# Estatística e Experimentação no Ambiente R

## 1. Introdução ao R

### 1.1 O que é o R?

O R é uma linguagem de programação e ambiente de computação estatística amplamente utilizado para análise de dados e gráficos. Ele é uma ferramenta de código aberto que fornece uma variedade de funções estatísticas e recursos para manipular, visualizar e modelar dados. O R é apreciado por sua flexibilidade e extensibilidade, permitindo aos usuários desenvolver seus próprios pacotes e funções. É uma escolha popular entre estatísticos, cientistas de dados e pesquisadores de diversas áreas devido à sua capacidade de lidar com uma ampla gama de tarefas analíticas e sua comunidade ativa de desenvolvedores e usuários. Durante esta aula, vamos explorar os conceitos fundamentais do R e como utilizá-lo para realizar análises estatísticas e experimentações.

### 1.2 Instalação e configuração básica.

Antes de começar a utilizar o R, é necessário instalar o software e configurar o ambiente. A instalação do R é simples e está disponível para várias plataformas (Windows, macOS, Linux). Após a instalação, é comum utilizar um ambiente de desenvolvimento integrado (IDE) como o RStudio, que facilita a interação com o R. A configuração básica envolve a definição de diretórios de trabalho, opções de exibição e a instalação de pacotes adicionais. Com um ambiente de trabalho bem configurado, você estará pronto para começar a escrever código em R, realizar análises de dados e executar experimentações estatísticas.

### 1.3 Interface do RStudio.

O RStudio é uma das interfaces mais populares para o ambiente R. Ele oferece um ambiente integrado que simplifica o uso do R, tornando-o mais amigável e produtivo. A interface do RStudio é composta por várias janelas e painéis:

Console R: O console é onde você digita e executa comandos R. Você pode ver a saída imediata dos comandos aqui.  
  
Editor de Script: Permite que você escreva, edite e salve scripts R. É útil para criar código reutilizável e documentado.  
  
Ambiente e Histórico: Mostra uma lista de objetos carregados na sessão atual e o histórico de comandos usados.  
  
Painel de Plots: Exibe gráficos e visualizações gerados a partir dos comandos R.  
  
Painel de Pacotes: Gerencia a instalação e carregamento de pacotes R adicionais.  
  
Painel de Arquivos e Ajuda: Permite navegar em diretórios, acessar ajuda online e visualizar arquivos.  
  
Configurações Globais: Aqui, você pode personalizar configurações, temas e atalhos do RStudio.

A interface do RStudio é altamente personalizável para atender às suas necessidades e fluxo de trabalho. Ele oferece recursos avançados, como depuração de código, controle de versão integrado e suporte a projetos. Dominar a interface do RStudio é essencial para se tornar produtivo na análise de dados e programação em R.

### 1.4 Instalação de pacotes.

Os pacotes são extensões de funcionalidades que podem ser adicionadas ao ambiente R. Eles contêm funções, conjuntos de dados e documentação relacionada a tarefas específicas. Para instalar um pacote em R, você pode usar a função install.packages(). Veja como funciona:

Seleção do Pacote: Primeiro, você precisa decidir qual pacote deseja instalar. Escolha pacotes que atendam às necessidades da sua análise de dados ou experimentação.  
  
Instalação do Pacote: Utilize o comando install.packages("nome\_do\_pacote"), substituindo "nome\_do\_pacote" pelo nome do pacote desejado. Por exemplo, para instalar o pacote "ggplot2", você digitaria install.packages("ggplot2").  
  
Seleção do Repositório: O R oferece vários repositórios de pacotes. Você pode selecionar um repositório de onde deseja baixar o pacote ou simplesmente pressionar Enter para usar o repositório padrão, o CRAN (Comprehensive R Archive Network).  
  
Aprovação de Dependências: O RStudio geralmente perguntará se você deseja instalar pacotes dependentes necessários para o funcionamento do pacote selecionado. Confirme a instalação dessas dependências, se solicitado.  
  
Conclusão: Após a instalação bem-sucedida, o pacote estará pronto para ser carregado e usado em sua sessão R. Você pode carregá-lo usando a função library().

A instalação de pacotes é fundamental, pois permite estender as capacidades do R para atender a necessidades específicas. Essa flexibilidade é uma das razões pelas quais o R é tão poderoso para análise de dados e estatísticas.

