A Inteligência Artificial nas Ciências de Dados

Luciane Ferreira Alcoforado
João Paulo Martins dos Santos
Orlando Celso Longo
Ariel Levy
Juan López Linares
Jessica Kubrusly

ORGANIZADORES
LUCIANE FERREIRA ALCOFORADO
JOÃO PAULO MARTINS DOS SANTOS
ORLANDO CELSO LONGO
ARIEL LEVY
JUAN LÓPEZ LINARES
JESSICA KRUBUSLY

A Inteligência Artificial nas Ciências de Dados

DOI: 10.11606/9786587023465

Pirassununga - SP
FACULDADE DE ZOOTECNIA E ENGENHARIA DE ALIMENTOS (FZEA)
UNIVERSIDADE DE SÃO PAULO (USP)
2024

UNIVERSIDADE DE SÃO PAULO

Reitor: Prof. Dr. Carlos Gilberto Carlotti Junior

Vice-Reitora: Profa. Dra. Maria Arminda do Nascimento Arruda

FACULDADE DE ZOOTECNIA E ENGENHARIA DE ALIMENTOS

Avenida Duque de Caxias Norte, 225 - Pirassununga, SP

CEP 13.635-900

http://www.fzea.usp.br

Diretor: Prof. Dr. Carlos Eduardo Ambrósio

Vice-Diretor: Prof. Dr. Carlos Augusto Fernandes de Oliveira

Dados Internacionais de Catalogação na Publicação

Serviço de Biblioteca e Informação da Faculdade de Zootecnia e Engenharia de Alimentos da Universidade de São Paulo

```
         Alcoforado, Luciane Ferreira
A354i       A inteligência artificial nas ciências de dados /
         Organizadores: Luciane Ferreira Alcoforado, João Paulo
         Martins dos Santos, Ariel Levy, Orlando Celso Longo,
         Juan López Linares, Jessica Kubrusly. -- Pirassununga:
         Faculdade de Zootecnia e Engenharia de Alimentos da
         Universidade de São Paulo, 2024.
            190 p.

            ISBN 978-65-87023-46-5 (e-book)
            DOI: 10.11606/9786587023465


            1. Linguagem R. 2. Visualização. 3. Estatística.
         4. Inteligência artificial. I. Santos, João Paulo
         Martins dos (org.). II. Levy, Ariel (org.). III. Longo,
         Orlando Celso (org.). IV. Linares, Juan López (org.).
         V. Kubrusly, Jessica (org.). VI. Título.
```

Ficha catalográfica elaborada por Girlei Aparecido de Lima, CRB-8/7113

"Com a chegada da inteligência artificial e a robótica em larga escala, a função humana deixa de ser produzir com eficiência, e uma nova missão surge para nós e as próximas gerações: voltar a criar, sonhar e imaginar."

trecho do livro Metaverso de Walter Longo & Flávio Tavares

AGRADECIMENTOS

Gostaria de expressar minha mais profunda e sincera gratidão a todos que dedicaram seu tempo de maneira especial para a concretização deste livro. Em primeiro lugar, agradeço aos organizadores desta obra: Luciane, João Paulo, Orlando, Ariel, Juan e Jéssica, com destaque para Luciane e Ariel, cuja amizade e parceria se mantém consolidadas desde a primeira edição do SER, em 2016, foram verdadeiros alicerces desta jornada, suas colaborações transcenderam o âmbito profissional, criando uma sinergia que foi essencial para a concretização de todos os produtos gerados pelo evento SER, incluindo esta obra. Suas capacidades em superar desafios com resiliência e manter um compromisso inabalável ao longo dos anos foi fundamental para que este trabalho alcançasse a profundidade, a relevância e o impacto que vemos hoje. Aos autores Ariel, Marcus, Jéssica, Luciane, Marco, João Paulo e Ariane, expresso meus mais sinceros reconhecimentos. Suas contribuições trazem uma visão atualizada e inovadora face as novas tendências para a análise de dados na era da Inteligência Artificial. Cada um deles ofereceram uma perspectiva única, enriquecendo o conteúdo com profundidades e reflexões que ampliaram as fronteiras do conhecimento. O comprometimento e a capacidade de gerar discussões instigantes foram fundamentais para transformar este projeto em um trabalho que não apenas informa, mas provoca e estimula debates essenciais nas áreas abordadas. O rigor intelectual e dedicação de cada um foi, sem dúvida, o motor que impulsionou o valor e a qualidade desta obra. Agradeço igualmente às revisoras Simone e Vitória, cuja atenção minuciosa aos detalhes foi determinante para elevar a qualidade final deste texto. Com um olhar atento e criterioso, elas garantiram que cada conceito fosse expresso com precisão e clareza, eliminando ambiguidades e garantindo a consistência em todo o conteúdo. Da mesma forma, minha gratidão aos editores João Paulo e Jéssica, que, com maestria, moldaram e refinaram as ideias, garantindo coesão e fluidez ao texto. Seus olhares críticos e expertise editorial foram indispensáveis para proporcionar uma leitura acessível e envolvente. Um agradecimento especial vai a Ariel pela arte da capa, que, com uma sensibilidade ímpar, capturou visualmente a essência do tema central do livro, traduzindo em imagem o espírito e a profundidade da obra. Igualmente, expresso minha gratidão à autora do prefácio, Mayra, cuja introdução instiga o leitor a se aprofundar nos temas abordados. Agradeço o apoio das instituições que tornaram este projeto possível: a Universidade Federal Fluminense (UFF) e a Academia da Força Aérea (AFA). Essas instituições forneceram não apenas os recursos necessários, mas também um ambiente intelectual propício para a investigação aprofundada e o avanço no campo da linguagem R e da inteligência artificial aplicada à análise de dados. O suporte foi decisivo para a realização deste trabalho.

Por fim, dedicamos este livro às nossas famílias e amigos, cujo apoio incondicional e encorajamento constante foram a força motriz que nos sustentou em cada etapa deste processo. Sem o suporte e a paciência deles, este projeto não teria sido possível. Esta obra é, em grande parte, resultado do amor e da confiança que eles depositaram em nós.

Orlando Celso Longo
Coordenador do Programa de Pós Graduação
em Engenharia Civil/UFF

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

ORGANIZADORES

Dra. LUCIANE FERREIRA ALCOFORADO (*lucianea@id.uff.br*). Possui graduação em Licenciatura em Matemática pela Universidade Federal de Santa Maria (1994), mestrado em Engenharia de Computação pela Universidade Federal do Rio de Janeiro (1998) e Doutora em Engenharia Civil pela Universidade Federal Fluminense (2009). É professora na Academia da Força Aérea (AFA) em Pirassununga/SP e professora colaboradora no Programa de Pós-Graduação em Engenharia Civil da UFF. Tem experiência na linguagem R, análise e visualização de dados e em problemas de otimização. Sua atuação na área de Matemática e Estatística destaca-se nos seguintes temas: regressão linear, regressão logística, testes de hipóteses, análise multivariada, métodos de apoio à tomada de decisão como Simplex, AHP entre outros e arte computacional com R. É integrante do Grupo de Pesquisa em Modelagem Matemática e Computacional da Academia da Força Aérea (GMMC/AFA). É líder do grupo de Pesquisa Estatística com R, coordenadora do SER - Seminário Internacional de Estatística com R, coorganizadora do capítulo R-ladies Niterói, autora de diversos livros sobre a linguagem R e dos pacotes disponíveis no CRAN, MandalaR e AHPWR.

Dr. JOÃO PAULO MARTINS DOS SANTOS (*jp2@alumni.usp.br*). Possui graduação em Licenciatura em Matemática pela Universidade Estadual Paulista Júlio de Mesquita Filho (2006), mestre em Matemática pela Universidade Estadual Paulista Júlio de Mesquita Filho (2009) e Doutor em Ciências pela Escola de Engenharia de São Carlos - EESC-USP. É professor na Academia da Força Aérea (AFA) em Pirassununga/SP, professor colaborador no Programa de Pós-Graduação em Engenharia Civil da UFF e líder do Grupo de Pesquisa em Modelagem Matemática e Computacional da AFA (GMMC/AFA). Tem experiência na área de Sistemas Dinâmicos não lineares e não ideais com pesquisa desenvolvida em métodos de perturbação. Tem experiência na área de Matemática e interesse nos seguintes temas: método numéricos para solução de equações diferenciais ordinárias e parciais, estimador de erro do tipo residual para a equação do transporte de poluentes, linguagem Python de programação, Computação Científica em Python e métodos numéricos para solução de sistemas lineares. Textos completos e gratuitos das publicações do autor podem ser encontrados aqui.

Dr. ORLANDO CELSO LONGO (*orlandolongo@id.uff.br*). Possui graduação em Engenharia Civil, Mestrado em Engenharia Civil e Doutorado em Engenharia de Transportes. Atualmente é Professor Titular da Universidade Federal Fluminense. Coordenador do Programa de Pós-graduação em Engenharia Civil da Universidade Federal Fluminense no período 2005 – 2013 e 2017 até a data atual. Diretor do DATAUFF de 2019 até data atual. Coordenou vários eventos nacionais e internacionais tais como IV Semana de Engenharia - III Seminário Fluminense de Engenharia, 4th International Conference on the Behaviour of Damaged Structures, VII Seminário Internacional de Estatística com R. Autor do pacote disponível no CRAN AHPWR. Tem experiência na área de Engenharia Civil e ambiente construído com ênfase em Construção Civil, atuando principalmente nos seguintes temas: construção civil, custos, gerenciamento / acompanhamento fiscalização, orçamento, administração de projetos e elaboração e desenvolvimento de projetos de infraestrutura para cidades inteligentes.

Dr. ARIEL LEVY (*alevy@id.uff.br*). Doutor em Economia (Universidade Federal Fluminense - 2013), mestre em Administração (IBMEC -2003) e engenheiro eletricista (Universidade Federal Fluminense - 1982). É Professor Associado da Universidade Federal Fluminense vinculado ao Departamento de Administração da Faculdade de Administração e Ciências Contábeis e foi coordenador do Curso de Graduação em Administração (2016-2021). Professor do quadro permanente do PPGAd - UFF e colaborador no MBA de Logística (LOGEMP - UFF), no MBA de Finanças (UFF), no MBA de Marketing (UFF) e dos cursos de Extensão em Ciências dos Dados (UFF) onde atua como coordenador. Possui experiência em Administração, com ênfase em Finanças Quantitativas; Finanças públicas; Planejamento e Controle; na linguagem R e na análise e visualização de dados. Organizador dos Seminários de Estatística R - Evento Internacional de Divulgação de Aplicações e Desenvolvimento de Linguagens R. Coordenador do Grupo de Pesquisa (CNPQ/UFF) - Métodos Quantitativos Aplicados à Administração.

Dr. JUAN LÓPEZ LINARES (*jlopez@usp.br*). Professor Associado do Departamento de Ciências Básicas (ZAB) da Faculdade de Zootecnia e Engenharia de Alimentos (FZEA) da Universidade de São Paulo (USP). Atualmente ministra as disciplinas de Cálculo II e IV para estudantes de engenharias e os cursos de "Treinamento Olímpico em Matemática para estudantes do Ensino Fundamental e Médio" e "Geometria olímpica com GeoGebra" para professores. Desenvolve projetos de pesquisa nas áreas de ensino de Cálculo e na resolução de problemas de Olimpíadas. Graduação e Mestrado em Física na Universidade da Havana, Cuba, em 1994 e 1996, respetivamente. Curso de Diploma da Matéria Condensada no Centro Internacional de Física Teórica Abdus Salam, em Trieste, na Itália em 1997-1998. Estágio no Instituto de Espectroscopia Molecular (CNR), Bolonha, Itália em 1998-1999. Doutor em Física pela Universidade Federal de São Carlos (UFSCar) em 1999-2001. Pós-doutorado de 4 anos (2002-2005) na Universidade Estadual de Campinas (Unicamp). Mestre Profissional em Matemática em Rede Nacional (PROFMAT) pela UFSCar em 2019 e Livre Docente na área de Ensino de Matemática Olímpica na FZEA USP em 2022. Textos completos e gratuitos das publicações do autor podem ser encontrados aqui.

Dra. JESSICA KUBRUSLY (*jessicakubrusly@id.uff.br*). Professora do Departamento de Estatística do Instituto de Matemática e Estatística da UFF desde 2009. Formada em Engenharia Elétrica, Mestre e Doutora em Matemática, sempre pela PUC-Rio. Sua área de atuação é Aprendizado de Máquinas, Sistemas de Recomendação e Mineração de Textos. Nestes temas a pesquisadora tem orientado alunos, desenvolvido projetos de pesquisa e de extensão, e também publicando artigos ao longo dos últimos anos. Recentemente publicou o livro 'Uma Introdução à Programação no R' e desenvolveu o pacote CFilt, disponível pelo CRAN do R.

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

AUTORES

ARIEL LEVY (*alevy@id.uff.br*): Doutor em Economia pela Universidade Federal Fluminense (2013), mestre em Administração pelo IBMEC (2003) e engenheiro eletricista pela Universidade Federal Fluminense (1982). Atualmente, é Professor Associado da Universidade Federal Fluminense, vinculado ao Departamento de Administração da Faculdade de Administração e Ciências Contábeis, onde também exerceu o cargo de coordenador do Curso de Graduação em Administração entre 2016 e 2021. Além disso, é professor do quadro permanente do PPGAd - UFF e colabora no MBA de Logística (LOGEMP - UFF), no MBA de Finanças (UFF), no MBA de Marketing (UFF) e nos cursos de Extensão em Ciências dos Dados (UFF), onde atua como coordenador. Sua experiência abrange a área de Administração, com ênfase em Finanças Quantitativas, Finanças públicas, Planejamento e Controle, além de conhecimentos na linguagem R e na análise e visualização de dados. Dr. Levy também é organizador dos Seminários de Estatística R - Evento Internacional de Divulgação de Aplicações e Desenvolvimento de Linguagens R e coordenador do Grupo de Pesquisa (CNPQ/UFF) - Métodos Quantitativos Aplicados à Administração.

MARCUS ANTONIO CARDOSO RAMALHO (*marcusantonio@id.uff.br*): Graduado e mestre em Administração pela Universidade Federal Fluminense (UFF), atua como professor nos cursos de formação executiva em Ciências de Dados e Finanças Corporativas e Mercados de Capitais na mesma instituição. Paralelamente, exerce a função de cientista de dados e desenvolvedor de sistemas no projeto Lagoa Viva (UFF/Maricá, RJ). Tem experiência em ciência de dados com R e Python, programação funcional, desenvolvimento de sistemas com Django, visualização de dados e automação de processos administrativos com Python, R e VBA. Seus interesses acadêmicos e profissionais incluem Administração da Informação, Desenvolvimento de sistemas, Economia Política e Finanças, com ênfase nas linguagens R e Python.

JESSICA KUBRUSLY (*jessicakubrusly@id.uff.br*): Professora do Departamento de Estatística do Instituto de Matemática e Estatística da UFF desde 2009. Formada em Engenharia Elétrica, Mestre e Doutora em Matemática, sempre pela PUC-Rio. Sua área de atuação é Aprendizado de Máquinas, Sistemas de Recomendação e Mineração de Textos. Nestes temas a pesquisadora tem orientado alunos, desenvolvido projetos de pesquisa e de extensão, e também publicando artigos ao longo dos últimos anos. Recentemente publicou o livro 'Uma Introdução à Programação no R' e desenvolveu o pacote CFilt, disponível pelo CRAN do R.

LUCIANE FERREIRA ALCOFORADO: Possui graduação em Licenciatura em Matemática pela Universidade Federal de Santa Maria (1994), mestrado em Engenharia de Computação pela Universidade Federal do Rio de Janeiro (1998) e Doutora em Engenharia Civil pela

Universidade Federal Fluminense (2009). É professora na Academia da Força Aérea em Pirassununga/SP e professora colaboradora no Programa de Pós-Graduação em Engenharia Civil da UFF. Tem experiência na linguagem R, análise e visualização de dados e em problemas de otimização. Sua atuação na área de Matemática e Estatística destaca-se nos seguintes temas: regressão linear, regressão logística, testes de hipóteses, análise multivariada, métodos de apoio à tomada de decisão como Simplex, AHP entre outros e arte computacional com R. É líder do grupo de Pesquisa Estatística com R, coordenadora do SER - Seminário Internacional de Estatística com R, coorganizadora do capítulo R-ladies Niterói, autora de diversos livros sobre a linguagem R e dos pacotes disponíveis no CRAN, MandalaR e AHPWR.

MARCO AURÉLIO CHAVES FERRO (*marcoferro@id.uff.br*): Professor dos Cursos de Graduação e Pós-Graduação em Engenharia Civil da Universidade Federal Fluminense (UFF). Graduado em Engenharia Civil pela Universidade Federal do Rio de Janeiro (UFRJ) em 1987, Mestre pela COPPE/UFRJ em 1997, Doutor pela COPPE/UFRJ em 2002 e Pós-Doutor pela COPPE/UFRJ em 2008 e Pós-Doutor pela FGV/EBRAPE em 2012. Atua nas áreas de Simulação Numérica e Análise e Cálculo Estrutural, Métodos Numéricos e Inteligência Artificial em Engenharia."

JOÃO PAULO M. DOS SANTOS (*jp2@alumni.usp.br*): Possui graduação em Licenciatura em Matemática pela Universidade Estadual Paulista Júlio de Mesquita Filho (2006), mestre em Matemática pela Universidade Estadual Paulista Júlio de Mesquita Filho (2009) e Doutor em Ciências pela Escola de Engenharia de São Carlos - EESC-USP. Docente na Academia da Força Aérea em Pirassununga/SP, e colaboradora no Programa de Pós-Graduação em Eng. Civil (UFF). Tem interesse em Matemática Aplicada e Estatística.

ARIANE HAYANA THOMÉ DE FARIAS (*ariane.hayana@gmail.com*): Possui graduação em Estatística e Economia, ambas pela Universidade Federal do Amazonas (UFAM). Atua como Assessora Estatística no Tribunal de Justiça do Estado de Roraima (TJRR) e está cursando MBA em Data Science e Analytics na USP/ESALQ.

REVISORAS

VITÓRIA SPANGHERO (*vspanghero@gmail.com*): Possui graduação em Letras/Linguística pela Universidade de São Paulo (USP), Mestrado e Doutorado em Linguística pela Universidade Estadual de Campinas (Unicamp) e Pós-doutorado em Linguística pela Universidade de São Paulo (USP). Atua nas seguintes áreas: Teoria e Análise Linguística, Linguística e Língua Portuguesa, Fonologia, Semântica e Lexicografia. Foi bolsista CAPES e FAPESP e corretora das provas de redação do Exame Nacional do Ensino Médio (Enem) e Vestibular Unicamp. Fez estágio no escritório da ORSTOM (*Institut Français de Recherche Scientifique pour le Développement*), em Cayenne, Guiana Francesa, desenvolvendo pesquisa linguística. Por ocasião do Pós-doutorado, ministrou uma disciplina no programa de Pós-graduação da FFLCH-USP. Trabalhou na Universidade Federal de Mato Grosso do Sul, atuando como docente nos cursos de Graduação e Pós-graduação. Atualmente ministra aulas de Língua Portuguesa 2 (POT2) na Academia da Força Aérea, em Pirassununga.

SIMONE ALVES LIMA BOERO (*slboero@gmail.com*): Possui graduação em Letras pela Faculdade de Ciências e Letras da Universidade Estadual Paulista Júlio de Mesquita Filho UNESP - Araraquara (1991), mestrado e doutorado em Estudos Literários pela mesma instituição (2006 e 2013). Atualmente é professora de Língua Portuguesa da Academia da Força Aérea. Trabalha com o enfoque em produção textual.

Título

A Inteligência Artificial nas Ciência de Dados

Prefácio

Em sua oitava edição, o SER – Seminário Internacional de Estatística com R – reuniu profissionais extremamente qualificados para apresentar um tema atual e de grande relevância.

A Inteligência Artificial (IA) é uma ferramenta poderosa e possibilita a análise de grandes volumes de dados com rapidez e precisão, acelerando descobertas nas diversas áreas do conhecimento. A IA expande possibilidades e transforma a forma como interagimos com o conhecimento e com o mundo ao nosso redor.

De que forma a Inteligência Artificial pode ser aplicada na programação em R, que é de código aberto, para atender a uma variedade de temas relacionados à estatística?

O título do livro, "A Inteligência Artificial na Ciência de Dados", resume de forma brilhante a essência do que foi tratado no evento de interesse ao meio técnico e científico, pois possibilita abordar técnicas e algoritmos da Inteligência Artificial (IA) aplicados na Ciência de Dados, permitindo análise, predição e automação. Em contrapartida, esta utiliza dados para treinar e melhorar modelos de IA.

O livro é composto por seis capítulos, elaborados por sete autores de distintas instituições de renome, abordando temas variados que circundam o tema central delineado pelo título da obra.

O primeiro capítulo, de autoria de Ariel Levy e Marcus Antonio Cardoso Ramalho, apresenta como analisar e otimizar uma carteira de ações com R em relação ao retorno e ao risco dos ativos de forma que supere o rendimento do mercado. A relevância do tema está na constante busca dos investidores por estratégias que otimizem a rentabilidade de seus portfólios e reduzam a exposição a riscos não desejados.

O capítulo 2, escrito por Jessica Kubrusly, introduz o leitor à programação com R. São apresentadas as funções básicas, bem como os controles de fluxo, a criação de funções, com diversos exemplos práticos para melhor entendimento do leitor. A programação é uma ferramenta imprescindível na simulação e solução de problemas reais.

O capítulo 3 de Luciane Ferreira Alcoforado explora a programação em R, amplamente utilizada em análise de dados e estatística, com assistência da Inteligência Artificial (IA), em particular o ChatGPT. As vantagens do uso da IA na linguagem de programação com R são mostradas, como por exemplo: oferta de assistência instantânea, explicação de conceitos complexos, além de auxílio na geração de códigos; as desvantagens podem ser observadas quanto à imprecisão das respostas, no custo de aquisição de versões mais completas e com recursos mais avançados.

No capítulo 4, Marco Aurélio Ferro apresenta algumas aplicações da Inteligência Artificial (IA), com seus impactos e desafios em diversas áreas do conhecimento. Inicialmente, são abordados seu histórico e sua evolução, com início na segunda metade do século XX. Em seguida, mostram-se abordagens e técnicas utilizadas na IA incluindo aprendizado de máquina, aprendizado profundo, redes neurais artificiais, processamento de linguagem natural, visão computacional, sistemas especialistas e robótica. O autor também aborda e discute os avanços e o futuro da IA.

O capítulo 5, apresentado pelo João Paulo Martins dos Santos, explora a interseção entre a geometria fractal e a arte por meio da computação viabilizada pela linguagem R. Os resultados evidenciam tanto os aspectos computacionais baseados nos processos iterativos, em geral, associados à geometria, proporcionando uma aplicação direta e visual. Um tema central na investigação é a aplicação sistemática das transformações (translações, rotações e homotetias), tema fortemente relacionado à Matemática do ponto de vista teórico, aos fractais fundamentais tais como a curva de Koch e a árvore fractal. O capítulo faz uma contribuição notável ao oferecer uma nova perspectiva para a geração de padrões geométricos intrincados, ao mesmo tempo em que aponta diferentes possibilidades de implicações pedagógicas para a educação interdisciplinar.

Com o propósito de auxiliar programadores que desejam aprimorar suas habilidades em análise de dados, o capítulo 6 de Ariane Hayana Thomé de Farias é voltado àqueles que buscam inovar em práticas analíticas mediante o uso de inteligência artificial, agregando valor aos resultados obtidos. São apresentadas etapas práticas para integrar a API do ChatGPT ao RStudio e explorar as funcionalidades do pacote gptstudio. O conteúdo abrange desde a configuração inicial do RStudio até a aplicação prática das ferramentas por meio de exemplos, proporcionando um estudo abrangente e fundamentado sobre o uso da inteligência artificial em análises de dados.

A organização deste livro foi feita de forma primorosa por Luciane Ferreira Alcoforado, João Paulo Martins dos Santos, Orlando Celso Longo, Ariel Levy, Juan López Linares e Jessica Kubrusly, e será de grande valor aos que desejam iniciar o uso da IA em linguagem de programação, em especial o R.

Mayra Soares Pereira Lima Perlingeiro
Professora do Programa de Pós Graduação em Engenharia
Civil (PPGEC) da Universidade Federal Fluminense
Vice Coordenadora do PPGEC (2022-2025)

Palavras-chave: Linguagem R, Visualização, Estatística, Inteligência Artificial.

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

Lista de Figuras

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

Conteúdo

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

Capítulo 1

Análise e Otimização de uma Carteira de Ações com R

Autores: Ariel Levy e Marcus Antonio Cardoso Ramalho[1]
Instituição: Universidade Federal Fluminense, RJ
e-mails: alevy@id.uff.br, marcusantonio@id.uff.br

1.1 INTRODUÇÃO

O objetivo de investir é obter lucros. Os lucros ou prejuízos em ativos de renda variáveis dependem dos valores investidos e das variações nos preços dos ativos escolhidos. O desejável é obter o maior lucro possível ante o valor investido. A medida para isso se dá por meio dos retornos percentuais, obtidos pelo cálculo da variação dos preços em certo horizonte de tempo, por exemplo, diário, mensal ou mesmo anual. Entretanto, nem tudo são flores. Cada tipo de investimento tem um risco associado. Então, podemos redefinir nosso objetivo: para dada quantia investida, pretende-se obter o maior lucro com o menor risco possível.

Ao longo do texto, trabalharemos com dois diferentes paradigmas da utilização de dados com R, mais utilizados pela indústria, a saber: xts; (**tidyverse**) *tidyverse*, *tibble*. Cada um deles tem facilidades específicas que serão exploradas em nosso aprendizado.

Aqui utilizaremos um portfólio genérico e acadêmico, apenas para mostrar as aplicações técnicas. Na prática você deveria escolher um conjunto de empresas quatro a cinco vezes maior do que aquele que pretende efetivamente colocar em sua carteira. Isso porque, eventualmente, precisará substituir os ativos. A melhor fonte de informações sobre as empresas é seu passado, apesar de saber que o passado não garantirá o futuro. Para a escolha das empresas, você deverá avaliar seus indicadores, a evolução das contas do balanço e demais demonstrações contábeis.

Essa parte, embora de extrema importância, não será objeto de estudo neste capítulo que tem o objetivo de apresentar a análise e otimização de uma carteira de ações com R. O trabalho está dividido nas seguintes etapas: Objetivo, Referencial teórico, Metodologia, Resultados e Considerações finais.

Nas seções seguintes exploraremos a moderna teoria de portfólios e o índice *Sharpe* que fundamentam a escolha de carteiras sob o aspecto de finanças. Posteriormente, analisaremos como podemos utilizar o R e suas bibliotecas para otimizar estas carteiras resolvendo os pesos para os artigos escolhidos.

1.2 OBJETIVO

O objetivo deste trabalho é apresentar como se pode otimizar um portfólio de investimento em relação ao retorno e o risco dos ativos de forma que supere o rendimento do mercado.

Como objetivos secundários, mostraremos como obter os dados de preços de fechamento de ações diretamente da internet, como calcular os retornos percentuais, a matriz de covariância dos retornos, o retorno e o risco de um portfólio e como otimizar o portfólio.

1.3 REFERENCIAL TEÓRICO

A moderna teoria de portfólios (**markowitz_portfolio_1952**) fundamenta-se no princípio de que os retornos são desejáveis e os riscos são indesejáveis. Markowitz inovou ao combinar ativos com diferentes perfis de retorno e risco,

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

permitindo, assim, a construção de uma carteira que otimiza o retorno para um dado nível de risco, ou minimiza o risco para um determinado nível de retorno.

Para melhor entendermos a confecção dessas carteiras, é necessário introduzir alguns conceitos. Inicialmente, iremos calcular os retornos dos ativos de forma isolada, utilizando a subtração logarítmica.

$$k_i = \log(P_{t+1}) - \log(P_t) \tag{1.3.1}$$

Onde: k= logretorno do ativo x no período t, e P = Preço do ativo no período t.

Faz-se assim, porque é mais fácil subtrair do que dividir computacionalmente. Para obter na forma linear, percentual, o resultado seria dado por:

$$k_{linear} = (e^{k_i} - 1) \times 100 \tag{1.3.2}$$

O desvio-padrão (σ) de um ativo representa seu risco e é calculado usando-se a fórmula:

$$\sigma = \sqrt{\frac{1}{T}\sum_{i=1}^{T}(k_t - \mu)^2} \tag{1.3.3}$$

onde: T é o número total de observações, k_t é o valor da i-ésima observação do retorno, μ é a média das observações.

Denomina-se por "carteira"ou "portfólio"o conjunto de ativos mantidos por um agente. Este conjunto representará a totalidade dos ativos mantidos em suas diversas posições. Assim, determinados ativos apresentarão posições compradas, e outros, vendidas. Como estes ativos representam a totalidade, a soma dos pesos destes na carteira deverá ser 100%. O peso w será representado normalmente pelo anagrama em inglês de *weight*.

E podem ser obtidos por:

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

$$w_i = \frac{\text{valor aplicado em } i}{\text{total da carteira}} = \frac{V_i}{\sum_{j=1}^{n} V_j} \tag{1.3.4}$$

onde: w_i será o peso, e V_i, o valor aplicado do i-ésimo ativo.

Então, para uma carteira composta por N ativos, o retorno da carteira será dado pela média ponderada dos retornos pelos pesos, a saber:

$$\sum_{i}^{N} k_i w_i \tag{1.3.5}$$

Lembre que a soma será de 100% ou 1 o que facilita a fórmula 1.3.5.

O risco de um ativo foi calculado como sendo o desvio-padrão de seus retornos, para a carteira um novo componente entrará em ação, a covariância dos ativos, que mede como um ativo alterará seu retorno ante o movimento do outro. A forma mais utilizada desta covariância é expressa pelo produto do coeficiente de correlação de *Pearson*, ρ, pelo produto dos desvios-padrões dos dois ativos.

Sejam x e y dois ativos, a covariância deles será dada por:

$$cov(x, y) = \sigma_{x,y} = \sigma_[xy] = \sigma_{yx}\rho - x, y\sigma_x\sigma_y \tag{1.3.6}$$

A variância de uma carteira C de dois ativos (x e y) será:

$$\sigma_C^2 = w_x^2\sigma_x^2 + w_y^2\sigma_y^2 + 2w_x w_y \sigma_{xy} \tag{1.3.7}$$

ou

$$\sigma_C^2 = w_X^2\sigma_X^2 + w_Y^2\sigma_Y^2 + 2w_X w_Y \rho_{XY}\sigma_X\sigma_Y \tag{1.3.8}$$

Com dois ativos verifica-se facilmente que os termos em covariância igualam o número de termos dos ativos em que as variâncias dos ativos estão presentes.

E, quando combinamos N ativos em uma carteira, a variância da carteira é dada por:

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

$$\sigma_c^2 = \sum_{i=1}^{n} w_i^2 \sigma_i^2 + 2 \sum_{i=1}^{n} \sum_{j=1}^{n} w_i w_j \rho_{ij} \sigma_i \sigma_j \tag{1.3.9}$$

e, claro, o risco da carteira será:

$$\sigma_c = \sqrt{\sigma_c^2} \tag{1.3.10}$$

Um ativo carrega consigo os riscos associados à organização emissora. Isso significa que parte das oscilações de preços observadas nos mercados é decorrente das informações adquiridas pelos investidores sobre as decisões daquela organização, seus dirigentes ou mesmo sobre seu setor ou ramo de atuação. Assim, uma notícia favorável elevará o preço do ativo, enquanto uma notícia desfavorável o reduzirá. Isso ocorre, porque os investidores estão formando expectativas sobre os resultados futuros e determinam o preço do ativo comprando e vendendo o papel.

À medida que o número de ativos em uma carteira aumenta, ocorre uma diversificação ou eliminação parcial ou total desses riscos individuais, pelos quais o investidor não será remunerado e, portanto, não precisa assumir. No entanto, existem riscos presentes para todos os participantes de um determinado mercado. Esses riscos estão relacionados a projeções sobre a economia global e, principalmente, à economia local, tais como: inflação, projeção de crescimento, taxa de desemprego, taxa de juros, acordos internacionais e qualquer outra política ou indicador que possa influenciar o desempenho das organizações como um todo. Isso significa que haverá um limite para a diversificação, e que as empresas experimentarão uma parcela de risco relativo ao seu mercado, denominado risco sistêmico ou não diversificável, embora tais riscos possam ser mitigados por meio dos mercados de derivativos.

No cálculo do risco de uma carteira, o número de termos da equação será o quadrado do número de ativos, (n^2). Os termos em variância estarão presentes no mesmo número que o de ativos, n. E os termos em covariâncias serão todos os demais $(n^2 - n)$. Esses termos crescem em número mais rapidamente e assumem uma maior importância na determinação do risco da carteira. Assim, à medida que o número de ativos em uma carteira cresce, os termos em covariância superam os termos em variância, tornando-se cada vez mais importantes e promovendo a redução do risco da carteira.

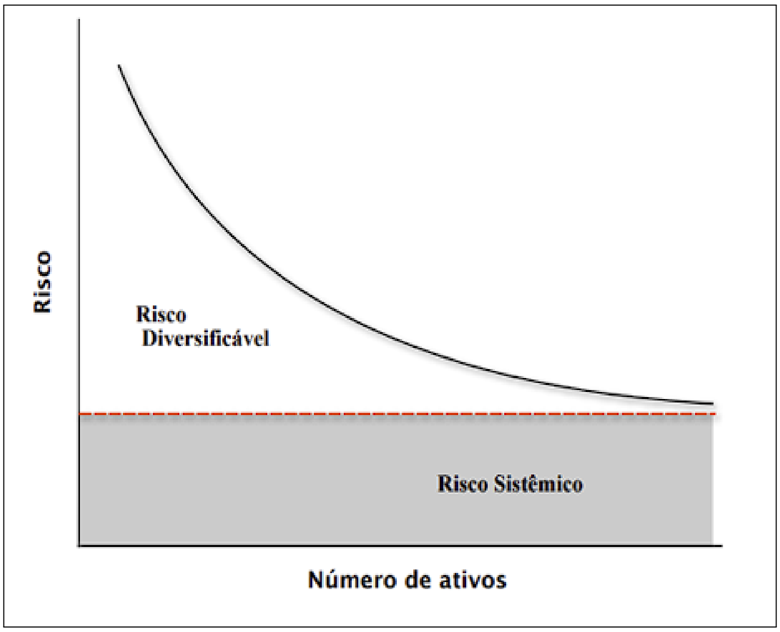
Tabela 1.1: Quantidade de termos em uma carteira de N ativos.

Quantidade de ativos	Total de termos	Termos em variância	Termos em covariância
n	n^2	n	$n^2 - n$
2	4	2	2
3	9	3	6
5	25	5	20
10	100	10	90
20	400	20	380
30	900	30	870

Fonte: Os autores, 2024.

A partir de um certo número, apesar de o efeito de diversificação não cessar, o risco da carteira praticamente igualará ao risco de sistêmico e, talvez, não se justifique a inclusão de novos ativos pela razão de diversificação, ver 4.1.

Figura 1.1: Risco diversificável *versus* não diversificável.



Fonte: Adaptado de (**galvaoFinancas**).

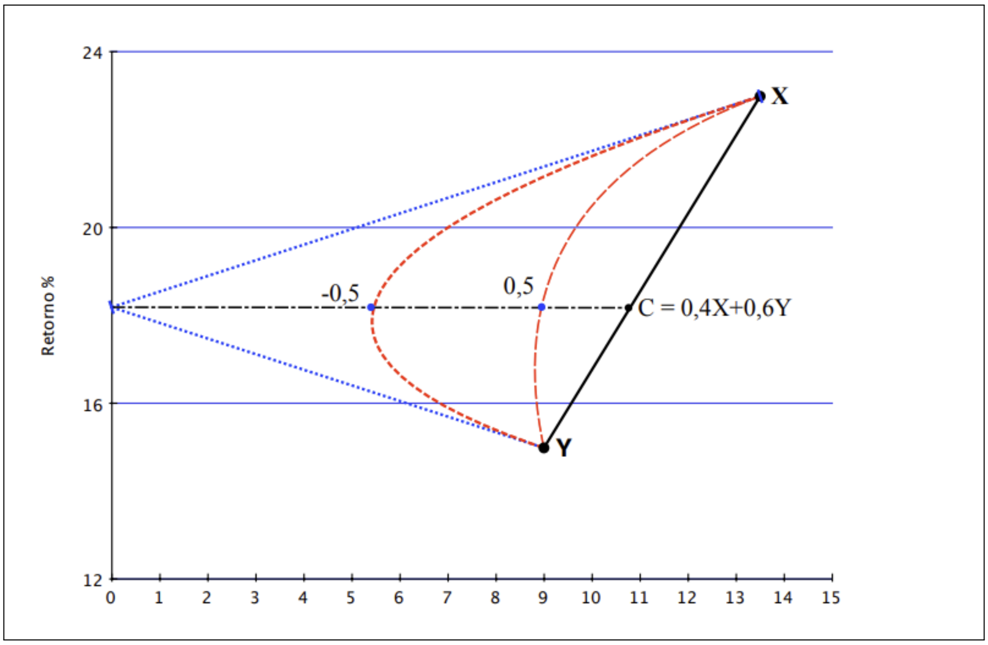
A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

A importância da diversificação de ativos, um conceito intuitivo comparável ao provérbio 'não colocar todos os ovos em uma cesta', foi evidenciada pelos estudos de Markowitz. A seguir, exploraremos a fronteira eficiente, um conceito fundamental em sua teoria.

1.3.1 Fronteira Eficiente

Normalmente combinamos ativos com diversas correlações e, como resultado, obtêm-se gráficos das carteiras possíveis, com a análise de média e variância, como realizada na figura 1.2 que reproduzimos parcialmente na figura 1.3.

Figura 1.2: Análise de Média e Variância.

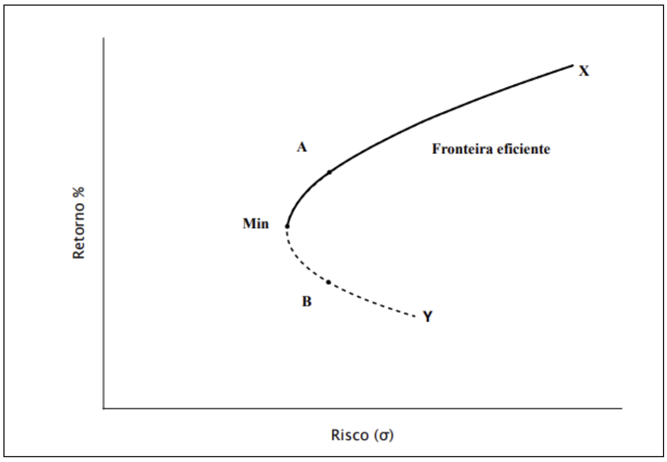


Fonte: Adaptado de (**galvaoFinancas**).

A figura 1.2 representa a combinação de dois ativos, x e y, numa carteira na qual, se, com correlação unitária, teríamos, $\rho = 1$, a carteira C movendo-se sobre o segmento de reta $\overline{xy}$ que, com os pesos respectivos de 40% e 60% no ponto **C**, nos apresenta três outros lugares geométricos, onde $\rho = 0,5$, na primeira curva da direita para esquerda, $\rho = -0,5$, na segunda curva, e, finalmente, para $\rho = -1$, a carteira se apresenta sem risco, já que o ponto **C** coincide com o eixo das ordenadas. Na prática não será tão simples encontrar correlações perfeitamente negativas. Entretanto, a diversificação ocorrerá mesmo em casos de correlações positivas.

Os agentes são racionais, então irão preferir a carteira A à carteira B, 1.3. Isto porque, para um mesmo nível de risco, irão preferir um maior retorno. Toda carteira formada por uma combinação dos ativos **X** e **Y** que estiver no segmento inferior da curva (tracejado) terá uma correspondente de maior retorno com mesmo risco. Assim nossa escolha de carteiras ficará restrita ao trecho da curva compreendido entre a carteira de variância mínima (Min) e a carteira X formada 100% pelo ativo X. Este trecho da curva é denominado de **fronteira eficiente** .

Figura 1.3: Fronteira Eficiente.



Fonte: Adaptado de (**galvaoFinancas**).

A escolha da carteira ideal, dentro do trecho eficiente, depende diretamente do perfil de risco do investidor. Quanto maior a tolerância ao risco, mais próxima da carteira com máxima variância (ponto X) estará a escolha do investidor. Por outro lado, investidores mais avessos ao risco tenderão a optar por carteiras mais próximas da carteira de mínima variância.

A carteira de variância mínima (Min) é, por definição, aquela cuja covariância entre seu retorno e o retorno dos n ativos a ela pertencentes é igual para todos os ativos.

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

$$\sigma_{x_i,Min} = k, \quad i \in [1, 2, 3 \cdots] \tag{1.3.11}$$

Onde k é uma constante. Sem perder a generalidade, podemos admitir k=1. Assim,

$$cov(k_1, MIn) = cov(k_2, MIn) = cov(k_3, MIn) \cdots = cov(k_n, MIn) = 1 \tag{1.3.12}$$

O retorno da carteira de variância mínima será obtido pelo uso da equação (13), onde w_i é o peso do i-ésimo ativo da carteira de variância mínima.

$$k_{Min} = \sum_{i=1}^{n} k_i w_i \tag{1.3.13}$$

Utilizando estes resultados, montamos o sistema:

$$\begin{bmatrix} \sigma^2 & \sigma_{12} & \cdots & \sigma_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n1} & \sigma_{n2} & \cdots & \sigma_n^2 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}, \text{ ou, } M.W = I \tag{1.3.14}$$

A solução desta equação nos dará os pesos ideais para construir uma carteira com o menor risco possível. Entretanto deveremos observar a proporcionalidade já que a soma deverá ser 100%.

Na próxima seção, determinaremos os pesos da carteira tangente, combinando uma carteira de risco com um título do Tesouro Nacional, que é considerado um investimento livre de risco.

1.3.2 Portfólio Tangente

Até o momento, nossa análise enfocou carteiras compostas exclusivamente por ativos de risco. A introdução de um ativo livre de risco, no entanto, altera

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

significativamente essa perspectiva.

Um ativo livre de risco é um título de renda fixa com baixíssimo potencial de perda. Tradicionalmente, títulos governamentais são considerados os ativos livres de risco de um país. A justificativa reside na capacidade dos governos de emitir moeda e honrar suas dívidas, mesmo que isso implique inflação. Os Estados Unidos, em particular, são vistos como a referência mundial em termos de segurança, devido à improbabilidade de *default* e à baixa volatilidade de seus títulos.

