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).

Para esta etapa inicial foram usadas as referências:

* Montgomery (2013)
* “Mermaid.js” (s.d.)

# 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.

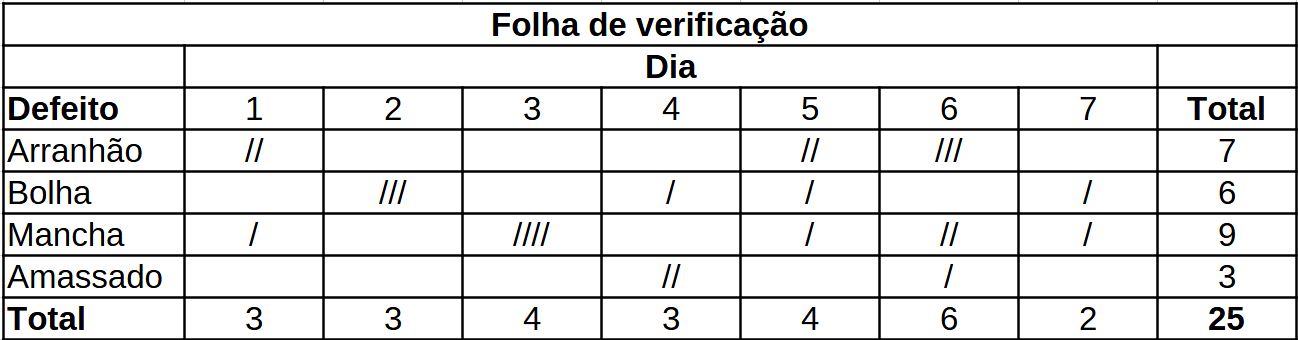
#### 1.1.2.1 Definição de Folha de Verificação

““*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**

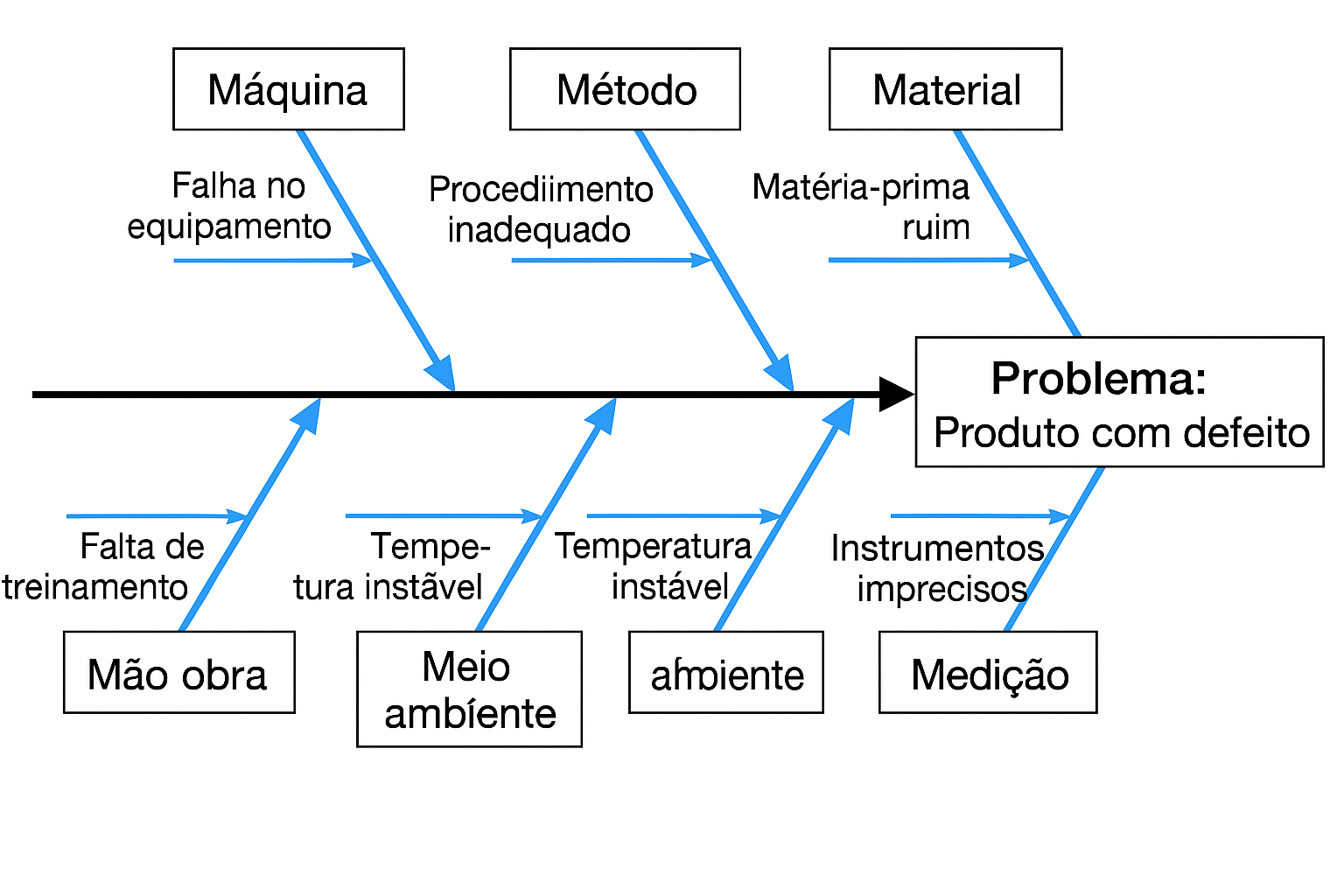
### 1.1.3 Diagrama de Ishikawa

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. Gráficos de Controle

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

## 3.1 Visão Geral da Inferência Estatística

A **inferência estatística** permite tirar conclusões sobre um processo a partir de uma **amostra**.

* Estimamos parâmetros populacionais (como média e variância).
* Avaliamos a **variabilidade natural** dos dados.
* Essencial para construir **gráficos de controle** com base em dados amostrais.

### 3.1.1 Duas abordagens principais

* **Estimação:** Processo de aplicação do *estimador* (fórmula) para se obter *estimativas*.
  + **Pontual:** Cálculo de valores pontuais para parâmetros do processo.
  + **Intervalar:** Cálculo de intervalos de confiança para parâmetros do processo.
* **Teste de Hipóteses:** Decisões sobre mudanças no processo com base em evidência estatística.

### 3.1.2 Propriedades dos Estimadores

Um estimador é uma função da amostra usada para estimar um parâmetro.

**Boas propriedades desejáveis:**

1. *Não-viesado:* Valor esperado (médio) do estimador é converge (igual) ao parâmetro verdadeiro, isto é,
2. *Consistência:* A medida que (tamanho amostral) tende ao infinito, o estimador converge, em probabilidade, ao verdadeiro valor do parâmetro. Isto é, seja uma amostra aleatória de uma variável aleatória com média e variância .

