UNIVERSIDADE ABERTA

ANÁLISE COMPREENSIVA DE MÉTODOS DE TRANSFORMAÇÃO DE DADOS

João Paulo Marques Correia

Dissertação de Mestrado

Mestrado em Bioestatística e Biometria

Orientadora: Professora Doutora Maria do Rosário Ramos, Universidade Aberta

Co-orientadora: Professora Doutora Patrícia Engrácia, ISCTE-IUL

05/2025

Condições de Utilização do Trabalho por Terceiros

Agradecimentos

Quero agradecer a ambas as orientadoras, Professora Doutora Maria do Rosário Ramos e à Professora Doutora Patrícia Engrácia pela disponibilidade e *feedback* dado e acima de tudo por terem aceitado o meu convite.

Quero dedicar o meu trabalho às três pessoas que mais me são próximas - Ana, Clara e Salvador.

UNIVERSIDADE ABERTA
www.uab.pt

DECLARAÇÃO DE INTEGRIDADE

STATEMENT OF INTEGRITY

Declaro ter atuado com integridade na elaboração da presente dissertação/tese. Confirmo que em todo o trabalho conducente à sua elaboração não recorri à prática de plágio ou a qualquer outra forma de falsificação de resultados.
Mais declaro que tomei conhecimento integral do Regulamento Disciplinar da Universidade Aberta, publicado no Diário da República, 2.ª série, n.º 215, de 6 de novembro de 2013.

I hereby declare having conducted my thesis with integrity. I confirm that I have not used plagiarism or any form of falsification of results in the process of the thesis elaboration.
I further declare that I have fully acknowledged Disciplinary Regulations of the Universidade Aberta (regulation published in the official journal Diário da República, 2.ª série, N.º 215, de 6 de novembro de 2013).

Universidade Aberta, 4 de Junho de 2025

Nome completo/Full name: João Paulo Marques Correia

Assinatura/Signature:

Assinado por: **JOÃO PAULO MARQUES CORREIA**
Num. de Identificação: 13389081
Data: 2025.06.04 22:53:55 +0100

CARTÃO DE CIDADÃO

Análise Compreensiva de Métodos de Transformação de Dados

Resumo

Esta dissertação apresenta uma análise detalhada de vários métodos de transformação de dados, com particular ênfase na sua aplicabilidade, culminando no desenvolvimento de uma ferramenta (*framework*) para o apoio à decisão de técnicas de transformação. A pesquisa aborda a complexidade na preparação de dados para análise estatística, onde a escolha apropriada de transformações pode ter impacto na qualidade da análise estatística. O estudo examina detalhadamente cada método de transformação, considerando as suas propriedades matemáticas, limitações, pressupostos subjacentes e impacto nos dados originais. A implementação dos métodos é ilustrada com código elaborado na linguagem Python. A *framework* proposta integra características fundamentais dos dados (como a distribuição, formato, tipo e intervalo de dados) com os requisitos específicos da análise estatística a ser realizada, fornecendo um processo para a seleção da transformação mais adequada. Para complementar, foi desenvolvida uma ferramenta *web* de forma a agilizar o processo de escolha. A metodologia foi aplicada a alguns casos de estudo de áreas distintas permitindo uma comparação do seu desempenho. Os resultados demonstram que a eficácia das transformações está dependente das características dos dados e dos objetivos.

Palavras-chave: Transformação de dados, Análise estatística, Framework de decisão, Pré-processamento de dados

Comprehensive Analysis of Data Transformation Methods

Abstract

This dissertation presents a detailed analysis of various data transformation methods, with particular emphasis on their applicability, culminating in the development of a framework for decision support of transformation techniques. The research addresses the complexity of preparing data for statistical analysis, where the appropriate choice of transformations can have an impact on the quality of the statistical analysis. The study analyses each transformation method in detail, considering its mathematical properties, limitations, underlying assumptions and impact on the original data. The implementation of the methods is illustrated with code written in the *Python* language. The proposed framework integrates fundamental data characteristics (such as data distribution, format, type and range) with the specific requirements of the statistical analysis to be performed, providing a process for selecting the most appropriate transformation. To complement this, a web tool was developed to speed up the selection process. The methodology was applied to a number of case studies from different areas, allowing a comparison of their performance. The results show that the effectiveness of the transformations depends on the characteristics of the data and the objectives.

Keywords: Data transformation, Statistical analysis, Decision framework, Data pre-processing

Índice

Índice de Tabelas

Índice de Figuras

Abreviaturas

ACP - Análise de Componentes Principais

AED - Análise Exploratória de Dados

ALR – *Additive Log Ratio*

ANCOVA – Análise de covariância

ANOVA - Análise de variância

CLR – *Center Log Ratio*

DVS - Decomposição do Valor Singular

DBN - *Deep Belief Networks*

ECU - Escala de Comprimento de Unidade

EMQ - Erro Médio Quadrático

FDC - Função de Distribuição Cumulativa

FIV - Fator de Inflação da Variância

ILR – *Isometric Log Ratio*

KNN - *K-Nearest Neighbors*

MVS – Máquina de Vetores de Suporte

RBM - *Restricted Boltzmann Machines*

REQM - Raiz do Erro Quadrático Médio

TLN - Transformação Logarítmica Negativa

TIP - Transformação Integral de Probabilidade

TP - Transformação por Posições

TPA - Transformação por Posições Alinhadas

WAPABA - *WAter PArtition and BAlance*

1. Introdução

As transformações de dados são funções aplicadas para modificar a distribuição ou a escala dos dados originais, de modo a satisfazer pressupostos ou requisitos específicos e permitir análises mais significativas. Estas transformações procuram adaptar os dados a pressupostos estatísticos como a normalidade, a homocedasticidade e a aditividade, facilitando a sua interpretação (Hoyle, 1973) e reduzindo a sua amplitude (Zhang, 2015).

Embora já exista uma vasta investigação sobre técnicas de transformação de dados, este trabalho procura compilar várias transformações, focando não só na teoria, mas também na sua aplicação e análise de situação de utilização. Para facilitar a sua aplicação, foram integrados elementos de programação, nomeadamente exemplos de aplicação das transformações usando a linguagem de programação *Python* e o desenvolvimento de uma ferramenta *web* (https://whichtransformation.com) para o auxílio na escolha da transformação de dados.

A presente dissertação foi assim dividida em oito capítulos, no primeiro capítulo (Capítulo 2) apresenta-se uma breve revisão da literatura e da história das transformações, posteriormente são exploradas as caraterísticas e propriedades das transformações de dados (Capítulo 3), sendo seguida da definição dos factores de decisão para a seleção de transformações no Capítulo 4. Os capítulos subsequentes são dedicados à realização de casos de estudo (Capítulo 5), discussão das limitações metodológicas (Capítulo 6), e apresentação de conclusões e exploração de potenciais direções para trabalho futuro (Capítulo 7 e 8 respetivamente).

O apoio à tomada de decisão na escolha da transformação a utilizar, consoante os objetivos da análise e/ou o conjunto de dados que se pretende analisar, é uma lacuna que esta dissertação tenta preencher.

2. Revisão de Literatura

A utilização de transformações de dados na análise estatística tem sido objeto de extensa investigação e desenvolvimento desde o início do século XX.

Nas décadas de 1920 e 1930, estatísticos como *R.A. Fisher*, *Edwin B. Wilson* e *Margaret M. Hilferty* estabeleceram fundamentos no trabalho sobre transformações estatísticas. *R.A. Fisher* introduziu pela primeira vez a transformação arco seno em 1922, aperfeiçoando-a em 1930 como um método de estabilização da variância.

Em 1931, *Wilson* e *Hilferty* introduziram a transformação da raiz cúbica na distribuição do qui-quadrado, demonstrando que esta podia ser aproximada por uma distribuição normal com média e variância específicas. Esta abordagem provou ser mais exata do que as aproximações anteriores, especialmente para graus de liberdade menores, simplificando significativamente os cálculos estatísticos.

Em 1932, *R.A. Fisher* concebeu a transformação Integral de Probabilidade, o que permitiu tanto utilizar testes de adequação dos dados a distribuições como a combinação de testes de significância.

O artigo de *Bartlett* de 1936, sobre a transformação da Raiz Quadrada na análise de variância (*ANOVA*) abordou questões de não normalidade e heterogeneidade na análise de dados, demonstrando a eficácia desta transformação no tratamento de dados que se desviam dos pressupostos.

Na década de 1940, *Eisenhart* investigou a transformação do Arco Seno da Raiz Quadrada, demonstrando a sua eficácia na estabilização da variância para pequenas amostras (particularmente para n=10). A sua análise revelou o impacto da proporção na variância dos dados transformados, o que estudos posteriores vieram a confirmar as vantagens desta transformação na melhoria das propriedades dos dados de proporção.

Em 1948, *F.N. David* e *N.L. Johnson* expandiram a investigação ao analisarem a transformação integral de probabilidade no contexto da estimativa de parâmetros a partir de dados amostrais. Exploraram as implicações da estimativa de parâmetros no processo de transformação, evidenciando a sua aplicação na uniformização de variáveis aleatórias

para a distribuição uniforme em testes estatísticos, para além de abordarem as implicações estatísticas do uso de momentos amostrais para a estimação de parâmetros.

No início dos anos 1960, *Howard F. Hjelm* e *Raymond C. Norris* realizaram um estudo empírico sobre as distribuições de amostragem de coeficientes de correlação, explorando a exatidão de várias aproximações a essas distribuições. Os seus resultados demonstraram que para amostras de maior dimensão, as distribuições obtidas aproximavam-se significativamente das distribuições teóricas, proporcionando assim um método eficaz para testar a significância dos coeficientes de correlação.

Em 1964, *Box* e *Cox* fizeram um avanço significativo com a obra "*An Analysis of Transformations*", que introduziu a transformação Box-Cox. Esta transformação específica, que envolve a escolha de um parâmetro (λ) para maximizar a função de log-verossimilhança dos dados transformados, visa melhorar a normalidade. Este trabalho teve um impacto duradouro na prática da estatística, tornando-se numa ferramenta fundamental na análise estatística e inspirando numerosas extensões e aplicações nas décadas seguintes.

Na década de 1970, *M.H. Hoyle* (1973) publicou uma análise abrangente sobre múltiplas transformações. Onde explorava métodos para desenvolver transformações adequadas e desafios na apresentação de resultados baseados em dados transformados. Destacou ainda que as análises estatísticas fundamentam-se em pressupostos de aditividade, variância constante e normalidade, que frequentemente exigem transformações para corrigir possíveis desvios destes pressupostos.

B.F.J. Manly, em 1976, introduziu uma nova abordagem utilizando funções exponenciais como alternativa às transformações de potência tradicionais, o que ofereceu vantagens particularmente no tratamento de valores negativos e na aproximação de distribuições normais.

Conover e *Iman* entre 1979 e 1982, desenvolveram uma série de estudos que estabeleceram uma ponte entre algumas estatísticas paramétricas e não paramétricas, proporcionando uma abordagem unificada a diversos métodos estatísticos. Os seus trabalhos sobre o método da Transformação por Posições (TP) em diferentes contextos, incluindo na Análise Discriminante e na Análise de Covariância (ANCOVA), demonstraram a robustez das transformações baseadas em posições.

Em 1982, *Victor M. Guerrero* e *Richard A. Johnson* estenderam o trabalho de *Box* e *Cox* aos modelos de resposta binária, particularmente à regressão logística, permitindo uma transformação do rácio de probabilidade orientada pelos dados e generalizando o modelo logístico padrão.

Nos anos 2000, *In-Kwon Yeo* e *Richard A. Johnson* propuseram uma nova transformação de potência que se aplica tanto a valores positivos como negativos, superando assim as limitações dos métodos existentes como a transformação de *Box-Cox*.

Em 2003, *Egozcue et al.* introduziram transformações de razão isométrica para a análise de dados composicionais. Esta contribuição foi fundamental pois fornece transformações que mantêm a isometria entre o *simplex* e o espaço real, de forma a responder às limitações das abordagens anteriores neste domínio.

Em 2012, *Wang et al.* propuseram a transformação *Log-Sinh* para a normalização de dados e estabilização de variância em modelação hidrológica. *Zhang* em 2015 introduziu uma nova abordagem baseada no modelo cinético cicloidal invertido para dados de séries temporais económicas, enquanto *Michael Greenacre* apresentou em 2023 a transformação *chiPower* como uma alternativa às transformações de *logratio* em análise de dados composicionais.

O impacto das transformações no desempenho de tarefas de classificação em *Machine Learning* também tem sido foco de investigação recente, como demonstrado por de Amorim et al. (2022). O seu estudo sobre 82 conjuntos de dados e 20 modelos de classificação destacou a influência significativa das técnicas de transformação no desempenho dos modelos.

A evolução das transformações estatísticas desde a década de 1920 até ao presente exibe um percurso notável de inovação e adaptação às necessidades crescentes da análise de dados. Começando com os trabalhos fundamentais de *Fisher*, *Wilson* e *Hilferty* nas décadas de 1920 e 1930, passando pelas contribuições significativas de *Eisenhart* nos anos 1940 e a revolucionária transformação *Box-Cox* dos anos 1960, até às abordagens mais recentes de *Yeo-Johnson* e *Egozcue et al.,* observa-se uma progressão constante nas técnicas de transformação.

Este desenvolvimento histórico reflete que não existe propriamente um avanço da teoria estatística, mas sim uma adaptação às necessidades nos diversos campos ou tipo de dados. Desde as primeiras preocupações com a normalidade até às aplicações modernas em *Machine Learning* e análise de dados composicionais, as transformações estatísticas continuarão a evoluir para enfrentar novos desafios.

3. Métodos de Transformações de Dados

Neste capítulo será apresentada uma análise aprofundada de cada método de transformação, incluindo os seus pressupostos e exemplos de aplicação. Para além disso, serão discutidos os critérios de seleção para, que no capítulo seguinte, seja escolhido o método de transformação de dados mais adequado com base nas características específicas dos dados e no objetivo da análise a realizar.

3.1. Transformações de Potência

As transformações de potência são maioritariamente usadas em conjuntos de dados contínuos. A sua importância reside na capacidade de resolver questões como a não normalidade e a assimetria nas distribuições de dados (Stoto & Emerson, 1983). Tendo como principais caraterísticas a preservação da ordem, sendo que a classificação relativa dos valores permanece inalterada após a transformação (Stoto & Emerson, 1983).

3.1.1. Transformação Box-Cox

A transformação *Box-Cox* (algumas vezes também referida como *Cox-Box*), é uma transformação utilizada em estatística para estabilizar variância, linearização e aproximar os dados de uma distribuição normal (Box & Cox, 1964). É frequentemente atribuída a *Tukey* (1957) a apresentação da ideia inicial de que as transformações podem ser consideradas uma classe ou família de funções matematicamente semelhantes (Osborne, 2010).

Esta ideia foi criada por *Box* e *Cox* em 1964 no artigo *"An Analysis of Transformations",* ganhando assim o nome *Box-Cox* (Osborne, 2010) e é atualmente uma das transformações mais usadas.

A transformação foi concebida para dados contínuos e estritamente positivos (não devendo ser aplicada a dados ordinais) e pode ainda ser usada em dados que seguem distribuições não normais, e quando a variância dos dados não for constante em diferentes níveis (Heterocedasticidade) (Box & Cox, 1964). A aplicação da transformação Box-Cox contribui para a aproximação dos dados à normalidade e à homocedasticidade, atendendo

aos pressupostos fundamentais de modelos estatísticos paramétricos, como a regressão linear (Box & Cox, 1964). No entanto, no caso da *ANOVA*, a transformação pode ser problemática, porque por vezes, elimina resultados significativos, ou não altera os resultados, ou então torna os resultados significativos (Field, Miles, & Field, 2012).

A transformação é definida por (1) (Box & Cox, 1964) onde λ é o parâmetro escolhido através do método da Máxima Verossimilhança ou do método *bayesiano*, sendo (2) a operação inversa (Box & Cox, 1964).

$$f(x,\lambda) = \begin{cases} \frac{x^{\lambda}-1}{\lambda}, & \lambda \neq 0 \\ \ln(x), & \lambda = 0 \end{cases} \tag{1}$$

$$f^{-1}(y,\lambda) = \begin{cases} (\lambda y + 1)^{\frac{1}{\lambda}}, & \lambda \neq 0 \\ \exp(y), & \lambda = 0 \end{cases} \tag{2}$$

Os diferentes valores de λ produzem diferentes resultados nos dados (Figura 3.1) a serem transformados como, por exemplo (Osborne, 2010):

- $\lambda = 1$: Não transforma os dados.
- $\lambda = 0$: Transforma os dados utilizando o logaritmo natural. Recomendado para dados que exibem assimetria positiva.
- $\lambda = 0{,}5$: Os dados são transformados de forma a reduzir a assimetria e a estabilizar a variância.
- Valores negativos: Valores como $\lambda = -1$, podem ser úteis para dados altamente enviesados, especialmente quando os dados têm uma cauda longa à direita (Assimetria positiva).

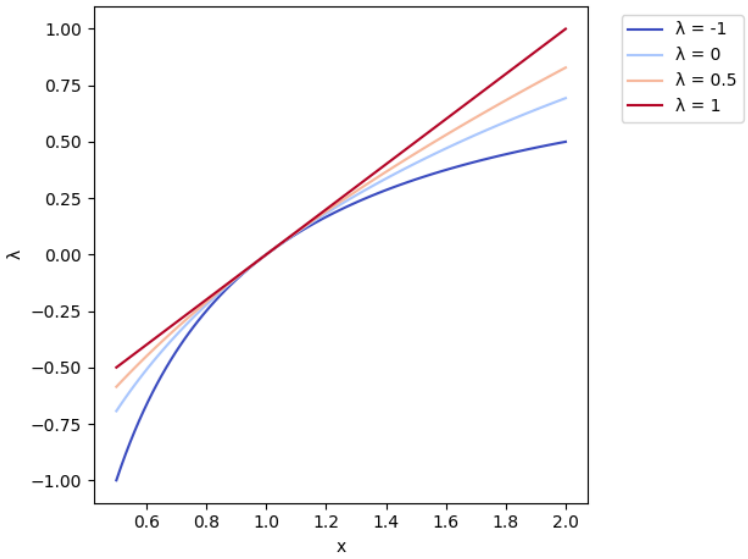

Figura 3.1 - Valores de Lambda de Box-Cox

Considere-se uma amostra com $\bar{X} = 1{,}837$ e $S = 1{,}886$. Após transformar os dados tomando $\lambda = 0{,}243$ as estatísticas dos dados transformados são $\bar{X} \approx 0$ e $S = 1{,}005$.

O exemplo seguinte, mostra como a biblioteca *scipy* da linguagem de programação *Python* para aplicar a transformação *Box-Cox* num conjunto de dados que segue uma distribuição exponencial, onde o parâmetro (λ) é calculado automaticamente pela biblioteca.

```
import numpy as np
from scipy import stats
import matplotlib.pyplot as plt

# Definição de uma semente aleatória para reprodutibilidade
np.random.seed(42)

# Geração de dados que sigam uma distribuição exponencial
data = np.random.exponential(scale=2, size=1000)

# Transformação Box-Cox
transformed_data, lambda_value = stats.boxcox(data)

print(lambda_value)
# 0.290636436945648

plt.subplot(2, 1, 1)
kde_transformed = stats.gaussian_kde(data)
x_range_transformed = np.linspace(min(data), max(data), 100)
plt.hist(data, bins=10, density=True, alpha=0.6)
plt.plot(x_range_transformed, kde_transformed(x_range_transformed), 'r-')
plt.title('Dados originais')
plt.xlabel('x')
```

```
plt.ylabel('Densidade')

plt.subplot(2, 1, 2)
kde_transformed = stats.gaussian_kde(transformed_data)
x_range_transformed = np.linspace(min(transformed_data),
max(transformed_data), 100)
plt.hist(transformed_data, bins=10, density=True, alpha=0.6)
plt.plot(x_range_transformed, kde_transformed(x_range_transformed), 'r-')
plt.title('Dados transformados')
plt.xlabel('x')
plt.ylabel('Densidade')

plt.show()
```

Após a execução do código anterior obteve-se a Figura 3.2, que demonstra que os dados que inicialmente seguiam uma distribuição exponencial, passaram a seguir uma distribuição mais próxima da normal.

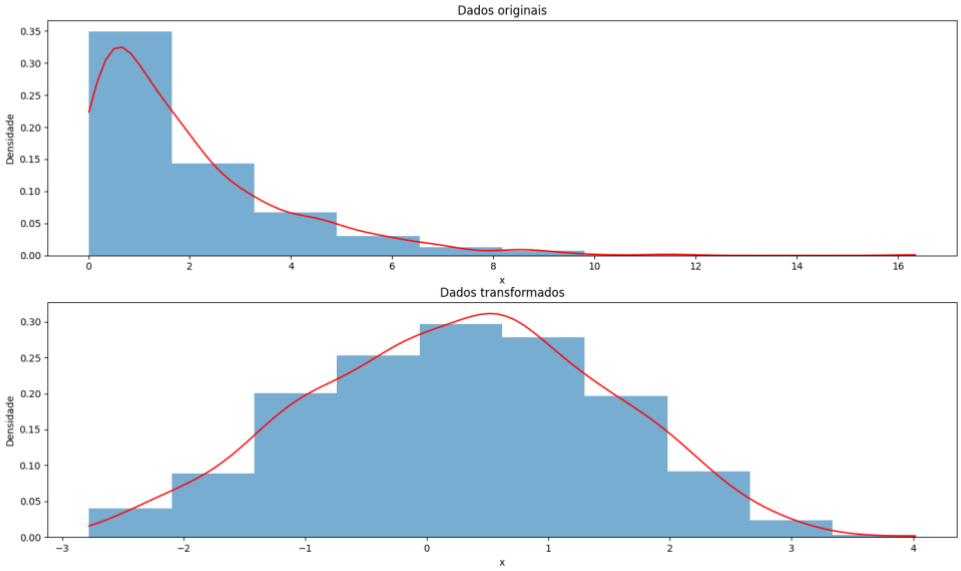

Figura 3.2 - Antes e depois da transformação de Box-Cox

Em termos práticos a transformação *Box-Cox*, tem sido aplicada com sucesso como, por exemplo, na regressão linear, visto que ajuda a estabilizar a variância dos resíduos e melhora o desempenho do modelo, cumprindo os pressupostos do modelo de regressão (Box & Cox, 1964).

Bickel e *Doksum* (1981) aprofundaram a transformação ao investigar a robustez desta em contextos em que o parâmetro λ é desconhecido e precisa de ser estimado, demonstrando que a variância dos estimadores pode aumentar significativamente,

especialmente em modelos estruturados com baixa variância residual. Adicionalmente, foram introduzidos métodos generalizados e robustos que permitem a aplicação da transformação mesmo quando os erros não seguem uma distribuição normal, aumentando a sua aplicabilidade.

O estudo também apresenta uma análise assintótica em situações de baixa variabilidade dos dados, revelando que, nesses casos, a transformação pode tornar-se instável ou pouco informativa.

Na análise de séries temporais, a aplicação da transformação pode ajudar a tornar a série temporal mais estacionária (estacionariedade), sendo frequentemente um requisito para muitos modelos de previsão de séries temporais (Godahewa, Bergmeir, Webb, & Montero-Manso, 2020). *Guerrero* (1993) desenvolveu métodos práticos para selecionar transformações de potência que estabilizam a variância sem depender de modelos prévios, baseando-se no coeficiente de variação e regressões logarítmicas para identificar o parâmetro λ ideal, além de criar um fator de correção de viés que melhora a precisão das previsões ao convertê-las para a escala original.

A transformação foi igualmente aplicada num estudo com vista a tratar a natureza não Gaussiana dos dados provenientes de radares e pluviómetros, conferindo-lhes compatibilidade com os pressupostos teóricos da *krigagem* (para detalhes, ver Erdin, Frei , & Künsch, 2012).

3.1.2. Transformação *Yeo-Johnson*

Esta transformação de potência foi desenvolvida para resolver as limitações de métodos de transformação existentes, que apenas aceitam valores estritamente positivos tais como a transformação de *Box-Cox* e de algumas outras transformações de potência. Por conseguinte, *In-Kwon Yeo* e *Richard A. Johnson* pretendiam criar uma transformação que aceitasse igualmente valores não positivos, fosse bem definida em toda a linha real e eficaz na redução da assimetria e na aproximação à normalidade, assim como na estabilização da variância e na linearização (Yeo & Johnson, 2000). Tal como acontece na transformação de *Box*-Cox, indicada para dados contínuos, não deve ser aplicada a dados ordinais (Yeo & Johnson, 2000).

Ao permitir valores negativos, a transformação proporciona maior flexibilidade no tratamento de dados enviesados e pode ser mais eficaz na obtenção de normalidade ou simetria em vários conjuntos de dados. Tal como na transformação *Box-Cox* o valor λ é estimado usando o método da Máxima Verossimilhança ou o método *Bayesiano* (Yeo & Johnson, 2000).

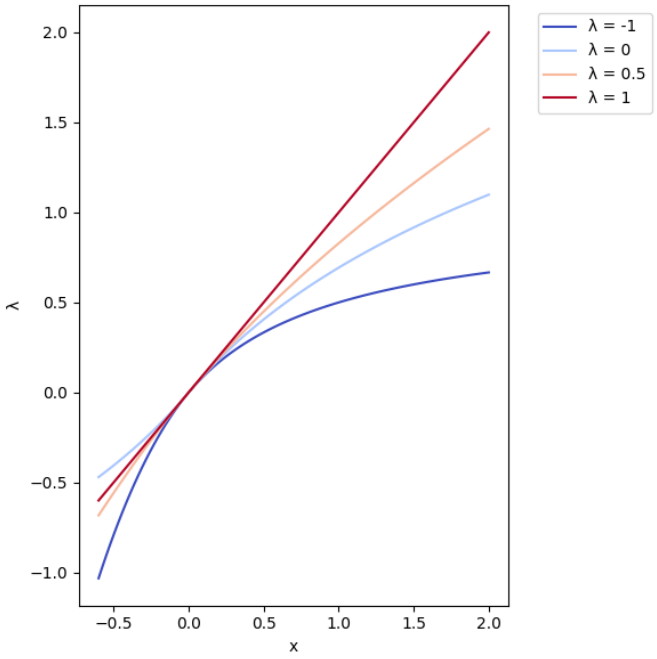

Figura 3.3 - Valores Lambda de Yeo-Johnson

A Figura 3.3 mostra a transformação para alguns valores λ, em que o valor $\lambda = 1$ produz uma relação linear. As transformações com λ < 1 comprimem a cauda direita da distribuição enquanto expandem a cauda esquerda, tornando-as adequadas para transformar distribuições enviesadas para a direita no sentido da simetria. Do mesmo modo, as transformações com λ > 1 são concebidas para tornar mais simétricas as distribuições com enviesamento à esquerda (Raymaekers & Rousseeuw, 2021). A transformação é definida como (3) (Yeo & Johnson, 2000).

$$f(x,\lambda) = \begin{cases} \frac{(x+1)^{\lambda}-1}{\lambda} & \text{se } x \geq 0, \lambda \neq 0 \\ \ln(x+1) & \text{se } x \geq 0, \lambda = 0 \\ -\frac{(-x+1)^{2-\lambda}-1}{2-\lambda} & \text{se } x < 0, \lambda \neq 2 \\ -\ln(-x+1) & \text{se } x < 0, \lambda = 2 \end{cases} \quad (3)$$

Onde λ é o parâmetro escolhido através do método da Máxima Verossimilhança ou do método *bayesiano*.

Considerando uma amostra semelhante à utilizada no exemplo da transformação de *Box-Cox* com $\bar{X} = 1{,}837$ e $S = 1{,}886$, após a transformação com $\lambda = -0{,}41$, a amostra apresenta $\bar{X} \approx 0$ e $S = 1{,}005$. Considerando uma amostra, com valores negativos, $\bar{X} = -0{,}538$ e $S = 5{,}83$, após a aplicação da transformação obtêm-se $\bar{X} \approx 0$ e $S = 1$ padronizando assim a amostra.

Utilizando novamente a biblioteca *scipy*, o exemplo seguinte demonstra como a transformação pode ser utilizada em *Python*, utilizando novamente uma distribuição exponencial, tal como na transformação de *Box-Cox*, o valor lambda (λ) é calculado na própria biblioteca.

```
import numpy as np
from scipy import stats
import matplotlib.pyplot as plt

# Definição de uma semente aleatória para reprodutibilidade
np.random.seed(42)

# Geração de dados que sigam uma distribuição exponencial
data = np.random.exponential(scale=2, size=1000)

# Transformação Yeo-Johnson
transformed_data, lambda_value = stats.yeojohnson(data)

print(lambda_value)
# -0.3726467777332597

plt.subplot(2, 1, 1)
kde_transformed = stats.gaussian_kde(data)
x_range_transformed = np.linspace(min(data), max(data), 100)
plt.hist(data, bins=10, density=True, alpha=0.6)
plt.plot(x_range_transformed, kde_transformed(x_range_transformed), 'r-')
plt.title('Dados originais')
plt.xlabel('x')
plt.ylabel('Densidade')
```

```
plt.subplot(2, 1, 2)
kde_transformed = stats.gaussian_kde(transformed_data)
x_range_transformed = np.linspace(min(transformed_data),
max(transformed_data), 100)
plt.hist(transformed_data, bins=10, density=True, alpha=0.6)
plt.plot(x_range_transformed, kde_transformed(x_range_transformed), 'r-')
plt.title('Dados transformados')
plt.xlabel('x')
plt.ylabel('Densidade')

plt.show()
```

Na Figura 3.4 pode-se observar os gráficos resultantes do código anterior, onde inicialmente os dados seguiam uma distribuição exponencial e após a transformação os dados seguem uma distribuição próxima da normal.

