Controle Estatístico de Qualidade

Breno Cauã Rodrigues da Silva

2025-05-06

Índice

# 1. Introdução

Material de apoio para a disciplina de **Controle Estatístico de Qualidade** da *Falculdade de Estatística* (FAEST) da *Universidade Federal do Pará* (UFPA).

# Pacotes necessários  
library(ggplot2)  
library(dplyr)  
  
library(reticulate)  
use\_python("C:/Users/user/anaconda3/python.exe", required = TRUE)

# Tratamentos de Dados  
import numpy as np  
import pandas as pd  
  
# Funções Estatísticas  
from scipy import stats  
  
# Visualizações Gráficas  
import matplotlib.pyplot as plt  
import seaborn as sns

## 1.1 Ferramentas Básicas do Controle da Qualidade

As sete ferramentas da qualidae são técnicas estatísticas simples para resolver problemas na indústria.✅

* Estratificação
* Folhas de Verificação
* Diagrama de Ishikawa
* Histograma
* Diagrama de Pareto
* Gráfico de Dispersão
* Gráfico de Controle

### 1.1.1 Estratificação

É uma técnica usada para **separar dados em grupos significativos** para facilitar a análise.

* Permite observar padrões escondidos em dados mistos.
* Ajuda identificar **fontes de variação**.

# Geração de Dados: Exemplo de Dados Estratificados  
np.random.seed(11111)  
group1\_size = 30  
group1\_x = np.random.normal(loc=10, scale=2.5, size=group1\_size)  
group1\_y = np.random.normal(loc=20, scale=3.5, size=group1\_size)  
  
group2\_size = 40  
group2\_x = np.random.normal(loc=20, scale=3.5, size=group2\_size)  
group2\_y = np.random.normal(loc=30, scale=4.5, size=group2\_size)  
  
group3\_size = 30  
group3\_x = np.random.normal(loc=30, scale=4.5, size=group3\_size)  
group3\_y = np.random.normal(loc=15, scale=2.5, size=group3\_size)  
  
df = pd.DataFrame({  
 "x": np.concatenate([group1\_x, group2\_x, group3\_x]),  
 "y": np.concatenate([group1\_y, group2\_y, group3\_y]),  
 "Grupo": ["Grupo A"] \* group1\_size + ["Grupo B"] \* group2\_size + ["Grupo C"] \* group3\_size  
})  
  
# Configurações de Figura  
fig, ax = plt.subplots(figsize=(6, 4), dpi=600)  
  
# Scatterplot  
sns.scatterplot(x="x", y="y", data=df, hue="Grupo", palette="viridis", s=100, ax=ax)  
  
# Configurações de eixos e títulos  
ax.set\_xlabel("X", fontsize=12)  
ax.set\_ylabel("y", fontsize=12)  
  
# Configurações de legenda  
ax.legend(loc="upper right", frameon=False)  
  
# Outras configurações  
ax.spines["top"].set\_visible(False)  
ax.spines["right"].set\_visible(False)  
  
# Exibição do gráfico  
plt.show()

|  |
| --- |
| Figura 1.1: Exemplo de Simulação de Dados Estratificados em Python. |

# Definir semente  
set.seed(11111)  
  
# Grupo A  
group1\_size <- 30  
group1\_x <- rnorm(group1\_size, mean = 10, sd = 2.5)  
group1\_y <- rnorm(group1\_size, mean = 20, sd = 3.5)  
  
# Grupo B  
group2\_size <- 40  
group2\_x <- rnorm(group2\_size, mean = 20, sd = 3.5)  
group2\_y <- rnorm(group2\_size, mean = 30, sd = 4.5)  
  
# Grupo C  
group3\_size <- 30  
group3\_x <- rnorm(group3\_size, mean = 30, sd = 4.5)  
group3\_y <- rnorm(group3\_size, mean = 15, sd = 2.5)  
  
# DataFrame unificado  
df <- data.frame(  
 x = c(group1\_x, group2\_x, group3\_x),  
 y = c(group1\_y, group2\_y, group3\_y),  
 Grupo = factor(c(  
 rep("Grupo A", group1\_size),  
 rep("Grupo B", group2\_size),  
 rep("Grupo C", group3\_size)  
 ))  
)  
  
# Gráfico com ggplot2  
ggplot(df, aes(x = x, y = y, color = Grupo)) +  
 geom\_point(size = 3) +  
 scale\_color\_viridis\_d() +  
 labs(x = "X", y = "y", color = "Grupo") +  
 theme\_minimal(base\_size = 12) +  
 theme(  
 legend.position = "top",  
 panel.grid.minor = element\_blank(),  
 panel.grid.major = element\_line(color = "gray90"),  
 axis.line = element\_line(color = "black"),  
 axis.ticks = element\_line(color = "black"),  
 panel.border = element\_blank()  
 )

|  |
| --- |
| Figura 1.2: Exemplo de Simulação de Dados Estratificados em R. |

#### 1.1.1.1 Definição de Estratificação

“*Processo de* ***dividir dados em subgrupos (estratos)*** *com base em características relevantes como turno, máquina, operador, etc.*”

* **Exemplo:** Existe diferênça de desempenho entre os turnos?