Um estimador para , é dito consistente se:

Em geral, a desigualdade de Chebyshev pode ser usada para verificar essa propriedade:

Portanto, à medida que cresce, a probabilidade de o estimador estar distante do verdadeiro parâmetro tende a zero, garantindo sua consistência.

1. *Eficiência:* Possui a menor variância possível entre os estimadores não-viesados. Isto é, seja uma amostra aleatória de uma variável com parâmetro

Um estimador é eficiente se sua variância atinge o limite inferior dado por:

em que é a ***Informação de Fisher*** expressa por:

## 3.2 Estimações do Processo

### 3.2.1 Estimando a Dispersão do Processo

A dispersão do processo pode ser estimada usando:

* Desvio padrão amostral ();
* Amplitude Total ();
* Desvio médio absoluto.

### 3.2.2 Estimando o Nível do Processo

O nível do processo geralmente se refere à média do processo.

* É representado por , a média verdadeira da variável de interesse;
* É estimada fazendo uso da distribuição amostral de , isto é, .

## 3.3 Resumo Geral

* A inferência estatística fornece as ferramentas para definir limites e linhas centrais ✅;
* A precisão da estimativa depende do tamanho e representatividade da amostra ✅;
* Com base nestes conceitos, constrói-se gráficos de controle como . entre outros ✅.

## 3.4 Visão Geral de Gráficos de Controle

Os gráficos de controle são ferramentas gráficas usadas para monitorar processos ao longo do tempo.

* Detectam **variações comuns** (naturais) e **variações especiais** (anomalias).
* Ajudam na **manutenção da qualidade** e na **tomada de decisões**.

### 3.4.1 Princípios dos Gráficos de Controle

* Baseiam-se em **amostras coletadas periodicamente**;
* Mostram uma **linha central** () representando o comportamento esperado do processo;
* Possuem **limites de controle** ( e ) que indicam variações aceitáveis;
* Proposto ppr Shewhart (1926);
* Controlar a variabilidade dos processos.

**Proposta geral:**

Temos que:

e

Se , então

que é equivalente a

.

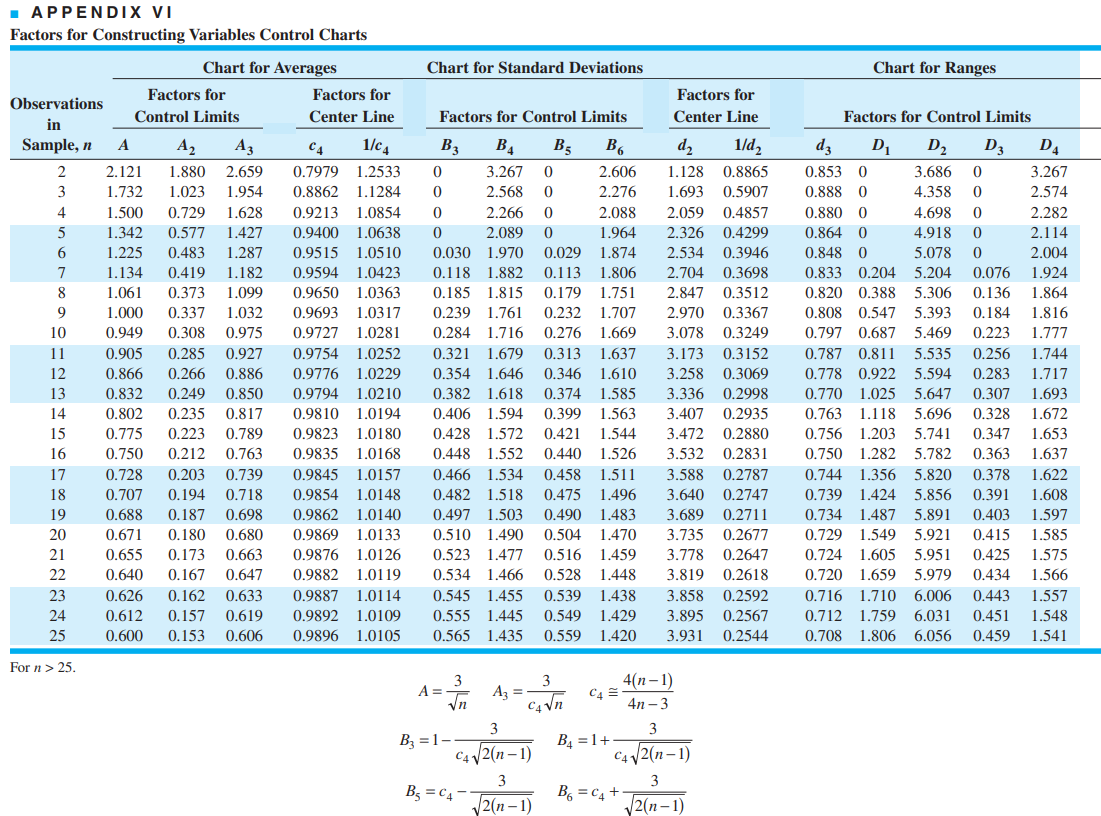
### 3.4.2 Gráficos de controle para variáveis

#### 3.4.2.1 Gráficos com valores padrão

* **Variância Conhecida:** Quando se tem conhecimento da variância populacional, usa-se as expressões abaixo:
* Ou, de forma sintetizada:
* Sendo

#### 3.4.2.2 Gráficos com valores tabelados

* **Principais gráficos:**
  + e ;
  + e ;
  + Medições individuais ()



**Tabela com fatores para contrução dos gráficos de controle para variáveis.**

**Nota:** Iremos utilizar como o números de características mensuradas (colunas) no processo.

##### 3.4.2.2.1 Gráfico de Controle: e

* **Quando utilizar:**
  + Controle da Média e Amplitude;
  + Indicados para subgrupos pequenos ()
* **Expressões para:**
  + **Controle de :** , e ;
  + **Controle de :** , e ;

##### 3.4.2.2.2 Gráfico de Controle: e

* **Quando utilizar:**
  + Usa Desvio Padrão ao invés da Amplitude;
  + Preferido para amostras maiores ().
* **Expressões para:**
  + **Controle de :** , e ;
  + **Controle de :** , e ;

##### 3.4.2.2.3 Gráfico de Controle:

* Menos comum;
* Baseado na distribuição Qui-Quadrado ().
* **Expressões:**
  + **Controle de :** , e ;
* Útil quando se deseja monitorar diretamente a variância do processo.

