Sociedade Brasileira de Computação (SBC) - CNPJ: 29.532.264/0001-78

SBSI 2024

Juiz de Fora - MG

XX Simpósio Brasileiro de Sistemas de Informação

Perspectivas e Tendências dos Sistemas de Informação na Agricultura Digital

TÓPICOS ESPECIAIS EM SISTEMAS DE INFORMAÇÃO

Minicursos SBSI 2024

Organizadores:

Daniela Barreiro Claro (UFBA)
Fernanda Araujo Baião (PUC-RIO)
Ronney Moreira de Castro (UFJF)
José Maria David (UFJF)

ufjf Universidade Federal de Juiz de Fora
SBC Sociedade Brasileira de Computação

Simpósio Brasileiro de Sistemas de Informação
sbsi2024.ufjf.br @sbsibrasil

Organizadores
Daniela Barreiro Claro
Fernanda Araujo Baião
Ronney Moreira de Castro
José Maria David

TÓPICOS ESPECIAIS EM SISTEMAS DE INFORMAÇÃO
Minicursos do SBSI 2024

https://sbsi2024.ufjf.br
https://sol.sbc.org.br/livros/index.php/sbc/catalog/category/si

Sociedade Brasileira da Computação
Porto Alegre
2024

TÓPICOS ESPECIAIS EM SISTEMAS DE INFORMAÇÃO
Minicursos SBSI 2024

Sociedade Brasileira de Computação –SBC
CNPJ: 29.532.264/0001-78

ISBN: 978-85-7669-580-6

Coordenação Geral
Ronney Moreira de Castro (UFJF)
José Maria David (UFJF)

Coordenação do Comitê de Programa - Minicursos
Daniela Barreiro Claro (UFBA)
Fernanda Araujo Baião (PUC-RIO)

Edição dos Anais
Williamson Alison Freitas Silva (UNIPAMPA)

Editora
Sociedade Brasileira de Computação – SBC

Realização

Organização

Patrocinadores

Editores

Daniela Barreiro Claro (UFBA)
Fernanda Araujo Baião (PUC-RIO)
Williamson Alison Freitas Silva (UNIPAMPA)
Johnny C. Marques (ITA)
Tadeu Moreira de Classe (UNIRIO)
Victor Stroele (UFJF)
Ronney Moreira de Castro (UFJF)
José Maria David (UFJF)

Comitê técnico

Coordenação Geral
Ronney Moreira de Castro (UFJF)
José Maria David (UFJF)

Coordenação de Programa
Johnny C. Marques (ITA)

Coordenação da Trilha de Pesquisa em Sistemas de Informação
Tadeu Moreira de Classe (UNIRIO)
Victor Stroele (UFJF)

Coordenação dos Anais
Williamson Alison Freitas Silva (UNIPAMPA)

Coordenação do Comitê - Minicursos
Daniela Barreiro Claro (UFBA)
Fernanda Araujo Baião (PUC-RIO)

Membros da Comissão Especial de Sistemas de Informação da SBC (CESI)

Coordenador Geral

Valdemar Vicente Graciano Neto (UFG)

Vice-coordenador

Rodrigo Pereira dos Santos (UNIRIO)

Comitê Gestor da CESI

Allysson Allex de Paula Araújo (UFCA)
Andrea Magalhães Magdaleno (dheka)
Célia Ghedini Ralha (UnB)
Johnny Cardoso Marques (ITA)
Marcílio Ferreira de Souza Júnior (UFRPE)
Marcos Kalinowski (PUC-Rio)
Maria Claudia Figueiredo Pereira Emer (UTFPR)
Mônica Ximenes Carneiro da Cunha (IFAL)
Rodrigo Pereira dos Santos (UNIRIO)
Sean Wolfgand Matsui Siqueira (UNIRIO)
Vera Maria Benjamim Werneck (UERJ)
Valdemar Vicente Graciano Neto (UFG)
Williamson Alison Freita Silva (UNIPAMPA)

Comitê Diretivo do SBSI (2023-2024)

Mônica Ximenes Carneiro da Cunha (IFAL) - Coordenadora do CD-SBSI
Williamson Silva (UNIPAMPA) - Vice-Coordenador do CD-SBSI
Rafael D. Araújo (UFU)
Johnny Cardoso Marques (ITA)
Ronney Moreira de Castro (UFJF)
José Maria David (UFJF)
Tadeu Moreira de Classe (UNIRIO)
Victor Stroele (UFJF)

Comitê de Programa – Minicursos SBSI 2024

Alessandro Cerqueira (UniLaSalle-RJ / FAETERJ)
Andre Martinotto (Universidade de Caxias do Sul)
Carla Merkle Westphall (UFSC)
Carlos Santos Pires (Federal University of Campina Grande)
Daniel Notari (UCS)
Daniela Barreiro Claro (Federal University of Bahia)
Emanuel Coutinho (UFC)
Fernanda Baião (PUC-Rio)
Flávio Soares Corrêa da Silva (Universidade de São Paulo)
Glauco Carneiro (Universidade Federal de Sergipe)
Heitor Costa (Federal University of Lavras)
José Maria David (Universidade Federal de Juiz de Fora)
Maria Istela Cagnin (UFMS)
Morganna Diniz (UNIRIO)
Paulo Sérgio Santos (Federal University of the State of Rio de Janeiro)
Renata Araujo (Universidade Presbiteriana Mackenzie)
Ricardo Choren (IME / RJ)
Valdemar Vicente Graciano Neto (Universidade Federal de Goiás)

Dados Internacionais de Catalogação na Publicação (CIP)

S612 Simpósio Brasileiro de Sistemas de Informação (20. : 20 – 23 maio 2024 : Minas Gerais)

Minicursos do SBSI 2024 [recurso eletrônico] / organização: Daniela Barreiro Claro ... [et al.]. – Dados eletrônicos. – Porto Alegre: Sociedade Brasileira de Computação, 2024.

110 p. : il. : PDF ; 12 MB

Modo de acesso: World Wide Web.
ISBN 978-85-7669-580-6 (e-book)

1. Computação – Brasil – Evento. 2. Sistemas de informação. 3. Abordagens práticas. I. Claro, Daniela Barreiro. II. Baião, Fernanda Araujo. III. Castro, Ronney Moreira de. IV. David, José Maria. V. Sociedade Brasileira de Computação. VI. Título.

CDU 004(063)

Ficha catalográfica elaborada por Annie Casali – CRB-10/2339
Biblioteca Digital da SBC – SBC OpenLib

Prefácio Minicursos SBSI 2024

Este livro reúne trabalhos apresentados nos minicursos ministrados no XX Simpósio Brasileiro de Sistemas de Informação (SBSI 2024), sob a organização do Departamento de Ciência da Computação da Universidade Federal de Juiz de Fora (UFJF), no período de 20 a 23 de maio de 2024. Participam do SBSI, fórum nacional de debates da área de Sistemas de Informação (SI), estudantes e pesquisadores com apresentação de trabalhos científicos e discussão de temas contemporâneos relacionados à área. Esta edição foi realizada de forma presencial. O SBSI 2024 teve como temática principal "**Perspectivas e Tendências dos Sistemas de Informação na Agricultura Digital"**, voltando o olhar para o agronegócio brasileiro, que tem buscado ferramentas tecnológicas para otimizar processos, reduzir custos e aumentar a produtividade através da agricultura digital e iniciativas de transformação digital no campo, representando uma área em ascensão e que permite muitas oportunidades de pesquisas e para o mercado de trabalho.

Neste ano, foram submetidas 05 (cinco) propostas de minicursos válidas e 02 (duas) foram selecionadas, tal qual decidido em reunião da Comissão Especial de Sistemas de Informação. Tais propostas foram avaliadas por, no mínimo, três professores doutores que fazem parte do comitê científico da trilha de Minicursos do SBSI 2024.

Os dois minicursos contidos neste livro abordam tópicos de interesse da comunidade de Sistemas de Informação. O primeiro capítulo, intitulado **"Séries Temporais: Uma abordagem prática em Python**," apresenta os principais conceitos relacionados ao uso de modelos auto-regressivos e às técnicas de Aprendizado de Máquina, incluindo exemplos e aplicações com dados de diversos setores como Finanças, Saúde e Agronegócio. Já o segundo e último capítulo, intitulado **"UX e Linguagem Simples na Web: Práticas para um Design de Interação mais Compreensível"** apresenta as principais práticas da Linguagem Simples voltada para melhoria do entendimento de elementos textuais e visuais em interfaces, websites, aplicativos e outros produtos interativos

Esperamos que este livro auxilie estudantes, pesquisadores e profissionais da área de Sistemas de Informação na construção do conhecimento em temas específicos relacionados ao que foi aqui apresentado e sobre um novo assunto vinculado à sua área de atuação e também de extrair elementos para serem aplicados em sua pesquisa e/ou prática.

Daniela Claro (UFBA)
Fernanda Baião (PUC-Rio)
Coordenadoras da Trilha de Minicursos do SBSI 2024
Juiz de Fora/MG, Maio de 2024

Coordenadoras dos Minicursos do SBSI 2024

Fernanda Baião é Professora Associada do Departamento de Engenharia Industrial e coordenadora do Programa de Pós-Graduação em Engenharia de Produção da PUC-Rio. É Doutora (2001) e Mestre (1997) em Engenharia de Sistemas e Computação pela COPPE/UFRJ. Pesquisa sobre suporte à tomada de decisões através de abordagens quantitativas e orientadas por dados, em especial em cenários intensivos em conhecimento. Atua nos temas de Ciência de Dados, Modelagem Conceitual e Ontologias, Gestão de Processos de Negócio, Economia Comportamental e Alinhamento de Ontologias. É Cientista do Nosso Estado da FAPERJ e Bolsista de Produtividade do CNPq. Faz parte do conselho deliberativo da diretoria da ANPEPRO, é associada à SOBRAPO e à IAOA. Tem vasta experiência nacional e internacional em projetos de pesquisa, desenvolvimento e inovação em Engenharia e Ciência de Dados, BPM, e Arquitetura Empresarial, em domínios de Energia, Gestão de Serviços de TI, Predição de Fraudes, Gestão de Saúde Pública, que incluíram financiamento da FAPERJ, CNPq, Stone, Fundação Bill e Melinda Gates, e União Europeia.

Daniela Barreiro Claro é professora Associada da Universidade Federal da Bahia, fez seu Mestrado em Ciências da Computação pela Universidade Federal de Santa Catarina (2000) e o seu Doutorado em Ciência da Computação pela Université d'Angers na França (2006). Em 2009, ela fundou o FORMAS - Centro de Pesquisa em Dados e Linguagem Natural. Ela tem atuado na área de Interoperabilidade de Dados Semânticos e Pragmáticos e no Processamento de Linguagem Natural, particularmente na Extração de Informação Aberta e Multimodal em Português e modelos gerativos de língua. Atualmente, Daniela é coordenadora do Programa de Pós-Graduação em Ciência da Computação (PGComp) da UFBA e tem atuado na orientação de alunos de Mestrado e Doutorado. Ela publicou mais de 70 artigos em conferências e jornais científicos, orientou mais de 55 trabalhos finais de Graduação/Mestrado/Doutorado e aprovou mais de 13 projetos de pesquisa atuando como membro ou coordenadora com

financiamento do CNPQ e FAPESB. Atualmente suas áreas de pesquisa são Processamento de Linguagem Natural, Extração da Informação Aberta, Extração Multimodal e Interoperabilidade Semântica e Pragmática.

Organizadores Gerais do SBSI 2024

Ronney Moreira de Castro é professor associado do Departamento de Ciência da Computação (DCC) da Universidade Federal de Juiz de Fora (UFJF). Possui Doutorado em Informática (ênfase em Sistemas de Informação) pela Universidade Federal do Estado do Rio de Janeiro (UNIRIO), Mestre em Ciência da Computação pela Universidade de Viçosa (UFV) e Graduado em Sistemas de Informação pelo Centro de Ensino Superior de Juiz de Fora (CES-JF). Já foi Coordenador do Curso de Bacharelado em Sistemas de Informação em duas Instituições por um período de 8 anos. É pesquisador e entusiasta da área de Educação em Computação, mais especificamente no uso de Aprendizagem Ativa em disciplinas da área da Computação, especificamente em Sistemas de Informação (SI). Pesquisa também didática nos cursos da área de SI e desenvolve projetos e técnicas voltadas para o uso de Aprendizagem Ativa em diversas disciplinas da Computação. É membro da Sociedade Brasileira de Computação (SBC) e do Comitê Diretivo do Simpósio Brasileiro de Sistemas de Informação 2024.

José Maria Nazar David possui graduação em Engenharia Elétrica pelo Instituto Militar de Engenharia (1983), mestrado (1991) e doutorado (2004) em Engenharia de Sistemas e Computação pela COPPE/Universidade Federal do Rio de Janeiro. Atualmente é professor associado e membro do Programa de Pós-graduação em Ciência da Computação da Universidade Federal de Juiz de Fora. Tem experiência na área de Ciência da Computação, com ênfase em Engenharia de Software, atuando principalmente nos seguintes temas: sistemas colaborativos, desenvolvimento distribuído de software, arquitetura de software, manutenção e evolução de software, e-Science, informática e educação, internet of things (IoT) e ecossistemas de software. É membro da Sociedade Brasileira de Computação (SBC), membro do Comitê Diretivo do Simpósio Brasileiro de Sistemas de Informação 2024.

Sumário

Capítulo 1

Introdução às Séries Temporais: Uma Abordagem Prática em Python

Rogério de Oliveira, Orlando Y. E. Albarracín, Gustavo Rocha da Silva

Abstract

This short course is a practical course that introduces the main concepts of time series and how to manipulate and create time series models in Python. Includes the implementation and analysis of ARIMA-type statistical models and implementations that employ classical and deep machine learning. Practical exercises are included in the course and a final project is suggested. As support material for the course, the language of this text and its references were adapted for this purpose. In the end, the participant is expected to be able to understand, analyze and make predictions from series of data in different contexts, and apply these tools in their own research and professional practice.

Resumo

Este minicurso é um curso prático que apresenta os principais conceitos de séries temporais e como manipular e criar modelos de séries temporais em Python. Inclui a implementação e análise de modelos estatísticos do tipo ARIMA e implementações que empregam aprendizado de máquina clássico e profundo. Exercícios práticos estão incluídos no curso e é sugerido um projeto final. Sendo material de apoio ao curso, a linguagem deste texto e suas referências foram adaptadas para esse propósito. Ao final, espera-se que o participante seja capaz de compreender, analisar e fazer previsões a partir de séries de dados em diferentes contextos, e aplicar essas ferramentas na sua própria pesquisa e prática profissional.

1.1. Introdução

Séries temporais fazem parte de nosso dia a dia e podem ser encontradas em praticamente qualquer área, das ciências físicas e engenharias, às áreas de saúde e de negócios. Análises do comportamento da série, busca de padrões e sazonalidades, simulações e previsões são alguns dos resultados úteis que podemos obter de séries de dados, o que se tornou uma

necessidade e um desafio em muitos campos. Este minicurso oferece uma introdução prática à análise e previsão de séries temporais utilizando a linguagem de programação Python. Embora existam várias formas de se abordar o problema de séries temporais, este minicurso se concentra no uso de modelos Autorregressivos integrados de médias móveis (ARIMA), um dos modelos estatísticos mais aplicados a séries temporais, e no uso de técnicas de aprendizado de máquina. O curso traz exemplos, aplicações e exercícios com dados sintéticos e reais de diversas áreas. Ao final, espera-se que o participante seja capaz de entender, analisar e fazer previsões de séries de dados, podendo aplicar essas ferramentas em suas próprias pesquisas e prática profissional.

O curso está baseado no livro Introdução às Séries Temporais: Uma Abordagem Prática em Python (Oliveira, Albarracín e Silva, 2024), dos mesmos autores deste curso. Uma versão online está disponível no site do livro, *https://github.com/Introducao-Series-Temporais-em-Python*, onde podem ser encontrados, incluindo este texto, todos os códigos e dados empregados, os exercícios e soluções, e outros materiais complementares.

1.2. Principais conceitos

Uma Série Temporal é uma sequência de observações registradas em intervalos de tempo regulares.

Essas observações são medidas tomadas ou encontradas a tempos regulares, e você certamente já se deparou com dados como valores anuais do PIB, preços diários de ações e commodities, quantidade de hits diários em página Web ou site, ou aumento de temperatura global anual. São dados muito comuns e medidos a intervalos regulares (hora, dia, mês etc.). Séries com intervalos muito curtos (segundos ou menos) são ainda encontradas na física e biomedicina, e intervalos muito longos, de décadas ou mais, são encontradas na astronomia e geologia. Aqui a maior parte dos exemplos e exercícios empregam séries diárias, mensais ou anuais, mas os mesmos procedimentos são igualmente aplicáveis a qualquer série. Na Figura 1.1 apresentam-se exemplos de séries temporais.

De acordo com o seu objetivo você pode estar interessado em diferentes tarefas aplicadas a uma série temporal:

1. Fazer previsões de valores futuros
2. Entender o mecanismo gerador da série
3. Descrever e comparar o comportamento da série
4. Procurar periodicidades e padrões relevantes
5. Identificar anomalias
6. Simular séries de dados

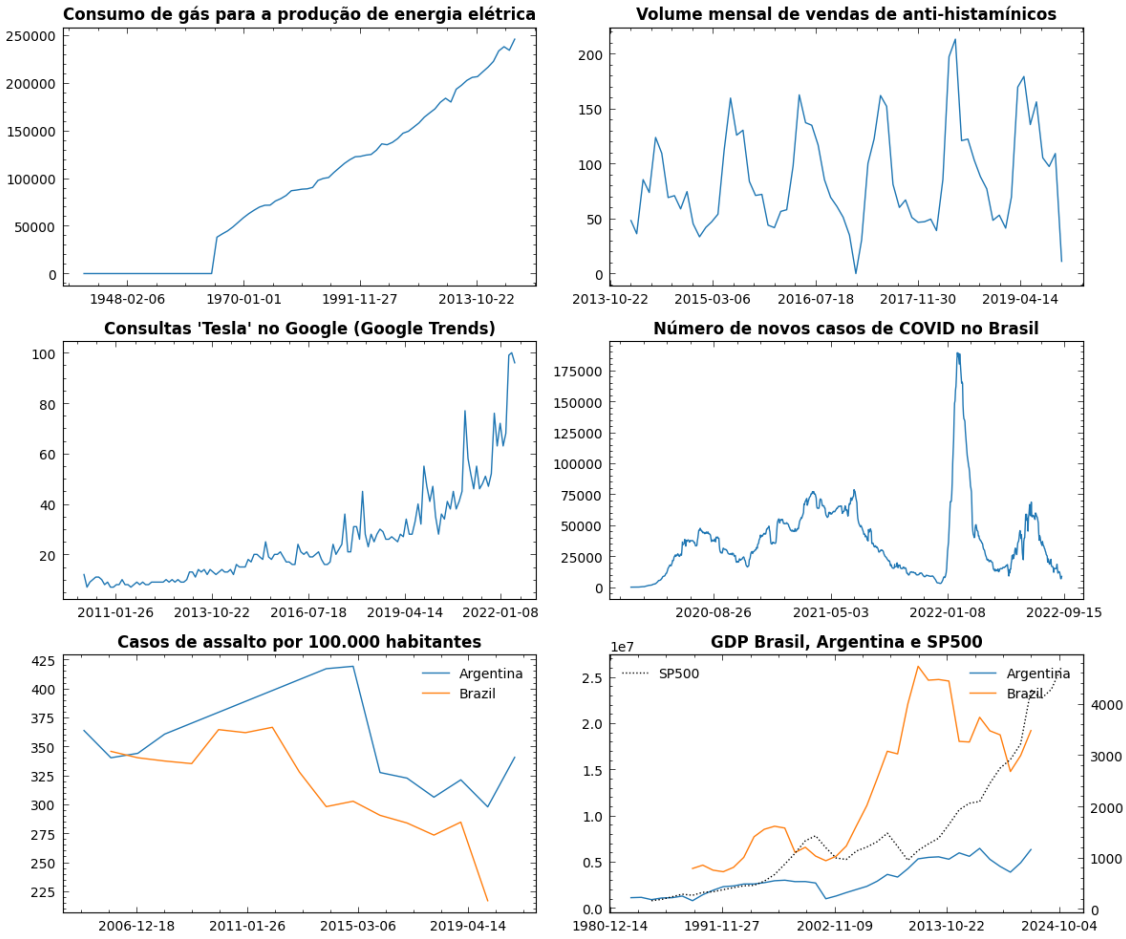


Figura 1.1. Diversos exemplos de séries temporais em diferentes áreas.

e, certamente, fazer previsões de valores desempenha um papel bastante importante. Aqui nos concentraremos unicamente em modelos ARIMA e de aprendizado de máquina supervisionado para previsões. Ambos são modelos paramétricos (possuem um número finito de parâmetros) e modelam as séries no domínio do tempo (diferentemente de modelos que empregam a frequência ou o espaço de estados), estes estão entre os modelos mais amplamente aplicados na economia, finanças, engenharias e outras áreas.

1.2.1. Decomposição de séries temporais

A ideia de construir um modelo é a de criarmos uma simplificação útil dos dados e no caso de séries temporais decompor os dados é o procedimento clássico.

Em geral decompomos uma série temporal em três componentes:

- Tendência
- Sazonalidade
- Resíduos

A tendência representa o comportamento da série no longo do tempo (aumento e/ou diminuição dos valores da série no período estudado), por exemplo, o crescimento da temperatura global ano a ano. A componente sazonalidade apresenta o padrão sazonal da série, como mudanças que ocorrem com alguma periodicidade ao longo do tempo. É o caso dos acréscimos e decréscimos de temperatura que ocorrem ao longo das estações

do ano, independentemente da elevação das temperaturas no longo prazo. Por último, os resíduos, representam os valores da série após retirada a tendência e sazonalidade. Esses resíduos representam a variação não explicada pelos dois componentes anteriores e podem conter ruído ou outros padrões não capturados. Correspondem, por exemplo, às diferenças de temperatura entre dois dias consecutivos da mesma estação causadas por inúmeros fatores como a maior presença de nuvens ou de raios solares naqueles dias.