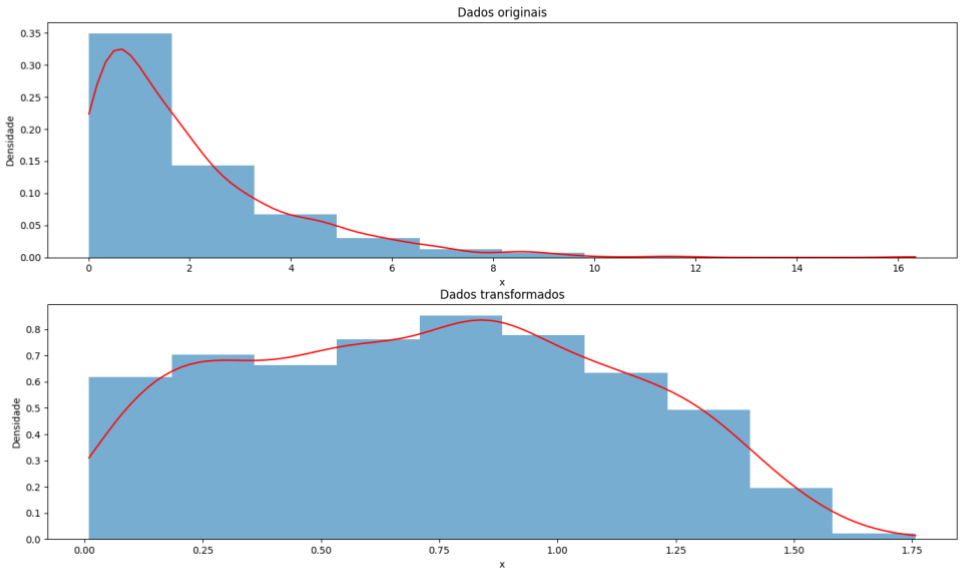

Figura 3.4 - Antes e depois da transformação Yeo-Johnson

Após a transformação, os dados podem ser utilizados em análises de regressão, como a regressão linear e em testes que tenham a normalidade como pressuposto, como referido anteriormente (Yeo & Johnson, 2000) (Othman & Ali, 2021).

A utilização da transformação *Yeo-Johnson* em series temporais levou a previsões mais exatas, com menor erro de previsão quando comparada com a transformação *Box-Cox*, o que indica que a transformação pode aumentar a exatidão dos modelos de previsão (Othman & Ali, 2021).

3.1.3. Transformação de *Tukey Ladder of Powers*

Tukey propôs um conjunto de transformações de potência que poderiam ajudar a linearizar relações, facilitando a análise e a interpretação dos dados (Wicklin, 2024).

As transformações de *Tukey* podem ser aplicadas a dados contínuos multivariados, bem como num contexto de dados bivariados em que uma variável (*Y*) é transformada relativamente a outra variável (Wicklin, 2024) e é normalmente utilizada com dados que podem seguir várias distribuições, incluindo distribuições normais e enviesadas, e a qualquer valor ($[-\infty, +\infty]$) (Tukey, 1957).

Na prática, os vários valores de λ são testados para determinar qual a transformação que produz a relação mais linear nos gráficos de dispersão ou maximiza o coeficiente de correlação entre *Y* e *X* transformados. As transformações foram concebidas principalmente para valores positivos de *Y*, enquanto valores nulos e negativos podem exigir transformações ou alterações específicas (Tukey, 1957).

Sendo que o valor ótimo de λ é aquele que normalmente obtém um maior valor absoluto no coeficiente de correlação, indicando a relação linear mais forte após a transformação (Wicklin, 2024).

As transformações dependem de um expoente, X^{λ}, que pode assumir vários valores reais. Cada valor de λ corresponde a uma transformação diferente, como raízes quadradas, logaritmos ou transformações recíprocas (Wicklin, 2024), que pode ser simplificada como demonstrado em (4).

$$f(x,\lambda) = \begin{cases} -x^{\lambda}, & \lambda < 0 \\ \ln(x), & \lambda = 0 \\ x^{\lambda}, & \lambda > 0 \end{cases} \tag{4}$$

Na implementação seguinte em *Python*, podemos observar todos os lambdas (λ) a serem calculados individualmente, sendo usada a biblioteca *scipy* para o cálculo do coeficiente de correlação de *Pearson*.

```
import numpy as np
from math import log
from scipy.stats import pearsonr
```

```
# Definição de uma semente aleatória para reprodutibilidade
np.random.seed(42)

# Dados gerados aleatoriamente que seguem uma distribuição lognormal
x = np.random.lognormal(0, 1, 1000)
y = np.random.lognormal(0, 1, 1000)

def calc_val(lamb, val):
    if lamb < 0:
        return -val**lamb
    elif lamb == 0:
        return log(val)
    elif lamb > 0:
        return val**lamb

# Cálculo de múltiplos lambdas e respectivos Coeficientes de Correlação
lambdas = [-2, -1, -0.5, 0, 0.5, 1, 2]
correlation_coefficients = []

for lamb in lambdas:
    x_transformed, y_transformed = [], []
    for i in range(len(x)):
        x_transformed.append(calc_val(lamb, x[i]))
        y_transformed.append(calc_val(lamb, y[i]))
    correlation_coefficient, p_value = pearsonr(x_transformed,
y_transformed)
    correlation_coefficients.append(correlation_coefficient)

print(correlation_coefficients)
# [-0.015603654847850876, -0.02407434922132374, -0.03699990536314564, -
0.04039987129344416, -0.01837913354362453, 0.010428312556350527,
0.01746696157994344]
# Onde se pode observar que Lambda=2 é onde se obtêm maior coeficiente de
correlação
```

A transformação é frequentemente comparada com a transformação *Box-Cox*. Ambas visam estabilizar a variância e melhorar a linearidade, embora a transformação de *Box-Cox* tenha como principal objetivo, alcançar a normalidade dos dados, enquanto as de *Tukey* debruçam-se sobre relações lineares (Wicklin, 2024).

Após aplicar a transformação, pode ser efetuada a *ANOVA*, assim como análises de correlação para quantificar a força da relação entre a variável *Y* transformada e a variável *X*.

A análise de regressão pode ser efetuada para modelar a relação e fazer previsões com base nos dados transformados. A Análise Exploratória de Dados (AED) pode ser melhorada através da visualização dos dados transformados para avaliar a linearidade e identificar potenciais valores anómalos (Wicklin, 2024).

3.1.4. Transformação *Guerrero-Johnson*

A transformação de *Guerrero-Johnson* é uma generalização do modelo logístico (também conhecida como regressão logística) que é utilizada quando a variável dependente é binária e as variáveis independentes são discretas ou contínuas e seguem uma distribuição logística (Zedeck, 2014).

Esta contem um parâmetro de transformação (λ), tal como na transformação *Box-Cox*, que é obtido através do método da Máxima Verossimilhança. Quando $\lambda = 0$, a transformação reduz-se ao modelo logístico padrão (Guerrero & Johnson, 1982).

Aplica uma transformação de potência ao rácio de probabilidades (5), permitindo uma maior flexibilidade na modelação de variáveis de resposta binárias (Guerrero & Johnson, 1982).

$$f(p,\lambda) = \begin{cases} \log\left(\frac{p}{1-p}\right), \text{se } \lambda = 0 \\ \frac{\left(\frac{p}{1-p}\right)^{\lambda-1}}{\lambda}, \text{se } \lambda \neq 0 \end{cases} \quad (5)$$

Onde:

p – Probabilidade sucesso (Variável de resposta)

λ – Parâmetro da transformação

Utilizando as bibliotecas *numpy* e *scipy* é possível implementar a transformação *Guerrero-Johnson* em *Python*.

```
import numpy as np
from scipy.optimize import minimize
from scipy.special import expit

# Transformação Guerrero-Johnson
def guerrero_johnson_transform(p, lambda_):
    if lambda_ == 0:
        return np.log(p / (1 - p))
    return ((p / (1 - p)) ** lambda_ - 1) / lambda_

# Cálculo da Máxima Verossimilhança
def log_likelihood(params, X, y):
    beta = params[:-1]
```

```
    lambda_ = params[-1]

    linear_pred = np.dot(X, beta)
    p = expit(linear_pred)
    transformed_p = guerrero_johnson_transform(p, lambda_)
    epsilon = 1e-9
    transformed_p = np.clip(transformed_p, epsilon, 1 - epsilon)
    log_lik = y * np.log(transformed_p) + (1 - y) * np.log(1 -
transformed_p)

    return -np.sum(log_lik)

# Minimização para obtenção do melhor valor de Lambda
def fit_guerrero_johnson(X, y):
    X = np.column_stack((np.ones(X.shape[0]), X))
    initial_params = np.zeros(X.shape[1] + 1)
    initial_params[-1] = 0.0
    result = minimize(log_likelihood, initial_params, args=(X, y),
method='BFGS')

    if result.success:
        return result.x
    else:
        raise RuntimeError("Falha na optimização")

# Exemplo (Variáveis Independentes)
X = np.array([
    [1.0, 0.5],
    [2.0, 1.5],
    [3.0, 3.5],
    [4.0, 4.5],
    [5.0, 5.5]
])

# Dados de exemplo (Variáveis Dependentes)
y = np.array([0, 0, 1, 1, 1])

# Minimização para obtenção do melhor valor de Lambda
fit_params = fit_guerrero_johnson(X, y)

# Melhor lambda obtido
print(fit_params[-1])
# 26.335757184114954
```

A transformação pode ser utilizada para avaliar a qualidade de ajuste em modelos logísticos e para analisar dados de ensaios clínicos (por exemplo, experiências de dose-resposta) através da transformação das variáveis explicativas, bem como em desenhos fatoriais (Guerrero & Johnson, 1982).

3.1.5. Transformação Inversa/Recíproca

A transformação inversa (também conhecida por transformação recíproca) tem sido utilizada há muito tempo e o seu inventor exato é difícil de identificar. Envolve a substituição de dados contínuos pelos seus valores recíprocos e é habitualmente utilizada quando as distribuições têm assimetria positiva (Zedeck, 2014) e estabiliza a variância (Sahai & Khurshid, 2002).

Só pode ser utilizada para valores estritamente positivos ($x > 0$), uma vez que a divisão por zero cria descontinuidade (Bland) e os dados são transformados usando a fórmula $y = \frac{1}{x}$ (Sahai & Khurshid, 2002).

Considerando uma amostra de dados, $\bar{X} = 57{,}83$ e $S = 854{,}69$. Após a transformação obtemos $\bar{X} = 0{,}048$ e $S = 0{,}008$.

Os dados transformados podem ser utilizados em análises que tenham como pressupostos a normalidade e a estabilização da variância (Bland). Como, por exemplo, regressões ou testes de significância (ex: testes *t*). É de salientar que a transformação recíproca pode ser difícil de interpretar, especialmente quando se comparam grupos, uma vez que inverte a direção das diferenças e os intervalos de confiança dos dados transformados podem ser contraintuitivos (Bland).

Em *Python*, a transformação pode ser realizada sem o uso de quaisquer bibliotecas adicionais.

```
import numpy as np

# Definição de uma semente aleatória para reprodutibilidade
np.random.seed(42)

# Geração de dados que sigam uma distribuição exponencial
data = np.random.exponential(scale=1.0, size=1000)

# Transformação Inversa
data_transformed = [1/data[i] for i in range(len(data))]

# Impressão dos valores transformados
print(data_transformed)
```

3.1.6. Transformação da Raiz Quadrada

A transformação da Raiz Quadrada é realizada aplicando a raiz quadrada a valores não negativos, sendo adequada para dados que sigam uma distribuição de *Poisson*, binomial (Freeman & Tukey, 1950) ou com assimetria positiva (Norris & Aroian, 2004).

A transformação é principalmente usada em dados discretos (Freeman & Tukey, 1950) (Pek, Wong, & Wong, 2020) mas também pode ser usada em dados contínuos (Pek, Wong, & Wong, 2020) com vista a estabilizar a variância e a obter a normalidade (Freeman & Tukey, 1950).

Se algumas das observações forem muito pequenas (particularmente zero), é provável que a estabilização da variância seja alcançada pelas transformações na forma $f(x) = \sqrt{x + \frac{1}{2}}$ (Bartlett, 1936) ou na forma de *Anscombe*, $f(x) = \sqrt{x + \frac{3}{8}}$, que demonstrou ser igualmente robusta para a estabilização da variância (Hoyle, 1973).

Com o objetivo de avaliar a eficácia dos métodos em cenários com amostras reduzidas, foi conduzida uma simulação de *Monte Carlo* (*n* = 24). A normalidade dos dados transformados foi avaliada através do teste de *Shapiro-Wilk*, adequado para amostras pequenas.

Tabela 3.1 - Teste de normalidade dos 3 métodos (Normal, Bartlett e Anscombe)

Método	**Estatística *W***	**Valor-p**
Normal	0,93	0,12
Bartlett	0,938	0,141
Anscombe	0,937	0,137

Na Tabela 3.1 podemos observar que os resultados são similares na estatística *W* e também no valor-p, não sendo rejeitada a hipótese de normalidade em qualquer dos métodos. As operações necessárias à implementação dos 3 métodos são as seguintes:

```
import numpy as np
from math import sqrt
from scipy import stats
import statistics

# Definição de uma semente aleatória para reprodutibilidade
np.random.seed(42)
```

```
# Transformação de Raiz Quadrada - Método normal
def normal(X):
    return [sqrt(x) for x in X]

# Transformação de Raiz Quadrada - Método Bartlett
def bartlett(X):
    return [sqrt(x + 0.5) for x in X]

# Transformação de Raiz Quadrada - Método Anscombe
def anscombe(X):
    return [sqrt(x + 3/8) for x in X]

# Configurações para geração dos dados e monte-carlo
lambda_rate = 5
sample_size = 24
iterations = 100

normal_results = []
bartlett_results = []
anscombe_results = []

for _ in range(iterations):
    poisson_data = np.random.poisson(lambda_rate, sample_size)

    # Transformações
    normal_transformed = normal(poisson_data)
    bartlett_transformed = bartlett(poisson_data)
    anscombe_transformed = anscombe(poisson_data)

    # Teste de Shapiro-Wilk
    normal_stat = stats.shapiro(normal_transformed).statistic
    bartlett_stat = stats.shapiro(bartlett_transformed).statistic
    anscombe_stat = stats.shapiro(anscombe_transformed).statistic

    normal_results.append(normal_stat)
    bartlett_results.append(bartlett_stat)
    anscombe_results.append(anscombe_stat)

# Método de Monte-Carlo
def monte_carlo_p_value(statistics, observed_stat):
    count = sum(1 for stat in statistics if stat >= observed_stat)
    return count / len(statistics)

# Teste de Shapiro-Wilk
observed_statistic = stats.shapiro(np.random.poisson(lambda_rate,
sample_size)).statistic

# Obtenção dos valores para a tabela
normal_p_value = monte_carlo_p_value(normal_results, observed_statistic)
bartlett_p_value = monte_carlo_p_value(bartlett_results,
observed_statistic)
anscombe_p_value = monte_carlo_p_value(anscombe_results,
observed_statistic)

print(f"Monte Carlo valor-p para o método normal: {normal_p_value:.4f}")
```

```
print(f"Monte Carlo valor-p para o método de Bartlett:
{bartlett_p_value:.4f}")
print(f"Monte Carlo valor-p para o método de Anscombe:
{anscombe_p_value:.4f}")

normal_mean = statistics.mean(normal_results)
bartlett_mean = statistics.mean(bartlett_results)
anscombe_mean = statistics.mean(anscombe_results)

print(f"Estatística Shapiro-Wilk para o método normal:
{normal_mean:.4f}")
print(f"Estatística Shapiro-Wilk para o método de Bartlett:
{bartlett_mean:.4f}")
print(f"Estatística Shapiro-Wilk para o método de Anscombe:
{anscombe_mean:.4f}")
```

Ao cumprir o pressuposto de normalidade, é possível realizar a *ANOVA* (Freeman & Tukey, 1950), assim como testes estatísticos paramétricos que pressupõem normalidade, como os testes *t* (Everitt & Skrondal, 2010) (Zedeck, 2014) (Bartlett, 1947).

É viável também usar a transformação em regressões lineares (que têm igualmente o pressuposto de normalidade), (Pek, Wong, & Wong, 2020) e no cálculo do Alfa de *Cronbach* e no coeficiente de correlação de *Pearson* (Norris & Aroian, 2004).

A transformação é recomendada especialmente em domínios como a ecologia e a botânica, onde se lida com contagens, como o número de colónias de bactérias ou o número de plantas numa determinada área (Bartlett, 1947) ou medições naturalmente distorcidas (Ahrens, Cox, & Budhwar, 1990).

3.1.7. Transformação da Raiz Cúbica

Originalmente desenvolvida em 1931 por *Wilson* e *Hilferty* no contexto da distribuição qui-quadrado, a transformação da Raiz Cúbica é aplicada para tornar os dados mais próximos da normalidade (Wilson & Hilferty, 1931).

A transformação também pode ser aplicada a variáveis de resposta que assumem valores positivos e negativos, sendo útil no tratamento de distribuições com caudas longas e elevada curtose — especialmente quando o zero é contextualmente relevante para a interpretação dos dados (Cox, 2011) (Hoyle, 1973).

Ao tomar a raiz cúbica dos quantis do qui-quadrado, tipicamente enviesados à direita, os valores transformados tendem a mostrar uma aproximação mais próxima de

uma distribuição normal, como demonstrado nos gráficos de probabilidade normal (Cox, 2011) (Hoyle, 1973). A transformação é representada por $f(x) = x^{\frac{1}{3}}$ ou por $f(x) = \sqrt[3]{x}$ (Hoyle, 1973) (Cox, 2011).

A título de exemplo, considere-se um conjunto de dados que contém os rendimentos (em unidades monetárias) de um conjunto de indivíduos, evidenciando uma assimetria positiva. Para efetuar uma análise estatística, que pressuponha a normalidade, como a regressão linear (Emerson & Stoto, 1982) necessitamos de transformar os dados de modo a aproximá-los de uma distribuição normal, recorrendo à transformação por raiz cúbica (Figura 3.5).

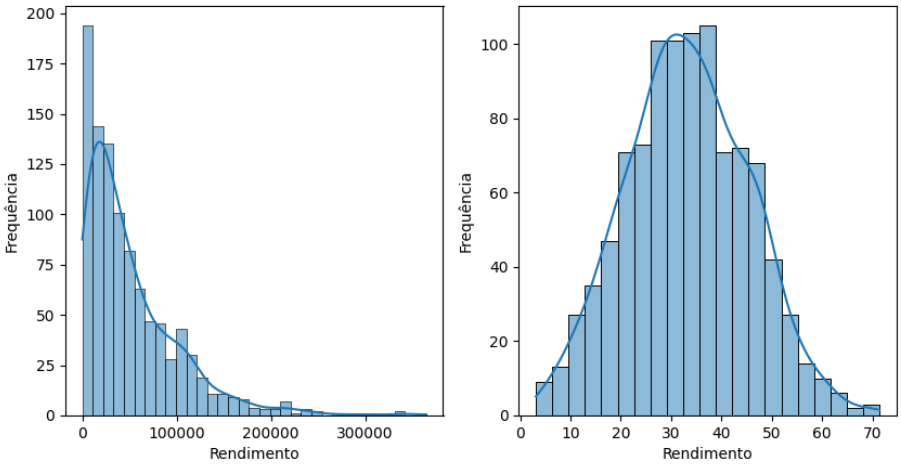

Figura 3.5 - Antes e depois da aplicação da Transformação Raiz Cúbica

As visualizações acima foram desenvolvidas utilizando o seguinte código:

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

# Definição de uma semente aleatória para reprodutibilidade
np.random.seed(42)

# Geração de dados que sigam uma distribuição exponencial
data = np.random.exponential(scale=50000, size=1000)

# Transformação da Raiz Cúbica
data_transformed = np.cbrt(data)

# Construção do gráfico
plt.subplot(1, 2, 1)
sns.histplot(data, kde=True)
plt.xlabel('Rendimento')
```

```
plt.ylabel('Frequência')
plt.subplot(1, 2, 2)
sns.histplot(data_transformed, kde=True)
plt.xlabel('Rendimento')
plt.ylabel('Frequência')
plt.show()
```

A transformação pode ser aplicada na visualização de dados para representar graficamente resíduos e enfatizar as caudas de grandes resíduos positivos e negativos em torno de zero. Esta transformação pode tornar os dados mais fáceis de visualizar e interpretar, mesmo que não conduza necessariamente a distribuições aproximadamente normais (Cox, 2011).

3.2. Transformação Logarítmica

A transformação Logarítmica consiste na utilização de logaritmos para transformar dados quantitativos contínuos (Zedeck, 2014) (Lee, 2020) e é frequentemente aplicada para obter a normalidade, ou estabilização de variâncias ou linearização (Lee, 2020) (Pek, Wong, & Wong, 2020).

Na transformação apenas podem ser usados valores positivos (Lee, 2020) (Pek, Wong, & Wong, 2020) e também pode ajudar a reduzir o impacto dos valores anómalos na extremidade superior da distribuição (Pek, Wong, & Wong, 2020). Assim como deve ser tida em conta quando se lida com distribuições de dados com assimetria positiva (principalmente *Lognormal*) (Lee, 2020).

A título de exemplo, pode observar-se, considerando uma amostra de dados com $\bar{X} = 59{,}33$ e $S = 26{,}49$ que, após a transformação, apresenta para as mesmas estatísticas os valores $\bar{X} = 3{,}93$ e $S = 0{,}63$.

A transformação dos dados para uma distribuição normal permite a aplicação do teste *t*, uma vez que a normalidade é um dos seus pressupostos fundamentais - requisito também essencial para a regressão linear. No caso da regressão linear, a transformação tem o benefício adicional de produzir linearidade (Lee, 2020) (Pek, Wong, & Wong, 2020). No entanto é de realçar que no teste *t* para duas amostras, pode não ser possível abordar o pressuposto de interesse, basta que a variância de uma das amostras seja enviesada (Feng, et al., 2014).

A homogeneidade das variâncias pode ser igualmente obtida pela transformação, possibilita-a de ser utilizada na *ANOVA* (Lee, 2020).

Quando se lida com dados de contagem, especialmente com muitos valores nulos, a transformação Logarítmica não pode ser utilizada, apesar de existir a possibilidade de adicionar uma constante que elimine esse valores nulos ou até valores negativos, essa mesma constante pode interferir nos valores transformados (O'Hara & Kotze, 2010) (Lee, 2020).

Um último ponto negativo é a necessidade de uma operação inversa para facilitar a interpretação dos valores transformados (Lee, 2020).

A transformação Logarítmica, para além do uso da função logarítmica de forma isolada, pode ser usada em diferentes vertentes com propósitos igualmente diferentes.

Introduzida num trabalho intitulado "*A log-sinh transformation for data normalization and variance stabilization*" por *Wang et al.* em 2012 a variante *Log-Sinh*, foi criada para lidar com situações em que as variáveis de previsão apresentam uma assimetria positiva e a dispersão do erro aumenta rapidamente. Depois "abranda", acabando por se tornar constante à medida que os valores das variáveis de previsão aumentam (Wang, Shrestha, Robertson, & Pokhrel, 2012).

A transformação não aceita valores negativos e envolve a aplicação da função hiperbólica seguida da função logarítmica aos dados, ajudando a normalizar os dados e a estabilizar a variância (6) (Wang, Shrestha, Robertson, & Pokhrel, 2012).

$$f(x, a, b) = \frac{1}{b} \log(\sinh[a + bx]) \tag{6}$$

Onde a, b são os parâmetros estimados.

Wang et al. demonstram um caso de estudo aplicando a transformação a um exemplo específico que envolve a simulação do volume de caudal mensal para a bacia hidrográfica do Lago *Hume* utilizando o modelo *WAter PArtition and BAlance* (*WAPABA*) (com dados de 1950 até 1955) e onde foi efetuada a inferência bayesiana dos parâmetros a, b (Wang, Shrestha, Robertson, & Pokhrel, 2012).

Sugerem ainda que para estimativas pontuais ou inferências probabilísticas sejam usados *log(a)* e *log(b)* em vez de *a* e *b*. Isto deve-se ao facto de a transformação ser altamente sensível quando *a* e *b* têm valores pequenos e torna-se menos sensível à medida que aumentam (Wang, Shrestha, Robertson, & Pokhrel, 2012).

A variante Logarítmica Negativa (TLN) é utilizada em vários domínios da estatística e da análise de dados, tendo ganho destaque no início da década de 2000, quando *Joe Whittaker, Chris Whitehead e Mark Somers* estudaram e aplicaram extensivamente a TLN no seu trabalho com dados de análise de crédito, definida em (7) (Whittaker, Whitehead, & Somers, 2005).

A transformação define-se da seguinte maneira:

$$f(x) = \begin{cases} -\log(-x+1) & x \leq 0 \\ \log(x+1) & x > 0 \end{cases} \quad (7)$$

Esta transformação é usada para variáveis contínuas e discretas que exibam características como valores anómalos, distribuições não gaussianas e assimetria. É monótona crescente, sendo útil em questões relacionadas com a não normalidade dos dados. A transformação ajuda a melhorar a distribuição conjunta de co-variáveis e aumenta a eficiência das estimativas na modelação estatística (Whittaker, Whitehead, & Somers, 2005).

Considere-se uma instituição financeira que analisa uma base de dados de análise de crédito para avaliar o risco de incumprimento dos titulares de cartões de crédito. Uma das variáveis de interesse é o saldo da conta, que apresenta valores anómalos, onde se inclui saldos positivos e/ou negativos. A instituição deseja transformar a variável saldo da conta usando a TLN para melhorar a modelação dessa mesma variável. O saldo da conta ao longo dos últimos meses assumiu os seguintes valores $X = \{-500, 100, 300, -200, 50, 700, -100, 400, -300\}$, após a aplicação da TLN, obtemos $X' = \{2.71, 0, 0, 2.30, 0.69, 0, 2.00, 0, 2.48\}$.

Os valores transformados dos saldos das contas apresentam agora uma distribuição mais equilibrada, resolvendo as questões dos valores anómalos e da não normalidade e ajuda a criar uma representação mais adequada dos saldos das contas para posterior análise estatística, como a modelação na regressão Quantílica ou para efeitos de visualização (Whittaker, Whitehead, & Somers, 2005).

Construindo gráficos *Q-Q* comparando os dados que sigam uma distribuição lognormal e as suas transformações (*Log*, *Log-Sinh* e TLN). Os dados transformados com logarítmico alinham-se estreitamente com a linha de normalidade (Figura 3.6).

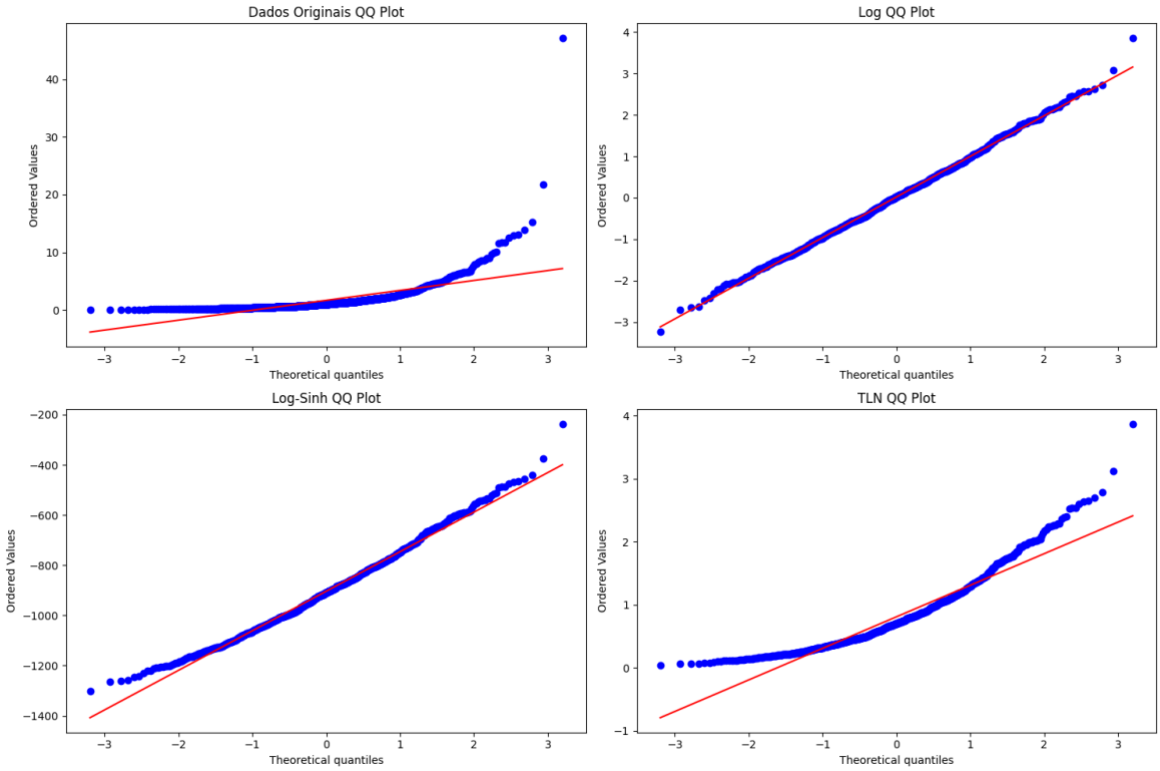

Figura 3.6 - Gráficos Q-Q das transformações Log, Log-Sinh e TLN

A Figura 3.6 foi construída da seguinte forma:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats

# Método para gerar gráficos QQ
def create_qq_plots(data, transformations):
    n_plots = len(transformations) + 1
    n_rows = (n_plots + 1) // 2
    fig = plt.figure(figsize=(15, 5*n_rows))

    plt.subplot(n_rows, 2, 1)
    stats.probplot(data, dist="norm", plot=plt)
    plt.title("Dados Originais QQ Plot")

    for i, (name, func) in enumerate(transformations.items(), 2):
        plt.subplot(n_rows, 2, i)
        transformed_data = func(data)

        stats.probplot(transformed_data, dist="norm", plot=plt)
        plt.title(f"{name} QQ Plot")

    plt.tight_layout()
    return fig

# Transformação Log
def log_transformation(x):
```

```
    if np.any(x <= 0):
        raise ValueError("Todos os valores de entrada devem ser
superiores a 0.")
    return np.log(x)

# Transformação Log-Sinh
def log_sinh_transformation(x):
    a = 0.000472
    b = 0.00559

    sinh_values = np.sinh(a + (b * x))
    if np.any(sinh_values <= 0):
        raise ValueError("A transformação Sinh produziu valores não
positivos; verifique os parâmetros a e b.")

    return (1 / b) * np.log(sinh_values)

# Transformação Logaritmica Negativa
def negative_log_transformation(X):
    data_transformed = []
    for x in X:
        if x > 0:
            data_transformed.append(np.log(x+1))
        elif x <= 0:
            data_transformed.append(-np.log(-x+1))
    return data_transformed

# Definição de uma semente aleatória para reprodutibilidade
np.random.seed(42)

# Geração de dados que seguem uma distribuição lognormal com uma amostra
de tamanho 1000
data = np.random.lognormal(0, 1, 1000)

create_qq_plots(data, {
    "Log": log_transformation,
    "Log-Sinh": log_sinh_transformation,
    "TLN": negative_log_transformation
})

plt.show()
```

A variante *Log-Log*, introduzida por *K. Mather* em 1949, que consiste no logaritmo do logaritmo negativo (8) (Hoyle, 1973).

$$f(x) = \log(-\log x), x \in (0,1) \tag{8}$$

A transformação *Log-Log* requer que os dados sejam proporções entre 0 e 1, e que não existam valores anómalos, que podem distorcer os resultados (Mather, 1949). É frequentemente aplicada a dados univariados que possam ser modelados utilizando distribuições exponenciais ou logísticas, particularmente no contexto da análise de sobrevivência e em bioensaios. A transformação também é utilizada em análises de regressão logística com o objetivo de linearizar a relação entre a variável transformada e o tempo, facilitando a interpretação dos resultados, estabilizando a variância e aproximando a distribuição dos dados da normalidade (Mather, 1949).