#### 1.1.1.2 Tipos de Estratificação

* Por **tempo:** turno, dia da semana, mês;
* Por **local:** máquina, setor, linha de produção;
* Por **pessoas:** operador, equipe;
* Por **método** ou **material**.

df <- data.frame(  
 Tipo = c("Tempo", "Local", "Pessoa", "Método"),  
 Exemplo = c("Turno", "Máquina", "Operador", "Matéria-prima")  
)  
  
knitr::kable(  
 df,  
 escape = FALSE,  
 align = "c",  
 booktabs = TRUE  
)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Tabela 1.1: Exemplos de Tipos de Estratificação.   | Tipo | Exemplo | | --- | --- | | Tempo | Turno | | Local | Máquina | | Pessoa | Operador | | Método | Matéria-prima | |

# Exemplo de Estratificação: Defeitos por Turno  
np.random.seed(11111)  
n = 150  
turno = np.random.choice(["Manhã", "Tarde", "Noite"], size=n)  
defeitos = np.random.binomial(n=10, p=1/3, size=n)  
  
# Formato Data Frame  
df = pd.DataFrame({"Turno": turno, "Defeitos": defeitos})  
  
# Configurações de Figura  
fig, ax = plt.subplots(figsize=(6, 4), dpi=600)  
  
# Scatterplot  
sns.boxplot(x="Turno", y="Defeitos", data=df, hue="Turno", palette="Set2", ax=ax)  
  
# Configurações de eixos e títulos  
ax.set\_xlabel(ax.get\_xlabel(), fontsize=12)  
ax.set\_ylabel(ax.get\_ylabel(), fontsize=12)  
  
# Outras configurações  
ax.spines["top"].set\_visible(False)  
ax.spines["right"].set\_visible(False)  
  
# Exibição do gráfico  
plt.show()

|  |
| --- |
| Figura 1.3: Exemplo 2 de Simulação de Dados Estratificados em Python. |

# Geração de dados  
set.seed(11111)  
n <- 150  
Turno <- sample(c("Manhã", "Tarde", "Noite"), size = n, replace = TRUE)  
Defeitos <- rbinom(n, size = 10, prob = 1/3)  
  
# Data frame  
df <- data.frame(Turno = Turno, Defeitos = Defeitos)  
  
# Gráfico boxplot  
ggplot(df, aes(x = Turno, y = Defeitos, fill = Turno)) +  
 geom\_boxplot() +  
 scale\_fill\_brewer(palette = "Set2") +  
 labs(x = "Turno", y = "Defeitos") +  
 theme\_minimal(base\_size = 12) +  
 theme(  
 legend.position = "none", # Para imitar o `hue="Turno"` do seaborn  
 panel.grid.minor = element\_blank(),  
 panel.grid.major = element\_line(color = "gray90"),  
 axis.line = element\_line(color = "black"),  
 axis.ticks = element\_line(color = "black"),  
 panel.border = element\_blank()  
 )

|  |
| --- |
| Figura 1.4: Exemplo 2 de Simulação de Dados Estratificados em R. |

### 1.1.2 Folhas de Verificação

São formulários usados para **coletar e organizar dados** de forma sistemática.

* Facilitam a visualização e interpretação de dados.
* Podem ser adaptados para diversos propósitos.

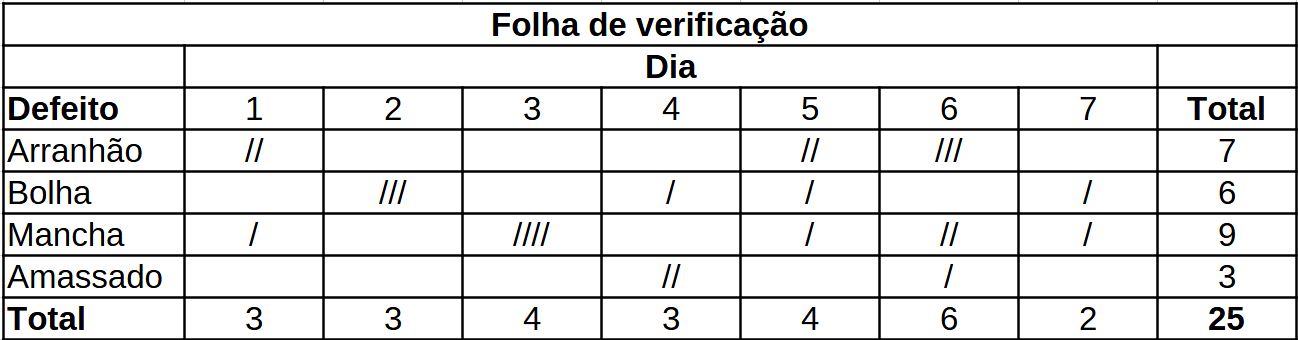
#### 1.1.2.1 Definição de Folha de Verificação

““*Documento estruturado para registrar dados observacionais em tempo real.*

Usada para:

* Contagem de defeitos
* Localização de falhas
* Frequência de ocorrências

🤔 Verificação: Distribuição do Processo de Produção  
🤔 Verificação: Item Defeituoso  
🤔 Verificação: Localização de Defeitos  
🤔 Verificação: Causas de um defeito ou falha  
🤔 Verificação: Satisfação do Cliente (ex.: questionários de satisfação)



**Exemplo de folha de verificação de defeitos na lataria de um carro.**

# Dados da Folha de Verificação  
defect\_types = ["Amassado", "Arranhão", "Bolha", "Mancha"]  
defect\_counts = [3, 7, 6, 9]  
  