No Brasil, o Tesouro Direto popularizou a aplicação em títulos públicos, permitindo investimentos a partir de pequenas quantias. A taxa SELIC, que remunera esses títulos, é considerada a taxa livre de risco doméstica K_f.

Ao investir em ativos de risco, os investidores naturalmente esperam retornos superiores aos oferecidos por títulos públicos. Essa diferença, denominada prêmio de risco, compensa tanto o risco de perda de capital quanto a volatilidade dos retornos. Matematicamente, o prêmio de risco ΔK pode ser calculado pela diferença entre o retorno esperado do ativo de risco k_a e a taxa livre de risco k_f:

$$\Delta k = k_a - k_f \tag{1.3.15}$$

A combinação da carteira A com um ativo livre de risco, F, gera um novo perfil de risco e retorno. A inclusão desse ativo, por não apresentar volatilidade, reduz o risco total da carteira. Pode-se inferir que σ_F e $\sigma_{AF} = 0$. Daí decorre:

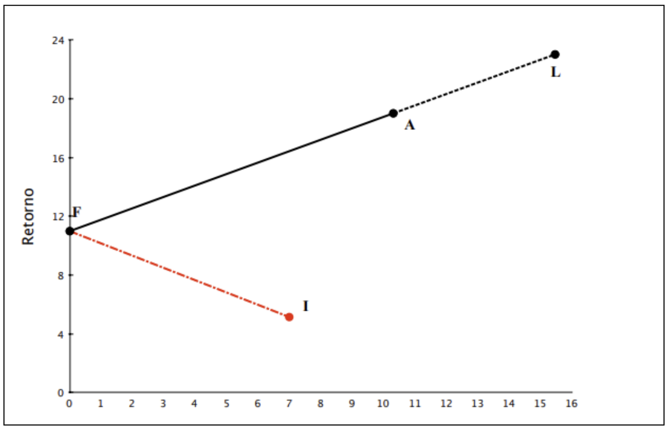
$$\bar{k}_{AF} = \bar{k}_F + \frac{(\bar{k}_A - \bar{k}_F)}{\sigma_A}\sigma_{AF} \tag{1.3.16}$$

A Equação 1.3.16 descreve a linha de mercado de capitais, representando a relação linear entre o retorno esperado e o desvio-padrão (risco) de uma carteira composta por ativos de risco e um ativo livre de risco.

As carteiras A e F representam os extremos do espectro de investimento: 100% em ativos de risco (A) e 100% em ativos livres de risco (F), respectivamente.

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

Figura 1.4: Combinações possíveis da carteira A (ativo de risco) com o ativo F (livre de risco).



Fonte: Adaptado de (**galvaoFinancas**).

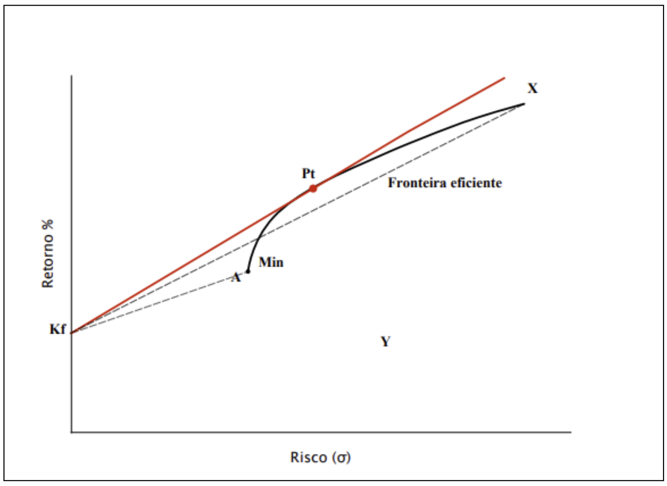
A carteira L, por sua vez, representa uma posição de alavancagem financeira, onde o investidor toma emprestado um valor equivalente a um terço do valor investido na carteira A, ampliando assim sua exposição ao risco e ao retorno.

A carteira I, por outro lado, apresenta uma posição vendida a descoberto no ativo A. O investidor vende o ativo A, recebendo o valor da venda e aplicando esse valor, somado a um montante duas vezes maior, em ativos de renda fixa. Essa estratégia é adotada quando o investidor espera uma queda no preço do ativo A.

A Figura 1.4 ilustra graficamente a relação entre o risco e o retorno das carteiras A, F, L e I. Cada ponto no gráfico representa uma combinação específica de risco e retorno, e a curva que conecta esses pontos mostra o conjunto de todas as carteiras possíveis, dadas as restrições de investimento. A carteira L, por exemplo, encontra-se acima da curva de eficiência, indicando um maior retorno para um determinado nível de risco, o que é típico de estratégias alavancadas. A carteira I, por sua vez, pode apresentar um retorno negativo se o preço do ativo A subir, refletindo o risco associado a essa estratégia.

A alavancagem está caracterizada por todas as carteiras no segmento de reta $\overline{LA}$, até o índice correspondente ao peso de L, se fosse maior a alavancagem desta carteira, este segmento se prolongaria se afastando dos eixos. No segmento de reta $\overline{AF}$, estão representadas todas as carteiras em que ambos os pesos são

Figura 1.5: Determinação da fronteira eficiente entre ativos de risco e um ativo livre de risco.



Fonte: Adaptado de (**galvaoFinancas**).

positivos, representando posições compradas tanto na carteira A como no Ativo F. Finalmente, no segmento $\overline{FI}$, estão as carteiras com posições vendidas em A e compradas em F.

Uma posição comprada de ativos de risco a um ativo livre de risco será uma reta, na figura 1.4 representada pelo segmento $\overline{FL}$. Pode-se, então, partir ao exame de qual seria, neste caso, a fronteira eficiente. Para isto retomaremos a figura 1.3 apenas para facilitar o raciocínio e sem perda de generalização, tornemos a carteira A como sendo a de variância mínima, de menor risco. Assim, nosso ponto de partida será um dos extremos da curva de fronteira eficiente das possíveis carteiras formadas por estes ativos de risco.

De forma similar ao procedimento anterior, liga-se o ponto Kf, que representa a taxa de retorno do ativo livre de risco (F), ao ponto que representa a carteira A. Este segmento será o lugar geométrico de todas as carteiras formadas pela carteira de variância mínima e o ativo livre de risco, cujos pesos sejam positivos. Se, em vez de ligar ao ponto A, se ligasse por um segmento o ponto Kf ao ponto X, se obteria o lugar geométrico de todas as carteiras de peso positivo que são formadas por X e o ativo livre de risco. É fácil verificar que este novo segmento domina o anterior, pois oferece maiores retornos para os mesmos níveis de risco.

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

Ao deslocar a reta ao longo da curva da fronteira eficiente formada pelos ativos de risco $\overline{(}AF)$, verifica-se que o segmento de reta que passa pelo ponto tangente dominará todos os demais. Esta carteira será denominada de Portfólio Tangente (P_t).

O procedimento para determinar os pesos do P_t é similar ao utilizado na determinação da carteira de variância mínima. Note que o portfólio tangente se caracteriza por apresentar uma composição em todos os ativos a ele pertencentes que apresentarão uma mesma razão entre o prêmio de risco pela respectiva covariância destes ativos com o P_t deve ser constante. Novamente, sem perda de generalidade, vamos igualar esta constante à unidade e obter os pesos provisórios para cada um dos ativos componentes deste portfólio.

$$\frac{k_i - k_f}{\sigma_{iP_t}} = 1,\ i \in [1, 2, 3 \cdots n] \tag{1.3.17}$$

Então, pode-se reescrever estas relações como:

$$k_i - k_f = \sigma_{iP_t},\ i \in [1, 2, 3 \cdots n] \tag{1.3.18}$$

Resultando num sistema de equações:

$$\begin{bmatrix} \sigma^2 & \sigma_{12} & \cdots & \sigma_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n1} & \sigma_{n2} & \cdots & \sigma_n^2 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} k_1 - K_f \\ k_2 - K_f \\ \vdots \\ k_n - K_f \end{bmatrix},\ \text{ou},\ M.W = P \tag{1.3.19}$$

Outra vez, em virtude da assunção da constante como a unidade, será necessário o ajuste dos pesos provisórios por proporcionalidade direta, vez que a soma destes não corresponde a 100

1.3.3 Índice de *Sharpe*

O Índice de *Sharpe* (**sharpe_capital_1964**) é uma medida amplamente utilizada para avaliar o desempenho de investimentos, comparando o retorno de uma carteira com o risco associado. Calculado como:

$$I_S = \frac{(\bar{k}_i - k_f)}{\sigma} \tag{1.3.20}$$

Onde $\bar{K}_i$ é o retorno médio da carteira, σ é o risco da carteira, e k_f, a taxa livre de risco.

Um Índice de *Sharpe* elevado sugere que a carteira obteve um retorno superior em relação ao risco assumido. Isso significa que o investidor foi devidamente recompensado pelo risco adicional. Essa métrica possibilita a comparação entre diversas carteiras, o que ajuda a identificar aquelas que proporcionam o melhor balanço entre risco e retorno.

Apesar de ser uma ferramenta valiosa, o Índice de *Sharpe* tem suas limitações. Ele pressupõe que os retornos seguem uma distribuição normal e não leva em conta outros fatores de risco, como a liquidez.

O Índice de *Sharpe* é uma ferramenta poderosa para avaliar o desempenho de investimentos e, quando combinado com técnicas de otimização, pode ser utilizado para construir carteiras altamente eficientes.

1.4 OTIMIZAÇÃO DE CARTEIRAS UTILIZANDO O R

Nas seções anteriores, exploramos a moderna teoria de portfólio e o Índice de *Sharpe* como ferramentas para avaliar o desempenho de investimentos. Na sequência, aprofundaremos nossa análise, utilizando o Índice de *Sharpe* (**sharpe_capital_1964**) como base para construir carteiras otimizadas. Veremos como o R e algumas bibliotecas podem ser usados para encontrar a alocação de ativos que maximiza o Índice de *Sharpe*, sujeito a diversas restrições. Além disso, discutiremos as limitações do Índice de *Sharpe* e exploraremos outras métricas de risco que podem

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

ser utilizadas na otimização de carteiras.

Antes de mais nada, é preciso instalar todos os pacotes necessários. Para isso, basta rodar o seguinte código:

```
# Função para instalar e carregar pacotes
instalar_e_carregar <- function(pacotes) {
novos_pacotes <- pacotes[!(pacotes %in% installed.packages()[, "Package"
   ])]
if (length(novos_pacotes)) install.packages(novos_pacotes)
sapply(pacotes, require, character.only = TRUE)  }
# Lista de pacotes necessários
pacotes <- c("tidyverse", "PortfolioAnalytics", "PerformanceAnalytics",
   "quantmod", "timetk")
# Instalar e carregar os pacotes
instalar_e_carregar(pacotes)
```

Após a instalação dos pacotes, vamos definir as ações que comporão a carteira, por exemplo, avaliando as empresas por seus indicadores de análise fundamentalista. A próxima etapa será obter os dados utilizando o pacote *quantmod*(**quantmod**) que nos permite obter preços de ações diretamente da internet pelo *site Yahoo Finance*(**yahoofi2023**). A seleção de ativos utilizada aqui é apenas um exemplo, e você pode escolher as ações que desejar.

```
# Definindo as ações que comporão a carteira
acoes <- c("VALE3.SA","PETR4.SA","CMIG4.SA","RADL3.SA","ITUB4.SA","^BVSP
   ")
# Obtendo os preços de ajustados das ações
prices_raw <-  getSymbols(tickers,
                      src = 'yahoo',
                      from = "2019-12-31",
                      to = "2024-05-31",
                      auto.assign = TRUE,
                      warnings = FALSE,)
prices<-prices_raw %>%   map(~Ad(get(.))) %>%
    reduce(merge) %>% `colnames<-`(tickers)
    prices_monthly <- to.monthly(prices, indexAt = "lastof", OHLC =
       FALSE)
```

O código acima obtém os preços mensais ajustados das ações selecionadas, no período de 31/12/2019 a 31/05/2024, e os armazena na variável *prices*.

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

O cálculo dos retornos logarítmicos, de acordo com a fórmula 1.3.1, pode ser feito de forma simples usando a função *Return.calculate* do pacote *PerformanceAnalytics* (**PerformanceAnalytics**). Em seguida é preciso separar os retornos dos ativos dos retornos do índice Bovespa, pois o índice será utilizado como *benchmark*.

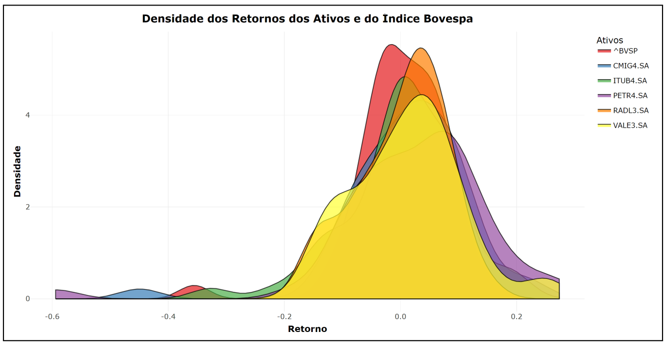
```
asset_returns_xts <-
PerformanceAnalytics::Return.calculate(prices_monthly,
        method = "log"
) %>%    na.omit()
# removendo o índice do IBOV da lista de ativos
retorno_ativos <- asset_returns_xts[, tickers[1:5]]
```

Agora que temos os retornos dos ativos e do índice Bovespa, podemos plotar um gráfico de densidade dos retornos dos ativos e do índice para visualizarmos a distribuição dos retornos. Assim, poderemos ter uma ideia da volatilidade dos retornos dos ativos e do índice. Para tanto, primeiro é necessário transformar o objeto *xts* em um *dataframe*, usando a função *tk_tbl* do pacote *timetk* (**timetk**), e, depois, a função *pivot_longer* do pacote *dplyr* (**wickham_dplyr_2022**) para transformar para o formato *long* e facilitar a plotagem do gráfico com o pacote *ggplot2* (**ggplot2**).

```
# Transforma o objeto xts em um dataframe e depois em formato longo
asset_returns_tbl <- asset_returns_xts %>%
tk_tbl(rename_index = "Data") %>% #converte em um data frame para plotar
    com ggplot
pivot_longer(!"Data" ,names_to = "assets")#seleciona todas as colunas
   menos data

# Plotando o gráfico de densidade dos retornos dos ativos e do índice
   Bovespa
ggplot(asset_returns_tbl) +
    aes(x = value, fill = assets) +
    geom_density(alpha = 0.7) +  # Adiciona transparência para sobreposi
       ção
    scale_fill_brewer(palette = "Set1") +
    labs(
    title = "Densidade dos Retornos dos Ativos e do Índice Bovespa",
    x = "Retorno",
```

```
    y = "Densidade",
    fill = "Ativos"
    ) +
    theme_minimal(base_size = 15) +  # Ajusta o tamanho base da fonte
    theme(
    plot.title = element_text(hjust = 0.5, face = "bold"),  #
        Centralizar e negritar o título
    axis.title.x = element_text(face = "bold"),  # Negritar o título do
        eixo x
    axis.title.y = element_text(face = "bold"),  # Negritar o título do
        eixo y
    legend.position = "bottom"  # Mover a legenda para a parte inferior)
```

Figura 1.6: Densidade dos retornos dos ativos e do índice Bovespa.



Fonte: Adaptado de (**galvaoFinancas**).

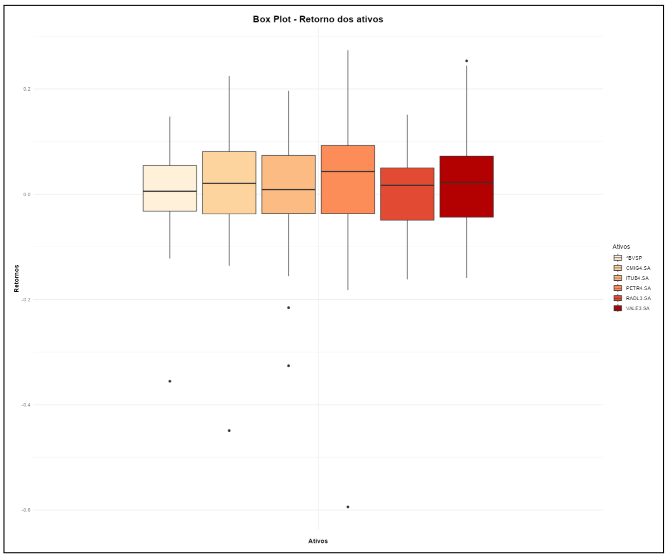
A partir do gráfico 1.6, podemos observar que os retornos dos ativos e do índice Bovespa têm distribuições semelhantes, o que é esperado, pois os retornos dos ativos influenciam o índice.

Outra forma de analisar os retornos é pelo *boxplot*, que nos permite visualizar e comparar melhor a distribuição dos retornos dos ativos e do índice Bovespa e identificar possíveis *outliers*.

```
# Plotando o boxplot dos retornos dos ativos e do índice Bovespa
ggplot(asset_returns_tbl) +
aes(x = "", y = value, fill = assets) +
geom_boxplot() +
scale_fill_brewer(palette = "OrRd", direction = 1) +
labs( x = "Ativos",
    y = "Retornos",
    title = "Box Plot - Retorno dos ativos",
```

```
        fill = "Ativos"  ) +
    theme_minimal() +
    theme(plot.title = element_text(size = 16L,
        face = "bold",
        hjust = 0.5),
        axis.title.y = element_text(face = "bold"),
        axis.title.x = element_text(face = "bold") )
```

Figura 1.7: Boxplot dos retornos dos ativos e do índice Bovespa.



Fonte: Adaptado de (**galvaoFinancas**).

Na figura 1.7, podemos observar a presença de *outliers* nos retornos dos ativos e do índice Bovespa, CEMIG, Itaú e Petrobras, o que indica que esses ativos tiveram retornos atípicos em algum período e podem ter influenciado a média dos retornos.

A próxima etapa é calcular o retorno e o risco da carteira utilizando a função *Return.portfolio* do pacote *PerformanceAnalytics* (**PerformanceAnalytics**). Esta função calcula o retorno do portfólio com base nos retornos dos ativos e seus respectivos pesos. Antes disso, precisamos definir os pesos dos ativos na carteira. Embora os pesos possam ser definidos de forma igualitária (1/N, onde N é o número de ativos), neste exemplo utilizaremos pesos otimizados para maximizar o retorno, o índice de *Sharpe* e minimizar o risco.

A otimização dos pesos da carteira é feita por meio da função *optimize.portfolio* do pacote *PortfolioAnalytics* (**PortfolioAnalytics**). Esta função recebe como argumentos os retornos dos ativos, um objeto que contém uma lista de restrições, objetivos e o método de otimização como mostrado no código abaixo.

```
portf <- portfolio.spec(assets = colnames(retorno_ativos))
portf <- add.constraint(portf,  type = "weight_sum",
    min_sum = 0.99, max_sum = 1.01 )
portf <- add.constraint(portf,
    type = "long_only"   )
portf <- add.constraint(portf,
 type = "box", min = 0.05, max = 1)
portf <- add.objective(portf,
    type = "risk", name = "StdDev"  )
portf <- add.objective(portf,
    type = "return", name = "mean"  )
opt_result <- optimize.portfolio(
    R = retorno_ativos,
    portfolio = portf,
    optimize_method = "ROI",
    maxSR = TRUE, trace = TRUE  )
    w <- extractWeights(opt_result)
    portfolio_returns_xts <- PerformanceAnalytics::Return.portfolio(
       retorno_ativos,
    weights = w,
    type = "discrete",
    verbose = FALSE ) %>% `colnames<-`("CARTEIRA")
portfolio_returns_df <- portfolio_returns_xts %>% tk_tbl(rename_index =
   "Data")
```

No código acima, a variável *portf* representa um objeto que encapsula as restrições e objetivos da otimização de portfólio. A função *add.constraint* é empregada para impor restrições ao objeto *portf*, assegurando que a soma dos pesos dos ativos seja igual a 0.8, em função da restrição de participação mínima. A função *add.objective* é utilizada para definir os objetivos da otimização, como a minimização do risco e a maximização do retorno. A função *optimize.portfolio* realiza a otimização dos pesos da carteira com base nos retornos dos ativos, nas restrições e nos objetivos previamente estabelecidos. Por fim, a função *extractWeights* extrai

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

os pesos otimizados da carteira.

O objeto de portfólio é uma estrutura que contém as seguintes informações:

- **assets**: vetor de nomes dos ativos.
- **constraints**: lista de restrições, incluindo:
  - **weight_sum**: restrição de que a soma dos pesos dos ativos deve ser igual a um valor específico.
  - **full_investment**: restrição de que a soma dos pesos dos ativos deve ser igual a 1.
  - **dollar_neutral**: restrição de que a soma dos pesos dos ativos deve ser igual a zero.
  - **box**: restrição de que os pesos dos ativos devem estar dentro de um intervalo específico.
  - **long_only**: restrição de que os pesos dos ativos devem ser maiores ou iguais a zero e menores ou iguais a um (caso especial da restrição box).
  - **group**: restrição de que os pesos dos ativos de um grupo específico devem ser iguais.
  - **turnover**: restrição de que o *turnover* da carteira deve ser menor ou igual a um valor específico a partir de uma carteira inicial.
  - **diversification**: especifica um valor de diversificação para a carteira.
  - **position_limit**: permite ao usuário definir o número máximo de posições (ativos com pesos não nulos) e o número máximo de posições compradas e vendidas.
  - **return**: especifica o retorno médio esperado da carteira.
  - **factor_exposure**: especifica a exposição da carteira a fatores de risco.

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

  - **leverage_exposure**: especifica a alavancagem máxima da carteira, onde a alavancagem é definida como a soma dos pesos absolutos dos ativos.

- **objectives**: lista de objetivos, incluindo:
  - **type**: 'return', 'risk', 'risk_budget', 'quadratic_utility', ou 'weight_concentration'.

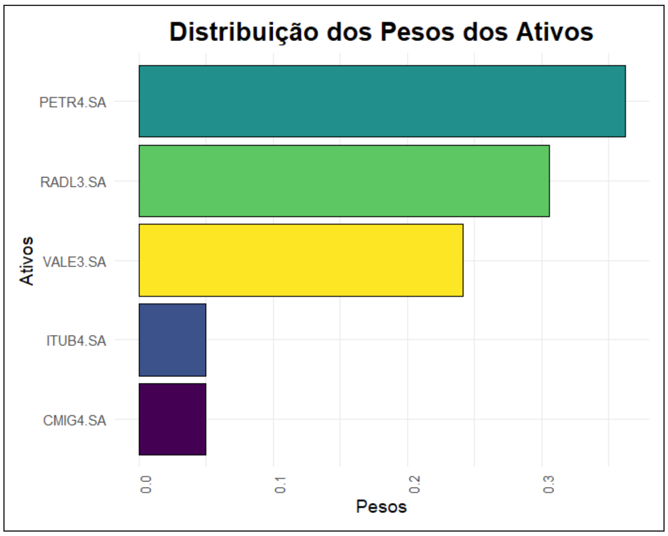
A função *optimize.portfolio* também recebe dois argumentos adicionais: *maxSR = TRUE*, que maximiza o índice de *Sharpe*, e *trace = TRUE*, que exibe o log da otimização. Vale ressaltar que o índice de *Sharpe* só pode ser usado quando são otimizados o risco e o retorno no mesmo problema. Além disso, o argumento *optimize_method = "ROI"* define o método de otimização utilizado, que, neste caso, seleciona o solver mais adequado dependendo da modelagem do problema de otimização.

A figura 1.8 mostra os pesos otimizados da carteira. Verificamos que Petrobras, Droga Raia e Vale possuem os maiores pesos na carteira.

```
weights <- data.frame(tickers = colnames(retorno_ativos), w, row.names =
    NULL)
ggplot(weights, aes(x = reorder(tickers, weights), y = weights, fill =
   tickers)) +
  geom_col(color = "black", show.legend = FALSE) +
  theme_minimal(base_size = 14) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1),
        plot.title = element_text(hjust = 0.5, size = 20, face = "bold")
            ,
        plot.subtitle = element_text(hjust = 0.5, size = 16),
        legend.position = "none") +
  scale_fill_viridis_d() +
  labs(title = "Distribuição dos Pesos dos Ativos",
       x = "Ativos",
       y = "Pesos") +
  coord_flip()
```

Por fim, é possível calcular o retorno, com base nos pesos otimizados, e comparar com o retorno do índice Bovespa, conforme o código abaixo:

Figura 1.8: Pesos dos ativos na carteira otimizada.

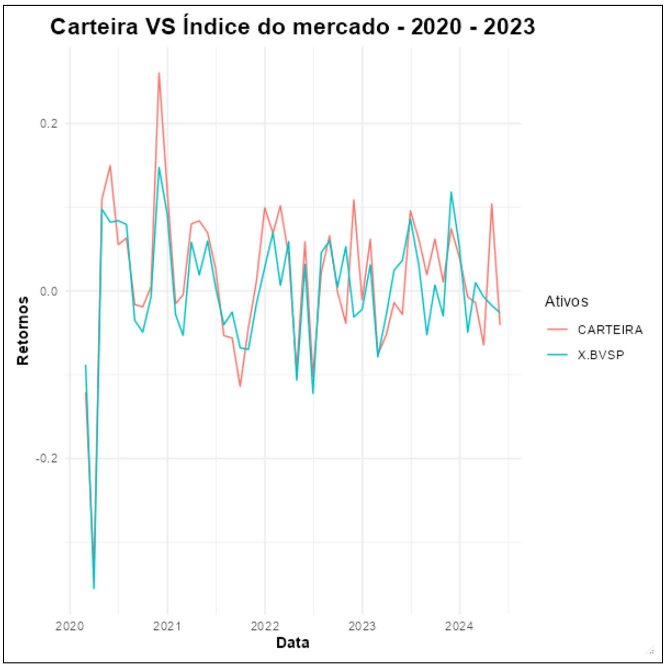


Fonte: Os autores, 2024.

```
comparacao <- cbind(asset_returns_xts, portfolio_returns_xts) %>%
tk_tbl(rename_index = "Data") %>%
pivot_longer(-Data, names_to = "assets", values_to = "value")
comp_mkt <- comparacao %>%
filter(assets %in% c("X.BVSP", "CARTEIRA")) %>%
ggplot() +  aes(x = Data, y = value, colour = assets) +
   geom_line() +
   scale_color_hue(direction = 1) +
   labs(title = "Carteira VS Índice do mercado - 2020 - 2023",
               x = "Data",
               y = "Retornos",
               colour = "Ativos" )+
   theme_minimal() +
   theme( plot.title = element_text(size = 16L, face = "bold", hjust =
      0.5),
       axis.title.y = element_text(face = "bold"),
       axis.title.x = element_text(face = "bold") )
```

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

Figura 1.9: Comparação dos retornos da carteira otimizada e do índice Bovespa.



Fonte: Adaptado de (**galvaoFinancas**).

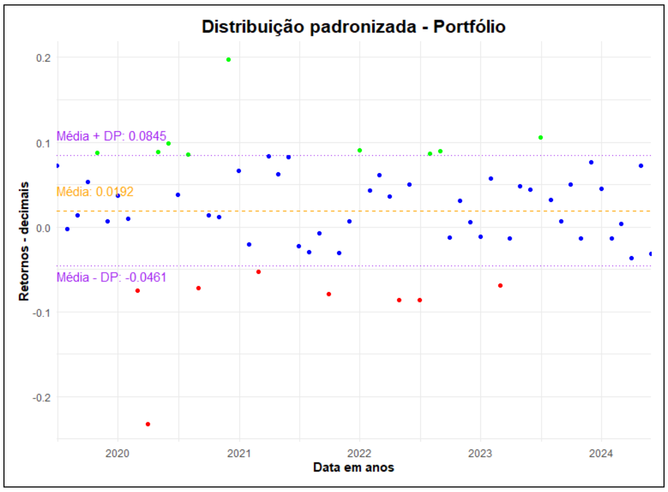
Na figura 1.9 podemos observar que a carteira otimizada superou o índice Bovespa em alguns períodos, o que indica que a otimização dos pesos dos ativos foi eficaz em maximizar o retorno e minimizar o risco da carteira, porém, por ser composta por ativos que influenciam fortemente o índice, a carteira não o conseguiu superar consistentemente.

Outra análise que pode ser feita é a visualização do desvio-padrão e do retorno médio da carteira ao longo do tempo. Para isso, basta utilizar o código abaixo:

```
mean_value <- mean(portfolio_returns_df$CARTEIRA, na.rm = TRUE)
sd_value <- sd(portfolio_returns_df$CARTEIRA, na.rm = TRUE)
portfolio_returns_df %>%
mutate(      faixa_inferior = if_else(CARTEIRA < (mean_value - sd_value)
    , CARTEIRA, as.numeric(NA)),
  faixa_superior = if_else(CARTEIRA > (mean_value + sd_value), CARTEIRA,
      as.numeric(NA)),
  faixa_central = if_else(CARTEIRA > (mean_value - sd_value) & CARTEIRA
     < (mean_value + sd_value), CARTEIRA, as.numeric(NA))
) %>%
ggplot() +
geom_point(aes(x = Data, y = faixa_inferior), color = "red") +
```

```
  geom_point(aes(x = Data, y = faixa_superior), color = "green") +
  geom_point(aes(x = Data, y = faixa_central), color = "blue") +
  geom_hline(yintercept = (mean_value + sd_value), color = "purple",
     linetype = "dotted") +
  geom_hline(yintercept = (mean_value - sd_value), color = "purple",
     linetype = "dotted") +
  geom_hline(yintercept = mean_value, color = "gray", linetype = "dashed")
      + # Linha tracejada para a média
  labs( x = "Data",  y = "Retornos",
    title = "Distribuição padronizada - Portfólio",
    color = "Ativo" ) +   theme_minimal() +
  theme( plot.title = element_text(size = 16L, face = "bold", hjust = 0.5)
     ,
    axis.title.y = element_text(face = "bold"),
    axis.title.x = element_text(face = "bold") ))
```

Figura 1.10: Distribuição padronizada do retorno da carteira.



Fonte: Adaptado de (**galvaoFinancas**).

Na figura 1.10 podemos observar que a maior parte dos retornos da carteira está dentro do intervalo de um desvio-padrão da média, o que indica que a carteira é relativamente estável e não tem retornos extremos, com exceção de alguns períodos, como em 2020, durante a pandemia de COVID-19, e em 2022, ano de eleições presidenciais.

Pode-se também fazer a otimização com o rebalanceamento da carteira. O rebalanceamento da carteira é uma prática comum no mercado financeiro que

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

consiste em ajustar os pesos dos ativos na carteira para manter a alocação de ativos desejada. Pode ser feito de forma periódica, mensal, trimestral, semestral ou anual, ou de forma dinâmica, com base em eventos específicos, como a mudança nas condições de mercado ou nos objetivos de investimento.

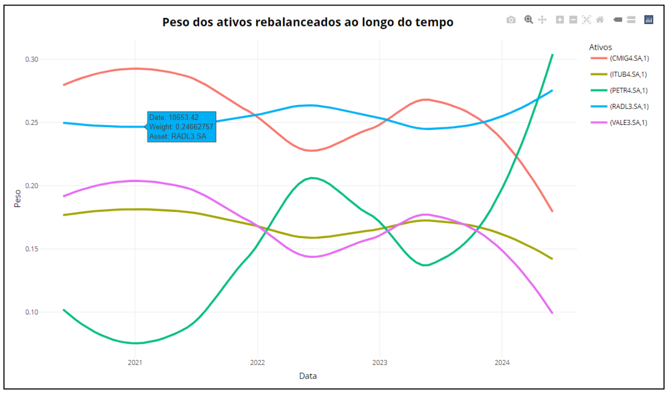
A função *optimize.portfolio.rebalancing()* é utilizada para otimizar os pesos da carteira com rebalanceamento. Esta função recebe como argumentos os retornos dos ativos, um objeto que contém uma lista de restrições, objetivos e o método de otimização, a frequência de rebalanceamento e o método de rebalanceamento, como pode ser visto no código abaixo:

```
    # Criar o objeto de portfólio
    port_specr <- portfolio.spec(assets = colnames(asset_returns_xts))
    # Adicionar restrições - full investment com soma de pesos igual a 1
    port_specr <- add.constraint(port_specr, type = "long_only")
    # restrição para a soma dos pesos
    port_specr <- add.constraint(port_specr, type = "weight_sum", min_sum =
       0.99, max_sum = 1.01)
    # Adicionar a função objetivo - maximizar o retorno esperado
    port_specr <- add.objective(port_specr, type = "return", name = "mean")
    # Adicionar a função objetivo - minimizar o risco
    port_specr <- add.objective(port_specr, type = "risk", name = "StdDev")
    # Adicionar a função objetivo - minimizar a contribuição marginal do
       risco de cada ativo para o risco total da carteira
    port_specr <- add.objective(port_specr, type = "risk_budget", name = "
       StdDev", min_prisk = 0.05, max_prisk = 0.1)
    rp <- random_portfolios(port_specr, permutations = 50, rp_method = '
       simplex')
    # Otimização com rebalanceamento mensal
opt_rebal <- optimize.portfolio.rebalancing(R = asset_returns_xts,
                     portfolio = port_specr,
                     optimize_method = "random",
                     rp = rp,
                     trace = TRUE,
                     search_size = 1000,
                     rebalance_on = "months",
                     training_period = 12,
                     rolling_window = 12,
                     maxSR=TRUE)
print(opt_rebal)
```

No código acima os argumentos *rebalance_on = "months"* e *training_period = 12* definem que o rebalanceamento será feito mensalmente e que o período de treinamento será de 12 meses. O argumento *rolling_window = 12* define que o período de rebalanceamento será de 12 meses. O argumento *maxSR = TRUE* maximiza o índice de *Sharpe*. Já o argumento *search_size = 1000* define o tamanho da busca aleatória para a otimização. O argumento *rp = rp* define os portfólios aleatórios gerados para a otimização.

A figura 1.11 mostra a evolução dos pesos dos ativos na carteira com rebalanceamento ao longo do tempo.

Figura 1.11: Evolução dos pesos dos ativos na carteira com rebalanceamento.



Fonte: Adaptado de (**galvaoFinancas**).

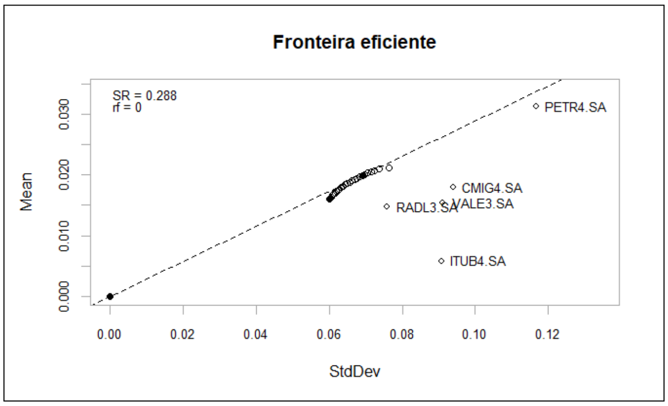
```
# Extraindo os pesos dos ativos ao longo do tempo
weights_xts <- extractWeights(opt_rebal)
# Convertendo os pesos para um data frame
weights_df <- weights_xts %>%
              tk_tbl(preserve_index = TRUE, rename_index = "Date")
# Colocando o data frame em formato longo
weights_long_df <- weights_df %>%
                   pivot_longer(cols = -Date, names_to = "Asset", values_to
                       = "Weight")
plotly::ggplotly(ggplot(weights_long_df) +
  aes(x = Date, y = Weight, colour = Asset) +
  geom_smooth(se = FALSE ) +
  scale_fill_hue(direction = 1) +
  labs(  x = "Data",  y = "Peso",
    title = "Peso dos ativos rebalanceados ao longo do tempo",
    colour = "Ativos") +   theme_minimal() +
```

```
theme( plot.title = element_text(size = 16L,
  face = "bold",hjust = 0.5)))
```

A fronteira eficiente pode ser visualizada utilizando-se as funções *create.EfficientFrontier* e *chart.EfficientFrontier* do pacote *PortfolioAnalytics* (**PortfolioAnalytics**). Esta curva representa todas as combinações de ativos que maximizam o retorno esperado para um dado nível de risco. A figura 1.12 ilustra a fronteira eficiente para a carteira otimizada. O código para gerar a fronteira eficiente é mostrado abaixo, considerando uma taxa de retorno livre de risco de 0. Deve-se ressaltar que foram usados 30 portfólios que buscam aproximar a fronteira eficiente, o uso de mais portfólios pode melhorar a visualização da fronteira eficiente, porém aumenta o tempo de processamento.

```
# Calculando a fronteira eficiente
frontier <- create.EfficientFrontier(R = asset_returns_xts, portfolio =
   port_spec, type = "mean-StdDev")
# Plotando a fronteira eficiente
chart.EfficientFrontier(frontier, match.col = "StdDev",n.portfolios =
   30, main = "Fronteira eficiente")
```

Figura 1.12: Fronteira eficiente.



Fonte: (**galvaoFinancas**).

1.5 CONSIDERAÇÕES FINAIS

O presente estudo se insere no escopo da otimização de carteiras de investimento, um campo de pesquisa tradicional em finanças quantitativas que busca determinar a alocação ótima de recursos entre diferentes ativos, considerando a relação risco-retorno. A relevância do tema reside na busca incessante por parte de investidores por estratégias que maximizem a rentabilidade de seus investimentos e minimizem a exposição a riscos indesejados.

Portanto, foi apresentado um arcabouço teórico-metodológico para a otimização de carteiras de ações, utilizando o Índice de *Sharpe* como critério de decisão e a linguagem R como ferramenta de aplicação. A escolha do Índice de *Sharpe* se justifica por sua ampla difusão no meio acadêmico e profissional, e pela sua capacidade de capturar, em uma única métrica, a relação entre risco e retorno.

O estudo apresenta algumas limitações, pois o foco está no processo metodológico usando a linguagem R e não na teoria. Portanto, embora tenha abrangido os conceitos basilares da Moderna Teoria de Portfólios, não explorou a totalidade da temática, optando por omitir temas relevantes como o Modelo de Precificação de Ativos de Capital (CAPM), o modelo de Precificação por Arbitragem (APT) e outras métricas de risco-retorno, como o Índice de *Treynor*. Em trabalhos futuros, a incorporação de tais temas poderá contribuir para uma análise mais completa do tema.

Em segundo lugar, as escolhas metodológicas realizadas, tais como a seleção das empresas que compuseram a carteira, a definição do período de rebalanceamento, o estabelecimento do mínimo de participação por ativo e a utilização do próprio Índice de *Sharpe* como métrica de otimização, foram motivadas por uma busca por parcimônia e clareza expositiva, mas podem ser reavaliadas e expandidas em trabalhos subsequentes. Outros métodos de otimização, menos suscetíveis à natureza dos dados e que considerem outros tipos de risco, como o *drawdown*, poderiam ser explorados, além da investigação da sensibilidade à variação dos parâmetros na composição da carteira.

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

Capítulo 2

Uma Introdução à Programação com o R

Autor: Jessica Kubrusly
Instituição: Universidade Federal Fluminense
e-mail: jessicakubrusly@id.uff.br

2.1 INTRODUÇÃO

Os profissionais dispostos a trabalhar com análise de dados precisam não só dominar o uso de pacotes em uma linguagem de programação, conhecidos também como bibliotecas, como também a linguagem de programação em si. Saber criar sua próprias rotinas garante a estes profissionais maior flexibilidade em sua programação e, consequentemente, uma maior capacidade de resolver problemas.

Seja no Programa R, no Python ou em outra linguagem de programação qualquer, um cientista de dados precisa dominar os controles de fluxos, conhecer a estrutura da linguagem de programação e saber criar funções. Com isso será possível combinar os recursos dos pacotes e funções já existentes com as particularidades da sua demanda e assim se tornar um profissional mais completo.

Neste capítulo será apresentado, de forma breve, como podemos usar o Programa R como linguagem de programação. O capítulo foi inspirado no livro **kubrusly2021**, de mesmo título e autoria, onde pode ser encontrada uma discussão mais ampla sobre o assunto. Aqui, primeiro serão apresentados os objetos básicos do Programa R, os controles de fluxo e o construtor de funções. Em seguida três problemas práticos serão simulados e resolvidos com o auxílio do computador.

2.2 OBJETIVOS

O objetivo geral deste capítulo é apresentar o uso do Programa R como linguagem de programação. De forma específica, os objetivos são: apresentar os controles de fluxo; motivar a criação de funções; exemplificar simulações e soluções de problemas práticos. Sempre contextualizado no Programa R.

2.3 OBJETOS NO R

O Programa R permite que você salve dados armazenando-os dentro de um objeto R. Mas o que é um objeto? É apenas um nome que você pode usar para acessar os dados armazenados (**grolemund2014hands**). A seguir serão apresentados alguns dos objetos mais básicos do R e como trabalhar com cada um deles.

2.3.1 Números

No R, quando se define um objeto x por um número qualquer, cria-se um objeto do tipo `numeric`. A partir deles é possível fazer operações matemáticas.

```
> x = 5
> x + 4
[1] 9
```

O R pode ser usado como uma grande calculadora. Ele realiza operações de soma, subtração, multiplicação, divisão, potencialização, entre outras.

```
> x^3
[1] 125
```

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

O exemplo anterior mostra o comando ^ (acento circunflexo), que realiza a operação de potencialização. O número que vem à esquerda é elevado ao número que aparece à direta.

O próximo exemplo ilustra a importância do uso dos parênteses para que em uma única linha de comando seja possível realizar uma expressão mais complexa.

```
> ((x^3 + 2*x - 1)/(x-3) + x^2)/4
[1] 23
```

Além das operações básicas de +, -, * e \ é possível usar o R para calcular as funções matemáticas mais conhecidas, como por exemplo a Função Logarítimica.

```
> log((2*x - 1)/3)
[1] 1.098612
```

No R o comando `log` se refere ao Logaritmo Neperiano, isto é, na base e. Caso se queira usar outra base, basta indicar a base como um segundo argumento da função `log`.

```
> log(2,base=10)
[1] 0.30103
```

A seguir, exemplos das funções Exponencial, Seno e Cosseno. Primeiro o valor de e^2, sendo e o número irracional que é base do Logarítmo Neperiano.

```
> exp(2)
[1] 7.389056
```

O exemplo a seguir calcula o valor de $\cos(\pi)$. No R a variável `pi` guardo o valor (aproximado) para π.

```
> cos(pi)
[1] -1
```

Agora o cálculo de $\operatorname{sen}(\pi)$.

```
> sin(pi)
[1] 1.224606e-16
```

O número resultante desta operação `1.224606e-16` representa $1,224606 \times 10^{-16}$. Sabe-se que $sen(\pi) = 0$, mas o que o programa R retorna para a função `sin` é uma aproximação para o valor do seno e não o valor exato. E na hora de fazer essa aproximação ele chega muito perto de zero, mas não exatamente no zero. Para saber mais sobre aproximações de funções veja a Parte II de (**kubrusly2021**).