Diversos desses padrões podem ser observados nas séries da Figura 1, como a tendência linear crescente no consumo de gás para geração de energia e o comportamento sazonal nas vendas de anti-histamínicos

Ciclos e mudanças sazonais são comportamentos bem diferentes nas séries temporais. A sazonalidade é um comportamento recorrente que se repete a intervalos fixos, regulares (a maior temperatura em certas estações do ano, o maior número de visitas em um site de entretenimento aos finais de semana). Já os ciclos são comportamentos recorrentes, mas que ocorrem a intervalos não regulares. Vulcões e terremotos, por exemplo, têm um comportamento recorrente, mas não sabemos quando irão ocorrer, e uma série que represente as temperaturas ou o tremor em torno da cratera do Vulcão Eyjafjallajökull apresentará comportamentos repetitivos, mas que não são sazonais. O mesmo ocorre com os ciclos econômicos que alternam recessão e crescimento, e não sabemos quando irá ocorrer a próxima crise ou o estouro de uma bolha do mercado. Nas séries de Figura 1, por exemplo, podemos observar alguns 'ciclos' de séries financeiras (GDP e SP500) e das ondas de variantes da pandemia de COVID.

1.2.2. Séries temporais em Python

Aquisição dos dados de uma série temporal pode ser feita na forma de dados tabulares. Em Python, o Pandas fornece suporte a vários tipos de formato de arquivos como .csv,.xlsx, .json, .html, .hdf5 e .sql para a criação de um *DataFrame.*

```
import pandas as pd

df = pd.read_csv(course_path + '/data/capea-consulta-cafe.csv')
df.head()
      Data  vista R$  vista US$
0  01/2014    288.98     119.88
1  02/2014    366.32     153.96
2  03/2014    437.24     187.79
3  04/2014    449.45     201.45
4  05/2014    429.28     193.22
```

O Pandas é uma biblioteca para manipulação de dados tabulares e oferece várias funcionalidades para a seleção e transformação dos dados.

```
df[ df['vista R$'] > 1400 ][ ['Data', 'vista R$'] ]
       Data  vista R$
95  12/2021   1452.15
96  01/2022   1482.59
97  02/2022   1485.35
```

Um caso particularmente importante quando se trata de séries temporais trata-se da manipulação de datas.

```
df.dtypes
Data          object
vista R$     float64
vista US$    float64

# seleção errônea dos dados com o atributo Data no formato de object
df[ df['Data'] > '10/2023' ].head(3)
       Data  vista R$  vista US$
10  11/2014    460.96     180.61
11  12/2014    455.20     172.39
22  11/2015    469.39     124.29

# seleção correta dos dados com o atributo Data no formato de datetime
df.Data = pd.to_datetime(df.Data)
df[ df['Data'] > '10/2023' ].head()
          Data  vista R$  vista US$
118 2023-11-01    888.00     181.31
119 2023-12-01    974.46     198.90
120 2024-01-01   1003.74     204.34

df.Data = pd.to_datetime(df.Data)
df[ df['Data'].dt.year > 2023 ].head()
          Data  vista R$  vista US$
120 2024-01-01   1003.74     204.34
```

1.2.2.1. Time index

Em Python, muitas funções úteis para a manipulação de séries temporais como *resample*, gráficos e uso de outros pacotes, requerem que o atributo de tempo da série temporal esteja representado no índice dos dados.

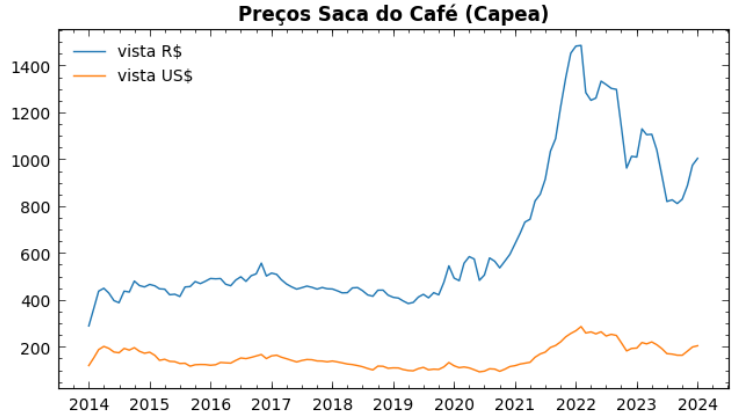
```
df.index = pd.to_datetime(df.Data)
df = df.drop(columns='Data')
df.head()
            vista R$  vista US$
2014-01-01    288.98     119.88
2014-02-01    366.32     153.96
2014-03-01    437.24     187.79
2014-04-01    449.45     201.45
2014-05-01    429.28     193.22

# resample dos dados por ano
df_year = df.resample('Y').mean()
df_year.head()
              vista R$   vista US$
2014-12-31  418.572500  177.984167
2015-12-31  451.494167  137.497500
2016-12-31  494.522500  142.853333
2017-12-31  465.690833  146.050833
2018-12-31  435.646667  120.087500
```

1.2.2.2. Gráficos

Gráficos de séries de dados são fundamentais para explorar e entender o comportamento dos dados e podem ser obtidos com bibliotecas como *matplotlib* ou *seaborn.* Um índice

do tipo `datetime` permite que identificar os dados como uma série temporal e formatar a escala de tempo dos gráficos.

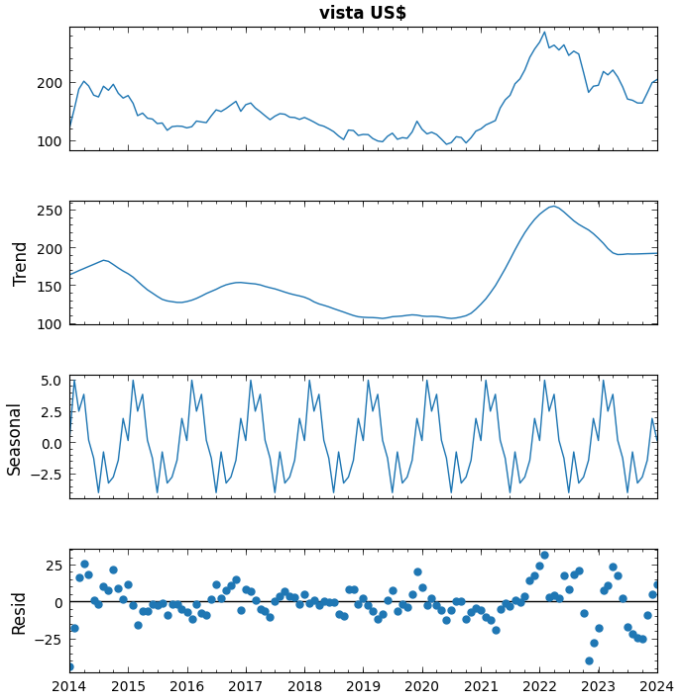
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.plot(df,label=['vista R$','vista US$'])
plt.title('Preços Saca do Café (Capea)')
plt.legend()
```



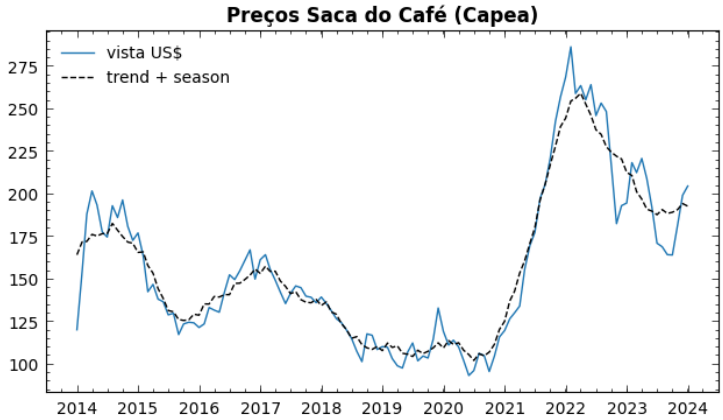
1.2.2.3. Decompondo uma série temporal

O principal pacote para modelos estatísticos de séries de dados em Python é o statsmodel, e a função `seasonal_decompose()` permite decompor uma série em suas componentes de tendência, sazonalidade e resíduos.

```
from statsmodels.tsa.seasonal import seasonal_decompose

result = seasonal_decompose(df['vista US$'], model='additive',
extrapolate_trend=1)
fig = result.plot()
fig.set_size_inches((7, 8))
plt.show()

df_compose = pd.DataFrame()
df_compose['trend'] = result.trend
df_compose['seasonal'] = result.seasonal
df_compose['resid'] = result.resid

df_compose.head()
                 trend  seasonal      resid
2014-01-01  163.774583  0.142035 -44.036618
2014-02-01  166.537917  4.960345 -17.538262
2014-03-01  169.301250  2.486304  16.002446
2014-04-01  172.064583  3.833470  25.551946
2014-05-01  174.827917  0.168887  18.223196

# Acesse o código completo no site do material complementar
```




```
plt.plot(df,label=['vista R$','vista US$'])
plt.plot(df_compose['compose'],color='k',linestyle='dashed',
       label='trend + season')

plt.title('Preços Saca do Café (Capea)')
plt.legend()
```



1.2.3. Séries aditivas e multiplicativas

A depender da natureza da série as componentes de tendência, sazonalidade e resíduo podem ser combinadas de forma aditiva ou multiplicativa, isto é:

$$Y_t = T_t + S_t + R_t$$

para modelos aditivos, ou:

$$Y_t = T_t \times S_t \times R_t$$

para modelos multiplicativos, sendo Y_t a série observada e T_t, S_t, R_t respectivamente as componentes de tendência, sazonalidade e resíduo.

Geralmente, o comportamento aditivo é apropriado quando a magnitude da sazonalidade ou tendência não depende do nível da série temporal já o comportamento multiplicativo é apropriado quando a magnitude da sazonalidade ou tendência varia com o nível da série temporal. De qualquer modo, lembre-se que os modelos são simplificações que buscam ser úteis, e uma série, por exemplo a atividade solar a cada mês, não tem qualquer obrigação de se comportar de forma aditiva ou multiplicativa, e nem sempre é muito simples identificar se uma série é aditiva ou multiplicativa.

1.2.4. Estacionariedade

A maior parte dos modelos estatísticos assumem que a série seja estacionária. Uma série temporal é dita estacionária, no sentido amplo, quando tem média e variância constantes e função de autocovariância entre dois períodos distintos depende apenas da defasagem de tempos entre os períodos. Os valores destas séries se desenvolvem ao redor de um certo nível com variância constante, isto é, ausência completa de tendência e sazonalidade da série.

Há vários testes estatísticos para verificar a estacionariedade de uma série temporal, mas os testes mais comuns são o teste Augmented Dickey Fuller (“ADF”) e o teste Kwiatkowski-Phillips-Schmidt-Shin (“KPSS”) e que, basicamente, verificam a presença de tendência na série. A análise exploratória dos dados é essencial para uma interpretação adequada desses testes.

1.2.4.1. Teste Augmented Dickey-Fuller (ADF)

O teste de Dickey-Fuller Aumentado (ADF) é um teste estatístico comumente usado para verificar a presença de uma tendência estocástica em uma série temporal.

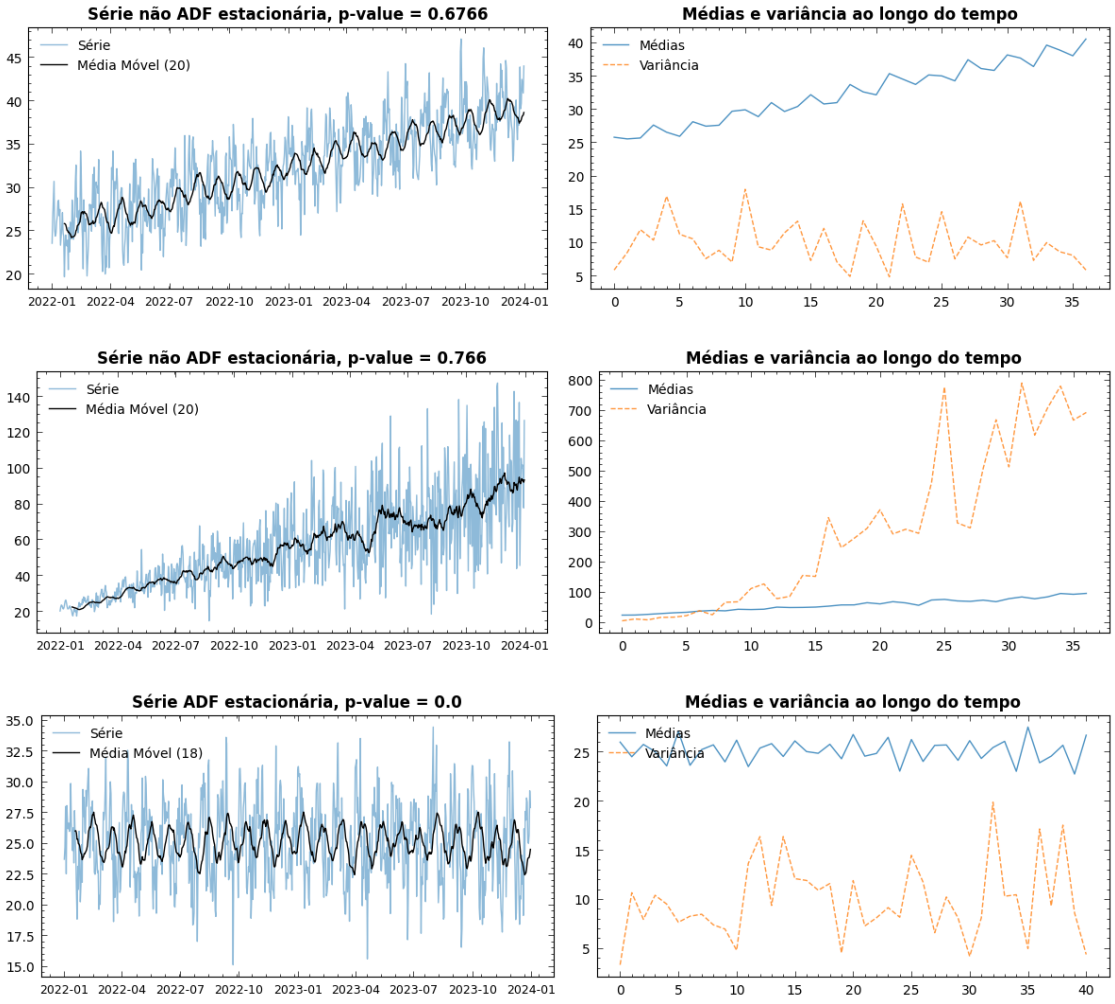
$$H_0\text{:A série não é estacionária}$$

$$H_1\text{:A série é estacionária}$$

Se o valor-p for menor que um determinado nível de significância (geralmente 0,05), então rejeitamos a hipótese nula e concluímos que a série é estacionária, ou seja, não possui uma tendência estocástica significativa. Os gráficos a seguir ilustram diferentes casos de estacionariedade e não estacionariedade com os respectivos p-values do teste ADF.

```
# Acesse o código completo no site do material complementar
from statsmodels.tsa.stattools import adfuller, kpss
def ADF(df,verbose=False):
  result = adfuller(df)
  if verbose:
    print('ADF Statistic: %f' % result[0])
    print('p-value: %f' % result[1])

    if result[1] < 0.05:
      print('Série é ADF estacionária')
    else:
      print('Série é não ADF estacionária')
  return result
```

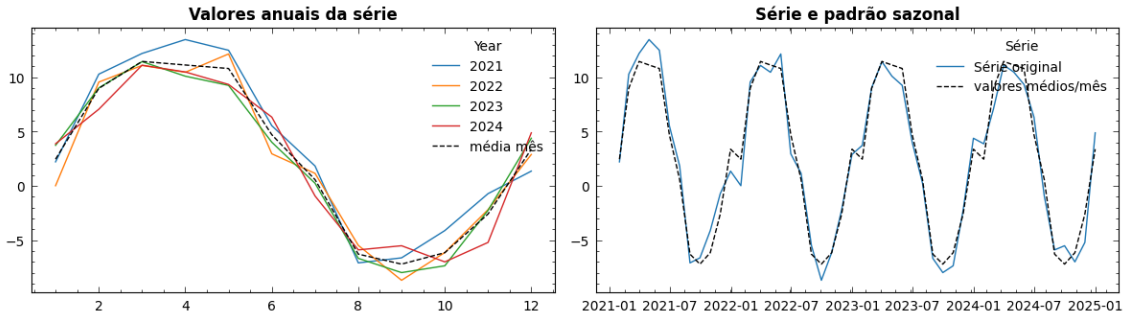
```
PLOT(no_stacionary_avg, ADF(no_stacionary_avg))
PLOT(no_stacionary_var, ADF(no_stacionary_var))
PLOT(stacionary, ADF(stacionary))
```



1.2.5. Sazonalidade

A inspeção visual das séries desempenha um papel fundamental na identificação de sazonalidades, como também da estacionariedade apesar dos testes disponíveis. Gráficos de agrupamentos com os valores médios por dia, mês etc. são bastantes empregados e constituem a base de muitos modelos de sazonalidade.

```
# Criando uma série de periodicidade anual
date_range = pd.date_range(start='2021-01-01', periods=4*12, freq='M')

# Para o código completo ver o material complementar
```



1.2.5.1. Periodograma

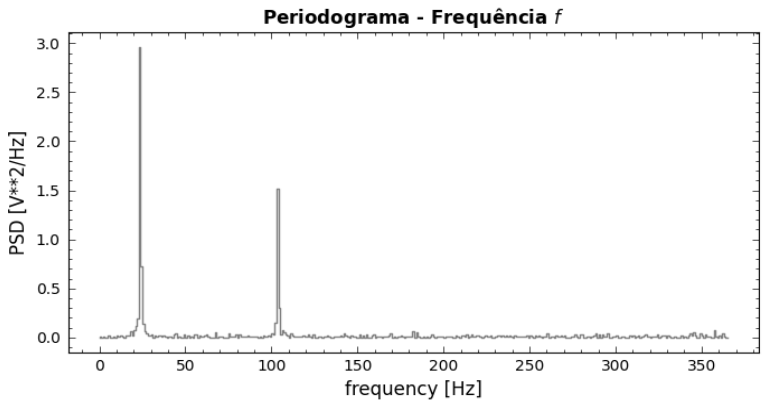
Apesar da importância e predominância da inspeção visual, o periodograma pode ser uma ferramenta bastante útil e mais adiante introduziremos os gráficos autocorrelação que também nos ajudam a identificar sazonalidades.

O periodograma apresenta a distribuição das frequências em um sinal ao longo do tempo e as frequências mais presentes podem ser empregadas para identificar as sazonalidades presentes na série.

$$P_i = \frac{\text{Total de Períodos}}{freq_i}$$

onde, P_i é a periodicidade da frequência $freq_i$. Pode haver inúmeras frequências, mas podemos nos limitar as mais predominantes (1, 2 ou 3 mais presentes).

```
from scipy import signal

frequencies, spectrum = signal.periodogram(stacionary['values'],
fs=len(stacionary))

# Para o código completo ver o material complementar
     frequencies  spectrum    periods
24          24.0  2.960242  30.416667
104        104.0  1.515986   7.019231
25          25.0  0.723724  29.200000
105        105.0  0.305413   6.952381
23          23.0  0.188924  31.739130
```



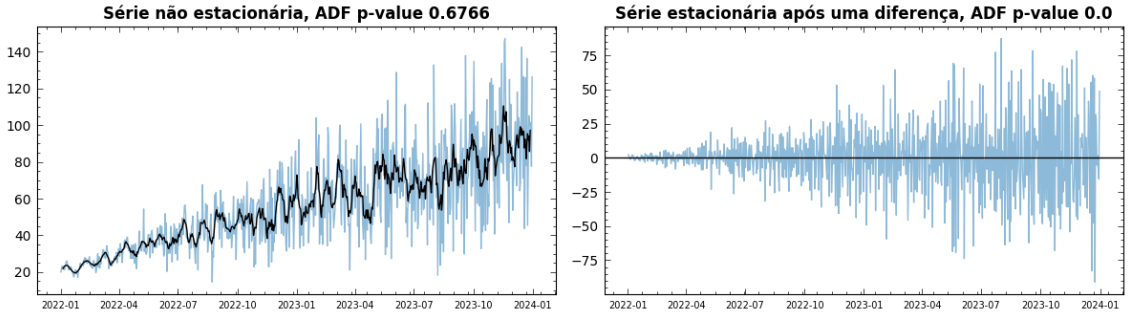
1.2.6. Diferenciação e Log: eliminando a tendência e sazonalidade

Um dos métodos mais simples para se eliminar a tendência de uma série temporal é construir uma nova série por diferenciação. Nesta nova série, o valor é calculado como a diferença entre o valor no instante t e o valor no intervalo de tempo anterior.

$$\Delta^1 y_t = y_t - y_{t-1}$$
$$\Delta^d y_t = y_t - y_{t-d}$$

A diferenciação acima é de ordem 1 e diferenciações maiores podem ser aplicadas para, por exemplo, eliminar uma tendência polinomial. Para tendências exponenciais pode ser necessário aplicar, do mesmo modo, a transformação logarítmica da série (tornando-a linear) antes de se aplicar a diferenciação.