A variante *Log-Log Complementar*, é bastante similar à variante anterior, apenas com uma pequena alteração (9) (Everitt & Skrondal, 2010).

$$f(x) = \log(-\log(1-x)), x \in (0,1) \tag{9}$$

A transformação é usada em tabelas de contingência tipicamente quadradas com pares emparelhados de variáveis ordinais que assumam uma distribuição *Gumbel*. Para que posteriormente os dados transformados sejam usados em testes de homogeneidade e de adequação (Everitt & Skrondal, 2010). Tal como na transformação Logarítmica, o objetivo é a normalidade dos dados, na Figura 3.7 são mostrados gráficos *Q-Q* comparando dados (em cima à esquerda) com uma distribuição uniforme, ilustrando o seu desvio da normalidade. As transformações *Log-Log* (em cima à direita) e *Log-Log* complementar (em baixo) melhoram o alinhamento com os quantis normais, particularmente nas caudas, embora nenhuma delas normalize totalmente os dados.

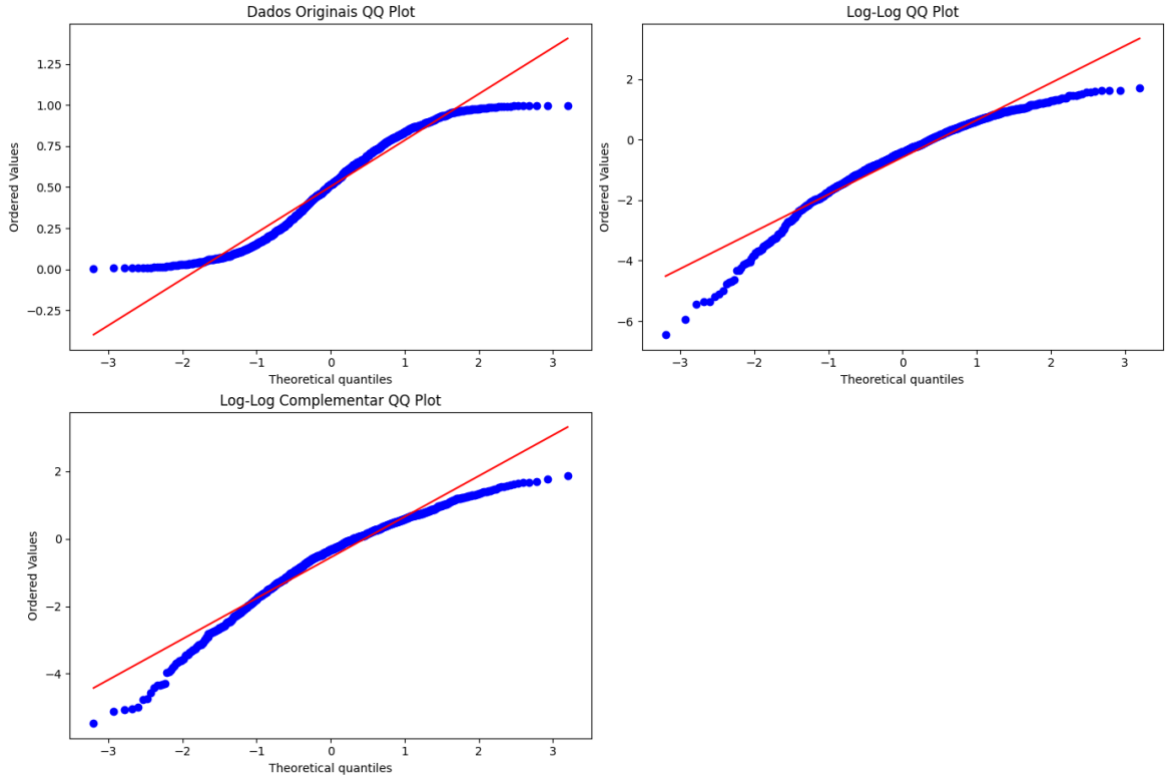

Figura 3.7 - Gráficos Q-Q das transformações Log-Log e Log-Log Complementar

O gráfico acima (Figura 3.7) foi construído utilizando o seguinte código:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats

# Método para gerar gráficos QQ
def create_qq_plots(data, transformations):
    n_plots = len(transformations) + 1
    n_rows = (n_plots + 1) // 2
    fig = plt.figure(figsize=(15, 5*n_rows))

    plt.subplot(n_rows, 2, 1)
    stats.probplot(data, dist="norm", plot=plt)
    plt.title("Dados Originais QQ Plot")

    for i, (name, func) in enumerate(transformations.items(), 2):
        plt.subplot(n_rows, 2, i)
        transformed_data = func(data)

        stats.probplot(transformed_data, dist="norm", plot=plt)
        plt.title(f"{name} QQ Plot")

    plt.tight_layout()
    return fig

# Transformação Log-Log
def log_log_transformation(x):
```

```
    if np.any((x <= 0) | (x >= 1)):
        raise ValueError("Todos os valores de entrada devem estar
compreendidos entre 0 e 1 (exclusivo).")
    return np.log(-np.log(x))

# Transformação Log-Log Complementar
def log_log_complementary_transformation(x):
    if np.any((x <= 0) | (x >= 1)):
        raise ValueError("Todos os valores de entrada devem estar
compreendidos entre 0 e 1 (exclusivo).")
    return np.log(-np.log(1 - x))

# Definição de uma semente aleatória para reprodutibilidade
np.random.seed(42)

# Geração de dados que seguem uma distribuição uniforme com uma amostra
de tamanho 1000
uniform_data = np.random.uniform(0.001, 0.999, 1000)

create_qq_plots(uniform_data, {
    "Log-Log": log_log_transformation,
    "Log-Log Complementar": log_log_complementary_transformation
})

plt.show()
```

Para um contexto de econometria a variante denominada Retorno Logarítmico (*Log Return*) é utilizada para medir a variação percentual do retorno de um ativo financeiro. É expressa por (10) e tem algumas vantagens comparativamente aos retornos brutos (Gundersen, 2022).

$$f(p_t) = \log\left(\frac{p_t}{p_{t-1}}\right) \tag{10}$$

Onde:

p_t - Preço do ativo p no tempo t

p_{t-1} - Preço do ativo p anterior ao tempo t

Uma vez que tem um suporte infinito, podendo assumir qualquer valor real, sendo que esta caraterística permite que os retornos logarítmicos captem melhor os movimentos anómalos nos preços dos ativos. Quando os retornos brutos são próximos de zero, os retornos logarítmicos fornecem uma boa aproximação. Por último, e tal como na

transformação Logarítmica pode ajudar no pressuposto da normalidade (Figura 3.8) (Gundersen, 2022).

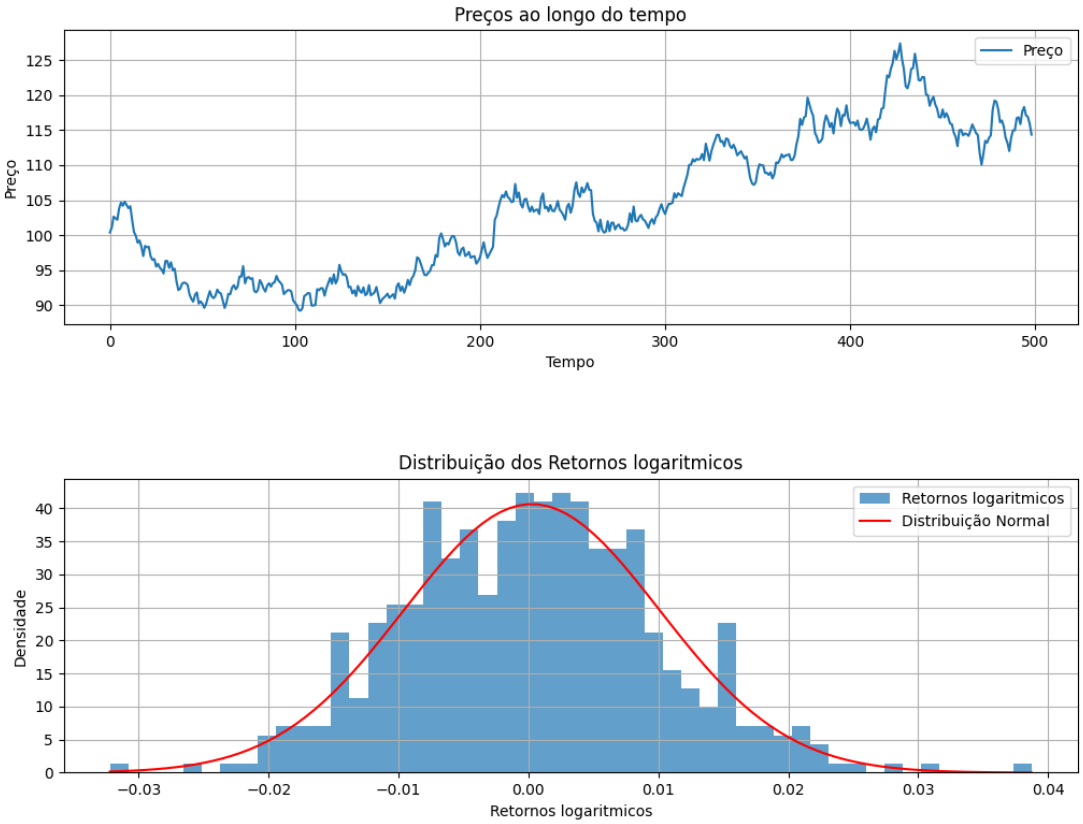

Figura 3.8 - Exemplo de retornos logarítmicos

A Figura 3.8 foi construída usando o código:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy import stats

def analyze_log_returns(prices, window=20):
    # Calculo dos retornos logarítmicos
    log_returns = np.log(prices[1:] / prices[:-1])

    return pd.DataFrame({
        'Price': prices[1:],
        'Log_Returns': log_returns,
    })

# Definição de uma semente aleatória para reprodutibilidade
np.random.seed(42)
n_days = 500
```

```
# Simulação de um random walk com desvio
returns = np.random.normal(0.0002, 0.01, n_days)
prices = 100 * np.exp(np.cumsum(returns))
df_returns = analyze_log_returns(prices)
x = np.linspace(df_returns['Log_Returns'].min(),
df_returns['Log_Returns'].max(), 100)

# Construção da visualização
fig, axes = plt.subplots(2, 1)

axes[0].plot(df_returns.index, df_returns['Price'], label='Preço')
axes[0].set_title('Preços ao longo do tempo')
axes[0].set_xlabel('Tempo')
axes[0].set_ylabel('Preço')
axes[0].grid(True)
axes[0].legend()

axes[1].hist(df_returns['Log_Returns'], bins=50, density=True, alpha=0.7,
label='Retornos logaritmicos')
axes[1].plot(x, stats.norm.pdf(x, df_returns['Log_Returns'].mean(),
df_returns['Log_Returns'].std()),
         'r-', label='Distribuição Normal')
axes[1].set_title('Distribuição dos Retornos logaritmicos')
axes[1].set_xlabel('Retornos logaritmicos')
axes[1].set_ylabel('Densidade')
axes[1].grid(True)
axes[1].legend()

plt.tight_layout()
plt.show()
```

3.3. Normalização e Estandardização de Dados

A normalização de dados é um processo que transforma os dados num intervalo de dados, que normalmente se situa entre 0 e 1 (Schendzielorz, 2020). De forma geral, a normalização é útil para melhorar a precisão e a eficiência de algoritmos que envolvem medições de distância, especialmente ao comparar variáveis em diferentes escalas, preservando assim a distribuição original dos dados enquanto ajusta apenas a escala (Han, Kamber, & Pei, 2011). A estandardização, por outro lado, transforma os dados de modo a terem média de zero e desvio padrão de 1 (Raschka, 2014).

3.3.1. Normalização *Min-Max*

A normalização *Min-Max* realiza uma transformação linear em dados contínuos, convertendo-os para um intervalo fixo (tipicamente entre 0 e 1) (de Amorim, Cavalcanti, & Cruz, 2022) (11). Este método preserva as relações proporcionais entre os valores originais (Gundersen, 2022), porém não atenua o impacto de valores anómalos nos dados e em valores não anómalos tem um efeito semelhante á Normalização *Z-Score* (de Amorim, Cavalcanti, & Cruz, 2022) abordada posteriormente.

$$f(x) = \frac{x - min_x}{max_x - min_x}(novo\_max_x - novo\_min_x) + novo\_min_x \quad (11)$$

Onde:

min_x – Valor mínimo de x

max_x – Valor máximo de x

$novo\_min_x$ – Valor mínimo do intervalo de dados transformados

$novo\_max_x$ – Valor máximo do intervalo de dados transformados

É possível também efetuar a inversão da transformação (12). No entanto, ao realizar a operação inversa não é possível "inverter" um valor fora do intervalo de dados original (Han, Kamber, & Pei, 2011).

$$f^{-1}(y) = (y - novo\_min)\frac{\max - min}{novo\_max - novo\_min} + min \quad (12)$$

Onde:

min_x – Valor mínimo de x

max_x – Valor máximo de x

Considere-se o seguinte exemplo: para um conjunto de dados com $\bar{X} = 23{,}898$ e $S = 17{,}993$ e num intervalo com um valor mínimo de 9,672 e valor máximo de 66,298 obtemos as características $\bar{X} = 0{,}251$ e $S = 0{,}318$.

A biblioteca *Sci-kit Learn* disponível para *Python*, contêm vários métodos de normalização de dados, incluindo a normalização *Min-Max*. Considere-se o seguinte exemplo retirado da documentação da biblioteca (Scikit-Learn, s.d.).

```
from sklearn.preprocessing import MinMaxScaler

# Dados de exemplo
data = [
    [-1, 2],
    [-0.5, 6],
    [0, 10],
    [1, 18]
]

# Normalização Min-Max
data_transformed = MinMaxScaler().fit_transform(data)

# Impressão dos valores transformados
print(data_transformed)
```

A normalização pode ser aplicada em *Machine Learning*, em algoritmos de classificação que envolvem redes neuronais ou medidas de distância, como o *K-Nearest Neighbors* (*KNN*), Máquina de Vetores de Suporte (MVS) e agrupamento (*Clustering*) (Han, Kamber, & Pei, 2011).

Nas redes neuronais, a normalização dos valores de entrada para cada variável ajuda a otimizar o treino do modelo (Han, Kamber, & Pei, 2011).

Para as análises que usam distâncias, a normalização ajuda a evitar que as variáveis em intervalos inicialmente grandes tenham mais peso do que as variáveis em intervalos inicialmente menores (Han, Kamber, & Pei, 2011).

3.3.2. Escala de Comprimento de Unidade

Na Escala de Comprimento de Unidade (ECU) transformam-se as componentes de um vetor de variáveis contínuas de modo que o vetor transformado tenha um comprimento de 1. A ECU transforma o vetor de variáveis e não os valores de cada variável individualmente. Um vetor de variáveis contém os valores de cada variável para uma única observação. Ao transformar para o comprimento da unidade do vetor, dividimos cada vetor de variável pela sua norma (Galli, 2020) e cada caraterística contribui igualmente para os cálculos de distância. Isto evita que as caraterísticas com escalas maiores influenciem desproporcionadamente os resultados, o que é particularmente importante em algoritmos baseados em métricas de distância, como os algoritmos de *KNN* e de agrupamento (Wu, 2017).

A transformação para a norma unitária é conseguida dividindo cada vetor de observação pela Distância de *Manhattan* (Norma L^1) ou pela Distância Euclidiana (Norma L^2) do vetor (13) (Galli, 2020).

$$f(x) = \frac{x}{\|x\|_1} \tag{13}$$

Onde:

$\|x\|_1$ - Norma L^1 (Distância de *Manhattan*) definida por $\|x\|_1 = \sum_{i=1}^{n} |x_i|$ com $x \in \mathbb{R}^n$

ou

$\|x\|_2$ - Norma L^2 (Distância Euclidiana) definida por $\|x\|_2 = \sqrt{\sum_{i=1}^{n} {x_i}^2}$ com $x \in \mathbb{R}^n$

Considerando o vetor observado:

$$x = (3, 4, 0)$$

Usando a Norma L^1 no vetor de observação X, obtemos.

$$\|x\|_1 = |3| + |4| + |0| = 7$$

$$f(x) = \left(\frac{3}{7}, \frac{4}{7}, \frac{0}{7}\right) = \left(\frac{3}{7}, \frac{4}{7}, 0\right)$$

Da mesma forma pode ser usada a Norma L^2.

$$\|x\|_2 = \sqrt{3^2 + 4^2 + 0^2} = \sqrt{25} = 5$$

$$f(x) = \left(\frac{3}{5}, \frac{4}{5}, \frac{0}{5}\right) = \left(\frac{3}{5}, \frac{4}{5}, 0\right)$$

A norma L^1 deve ser usada quando se lida com dados esparsos, quando são necessárias interpretações probabilísticas ou quando a robustez em relação a valores anómalos é uma prioridade. Utiliza-se a norma L^2 quando as propriedades geométricas são importantes, quando se trabalha com dados contínuos ou se utilizam algoritmos que beneficiam de uma contribuição igual das caraterísticas, podendo amplificar a influência de valores anómalos. No entanto, a norma L^1 é geralmente mais robusta a valores anómalos, uma vez que se concentra nos valores absolutos e pode atenuar o seu impacto (Wu, 2017).

Do ponto de vista aplicacional, como já referido anteriormente, pode ser usada em algoritmos de agrupamento (como o *k-means*), e *KNN* e também em MVS, na Análise de Componentes Principais (ACP) e Semelhança de Cosseno (Wu, 2017).

Tal como na Normalização *Min-Max*, a biblioteca *Sci-kit Learn* também pode realizar esta transformação da seguinte forma:

```
from sklearn.preprocessing import Normalizer

# Dados de exemplo
X = [[4, 1, 2, 2],
     [1, 3, 9, 3],
     [5, 7, 5, 1]]

# Escala de Comprimento de Unidade
data_transformed = Normalizer().fit_transform(X)

# Impressão dos valores transformados
print(data_transformed)
```

3.3.3. Normalização *Z-Score* (Estandardização)

A Normalização *Z-Score* pressupõe que os dados se aproximem de uma distribuição normal e transforma os valores amostrais para apresentarem propriedades padronizadas, com média ($\bar{X}$) igual a 0 e desvio padrão (S) igual a 1. A normalização é usada para melhorar o desempenho de análises que requerem dados com características de normalidade, para além de uniformizar variáveis com escalas ou unidades distintas (Jaiswal, 2024), é realizada subtraindo a média a cada valor e dividindo o resultado pelo desvio padrão da distribuição (14) (Raschka, 2014).

$$f(x) = \frac{x - \mu_x}{\sigma_x} \tag{14}$$

Onde:

μ_X – Média da população

σ_X – Desvio padrão da população

É possível também realizar a operação inversa da transformação (15).

$$f^{-1}(y) = y\sigma_y + \mu_y \tag{15}$$

O método do gradiente descendente surge como um importante exemplo (um algoritmo de otimização frequentemente utilizado na MVS, redes neuronais, etc.), em que quando os dados estão normalizados, os pesos podem ser atualizados mais rapidamente (Raschka, 2014).

Outros exemplos podem incluir os algoritmos *KNN* e algoritmos de agrupamento (*k-means*) que utilizem a distância euclidiana e caso se pretenda que todas as variáveis contribuam de forma igual, sendo que os algoritmos de classificação baseados em árvores são provavelmente os únicos classificadores onde a normalização não faz diferença (Raschka, 2014).

Em metodologias como a análise discriminante, a análise em componentes principais (ACP e a análise de componentes principais de *kernel*) o objetivo é encontrar

direções que maximizem a variância, assegurando que essas direções (ou componentes principais) permanecem ortogonais. Para obter resultados exatos, é importante dimensionar as caraterísticas, uma vez que as variáveis com escalas de medição maiores podem dominar e distorcer a análise (Raschka, 2014).

Utilizando a função *StandardScaler* da biblioteca *Sci-kit Learn* podemos realizar a transformação da seguinte forma:

```
from sklearn.preprocessing import StandardScaler


# Dados de exemplo
data = [
    [170, 65, 25],
    [180, 70, 30],
    [160, 55, 20],
    [175, 75, 35],
    [165, 60, 28]
]
# Normalização Z-Score (Estandardização)
data_transformed = StandardScaler().fit_transform(data)

# Impressão dos valores transformados
print(data_transformed)
```

3.3.4. Escala Decimal

A transformação por Escala Decimal normaliza os valores por uma potência de 10, garantido que o maior valor absoluto de cada variável seja inferior a 1 (Han, Kamber, & Pei, 2011). Deve ser usada quando o intervalo de valores do conjunto de dados é conhecido, e com valores contínuos (Han, Kamber, & Pei, 2011) (Sinsomboonthong , 2022). A transformação é definida por (16):

$$f(x, j) = \frac{x}{10^j} \tag{16}$$

onde j é o menor número inteiro tal que $Max(|x|) < 1$, podendo aceitar qualquer valor (positivo, zero ou negativo).

É utilizada para normalizar valores para uma escala comum e para melhorar o desempenho em tarefas de classificação em algoritmos de *Machine Learning*, como por exemplo, redes neuronais (Sinsomboonthong , 2022).

Sem a necessidade de recorrer a bibliotecas externas, apenas a biblioteca *standard math* no *Python* é possível realizar esta transformação:

```
from math import ceil, log10

# Dados de exemplo
X = [65,28,80,77,89,72,65,72,42,13,88,3]

# Obtenção do número inteiro mais pequeno
max_abs = max([abs(x) for x in X])
j = ceil(log10(max_abs))

# Escala Decimal
data_transformed = [x / (10 ** j) for x in X]

# Impressão dos valores transformados
print(data_transformed)
```

3.3.5. Escala Robusta

A Escala Robusta é uma transformação em que se subtrai a mediana e divide-se pelo intervalo interquartil entre o 1º e o 3º quartis (17) (Hvitfeldt, 2024) (de Amorim, Cavalcanti, & Cruz, 2022) e tem como principal objetivo reduzir a influência dos valores anómalos no conjunto de dados (de Amorim, Cavalcanti, & Cruz, 2022).

$$f(x) = \frac{x - \tilde{x}}{Q3_x - Q1_x} \tag{17}$$

Onde:

$\tilde{x}$ - Mediana da amostra de x

$Q1_x$ - 1º Quartil

$Q3_x$ - 3ª Quartil

Normalmente esta transformação é usada em variáveis que contenham dados contínuos e que possam conter dados anómalos, visto que os quartis exteriores não são considerados. No entanto, não é adequada para dados cuja variância seja muito próxima de zero ou igual a zero.

O intervalo predefinido significa que apenas 50% das observações são utilizadas, em caso de os valores anómalos serem necessários para o modelo/análise esta transformação não deve ser usada (Raschka, 2014). A transformação inversa é definida por (18).

$$f^{-1}(y) = y(Q3_y - Q1_y) + \tilde{y} \quad (18)$$

A transformação tem aplicabilidade em algoritmos de classificação que envolvem redes neuronais, algoritmos de agrupamento (*k-means*), como o *KNN* e também em MVS e na Análise Discriminante (de Amorim, Cavalcanti, & Cruz, 2022).

Através da função *RobustScaler* da biblioteca *Sci-kit Learn* podemos usar a transformação utilizando *Python*.

```
from sklearn.preprocessing import RobustScaler

# Dados de exemplo
X = [[ 1., -2.,  2.],
     [ -2.,  1.,  3.],
     [ 4.,  1., -2.]]

# Escala Robusta
data_transformed = RobustScaler().fit_transform(X)

# Impressão dos valores transformados
print(data_transformed)
```

3.3.6. Escala do Máximo Absoluto

Similar à Escala Decimal na Escala de Máximo Absoluto os dados são divididos pelo valor máximo do conjunto de dados (19) e tem como principal objetivo a normalização de dados contínuos, em distribuições enviesadas, de forma que cada variável esteja no intervalo de [-1, 1]. Simultaneamente ajuda a manter a distribuição original dos dados e assegura que as caraterísticas contribuem igualmente para a análise, especialmente em métodos baseados na semelhança (Lima & Souza, 2023) ou quando os dados contêm valores positivos e negativos, uma vez que preserva o sinal dos dados enquanto normaliza a escala (de Amorim, Cavalcanti, & Cruz, 2022).

$$f(x) = \frac{x}{max_x} \quad (19)$$

onde max_x é o valor máximo de x.

Exemplo: Após o uso da transformação, o conjunto de dados usado nas transformações de escala com as características $\bar{X} = 57{,}833$ e $S = 20{,}235$ estas transformam-se em $\bar{X} = 0{,}649$ e $S = 0{,}328$ onde podemos observar que tal como a Escala Decimal, os valores tendem para zero, a única é diferença é que o valor máximo positivo ficará igual.

Em termos de análises, os dados transformados têm aplicabilidade em algoritmos de agrupamento, como o *KNN* e também em MVS, assim como redes neuronais (de Amorim, Cavalcanti, & Cruz, 2022).

Utilizando *Python*, a transformação pode ser realizada usando novamente a biblioteca *Sci-kit Learn*, desta vez com a função *MaxAbsScaler*.

```
from sklearn.preprocessing import MaxAbsScaler

# Dados de exemplo
X = [[ 1., -1.,  2.],
     [ 2.,  0.,  0.],
     [ 0.,  1., -1.]]

# Escala do Máximo Absoluto
data_transformed = MaxAbsScaler().fit_transform(X)

# Impressão dos valores transformados
print(data_transformed)
```

3.4. Transformações Trigonométricas

3.4.1. Transformação do Arco Seno

A transformação do Arco seno foi sugerida pela primeira vez por *Fisher* em 1922 e novamente em 1930, para estabilizar a variância (Hoyle, 1973) assim como para proporções assumirem uma distribuição próxima da normal (Zedeck, 2014) (Wilson, et al., 2010).

É aplicada a dados univariados, especificamente a proporções que variam de 0 a 100% (ou 0 a 1), e em distribuições binomiais, sendo a transformação calculada usando função *arcsin* (Wilson, et al., 2010).

A função *arcsin* encontra-se na biblioteca *numpy* do *Python*, podendo assim esta transformação ser realizada da seguinte forma:

```
import numpy as np

# Dados de exemplo
proportions = [0.1, 0.25, 0.5, 0.75, 0.9, 1.0]

# Transformação do Arco Seno
transformed_data = np.arcsin(proportions)

print(transformed_data)
# [0.10016742 0.25268026 0.52359878 0.84806208 1.11976951 1.57079633]
```

Ao ter a capacidade de estabilizar a variância e de normalização pode cumprir os pressupostos subjacentes da *ANOVA*, de Modelos Generalizados e da Regressão Linear (Wilson, et al., 2010).

3.4.2. Transformação do Arco Seno da Raiz Quadrada (Angular)

A transformação do Arco Seno da Raiz Quadrada, também conhecida por transformação angular foi desenvolvida como um método para transformar proporções em ângulos, estabilizando a variabilidade das proporções, envolvendo a utilização da função arco seno com a raiz quadrada da proporção (20) (Lin & Xu, 2020).

$$f(x,n) = arcsin\left(\sqrt{P}\right) \text{com } P = \frac{x + w_1}{n + w_2} \quad (20)$$

Onde P são as Proporções, n o tamanho da amostra e w_1, w_2 o pesos da variante da transformação.

Na implementação de *Eisenhart* são considerados pesos ($w_1 = w_2 = 0$), mas *Bartlett* (1936) sugeriu que colocar $w_1 = \frac{1}{2}$ e $w_2 = 0$ dando uma variância mais estável. Em 1948, *Anscombe* sugeriu posteriormente uma fórmula ligeiramente diferente com $w_1 = \frac{3}{8}$ e $w_2 = \frac{3}{4}$ (Hoyle, 1973).

Através da Figura 3.9 é possível ver que numa experiência ($n = 10000$), com dados que seguem uma distribuição binomial, que após o uso da transformação em que qualquer dos métodos descritos anteriormente a variância é estabilizada.

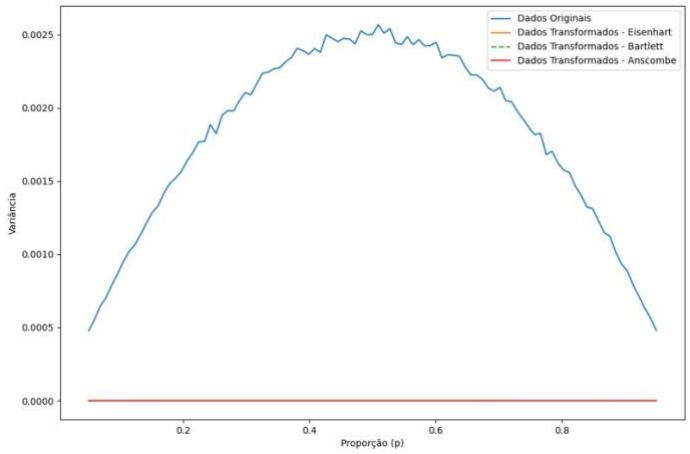

Figura 3.9 - Variância dos Dados Originais vs Dados Transformados

A Figura 3.10 apresenta exatamente os mesmos dados transformados da Figura 3.9, sem incluir os dados originais, de forma a mostrar as diferenças entre os três métodos da transformação que antes apareciam sobrepostos. O método de *Bartlett* demonstra uma eficácia superior tanto ao método de *Anscombe* como ao método de *Eisenhart* na estabilização da variância.