Vamos instalar alguns pacotes

install.packages("tidyverse")  
install.packages("agricolae")  
install.packages("ExpDes.pt")  
install.packages("nortest")  
install.packages("readxl")

Vamos carregar o pacotes ao ambiente

library(tidyverse)  
library(agricolae)  
library(ExpDes.pt)  
library(nortest)  
library(readxl)

## 2. Conceitos Básicos do R

### 2.1 Tipos de objetos (vetores, matrizes, listas, data frames).

R oferece diversos tipos de objetos para armazenar e manipular dados. Aqui estão alguns dos tipos mais comuns:

#### **Vetores**:

Os vetores são sequências unidimensionais de elementos do mesmo tipo (por exemplo, números inteiros, números reais, caracteres). Eles podem ser criados usando a função c() (combine) e são a estrutura de dados fundamental em R.

OBS: a atribuição de valores a um objeto no R será sempre feita pelo operador de atribuição <-.

x <- 1  
is.vector(x)  
length(x)

Observe que um escalar para o R é um vetor com dimensão de elementos.

y <- c(pi, 2, 5)  
y

Vamos dar um exemplo de vetor com caracteres:

nome <- c("Arthur", "Beatriz", "Carlos", "Davi")  
nome

Agora observe o retorno das funções

is.character(nome)  
mode(nome)  
class(nome)

#### **Matrizes**:

As matrizes são estruturas bidimensionais que consistem em elementos organizados em linhas e colunas. Todos os elementos de uma matriz devem ter o mesmo tipo de dado. Você pode criar matrizes com a função matrix().

fornecido o vetor x, contendo 9 elementos, podemos criar a matriz A, com 3 linhas e 3 colunas. O preenchimento da matriz é controlado pelo argumento byrow da função matrix(), sendo por linhas (byrow = TRUE) ou por colunas (byrow = FALSE).

x <- c(1,2,3,4,5,6,7,8,10)  
A <- matrix(x, ncol=3, byrow = FALSE)  
A

A determinante de A é dada por:

det(A)

Podemos trasnpor a matriz A com a função t().

t(A)

Podemos inverter a matriz A com a função solve().

solve(A)

Podemos multiplicar a matriz A pela sua inversa e termos a matriz identidade.

round(solve(A) %\*% A, 2)

OBS: A função round() foi utilizada para aproximação das casas decimais dos valores.

#### **Listas**:

Listas são estruturas de dados que podem conter elementos de tipos diferentes, incluindo vetores, matrizes, listas e objetos individuais. Listas são criadas com a função list().

Vamos criar uma lista com os vetores nome e x e a matriz A.

minha\_lista <- list(  
 nomes = nome,  
 elementos = x,  
 matrix = A  
)  
minha\_lista

#### **Data Frames**:

Os data frames são semelhantes a matrizes, mas têm uma flexibilidade adicional. Eles são estruturas bidimensionais onde as colunas podem conter diferentes tipos de dados. Data frames são frequentemente usados para representar dados tabulares, como planilhas. Você pode criar data frames usando a função data.frame().

Esses tipos de objetos são fundamentais para armazenar e organizar dados em R. Eles desempenham um papel essencial na manipulação e análise de dados, permitindo que você realize operações estatísticas, visualizações e experimentações de maneira eficaz. Cada tipo de objeto tem suas próprias propriedades e métodos específicos para operações e análises.

Vamos crias um Data frame com nome dos alunos, data de nascimentos, e notas das provas 1 e 2.

meu\_df <- data.frame(  
 nome = nome,  
 data\_nascimento = c("25/05/2003","26/08/2006","27/07/2003","28/08/2003"),  
 nota\_p1 = c(5,3,8,10),  
 nota\_p2 = c(2,1,3,6)  
)  
meu\_df

### 2.2 Operações básicas com objetos.

O acesso aos elementos da lista pode ser realizado por meio da indexação de sua posição entre colchetes, declarado após o nome do objeto. Tal indexação funciona para todos os elementos até aqui criados.

O terceiro nome é:

nome[3]

O primeiro e o quarto nomes são:

nome[c(1,4)]

A terceira coluna de A é:

A[,3]

A primeira e terceira colunas de A são:

A[,c(1,3)]

As segunda e terceira linhas de A são:

A[c(2,3),]

O primeiro elemento da lista é:

minha\_lista[1]

Retirar a segunda coluna da matriz da lista:

minha\_lista[[3]][,2]

Agora, podemos aproveitar o operador de pipe (%>%) para efetuar alterações no Data Frame que construímos. Esse operador é uma implementação do pacote magrittr, que é carregado no ambiente R quando utilizamos o meta-pacote tidyverse. A seguir, serão apresentadas algumas funções para a manipulação de Data Frames. A grande vantagem desse método de manipulação é que o objeto original a ser trabalhado (meu\_df) é sempre preservado, tornando-o ideal para transformações sem destruir os dados originais.