```
fig, ax = plt.subplots(1,2,figsize=(12,3.5))
ax[0].plot(no_stacionary_var,alpha=0.5,lw=1)
ax[1].plot(no_stacionary_var.diff().dropna(),alpha=0.5,lw=1);

# Para o código completo ver o material complementar
```



1.2.7. Resíduos

Os pressupostos do modelo ARIMA incluem a estacionariedade da série temporal (ou a estacionariedade por diferenciação), e a normalidade dos resíduos. O termo "resíduo" é utilizado para designar a diferença entre os valores reais e o ajuste obtido da série por algum modelo, consistindo em uma componente não explicada da série.

Geralmente, os resíduos são utilizados para selecionar o 'melhor' modelo, definido como aquele que gera os menores resíduos. Essa diferença pode ser medida de diversas formas, mas ao final todas refletem as diferenças da série real Y_t e os valores estimados $\hat{Y}_t$ que queremos minimizar. Por serem muito empregadas algumas dessas medidas recebem nomes especiais:

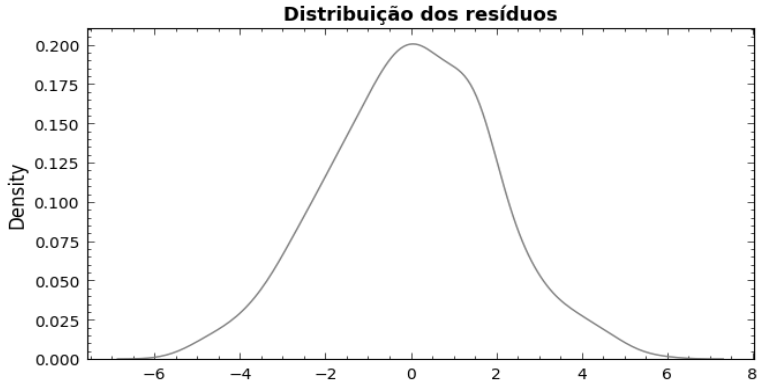
- Erro Médio Absoluto, $MAE = \frac{1}{n}\sum\left|Y_t - \hat{Y}_t\right|$
- Erro Médio Quadrático, $MSE = \frac{1}{n}\sum\left(Y_t - \hat{Y}_t\right)^2$
- Raiz do Erro Médio Quadrático, $RMSE = \sqrt{\frac{1}{n}\sum\left(Y_t - \hat{Y}_t\right)^2}$
- Erro Percentual Absoluto Médio, $MAPE = \frac{1}{n}\sum\left|\frac{Y_t - \hat{Y}_t}{Y_t}\right|$

Todas medidas que podem ser fácil e diretamente calculadas ou pode-se empregar algum pacote.

```
import statsmodels.tools.eval_measures as eval_measures

def error_measures(y, y_pred):
# Para o código completo ver o material complementar

result = seasonal_decompose(stacionary, model='additive')

non_NA = pd.merge(stacionary, result.trend, how='inner', left_index=True, rig
ht_index=True).dropna().index # exclui valores nulos das previsões

y = stacionary.loc[non_NA]['values']
y_pred = result.trend.loc[non_NA] + result.seasonal.loc[non_NA]

_ = error_measures(y,y_pred)

MSE:    3.6922
MAE:    1.5335
RMSE:   1.9215
MAPE:   0.0628
```

Além de buscar minimizar o erro, ou resíduos, queremos que seus valores sejam independentes, no sentido de não estarem correlacionados, e que apresentem uma distribuição normal.

```
sns.kdeplot(y - y_pred)
plt.title('Distribuição dos resíduos')
```



A normalidade dos resíduos pode ser verificada através de um gráfico de distribuição ou do tipo qqplot, o que é mais comum que o uso de testes de hipótese de normalidade. Para independência dos resíduos pode-se empregar um gráfico de autocorrelação dos resíduos. Veremos esse gráfico mais adiante como também uma função `plot_diagnostics()` do statsmodels que agrega todos esses gráficos para uma melhor análise dos resíduos.

1.2.8. Exercícios e referências

Além do livro base deste curso há muitos livros texto que tratam desde os conceitos de séries temporais. Montgomery et al. (2015) e Chatfield (1996) são livros texto clássicos

e um ótimo texto nacional é Morettin e Toloi (2006). Eles tratam dos conceitos de séries temporais e de modelos estatísticos, mas não trazem código ou implementações. Hyndman e Athanasopoulos (2018) é um livro texto bastante tradicional de séries temporais e que também traz exemplos de implementação, embora empregue a linguagem R, bastante empregada no tratamento de séries de dados. O texto também tem uma versão disponível online. Box et. al. (2015) é um clássico e uma referência obrigatória para quem quer se aprofundar formalmente em modelos estatísticos para séries temporais.

Para conceitos introdutórios de Python, Pandas e aprendizado de máquina clássico, Vanderplas (2016) é um texto muito útil, possui uma seção dedicada a séries temporais e, assim como o material deste curso e nosso livro texto, está disponível online em formato de notebooks Python. Também disponíveis online, Oliveira (2022) é uma introdução útil para visualização de dados em Python, incluindo séries de dados, e Kong et al. (2020) apresenta a implementação de vários métodos numéricos em Python.

Os sites de documentação das bibliotecas Pandas (2024) e statsmodels (2024) são úteis para a melhor compreensão de muitas das funções empregadas nesta e nas seções seguintes.

Por último, para complementar esta seção, De Gooijer e Hyndman (2006) é um artigo curto que revisa, sem fórmulas ou implementações, as ideias dos principais modelos de séries temporais clássicos e como eles evoluíram antes da introdução dos modelos de aprendizado de máquina.

Os exercícios dessa seção e suas soluções podem ser acessadas no material complementar do curso *exerc_parte1_introd*.

1.3. Modelos autorregressivos

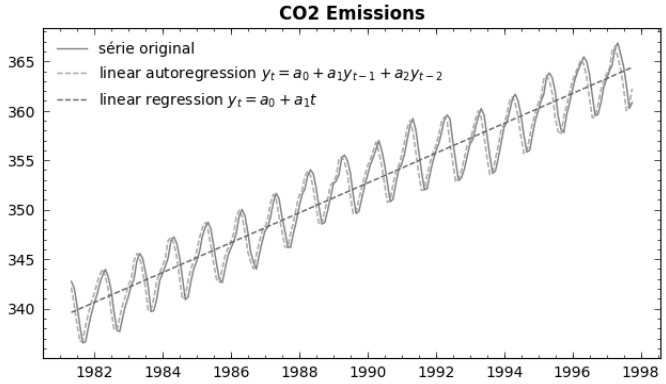
A ideia principal dos modelos autorregressivos, incluindo o ARIMA, consiste em modelar a dependência serial dos dados, uma vez que, na maioria das séries, observa-se que os valores recentes estão correlacionados com seus valores passados e que a força dessa dependência diminui quando considerados valores mais distantes no tempo. Assim, é razoável pensar que o valor de amanhã das vendas de uma safra ou do volume de chuvas está correlacionado com os valores observados ontem, ou no dia anterior, e que essa correlação diminui conforme nos afastamos no tempo.

1.3.1. Modelos de regressão e autorregressão

Modelos de regressão se baseiam em variáveis independentes para prever a variável dependente, os modelos de autorregressão usam os próprios valores passados da variável dependente para fazer previsões. Para séries temporais, em geral, faz mais sentido empregarmos a forma autorregressiva que empregar o tempo (que seria única variável independente em uma única série de dados). Entretanto, em ambos os casos o cálculo dos coeficientes pode ser feito do mesmo modo (em geral um método de mínimos quadrados).

```
co2 = pd.read_csv(path + 'co2.csv',index_col=0,parse_dates=True)
co2['CO2_t-1'] = co2.CO2.shift()
co2['CO2_t-2'] = co2.CO2.shift().shift()
co2 = co2.dropna()
co2.head()
```

```
            time     CO2  CO2_t-1  CO2_t-2
1981-07-01     2  340.32   342.08   342.74
1981-08-01     3  338.26   340.32   342.08
1981-09-01     4  336.52   338.26   340.32
1981-10-01     5  336.68   336.52   338.26
1981-11-01     6  338.19   336.68   336.52

import statsmodels.api as sm

# regressão
X = co2[['time']]
y = co2[['CO2']]

X = sm.add_constant(X)
y_regression = sm.OLS(y, X).fit().predict()

# auto regressão
X = co2[['CO2_t-1','CO2_t-2']]

X = sm.add_constant(X)
y_autoregression = sm.OLS(y, X).fit().predict()

# Para o código completo ver o material complementar
```



1.3.2. Modelo ARIMA

Existem vários tipos de modelos, empregando diferentes princípios, para análise e previsões de séries temporais. O modelo ARIMA (Autorregressivos Integrados de Médias Móveis), é um modelo de análise estatística amplamente utilizado para modelar séries temporais estacionárias e não estacionárias, e constituir a base de modelos mais complexos (ARIMAX, VARIMAX, SARIMAX, ARCH, GARCH etc. envolvendo variáveis exógenas, sazonalidades e volatilidade de séries temporais). O ARIMA é, portanto, um modelo fundamental. O modelo ARIMA consiste nos seguintes componentes:

- $AR(p)$: Termo autoregressivo que incorpora a dependência entre uma observação e uma série de observações defasadas até a ordem p, o que pode ser escrito do seguinte modo:

$$Y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \ldots + \phi_p Y_{t-p} + \epsilon_t$$, onde ϵ_t é o erro do modelo.

- $I(d)$: Termo integrado que envolve diferenciar na ordem d os dados da série temporal para torná-los estacionários.

$$W_t = \Delta^d Y_t = Y_t - Y_{t-d}, \text{ e, portanto,}$$
$$Y_t = W_t + Y_{t-d}$$

- $MA(q)$: Termo de média móvel que leva em conta a dependência entre uma observação e um erro residual de um modelo de média móvel de ordem q.

$$Y_t = \mu + e_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \ldots + \theta_q \epsilon_{t-q}, \text{ onde } \mu \text{ é a média da série.}$$

Tanto o modelo AR como o modelo MA são conceitualmente e podem ser calculados como uma regressão linear do valor atual da série, respectivamente sobre seus valores passados (AR) e os termos de erro com relação à média móvel (MA). Ao final o modelo ARIMA completo pode ser escrito como:

$$W_t = \underbrace{\Delta^d Y_t}_{I(d)}$$

$$W_t = \underbrace{\phi_1 W_{t-1} + \phi_2 W_{t-2} + \cdots + \phi_p W_{t-p}}_{AR(p)} \underbrace{+\theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q}}_{MA(q)} + \epsilon_t,$$

Assim, o modelo ARIMA é a junção de modelos que podem também ser empregados de forma independente como segue (Tabela 1.1).

Tabela 1.1. Junção de modelos ARIMA

Model	ARIMA(p,q,d)	Tipo de Série
$AR(p)$	$ARIMA(p,0,0)$	estacionária
$MA(q)$	$ARIMA(0,0,q)$	estacionária
$ARMA(p,q)$	$ARIMA(p,0,q)$	estacionária
$ARIMA(p,d,q)$	$ARIMA(p,d,q)$	não estacionária

E, em todos os casos, como vimos assumir-se que os resíduos seguem uma distribuição normal.

1.3.2.1. Exemplo 1

Considerando a série Y_t estacionária podemos considerar os seguintes modelos (ver Tabela 1.2) de ordem 1:

Tabela 1.2. Modelos estacionários

Modelo	ARIMA(p,q,d)
$AR(1)$ ou $ARIMA(1,0,0)$	$Y_t = \phi_0 + \phi_1 Y_{t-1} + \epsilon_t$
$MA(1)$ ou $ARIMA(0,0,1)$	$Y_t = \epsilon_t + \theta_1 \epsilon_{t-1}$
$ARMA(1,1)$ ou $ARIMA(1,0,1)$	$Y_t = \phi_0 + \phi_1 Y_{t-1} + \epsilon_t + \theta_1 \epsilon_{t-1}$

1.3.2.2. Exemplo 2

Considerando a série Y_t estacionária para a diferenciação de ordem 1, podemos construir a série estacionária:

$W_t = Y_t - Y_{t-1}$, e, então, o modelo da Tabela 1.3:

Tabela 1.3. Modelos estacionários

Modelo	ARIMA(p,q,d)
$ARIMA(1,1,1)$	$W_t = \phi_0 + \phi_1 W_{t-1} + \epsilon_t + \theta_1 \epsilon_{t-1}$

Em que é ajustado um modelo ARMA(1,1) à série diferenciada W_t.

1.3.2.3. Exemplo 3

Modelos de suavização, uma outra técnica bastante empregada em séries temporais, também podem ser derivados de modelos ARIMA como o modelo básico de suavização (ARIMA(0,1,1)), o modelo de Holt Amortecido (ARIMA(0,1,2)) e o método linear de Holt (ARIMA(0,2,2)). E o modelo ARIMA Sazonal, SARIMA(p,d,q)(P,D,Q), consiste em um modelo ARIMA em que a componente sazonal é também modelada do mesmo modo com os parâmetros P, D, Q.

1.3.3. Autocorrelação e autocorrelação parcial

A correlação de duas variáveis x, y refere-se à sua dependência linear e é dada por:

$$\rho(x,y) = \frac{cov(x,y)}{\sqrt{var(x)var(x)}}$$

Em séries temporais, nos referimos à autocorrelação à correlação entre valores da mesma série para intervalos de tempo diferentes e, por exemplo a correlação do valor atual com o valor do instante anterior da série é dado por:

$$\rho(x_t, x_{t-1}) = \frac{cov(x_t, x_{t-1})}{\sqrt{var(x_t)var(x_{t-1})}}$$

A função ACF (Autocorrelation Function) fornece os valores de autocorrelação para diferentes defasagens de valores. A função ACF considera a correlação entre os valores x_t e x_{t-k}, $k = 0,1,\ldots$. A função PACF (Partial Autocorrelation Function) estima os valores de autocorrelação entre x_t e x_{t-k} excluindo as dependências anteriores. Ela fornece uma boa estimativa para os valores p do modelo AR. Na Tabela 1.4. apresentam-se o comportamento das ACF e PACF.

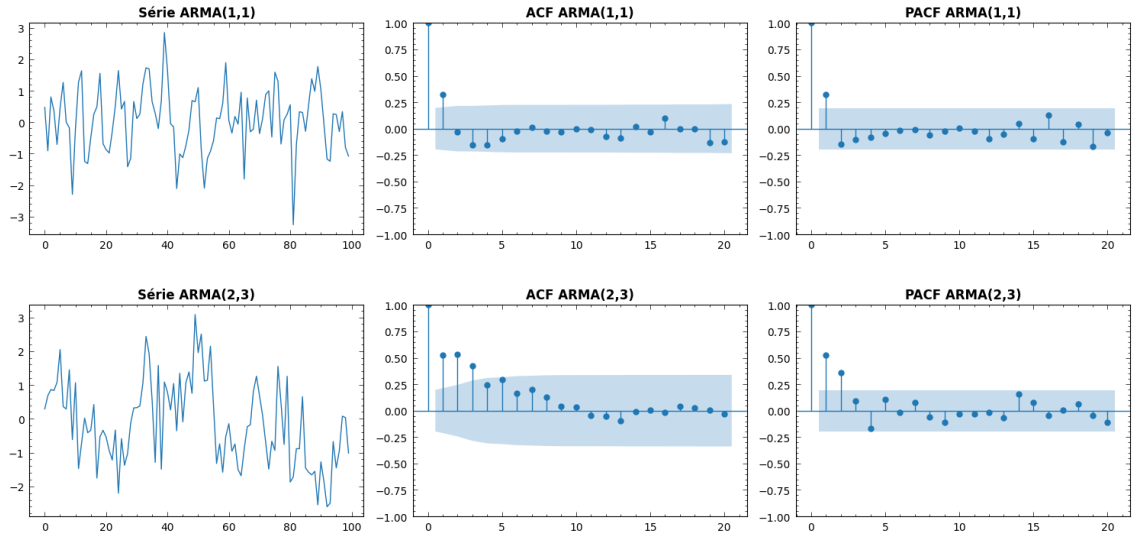
Tabela 1.4. Comportamento das funções ACF e PACF

Função	$MA(q)$	$AR(p)$	$ARMA(p,q)$
ACF	Desprezível após q	Decaimento[1]	Decaimento[1] após q
$PACF$	Decaimento[1]	Desprezível após p	Decaimento[1] após p

[1] *Decaimento tipo exponencial ou sinusoide*

```
# Para o código completo ver o material complementar
model_name = 'ARMA(1,1)'
ar= np.array([1,-0.3,0,0,0,0]) # ar_coefs has the form [1,-a_1,-a_2,...,-a_p]
ma= np.array([1,0.3,0,0,0,0])  # ma_coefs has the form [1,m_1,m_2,...,m_q]
ts= arma_generate_sample(ar, ma, nsample=100)
```

```
plot_ts_acf_pacf(ts,model_name)

model_name = 'ARMA(2,3)'
ar = np.array([1,-0.4,-0.3,0,0,0]); ma = np.array([1])
ts = arma_generate_sample(ar, ma, nsample=100)
plot_ts_acf_pacf(ts,model_name)
```



1.3.4. Aplicando um modelo ARIMA

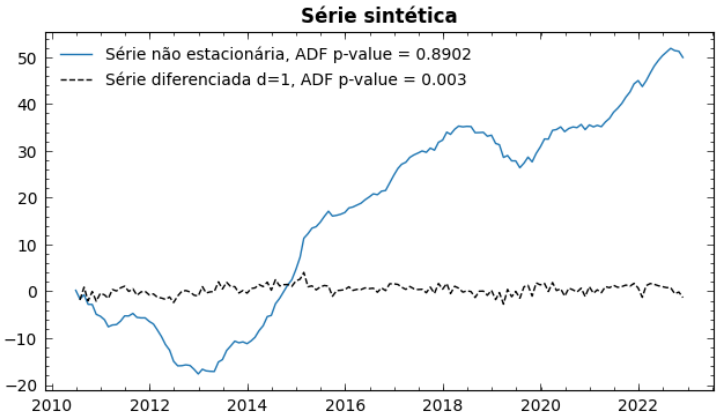
Após uma exploração e entendimento dos dados a aplicação do modelo ARIMA envolve, desse modo, os seguintes passos:

1. Análise da estacionariedade da série, e escolha do parâmetro d de diferenciação que torna a série estacionária nos casos em que não é estacionária.
2. Verificação da autocorrelação e autocorrelação parcial da série, identificando os potenciais valores p (através da PACF) e q (através da ACF) do modelo.
3. Análise dos modelos e seleção dos parâmetros, seguindo alguma métrica de desempenho do modelo como o critério de informação de Akaike (AIC) ou o critério Bayesiano de Schwarz (BIC) [1].
4. Análise dos resíduos, observando-se a normalidade e independência dos resíduos.
5. Previsões, com o modelo selecionado.

1.3.4.1. Análise da estacionariedade da série

Vamos produzir uma série sintética simulando uma série ARIMA(2,1,0). A série original é não estacionária, mas a série diferenciada de ordem 1 torna-se estacionária. Obtemos então nosso parâmetro $d = 1$.

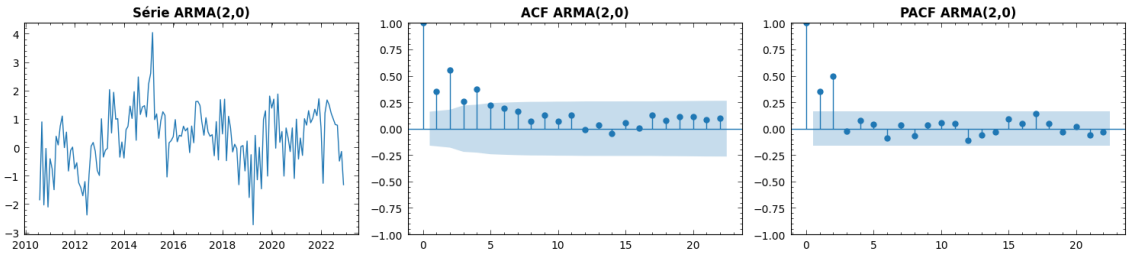
```
# Para o código completo ver o material complementar

df = simulate_ARIMA(phi = np.array([0.2, 0.6]),
                    theta = np.array([0.0]), d=1, n = 150)
df = pd.DataFrame(df.flatten(),columns=['values'])
df.index = pd.date_range(start='6/1/2010', periods = 150, freq='M')
```



1.3.6. Verificação da autocorrelação e autocorrelação parcial da série

Os parâmetros p e q são obtidos respectivamente da análise dos gráficos de autocorrelação parcial e autocorrelação, e obtemos assim os valores $p = 2$ e $q = 0$.

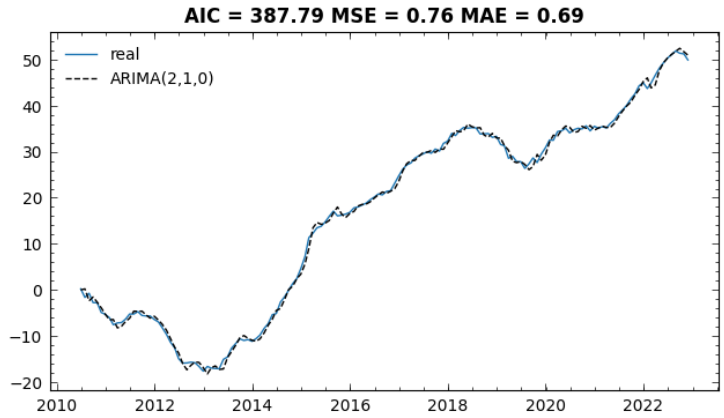
```
plot_ts_acf_pacf(df.diff().dropna(),'ARMA(2,0)')
```



1.3.7. Análise dos modelos e seleção dos parâmetros

Podemos então aplicar esses parâmetros e analisar o ajuste da série e métricas do modelo obtido.