### 3.4.3 Exemplo de Construção Didática de Gráficos de Controle

* **Exemplo (6.1) página 239 (Montgomery 2013) - *Medições da Largura de Fluxo (mícrons) para o Processo de Cozimento Duro*:** *“O processo de cozimento duro é utilizado em conjunto com a fotolitografia na fabricação de semicondutores. Desejamos estabelecer o controle estatístico da largura de fluxo do resistor neste processo utilizando gráficos de e . Vinte e cinco amostras, cada uma com wafers de tamanho cinco, foram coletadas quando acreditamos que o processo está sob controle. O intervalo de tempo entre as amostras ou subgrupos é de uma hora. Os dados de medição da largura de fluxo (em x mícrons) dessas amostras são mostrados na Tabela 6.1.”*

# Dados de Exemplo  
list\_of\_data = [  
 [1.3235, 1.4128, 1.6744, 1.4573],  
 [1.4314, 1.3592, 1.6075, 1.4666],  
 [1.4284, 1.4871, 1.4932, 1.4324],  
 [1.5028, 1.6352, 1.3841, 1.2831],  
 [1.5604, 1.2735, 1.5265, 1.4362],  
 [1.5955, 1.5451, 1.3574, 1.3281],  
 [1.6274, 1.5064, 1.8366, 1.4177],  
 [1.4190, 1.4303, 1.6637, 1.6067],  
 [1.3884, 1.7277, 1.5355, 1.5176],  
 [1.4039, 1.6697, 1.5089, 1.6477],  
 [1.4158, 1.7667, 1.4278, 1.5927],  
 [1.5821, 1.3355, 1.5777, 1.3908],  
 [1.2856, 1.4106, 1.4447, 1.6388],  
 [1.4951, 1.4036, 1.5893, 1.6458],  
 [1.3589, 1.2863, 1.5996, 1.2497],  
 [1.5747, 1.5301, 1.5171, 1.1839],  
 [1.3680, 1.7269, 1.3957, 1.5019],  
 [1.4163, 1.3864, 1.3057, 1.6210],  
 [1.5796, 1.4185, 1.6541, 1.5116],  
 [1.7106, 1.4412, 1.2361, 1.3824],  
 [1.4371, 1.5051, 1.3485, 1.5670],  
 [1.4738, 1.5936, 1.6583, 1.4973],  
 [1.5917, 1.4333, 1.5551, 1.5295],  
 [1.6399, 1.5243, 1.5705, 1.5563],  
 [1.5797, 1.3663, 1.6240, 1.3732]  
]  
  
# Transformando em Array Multidimensional  
arr = np.array(list\_of\_data)  
  
# Visualizar  
arr

array([[1.3235, 1.4128, 1.6744, 1.4573],  
 [1.4314, 1.3592, 1.6075, 1.4666],  
 [1.4284, 1.4871, 1.4932, 1.4324],  
 [1.5028, 1.6352, 1.3841, 1.2831],  
 [1.5604, 1.2735, 1.5265, 1.4362],  
 [1.5955, 1.5451, 1.3574, 1.3281],  
 [1.6274, 1.5064, 1.8366, 1.4177],  
 [1.419 , 1.4303, 1.6637, 1.6067],  
 [1.3884, 1.7277, 1.5355, 1.5176],  
 [1.4039, 1.6697, 1.5089, 1.6477],  
 [1.4158, 1.7667, 1.4278, 1.5927],  
 [1.5821, 1.3355, 1.5777, 1.3908],  
 [1.2856, 1.4106, 1.4447, 1.6388],  
 [1.4951, 1.4036, 1.5893, 1.6458],  
 [1.3589, 1.2863, 1.5996, 1.2497],  
 [1.5747, 1.5301, 1.5171, 1.1839],  
 [1.368 , 1.7269, 1.3957, 1.5019],  
 [1.4163, 1.3864, 1.3057, 1.621 ],  
 [1.5796, 1.4185, 1.6541, 1.5116],  
 [1.7106, 1.4412, 1.2361, 1.3824],  
 [1.4371, 1.5051, 1.3485, 1.567 ],  
 [1.4738, 1.5936, 1.6583, 1.4973],  
 [1.5917, 1.4333, 1.5551, 1.5295],  
 [1.6399, 1.5243, 1.5705, 1.5563],  
 [1.5797, 1.3663, 1.624 , 1.3732]])

# Dados do Problema  
dados <- c(  
 1.3235, 1.4128, 1.6744, 1.4573,  
 1.4314, 1.3592, 1.6075, 1.4666,  
 1.4284, 1.4871, 1.4932, 1.4324,  
 1.5028, 1.6352, 1.3841, 1.2831,  
 1.5604, 1.2735, 1.5265, 1.4362,  
 1.5955, 1.5451, 1.3574, 1.3281,  
 1.6274, 1.5064, 1.8366, 1.4177,  
 1.4190, 1.4303, 1.6637, 1.6067,  
 1.3884, 1.7277, 1.5355, 1.5176,  
 1.4039, 1.6697, 1.5089, 1.6477,  
 1.4158, 1.7667, 1.4278, 1.5927,  
 1.5821, 1.3355, 1.5777, 1.3908,  
 1.2856, 1.4106, 1.4447, 1.6388,  
 1.4951, 1.4036, 1.5893, 1.6458,  
 1.3589, 1.2863, 1.5996, 1.2497,  
 1.5747, 1.5301, 1.5171, 1.1839,  
 1.3680, 1.7269, 1.3957, 1.5019,  
 1.4163, 1.3864, 1.3057, 1.6210,  
 1.5796, 1.4185, 1.6541, 1.5116,  
 1.7106, 1.4412, 1.2361, 1.3824,  
 1.4371, 1.5051, 1.3485, 1.5670,  
 1.4738, 1.5936, 1.6583, 1.4973,  
 1.5917, 1.4333, 1.5551, 1.5295,  
 1.6399, 1.5243, 1.5705, 1.5563,  
 1.5797, 1.3663, 1.6240, 1.3732  
)  
  
# Formatação de Vetor para Matriz & de Matriz para Data Frame  
tbl <- matrix(dados, ncol = 4, byrow = TRUE)  
tbl.df <- data.frame(tbl)  
  
# Visualizar  
tbl.df

X1 X2 X3 X4  
1 1.3235 1.4128 1.6744 1.4573  
2 1.4314 1.3592 1.6075 1.4666  
3 1.4284 1.4871 1.4932 1.4324  
4 1.5028 1.6352 1.3841 1.2831  
5 1.5604 1.2735 1.5265 1.4362  
6 1.5955 1.5451 1.3574 1.3281  
7 1.6274 1.5064 1.8366 1.4177  
8 1.4190 1.4303 1.6637 1.6067  
9 1.3884 1.7277 1.5355 1.5176  
10 1.4039 1.6697 1.5089 1.6477  
11 1.4158 1.7667 1.4278 1.5927  
12 1.5821 1.3355 1.5777 1.3908  
13 1.2856 1.4106 1.4447 1.6388  
14 1.4951 1.4036 1.5893 1.6458  
15 1.3589 1.2863 1.5996 1.2497  
16 1.5747 1.5301 1.5171 1.1839  
17 1.3680 1.7269 1.3957 1.5019  
18 1.4163 1.3864 1.3057 1.6210  
19 1.5796 1.4185 1.6541 1.5116  
20 1.7106 1.4412 1.2361 1.3824  
21 1.4371 1.5051 1.3485 1.5670  
22 1.4738 1.5936 1.6583 1.4973  
23 1.5917 1.4333 1.5551 1.5295  
24 1.6399 1.5243 1.5705 1.5563  
25 1.5797 1.3663 1.6240 1.3732