Vamos criar a média das provas, nota final dos alunos

meu\_df %>%   
 mutate(nota\_final = 0.5\* nota\_p1 + 0.5\*nota\_p2)

Vamos criar a média das provas, nota final dos alunos, e ordenar da maior para a menor média

meu\_df %>%   
 mutate(nota\_final = 0.5\* nota\_p1 + 0.5\*nota\_p2) %>%   
 arrange(desc(nota\_final))

Vamos criar a média das provas, nota final dos alunos, e ordenar da maior para a menor média, e selecionar somente as colunas nome e nota\_final.

meu\_df %>%   
 mutate(nota\_final = 0.5\* nota\_p1 + 0.5\*nota\_p2) %>%   
 arrange(desc(nota\_final)) %>%   
 select(nome, nota\_final)

Vamos criar a média das provas, nota final dos alunos, e ordenar da maior para a menor média, e selecionar somente as colunas nome e nota\_final e adicionar a situação do aluno, se a nota\_final por maior ou igual a 5 ele estará aprovado, caso contrário, estará reprovado.

meu\_df %>%   
 mutate(nota\_final = 0.5\* nota\_p1 + 0.5\*nota\_p2) %>%   
 arrange(desc(nota\_final)) %>%   
 select(nome, nota\_final) %>%   
 mutate(situação = ifelse(nota\_final >=5, "Aprovado","Reprovado"))

Qual a média geral das notas?:

meu\_df %>%   
 mutate(nota\_final = 0.5\* nota\_p1 + 0.5\*nota\_p2) %>%   
 arrange(desc(nota\_final)) %>%   
 select(nome, nota\_final) %>%   
 mutate(situação = ifelse(nota\_final >=5, "Aprovado","Reprovado")) %>%   
 summarise(media\_geral = mean(nota\_final))

Qual a média geral das notas por situação?:

meu\_df %>%   
 mutate(nota\_final = 0.5\* nota\_p1 + 0.5\*nota\_p2) %>%   
 arrange(desc(nota\_final)) %>%   
 select(nome, nota\_final) %>%   
 mutate(situação = ifelse(nota\_final >=5, "Aprovado","Reprovado")) %>%   
 group\_by(situação) %>%   
 summarise(media\_geral = mean(nota\_final))

Quais os desvios de nota\_final em relação à média geral?:

meu\_df %>%   
 mutate(nota\_final = 0.5\* nota\_p1 + 0.5\*nota\_p2) %>%   
 arrange(desc(nota\_final)) %>%   
 select(nome, nota\_final) %>%   
 mutate(situação = ifelse(nota\_final >=5, "Aprovado","Reprovado")) %>%   
 mutate(desvio = nota\_final - mean(nota\_final))

Quais os desvios de nota\_final em relação à média de nota\_final por grupo de situação?:

meu\_df %>%   
 mutate(nota\_final = 0.5\* nota\_p1 + 0.5\*nota\_p2) %>%   
 arrange(desc(nota\_final)) %>%   
 select(nome, nota\_final) %>%   
 mutate(situação = ifelse(nota\_final >=5, "Aprovado","Reprovado")) %>%   
 group\_by(situação) %>%   
 mutate(desvio = nota\_final - mean(nota\_final))

### 2.3 Funções e operadores.

#### Funções Básicas:

Operações Matemáticas:

Soma:

Subtração: 5 - 1

Multiplicação: 4 \* 7

Divisão: 11 / 2

Divisão inteira: 11 %/% 3

Resto da divisão inteira: 11 %% 3

#### Funções Matemáticas:

Raiz quadrada: sqrt(25)

Potência: 2^3 ou 2\*\*

Valor absoluto: abs(-10)

Funções trigonométricas seno: sin(30\*pi/180)

Função arco-seno: asin(0.5)\*180/pi

Função exponencial: exp(1)

Função logarítmo natual: log(2.718282)

Função log na base 10: log10(100)

Função log definição da base: log(1024,2)

#### Operadores Lógicos e relacionais:

Igualdade:

> 3 == 3 (verdadeiro)  
  
> 3 == 4 (falso)  
  
> 3 != 4 (verdadeiro)

Maior e Menor (ou igual):

> 5 > 3 (verdadeiro)  
  
 > 2 <= 1 (falso)

Operadores Lógicos Compostos:

> TRUE && FALSE (AND lógico, falso)  
  
> TRUE || FALSE (OR lógico, verdadeiro)

#### Criando uma função no R

Como exemplo vamos criar uma funão que calcule o coeficiente de variação que será dado pela relação percentual do desvio-padrão amostral e a média amostral:

Criando a função no R.