```
from statsmodels.tsa.arima.model import ARIMA

p = 2; d = 1; q = 0
model = ARIMA(df, order=(p, d, q))
results = model.fit(method_kwargs={'maxiter':700})

# Para o código completo ver o material complementar
```



O sumário do statsmodels fornece uma série de informações para análise do modelo.

```
print(results.summary())
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                 values   No. Observations:                  150
Model:                 ARIMA(2, 1, 0)   Log Likelihood                -190.896
Date:                Mon, 22 Jan 2024   AIC                            387.791
Time:                        21:58:26   BIC                            396.803
Sample:                    06-30-2010   HQIC                           391.453
                         - 11-30-2022
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.1909      0.062      3.071      0.002       0.069       0.313
ar.L2          0.5302      0.062      8.583      0.000       0.409       0.651
sigma2         0.7549      0.091      8.259      0.000       0.576       0.934
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):                 0.15
Prob(Q):                              0.99   Prob(JB):                         0.93
Heteroskedasticity (H):               1.25   Skew:                            -0.06
Prob(H) (two-sided):                  0.43   Kurtosis:                         2.90
===================================================================================
```

O resultado do ajuste do modelo é apresentado em três partes. Na primeira parte há informações gerais do modelo, como ordem do modelo, número de amostras e métric as como o AIC e BIC. Na segunda, há os coeficientes estimados, sua significância e inte rvalo de confiança. A terceira parte traz indicadores para análise dos resíduos.

Os gráficos das funções PACF e ACF sugerem valores de p e q a serem empregados. Mas podemos buscar alguma métrica de desempenho do modelo, como uma métrica de erro ou o AIC, para busca dos melhores parâmetros. Na prática, geralmente, os termos MA são adicionados para melhor o ajuste do modelo.

```
model_list = []; AIC_list = []

d = 1
for p in range(0,5):
  for q in range(0,5):
```

```
    model = ARIMA(df, order=(p, d, q))
    results = model.fit(method_kwargs={'maxiter':700})
    model_list.append('ARIMA(' + str(p) + ',' + str(d) + ',' + str(q) + ')')
    AIC_list.append(np.round(results.aic,4))

results_df = pd.DataFrame({'model': model_list, 'AIC': np.array(AIC_list)
}).sort_values('AIC',ascending=True)
display(results_df)
```

```
model       AIC
10  ARIMA(2,1,0)  387.7915
22  ARIMA(4,1,2)  388.3696
11  ARIMA(2,1,1)  389.7083
...
1   ARIMA(0,1,1)  448.2978
0   ARIMA(0,1,0)  459.6912
```

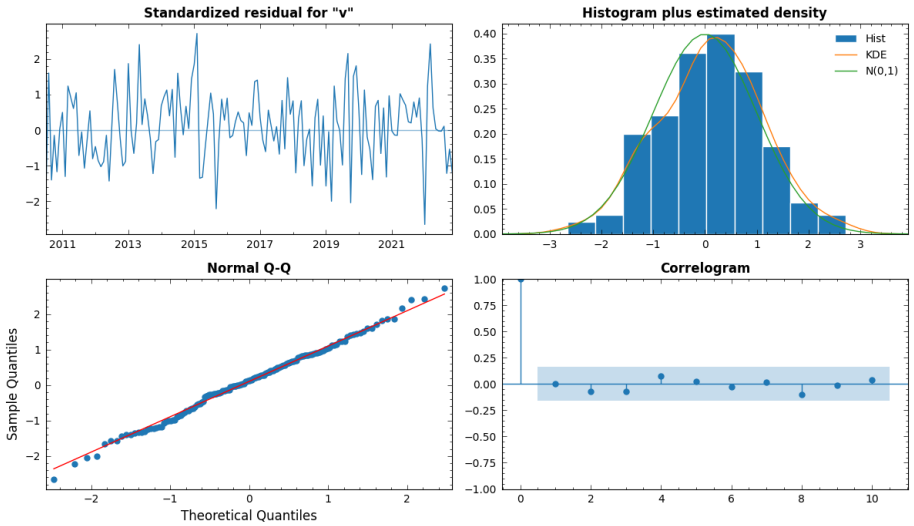
Assim, para o melhor desempenho pela métrica AIC podemos escolher os valores $p = 2, d = 1, q = 0$ e, então, analisar se os erros obtidos satisfazem as premissas do modelo.

1.3.8. Análise dos resíduos

Na parte dos indicadores de resíduos os valores Prob correspondem a p-valores das métricas indicadas (no exemplo, p-value Ljung-Box = 0.99 > 0.05 indica que os resíduos são ruído branco e p-value Jarque-Bera = 0.93 > 0.05 indica que os resíduos têm uma distribuição normal – ver `ARIMA(df, order=(2, 1, 0)).fit.summary()`).

A análise dos resíduos ainda é normalmente complementada com análise dos gráficos da função `plot_diagnostics()` para, além da normalidade, verificarmos a independência dos valores de erro. No exemplo, como podemos ver, os erros não estão corelacionados.

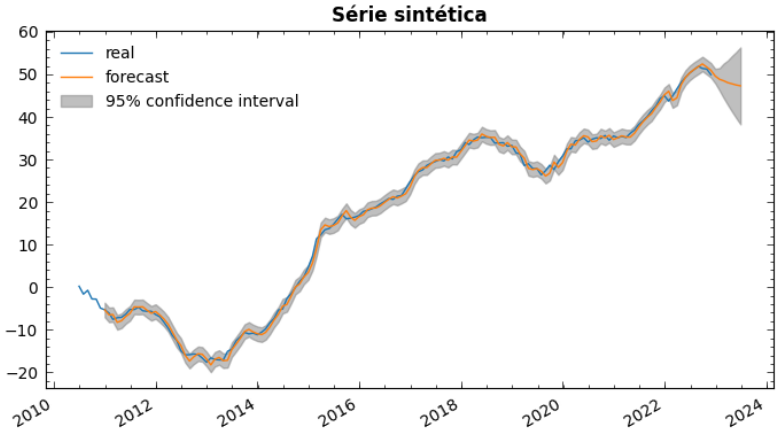
```
fig = plt.figure(figsize=(12,7))
results.plot_diagnostics(fig=fig)
plt.tight_layout()
plt.show()
```



1.3.9. Previsões

Uma vez selecionado o modelo podemos fazer previsões de valores futuros da série. Por exemplo, a predição para os próximos 6 meses.

```
from statsmodels.graphics.tsaplots import plot_predict
fig, ax = plt.subplots(figsize=(7,4))
ax.plot(df,label='real')

plot_predict(results, start=pd.to_datetime('2010-12-31'),
       end=pd.to_datetime('2023-06-30'),ax=ax)
ax.set_title('Série sintética')
plt.tight_layout()
plt.show()
```




```
results.forecast(6)

2022-12-31    49.595265
2023-01-31    48.833805
2023-02-28    48.513812
2023-03-31    48.048958
2023-04-30    47.790525
2023-05-31    47.494697
```

O segundo modelo com menor AIC é ARIMA(4,1,2), a diferença dos AIC é pequena, será que podemos usá-lo também? Na prática, você pode implementar os n primeiros modelos com menor AIC e fazer previsões das últimas observações registradas (não usadas no ajuste do modelo) e selecionar o modelo com menor erro, empregando alguma das métricas que vimos como MSE, MAE, RMSE ou MAPE.

1.3.10. Exercícios e referências

As referências da seção anterior, Montgomery et al. (2015), Chatfield (1996), Morettin e Toloi (2006) são livros texto acessíveis e bastante úteis aqui para compreender os modelos ARIMA e seus submodelos, enquanto Box et. al. (2015) é um texto avançado.

Análise e implementações de modelos clássicos em R podem ser encontradas em Coghlan (2024), Shmueli et al. (2016) e Hyndman e Athanasopoulos (2018). Peixeiro (2022) traz implementações em Python com o pacote statsmodels, sendo mais próximo das implementações empregadas aqui. Nielsen (2019) é outro texto voltado para modelos

práticos de séries temporais em Python, incluindo modelos de espaço de estados e de aprendizado de máquina.

Por último a documentação statsmodels (2024) é essencial para a compreensão das implementações e exercícios desta seção que podem ser acessados no material complementar do curso em exerc_parte2_arima.

1.4. Aprendizado de máquina

Os modelos autorregressivos, como vimos, baseiam-se em modelos de regressão linear. Modelos de aprendizado de máquina supervisionado também podem ser empregados para fazer previsões de valores futuros de séries temporais e têm tido um sucesso grande em fornecer previsões mais precisas para séries muito complexas, em particular modelos baseados em redes neurais profundas desenvolvidas com PyTorch ou TensorFlow/Keras, ou modelos híbridos como o Greykite (2024) (LinkedIn) e Prophet (2024) (Meta). As diferenças entre um modelo estatístico e um modelo de aprendizado de máquina residem na capacidade de precisão e explicabilidade de cada modelo e, de modo bastante simples, poderíamos dizer que se trata de uma escolha entre fazer previsão mais precisa (modelos de aprendizado de máquina) ou compreender os processos subjacentes aos dados (modelos estatísticos), não excluindo o caso em que ambas as técnicas podem ser aplicadas.

Independente se empregamos um modelo de aprendizado de máquina clássico (como regressores de árvores de decisão, florestas de árvores aleatórias e k-vizinhos mais próximos) ou de redes neurais, incluindo aprendizado profundo (deep learning), o processo é bastante semelhante partindo por empregar os valores defasados da série como variáveis preditoras (como vimos no modelo autorregressivo anteriormente). Este processo está representado na Figura 1.2.

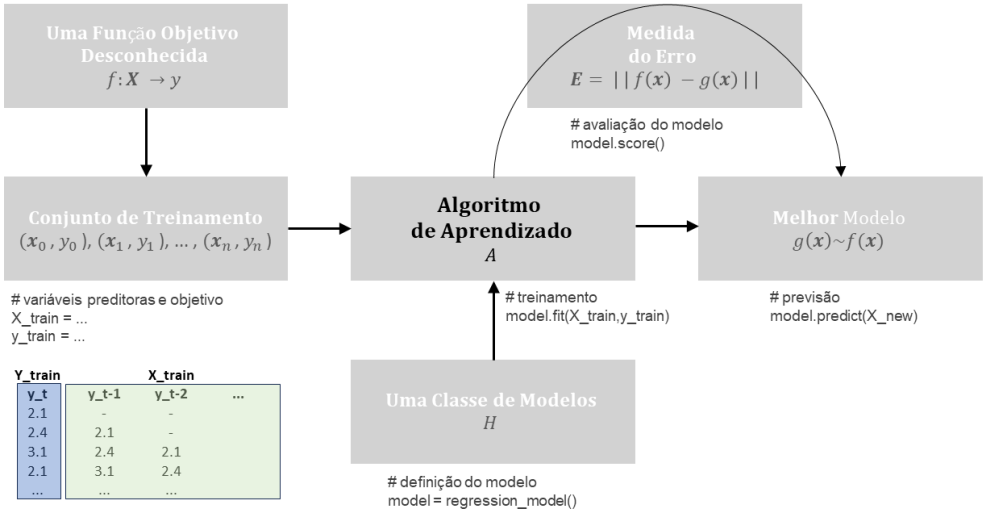


Figura 1.2. Esquema geral do aprendizado de máquina (regressão) e da engenharia de atributos para séries temporais.

Em todos os casos, dado uma classe de modelos que desejamos empregar para aproximar uma série, o modelo é ajustado até que se alcance um erro mínimo por alguma métrica (como o MSE). Diferentemente de um modelo estatístico, como o ARIMA, a aplicação do modelo de não requer nenhum pressuposto como estacionariedade, não sazonalidade ou independência e normalidade dos resíduos, sendo seu único objetivo aproximar ao máximo a série que se deseja modelar. Esse aspecto de caixa-preta ou de força-bruta, principalmente nos modelos de aprendizado profundo, é uma das principais críticas na aplicação desses tipos de modelo em séries temporais. Entretanto, eles podem obter resultados de previsibilidade bastante bons em casos muito complexos.

São inúmeras as classes de modelos e bibliotecas disponíveis. Mas sendo o esquema geral de aplicação do método o mesmo, vamos apresentar uma aplicação com modelos clássicos de aprendizado empregando o pacote scikit-learn e outra de aprendizado profundo, com TensorFlow/Keras, o que para os nossos propósitos parece ser suficiente.

1.4.1. Scikit-learn

O scikit-learn é a principal biblioteca de aprendizado de máquina de modelos clássicos (não profundo), incluindo modelos de aprendizado supervisionado (classificação, regressão) e não supervisionado (clusterização, detecção de anomalias, redução de dimensionalidade). Para predição de séries temporais empregamos modelos regressores e uma relação dos modelos disponíveis encontram-se a seguir.

```
# alguns dos estimadores de regressão disponíveis no scikit-learn
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR
from xgboost.sklearn import XGBRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.linear_model import BayesianRidge
from sklearn.linear_model import ElasticNet
from sklearn.kernel_ridge import KernelRidge
from sklearn.linear_model import SGDRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.neural_network import MLPRegressor
```

Cada um desses modelos tem um princípio, ou fundamento, mas que não entraremos em detalhe aqui. Assim, um estimador como MLPRegressor empregará uma rede neural para aproximar uma série, enquanto um DecisionTreeRegressor empregará uma árvore de decisão baseada na distribuição dos valores. Quaisquer desses estimadores podem ser igualmente aplicados.

1.4.1.1.Exemplo 1

Vamos empregar a série co2 utilizada antes. Já criamos os dados defasados da série para serem as variáveis preditoras dos modelos. Desse modo, faremos fazer o ajuste do modelo para,

$$(CO2_{t-1}, CO2_{t-2}) \rightarrow CO2_t$$

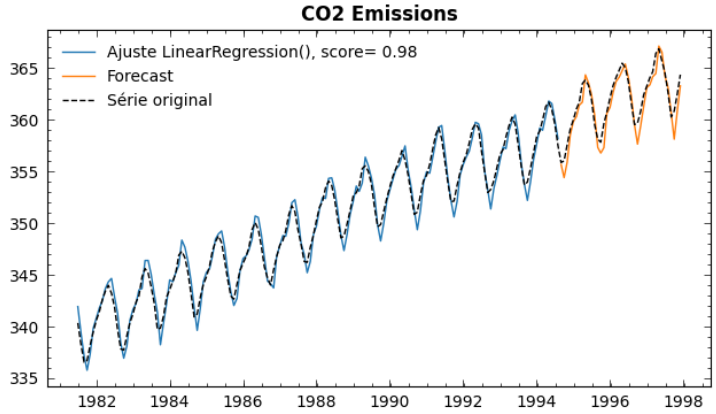
```
df = co2
df.head()
            time     CO2  CO2_t-1  CO2_t-2
1981-07-01     2  340.32   342.08   342.74
1981-08-01     3  338.26   340.32   342.08
1981-09-01     4  336.52   338.26   340.32
1981-10-01     5  336.68   336.52   338.26
1981-11-01     6  338.19   336.68   336.52
```

Um conceito importante na construção de modelos de aprendizado de máquina é o conceito de conjuntos de treinamento e teste. A ideia é separar dados empregados para o treinamento do modelo daqueles que são empregados para mensurar o seu desempenho. Há várias formas de fazer isso e, tipicamente, são empregados um percentual de dados aleatórios para treinamento (80% por exemplo) e teste (20%). Mas uma alternativa ao tratarmos de séries temporais considerarmos para teste os dados mais recentes (por exemplo, os 20% que correspondem aos dados mais recentes).

```
# conjuntos de treinamento e teste
train_size = int(len(df) * 0.80)
train_data, test_data = df.iloc[0:train_size], df.iloc[train_size:len(df)]
```

O código a seguir é um padrão para o uso de estimadores do scikit-learn e o estimador (modelo) empregado a seguir pode ser substituído por quaisquer dos outros regressores disponíveis.

```
X_train, y_train, X_test, y_test = train_data[['CO2_t-1','CO2_t-2']],
                                   train_data[['CO2']],
                                   test_data[['CO2_t-1','CO2_t-2']],
                                   test_data[['CO2']]

# modelo
model = LinearRegression()
# model = DecisionTreeRegressor() # try this!

# treinamento do modelo
model.fit(X_train, y_train)

# pedição com o modelo treinado
train_predict = model.predict(X_train)
test_predict = model.predict(X_test)

# Para o código completo ver o material complementar

_ = error_measures(y_test.values.flatten(), test_predict.flatten())
```

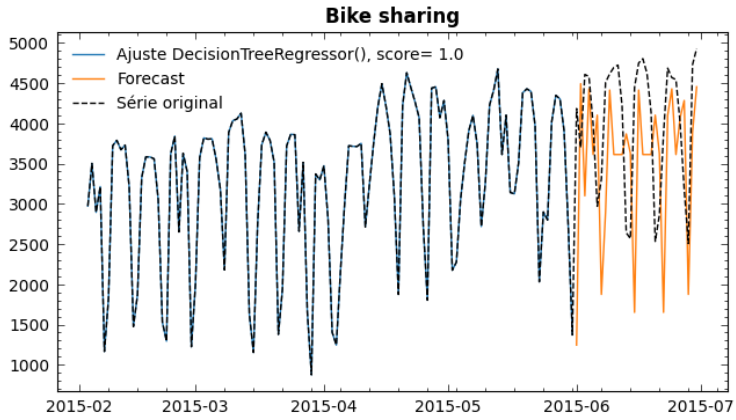



```
MSE:        1.1527
MAE:        0.8495
RMSE:       1.0737
MAPE:       0.0023
```

1.4.1.2. Exemplo 2

Um conjunto de dados mais complexo pode, entretanto, exigir um modelo mais robusto.

```
df = pd.read_csv(course_path + '/data/dados_bike_cnt.csv',
       index_col=0, parse_dates=True)

# valores defasados
df['cnt_t-1'] = df['cnt'].shift()
df['cnt_t-2'] = df['cnt'].shift(2)
df = df.dropna()
df.head()
             cnt  cnt_t-1  cnt_t-2
2015-02-03  2972   3556.0   1117.0
2015-02-04  3502   2972.0   3556.0
2015-02-05  2894   3502.0   2972.0
2015-02-06  3214   2894.0   3502.0
2015-02-07  1165   3214.0   2894.0

# conjuntos de treinamento e teste
train_size = int(len(df) * 0.80)
train_data, test_data = df.iloc[0:train_size], df.iloc[train_size:len(df)]

X_train, y_train, X_test, y_test = train_data[['cnt_t-1','cnt_t-2']], train_d
ata[['cnt']], test_data[['cnt_t-1','cnt_t-2']], test_data[['cnt']]

# modelo
model = DecisionTreeRegressor()

# treinamento do modelo
model.fit(X_train, y_train)

# predição com o modelo treinado
train_predict = model.predict(X_train)
test_predict = model.predict(X_test)

# Para o código completo ver o material complementar

_ = error_measures(y_test.values.flatten(), test_predict.flatten())
```




```
MSE:   1516080.7667
MAE:   1014.1000
RMSE:  1231.2923
MAPE:  0.2694
```

1.4.2. TensorFlow/Keras

O TensorFlow é uma biblioteca para a implementação de aprendizado profundo que permite criar modelos bastante complexos e uso de recursos de processamento em gpu, e o Keras fornece uma interface de mais alto nível para sua programação. Diferentes tipos de redes podem ser empregados e vamos empregar aqui dois modelos, um modelo de camadas sequenciais (ou rede multilayer perceptron) e um modelo LSTM (long short-term memory) que é uma arquitetura de rede neural recorrente bastante empregados em modelos de séries temporais. Modelos profundos vem tendo sucesso no tratamento de dados muitos complexos que envolvem grandes volumes de dados, grande ruído e variabilidade, além de séries múltiplas de dados.

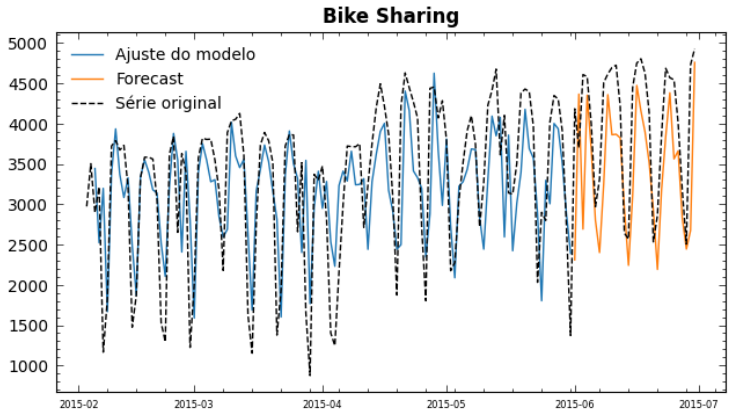
1.4.2.1. Exemplo 1

Empregamos os mesmos dados de Bike Sharing do exemplo anterior para buscarmos resultados melhores com um modelo neural. O uso de pacotes de aprendizado profundo é em geral mais complexo e o leitor pode buscar mais detalhes da programação envolvida aqui na documentação do Keras (2024).

Em um modelo neural é comum normalizarmos os dados (que são "desnormalizados" na previsão dos valores). O código a seguir implementa um modelo sequencial, mas que pode ser substituído pelo modelo recorrente LSTM (ver código completo no material complementar). Ambos os modelos apresentam um resultado melhor que o obtido pelo modelo clássico anterior.