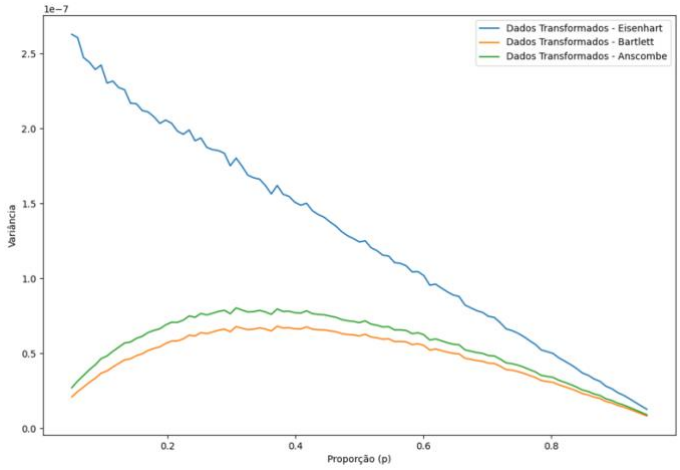

Figura 3.10 - Dados Transformados (Eisenhart, Bartlett e Anscombe)

Para realizar a transformação e consequentemente construir os gráficos anteriores foram utilizadas as bibliotecas *numpy* para os cálculos e a biblioteca *matplotlib* para construir os gráficos.

```
import numpy as np
import matplotlib.pyplot as plt

# Transformação da Raiz Quadrada do Arco Seno
def squared_arc_sine_transform(data, method = 'eisenhart'):
    if method == 'eisenhart':
        p = data / n_samples
    elif method == 'bartlett':
        p = (data+(1/2)) / n_samples
    elif method == 'anscombe':
        p = (data+(3/8)) / (n_samples+(3/4))
    return np.arcsin(np.sqrt(p))

# Geração de dados binomiais
def generate_binomial_data(n_samples, n_trials, p):
    return np.random.binomial(n_trials, p, n_samples) / n_trials

n_samples = 10000
n_trials = 100
p_values = np.linspace(0.05, 0.95, n_trials-1)

original_variances = []
transformed_variances = []
transformed_bartlett_variances = []
transformed_anscombe_variances = []

# Aplicação da transformação aos 3 métodos
for p in p_values:
    data = generate_binomial_data(n_samples, n_trials, p)
    original_variances.append(np.var(data))

    transformed_data = squared_arc_sine_transform(data)
```

```
    transformed_variances.append(np.var(transformed_data))

    transformed_bartlett_data = squared_arc_sine_transform(data,
'bartlett')

transformed_bartlett_variances.append(np.var(transformed_bartlett_data))

    transformed_anscombe_data = squared_arc_sine_transform(data,
'anscombe')

transformed_anscombe_variances.append(np.var(transformed_anscombe_data))

# Construção do gráfico
plt.figure(figsize=(12, 8))
plt.plot(p_values, original_variances, label='Dados Originais')
plt.plot(p_values, transformed_variances, label='Dados Transformados -
Eisenhart')
plt.plot(p_values, transformed_bartlett_variances, label='Dados
Transformados - Bartlett')
plt.plot(p_values, transformed_anscombe_variances, label='Dados
Transformados - Anscombe')
plt.xlabel('Proporção (p)')
plt.ylabel('Variância')
plt.legend()
plt.show()
```

A aplicação da transformação arco seno da raiz quadrada pode tornar os dados mais adequados para determinadas análises estatísticas, especialmente quando as observações possuem o mesmo peso ou representam proporções em escalas (Eisenhart, 1947), como regressões lineares, *ANOVA* e *ANCOVA* ou modelos lineares generalizados, ou quando se lida com acontecimentos raros e de pequenas dimensões de amostras (Lin & Xu, 2020) (Eisenhart, 1947).

3.5. Transformações Exponenciais

3.5.1. Transformação Exponencial

Apresentada por *B. F. J. Manly* em 1976, a transformação fornece uma abordagem alternativa à transformação de potência introduzida por *Box e Cox* em 1964. A transformação Exponencial foi desenvolvida para dados contínuos univariados com capacidade para lidar com dados negativos, o que por vezes noutros métodos de transformação constitui uma limitação, e para dados que sigam uma distribuição assimétrica (Manly, 1976).

A escolha do parâmetro de transformação λ (21) na transformação Exponencial é crucial para obter uma distribuição que se aproxime de uma distribuição normal. Podem ser considerados vários critérios ao selecionar o valor λ adequado para transformar os dados eficazmente (Manly, 1976). Alguns critérios que podem ser considerados (Manly, 1976):

- Normalidade: Quando o objetivo é transformar os dados de forma que a distribuição resultante seja o mais próxima possível de uma distribuição normal.
- Simetria: O parâmetro λ pode ser escolhido para minimizar a assimetria dos dados transformados, tornando a distribuição mais simétrica.
- Curtose: Considerar a curtose da distribuição original dos dados pode orientar a seleção do parâmetro λ para normalizar a curtose nos dados transformados.
- Estimativa de Máxima Verossimilhança: Em alguns casos, a escolha do parâmetro λ com base na maximização da função de verossimilhança para os dados transformados, assumindo a normalidade, pode ser um critério adequado.
- Análise gráfica: Traçar os dados transformados relativamente aos dados originais e inspecionar visualmente a distribuição também pode ajudar a determinar um valor adequado para o parâmetro λ.

$$f(x,\lambda) = \begin{cases} \frac{\exp(\lambda x)-1}{\lambda} & ,\text{se } \lambda \neq 0 \\ x & ,\text{se } \lambda = 0 \end{cases} \quad (21)$$

Onde λ é o parâmetro de transformação.

Cumprindo o pressuposto de normalidade, possibilita-se o uso da transformação em análises de regressão e *ANOVA* assim como em testes *t* (Manly, 1976). Uma possível implementação desta transformação, utilizando $\lambda = 0{,}5$ em *Python* seria:

```
import numpy as np

# Transformação Exponencial
def manly_exponential_transform(x, lambd):
    if lambd != 0:
        return (np.exp(lambd * x) - 1) / lambd
    return x

# Definição de uma semente aleatória para reprodutibilidade
np.random.seed(42)

# Geração de dados que sigam uma distribuição uniforme
data = np.random.exponential(scale=1.0, size=1000)

#Valor lambda de exemplo
lambda_value = 0.5

# Transformação Exponencial
data_transformed = manly_exponential_transform(data, lambda_value)

# Impressão dos valores transformados
print(data_transformed)
```

3.5.2. Transformação Exponencial Dobrada

Proposta em 2003 por *Hans-Peter Piepho*, foi desenvolvida como uma extensão da transformação exponencial. Este método oferece uma solução para situações onde outras transformações normalmente falham devido à presença de valores limite (Piepho, 2003).

A Transformação Exponencial Dobrada é uma transformação usada em cenários que envolvem dados proporcionais univariados.

As proporções, que representam percentagens ou frações, são comuns em vários domínios, como a biologia, a economia e as ciências sociais. No entanto, quando as proporções se aproximam de 0 ou 1, as transformações padrão, como a transformação *Logit* ou do Arco Seno, podem conduzir a distribuições enviesadas ou não normais, afetando a validade das análises estatísticas (Piepho, 2003).

A transformação é indicada para distribuições unimodais moderadamente enviesadas. No entanto, pode ser menos adequada para distribuições altamente enviesadas ou em forma de *J* (Piepho, 2003).

A transformação contém um parâmetro λ que permite alinhar os dados a uma distribuição normal e a estabilização de variância (22), mesmo quando estão presentes valores anómalos. Tal como outras transformações, a Transformação Exponencial Dobrada também pode ser invertida (23).

$$f(x,\lambda) = \begin{cases} \frac{\exp(\frac{\lambda}{2})}{\lambda} \sinh\left(\lambda\left(x-\frac{1}{2}\right)\right), & \text{se } \lambda \neq 0 \\ x-\frac{1}{2} \quad , & \text{se } \lambda = 0 \end{cases} \tag{22}$$

Onde λ é o parâmetro escolhido.

$$f^{-1}(y,\lambda) = \begin{cases} \frac{1}{2} + \frac{1}{\lambda}\sinh^{-1}\left(\frac{y\lambda}{\exp(\frac{\lambda}{2})}\right), & \text{se } \lambda \neq 0 \\ y+\frac{1}{2} \quad , & \text{se } \lambda = 0 \end{cases} \tag{23}$$

O procedimento para estimar o parâmetro λ envolve a maximização da função de Máxima Verossimilhança ou a minimização de um critério escolhido para encontrar um valor ótimo que melhor transforme os dados para aproximar da normalidade (Piepho, 2003).

Em termos aplicacionais, ao estabilizar as variâncias, pode ser usada na *ANOVA* e em modelos lineares mistos (Piepho, 2003).

Tendo uma implementação similar à Transformação Exponencial, pode ser realizada da seguinte forma.

```
import numpy as np

# Transformação Exponencial Dobrada
def folded_exponential_transform(x, lambd):
    if lambd != 0:
        return (np.exp(lambd/2) / lambd) * np.sinh(lambd*(x-0.5))
    return x - 0.5


# Definição de uma semente aleatória para reprodutibilidade
np.random.seed(42)

# Geração de dados que sigam uma distribuição uniforme
data = np.random.exponential(scale=1.0, size=1000)
```

```
#Valor lambda de exemplo
lambda_value = 0.5

# Transformação Exponencial Dobrada
data_transformed = folded_exponential_transform(data, lambda_value)

# Impressão dos valores transformados
print(data_transformed)
```

3.6. Transformações de Dados Composicionais

Os dados composicionais representam descrições quantitativas das partes de um todo, fornecendo informação apenas relativamente ao todo. Exemplos de dados composicionais incluem medições de probabilidades, proporções e percentagens. A principal característica desse tipo de dados é que a sua soma totaliza uma constante. Esses dados são frequentemente encontrados em pesquisas de diversas áreas, como a geologia (Aitchison, 1986).

3.6.1. Transformações Additive/Center/Isometric Log Ratio

As transformações *Addictive Log Ratio* (*ALR*), *Center Log Ratio* (*CLR*) e *Isometric Log Ratio* (*ILR*) são usadas na análise de dados composicionais para transformar dados num espaço linear onde podem ser aplicadas análises estatísticas *standard* (Egozcue, Pawlowsky-Glahn, Mateu-Figueras, & Barceló-Vidal, 2003).

Introduzida por *Aitchison* em 1986 a *ALR* é uma transformação definida como o logaritmo da razão das partes sobre uma dada parte (24). Não é isométrica e carece de interpretação intuitiva devido à ausência de isometria do *simplex* para o espaço real.

$$alr(x) = \left[\ln\left(\frac{x_1}{x_D}\right), \ln\left(\frac{x_2}{x_D}\right), \dots, \ln\left(\frac{x_{D-1}}{x_D}\right)\right] \tag{24}$$

A isometria do *simplex* é a propriedade de transformar dados representados num espaço geométrico que representa distribuições probabilísticas ou proporções sem alterar a sua geometria essencial, ou seja, preservando distâncias, proporções e a estrutura geométrica do espaço (Aitchison, 1986).

Num primeiro passo, calcula-se os *logratios* para cada uma das amostras e, por fim, utilizando os *logratios* calculados, calcula-se as coordenadas *ALR* para cada amostra (Egozcue, Pawlowsky-Glahn, Mateu-Figueras, & Barceló-Vidal, 2003), visando a preservação da soma das partes de uma composição, enquanto revela as diferenças relativas entre elas (Egozcue, Pawlowsky-Glahn, Mateu-Figueras, & Barceló-Vidal, 2003).

A *CLR*, tal com a *ALR*, é outra transformação Logarítmica que procura centrar os dados em torno de zero, facilitando a identificação de desvios de um valor central, e foi

igualmente introduzida por *Aitchison* em 1986 (25). Em primeiro lugar são calculadas as médias geométricas para cada amostra, $g(x)$. Em seguida, os rácios logarítmicos para cada componente utilizando a média geométrica calculada anteriormente e, por fim, os rácios logarítmicos anteriormente calculados são centrados (Egozcue, Pawlowsky-Glahn, Mateu-Figueras, & Barceló-Vidal, 2003).

$$clr(x) = \left[\ln\left(\frac{x_1}{g(x)}\right), \ln\left(\frac{x_2}{g(x)}\right), .., \ln\left(\frac{x_D}{g(x)}\right)\right] = \ln\left(\frac{x}{g(x)}\right) \quad (25)$$

Por último, a *ILR* é uma transformação que preserva as propriedades métricas como ângulos e distâncias no *simplex* (26), permitindo-nos associar ângulos e distâncias no *simplex* com ângulos e distâncias no espaço real (Egozcue, Pawlowsky-Glahn, Mateu-Figueras, & Barceló-Vidal, 2003).

$$ilr(x) = [\langle x, e_1\rangle_a, \langle x, e_2\rangle_a, \dots, \langle x, e_{D-1}\rangle_a] \quad (26)$$

Onde:

x - Composição no simplex S_D.

e_i - Vetores da base no espaço simplex S_D. Cada e_i corresponde a uma direção específica no simplex.

$\langle x, e_1\rangle$ - Produto interno de *Aitchison* entre a composição x e o vetor da base e_i.

D - Dimensão do *simplex*, que corresponde ao número de partes da composição.

A transformação *ILR* representa os dados composicionais pelas suas coordenadas numa base ortonormal escolhida, permitindo um manuseamento e análise eficazes como um conjunto de dados multivariados padrão no espaço real. (Egozcue, Pawlowsky-Glahn, Mateu-Figueras, & Barceló-Vidal, 2003).

Para demonstrar a utilização da *ILR*, vamos considerar o seguinte cenário em que dispomos de dados composicionais de cinco amostras de rocha (Tabela 3.2), que representam as percentagens de três minerais diferentes (A, B e C).

Tabela 3.2 - Dados composicionais de amostras de rocha

Amostra	Mineral A (%)	Mineral B (%)	Mineral C (%)
1	20	30	50
2	25	35	40
3	15	45	40
4	10	50	40
5	30	40	30

No primeiro passo, todas as amostras passam por um processo de "fecho" (27), ou seja, são normalizadas para que a soma das proporções de todos os minerais em cada amostra seja igual a 1 (Tabela 3.3).

$$x_{fecho}[i,j] = \frac{x[i,j]}{\sum_j x[i,j]} \tag{27}$$

Tabela 3.3 - Amostras após processo de "fecho"

Amostra	Mineral A (%)	Mineral B (%)	Mineral C (%)
1	0.2	0.3	0.5
2	0.25	0.35	0.4
3	0.15	0.45	0.4
4	0.1	0.5	0.4
5	0.3	0.4	0.3

Para uma composição de 3 partes ($D = 3$), a ILR projecta os dados num espaço euclidiano bidimensional utilizando vetores de base predefinidos. Cada linha da composição é transformada em duas coordenadas (y_1, y_2) (28 e 29):

Coordenada 1:

$$y_1 = \sqrt{\frac{1}{2}} \ln\left(\frac{x_1}{x_2}\right) \tag{28}$$

Coordenada 2:

$$y_2 = \sqrt{\frac{2}{3}} \ln\left(\frac{g(x_1, x_2)}{x_3}\right) \tag{29}$$

Por último, cada linha do conjunto de dados normalizado é transformada através do cálculo de y_1 e y_2 utilizando as fórmulas referidas anteriormente (28 e 29), tomando a primeira amostra como exemplo:

$$y_1 = \sqrt{\frac{1}{2}} \ln\left(\frac{0{,}2}{0{,}3}\right) = -0{,}287$$

$$y_2 = \sqrt{\frac{2}{3}} \ln\left(\frac{\sqrt{0{,}2 \,.\, 0{,}3}}{0{,}5}\right) = -0{,}583$$

Ao aplicar o mesmo procedimento a todas as amostras obtemos os valores da Tabela 3.4.

Tabela 3.4 - Cálculo das Coordenadas do ILR

Amostra	y_1	y_2
1	-0,287	-0,583
2	-0,238	-0,246
3	-0,777	-0,352
4	-1,138	-0,475
5	-0,203	0,117

A computação da *ILR*, utilizando *Python*, pode ser realizada utilizando as biblioteca *standard statistics* e a biblioteca numpy da seguinte forma:

```
from statistics import geometric_mean
import numpy as np

# Dados de exemplo
x_data = np.array([
    [20, 30, 50],
```

```
    [25, 35, 40],
    [15, 45, 40],
    [10, 50, 40],
    [30, 40, 30]
])

# Normalização de cada amostra (fecho)
x_data_closure = x_data / x_data.sum(axis=1, keepdims=True)
print(x_data_closure)

# Cálculo das coordenadas ILR para cada composição do conjunto de dados
def calculate_ilr(composition):
    y1 = np.sqrt(1 / 2) * np.log(composition[0] / composition[1])
    y2 = np.sqrt(2 / 3) * np.log(geometric_mean(composition[:2]) /
composition[2])
    return [y1, y2]

# Aplicação da transformação ILR para cada amostra
data_transformed = np.array([calculate_ilr(row) for row in
x_data_closure])
print(data_transformed)
```

A ACP pode ser aplicada aos dados depois de estes terem sido transformados e permite reconhecer padrões e relações entre os elementos da composição. Esta técnica, tal como a decomposição do valor singular (DVS), é um método de álgebra linear que decompõe uma matriz complexa em três matrizes de componentes mais simples (Press, Teukolsky, Vetterling, & Flannery, 1992). Quando aplicada às coordenadas ILR, a ACP permite a extração de informações sobre a estrutura e a variabilidade dos dados de composição.

Ambas as análises ajudam a identificar padrões e relações dominantes no conjunto de dados (Egozcue, Pawlowsky-Glahn, Mateu-Figueras, & Barceló-Vidal, 2003). Como o *simplex* apenas pode conter valores positivos, o mesmo se aplica às transformações (Egozcue, Pawlowsky-Glahn, Mateu-Figueras, & Barceló-Vidal, 2003).

3.6.2. Transformação *chiPower*

A transformação *chiPower* foi concebida para tratar dados composicionais, em que as partes somam uma constante, combinando a normalização inerente da distância qui-quadrado com elementos da transformação de potência *Box-Cox* (Greenacre, 2024). Foi desenvolvida como alternativa às transformações *logratio*, para resolver as limitações do tratamento de zeros (sem necessidade de substituição de valores) sem comprometer a

coerência e a isometria e preservando as relações entre as partes da composição (Greenacre, 2024). As transformações *logratio* requerem a substituição dos zeros, o que pode introduzir enviesamento e afetar a interpretação dos dados (Greenacre, 2024).

A transformação define distâncias intra-amostrais que tendem para distâncias de *logratio* em dados estritamente positivos à medida que o parâmetro de potência se aproxima de zero, tornando-a equivalente à transformação para logratios nesse limite.

Ao identificar um valor ótimo para o parâmetro de potência λ, a transformação pode aproximar-se de uma transformação de proporção sem exigir a substituição de zeros nos dados (Greenacre, 2024).

A transformação é definida algoritmicamente, combinando o conceito de λ utilizado na transformação de *Box-Cox* e a normalização do qui-quadrado (Greenacre, 2024).

O cálculo do valor ótimo para o parâmetro de potência λ envolve a identificação de um valor que aproxime os dados transformados o mais possível de uma transformação *logratio* sem a necessidade de substituição por zero (Greenacre, 2024).

Na prática, diferentes valores de λ no intervalo de $0 < \lambda \leq 1$, podem ser explorados para identificar o parâmetro ótimo que equilibra as propriedades da transformação e alinha as distâncias *chiPower* com as distâncias de *logratio*.

A seleção de λ depende do objetivo específico da análise estatística, quer se trate de obter isometria num modelo de *Machine Learning* não supervisionado ou de otimizar o desempenho do modelo em cenários com *Machine Learning* supervisionado.

Como já referido, a transformação é algorítmica, realizada em múltiplos passos, considerando uma matriz de dados *X* de dados composicionais tal que $X = \begin{bmatrix} 2 & 3 & 5 \\ 4 & 1 & 5 \\ 3 & 3 & 4 \end{bmatrix}$.

Como primeiro passo, considerando $\lambda = 0{,}212$ transforma-se a matriz com potência nos valores da matriz $X[\lambda] = x_{ij}^{\lambda}$ obtemos $X[\lambda] = \begin{bmatrix} 1{,}158 & 1{,}262 & 1{,}407 \\ 1{,}342 & 1 & 1{,}407 \\ 1{,}262 & 1{,}262 & 1{,}342 \end{bmatrix}$.

A seguir as linhas da matriz $X[\lambda]$ são fechadas de forma que cada linha da matriz (27) represente uma composição $Y[\lambda]$, ou seja, a soma de todos os valores de cada linha seja 1.

$$Y[\lambda] = \begin{bmatrix} 0{,}303 & 0{,}33 & 0{,}368 \\ 0{,}358 & 0{,}267 & 0{,}375 \\ 0{,}326 & 0{,}326 & 0{,}347 \end{bmatrix}$$

É calculado um vetor com as médias de cada coluna de $Y[\lambda]$: $\bar{y}[\lambda] = \left(\bar{y}[\lambda]_1, \bar{y}[\lambda]_2, \dots, \bar{y}[\lambda]_j\right)$.

$$\bar{y}[\lambda] = (0{,}329 \quad 0{,}308 \quad 0{,}363)$$

Por último, as colunas de $Y[\lambda]$ são dividas pelas raízes quadradas das médias das respetivas colunas (ou seja, a normalização do qui-quadrado) e aplicar o fator $\frac{1}{\lambda}$:

$Z_{ij}[\lambda] = \frac{1}{\lambda} \frac{y_{ij}^{\lambda}}{\sqrt{\bar{y}[\lambda]_j}}$ e assim os dados transformados são-nos dados pelo $Z[\lambda] = \left[Z_{ij}[\lambda]\right]$.

$$Z[\lambda] = \begin{bmatrix} 6{,}383 & 6{,}722 & 6{,}33 \\ 6{,}614 & 6{,}426 & 6{,}358 \\ 6{,}486 & 6{,}707 & 6{,}253 \end{bmatrix}$$

Os passos acima podem ser replicados utilizando o seguinte algoritmo:

```
from math import sqrt
from operator import itemgetter

# Lambda de exemplo
lambd = 0.212

# Dados de exemplo
data = [
    [2, 3, 5],
    [4, 1, 5],
    [3, 3, 4]
]

# Passo 1
Xl = []
for i in range(len(data)):
    Xl.append([data[i][j]**lambd for j in range(len(data[i]))])


# Passo 2
Yl = []
for i in range(len(Xl)):
    Yl.append([Xl[i][j]/sum(Xl[i]) for j in range(len(Xl[i]))])

# Passo 3
col_mean_vector = []
for i in range(len(Yl[0])):
    col_values = list(map(itemgetter(i), Yl))
    col_mean_vector.append(sum(col_values)/len(col_values))
```

```
# Passo 4
data_transformed = []
for i in range(len(Yl)):
    data_transformed.append([])
    for j in range(len(Yl[i])):
        data_transformed[i].append(
            (1/lambd) * (
                (Yl[i][j]**lambd)/sqrt(col_mean_vector[j])
            )
        )

# Impressão dos valores transformados
print(data_transformed)
```

A transformação *chiPower* oferece um elevado nível de coerência e isometria na análise de dados composicionais, o que a torna uma alternativa válida às transformações tradicionais *logratio*, especialmente em contextos de dados de elevada dimensão (Greenacre, 2024). Pode ser utilizada em variadas análises, nomeadamente na Análise de Correspondência para analisar as relações entre variáveis categóricas em dados de composição. Ao transformar os dados, a interpretação dos resultados pode ser melhorada e manter a coerência na análise (Greenacre, 2024).

No contexto da análise de dados composicionais, a ACP pode ser efetuada com dados transformados para identificar padrões e relações entre variáveis, uma vez que permite a normalização e a transformação de dados composicionais para melhorar os resultados (Greenacre, 2024).

Em técnicas de agrupamento, como o agrupamento hierárquico ou o agrupamento *k-means*, podem beneficiar da transformação *chiPower* no agrupamento de pontos de dados com composições semelhantes. Ao transformar os dados, as análises de agrupamento podem ser efetuadas de forma mais eficaz, preservando a estrutura inerente dos dados (Greenacre, 2024).

Em tarefas de *Machine Learning* supervisionado, em que os dados de composição servem como variáveis independentes de uma variável de resposta, a transformação pode ser utilizada para otimizar o ajuste do modelo e a precisão da previsão. Ao otimizar o parâmetro de potência λ na transformação, o desempenho dos modelos de previsão pode ser melhorado (Greenacre, 2024).

3.7. Outras Transformações

3.7.1. Transformação por Posições

A transformação por Posições (TP) é uma transformação conhecida principalmente pela sua utilização em testes não paramétricos, na análise discriminante e em desenhos fatoriais (Zimmerman, 2011) (Conover & Iman, 1981) (Iman & Conover, 1979) (Wobbrock, Findlater, Gergle, & Higgins, 2011), em que os dados podem seguir uma distribuição normal, lognormal, exponencial, uniforme ou de *Cauchy* (Conover & Iman, 1982).

Em 1904, o trabalho *"The proof and measurement of an association between two things."* de *Spearman*, introduziu o conceito de correlação por posição (atualmente conhecido como *ρ* de *Spearman*). *Spearman* substituiu os valores originais dos dados (*X* e *Y*) pelas suas classificações e, a seguir, calculou o coeficiente de correlação de *Pearson* nos dados posicionados. Esta abordagem permitiu-lhe obter a distribuição de probabilidade exata do coeficiente de correlação das posições sem assumir a normalidade bivariada, criando assim um método não paramétrico para testar a independência entre as variáveis emparelhadas *X* e *Y*. Os métodos de análise estatística que utilizavam posições em vez dos dados proliferaram, inspirando assim *Conover* e *Iman* a cunhar o termo "*Rank Transform*" (Transformação por Posições) para incluir as muitas formas de utilização de posicionamento (Conover W. J., 2012). Na TP existem vários métodos de atribuir posições aos dados.

O primeiro método envolve a substituição dos dados originais pelas suas posições correspondentes, atribuindo a posição 1 à observação com menor valor e continuando até à posição *n* para a observação com maior valor (Iman & Conover, 1979). No caso de existirem empates, os mesmos são tratados através da atribuição de posições médias aos valores empatados (Conover & Iman, 1981).

A biblioteca *scipy* providencia a função *rankdata* que permite utilizar este método.

```
from scipy.stats import rankdata

# Dados de exemplo
data = [0, 2, 3, 2, 10, 10, 2, 1, 1, 4, 7, 8]

# Primeiro método da transformação por posições
data_transformed = rankdata(data)

# Impressão dos valores transformados
```

```
print(data_transformed)
#[ 1.   5.   7.   5.  11.5 11.5  5.   2.5  2.5  8.   9.  10. ]
```

No segundo método, quando os dados estão divididos em subconjuntos, o posicionamento pode ser atribuído de duas formas: independente, onde a transformação é aplicada separadamente a cada subconjunto, ou dependente, onde a transformação é aplicada considerando todos os subconjuntos de forma conjunta (Conover & Iman, 1981) (Conover & Iman, 1982).

Por exemplo, considere-se dois grupos de alunos (Turma A e Turma B) com as respetivas notas:

- Turma A: 14, 12, 14, 19
- Turma B: 20, 17, 15, 19

No posicionamento independente, a classificação é feita dentro de cada turma:

- Turma A: 14 (2,5), 12 (1), 14 (2,5), 19 (4)
- Turma B: 20 (4), 17 (2), 15 (1), 19 (3)

No posicionamento dependente, a classificação considera todas as notas juntas:

Global: 14 (2,5), 12 (1), 14 (2,5), 19 (6,5), 20 (8), 17 (5), 15 (4), 19 (6,5)

Neste caso, um aluno que é primeiro lugar na sua turma (posicionamento independente) pode ter uma classificação diferente quando considerado no conjunto total de alunos (posicionamento dependente).

Aplicando o método de transformação por posições ao exemplo anterior, o cálculo é realizado da seguinte forma:

```
import numpy as np
from scipy.stats import rankdata


# Grupos de dados de exemplo
A = np.array([14, 12, 14, 19])
B = np.array([20, 17, 15, 19])

# Calcular as posições independentes
rank_A_independent = rankdata(A)
print(rank_A_independent)
# [2.5 1.  2.5 4. ]

rank_B_independent = rankdata(B)
print(rank_B_independent)
# [4. 2. 1. 3.]
```

```
# Combinar todos os grupos
combined_data = np.concatenate((A, B))

# Calcular as posições dependentes
ranks_combined = rankdata(combined_data)
print(ranks_combined)
# [2.5 1.  2.5 6.5 8.  5.  4.  6.5]
```

O terceiro método, em que os dados podem ser transformados, envolve a re-expressão dos dados antes do posicionamento, como a subtração de uma média ou mediana da amostra e obtendo o valor absoluto da diferença antes do posicionamento. Usando como dados originais, $N = \{7, 9, 11\}$, que nos dá uma média de 9, após o posicionamento obtemos $R = \{2, 0, 2\}$.

Neste terceiro não existe necessidade de recorrer a bibliotecas externas, basta apenas utilizar funções *standard* do *Python*.

```
# Dados de exemplo
data = [7, 9, 11]

# Cálculo da média
mean = sum(data) / len(data)

# Terceiro método da transformação por posições
data_transformed = [abs(x-mean) for x in data]

# Impressão dos valores transformados
print(data_transformed)
# [2.0, 0.0, 2.0]
```

Por último, existe também uma derivação da TP, chamada transformação por Posições Alinhadas (TPA). Enquanto a TP é adequada para analisar dados de desenhos experimentais de dois fatores onde os efeitos de interação não estão presentes e o alinhamento dos dados não é necessário. A TPA é mais apropriada para analisar dados de desenhos fatoriais com um número arbitrário de fatores, onde os efeitos de interação estão presentes e o alinhamento dos dados é necessário. Além disso, a TPA pode lidar com *designs* de medidas repetidas, tornando-a mais versátil em cenários onde as mesmas estão presentes (Wobbrock, Findlater, Gergle, & Higgins, 2011).