meu\_cv <- function(x){  
 media <- mean(x)  
 desv\_pad <- sd(x)  
 valor\_cv <- 100\*desv\_pad/media  
 return(valor\_cv)  
}

Agora podemos chamar a função.

y <- c(8,5.5,3.5,2)  
meu\_cv(y)

## 3. Manipulação de Dados no R

### 3.1 Importação e exportação de dados (leitura de CSV, Excel, etc.).

Suponha que você tenha um arquivo de texto chamado “dados.txt” com o seguinte conteúdo:

Nome Idade João 30 Maria 28 Pedro 35 Ana 25

Você pode usar a função read.table() para importar esses dados em R da seguinte forma:

# Especifique o caminho para o arquivo de texto  
caminho\_do\_arquivo <- "caminho/para/seu/arquivo/dados.txt"  
  
# Use a função read.table() para importar os dados  
seu\_dataframe <- read.table(caminho\_do\_arquivo, header = TRUE)

Para importar dados de um arquivo CSV (Comma-Separated Values), você pode usar a função read.csv():

meu\_dataframe <- read.csv("seu\_arquivo.csv")

Para importar dados de uma planilha Excel, você pode usar a biblioteca readxl e a função read\_excel(), o argumento sheet especifica o nome da planilha que se quer importar:

library(readxl)  
meu\_dataframe <- read\_excel("seu\_arquivo.xlsx",  
 sheet = "Plan1")

Leitura de Outros Formatos:

O R suporta a importação de dados de muitos outros formatos, como arquivos JSON, SQL, HDF5, e mais. Você pode usar pacotes específicos para cada formato.

### 3.1 Limpeza e transformação de dados.

Vamos simular um conjunto de dados:

meus\_dados <- data.frame(  
 FCO2 = sort(rnorm(100, 4, 0.95)),  
 Temp = rnorm(100, 24,2),  
 umidade = sort(rnorm(100, 15,5))  
)

Padronização dos nomes das variáveis:

meus\_dados %>%   
 janitor::clean\_names() %>%   
 head()

Renomeando a variável temp para ts e a variável umidade para us.

meus\_dados\_pad <- meus\_dados %>%   
 janitor::clean\_names() %>%   
 rename(ts = temp, us = umidade)

Valores perdidos poderão ser descartados da base de dados pela função drop\_na do pacote tidyr.

meus\_dados\_pad %>%   
 drop\_na() %>%   
 head()

Vamos transformar os dados de fco2, passando-os para a escala logarítmica:

meus\_dados\_pad %>%   
 drop\_na() %>%   
 mutate(fco2\_log = log10(fco2)) %>%   
 head()

Criar as categorais para emissão em “baixa”, “média” e “alta”, por meio da função cut. Mude a posição das colunas fco2\_log e categoria para logo depois da coluna fco2.

meus\_dados\_pad <- meus\_dados\_pad %>%   
 drop\_na() %>%   
 mutate(fco2\_log = log10(fco2),  
 categoria = cut(fco2,  
 breaks = 3,  
 labels = c("baixa","média","alta")  
 )  
 ) %>%   
 relocate(fco2, fco2\_log, categoria)   
meus\_dados\_pad %>% head

### 3.3 Resumo estatístico dos dados.

Vamos agora criar um resumo estatístico simples para os dados

Emissão de CO2 do solo

meus\_dados\_pad %>%   
 pull(fco2) %>%   
 summary()

## 4. Visualização de Dados

### 4.1 Gráficos básicos (histogramas, boxplot, gráficos de barras, gráficos de dispersão).

Histograma da emissão de CO2 do solo

meus\_dados\_pad %>%   
 ggplot(aes(x=fco2)) +   
 geom\_histogram(bins = 12, color="black",fill="orange")

Boxplot da emissão de CO2 do solo por categoria

meus\_dados\_pad %>%   
 ggplot(aes(x= categoria, y=fco2, fill = categoria)) +   
 geom\_boxplot()

Gráfico de colunas da emissão de CO2 do solo

meus\_dados\_pad %>%   
 group\_by(categoria) %>%   
 summarise(media = mean(fco2)) %>%   
 ggplot(aes(x=categoria, y=media, fill=categoria)) +  
 geom\_col()

Gráfico de barras da emissão de CO2 do solo

meus\_dados\_pad %>%   
 group\_by(categoria) %>%   
 summarise(media = mean(fco2)) %>%   
 ggplot(aes(x=categoria, y=media, fill=categoria)) +  
 geom\_col() +  
 coord\_flip()

Plotar os dados de emissão de co2 em função da temperatura do solo, mapeando as categorias

meus\_dados\_pad %>%   
 ggplot(aes(x=ts, y=fco2, color=categoria)) +  
 geom\_point()