1. Calcule os limites de controle para o gráfico .

* **Exemplo em Python:**

# Média por Linha e Média das Médias  
row\_mean = np.mean(arr, axis=1)  
mean\_X\_bar = np.mean(row\_mean)  
  
# Amplitude por Linha e Média da Amplitude  
row\_range = np.ptp(arr, axis=1)  
mean\_range = np.mean(row\_range)  
  
# Valor de Referência  
A2 = 0.729  
  
# Limites de Controle  
LIC\_X\_bar = mean\_X\_bar - A2 \* mean\_range  
LSC\_X\_bar = mean\_X\_bar + A2 \* mean\_range  
  
# Impressão dos Resultados  
print(f"LIC = {LIC\_X\_bar:.4f}")

LIC = 1.2861

print(f"LC = {mean\_X\_bar:.4f}")

LC = 1.4929

print(f"LSC = {LSC\_X\_bar:.4f}")

LSC = 1.6997

* **Exemplo em R:**

# Média por Linha e Média das Médias  
row\_mean <- apply(tbl.df, 1, mean)  
mean\_X\_bar <- mean(row\_mean)  
  
# Função de Amplitude Total  
f.range <- function(vec) max(vec) - min(vec)  
  
# Amplitude por Linha e Média da Amplitude  
row\_range <- apply(tbl.df, 1, f.range)  
mean\_range <- mean(row\_range)  
  
# Valor de Referência  
A2 <- 0.729  
  
# Limites de Controle  
LIC\_X\_bar = mean\_X\_bar - A2 \* mean\_range  
LSC\_X\_bar = mean\_X\_bar + A2 \* mean\_range  
  
# Impressão dos Resultados  
cat("LIC =", round(LIC\_X\_bar, 4), "\n")

LIC = 1.2861

cat("LC =", round(mean\_X\_bar, 4), "\n")

LC = 1.4929

cat("LSC =", round(LSC\_X\_bar, 4))

LSC = 1.6997

1. Calcule os limites de controle para o gráfico .

* **Exemplo em Python:**

# Valores de Referência  
D3 = 0  
D4 = 2.282  
  
# Limites de Controle  
LIC\_AT = D3 \* mean\_range  
LSC\_AT = D4 \* mean\_range  
  
# Impressão dos Resultados  
print(f"LIC = {LIC\_AT:.4f}")

LIC = 0.0000

print(f"LC = {mean\_range:.4f}")

LC = 0.2837

print(f"LSC = {LSC\_AT:.4f}")

LSC = 0.6473

* **Exemplo em R:**

# Valores de Referência  
D3 <- 0  
D4 <- 2.282  
  
# Limites de Controle  
LIC\_AT = D3 \* mean\_range  
LSC\_AT = D4 \* mean\_range  
  
# Impressão dos Resultados  
cat("LIC =", round(LIC\_AT, 4), "\n")

LIC = 0

cat("LC =", round(mean\_range, 4), "\n")

LC = 0.2837

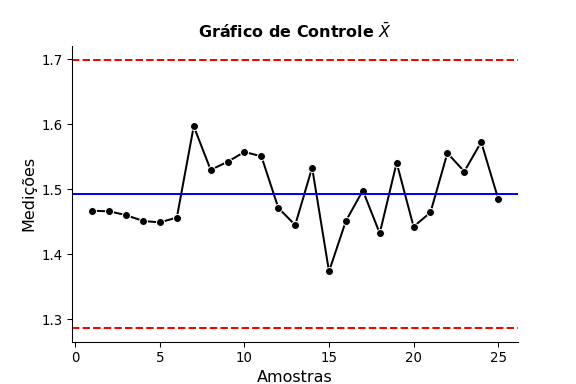
cat("LSC =", round(LSC\_AT, 4))

LSC = 0.6473

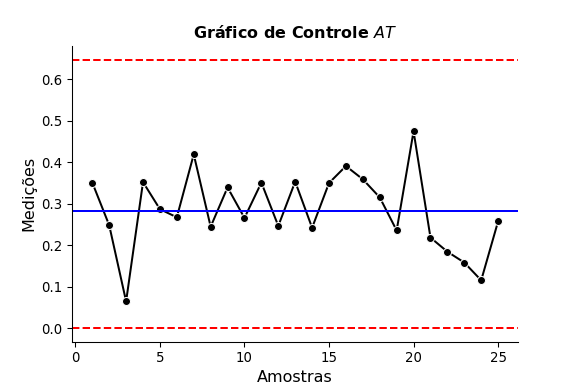
1. Construa os dois gráficos.

* **Exemplo em Python:**

# Configurações de Figura  
fig, ax = plt.subplots(figsize=(6, 4), dpi=600)  
  
# Gráfico de Controle  
sns.lineplot(x=np.arange(1, len(arr) + 1), y=row\_mean, marker="o", color="black", ax=ax)  
ax.axhline(mean\_X\_bar, color="blue", label=f"Média = {mean\_X\_bar:.2f}", linewidth=1.5)  
ax.axhline(LIC\_X\_bar, color="red", linestyle="--", label=f"LIC = {LIC\_X\_bar:.2f}", linewidth=1.5)  
ax.axhline(LSC\_X\_bar, color="red", linestyle="--", label=f"LSC = {LSC\_X\_bar:.2f}", linewidth=1.5)  
  
# Configurações de eixos e títulos  
ax.set\_title(r"Gráfico de Controle $\bar{X}$", fontsize=12, weight="bold")  
ax.set\_xlabel("Amostras", fontsize=12)  
ax.set\_ylabel("Medições", fontsize=12)  
  
# Configuração de Legenda  
ax.legend(loc="lower center", prop={"size": 10}, ncol=3, bbox\_to\_anchor=(0.5, - 0.25), frameon=False)  
  
# Outras configurações  
ax.spines["top"].set\_visible(False)  
ax.spines["right"].set\_visible(False)  
  
# Exibição da Figura  
plt.show()

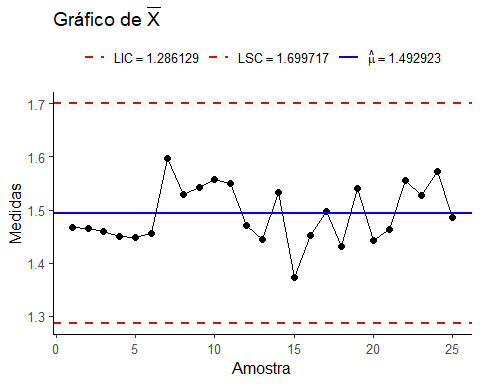


# -----------------------------  
  
# Configurações de Figura  
fig, ax = plt.subplots(figsize=(6, 4), dpi=600)  
  
# Gráfico de Controle  
sns.lineplot(x=np.arange(1, len(arr) + 1), y=row\_range, marker="o", color="black", ax=ax)  
ax.axhline(mean\_range, color="blue", label=r"$\bar{AT}$"+f" = {mean\_range:.2f}", linewidth=1.5)  
ax.axhline(LIC\_AT, color="red", linestyle="--", label=f"LIC = {LIC\_AT:.2f}", linewidth=1.5)  
ax.axhline(LSC\_AT, color="red", linestyle="--", label=f"LSC = {LSC\_AT:.2f}", linewidth=1.5)  
  