Segundo *Wobbrock, Findlater, Gergle, & Higgins*, inicialmente o primeiro passo é o cálculo dos resíduos. Para cada valor de resposta, calcula-se o seu resíduo subtraindo a

média da célula do valor da resposta. Isto dá-lhe uma medida de quanto cada observação se desvia da média da sua célula. Uma célula refere-se a uma combinação específica de níveis de factores no seu desenho experimental. Por exemplo, se tiver dois factores em que o Fator A tem os níveis "a" e "b" e o Fator B tem os níveis "x" e "y", então haverá quatro células possíveis: *ax*, *ay*, *bx* e *by*. A média da célula é a média de todos os valores de resposta que partilham essa combinação específica de níveis de factores.

O segundo passo envolve o cálculo dos efeitos estimados para todos os efeitos principais e interações. Para efeitos unidirecionais (uma interação), o efeito principal estimado para um fator *A* com resposta Y_i é $\overline{A_\iota} - \mu$, onde $\overline{A_\iota}$ é a resposta média para o nível *i* do fator *A*, e μ é a média geral. Para efeitos bidirecionais (duas interações), o valor é dado por $\overline{A_\iota B_J} - \overline{A_\iota} - \overline{B_J} + \mu$, onde $\overline{A_\iota B_J}$ é a resposta média para a combinação do nível *i* do fator *A* e do nível *j* do fator *B*. Para três interações é dado por $\overline{A_\iota B_J C_k} - \overline{A_\iota B_J} - \overline{A_\iota C_k} - \overline{B_J C_k} - \overline{A_\iota} - \overline{B_J} - \overline{C_k} - \mu$, onde $\overline{A_\iota B_J C_k}$ é a resposta média para a combinação do nível *i* do fator *A*, do nível *j* do fator *B* e do nível *k* do fator *C*. O padrão continua para interações de ordem superior.

De seguida, calcula-se a resposta alinhada *Y'* adicionando o resíduo do primeiro passo ao efeito estimado do segundo passo: *Y'* = resíduo + efeito estimado. Este processo de alinhamento isola efetivamente o efeito de interesse enquanto controla outros efeitos no desenho.

O quarto passo envolve a atribuição de classificações médias, denotadas como *Y",* às respostas alinhadas. Classifica-se todos os valores alinhados do menor para o maior, com o menor recebendo a classificação 1. Quando há empate entre *k* valores, a classificação é calculada como a soma das classificações dividida por *k*. Por exemplo, se os valores estiverem empatados nas classificações 2, 3 e 4, cada um receberá a classificação (2+3+4) / 3 = 3.

A etapa final é executar uma *ANOVA* fatorial completa nos dados classificados. Embora se incluam todos os efeitos principais e interações no modelo, apenas se interpretam os resultados para o efeito específico para o qual os dados foram alinhados. Este processo precisa de ser repetido para cada efeito de interesse no seu estudo.

Para se calcular a TPA, utiliza-se a função *rankdata* já utilizada anteriormente.

```
import pandas as pd
from scipy import stats
from itertools import combinations

# Cálculo dos residuos
def compute_residuals(data, response_col, factors):
    cell_means = data.groupby(factors)[response_col].transform('mean')
    return data[response_col] - cell_means

# Calculo do efeitos estimados
def compute_estimated_effects(data, response_col, effect):
    grand_mean = data[response_col].mean()

    # Uma interação
    if len(effect) == 1:
        effect_mean =
data.groupby(effect)[response_col].transform('mean')
        return effect_mean - grand_mean

    # Duas interações
    elif len(effect) == 2:
        AB_mean = data.groupby(effect)[response_col].transform('mean')
        A_mean = data.groupby(effect[0])[response_col].transform('mean')
        B_mean = data.groupby(effect[1])[response_col].transform('mean')
        return AB_mean - A_mean - B_mean + grand_mean

    # Três interacções
    elif len(effect) == 3:
        ABC_mean = data.groupby(effect)[response_col].transform('mean')
        AB_mean =
data.groupby(effect[:2])[response_col].transform('mean')
        AC_mean = data.groupby([effect[0],
effect[2]])[response_col].transform('mean')
        BC_mean = data.groupby([effect[1],
effect[2]])[response_col].transform('mean')
        A_mean = data.groupby(effect[0])[response_col].transform('mean')
        B_mean = data.groupby(effect[1])[response_col].transform('mean')
        C_mean = data.groupby(effect[2])[response_col].transform('mean')
        return (ABC_mean - AB_mean - AC_mean - BC_mean +
                A_mean + B_mean + C_mean - grand_mean)

def align_and_rank(data, response_col, factors, effect):
    # Cálculo dos resíduos e dos efeitos estimados
    residuals = compute_residuals(data, response_col, factors)
    estimated_effects = compute_estimated_effects(data, response_col,
effect)

    # Cálculo da resposta alinhada
    aligned_response = residuals + estimated_effects

    # Cáclulo das posições
    ranked_response = pd.Series(stats.rankdata(aligned_response))

    return aligned_response, ranked_response
```

```
def art_transform(data, response_col, factors):
    results = data.copy()

    # Gerar todos os efeitos possíveis (efeitos principais e interações)
    all_effects = []
    for i in range(1, len(factors) + 1):
        all_effects.extend(list(combinations(factors, i)))

    # Efetuar o alinhamento e a posição para cada efeito
    for effect in all_effects:
        effect_list = list(effect)
        aligned, ranked = align_and_rank(data, response_col, factors,
effect_list)
        effect_name = '*'.join(effect_list)
        results[f'aligned_{effect_name}'] = aligned
        results[f'ranked_{effect_name}'] = ranked

    return results

# Dados de exemplo
sample_data = pd.DataFrame({
    'subject': [1, 1, 1, 1, 2, 2, 2, 2],
    'A': ['a1', 'a1', 'a2', 'a2', 'a1', 'a1', 'a2', 'a2'],
    'B': ['b1', 'b2', 'b1', 'b2', 'b1', 'b2', 'b1', 'b2'],
    'response': [5, 6, 7, 8, 4, 5, 6, 7]
})

# Transformação dos dados
results = art_transform(sample_data, 'response', ['A', 'B'])

# Impressão dos valores transformados
print(list(results['ranked_A*B']))
# [6.5, 6.5, 6.5, 6.5, 2.5, 2.5, 2.5, 2.5]
```

Tal como na TPA, a TP pode também ser utilizada na *ANCOVA* e na *ANOVA*, para tornar a distribuição de dados menos sensível a pressupostos sobre a distribuição subjacente (Wobbrock, Findlater, Gergle, & Higgins, 2011). Na *ANCOVA*, a transformação envolve a aplicação da TP à variável dependente e às co-variáveis antes de a análise ser efetuada. Esta transformação reduz o impacto de valores anómalos, tornando-a mais robusta a violações de pressupostos como o da normalidade (Conover & Iman, 1982) (Akritas, 1990).

As TP podem fornecer uma ligação entre os métodos paramétricos e não paramétricos, permitindo a aplicação de técnicas paramétricas a distribuições de dados não normais (Conover & Iman, 1981). Esta ligação pode ser aplicada em situações em que os pressupostos paramétricos não se podem verificar (Conover & Iman, 1981).

Num estudo de *Conover* e *Iman*, a TP demonstrou uma capacidade de discriminação entre populações normais que igualou a função discriminante linear e a função discriminante quadrática, e ultrapassou-as com populações não normais. Ao contrário dos concorrentes não paramétricos, estes métodos podem ser utilizados com amostras pequenas (Conover & Iman, 1981) (Conover & Iman, 1980) .

Concluindo a TP pode ser usada quando a distribuição dos dados não se aproxima de uma distribuição normal, visto que não requer pressupostos sobre a distribuição (Iman & Conover, 1979).

No entanto, é importante reconhecer as limitações da TP: embora possa fornecer soluções rápidas para muitos problemas estatísticos, as soluções resultantes podem nem sempre ser as melhores ou mais adequadas. Em alguns casos, o problema resolvido pelo método TP pode ser diferente do que é resolvido pelo método paramétrico original (Conover W. J., 2012). Por exemplo, se os dados apresentarem enviesamento, a transformação reduzirá esse enviesamento, o que pode ser indesejável se as distribuições forem significativas (Wobbrock, Findlater, Gergle, & Higgins, 2011).

3.7.2. Transformação de Pontuações Normais

As Pontuações Normais (*Normal Scores*) são os valores esperados das estatísticas ordenadas de uma amostra da distribuição normal (Davison, 2014). São amplamente utilizadas em gráficos (gráficos *Q-Q*) e em testes para avaliar a adequação da distribuição normal aos dados e, como tal, podem ser utilizadas em análises para cumprir o pressuposto de normalidade em análises (Davison, 2014) como a *ANOVA* (Lu & Smith, 1979). O conceito foi introduzido por *Fisher* e *Yates* na primeira edição de "*Statistical Tables for Biological, Agricultural and Medical Research*" publicada em 1938 (Hoyle, 1973).

Fisher e *Yates* sugeriram que as Pontuações Normais devem ser utilizadas para testes de significância que envolvam comparações com dados ordenados, que se assume serem derivados de uma distribuição normal subjacente (Hoyle, 1973).

As Pontuações Normais são similares às posições usadas na TP, em que as posições são substituídas por pontuações. No entanto, a TP torna-se mais eficiente quando a distribuição dos dados está mais próxima da normalidade (Davison, 2014).

A base intuitiva deste procedimento é, evidentemente, o facto de as Pontuações Normais fornecerem uma boa reconstrução dos valores numa escala. Este raciocínio foi formalizado por *Terry* em 1952 e *Hoeffding* em 1951 e 1953 que, entre outras coisas, mostram que, assintoticamente, existe uma correlação perfeita entre os valores esperados das estatísticas de ordem e os valores das variantes que substituem (Hoyle, 1973).

A transformação pode ser realizada por vários métodos distintos: método de *van der Waerden* (1952), o método de *Blom* (1954), o método de *Rankit* (1956), o método dos resultados Esperados de Ordem Normal (1961) e por último o método de *Tukey* (1962) (Solomon & Sawilowsky, 2009) (StatsDirect Limited, 2024).

O método *Van der Waerden* (30), foi desenvolvido por *Willem Jacob van der Waerden* no início dos anos 50, e consiste em calcular os quantis com base na classificação de um determinado valor de pontuação relativamente à dimensão da amostra, em vez de se basear estritamente nas classificações (Solomon & Sawilowsky, 2009). É um teste não paramétrico que testa se a hipótese nula de que todas as funções de distribuição das populações são iguais, mas que pode ser usado também para transformar dados (Conover W. J., 1999).

$$f(r,n) = \Phi\left(\frac{r}{n+1}\right) \tag{30}$$

Onde:

r – Posições dos dados (Ver Transformação por Posições)

n – Tamanho da Amostra

Φ – Função de distribuição da distribuição normal padrão

No método de *Blom*, ao ser aplicada a Transformação Integral de Probabilidade (TIP) a um conjunto de variáveis aleatórias independentes com funções de densidade de probabilidade conhecidas, é obtido um conjunto de variáveis aleatórias uniformemente distribuídas no intervalo de 0 a 1. Estas variáveis são então ordenadas, resultando em distribuições de função *beta*. *Blom* também introduz correções para valores esperados e momentos de estatísticas de ordem, para derivar estimadores lineares quase imparciais e

eficientes para parâmetros populacionais. É representado pela fórmula seguinte (31) (Davison, 2014) (StatsDirect Limited, 2024), sendo os valores recomendados $\frac{3}{8}$ e $\frac{1}{4}$:

$$f(r,n) = \Phi^{-1}\left(\frac{r-\frac{3}{8}}{n+\frac{1}{4}}\right) \tag{31}$$

Onde:

r – Posições dos dados (Ver Transformação por Posições)

n – Tamanho da Amostra

Φ^{-1} – Inversa da função de distribuição da distribuição normal padrão

Desenvolvido por *Bliss, Greenwood e White* em 1956, o método *Rankit* (32) calcula a pontuação normalizada ajustando a classificação da pontuação relativamente ao tamanho da amostra. Este ajuste visa distribuir as pontuações de uma forma que se aproxime de uma distribuição normal, tornando-o eficaz na normalização das pontuações dos testes para fins de análise e comparação (Solomon & Sawilowsky, 2009).

$$f(r,n) = \frac{r-\frac{1}{2}}{n} \tag{32}$$

Onde:

r – Posições dos dados (Ver Transformação por Posições)

n – Tamanho da Amostra

O método dos valores esperados de ordem normal (33) centra-se na determinação das posições médias ou esperadas dos pontos de dados numa amostra quando dispostos por ordem crescente, assumindo uma distribuição normal. Este método envolve o cálculo dos valores esperados das estatísticas da ordem, que fornecem informações sobre a distribuição dos pontos de dados e a população subjacente (Harter, 1961) (Davison, 2014).

$$E\left(Z_{(r)}\right) = e(r,n) = \frac{n!}{(r-1)!\,(n-r)!}\int_{-\infty}^{\infty} z\Phi(z)^{r-1}[1-\Phi(z)]^{n-r}\,\phi(z)\,dz \quad (33)$$

Onde:

r – Posições dos dados (Ver Transformação por Posições)

n – Tamanho da Amostra

$\Phi(z)$ – Função de distribuição da distribuição normal

$\phi(z)$ – Função de Densidade Probabilística

Proposto por *John Tukey* em 1962, o método de *Tukey* (34) calcula a pontuação normalizada ajustando a classificação da pontuação relativamente ao tamanho da amostra, semelhante aos outros métodos de normalização baseados em classificação. Ao aplicar esta fórmula, pretende-se normalizar as pontuações dos testes e torná-las mais comparáveis, aproximando-as de uma distribuição normal.

$$f(r,n) = \frac{r-\frac{1}{3}}{n+\frac{1}{3}} \quad (34)$$

Onde:

r – Posições dos dados (Ver Transformação por Posições)

n – Tamanho da Amostra

Na tabela seguinte (Tabela 3.5), pode-se observar as diferenças entre os métodos de Pontuações Normais, após a suas aplicações no conjunto de dados ($n = 100$) univariável com $\bar{X} = 53{,}15$ e $S = 30{,}632$.

Tabela 3.5 - Diferentes métodos da transformação de Pontuações Normais

	van der Waerden	***Blom***	***Rankit***	**Resultados de Ordem Normal Esperados**	***Tukey***
Média	0,682	-0,03	0,493	0,493	0,493
Desvio Padrão	0,1	1,025	0,291	0,291	0,290
Variância	0,01	1,012	0,085	0,085	0,084

Em termos de média, com exceção do método de *Blom*, todos os métodos são bastante próximos (Tabela 3.5). Sendo, no entanto, o método *Blom* o que mais se aproxima de zero. No entanto, *Blom* apresenta um desvio padrão superior ao dos outros métodos, assim como uma variância superior. Os métodos de *Rankit*, Resultados de Ordem Normal Esperados e de *Tukey*, em termos de desvio padrão e de variância, apresentam valores similares, mas ligeiramente superiores ao método *van der Waerden*.

O estudo de comparação entre os vários métodos de Pontuações Normais foi realizado utilizado *Python*, onde se inclui a utilização da biblioteca *scipy*.

```
from scipy.stats import norm
import numpy as np
from scipy.integrate import quad
import random

# Método para colocar posições nos dados
def rank(X):
    return [sorted(X).index(x)+1 for x in X]

# Método Blom
def blom_transform(X):
    ranks = rank(X)
    n = len(X)
    trans_raw = []
    for x in range(len(ranks)):
        trans_raw.append((ranks[x] - 0.375) / (n + 0.25))
    return norm.ppf(trans_raw)


# Método van der Waerden
def waerden_transform(X):
    ranks = rank(X)
    n = len(X)
    trans_raw = []
    for x in range(len(ranks)):
```

```
        trans_raw.append(ranks[x]/ (n + 1))
    return norm.cdf(trans_raw)

# Método Resultados de Ordem Normal Esperados
def expected_normal_order_score(rank, sample_size):
    def integrand(x):
        return standard_normal_density(x) * (rank - 0.5) / sample_size

    result, _ = quad(integrand, -np.inf, np.inf)
    return result

def expected_normal_transform(X):
    ranks = rank(X)
    n = len(X)
    trans = []
    for x in range(len(ranks)):
        trans.append(expected_normal_order_score(ranks[x], n))
    return trans

def standard_normal_density(x):
    return np.exp(-x**2 / 2) / np.sqrt(2 * np.pi)

# Método Rankit
def rankit_transform(X):
    ranks = rank(X)
    n = len(X)
    trans_raw = []
    for x in range(len(ranks)):
        trans_raw.append((ranks[x] - 0.5) / n)
    return trans_raw

# Método Tukey
def tukey_transform(X):
    ranks = rank(X)
    n = len(X)
    trans_raw = []
    for x in range(len(ranks)):
        trans_raw.append((ranks[x] - (1/3)) / (n+(1/3)))
    return trans_raw

# Transformação de Pontuações Normais
def normal_scores(X, mode):
    data_transformed = []
    if mode == 'waerden':
        data_transformed = waerden_transform(X)
    elif mode == 'blom':
        data_transformed = blom_transform(X)
    elif mode == 'expected':
        data_transformed = expected_normal_transform(X)
    elif mode == 'rankit':
        data_transformed = rankit_transform(X)
    elif mode == 'tukey':
        data_transformed = tukey_transform(X)
    return data_transformed
```

```
# Dados de exemplo
n = [random.randint(1, 100) for _ in range(100)]

# Método van der Waerden
waerden_transformed = normal_scores(n, 'waerden')

# Método Blom
blom_transformed = normal_scores(n, 'blom')

# Método Resultados de Ordem Normal Esperados
expected_transformed = normal_scores(n, 'expected')

# Método Rankit
rankit_transformed = normal_scores(n, 'rankit')

# Método Tukey
tukey_transformed = normal_scores(n, 'tukey')
```

3.7.3. Transformação de Branqueamento (*Whitening)*

A transformação de branqueamento serve para remover a correlação num conjunto de dados, decompondo em componentes, em que separa a parte correlacionada da parte não-correlacionada, onde as componentes não-correlacionadas têm variância unitária e assemelham-se a um processo de ruído branco (Everitt & Skrondal, 2010) (Hossain, 2014).

Antes do início da transformação é necessário verificar se os dados têm média zero, caso não se verifique, subtrai-se a média dos mesmos aos dados (Hossain, 2014).

O primeiro passo para o uso da transformação envolve a determinação dos vetores próprios e dos valores próprios da matriz de covariância original Σ a este processo chama-se decomposição de *Eigen* sendo dada por (35):

$$\Sigma = \Phi \Lambda \Phi^T \tag{35}$$

Onde:

Φ - Matriz de vetores próprios

Λ - Matriz diagonal dos valores próprios

Φ^T - Transposta da matriz Φ

Apesar da decomposição de Eigen ser dada por $\Sigma = \Phi \Lambda \Phi^{-1}$, tendo em conta que as colunas da matriz Φ são ortogonais entre si, sai que $\Phi^{-1} = \Phi^T$ obtendo-se a decomposição $\Phi \Lambda \Phi^{-1}$.

De seguida os dados são transformados utilizando os vetores próprios para obter as componentes não correlacionadas. Este passo produz amostras extraídas de uma distribuição com uma matriz de covariância diagonal (*Y*) usando (36) (Hossain, 2014):

$$\Phi^{-1}X = \Phi^{T}X \tag{36}$$

onde *X* são os dados centrados.

Os dados sem correlação são então redimensionados de modo que as diferentes componentes tenham uma variância unitária, multiplicando-os pela raiz quadrada dos valores próprios (37) (Hossain, 2014).

$$W = \Lambda^{-\frac{1}{2}}\Phi^{T}X \tag{37}$$

Onde:

Λ - Matriz diagonal dos valores próprios

Φ^{T} - Matriz transposta de Φ

X - Dados centrados

Os dados resultantes (W), são agora não correlacionados com variância unitária, assemelhando-se a um vetor de ruído branco (Hossain, 2014).

Considerando o exemplo $\mathrm{X} = \begin{bmatrix} 0{,}371 & -0{,}871 & 1{,}034 \\ 2{,}588 & 0{,}269 & 4{,}511 \\ 3{,}418 & 2{,}432 & 3{,}473 \end{bmatrix}$, após o cálculo das médias vetoriais obtém-se $\mu = [0{,}178 \quad 2{,}456 \quad 3{,}108]$ e subtraindo as mesmas por cada coluna de *X* resulta em $\mathrm{X} = \begin{bmatrix} 0{,}193 & -1{,}049 & 0{,}856 \\ 0{,}132 & -2{,}187 & 2{,}055 \\ 0{,}310 & -0{,}676 & 0{,}365 \end{bmatrix}$.

Seguidamente, calcula-se a matriz de covariância onde se obtém $\bar{Z} = \begin{bmatrix} 0{,}623 & 1{,}359 & 0{,}360 \\ 1{,}359 & 3{,}007 & 0{,}756 \\ 0{,}360 & 0{,}756 & 0{,}228 \end{bmatrix}$.

Utilizando a matriz de covariância Σ calculam-se os valores próprios $\Lambda = [3{,}819 \quad 0{,}04 \quad 10^{-16}]$ e os vetores próprios $\Phi = \begin{bmatrix} -0{,}402 & 0{,}303 & -0{,}863 \\ -0{,}886 & -0{,}365 & 0{,}285 \\ -0{,}227 & 0{,}880 & 0{,}415 \end{bmatrix}$. Após a

aplicação da transformação de Branqueamento obtém-se $W = \begin{bmatrix} -0{,}206 & -0{,}453 & -0{,}116 \\ 1{,}504 & -1{,}802 & 4{,}367 \\ 3{,}771 \times 10^{7} & 1{,}248 \times 10^{7} & 1{,}814 \times 10^{7} \end{bmatrix}$.

Os cálculos anteriores foram realizados em *Python*, e onde a grande maioria dos cálculos foram realizados com a biblioteca numpy e é escrita da seguinte forma:

```
import numpy as np

# Dados de exemplo
X = np.array([
    [0.371, -0.871, 1.034],
    [2.588, 0.269, 4.511],
    [3.418, 2.432, 3.473]
])

# Calculo de média
mean = np.mean(X, axis=1, keepdims=True)

# Centrar dados
X_centered = X - mean

# Cálculo da matriz de covariância
n_samples = X.shape[1]
covariance = np.dot(X_centered, X_centered.T) / n_samples

# Cálculo dos valores próprios e dos vectores próprios
eigenvalues, eigenvectors = np.linalg.eigh(covariance)
idx = eigenvalues.argsort()[::-1]
eigenvalues = eigenvalues[idx]
eigenvectors = eigenvectors[:, idx]

# Cálculo da transformação
Lambda_inv_sqrt = np.diag(1.0 / np.sqrt(eigenvalues))
whitening_matrix = np.dot(Lambda_inv_sqrt, eigenvectors.T)
```

Esta transformação pode ser usada em cenários de séries temporais (dados univariados) que apresentam uma correlação elevada, como nas observações de precipitação obtidas através de radar meteorológico. Neste contexto, o branqueamento de dados envolve a aplicação de uma transformação aos dados para reduzir a correlação entre medições consecutivas. Ao fazê-lo, o processo de branqueamento aumenta o número efetivo de amostras independentes no conjunto de dados, o que pode levar a uma estimativa mais precisa e fiável de parâmetros como os níveis de potência nos retornos de radar (Koivunen & Kostinski, 1999).

A transformação de Branqueamento também pode ser aplicada no reconhecimento de objetos utilizando modelos *Restricted Boltzmann Machines* (*RBM*) e *Deep Belief Networks* (*DBN*), motivado pelas fortes correlações entre pixéis próximos em imagens naturais. Ao branquear os dados, o modelo é encorajado a concentrar-se em correlações de ordem superior, em vez de se distrair com a modelação de correlações bidirecionais, levando potencialmente à descoberta de regularidades mais significativas nas imagens (Krizhevsky, 2009).

Ao reduzir a colinearidade, a transformação pode ser utilizada em regressões lineares, sendo que a não multicolinearidade é um dos pressupostos da mesma, principalmente quando as variáveis dependentes contêm linearidade, tornando-se assim mais fácil realizar regressões com mais precisão. Esta transformação pode ser igualmente aplicada durante o pré-processamento de dados antes da aplicação de métodos como a ACP e análise de componentes independentes (Hossain, 2014).

3.7.4. Transformação *Cinética*

A transformação Cinética baseia-se no modelo cinético cicloidal invertido, utilizado em física. Esta abordagem envolve a conversão de dados não lineares numa forma linear para aumentar a precisão da análise em séries temporais. O método de transformação baseado no modelo cinético cicloidal invertido pode ajudar a alcançar a normalidade dos dados através da linearização de padrões não lineares, aumentando a interpretabilidade dos indicadores económicos e melhorando a precisão das análises estatísticas subsequentes (Zhang, 2015).

O processo começa por normalizar os dados da série temporal e representá-los numa curva cicloide invertida (Zhang, 2015).

Tome-se como exemplo o seguinte conjunto de dados temporais de temperaturas diárias.

$$x = \{27, 30, 25, 28, 28, 25, 29, 30, 28, 28, 30, 32, 26, 23, 25, 27, 23, 30, 26, 26,$$
$$23, 28, 28, 25, 26, 29, 27, 25, 30, 27\}$$

Em primeiro lugar, efetua-se a normalização dos dados usando a Escala do Máximo Absoluto (38):

$$f(x) = \frac{x}{x_{max}} \tag{38}$$

onde max_x é o valor máximo de x.

Por último usando a fórmula (39):

$$F_n = (arcsin(y))^2 \tag{39}$$

Onde y são os dados normalizados no passo anterior.

Obtendo assim uma maior linearização dos dados temporais (Figura 3.11), podendo também ser usada obter o pressuposto de normalidade (Zhang, 2015).

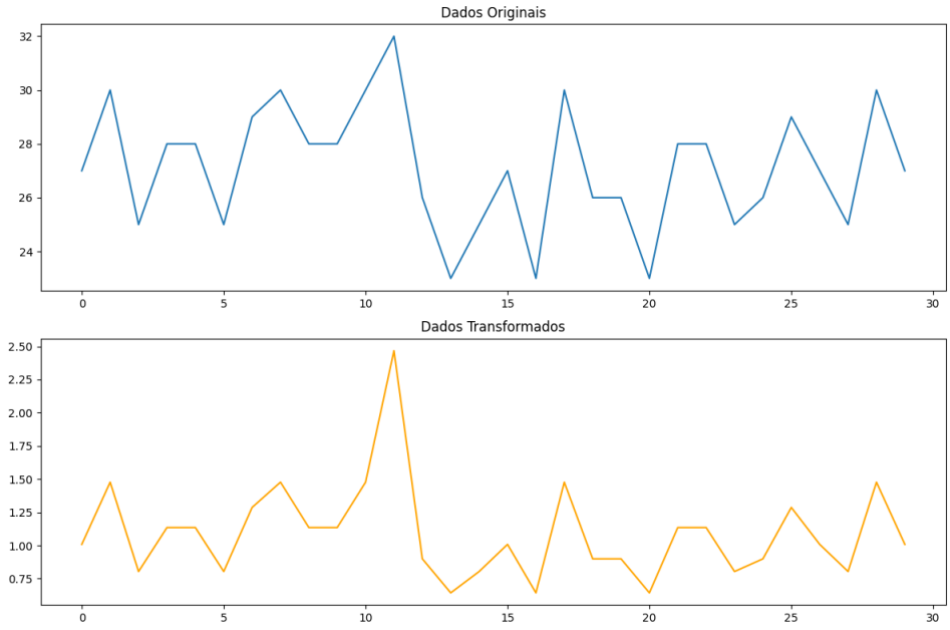

Figura 3.11 - Antes e depois da Transformação Cinética

As visualizações acima foram construídas usando a biblioteca *matplotlib* e computadas da seguinte forma:

```
import numpy as np
import matplotlib.pyplot as plt

# Dados de exemplo
data = [27, 30, 25, 28, 28, 25, 29, 30, 28, 28, 30, 32, 26, 23, 25, 27,
23, 30, 26, 26, 23, 28, 28, 25, 26, 29, 27, 25, 30, 27]
```

```
# Passo 1: Normalização usando a Escala do Máximo Absoluto
normalized_data = np.array(data) / max(data)

# Passo 2: Calcular valores de força
force_values = np.square(np.arcsin(normalized_data))

# Construção do gráfico
plt.figure(figsize=(12, 8))
plt.subplot(2, 1, 1)
plt.plot(data, label='Dados Originais')
plt.title('Dados Originais')
plt.subplot(2, 1, 2)
plt.plot(force_values, label='Dados Transformados', color='orange')
plt.title('Dados Transformados')
plt.tight_layout()
plt.show()
```

O método de transformação aproveita as características de maior desaceleração e do isocronismo da cicloide invertida para decompor o movimento não linear complexo em movimento linear unidirecional. Este processo de decomposição baseia-se em leis mecânicas de física e simplifica os dados para análise (Zhang, 2015).

Os dados transformados podem ser então utilizados para construir um modelo *Autoregressive Integrated Moving Average* (ARIMA), um modelo estatístico para analisar e prever dados de séries temporais (Everitt & Skrondal, 2010), beneficiado com dados linearizados para melhorar a precisão da análise de séries temporais em comparação com os métodos de transformação tradicionais (Zhang, 2015).

Ao linearizar dados não lineares, esta transformação pode ajudar a cumprir os pressupostos da regressão linear, como a normalidade e a homocedasticidade dos resíduos (Zhang, 2015).