Plotar os dados de emissão de co2 em função da umidade do solo, mapeando as categorias

meus\_dados\_pad %>%   
 ggplot(aes(x=us, y=fco2, color=categoria)) +  
 geom\_point()

### 4.2 Personalização de gráficos.

meus\_dados\_pad %>%   
 ggplot(aes(x=us, y=fco2, color=categoria)) +  
 geom\_point() +  
 theme\_bw() +  
 labs(title="Gráfico de Dispersão",  
 x= "Umidade do Solo",  
 y=expression(paste("Emissão de ", CO[2]," do solo")),  
 color = "Categorais de FCO2:"  
 ) +   
 theme(legend.position = "bottom",  
 plot.title = element\_text(hjust = 0.5))

### 4.3 Verificação de normalidade (teste de Shapiro-Wilk, gráfico Q-Q).

meus\_dados\_pad %>%   
 ggplot(aes(sample = fco2)) +   
 stat\_qq(color="blue") +  
 stat\_qq\_line(color = "red")

Teste de Normalidade

library(nortest)  
meus\_dados\_pad %>% pull(fco2) %>% shapiro.test()  
meus\_dados\_pad %>% pull(fco2) %>% lillie.test()  
meus\_dados\_pad %>% pull(fco2) %>% ad.test()  
meus\_dados\_pad %>% pull(fco2) %>% cvm.test()

## 5. Estatística Descritiva

### 5.1 Medidas de tendência central (média, mediana, moda).

Vamos criar uma função contendo as medidas de tendência central.

medidas\_tendencia\_central <- function(x){  
 xb <- mean(x)  
 md <- median(x)  
 uniq\_x <- unique(x)  
 moda <- uniq\_x[which.max(  
 tabulate(  
 match(x, uniq\_x)))]  
 return( c(  
 Média = xb,  
 Mediana = md,  
 Moda = moda  
 ))  
}  
meus\_dados\_pad %>% pull(fco2) %>% medidas\_tendencia\_central()

### 5.2 Medidas de dispersão (desvio padrão, variância, intervalos).

Semelhante ao exemplo anterior

medidas\_dispersao <- function(x){  
 xb <- mean(x)  
 maior <- max(x)  
 menor <- min(x)  
 variancia <-var(x)  
 desvpad <- sd(x)  
 erropad <- sd(x)/length(x)^.5  
 cv <- meu\_cv(x)  
   
 return( c(  
 Média = xb,  
 Mínimo = menor,  
 Máximo = maior,  
 Variância = variancia,  
 `Desvio Padrão` = desvpad,  
 `Erro Padrão` = erropad,  
 `Coef. Variação` = cv  
 ))  
}  
meus\_dados\_pad %>% pull(fco2) %>% medidas\_dispersao()

## 6. Ajuste de Regressão linear simples.

A regressão linear simples é uma técnica estatística que modela a relação entre uma variável independente (preditora) e uma variável dependente (alvo) por meio de uma equação linear.

O modelo de regressão linear simples pode ser expresso como:

é a variável dependente (alvo).  
 é a variável independente (preditora).  
 é o intercepto.  
 é o coeficiente de regressão.  
 é o erro residual.

Através da análise de regressão, você encontra os melhores valores para β0 e β1 que minimizam a soma dos quadrados dos erros residuais. Vamos fazer um ajuste para FCO2 em função da umidade do solo.

mod <- lm(fco2 ~ us, data=meus\_dados\_pad)  
summary.lm(mod)

beta0 <- mod %>% pluck(1) %>% pluck(1)  
beta1 <- mod %>% pluck(1) %>% pluck(2)  
meus\_dados\_pad %>%   
 ggplot(aes(x=fco2,y=us)) +   
 geom\_point()+  
 geom\_smooth(method="lm",se=FALSE) +  
 geom\_text(x=3,y=20, label=paste0("Y = ",  
 round(beta0,3),  
 " + ",  
 round(beta1,3),"X")  
 ,size=rel(5)) +   
 geom\_text(x=3,y=18, label="R² = 0,99",size=rel(5))

## 7. Delineamentos Experimentais

### 7.1 Delineamento Inteiramente Casualizados (DIC)

O delineamento inteiramente casualizado (DIC) é o mais simples de todos os delineamentos. No DIC, os tratamentos se distribuem ao acaso em todas as unidades experimentais e o número de repetições ou unidades experimentais por tratamento pode ser igual ou diferente.

É muito útil para o estudo de métodos e técnicas de trabalho de laboratório, ensaios de vegetação e em experimentos com animais.