# Configurações de Figura  
fig, ax = plt.subplots(figsize=(6, 4), dpi=600)  
  
# Gráfico de Barras  
sns.barplot(x=defect\_types, y=defect\_counts, ax=ax)  
  
# Configurações de eixos e títulos  
ax.set\_xlabel("Defeito", fontsize=12)  
ax.set\_ylabel("Quantidade", fontsize=12)  
  
# Outras configurações  
ax.spines["top"].set\_visible(False)  
ax.spines["right"].set\_visible(False)  
  
# Exibição do gráfico  
plt.show()

|  |
| --- |
| Figura 1.5: Exemplo Gráfico da Folha de Verificação em Python. |

# Dados da Folha de Verificação  
tipos <- c("Amassado", "Arranhão", "Bolha", "Mancha")  
quantidades <- c(3, 7, 6, 9)  
  
# Gráfico de Barras  
ggplot(data = NULL, aes(x = tipos, y = quantidades)) +  
 geom\_bar(stat = "identity", fill = "blue") +  
 labs(x = "Defeito", y = "Quantidade") +  
 theme\_classic(base\_size = 12)

|  |
| --- |
| Figura 1.6: Exemplo Gráfico da Folha de Verificação em R. |

#### 1.1.2.2 Conclusão sobre Folhas de Verificação

✅ Facilitam a **padronização da coleta de dados**  
✅ Auxiliam na **identificação de padrões**  
✅ São a **base para análises gráficas e estatísticas posteriores**

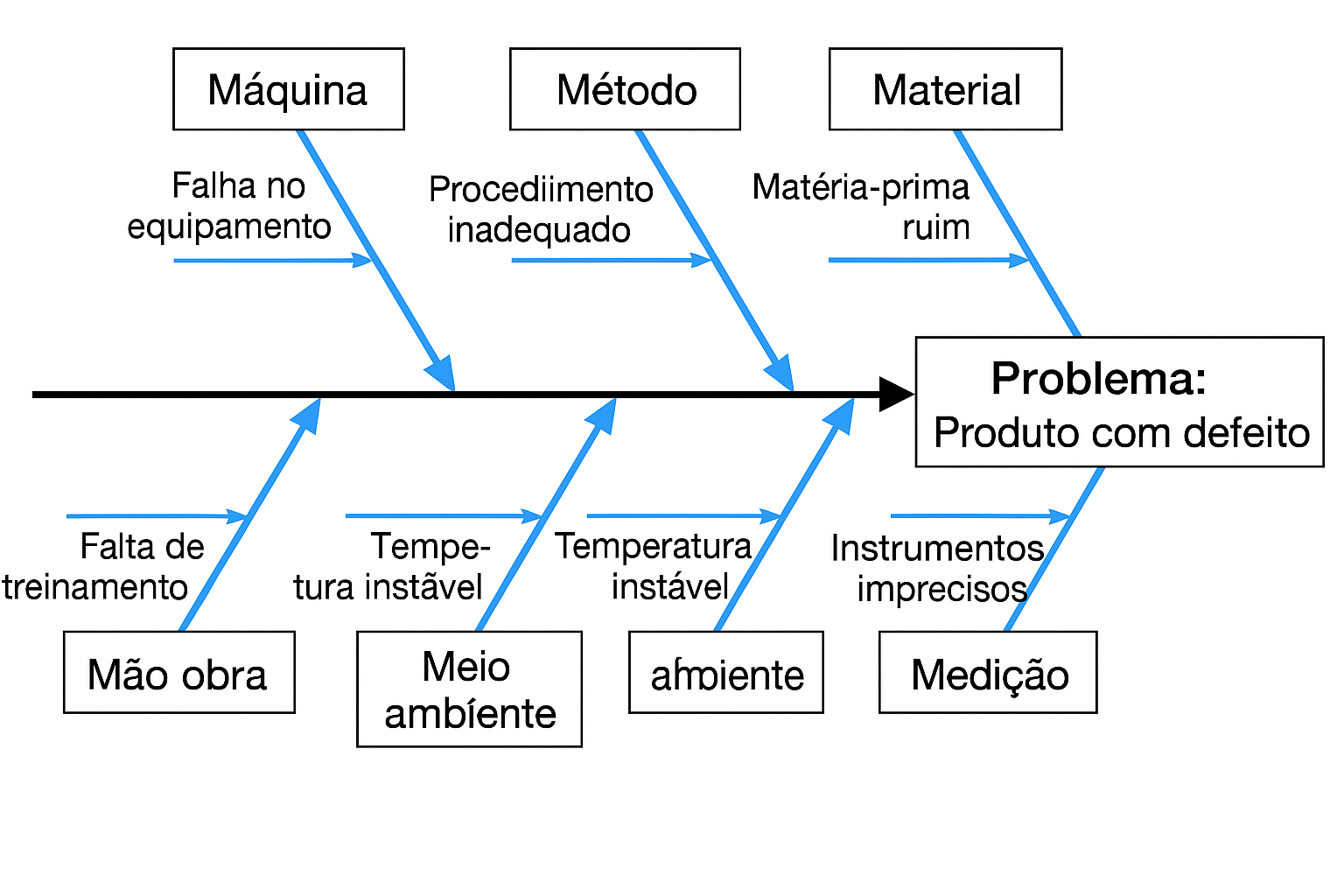
### 1.1.3 Diagrama de Ishikawa

Também conhecido como diagrama de causa e efeito ou espinha de peixe.

* Ferramenta para análise de problemas.
* Organiza causas potenciais de um efeito específico.