# Configurações de eixos e títulos  
ax.set\_title(r"Gráfico de Controle $AT$", fontsize=12, weight="bold")  
ax.set\_xlabel("Amostras", fontsize=12)  
ax.set\_ylabel("Medições", fontsize=12)  
  
# Configuração de Legenda  
ax.legend(loc="lower center", prop={"size": 10}, ncol=3, bbox\_to\_anchor=(0.5, - 0.25), frameon=False)  
  
# Outras configurações  
ax.spines["top"].set\_visible(False)  
ax.spines["right"].set\_visible(False)  
  
# Exibição da Figura  
plt.show()

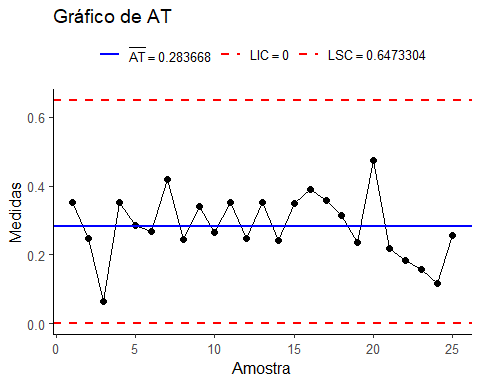


* **Exemplo em R:**

ggplot(data = NULL, aes(x = 1:length(row\_mean), y = row\_mean)) +  
 geom\_point(color = "black", size = 2) +  
 geom\_line(color = "black") +  
 geom\_hline(aes(yintercept = mean\_X\_bar, color = "Média Geral"), linewidth = 1) +  
 geom\_hline(aes(yintercept = LIC\_X\_bar, color = "LIC"), linetype = 2, linewidth = 1) +  
 geom\_hline(aes(yintercept = LSC\_X\_bar, color = "LSC"), linetype = 2, linewidth = 1) +  
 scale\_color\_manual(  
 name = NULL,  
 values = c("LIC" = "red", "LSC" = "red", "Média Geral" = "blue"),  
 labels = c(  
 bquote(LIC == .(LIC\_X\_bar)),   
 bquote(LSC == .(LSC\_X\_bar)),   
 bquote(hat(mu) == .(mean\_X\_bar))  
 )  
 ) +  
 labs(  
 x = "Amostra",  
 y = "Medidas",  
 title = bquote("Gráfico de" ~ bar(X))  
 ) +  
 theme\_classic(base\_size = 12) +  
 theme(legend.position = "top")



# -----------------------------  
  
ggplot(data = NULL, aes(x = 1:length(row\_range), y = row\_range)) +  
 geom\_point(color = "black", size = 2) +  
 geom\_line(color = "black") +  
 geom\_hline(aes(yintercept = mean\_range, color = "Amplitude Média"), linewidth = 1) +  
 geom\_hline(aes(yintercept = LIC\_AT, color = "LIC"), linetype = 2, linewidth = 1) +  
 geom\_hline(aes(yintercept = LSC\_AT, color = "LSC"), linetype = 2, linewidth = 1) +  
 scale\_color\_manual(  
 name = NULL,  
 values = c("LIC" = "red", "LSC" = "red", "Amplitude Média" = "blue"),  
 labels = c(  
 bquote(bar(AT) == .(mean\_range)),  
 bquote(LIC == .(LIC\_AT)),   
 bquote(LSC == .(LSC\_AT))  
 )  
 ) +  
 labs(  
 x = "Amostra",  
 y = "Medidas",  
 title = bquote("Gráfico de" ~ AT)  
 ) +  
 theme\_classic(base\_size = 12) +  
 theme(legend.position = "top")



1. Calcule o desvio padrão de cada subgrupo.

* **Exemplo em Python:**

# Desvio Padrão por Linha  
row\_std = np.std(arr, axis=1)  
  
# Visualizar  
row\_std

array([0.1290724 , 0.09031319, 0.02998611, 0.13160408, 0.11110829,  
 0.11562691, 0.15707416, 0.10726095, 0.12115558, 0.10802753,  
 0.14293566, 0.11013547, 0.12663948, 0.09229503, 0.13625765,  
 0.15594014, 0.14122292, 0.1161874 , 0.0868219 , 0.17182276,  
 0.08118089, 0.07429894, 0.05865364, 0.04222615, 0.11712756])

* **Exemplo em R:**

# Desvio Padrão por Linha  
row\_std = apply(tbl.df, 1, sd)  
  
# Visualizar  
row\_std

[1] 0.14903997 0.10428469 0.03462498 0.15196331 0.12829680 0.13351445  
 [7] 0.18137362 0.12385428 0.13989842 0.12473945 0.16504788 0.12717349  
[13] 0.14623067 0.10657312 0.15733678 0.18006416 0.16307018 0.13416166  
[19] 0.10025330 0.19840384 0.09373962 0.08579303 0.06772739 0.04875855  
[25] 0.13524726

1. Calcule a média dos desvios e os limites de controle do gráfico .

* **Exemplo em Python:**

# Média do Desvio Padrão  
mean\_S = np.mean(row\_std)  
  
# Valores de Referência  
B3 = 0  
B4 = 2.266  
  
# Limites de Controle  
LIC\_S = B3 \* mean\_S  
LSC\_S = B4 \* mean\_S  
  
# Impressão dos Resultados  
print(f"LIC = {LIC\_S:.4f}")

LIC = 0.0000

print(f"LC = {mean\_S:.4f}")

LC = 0.1102

print(f"LSC = {LSC\_S:.4f}")

LSC = 0.2497

* **Exemplo em R:**

# Média do Desvio Padrão  
mean\_S <- mean(row\_std)  
  
# Valores de Referência  
B3 <- 0  
B4 <- 2.266  
  
# Limites de Controle  
LIC\_S = B3 \* mean\_S  
LSC\_S = B4 \* mean\_S  
  
# Impressão dos Resultados  
cat("LIC =", round(LIC\_S, 4), "\n")

LIC = 0

cat("LC =", round(mean\_S, 4), "\n")