Modelo matemático do DIC

é a observação na parcela que recebeu o tratamento i (i=1, …, k) na repetição j ( j=1, … , r);

é a média geral comum a todas as observações;

é o efeito do i-ésimo tratamento;

é o erro aleatório (efeitos não controlados na unidade);

Exemplo: Vamos simular um conjunto de dados com 4 tratamentos e 5 repetições

dados\_dic <- data.frame(  
 trat = gl(4,5,labels = c("A","B","C","D")),  
 y = c(rnorm(5,26,8),rnorm(5,39,5),rnorm(5,32,9),rnorm(5,22,8))  
)  
dados\_dic %>% head()

MODELO MATEMÁTICO NO R

mod <- aov(y ~ trat)

Onde: y – é a variável resposta (vetor Numérico)  
trat – tratamentos (deve ser uma variável categórica alocada como fator no R)

Então,

trat <- dados\_dic %>% pull(trat) %>% as\_factor()  
y <- dados\_dic %>% pull(y)   
mod <- aov(y ~trat)  
anova(mod)

Podemos realizar o teste de comparação de médias de Tukey

Tukey<-HSD.test(mod,"trat",group=TRUE,alpha=0.05)  
Tukey

Os mesmos resultados podem ser consguidos por meio da função dic do pacote ExpDes.pt.

dic(trat, y, quali = TRUE, mcomp = "tukey")

Rápida checagem dos resíduos

plot(mod)

### 7.2 Delineamento em Blocos Casualizados (DBC)

Neste delineamento os tratamentos são designados aleatoriamente a um grupo de unidades experimentais denominadas BLOCOS. O objetivo é manter a variabilidade entre as unidades dentro de cada BLOCO a menor possível, assim bem como maximizar as diferenças entre os BLOCOS. Se não existe diferença entre BLOCOS, este delineamento não contribuíra para a precisão em detectar diferenças entre tratamentos.

Modelo matemático do DBC

é a observação na parcela que recebeu o tratamento i (i=1, …, k) no bloco j ( j=1, … , r);

é a média geral comum a todas as observações;

é o efeito do i-ésimo tratamento;

é o efeito do j-ésimo bloco;

é o erro aleatório (efeitos não controlados na unidade);

Exemplo: Vamos simular um conjunto de dados com 3 tratamentos e 4 blocos

dados\_dbc <- data.frame(  
 trat = gl(3,4,labels = c("A","B","C")),  
 bloco = gl(4,1,12,labels = c("I","II","III","IV")),  
 y = c(sort(rnorm(4,26,6)),  
 sort(rnorm(4,39,5)),  
 sort(rnorm(4,32,5)))  
)  
dados\_dbc %>% head()

MODELO MATEMÁTICO NO R

mod <- aov(y ~ trat + bloco)

Onde: y – é a variável resposta (vetor Numérico)  
trat – tratamentos (deve ser uma variável categórica alocada como fator no R) bloco – blocos (deve ser uma variável categórica alocada como fator no R)

Então,

trat <- dados\_dbc %>% pull(trat) %>% as\_factor()  
bloco <- dados\_dbc %>% pull(bloco) %>% as\_factor()  
y <- dados\_dbc %>% pull(y)   
mod <- aov(y ~ trat + bloco)  
anova(mod)

Podemos realizar o teste de comparação de médias de Tukey

Tukey<-HSD.test(mod,"trat",group=TRUE,alpha=0.05)  
Tukey

Os mesmos resultados podem ser consguidos por meio da função dbc do pacote ExpDes.pt.

dbc(trat, bloco, y, quali = TRUE, mcomp = "tukey")

Rápida checagem dos resíduos

plot(mod)

### 7.3 Delineamento Quadrado Latino (DQL)

No delineamento Quadrado Latino os tratamentos são designados aos blocos de duas maneiras diferentes, geralmente designados por colunas e linhas. Temos dois controles locais.

Cada coluna e cada linha é um bloco completo de todos os tratamentos. Portanto, em um DQL, três fontes de variação explicáveis são identificáveis: linhas, colunas e tratamentos. Um particular tratamento é designado somente uma vez em cada linha e cada coluna. O número de tratamentos é igual ao número de linhas e colunas. O número total de observações é igual .