```
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense, LSTM

cnt_original = df[['cnt']] # salva antes da normalização
df = df[['cnt']]

# normalizando os dados
scaler = MinMaxScaler(feature_range=(0, 1))
df['cnt'] = scaler.fit_transform(df)
```

```
# valores defasados normalizados
df['cnt_t-1'] = df['cnt'].shift()
df['cnt_t-2'] = df['cnt'].shift(2)
df = df.dropna()

# conjuntos de treinamento e teste
train_size = int(len(df) * 0.80)
train_data, test_data = df.iloc[0:train_size], df.iloc[train_size:len(df)]

X_train, y_train, X_test, y_test = train_data[['cnt_t-1','cnt_t-2']], train_d
ata[['cnt']], test_data[['cnt_t-1','cnt_t-2']], test_data[['cnt']]

# modelo MLP ou LSTM
model = Sequential()

# MLP, MAPE ~ 0.14
model.add(Dense(64, input_dim=X_train.shape[1], activation='relu'))
model.add(Dense(512, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mean_squared_error')

# treinamento do modelo
model.fit(X_train, y_train, epochs=50, batch_size=16, verbose=0)

# predição com o modelo treinado
train_predict = model.predict(X_train)
test_predict = model.predict(X_test)

# invertendo a normalização
train_predict = scaler.inverse_transform(train_predict)
test_predict = scaler.inverse_transform(test_predict)

y_test = scaler.inverse_transform(y_test)

# Para o código completo ver o material complementar
```

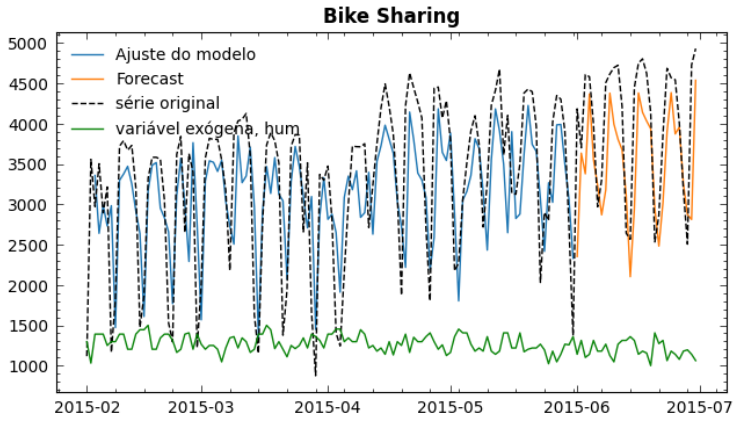


MSE: 751593.1959
MAE: 689.0012
RMSE: 866.9447
MAPE: 0.1678

1.4.2.2. Exemplo 2

Até aqui empregamos somente os valores da própria série para a previsão de novos valores. Entretanto nossa série pode ser influenciada por variáveis exógenas. O volume de vendas dependendo da taxa de inflação, o consumo de energia dependendo no volume de chuvas no período e assim por diante. No exemplo a seguir incluímos os dados de humidade para melhorar a nossa previsão do volume de compartilhamento de bicicletas, uma vez que para uma maior humidade do clima (chuva) é esperado uma menor procura por bicicletas. O resultado melhora a previsão anterior, embora um modelo com mais elementos seja necessário.

```
df_cnt_hum = pd.read_csv(course_path + '/data/dados_bike_cnt_hum.csv',
       index_col=0,parse_dates=True)
df_cnt_hum.head()
             cnt   hum
2015-02-01  1117  87.0
2015-02-02  3556  70.0
2015-02-03  2972  93.0
2015-02-04  3502  93.0
2015-02-05  2894  93.0