LC = 0.1272

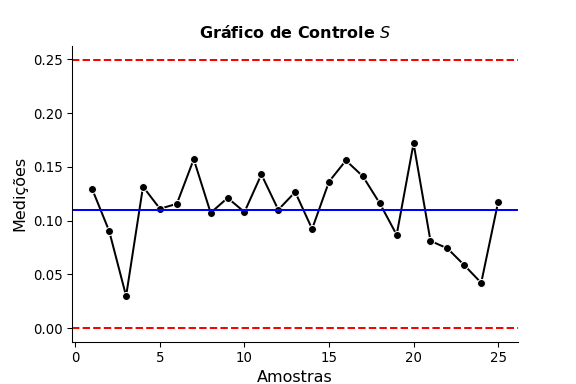
cat("LSC =", round(LSC\_S, 4))

LSC = 0.2883

1. Construa o gráfico do Desvio Padrão.

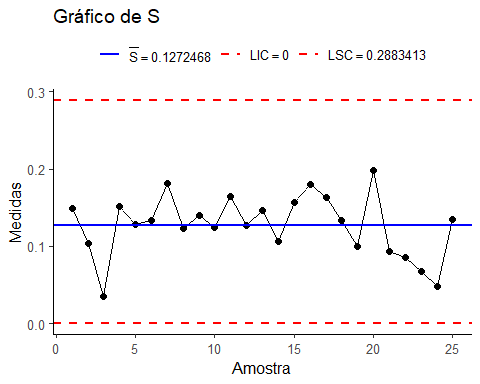
* **Exemplo em Python:**

# Configurações de Figura  
fig, ax = plt.subplots(figsize=(6, 4), dpi=600)  
  
# Gráfico de Controle  
sns.lineplot(x=np.arange(1, len(arr) + 1), y=row\_std, marker="o", color="black", ax=ax)  
ax.axhline(mean\_S, color="blue", label=r"Desvio Padrão Médio ($\bar{S}$)"+f" = {mean\_range:.2f}", linewidth=1.5)  
ax.axhline(LIC\_S, color="red", linestyle="--", label=f"LIC = {LIC\_S:.2f}", linewidth=1.5)  
ax.axhline(LSC\_S, color="red", linestyle="--", label=f"LSC = {LSC\_S:.2f}", linewidth=1.5)  
  
# Configurações de eixos e títulos  
ax.set\_title(r"Gráfico de Controle $S$", fontsize=12, weight="bold")  
ax.set\_xlabel("Amostras", fontsize=12)  
ax.set\_ylabel("Medições", fontsize=12)  
  
# Configuração de Legenda  
ax.legend(loc="lower center", prop={"size": 10}, ncol=3, bbox\_to\_anchor=(0.5, - 0.25), frameon=False)  
  
# Outras configurações  
ax.spines["top"].set\_visible(False)  
ax.spines["right"].set\_visible(False)  
  
# Exibição da Figura  
plt.show()



* **Exemplo em R:**

ggplot(data = NULL, aes(x = 1:length(row\_std), y = row\_std)) +  
 geom\_point(color = "black", size = 2) +  
 geom\_line(color = "black") +  
 geom\_hline(aes(yintercept = mean\_S, color = "Desvio Padrão Médio"), linewidth = 1) +  
 geom\_hline(aes(yintercept = LIC\_S, color = "LIC"), linetype = 2, linewidth = 1) +  
 geom\_hline(aes(yintercept = LSC\_S, color = "LSC"), linetype = 2, linewidth = 1) +  
 scale\_color\_manual(  
 name = NULL,  
 values = c("LIC" = "red", "LSC" = "red", "Desvio Padrão Médio" = "blue"),  
 labels = c(  
 bquote(bar(S) == .(mean\_S)),  
 bquote(LIC == .(LIC\_S)),   
 bquote(LSC == .(LSC\_S))  
 )  
 ) +  
 labs(  
 x = "Amostra",  
 y = "Medidas",  
 title = bquote("Gráfico de" ~ S)  
 ) +  
 theme\_classic(base\_size = 12) +  
 theme(legend.position = "top")



1. Alguma amostra está fora dos limites de controle?

* **Exemplo em Python:**

print("Não. Nenhuma amostra se encontra fora de controle!")

Não. Nenhuma amostra se encontra fora de controle!

* **Exemplo em R:**

print("Não. Nenhuma amostra se encontra fora de controle!")

[1] "Não. Nenhuma amostra se encontra fora de controle!"

1. Há alguma tendência ou padrão preocupante mesmo com os pontos dentro dos limites?

* **Exemplo em Python:**

print("Aparentemente não! As amostras estão dispersas de forma aleatória, sem padrões visíveis.")

Aparentemente não! As amostras estão dispersas de forma aleatória, sem padrões visíveis.

* **Exemplo em R:**

print("Aparentemente não! As amostras estão dispersas de forma aleatória, sem padrões visíveis.")

[1] "Aparentemente não! As amostras estão dispersas de forma aleatória, sem padrões visíveis."

1. O processo pode ser considerado sob controle estatístico?

* **Exemplo em Python:**

print("Sim. As amostras estão dispersas de forma aleatória, sem padrões visíveis e todas dentro dos Limites de Controle.")

Sim. As amostras estão dispersas de forma aleatória, sem padrões visíveis e todas dentro dos Limites de Controle.

* **Exemplo em R:**

print("Sim. As amostras estão dispersas de forma aleatória, sem padrões visíveis e todas dentro dos Limites de Controle.")

[1] "Sim. As amostras estão dispersas de forma aleatória, sem padrões visíveis e todas dentro dos Limites de Controle."

1. Compare o gráficos e com o gráfico e .
   1. Qual parece mais sensível às variações nos dados?
   2. Qual seria mais indicado para subgrupos maiores que 10?
   3. Em que situações o gráfico de variância seria mais apropriado?

* **Exemplo em Python:**

# Valor de Referência  
A2 = 0.729  
  
# Limites de Controle  
LIC\_X\_bar = mean\_X\_bar - A2 \* mean\_range  
LSC\_X\_bar = mean\_X\_bar + A2 \* mean\_range  
  
# Impressão dos Resultados  
print(f"LIC = {LIC\_X\_bar:.4f}")

LIC = 1.2861

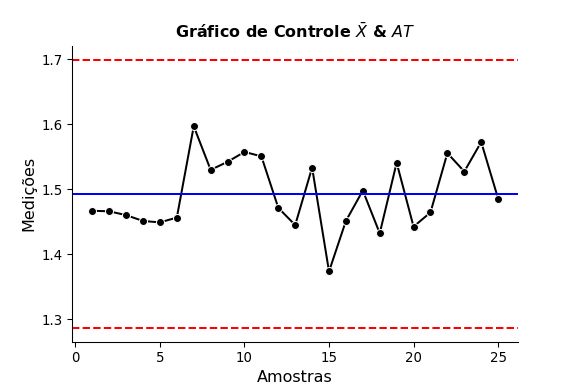
print(f"LC = {mean\_X\_bar:.4f}")

LC = 1.4929

print(f"LSC = {LSC\_X\_bar:.4f}")