Modelo matemático do DQL

é a observação na parcela que recebeu o tratamento k (i = 1, …, a) na linha i ( i = 1,…,a) e coluna j (j = 1,…, a);

é a média geral comum a todas as observações;

é o efeito da i-ésimo linha;

é o efeito da j-ésimo coluna;

é o efeito do k-ésimo tratamento;

é o erro aleatório (efeitos não controlados na unidade);

Exemplo: Vamos simular um conjunto de dados com 3 tratamentos e 4 blocos

dados\_dql <- data.frame(  
 trat = as\_factor(  
 c("D","C","E","B","A",  
 "A","E","B","D","C",  
 "B","A","C","E","D",  
 "C","B","D","A","E",  
 "E","D","A","C","B")  
 ),  
 linha = gl(5,1,25,labels = c("I","II","III","IV","V")),  
 coluna = gl(5,5,labels = c("I","II","III","IV","V")),  
 y = c((rnorm(5,26,6)),  
 (rnorm(5,39,5)),  
 (rnorm(5,32,5)),  
 (rnorm(5,22,4)),  
 (rnorm(5,12,4)))  
)  
dados\_dql %>% head()  
table(dados\_dql$trat, dados\_dql$linha, dados\_dql$coluna)

MODELO MATEMÁTICO NO R

mod <- aov(y ~ trat + linha + coluna)

Onde: y – é a variável resposta (vetor Numérico)  
trat – tratamentos (deve ser uma variável categórica alocada como fator no R) linha – (deve ser uma variável categórica alocada como fator no R)

coluna – (deve ser uma variável categórica alocada como fator no R) Então,

trat <- dados\_dql %>% pull(trat) %>% as\_factor()  
linha <- dados\_dql %>% pull(linha) %>% as\_factor()  
coluna <- dados\_dql %>% pull(coluna) %>% as\_factor()  
y <- dados\_dql %>% pull(y)   
mod <- aov(y ~ trat + linha + coluna)  
anova(mod)

Podemos realizar o teste de comparação de médias de Tukey

Tukey<-HSD.test(mod,"trat",group=TRUE,alpha=0.05)  
Tukey

Os mesmos resultados podem ser consguidos por meio da função dql do pacote ExpDes.pt.

dql(trat, linha, coluna, y, quali = TRUE, mcomp = "tukey")

Rápida checagem dos resíduos

plot(mod)

### 7.4 Experimentos Fatoriais

Um experimento fatorial tem dois ou mais conjuntos de tratamentos que são analisados ao mesmo tempo. Vimos que tratamentos denotam níveis particulares de uma variável categórica independente, frequentemente denominada fator. Portanto, se dois ou mais fatores são examinados em um experimento, ele é um experimento fatorial.

Uma característica de um experimento fatorial é que todas as combinações dos níveis dos fatores é testada.

O efeito de um fator sozinho é denominado de efeito principal.

O efeito de diferentes fatores agindo juntos é denominado interação.

O delineamento experimental pode ser o DIC, DBC ou DQL. As combinações dos níveis dos fatores são aplicados aleatoriamente às unidades experimentais de acordo com o sorteio do DIC, do DBC ou do DQL.

Modelo matemático do Fatorial em DIC

Modelo matemático do Fatorial em DBC

é a observação na parcela que recebeu o nível i do fator A (i = 1, …, a) o nível j do fator B ( j = 1,…,b) presente na repetição (bloco) k (k = 1,…, r);

é a média geral comum a todas as observações;

é o efeito da i-ésimo nível do fator A;

é o efeito da j-ésimo nível do fator B;

é o efeito da interação observado na parcela que recebeu o i-ésimo nível do fator A e o j-ésimo nível do fator B;

é o efeito do k-ésimo bloco; (MODELO DE BLOCOS SOMENTE)

é o erro aleatório (efeitos não controlados na unidade);

Exemplo: Vamos simular um conjunto de dados com 6 tratamentos e 4 blocos, sendo o Fator A em 3 níveis e o fator B em 2 níveis.

dados\_dbc\_fatorial <- data.frame(  
 fator\_a = gl(3,4\*2,labels=c("a1","a2","a3")),  
 fator\_b = gl(2,1,24,labels = c("b0","b1")),  
 bloco = gl(4,2,24,labels = c("I","II","III","IV")),  
 y = c(sort(rnorm(4,26,6)),  
 sort(rnorm(4,39,5)),  
 sort(rnorm(4,32,5)),  
 sort(rnorm(4,40,6)),  
 sort(rnorm(4,12,5)),  
 sort(rnorm(4,10,5)))  
)  
dados\_dbc\_fatorial

MODELO MATEMÁTICO NO R

mod <- aov(y ~ fator\_a + fator\_b + fator\_a:fator\_b + bloco)

Onde: y – é a variável resposta (vetor Numérico)  
fator\_a – (deve ser uma variável categórica alocada como fator no R) fator\_b – (deve ser uma variável categórica alocada como fator no R) bloco – (deve ser uma variável categórica alocada como fator no R)