#### 1.1.3.1 Como construir um Diagrama de Ishikawa

1. Defina claramente o problema (efeito).
2. Trace uma linha horizontal com o problema no final (efeito).
3. Adicione as categorias principais de causa (método, máquina, mão de obra, material, meio ambiente, medição, etc.)
4. Liste causas específicas em cada categoria.



**Exemplo de diagrama de causa e efeito (Ishikawa).**

# 2. Gráficos Usuais

library(ggplot2)  
library(gridExtra)  
library(dplyr)  
  
library(reticulate)  
use\_python("C:/Users/user/anaconda3/python.exe", required = TRUE)

# Tratamentos de Dados  
import numpy as np  
import pandas as pd  
  
# Funções Estatísticas  
from scipy import stats  
  
# Visualizações Gráficas  
import matplotlib.pyplot as plt  
import seaborn as sns

## 2.1 Histograma

* **O que é um Histograma?**
  + Um histograma é uma representação gráfica da distribuição de frequências de dados contínuos.
  + Mostra como os valores se distribuem por intervalos (classes).
  + Ajuda a visualizar:
    - Tencência central
    - Dispersão
    - Assimetria
    - Possíveis anomalias

### 2.1.1 Construção do Histograma com Limites de Especificação

* Limites de especificação:
  + **Limite inferior de especificação (LSE):** menor valor permitido para uma característica de qualidade.
  + **Limite superior de especificação (LIE):** maior valor permitido para uma característica de qualidade.
* Etapas principais:
  1. Coletar dados contínuos (ex.: tempo, peso, medida, etc.).
  2. Definir os intervalos de classe.
  3. Contar quantos dados caem em cada intervalo.
  4. Representar as frequências com barras adjacentes.

# Exemplo Prático: Histograma  
np.random.seed(11111)  
values = np.random.normal(loc=np.mean([85, 120]), scale=5.5, size=100)  
  
# Configurações de Figura  
fig, axes = plt.subplots(1, 2, figsize=(12, 4), dpi=600)  
  
# Histogrma  
sns.histplot(data=values, bins=7, binrange=(85, 120), binwidth=5, edgecolor="white", ax=axes[0])  
sns.histplot(data=values, bins=7, binrange=(85, 120), binwidth=5, edgecolor="white", ax=axes[1])  
  
# LIE & LSE  
axes[1].axvline(x=90, color="red", linestyle="dashdot", label="LIE")  
axes[1].axvline(x=110, color="red", linestyle="dashed", label="LSE")  
  
# Configurações de legenda  
axes[1].legend(prop={"size":8}, loc="upper right", frameon=False)  
  
# Configurações de eixos e títulos  
axes[0].set\_title("Histograma das Medidas", fontsize=12, weight="bold")  
axes[1].set\_title("Histograma das Medidas c/ Limites de Especificação", fontsize=12, weight="bold")  
  
for ax in axes:  
 # Configurações de eixos e títulos  
 ax.set\_xlabel("Valor Medido", fontsize=12)  
 ax.set\_ylabel("Frequência", fontsize=12)  
  
 # Outras configurações  
 ax.spines["top"].set\_visible(False)  
 ax.spines["right"].set\_visible(False)  
  
# Ajuste do Layout e Exibição da Figura  
fig.tight\_layout()  
plt.show()

|  |
| --- |
| Figura 2.1: Exemplo de Histograma com Limites de Especificação em Python. |

# Exemplo Prático: Histograma  
set.seed(11111)  
valores <- rnorm(100, mean = mean(c(85, 120)), sd = 5.5)  
  
# Base para histograma  
df <- data.frame(valor = valores)  
  
# Histograma 1  
hist1 <- ggplot(df, aes(x = valor)) +  
 geom\_histogram(binwidth = 5, boundary = 85, color = "white", fill = "#3B9AB2") +  
 labs(title = "Histograma das Medidas", x = "Valor Medido", y = "Frequência") +  
 theme\_minimal(base\_size = 12) +  
 theme(  
 panel.grid.minor = element\_blank(),  
 panel.grid.major = element\_line(color = "gray90"),  
 axis.line = element\_line(color = "black"),  
 axis.ticks = element\_line(color = "black"),  
 panel.border = element\_blank()  
 )  
  
# Histograma 2 com LIE e LSE  
hist2 <- ggplot(df, aes(x = valor)) +  
 geom\_histogram(binwidth = 5, boundary = 85, color = "white", fill = "#3B9AB2") +  
 geom\_vline(xintercept = 90, color = "red", linetype = "dotdash", size = 0.8) +  
 geom\_vline(xintercept = 110, color = "red", linetype = "dashed", size = 0.8) +  
 annotate("text", x = 90, y = Inf, label = "LIE", vjust = 2, hjust = -0.1, size = 3, color = "red") +  
 annotate("text", x = 110, y = Inf, label = "LSE", vjust = 2, hjust = -0.1, size = 3, color = "red") +  
 labs(title = "Histograma das Medidas c/ Limites de Especificação", x = "Valor Medido", y = "Frequência") +  
 theme\_minimal(base\_size = 12) +  
 theme(  
 panel.grid.minor = element\_blank(),  
 panel.grid.major = element\_line(color = "gray90"),  
 axis.line = element\_line(color = "black"),  
 axis.ticks = element\_line(color = "black"),  
 panel.border = element\_blank()  
 )  
  
# Exibição lado a lado  
grid.arrange(hist1, hist2, ncol = 2)

|  |
| --- |
| Figura 2.2: Exemplo de Histograma com Limites de Especificação em R. |

* ✅ Quando a maioria dos dados está entre LIE e LSE ➡ **Processo Capaz**.
* ⚠ Quando muitos dados estão fora dos limites ➡ **Processo Não Capaz**.