# Para o código completo ver o material complementar
```




```
MSE:   619213.9539
MAE:   618.7660
RMSE:  786.9015
MAPE:  0.1517
```

1.4.3. Exercícios e referências

Kong et al. (2020) e Vanderplas (2016) trazem uma introdução prática ao aprendizado de máquina clássico (não profundo) em Python com scikit-learn. Entretanto, assim como a maioria dos livros textos de aprendizado de máquina, o foco está em modelos de classificação e regressão de valores, e não em séries temporais. Se você quiser saber mais sobre implementações de aprendizado profundo Zhang et al. (2022) é um texto avançado, completo e online que traz exemplos em TensorFlow, PyTorch e MXNET. Nielsen (2019) traz modelos práticos de aprendizado clássico e profundo de séries em Python, mas os modelos de redes profundas empregam MXNET, bem menos empregado que as bibliotecas TensorFlow e PyTorch.

Embora tenham um esquema geral simples de implementação (engenharia dos atributos, ajuste do modelo e previsão), os possíveis modelos e suas implementações, sejam de aprendizado clássico ou profundo, são bastante variados e a consulta da

documentação das bibliotecas scikit-learn (2024), TensorFlow (2024) e Keras (2024) é essencial para entendimento dos exemplos aqui e para que você possa realizar as suas implementações.

Para esta seção, os exercícios e soluções podem ser acessados em exerc_parte3_ml.

1.5. Conclusão

Neste minicurso você pode entender alguns dos conceitos fundamentais de séries temporais e como lidar com séries temporais em Python e Pandas. Pode aprender como construir e analisar modelos estatísticos do tipo ARIMA (incluídos os modelos AR, MA, ARMA) com a biblioteca statsmodels, e a partir deles você poderá facilmente evoluir modelos mais complexos como o SARIMA, SARIMAX. Você também aprendeu como aplicar modelos de aprendizado de máquina clássicos e de aprendizado profundo com as bibliotecas scikit-learn e TensorFlow/Keras a partir da engenharia dos atributos da série, incluindo o uso de variáveis exógenas.

Esse poderá ser um ferramental bastante útil para você, na vida profissional ou para seus projetos de pesquisa, e a ferramenta certa - aprendizado de máquina, ou estatística clássica - e o modelo específico a ser utilizado, em última análise, dependerá do contexto mais amplo do seu problema.

Há muitos temas que não foram abordados aqui como intervalos de confiança, diversos outros modelos estatísticos, critérios de seleção de modelos, conceitos de ajuste no aprendizado de máquina ou validação cruzada etc. mas contamos que você encontrará aqui um ótimo ponto de partida para se aprofundar nesses pontos quando precisar.

Material complementar

Os códigos completos, exercícios, conjuntos de dados e outros materiais do curso podem ser acessados no site permanente *https://github.com/Introducao-Series-Temporais-em-Python/minicurso-SBC-SBSI-2024*.

Referências

Box, G. E., Jenkins, G. M., Reinsel, G. C., Ljung, G. M. (2015). Time series analysis: forecasting and control. John Wiley & Sons.

Chatfield, C. (1996) The analysis of time series: an introduction. Chapman and hall/CRC.

Coghlan, A. (2024) Welcome to a little book of R for time series. Parasite Genomics Group, Cambridge, U.K. https://a-little-book-of-r-for-time-series.readthedocs.io/en/latest/. Acesso em: 12 jan. 2024.

De Gooijer, J. G., Hyndman, R. J. (2006) 25 years of time series forecasting. International journal of forecasting, v. 22, n. 3, p. 443-473.

Greykite. Disponível em: https://linkedin.github.io/greykite/. Acesso em: 12 jan. 2024.

Hyndman, R. J., Athanasopoulos, G. (2018) Forecasting: principles and practice.

Keras. Disponível em: https://keras.io/. Acesso em: 12 jan. 2024.

Kong, Q., Siauw, T., Bayen, A. (2020) Python Programming and Numerical Methods: A Guide for Engineers and Scientists. Academic Press, 2020. Disponível em: https://pythonnumericalmethods.berkeley.edu/notebooks/Index.html. Acesso em: 12 jan. 2024.

Montgomery, D. C., Jennings, C. L.; Kulahci, M. (2015) Introduction to time series analysis and forecasting. John Wiley & Sons.

Morettin, P. A., Toloi, C. (2006) Análise de séries temporais. In: Análise de séries temporais.

Nielsen, A. (2019) Practical time series analysis: Prediction with statistics and machine learning. O'Reilly Media.

Oliveira, R. (2022). Visualizacao de Dados em Python. ISBN: 978-65-5545-511-3. Coleção Conexão Inicial. Editora Mackenzie.

Oliveira, R., Albarracin, O. Y. E., Silva, G. R. (2024) Introdução às Séries Temporais: Uma Abordagem Prática em Python (in printing). Coleção Conexão Inicial. Editora Mackenzie. Disponível em: https://github.com/Introducao-Series-Temporais-em-Python Acesso em: 12 jan. 2024.

Pandas. Disponível em: https://pandas.pydata.org/. Acesso em: 12 jan. 2024.

Peixeiro, M. (2022) Time Series Forecasting in Python.

Prophet. Disponível em: https://facebook.github.io/prophet/. Acesso em: 12 jan. 2024.

scikit-learn. Disponível em: https://scikit-learn.org/stable/. Acesso em: 12 jan. 2024.

Shmueli, G., Lichtendahl Jr. K. C. (2016) Practical time series forecasting with r: A hands-on guide. Axelrod schnall publishers.

statsmodels. Disponível em: https://www.statsmodels.org/stable/index.html. Acesso em: 12 jan. 2024.

TensorFlow. Disponível em: https://www.tensorflow.org/. Acesso em: 12 jan. 2024.

Vanderplas, J. (2016) Python data science handbook: Essential tools for working with data. O'Reilly Media, Inc., 2016. Disponível em: https://jakevdp.github.io/PythonDataScienceHandbook/ Acesso em: 12 jan. 2024.

Zhang, A. et al. (2021) Dive into deep learning. arXiv preprint arXiv:2106.11342, 2021. Disponível em: https://d2l.ai/. Acesso em: 12 jan. 2024.

Capítulo 2

UX e Linguagem Simples na Web: Práticas para um Design de Interação mais Compreensível

UX and Plain Language in Web: Practices for More Understandable Interaction Design

Rodrigo Oliveira e Claudia Cappelli

Abstract

This theoretical-practical short course aims to present the main practices of Plain Language aimed at improving the understanding of textual and visual elements in interfaces, websites, applications and other interactive products. During the course, the importance of this language, its history and foundations for clear communication, accessibility and efficiency in interaction will be explored. Participants will be invited to exercise the practices and learn about automation tools. In the end, they will have acquired skills to apply Plain Language in their projects and will be able to pass on their knowledge on the subject..

Resumo

Este Minicurso teórico-prático tem como objetivo apresentar as principais práticas da Linguagem Simples voltada para melhoria do entendimento de elementos textuais e visuais em interfaces, websites, aplicativos e outros produtos interativos. Durante o curso, será explorada a importância dessa linguagem, seu histórico e fundamentos para a clareza de comunicação, acessibilidade e eficiência na interação. Os participantes serão convidados a exercitar as práticas e conhecer ferramentas de automação. Ao final, terão adquirido habilidades para aplicar a Linguagem Simples em seus projetos e poderão repassar seus conhecimentos no assunto.

2.1. Introdução

No cenário digital contemporâneo, onde a acessibilidade e a experiência do usuário são essenciais, o design de interação emerge como um elemento crucial para garantir a compreensibilidade e a inclusão na web. A complexidade inerente à tecnologia digital muitas vezes representa uma barreira para a acessibilidade, limitando a compreensão eficaz das informações disponíveis na internet. A interseção entre UX e Linguagem Simples surge como uma abordagem inovadora para superar esses desafios, visando tornar a experiência online mais acessível para um público diversificado, incluindo

pessoas com diferentes níveis de habilidade, conhecimento e familiaridade com a tecnologia. Linguagem Simples é uma técnica que apresenta, o texto, a estrutura e o design da informação tão claros que o leitor consiga encontrar facilmente o que procura, compreender o que encontrou e usar essa informação (Plain, 2024). Ela contribui diretamente para a transparência da informação, um princípio essencial em governos e organizações que buscam evitar a obscuridade das informações apresentadas que dificultam o entendimento e limitam o acesso às informações vitais, além de promover a confiança e a participação do seu público.

Neste contexto, este trabalho oferece um minicurso de caráter introdutório com abordagem teórico-prática onde serão apresentadas as principais práticas e diretrizes desta tradução intralinguística (Jakobson, 2013). O conteúdo será apresentado a partir de exemplos reais de projetos de pesquisa, cases de sucesso do mercado e também com a demonstração de ferramentas de apoio desenvolvidas em pesquisas acadêmicas recentes, visando automatizar as práticas da Linguagem Simples e aumentar sua aderência aos diversos tipos de apresentação de informação, como textos, interfaces de sistemas, gráficos, fluxos de processo, entre outros. Em seguida, aplicaremos os conceitos apresentados em portais de dados abertos na web na forma de exercícios de avaliação destas práticas utilizando um *checklist* de verificação. Assim os participantes poderão conferir na prática como a utilização da linguagem simples pode apoiar a transparência da informação e a melhoria da experiência do usuário (UX).

Ao longo deste artigo, serão exploradas práticas eficazes de design de interação, com foco especial na otimização da usabilidade por meio de interfaces intuitivas e na aplicação da Linguagem Simples para aprimorar a compreensão de conteúdos complexos. Além disso, será discutido o impacto positivo dessas práticas na promoção da inclusão digital e na garantia de uma experiência online mais acessível e agradável para todos os usuários.A proposta é se ter uma análise aprofundada das práticas de User Experience (UX) aliadas à utilização de Linguagem Simples como estratégias fundamentais para a promoção de um design de interação mais compreensível na web. Este trabalho busca, assim, contribuir para o avanço do conhecimento na área de UX, fornecendo insights valiosos para designers, desenvolvedores e pesquisadores interessados em criar interfaces digitais que transcendam barreiras cognitivas e promovam a democratização do acesso à informação na era digital.

Este capítulo está organizado da seguinte forma: A Seção 1.2 apresenta os fundamentos teóricos do tema. Em seguida destacamos as principais práticas gerais da linguagem oriundas do *Federal Plain Language Guidelines* na Seção 1.3. Posteriormente, apresentamos práticas para interfaces da web na Seção 1.4. Na Seção 1.5 listamos algumas ferramentas úteis para aplicação da Linguagem Simples e concluímos com a Seção 1.6, seguido das referências utilizadas neste capítulo.

2.2. Fundamentação Teórica

2.2.1. Entendimento

Esta característica na definição de transparência de Cappelli (2009) se apresenta como um dos pontos mais importantes no trato com o cidadão. Apesar de diversas leis se preocuparem bastante com dar acesso, e garantir o uso, além claro, de que os dados

tenham qualidade, não se percebe ainda um estímulo a garantir que as informações apresentadas sejam de fato entendidas pela sociedade. O que nos leva a outro problema dado que se o cidadão não entende ele não faz uso e consequentemente não participa de discussões e decisões. O nível do entendimento é uma característica, que segundo Cappelli e Leite (2008) se refere à capacidade de alcançar o significado e o sentido. Ainda que os demais aspectos sejam necessários para o cidadão ter transparência de informações, o entendimento das informações é que fará com que ele de fato se aproprie das mesmas e consiga fazer uso. Para isso precisamos cada vez mais de mecanismos que propiciem este entendimento e a técnica da Linguagem Simples é um destes.

2.2.2. Linguagem Simples

Linguagem simples é uma das traduções do termo em inglês "Plain Language". Podemos ainda traduzir como Linguagem Clara ou Linguagem Cidadã. O conceito surgiu devido à necessidade de facilitar o entendimento da informação, principalmente aquelas que as organizações apresentam à sociedade. Na década de 1970, um movimento mundial foi iniciado de modo a consolidar orientações para a escrita e a organização visual da informação. Desde então, a prática de Linguagem Simples se disseminou por diversos países. Ao longo deste tempo também nasceram algumas associações e uma Federação que criaram juntas o *Federal Plain Language Guidelines*, que contém um corpo de conhecimento sobre as melhores práticas indicadas para a aplicação da Linguagem Simples. Neste são apresentadas diretrizes e recomendações para apoiar uma escrita com mais clareza e objetividade. Esta Federação definiu também três princípios para esta Linguagem, indicando que uma comunicação está em linguagem simples quando a escrita, a organização, o design e todo e qualquer outro elemento utilizado para apresentar informação sejam tão claros que o público-alvo consiga encontrar facilmente o que procura, compreender o que encontrou e usar essa informação. Seguimos com essa declaração pela sua abrangência, conceituando o objetivo da linguagem na comunicação de forma ampla, não apenas textual. Apesar da escrita ser o principal meio de aplicação da técnica.

A lei da redação simples (*Plain Writing Act*) nos Estados Unidos sancionada em 2010 (Gov Info, 2010) estabelecendo que os documentos governamentais emitidos ao público fossem redigidos de forma clara e a política de comunicação do governo canadense que institui também o uso da Linguagem Simples são iniciativas proeminentes na parte norte-americana. As ações oficiais da União Europeia são encontradas no serviço de publicações, como a distribuição de manuais de como escrever claramente. Na América Latina o governo colombiano desponta com um Programa Nacional de Serviços ao Cidadão desde 2013, estabelecendo a Linguagem Clara como uma das prioridades da administração pública. Guias, métodos e seminários são divulgados e já foram traduzidos mais de 150 documentos de alto impacto.

No Brasil temos referências explícitas à aplicação da Linguagem Simples em legislações e uma política nacional está em tramitação no Congresso. A Linguagem Simples é projetada para comunicar precisamente as informações para o público pretendido, com documentos claros e concisos (Barboza, 2010). A organização norte-americana FDA (*Food and Drug Administration*) apresenta muitos benefícios que a experiência com a Linguagem Simples oferece, entre eles: (i) Melhorar a compreensão

pública das comunicações do governo; (ii) Economizar dinheiro e aumentar a eficiência; (iii) Reduzir a necessidade de esclarecimento público; (iv) Diminuir os recursos gastos com fiscalização; (v) Minimizar erros cometidos e o esforço para corrigi-los.

Apesar de toda a sua contribuição, ainda existem muitas oportunidades de trabalho a partir deste conhecimento disponibilizado. Quanto à dinâmica escrita, a Linguagem Simples oferece métodos para melhorar a inteligibilidade, entretanto no carácter visual e interativo ainda pode ser bastante aprofundada. A partir do que diz a própria definição da linguagem, que além da escrita do texto, a estrutura e o design também devem se apresentar de forma melhor compreensível, pouco se encontra no corpo da própria linguagem sobre o uso de elementos visuais. Estes outros recursos são muito importantes por serem amplamente integrados nas interfaces de sistemas, justamente pela sua fácil compreensão e carácter atrativo.

2.2.3. Principais Ações em Linguagem Simples

A relevância da Linguagem Simples é demonstrada pelas inúmeras iniciativas, públicas e privadas, com intuito de fomentar e aplicar o uso desta linguagem no Brasil e no Mundo. Destacam-se:

Ações Internacionais

- Norma ISO já publicada para estabelecer os primeiros princípios normativos da Linguagem Simples oficialmente (ISO, 2023).

- Plain Language Association International[1] (PLAIN) é uma organização canadense sem fins lucrativos que reúne defensores da linguagem simples e profissionais de todo o mundo em mais de 30 países que trabalham em comunicação clara em pelo menos 15 idiomas.

- Clarity International[2] é uma organização britânica sem fins lucrativos que reúne profissionais empenhados em promover uma linguagem jurídica simples. Possui representantes oficiais em cerca de 50 países que defendem o uso de linguagem jurídica simples no lugar do “juridiquês”.

- Center for Plain Language[3] é uma organização norte-americana composta por profissionais de diversas áreas como acadêmicos, consultores, organizações de saúde e a comunidade de negócios com a missão de criar uma cultura de clareza onde com uma comunicação clara as pessoas e organizações possam prosperar.

- International Plain Language Federation[4] coordena as ações conjuntas das três organizações citadas anteriormente para promover o uso e a prática da linguagem simples.

[1] https://plainlanguagenetwork.org/
[2] http://www.clarity-international.org/
[3] https://centerforplainlanguage.org/
[4] https://www.iplfederation.org/

- The Plain Language Action and Information Network[5] é uma plataforma mantida por funcionários do governo norte-americano. Seu objetivo é difundir o uso da linguagem simples em agências e órgãos estatais.

- Plain Language Comission[6] é uma organização britânica que oferece serviços de edição de textos em linguagem simples e cursos no tema. Criou a certificação Clear English Standard.

- Plain English Foundation[7] é uma fundação australiana que oferece consultorias e treinamentos em linguagem simples.

- Red de Lenguaje Claro[8] é uma organização chilena que reúne 7 instituições públicas visando trabalhar em conjunto na implementação de ações que visem gerar iniciativas, projetos e medidas que promovam, divulguem e facilitem o uso da linguagem clara.

Ações Nacionais

- Guia de Edição de Serviços do Gov.br[9] é um guia desenvolvido pelo Governo Federal com diretrizes e recomendações para o uso de linguagem simples para editores do portal do governo federal, auxiliando a manter os textos mais claros e úteis para os usuários desses serviços.

- O Laboratório Interdisciplinar de Linguagem Cidadã[10] sediado no Rio de Janeiro, congrega pesquisadores atuando em pesquisa teórica e aplicada sobre linguagem cidadã.

- O Programa Municipal de Linguagem Simples da prefeitura de São Paulo[11], criado em 2019, visando simplificar a linguagem que a Prefeitura de São Paulo usa na comunicação com a população.

- As orientações para Adoção de Linguagem Clara no Portal Governo Aberto SP[12], criadas em 2016, oferecem um guia que é parte integrante do projeto de cooperação entre o Governo do Estado de São Paulo e o Reino Unido.

- O Laboratório de Inovação e Dados do Governo do Estado do Ceará[13] (Iris Lab Gov) foi criado com a proposta de fomentar e ampliar a inovação no setor público que tem como um de objetos de pesquisa e investimento a discussão e implantação da linguagem simples no Estado.

[5] https://www.plainlanguage.gov/
[6] https://www.clearest.co.uk/
[7] https://www.clearest.co.uk/about
[8] http://www.lenguajeclarochile.cl/
[9] https://www.gov.br/pt-br/guia-de-edicao-de-servicos-do-gov.br/escrevendo-para-o-seu-usuario
[10] https://linclab.com.br/
[11] https://011lab.prefeitura.sp.gov.br/linguagem-simples/inicio
[12]http://www.governoaberto.sp.gov.br/wp-content/uploads/2017/12/orientacoes_para_adocao_linguagem_clara_ptBR.pdf
[13] https://www.egp.ce.gov.br/irislabgov/

- Linguagem Simples Lab[14], criado em fevereiro de 2023 visando melhorar a comunicação pública por meio da Linguagem Simples e soma mais de 310 integrantes de órgãos federais, estaduais e municipais, laboratórios de inovação e da sociedade civil.
- Rede Linguagem Simples Brasil[15], se propõe a ser um espaço público de debate, fomento e construção em torno da Linguagem Simples. Ele foi cocriado a partir de experiências de servidores em governos e suas interações com a sociedade civil.
- ABEP-TIC (Associação Brasileira de Entidades Estaduais de Tecnologia de Informação e Comunicação) que congrega todos os órgãos administradores dos portais de serviços públicos dos estados do Brasil, desenvolveu um Guia de Linguagem Simples, incentivando a sua aplicação em todos os estados (ABEPTIC, 2022) como incluirá um índice específico de avaliação de linguagem simples na oferta de serviços digitais dos estados[16].

Leis Nacionais

- Lei de Acesso à Informação (Lei 12.527 de 18 de novembro de 2011), determina que todo órgão e entidade pública ofereça o acesso às suas informações utilizando de procedimentos objetivos, ágeis, transparentes, claros e em linguagem de fácil compreensão.
- Lei do Governo Digital (Lei 14.129 de 29 de março de 2021) dispõe sobre princípios, regras e instrumentos para o Governo Digital e para o aumento da eficiência pública.
- Lei 13.460 de 26 de junho de 2017 dispõe sobre participação, proteção e defesa dos direitos do usuário dos serviços públicos da administração pública. Esta lei regulamenta a Carta de Serviços, onde cada município deverá elaborar a sua carta com informações claras e precisas, tendo que dispor em locais de fácil acesso, tais como o portal institucional e o local onde os serviços serão prestados.
- Projeto de Lei (PL 6256/2019) que Institui a Política Nacional de Linguagem Simples nos órgãos e entidades da administração pública direta e indireta. Atualmente está Aguardando Apreciação pelo Senado Federal[17].

2.2.4. Experiência do Usuário (UX) e Design de Interfaces (UI)

Conforme a norma ISO (2019) sobre design centrado no ser humano para sistemas interativos, a experiência do usuário, comumente abreviada por UX (User Experience), é o conjunto de percepções e respostas do usuário no uso de um sistema, produto ou serviço. Ou seja, ela está intimamente ligada às percepções e respostas dos usuários,

[14] https://www.linkedin.com/in/linguagemsimpleslab/
[15] https://www.linkedin.com/company/rede-linguagem-simples-brasil/
[16] https://www.jornaldaabep.com.br/indice-de-linguagem-simples
[17] https://www.camara.leg.br/proposicoesWeb/fichadetramitacao?idProposicao=2231632

incluindo as emoções, crenças, preferências, percepções e comportamentos que ocorrem antes, durante e após o uso. Segundo Don Norman e Jacob Nielsen[18], pioneiros na área, é importante também distinguir UX de usabilidade, um dos conceitos mais comuns quando o assunto é sistemas interativos. A usabilidade é um atributo de qualidade de uma interface, abrangendo se o sistema é fácil de aprender, eficiente de usar, etc. Enquanto para UX temos diversos outros atributos de qualidade relevantes. Peter Morville (2004) criou um diagrama conhecido como "Honeycomb" (favo de mel) de UX ilustrado na Figura 1.1. Nele há as facetas principais da experiência do usuário como: desejável, acessível, valioso, útil, etc. O diagrama é especialmente útil para entender por que precisamos pensar além da usabilidade.

Figura 1.1. User Experience Honeycomb. Fonte: Morville, 2004.

Ainda para Norman e Nielsen, para alcançarmos uma experiência de usuário de alta qualidade, são necessárias várias competências de diversas áreas como engenharia, marketing, design gráfico, industrial e especialmente design de interface (UI). A área sobre UI, sigla para *User Interface* ou interface do usuário, propriamente dita, surgiu a partir das interfaces gráficas em computadores pessoais. Ela é totalmente baseada no paradigma de interações a partir de ícones, menus, ponteiros do mouse e mais atualmente das telas sensíveis ao toque e comandos de voz (Bowman, 2014). O design, ou a construção destas interfaces é o processo que os designers usam para construir interfaces em software ou dispositivos computadorizados, visando contribuir com uma melhor experiência (UX) (IxDF, 2016a). Portanto, uma boa interface web, considerando o caso das interfaces acessadas via internet, precisa atingir alguns destes elementos importantes já citados. Ela precisa permitir ao usuário uma jornada adequada às suas necessidades, precisa fornecer um contexto claro e fazer com que o usuário consiga realizar ações e obter o melhor resultado. Isso está diretamente relacionado com os três princípios da Linguagem Simples: Encontrar o que precisa, compreender o que encontrou e conseguir usar as informações para atingir seus objetivos. Para isso, apresentaremos abaixo um conjunto de práticas que podem ajudar a construir melhores interfaces. Elas já atendem às práticas propostas pela Linguagem Simples que compartilham as características que uma boa interface deve ter.

[18] https://www.nngroup.com/articles/definition-user-experience/

2.3. Práticas Gerais de Linguagem Simples

Dado que o objetivo principal da Linguagem Simples é facilitar a compreensão, a localização e o uso das informações, podemos dizer que certamente ela é base para a transparência e consequentemente o ponto inicial para a participação de todos. Para isso faz-se necessário que as organizações, principalmente as da esfera pública, estejam preparadas para o uso desta linguagem. Textos com jargões e termos técnicos, elaborados de maneira complexa e com palavras de difícil entendimento apenas geram desinteresse e confusão. Deve-se considerar que uma Linguagem Simples também torna a comunicação mais inclusiva, promovendo a acessibilidade. Para entender melhor as práticas da Linguagem Simples e poder aplicá-las é importante ter domínio do conteúdo apresentado no *Federal Plain Language Guidelines*. Este guia, como já citado anteriormente, está sob a tutela de uma federação, sendo organizado e mantido por organizações ligadas a esta federação. Este guia é estruturado em 5 grandes blocos de conteúdo como apresentado na Figura 1.2.