2.3.2 Caracteres

Outro tipo bem comum de dados são os caracteres. Tratam-se de objetos entendidos como texto.

```
> a = "oi"
> a
[1] "oi"
```

Veja que o espaço também é um caracter, então ele pode fazer parte de um texto armazenado em uma variável.

```
> b = "com vai?"
> b
[1] "com vai?"
```

Quando se trabalha com objetos do tipo caracteres, é comum o uso de alguns comandos e funções específicas para esse tipo de objeto. Por exemplo, a função `paste`, que une, ou "cola", objetos de texto em um único texto.

```
> paste(a,b)
[1] "oi como vai?"
```

Essa função de forma padrão junta os objetos com um espaço entre eles. Para unir os objetos sem esse espaço entre eles existe a função `paste0`.

```
> paste0(a,b)
[1] "oicom vai?"
```

2.3.3 Lógicos

Os operadores lógicos são uma ferramenta importante em qualquer linguagem de programação. No `R` são eles: `T` ou `TRUE` para representar a opção VERDADEIRO; e `F` ou `FALSE` para a FALSA.

```
> 3 < 2
[1] FALSE
> 2 == 2.0
[1] TRUE
> 2 != 2.0
[1] FALSE
> x >= 0
[1] TRUE
```

Lembre-se que no código acima o valor guardado no objeto `x` é o número 5.

Entre objetos lógicos é possível realizar operações como interseção e união a partir dos operadores lógicos `&` e |, respectivamente.

```
> (3==4) & (2>=1)
[1] FALSE
> (3==3) | (2>=1)
[1] TRUE
```

2.4 VETORES

Vetores atômicos, ou simplesmente vetores, são objetos existentes em toda linguagem de programação. De acordo com (**grolemund2014hands**), um vetor atômico é apenas um vetor simples de dados. Trata-se de uma lista de objetos, todos do mesmo tipo, que podem ser acessados pelo seu índice.

Existem quatro tipos principais de vetores atômicos: lógico, inteiro, *double* e caracter. Os vetores inteiros e *double* são conhecidos como vetores do tipo `numeric` (**wickham2019advanced**).

No `R` a função `c` concatena objetos do mesmo tipo em forma de vetores. Para os exemplos a seguir, `y` é um vetor de números, `texto1` é um vetor de caracteres e `logico` é um vetor de lógicos.

```
> y = c(2,3,4,5,6)
> y
[1] 2 3 4 5 6
```

```
> texto1 = c("A","AA","ABB")
```

```
> texto1
[1] "A"   "AA"  "ABB"
```

```
> logico = c(F,F,T)
> logico
[1] FALSE FALSE  TRUE
```

Os vetores numéricos também podem ser criados com o comando `:`, onde à esquerda é informado o primeiro numero do vetor e à esquerda o último. Cada posição deste vetor é definido como uma unidade a mais (ou a menos) da posição anterior. Veja dois exemplos.

```
> -1:5
[1] -1  0  1  2  3  4  5
> 15:5
[1] 15 14 13 12 11 10  9  8  7  6  5
```

A função `c` pode ser usada não só para concatenar objetos como também para concatenar vetores.

```
> w=0:-4;w
[1]  0 -1 -2 -3 -4
> y
[1] 2 3 4 5 6
> c(w,y)
[1]  0 -1 -2 -3 -4  2  3  4  5  6
> c(y,w)
[1]  2  3  4  5  6  0 -1 -2 -3 -4
```

Essa operação pode ser realizada entre qualquer tipo de vetor.

```
> texto2 = c("1","2","3","4","5","6")
> c(texto1,texto2)
[1] "A"   "AA"  "ABB" "1"   "2"   "3"   "4"   "5"   "6"
> a
[1] " oi "
> b
[1] " com vai ? "
> ab = c(a,b)
> ab
[1] " oi "        " com vai ? "
```

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

Quando se trabalha com vetores é possível realizar operações já apresentadas. Por exemplo, a partir do vetor y é possível somar 4 unidades a cada elemento com o comando a seguir.

```
> y
[1] 2 3 4 5 6
> y+4
[1]  6  7  8  9 10
```

Com dois vetores do mesmo tipo e do mesmo tamanho, é possível realizar operações entre eles e tais operações serão realizadas em cada posição, uma a uma.

```
> y
[1] 2 3 4 5 6
> w
[1]  0 -1 -2 -3 -4
> y+w
[1] 2 2 2 2 2
```

Se os vetores não têm o mesmo tamanho, mas o tamanho de um é múltiplo do tamanho do outro, o de menor tamanho será sequenciado as vezes necessárias para obter o mesmo tamanho do maior e então a operação será realizada posição a posição.

```
> y
[1] 2 3 4 5 6
> z = 1:15
> z
[1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
> y+z
[1]  3  5  7  9 11  8 10 12 14 16 13 15 17 19 21
```

Também é possível realizar operações próprias para caracteres entre vetores de caracteres.

```
> texto1
[1] "A"   "AA"  "ABB"
> paste(texto1,"0")
[1] "A 0"   "AA 0"  "ABB 0"
> texto3 = c("0","00","000","0000","00000","000000")
```

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

```
> texto3
[1] "0"      "00"     "000"    "0000"   "00000"  "000000"
> paste(texto1,texto3)
[1] "A 0"         "AA 00"       "ABB 000"     "A 0000"      "AA 00000"    "
   ABB 000000"
```

O mesmo vale para vetores de objetos lógicos.

```
> logico
[1] FALSE FALSE  TRUE
> logico & TRUE
[1] FALSE FALSE  TRUE
> logico & c(TRUE,FALSE,FALSE)
[1] FALSE FALSE FALSE
> logico | TRUE
[1] TRUE TRUE TRUE
> logico | c(TRUE,FALSE,FALSE)
[1]  TRUE FALSE  TRUE
```

2.5 CONTROLES DE FLUXO

Os controles de fluxo são operações definidas em todas as linguagens de programação. Como apresentou (**wickham2019advanced**), há duas ferramentas principais de controle de fluxo: escolhas e laços. As escolhas, como o `if`, permitem que você execute diferentes códigos dependendo da entrada. Os laços, também conhecidos pelo nome em inglês *loops*, como o `for`, permitem que você execute repetições de uma tarefa, o que os torna uma ferramenta útil para programar simulações (**grolemund2014hands**).

Cada linguagem de programação tem a sua própria sintaxe, isto é, sua própria estrutura ou regra de como essas operações devem ser usadas. Veja alguns exemplos de controles de fluxo mais comuns e suas sintaxes.

2.5.1 If/else

A sintaxe do comando `if/else`, que pode ser também somente `if`, é dada da seguinte forma:

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

```
if (objeto lógico) {
#comandos caso o objeto seja TRUE
} else {
#comandos caso o objeto seja FALSE
}
```

ou, quando não há algo a ser executado quando o objeto for `FALSE`,

```
if (objeto lógico) {
#comandos caso o objeto seja TRUE
}
```

Seguem alguns exemplos práticos. O primeiro deles é um código que imprime um texto na tela que depende do valor guardado na variável `x`.

```
> x = 2
> if( x < 5 ){
+     print(paste(x,"e menor que",5))
+ } else {
+     print(paste(x,"e maior ou igual a",5))
+ }
[1] "2 e menor que 5"
```

Outro valor para `x` pode resultar em outro texto impresso na tela. Ou seja, o texto depende do valor guardado na variável `x`.

```
> x = 10
> if( x < 5 ){
+     print(paste(x,"e menor que",5))
+ } else {
+     print(paste(x,"e maior ou igual a",5))
+ }
[1] "10 e maior ou igual a 5"
```

O segundo exemplo deste controle de fluxo atribuí à variável `y` um valor que depende do valor guardado na variável `x`. Dessa forma, se o valor de `x` no início do código mudar, o valor de `y` mudará também.

```
> x = 3
> if (x > 2){
+     y <- 2*x
+ } else {
+     y <- 3*x
```

Qual o valor da variável `y` ao final desse código? Você consegue descobrir a resposta? Respondeu certo quem disse `6`.

2.5.2 for

Veja a sintaxe para se criar um comando com o `for` no R.

```
for(i in valores){
#comandos, que em geral dependem do valor de i
}
```

Dentro do par de parênteses está o vetor `valores`, que é um vetor de objetos e que pode ser de qualquer tipo. Os comandos de dentro do par de chaves serão executados, repetidamente, e em cada iteração o objeto `i` assume um valor diferente, valor esse guardado no vetor `valores`.

```
> y = 0
> for(i in 1:10){
+     y = y+1
+ }
> y
[1] 10
```

A variável `y` começa com o valor `0`. Quando `i=1`, `y` é incrementado de `1` unidade e passa a guardar o valor `0 + 1 = 1`. Quando `i=2`, `y` é incrementado de mais `1` unidade e passa a guardar o valor `1 + 1 = 2`. Assim por diante até que temos `i = 10`, quando `y` recebe seu último incremente e passa a guardar o valor `10`.

O próximo exemplo apresenta um for considerando um vetor de valores não numéricos.

```
> alfabeto = NULL
> for(a in LETTERS){
+     alfabeto = paste(alfabeto,a)
+ }
> alfabeto
[1] " A B C D E F G H I J K L M N O P Q R S T U V W X Y Z"
```

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

Veja como uma pequena mudança no código pode mudar totalmente o resultado final.

```
> alfabeto = NULL
> for(a in LETTERS){
+     alfabeto = paste(a,alfabeto)
+ }
> alfabeto
[1] "Z Y X W V U T S R Q P O N M L K J I H G F E D C B A "
```

Qual a diferença entre o último e o penúltimo código? O que isso resultou de diferente na execução dos comandos?

2.5.3 while

Uma outra opção de controle de floxo de repetição é o `while`. Este controle repete os comandos enquanto uma certa condição for verdadeira. Esta condição está representada na sintaxe, do exemplo abaixo, como `objeto lógico`.

```
while (objeto lógico) {
# comandos
}
```

Os comandos dentro das chaves são executados repetidamente enquanto `objeto lógico` for verdadeiro. É importante garantir que em algum momento `objeto lógico` torna-se falso, senão o programa não para de rodar e isto é chamado de *loop* infinito.

Seguem um exemplo.

```
> vet = 1
> while(length(vet) < 100){
+     i = length(vet)
+     vet[i+1] = i+1
+ }
> vet
[1]    1  2  3  4  5  6  7  8  9 10 11 12 13 14
[15]  15 16 17 18 19 20 21 22 23 24 25 26 27 28
[29]  29 30 31 32 33 34 35 36 37 38 39 40 41 42
[43]  43 44 45 46 47 48 49 50 51 52 53 54 55 56
[57]  57 58 59 60 61 62 63 64 65 66 67 68 69 70
```

```
[71]  71  72  73  74  75  76  77  78  79  80  81  82  83  84
[85]  85  86  87  88  89  90  91  92  93  94  95  96  97  98
[99]  99 100
```

O `for` e o `while` podem ser trocados um pelo outro sempre. A escolha entre um dos dois será feita baseada naquele controle de fluxo que torna o código mais limpo e claro.

2.6 FUNÇÕES

O R é uma linguagem funcional, o que significa que ela possui certas propriedades técnicas, mas, mais importante do que isso, ela segue um estilo de resolução de problemas centrado em funções (**wickham2019advanced**). A criação de funções no R permite que o programador organize seu código de forma que as operações repetidas muitas vezes possam ser chamas com um único comando, que é a chamada da função.

Toda função no `R` tem três partes básicas: um nome, um corpo de código e um conjunto de argumentos (**grolemund2014hands**).

A seguir um exemplo de como é a sintaxe do construtor de funções, chamado de `function`.

```
> nome_da_funcao = function(argumento1,argumento2){
+     # operações que dependem dos argumentos de entrada,
+     # por exemplo:
+     saida = (argumento1 + argumento2 + 10)/2
+     # o comando return define o que a função retorna como saída
+     return(saida)
+ }
> nome_da_funcao(3,4)
[1] 8.5
```

Mais exemplos de criação de função serão apresentados nas próximas seções. Tais exemplos serão expostos durante a solução dos problemas trabalhados nas seções que seguem.

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

2.7 PROBLEMAS

Esta última apresenta três problemas reais que serão explorados e irão exemplificar o uso dos conceitos vistos nas primeiras seções.

Problema de Investimento

Suponha que um indivíduo tenha R$1.000,00 para investir na poupança. Suponha que a poupança rende 0,5% ao mês. Como usar o Programa R para saber o montante que este indivíduo terá depois de 3 anos do investimento inicial?

Se essa conta for feita no “lápis e papel”, ela seria resolvida mais ou menos assim.

$$\begin{aligned} v_0 &= 1000 \\ v_1 &= 1000 + (0,5/100) \times 1000 = 1005 \\ v_2 &= 1005 + (0,5/100) \times 1005 = 1.010,025 \\ &\vdots \end{aligned}$$

Mas são muitas contas repetidas. Para chegar no valor investido depois de 3 anos é preciso realizar essa operação para cada um dos 36 meses. Muito trabalhoso e repetitivo. Tudo que é trabalhoso e repetitivo pode ser facilitado com o uso do computador.

```
> valor = 1000
> for(i in 1:36){
+     valor = valor + valor*0.5/100
+ }
> valor
[1] 1196.681
```

Então, depois de 3 anos o indivíduo passa a ter R$ 1.196,68 investido. E depois de 10 anos? Veja que dez anos são 120 meses.

```
> valor = 1000
> for(i in 1:120){
+     valor = valor + valor*0.5/100
+ }
```

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

```
> valor
[1] 1819.397
```

Talvez esse indivíduo está na dúvida se deve investir todo o seu dinheiro ou parte dele. Qual seria a diferente do montante guardado após 10 anos se em vez de R$ 1.000,00 o investimento inicial fosse de R$ 900,00?

```
> valor = 900
> for(i in 1:120){
+     valor = valor + valor*0.5/100
+ }
> valor
[1] 1637.457
```

Ou seja, se forem guardados R$ 1.000,00 por 10 anos nesse investimento ao final do período o montante acumulado será de R$ 1.819,39. Já se forem guardados 100 reais a menos, isto é, R$ 900,00 , depois de 10 anos o montante acumulado será de R$ 1.637,45.

Suponha agora que exista a opção de escolher entre 2 investimentos distintos, um com o rendimento de 0,5% e outro de 0,6% ao mês. Qual seria a diferença no montante guardado depois de 10 anos?

```
> valor = 1000
> for(i in 1:120){
+     valor = valor + valor*0.6/100
+ }
> valor
[1] 2050.018
```

Se o investimento rendesse 0,6% ao mês e não 0,5%, em vez de R$ 1.819,39 o montante guardado seria de R$ 2.050,01.

Veja que é possível descobrir o valor do montante guardado para diferentes valores de: investimento inicial, rendimento, período. Uma alternativa para facilitar essa conta e as análises que surgirão é a criação de uma função que faça essa conta para os valores escolhidos.

```
> investimento = function(valor,n,taxa){
+     for(i in 1:n){
```

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

```
+          valor = valor + valor*taxa/100
+      }
+      return(valor)
+ }
```

A partir da função `investimento` recém criada é possível calcular muitas possibilidades de investimento de forma mais direta e com o código mais limpo.

```
> investimento(1000,36,0.5)
[1] 1196.681
> investimento(1000,120,0.5)
[1] 1819.397
> investimento(900,120,0.5)
[1] 1637.457
> investimento(1000,120,0.6)
[1] 2050.018
```

Outra pergunta que talvez o indivíduo queira saber é: quantos meses é preciso esperar até conseguir juntar uma certa quantia? Aqui as contas serão feitas em repetição não como antes, que era um número fixo de vezes, 36 para 3 anos ou 120 para 10 anos, mas sim até que o montante acumulado seja igual ou maior do que a quantia definida.

Suponha que deseja-se juntar R$ 15.000,00, quantos meses é preciso esperar considerando um investimento inicial de R$ 1.000,00 e juros de 0,5% ao mês?

```
> valor = 1000
> r = 0.5
> parcelas = 60
> n = 1
> while(valor < 15000){
+     valor = valor + valor*r/100 + parcelas
+     n = n + 1
+ }
> n
[1] 148
> valor
[1] 15061.66
```

Ao final do código a variável `n` guarda o número de meses que se passaram e a variável `valor` guarda o valor total acumulado.

Veja uma função para avaliar diferentes situações desse mesmo problema.

```
> meses_para_juntar = function(montante,investimento,parcelas,r){
+     n = 1
+     valor = investimento
+     repeat{
+         valor = valor + valor*r/100 + parcelas
+         n = n + 1
+         if(valor > montante){
+             break
+         }
+     }
+     return(c(n,valor))
+ }
```

A função irá retornar um vetor de valores. A primeira posição desse vetor será a quantidade de meses para juntar a quantia estipulada. A segunda posição será o valor final depois dessa quantidade de meses.

```
> meses_para_juntar(15000,1000,60,0.5)
[1]   148.00 15061.66
> meses_para_juntar(20000,1200,100,0.5)
[1]   129.00 20141.33
> meses_para_juntar(15000,1000,60,0.5)
[1]   148.00 15061.66
```

Problema de Crescimento Populacional

A proposta deste segundo problema é simular o crescimento de uma população de microrganismos. Para isso, primeiro algumas definições.

Chama-se de **taxa de nascimento** semanal a proporção de nascimentos, por semana, em relação ao tamanho da população. Chama-se de **taxa de mortalidade** semanal a proporção de mortes, por semana, em relação ao tamanho da população.

Suponha uma população inicial de 1000 microrganismos, uma taxa de nascimento de 20% e uma taxa de mortalidade de 5%. Como usar o computador, isto é, como simular, para descobrir o tamanho desta população depois de 4 semanas?

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

Considere o código a seguir, onde, a partir do `for` é simulado os nascimentos e mortes de 4 semanas. A variável `pop` guarda o valor da população em cada semana.

```
> pop = 1000
> for(i in 1:4){
+     pop = pop + 0.2*pop - 0.05*pop
+ }
> pop
[1] 1749.006
```

Ao final de 4 semanas a população será de 1.794 indivíduos.

E depois de 8 semanas, qual o tamanho da população? Qual mudança deve ser feita no código anterior para encontrar essa resposta?

```
> pop = 1000
> for(i in 1:8){
+     pop = pop + 0.2*pop - 0.05*pop
+ }
> pop
[1] 3059.023
```

Ao final de 8 semanas a população será de 3059 indivíduos.

Seria interessante fazer as mesmas contas considerando outros valores para a população inicial, ou outra taxa de nascimento ou até outra taxa de mortalidade. Para facilitar essas contas e a mudanças desses valores é recomendado construir uma função. Essa função recebe como entrada: o tamanho inicial da população; a taxa de nascimento; a taxa de morte; e o tempo de espera. Essa mesma função retorna o tamanho da população ao final do período estipulado.

```
> populacao = function(pop,nas,mor,n){
+     for(i in 1:n){
+         pop = pop + nas*pop - mor*pop
+     }
+     return(pop)
+ }
```

Uma vez criada a função é possível simular o crescimento de diferentes populações e descobrir o tamanho dela após um dado período de tempo. Por exemplo,

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

para uma população inicial de 1.000 indivíduos, com taxa de nascimento de 20% e taxa de mortalidade de 5%, depois de 20 dias a população será de 16.366 indivíduos, como m ostra o código a seguir.

```
> populacao(1000,0.2,0.05,20)
[1] 16366.54
```

Se a taxa de nascimento for alterada para 5% e a de mortalidade para 20%, após 20 dias a população será de 38 indivíduos.

```
> populacao(1000,0.05,0.2,20)
[1] 38.75953
```

Ainda, no estudo sobre o crescimento populacional, considere agora um cenário diferente, onde exite limitações de alimento. Esta limitação será modelada da seguinte forma: para uma população menor que 2.000, a taxa de mortalidade é a mesma de 5%, porém se a população ficar acima de 2.000 a a taxa de mortalidade aumenta e passa a ser 5% + (tamanho da população - 2000)/10000. Isso representa a situação de escassez de alimentos.

Para esse novo cenário, qual o tamanho da população depois de 20 dias?

```
> for(i in 1:20){
+   if(pop < 2000){
+     pop = pop + 0.2*pop - 0.05*pop
+   } else {
+     pop = pop + 0.2*pop - (0.05 + (pop-2000)/10000)*pop
+   }
+ }
> pop
[1] 3499.902
```

Sob as condições apresentadas, onde a taxa de mortalidade aumenta quando a população ultrapassa o valor de 2000 indivíduos, após 20 dias o número de indivíduos é praticamente 3500 indivíduos.

Uma função pode ser útil para analisar esse novo cenário para diferentes valores iniciais.

```
> populacao2 = function(pop,nas,mor,n){
+   for(i in 1:n){
```

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

```
+     if(pop < 2000){
+       pop = pop + nas*pop - mor*pop
+     } else {
+       pop = pop + 0.2*pop - (0.05 + (pop-2000)/10000)*pop
+     }
+   }
+   return(pop)
+ }
```

Agora é possível, por exemplo, comparar as populações dos dois cenários após um período determinado.

```
> populacao(1000,0.2,0.05,20)
[1] 16366.54
> populacao2(1000,0.2,0.05,20)
[1] 3494.66
```

As funções `populacao` e `populacao2` retornam o tamanho da população após um período de tempo. Talvez uma análise mais detalhada seja interessante, como por exemplo, observar o tamanho da população em cada semana, e não apenas no final do período.

Para isso basta uma pequena mudança nas funções já criadas. A cada semana da simulação o tamanho da população será guardada em um vetor e este vetor que será o objeto retornado pela função.

Primeiro para o cenário em que não há restrição de alimentos.

```
> populacao_vec = function(pop,nas,mor,n){
+   vec_pop = pop
+   for(i in 1:n){
+     pop = pop + nas*pop - mor*pop
+     vec_pop = c(vec_pop,pop)
+   }
+   return(vec_pop)
+ }
> populacao_vec(1000,0.2,0.05,100)
[1] 1.000000e+03 1.150000e+03 1.322500e+03 1.520875e+03
[5] 1.749006e+03 2.011357e+03 2.313061e+03 2.660020e+03
[9] 3.059023e+03 3.517876e+03 4.045558e+03 4.652391e+03
[13] 5.350250e+03 6.152788e+03 7.075706e+03 8.137062e+03
[17] 9.357621e+03 1.076126e+04 1.237545e+04 1.423177e+04
```

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

```
[21] 1.636654e+04 1.882152e+04 2.164475e+04 2.489146e+04
[25] 2.862518e+04 3.291895e+04 3.785680e+04 4.353531e+04
[29] 5.006561e+04 5.757545e+04 6.621177e+04 7.614354e+04
[33] 8.756507e+04 1.006998e+05 1.158048e+05 1.331755e+05
[37] 1.531519e+05 1.761246e+05 2.025433e+05 2.329248e+05
[41] 2.678635e+05 3.080431e+05 3.542495e+05 4.073870e+05
[45] 4.684950e+05 5.387693e+05 6.195847e+05 7.125224e+05
[49] 8.194007e+05 9.423108e+05 1.083657e+06 1.246206e+06
[53] 1.433137e+06 1.648108e+06 1.895324e+06 2.179622e+06
[57] 2.506566e+06 2.882550e+06 3.314933e+06 3.812173e+06
[61] 4.383999e+06 5.041599e+06 5.797838e+06 6.667514e+06
[65] 7.667641e+06 8.817787e+06 1.014046e+07 1.166152e+07
[69] 1.341075e+07 1.542237e+07 1.773572e+07 2.039608e+07
[73] 2.345549e+07 2.697381e+07 3.101989e+07 3.567287e+07
[77] 4.102380e+07 4.717737e+07 5.425397e+07 6.239207e+07
[81] 7.175088e+07 8.251351e+07 9.489054e+07 1.091241e+08
[85] 1.254927e+08 1.443166e+08 1.659641e+08 1.908588e+08
[89] 2.194876e+08 2.524107e+08 2.902723e+08 3.338132e+08
[93] 3.838852e+08 4.414679e+08 5.076881e+08 5.838413e+08
[97] 6.714175e+08 7.721302e+08 8.879497e+08 1.021142e+09
[101] 1.174313e+09
```

Agora considerando a restrição de alimentos.

```
> populacao2_vec = function(pop,nas,mor,n){
+   vec_pop = pop
+   for(i in 1:n){
+     if(pop < 2000){
+       pop = pop + nas*pop - mor*pop
+     } else {
+       pop = pop + 0.2*pop - (0.05 + (pop-2000)/10000)*pop
+     }
+     vec_pop = c(vec_pop,pop)
+   }
+   return(vec_pop)
+ }
> populacao2_vec(1000,0.2,0.05,100)
[1] 1000.000 1150.000 1322.500 1520.875 1749.006 2011.357
[7] 2310.776 2585.579 2822.010 3013.340 3159.987 3267.431
[13] 3343.421 3395.772 3431.165 3454.784 3470.405 3480.676
[19] 3487.402 3491.795 3494.660 3496.526 3497.741 3498.531
[25] 3499.045 3499.379 3499.596 3499.738 3499.829 3499.889
[31] 3499.928 3499.953 3499.970 3499.980 3499.987 3499.992
[37] 3499.995 3499.996 3499.998 3499.999 3499.999 3499.999
```

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

```
[43] 3500.000 3500.000 3500.000 3500.000 3500.000 3500.000
[49] 3500.000 3500.000 3500.000 3500.000 3500.000 3500.000
[55] 3500.000 3500.000 3500.000 3500.000 3500.000 3500.000
[61] 3500.000 3500.000 3500.000 3500.000 3500.000 3500.000
[67] 3500.000 3500.000 3500.000 3500.000 3500.000 3500.000
[73] 3500.000 3500.000 3500.000 3500.000 3500.000 3500.000
[79] 3500.000 3500.000 3500.000 3500.000 3500.000 3500.000
[85] 3500.000 3500.000 3500.000 3500.000 3500.000 3500.000
[91] 3500.000 3500.000 3500.000 3500.000 3500.000 3500.000
[97] 3500.000 3500.000 3500.000 3500.000 3500.000
```

Para melhor visualização e comparação dos dois cenários é possível gerar gráficos que apresentem a curva de crescimento ao longo das semanas. O gráfico gerado pelo código a seguir encontra-se na Figura 2.1. Percebe-se um crescimento exponencial para o caso sem restrição de alimentos, e o tamanho da população ultrapassa $10^9 = 1.000.000.000$ após 100 semanas.

```
> v1 = populacao_vec(1000,0.2,0.05,100)
> plot(v1,axes = F,ylab="População",xlab="Semanas",
+     main="Crescimento populacional sem restrição de alimentos")
> box();grid();axis(2,c(1000,500000000,1000000000))
```

O código abaixo gera o gráfico para o cenário com escassez de alimentos, que é apresentado na Figura 2.2. Nesta figura o comportamento da população apresenta uma estabilidade e não ultrapassa os 3.500 habitantes após 100 semanas.

```
> v2 = populacao2_vec(1000,0.2,0.05,100)
> plot(v2,axes = F, ylab="População", xlab="Semanas",
+     main="Crescimento populacional com restrição de alimentos")
> box(); grid()
> axis(2,c(1000,2000,3000))
```

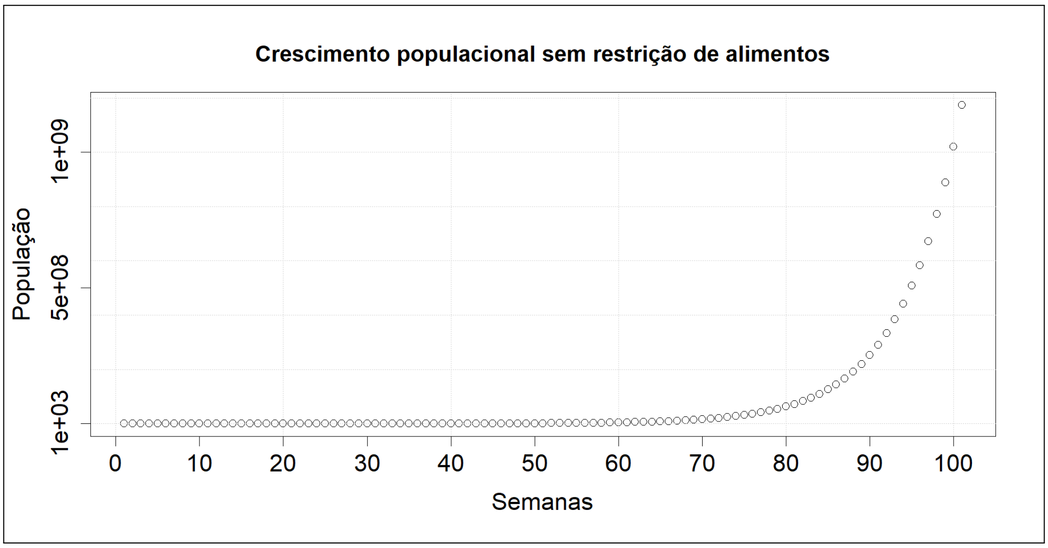
Problema da Fatoração

O último problema será como fatorar um número inteiro positivo em fatores primos.

Para elaborar um código que faça isso primeiro pense em como se faz essa conta no papel. Por exemplo, como fatorar o número 1234567890 em fatores

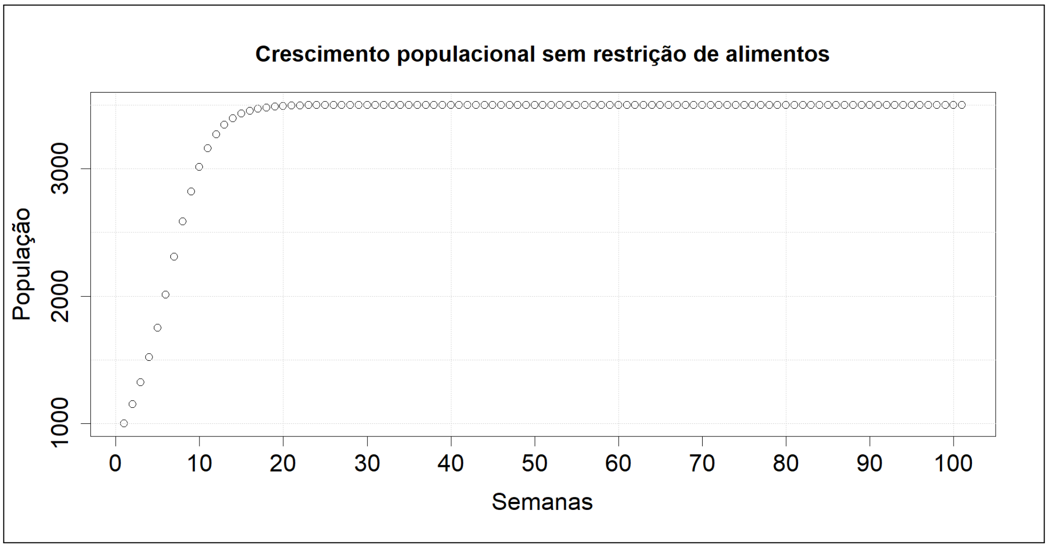
A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

Figura 2.1: Curva de crescimento para o cenário sem restrição de alimentos.



Fonte: A autora, 2024.

Figura 2.2: Curva de crescimento para o cenário com restrição de alimentos.



Fonte: A autora, 2024.

primos?

1. Comece com $q = 1234567890$.

2. O primeiro divisor que vamos testar é $d = 2$.

3. Se q não for divisível por d, incrementamos o valor de d, $d = d + 1$, e volte ao passo 2.

4. Se q for divisível por d, guarde d como um dos divisores de q e atualize o valor de q, $q = q/d$.

5. Se $q \neq 1$, volte ao passo 2.

6. Se $q = 1$, fim.

Tente seguir os passos acima para encontrar a fatoração de 1234567890 em números primos. Ou pelos menos os primeiros fatores primos, porque o processo será bem trabalhoso.

Tudo que é trabalhoso e repetitivo pode ser facilitado com o uso do computador. Mas pra isso é preciso programar.

```
> q = 1234567890
> fatores = NULL
> d = 2
> while(q > 1){
+   if(q %% d==0){
+     q = q/d
+     fatores = c(fatores,d)
+   } else {
+     d = d + 1
+   }
+ }
> fatores
[1]    2    3    3    5 3607 3803
```

Será que deu certo? Tem um jeito simples de verificar se uma fatoração está correta, basta multiplicar os fatores e verificar se o resultado é o número 1234567890.

```
> prod(fatores)
[1] 1234567890
```

Percebe que para esse processo iterativo é mais adequado usar o controle de fluxo `while` em vez do `for`? Veja também que ganha-se de brinde a informação de que os números 3607 e 3803 são primos.

Para encontrar a fatoração de outro número, o que mudar no código anterior? Ou melhor, como criar uma função que recebe como entrara um número inteiro positivo qualquer e retorna um vetor com os fatores primos desse número?

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

```
> fatoracao  = function(numero){
+   q = numero
+   fatores = NULL
+   d = 2
+   while(q > 1){
+     if(q %% d==0){
+       q = q/d
+       fatores = c(fatores,d)
+     } else {
+       d = d + 1
+     }
+   }
+   return(fatores)
+ }
> fatoracao(9876543210)
[1]      2      3      3      5     17     17 379721
```

O que acontece se a função fatoração for chamada para fatorar um número primo? Ela retorna o próprio número. Veja.

```
> fatoracao(2)
[1] 2
> fatoracao(5)
[1] 5
> fatoracao(7)
[1] 7
> fatoracao(191)
[1] 191
```

Isso porque um número primo não pode ser fatorado em número menores, sendo essa a definição de números primos. Dessa forma, o processo de encontrar fatores de um número pode ser usado também para decidir se um número qualquer é primo ou não. Para isso pode-se criar uma outra função, muito parecida com a função `fatoracao`, só que essa nova função será interrompida se for enocntrado um fator primo qualquer, diferente do próprio número, pois nesse caso já pode-se concluir que o número não é primo.

```
> eh_primo  = function(numero){
+   if(numero == 1 || numero == 2){
+     return(TRUE)
```

```
+   }
+   d = 2
+   while(d<numero){
+     if(numero%%d==0){
+       return(FALSE)
+     } else {
+       d = d + 1
+     }
+   }
+   return(TRUE)
+ }
> eh_primo(12345678989)
[1] FALSE
> eh_primo(268969)
[1] TRUE
```

Antes de terminar veja mais uma variação deste mesmo problema. Como criar uma função que retorna todos os primos menores que um número qualquer passado como argumento? Esse processo que parece complicado pode ficar bem simples com o auxílio da função `eh_primo` já criada.

```
> primos_menores_que = function(N){
+   primos = NULL
+   for(i in 1:N){
+     if(eh_primo(i)){
+       primos = c(primos,i)
+     }
+   }
+   return(primos)
+ }
```

A função acima simplesmente percorre todos os números de 1 até N, onde N é o argumento de entrada, "perguntando" para cada um deles: você é primo? Esse não é o processo mais rápido para executar essa tarefa, mas sem dúvida é um processo que funciona.

```
> primos_menores_que(1000)
[1]    1   2   3   5   7  11  13  17  19  23  29  31  37  41  43
[16]  47  53  59  61  67  71  73  79  83  89  97 101 103 107 109
[31] 113 127 131 137 139 149 151 157 163 167 173 179 181 191 193
[46] 197 199 211 223 227 229 233 239 241 251 257 263 269 271 277
```

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

```
 [61] 281 283 293 307 311 313 317 331 337 347 349 353 359 367 373
 [76] 379 383 389 397 401 409 419 421 431 433 439 443 449 457 461
 [91] 463 467 479 487 491 499 503 509 521 523 541 547 557 563 569
[106] 571 577 587 593 599 601 607 613 617 619 631 641 643 647 653
[121] 659 661 673 677 683 691 701 709 719 727 733 739 743 751 757
[136] 761 769 773 787 797 809 811 821 823 827 829 839 853 857 859
[151] 863 877 881 883 887 907 911 919 929 937 941 947 953 967 971
[166] 977 983 991 997
```

2.8 CONSIDERAÇÕES FINAIS

Este capítulo apresentou conceitos básicos de programação na Linguagem R. Tais conceitos foram usados para simular e resolver problemas reais. Os exemplos apresentados mostraram a importância e as possibilidades de solução quando se sabe programar. Saber programar é uma ferramente fundamental na formação de um profissional em diversas áreas. Espera-se que o leitor tenha tido esta persepsão e que depois desta leitura busque mais conhecimento e mais prática na área. Divirtam-se!

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

Capítulo 3

Programação Aprimorada com R Utilizando Assistência Cognitiva Artificial

Autora: Luciane Ferreira Alcoforado[1]
Instituição: Academia da Força Aérea (AFA)
e-mail: lucianea@id.uff.br

3.1 INTRODUÇÃO

Entrar no mundo da programação pode ser desafiador para muitos, especialmente quando se trata de dominar uma linguagem específica como R. No entanto, com os avanços da tecnologia, novas ferramentas estão disponíveis para facilitar esse processo. Neste capítulo, exploraremos como a assistência cognitiva artificial, em particular o ChatGPT, pode ser uma aliada poderosa para quem está aprendendo ou trabalhando com R. Vamos mergulhar em algumas conversas exemplares com o ChatGPT, destacando tanto as dificuldades quanto as vantagens encontradas ao programar em R com sua assistência.

A motivação para explorar a programação em R com a assistência cognitiva artificial surge da necessidade de superar desafios e acelerar o processo de aprendizado e desenvolvimento de projetos. A linguagem R é amplamente utilizada em análise de dados e estatística, mas pode apresentar obstáculos para iniciantes e até mesmo para programadores experientes. A busca por ferramentas que facilitem esse processo levou à investigação de como a assistência cognitiva artificial, aqui representada pelo ChatGPT, pode ser aproveitada nesse contexto.

3.2 OBJETIVOS

O principal objetivo deste capítulo é explorar conceitos de programação computacional utilizando a linguagem R em conjunto com a assistência cognitiva artificial do chatGPT.

Os objetivos específicos são:

- mostrar solicitações ao chatGPT sobre comandos da linguagem R, exibindo exemplos práticos;
- refletir sobre as vantagens e desvantagens que a assistência do chatGPT pode fornecer a um iniciante na linguagem de programação R.

3.3 DEFINIÇÕES

Apresentamos algumas definições baseadas em (**Barr1981**), (**zhang2021study**), (**Warwick2012**) no contexto deste capítulo, para que o leitor possa se familiarizar com os termos relacionados ao universo da programação:

Algoritmo: um algoritmo é uma receita passo a passo que define como resolver um problema, sendo a base fundamental para o desenvolvimento de software e para a resolução de problemas computacionais.

Lógica de Programação: A lógica de programação é o conjunto de regras e técnicas utilizadas para definir a sequência de passos lógicos necessários para resolver um problema computacional. Ela envolve a formulação de algoritmos, que são sequências de instruções precisas e bem definidas, capazes de solucionar um problema específico.

Linguagem Natural: A linguagem natural é a forma de comunicação que os seres humanos utilizam para se expressar, seja falada ou escrita. Ela se refere ao modo como as pessoas se comunicam de maneira natural, utilizando palavras, frases e expressões que são compreendidas intuitivamente pelos falantes de uma determinada língua ou grupo linguístico.

Linguagem de Programação: Uma linguagem de programação é uma linguagem formal utilizada para escrever instruções que podem ser executadas por um computador. Ela fornece um conjunto de regras sintáticas e semânticas que permite aos programadores escreverem códigos fonte que serão traduzidos para instruções compreensíveis pelo computador. As linguagens de programação podem variar em complexidade e propósito, desde linguagens de baixo nível, como Assembly, até linguagens de alto nível, como Python, R entre outras.

Linguagem R: A linguagem R é uma linguagem de programação interativa e integrativa, o que significa que os comandos podem ser executados linha por linha, facilitando a exploração e manipulação de dados utilizada principalmente para computação estatística e análise de dados, além de pode ser integrada com outras linguagens de programação, como C/C++, Python e Java, permitindo a extensão de suas funcionalidades e a integração com sistemas existentes. Ela foi desenvolvida inicialmente por Ross Ihaka e Robert Gentleman na Universidade de Auckland, Nova Zelândia, na década de 1990 e atualmente é gerenciada pela R Foundation e conta com colaboradores do mundo todo.

Inteligência: A inteligência é a capacidade de compreender, aprender, raciocinar, tomar decisões e resolver problemas de forma adaptativa em um ambiente em constante mudança. Ela envolve a aplicação de conhecimento e habilidades para alcançar objetivos e resolver desafios de maneira eficaz.

Inteligência Artificial (IA): A inteligência artificial refere-se à capacidade de máquinas ou sistemas computacionais executarem tarefas que normalmente exigiriam inteligência humana. Isso inclui atividades como reconhecimento de padrões, aprendizado, raciocínio, tomada de decisões e compreensão da linguagem natural. A IA busca criar sistemas que possam simular ou replicar o comportamento humano inteligente.

Assistência Cognitiva Artificial: A assistência cognitiva artificial é uma aplicação específica da inteligência artificial que visa oferecer suporte e orientação a indivíduos humanos em suas atividades cognitivas e tarefas diárias. Isso pode incluir a automação de tarefas, sugestões personalizadas, previsões baseadas em dados e interações naturais entre humanos e máquinas.

ChatGPT: O ChatGPT é um modelo de linguagem desenvolvido pela OpenAI que se baseia em inteligência artificial avançada para compreender e gerar texto em linguagem natural. Ele é treinado em uma vasta quantidade de dados textuais e pode responder a perguntas, realizar conversas, gerar textos criativos e muito mais. O ChatGPT é frequentemente utilizado como um assistente virtual para oferecer suporte em diversas áreas, desde esclarecimento de dúvidas até geração de conteúdo.

Copiloto: é um assistente de IA, diretamente integrado ao Windows 10 e Windows 11, permitindo que os usuários o acessem por meio de um botão na barra de tarefas. Além de texto, o Copilot aceita anexos de imagens, arquivos e capturas de tela para dar suporte às solicitações.

3.4 COMPREENSÃO BÁSICA DE PROGRAMAÇÃO

Diversos autores têm se dedicado ao ensino de programação na linguagem R, destacando-se (**Gillespie2016**), (**Grolemund2014**), (**Wickham2019**), (**Peng2022**). No Brasil, embora a representação feminina na área de Ciência de Dados e Programação ainda esteja em crescimento, há mulheres que contribuem para a literatura sobre R, destacando-se as autoras (**Alcoforado2021**), que fornece uma panorama geral e possibilidades de aplicação da linguagem R e (**Kubrusly2022**), que se concentra nos conceitos de introdução à programação voltada para estudantes de Estatística e áreas afins.