## 2.2 Gráfico de Pareto

É um gráfico de barras que **ordena as causas ou categorias em ordem decrescente de frequência**.

* Baseado no Princípio de Pareto (80/20)
* 80% dos resultados provêm de 20% das causas
* Ajuda a identificar os principais problemas
* **Exemplo:** Dados Simulados de Defeitos.

df <- data.frame(  
 Categoria = c("Erro A", "Erro B", "Erro C", "Erro D", "Erro E"),  
 Freq = c(40, 25, 15, 12, 8)  
)  
  
knitr::kable(  
 df,  
 escape = FALSE,  
 align = "c",  
 booktabs = TRUE  
)

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Tabela 2.1: Dados Simulados de Defeitos.   | Categoria | Freq | | --- | --- | | Erro A | 40 | | Erro B | 25 | | Erro C | 15 | | Erro D | 12 | | Erro E | 8 | |

# Exemplo do Gráfico de Pareto  
df = pd.DataFrame({  
 "Causa": ["Erro A", "Erro B", "Erro C", "Erro D", "Erro E"],  
 "Frequência": [40, 25, 15, 12, 8],  
})  
  
df["Percentual Acumulado"] = df["Frequência"].cumsum() / df["Frequência"].sum() \* 100  
  
# Configurações de Figura  
fig, ax1 = plt.subplots(figsize=(6, 4), dpi=600)  
  
# Criando um 2º eixo y  
ax2 = ax1.twinx()  
  
# Gráfico de Barras - Frequência  
sns.barplot(data=df, x="Causa", y="Frequência", ax=ax1)  
  
# Gráfico de Linhas - Percentual Acumulado  
sns.lineplot(data=df, x="Causa", y="Percentual Acumulado", marker="o", linestyle="-", color="red", ax=ax2)  
  
# Configurações de eixos e títulos  
ax1.set\_xlabel("Causa", fontsize=12)  
ax1.set\_ylabel("Frequência", fontsize=12)  
#ax1.set\_ylim(0, 40)  
  
ax2.set\_ylabel("Percentual Acumulado", fontsize=12)  
#ax2.set\_ylim(0, 100)  
ax2.set\_yticklabels([f"{fr}%" for fr in np.arange(0, 101, 20)])  
  
# Outras configurações  
ax1.spines["top"].set\_visible(False)  
ax2.spines["top"].set\_visible(False)  
ax1.spines["bottom"].set\_visible(False)  
ax2.spines["bottom"].set\_visible(False)  
  
# Exibição da Figura  
plt.show()

|  |
| --- |
| Figura 2.3: Exemplo do Gráfico de Pareto em Python. |

df <- data.frame(  
 Causa = c("Erro A", "Erro B", "Erro C", "Erro D", "Erro E"),  
 Frequencia = c(40, 25, 15, 12, 8)  
)  
  
# Cálculo do Percentual Acumulado  
df$PercentualAcumulado <- cumsum(df$Frequencia) / sum(df$Frequencia) \* 100  
  
# ------------------------------  
# Gráfico de Pareto  
# ------------------------------  
# Base do gráfico com barras  
p <- ggplot(df, aes(x = Causa)) +  
 geom\_bar(aes(y = Frequencia), stat = "identity", fill = "steelblue") +  
 geom\_line(aes(y = PercentualAcumulado \* max(Frequencia) / 100),   
 group = 1, color = "red", size = 1) +  
 geom\_point(aes(y = PercentualAcumulado \* max(Frequencia) / 100),   
 color = "red", size = 3) +  
   
 # Eixos primário e secundário  
 scale\_y\_continuous(  
 name = "Frequência",  
 limits = c(0, 40),  
 sec.axis = sec\_axis(  
 trans = ~ . / max(df$Frequencia) \* 100,  
 name = "Percentual Acumulado",  
 labels = function(x) paste0(x, "%")  
 )  
 ) +  
   
 labs(x = "Causa") +  
 theme\_classic(base\_size = 12) +  
 theme(  
 axis.line.y.right = element\_line(color = "red"),  
 axis.ticks.y.right = element\_line(color = "red"),  
 axis.text.y.right = element\_text(color = "red"),  
 axis.title.y.right = element\_text(color = "red"),  
 panel.grid.minor = element\_blank()  
 )  
  
# ------------------------------  
# Exibição do gráfico  
# ------------------------------  
print(p)

|  |
| --- |
| Figura 2.4: Exemplo do Gráfico de Pareto em R. |

## 2.3 Diagrama de Correlação ou Diagrama de Dispersão

### 2.3.1 Construção do Diagrama de Correlação

* **Passos:**
  1. Coletar pares de observações (, );
  2. Plotar os pontos em um gráfico de dispersão;
  3. Analisar visualmente a existência e o tipo de correlação.
* **Correlação Linear Positiva:**
  + Quando uma variável aumenta, a outra também tende a aumentar.
  + Os pontos seguem uma tendência crescente.
* **Correlação Linear Negativa:**
  + Quando uma variável aumenta, a outra também tende a diminuir.
  + Os pontos seguem uma tendência decrescente.