3.7.5. Transformação *Probit*

A transformação *Probit* é utilizada para modelar a relação entre uma variável dependente que segue uma distribuição binomial e um conjunto de variáveis explicativas (Armitage, Berry, & Matthews, 2002). Foi utilizada pela primeira vez por *Gaddum* (1933) como um desvio normal equivalente, mas *Bliss* (1935) diminuiu o desvio normal em 5 (40), para tornar muito rara a ocorrência de valores negativos (Hoyle, 1973). Esta transformação tem como objetivos a linearização visto que a transformação *Probit* converte a curva de

regressão em forma sigmoide inerente entre a proporção/probabilidade e as variáveis explicativas numa relação linear. Isto torna os dados mais adequados para a modelação de regressão linear (Armitage, Berry, & Matthews, 2002).

$$f(\Phi^{-1}, p) = 5 + \Phi^{-1}(p), p \in (0,1) \quad (40)$$

Onde:

Φ^{-1} - Função inversa da distribuição de probabilidade cumulativa

p – Proporção

Após a aplicação da transformação *Probit*, os dados podem ser analisados através da análise de regressão *Probit*. Isto permite modelar a relação entre a variável dependente transformada em *probit* e as variáveis explicativas utilizando métodos de regressão linear (Bliss, 1935).

A transformação é aplicada em estudos de resposta quantitativa, como ensaios clínicos (por exemplo, experiências de dose-resposta), em que o objetivo é modelar a probabilidade de resposta em função da dose ou de outros fatores explicativos e pode ser aplicada tanto a dados univariados (uma única proporção ou probabilidade) como a dados multivariados (múltiplas proporções/probabilidades associadas a variáveis explicativas) (Armitage, Berry, & Matthews, 2002).

É importante salientar que a transformação *Probit* pressupõe que a distribuição subjacente dos dados segue uma distribuição binomial ou normal. Pode tratar valores positivos, negativos e zero, embora possam ser necessários ajustamentos para valores de 0 ou 1. É geralmente robusta a valores anómalos, uma vez que a distribuição normal subjacente à escala *Probit* não é tão sensível a valores anómalos (ou discrepantes) como algumas outras distribuições (Armitage, Berry, & Matthews, 2002).

Com a utilização da função *norm* da biblioteca *scipy*, para o cálculo da função inversa da distribuição cumulativa, podemos desenvolver a transformação da seguinte forma:

```
import numpy as np
from scipy.stats import norm
```

```
# Transformação probit
def probit_transformation(p):
    if np.any((p <= 0) | (p >= 1)):
        raise ValueError("Parametro de entrada p deve estar entre [0,
1]")

    return 5 + norm.ppf(p)

# Dados de exemplo
p_values = np.array([0.1, 0.5, 0.9])

# Transformação probit
data_transformed = probit_transformation(p_values)

# Impressão dos valores transformados
print(data_transformed)
# [3.71844843 5.         6.28155157]
```

3.7.6. Transformação *Logit*

A transformação *Logit* é uma função de probabilidade não linear (Zedeck, 2014), utilizada principalmente para transformar dados de resposta binária (como sobrevivência/não sobrevivência ou presença/ausência), num valor contínuo que varia entre $-\infty$ e ∞ (Smith, 2018) e para ajustar as relações em forma de *S* (logísticas) entre a resposta e as variáveis explicativas, linearizando assim os dados (Smith, 2018).

Por outras palavras, a transformação é o logaritmo (*Log*) das probabilidades de ocorrência de um acontecimento (41), onde o *Logit* é o inverso das transformações logísticas (Zedeck, 2014).

$$Logit(p) = f(p) = log\left(\frac{p}{1-p}\right), p \in (0,1) \quad (41)$$

Onde p é a proporção de sobrevivência de cada amostra.

À medida que p tende a 0, $Logit(p)$ tende a $-\infty$ e à medida que p tende a 1, $Logit(p)$ tende a ∞. A função resulta numa curva sigmoide (Figura 3.12) simétrica em torno de $p = 0{,}5$ (Everitt & Skrondal, 2010).

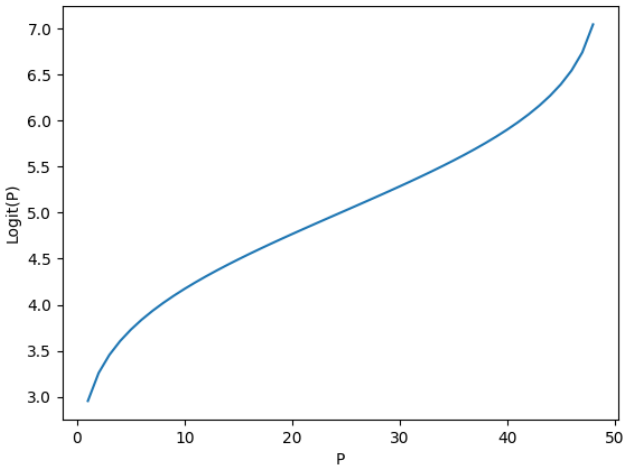

Figura 3.12 - Transformação Logit

Esta transformação ajuda a evitar o agrupamento de valores anómalos do intervalo. Quando se trata de amostras nas quais as proporções p estão próximas de 0 ou 1, resultando em valores transformados positivos ou negativos muito grandes, pode ser aplicada uma versão modificada da transformação. Esta modificação envolve normalmente a adição de $\frac{1}{2n}$ ao numerador e ao denominador, com n a representar a dimensão da amostra (Smith, 2018), a operação inversa desta transformação pode ser obtida através da função logística (42) (Smith, 2018) (Hoyle, 1973).

$$f^{-1}(p) = \frac{exp(p)}{1 + exp(p)} \tag{42}$$

A função logística também pode ser usada como transformação em que as medidas numa escala não-linear são convertidas em probabilidades, sendo normalmente utilizada em regressões logísticas (Zedeck, 2014).

Apesar de similar à transformação *Probit*, mas sem a necessidade se usar a biblioteca *scipy* a computação da transformação pode ser da seguinte forma:

```
import numpy as np

# Transformação logit
def logit_transformation(p):
    if np.any((p <= 0) | (p >= 1)):
        raise ValueError("Parametro de entrada p deve estar entre [0,
1]")
```

```
    return np.log(p / (1 - p))

# Dados de exemplo
p_values = np.array([0.1, 0.5, 0.9])

# Transformação logit
data_transformed = logit_transformation(p_values)

# Impressão dos valores transformados
print(data_transformed)
# [-2.19722458  0.          2.19722458]
```

3.7.7. Transformação Integral de Probabilidade

Em 1932, Fisher introduziu a transformação integral de probabilidade (TIP) para combinar testes de significância e testar a adequação dos dados a distribuições. Embora o seu principal objetivo fosse combinar testes de significância independentes, o seu trabalho lançou as bases para várias aplicações desta transformação na análise estatística (David & Johnson, 1948).

A TIP (também conhecida como a universalidade uniforme) é aplicada inicialmente a dados univariados, em que uma única variável aleatória é transformada (43), tornando assim possível testar a qualidade do ajuste (David & Johnson, 1948).

A transformação também pode ser alargada a dados multivariados, em que múltiplas variáveis aleatórias são consideradas simultaneamente. Neste caso, a transformação pode ajudar a analisar a distribuição conjunta das variáveis e a simplificar as relações entre elas, transformando variáveis correlacionadas em independentes (David & Johnson, 1948).

Em cenários em que existe uma variável de resposta específica (variável dependente) e uma ou mais variáveis preditoras (variáveis independentes), a transformação pode ser aplicada à variável de resposta, considerando os efeitos dos preditores, isto permite analisar o comportamento da variável dependente sob a transformação, facilitando a análise de regressão linear multivariada. Com a TIP é possível converter qualquer distribuição de probabilidade continua, para uma distribuição uniforme com o intervalo (0, 1) (David & Johnson, 1948).

$$f(x) = \int_{-\infty}^{x} p(x)\, dx \tag{43}$$

Onde $p(x)$ é a função de densidade de probabilidade de *X*.

Neste caso, x é uniformemente distribuído no intervalo (0, 1), pelo que esta transformação não induz normalidade, embora estabilize perfeitamente a variância (Figura 3.13) (Hoyle, 1973) (David & Johnson, 1948).

A transformação envolve o cálculo da função de distribuição cumulativa (FDC) da variável aleatória original e, em seguida, a aplicação da inversa da FDC à variável aleatória. No caso de dados multivariados, as variáveis são consideradas simultaneamente.

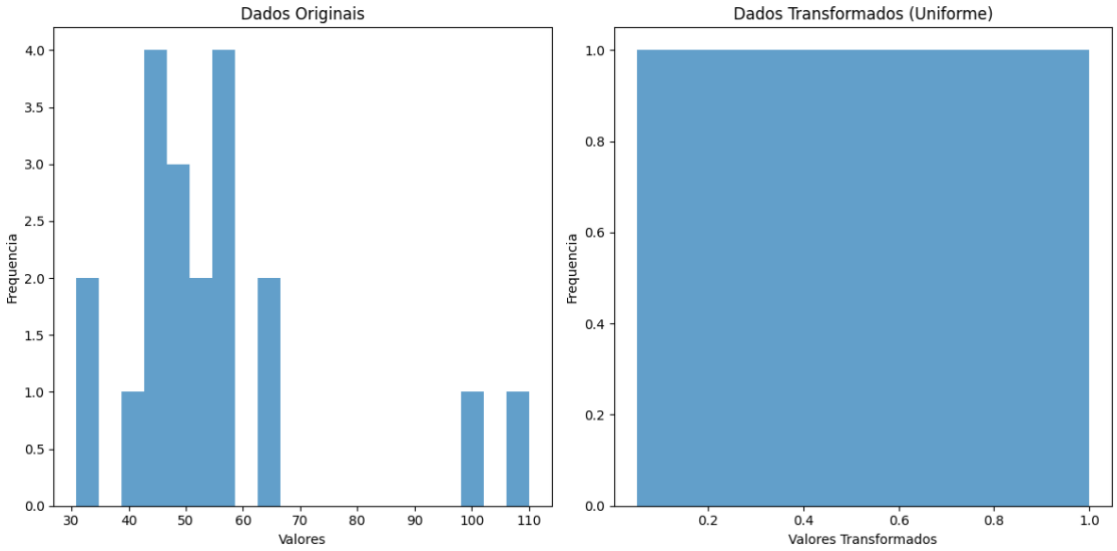

Figura 3.13 - Antes e após a TIP

As visualizações geradas (Figura 3.13) utilizando a biblioteca *matplotlib* da seguinte forma:

```
import numpy as np
import matplotlib.pyplot as plt

# Definição de uma semente aleatória para reprodutibilidade
np.random.seed(42)

# Geração de dados normalmente distribuídos
data = np.random.normal(loc=50, scale=10, size=18)

# Adição de valores anómalos
outliers = np.array([100, 110])
```

```
data_with_outliers = np.concatenate([data, outliers])

# Ordenação dos dados
sorted_data = np.sort(data_with_outliers)

# Cálculo da função de distribuição cumulativa empírica
ecdf = np.arange(1, len(sorted_data) + 1) / len(sorted_data)

# Mapeamento dos dados originais para os dados transformados
transformed_data = np.zeros_like(data_with_outliers)
for original_value in data_with_outliers:
    transformed_data[np.where(sorted_data == original_value)[0][0]] =
ecdf[np.where(sorted_data == original_value)[0][0]]

# Construção do gráfico
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.hist(data_with_outliers, bins=20, alpha=0.7)
plt.title('Dados Originais')
plt.xlabel('Valores')
plt.ylabel('Frequencia')
plt.subplot(1, 2, 2)
plt.hist(transformed_data, bins=20, alpha=0.7)
plt.title('Dados Transformados (Uniforme)')
plt.xlabel('Valores Transformados')
plt.ylabel('Frequencia')
plt.tight_layout()
plt.show()
```

4. Fatores de Decisão

No capítulo anterior, foram apresentadas várias transformações de dados, cada uma delas adaptada para servir objetivos específicos e/ou com dados específicos. Estes objetivos, tal como os pressupostos que muitas análises possuem, são fundamentais para selecionar uma transformação adequada.

Vários atributos dos dados - tais como o seu formato (univariados, multivariados), o seu tipo (contínuo, categórico, proporcional, etc.) e que intervalo têm - devem ser verificados. Por exemplo, certas transformações funcionam melhor (ou apenas) com dados univariados, enquanto outras são mais adequadas a conjuntos de dados multivariados. De forma semelhante, os dados contínuos podem exigir uma abordagem diferente em comparação com os dados categóricos ou com proporções, pois podem ter restrições inerentes, e limitando as opções de transformação. Além disso, as transformações impõem frequentemente requisitos ou limitações específicas aos dados. Algumas transformações, por exemplo, são concebidas para aceitar todos os valores, enquanto outras podem apenas aceitar valores estritamente positivos, assim como outras são mais suscetíveis a dados anómalos. É fundamental compreender estas restrições, uma vez que a aplicação de uma transformação a dados que não satisfazem os seus critérios pode conduzir a resultados errados ou até mesmo impedir que a transformação seja realizada. Essencialmente, tanto os objetivos da transformação como a natureza dos dados são os principais fatores na seleção da transformação mais adequada. Estes fatores devem ser verificados para garantir que a transformação fica em linha com os objetivos da análise e as características dos dados.

No entanto, nem todos os dados devem ou podem ser transformados. Por vezes, encontramos distribuições com caudas muito longas em ambas as extremidades, tornando a maioria das transformações ineficazes. Em alternativa, podemos utilizar métodos não paramétricos que não envolvem pressupostos de distribuição, embora estes forneçam frequentemente um teste de significância válido sem um intervalo de confiança. É importante lembrar que, embora as transformações possam ser ferramentas poderosas,

nem sempre são necessárias ou apropriadas para todos os conjuntos de dados que estão a ser analisados (Bland).

4.1. Dados

Como já referido, o conjunto de dados a ser analisado contém várias características que podem ditar a escolha da transformação. Nas transformações estudadas, os valores dos dados estão, geralmente, nos seguintes intervalos: $(-\infty, \infty)$, $[0\ ,1]$, $(0\ ,1)$, $(0, \infty)$ e $[0, \infty)$. Por exemplo, transformações como a *Yeo-Johnson*, TP e a TIP são aplicáveis a todos os valores reais. Por outro lado, várias transformações, como as transformações Logarítmica, Inversa/Recíproca e de *Box-Cox*, são especificamente adequadas para valores estritamente positivos $(0, \infty)$. Outras transformações, como as transformações do Arco Seno da Raiz Quadrada e *Logit*, foram desenvolvidas para dados de probabilidade ou proporção estando limitadas a dados no intervalo [0, 1] (Tabela 4.1).

Tabela 4.1 - Aplicabilidade da transformação aos tipos de intervalo de dados

Transformações	(-∞, ∞)	[0,1]	(0,1)	(0, ∞)	[0, ∞)	(-1, ∞)
Box-Cox				✓		
Yeo-Johnson	✓					
Tukey Ladders of Power	✓					
Guerrero-Johnson		✓				
Inversa/Recíproca				✓		
Raiz Quadrada				✓		
Raiz Cúbica					✓	
Logarítmica				✓		
Log-Sinh				✓		
TLN						✓
Log-Log			✓			
Log-Log Complementar			✓			
Retorno Logarítmico				✓		
Normalização *Min-Max*	✓					
ECU	✓					
Normalização de *Z-score* (Estandardização)	✓					
Escala Decimal	✓					
Escala Robusta	✓					
Escala do Máximo Absoluto	✓					
Arco Seno		✓				
Arco Seno da Raiz Quadrada		✓				
Exponencial	✓					
Exponencial Dobrada		✓				
ILR				✓		
chiPower					✓	
TP	✓					
Pontuações Normais	✓					
Branqueamento	✓					
Cinética	✓					
Probit			✓			
Logit			✓			
TIP	✓					

Os valores anómalos são tratados de forma diferente em várias transformações de dados, algumas transformações são especificamente concebidas para ter em conta ou atenuar o impacto dos valores anómalos (*outliers*, por exemplo), enquanto outras podem ser sensíveis a valores anómalos. A TP, por exemplo, ao substituir os dados pelas posições torna-se robusta a valores anómalos, enquanto a Escala Robusta foi desenvolvida com os valores anómalos em consideração (Tabela 4.2).

Tabela 4.2 - Tratamento para dados anómalos

Dados Anómalos	**Transformações**
Sim	*Box-Cox, Yeo-Johnson*, Inversa/Recíproca, Logarítmica, TLN, Escala Robusta, TP, Pontuações Normais, *Probit, Logit*
Não	*Tukey Ladders of Power, Guerrero-Johnson,* Arco Seno, Arco Seno da Raiz Quadrada, *Log-Log, Log-Log* Complementar, *Log-Sinh,* Raiz Quadrada, Raiz Cúbica, Exponencial, Exponencial Dobrada, *ILR, chiPower,* Normalização *Min-Max,* ECU, Normalização de *Z-Score,* Escala Decimal, Escala do Máximo Absoluto, Branqueamento, Cinética e TIP

A grande maioria das transformações pode ser aplicada a dados contínuos (ou até mesmo exclusivamente a dados contínuos) (Tabela 4.3), no entanto, existem transformações talhadas apenas para tipos de dados específicos, como nos casos dos dados composicionais (*ILR* e *chiPower*) ou de proporções (Arco Seno, Arco Seno da Raiz Quadrada, *Log-Log*, *Log-Log* Complementar, Exponencial Dobrada, *Probit* e *Logit*).

Tabela 4.3 - Tipos de dados

Transformações	Contínuos	Discretos	Ordinais	Binário	Composicionais	Proporcionais
Box-Cox	✓					
Yeo-Johnson	✓					
Tukey Ladders of Power	✓					
Guerrero-Johnson	✓	✓		✓		
Inversa/Recíproca	✓					
Raiz Quadrada	✓	✓				
Raiz Cúbica	✓					
Logarítmica	✓					
Log-Sinh	✓					
TLN	✓	✓				
Log-Log						✓
Log-Log Complementar						✓
Retorno Logarítmico	✓					
Normalização *Min-Max*	✓					
ECU	✓					
Normalização de *Z-score* (Estandardização)	✓					
Escala Decimal	✓					
Escala Robusta	✓					
Escala do Máximo Absoluto	✓					
Arco Seno						✓
Arco Seno da Raiz Quadrada						✓
Exponencial	✓					
Exponencial Dobrada						✓
ILR					✓	
chiPower					✓	
TP	✓					
Pontuações Normais	✓		✓			
Branqueamento	✓					
Cinética	✓					
Probit						✓
Logit						✓
TIP	✓					

A grande maioria das transformações não têm quaisquer restrições sobre o formato de dados, sendo algumas especificas apenas para dados univariados (Tabela 4.4), como acontece nas transformações Cinética, *Log-Log*, Pontuações Normais (Método de *Blom*), Exponencial e Exponencial Dobrada, enquanto outras como *Guerrero-Johnson*, TP, *Log-Log Complementar* são específicas para dados bivariados (2 variáveis, incluindo variável dependente) ou multivariados (3 ou mais variáveis incluindo variável ou variáveis dependentes).

Tabela 4.4 - Formato dos dados

Transformações	Univariados	Bivariados	Multivariados
Box-Cox	✓	✓	✓
Yeo-Johnson	✓	✓	✓
Tukey Ladders of Power	✓	✓	✓
Guerrero-Johnson		✓	✓
Inversa/Recíproca	✓	✓	✓
Raiz Quadrada	✓	✓	✓
Raiz Cúbica	✓	✓	✓
Logarítmica	✓	✓	✓
Log-Sinh	✓		
TLN	✓	✓	✓
Log-Log	✓		
Log-Log Complementar		✓	✓
Retorno Logarítmico	✓		
Normalização *Min-Max*	✓	✓	✓
ECU	✓	✓	✓
Normalização de *Z-score* (Estandardização)	✓	✓	✓
Escala Decimal	✓	✓	✓
Escala Robusta	✓	✓	✓
Escala do Máximo Absoluto	✓	✓	✓
Arco Seno	✓	✓	✓
Arco Seno da Raiz Quadrada	✓	✓	✓
Exponencial	✓		
Exponencial Dobrada	✓		
ILR			✓
chiPower			✓
TP		✓	✓
Pontuações Normais	✓		
Branqueamento	✓	✓	✓
Cinética	✓		
Probit	✓	✓	✓
Logit	✓	✓	✓
TIP	✓		

A escolha da transformação pode depender da distribuição original dos dados e das propriedades desejadas para a análise subsequente. É importante notar que algumas transformações são versáteis e podem ser aplicadas a vários tipos de distribuições, enquanto outras são mais indicadas para distribuições específicas (Tabela 4.5). Considerando que a maioria dos métodos estatísticos visa reduzir ou corrigir a assimetria dos dados devido ao pressuposto de normalidade exigido em diversas análises estatísticas.

Tabela 4.5 - Transformações aplicáveis consoante o tipo de distribuição

Transformações	Assimetria Positiva	Simétrica (Normal/Gaussian)	Assimetria Negativa	Uniforme	Binomial	Poisson	Logística	Exponencial
Box-Cox	✓		✓					
Yeo-Johnson	✓		✓					
Tukey Ladders of Power	✓	✓	✓					
Guerrero-Johnson							✓	
Inversa/Recíproca	✓							
Raiz Quadrada	✓				✓	✓		
Raiz Cúbica	✓							
Logarítmica	✓							
Log-Sinh	✓							
TLN	✓		✓					
Log-Log							✓	✓
Log-Log Complementar	✓							
Retorno Logarítmico	✓		✓					
Normalização *Min-Max*	✓	✓	✓					
ECU	✓	✓	✓	✓				
Normalização de *Z-score* (Estandardização)		✓						
Escala Decimal		✓						
Escala Robusta	✓	✓	✓					
Escala do Máximo Absoluto	✓		✓					
Arco Seno					✓			
Arco Seno da Raiz Quadrada					✓			
Exponencial	✓		✓					
Exponencial Dobrada	✓		✓					
TP	✓	✓		✓				✓
Pontuações Normais	✓		✓					
Branqueamento		✓						
Cinética	✓		✓					
Probit		✓			✓			
Logit					✓			
TIP	✓	✓		✓				

4.2. Objetivos

Os objetivos abordados representam princípios e métodos fundamentais para preparar e validar análises (Tabela 4.6). Para auxiliar a tomada de decisão na escolha da transformação mais apropriada, foram construídos fluxogramas utilizando o *software Apple® Freeform*, abordando cenários com múltiplas transformações disponíveis. Os elementos são codificados por cores específicas: amarelo para análises, azul-marinho para intervalos de dados, azul para formatos de dados, verde para distribuições, vermelho para tipos de dados e laranja para as transformações. Na elaboração dos fluxogramas, priorizou-se a aplicação em análises estatísticas, sendo a isometria a única exceção. Devido à complexidade das análises e características estudadas, alguns fluxogramas tornaram-se extensos, dificultando sua interpretação. Consequentemente, para objetivos como normalidade, estabilização da variância e normalização, foram apresentados fluxogramas simplificados com apenas uma análise.

Tabela 4.6 - Transformações para os objetivos selecionados

Objetivos	**Transformações**
Normalidade	Arco Seno, *Box-Cox*, *Yeo-Johnson*, Inversa/Recíproca, Raiz Quadrada, Raiz Cúbica, Arco Seno, Logarítmica, *Log-Sinh*, TLN, *Log-Log*, *Log-Log* Complementar, Retorno Logarítmico, Exponencial, Exponencial Dobrada, Por Posições, Pontuações Normais, TIP, TLN e Cinética
Estabilização da Variância	Arco Seno, *Box-Cox*, *Yeo-Johnson*, *Tukey Ladder of Powers*, Inversa/Recíproca, Raiz Quadrada, Arco Seno, Arco Seno da Raiz Quadrada, Logarítmica, *Log-Sinh* e *Log-Log*
Isometria	*ILR* e *chiPower*
Colinearidade	TIP e Branqueamento
Linearização	*Box-Cox*, *Yeo-Johnson*, *Tukey Ladder of Powers*, Logarítmica, *Log-Log*, Cinética, *Probit* e *Logit*
Distribuição conjunta de Co-variáveis	TLN
Generalização o Modelo Logístico	*Guerrero-Johnson*
Normalização (Entre escalas)	Normalização *Min-Max*, ECU, Normalização de *Z-Score*, Escala Decimal, Escala Robusta e Escala de Máximo Absoluto

- **Normalidade** - O pressuposto de normalidade refere-se à suposição de que os dados seguem uma distribuição normal (ou *gaussiana*), com uma curva em forma de sino (Zedeck, 2014). A normalidade é um requisito básico para várias análises estatísticas, como a regressão linear, os testes *t* e a *ANOVA*. É o pré-requisito que mais frequentemente motiva a utilização de transformações de dados. Tomando como exemplo a escolha de uma transformação, num cenário onde se realiza uma regressão linear simples (ou múltipla) e em que os resíduos não cumprem o pressuposto da normalidade exigida pela regressão (Figura 4.1).

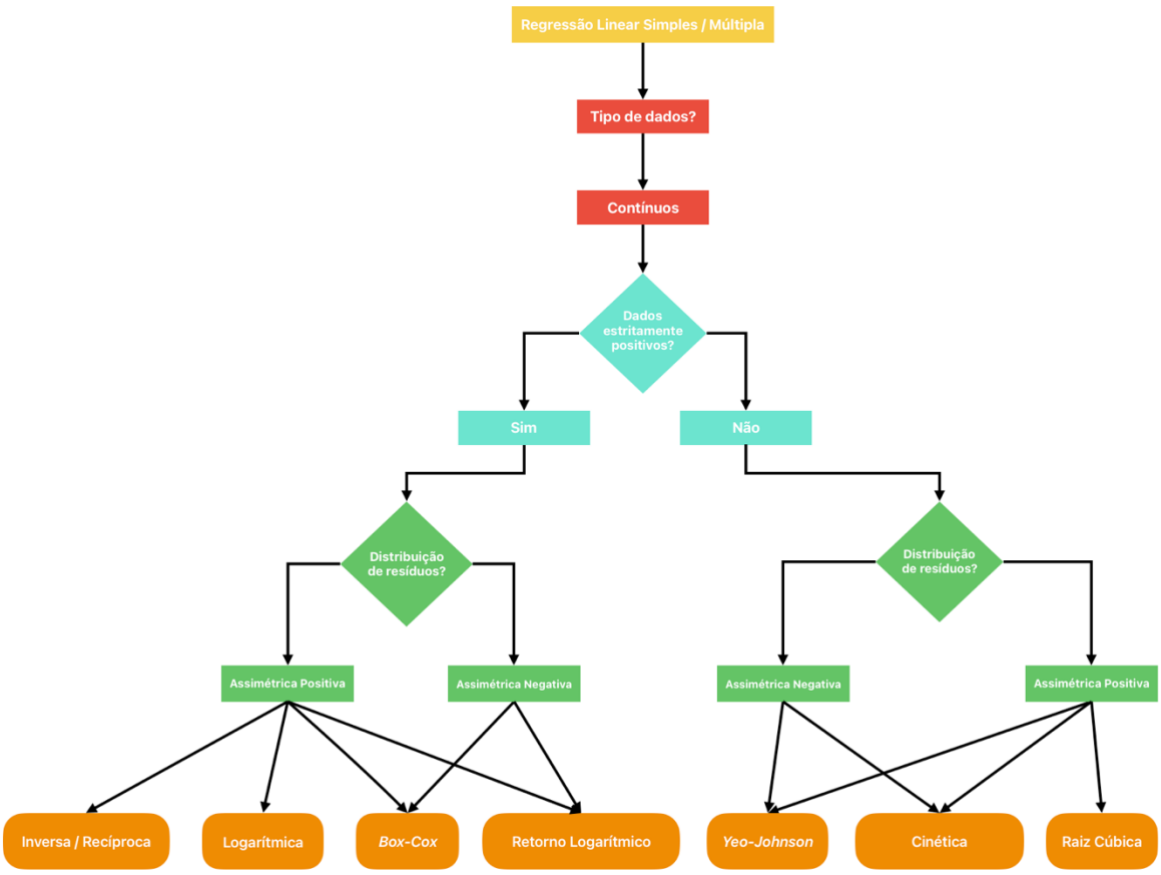

Figura 4.1 - Fluxograma de escolha de transformação para cumprir pressuposto de normalidade de resíduos na Regressão Linear

- **Estabilização da Variância** – Várias transformações, incluindo Inversa/Recíproca, Logarítmica e várias transformações de potência como, por exemplo, Raiz Quadrada e o Arco Seno da Raiz Quadrada, são utilizadas para estabilizar a variância. A estabilização da variância é um objetivo em algumas análises, como na *ANOVA*, constante em diferentes níveis dos dados (homocedasticidade). Se este pressuposto for violado - por exemplo, se a variância aumentar com a média, como acontece frequentemente com os dados biológicos ou financeiros - pode produzir resultados pouco fiáveis. Considerando a realização de uma *ANOVA*, a processo de escolha da transformação pode ser realizado através da Figura 4.2.

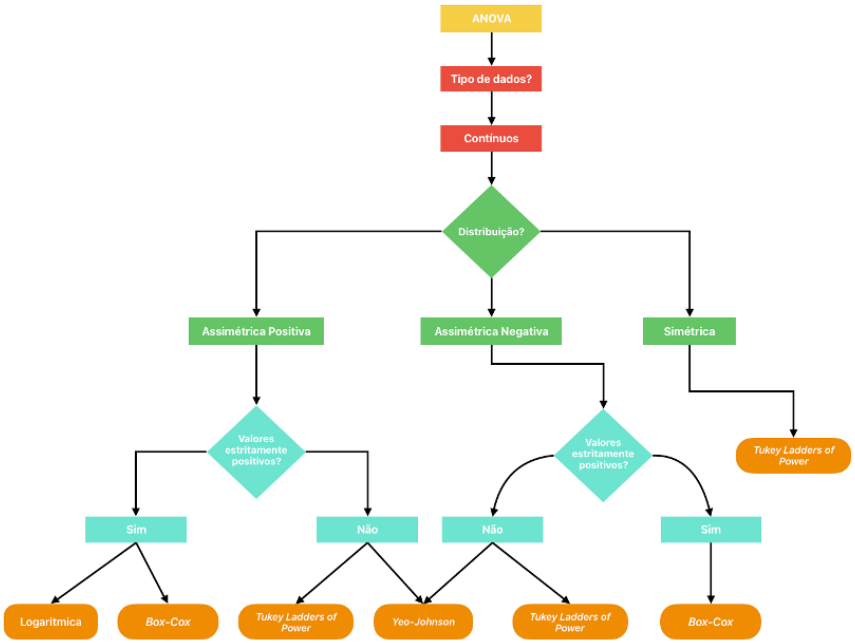

Figura 4.2 - Fluxograma de escolha de transformação para cumprir pressuposto da estabilização da variância na ANOVA

- **Isometria** - A preservação da propriedade geométrica que preserva distâncias é frequentemente aplicada em análises de dados para manter relações geométricas intactas (Egozcue, Pawlowsky-Glahn, Mateu-Figueras, & Barceló-Vidal, 2003) sendo o objetivo que as transformações da família de dados composicionais pretendem alcançar (Figura 4.3).