Antes de começar a programar em R, é útil ter uma compreensão básica de conceitos fundamentais de programação. Familiarizar-se com estes conceitos essenciais pode facilitar bastante o aprendizado e o uso eficaz da linguagem R.

Aqui estão alguns pontos-chave a serem considerados:

Variáveis: Em programação, as variáveis são usadas para armazenar dados temporariamente na memória do computador. Em R, você pode atribuir valores a variáveis usando o operador $<-$ ou o sinal de igual $=$.

Exemplo 1. *Criando **variáveis** com R.*

```
x <- 10
nome <- "João"
```

Vetor: estrutura de dados *unidimensional* que contém elementos do mesmo tipo.

Exemplo 2. *Criando **vetores** com R.*

```
vetor_numerico <- c(1, 2) #armazena 1 2
vetor_logico <- c(TRUE, FALSE) #armazena TRUE FALSE
vetor_caracter <- c("a", "b") #armazena "a"  "b"
```

Matriz: estrutura de dados *bidimensional* que contém elementos do mesmo tipo.

Exemplo 3. *Criando **matriz** com R.*

```
matriz <- matrix(1:6,
                 nrow = 2,
                 ncol = 3)
print(matriz)
```

```
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

Array: generalização de matriz, todos os elementos devem ser do mesmo tipo.

Exemplo 4. *Criando **array** com R.*

```r
array4d <- array(1:24,
        dim = c(2, 2, 2, 3))
# 2 linhas x 2 colunas
# 2 linhas 3 colunas
# 6 matrizes 2x2
print(array4d)
```

```
, , 1, 1

     [,1] [,2]
[1,]    1    3
[2,]    2    4

, , 2, 1

     [,1] [,2]
[1,]    5    7
[2,]    6    8

, , 1, 2

     [,1] [,2]
[1,]    9   11
[2,]   10   12

, , 2, 2

     [,1] [,2]
[1,]   13   15
[2,]   14   16

, , 1, 3

     [,1] [,2]
[1,]   17   19
[2,]   18   20

, , 2, 3

     [,1] [,2]
[1,]   21   23
[2,]   22   24
```

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

Lista: pode conter objetos de diferentes tipos e tamanhos.

Exemplo 5. *Criando **lista** com R.*

```
lista <- list(numeros = 1:3,
        logicos = c(TRUE, FALSE),
        caracteres = c("r", "R"))
print(lista)
```

```
$numeros
[1] 1 2 3

$logicos
[1]  TRUE FALSE

$caracteres
[1] "r" "R"
```

Data frame: Estrutura para dados tabulares, cada coluna um vetor de qualquer tipo e mesmo tamanho.

Exemplo 6. *Criando **data frame** com R.*

```
df <- data.frame(
numeros = 1:3,
caracteres = c("R1", "R2", "R3"),
logicos = c(TRUE, FALSE, TRUE) )
print(df)
```

```
numeros caracteres logicos
1       1          R1     TRUE
2       2          R2    FALSE
3       3          R3     TRUE
```

Estruturas de Controle: As estruturas de controle permitem que você altere o fluxo de execução do programa com base em certas condições. As estruturas de controle mais comuns são as declarações **if**, **else**, **ifelse** e os **loops for** e **while**.

Exemplo 7. *Criando um teste com uma estrutura condicional usando **if else**, considerando* $x = 10$.

```
if (x > 5) {
print("x é maior que 5")
} else {
print("x é menor ou igual a 5")
}
```

```
[1] "x é maior que 5"
```

Exemplo 8. *Imprimindo valores com uma estrutura de repetição usando **for**.*

```
for (i in 1:5) {
print(i)
}
```

```
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
```

Funções: As funções são blocos de código que realizam uma tarefa específica e podem ser reutilizadas em diferentes partes de um programa. Em R, assim como em muitas outras linguagens, você pode definir suas próprias funções usando o operador `function()`.

Exemplo 9. *Criando uma função simples para somar dois valores.*

```
# Definição de uma função em R
minha_funcao <- function(x, y) {
resultado <- x + y
return(resultado)
}

# Chamando a função e armazenando o resultado em uma variável
resultado <- minha_funcao(3, 5)
print(resultado)  # Saída: 8
```

Tempo de processamento: Tempo de execução de uma expressão ou código. Em R pode ser medido pela função *system.time()*. Serve para avaliar a

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

rapidez com que um código é executado, auxiliando o desenvolvedor na otimização da eficiência do código. O resultado incluirá três valores: 1-**usuário:** O tempo da CPU gasto pelo processo atual (ou seja, a sessão R atual); 2-**sistema:** O tempo da CPU gasto pelo *kernel* (sistema operacional) em nome do processo atual e 3-**decorrido:** O tempo real decorrido desde o início do processo.

Aplicação sobre o tempo de processamento pode ser visto em Exemplo 10 e em Exemplo 15. Observa-se a comparação destes tempos, evidenciando-se o código mais eficiente. Adicionalmente na Subseção 3.8.3, apresenta-se uma avaliação de códigos fornecidos pelo ChatGPT utilizando-se do tempo de execução como um dos fatores de seleção do melhor código.

Exemplo 10. *Calculando o tempo de execução de uma função recursiva que calcula o fatorial de um número.*

```
# Função para calcular o fatorial
fatorial <- function(n) {
resultado <- 1
for (i in 1:n) {
    resultado <- resultado * i
}
return(resultado)
}

# Medindo o tempo de execução
resultado <- system.time(fatorial(900000))
print(resultado)
```

```
usuário   sistema decorrido
    0.04      0.00      0.05
```

3.5 PROGRAMAÇÃO FUNCIONAL EM R

Na programação funcional, as funções são tratadas como entidades imutáveis e independentes do estado do programa, o que promove a clareza, a modularidade e a reutilização do código.

Funções de Ordem Superior: Em R, as funções de ordem superior são funções que podem receber outras funções como argumentos e/ou retornar funções como resultado. Veja o exemplo de uma função que recebe outra função:

Exemplo 11. *Criando uma função que retorna um vetor numérico.*

```
# Função que aplica outra função a cada elemento de um vetor
aplicar_funcao <- function(vetor, funcao) {
resultado <- sapply(vetor, funcao)
return(resultado) }

# Definindo uma função de exemplo
quadrado <- function(x) {
return(x^2) }

# Aplicando a função de exemplo a um vetor usando a função de ordem
   superior
resultado <- aplicar_funcao(c(1, 2, 3, 4), quadrado)
print(resultado)  # Saída: 1 4 9 16
```

```
[1]  1  4  9 16
```

Exemplo 12. *Criando uma função que gera outra função.*

Neste exemplo, a função `criar_funcao_media_ponderada()` retorna uma função `media_ponderada`, onde os pesos são configuráveis. Isso nos permite criar diferentes funções de média ponderada com diferentes conjuntos de pesos e, em seguida, aplicá-las a diferentes conjuntos de valores. Isso oferece flexibilidade e reutilização de código ao lidar com diferentes cenários onde a média ponderada é necessária.

```
# Função de ordem superior que retorna uma função para calcular a média
   ponderada
criar_funcao_media_ponderada <- function(pesos) {
media_ponderada <- function(valores) {
    soma <- sum(valores * pesos)
    peso_total <- sum(pesos)
    return(soma / peso_total)
}
```

```
return(media_ponderada)
}

# Criar função para calcular a média ponderada com diferentes pesos
media_ponderada_pesos1 <- criar_funcao_media_ponderada(c(0.3, 0.5, 0.2))
media_ponderada_pesos2 <- criar_funcao_media_ponderada(c(0.4, 0.4, 0.2))

# Calcular a média ponderada para diferentes conjuntos de valores
valores1 <- c(10, 20, 30)
valores2 <- c(15, 25, 35)
resultado1 <- media_ponderada_pesos1(valores1)
resultado2 <- media_ponderada_pesos2(valores2)

print(resultado1)  # Saída: 19
print(resultado2)  # Saída: 23
```

A linguagem R dispõe de uma família de funções de ordem superior conhecidas como família *apply*, assim como o pacote **purrr** que pode ser um coadjuvante, fornecendo um conjunto de ferramentas para programação funcional em R.

No estilo funcional de programação, procuramos evitar o uso de estruturas de repetição como o **for**, procurando utilizar funções que demonstrem o que está sendo feito através de um mapeamento do processo utilizando um procedimento em paralelo para cada elemento do mapeamento. O Exemplo 13 mostra um código executado com **for** e o Exemplo 14 mostra o mesmo procedimento executado com **apply** e com o pacote **purrr**. No Exemplo 15 podemos observar a diferença nos tempos de processamento de cada função, notadamente o código utilizando **apply** é o que se apresenta como mais eficiente em termos de tempo.

Exemplo 13. *Usando o looping **for** para obter a soma de cada coluna de uma matriz.*

```
# Criando uma matriz de exemplo
matriz <- matrix(1:9, nrow = 3)
print(matriz)
```

```
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
```

```
[3,]    3    6    9
```

```
# Inicializando um vetor para armazenar as somas das colunas
soma_colunas <- numeric(ncol(matriz))

# Calculando a soma das colunas
for (col in 1:3) {
soma_colunas[col] <- sum(matriz[, col])
}

# Exibindo as somas das colunas
print(soma_colunas)
```

```
[1]  6 15 24
```

A seguir, veremos diversas funções da família *apply* seguida de exemplos, procurando mostrar o mesmo procedimento usando a família de funções apply e a função equivalente do pacote **purrr**.

apply(): Aplica uma função a linhas ou colunas de uma matriz ou array. Matrizes são bidimensionais, consistindo de linhas e colunas, enquanto arrays podem ter qualquer número de dimensões, o que significa que podem representar volumes de dados mais complexos, como imagens em 3D ou tensores em aprendizado de máquina.

Exemplo 14. *Usando a função* ***apply*** *para obter a soma de cada coluna de uma matriz.*

```
# Aplicando a função sum a cada coluna da matriz
resultado <- apply(matriz , 2, sum)

# Imprimindo o resultado
print(resultado) # Saída: [1]  6 15 24
```

Usando o pacote **purrr**, função `map_dbl`:

```
# Carregando o pacote purrr
library(purrr)

# Aplicando a função sum a cada coluna da matriz
```

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

```
resultado <- map_dbl(as.data.frame(matriz), sum)

# Imprimindo o resultado
print(resultado) # Saída: [1]  6 15 24
```

Exemplo 15. *Comparando a eficiência entre as funções usando* ***system.time****.*

```
# Criando uma matriz de exemplo
matriz <- matrix(1:900000, nrow = 3)

# Obtendo o tempo de processamento
tempo1<-system.time(
for (col in 1:300000) {
soma_colunas[col] <- sum(matriz[, col]) } )

tempo2<-system.time(
apply ( matriz , 2 , sum ) )

tempo3<-system.time(purrr::map_dbl(as.data.frame(matriz), sum) )
# Exibindo os tempos obtidos
print(tempo1)
```

```
usuário   sistema decorrido
    0.52      0.03      0.66
print(tempo2)
usuário   sistema decorrido
    0.47      0.02      0.51
print(tempo3)
usuário   sistema decorrido
    1.39      0.00      1.67
```

Exemplo 16. *Usando a função* ***apply*** *em um array, obtendo a média dos elementos da terceira dimensão do array:*

```
# Criando um array tridimensional de exemplo
array_exemplo <- array(data = 1:24, dim = c(2, 3, 4))  # 2 experimentos,
    3 medidas, 4 repetições
array_exemplo
```

```
, , 1
```

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

```
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6

, , 2

     [,1] [,2] [,3]
[1,]    7    9   11
[2,]    8   10   12

, , 3

     [,1] [,2] [,3]
[1,]   13   15   17
[2,]   14   16   18

, , 4

     [,1] [,2] [,3]
[1,]   19   21   23
[2,]   20   22   24
```

```
# Aplicando a função mean() ao longo da terceira dimensão do array (
   repetições)
media_experimentos <- apply(array_exemplo, MARGIN = 3, FUN = mean)

# Exibindo o resultado
print(media_experimentos)
```

```
[1]  3.5  9.5 15.5 21.5
```

O pacote **purrr** não lida bem com *arrays* multidimensionais diretamente, desse modo, transformamos o *array* em uma lista de matrizes e, em seguida, usamos a função `map_dbl` pacote **purrr** para operar em cada matriz dessa lista.

```
# Carregando o pacote purrr
library(purrr)

# Transformando o array em uma lista de matrizes
lista_matrizes <- lapply(seq_len(dim(array_exemplo)[3]),
                         function(i) array_exemplo[ , , i, drop = FALSE])
print(lista_matrizes)
```

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

```
[[1]]
, , 1

     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6


[[2]]
, , 1

     [,1] [,2] [,3]
[1,]    7    9   11
[2,]    8   10   12


[[3]]
, , 1

     [,1] [,2] [,3]
[1,]   13   15   17
[2,]   14   16   18


[[4]]
, , 1

     [,1] [,2] [,3]
[1,]   19   21   23
[2,]   20   22   24
```

```
# Calculando a média ao longo da terceira dimensão (repetições)
media_experimentos <- lista_matrizes %>%
map_dbl(~ mean(.x, na.rm = TRUE))

# Exibindo o resultado
print(media_experimentos)
```

```
[1]  3.5  9.5 15.5 21.5
```

lapply(): Aplica uma função a cada elemento de uma lista e retorna uma lista com os resultados.

Exemplo 17. *Usando a função* ***lapply*** *para obter a soma dos elementos de uma lista.*

```
# Criando uma lista de exemplo
lista <- list(a = 1:3, b = 4:6, c = 7:9)
print(lista)
```

```
$a
[1] 1 2 3

$b
[1] 4 5 6

$c
[1] 7 8 9
```

```
# Aplicando a função sum a cada elemento da lista
resultado <- lapply(lista, sum)

# Imprimindo o resultado
print(resultado)
```

```
$a
[1] 6

$b
[1] 15

$c
[1] 24
```

Usando o pacote **purrr**, função `map`:

```
# Carregando o pacote purrr
library(purrr)

# Aplicando a função sum a cada elemento da lista
resultado <- map(lista, sum)

# Imprimindo o resultado
print(resultado)  # Saída: $a [1] 6 $b [1] 15 $c [1] 24
```

sapply(): Aplica uma função a cada elemento de uma lista e simplifica o resultado, se possível.

Exemplo 18. *Usando a função **sapply** para obter a soma dos elementos de uma lista.*

```
# Criando uma lista de exemplo
lista <- list(a = 1:3, b = 4:6, c = 7:9)
#lista já exibida anteriormente

# Aplicando a função sum a cada elemento da lista e simplificando o
   resultado
resultado <- sapply(lista, sum)

# Imprimindo o resultado
print(resultado)
```

```
a  b  c
6 15 24
```

Utilizando o pacote **purrr**, função `map_dbl`:

```
# Carregando o pacote purrr
library(purrr)

# Aplicando a função sum a cada elemento da lista
resultado <- map_dbl(lista, sum)

# Imprimindo o resultado
print(resultado)
```

```
a  b  c
6 15 24
```

tapply(): Aplica uma função a subsets de um vetor ou dataframe, agrupados por um ou mais fatores.

Exemplo 19. *Usando a função **tapply** para obter a soma dos elementos de um dataframe de acordo com um agrupamento.*

```
# Criando um dataframe de exemplo
df <- data.frame(grupo = c("A", "B", "A", "B"), valor = c(1, 2, 3, 4))
print(df)
```

```
Description:df [4 x 2]
grupo  valor
<chr>  <dbl>
A    1
B    2
A    3
B    4
4 rows
```

```
# Aplicando a função sum aos valores agrupados por grupo
resultado <- tapply(df$valor, df$grupo, sum)

# Imprimindo o resultado
print(resultado)
```

```
A B
4 6
```

Utilizando o pacote **dplyr**, função `group_by` e `summarise`.

```
# Carregando os pacotes necessários
library(dplyr)

# Agrupando os dados pelo grupo e aplicando a função sum aos valores
resultado <- df %>%
group_by(grupo) %>%
summarise(soma = sum(valor))

# Exibindo o resultado
print(resultado)
```

```
A tibble:2 x 2
grupo  soma
<chr>  <dbl>
A    4
B    6
2 rows
```

mapply(): Aplica uma função a múltiplos argumentos em paralelo, retornando um resultado simplificado, como por exemplo um vetor.

Exemplo 20. *Usando a função* ***mapply*** *para obter a soma dos elementos processados em paralelo.*

```r
# Definindo uma função de exemplo que retorna a soma de dois números
soma <- function(x, y) {
return(x + y)
}

# Aplicando a função soma a dois vetores em paralelo
resultado <- mapply(soma, c(1, 2, 3), c(4, 5, 6))

# Imprimindo o resultado
print(resultado)  # Saída: 5 7 9
```

Utilizando o pacote **purrr**, função `map2` ou `map2_dbl`:

```r
# Carregando o pacote purrr
library(purrr)


# Aplicando a função soma a dois vetores em paralelo e convertendo o
   resultado em um vetor
resultado <- map2(c(1, 2, 3), c(4, 5, 6), soma) %>%
            unlist()
# Imprimindo o resultado
print(resultado)  # Saída: 5 7 9

#Equivalente, retornando diretamente o vetor

resultado1 <- map2_dbl(c(1, 2, 3), c(4, 5, 6), soma)

# Imprimindo o resultado
print(resultado1)  # Saída: 5 7 9
```

Outra função a ser considerada, semelhante à *mapply* é a *Map()* que aplica uma função a cada elemento de um vetor e retorna uma lista com os resultados. Neste caso, o resultado não é simplificado na forma de um vetor.

Exemplo 21. *Usando a função **Map** para obter a soma de elementos, exibindo o resultado na forma de lista.*

```r
# Aplicando a função soma a cada elemento de um vetor
resultado <- Map(soma, c(1, 2, 3), c(4,5,6) )

# Imprimindo o resultado
```

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

```
print(resultado)
```

```
[[1]]
[1] 5

[[2]]
[1] 7

[[3]]
[1] 9
```

*Utilizando o pacote **purrr**, função* `map2`*:*

```
# Carregando o pacote purrr
library(purrr)

# Aplicando a função soma a cada elemento de um vetor
resultado <- map2(c(1, 2, 3),c(4, 5, 6), soma)

# Imprimindo o resultado
print(resultado)
```

```
[[1]]
[1] 5

[[2]]
[1] 7

[[3]]
[1] 9
```

O pacote **purrr** apresenta grande habilidade em lidar com operações em objetos do tipo lista, retornando resultados no formato de listas de forma natural através da função *map* ou *map2*, podendo ser simplificada para resultados como vetor numérico, utilizando, por exemplo, `map_dbl` ou `map2_dbl`.

Outras funções do pacote *base* na categoria das funções de ordem superior são listadas a seguir. A estrutura destas funções considera fornecer a função de operação seguida do vetor a ser operado.

Reduce(): Aplica uma função binária cumulativamente a elementos de uma lista, reduzindo-a a um único valor.

Exemplo 22. *Aplicando a função* ***Reduce*** *para obter a soma de elementos de um vetor de forma cumulativa.*

```
# Definindo uma função de exemplo que soma dois números
soma <- function(x, y) {
return(x + y)
}

# Aplicando a função soma cumulativamente aos elementos de um vetor
resultado <- Reduce(soma, c(1, 2, 3, 4, 5))

# Imprimindo o resultado
print(resultado)  # Saída: 15
```

Filter(): Retorna um subconjunto dos elementos de um vetor, lista ou data-frame que satisfazem uma condição.

Exemplo 23. *Aplicando a função* ***Filter*** *para obter elementos de um vetor que atendam a uma condição.*

```
# Criando um vetor de exemplo
vetor <- c(1, 2, 3, 4, 5)

# Filtrando os elementos do vetor que são maiores que 2
resultado <- Filter(function(x) x > 2, vetor)

# Imprimindo o resultado
print(resultado)  # Saída: 3 4 5
```

Find(): Retorna o primeiro elemento de um vetor, lista ou dataframe que satisfaz uma condição.

Exemplo 24. *Aplicando a função* ***Find*** *para obter o primeiro elemento de um vetor que atenda a uma condição.*

```
# Criando um vetor de exemplo
vetor <- c(1, 2, 3, 4, 5)

# Encontrando o primeiro elemento do vetor que é maior do que 2
resultado <- Find(function(x) x > 2, vetor)
```

```
# Imprimindo o resultado
print(resultado)  # Saída: 3
```

Position(): Retorna a primeira posição da esquerda para direita dos elementos de um vetor que satisfazem uma condição.

Exemplo 25. *Aplicando a função **Position** para obter a primeira posição dos elementos de um vetor que atendam a uma condição.*

```
# Criando um vetor de exemplo
vetor <- c(3, 2, 1, 4, 5)

# Encontrando a primeira posição dos elementos do vetor que é maior do
   que 2
resultado <- Position(function(x) x > 2, vetor)

# Imprimindo o resultado
print(resultado)  # Saída: 1
```

3.6 FUNÇÕES RECURSIVAS DO R

Recursão: A recursão é uma técnica na qual uma função chama a si mesma para resolver um problema. Ela é comumente usada na programação funcional para simplificar a solução de problemas complexos.

Exemplo 26. *Criando uma função recursiva com **if else**.*

```
# Função recursiva para calcular o fatorial de um número
fatorial <- function(n) {
if (n == 0 || n == 1) {
    return(1)
} else {
    return(n * fatorial(n - 1))  } }

# Calculando o fatorial de 5 usando a função recursiva
resultado <- fatorial(5)
print(resultado)  # Saída: 120
```

Neste exemplo, a função fatorial calcula o fatorial de um número n chamando a si mesma recursivamente até que n seja igual a 1 ou 0.

Na sequência, apresenta-se algumas funções nativas do R, que são implementadas de forma recursiva, juntamente com exemplos de uso.

factorial(): Calcula o fatorial de um número.

Exemplo 27. *Aplicando a função recursiva **factorial**.*

```
# Exemplo de uso
resultado <- factorial(5)
print(resultado)  # Saída: 120
```

identical(): Verifica se dois objetos são exatamente iguais.

Exemplo 28. *Aplicando a função recursiva **identical**.*

```
# Exemplo de uso
x <- c(1, 2, 3)
y <- c(1, 2, 3)
z <- c(3, 2, 1)
resultado1 <- identical(x, y)
print(resultado1)  # Saída: TRUE

resultado2 <- identical(x, z)
print(resultado2)  # Saída: FALSE
```

colSums(): Calcula a soma das colunas de uma matriz.

Exemplo 29. *Aplicando a função recursiva **colSums**.*

```
# Exemplo de uso
matriz <- matrix(1:9, nrow = 3)
matriz
resultado <- colSums(matriz)
print(resultado)  # Saída: 12 15 18
```

rowSums(): Calcula a soma das linhas de uma matriz.

Exemplo 30. *Aplicando a função recursiva **rowSums**.*

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

```
# Exemplo de uso
matriz <- matrix(1:9, nrow = 3)
resultado <- rowSums(matriz)
print(resultado)  # Saída: 6 15 24
```

sort(): Ordena um vetor em ordem crescente.

Exemplo 31. *Aplicando a função recursiva **sort**.*

```
# Exemplo de uso
vetor <- c(3, 1, 5, 2, 4)
resultado <- sort(vetor)
print(resultado)  # Saída: 1 2 3 4 5
```

rev(): Inverte a ordem dos elementos de um vetor.

Exemplo 32. *Aplicando a função recursiva **rev**.*

```
# Exemplo de uso
vetor <- c(1, 2, 3, 4, 5)
resultado <- rev(vetor)
print(resultado)  # Saída: 5 4 3 2 1
```

Entender as funções básicas do R e evitar o uso excessivo de pacotes é fundamental para garantir a portabilidade do código desenvolvido e evitar a vulnerabilidade das possíveis mudanças que os pacotes podem sofrer. Aprofundar-se nos conceitos básicos de programação e ter uma noção da programação funcional em R ajudará o desenvolvedor a escrever códigos mais eficientes e expressivos ao trabalhar com análise de dados, estatísticas, criação de funções próprias, pacotes ou mesmo aplicativos shiny.

3.7 POR QUE CONHECER OS COMANDOS NOS TORNA CRÍTICOS DA ASSISTÊNCIA COGNITIVA ARTIFICIAL?

Observe o diálogo entre o usuário e o ChatGPT. Nem todos os pormenores de sua resposta estão corretos. A função *reduce()* pertence ao pacote **purrr** e não ao pacote **dplyr**. Após alguma insistência no *chat*, foi criado um exemplo bem contextualizado para aplicação da função.

Usuário: *Para que serve a função reduce()? Mostre-me algo realmente valioso.*

ChatGPT: *A função reduce() é uma função de ordem superior disponível em diversos pacotes do R, incluindo o **base**, **purrr** e **dplyr**. Essa função é utilizada para combinar os elementos de uma lista, vetor ou data frame em um único valor, aplicando uma função iterativamente aos pares de elementos.*

O propósito principal da função reduce() é reduzir uma coleção de valores a um único valor, aplicando uma operação acumuladora iterativamente. Ela é especialmente útil em situações nas quais é necessário computar um valor agregado a partir de uma sequência de valores.

Vamos apresentar um exemplo significativo. Suponha que você tenha uma lista de números e queira encontrar o produto acumulado de todos esses números. A função reduce() pode ser útil nesse cenário. Aqui está um exemplo:

Exemplo 33. *Exemplo fornecido pelo ChatGPT para a função **reduce**.*

```r
# Carregando o pacote purrr
library(purrr)

# Lista de números
numeros <- c(2, 3, 5, 7, 11)

# Usando reduce para calcular o produto acumulado dos elementos da
   lista
produto_acumulado <- reduce(numeros, `*`)
print(produto_acumulado)  # Saída: 2310
```

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

Neste exemplo, reduce() é aplicado à lista de números, utilizando a operação de multiplicação (*). Isso efetivamente calcula o produto acumulado de todos os elementos da lista, resultando em 2310. Este é um exemplo útil de como a função reduce() pode ser utilizada para operações de acumulação mais complexas, indo além da simples soma.
Usuário: *Mas a função prod() não faria isso diretamente?* ***ChatGPT:*** *Você está absolutamente correto. A função prod() do R já está projetada para calcular o produto de todos os elementos de um vetor ou lista. Portanto, para calcular o produto acumulado de uma lista de números, você pode usar diretamente a função prod() sem a necessidade da função reduce(). Aqui está como você poderia fazer isso:*

Exemplo 34. *Exemplo fornecido pelo ChatGPT para a função* ***prod****.*

```
# Lista de números
numeros <- c(2, 3, 5, 7, 11)

# Usando a função prod para calcular o produto acumulado dos
   elementos da lista
produto_acumulado <- prod(numeros)
print(produto_acumulado)  # Saída: 2310
```

Neste caso, a função prod() é uma escolha mais direta e simples para calcular o produto acumulado de uma lista de números. O exemplo anterior com reduce() foi apenas uma demonstração do uso dessa função para operações de acumulação mais complexas, mas para calcular o produto acumulado, a função prod() é a opção mais adequada. Obrigado por apontar!
Usuário: *Estou frustrada, seu exemplo não me apontou algo realmente valioso para se usar reduce(). Teria outro exemplo que não pudesse ser resolvido de forma tão direta?*

ChatGPT: *Entendi, você está procurando por um exemplo mais genuinamente útil de reduce(), que não poderia ser facilmente replicado com outras funções embutidas mais diretas.*
Vamos pensar em uma situação na qual reduce() seja mais apropriado. Imagine que você tenha uma lista de preços de produtos e você queira calcular o preço total depois de aplicar um desconto a todos eles. O desconto pode ser uma operação mais complexa do que uma simples subtração, talvez envolvendo uma lógica específica de negociação. Nesse caso, reduce() poderia ser útil para aplicar o desconto iterativamente a cada preço e calcular o preço total. Aqui está como você poderia fazer isso:

Exemplo 35. *Um exemplo valioso fornecido pelo ChatGPT para a função* ***reduce()****.*

```
# Carregando o pacote purrr
library(purrr)
# Lista de preços dos produtos
preco <- c(100, 50, 75, 120)
# Função para aplicar um desconto a um preço
aplicar_desconto <- function(acc, preco) {
# Suponha que o desconto seja de 10% para preços acima de 50 e 5%
   caso contrário
if (preco > 50) {
desconto <- preco * 0.1
} else {
desconto <- preco * 0.05 }
# Aplica o desconto ao preço atual e acumula o resultado
return(acc + (preco - desconto)) }
# Usando reduce para calcular o preço total após aplicar o desconto
    a todos os produtos
preco_total <- reduce(precos, aplicar_desconto, .init = 0)
print(preco_total) # Saída: 313
```

Neste exemplo, `reduce()` *é usado para calcular o preço total após aplicar um desconto a todos os preços no objeto do tipo vetor denominado 'preco'. A função* `aplicar_desconto()` *define a lógica para aplicar o desconto a cada preço individualmente, e* ***reduce()*** *é usado para aplicar essa função iterativamente a todos os elementos da lista e calcular o preço total resultante. Isso destaca uma aplicação útil de* ***reduce()*** *em uma situação na qual a lógica de redução é mais complexa e não pode ser facilmente substituída por outras funções embutidas.*

Esta conversa procura ilustrar que se o desenvolvedor já possui algum conhecimento básico da linguagem R, isso o permite ser mais crítico ao receber uma resposta ou código. Nos exemplos desta seção ilustramos que conhecer algumas funções básicas como a função *prod*, pode ser útil para uma crítica ou discussão sobre a melhor aplicabilidade das respostas recebidas pela assistência. De qualquer forma, a criatividade em fornecer um exemplo contextualizado é algo que nos impressiona na atual conjuntura em que vivemos, uma tecnologia super recente e pouco conhecida até o momento, que vem ganhando adeptos com grande rapidez.

3.8 CONVERSAS COM CHATGPT

Vamos observar nesta seção algumas conversas entre a autora e a assistência do chatGPT como uma forma de exemplificar como um aprendiz iniciante na linguagem R pode se aproveitar desse recurso. Algumas destas conversas são adaptadas de problemas reais que surgiram no desenvolvimento de projetos coordenados pela autora.

Para um bom aproveitamento, no entanto, deve-se fazer perguntas bem direcionadas, o que pode ser difícil para alguém com nenhum conhecimento ou orientação de um professor humano especializado no assunto.

Cabe lembrar que um curso formal sobre programação passa por diversas fases que vão desde a ementa, os objetivos e o conteúdo a ser abordado, até se chegar a uma avaliação final. Esta tarefa, apesar de poder ser realizada por uma assistência artificial, não possui a mesma credibilidade de um professor especialista no assunto.

Diante disso, recomenda-se que o iniciante procure seguir uma bibliografia confiável, utilizando a assistência para agilizar e melhorar seu aprendizado, sempre acompanhado de orientação de um professor humano especialista em programação com R.

3.8.1 Conversa 1: Explorando Conceitos Básicos

Usuário: *Olá, ChatGPT! Estou começando a aprender R e estou um pouco confuso com a diferença entre listas e vetores. Você pode me ajudar a entender melhor?*

ChatGPT: *Claro! Em R, os vetores são estruturas de dados unidimensionais que podem conter elementos de um único tipo, enquanto as listas são estruturas que podem conter elementos de diferentes tipos e até outras listas. Posso te ajudar a criar exemplos para ilustrar isso?*

Usuário: *Sim, por favor!*

ChatGPT: *Ok, vamos lá.*

Exemplo 36. *Exemplo fornecido pelo ChatGPT para criar um vetor em R.*

```
meu_vetor <- c(1, 2, 3, 4, 5)
meu_vetor
```

```
[1] 1 2 3 4 5
```

Exemplo 37. *Exemplo fornecido pelo ChatGPT para criar uma lista em R.*

```
minha_lista <- list(nome = "João", idade = 25, notas = c(8, 7, 9))
minha_lista
```

```
$nome
[1] "João"
$idade
[1] 25
$notas
[1] 8 7 9
```

3.8.2 Conversa 2: Escrevendo Funções em R

Usuário: *Estou tentando escrever uma função em R que leia uma lista contendo diversos tipos de objetos, como vetores, textos e matrizes, bem como filtre as matrizes desta lista e retorne a média de suas colunas na forma de lista, mas estou tendo dificuldades. Pode me ajudar?*
ChatGPT: *Claro! Vou criar uma função em R que faz exatamente isso: irá ler uma lista contendo diversos tipos de objetos, filtrar as matrizes dessa lista e retornar a média de suas colunas na forma de uma nova lista.*

Exemplo 38. *Um exemplo de um código de função criado pela assistência.*

```
# Função para calcular a média das colunas das matrizes em uma
   lista
calcular_media_colunas <- function(lista) {
# Filtrar apenas as matrizes da lista
matrizes <- lista[sapply(lista, is.matrix)]

# Calcular a média das colunas de cada matriz
medias_colunas <- lapply(matrizes, function(matriz) {
    colMeans(matriz, na.rm = TRUE)  })

return(medias_colunas) }
```

```
# Exemplo de uso
minha_lista <- list(
"esta lista tem vários objetos",
matrix(1:8, nrow = 2),
matrix(1:16, nrow = 4),
matrix(1:9, nrow = 3, dimnames = list(c("l1", "l2", "l3"), c("Tr1",
    "Tr2", "Tr3"))),
c("texto1", "texto2"),
1:10 )
print(minha_lista)
```

```
[[1]]
[1] "esta lista tem vários objetos"
[[2]]
    [,1] [,2] [,3] [,4]
[1,]   1    3    5    7
[2,]   2    4    6    8
[[3]]
    [,1] [,2] [,3] [,4]
[1,]   1    5    9   13
[2,]   2    6   10   14
[3,]   3    7   11   15
[4,]   4    8   12   16
[[4]]
Tr1 Tr2 Tr3
l1   1   4   7
l2   2   5   8
l3   3   6   9
[[5]]
[1] "texto1" "texto2"
[[6]]
[1]  1  2  3  4  5  6  7  8  9 10
```

```
medias_colunas_resultado <- calcular_media_colunas(minha_lista)
print(medias_colunas_resultado)
```

```
    [[1]]
[1] 1.5 3.5 5.5 7.5

[[2]]
[1]  2.5  6.5 10.5 14.5

[[3]]
Tr1 Tr2 Tr3
2   5   8
```

No Exemplo 38, a função `calcular_media_colunas` filtra as matrizes da lista `minha_lista` e calcula a média das colunas de cada matriz. O resultado é uma lista com as médias das colunas de cada matriz contidas na lista original.

3.8.3 Conversa 3: Otimizando Scripts em R

Usuário: *Estou escrevendo um script em R para analisar um grande conjunto de dados e está demorando muito para rodar. Alguma sugestão de como posso otimizá-lo?*
ChatGPT: *Uma maneira de otimizar seu script é verificar se está usando funções vetorizadas sempre que possível, em vez de loops, pois isso pode melhorar significativamente o desempenho. Posso te ajudar a identificar trechos do seu código que podem ser vetorizados?*
Usuário: *Tenho uma matriz com 90.000 valores distintos. Estou usando o seguinte script para elevar os valores ao quadrado, mas gostaria de substituir o* ***for*** *por uma função de ordem superior e incluir uma condição de que o número só seja elevado ao quadrado se for par. Posso resolver com* ***apply****?*

```
matriz <- matrix(1:90000, nrow = 300)
resultado_for <- numeric(length(matriz))
for (i in seq_along(matriz)) {
    resultado_for[i] <- matriz[i] ^ 2 }
```

ChatGPT: *Você pode usar a função **apply** em combinação com **ifelse** para elevar ao quadrado apenas os valores pares da matriz.*

Exemplo 39. *Código de função gerado pela assistência para elevar valores pares ao quadrado, usando **apply** e **ifelse**.*

```
# Usa apply para aplicar uma função a cada elemento da matriz
resultado_apply <- apply(matriz, c(1, 2), function(x) ifelse(x %% 2
    == 0, x^2, x))
```

Neste código do Exemplo 39, *apply* é usado para aplicar uma função a cada elemento da matriz. A função passada para *apply* verifica se o número é par (`x %% 2 == 0`) e, em caso afirmativo, eleva-o ao quadrado (`x^2`), caso contrário, deixa o número inalterado (x).

Se preferir usar *lapply* em vez de *apply*, você pode fazer isso da seguinte maneira:

Exemplo 40. *Código de função gerado pela assistência para elevar valores pares ao quadrado, usando **lapply** e **ifelse**.*

```
# Converte a matriz para um vetor e aplica lapply
resultado_lapply <- lapply(matriz, function(x) ifelse(x %% 2 == 0,
   x^2, x))

# Converte o resultado de volta para uma matriz
resultado_lapply <- matrix(unlist(resultado_lapply), nrow = 300)
```

Pode ainda dispensar a função *apply* e aplicar somente *ifelse* da seguinte maneira:

Exemplo 41. *Código de função gerado pela assistência para elevar valores pares ao quadrado, usando apenas **ifelse**.*

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

```
matriz_quadrados_pares <- ifelse(matriz %% 2 == 0, matriz^2, matriz
   )
```

Ambas as abordagens dos Exemplos 39, 40 e 41 devem fornecer o mesmo resultado, mas a primeira que usa *apply* é mais direta para operações em matrizes, e a última abordagem é ainda mais direta, usando apenas o condicional *ifelse* aplicado diretamente na matriz.

Agora que temos três funções diferentes para realizar a mesma operação proposta inicialmente com o *loop for*, podemos analisar sua eficiência quanto ao tempo de processamento.

Exemplo 42. *Código para analisar o tempo de processamento das funções produzidas com ajuda da assistência, comparada ao código proposto pelo usuário com* ***loop for****.*

```
# Cria a matriz
matriz <- matrix(1:90000, nrow = 300)
# Usando for loop
resultado_for <- numeric(length(matriz))
t1 <- system.time({
for (i in seq_along(matriz)) {
    resultado_for[i] <- matriz[i] ^ 2  } })
# Usando apply com ifelse
t2 <- system.time({
resultado_apply_if <- apply(matriz, c(1, 2), function(x) ifelse(x
   %% 2 == 0, x^2, x)) })
# Usando lapply com ifelse
t30 <- system.time({
resultado_lapply_if <- lapply(matriz, function(x) ifelse(x %% 2 ==
   0, x^2, x)) })
t31 <- system.time({
resultado_lapply <- matrix(unlist(resultado_lapply_if), nrow = 300)
    })
t3 <- t30 + t31
```

```
# Usando ifelse diretamente na matriz
t4 <- system.time({
matriz_quadrados_pares <- ifelse(matriz %% 2 == 0, matriz^2, matriz
   ) })

# Resultados dos tempos de execução
t1
```

```
usuário   sistema decorrido
     0.26      0.01      0.28
t2
usuário   sistema decorrido
     3.23      0.03      3.56
t3
usuário   sistema decorrido
     2.44      0.02      2.62
t4
usuário   sistema decorrido
     0.03      0.02      0.04
```

Podemos observar no Exemplo 42 que t1 é o tempo de execução usando o *loop for*; t2 é o tempo de execução usando *apply* com *ifelse*; t3 é o tempo de execução usando *lapply* com *ifelse*, incluindo a conversão de volta para matriz e t4 é o tempo de execução usando *ifelse* diretamente na matriz.

O *loop for* (t1) é mais rápido que *apply* (t2) e *lapply* (t3) no sistema em que foi executado. Isso pode ser devido a otimizações específicas de compilação e execução no ambiente R. Embora *for* geralmente não seja a abordagem recomendada para grandes conjuntos de dados, devido à falta de generalização e clareza, neste caso específico ele oferece um desempenho melhor que *apply* e *lapply*, sendo esta uma opção viável e eficiente.

A melhor abordagem em termos de eficiência é usar *ifelse* (t4) diretamente na matriz. Esta solução é a mais rápida porque aproveita as operações vetorizadas, as quais são muito otimizadas em R para manipulação de grandes conjuntos de dados.

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

3.8.4 Conversa 4: Criando um gráfico com ggplot2

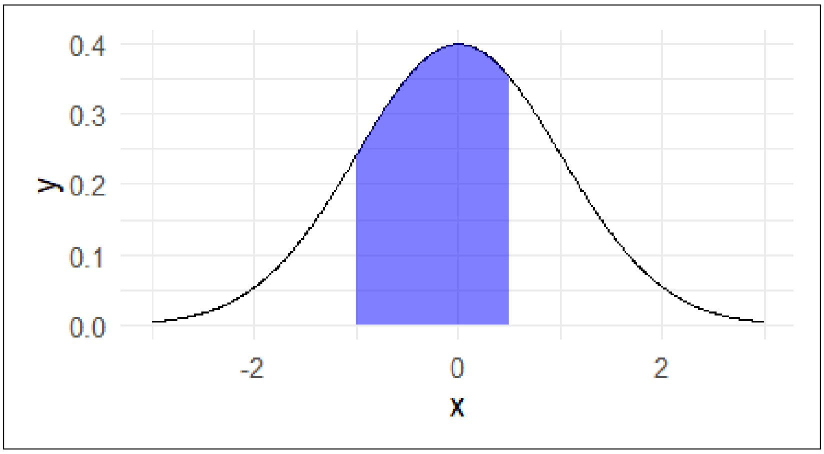
Usuário: *Crie no R o gráfico da distribuição normal padrão, pintando a área sob a curva no intervalo -1 a 0.5.* ***ChatGPT:*** *Para criar o gráfico da distribuição normal padrão no intervalo de -1 a 0.5 em R e pintar a área sob a curva, você pode usar a função dnorm para obter os valores da densidade de probabilidade e a função polygon para preencher a área sob a curva.*

Exemplo 43. *Exemplo fornecido pelo ChatGPT para criar o gráfico da distribuição normal.*