# Exemplos: Para Diagramas de Dispersão  
np.random.seed(11111)  
  
x1 = np.random.normal(loc=50, scale=10, size=100)  
y1 = 3/2 \* x1 + np.random.normal(loc=0, scale=10, size=100)  
  
x2 = np.random.normal(loc=50, scale=10, size=100)  
y2 = -3/2 \* x2 + np.random.normal(loc=0, scale=10, size=100)  
  
# Configurações de Figura  
fig, axes = plt.subplots(1, 2, figsize=(12, 4), dpi=600)  
  
# Diagramas de Dispersão  
sns.scatterplot(x=x1, y=y1, ax=axes[0])  
sns.scatterplot(x=x2, y=y2, ax=axes[1])  
  
# Configurações de eixos e títulos  
axes[0].set\_title("Correlação Linear Positiva", fontsize=12, weight="bold")  
axes[1].set\_title("Correlação Linear Negativa", fontsize=12, weight="bold")  
  
for ax in axes:  
 # Configurações de eixos e títulos  
 ax.set\_xlabel("Variável X", fontsize=12)  
 ax.set\_ylabel("Variável Y", fontsize=12)  
  
 # Outras configurações  
 ax.spines["top"].set\_visible(False)  
 ax.spines["right"].set\_visible(False)  
  
# Ajuste do Layout e Exibição da Figura  
fig.tight\_layout()  
plt.show()

|  |
| --- |
| Figura 2.5: Exemplo de Diagrama de Dispersão em Python. |

# Semente para reprodutibilidade  
set.seed(11111)  
  
# Geração dos dados  
x1 <- rnorm(100, mean = 50, sd = 10)  
y1 <- (3/2) \* x1 + rnorm(100, mean = 0, sd = 10)  
  
x2 <- rnorm(100, mean = 50, sd = 10)  
y2 <- (-3/2) \* x2 + rnorm(100, mean = 0, sd = 10)  
  
# Data frames para os gráficos  
df1 <- data.frame(X = x1, Y = y1)  
df2 <- data.frame(X = x2, Y = y2)  
  
# Diagrama de Dispersão: Correlação Positiva  
scatter1 <- ggplot(df1, aes(x = X, y = Y)) +  
 geom\_point(color = "#1f77b4") +  
 labs(title = "Correlação Linear Positiva", x = "Variável X", y = "Variável Y") +  
 theme\_minimal(base\_size = 12) +  
 theme(  
 panel.grid.minor = element\_blank(),  
 panel.grid.major = element\_line(color = "gray90"),  
 axis.line = element\_line(color = "black"),  
 axis.ticks = element\_line(color = "black"),  
 panel.border = element\_blank()  
 )  
  
# Diagrama de Dispersão: Correlação Negativa  
scatter2 <- ggplot(df2, aes(x = X, y = Y)) +  
 geom\_point(color = "#d62728") +  
 labs(title = "Correlação Linear Negativa", x = "Variável X", y = "Variável Y") +  
 theme\_minimal(base\_size = 12) +  
 theme(  
 panel.grid.minor = element\_blank(),  
 panel.grid.major = element\_line(color = "gray90"),  
 axis.line = element\_line(color = "black"),  
 axis.ticks = element\_line(color = "black"),  
 panel.border = element\_blank()  
 )  
  
# Exibição lado a lado  
grid.arrange(scatter1, scatter2, ncol = 2)

|  |
| --- |
| Figura 2.6: Exemplo de Diagrama de Dispersão em R. |

* **Ausência de Correlação Linear:**
  + Os pontos não seguem padrão algum.
  + Indica ausência de relação linear.

# Exemplos: Para Diagramas de Dispersão  
np.random.seed(11111)  
  
x1 = np.random.normal(loc=50, scale=5, size=100)  
y1 = np.random.normal(loc=50, scale=15, size=100)  
  
# Configurações de Figura  
fig, ax = plt.subplots(figsize=(6, 4), dpi=600)  
  
# Diagramas de Dispersão  
sns.scatterplot(x=x1, y=y1, ax=ax)  
  
# Configurações de eixos e títulos  
ax.set\_title("Ausência de Correlação Linear", fontsize=12, weight="bold")  
ax.set\_xlabel("Variável X", fontsize=12)  
ax.set\_ylabel("Variável Y", fontsize=12)  
  
# Outras configurações  
ax.spines["top"].set\_visible(False)  
ax.spines["right"].set\_visible(False)  
  
# Exibição da Figura  
plt.show()

|  |
| --- |
| Figura 2.7: Exemplo de Diagrama de Dispersão (Sem Relação Linear) em Python. |

# Semente para reprodutibilidade  
set.seed(11111)  
  
# Geração dos dados  
x1 <- rnorm(100, mean = 50, sd = 5)  
y1 <- rnorm(100, mean = 50, sd = 15)  
  
# Data frame  
df <- data.frame(X = x1, Y = y1)  
  
# Diagrama de Dispersão  
ggplot(df, aes(x = X, y = Y)) +  
 geom\_point(color = "#1f77b4") +  
 labs(  
 title = "Ausência de Correlação Linear",  
 x = "Variável X",  
 y = "Variável Y"  
 ) +  
 theme\_minimal(base\_size = 12) +  
 theme(  
 panel.grid.minor = element\_blank(),  
 panel.grid.major = element\_line(color = "gray90"),  
 axis.line = element\_line(color = "black"),  
 axis.ticks = element\_line(color = "black"),  
 panel.border = element\_blank()  
 )

|  |
| --- |
| Figura 2.8: Exemplo de Diagrama de Dispersão (Sem Relação Linear) em R. |

### 2.3.2 Cálculo do Coeficiente de Correlação Linear de Pearson

O *Coeficiente de Correlação Linear de Pearson* mede a força e direção da relação linear entre duas variáveis. Tal medida é obtida a partir da expressão:

* **Obeservações Importantes:**
  + Varia de :
    - : correlação positiva perfeita;
    - : sem correlação linear;
    - : correlação negativa perfeita.