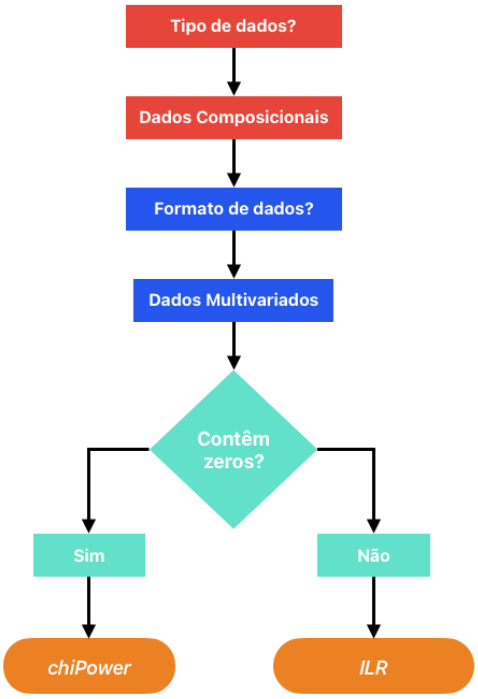

Figura 4.3 - Fluxograma de escolha da transformação para obter isometria

- **Colinearidade** - Quando duas ou mais variáveis estão altamente correlacionadas, o que pode prejudicar a interpretação de modelos de regressão (Zedeck, 2014) é um outro cenário onde se pode utilizar as transformações, para mitigar o problema. Utilizando as características dos dados e das respetivas análises foi construído um fluxograma de apoio á decisão na escolha da transformação (Figura 4.4).

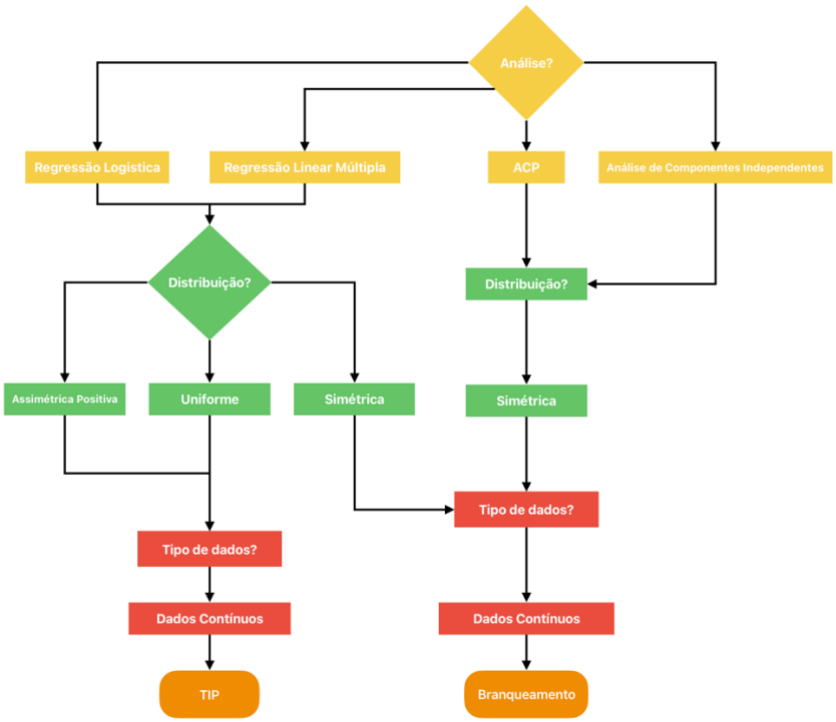

Figura 4.4 - Fluxograma de escolha de transformação para resolver colinearidade

- **Linearização** – O processo de transformar dados ou modelos para poderem ser representados ou ajustados linearmente. Permite-nos converter relações complexas e não lineares em relações lineares mais simples, que são mais fáceis de analisar. Considerando as transformações que têm como objetivo a linearização, assim como as análises que necessitam de linearização, foi possível construir o fluxograma da Figura 4.5.

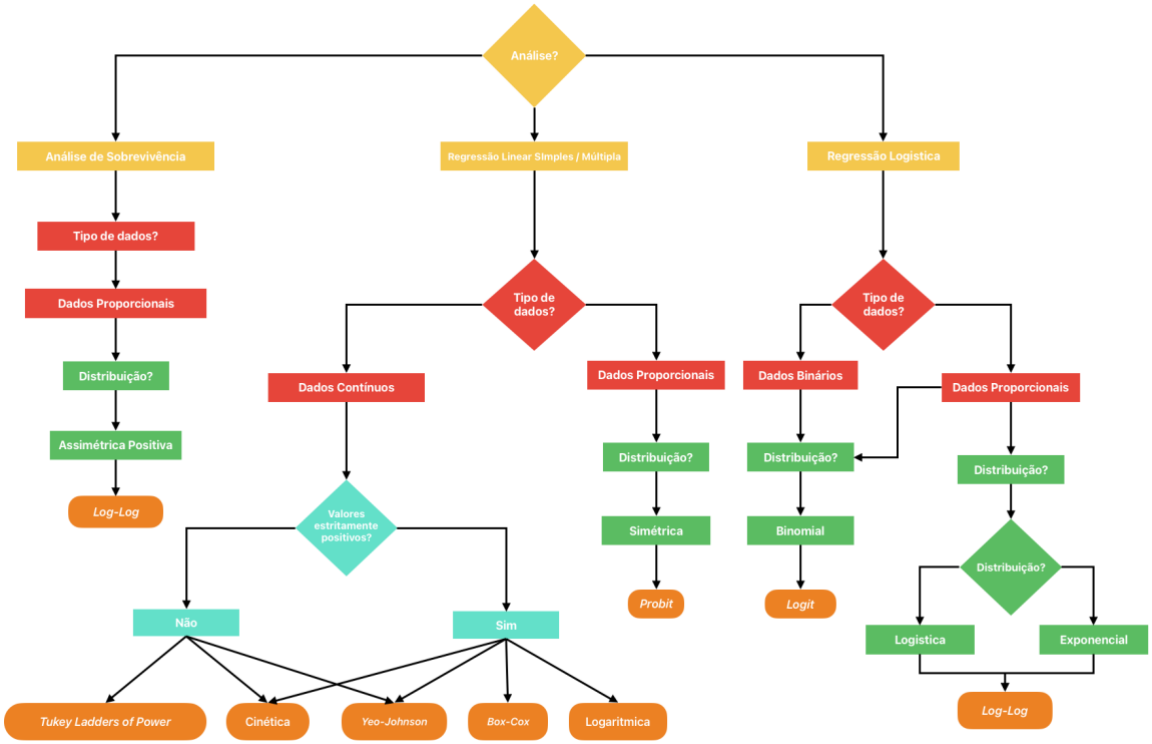

Figura 4.5 - Fluxograma de escolha de transformação para ajudar a linearização

- **Distribuição Conjunta de Co-variáveis** - A distribuição simultânea de duas ou mais variáveis explicativas (co-variáveis) num modelo, geralmente analisada para entender as suas interações.

- **Generalização do Modelo Logístico** - Aplicação de modelos logísticos a situações mais complexas, como a regressão logística multinomial ou outras extensões que envolvem múltiplos resultados.

- **Normalização (Entre Escalas)** - Ao ajustar os dados para estarem na mesma escala, faz com que os dados contribuem de forma igual no modelo, facilitando comparações entre variáveis de diferentes magnitudes. Por exemplo, as redes neuronais baseiam-se na otimização baseada em gradientes, em que caraterísticas com escalas muito diferentes podem fazer com que os gradientes se comportem de forma irregular, o que atrasa a convergência ou conduz a soluções não ótimas. Neste cenário da utilização de uma rede neuronal, a transformação a ser utilizada pode ser definida pelo fluxograma da Figura 4.6.

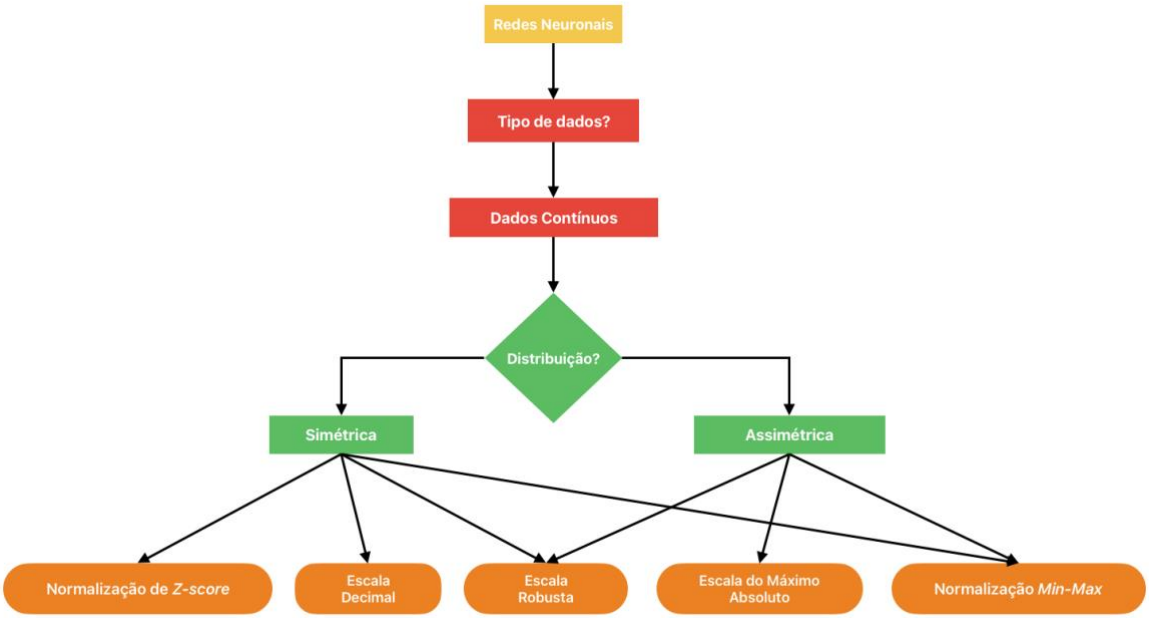

Figura 4.6 - Fluxograma de ajuda á escolha de transformação para normalização de dados (entre escalas)

4.3. Análises

O estudo das transformações mostrou que existe uma compatibilidade natural entre certas transformações e certos métodos de análise, baseada principalmente nos pressupostos específicos necessários para cada análise (Tabela 4.7).

Para a regressão linear, as transformações como *Box-Cox*, *Yeo-Johnson*, Logarítmica e Raiz Quadrada, que ajudam a obter o pressuposto da normalidade e a estabilizar a variância. A *ANOVA* utiliza transformações semelhantes para cumprir os pressupostos de homogeneidade das variâncias e normalidade dos resíduos.

Na regressão logística, técnicas como *Guerrero-Johnson*, *Log-Log* e *Logit* lidam com distribuições assimétricas e melhoram a interpretação dos coeficientes. Os testes *t* utilizam transformações como o Inversa/Recíproca, a Raiz Quadrada e a Logarítmica para transformar os dados.

Em *Machine Learning*, métodos como o KNN, as redes neuronais e o *k-means* utilizam a normalização *min-max*, a normalização *z-score*, MVS e a escala robusta para normalizar os dados e reduzir o impacto dos valores anómalos, enquanto técnicas como o *RBM* e o *DBN* utilizam o branqueamento para reduzir as correlações entre variáveis. Análises como a Análise de Sobrevivência utilizam transformações *Log-Log* para ajustar as curvas, enquanto o coeficiente de correlação de *Pearson* utiliza a raiz quadrada para lidar com a assimetria. Em modelos avançados, como o *ARIMA*, pode ser aplicada a transformação cinética para linearizar dados temporais.

Tabela 4.7 - Métodos e transformações mais utilizadas

Análises	Transformações
Regressão Linear	*Box-Cox, Yeo-Johnson, Tukey Ladder of Powers, Inversa/Recíproca, Raiz Quadrada, Raiz Cúbica, Logarítmica, Arco Seno, Arco Seno da Raiz Quadrada, Cinética, TIP*
ANOVA	*Tukey Ladder of Powers, Raiz Quadrada, Logarítmica, Arco Seno, Arco Seno da Raiz Quadrada, Exponencial, Exponencial, TP, Pontuações Normais*
Regressão Logistica	*Guerrero-Johnson, Log-Log, Logit*
Desenhos Factoriais	*Guerrero-Johnson*
Teste t	*Inversa/Recíproca, Raiz Quadrada, Logarítmica, Exponencial*
Alfa de Cronbach	*Raiz Quadrada*
Coeficiente de Correlação de Pearson	*Raiz Quadrada*
WAPABA	*Log-Sinh*
Regressão Quantilica	*TLN*
Análise de Sobrevivência	*Log-Log*
KNN	*Normalização Min-Max, ECU, Normalização Z-Score, Escala Robusta, Escala do Máximo Absoluto*
MVS	*Normalização Min-Max, ECU, Normalização Z-Score, Escala Robusta, Escala do Máximo Absoluto*
Redes Neuronais	*Normalização Min-Max, Normalização Z-Score, Escala Decimal, Escala Robusta, Escala do Máximo Absoluto*
ACP	*ECU, ILR, chiPower, Branqueamento, Normalização Z-Score*
Semelhança de Cosseno	*ECU*
k-means	*Normalização Min-Max, ECU, Normalização Z-Score, Escala Robusta, chiPower*
Análise de Componentes Principais de Kernel	*Normalização Z-Score*
Análise Discriminante	*Normalização Z-Score, Escala Robusta*
ANCOVA	*Arco Seno da Raiz Quadrada, TP*
Modelos Lineares Mistos	*Exponencial*
DVS	*ILR*
Função Discriminante Linear	*TP*
Função Discriminante Quadrática	*TP*
RBM	*Branqueamento*
DBN	*Branqueamento*
Análise de Componentes Independentes	*Branqueamento*
ARIMA	*Cinética*
Modelo Probit	*Probit*

5. Casos de Estudo

Foram cuidadosamente selecionados quatro estudos de caso que apresentam diferentes cenários de aplicação das transformações, desde situações em que são desnecessárias até outras que exigem uma normalização específica. A seleção e aplicação das transformações em cada caso de estudo foi orientada pelos fluxogramas de decisão previamente apresentados, que consideram aspectos como a preservação de isometria em análises espaciais, a normalidade em regressões lineares para modelação hidrogeológica e a normalização de dados para redes neuronais em estudos de supercondutores.

5.1. Classificação de caraterísticas biomecânicas em pacientes ortopédicos

A equipa médica de um centro ortopédico de reabilitação lida diariamente com pacientes que apresentam diversas patologias que afetam a coluna vertebral, como o alinhamento pélvico e o grau de espondilolistese. A equipa desenvolveu um modelo preditivo para a classificação de pacientes para identificar se estes apresentam valores normais ou anómalos, permitindo deste modo determinar a necessidade de exames complementares de diagnóstico.

O conjunto de dados (Lichman, 2013) utilizado para a construção do modelo contêm as variáveis independentes – o ângulo da incidência pélvica (em graus), o ângulo da inclinação pélvica (em graus), o ângulo de lordose lombar (em graus), o ângulo da inclinação sacral (em graus), o raio pélvico (em mm) e o grau de espondilolistese (em percentagens), onde todas as variáveis são contínuas. E a variável dependente que pode assumir os valores categóricos, *normal* ($n = 100$) ou *anómalo* ($n = 210$) (Tabela 5.1).

Estes parâmetros são calculados a partir de exames imagiológicos e/ou medições. Se o paciente apresentar medidas anómalas, isto significa que poderá padecer ou de uma hérnia discal, ou de espondilolistese.

Tabela 5.1 - AED do conjunto de dados das caraterísticas biomecânicas

Variável	Média	Desvio Padrão	Valor Mínimo	Valor Máximo
Incidência Pélvica	60,496	17,236	26,147	129,834
Inclinação Pélvica	17,542	10	-6,554	49,431
Ângulo de Lordose Lombar	51,930	18,554	14	125,742
Inclinação Sacral	42,953	13,423	13,366	121,429
Raio Pélvico	117,920	13,317	70,082	163,071
Grau de Espondilolistese	26,296	37,559	-11,058	418,543

A *Incidência Pélvica* e a *Inclinação Pélvica* apresentam distribuições ligeiramente enviesadas para a direita, com valores concentrados em torno de 50-70 e 10-30, respetivamente. O *Ângulo de Lordose Lombar* é mais fortemente enviesado para a direita, com um pico em 30-50 e uma longa cauda até 120, enquanto a *Inclinação Sacral* mais simétrica, com a maioria dos valores entre 30 e 60.

O *Raio Pélvico* apresenta uma distribuição em forma de sino, semelhante à normal, centrada em 120, enquanto o *Grau de Espondilolistese* é altamente enviesado para a direita, com a maioria dos valores perto de 0-100, mas com valores anómalos que excedem 400 (Figura 5.1).

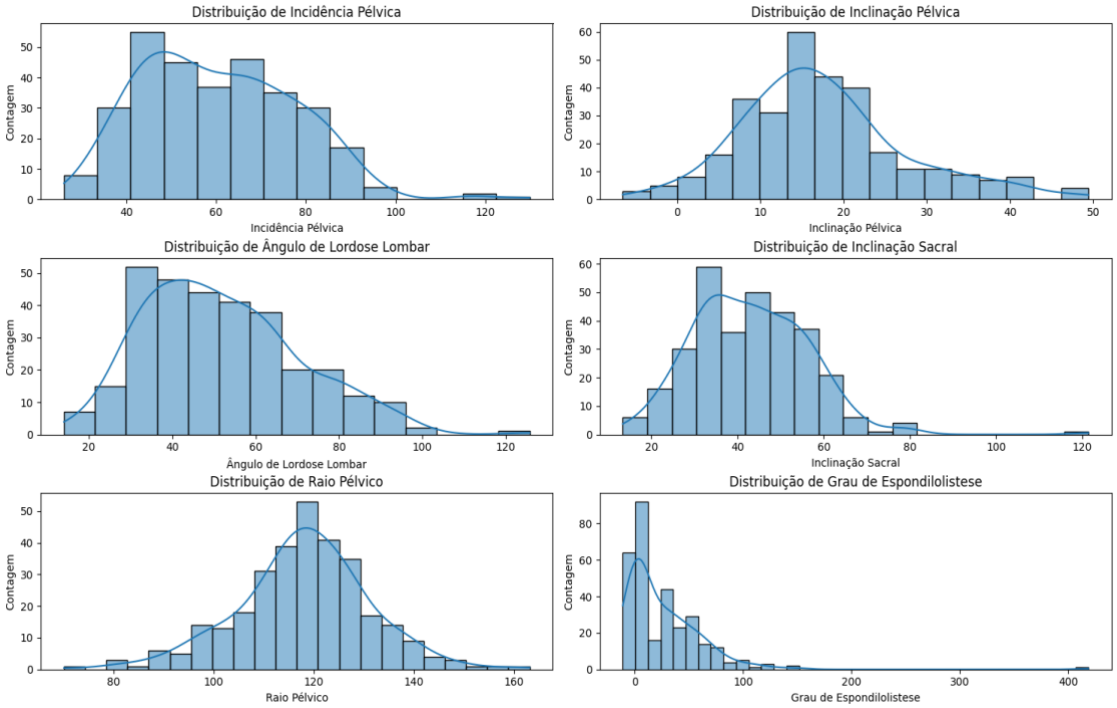

Figura 5.1 - Distribuições das variáveis biomecânicas

Pretende-se implementar um modelo de regressão logística para classificar os valores dos pacientes em normais ou anómalos, permitindo uma avaliação sistemática dos resultados clínicos. O primeiro passo implementado foi a verificação dos pressupostos necessários para a regressão logística, começando na verificação da existência da multicolinearidade através do teste Fator de Inflação da Variância (FIV), um FIV inferior a 5 indica uma correlação baixa, enquanto valores FIV superiores a 5 são um sinal de correlação (James, Witten, Hastie, & Tibshirani, 2013), neste cenário quase todas as variáveis apresentam colinearidade, com particular ênfase nas variáveis que apresentam uma FIV infinita (Tabela 5.2).

Tabela 5.2 - FIV

Variável	FIV
Incidência Pélvica	∞
Inclinação Pélvica	∞
Inclinação Sacral	∞
Ângulo de Lordose Lombar	18,94
Raio Pélvico	12,28
Grau de Espondilolistese	2,36

De forma a combater a colinearidade existente nas variáveis, principalmente nas que apresentam valores infinitos, foi usada a seleção *forward* baseada no teste de *Wald* (Marôco, 2021). O teste de *Wald* foi escolhido por ser computacionalmente eficiente, uma vez que requer apenas um ajuste de modelo por variável e testa diretamente a significância do coeficiente tornando-o ideal para grandes conjuntos de dados e seleção automática de variáveis. Após o teste de *Wald* apenas as variáveis *Raio Pélvico* e *Grau de Espondilolistese* foram escolhidas.

Tabela 5.3 - Variáveis após seleção forward baseada no teste de Wald

Variável	FIV
Raio Pélvico	1,476
Grau de Espondilolistese	1,476

Assim sendo, como ambas as variáveis apresentam um FIV baixo (<5), demonstrando que nem sempre é necessário realizar a transformação dos dados.

Para a realização da regressão logística os dados foram divididos, 80% dos dados para treino e os restantes 20% para teste. Após a realização da regressão, foi obtido um desempenho de aproximadamente 80% de precisão nos conjuntos de treino e teste. O desempenho semelhante entre as precisões de treino (79,4%) e de teste (80,6%) sugere que o modelo alcançou uma boa generalização dos dados.

A capacidade de discriminação do modelo é particularmente forte, como evidenciado pelas elevadas pontuações da área sob a curva (AUC) de 0,89 para o treino e 0,91 para o teste (Figura 5.2). Os resultados da validação cruzada, com uma média de 0,79 e um desvio padrão de 0,048, confirmam a estabilidade do modelo em diferentes subconjuntos de dados.

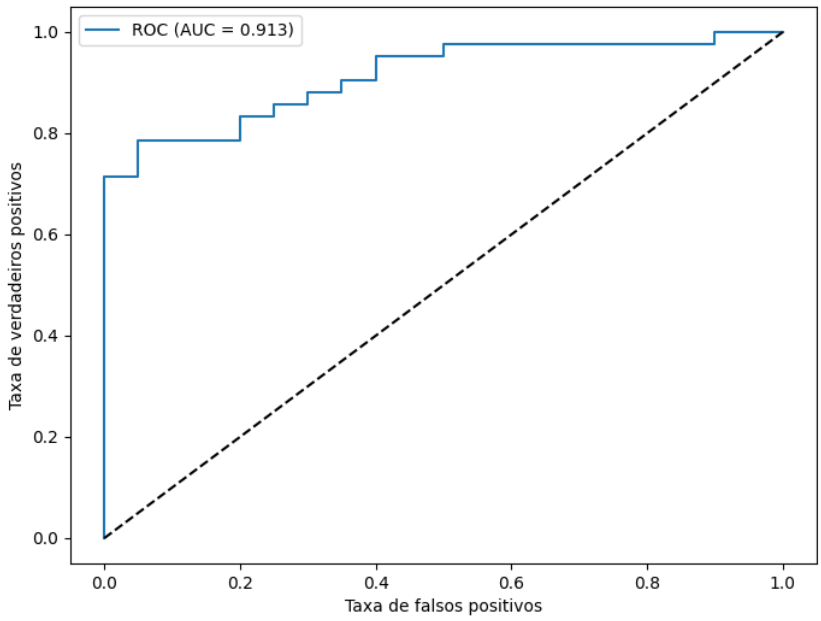

Figura 5.2 - Curva ROC dos dados de teste

Em termos de desempenho específico por classe, o modelo revela uma maior capacidade de previsão para casos anómalos (classe 1) com 86% de precisão e de recuperação, em comparação com 70% para casos normais (classe 0). Este desequilíbrio pode ser justificado pelo facto de haver mais casos anómalos que normais, no conjunto de dados.

A análise de regressão logística identificou o *Grau de Espondilolistese* e o *Raio Pélvico* como preditores significativos, com coeficientes de 0,107 e -0,072, respetivamente. Estes resultados indicam que um maior *Grau de Espondilolistese* ou um menor *Raio Pélvico* aumentam a probabilidade de casos anómalos.

5.2. Análise espacial de múltiplas secções de cultivo de milho

Numa exploração agrícola onde a tomada de decisão é baseada em dados, os investigadores aprofundam a dinâmica de crescimento de várias secções da exploração, utilizando coordenadas de longitude e latitude para mapear a sua distribuição geográfica.

O objetivo principal é visualizar e analisar a evolução do crescimento destas secções, com um foco específico no cultivo de milho durante os primeiros cinco meses. Para tal, a equipa recolhe os valores de longitude e latitude de cada exploração, bem como os níveis de macro e micronutrientes (em partes por milhão - ppm) presentes nas amostras de milho semeadas durante a fase inicial de cultivo.

Este conjunto de dados (Mineral Earth Sciences LLC), com $n = 724$ contém 14 variáveis contínuas, onde se inclui as variáveis sobre a localização das culturas de milho (*Latitude* e *Longitude*), as concentrações dos macronutrientes (nitrogénio, fósforo, potássio, sulfúrico, cálcio, magnésio) e as concentrações dos micronutrientes (zinco, ferro, manganésio, cobre, boro e molibdénio).

Tendo em conta que as variáveis que contêm os valores *ppm* são dados composicionais, é pretendido que haja isometria para assegurar que cada nutriente (tanto macro como micro) tenha uma representação equilibrada na análise, independentemente da sua magnitude original. Após os valores composicionais serem “fechados” (somados numa única constante), é necessário transformar os dados para garantir que existe isometria entre os mesmos.

Usando o método para a escolha da transformação que produza isometria (Figura 4.3), uma vez que não existem valores zero (e tendo em conta que são dados multivariados sem dados anómalos) é usada a transformação *ILR*. Após a transformação, é criada a visualização para a análise espacial (Figura 5.3) onde cada exploração agrícola é representada por um ponto, sendo o seu tamanho proporcional aos níveis médios de macro e micronutrientes nas amostras de milho.

Tabela 5.4 - AED do conjunto de dados das variáveis das amostras de milho

Variável	Média	Desvio Padrão	Valor Mínimo	Valor Máximo
Concentração Nitrogénio	262370,600	76688,459	119457,806	641848,523
Concentração Fósforo	31022,729	25739,881	1338,685	190369,94
Concentração Potássio	396983,816	113421,725	90993,387	688243,245
Concentração Sulfúrico	28305,987	7046,812	8873,992	59653,334
Concentração Cálcio	172170,048	50505,867	15579,176	310810,354
Concentração Magnésio	101330,488	37030,465	23860,632	247312,631
Concentração Zinco	22,078	16,569	1	192
Concentração Ferro	107,408	77,463	13	491
Concentração Manganésio	56,055	29,316	4	323
Concentração Cobre	4,885	2,659	0,9	24,7
Concentração Boro	6,901	2,827	1,3	23,8
Concentração Molibdénio	0,319	0,329	0,01	3,32

Perante esta análise podemos concluir que existem diferentes níveis de crescimento entre as secções da exploração agrícola e que, inclusive, em algumas esta possa não ser a melhor cultura a realizar. Em situação inversa nalgumas secções verifica-se um crescimento superior e o cultivo de milho deve ser prioritizado face a outras culturas para otimizar a rentabilidade da mesma.

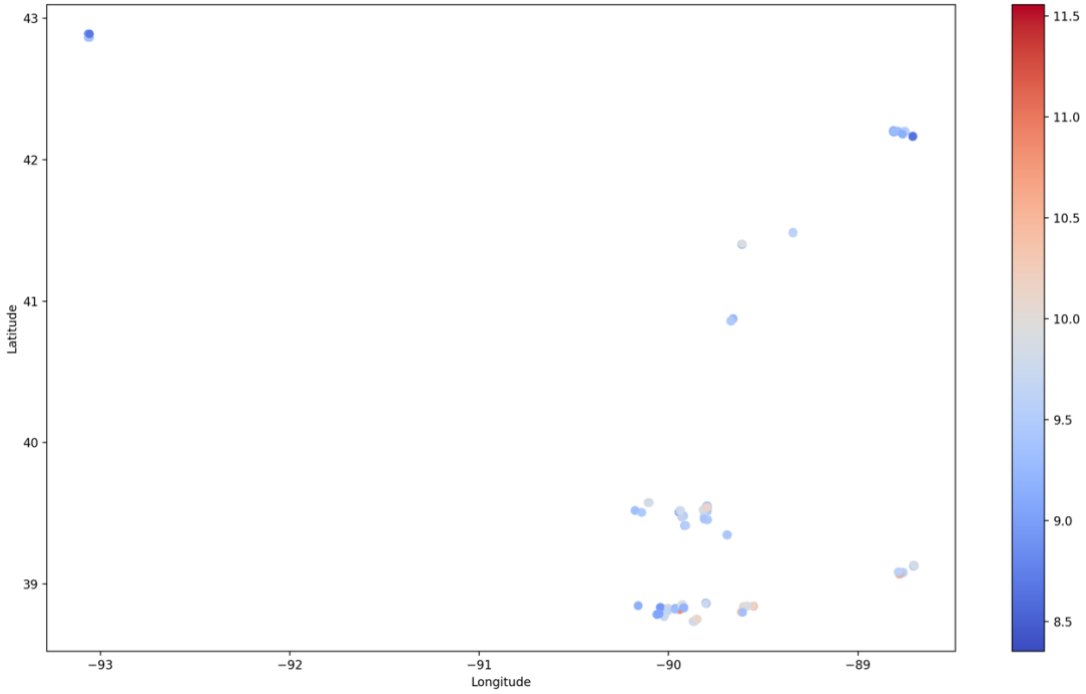

Figura 5.3 - Análise espacial das secções agrícolas

5.3. Modelação da Profundidade do Aquífero Luco

O Aquífero *Luco*, situado na bacia hidrogeológica do centro de Itália, representa um recurso hídrico subterrâneo para o abastecimento de água agrícola e municipal. Dentro deste sistema aquífero, o poço de monitorização *"Pozzo 1"* serve como ponto de observação para acompanhar as variações temporais dos níveis de água subterrânea. Este poço, estabelecido no final dos anos 80, fornece medições contínuas da profundidade das águas subterrâneas, utilizadas para compreender a resposta do aquífero às mudanças sazonais, padrões de precipitação e pressões de extração.