```
# Instale o pacote necessário se ainda não estiver instalado
if (!require("ggplot2")) {
install.packages("ggplot2")    }
library(ggplot2) # Carregue a biblioteca ggplot2
x <- seq(-3, 3, 0.01)# Crie um vetor de valores no intervalo
    desejado
# Calcule a densidade de probabilidade para a distribuição normal
    padrão
y <- dnorm(x)
# Crie um dataframe com os valores de x e y
df <- data.frame(x, y)
# Filtrar os valores de x no intervalo de -1 a 0.5
df_subset <- subset(df, x >= -1 & x <= 0.5)
# Crie o gráfico da distribuição normal padrão
plot <- ggplot(df, aes(x, y)) + geom_line() +
geom_ribbon(data = df_subset, aes(x = x, ymin = 0, ymax = y), fill
    = "blue", alpha = 0.5) + theme_minimal()
print(plot)    # Exiba o gráfico
```

Este código utiliza o pacote **ggplot2** para criar o gráfico da Figura 3.1 . Ele calcula a densidade de probabilidade para a distribuição normal padrão em um intervalo mais amplo e, em seguida, usa a função `geom_ribbon` para preencher a área sob a curva no intervalo de -1 a 0.5 com uma cor azul semi-transparente. A função `theme_minimal` é usada para um layout mais simples. Certifique-se de ter o pacote ggplot2 instalado antes de executar o código. Se não tiver instalado, você pode instalá-lo com install.packages("ggplot2").

Figura 3.1: Gráfico da distribuição normal.



Fonte: A autora, 2024.

Usuário: *Adicione no eixo x os pontos do -1 ao 0.5*

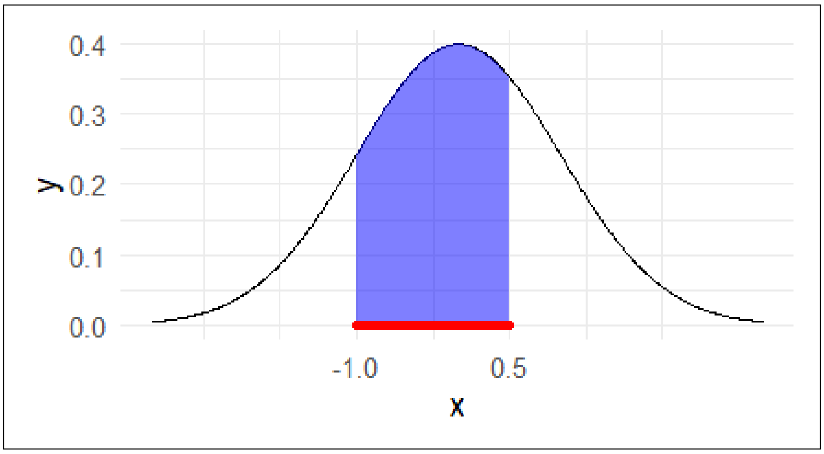
ChatGPT: *Claro, para adicionar os pontos no eixo x correspondentes aos valores de -1 e 0.5, você pode utilizar a função* `geom_point`.

Exemplo 44. *Exemplo fornecido pelo ChatGPT para adicionar pontos no eixo x.*

```
# A parte anterior do código não se altera...
plot <- ggplot(df, aes(x, y)) +    geom_line() +
geom_ribbon(data = df_subset, aes(x = x, ymin = 0, ymax = y), fill
   = "blue", alpha = 0.5) +
geom_point(data = df_subset, aes(x = x, y = 0), color = "red", size
    = 3) + scale_x_continuous(breaks = c(-1, 0.5)) +
theme_minimal()
print(plot)    # Exiba o gráfico
```

Este código adiciona a função `geom_point` para adicionar os pontos vermelhos no eixo x correspondentes aos valores de -1 e 0.5. A função `scale_x_continuous` é usada para definir os pontos de quebra no eixo x. O resultado pode ser visto no gráfico da Figura 3.2.

Figura 3.2: Gráfico da distribuição normal, adicionando pontos no eixo x.



Fonte: A autora, 2024.

Usuário: *inclua uma linha tracejada no sentido vertical passando pela origem*

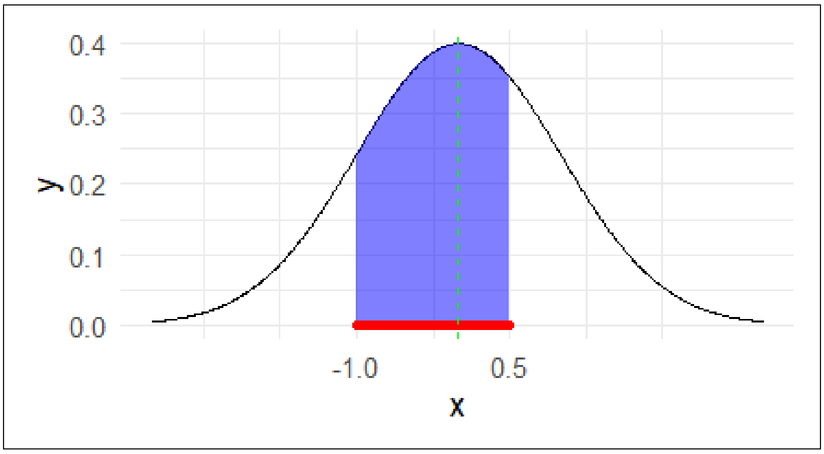
ChatGPT: *Para adicionar uma linha tracejada vertical passando pela origem, você pode usar a função* `geom_vline`.

Exemplo 45. *Exemplo fornecido pelo ChatGPT para adicionar uma linha tracejada vertical.*

```
plot <- ggplot(df, aes(x, y)) +
geom_line() +
geom_ribbon(data = df_subset, aes(x = x, ymin = 0, ymax = y), fill
   = "blue", alpha = 0.5) +
geom_point(data = df_subset, aes(x = x, y = 0), color = "red", size
    = 3) +
geom_vline(xintercept = 0, linetype = "dashed", color = "green") +
scale_x_continuous(breaks = c(-1, 0.5)) +
theme_minimal()
print(plot)
```

Este código adiciona a linha tracejada vertical (verde) passando pela origem usando a função `geom_vline`. A opção xintercept = 0 especifica que a linha tracejada deve passar pela coordenada x = 0. O parâmetro linetype = "dashed"define o estilo da linha como tracejada. O resultado pode ser visto na Figura 3.3

Figura 3.3: Gráfico da distribuição normal, adicionando linha tracejada vertical passando pela origem.



Fonte: A autora, 2024.

Usuário: *coloque rótulos nos eixo x e y.*

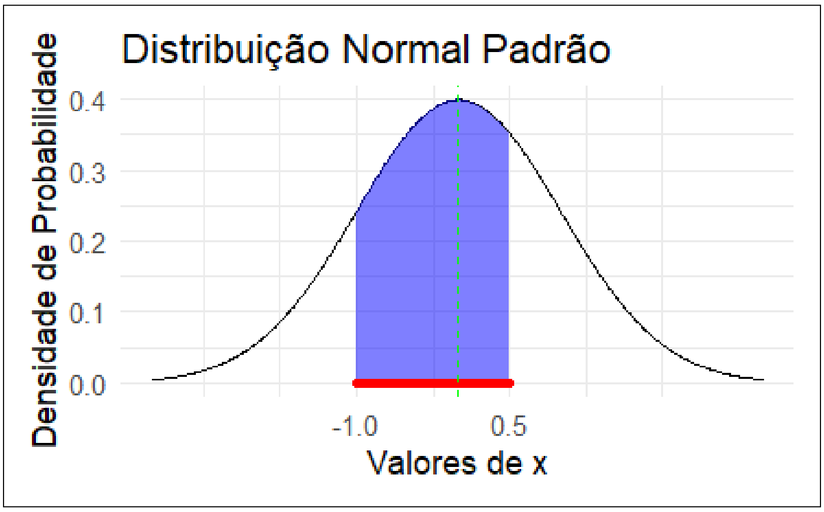
ChatGPT: *Para adicionar rótulos aos eixos x e y, você pode usar as funções labs e xlab e ylab em combinação com o código existente.*

Exemplo 46. *Exemplo fornecido pelo ChatGPT para colocar rótulos nos eixos do gráfico.*

```
# Crie o gráfico da distribuição normal padrão
plot <- ggplot(df, aes(x, y)) + geom_line() +
geom_ribbon(data = df_subset, aes(x = x, ymin = 0, ymax = y), fill
   = "blue", alpha = 0.5) +
geom_point(data = df_subset, aes(x = x, y = 0), color = "red", size
    = 3) +
geom_vline(xintercept = 0, linetype = "dashed", color = "green") +
    # Adiciona a linha tracejada
scale_x_continuous(breaks = c(-1, 0.5)) +
labs(x = "Valores de x", y = "Densidade de Probabilidade", title =
   "Distribuição Normal Padrão") + theme_minimal()
print(plot)    # Exiba o gráfico
```

Este código adiciona rótulos aos eixos x e y usando labs e também adiciona um título ao gráfico usando o argumento title. Os rótulos podem ser personalizados conforme necessário. Veja o resultado final no gráfico da Figura 3.4.

Figura 3.4: Gráfico da distribuição normal, adicionando rótulos ao eixos.



Fonte: A autora, 2024.

3.8.5 Conversa 5: Extraindo caracteres de um vetor e organizando-os em um dataframe

Usuário: *Suponha um vetor de caracteres com o seguinte padrão* `x_n1,n2,n3`*, sendo n1, n2 e n3 um número qualquer. Preciso extrair esses 3 números de cada posição do vetor para criar um data frame com 3 colunas i, j, k, cada uma correspondendo a n1, n2 e n3. Faça no R.*

ChatGPT: *No R você pode usar expressões regulares e funções do pacote tidyverse, como* `str_extract()` *e* `separate()`*, para extrair os números do vetor de caracteres e criar um data frame com três colunas correspondentes.*

Exemplo 47. *Exemplo fornecido pelo ChatGPT para extrair textos de um vetor e criar um data frame, com parte do texto distribuído em colunas.*

```
library(tidyverse)
# Vetor de caracteres
vetor <- c("x_1,2,3", "x_4,5,6", "x_7,8,9")
# Extrair os números usando expressões regulares
df <- data.frame(vetor) %>%
```

```
mutate(
    n1 = as.numeric(str_extract(vetor, "(?<=x_)\\d+")),
    n2 = as.numeric(str_extract(vetor, "(?<=,)\\d+(?=,)")),
    n3 = as.numeric(str_extract(vetor, "(?<=,)\\d+$"))
) %>%
select(-vetor)

# Renomear as colunas
names(df) <- c("i", "j", "k")

# Imprimir o data frame resultante
print(df)
```

```
Description:df [3 x 3]
i      j      k
<dbl> <dbl> <dbl>
1     2     3
4     5     6
7     8     9
3 rows
```

Neste código usamos `str_extract()` para extrair os números usando expressões regulares. `(?\<=x\_)\\d+` corresponde ao número após o `"x\_"`. `(?\<=,)\\d+(?=,)` corresponde aos números entre as vírgulas. `as.numeric()` é usado para converter as strings de números em valores numéricos e `mutate()` para adicionar novas colunas ao dataframe. `select()` é usado para remover a coluna vetor.

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

3.8.6 Conversa 6: Criando um aplicativo Shiny

Usuário: *Crie um aplicativo Shiny para exposição de mandalas. São 7 arquivos M1.png, ..., M7.png. O aplicativo deve disponibilizar um botão para o usuário escolher qual mandala quer visualizar. Ao exibir o desenho, deve aparecer junto uma janela explicativa contendo informações sobre aquela mandala. Cada mandala tem um texto diferente.*

ChatGPT: *Claro, vamos criar um aplicativo Shiny para a exposição das mandalas. O aplicativo permitirá que os usuários escolham qual mandala querem visualizar e exibirá uma janela explicativa com informações sobre a mandala selecionada.*

Estrutura dos Arquivos

1. *Imagens das Mandalas: Devem estar nomeadas como M1.png, M2.png, ..., M7.png e localizadas na pasta www dentro do diretório do aplicativo Shiny.*

2. *Textos Explicativos: Cada mandala terá um texto explicativo diferente que será exibido na janela.*

Exemplo 48. *Código do aplicativo shiny criado pelo ChatGPT.*

```
library(shiny)
# Textos explicativos para cada mandala
mandala_texts <- c(
"Texto explicativo para a Mandala 1.",
"Texto explicativo para a Mandala 2.",
"Texto explicativo para a Mandala 3.",
"Texto explicativo para a Mandala 4.",
"Texto explicativo para a Mandala 5.",
"Texto explicativo para a Mandala 6.",
"Texto explicativo para a Mandala 7." )
```

```r
ui <- fluidPage(  titlePanel("Exposição de Mandalas"),
sidebarLayout(  sidebarPanel(
    selectInput("mandala", "Escolha uma mandala:",
                choices = paste("Mandala", 1:7),
                selected = "Mandala 1") ),
    mainPanel( imageOutput("mandalaImage"),
    textOutput("mandalaText") ) ) )
server <- function(input, output, session) {
output$mandalaImage <- renderImage({
    list( src = paste0("www/M", substr(input$mandala, 8, 8), ".png"
       ),
    contentType = 'image/png',
    alt = paste("Mandala", substr(input$mandala, 8, 8)) ) },
    deleteFile = FALSE)
output$mandalaText <- renderText({
    mandala_index <- as.numeric(substr(input$mandala, 8, 8))
    mandala_texts[mandala_index]}) }
shinyApp(ui = ui, server = server)
```

As imagens das mandalas devem estar na pasta www dentro do diretório do aplicativo *Shiny*. A estrutura dos arquivos deve ser assim de acordo com a Figura 3.5.

Figura 3.5: Estrutura dos arquivos no diretório do aplicativo shiny.