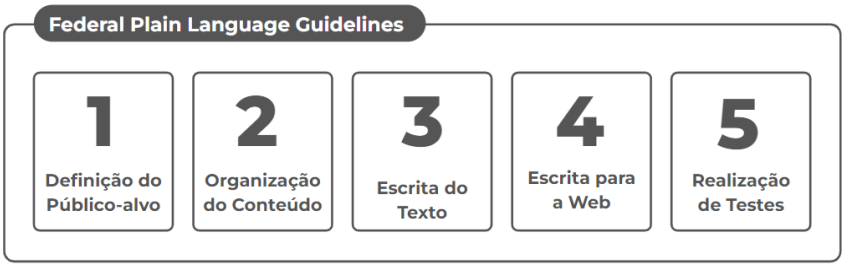


Figura 1.2. Conteúdo do *Federal Plain Language Guidelines.* Fonte: Os Autores.

2.3.1. Definição do Público-alvo

Esta prática indica que devemos buscar conhecer o público alvo da informação para quem estamos tentando transmitir, entender o quanto ele já sabe sobre o tema, o que mais ele precisa saber e que perguntas ele faria sobre aquele conjunto de informações. Além disso, é importante pensar qual será o melhor resultado a ser apresentado e como devemos nos expressar para apresentar este resultado dessa forma. Esta prática se divide em duas diretrizes: como escrever para sua audiência e como tratar diferentes audiências.

2.3.1.1. Como Escrever para sua Audiência

A técnica de definir Personas pode ajudar neste caso. Persona é a representação fictícia do seu cliente típico. Ela é baseada em: dados reais sobre o comportamento, características demográficas, histórias pessoais, motivações, objetivos, desafios e preocupações. A Figura 1.3 apresenta um exemplo e a Tabela 1 apresenta um exemplo de uso da Linguagem Simples adaptada a uma possível persona definida.

Figura 1.3. Exemplo de Persona. Fonte: ABEPTIC, 2022

2.3.1.2. Tratar diferentes audiências

A depender da quantidade de personas definidas como público-alvo da sua informação, pode ser importante abordar públicos diferentes, separadamente. Muitos conteúdos abordam mais de um público. Conjuntos de informação que misturam elementos destinados a públicos diferentes podem confundir os leitores, pois, ao abordar diferentes públicos no mesmo lugar, você dificulta para cada público encontrar o material que se aplica a eles, dificultando que cada público possa cumprir os requisitos para sua necessidade. Neste caso, o mais indicado é que se usem formas de apresentação de informação ou até mesmo conteúdos diferenciados para as personas ou grupos de personas definidos.

Um bom exemplo disso são as interfaces bancárias que permitem ao cliente customizar os itens principais que desejam que apareçam como principais opções. A customização de interface pelo próprio cliente é uma técnica que permite a divulgação de informação para públicos diferentes numa mesma interface.

2.3.2. Organização do Conteúdo

Organizar conteúdos na maioria das vezes é a chave para o entendimento. Em geral, lemos documentos, analisamos informações ou visitamos sites para obter respostas para nossos problemas ou para buscar informações para atender nossas necessidades. Queremos saber como fazer algo ou o que acontece se não fizermos nada. Também queremos obter esse conhecimento o mais rápido possível. Para tal o *Federal Plain Language Guidelines* sugere algumas práticas para fazer melhor esta organização. Ele aborda 4 diretrizes: atender às necessidades da audiência, falar para uma pessoa e não para um grupo, usar títulos úteis e escrever seções curtas.

2.3.2.1. Atender às necessidades da audiência

Para todo conteúdo que se queira transmitir é importante começar sempre declarando a finalidade do documento e seus resultados. Deve-se eliminar qualquer preenchimento e conteúdo desnecessário e atentar para colocar as informações mais importantes no início da apresentação. Informações básicas (quando necessário) devem ser colocadas no final. Outro ponto bastante importante é que o seu conteúdo deve estar organizado para responder às perguntas que o seu público provavelmente fará. Busque estruturar na ordem em que ele as faria.

Para conjuntos de informação, mas complexos, podem ser criados índices que facilitem a navegação ou a busca direta. No caso de sites, os menus podem também ser um grande aliado para isso. Estes mecanismos ajudam os usuários a seguir e encontrar rapidamente as informações de que precisam. Porém, não crie muitos níveis de hierarquia. Em interfaces web podemos seguir a lei de Miller, um pesquisador da área da psicologia e ciência cognitiva que determinou que as pessoas processam até sete porções de informação a cada vez, variando de mais ou menos duas (Yablonski, 2020).

No caso de um processo ou um serviço, pode-se apresentar as etapas dos mesmos na ordem em vão ser usadas ou executadas. Isso não só facilita o entendimento como também guia o usuário diminuindo a incidência de erros. Já considerando um conteúdo que possui informações gerais e específicas, coloque as informações gerais em primeiro lugar e as especializadas ou exceções depois. O que se dirige à maioria dos leitores na maioria das situações deve vir em primeiro lugar.

2.3.2.2. Fale para uma pessoa, não para um grupo

Um fator bastante importante é falar com quem está lendo a informação. Mesmo que sua comunicação vá atingir um milhão de pessoas, lembre que você está falando com a única pessoa que o está lendo. Não dar atenção a isso pode gerar confusão se uma determinada ação que está sendo descrita deve ser realizada pela pessoa que lê ou por um grupo. Uma boa prática bastante recomendada é usar “você” para se dirigir ao usuário. O uso do “ele ou ela” pode fazer com que o leitor não perceba que ele é o agente da ação. A Tabela 1 ilustra como fazer uso desta prática.

Tabela 1. Exemplo de uso de fala direta

Texto Comum	Texto com Linguagem Simples
Os residentes do Rio de Janeiro que buscam assistência no processo de preenchimento e envio de sua inscrição podem entrar em contato com a agência usando o número de telefone listado abaixo.	Se você precisar de ajuda com sua inscrição, ligue para a agência usando o número abaixo.

2.3.2.3. Use Títulos úteis

Uma maneira eficaz de facilitar o entendimento de um conjunto de informações em um site é usar títulos autoexplicativos e úteis. Existem algumas categorias de títulos:

- ***Título de perguntas***: Muito úteis, mas é necessário perceber quais perguntas o seu público fará. Se essas perguntas são possivelmente conhecidas, use-as como títulos. Elas ajudarão o público a encontrar rapidamente as informações que procuram. Um bom exemplo disso são os FAQs (*Frequently Asked Questions*) que atualmente a maioria das organizações disponibiliza em seus sites reunindo as perguntas mais frequentes feitas e as respostas das mesmas.
- ***Título de declarações***: Define em poucas palavras o que se segue no conteúdo. É sempre uma ótima escolha porque resume bem o que segue.
- ***Título de Tópicos***: São do tipo "Geral", "Aplicativo" e "Escopo". Devemos evitar sempre que possível, pois são vagos e não ajudam a saber o que tem de fato no que segue.

Estruturar primeiramente os títulos, seja de um texto, um conjunto de dados ou de um menu pode nos ajudar a perceber a estrutura, verificar se não faltou nada e também nos mostrar possíveis agrupamentos. Em todos os casos citados, os títulos nunca devem ser muito longos. Frases com até 5 palavras podem ser uma boa prática.

2.3.2.4. Escrever seções curtas

Seções curtas dão a sensação de que entenderemos de maneira mais fácil. Seções densas e longas são visualmente desagradáveis e dão a impressão de que a informação é difícil de entender. Para darmos títulos, uma seção curta também nos ajuda, por ter apenas um assunto em geral, o que não acontece com seções longas, por serem mais difíceis de resumir. Muitas vezes tratam de mais de um assunto. A Tabela 2 mostra um exemplo.

Tabela 2. Exemplo de seções curtas

Texto Comum	Texto com Linguagem Simples
O programa tem uma linha telefônica Punjabi em 604-877-6163 na qual você pode deixar uma mensagem de correio de voz para alguém retornar sua chamada (as chamadas só são retornadas às sextas-feiras entre 10h00 e 12h00). Além disso, atendentes de reservas que falam chinês geralmente estão disponíveis pelo telefone 1-800-663-9203. Se não houver um disponível para atender sua chamada, você será questionado se alguém pode traduzir em seu nome ou para ligar de volta quando houver assistência de tradução disponível. Você também pode pedir a um membro da família ou amigo para ajudá-lo com a ligação.	O programa tem uma linha telefônica Punjabi em 604-877-6163 que você pode deixar uma mensagem de voz para alguém retornar sua chamada às sextas-feiras entre 10h00 e 12h00. Atendentes de reservas que falam chinês estão disponíveis pelo telefone 1-800-663-9203. Se não houver um disponível para atender sua chamada, será indicado ligar de volta quando houver tradução disponível. Você pode pedir a um membro da família ou amigo para ajudá-lo com a ligação.

2.3.3. Escrita do Texto

A escrita do conteúdo é a terceira prática fundamental. A escolha cuidadosa das palavras, os verbos que devem dizer exatamente o que será feito, o cuidado com as abreviações fazem parte desta prática. As diretrizes de escrita do texto na Linguagem Simples está direcionada para palavras, sentenças e parágrafos.

2.3.3.1. Palavras

A escolha das palavras na escrita do texto é algo muito importante. Podemos dizer que elas são o bloco mais básico do texto. Algumas características devem ser observadas quando estamos escolhendo palavras:

- Concisão: Escolha sempre o menor número de palavras possível para dizer o mesmo.
- Precisão: Escolha sempre a melhor palavra para transmitir uma ideia. Buscar sinônimos é sempre uma ótima prática.
- Simplicidade: Se tiver que escolher entre mais de uma palavra com o mesmo significado, escolha a mais simples, a que tem mais oportunidades de ser entendida pela sua audiência.
- Domínio comum: Escolha sempre as palavras de domínio da maior parte do seu público-alvo.

Outra prática que traz bons resultados é usar a voz ativa. A voz ativa é a voz verbal que indica que o sujeito da oração prática a ação. E na Linguagem Simples sempre indicamos falar diretamente com quem é o responsável pela ação. Assim sendo, ela será sempre a mais indicada. Quanto aos verbos, use sempre as formas mais simples. O presente do indicativo é o tempo verbal mais recomendado. Verbos ocultos não devem ser utilizados. Em geral, aparecem convertidos em substantivos, dificultando o entendimento do texto. Além disso, acaba obrigando que exista um verbo extra para completar o sentido da frase. Inca-se então que a conversão de verbos em substantivos seja minimizada o máximo possível.

Outro ponto importante é evitar jargões ou termos técnicos, ou estrangeiros, quando existirem outras palavras com mesmo significado que possam substituí-los. Sempre é melhor usar palavras conhecidas da audiência. É importante também evitar ou reduzir ao máximo o uso de abreviações. Muitas vezes para poder entendê-las o leitor terá que repetidamente recorrer a outro local do texto ou da imagem onde esta está explicada. Neste ponto podemos falar também da boa prática de evitar apresentar definições. Em geral, elas são apresentadas no meio do texto, gerando uma quebra de continuidade da ideia que está sendo apresentada. O uso de palavras curtas também é recomendado. Se tiver que escolher entre duas conhecidas pela audiência e que tenham o mesmo significado, opte pela menor. Além destas práticas citadas na estrutura das palavras, recomenda-se usar sempre o mesmo termo para se referir a um mesmo objeto. Neste caso podemos exemplificar o termo “patrão” e “empregador” sendo sinônimos, mas que para um público mais leigo pode levar a uma interpretação de que o primeiro tem relação com emprego informal e o segundo com emprego formal. A Tabela 3 nos mostra alguns exemplos.

Tabela 3. Alguns exemplos de uso de palavras

Texto Comum	Texto com Linguagem Simples
Os agendamentos podem ser feitos em diversos locais em todo o município.	Você pode fazer a marcação em qualquer local da cidade.

Se o paciente sentir dor no peito, suor e falta de ar, deve ligar imediatamente para 190.	Se você sentir dor no peito, suor e falta de ar, ligue para 190.
Ao final será solicitado que o cliente dê seu feedback sobre o serviço.	Ao final pedimos a você que informe seu grau de satisfação com o serviço.

2.3.3.2. Sentenças

Escolha suas palavras e as organize com cuidado. Comece sempre com a ideia principal. As exceções devem ser tratadas depois. Coloque suas palavras com cuidado, procurando ser sempre breve. Não é ruim usar muitos pontos. É melhor escrever frases curtas com uma única ideia, do que longas com várias ideias nela. Isso acabará levando para outra boa prática de escrever sentenças também curtas.

A ordem natural das palavras de uma frase é sujeito-verbo-objeto. Busque sempre escrever desta forma. Sempre procure escrever frases completas. Neste caso de estrutura de frases, outro ponto importante são as negativas. Elas devem sempre ser evitadas. Estamos acostumados a pensar e falar positivamente. Negativas tendem a nos confundir. No caso do uso de duplas negativas e exceções, isso fica ainda um pouco mais complexo para o entendimento. Quando escrevemos no negativo, colocamos um empecilho no caminho do entendimento do público-alvo. Quando você escreve uma frase contendo dois negativos, eles se cancelam, ou seja, na verdade, ela seria uma afirmativa. A melhor prática é sempre usar a frase afirmativa, colocando a ideia principal no início e as exceções depois. A Tabela 4 nos mostra alguns exemplos.

Tabela 4. Alguns exemplos de sentenças

Texto Comum	Texto com Linguagem Simples
O treinamento, que contou com a participação de 60 funcionários e obteve índices de aprovação acima de 95%, foi um grande sucesso.	O treinamento foi bem-sucedido: 60 funcionários compareceram e deram índices de aprovação de mais de 95%.
O recurso ao Tribunal de Primeira Instância, embora organizado de forma ineficaz, foi, no entanto, apresentado pelo escritório de advogados.	O escritório de advocacia apresentou seu requerimento ao Tribunal de Primeira Instância, apesar da organização ineficaz.

2.3.3.3. Parágrafos

Escreva parágrafos curtos que contenham somente uma ideia. Inicie sempre com uma sentença que define o tópico que será detalhado. Uma frase de tópico pode fornecer uma transição de um parágrafo para outro. Uma palavra ou frase de transição diz claramente ao público-alvo se: (i) o parágrafo se expande no parágrafo seguinte, (ii) se contrasta com ele, (iii) se haverá uma direção completamente diferente. Outro ponto bastante importante é o título do parágrafo. Este deve conter um resumo do mesmo de forma que ao consumir tal informação já tenha um resumo da mesma pelo título. A Tabela 5 nos mostra alguns exemplos.

Tabela 5. Exemplo de uso em parágrafos

Texto Comum	Texto com Linguagem Simples
Quando duas ou mais declarações são feitas em substituição e uma delas, se feita de forma independente, seria suficiente, a argumentação não é insuficiente pela insuficiência de uma ou mais das declarações alternativas.	Se uma das partes fez declarações alternativas, o pedido é suficiente se qualquer uma delas for suficiente.

2.3.4. Outras Práticas Recomendadas

Além de todas as práticas citadas anteriormente, algumas outras mais gerais também podem auxiliar na melhoria do entendimento e na clareza da informação. Aqui temos algumas delas:

- O uso de exemplos pode ajudar seu público-alvo a entender melhor a informação.
- A criação de listas e tabelas pode organizar melhor um texto muito extenso e cheio de itens.
- A inclusão de uma ilustração pode ser mais útil do que uma extensa descrição textual.
- O uso de recursos como negrito e itálico podem ser interessantes para destacar coisas importantes.
- A redução do número de referências cruzadas pode auxiliar o consumidor da informação a não se dispersar muito.

2.4. Práticas para Interface Web em Linguagem Simples

As práticas a seguir foram compiladas a partir do guia da Associação Brasileira de Entidades Estaduais de Tecnologia da Informação e Comunicação (ABEPTIC, 2022). Essa instituição divulga um material rico em exemplos e práticas que aplicam a Linguagem Simples a diversos tipos de informação. Neste caso citamos os casos referentes às interfaces para aplicações WEB.

2.4.1. Identifique o objetivo do seu site e direcione funções para cumpri-lo.

Se o seu site não ajudar os usuários a concluir a tarefa que desejam realizar, eles sairão. Você precisa deixar bem claro o propósito do seu site. Isso ajudará a esclarecer o que seu site faz de fato e como ele pode ajudar a resolver o problema do usuário. Não adianta rechear o seu site de funcionalidades se o seu visitante não sabe como lidar com elas. Ao invés disso, coloque apenas as funções essenciais na página e o auxilie a se decidir mais rápido.

Figura 1.4. Exemplos de carrosséis de funcionalidades de acesso rápido em site bancário. Fonte: Santander[19]

Por exemplo, se a página tem por objetivo a oferta de serviços, pode ser importante destacar a busca em primeiro plano ou então os serviços principais, ou recomendados em destaque logo nas primeiras seções do site. A Figura 1.4 demonstra alguns exemplos comuns de botões dispostos em um carrossel ou slider nas primeiras partes da página de sites bancários com funções mais buscadas, ou ofertas de serviços recomendados.

2.4.2. Busque saber o que o usuário quer encontrar no site

Você deve começar a construção de qualquer site tentando entender quem são os seus usuários. O processo de caracterizar as personas ajudará você a descobrir os interesses e necessidades de cada usuário. Em seguida procure analisar o que as pessoas vão procurar em seu site, pois isso irá servir para saber o que deve ser apresentado, onde e de que forma:

- Quais são as páginas que os usuários irão visitar mais?
- Quais seções ou conteúdos passarão mais tempo?
- Quais as principais tarefas que os seus usuários vão executar?

Uma técnica que pode ajudar a definir a importância, ordem ou até nomes mais claros para páginas, menus e funções do seu site é o **card sorting**. Esta técnica é muito utilizada para arquitetura de informação. Um grupo de pessoas do seu público-alvo recebe cartões ou post-its com os itens do site e devem organizá-los conforme as categorias que mais fazem sentido para eles. Eles podem ainda definir os nomes de cada grupo que mais tenha significado num contexto de uso. Assim entendemos o modelo mental das pessoas sobre o conteúdo e aumentamos a capacidade do usuário de conseguir se localizar de forma rápida no sistema.

2.4.3. Faça seu site funcionar num padrão que o usuário já conhece

Os usuários, em geral, transferem as expectativas que construíram no uso de um site para outro que pareça semelhante. Então, procure manter os padrões para sites com o mesmo propósito para evitar criar experiências onde o usuário tenha que reaprender aonde ir para conseguir executar as suas tarefas. Isso também é importante, por reduzir o tempo gasto do usuário na execução de suas atividades. Em geral, podemos citar alguns padrões ou formas de organização que já se tornaram de uso comum, como:

- Crie sites com estruturas de cabeçalho, conteúdo e rodapé;
- Mantenha a barra lateral auxiliar à direita, se o seu site possui um layout com mais de uma coluna;

[19] https://www.santander.com.br/

- Mantenha um layout de coluna única no formato acessado por smartphones;
- O logo costuma ficar no cabeçalho à esquerda;
- Geralmente se utilize um menu na parte superior;
- Diferencie os tipos de conteúdo em textos com títulos, subtítulos e parágrafos; e
- Use ícones de uso comum para os usuários, mas ao utilizá-los veja se é preciso rótulos de textos que podem ajudar a reduzir a ambiguidade.

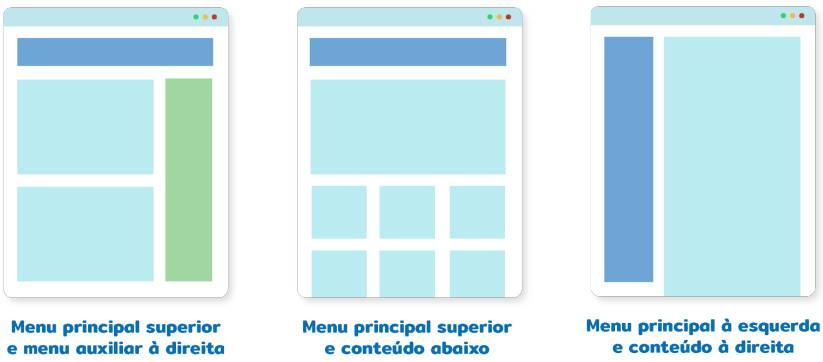


Figura 1.5. Exemplos de layouts web comuns. Fonte: os autores.

Existem diversas composições, estruturas e layouts disponíveis atualmente na web. Uma recomendação valiosa, é realizar uma busca por páginas similares ou do nicho que você está desenvolvendo para identificar os padrões comuns. Esta técnica é conhecida como **benchmarking**, um processo contínuo de comparar produtos, serviços e práticas com empresas do mesmo segmento. Na Figura 1.5 exemplificamos alguns layouts bastante utilizados pela web em blogs, sites institucionais e sistemas interativos.

2.4.4. Use padrões de leitura

Em média, os usuários leem apenas as duas primeiras palavras de cada linha. Ou, geralmente, apenas "passam os olhos" por um texto em busca das informações que precisam. Esta prática convencionou-se chamar de escaneabilidade. Com isso, eles podem decidir em apenas cinco segundos se seu site é útil para eles ou não. Então use poucas palavras no início para ele entender facilmente o que vem adiante. Em geral, também acessam as páginas com foco no lado superior esquerdo da página, seguindo nosso padrão de leitura. Portanto, explore estes padrões que já são consenso. Bons padrões de conteúdo da web usam:

- **O estilo pirâmide invertida:** Comece com a declaração mais curta e clara que você pode fazer sobre o seu tópico. Coloque as informações mais importantes na parte superior.
- **Conteúdo fragmentado:** Não tente empacotar tudo em parágrafos longos. Divida os tópicos em seções lógicas separadas por títulos informativos.
- **Padrão F de leitura:** A maioria dos usuários em uma página lê de cima para baixo, da esquerda para direita, criando um formato similar a de uma letra F. Por isso, destaque informações mais importantes logo nos primeiros parágrafos e evite colocar informações importantes à direita.

Figura 1.6. Exemplos de páginas web com padrões de leitura. Fonte: ABEPTIC, 2022

2.4.5. Não leve o conteúdo de documentos textuais direto para a web

Se você pensa que precisa somente apresentar as informações que existem em documentos impressos, você está errado. Em geral, informações escritas em documentos não estão no formato certo para a web. Você precisará transformar estas informações em algo que esteja simples para ser apresentado ao público. Algumas sugestões de boas práticas para você fazer esta adaptação são:

- Mantenha a mensagem mais importante, de forma clara, no topo da página.
- Divida seu conteúdo em seções do mesmo tópico ou que mantenham alguma relação entre si. O desencadeamento de ideias é um valioso elemento para o leitor de internet. Isso estimula a continuidade da leitura e aumenta o interesse do usuário no que está sendo dito.
- Use títulos para auxiliar os usuários a navegar pelo conteúdo.
- Destaque os principais fatos em uma lista com marcadores.
- Explique instruções complexas de forma visualmente atraente.
- Verifique se o site está sendo rápido e direto na entrega da informação. Deve-se entregar o que se precisa o mais diretamente possível.
- Gere uma boa navegabilidade para os seus usuários. Uma boa arquitetura da informação utilizada tanto na estrutura das páginas de um site, como nas suas seções, nas suas categorias de assuntos, e também nos seus links.
- Complemente as informações: Imagens, vídeos, áudios, infográficos, etc. Diversos recursos podem ser utilizados para acrescentar entendimento ao seu leitor. Porém, adicionar somente conteúdo relevante para o público.

2.4.6. Evite o uso de PDF

Em geral, usuários tentam evitar ler PDFs. Mas, se você precisar postar um, use uma página para isso. Lá você deve explicar do que se trata, qual o tamanho do arquivo e que informações podem ser encontradas lá. Em seguida, coloque a possibilidade de download do PDF. A Figura 1.7 exemplifica um exemplo interessante na página do portal de dados abertos brasileiro. E lembre-se! Se for um PDF para ser preenchido como, por exemplo, um formulário, permita que isso seja feito no próprio documento, evitando que a pessoa tenha que imprimir para poder preencher.

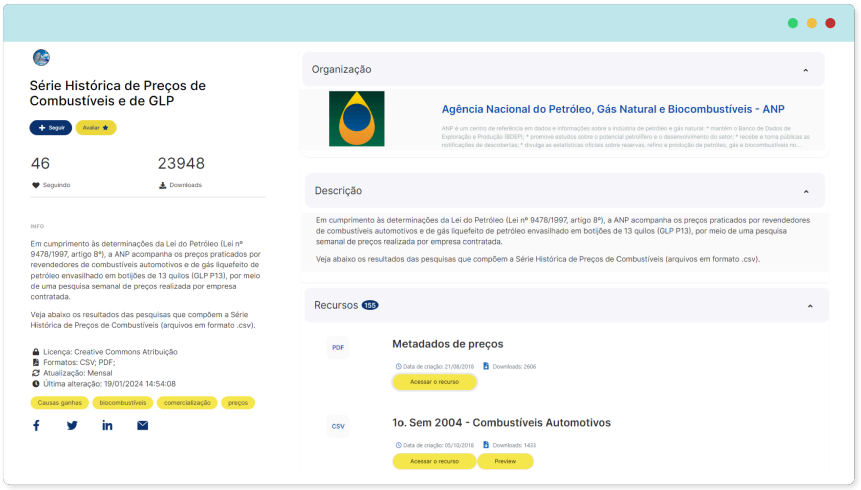


Figura 1.7. Página do portal de dados abertos do Brasil com descrição do PDF para download. Fonte: dados.gov.br[20]

2.4.7. Use mensagens assertivas, curtas, claras e de entendimento geral

Não desperdice o tempo dos seus usuários. Vá sempre direto ao ponto. Quando precisamos dar mensagens aos usuários em uma interface web, estas precisam explicar claramente e de forma simples o que aconteceu e deixar claro também como o usuário deve prosseguir no próximo passo. Veja algumas práticas e exemplos a seguir:

- Evite duplo negativo, pois, confundem o usuário e aumentam a sua carga
- Comece com o objetivo da ação e em seguida apresente o resultado da interação
- Use palavras simples e reduza sempre ao máximo palavras desnecessárias.
- Escreva as ações com clareza. Evite palavras e expressões muito genéricas.

2.4.8. Formate os links corretamente

Nomes de links eficazes são fundamentais para a satisfação do usuário. Os links devem estar escritos em uma linguagem simples de forma que o usuário entenda exatamente para onde o link o levará. Algumas boas práticas devem ser seguidas:

- Os nomes dos links devem ser iguais ao nome da página vinculada.
- Não use o nome completo de um documento ou programa como nome de link.
- Seja o mais explícito possível — muito longo é melhor que muito curto.
- Faça o link significativo. Não use “clique aqui” ou “mais”.
- Não incorpore links no texto. Apenas convida as pessoas a deixar seu texto!
- Adicione uma breve descrição, quando necessário, para esclarecer o link.
- Lembre-se de que alguns de seus usuários podem ter deficiência visual. Não use links como “Clique aqui” ou “Clique no botão verde”.
- Certifique-se de que seus links estejam acessíveis a todos os usuários.
- Você deve usar links que explicam o conteúdo da página para a qual ele está vinculado. Se o seu link disser “Relatórios Anuais”, a página de destino deverá ser intitulada “Relatórios Anuais”.

[20] https://dados.gov.br/dados/conjuntos-dados/serie-historica-de-precos-de-combustiveis-e-de-glp

2.4.9. Apresente informações relevantes