LSC = 1.6997

# Configurações de Figura  
fig, ax = plt.subplots(figsize=(6, 4), dpi=600)  
  
# Gráfico de Controle  
sns.lineplot(x=np.arange(1, len(arr) + 1), y=row\_mean, marker="o", color="black", ax=ax)  
ax.axhline(mean\_X\_bar, color="blue", label=f"Média = {mean\_X\_bar:.2f}", linewidth=1.5)  
ax.axhline(LIC\_X\_bar, color="red", linestyle="--", label=f"LIC = {LIC\_X\_bar:.2f}", linewidth=1.5)  
ax.axhline(LSC\_X\_bar, color="red", linestyle="--", label=f"LSC = {LSC\_X\_bar:.2f}", linewidth=1.5)  
  
# Configurações de eixos e títulos  
ax.set\_title(r"Gráfico de Controle $\bar{X}$ & $AT$", fontsize=12, weight="bold")  
ax.set\_xlabel("Amostras", fontsize=12)  
ax.set\_ylabel("Medições", fontsize=12)  
  
# Configuração de Legenda  
ax.legend(loc="lower center", prop={"size": 10}, ncol=3, bbox\_to\_anchor=(0.5, - 0.25), frameon=False)  
  
# Outras configurações  
ax.spines["top"].set\_visible(False)  
ax.spines["right"].set\_visible(False)  
  
# Exibição da Figura  
plt.show()



# -----------------------------  
  
# Valor de Referência  
A3 = 1.628  
  
# Limites de Controle  
LIC\_X\_bar = mean\_X\_bar - A3 \* mean\_S  
LSC\_X\_bar = mean\_X\_bar + A3 \* mean\_S  
  
# Impressão dos Resultados  
print(f"LIC = {LIC\_X\_bar:.4f}")

LIC = 1.3135

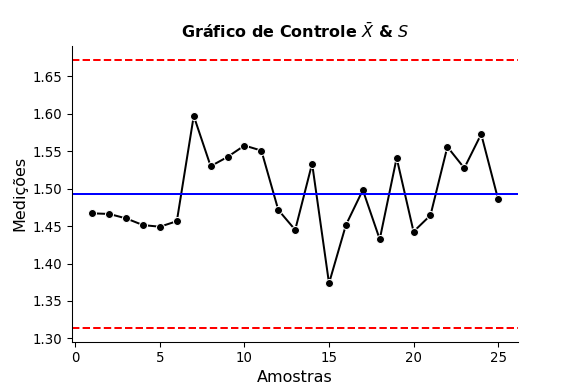
print(f"LC = {mean\_X\_bar:.4f}")

LC = 1.4929

print(f"LSC = {LSC\_X\_bar:.4f}")

LSC = 1.6723

# Configurações de Figura  
fig, ax = plt.subplots(figsize=(6, 4), dpi=600)  
  
# Gráfico de Controle  
sns.lineplot(x=np.arange(1, len(arr) + 1), y=row\_mean, marker="o", color="black", ax=ax)  
ax.axhline(mean\_X\_bar, color="blue", label=f"Média = {mean\_X\_bar:.2f}", linewidth=1.5)  
ax.axhline(LIC\_X\_bar, color="red", linestyle="--", label=f"LIC = {LIC\_X\_bar:.2f}", linewidth=1.5)  
ax.axhline(LSC\_X\_bar, color="red", linestyle="--", label=f"LSC = {LSC\_X\_bar:.2f}", linewidth=1.5)  
  
# Configurações de eixos e títulos  
ax.set\_title(r"Gráfico de Controle $\bar{X}$ & $S$", fontsize=12, weight="bold")  
ax.set\_xlabel("Amostras", fontsize=12)  
ax.set\_ylabel("Medições", fontsize=12)  
  
# Configuração de Legenda  
ax.legend(loc="lower center", prop={"size": 10}, ncol=3, bbox\_to\_anchor=(0.5, - 0.25), frameon=False)  
  
# Outras configurações  
ax.spines["top"].set\_visible(False)  
ax.spines["right"].set\_visible(False)  
  
# Exibição da Figura  
plt.show()



# ----------------------------------------------------------------------  
# a. Qual parece mais sensível às variações nos dados?  
# ----------------------------------------------------------------------  
  
print("O gráfico de controle com base no desvio padrão (S) apresentou limites ligeiramente maiores que o gráfico de controle com base na amplitude (AT).")

O gráfico de controle com base no desvio padrão (S) apresentou limites ligeiramente maiores que o gráfico de controle com base na amplitude (AT).

# ----------------------------------------------------------------------  
# b. Qual seria mais indicado para subgrupos maiores que 10?  
# ----------------------------------------------------------------------  
  
print("O mais indicado para amostras maiores que 10 é o gráfico de controle com base no desvio padrão (S).")

O mais indicado para amostras maiores que 10 é o gráfico de controle com base no desvio padrão (S).

# ----------------------------------------------------------------------  
# c. Em que situações o gráfico de variância seria mais apropriado?  
# ----------------------------------------------------------------------  
  
print("Sob o mesmo raciocínio do item anterior, o gráfico de controle com base no desvio padrão é mais indicado para amostras maiores que 10.")

Sob o mesmo raciocínio do item anterior, o gráfico de controle com base no desvio padrão é mais indicado para amostras maiores que 10.

* **Exemplo em R:**

# Valor de Referência  
A2 <- 0.729  
  
# Limites de Controle  
LIC\_X\_bar <- mean\_X\_bar - A2 \* mean\_range  
LSC\_X\_bar <- mean\_X\_bar + A2 \* mean\_range  
  
# Impressão dos Resultados  
cat("LIC =", round(LIC\_X\_bar, 4), "\n")

LIC = 1.2861

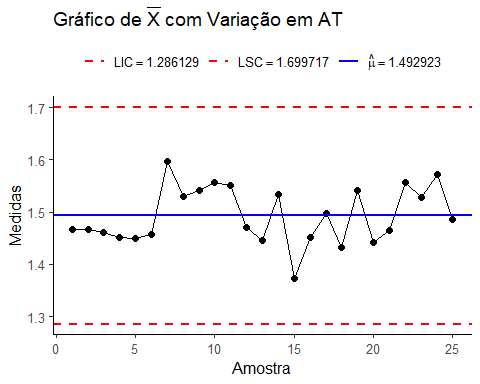
cat("LC =", round(mean\_X\_bar, 4), "\n")

LC = 1.4929

cat("LSC =", round(LSC\_X\_bar, 4))