```bash
/seu_diretorio_do_app_shiny
  |- app.R
  |- www/
      |- M1.png
      |- M2.png
      |- M3.png
      |- M4.png
      |- M5.png
      |- M6.png
      |- M7.png
```

Fonte: A autora, 2024.

Como Executar o Aplicativo

1. Salve o código acima em um arquivo chamado `app.R`.

2. Crie uma pasta www dentro do diretório onde o arquivo `app.R` está salvo.

3. Coloque as imagens M1.png, M2.png, ..., M7.png dentro da pasta www.

4. Execute o aplicativo Shiny usando o comando:
   `shiny::runApp('caminho_para_o_seu_diretorio_do_app_shiny')`.

Explicação do Código

1. UI (*User Interface*): O layout do aplicativo é definido usando fluidPage com um painel lateral (sidebarPanel) para a seleção da mandala e um painel principal (mainPanel) para exibir a imagem e o texto explicativo.

2. Server: No servidor, a função renderImage é usada para exibir a imagem da mandala selecionada e renderText para exibir o texto explicativo correspondente.

3. Agora você tem um aplicativo Shiny funcional que permite aos usuários escolher e visualizar diferentes mandalas com suas respectivas explicações.

Destaca-se, aqui nesta conversa, que o código fornecido pelo ChatGPT no Exemplo 48 não funcionou conforme o esperado, porém toda a explicação da estrutura do aplicativo e os procedimentos que o usuário precisaria fazer para finalizar o aplicativo estão corretos. Foi possível, a partir da resposta inicial, identificar os ajustes necessários para obter o código desejado e publicar o aplicativo, cujo resultado final pode ser visto na Figura 3.6.

Figura 3.6: O aplicativo final após ajustes no código fornecido pelo ChatGPT.

Fonte: https://lucianealcoforado.shinyapps.io/exposicao_mandala_2024/, 2024.

3.9 REFLEXÕES SOBRE AS VANTAGENS E AS DESVANTAGENS

As vantagens de utilizar a inteligência artificial, como o ChatGPT, ao programar em R são diversas. Em primeiro lugar, o ChatGPT oferece assistência instantânea, disponível a qualquer momento, para responder dúvidas, explicar conceitos complexos e oferecer exemplos práticos. Além disso, ele pode ajudar na geração de código, na sugestão de boas práticas de programação e na otimização de *scripts*. A capacidade de interação natural também facilita o processo de aprendizado, permitindo que os usuários experimentem e testem hipóteses sem medo de cometer erros.

As desvantagens residem na imprecisão das respostas, na confiabilidade e no custo de adquirir versões mais completas e com recursos mais avançados, bem como nos limites técnicos de uso e processamento da ferramenta. Do ponto de vista do desenvolvimento de códigos em R, para obter a solução correta, muitas vezes é necessário uma longa conversa que não pode ultrapassar 4096 caracteres em cada pergunta. Assim, se o seu código for muito longo, essa interação não será muito eficiente, pois será necessário dividir o problema em perguntas menores, além de exigir um tempo maior de iteração para corrigir os possíveis erros na resposta do ChatGPT.

Uma desvantagem específica ao usar o ChatGPT é o fenômeno conhecido como "alucinação", no qual o modelo pode fornecer informações incorretas ou inventadas de maneira convincente. Isso pode ser especialmente problemático quando se confia na geração automática de código ou na explicação de conceitos complexos, pois sua resposta pode levar a erros difíceis de detectar. É sempre crucial validar e testar qualquer código ou informação fornecida pelo ChatGPT para garantir sua precisão.

Assim, para uma boa experiência utilizando assistência como a do ChatGPT, sugerimos que ao solicitar uma ajuda na elaboração de um código em R, procure esclarecer o contexto do seu problema, o objetivo do seu código, o tipo de objeto ou dado com o qual está lidando e as restrições que devem ser atendidas, sempre testando o código fornecido e realizando ajustes ou solicitando melhorias. Procure documentar a totalidade do processo, mantendo todo o código e alterações realizadas devidamente registrados para compreensão de outros colaboradores que poderão surgir.

Por fim, cabe registrar que todas as conversas aqui expostas foram baseadas na versão gratuita GPT-3.5 da empresa OpenAI e que a versão Plus, com acesso ao GPT-4, atualmente apresenta um custo de `U$20,00` por mês.

Capítulo 4

Aplicações da Inteligência Artificial: seus Desafios e Oportunidades

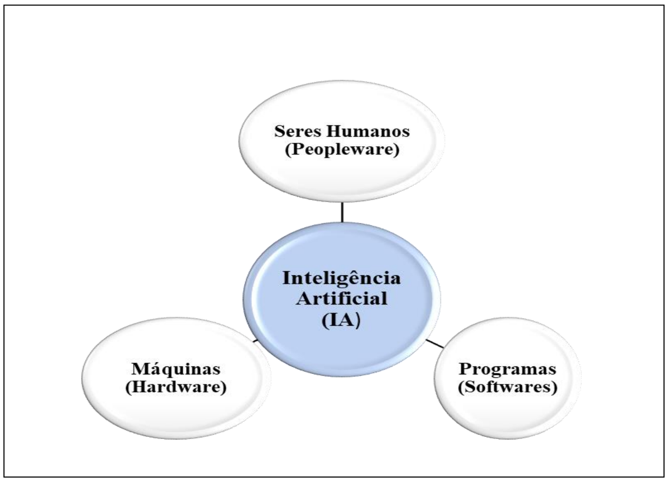
Autor: Marco Aurélio Chaves Ferro, D. Sc.[1]
Instituição: PPGEC/UFF, Niterói, RJ
e-mail: marcoferro@id.uff.br

4.1 INTRODUÇÃO

Existem inúmeras definições de Inteligência Artificial descritas pela comunidade científica, cada uma delas realçando algum aspecto específico, dentre vários possíveis. Uma delas pode ser sintetizada como um campo de estudo multidisciplinar da ciência que visa desenvolver sistemas capazes de realizar tarefas que, quando realizadas por humanos, exigem inteligência. Isso inclui habilidades como reconhecimento de fala, aprendizado, planejamento, resolução de problemas e até a capacidade de tomar decisões complexas. Para se obterem esses resultados, faz-se necessária a interação entre três pilares, que formam um tripé, quais sejam seres humanos (*peopleware*), máquinas (*hardware*) e seres humanos, segundo Ferro (**Ferro2020**) e mostrado na Figura 4.1. Desde seus primórdios, na metade do século XX, a IA tem sido um impulsionador da inovação tecnológica, influenciando áreas tão diversas quanto a medicina, finanças, indústria, engenharia e entretenimento, de acordo com Krishnamoorthy (**Krishnamoorthy1996**), Rosa

(**Rosa2011**) e Russell (**Russell2020**).

Figura 4.1: Tripé da Inteligência Artificial.



Fonte: (**Ferro2020**).

4.2 HISTÓRICO E EVOLUÇÃO

A história da Inteligência Artificial (IA) começa na segunda metade do século XX, embora suas raízes conceituais possam ser encontradas na antiguidade clássica, com a criação de autômatos e o desejo de construir máquinas que pudessem realizar tarefas sem a interferência humana. No entanto, foi em 1956, durante a conferência de *Dartmouth College*, nos Estados Unidos, que o termo 'Inteligência Artificial' foi oficialmente cunhado, marcando o início formal da área.

Nos primeiros anos, os pesquisadores estavam otimistas sobre a possibilidade de, rapidamente, desenvolver máquinas capazes de realizar tarefas cognitivas complexas, como jogar xadrez e entender a linguagem natural. No entanto, o progresso inicial foi mais lento do que o esperado. Os primeiros sistemas de IA, baseados em regras e lógica simbólica, falharam em lidar com a complexidade do mundo real, levando ao que passou a ser denominado primeiro "inverno da IA"nas décadas de 1970 e 1980, quando as pesquisas diminuíram.

A virada positiva começou no início nos anos 1980, com o desenvolvimento das redes neurais artificiais, que tentavam copiar o funcionamento do cérebro humano ao processar informações. Apesar de, inicialmente, terem um desempenho limitado, as redes neurais ganharam força com a chegada de computadores mais poderosos e grandes volumes de dados. Isso levou ao impulsionamento da IA nos anos 2000, principalmente por avanços em aprendizado de máquina (*machine learning*) e, mais tarde, aprendizado profundo (*deep learning*).

No início do século XXI, a IA começou a ser aplicada em uma grande variedade de casos, desde o reconhecimento de fala e imagem até a automação industrial e a medicina. Grandes empresas de tecnologia realizaram altos investimentos na área, acelerando ainda mais seu desenvolvimento. Hoje, a IA está presente em inúmeros aspectos da vida cotidiana, como assistentes virtuais, veículos autônomos, sistemas de recomendação, dentre outros

Contudo, o rápido avanço da IA também trouxe novos desafios, como a necessidade de lidar com questões éticas, a transparência dos algoritmos e os impactos econômico e social da automação. Enquanto os pesquisadores continuam a explorar a fronteira da inteligência artificial geral (AGI), que seria uma IA capaz de realizar qualquer tarefa cognitiva humana, a sociedade debate como melhor integrar e regular essa poderosa tecnologia.

O futuro da IA promete inovações ainda mais surpreendentes, mas também exige uma abordagem cuidadosa para garantir que seus benefícios sejam amplamente compartilhados e seus riscos, mitigados.

4.3 ABORDAGENS E TÉCNICAS

A Inteligência Artificial (IA) é uma área vasta e multidisciplinar, com diversas abordagens e técnicas que buscam simular a inteligência humana em máquinas, de acordo com Ferro (**Ferro2023**). Essas abordagens variam desde métodos baseados em regras até técnicas que envolvem o aprendizado a partir de grandes quantidades de dados. Neste capítulo são apresentadas algumas das principais abordagens e técnicas utilizadas na IA, incluindo aprendizado de máquina, aprendizado profundo, redes neurais artificiais, processamento de linguagem natural, visão computacional, sistemas especialistas e robótica.

4.3.1 Aprendizado de Máquina (*Machine Learning*)

O aprendizado de máquina é uma das subáreas mais utilizadas da IA. Trata-se de uma abordagem em que sistemas são treinados para identificar padrões e fazer previsões com base em dados, sem serem explicitamente programados para realizar uma tarefa específica. Existem várias técnicas dentro do aprendizado de máquina, mas as três mais comuns são: aprendizado supervisionado, não supervisionado e por reforço.

No Aprendizado Supervisionado, o sistema é treinado com um conjunto de dados rotulados, estruturados ou não, em que as entradas são associadas a saídas conhecidas, denominadas alvos. O objetivo do modelo é aprender a mapear as entradas para as saídas corretas e, posteriormente, aplicar esse mapeamento a novos dados. Um exemplo comum é a classificação de *e-mails* em *spam* ou não *spam*.

No Aprendizado Não Supervisionado, o sistema recebe dados não rotulados e deve encontrar padrões ou estruturas ocultas dentro desses dados. Técnicas como a análise de agrupamento (*clustering*) e a redução de dimensionalidade são exemplos de aprendizado não supervisionado. Essa abordagem é amplamente utilizada em tarefas como segmentação de mercado e detecção de anomalias em processos ou sistemas.

No Aprendizado por Reforço, um agente aprende a tomar decisões em um ambiente específico, recebendo recompensas ou punições com base em suas ações. O objetivo é maximizar as recompensas e minimizar as penalidades totais ao longo do tempo. O aprendizado por reforço tem sido utilizado com sucesso em áreas como jogos e controle de robôs.

4.3.2 Aprendizado Profundo (*Deep Learning*)

O aprendizado profundo é uma subárea do aprendizado de máquina que utiliza redes neurais com muitas camadas (redes neurais profundas) para modelar padrões complexos em grandes volumes de dados. Essas técnicas têm revolucionado várias áreas da IA, devido à sua capacidade de aprender representações hierárquicas dos dados.

Em aprendizado profundo, redes com várias camadas de neurônios (muitas vezes, dezenas ou até centenas de camadas) são treinadas para extrair padrões e características cada vez mais abstratas dos dados. Por exemplo, em uma rede neural profunda treinada para reconhecer objetos em imagens, as camadas iniciais podem aprender a detectar bordas e formas básicas, enquanto as camadas mais profundas podem reconhecer partes de objetos, cores e, eventualmente, o objeto completo.

O treinamento de redes profundas é um processo intenso, sob o ponto de vista computacional, que requer grandes volumes de dados e capacidade de processamento significativo. Algoritmos específicos, como o gradiente descendente estocástico (SGD) e suas variantes, são usados para ajustar os pesos das conexões entre os neurônios de maneira eficiente. Além disso, técnicas como regularização, *dropout* e normalização em lote (*batch normalization*) são usadas para melhorar o desempenho e evitar o sobreajuste (*overfitting*).

4.3.3 Redes Neurais Artificiais

As redes neurais artificiais são modelos computacionais inspirados no cérebro humano, compostas por unidades chamadas neurônios, que estão interconectadas em camadas. Cada camada recebe entradas, processa-as e passa a saída para a próxima camada. As redes neurais são particularmente eficazes para tarefas que envolvem padrões complexos, como reconhecimento de imagem e processamento de linguagem natural, de acordo com Spatti (**Silva2016**). O *perceptron* é o tipo

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

mais simples de rede neural, com uma única camada de neurônios. Redes *feed-forward*, por sua vez, consistem em várias camadas, nas quais as conexões fluem em uma única direção, da entrada para a saída. Essas redes são a base de muitas aplicações de IA, como reconhecimento de dígitos manuscritos.

Redes Neurais Convolucionais (CNNs) são uma variante das redes neurais projetadas especificamente para processar dados que têm uma estrutura de grade, como imagens. São compostas por camadas convolucionais que aplicam filtros sobre as entradas, extraindo características como bordas e texturas. As CNNs são amplamente utilizadas em visão computacional, sendo fundamentais em sistemas de reconhecimento facial e análise de imagens na área da medicina.

Redes Neurais Recorrentes (RNNs) são redes neurais que têm conexões cíclicas, o que permite a manutenção de informações em uma "memória"interna ao longo do tempo. Isso as torna ideais para tarefas sequenciais, como processamento de texto e análise de séries temporais. Uma variante popular das RNNs é a *Long Short-Term Memory* (LSTM), que é eficaz na modelagem de dependências de longo prazo, em sequências de dados.

4.3.4 Processamento de Linguagem Natural (PLN)

O processamento de linguagem natural é uma subárea da IA que se concentra na interação entre computadores e linguagem humana. O objetivo é capacitar as máquinas a compreender, interpretar e gerar linguagem natural de maneira útil e significativa.

A modelagem de linguagem envolve o desenvolvimento de modelos que podem prever a probabilidade de uma sequência de palavras específica. Modelos como o n-grama, transformadores, e o GPT (*Generative Pre-trained Transformer*) são usados para tarefas como tradução automática, resumo de textos e geração de texto.

Essas técnicas permitem que os sistemas compreendam a estrutura e o significado das frases por intermédio de análises sintática e semântica. A análise sintática envolve o estudo da gramática e a identificação da função das palavras em uma sentença, enquanto a análise semântica busca entender o significado subjacente.

Reconhecimento de Fala e Conversão de Texto em Fala são aplicações importantes do PLN, em que a linguagem falada é convertida em texto, ou vice-versa. Essas tecnologias são amplamente usadas em assistentes virtuais como Siri, Alexa e *Google Assistant*, dentre outros.

4.3.5 Visão Computacional

A visão computacional é a área da IA que capacita as máquinas a interpretar e entender o mundo visual. Isso inclui tarefas como reconhecimento de objetos, segmentação de imagem, e reconstrução 3D. As técnicas de visão computacional permitem que os sistemas detectem e identifiquem objetos em imagens e vídeos. Isso é feito utilizando-se redes neurais convolucionais (CNNs) e algoritmos como o YOLO (*You Only Look Once*) e o R-CNN (*Region-Based Convolutional Neural Network*). Um exemplo recente é o reconhecimento de identificação de uma pintura como de autoria do pintor renascentista Rafael, como visto na Figura 4.2.

A segmentação envolve a divisão de uma imagem em regiões significativas, como separar o fundo do primeiro plano. Isso é crucial para aplicações em medicina, como na detecção de tumores em exames de imagem. Técnicas de visão estéreo permitem que os sistemas percebam a profundidade e reconstruam o ambiente em três dimensões, algo essencial para a navegação autônoma de veículos e robôs. A IA pode ainda fazer pinturas com características de determinadas épocas e artistas, como a vencedora do concurso de arte mostrada na Figura 4.3.

4.3.6 Sistemas Especialistas

Sistemas especialistas são programas de computador que simulam o julgamento e o comportamento de um humano ou organização que tem conhecimento especializado em determinada área do conhecimento, de acordo com Krishnamo-

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

Figura 4.2: Pintura atribuída a Rafael, identificada por IA.

Fonte: *https : //pt.wikipedia.org/wiki/Madona$_{s}$istina.*

orthy (**Krishnamoorthy1996**). Utilizam uma base de dados composta por fatos e regras para tomar decisões ou resolver problemas complexos. Um sistema especialista utiliza um motor de inferência para aplicar regras lógicas à base de conhecimento, derivando conclusões ou recomendando ações. Um exemplo clássico é o sistema MYCIN, desenvolvido na década de 1970, que auxiliava médicos no diagnóstico de infecções bacterianas. A criação de sistemas especialistas envolve a captura do conhecimento de especialistas humanos e sua codificação em um formato que o computador possa processar. Essa abordagem foi pioneira em áreas como medicina, finanças e diagnóstico de falhas industriais.

Figura 4.3: Pintura feita por por IA.

Fonte: *https://www.nytimes.com/2022/09/02/technology/ai-artificial-intelligence-artists.html.*

4.3.7 Robótica

A robótica é um campo da IA que envolve o desenvolvimento de robôs, máquinas que podem executar tarefas no mundo físico, com reduzida ou mesmo sem qualquer intervenção humana. A robótica combina sensores, atuadores e algoritmos de controle para permitir que os robôs percebam o ambiente, tomem decisões e realizem ações.

Robôs autônomos podem ser capazes de se mover em ambientes desconhecidos e dinâmicos. Técnicas de SLAM (*Simultaneous Localization and Mapping*) são usadas para criar mapas do ambiente e determinar a posição do robô em tempo real. A robótica também se preocupa com como os robôs interagem com os humanos. Isso envolve desde a interpretação de comandos de voz até a colaboração segura em tarefas compartilhadas. Em vez de substituir trabalhadores humanos, a IA pode colaborar com eles para aumentar a eficiência e a criatividade. A integração efetiva de sistemas de IA nas operações diárias pode levar a um aumento na produtividade e na resolução de problemas complexos.

Robôs industriais são amplamente utilizados para tarefas como montagem,

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

soldagem e pintura em fábricas. Além disso, robôs de serviço estão sendo cada vez mais usados em áreas como saúde, entrega e entretenimento.

4.4 IMPACTOS E DESAFIOS

A Inteligência Artificial (IA) é uma das forças mais transformadoras da tecnologia moderna, com potencial para revolucionar praticamente todos os aspectos da sociedade. No entanto, junto com os imensos benefícios que a IA pode proporcionar, surgem também desafios significativos e questões éticas complexas. Portanto, a partir daqui, serão abordados, em detalhes, os impactos da IA em várias áreas, incluindo economia, saúde, educação, segurança e privacidade, ao mesmo tempo em que serão discutidos os desafios associados à sua implementação, como o impacto no mercado de trabalho, a desigualdade, a transparência, a ética, e a governança.

4.4.1 Impactos da IA na Economia

A IA está transformando a economia global ao melhorar a eficiência, criar novas oportunidades de negócios e modificar a estrutura do mercado de trabalho. A automação de tarefas repetitivas e rotineiras, por meio da IA, permite que as empresas operem com maior eficiência e menor custo. Isso é particularmente evidente em setores como a manufatura, em que robôs e sistemas automatizados podem realizar tarefas que antes eram realizadas por humanos, com maior precisão e em menos tempo.

Os sistemas de IA podem analisar vastos conjuntos de dados, reconhecer padrões e prever resultados, oferecendo assim *insights* que podem ser imprescindíveis para a estratégia empresarial. Executivos e gestores de vários setores podem aproveitar a IA para tomar decisões mais acertadas. Com sua utilização desde a previsão de tendências de mercado até a otimização das cadeias de abastecimento, o papel da IA no planeamento estratégico deverá se aprofundar nos próximos anos.

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

Além disso, a IA está impulsionando a inovação em diversas indústrias, levando à criação de novos produtos e serviços que antes eram inconcebíveis. No setor financeiro, por exemplo, a IA é usada para análise preditiva, gestão de riscos e detecção de fraudes. Em serviços ao cliente, os assistentes virtuais baseados em IA estão transformando a forma como as empresas interagem com seus clientes, proporcionando experiências mais personalizadas e eficientes.

No entanto, a automação e a IA também estão alterando a demanda por diferentes tipos de habilidades no mercado de trabalho, segundo Nascimento Jr (**Nascimento2000**). Funções que envolvem tarefas repetitivas e previsíveis estão sendo progressivamente automatizadas, o que pode levar à substituição de trabalhadores em determinadas áreas. Por outro lado, há uma crescente demanda por habilidades em ciência de dados, programação e outras áreas relacionadas à IA, criando novas oportunidades de emprego, mas também exigindo uma força de trabalho mais qualificada.

4.4.2 Impactos da IA na Saúde

Na área da saúde, a IA tem o potencial de transformar completamente a forma como os cuidados médicos são prestados. Um dos principais impactos da IA na saúde é a melhoria dos diagnósticos com o uso de algoritmos de aprendizado de máquina que possam analisar grandes volumes de dados médicos, incluindo imagens, exames laboratoriais e históricos de pacientes, para identificar padrões e prever doenças com uma precisão que, muitas vezes, supera a dos médicos humanos.

Além disso, a IA está sendo usada no desenvolvimento de tratamentos personalizados, que levam em conta as características individuais de cada paciente, como genética, estilo de vida e histórico médico. Isso permite a criação de terapias mais eficazes e com menos efeitos colaterais. A IA também está revolucionando a descoberta de novos medicamentos, acelerando o processo de pesquisa e desenvolvimento ao identificar rapidamente compostos promissores e prever seus efeitos.

Outro impacto significativo da IA na saúde advém da telemedicina e da automação de tarefas administrativas. Sistemas baseados em IA podem ajudar na triagem de pacientes, no agendamento de consultas e na gestão de prontuários médicos, liberando os profissionais de saúde para se concentrarem em tarefas mais complexas e de maior valor.

4.4.3 Impactos da IA na Educação

A IA está desempenhando um papel crescente na educação, oferecendo novas maneiras de personalizar o aprendizado e melhorar os resultados educacionais. Com a IA, é possível criar sistemas de ensino adaptativo que ajustam o conteúdo e o ritmo de aprendizado de acordo com as necessidades individuais de cada aluno. Isso é particularmente útil em ambientes de ensino a distância, onde a IA pode fornecer *feedback* instantâneo e suporte personalizado aos estudantes.

Ferramentas de IA também estão sendo usadas para analisar o desempenho dos alunos, identificar áreas nas quais eles apresentam dificuldades e sugerir intervenções apropriadas. Isso pode ajudar os educadores a tomar decisões mais bem informadas sobre como apoiar melhor seus alunos.

Além disso, a IA está sendo usada para automatizar tarefas administrativas, como a correção de provas e a gestão de currículos, permitindo que os professores se concentrem mais no ensino e na interação direta com os alunos. No entanto, a crescente dependência da tecnologia na educação também levanta questões sobre a equidade no acesso às ferramentas de IA e o papel dos educadores no mundo digital.

4.4.4 Impactos da IA na Segurança e Privacidade

A IA também está impactando a segurança e a privacidade de maneira profunda. Em termos de segurança, a IA está sendo usada para monitorar e prever comportamentos suspeitos, tanto *on-line* quanto no mundo físico. Sistemas de IA podem analisar grandes volumes de dados em tempo real para detectar atividades incomuns ou potencialmente perigosas, como fraudes financeiras, ataques cibernéticos, ou até ameaças à segurança pública.

No entanto, o uso crescente da IA para vigilância e monitoramento também levanta preocupações significativas em relação à privacidade. Tecnologias como o reconhecimento facial e a análise preditiva estão sendo usadas por governos e empresas para rastrear e analisar o comportamento das pessoas, muitas vezes sem o seu consentimento ou conhecimento. Isso cria um dilema ético sobre o equilíbrio entre segurança e privacidade e levanta questões sobre o potencial de abuso dessas tecnologias.

4.4.5 Impacto Social e Cultural da IA

A IA está mudando não apenas a economia e as indústrias, mas também a forma como interagimos uns com os outros e percebemos o mundo ao nosso redor. As plataformas de mídia social, por exemplo, utilizam algoritmos de IA para personalizar o conteúdo que vemos, influenciando nossas opiniões, comportamentos, julgamentos e até decisões políticas. Isso levanta preocupações sobre a polarização, a desinformação e o impacto da IA na democracia. É crescente também a aplicação de ferramentas de IA na área de entretenimento. Grandes empresas do cinema já exploram o uso de IA generativa em todo o seu *pipeline* de produção. A tecnologia está sendo usada para sincronizar as performances dos atores com várias dublagens em idiomas estrangeiros e está reinventando o que é possível fazer com os efeitos especiais. Em 2023, "Indiana Jones e a Relíquia do Destino" foi estrelado por um Harrison Ford *deepfake*, como visto na Figura 4.4

Além disso, a IA tem o potencial de influenciar a cultura e as artes. Algoritmos estão sendo usados para criar música, arte e até literatura, desafiando nossas noções tradicionais de criatividade e autoria. Enquanto isso, os assistentes virtuais baseados em IA estão mudando a forma como nos comunicamos, tornando as interações com a tecnologia mais naturais e conversacionais.

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

Figura 4.4: 'Deepfake' Harrison Ford em 'Indiana Jones e a Relíquia do Destino'.

Fonte: *https://fastcompanybrasil.com/co-design/*.

4.4.6 Impactos da IA no Mercado de Trabalho

Um dos desafios mais discutidos da IA é o impacto no mercado de trabalho. A automação impulsionada pela IA está substituindo trabalhadores humanos em várias indústrias, especialmente em funções que envolvem tarefas repetitivas e previsíveis. Isso tem gerado preocupações sobre o desemprego em larga escala e a necessidade de requalificação da força de trabalho.

Estudos sugerem que, embora a IA possa criar novas oportunidades de emprego, a transição pode ser difícil para muitos trabalhadores, especialmente aqueles em setores de baixa qualificação. Há também o risco de aumento da desigualdade, à medida que as oportunidades de emprego se concentram em áreas que exigem habilidades tecnológicas avançadas, exacerbando as disparidades econômicas e sociais.

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

4.4.7 Impactos da IA na Transparência e Explicabilidade

Um dos principais desafios técnicos da IA é a falta de transparência e explicabilidade dos sistemas. Muitos dos algoritmos de IA, especialmente aqueles baseados em aprendizado profundo, funcionam como "caixas-pretas", onde as decisões são tomadas com base em padrões complexos que nem mesmo os desenvolvedores conseguem explicar claramente.

Essa falta de explicabilidade pode ser problemática em áreas em que a confiança e a responsabilidade são cruciais, como na saúde, no setor financeiro e na justiça. Por exemplo, se um sistema de IA é usado para determinar se uma pessoa deve receber um empréstimo ou se deve ser liberada sob fiança, é essencial que as decisões possam ser explicadas e justificadas de maneira clara e transparente. Caso contrário, há um risco significativo de injustiças, discriminação ou erro.

4.4.8 Questões Éticas

As questões éticas relacionadas à IA são vastas e complexas. Uma das principais preocupações é o potencial viés nos algoritmos de IA. Como esses sistemas são treinados em grandes conjuntos de dados, qualquer preconceito presente nesses dados pode ser amplificado pela IA, levando a decisões que discriminam certos grupos com base em raça, gênero ou outras características.

Além disso, há preocupações sobre a autonomia dos sistemas de IA. À medida que as máquinas se tornam mais inteligentes e capazes de tomar decisões por conta própria, surgem questões sobre quem deve ser responsabilizado pelas ações desses sistemas. Isso é particularmente relevante em contextos como veículos autônomos ou sistemas de armamento, nos quais erros podem ter consequências fatais.

4.4.9 Privacidade e Segurança

As preocupações com privacidade e segurança são exacerbadas pela coleta e análise de grandes volumes de dados pessoais por sistemas de IA. O uso de IA em vigilância em massa, por exemplo, levanta sérias questões sobre o direito à privacidade e o potencial de abuso por parte de governos e corporações. Além disso, a segurança dos sistemas de IA é uma preocupação crescente, especialmente no contexto de ataques cibernéticos que visam manipular ou sabotar esses sistemas.

A proteção contra esses riscos requer não apenas a adoção de melhores práticas de segurança, mas também o desenvolvimento de normas e regulamentos que limitem o uso abusivo da IA e protejam os direitos dos indivíduos.

4.4.10 Impactos da IA na Desigualdade e Exclusão

A desigualdade é outro grande desafio associado à IA. O acesso a tecnologias avançadas de IA tende a ser concentrado em países e empresas com maiores recursos financeiros e tecnológicos, o que pode ampliar as disparidades globais e regionais. Além disso, dentro dos países, há o risco de que as inovações em IA beneficiem desproporcionalmente aqueles que já estão em uma posição de vantagem, enquanto os mais vulneráveis são deixados para trás.

A exclusão digital também é uma preocupação, especialmente em comunidades de baixa renda ou regiões menos desenvolvidas, onde o acesso à tecnologia e à internet é limitado. Isso pode levar a uma exclusão de oportunidades educacionais, econômicas e sociais acarretada pela IA, exacerbando as desigualdades existentes.

4.4.11 Governança e Regulação

A governança da IA é um desafio crucial que exige a cooperação internacional, a criação de novos *frameworks* regulatórios e a participação de diversas partes interessadas, incluindo governos, empresas, academia e a sociedade civil. A regulação da IA precisa equilibrar a promoção da inovação com a proteção dos direitos humanos e a mitigação dos riscos associados.

No entanto, a rápida evolução da tecnologia de IA torna difícil a criação de regulamentações que sejam, ao mesmo tempo, eficazes e flexíveis o suficiente para

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

acompanhar as mudanças. Além disso, há a questão da soberania tecnológica, em que diferentes países podem ter abordagens e prioridades diferentes em relação à regulação da IA, potencialmente levando a conflitos e fragmentação global.

4.5 O FUTURO DA IA

O futuro da Inteligência Artificial (IA) é uma das áreas mais discutidas e especuladas no campo da ciência, tecnologia e inovação, dentre outros. À medida que continua a se desenvolver, a IA deve impactar significativamente a vida humana em quase todos os aspectos. Isso compreende desde a automação e aprimoramento da tomada de decisões até a potencial criação de inteligências artificiais mais avançadas que podem superar a capacidade cognitiva humana. A IA está no centro de uma revolução tecnológica que moldará o século XXI.

Isso posto, a partir daqui, exploram-se as possíveis direções que a IA pode tomar nas próximas décadas, considerando os avanços tecnológicos esperados, suas aplicações em várias áreas, os desafios éticos e sociais que acompanham essa evolução, e as implicações globais e filosóficas para a humanidade.

4.5.1 Avanços Tecnológicos Esperados

Um dos maiores sonhos e medos, no campo da IA, é a criação da Inteligência Artificial Geral (AGI). Diferentemente da IA estreita ou especializada que existe hoje, que é projetada para tarefas específicas, a AGI teria a capacidade de entender, aprender e aplicar conhecimentos de maneira tão ampla e flexível quanto um ser humano. Isso significaria que a AGI poderia realizar qualquer tarefa cognitiva que um humano pudesse e, possivelmente, de maneira mais eficiente.

O desenvolvimento da AGI representaria uma mudança de paradigma. Atualmente, as IAs são altamente especializadas, como sistemas de recomendação ou algoritmos de processamento de linguagem natural. No entanto, uma AGI seria capaz de resolver problemas complexos em uma variedade de domínios, sem a necessidade de reconfiguração. Embora os cientistas ainda estejam longe de alcan-

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

çar uma AGI verdadeira, o progresso em áreas como aprendizado profundo, redes neurais e processamento de linguagem natural sugere que estamos avançando em direção a essa possibilidade.

4.5.2 Aprendizado Profundo e Aprendizado por Reforço

O aprendizado profundo (*deep learning*) já revolucionou muitas áreas da IA, desde a visão computacional até o reconhecimento de fala. No futuro, espera-se que essas técnicas se tornem ainda mais sofisticadas, permitindo que as IAs compreendam e interajam com o mundo de maneiras mais naturais e intuitivas. Isso inclui a capacidade de aprender com menos dados e de transferir conhecimento de uma tarefa para outra.

Além disso, o aprendizado por reforço (*reinforcement learning*) – uma técnica em que os sistemas aprendem comportamentos ideais por meio de recompensas e punições – também deverá ver grandes avanços. Aplicações futuras podem incluir desde a otimização de processos industriais até a tomada de decisões estratégicas em ambientes complexos, como operações militares ou gestão de crises globais.

4.5.3 Computação Quântica e IA

A computação quântica, embora ainda em sua infância, tem o potencial de impulsionar enormemente a IA. Computadores quânticos (ver Figura 4.5) poderiam processar vastas quantidades de dados a velocidades incomparáveis às dos computadores clássicos, permitindo que IAs resolvam problemas complexos que atualmente estão além de nosso alcance. Isso poderia revolucionar áreas como criptografia, simulações de moléculas para desenvolvimento de medicamentos e otimização de grandes sistemas, como redes de transporte ou cadeias de suprimentos globais.

Figura 4.5: Computador Quântico.

Fonte: (**Ferro2023**).

4.5.4 Aplicações Futuras da IA

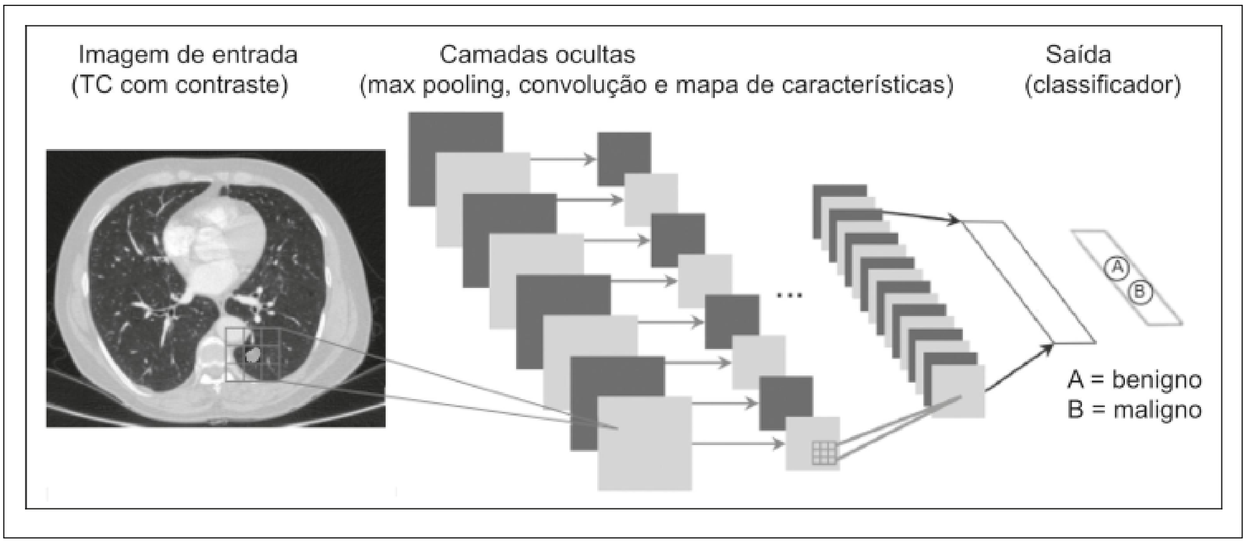
Na área da saúde, a IA tem o potencial de transformar desde a pesquisa e desenvolvimento de medicamentos até os cuidados personalizados. Com o avanço das técnicas de análise de *big data* e aprendizado de máquina, a medicina de precisão se tornará ainda mais eficaz, permitindo que tratamentos sejam adaptados às características genéticas e de estilo de vida de cada paciente.

Além disso, a IA pode desempenhar um papel crucial no diagnóstico precoce de doenças, pela análise de imagens médicas, exames de sangue e até mesmo mudanças sutis no comportamento e nas interações dos pacientes, ver Figura 4.6. No futuro, espera-se que a IA seja capaz de detectar doenças, como o câncer, em estágios iniciais, quando as chances de cura são muito maiores.

A IA poderá ainda melhorar a gestão de sistemas de saúde, otimizar o uso de recursos e auxiliar na previsão de surtos e pandemias. No entanto, esses avanços também trazem desafios significativos, como a privacidade dos dados dos pacientes e a necessidade de garantir que as decisões médicas baseadas em IA sejam transparentes e explicáveis.

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

Figura 4.6: Diagnóstico de tumor por IA.



Fonte: (**Santos2019**).

No setor educacional, a IA poderá personalizar a experiência de aprendizado para cada aluno, ajustando o conteúdo e o ritmo de acordo com as necessidades e o progresso individual. Sistemas tutoriais inteligentes, por exemplo, poderiam identificar áreas onde um aluno enfrenta obstáculos e fornecer materiais de suporte específicos para ajudá-lo a superar suas dificuldades. Técnicas como Realidade Aumentada podem criar ambientes virtuais que facilitem e estimulem o aluno a aprender assuntos como História e Biologia, por exemplo. O aluno poderá "caminhar"na Roma antiga quando estiver estudando o Império Romano, por exemplo.

A IA também pode ajudar a automatizar tarefas administrativas, como correção de provas e monitoramento do progresso dos alunos, permitindo que os educadores se concentrem mais no ensino e na interação humana. Além disso, com o avanço da realidade virtual e aumentada, a IA poderá criar ambientes de aprendizado imersivos, onde os alunos podem aprender com simulações interativas.

O impacto da IA na economia e no mercado de trabalho é talvez uma das questões mais discutidas e controversas. No futuro, a automação baseada em IA poderá eliminar muitos empregos, mas, por outro lado, a IA também criará novos empregos e empresas, especialmente em áreas que requerem habilidades avançadas em ciência de dados, robótica, computação e desenvolvimento de IA. A grande questão será como lidar com essa transição no mercado de trabalho e garantir que as pessoas tenham as habilidades necessárias para prosperar em um mundo onde a IA desempenha um papel central.

A Superinteligência Artificial (ASI) é um sistema de inteligência artificial (IA) baseado em *hardware* e *software*, com um escopo intelectual além da inteligência humana. No nível mais fundamental, essa IA superinteligente tem funções cognitivas de ponta e habilidades de pensamento altamente desenvolvidas mais avançadas do que qualquer ser humano, tornando possível a resolução de problemas sem que possamos compreender como elas foram realizadas. A questão é: como o ser humano vai lidar com a ASI?

Modelos econômicos podem precisar ser revisados para acomodar as mudanças trazidas pela automação e pela IA. Isso inclui explorar novas formas de distribuição de renda, como renda básica universal, ou políticas que incentivem a requalificação da força de trabalho. A IA também pode ser usada para melhorar a eficiência econômica, desde a otimização da produção industrial até a gestão de cadeias de suprimentos globais.

Com o aumento da dependência da IA em setores críticos, a segurança cibernética se tornará ainda mais vital. Sistemas de IA podem ser alvos de *hackers* ou manipulados para comportamentos prejudiciais. A segurança cibernética precisa evoluir para proteger essas tecnologias avançadas contra ameaças internas e externas. Além disso, a IA será crucial na defesa cibernética, utilizando algoritmos que detectam automaticamente ameaças em tempo real e a estas respondem.

Governos de todo o mundo também estão começando a explorar como a IA pode ser usada na governança, desde a previsão de crises econômicas e políticas até a automação de processos burocráticos. No entanto, a IA na governança também levanta preocupações sobre a transparência, a responsabilidade e a possibilidade de abuso de poder.

O uso crescente da IA levanta questões significativas sobre privacidade e controle de dados. À medida que as IAs se tornam mais integradas em nossa vida, elas coletam e analisam vastas quantidades de dados pessoais, o que pode levar a preocupações sobre como essas informações são usadas e protegidas. Há o risco de que a IA seja usada para vigilância ilegal de indivíduos ou grupos, tanto por governos quanto por corporações, o que poderia comprometer as liberdades individuais e os direitos à privacidade.

Para mitigar esses riscos, é essencial desenvolver estruturas de governança que garantam a proteção dos dados e que as práticas de IA sejam transparentes e responsáveis. Isso pode incluir o estabelecimento de regulamentos mais rigorosos sobre a coleta e o uso de dados, bem como o desenvolvimento de tecnologias de IA que respeitem a privacidade desde o início.

Outro desafio significativo é o viés nos sistemas de IA. Como as IAs são treinadas em grandes conjuntos de dados, qualquer viés presente nesses dados pode ser amplificado nos resultados. Isso pode levar a discriminação em áreas como emprego, crédito, saúde e justiça. Por exemplo, algoritmos de IA usados para decidir sobre empréstimos podem discriminar minorias se os dados históricos refletirem preconceitos passados.

Abordar o viés na IA requer tanto avanços técnicos quanto medidas regulatórias. Desenvolvedores de IA precisam criar métodos para identificar e corrigir vieses nos dados e nos algoritmos. Além disso, deve haver supervisão para garantir que as IAs sejam usadas de maneira justa e ética.

À medida que as IAs se tornam mais autônomas, surge a questão de quem é responsável pelas decisões tomadas por esses sistemas. Se uma IA toma uma decisão errada, como em um veículo autônomo que causa um acidente, quem deve ser responsabilizado? O fabricante do sistema, o programador ou o proprietário?

Estas questões éticas são complexas e exigem novas abordagens legais e filosóficas. Pode ser necessário desenvolver novas normas de responsabilidade para sistemas de IA, assim como diretrizes para garantir que as decisões automatizadas sejam explicáveis e justificáveis.

A disseminação da IA também tem o potencial de exacerbar desigualdades sociais e econômicas. Se o acesso à tecnologia de IA for concentrado em poucos indivíduos ou instituições, como grandes corporações ou países desenvolvidos, isso pode ampliar as disparidades entre ricos e pobres, tanto dentro das sociedades quanto entre nações.

É crucial que o desenvolvimento da IA seja acompanhado por políticas que garantam sua distribuição equitativa e que os benefícios sejam amplamente compartilhados. Isso inclui investir em educação e treinamento, para que mais pessoas possam participar da economia baseada em IA, e criar estruturas que promovam a colaboração global em vez da competição exacerbada.

A ascensão da IA levanta questões profundas sobre o papel da tecnologia na sociedade e o que significa ser humano em um mundo cada vez mais dominado por máquinas inteligentes. A IA pode alterar fundamentalmente o trabalho, a educação, as interações sociais e até mesmo a maneira como compreendemos nossa própria identidade.

Essas mudanças exigem uma reflexão filosófica sobre os valores que queremos preservar e como a IA pode ser usada para melhorar, em vez de degradar, a vida humana. O desafio será garantir que o desenvolvimento da IA seja guiado por princípios éticos sólidos e por uma visão clara do tipo de futuro que queremos criar para a humanidade

Finalmente, a IA tem implicações globais que exigem colaboração internacional. A criação de normas e regulamentos globais para a IA será essencial para evitar o uso abusivo da tecnologia e para promover um desenvolvimento sustentável e equitativo. Países e organizações internacionais precisarão trabalhar juntos para enfrentar os desafios comuns, como a segurança cibernética, o viés nos algoritmos e a proteção dos direitos humanos.

No entanto, essa cooperação não será fácil. A competição entre nações para

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

liderar o desenvolvimento da IA pode criar tensões e dificultar a criação de um consenso global. Ainda assim, a cooperação internacional será crucial para garantir que a IA beneficie a humanidade como um todo e não apenas alguns privilegiados.

4.6 CONSIDERAÇÕES FINAIS

As abordagens e técnicas de IA estão em constante evolução, à medida que os pesquisadores continuam a buscar métodos mais eficazes e eficientes para criar sistemas inteligentes. A combinação dessas técnicas permite que a IA seja aplicada em uma vasta gama de áreas, desde diagnósticos médicos até carros autônomos, transformando indústrias e melhorando a vida dos seres humanos. O futuro da Inteligência Artificial é repleto de possibilidades e desafios. À medida que a tecnologia continua a avançar, ela tem o potencial de transformar profundamente a sociedade, trazendo benefícios imensos, mas também novos riscos e questões éticas. A chave para um futuro positivo com a IA será equilibrar a inovação tecnológica com uma governança responsável, garantir que os benefícios sejam amplamente compartilhados e que os riscos sejam minimizados ou mitigados. A IA é não apenas uma ferramenta tecnológica e de inovação, mas uma ferramenta que pode redefinir o que significa ser humano no século XXI. Os impactos e desafios da IA são vastos e multifacetados, afetando quase todos os aspectos da vida moderna. Embora a IA tenha o potencial de trazer benefícios significativos para a economia, saúde, educação e segurança, também apresenta desafios consideráveis, como o impacto no mercado de trabalho, desigualdade, falta de transparência, questões éticas e a necessidade de uma governança robusta. À medida que a tecnologia de IA continua a evoluir, é essencial que a sociedade como um todo se mobilize em um diálogo contínuo sobre como melhor integrar e regular essa poderosa tecnologia. Somente por meio de uma abordagem colaborativa, multidisciplinar e responsável, será possível maximizar os benefícios da IA enquanto se mitigam seus riscos, garantindo que seus avanços sejam amplamente compartilhados e acessíveis a todos.

Capítulo 5

Mandalas Fractais: explorando simetrias em padrões recorrentes

Autor: João Paulo Martins dos Santos[1]
Instituição: Academia da Força Aérea, Pirassununga, SP
e-mail: jp2@alumni.usp.br

5.1 INTRODUÇÃO

Os fractais são estruturas geométricas complexas que exibem padrões auto-similares em diferentes escalas. Mandalas, no sentido explorado em **livro01**, são estruturas com simetria radial, provenientes de um processo de composição de formas geométricas mais simples. Ambas as estruturas exibem padrões geométricos com apelo visual e que capturam a atenção devido à complexidade resultante dos processos construtivos e de coloração. A combinação da beleza intrínseca das mandalas com a complexidade e a auto-similaridade dos fractais oferece um campo fértil para a criação de figuras complexas e visualmente impactantes. Nesse contexto, a exploração das mandalas, no sentido adotado em **livro01**, foi delineada como proposta do presente capítulo.

A principal motivação para explorar fractais na construção de mandalas foi o processo de construção e coloração desenvolvido em **livro01** com resultados complementares em **livro02** e **xornadas2023**. Os autores empregam curvas com expressões analíticas na forma paramétrica, mas destacam que o processo não é dependente da curva em questão. Além da versatilidade do método, os resultados são figuras cuja beleza remete, de certa forma, aos aspectos das formas fractais.

5.2 OBJETIVO

O principal objetivo deste capítulo é explorar a construção de figuras com simetria radial, denominadas Mandalas, por meio das curvas fractais.

Os objetivos específicos são:

- discutir as principais curvas fractais e possibilidades de construções;
- discutir os processos de construção e coloração;
- refletir sobre as vantagens e desvantagens que a assistência cognitiva pode fornecer a um iniciante na linguagem de programação R.

5.3 ELEMENTOS BÁSICOS

O termo Mandala, aqui, refere-se às figuras geradas por transformações geométricas de homotetias, rotações e translações que apresentem certa simetria radial. Outras transformações matemáticas ainda permanecem inexploradas, tais como transformações afins, transformações lineares ou mesmo transformações do tipo log-normal.

O conjunto de passos do processo de construção das mandalas, apresentado a seguir, não precisa seguir a ordem estrita de apresentação, ou seja, é um delineamento de elementos norteadores da construção (**livro02**) que podem ser executados em qualquer ordem. A questão principal é o estabelecimento da figura base e a composição e o método de coloração. Cada um desses elementos atribui uma característica específica à construção.

1.) Escolha das curvas ou figuras geométricas.

2.) Aplique as transformações geométricas de rotação, homotetia ou translação.

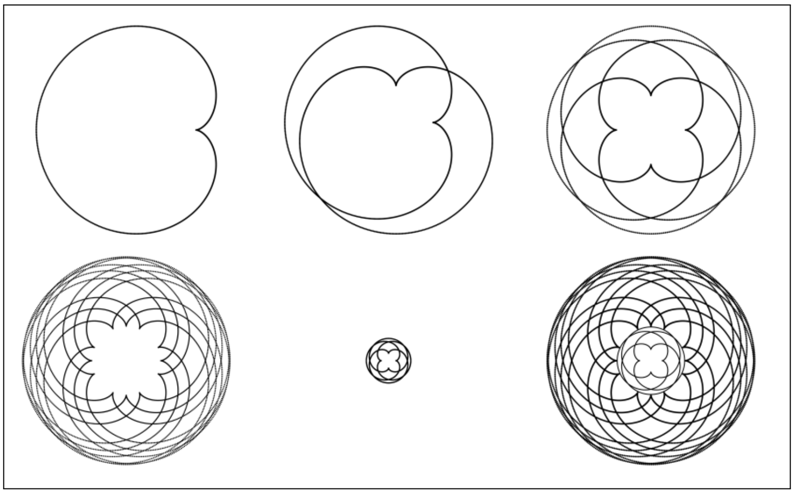
3.) Defina o modelo de cores e escolha da paleta de cores.

4.) Especifique o padrão de cores, ou seja, a sequência em que os objetos serão coloridos.

5.) Realize as composições de uma ou mais figuras.

Ilustrativamente, a Figura 5.1 mostra os passos 1. 2. e 5., os quais são, respectivamente, a escolha da curva cardioide, as rotações sucessivas para gerar a quarta figura (a segunda e terceira figuras são intermediárias) e a homotetia da terceira figura (quinta figura); por fim, a composição da quarta e da quinta figura é apresentada na última figura. É nesse sentido que o processo de construção não precisa seguir uma ordem estrita, pois diferentes estratégias de rotações combinadas com homotetias (ou vice-versa) ou curvas iniciais fornecem diferentes figuras.

Figura 5.1: O processo de composição utilizando das Mandalas: **a.)** rotação, **b.)** rotações sucessivas, **c.)** rotações sucessivas de b.), **d.)** homotetia de b.), **e.)** composição de b.) e c.)

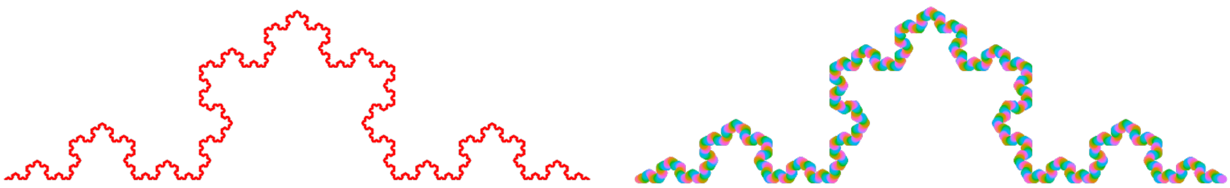


Fonte: (**livro02**).

No contexto dos fractais, cujos elementos principais serão discutidos na Seção 5.3.2, o item 1.) exige uma curva tal como a **curva de Coch**, por exemplo. A

Figura 5.2 ilustra o processo de construção com foco nas curvas fractais e os dois padrões de cores, utilizando modelo RGB, aplicados à curva de Koch.

Figura 5.2: A curva de Koch c composta de um conjunto de n pontos no plano e paletas de cores $p_1 = \{p_{1_i}\}_{i=1}^{n} = "red"$ e $p_2 = \{p_{2_i}\}_{i=1}^{n} = c("red", "blue", "green", ...)$.



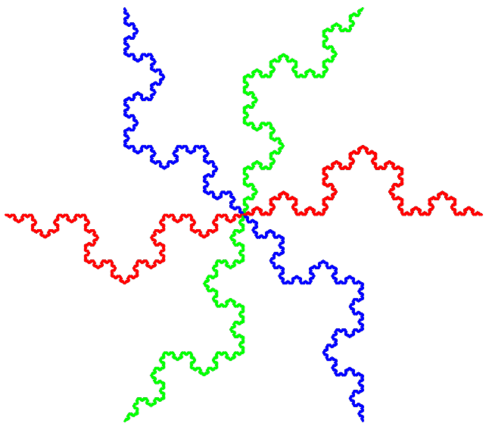
(a) Curva de Koch c com única cor. (b) Curva de Koch c com três cores.

Fonte: O autor, 2024.

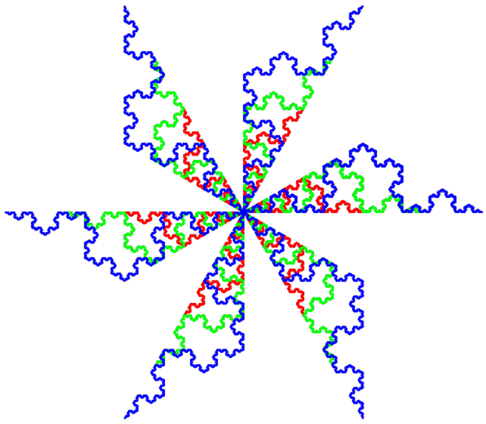
Os processos de coloração anteriores podem ser colocados de maneira mais formal. Seja c a curva de Koch, composta de um conjunto de n pontos no plano, e $p_1 = \{p_{1_i}\}_{i=1}^{n}$ e $p_2 = \{p_{2_i}\}_{i=1}^{n}$, as paletas de cores utilizadas considerando o modelo de cores RGB. Assim, o passo 4.) é ilustrado nas Figuras 5.2-a e 5.2-b com diferentes esquemas de coloração; no primeiro caso a sequência é $p_1 = \{p_{1_i}\} = "red"$, enquanto que no segundo, a paleta de cores é uma replicação ($rep(**)$) das cores $c("red", "blue", "green")$ para o número de rotações adotado, ou seja, $p_2 = \{p_{2_i}\} = c("red", "blue", "green", \cdots)$. No R a cor vermelha é utilizada diretamente, enquanto que a a replicação para as rotações é dependente da quantidade de rotações e feita por $rep(c("red", "blue", "green"), length.out = dim(dt)[1]))$, em que $dim(dt)[1]$ é a dimensão do *dataframe*.

Considere a Figura 5.5-a, a qual mostra uma sequência de rotações $rotacao = c(0.0, pi/k * (1 : (2 * k)))$ com $k = 3$, considerando que $\{cor_j\}_{j=1}^{Nrot}$ é a paleta de cores para cada rotação. Assim, cada rotação de c, c_j, é colorida com a mesma cor cor_j, ou seja, o conjunto de pontos rotacionados é colorido seguindo o esquema da Figura 5.2-a. Em sequência considere a Figura 5.5-b, a qual apresenta rotações da imagem apresentada na Figura 5.5-a, seguidas de homotetias de razões $r_l = 0.5, 0.75$ e 1; agora, as rotações de c constituem a curva c_1 (curva de nível 1 - composição de todas as rotações de c), à qual a paleta de cores será aplicada. Neste caso, cada composição possui a mesma cor, mas as homotetias são coloridas de forma distinta.

Figura 5.3: A curva de Koch c com n pontos no plano e paletas de cores $p_1 = \{p_{1_i}\}_{i=1}^{n} = c("red", "blue", "green")$ e $p_2 = \{p_{2_i}\}_{j=1}^{N} = rep(c("red", "blue", "green"), length.out = dim(dt1)[1])$, em que n é o comprimento do vetor de c, enquanto que N é o comprimento da composição de rotações sucessivas $N = 6 \cdot n$.



(a) Curva de Koch rotacionada e cores distintas para as rotações.



(b) Curva de Koch c rotacionada (composição c_1) coloridas da mesma cor; homotetias de c_1 coloridas de cores distintas.

Fonte: O autor, 2024.

Uma observação é que os passos 2.) e 3.) poderiam ter sido aplicados aos resultados da composição de 5.) e depois, novamente, as homotetias e rotações. É, portanto, nesse sentido, que o processo construtivo não precisa ser estritamente seguido na ordem apresentada.

Resultados que englobam tanto rotações quanto homotetias (passo 5) dos elementos principais das Figuras 5.2-a,b são apresentados nas Figuras 5.5-a,b.

Uma discussão sobre os elementos anteriores como foco na construção das Mandalas utilizando curvas planas pode ser encontrada em (**livro02**), disponível em Portal de Livros Abertos da USP. Os pormenores das discussões de construção das Mandalas e respectivos códigos podem ser obtidos diretamente em (**livro01**), disponível em Portal de Livros Abertos da USP. Apesar da diversidade de resultados das referências apontadas, há campo para a exploração; avanços em relação aos métodos de coloração podem ser consultados em (**xornadas2023**), no qual os autores discutem outras possibilidades de coloração das figuras que conduzem a resultados distintos. Deve ser observado, no entanto, que os métodos de coloração não são esgotados nas referências apresentadas.

5.3.1 Programação, Inteligência Artificial e softwares

A programação computacional é o elemento essencial na geração de fractais relacionados aos processos iterativos. De forma geral é o elemento fundamental para o desenvolvimento, por meio das linguagens de programação, de algoritmos que permitem a criação e manipulação do conjunto de pontos cuja visualização fornece as estruturas utilizadas na construção das mandalas.

As linguagens de programação R (Python, entre outras) fornecem pacotes (bibliotecas) capazes de gerar estruturas fractais que, de certa forma, facilitam a geração das figuras. A utilização dessas bibliotecas possibilitam, de certa forma, acelerar o processo de construção e obtenção das figuras, porém pode ser interessante compreender os conceitos elementares de programação para uma visão mais adequada do processo computacional. Um ponto de partida para tais elementos pode ser o Capítulo 2 (Uma Introdução à Programação com o R) e referências.

Um elemento que deve ser mencionado no contexto da programação é a utilização combinada dos modelos interativos do tipo *chat bots* (ChatGPT, Gemini, por exemplo), os quais podem ser facilitadores do aprendizado de codificação. Deve ser observado que as denominadas Inteligências Artificiais generativas (IA's generativas), tais como ChatGPT (Modelo de linguagem (Versão GPT-4), disponível em: https://www.openai.com) e Gemini (Modelo de linguagem Gemini, disponível em: https://gemini.google.com/app), por exemplo, são vistos no momento como ferramenta de produtividade e aprendizado e não substituem o conhecimento básico em programação computacional, fruto da dedicação pessoal. Este conhecimento é necessário para avaliar a qualidade dos resultados fornecidos e consequente ajuste ou rejeição da resposta. Uma discussão de resultados provenientes das interações, com foco na linguagem R, são fornecidos no Capítulo 3 (Programação Aprimorada com R Utilizando Assistência Cognitiva Artificial).

Retornando às questões dos fractais, há softwares especializados para a geração de figuras fractais, mas o objetivo do capítulo é a geração de mandalas. Nesse contexto, no momento, apenas fractais que derivam de processos iterativos associados à geometria. Como exemplo, a curva de Koch é obtida por meio de processo iterativo aplicado a um segmento de reta.

5.3.2 Fractais

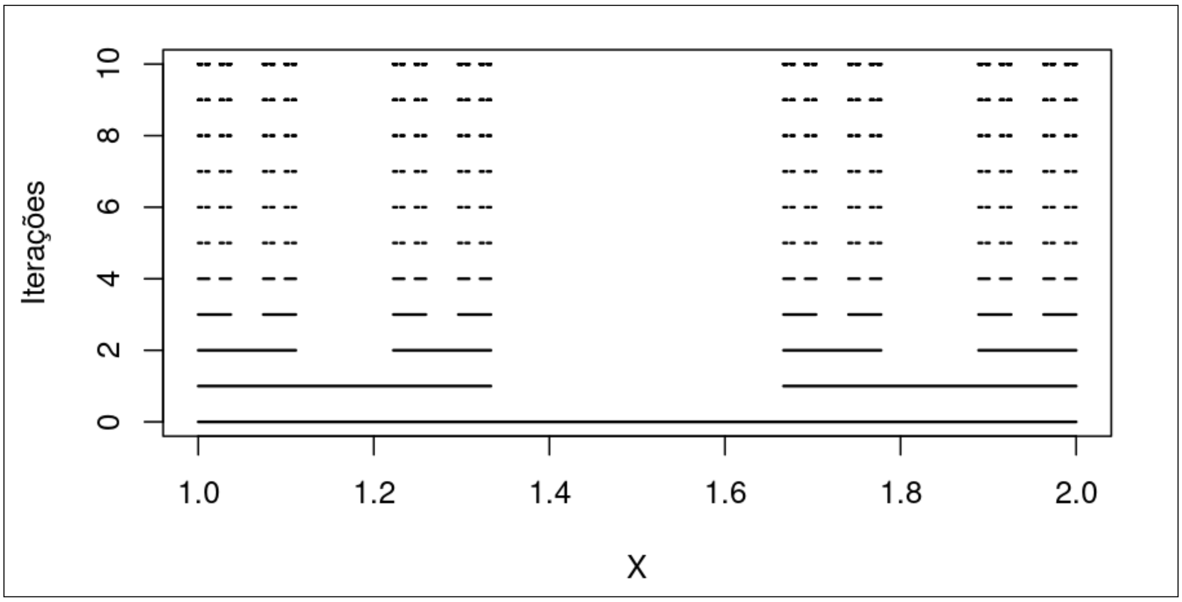
A definição precisa de um fractal requer um certo arcabouço de conhecimento na área de Matemática. Os elementos principais de um fractal, no entanto, podem ser ilustrados por meio do conjunto 1/3 de Cantor (**falconer2003**).

O conjunto um terço de Cantor é obtido por meio de um processo recursivo aplicado a um segmento de reta: i.) o terço medial de um segmento é extraído; ii.) extraia o terço medial de cada um dos segmentos resultantes de i.); iii.) repita o processo de extração dos terços medias de ii.). A repetição indefinida do processo descrito resulta no Conjunto de um terço de Cantor F, o conjunto dos pontos restantes após todo o processo.

Um estrutura fractal cujas 10 primeiras iterações $E^k, k = 1, \cdots, 10$ são mostradas na Figura 5.4.

Segundo **falconer2003**, as seguintes propriedades do conjunto 1/3 de Cantor são características dos fractais:

Figura 5.4: Conjunto 1/3 de Cantor com 10 interações. Iterações mostradas no eixo y.



Fonte: O autor, 2024.

Auto-similaridade: O conjunto **F** é auto-similar. As partes de **F** no intervalo **[0, 1/3]** e a parte de **F** em **[2/3, 1]** são geometricamente similares a **F**, escaladas por um fator **1/3**. De forma análoga, o raciocínio pode ser repetido aplicado sucessivamente para outros estágios da construção, resultando que o "conjunto de Cantor contém cópias de si mesmo em diversas escalas".

Estrutura Fina: O conjunto **F** possui uma "estrutura fina", ou seja, contém detalhes em escalas arbitrariamente pequenas. Quanto mais ampliamos a imagem do conjunto de Cantor, mais lacunas se tornam aparentes.

Definição Simples: Embora **F** tenha uma estrutura intrincada, sua definição é bastante simples com base em argumento geométrico.

Procedimento Recursivo: **F** é obtido por um procedimento recursivo.

Geometria Não Clássica: A geometria de **F** não é facilmente descrita em termos clássicos, ou seja, não pode ser descrita como o lugar geométrico dos pontos que satisfazem alguma condição geométrica simples, nem é o conjunto de soluções de qualquer equação simples.

Geometria Local: É difícil descrever a geometria local de **F**. Próximo a cada um de seus pontos, há um grande número de outros pontos, separados por lacunas de tamanhos variados.

Medida Nula: Embora **F** seja um conjunto incontável, seu tamanho não é quantificado pelas medidas usuais, como o comprimento, pois **F** tem comprimento zero.

O código a seguir foi utilizado na figura anterior.

```
# Função para gerar o conjunto de Cantor
generate_cantor_set <- function(iterations, start = 0, end = 1) {
  segments_list <- list(list(c(start, end)))

   for (i in 1:iterations) {
    current_segments <- segments_list[[i]]
    new_segments <- list()

    for (segment in current_segments) {
      start <- segment[1]
      end <- segment[2]
      third <- (end - start) / 3

      # Adiciona os dois segmentos restantes após remover o terço do
         meio
      new_segments <- append(new_segments, list(c(start, start + third))
         )
      new_segments <- append(new_segments, list(c(end - third, end)))
      }
      segments_list[[i + 1]] <- new_segments
      }
      return(segments_list)
      }

plot_cantor_set <- function(segments_list) {
  iterations <- length(segments_list) - 1
  plot(a, a, type = "n", xlim = c(a,b), ylim = c(0, iterations),
       xlab = "X", ylab = "Iterações", main = " ")

  for (i in 0:iterations) {
    segments <- segments_list[[i + 1]]
    for (segment in segments) {
```

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

```
        segments(segment[1], i, segment[2], i, lwd = 1.5) } }
        }

    iterations <- 10;  a=1;b=2.0
    plot_cantor_set(cantor_set)
```

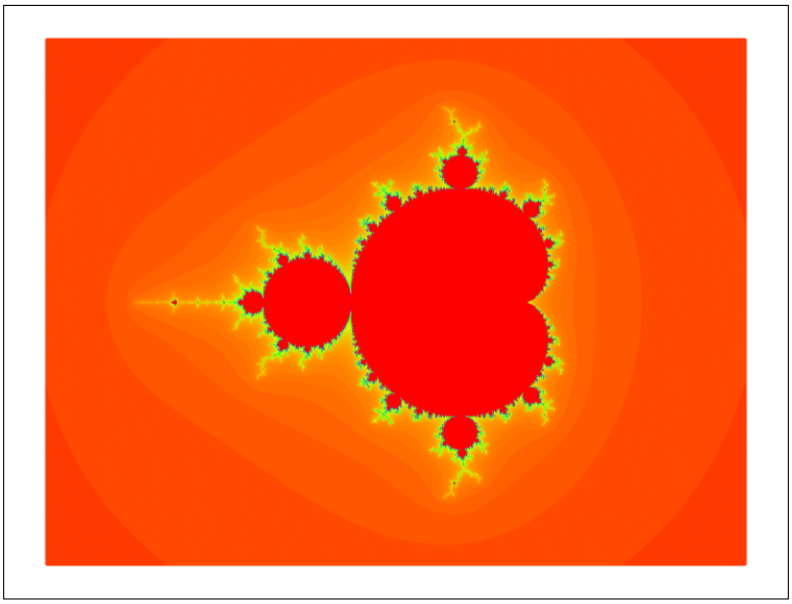
Outros exemplos e discussões adicionais são apresentados em (**falconer2003**); um texto com foco em Matemática é, por exemplo, (**barnsley1988fractals**). Do ponto de vista não matemático aqui adotado, um fractal é uma estrutura geométrica que pode ser dividida em partes, cada uma das quais possui similaridade com o todo, do ponto de vista qualitativo, e é obtido por meio de procedimento computacional recursivo. Também, o interesse é a figura gerada e não uma discussão pormenorizada do processo de construção, exploração de propriedades matemáticas ou mesmo demonstrações.

A seguir alguns exemplos de conjuntos fractais e respectivos códigos, em linguagem R (**Rcore**), por meio da interface gráfica *RStudio Posit* (**Rstudio**). Analogamente às referências (**livro01**), (**livro02**) e (**xornadas2023**), as cores são baseadas, primariamente, na função `colors()`, disponível no R Core. A visualização gráfica é focada na utilização do pacote **ggplot2** (**hardley**) por meio da utilização dos *tibbles* (**tibble**).

5.3.2.1 Conjunto de Mandelbrot

Um dos fractais mais famosos, o conjunto de Mandelbrot, é definido por uma simples fórmula matemática e exibe uma complexidade incrível. O fractal de Mandelbrot é gerado por meio da iteração da equação $z_{n+1} = z_n^2 + c$, em z e c são números complexos (**falconer2003**); a iteração em que os módulos dos valores divergem para o infinito são anotadas e coloridas de acordo com alguma regra específica. O resultado desse processo é mostrado na Figura 5.5.

Figura 5.5: O conjunto de Mandelbrot obtida com apoio de IA.



Fonte: O autor, 2024.

```
# Load required libraries
library(ggplot2)

# Function to generate Mandelbrot set
mandelbrot_set <- function(xlim, ylim, res, max_iter) {
  # Create a grid of complex numbers
  x <- seq(xlim[1], xlim[2], length.out = res)
  y <- seq
  grid <- expand.grid(x = x, y = y)
  c <- complex(real = grid$x, imaginary = grid$y)

  # Initialize the z array and iteration matrix
  z <- matrix(0, nrow = length(x), ncol = length(y))
  iterations <- matrix(0, nrow = length(x), ncol = length(y))

  for (i in 1:max_iter) {z <- z^2 + c
    divergent <- abs(z) > 4;
    iterations[divergent & iterations == 0] = i
    z[divergent] <- NA }

  # Create a data frame for ggplot
  df <- data.frame(
    x = rep(x, each = length(y)),
    y = rep(y, times = length(x)),
```

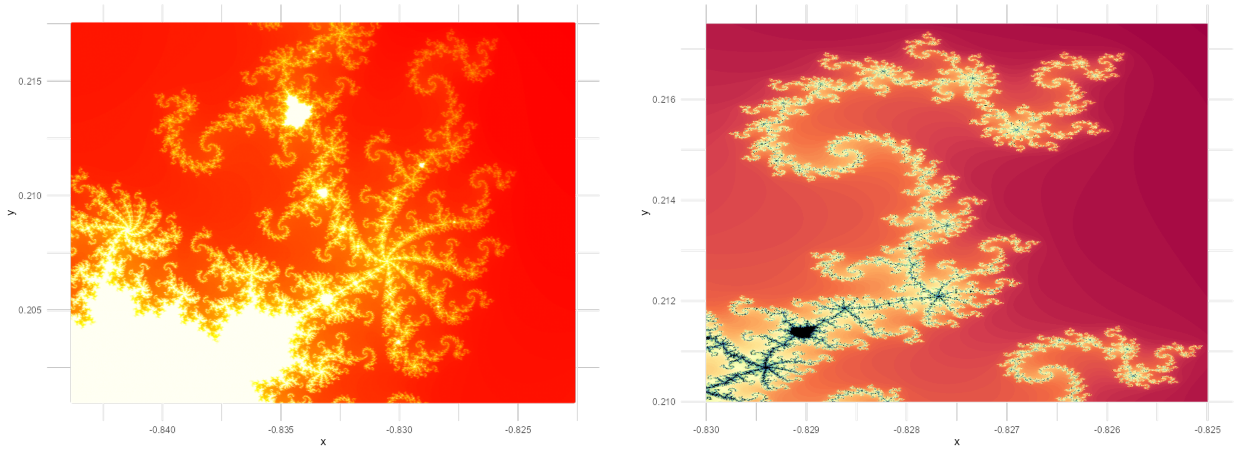
```
    iterations = as.vector(t(iterations))    )
  return(df)  }

# Parameters
xlim <- c(-2.5, 1.5)
ylim <- c(-1.5, 1.5)
res <- 1000
max_iter <- 100

# Generate Mandelbrot set
mandelbrot_df <- mandelbrot_set(xlim, ylim, res, max_iter)

# Plot the Mandelbrot set
p <- ggplot(mandelbrot_df,
aes(x = x, y = y, color = iterations)) +
  geom_point(alpha = 0.6, size = 0.5) +  # Adjust alpha and size as
     needed
  scale_color_gradientn(colors = rainbow(10)) +  # Use a gradient color
     scale
  coord_fixed(ratio = 1) +  # Ensure equal aspect ratio
  theme_void() +  # Remove axes and background
  theme(legend.position = "none")  # Remove legend
# Display the plot
print(p)
```

Construções de Mandelbrot com base em bibliotecas do R podem ser obtidas, por exemplo, no pacote **Mandelbrot** [2]. Nesse caso, há outras opções gráficas associadas às funções geradoras. As Figuras 5.6 e 5.7 são ilustrativas.

Apesar dos resultados interessantes, Mandelbrot resistiu ao processo de construção de Mandalas seguindo a metodologia delineada em (**livro01**), pois é uma figura no plano cartesiano. Este fato ocasiona distorções ao utilizar rotações sucessivas e, também, há sobreposição das imagens ao realizar composições. Dessa forma, restringiu-se à utilização das curvas de Koch, Floco de Neve de Koch e árvore fractal, pois podem ser colocadas em forma similar àquelas utilizadas em (**livro01**).

[2]disponível em https://github.com/blmoore/mandelbrot, acessado em 29 de agosto de 2024.

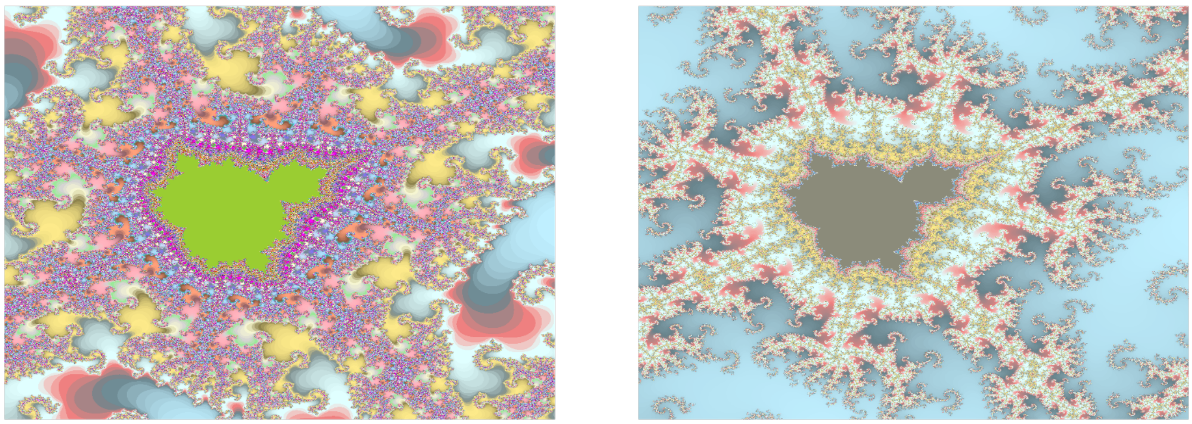
Figura 5.6: Conjunto de Mandelbrot restrito a uma região do plano com diferentes mapas de cores obtido do pacote **Mandelbrot**, disponível em https://github.com/blmoore/mandelbrot.



(a) Restrição 01 (b) Restrição 02

Fonte: O autor, 2024.

Figura 5.7: Conjunto de Mandelbrot restrito a uma região do plano com diferentes mapas de cores obtido do pacote **Mandelbrot**, disponível em https://github.com/blmoore/mandelbrot.



(a) Restrição 03 (b) Restrição 04

Fonte: O autor, 2024.

5.3.2.2 Curva de Koch

Um exemplo clássico de fractal geométrico é a Curva de Koch mostrada na Figura 5.2. Uma discussão com traçado de um perfil histórico pode ser obtida em (**rani01**); esta curva é um excelente exemplo de como uma construção geométrica simples, analogamente ao conjunto um terço de Cantor, pode levar a uma forma fractal. A construção da Curva de Koch começa com um segmento de linha reta e o processo de iteração consiste em dividir cada segmento de linha em três partes iguais e substituir a parte do meio por dois segmentos que formam um triângulo equilátero, sem a base.

A seguir os códigos utilizados para a geração das curva de Koch, da curva floco de neve de Koch e arvore fractal são apresentados a seguir. Os pormenores com exemplos mais específicos podem ser obtidos diretamente em Link.