### 2.3.3 Teste de Hipótese para o Coeficiente de Correlação Linear de Pearson

Normalmente, se testa a significância de com as seguintes hipóteses:

Usa-se para testar as hipótes da [Equação 2.2](#eq-HypotesisCorrPearson) a seguinte estatística de teste:

que sob a hipótese nula () segue uma distribuição . Vejamos o exemplo:

# Função equivalente ao shapiro.test do R  
def shapiro\_test(x):  
 W, p\_value = stats.shapiro(x)  
 print(  
 f"""  
 Estatística de Teste (W) = {W:.4f}  
 Nível Descritivo (p-value) = {p\_value:.4f}  
 """  
 )  
  
# Classe para calcular o coeficiente de correlação de Pearson e realizar testes de hipótese  
class PearsonCorrelation:  
 """  
 Classe para calcular o coeficiente de correlação de Pearson e realizar testes de hipótese.  
 """  
  
 def \_\_init\_\_(self, x, y, alpha=0.05):  
 self.x = x  
 self.y = y  
 self.alpha = alpha  
  
 self.r = None  
 self.interval\_confidence = None  
  
 self.t\_statistic = None  
 self.df = None  
 self.p\_value = None  
  
 def calculate\_correlation(self):  
 """  
 Calcula o coeficiente de correlação de Pearson.  
 """  
 r = np.corrcoef(self.x, self.y)[0, 1]  
 z = np.arctanh(r)  
 se = 1 / np.sqrt(len(self.y) - 3)  
 lower\_bound = np.tanh(z - stats.norm.ppf(1 - self.alpha / 2) \* se)  
 upper\_bound = np.tanh(z + stats.norm.ppf(1 - self.alpha / 2) \* se)  
  
 self.r = r  
 self.interval\_confidence = (lower\_bound, upper\_bound)  
  
 return r, (lower\_bound, upper\_bound)  
  
 def hypothesis\_test(self):  
 """  
 Realiza o teste de hipótese para o coeficiente de correlação.  
 """  
 n = len(self.y)  
 self.t\_statistic = self.r \* np.sqrt(n - 2) / np.sqrt(1 - self.r\*\*2)  
 self.df = n - 2  
 self.p\_value = stats.t.sf(np.abs(self.t\_statistic), self.df) \* 2  
  
 return self.t\_statistic, self.df, self.p\_value  
  
 def result\_print(self):  
 """  
 Imprime os resultados do cálculo e do teste de hipótese.  
 """  
 print(  
 f"""  
 - Coeficiente de Correlação Linear de Pearson:  
 r = {self.r:.4f}  
 Intervalo de Confiança ({self.alpha\*100:.0f}%):  
 ({self.interval\_confidence[0]:.4f}, {self.interval\_confidence[1]:.4f})  
  
 - Teste de Hipótese:  
 t = {self.t\_statistic:.4f}, df = {self.df}, p-value = {self.p\_value:.4f}  
 """  
 )

* **Geração do Dados & Visualização Gráfica:**

# Exemplo Completo: Passo a Passo  
np.random.seed(123)  
  
x1 = np.random.normal(loc=10, scale=1, size=30)  
y1 = x1 + np.random.normal(loc=0, scale=1, size=30)  
  
# Configurações de Figura  
fig, ax = plt.subplots(figsize=(6, 4), dpi=600)  
  
# Diagramas de Dispersão  
sns.scatterplot(x=x1, y=y1, color="lightblue", ax=ax)  
  
# Configurações de eixos e títulos  
ax.set\_title("Relação entre X e Y", fontsize=12, weight="bold")  
ax.set\_xlabel("Variável X", fontsize=12)  
ax.set\_ylabel("Variável Y", fontsize=12)  
  
# Outras configurações  
ax.spines["top"].set\_visible(False)  
ax.spines["right"].set\_visible(False)  
  
# Exibição da Figura  
plt.show()

|  |
| --- |
| Figura 2.9: Dados Simulados para Exemplo em Python. |

# Fixando semente para geração  
set.seed(123)  
  
# Gerando Dados  
x <- rnorm(30, mean = 10)  
y <- x + rnorm(30)  
  
# Ajustando Formato  
df <- data.frame(X = x, Y = y)  
  
# Gerando Plot  
ggplot(df, aes(x = X, y = Y)) +  
 geom\_point(color = "lightblue", size = 3) +  
 labs(title = "Relação entre X e Y",  
 x = "Variável X", y = "Variável Y") +  
 theme\_classic(base\_size = 12)

|  |
| --- |
| Figura 2.10: Dados Simulados para Exemplo em R. |

* **Cálculo do Coeficiente de Correlação:**

# Nível de Significância  
ns = 0.05  
  
# Exemplo de uso da classe PearsonCorrelation  
Pearson = PearsonCorrelation(x1, y1, alpha=ns)  
  
# Coeficiente de Correlação de Pearson  
r, ci = Pearson.calculate\_correlation()  
  
# Impressão dos resultados  
print(f"Coeficiente de Correlação de Pearson: r = {r:.4f}")

Coeficiente de Correlação de Pearson: r = 0.6424

print(f"Intervalo de Confiança ({(1 - ns) \* 100:.0f}%): {ci[0]:.4f}, {ci[1]:.4f}")