As interfaces não devem conter informações pouco necessárias. Cada informação a mais em uma interface compete com as informações relevantes e diminui a visibilidade destas. Você deve manter o design focado no essencial. Certifique-se de que os elementos visuais da interface suportem os objetivos principais do usuário. Veja também se o conteúdo atende ao objetivo do negócio. Não deixe que elementos desnecessários distraiam os usuários das informações que eles realmente precisam.

2.4.10. Reduza a complexidade

Tudo que é muito complexo afasta o usuário. Já vimos por diversos motivos que devemos descomplicar ao máximo nossas informações para fazer com que os usuários as entendam. Porém, temos que tomar cuidado para o "simples" não se tornar algo que não agrega muito valor para o usuário, deixando a experiência dele ruim. Você deve reduzir a complexidade da informação apresentada ao invés do usuário ter que dedicar mais tempo na tarefa para entendê-la. Para isso algumas práticas podem ser bem eficazes:

- Divida tarefas complexas em etapas menores para diminuir o nível de dificuldade de entendimento.
- Tenha cuidado para não simplificar ao ponto de escrever de forma muito genérica fazendo com que o usuário não entenda o que está sendo dito ali.
- Tente reduzir a duração de uma tarefa. Caso não seja possível, pense em como você pode tornar a espera mais aceitável.
- Forneça passos compreensíveis em direção ao resultado, garantindo que cada etapa da sua aplicação se encaixe apropriadamente no fluxo de utilização.
- Reduza a quantidade de distrações.

2.4.11. Minimize o número de escolhas

O número de escolhas deve sempre ser o menor possível. Muitas escolhas possíveis fazem com que o usuário tenha que memorizar coisas, pois a medida que avança na lista de escolhas tem que manter a percepção das demais que já foram apresentadas. Além disso, muitas opções de escolha dificultam a análise do usuário sobre a opção que lhe atende. Algumas boas práticas devem ser seguidas:

- Destaque as opções mais recomendadas para não sobrecarregar os usuários.
- Use tutoriais de primeiro acesso para diminuir o processo de entendimento para novos usuários.
- Crie menus principais que se abrem em várias opções quando passamos o cursor na navegação de um site. Eles podem auxiliar o usuário a atingir níveis mais profundos de um site com menos cliques. Mas tenha cuidado, pois para novos visitantes, esse menu acaba sendo uma barreira cognitiva. Geralmente acabam confundindo e não segue a lógica do mapa mental do usuário. A solução é investir em arquitetura da informação e deixar a navegação mais balanceada.

2.4.12. Seja liberal no que você aceita

Seja empático, flexível e tolerante, dando mais de uma opção para as entradas do usuário. Antecipe o que puder como entrada e acesso, fornecendo uma interface inteligente. Quanto mais pudermos antecipar e planejar no design, mais resiliente o sistema será.

- Em um formulário com campo numérico para CPF, permita que o usuário digite apenas os números e o sistema adicione os pontos para melhorar a visualização.
- Quando for solicitar o cadastro de uma senha, defina os limites para entrada e dê feedback claro ao usuário. Por exemplo: sua senha deve ter de 5 a 8 caracteres e possuir uma letra maiúscula.
- Permita ao usuário editar as respostas de um formulário, antes ou após enviá-lo. Assim, o usuário pode adequar o conteúdo enviado antes de submetê-lo.

2.4.13. Use listas com itens em ordem de importância

Usuários costumam lembrar mais dos primeiros e últimos itens de uma lista, esta é uma lei de UX chamada de efeito da posição em série (Yablonski, 2020). Assim, coloque os itens menos importantes no meio das listas porque esses itens tendem a ser armazenados com menos frequência na memória de longo prazo e na memória de trabalho. Por exemplo, na navegação posicione as principais ações à esquerda e à direita para aumentar a memorização.

2.4.14. Apresente informações sobre o estado do sistema

Os usuários devem estar sempre informados sobre o que está acontecendo num período razoável de tempo. Este é um dos princípios ou heurísticas bem famosas de Jakob Nielsen, sobre a visibilidade do estado do sistema (Nielsen, 2015). Portanto, apresente o que está acontecendo em uma atividade enquanto ela está sendo executada. Isso ajudará o usuário a se localizar e a decidir se mantém a execução ou se termina a tarefa em determinado momento. Uma indicação clara do progresso de atividades pode motivar os usuários a concluírem as tarefas. Algumas boas práticas podem ser seguidas:

- Apresente feedback ao usuário o mais rápido possível (idealmente, imediatamente).
- Comunique claramente aos usuários qual é o estado do sistema - nenhuma ação com consequências para os usuários deve ser tomada sem informá-los.
- Use barras de carregamento ou etapas a serem concluídas. Quando o usuário não tem clareza do que está acontecendo, ele muitas vezes pode tomar a decisão de desistir da tarefa por pensar que está muito demorada quando ela está prestes a ser finalizada.
- Use breadcrumb (também conhecido como “migalhas de pão” ou “trilha de migalhas de pão”) que é um tipo de esquema de navegação auxiliar que revela a localização do usuário em um site ou aplicação web. Eles indicam a localização atual do usuário e por onde passou até chegar onde está.

2.4.15. Use termos e elementos conhecidos do usuário

Use palavras e conceitos familiares ao usuário. Os usuários devem entender o significado dos termos utilizados sem precisar procurar suas definições. Apesar de que as definições devem estar sempre disponíveis, deve-se buscar ao máximo palavras de conhecimento do público a quem se destina a informação. As personas ajudarão você a descobrir os termos que seus usuários provavelmente conhecem.

2.4.16. Mantenha as mesmas características de elementos com funções iguais

Os usuários não devem se perguntar se palavras, situações ou ações diferentes significam o mesmo. Manter a consistência na escrita e no visual diminui a dificuldade de entendimento, auxiliando o usuário a não precisar aprender algo novo a cada momento. Portanto, temos algumas práticas a seguir:

- Diferencie claramente dois elementos que possuem funções distintas
- Mantenha o mesmo visual para elementos com a mesma função
- Use a mesma palavra para se referir a um mesmo elemento

Tais práticas advêm dos princípios da Gestalt, uma teoria da psicologia que descreve a percepção humana em diversas situações como agrupamentos, elementos semelhantes e reconhecimento de padrões (IxDF, 2016b). Na Figura 1.8 temos um exemplo a partir da interface do Airbnb, um serviço de hospedagem, onde os elementos associados são desenhados de forma similar.

Figura 1.8. Trecho da interface de hospedagem do Airbnb. Fonte: Airbnb[21]

2.4.17. Faça o usuário reconhecer os conteúdos

O usuário não deve ter que lembrar informações de uma interface para outra. As informações necessárias para ele usar devem estar sempre visíveis ou facilmente recuperáveis quando necessário. As interfaces que promovem este tipo de facilidade reduzem a quantidade de esforço para o entendimento exigido dos usuários. Ofereça auxílio no contexto, em vez de dar aos usuários um longo tutorial para memorizar.

[21] https://www.airbnb.com.br/

2.4.18. Mantenha próximo os itens relacionados entre si.

Para nossa mente, coisas que estão próximas parecem estar mais relacionadas entre si do que se estivessem distantes, sendo esse o princípio da proximidade também originado na Gestalt (IxDF, 2016b). De uma maneira visual, o usuário irá relacionar e agrupar os itens, botões, campos de um formulário ou até textos que estiverem próximos entre si do que se estiverem mais distantes. Na Figura 1.9 essa relação é bastante visível, onde os ícones e textos criam uma hierarquia clara para o usuário dos grupos para a leitura.

Figura 1.9. Página do Gerenciador de arquivos Dropbox com elementos relacionados. Fonte: Dropbox[22]

2.4.19. Torne itens visualmente similares para facilitar a identificação

Elementos parecidos visualmente, são percebidos como integrantes do mesmo grupo e tendo a mesma função. Essa noção é importante para dar clareza aos usuários de quais botões, menus ou regiões da sua interface estão relacionadas simplesmente pelas cores, ou formatos que elas possuem. Basta que alguma diferença seja percebida entre os elementos, para eles serem percebidos como de grupos distintos. Ao invés de cores diferentes, você pode diferenciar os itens por tamanhos, formas, entre outras diferenças.

2.4.20. Torne o uso acessível

Processos flexíveis e com boa acessibilidade podem ser realizados de diferentes maneiras, por todo tipo de pessoas, que podem escolher e buscar o acesso que melhor funcione para elas. Permitir que os usuários personalizem e tenham uma boa usabilidade é fundamental para todos. Algumas boas práticas podem ser seguidas:

- Forneça atalhos de teclado ou gestos de toque;
- Torne todas as funcionalidades acessíveis pelo teclado;
- Possibilite layouts alternativos (mais simples) ou separação de background, alto contraste ou tornando o conteúdo mais simples de ser lido e ouvido;
- Forneça páginas adaptadas a dispositivos móveis ou outros que usuários possam fazer seleções sobre como desejam que o produto funcione;

[22] https://www.dropbox.com/

- Permita a customização da interface com cores próprias ou opções comuns;
- Recomende assuntos relacionados;
- Evite animações, sons ou cores extravagantes que possam causar desconfortos;
- Maximize a compatibilidade com tecnologias de assistência virtual.

2.4.21. Ajude os usuários a reconhecer, diagnosticar e se recuperar de erros

Devemos sempre que possível criar mecanismos para evitar erros desnecessários que o usuário possa cometer. Desde evitar deslizes ao usar a interface até possibilitar desfazer alguma ação. Apresente mensagens de erro claras e com tratamentos visuais, pois auxiliarão os usuários a notá-las e reconhecê-las. Expresse mensagens em linguagem simples (sem códigos de erro) indicando com precisão o problema e sugerindo uma solução de forma construtiva. Algumas boas práticas podem ser seguidas:

- Use elementos visuais tradicionais para mensagens de erro, como texto em negrito e vermelho.
- Diga aos usuários o que deu errado na linguagem que eles entenderão — evite jargões técnicos ou termos estrangeiros. Por mais que sejam comuns muitas palavras voltadas para tecnologias advindas do inglês, evite palavras como login, logar, bug, entre outros.
- Ofereça aos usuários uma solução, como um atalho que pode resolver o erro imediatamente.
- Desabilite botões sem o preenchimento correto das opções.
- Exiba boas mensagens de erro na linguagem comum do usuário como, por exemplo, na Figura 1.10, ao invés de apresentar “ERRO 404” se apresenta ao usuário meios ou alternativas para corrigir o problema como: verificar o endereço acessado.
- Pergunte se o usuário tem certeza antes de realizar uma ação definitiva como excluir ou alterar informações.

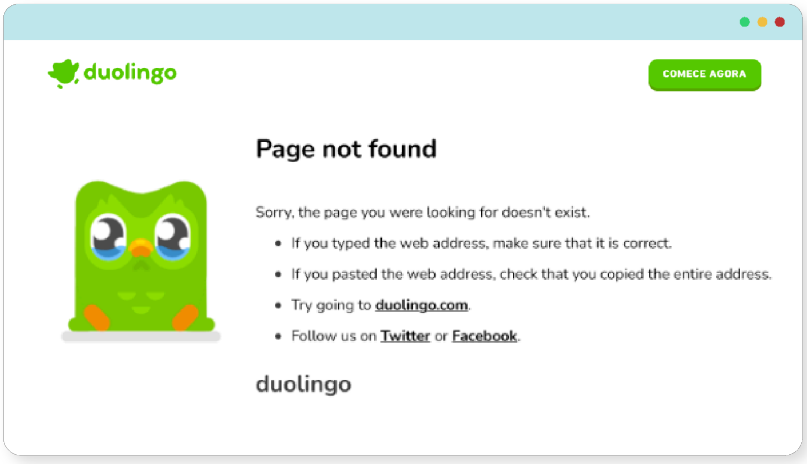


Figura 1.10. Mensagem de Erro: Página não encontrada no site de idiomas Duolingo. Fonte: Duolingo[23]

[23] https://pt.duolingo.com/

2.4.22. Ofereça ajuda

É melhor que o sistema não precise de nenhuma explicação adicional. No entanto, pode ser necessário fornecer ajuda para que usuários entendam como concluir suas tarefas. O conteúdo de ajuda e documentação deve ser fácil de pesquisar e focado na tarefa do usuário. Mantenha-o conciso e liste as etapas concretas que precisam ser realizadas. Se possível utilize chatbots ou assistentes virtuais para o auxílio com fluxos de conversas que representam os caminhos de diálogos reais do usuário ou redirecionam a conteúdos de ajuda.

2.5. Ferramentas de Apoio e Automação para a Linguagem Simples

Para aplicação da Técnica de Linguagem Simples podemos lançar mão do uso de ferramentas de apoio. Apresentamos aqui algumas das principais utilizadas e de livre acesso.

2.5.1. Coh-Metrix-Port 3.0

O Coh-Metrix-Port[24] é uma tradução para o português brasileiro da ferramenta Coh-Metrix em inglês (http://cohmetrix.com/). Ela fornece índices para avaliar a coesão (harmonia entre as partes textuais), a coerência (ligação lógica entre as ideias) e a dificuldade de compreensão de um texto utilizando diversos níveis de análise linguística, tais como: lexical (ocorrência de termos diferentes, mas de mesmo significado), sintática (organização dos elementos na oração), discursiva (exame da estrutura para compreender a construção ideológica do texto) e conceitual (busca de termos obscuros que possuam definições mais claras). Para usar, basta acessar o site do projeto, escrever ou colar o texto a ser analisado no campo correspondente e clicar no botão submit, aguardando que a página exiba o resultado da análise. Nessa metodologia, diversos recursos e ferramentas de processamento de linguagem são utilizados para calcular as 46 diferentes métricas, entre elas o Índice de Leiturabilidade Flesch, Complexidade Sintática, Diversidade Léxica entre outros.

2.5.2. CAMELoT – Citizen Automatic Model Translator

O CAMELoT[25] é uma aplicação web que realiza a tradução de modelos de processos de negócios BPMN para Linguagem Cidadã de Processos (Oliveira, et al., 2021). Ele faz isso de forma semiautomática, ou seja, após a tradução é necessário alguns ajustes manuais por parte do usuário, que pode, por exemplo, decidir quais atividades farão parte do novo modelo. O procedimento necessário é bastante simples e intuitivo: através de um fluxo de quatro etapas, o usuário seleciona seu arquivo contendo sem modelo de processo em formato XPDL e o envia ao sistema, que identifica os elementos existentes no arquivo. Na etapa de configurações, o usuário tem a opção de remover atividades ou atores, poder editar os nomes das tarefas e deixá-las mais objetivas, além de poder definir uma cor para cada ator. Na etapa de exportação, o sistema apresenta o modelo já pronto, que pode ser salvo nos formatos disponíveis ou retornar para fazer ainda algumas adaptações.

[24] http://fw.nilc.icmc.usp.br:23380/cohmetrixport
[25] https://camelot-5bf1b.firebaseapp.com/

Figura 1.11. Interface do Sistema CAMELoT.

2.5.3. QuillBot

Com modos personalizados ilimitados e 8 modos predefinidos, o sistema permite reformular o texto de inúmeras maneiras. Ele melhora a fluência e, ao mesmo tempo, garante que se tenha o vocabulário, o tom e o estilo adequados para qualquer ocasião. Basta inserir o texto na caixa de entrada e a IA trabalhará para criar a melhor paráfrase.

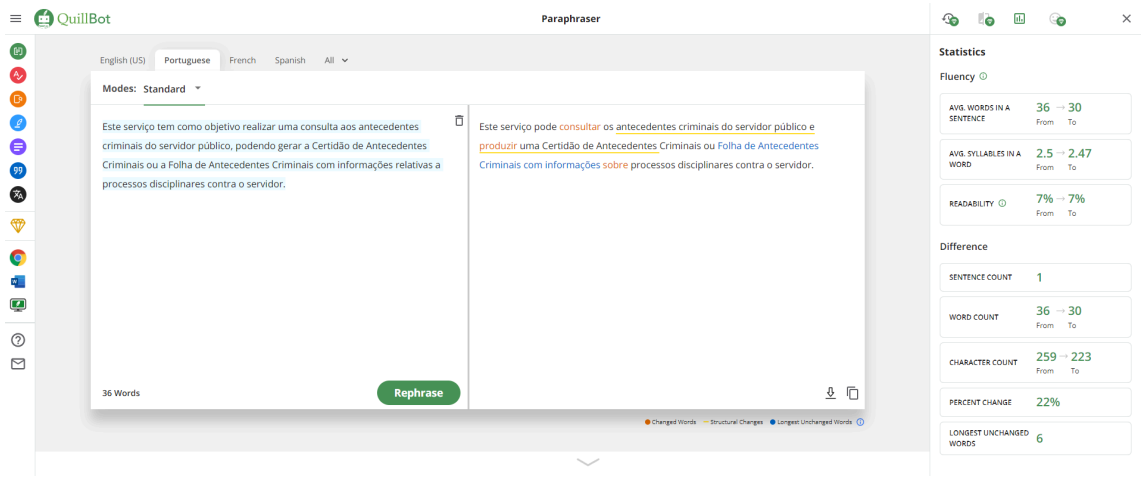


Figura 1.12. Interface do Sistema QuillBot[26].

2.5.4. Word Spinner

Word Spinner[27] é um programa de reescrita de texto online, ou seja, ele reescreve seu texto com outras palavras. Funciona em 2 etapas, faz primeiramente a paráfrase do texto e depois a troca das palavras por sinônimos. Portanto, além de realocar frases, o Word Spinner fornece uma lista de sinônimos no próprio texto parafraseado que pode ser usado para substituir as palavras que você usou no texto original. Este programa de parafrasear texto online usa Inteligência Artificial para reescrever o texto. Para reescrever o texto, este programa usa Inteligência Artificial. Este robô puxa todas as

[26] https://quillbot.com/
[27] https://wordspinner.com.br/

informações que encontra em seu material de origem e oferece paráfrases ideais ou sinônimos que podem ser trocados por palavras. O Word Spinner (também conhecido como Spinner em português, Reescritor, Parafraseador), é amplamente adotado por pessoas que precisam de um texto parafraseado totalmente em português.

2.5.5. Chatmind

Chatmind[28] é um mapeamento mental guiado por chat alimentado por GPT4. Qualquer pessoa pode fazer mapas mentais, conversando com GPT. Podem ser criados e aperfeiçoados instantaneamente mapas mentais em conversa com a IA.

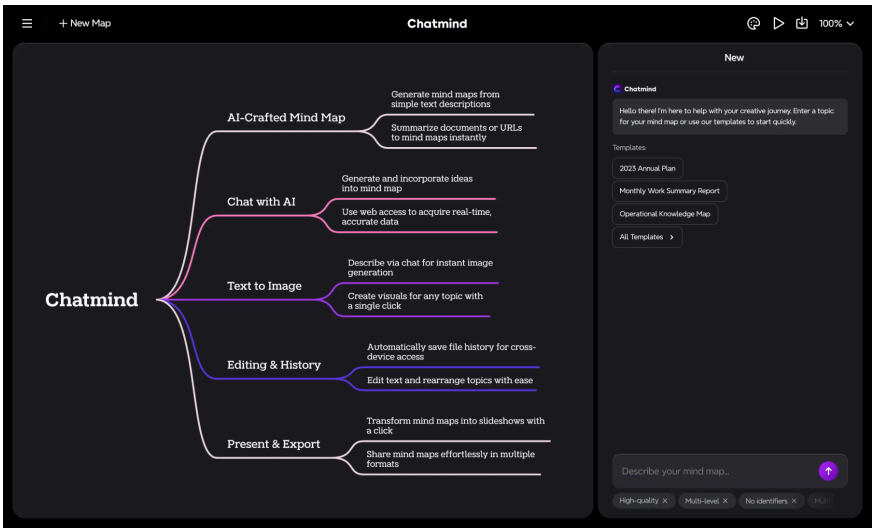


Figura 1.13. Interface do Sistema Chatmind.

2.5.6. AnswerThePublic

O AnswerThePublic[29] permite que você escolha uma palavra-chave como mecanismo de busca e em cima dela uma busca no Google é realizada, trazendo como resultado todas as frases e perguntas úteis que as pessoas estão fazendo sobre sua palavra-chave. É uma fonte preciosa de informações sobre o consumidor, que podemos usar para criar conteúdos, produtos e serviços super úteis e atuais. Dados precisos direto dos clientes com base no que eles andam buscando na internet. Todos os dias, 3 bilhões de buscas são realizadas no Google, sendo 20% delas inéditas.

[28] https://chatmind.tech/pt
[29] https://answerthepublic.com/

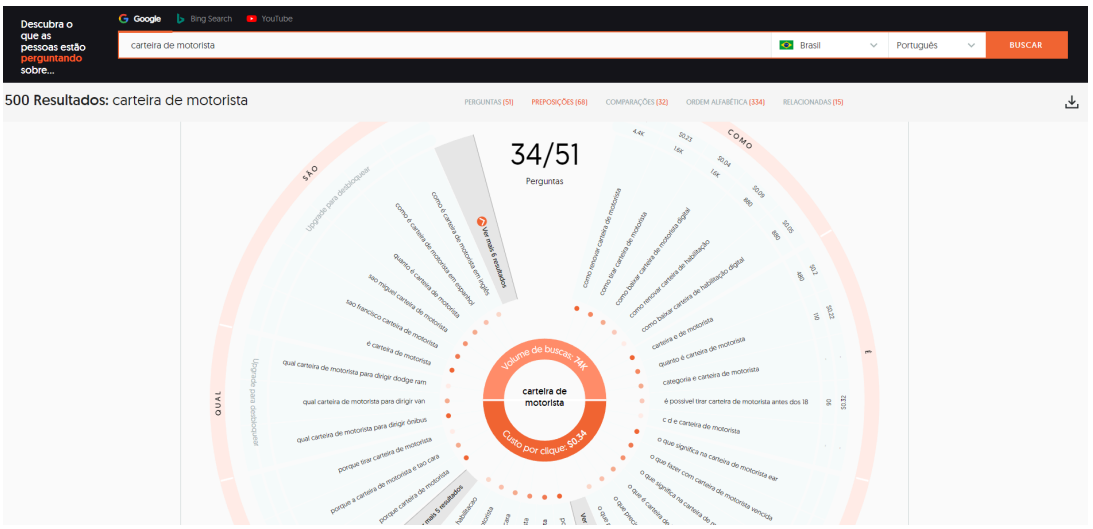


Figura 1.14. Interface do Sistema AnswerThePublic.

2.5.7. Chart Lab

A ferramenta Chart Lab, permite construir um gráfico com o apoio dessas práticas de forma automática, fácil e intuitiva (Oliveira et al., 2022). O propósito geral da ferramenta é fornecer uma validação dos componentes do gráfico frente às práticas da linguagem. Com essas orientações gerais, permite que o usuário possa corrigir e atualizar seu gráfico dentro destes parâmetros e assim melhorar a compreensibilidade dos dados pelos leitores em geral. O sistema ainda segue em desenvolvimento, mas sem um protótipo online, entretanto em publicações recentes é possível ter acesso ao repositório de códigos de um protótipo funcional.

Figura 1.15. Interface do Sistema ChartLab. Fonte: Oliveira et al. (2022)

2.6. Conclusão

Neste Capítulo apresentamos como a Técnica da Linguagem Simples pode apoiar a construção de conteúdos de fácil entendimento na Web. Um requisito necessário para a transparência das informações e seu uso para a transformação digital que vivenciamos.

Apresentamos os fundamentos teóricos da área de UX e de Linguagem Simples. Destacando os principais conceitos relacionados aos temas e demonstrando o inter relacionamento de cada tópico. Demos ênfase na transparência, reforçando a importância destes temas na implementação do conceito de Transformação Digital dos governos. O foco na transparência é reforçado também através do uso de serviços, dados e informações e no fato do entendimento destes ser uma característica primordial na transparência. A técnica da Linguagem Simples é usada na busca desse objetivo porque favorece o entendimento para uma efetiva transparência, e com isso impulsiona a transformação digital.

Foram apresentadas ainda as principais iniciativas no Brasil e no mundo no uso desta técnica de Linguagem Simples. Como pode ser visto hoje já contamos com diversas organizações que fazem coro na aplicação desta técnica, sendo estas, centros de pesquisa, organizações internacionais independentes, universidades e outras, o que nos mostra a importância e magnitude deste movimento, chamado de Plain Language Movement, para garantir à sociedade o direito ao entendimento. A partir do *Federal Plain Language Guidelines* apresentamos também as principais práticas para uma escrita mais simples. Foram mostradas práticas gerais para organização do texto como: definir o público-alvo adequado, usar bons títulos e escrever seções curtas de conteúdo. Seguindo a discussão para a escrita do texto propriamente dito, repassando orientações sobre as palavras, sentenças e parágrafos. Finalizando com boas práticas gerais sobre o design, o uso de exemplos e representações visuais. Podemos destacar ainda que a Linguagem Simples, se propõe a melhorias não apenas no âmbito da escrita de texto, mas na estrutura e no design da informação. Contudo, no *Federal Plain Language Guidelines* há uma única diretriz que se refere ao uso de imagens para clareza informacional, indicando apenas o uso ilustrativo. Não há nenhuma menção a como se deve construir eficientemente esses recursos visuais. Isso demonstra ainda mais a importância do aprofundamento e da exploração deste tema deste minicurso.

Referências

ABEPTIC. Guia ABEPTIC de Uso de Linguagem Simples para Apresentação de Serviços Públicos. 2022. Disponível em: https://abep-tic.org.br/as-praticas-da-linguagem-simples/. Acesso em: 8 jan. 2024.

Barboza, E. M. F. (2010). A linguagem clara em conteúdos de websites governamentais para promover a acessibilidade a cidadãos com baixo nível de escolaridade. Inc. Soc., Brasília, DF, v. 4 n. 1, pp. 52-66.

Bowman, Doug A. 2014. 3D User Interfaces Disponível em: https://www.interaction-design.org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed/3d-user-interfaces. Acesso em: 23 jan. 2024.

Cappelli, (2009) Uma abordagem para transparência em processos organizacionais utilizando aspectos. Tese de Doutorado. Disponível em: http://www-di.inf.puc-rio.br/~julio/tese-cappelli.pdf. Acesso em: 05 jan. 2024.

Cappelli, C., Leite, J. C. S. P. (2008). Transparência de processos organizacionais. II Simpósio Internacional de Transparência nos Negócios, Universidade Federal Fluminense, LATEC, Niterói, RJ, Brasil.

Govinfo. Public Law 111 - 274 - Plain Writing Act of 2010 (2010). Disponível em: https://www.govinfo.gov/app/details/PLAW-111publ274. Acesso em: 24 jan. 2024.

ISO, I. O. f. S. ISO/FDIS 24495-1: Plain language — Part 1: Governing principles and guidelines. 2023. Disponível em: https://www.iso.org/standard/78907.html. Acesso em: 20 jan. 2024.

ISO, ISO 9241-210:2019. Ergonomics of human-system interaction Part 210: Human-centred design for interactive systems. 2019. Disponível em: https://www.iso.org/standard/77520.html. Acesso em: 23 jan. 2024.

IxDF, Interaction Design Foundation. 2016a. What is User Interface (UI) Design? Disponível em: https://www.interaction-design.org/literature/topics/ui-design. Acesso em: 23 jan. 2024.

IxDF, Interaction Design Foundation. 2016b. What are the Gestalt Principles? Disponível em: https://www.interaction-design.org/literature/topics/gestalt-principles. Acesso em: 23 jan. 2024.

Jakobson, R. On linguistic aspects of translation. In: On translation. [S.l.]: Harvard University Press, 2013. p. 232–239.

Morville, P. User experience design, June 2004. Disponível em: https://semanticstudios.com/user_experience_design/. Acesso em: 23 jan. 2024.

Nielsen, J. (2015). 10 Heuristics for User Interface Design. Disponível em: https://www.nngroup.com/articles/ten-usability-heuristics/. Acesso em: 8 jan. 2024.

Oliveira, R., Cappelli, C., & Oliveira, J. (2022). Chart Lab: Uma ferramenta para o design de visualizações de dados em linguagem simples. In Anais do X Workshop de Computação Aplicada em Governo Eletrônico (pp. 192-203). SBC.

Oliveira, R., Cappelli, C., & Santoro, F. M. (2021). CAMELoT-Tradutor Semiautomático de Processos em BPMN para Modelos Compreensíveis aos Cidadãos. iSys-Brazilian Journal of Information Systems, 14(3), 5-24.

Plain, A. I. What is plain language? 2024. Disponível em: https://plainlanguagenetwork.org/plain-language/what-is-plain-language/. Acesso em: 20 jan. 2024.

Yablonski, J. (2020). Laws of UX: Using psychology to design better products & services. O'Reilly Media.

Rodrigo Oliveira

Doutorando em Computação na Universidade Federal Fluminense (UFF). Mestre em Informática, linha de pesquisa em Sistemas de Informação pela Universidade Federal do Rio de Janeiro, UFRJ (2019) possui Bacharel em Sistemas de Informação pela Universidade Federal do Estado do Rio de Janeiro, UNIRIO (2018) e Graduado em Design Gráfico pela Universidade do Grande Rio, UNIGRANRIO (2013). Pesquisador em Linguagem Cidadã no LincLab (Laboratório Interdisciplinar de Linguagem Clara). Analista de Geotecnologia na Imagem Geosistemas (img.com.br). Carreira desenvolvida na área de Design e Tecnologia, com experiência em comunicação visual, design de interação e desenvolvimento de software. Com interesses na área de Visualização de Dados, UX Design, Linguagem Simples e Gestão de Processos de Negócios, com ênfase em transparência da informação e automação de soluções em TI.
http://lattes.cnpq.br/1091110605525620

Claudia Cappelli

Professora do curso de Ciência de Computação da UERJ. Professora do Mestrado Profissional em Telemedicina e Telessaúde da UERJ. Colaboradora do programa de Pós Graduação em Sistemas de Informação da UFRJ. Doutora em Ciências - Informática pela PUC-Rio (2009). Mestre em Sistemas de Informação pela Universidade Federal do Rio de Janeiro (2000). Graduada em Informática pela Universidade do Estado do Rio de Janeiro (1985). Realizou estágio Pós-Doutoral junto ao Programa de Pós-Graduação em Informática da Unirio (2010) e na UFRJ (2020). Pesquisadora Jovem Cientista Nosso Estado pela FAPERJ. Fundadora do Instituto Nacional de Ciência e Tecnologia em Democracia Digital (INCT-DD). Pesquisadora em Literacia Digital do IBCIH (Instituto Brasileiro de Cidades Inteligentes e Humanas). Gestora de Conhecimento do Linguagem Simples Lab. Membro do Comitê Gestor do programa Meninas Digitais da SBC. Mentora da Fábrica de Startups. Representante do Brasil na Clarity. Membro da PLAIN. Pesquisadora em Inovação no INEI (Instituto Nacional de Empreendedorismo e Inovação). Foi Gerente da Área de Arquitetura Corporativa e Planejamento de Tecnologia do Citibank (1996 - 2001) e da Telemar (2001 - 2003) e Professora Adjunta do Departamento de Informática Aplicada da Universidade Federal do Estado do Rio de Janeiro (UNIRIO). Foi Diretora de Articulação com a Indústria na SBC (Sociedade Brasileira de Computação) (2016-2018). Atuou por 8 anos junto a Petrobras em Projeto de Gestão de Processos de Negócio (2008-2016) e por 2 anos na CEF em Data Science com foco em Linguagem Clara (2018-2020). Atua como revisora de diversos periódicos nacionais e internacionais. Atua na área de Sistemas de Informação, principalmente nos seguintes temas: Gestão de Processos de Negócio, Arquitetura Corporativa, Gestão de TI, Transparência, Linguagem Simples e Governo Digital.
http://lattes.cnpq.br/4930762936357558