```
# function to draw a single line
drawLine <- function(line, col="white", lwd=1) {
  segments(x0=line[1], y0=line[2], x1=line[3], y1=line[4], col=col, lwd=
     lwd) }
# wrapper around "drawLine" to draw entire objects
drawObject <- function(object, col="white", lwd=1) {
  invisible(apply(object, 1, drawLine, col=col, lwd=lwd))  }
```

```
# function to add a new line to an existing one
 newLine <- function(line, angle, reduce=1) {
  x0 <- line[1]; y0 <- line[2]; x1 <- line[3]; y1 <- line[4]
  dx <- unname(x1-x0)                       # change in x direction
  dy <- unname(y1-y0)                       # change in y direction
  l <- sqrt(dx^2 + dy^2)                    # length of the line
  theta <- atan(dy/dx) * 180 / pi           # angle between line and
     origin
  rad <- (angle+theta) * pi / 180           # (theta + new angle) in
     radians
  coeff <- sign(theta)*sign(dy)             # coefficient of direction
  if(coeff == 0) coeff <- -1
  x2 <- x0 + coeff*l*cos(rad)*reduce + dx   # new x location
  y2 <- y0 + coeff*l*sin(rad)*reduce + dy   # new y location
  return(c(x1,y1,x2,y2)) }
```

```
iterate= function(object, ifun, ...) {
  linesList <- vector("list",0)
  for(i in 1:nrow(object)) {
    old_line = matrix(object[i,], nrow=1)
    new_line =ifun(old_line, ...)
    linesList[[length(linesList)+1]] = new_line }
  new_object <- do.call(rbind, linesList)
```

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

```
  return(new_object)  }
```

```
# iterator function: koch curve
 koch <- function(line0) {
  # new triangle (starting at right)
  line1 <- newLine(line0, angle=180, reduce=1/3)
  line2 <- newLine(line1, angle=-60, reduce=1)
  line3 <- newLine(line2, angle=120, reduce=1)
  line4 <- newLine(line3, angle=-60, reduce=1)

  # reorder lines (to start at left)
  line1 <- line1[c(3,4,1,2)]
  line2 <- line2[c(3,4,1,2)]
  line3 <- line3[c(3,4,1,2)]
  line4 <- line4[c(3,4,1,2)]

  # store in matrix and return
  mat <- matrix(c(line4,line3,line2,line1), byrow=T, ncol=4)
  return(mat) }
```

A utilização das funções acima para obter a curva de Koch com seis iterações é fornecida a seguir.

```
# example: Koch curve (after six iterations)
fractal <- matrix(c(0,0,1,1e-9), nrow=1)
for(i in 1:6) fractal <- iterate(fractal, ifun=koch)
```

Para a visualização com o **ggplot2** as seguintes modificações foram introduzidas. Os códigos a seguir ilustram a visualização da curva de Koch (Figura 5.2-a) com cor única vermelha. Note que os resultados do processo iterativo foram convertidos para proporcionar a utilização do **ggplot2**.

```
df=data.frame(fractal)
xt=c(df$X1,df$X3); yt=c(df$X2,df$X4)
dt=tibble::tibble(xt,yt)
p=ggplot()+coord_fixed()+    theme_void()
p=p+ geom_point(data=dt, aes(x=X1, y=X2), color='red',size=.0015)
p
```

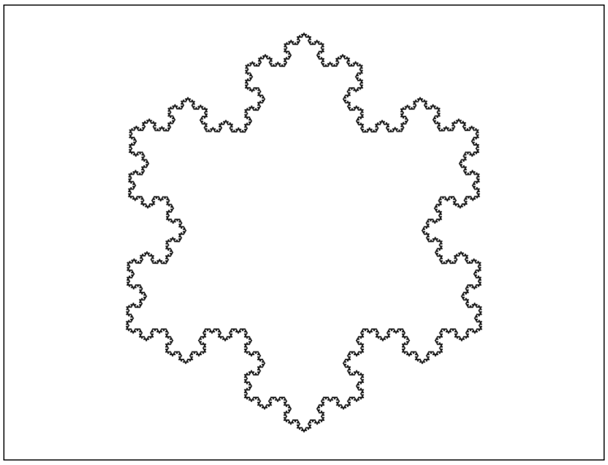
A modificação da paleta de cores permite obter o resultado da Figura 5.2-b de forma imediata. O código é apresentado a seguir:

```
dt$color= rep(c("red","blue","green"),length.out = dim(dt)[1])
p=p+ geom_point(data=dt, aes(x=X1, y=X2,color=color),size=1.5)+
theme(legend.position ="none")
```

5.3.2.3 Curva Floco de Neve de Koch

Uma consequência direta do processo de substituição do terço medial de um segmento (curva de Koch) é a aplicação do processo iterativo a um triângulo equilátero ΔABC. Neste caso, cada um dos lados é substituído de forma iterativa pelo processo descrito anteriormente. A Figura 5.8 é o resultado do processo iterativo descrito:

Figura 5.8: Curva floco de Neve de Koch.



Fonte: O autor com base nos códigos disponíveis em Link.

O código para a curva floco de neve de Koch é apresentado a seguir (Ver Link):

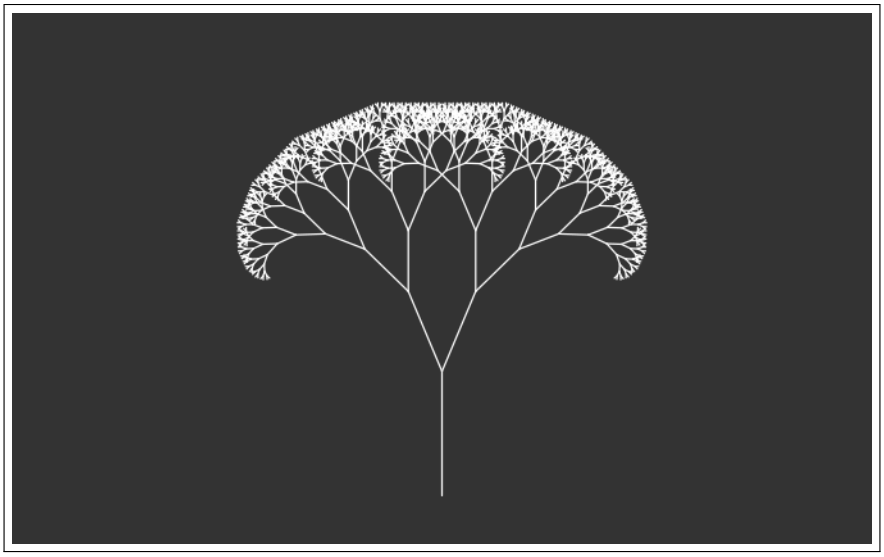
```
# Koch snowflake (after six iterations)
A <- c(0,1e-9)
B <- c(3,5)
C <- c(6,0)
fractal2 <- matrix(c(A,B,B,C,C,A), nrow=3, byrow=T)
for(i in 1:5) fractal2 <- iterate(fractal2, ifun=koch)

df=data.frame(fractal2)
xt=c(df$X1,df$X3)
```

```
yt=c(df$X2,df$X4)

dt=tibble::tibble(xt,yt)

names(dt)=c("X1","X2")
p=ggplot()+coord_fixed()+    theme_void()
#p=p+ geom_point(data=dt1, aes(x=xt1, y=yt1), color='black',size=.015)
p=p+ geom_point(data=dt, aes(x=X1, y=X2), color='black',size=.0015)
p
```

5.3.2.4 Árvore Fractal

A árvore fractal é um elemento definido em termos matemáticos por meio de um processo recursivo de construção dos ramos. A Figura 5.9 ilustra o processo:

Figura 5.9: Árvore Fractal.



Fonte: O autor com base nos códigos disponíveis em Link.

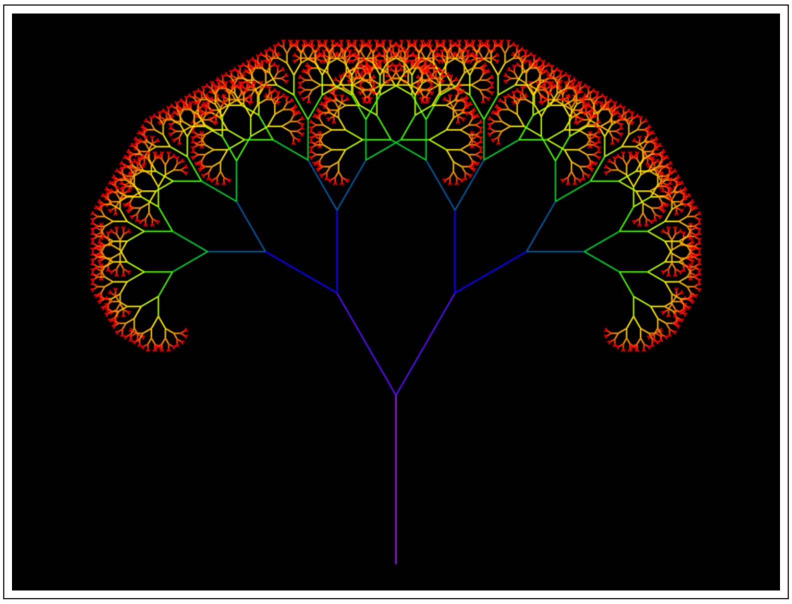
Os códigos a seguir são ilustrativos (Ver códigos disponíveis em Link):

```
# iterator function: recursive tree
tree <- function(line0, angle=30, reduce=.7, randomness=0) {
# angles and randomness
angle1 <- angle+rnorm(1,0,randomness)  # left branch
angle2 <- -angle+rnorm(1,0,randomness) # right branch
# new branches
line1 <- newLine(line0, angle=angle1, reduce=reduce)
```

```
  line2 <- newLine(line0, angle=angle2, reduce=reduce)
  # store in matrix and return
  mat <- matrix(c(line1,line2), byrow=T, ncol=4)
  return(mat)   }

  # example: recursive tree (after ten iterations)
  fractal <- matrix(c(0,0,0,10), nrow=1)
  emptyCanvas(xlim=c(-30,30), ylim=c(0,35))
  drawObject(fractal)
  for(i in 1:10) {
    fractal <- iterate(fractal, ifun=tree, angle=23)
    drawObject(fractal)  }
```

A Figura 5.9 ilustra uma metodologia distinta para a árvore fractal, obtida com apoio de IA. Ambas serão utilizadas para a construção das Mandalas Fractais na seção 5.4, originando figuras distintas.

Figura 5.10: Árvore Fractal.



Fonte: O autor, 2024.

Variações podem ser inseridas por meio de uma modificação direta dos códigos fornecidos, como por exemplo, alteração na simetria dos ramos. O código completo para os cálculos e a visualização da árvore fractal da Figura 5.10 são apresentados a seguir:

```r
library(ggplot2)
# Recursive function to generate the fractal tree with depth information
generate_tree <- function(x, y, angle, length, depth, max_depth) {
  if (depth == 0) return()
  # Calculate the end points of the branches
  x_end <- x + length * cos(angle);  y_end <- y + length * sin(angle)
  # Store the branches in a data frame with depth information
  branches <<- rbind(branches, data.frame(x = x, y = y, x_end = x_end, y
     _end = y_end, depth = depth))
  # Recursive calls for left and right branches
  generate_tree(x_end, y_end, angle - pi / 6, length * 0.7, depth - 1,
     max_depth)
  generate_tree(x_end, y_end, angle + pi / 6, length * 0.7, depth - 1,
     max_depth)
  }

# Initialize an empty data frame to store branches
branches <- data.frame(x = numeric(), y = numeric(), x_end = numeric(),
   y_end = numeric(), depth = numeric())
max_depth <- 12# Define the maximum depth
# Generate the tree
generate_tree(0, 0, pi / 2, 1, max_depth, max_depth)

# Create a gradient color palette
color_palette <- colorRampPalette(colors = c("red", "orange", "yellow",
   "green", "blue", "purple"))(max_depth)

# Plot the fractal tree using ggplot2 with the custom gradient color
   palette
p=ggplot(branches) +
  geom_segment(aes(x = x, y = y, xend = x_end, yend = y_end, color = as.
     factor(depth)), size = 0.5) + scale_color_manual(values = color_
     palette) +
  theme_void() + coord_equal() +
  theme(plot.background = element_rect(fill = "black"), panel.background
      = element_rect(fill = "black"),legend.position = "none") +
  guides(color = guide_legend(title = "Depth"))
p
```

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

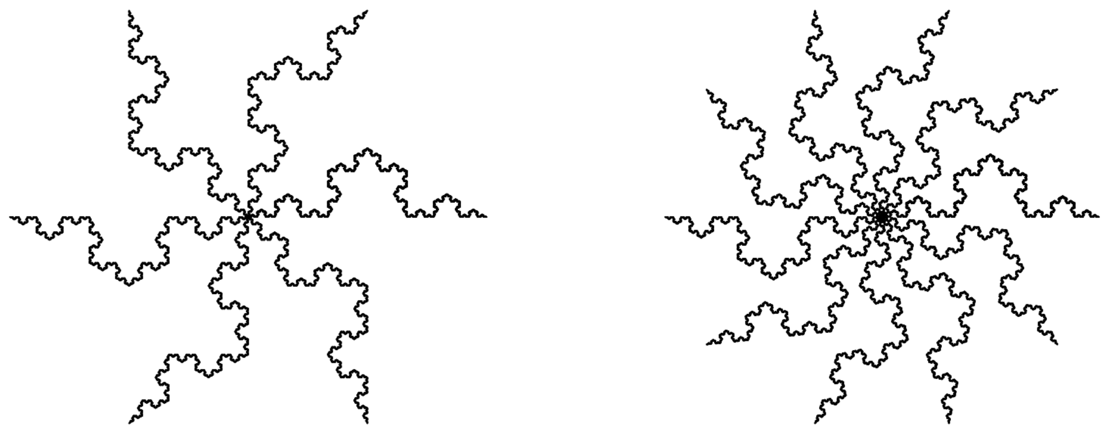
5.4 MANDALAS FRACTAIS

As mandalas fractais são obtidas por meio da aplicação de transformações rígidas no plano aplicadas aos fractais Curva de Koch, Curva Floco de Neve e Árvore Fractal. No momento, conjuntos do tipo Mandelbrot, por exemplo, resistem ao processo de construção, pois são figuras cuja aplicação de transformações geram distorções nos padrões existentes. Assim, uma análise mais cuidadosa sobre o processo deve ser considerada para o caso de conjuntos do tipo Mandelbrot, ou mais geralmente, conjuntos do tipo Julia.

5.4.1 Curva de Koch

As Figuras 5.11-a,b e 5.12-a,b ilustram as rotações aplicadas a curva de Koch com centro de rotação à esquerda em $P(0,0)$, com ângulos definidos por meio da sequência $rotacao = \dfrac{\pi}{k \cdot l}, l = 1, \cdots, 2k$.

Figura 5.11: Curva de Koch com número de rotações definido por $rotacao = \dfrac{\pi}{l}, l = 1, \cdots, 2k$ e paleta de cores $p = "black"$.



(a) Rotações para $k = 3$.

(b) Rotações para $k = 5$.

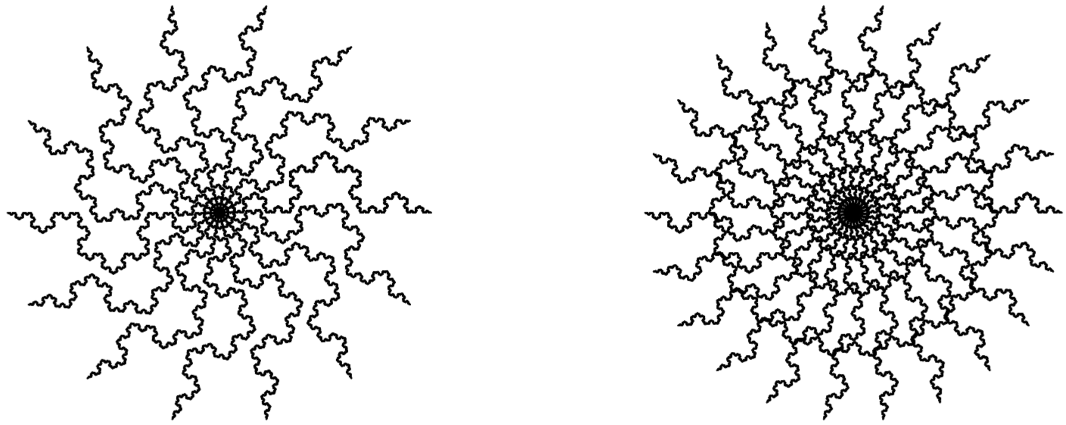
Fonte: O autor, 2024.

A paleta de cores "*black*" é mantida constante para evidenciar os elementos de rotações. Apesar disso, os resultados são, no mínimo, interessantes. Também

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

deve ser observada a formação de alguns padrões próximos aos pontos de rotação, especialmente no caso da Figura 5.12-b, devido à sobreposição das rotações.

Figura 5.12: Curva de Koch com número de rotações definido por $rotacao = \frac{\pi}{l}, l = 1, \cdots, 2k$ e paleta de cores $p = "black"$.

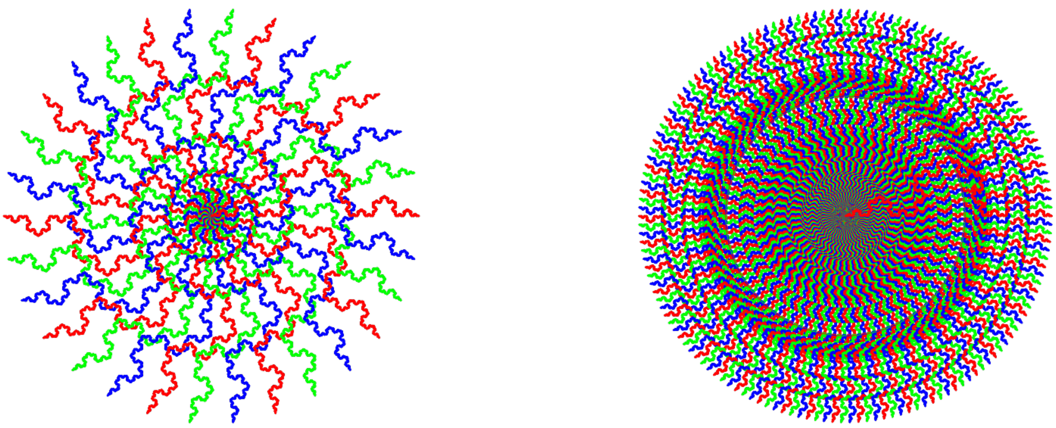


(a) Rotações para $k = 7$. (b) Rotações para $k = 11$.

Fonte: O autor, 2024.

As Figuras 5.13-a,b e 5.14-a,b ilustram rotações definidas por $k = 15, 75, 100, 101$.

Figura 5.13: Curva de Koch com número de rotações definido por $rotacao = \frac{\pi}{l}, l = 1, \cdots, 2k$ e paleta de cores $p = \{p_1, p_2\}$ para $k = 15, 75$.

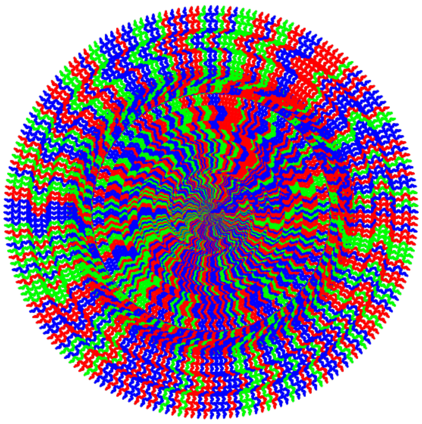


(a) Rotações para $k = 15$. (b) Rotações para $k = 75$.

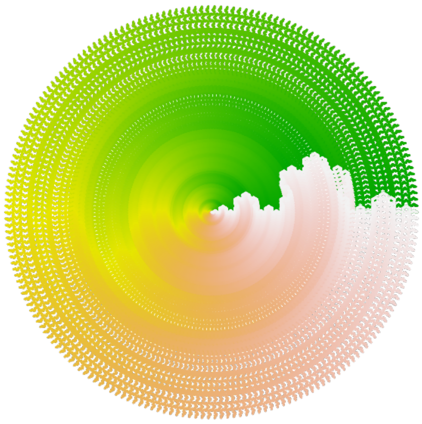
Fonte: O autor, 2024.

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

Figura 5.14: Curva de Koch com número de rotações definido por $rotacao = \frac{\pi}{l}, l = 1, \cdots, 2k$ e paleta de cores $p = \{p_1, p_2\}$ para $k = 100, 101$.



(a) Rotações para $k = 100$.



(b) Rotações para $k = 101$.

Fonte: O autor, 2024.

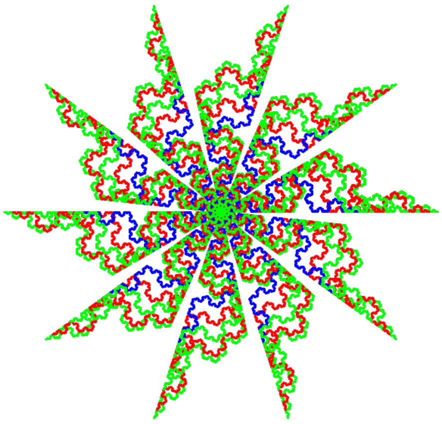
A paleta de cores utilizada pode ser reproduzida com base no código a seguir:

```
if (k==15){
cores=rep(c("red","green","blue"),length.out=length(rotacao))
}else if(k==75){cores=rep(c("red","green","blue"),length.out=length(
   rotacao))
}else if(k==100){cores=cores=sample(c("red","green","blue"),
size=length(rotacao),replace=T)[652:655],length.out=length(rotacao))
}else if(k==101){cores=terrain.colors(length(rotacao))}
```

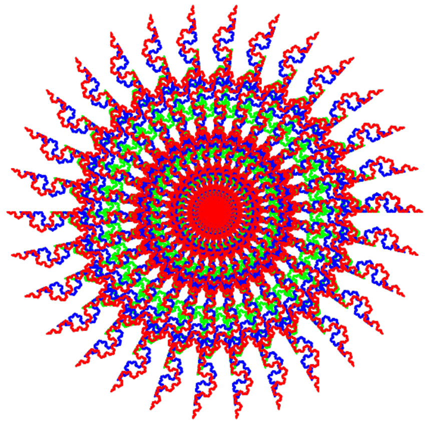
As Figuras 5.15-a,b,c,d ilustram rotações e homotetias com cores distintas para homotetias. O trecho de código a seguir define a paleta de cores utilizada.

```
if (k==5){set.seed(1000); contracao=runif(k,.5,1.0)
     cores=rep(c("red","green","blue"),length.out=length(contracao))
}else if(k==15){ contracao=c(.5, .75,.89, .95)#runif(5,.25,1.)
     cores=rep(c("red","green","blue"),length.out=length(contracao))
}else if(k==17){ contracao=runif(10,.5,1.0)
     cores=cores=sample(c("red","green","blue"),size=length(contracao),
        replace=T)
}else if(k==11){contracao=c(.5, .75,.89, .95)
     cores=terrain.colors(length(rotacao)) }
```

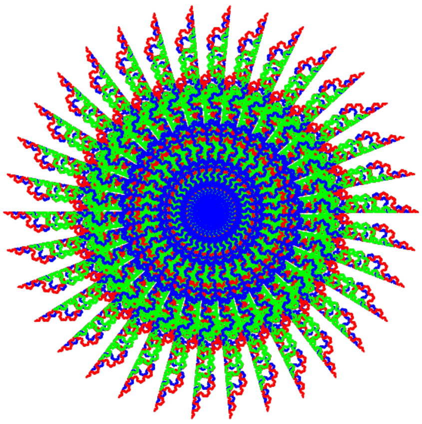
Figura 5.15: Curva de Koch com número de rotações definido por $rotacao = \dfrac{\pi}{l}, l = 1, \cdots, 2k$ e paleta de cores para as homotetias.



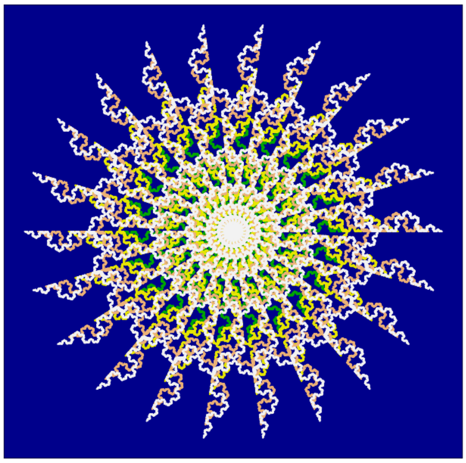
(a) Rotações para $k = 5$.



(b) Rotações para $k = 15$.



(c) Rotações para $k = 17$.



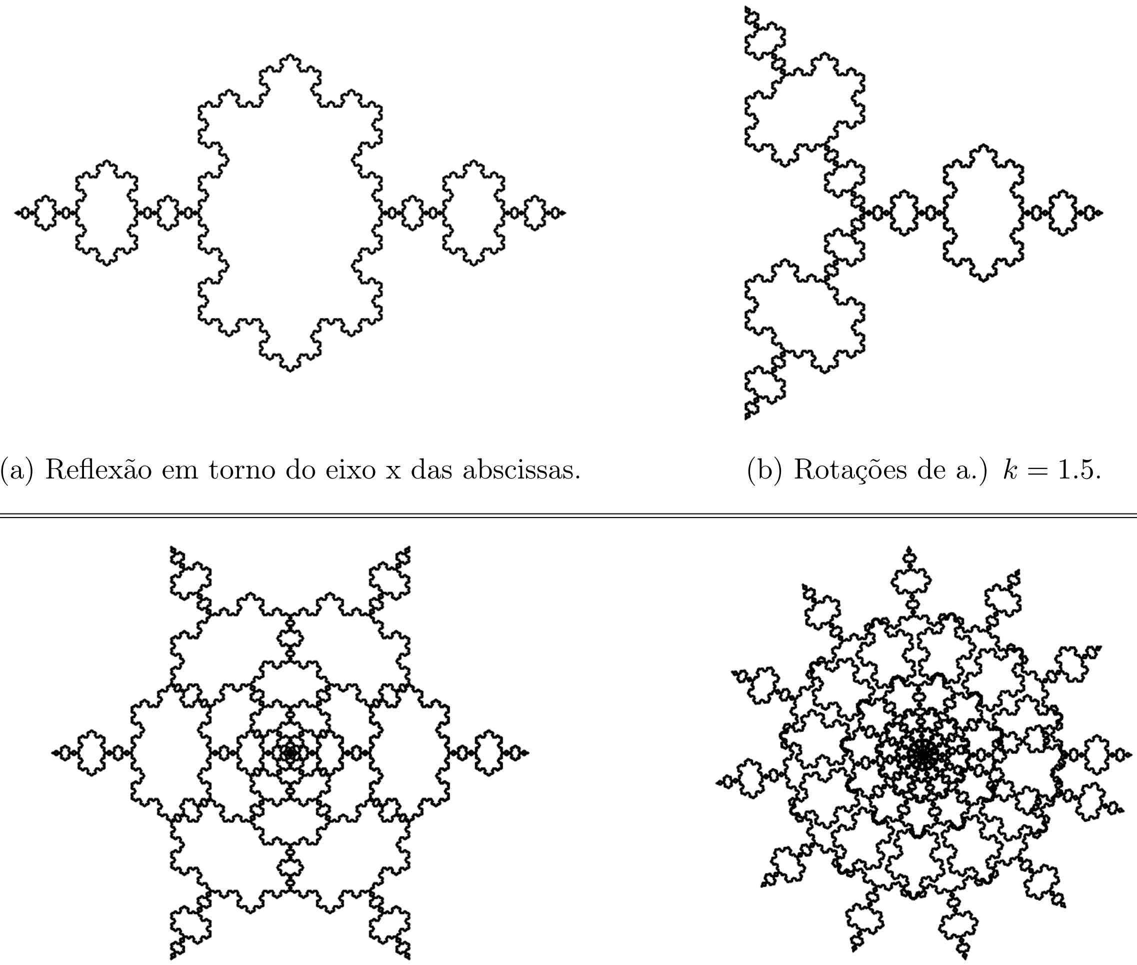
(d) Rotações para $k = 11$.

Fonte: O autor, 2024.

Uma variação imediata da curva de Koch é a reflexão em torno do eixo x das abscissas. As Figuras 5.16-a,b,c,d ilustram os resultados com cores idênticas para rotações e sem homotetias.

A Figura 5.16-b lembra, em aspecto, a curva fractal *Koch anti-snow flake.* A verificação requer uma análise mais pormenorizada, pois a curva *anti snow flake* é construída por meio de processo iterativo em direção ao interior do triângulo equilátero (na direção oposta da curva floco de neve de Koch). As demais figuras também apresentam outras estruturas que emergem ao final da composição.

Figura 5.16: Curva de Koch com reflexão e rotações número de rotações definido por $rotacao = \frac{\pi}{l}, l = 1, \cdots, 2k$.



(a) Reflexão em torno do eixo x das abscissas.

(b) Rotações de a.) $k = 1.5$.

(c) Rotações para $k = 3$.

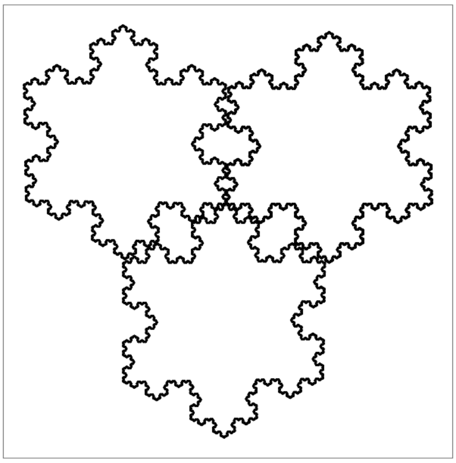
(d) Rotações para $k = 5.75$.

Fonte: O autor, 2024.

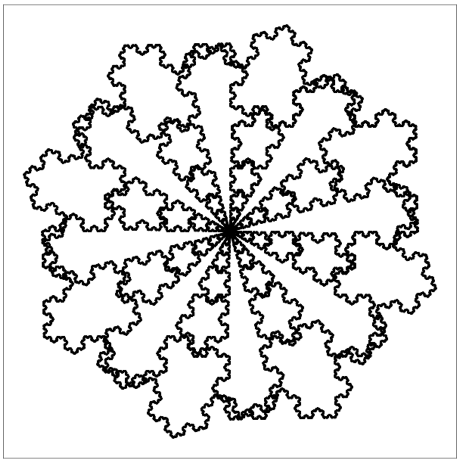
5.4.2 Curva Floco de Neve de Koch

A curva Floco de Neve de Koch, ilustrada na Figura 5.8, foi utilizada para gerar os resultados da Figura 5.17-a,b e 5.18-a,b. De forma geral, estas figuras apresentam padrões curiosos, os quais requerem uma análise mais pormenoriza-ada.

Figura 5.17: Curva de Floco de neve de Koch com número de rotações definido por $rotacao = \frac{\pi}{l}, l = 1, \cdots, 2k$ com $k = 1.5, 4$.



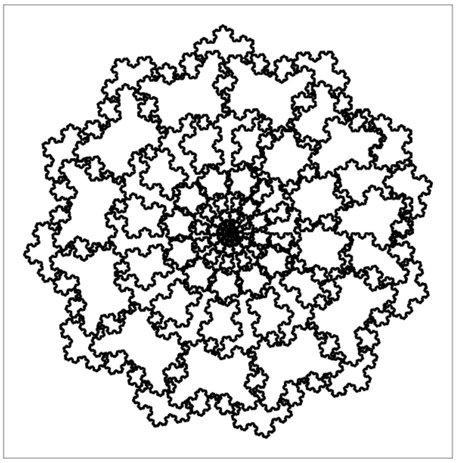
(a) Rotações para $k = 1.5$.



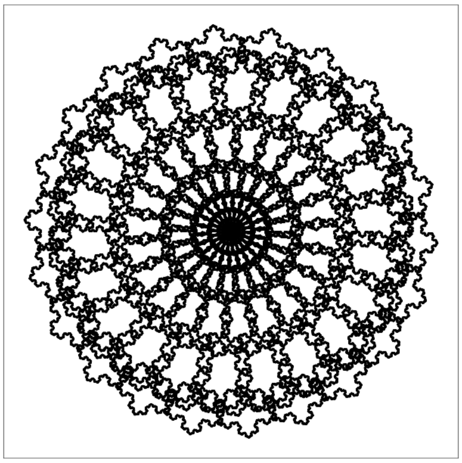
(b) Rotações para $k = 4$.

Fonte: O autor, 2024.

Figura 5.18: Curva de Floco de neve de Koch com número de rotações definido por $rotacao = \frac{\pi}{l}, l = 1, \cdots, 2k$ com $k = 5.75$.



(a) Rotações para $k = 5.75$.

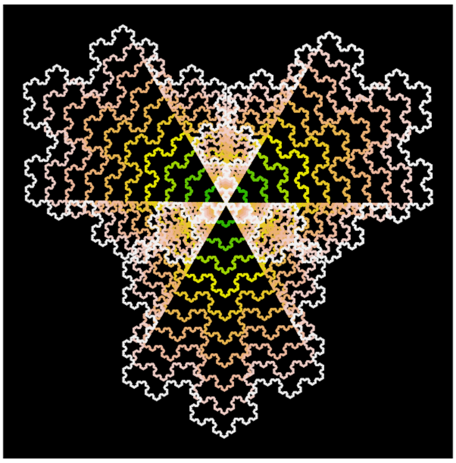


(b) Rotações para $k = 11$.

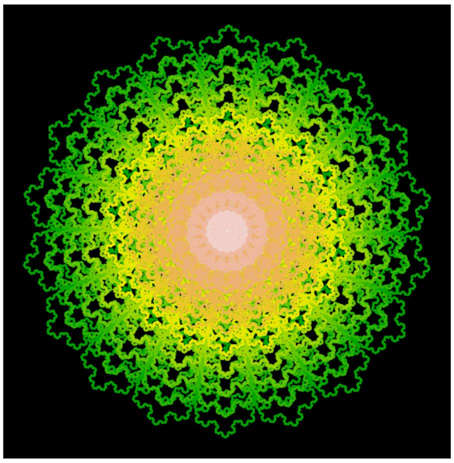
Fonte: O autor, 2024.

Analogamente aos resultados obtidos em (**livro01**), (**livro02**) e (**xornadas2023**), efeitos visuais interessantes surgem ao considerar a coloração combinada com as rotações e homotetias. A Figuras 5.19-a,b,c,d são ilustrativas.

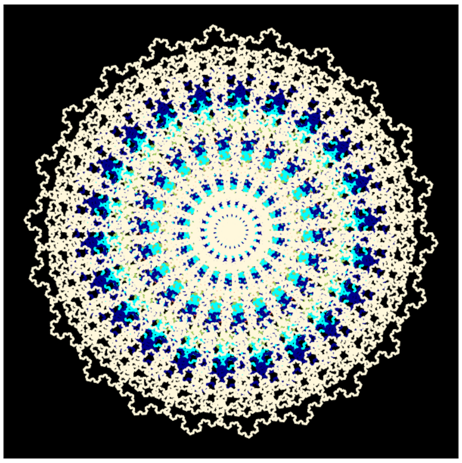
Figura 5.19: Curva de Floco de neve de Koch com número de rotações definido por $rotacao = \frac{\pi}{l}, l = 1, \cdots, 2k$ e homotetias $r_i = 0.1 \cdot i, i = 1, \cdots, 9$ para a), c) e d) e $-r_i$ para b).



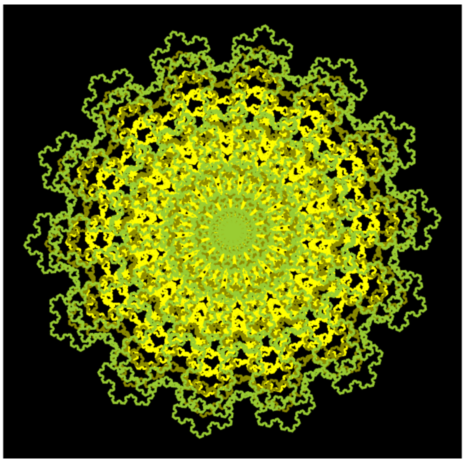
(a) Rotações com $k = 1.5$ e homotetias.



(b) Rotações com $k = 9$ e homotetias.



(c) Rotações com $k = 11$ e homotetias.



(d) Rotações com $k = 7$ e homotetias.

Fonte: O autor, 2024.

As rotações e homotetias utilizadas nas Figuras 5.19-a,b,c,d, bem como os esquemas de coloração, são descritos no código a seguir:

```
if (k==1.5){
    contracao=seq(0,1.,.1)
    cores=terrain.colors(length(contracao))
}else if (k==9){
    contracao=seq(1,0,-.1)
    cores=rep(colors()[550:557],length.out=length(contracao))
}else if(k==11){
    set.seed(10)
```

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

```
    contracao=seq(0,1.,.1)
    cores=sample(colors()[50:100],size=length(contracao),replace = T)
}else if(k==7){
    set.seed(10)
    contracao=seq(0,1.,.1)
    cores=sample(colors()[652:657],size=length(contracao),replace = T)}
```

5.4.3 Árvore Fractal

A árvore fractal com padrões das Figuras 5.9 foi utilizada nas construções das Mandalas de Árvore Fractal mostradas nas Figuras 5.20.

Em Figura 5.9, apenas os pontos finais dos galhos da árvore foram utilizados na construção (copa da árvore). Os códigos para a obtenção e paleta de cores são mostrados a seguir:

```
df=data.frame(fractal);  xt=c(df$X1,df$X3);   yt=c(df$X2,df$X4)
k=2.; contracao=c(seq(0,.25,.05)); rotacao=pi/k*(1:(2*k))
k=5; contracao=c(seq(0,1.,.1)); rotacao=pi/k*(1:(2*k))
k=10; contracao=c(seq(0,1.,.1)); rotacao=pi/k*(1:(2*k))
k=7; contracao=c(seq(0,1.,.05)); rotacao=pi/k*(1:(2*k))