As medições da profundidade refletem a *"saúde"* geral e a dinâmica do sistema Aquífero *Luco*, tornando-o um ponto de referência para as decisões locais de gestão dos recursos hídricos e para os estudos hidrogeológicos a longo prazo. Ao ser modelada, a profundidade das águas subterrâneas, pode ajudar a tomar decisões sobre as taxas sustentáveis de extração de água, prever os potenciais impactos das alterações climáticas e proteger os aquíferos vulneráveis da sobre-exploração e/ou contaminação.

Para a realização da modelação através da regressão linear, é necessário garantir que o pressuposto de normalidade é cumprido. Para a construção do modelo foram obtidos dados de medições diárias desde 2017 até 2020 (Musone, et al., 2020), consistindo numa série temporal (Figura 5.4), em que as medições da profundidade do aquífero face ao nível de mar, são valores contínuos exclusivamente negativos. Na análise os valores foram agrupados por mês usando a média dos valores de cada mês, onde o valor mínimo foi -12,232 e o valor máximo -9,34, com uma média de -11,089 e um desvio padrão de 0,716.

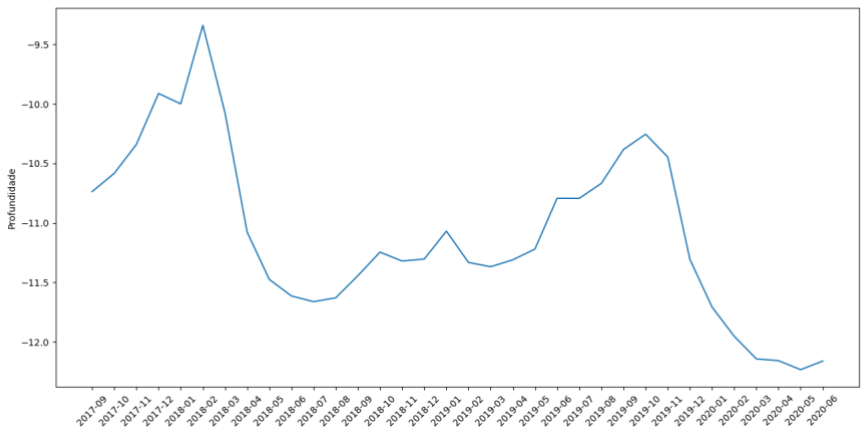

Figura 5.4 - Serie temporal da profundidade do aquífero Luco

Para a escolha da transformação dos dados foi usado o método da Figura 4.1, seguindo o caminho: Regressão Linear Simples / Múltipla -> Tipo de dados? [Contínuos] -> Dados estritamente positivos? [Não] -> Distribuição de resíduos? [Assimétrica Positiva] (Figura 5.5) sendo assim escolhida a transformação Cinética.

Para verificar a influência da transformação foi realizado o teste *Shapiro-Wilk* para testar a normalidade dos resíduos antes e após a transformação (Figura 5.5). Antes de aplicar a transformação, foi realizado o teste *Shapiro-Wilk* (considerando um nível de significância de 0,05) onde se obteve um valor-p $= 0{,}025$ com uma estatística de 0,926 onde se rejeita a H_0 violando pressuposto de normalidade exigida pela regressão linear. Após a transformação obteve-se o valor-p $= 0{,}768$ com uma estatística de 0,979, aceitando-se assim a H_0.

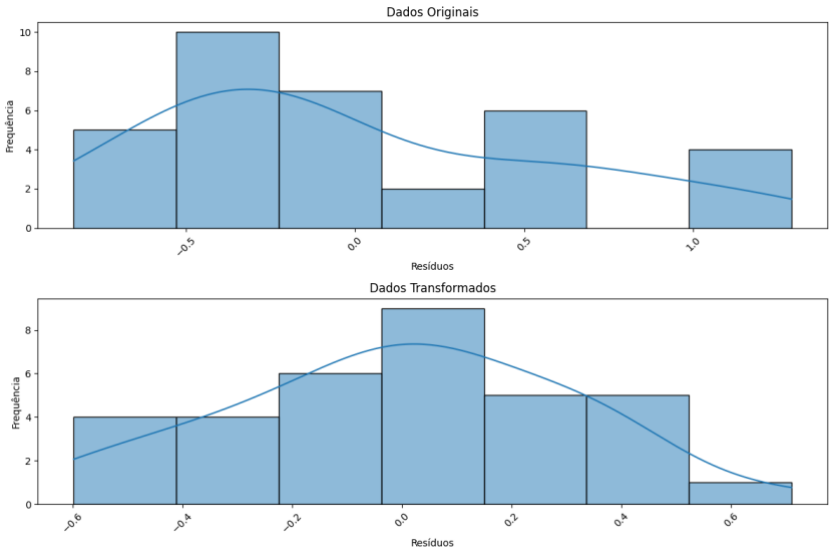

Figura 5.5 -Resíduos da Regressão antes e após a transformação Cinética

5.4. Rede neuronal para a previsão da temperatura crítica de supercondutores

Os supercondutores são materiais que apresentam uma resistência elétrica nula e a expulsão de campos magnéticos abaixo de uma temperatura crítica. A previsão da temperatura crítica a partir das propriedades químicas e físicas dos supercondutores é um passo vital para a descoberta de novos materiais supercondutores.

O conjunto de dados inclui 21263 materiais e 81 variáveis contínuas que incluem caraterísticas físicas e químicas como massa atómica, afinidade eletrónica, densidade, condutividade térmica e propriedades de valência e em que a variável dependente é a temperatura crítica (Hamidieh, 2018).

A distribuição da maioria das variáveis apresenta assimetria, com amplas gamas e elevados desvios-padrão, refletindo a natureza heterogénea dos materiais. Por exemplo, a condutividade térmica varia de 0,03 W/(m·K) a 406,96 W/(m·K) e a afinidade eletrónica varia de 1,5 kJ/mol a 326,1 kJ/mol. As medidas de entropia para várias propriedades, como a massa atómica e a condutividade térmica, agrupam-se geralmente abaixo de 2, sugerindo uma grande variabilidade na distribuição dos elementos.

Como existe uma grande variedade de escalas entre variáveis, é necessária uma normalização para garantir a convergência dos algoritmos de otimização durante o treino da rede neuronal. Utilizando a Figura 4.6 para decisão de que transformação utilizar, foi seguido o caminho: Análise? [Rede Neuronais] -> Tipo de dados? [Contínuos] -> Distribuição? [Assimétrica] -> Normalização *Min-Max*.

A rede neuronal foi estruturada com uma camada de entrada com 81 variáveis e com 3 camadas ocultas com funções de ativação *ReLU* (com 64, 32 e 16 neurónios respetivamente por cada camada) para captar as dependências não lineares e onde a camada de saída contém um único neurónio. Entre cada camada oculta foi incluída uma camada de abandono (*Dropout*) de formar a prevenir possíveis sobreajustes (*overfitting*).

Para o treino do modelo foi usado o optimizador *Adam* com uma taxa de aprendizagem (*learning rate)* inicial de 0,001. O erro médio quadrático (*EMQ*) foi usado como função de perda. Por fim, para avaliar o desempenho da rede neuronal, foi usada a raiz do erro quadrático médio (REQM) e o R-quadrado (Tabela 5.5).

Tabela 5.5 - Métricas da rede neuronal sem e com normalização de dados

Rede Neuronal	**Perda (*Loss*)**	**MSE**	**RMSE**	**R-quadrado**
Sem normalização	0,013	444,266	21,077	0,614
Com normalização	0,0065	223,768	14,958	0,805

Após o treino de ambos os cenários (com e sem normalização) com os mesmos dados de treino e teste, os resultados dos testes demonstram que a normalização melhora significativamente o desempenho do modelo. Com a normalização, a perda é reduzida para 0,0065, indicando um melhor ajuste. O EMQ e REQM diminuem, sugerindo menor magnitude de erro e melhores previsões. Além disso, o R-quadrado aumenta para 0,8056, o que significa que o modelo explica uma parte maior da variação.

Sem a normalização, a perda é maior, 0,0130, refletindo um ajuste mais fraco. O EMQ e REQM são substancialmente mais altos, indicando maiores erros de previsão, enquanto R-quadrado cai para 0,6140, mostrando uma capacidade reduzida de explicar a variação na variável dependente.

6. Discussão

A análise de conjuntos de dados que incorporam diferentes tipos de variáveis (nominais, ordinais e contínuas) apresenta desafios específicos quanto à adequação aos pressupostos paramétricos. Particularmente, nas variáveis categóricas que requerem técnicas especiais de codificação, como *one-hot encoding* ou *label encoding*, uma vez que não podem ser submetidas diretamente a transformações.

Quando os dados atendem aos pressupostos das análises, como normalidade e homogeneidade de variâncias, é possível utilizar os dados na sua forma original, sem necessidade de transformações. Outra exceção ficou evidente no Caso de Estudo 5.1, em que o processo de seleção de variáveis, retirou variáveis que introduziam colinearidade aos dados, não sendo necessária a utilização de uma transformação.

Contudo existem situações, em que mesmo após a aplicação de transformações, os dados podem não satisfazer adequadamente esses pressupostos, comprometendo assim a validade dos resultados ou tornando a sua interpretação significativamente mais complexa.

Nessas circunstâncias, os testes não paramétricos emergem como uma solução metodológica robusta e eficaz. Permitindo realizar análises estatísticas sem a necessidade de assumir distribuições específicas ou efetuar transformações nos dados, preservando tanto a integridade dos resultados como a facilidade de interpretação. Entre as opções disponíveis, destacam-se o teste de *Mann-Whitney* e *Kruskal-Wallis* para comparações entre grupos, e o teste de *Spearman* para análise de correlações em dados ordinais.

Todavia não devemos descurar que apesar de os mesmos fornecerem um teste de significância válido não apresentam um intervalo de confiança.

Nos objetivos onde se encontram a maioria das transformações, normalidade, estabilização da variância e normalização (entre escalas), houve uma maior dificuldade na construção dos fluxogramas. Estes objetivos ao serem representados deram origem a fluxogramas extensos e de difícil interpretação, o que dificultou o objetivo inicial de simplificação e de facilidade de compreensão.

Ainda com o objetivo de simplificar o processo de decisão, foi desenvolvida uma ferramenta *web* onde é possível especificar a análise e as características do conjunto de dados aa analisar.

7. Conclusão

Esta dissertação apresenta uma análise extensa de métodos de transformação de dados, que resultou no desenvolvimento de uma *framework* para a seleção de transformações.

Os resultados obtidos demonstram que a eficácia de uma transformação está intrinsecamente ligada tanto às caraterísticas inerentes dos dados como aos objetivos específicos da análise. A *framework* desenvolvida revelou-se uma ferramenta com potencial de orientar no processo de decisão de qual transformação utilizar, considerando múltiplos critérios e contextos.

As principais contribuições da pesquisa abrangem uma categorização das transformações de dados sob diferentes perspetivas, classificando-as conforme as suas propriedades matemáticas, pressupostos e impactos sobre os dados.

As limitações identificadas incluem a necessidade de considerar contextos específicos e a possível existência de casos especiais onde a *framework* pode exigir adaptações ou até mesmo os pressupostos não serem cumpridos após a transformação.

Os resultados obtidos sugerem que uma abordagem à seleção de transformações pode melhorar substancialmente a qualidade das análises estatísticas e, consequentemente, a fiabilidade das conclusões derivadas dos dados. A dificuldade encontrada na construção dos fluxogramas de apoio à escolha da transformação de dados mais adequada, permite que este trabalho possa ser mais aprofundado e desenvolvido assim como ajudar na escolha da transformação mais adequada, de futuro, de forma que o objetivo inicial seja cumprido.

Para aumentar a acessibilidade e a usabilidade dos métodos desenvolvidos, foi criada uma ferramenta *web* (https://whichtransformation.com). Nesta ferramenta, o utilizador poderá especificar os seus objetivos de análise e as caraterísticas do seu conjunto de dados, de forma a receber as recomendações sobre quais as transformações mais adequadas á sua análise.

8. Trabalho Futuro

Uma vez que ao longo desta dissertação foram apresentadas várias transformações, sendo estas acompanhadas de exemplos práticos implementados em *Python*, o desenvolvimento de uma biblioteca especializada torna-se uma evolução natural do trabalho. Tendo assim como objetivo facilitar a seleção da transformação mais adequada em função das caraterísticas específicas do conjuntos de dados analisado, e ainda permitir a aplicação individual de cada transformação.

Relativamente aos fluxogramas apresentados foram desenvolvidos com base rigorosa na literatura, cobrindo vários cenários e critérios de decisão. Como já referido anteriormente a dimensão de alguns dos fluxogramas tornou a sua consulta demasiado complexa, indo contra a simplificação que era um dos objetivos pretendidos. Permitindo assim que no futuro, estes sejam novamente analisados.

Esse passo poderá ser exequível, se forem efetuados estudos comparativos entre transformações, pois em algumas situações são sugeridas múltiplas transformações como alternativas viáveis. Uma forma otimizar a dimensão dos fluxogramas, de modo que estes sejam menores, poderá consistir em identificar eventuais vantagens em termos de eficácia e aplicabilidade nos diferentes contextos.

A exploração destas várias direções sugere que há ainda muito trabalho a desenvolver no sentido de conseguir indicar de forma mais simples e exata as transformações mais adequadas para cada caso de estudo.

9. Bibliografia

Ahrens, W. H., Cox, D. J., & Budhwar, G. (1990). Use of the Arcsine and Square Root Transformations for Subjectively Determined Percentage Data. *Weed Science, 38(4/5)*, 452-458. https://doi.org/10.1017/S0043174500056824.

Aitchison, J. (1986). *The Statistical Analysis of Compositional Data.* New York: Chapman and Hall.

Akritas, M. G. (1990). The Rank Transform Method in Some Two-Factor Designs. *Journal of the American Statistical Association, 85*, 73-78. https://doi.org/10.2307/2289527.

Armitage, P., Berry, G., & Matthews, J. N. (2002). *Statistical Methods in Medical Research.* Blackwell Science. https://doi.org/10.1002/9780470773666.

Bartlett, M. S. (1936). The Square Root Transformation in Analysis of Variance. *Supplement to the Journal of the Royal Statistical Society*, 68-78. https://doi.org/10.2307/2983678.

Bartlett, M. S. (1947). The Use of Transformations. *Biometrics*, 39-52. https://doi.org/10.2307/3001536.

Bickel, P. J., & Doksum, K. A. (1981). An Analysis of Transformations Revisited. *Journal of the American Statistical Association*, 296-311: https://doi.org/10.2307/2287831.

Bland, M. (s.d.). *Transformations.* Obtido em Agosto de 2024, de University of York: https://www-users.york.ac.uk/~mb55/msc/clinbio/week5/transfm_gif.pdf

Bliss, C. I. (1935). The Calculation of the Dosage-Mortality Curve. *Annals of Applied Biology, 22 (1)*, 134-167. https://doi.org/10.1111/j.1744-7348.1935.tb07713.x.

Box, G. E., & Cox, D. R. (1964). An Analysis of Transformations. *Journal of the Royal Statistical Society. Series B (Methodological), 26(2)*, 211-243. https://doi.org/10.1111/j.2517-6161.1964.tb00553.x.

Conover, W. J. (1999). *Practical Nonparametric Statistics, 3rd Edition.* John Wiley & Sons, Inc.

Conover, W. J. (2012). The rank transformation—an easy and intuitive way to connect many nonparametric methods to their parametric counterparts for seamless teaching

introductory statistics courses. *WIREs Comput Stat, 4(5)*, 432–438. https://doi.org/10.1002/wics.1216.

Conover, W. J., & Iman, R. L. (1980). The rank transformation as a method of discrimination with some examples. *Communications in Statistics - Theory and Methods, 9(5)*, 465-487. https://doi.org/10.1080/03610928008827895.

Conover, W. J., & Iman, R. L. (1981). Rank Transformations as a Bridge Between Parametric and Nonparametric Statistics. *The American Statistician, 35(3)*, 124-129. https://doi.org/10.1080/00031305.1981.10479327.

Conover, W. J., & Iman, R. L. (1982). Analysis of Covariance Using the Rank Transformation. *Biometrics, 38(3)*, 715-724. https://doi.org/10.2307/2530051.

Cox, N. J. (2011). Stata tip 96: Cube roots. *The Stata Journal, 11(1)*, 149–154. https://doi.org/10.1177/1536867x1101100112.

David, F. N., & Johnson, N. L. (1948). The Probability Integral Transformation When Parameters are Estimated from the Sample. *Biometrika, 35(1-2)*, 182-190. https://doi.org/10.1093/biomet/35.1-2.182.

Davison, A. C. (2014). Normal Scores. *Wiley StatsRef: Statistics Reference Online*, https://doi.org/10.1002/9781118445112.

de Amorim, L. B., Cavalcanti, G. D., & Cruz, R. M. (2022). The choice of scaling technique matters for classification performance. *133*, https://doi.org/10.1016/j.asoc.2022.109924.

Egozcue, J. J., Pawlowsky-Glahn, V., Mateu-Figueras, G., & Barceló-Vidal, C. (2003). Isometric Logratio Transformations for Compositional Data Analysis. *Mathematical Geology, 35*, 279-300. https://doi.org/10.1023/A:1023818214614.

Eisenhart, C. (1947). Inverse Sine Transformation of Proportions. Em C. Eisenhart, M. W. Hastay, & W. A. Wallis, *Techniques of Statistical Analysis* (pp. 397-416). New York: McGraw-Hill.

Emerson, J. D., & Stoto, M. A. (1982). Exploratory Methods for Choosing Power Transformations. *Journal of the American Statistical Association, 77(377)*, 103- 108. https://doi.org/10.1080/01621459.1982.10477772.

Erdin, R., Frei , C., & Künsch, H. R. (2012). Data Transformation and Uncertainty in Geostatistical Combination of Radar and Rain Gauges. *Journal of Hydrometeorology, 13(4)*, 1332-1346. https://doi.org/10.1175/jhm-d-11-096.1.

Everitt, B. S., & Skrondal, A. (2010). *The Cambridge Dictionary of Statistics.* Cambridge University Press.

Feng, C., Hongyue, W., Lu, N., Chen, T., He, H., Ying, L., & Tu, X. M. (2014). Log-transformation and its implications for data analysis. *Biostatistics in psychiatry*, 105-109. https://doi.org/10.3969/j.issn.1002-0829.2014.02.009.

Field, A., Miles, J., & Field, Z. (2012). *Discovering Statistics Using R.* SAGE Publications.

Freeman, M. F., & Tukey, J. W. (1950). Transformations Related to the Angular and the Square Root. *The Annals of Mathematical Statistics, 21(4)*, 607-611. https://doi.org/10.1214/aoms/1177729756.

Galli, S. (2020). *Python Feature Engineering Cookbook: Over 70 recipes for creating, engineering, and transforming features to build machine learning models.* Birmingham: Packt Publishing Ltd.

Godahewa, R., Bergmeir, C., Webb, G. I., & Montero-Manso, P. (2020). A Strong Baseline for Weekly Time Series Forecasting. https://dx.doi.org/10.48550/arXiv.2010.08158.

Greenacre, M. (2024). The chiPower transformation: a valid alternative to logratio transformations in compositional data analysis. *Advances in Data Analysis and Classification, 18(3)*, 769–796. https://doi.org/10.1007/s11634-024-00600-x.

Guerrero, V. M. (1993). Time-series analysis supported by Power Transformations. *Journal of Forecasting, 12(1)*, 37-48. https://doi.org/10.1002/for.3980120104.

Guerrero, V. M., & Johnson, R. A. (1982). Use of the Box-Cox transformation with binary response models. *Biometrika, 69(2)*, 309-314. https://doi.org/10.2307/2335404.

Gundersen, G. (6 de Fev de 2022). *Returns and Log Returns*. Obtido em Setembro de 2024, de Gregory Gundersen: https://gregorygundersen.com/blog/2022/02/06/log-returns/

Hamidieh, K. (2018). *Superconductivty Data - UCI Machine Learning Repository.* Obtido em Novembro de 2024, de UCI Machine Learning Repository: https://archive.ics.uci.edu/dataset/464/superconductivty+data

Han, J., Kamber, M., & Pei, J. (2011). *Data Mining: Concepts and Techniques.* Morgan Kaufmann.

Harter, H. L. (1961). Expected Values of Normal Order Statistics. *Biometrika, 48(1-2)*, 151-165. https://doi.org/10.1093/biomet/48.1-2.151.

Hossain, M. (2014). Whitening and coloring transformations for Multivariate gaussian data.

Hoyle, M. H. (1973). Transformations: An Introduction and a Bibliography. *International Statistical Review / Revue Internationale de Statistique, 41(2)*, 203-223. https://doi.org/10.2307/1402836.

Hvitfeldt, E. (28 de Janeiro de 2024). *Feature Engineering A-Z | Robust Scaling*. Obtido em Abril de 2024, de Feature Engineering A-Z: https://feaz-book.com/numeric-robust

Iman, R. L., & Conover, W. J. (1979). The Use of the Rank Transform in Regression. *Technometrics, 21(4)*, 499-509. https://doi.org/10.1080/00401706.1979.10489820.

Jaiswal, S. (Janeiro de 2024). *What is Normalization in Machine Learning? A Comprehensive Guide to Data Rescaling*. Obtido em Junho de 2024, de DataCamp: https://www.datacamp.com/tutorial/normalization-in-machine-learning

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning with Applications in R.* New York: Springer Science & Business Media.

Koivunen, A. C., & Kostinski, A. (1999). The feasibility of data whitening to improve performance of weather radar weather radar. *Journal of Applied Meteorology and Climatology*, 741-749. https://doi.org/10.1175/1520-0450(1999)038%3C0741:TFODWT%3E2.0.CO;2.

Krizhevsky, A. (2009). *Learning Multiple Layers of Features from Tiny Images.* Obtido em Maio de 2024, de https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

Lee, D. K. (2020). Data transformation: a focus on the interpretation. *Korean Journal of Anesthesiology, 73(6)*, 503-508. https://doi.org/10.4097/kja.20137.

Lichman, M. (2013). Biomechanical features of orthopedic patients. Irvine, CA: University of California, School of Information and Computer Science: UCI Machine Learning Repository [http://archive.ics.uci.edu/ml].

Lima, F. T., & Souza, V. M. (2023). A Large Comparison of Normalization Methods on Time Series. *Big Data Research, 34*, https://doi.org/10.1016/j.bdr.2023.100407.

Lin, L., & Xu, C. (2020). Perspective Arcsine-based transformations for meta-analysis of proportions: Pros, cons, and alternatives. *Health Science Reports, 3(3)*, https://doi.org/10.1002/hsr2.178.

Lu, H. T., & Smith, P. J. (1979). Distribution of the Normal Scores Statistic for Nonparametric One-Way Analysis of Variance. *74(367)*, pp. 715- 722. https://doi.org/10.1080/01621459.1979.10481676.

Manly, B. F. (1976). Exponential Data Transformations. *Journal of the Royal Statistical Society. Series D (The Statistician), 25(1)*, 37-42. https://doi.org/10.2307/2988129.

Marôco, J. (2021). *Análise Estatistica com SPSS Statistics.* Pêro Pinheiro: Report Number.

Mather, K. (1949). The Analysis of Extinction Time Data in Bioassay. *Biometrics, 5(2)*, 127-143. https://doi.org/10.2307/3001915.

Mineral Earth Sciences LLC. (s.d.). *Corn Tissue Samples.* Obtido em Outubro de 2024, de Kaggle: https://www.kaggle.com/datasets/mineral-earth-sciences-llc/corn-tissue-samples

Musone, A., Bergström, A., Federico, Marotta, L., Maggie, & Lucchesi, M. (2020). *Acea Smart Water Analytics.* Obtido em Novembro de 2024, de Kaggle: https://kaggle.com/competitions/acea-water-prediction

Norris, A. E., & Aroian, K. J. (2004). To Transform or Not Transform Skewed Data for Psychometric Analysis. *Nursing Research, 53(1)*, 67-71. https://doi.org/10.1097/00006199-200401000-00011.

O'Hara, R. B., & Kotze, D. J. (2010). Do not log-transform count data. *Methods in Ecology and Evolution, 1(2)*, 118-122. https://doi.org/10.1111/j.2041-210x.2010.00021.x.

Osborne, J. W. (2010). Improving your data transformations: Applying the Box-Cox transformation. *Practical Assessment, Research & Evaluation, 15(12)*, https://doi.org/10.7275/qbpc-gk17.

Othman, S. A., & Ali, H. T. (2021). Improvement of the Nonparametric Estimation of Functional Stationary Time Series Using Yeo-Johnson Transformation with Application to Temperature Curves. *Advances in Mathematical Physics*, https://doi.org/10.1155/2021/6676400.

Pek, J., Wong, O., & Wong, C. M. (2020). Data Transformations for Inference with Linear Regression: Data Transformations for Inference with Linear Regression: Clarifications and Recommendations. Em *Practical Assessment, Research, and Evaluation Practical Assessment, Research, and Evaluation* (Vol. 22).

Piepho, H.-P. (2003). The Folded Exponential Transformation for Proportions. *Journal of the Royal Statistical Society. Series D (The Statistician), 52(4)*, 575–589.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (1992). *Numerical Recipes in C.* Cambridge: Cambridge University Press.

Raschka, S. (2014). *About Feature Scaling and Normalization*. Obtido em Junho de 2024, de Sebastian Raschka: https://sebastianraschka.com/Articles/2014_about_feature_scaling.html#about-min-max-scaling

Raymaekers, J., & Rousseeuw, P. J. (2021). Transforming variables to central normality. *Machine Learning, 113(8)*, 4953–4975. https://doi.org/10.1007/s10994-021-05960-5.

Sahai, H., & Khurshid, A. (2002). *Pocket Dictionary of Statistics.* McGraw-Hill/Irwin.

Schendzielorz, T. M. (15 de Janeiro de 2020). *A guide to Data Transformation*. Obtido em Abril de 2024, de Medium: https://medium.com/analytics-vidhya/a-guide-to-data-transformation-9e5fa9ae1ca3

Scikit-Learn. (s.d.). *sklearn.preprocessing.MinMaxScaler — scikit-learn 1.4.2 documentation*. Obtido em Abril de 2024, de scikit-learn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html#sklearn.preprocessing.MinMaxScaler

Sinsomboonthong , S. (2022). Performance Comparison of New Adjusted Min-Max with Decimal Scaling and Statistical Column Normalization Methods for Artificial Neural Network Classification. *International Journal of Mathematics and Mathematical Sciences*, 1-9. https://doi.org/10.1155/2022/3584406.

Smith, M. J. (2018). *Statistical Analysis Handbook.* The Winchelsea Press.

Solomon, S. R., & Sawilowsky, S. S. (2009). Impact of Rank-Based Normalizing Transformations on the Accuracy of Test Score. *Journal of Modern Applied Statistical Methods, 8(2)*, 448-462. https://doi.org/10.22237/jmasm/1257034080.

StatsDirect Limited. (2024). *Normal Scores Transform - StatsDirect*. Obtido em Abril de 2024, de StatsDirect: https://www.statsdirect.com/help/data_preparation/transform_normal_scores.htm

Stoto, M. A., & Emerson, J. D. (1983). Power Transformations for Data Analysis. *Sociological Methodology, 14*, 126-168. https://doi.org/10.2307/270905.

Tukey, J. W. (1957). On the Comparative Anatomy of Transformations. *The Annals of Mathematical Statistics, 28(3)*, 602-632. https://doi.org/10.1214/aoms/1177706875.

Wang, Q. J., Shrestha, D. L., Robertson, D. E., & Pokhrel, P. (2012). A log-sinh transformation for data normalization and variance stabilization. *Water Resources Research, 48(5)*, https://doi.org/10.1029/2011wr010973.

Whittaker, J., Whitehead, C., & Somers, M. (2005). The Neglog Transformation and Quantile Regression for the Analysis of a Large Credit Scoring Database. *Journal of the Royal Statistical Society Series C (Applied Statistics), 54(5)*, 863-878. https://doi.org/10.1111/j.1467-9876.2005.00520.x.

Wicklin, R. (10 de Março de 2024). *Tukey's ladder of variable transformations - The DO Loop*. Obtido em Abril de 2024, de SAS: https://blogs.sas.com/content/iml/2022/08/15/tukeys-ladder-transformations.html

Wilson, E. B., & Hilferty, M. M. (1931). The Distribution of Chi-Square. *Proceedings of the National Academy of Sciences, 17(12)*, 684-688. https://doi.org/10.1073/pnas.17.12.684.

Wilson, E., Underwood, M., Puckrin, O., Letto, K., Doyle, R., Caravan, H., . . . Bassett, K. (2010). The Arcsine Transformation: Has the time come for retirement?

Wobbrock, J. O., Findlater, L., Gergle, D., & Higgins, J. J. (2011). The Aligned Rank Transform for Nonparametric Factorial Analyses Using Only ANOVA Procedures. *CHI 2011 -*

Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 143-146. https://doi.org/10.1145/1978942.1978963.

Wu, J. (2017). Distance Metrics and Data Transformation.

Yeo, I.-K., & Johnson, R. A. (2000). A New Family of Power Transformations to Improve Normality or Symmetry. *Biometrika, 87(4)*, 954-959. https://doi.org/10.1093/biomet/87.4.954.

Zedeck, S. (2014). *APA Dictionary of Statistics and Research Methods.* American Psychological Association.

Zhang, M. (2015). A New Data Transformation Method and Its Empirical Research Based on Inverted Cycloidal Kinetic Model. *Procedia Computer Science, 55*, 485-492. https://doi.org/10.1016/j.procs.2015.07.020.

Zimmerman, D. W. (2011). A note on consistency of non-parametric rank tests and related rank transformations. *British Journal of Mathematical and Statistical Psychology, 65(1)*, 122-144. https://doi.org/10.1111/j.2044-8317.2011.02017.x.