LSC = 1.6997

ggplot(data = NULL, aes(x = 1:length(row\_mean), y = row\_mean)) +  
 geom\_point(color = "black", size = 2) +  
 geom\_line(color = "black") +  
 geom\_hline(aes(yintercept = mean\_X\_bar, color = "Média Geral"), linewidth = 1) +  
 geom\_hline(aes(yintercept = LIC\_X\_bar, color = "LIC"), linetype = 2, linewidth = 1) +  
 geom\_hline(aes(yintercept = LSC\_X\_bar, color = "LSC"), linetype = 2, linewidth = 1) +  
 scale\_color\_manual(  
 name = NULL,  
 values = c("LIC" = "red", "LSC" = "red", "Média Geral" = "blue"),  
 labels = c(  
 bquote(LIC == .(LIC\_X\_bar)),   
 bquote(LSC == .(LSC\_X\_bar)),   
 bquote(hat(mu) == .(mean\_X\_bar))  
 )  
 ) +  
 labs(  
 x = "Amostra",  
 y = "Medidas",  
 title = bquote("Gráfico de" ~ bar(X) ~ "com Variação em" ~ AT)  
 ) +  
 theme\_classic(base\_size = 12) +  
 theme(legend.position = "top")



# -----------------------------  
  
# Valor de Referência  
A3 <- 1.628  
  
# Limites de Controle  
LIC\_X\_bar <- mean\_X\_bar - A3 \* mean\_S  
LSC\_X\_bar <- mean\_X\_bar + A3 \* mean\_S  
  
# Impressão dos Resultados  
cat("LIC =", round(LIC\_X\_bar, 4), "\n")

LIC = 1.2858

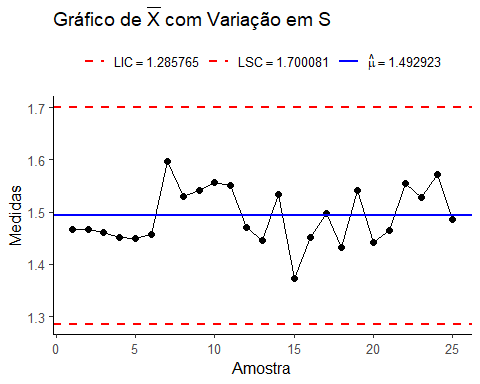
cat("LC =", round(mean\_X\_bar, 4), "\n")

LC = 1.4929

cat("LSC =", round(LSC\_X\_bar, 4))

LSC = 1.7001

ggplot(data = NULL, aes(x = 1:length(row\_mean), y = row\_mean)) +  
 geom\_point(color = "black", size = 2) +  
 geom\_line(color = "black") +  
 geom\_hline(aes(yintercept = mean\_X\_bar, color = "Média Geral"), linewidth = 1) +  
 geom\_hline(aes(yintercept = LIC\_X\_bar, color = "LIC"), linetype = 2, linewidth = 1) +  
 geom\_hline(aes(yintercept = LSC\_X\_bar, color = "LSC"), linetype = 2, linewidth = 1) +  
 scale\_color\_manual(  
 name = NULL,  
 values = c("LIC" = "red", "LSC" = "red", "Média Geral" = "blue"),  
 labels = c(  
 bquote(LIC == .(LIC\_X\_bar)),   
 bquote(LSC == .(LSC\_X\_bar)),   
 bquote(hat(mu) == .(mean\_X\_bar))  
 )  
 ) +  
 labs(  
 x = "Amostra",  
 y = "Medidas",  
 title = bquote("Gráfico de" ~ bar(X) ~ "com Variação em" ~ S)  
 ) +  
 theme\_classic(base\_size = 12) +  
 theme(legend.position = "top")



# ----------------------------------------------------------------------  
# a. Qual parece mais sensível às variações nos dados?  
# ----------------------------------------------------------------------  
  
print("O gráfico de controle com base no desvio padrão (S) apresentou limites ligeiramente maiores que o gráfico de controle com base na amplitude (AT).")

[1] "O gráfico de controle com base no desvio padrão (S) apresentou limites ligeiramente maiores que o gráfico de controle com base na amplitude (AT)."

# ----------------------------------------------------------------------  
# b. Qual seria mais indicado para subgrupos maiores que 10?  
# ----------------------------------------------------------------------  
  
print("O mais indicado para amostras maiores que 10 é o gráfico de controle com base no desvio padrão (S).")

[1] "O mais indicado para amostras maiores que 10 é o gráfico de controle com base no desvio padrão (S)."

# ----------------------------------------------------------------------  
# c. Em que situações o gráfico de variância seria mais apropriado?  
# ----------------------------------------------------------------------  
  
print("Sob o mesmo raciocínio do item anterior, o gráfico de controle com base no desvio padrão é mais indicado para amostras maiores que 10.")

[1] "Sob o mesmo raciocínio do item anterior, o gráfico de controle com base no desvio padrão é mais indicado para amostras maiores que 10."

### 3.4.4 Considerações Finais

* A escolha entre amplitude ou desvio padrão depende do tamanho da amostra. ✅
* Gráficos de variância são mais sensíveis, mas menos utilizados. ✅
* Gráficos e são os mais comuns na prática industrial. ✅

# 4. Listas e Exercícios

## 4.1 Lista I

1. Simule (no R **e** Python) um conjunto de dados com três turnos de produção e números de defeitos.
   1. Faça um boxplot para comparar os defeitos entre turnos.
   2. Comente se a estratificação revela alguma diferença relevante.
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. 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 |

* 1. Quais problemas devem ser atacados primeiro?
  2. Qual o percentual acumaludo dos dois problemas mais frquentes?

1. Simule 200 observações com e .
   1. Crie um histograma.
   2. Defina limites de especificaçãp mais estreitos: e .
2. Reúna-se com seu grupo faça o seguinte:
   * Colete um conjunto de dados reais (ex.: tempo para executar uma tarefa simples);
   * Classifique os dados usando estratificação (ex.: por turno, grupo, dia, etc.);
   * Construa um histograma, gráfico de Pareto e, se possível um diagrama de Ishikawa para o problema observado;
   * Apresente os resultados com uma breve conclusão.

# 5. Lista II

1. Utilize os vetores abaixo e construa o diagrama de dispersão. e .
   1. Descreva o tipo de relação entre as variáveis.
   2. Adicione uma reta de tendência.
2. Geração de dados com correlação negativa.
   1. Gere dois vetores de 30 elementos com correlação negativa.
   2. Construa o gráfico de dispersão.
   3. Calcule a correlação de Pearson.
3. Dados reais - mtcars. Utilize o conjunto de dados mtcars.
   1. Há relação entre mpg (milhas por galão) e wt (peso)?
   2. Faça o gráfico e interprete-o.
   3. Calcule a correlação de maneira adequada.
   4. A relação é positiva ou negativa?
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

“Mermaid.js”. s.d. <https://csilv7.atlassian.net/wiki/spaces/~71202019d1c3fc8d434cb59c0a9a68051b55ca/overview?homepageId=65794>.

Montgomery, Douglas C. 2013. *Introdução ao Controle Estatístico da Qualidade*. Rio de Janeiro: LTC.

Shewhart, W. A. 1926. “Quality Control Charts”. *The Bell System Technical Journal* 5 (4): 593–603. <https://doi.org/10.1002/j.1538-7305.1926.tb00125.x>.