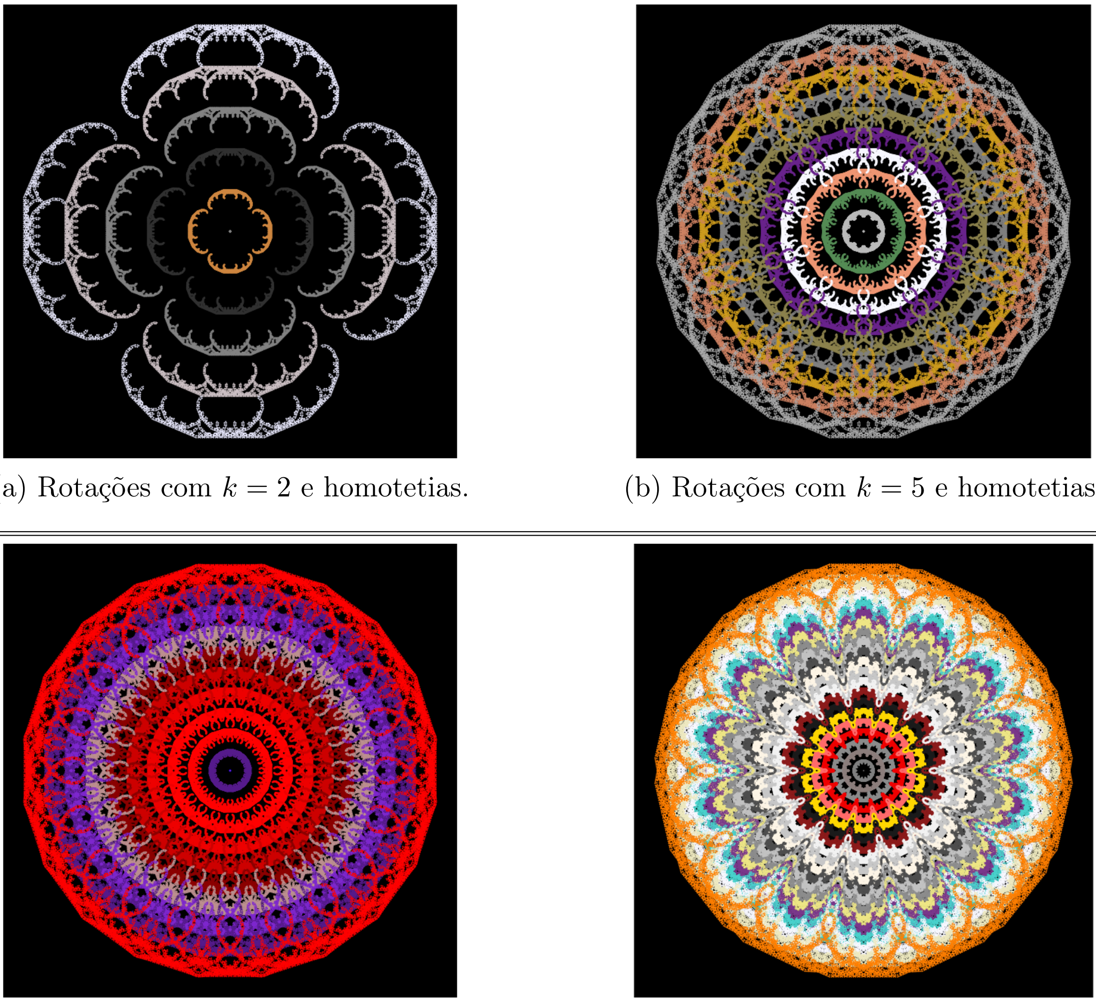
for(i in 1:length(rotacao)){
  xt=c(xt,xt[1:n]*cos(rotacao[i])-yt[1:n]*sin(rotacao[i]))
  yt=c(yt,xt[1:n]*sin(rotacao[i])+yt[1:n]*cos(rotacao[i]))}

p= ggplot()+ coord_fixed()+ theme_void()

if (k==2||k==5||k==7){cores=sample(colors()[50:557],size=length(
    contracao),replace = T)
}else{cores=rep(colors()[650:657],length.out=length(contracao))}

for(i in 1:length(contracao)){
  xt2=c(xt*contracao[i])
  yt2=c(yt*contracao[i])
  dt2=tibble::tibble(xt2,yt2)
  p=p+geom_point(data=dt2, aes(x=xt2, y=yt2),color=cores[i],size=0.015)}
p=p+theme(panel.background = element_rect(fill = "black" ))
p
```

Figura 5.20: Árvore fractal com número de rotações definido por $rotacao = \dfrac{\pi}{l}, l = 1, \cdots, 2k$.



(a) Rotações com $k = 2$ e homotetias.
(b) Rotações com $k = 5$ e homotetias.
(c) Rotações com $k = 10$ e homotetias.
(d) Rotações com $k = 7$ e homotetias.

Fonte: O autor, 2024.

A utilização da árvore fractal da Figura 5.10 produz resultados mostrados nas Figuras 5.21-a,b,c,d. Os códigos são similares aos códigos anteriores, exceto pelos números de rotações, homotetias e a paleta de cores utilizada em 5.10. Note que as duas primeiras figuras consideram apenas as rotações da figura original, enquanto as demais consideram homotetias distintas com padrões descritos a seguir.

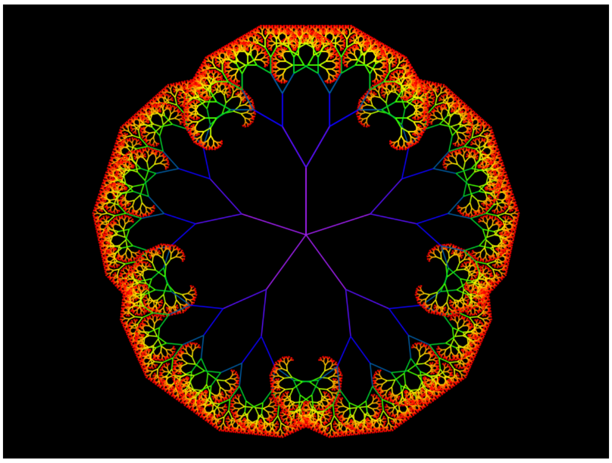
```
rotacao=c(pi/2,pi, 3*pi/2)
k=2.5
rotacao=pi/k*(1:(2*k)); contracao=c(1)
```

```
k=5
rotacao=pi/k*(1:(2*k)); contracao=c(1)

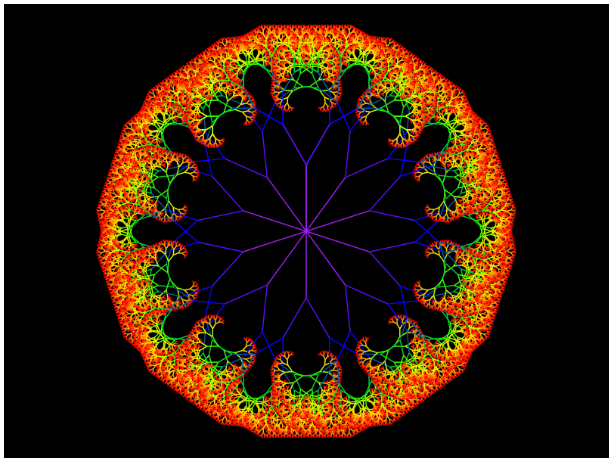
k=3
rotacao=pi/k*(1:(2*k)); contracao=c(seq(0,1,.25))

k=3.5
rotacao=pi/k*(1:(2*k));contracao=c(seq(0,.5,.125),1.1,1.2,1.5,2.0)
```

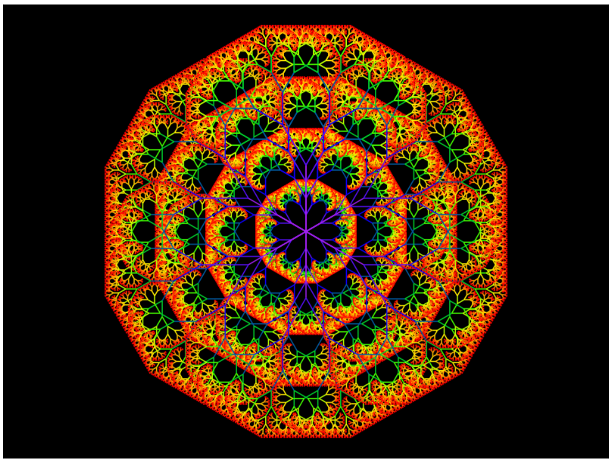
Figura 5.21: Árvore Fractal com número de rotações definido por $rotacao = \frac{\pi}{l}, l = 1, \cdots, 2k$ e homotetias .



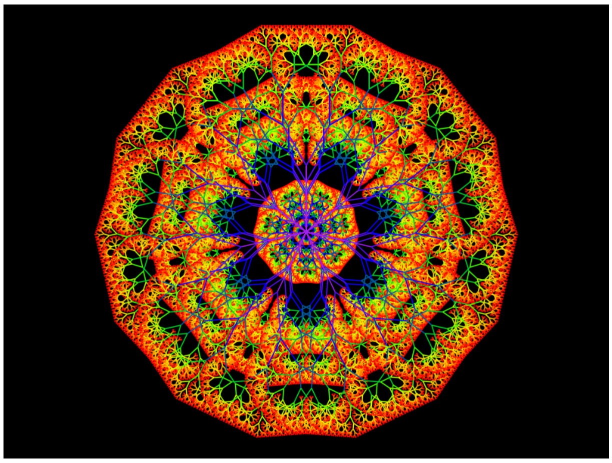
(a) Rotações com $k = 2.5$ e homotetias.



(b) Rotações com $k = 5$ e homotetias.



(c) Rotações com $k = 3$ e homotetias.



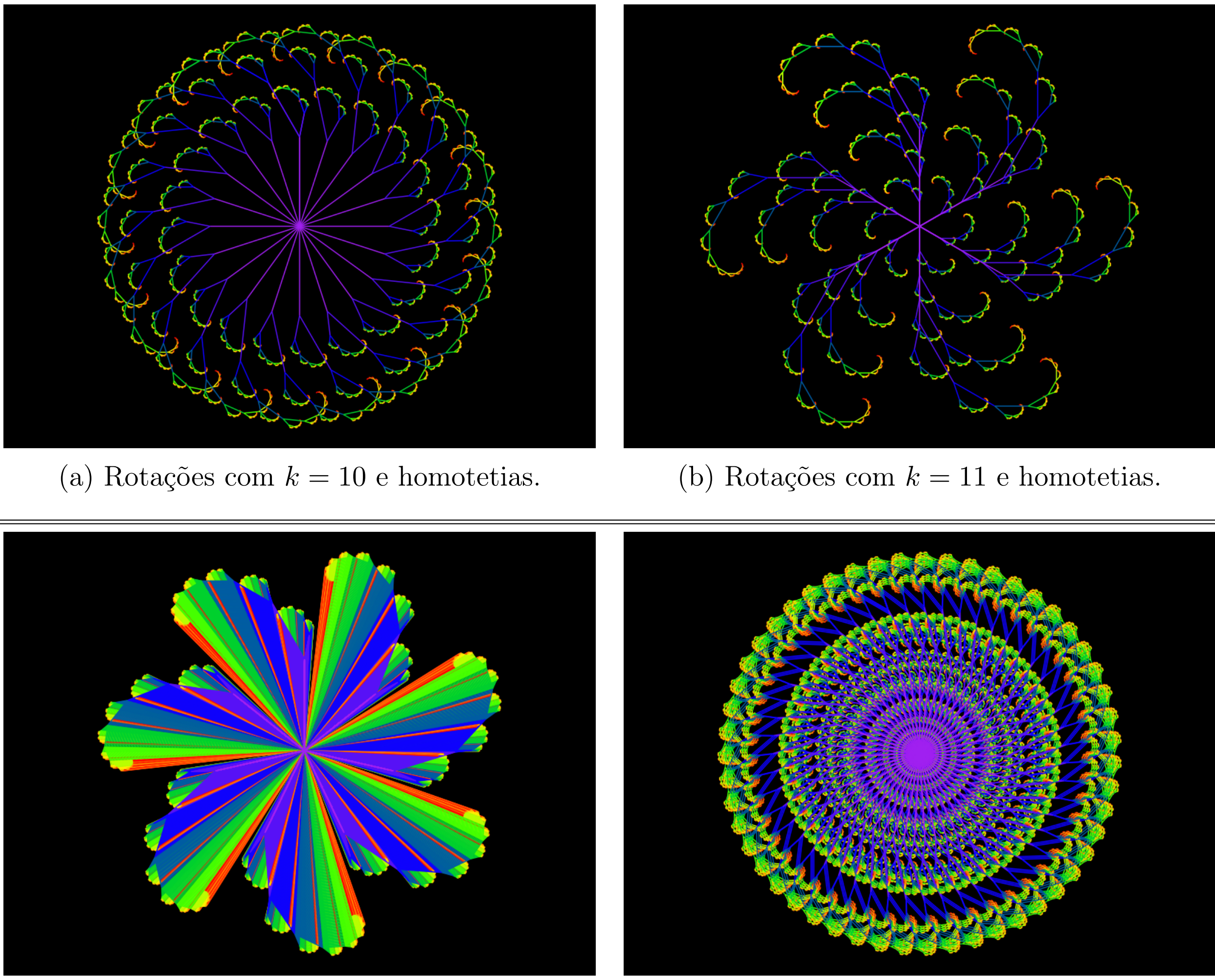
(d) Rotações com $k = 3.5$ e homotetias.

Fonte: O autor, 2024.

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

A seguir uma Mandala de Árvore Fractal com modificações da simetria dos ramos da árvore fractal 5.10. É interessante notar o efeito da composição das rotações e homotetias originando figuras com características peculiares.

Figura 5.22: Árvore Fractal assimétrica com número de rotações definido por $rotacao = \frac{\pi}{l}, l = 1, \cdots, 2k$ e homotetias .



(a) Rotações com $k = 10$ e homotetias.

(b) Rotações com $k = 11$ e homotetias.

(c) Rotações com $k = 25.5$ e homotetias.

(d) Rotações com $k = 13$ e homotetias.

Fonte: O autor, 2024.

```
rotacao=c(pi/2,pi, 3*pi/2)
k=10; rotacao=pi/k*(1:(2*k)); contracao=c(1)
k=3; rotacao=pi/k*(1:(2*k));  contracao=c(seq(0,1,.25))
k=25.5;rotacao=pi/k*(1:(2*k));contracao=c(seq(0,.5,.1),.75,.76,.77)
k=13; rotacao=pi/k*(1:(2*k)); contracao=c(.75,.76,.77,seq(.5,0,-.1))
k=3.5; rotacao=pi/k*(1:(2*k)); contracao=c(seq(0,1,.00500))
```

5.5 CONSIDERAÇÕES FINAIS

O processo de construção das mandalas é uma abordagem promissora e multifacetada para a geração de figuras com simetria radial. A complexidade e beleza das construções depende, em grande parte, da criatividade e não está restrita às curvas matemáticas empregadas. As composições e as colorações ainda permanecem quase inexploradas, pois apenas algumas das possibilidades de curvas matemáticas e métodos de coloração foram explorados. Ao considerar as curvas fractais uma nova perspectiva de possibilidades é inserida, pois a geometria fractal traz suas características peculiares, tais como auto-similaridade e estrutura fina.

De forma geral, os resultados alcançados apontam um potencial para exploração no ensino interdisciplinar envolvendo Matemática, Computação, Estatística e Arte. Também pode despertar o interesse de leitores de diferentes áreas, pois a construção das figuras, seja pelo aspecto computacional, pela matemática, pela estatística ou pelo viés artístico das construções, é um processo que pode ser dominado uma vez que os códigos completos são fornecidos ao leitor. Em termos matemáticos, o número de possibilidades aumentará significativamente à medida que novas curvas, novos modelos de cores, novos métodos de construção e coloração forem adicionados.

Capítulo 6

ChatGPT + R: Ampliando Horizontes em Análises de Dados

Autora: Ariane Hayana Thomé de Farias
Instituição: Tribunal de Justiça do Estado de Roraima (TJRR)
e-mail: ariane.hayana@gmail.com

6.1 INTRODUÇÃO

Nos últimos anos, o avanço tecnológico e a crescente demanda por informações rápidas têm ampliado os desafios na análise de dados, exigindo rapidez e consistência nos resultados. Nesse contexto, a automação de tarefas repetitivas e a geração de *insights* valiosos surgem como soluções estratégicas. A integração de ferramentas avançadas, especialmente aquelas baseadas em inteligência artificial (IA), tornou-se essencial para otimizar processos e aprimorar resultados, consolidando-se como uma prática comum em diversas operações diárias. A IA, em especial, desempenha um papel importante ao auxiliar desenvolvedores, estatísticos e outros profissionais na criação de *scripts* e modelos avançados, contribuindo para a análise de grandes volumes de dados, auxiliando na tomada de decisão e garantindo uma análise mais eficiente e precisa.

Atualmente, é possível integrar ferramentas de inteligência artificial a ambientes de desenvolvimento integrados (IDEs), como o *Visual Studio Code*, *PyCharm* e `RStudio`, que são amplamente utilizados por estatísticos e cientistas de dados. Esses IDEs permitem a integração de *Application Programming Interfaces* (APIs) de IA, como a do ChatGPT, diretamente em seus ambientes. Dessa forma, ao desenvolver um *script* em `R` para análise de dados, o usuário pode utilizar a API do ChatGPT para gerar trechos de código, resolver problemas específicos ou otimizar tarefas repetitivas, realizando consultas diretamente no ambiente de desenvolvimento.

Neste contexto, com o intuito de auxiliar programadores que desejam aprimorar suas análises de dados, este capítulo é voltado para aqueles que buscam formas de inovação em suas práticas analíticas, utilizando inteligência artificial para agregar valor aos seus resultados. Serão apresentadas etapas práticas sobre como integrar a API do ChatGPT ao `RStudio` e como explorar algumas funcionalidades do pacote `gptstudio`. O estudo abrange desde as etapas iniciais de configuração do `RStudio` até sua aplicação prática.

Este capítulo está estruturado da seguinte forma: inicialmente, são apresentados a introdução e o objetivo principal, descritos nas Seções 6.1 e 6.2, respectivamente. A Seção 6.3 detalha as etapas necessárias para a integração da API do ChatGPT e sua configuração ao `RStudio`. Na Seção 6.4, são expostos os resultados obtidos a partir de exemplos de aplicação da API do ChatGPT no ambiente do `RStudio`. A Seção 6.5 apresenta as considerações finais.

6.2 OBJETIVO

O objetivo principal deste capítulo é detalhar as etapas essenciais para integrar a API do ChatGPT ao IDE `RStudio`, com a exemplificação de aplicações práticas de utilização do pacote `gptstudio`.

6.3 APLICAÇÃO

Para utilizar a inteligência artificial do ChatGPT no `RStudio` por meio da API, é necessário seguir algumas etapas técnicas fundamentais:

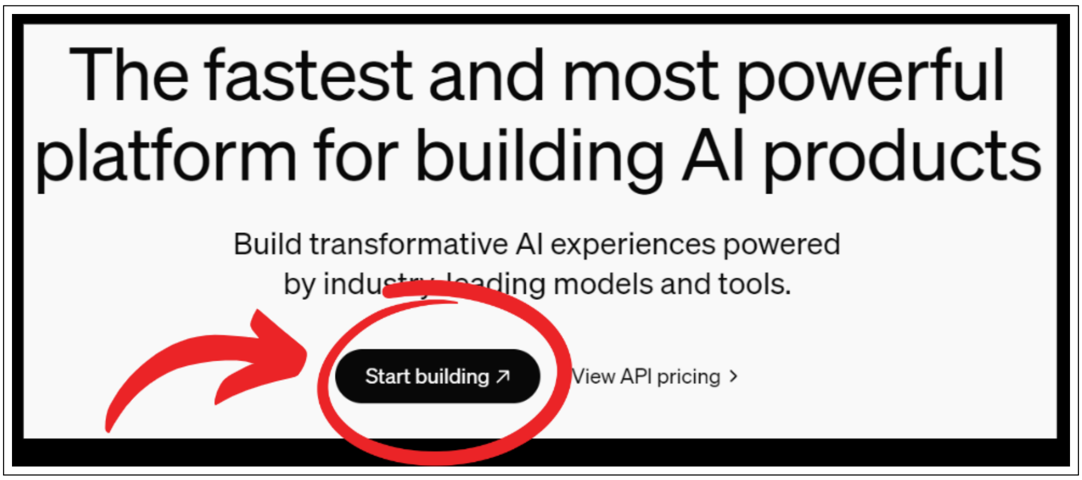
- Cadastrar-se no *site* da OpenAI;
- Criar uma chave de API;
- Configurar o `RStudio`;
- Ter o pacote `gptstudio`.

Cada etapa será detalhada nas subseções seguintes.

6.3.1 Configurações preliminares

Para a configuração da API do ChatGPT, o passo inicial consiste em realizar um cadastro no *site* da OpenAI. Para isso, basta acessar o *link* https://openai.com/api e clicar em *"Start building"*, conforme tela a seguir:

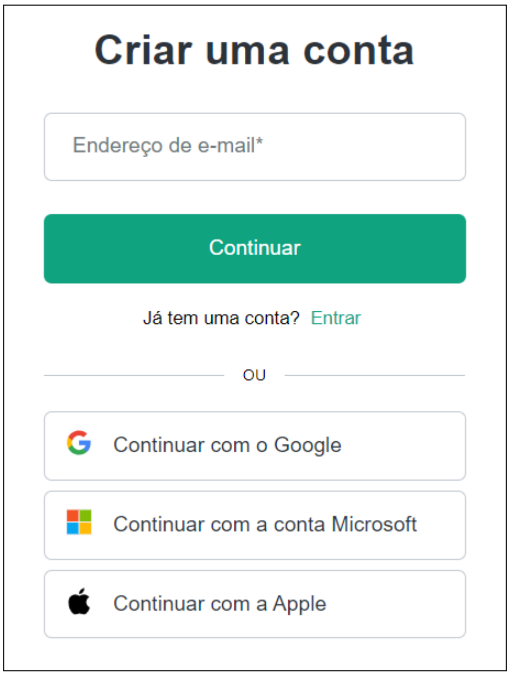
Figura 6.1: Tela de acesso na OpenAI.



Fonte: OpenAI, 2024.

Na sequência, uma nova janela aparecerá na tela e, no canto superior direito, haverá a opção de `"Log in"` e `"Sign up"`. Se o usuário já tem cadastro, basta clicar em `"Log in"` e acessar com seus dados cadastrados de *e-mail* e senha. Caso contrário, o cadastro pode ser realizado clicando em `"Sign up"`. A tela da Figura 6.2 será exibida, e o cadastro pode ser realizado com *e-mail* e contas do *Google*, *Microsoft* e *Apple*.

Figura 6.2: Tela de cadastro.

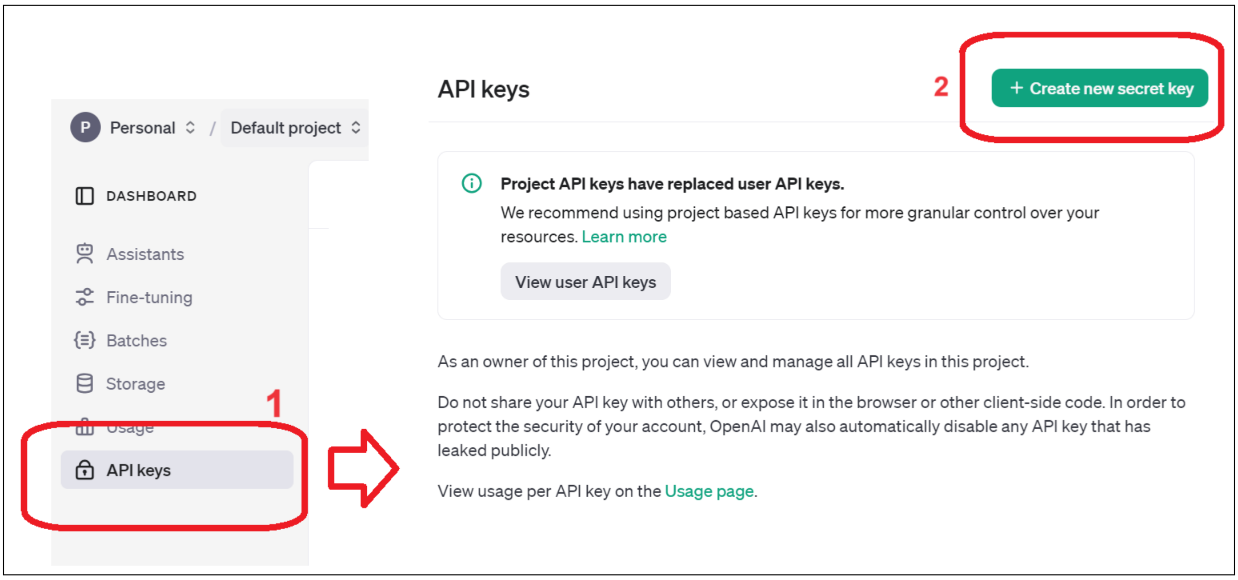


Fonte: OpenAI, 2024.

Com o cadastro efetivado, a próxima etapa consiste em obter a chave de acesso para utilizar o modelo do ChatGPT. Assim, após efetuar o `"Log in"`, em `"Dashboard"`, o usuário deve acessar a opção `"API keys"` (na barra lateral esquerda, conforme (1) na Figura 6.3). A seguir, o usuário deve clicar em `"+ Create new secret key"` (em (2) na Figura 6.3).

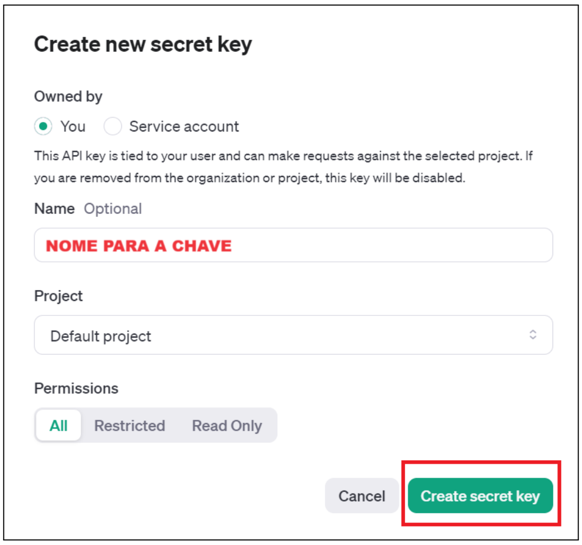
Após clicar em `"+ Create new secret key"`, aparecerá na tela uma janela para que o usuário insira as configurações que deseja para utilizar a chave da API (tais como o nome para identificação da chave de acesso, permissões e outros, conforme Figura 6.4).

Figura 6.3: Orientações para obter a chave de acesso.



Fonte: OpenAI, 2024.

Figura 6.4: Configurando chave de acesso.

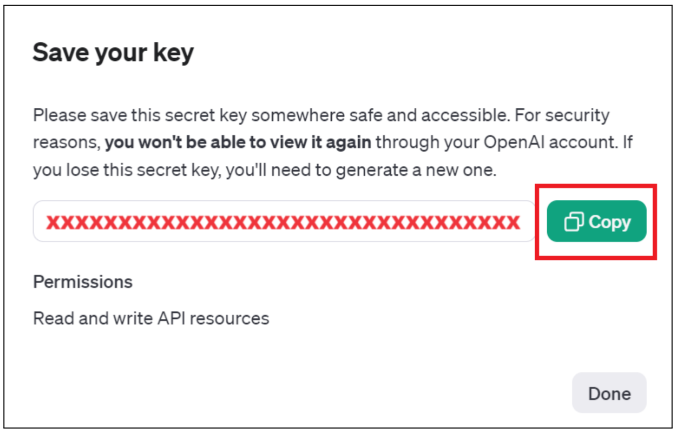


Fonte: OpenAI, 2024.

Após a criação da chave, um código aparecerá na tela (em vermelho, na Figura 6.5). Este deve ser copiado para que o usuário possa configurar o `RStudio` na sequência.

Com isso, as configurações preliminares estarão concluídas. Parte-se, então, para a próxima etapa, que consiste em integrar a chave de acesso da API

Figura 6.5: Gerando chave de acesso.



Fonte: OpenAI, 2024.

ao RStudio. Neste momento é importante destacar que, para utilizar a API do ChatGPT, é necessário contratar um plano pago específico de acordo com o objetivo pretendido em seus projetos. A OpenAI dispõe de modelos avançados, tais como GPT-4o, modelos de imagem, áudio, entre outros, e os preços podem variar conforme número de *tokens* processados. Para mais detalhes, os preços podem ser consultados no *site* em https://openai.com/api/pricing/.

6.3.2 Integrando API e RStudio

Esta etapa consiste em conectar o ChatGPT ao RStudio. Para isso, é necessário abrir o RStudio e instalar o pacote usethis:

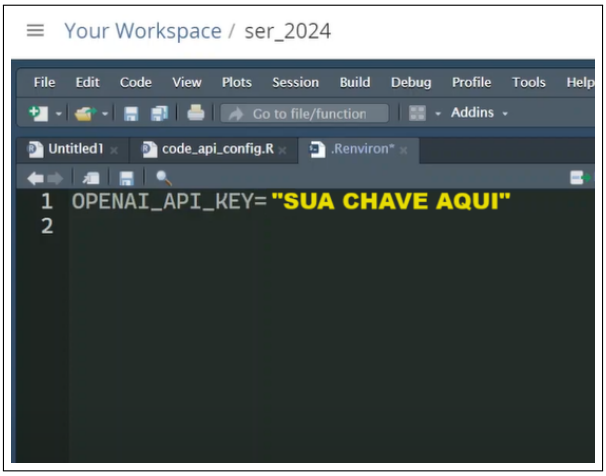
```
# Pacotes para instalar
install.packages("usethis")
```

Após a instalação, no *console* deve ser inserido o código a seguir e, posteriormente, apertar a tecla *enter*:

```
usethis::edit_r_environ()
```

Um arquivo .Renviron abrirá na tela. É neste arquivo que o usuário vai incluir a chave de acesso da API do ChatGPT. Tal chave deve ser inserida entre aspas, conforme a Figura 6.6.

Figura 6.6: Integrando API no `RStudio`.



Fonte: A autora, 2024.

O código será:

```
OPENAI_API_KEY="SUA CHAVE AQUI"
```

Após inserir a chave, basta salvar as alterações realizadas e reiniciar o `RStudio` com o atalho `CTRL + SHIFT + F10`.

Na sequência, precisamos carregar a chave de API do arquivo `.Renviron` com o código:

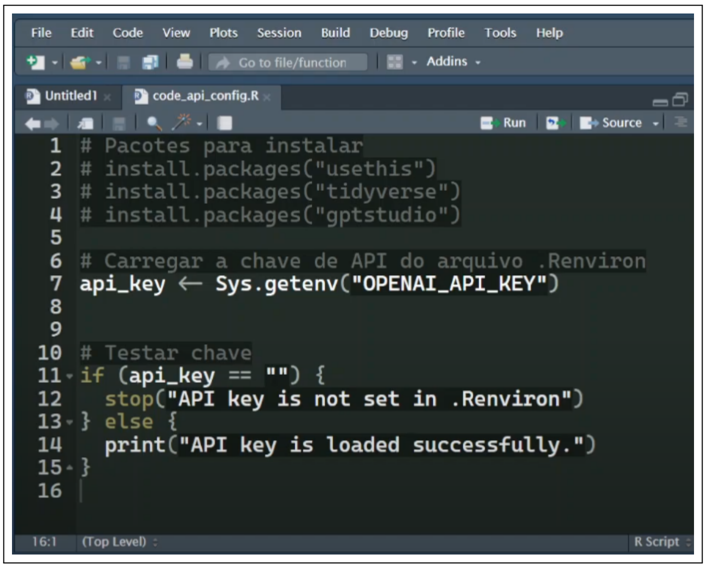
```
api_key <- Sys.getenv("OPENAI_API_KEY")
```

Para validar se a integração foi bem-sucedida, o código a seguir servirá de auxílio para tal validação. Se a conexão foi carregada com sucesso, aparecerá a mensagem "*A chave da API foi carregada com sucesso*". A Figura 6.7 mostra a estrutura do código.

```
if (api_key == "") {
  stop("A chave da API não está definida em .Renviron")
} else {
  print("A chave da API foi carregada com sucesso.")
}
```

Com as conexões realizadas com sucesso, parte-se para a etapa seguinte, que consiste em explorar as funcionalidades do ChatGPT com a utilização do pacote `gptstudio`. Uma observação importante que cabe salientar diz respeito à utilização de sistemas de controle de versão. Para evitar a exposição de sua chave de API, o arquivo `.Renviron` **deve** estar contido no arquivo `.gitignore`.

Figura 6.7: Código completo.



Fonte: A autora, 2024.

6.3.3 O pacote `gptstudio`

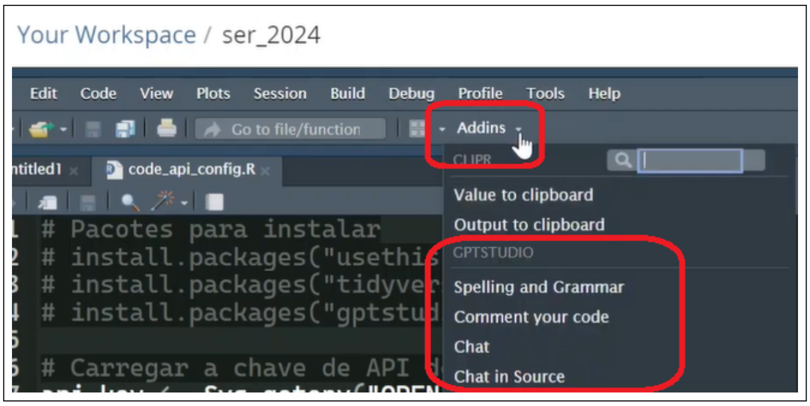
Segundo os autores do pacote (**gptstudio**), o objetivo do `gptstudio` é auxiliar os programadores `R` a incorporarem facilmente o uso de grandes modelos de linguagem (LLMs) em seus fluxos de trabalho de projeto. O pacote é configurado para trabalhar com diversos provedores de serviços de IA, como, por exemplo, OpenAI, HuggingFace, Google AI Studio, entre outros, permitindo flexibilidade de escolha ao usuário. Assim, para utilização no `RStudio`, o passo inicial será instalar os seguintes pacotes que auxiliarão também nas próximas etapas:

```
# Pacotes para instalar
install.packages("gptstudio")
install.packages("tidyverse")
```

Sugere-se reiniciar o `RStudio` após a instalação. A seguir, para verificar se a instalação foi bem-sucedida, em `Addins`, conferir se aparece o `gptstudio`, conforme apresenta a Figura 6.8.

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

Figura 6.8: `Addins` com `gptstudio`.



Fonte: A autora, 2024.

Após identificar que a instalação foi bem-sucedida, o `gptstudio` está pronto para ser utilizado e explorado. Na seção seguinte serão abordadas algumas aplicações.

6.4 RESULTADOS E DISCUSSÃO

Com a finalização das configurações necessárias, o `RStudio` está devidamente integrado à API do ChatGPT. Algumas aplicações são extremamente úteis para a otimização de tempo e produtividade de desenvolvedores e estatísticos.

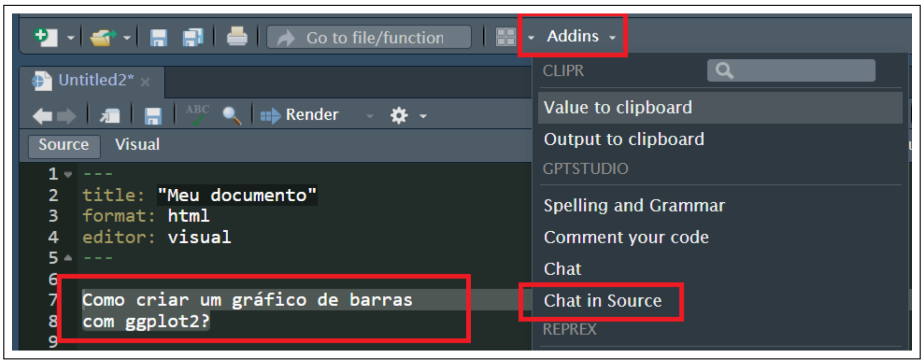
Ao longo do desenvolvimento de *scripts*, em diversos momentos, profissionais deparam com dúvidas sobre algum trecho de código, que podem ser rapidamente sanadas com auxílio de ferramentas de IA. No `RStudio`, por exemplo, é possível criar um documento Quarto `(.qmd)` e, com apoio da API do ChatGPT, fazer perguntas tais "*Como criar um gráfico de barras com `ggplot2`*"?

Para encontrar a resposta, basta escrever a dúvida, selecionar a frase e em `Addins` clicar em `"Chat in Source"`, como na Figura 6.9.

Como resposta, o resultado sugerido pelo ChatGPT foi:

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

Figura 6.9: Utilizando o ChatGPT com `gptstudio`.



Fonte: A autora, 2024.

Como criar um gráfico de barras com ggplot2?

```
```r
# Primeiro, você precisa instalar e carregar o pacote ggplot2.
# Para instalar (caso ainda não tenha feito isso), use:
# install.packages("ggplot2")
library(ggplot2)# Carregando o pacote ggplot2
# Exemplo de dados que serão usados para criar o gráfico de barras
dados <- data.frame(
  Categoria = c("A", "B", "C", "D"),
  Valores = c(23, 45, 12, 67))

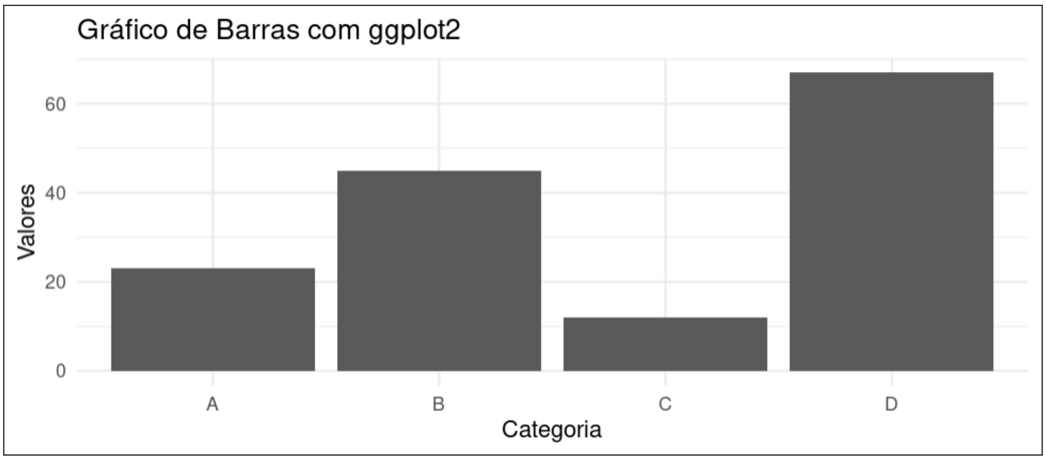
# Criando um gráfico de barras
ggplot(dados, aes(x=Categoria, y=Valores)) +
  geom_bar(stat="identity") +
  theme_minimal() +
  labs( title = "Gráfico de Barras com ggplot2",
    x = "Categoria", y = "Valores" )
```
```

Este trecho de código R cria um gráfico de barras simples usando o pacote ‘ggplot2‘, utilizando dados de exemplo. Para executá-lo, copie e cole o código em um editor de script R ou RStudio e execute.

Quanto ao gráfico sugerido pela IA, o resultado apresentado foi gerado, como mostra a Figura 6.10:

Além disso, podemos, ainda, comentar códigos com o auxílio da IA. Para

Figura 6.10: Sugestão dada pela IA.



Fonte: A autora, 2024.

inserir comentários em seu código, basta digitar o *prompt* desejado e selecionar o código em que deseja inserir comentários. Veja o exemplo a seguir, em que a primeira linha dá instruções sobre o que é solicitado:

Comente esse código em português (brasileiro)

```
library(plotly)

categorias <- c('Categoria A', 'Categoria B',
'Categoria C', 'Categoria D')

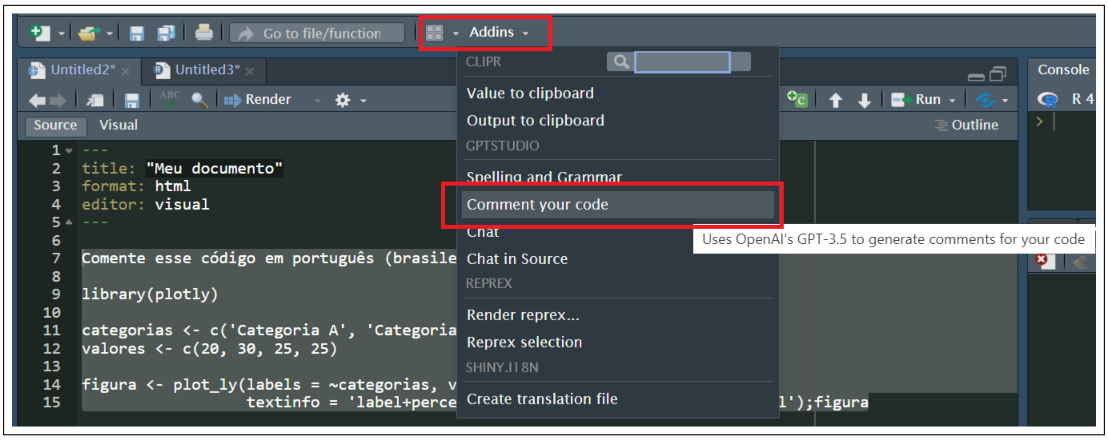
valores <- c(20, 30, 25, 25)

figura <- plot_ly(labels = ~categorias, values = ~valores,
type = 'pie', textinfo = 'label+percent',
insidetextorientation = 'radial');figura
```

Na sequência, clicar em `Addins` e depois em `Comment your code`, como na Figura 6.11:

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

Figura 6.11: Comentários em códigos com auxílio da IA.



Fonte: A autora, 2024.

O resultado da consulta foi:

```
# Carrega o pacote plotly para criar gráficos interativos
library(plotly)
# Define as categorias (ou rótulos) que aparecerão no gráfico de pizza
categorias <- c('Categoria A', 'Categoria B', 'Categoria C', 'Categoria D')
# Define os valores correspondentes a cada categoria no gráfico de pizza
valores <- c(20, 30, 25, 25)
# Cria uma figura usando o plotly, especificando:
# - labels: os rótulos das categorias
# - values: os valores de cada categoria
# - type: o tipo de gráfico, que é 'pie' (gráfico de pizza)
# - textinfo: a informação de texto para mostrar em cada fatia do gráfico de
    pizza ('label+percent' mostra o rótulo e a porcentagem)
# - insidetextorientation: a orientação do texto dentro das fatias ('radial'
    faz o texto seguir a curvatura do gráfico)
figura <- plot_ly(labels = ~categorias, values = ~valores, type = 'pie',
   textinfo = 'label+percent', insidetextorientation = 'radial')
# Exibe a figura criada
figura
```

6.5 CONCLUSÃO

Este capítulo abordou as etapas essenciais para a integração da API do ChatGPT ao `RStudio`. A aplicação da inteligência artificial para otimizar o tempo e a produtividade está em notável crescimento, o que beneficia não apenas desenvolvedores, mas também a sociedade como um todo. Os exemplos apresentados neste capítulo representam apenas uma fração das inúmeras possibilidades de utilização da IA no ambiente de desenvolvimento, destacando seu potencial para transformar processos e contribuir para a inovação. Ademais, os códigos apresentados neste capítulo estão consolidados em repositório Github, acessível no endereço https://github.com/a-hayana/ser2024. O passo a passo da apresentação está disponível em vídeo no YouTube do Seminário Internacional de Estatística com R (SER).

A Inteligência Artificial nas Ciências de Dados. Portal de Livros Abertos da USP, Pirassununga: Faculdade de Zootecnia e Engenharia de Alimentos, 2024. 190 p. ISBN 978-65-87023-46-5(e-book). Disponível em: https://doi.org/10.11606/9786587023465.

ISBN 978-65-87023-46-5 (e-book)