Intervalo de Confiança (95%): 0.3671, 0.8142

# Nível de Significância  
ns <- 0.05  
  
# Coeficiente de Correlação de Pearson  
result <- cor.test(x, y)  
  
# Impressão dos resultados  
cat("Coeficiente de Correlação de Pearson: r =", round(result$estimate, 4), "\n")

Coeficiente de Correlação de Pearson: r = 0.7175

cat("Intervalo de Confiança (", (1 - ns)\*100,"%):", round(result$conf.int[1], 4), round(result$conf.int[2], 4))

Intervalo de Confiança ( 95 %): 0.4818 0.8564

* **Teste de Hipótes para o Coeficiente de Correlação:**

# Verificação dos Pressupostos (Normalidade)  
shapiro\_test(x1)

Estatística de Teste (W) = 0.9621  
 Nível Descritivo (p-value) = 0.3509

shapiro\_test(y1)

Estatística de Teste (W) = 0.9627  
 Nível Descritivo (p-value) = 0.3630

# Teste de Hipótese  
result = Pearson.hypothesis\_test()  
  
# Teste de Hipótese  
print(f"Estatística de Teste (t) = {result[0]:.4f}")

Estatística de Teste (t) = 4.4353

print(f"Graus de Liberdade (df) = {result[1]}")

Graus de Liberdade (df) = 28

print(f"Nível Descritivo (p-value) = {result[2]:.4f}")

Nível Descritivo (p-value) = 0.0001

# Verificação dos Pressupostos (Normalidade)  
shapiro.test(x)

Shapiro-Wilk normality test  
  
data: x  
W = 0.97894, p-value = 0.7966

shapiro.test(y)

Shapiro-Wilk normality test  
  
data: y  
W = 0.96204, p-value = 0.3488

# Impressão dos resultados  
cat("Estatística de Teste: t =", round(result$statistic, 4), "\n")

Estatística de Teste: t = 5.4508

cat("Graus de Liberdade: gl =", result$parameter, "\n")

Graus de Liberdade: gl = 28

cat("Valor-p:", round(result$p.value, 4), "\n")

Valor-p: 0

### 2.3.4 Conclusão

* São úteis para investigar relação entre variáveis ✅
* Ajudam a detectar tendências visuais ✅
* O coeficiente de Pearson quantifica a força da relação ✅
* Há um teste que verifica a significância estatística dessa relação ✅

# 3. Listas e Exercícios

## 3.1 Lista I

### 3.1.1 1. Simule (no R **e** Python) um conjunto de dados com três turnos de produção e números de defeitos.

#### 3.1.1.1 a. Faça um boxplot para comparar os defeitos entre turnos.

#### 3.1.1.2 b. Comente se a estratificação revela alguma diferença relevante.

### 3.1.2 2. Monte um diagrama de espinha de peixe para o seguinte problema: “Produto entregue com atraso”. Use papel ou software. Sugestões de Pacote no R: Mermaid e DiagrammeR.

### 3.1.3 3. Com base nos dados a seguir, construa um gráfico de Pareto (no papel, R ou Python) e interprete os resultados.

| **Problemas** | **Frequência** |
| --- | --- |
| Risco | 80 |
| Mancha | 68 |
| Corte | 50 |
| Tinta Fraca | 45 |
| Erro de Montagem | 30 |

#### 3.1.3.1 a. Quais problemas devem ser atacados primeiro?

#### 3.1.3.2 b. Qual o percentual acumaludo dos dois problemas mais frquentes?

### 3.1.4 4. Simule 200 observações com e .

#### 3.1.4.1 a.Crie um histograma.

#### 3.1.4.2 b. Defina limites de especificaçãp mais estreitos: e .

### 3.1.5 5. Reúna-se com seu grupo faça o seguinte:

#### 3.1.5.1 ⚫ Colete um conjunto de dados reais (ex.: tempo para executar uma tarefa simples).

#### 3.1.5.2 ⚫ Classifique os dados usando estratificação (ex.: por turno, grupo, dia, etc.).

#### 3.1.5.3 ⚫ Construa um histograma, gráfico de Pareto e, se possível um diagrama de Ishikawa para o problema observado.

#### 3.1.5.4 ⚫ Apresente os resultados com uma breve conclusão.

# 4. Lista II

## 4.1 1. Utilize os vetores abaixo e construa o diagrama de dispersão. e .

### 4.1.1 a. Descreva o tipo de relação entre as variáveis.

### 4.1.2 b. Adicione uma reta de tendência.

## 4.2 2. Geração de dados com correlação negativa.

### 4.2.1 a. Gere dois vetores de 30 elementos com correlação negativa.

### 4.2.2 b. Construa o gráfico de dispersão.

### 4.2.3 c. Calcule a correlação de Pearson.

## 4.3 3. Dados reais - mtcars. Utilize o conjunto de dados mtcars.

### 4.3.1 a. Há relação entre mpg (milhas por galão) e wt (peso)?

### 4.3.2 b. Faça o gráfico e interprete-o.

### 4.3.3 c. Calcule a correlação de maneira adequada.

### 4.3.4 d. A relação é positiva ou negativa?

## 4.4 4. Construção de Função - Crie uma função correlacao\_diagnostico() que:

* Plote o gráfico de dispersão
* Calcule o r
* Execute o teste cor\_test()
* Apresente o valor-p do teste

# References