Então,

fator\_a <- dados\_dbc\_fatorial %>% pull(fator\_a) %>% as\_factor()  
fator\_b <- dados\_dbc\_fatorial %>% pull(fator\_b) %>% as\_factor()  
bloco <- dados\_dbc\_fatorial %>% pull(bloco) %>% as\_factor()  
y <- dados\_dbc\_fatorial %>% pull(y)   
mod <- aov(y ~ fator\_a + fator\_b + fator\_a:fator\_b + bloco)  
anova(mod)

Os mesmos resultados podem ser conseguidos por meio da função fat2\_dbc do pacote ExpDes.pt.

fat2.dbc(fator\_a, fator\_b, bloco, y, quali = c(TRUE,TRUE), mcomp = "tukey")

### 7.5 Experimentos em Parcelas Subdivididas

Nos experimentos fatoriais ou esquemas fatoriais os tratamentos gerados pelas combinações dos níveis dos fatores são designados às unidades experimentais de acordo com o procedimento de aleatorização do delineamento inteiramente casualizado (DIC), ou do delineamento em blocos casualizados (DBC), ou do delineamento em quadrado latino (DQL).

Entretanto, outros tipos de aleatorizações são possíveis. Uma dessas aleatorizações alternativas dá origem aos experimentos em parcelas subdivididas, os quais são um caso especial de blocos incompletos.

O princípio básico deste delineamento é que parcelas principais que recebem níveis de um fator são subdivididas em subparcelas ou subunidades, as quais recebem os níveis de um outro fator.

Assim cada parcela funciona como um bloco para as subparcelas. Os níveis do fator sorteado nas parcelas são denominados de tratamentos principais e os níveis do fator sorteados nas subparcelas são denominados de tratamentos secundários.

Modelo matemático de Parcelas subdivididas em DIC

Modelo matemático do de Parcelas subdivididas em DBC

é a observação na parcela que recebeu o nível i do fator A (i = 1, …, a) o nível j do fator B ( j = 1,…,b) presente na repetição (bloco) k (k = 1,…, r);

é a média geral comum a todas as observações;

é o efeito do k-ésimo bloco; (MODELO DE BLOCOS SOMENTE)

é o efeito da i-ésimo nível do fator A na parecela principal;

é o efeito aleatório do nível i do fator A na parcela principal k e constitui um erro expeirmental referente à parcela principal ik.

é o efeito da j-ésimo nível do fator B na subparcela;

é o efeito da interação observado na parcela que recebeu o i-ésimo nível do fator A e o j-ésimo nível do fator B;

é o erro aleatório (efeitos não controlados na unidade);

Exemplo: Vamos simular um conjunto de dados com 6 tratamentos e 4 blocos, sendo o Fator A em 3 níveis e o fator B em 2 níveis.

dados\_dbc\_fatorial <- data.frame(  
 fator\_a = gl(3,4\*2,labels=c("a1","a2","a3")),  
 fator\_b = gl(2,1,24,labels = c("b0","b1")),  
 bloco = gl(4,2,24,labels = c("I","II","III","IV")),  
 y = c(sort(rnorm(4,26,6)),  
 sort(rnorm(4,39,5)),  
 sort(rnorm(4,32,5)),  
 sort(rnorm(4,40,6)),  
 sort(rnorm(4,12,5)),  
 sort(rnorm(4,10,5)))  
)  
dados\_dbc\_fatorial

MODELO MATEMÁTICO NO R NO DIC

mod <- aov(y ~ fator\_a + (fator\_a:rep) + fator\_b + fator\_a:fator\_b)

Onde: y – é a variável resposta (vetor Numérico)  
fator\_a – (deve ser uma variável categórica alocada como fator no R) fator\_b – (deve ser uma variável categórica alocada como fator no R) rep – (deve ser uma variável categórica alocada como fator no R) referente às repetições das combinações dos fatores.

MODELO MATEMÁTICO NO R NO DBC

mod <- aov(y ~ bloco + fator\_a + Error(bloco/fator\_a) + fator\_b + fator\_a:fator\_b)

Onde: y – é a variável resposta (vetor Numérico)  
fator\_a – (deve ser uma variável categórica alocada como fator no R) fator\_b – (deve ser uma variável categórica alocada como fator no R) bloco – (deve ser uma variável categórica alocada como fator no R)

Então,

fator\_a <- dados\_dbc\_fatorial %>% pull(fator\_a) %>% as\_factor()  
fator\_b <- dados\_dbc\_fatorial %>% pull(fator\_b) %>% as\_factor()  
bloco <- dados\_dbc\_fatorial %>% pull(bloco) %>% as\_factor()  
y <- dados\_dbc\_fatorial %>% pull(y)   
mod <- aov(y ~ bloco + fator\_a + Error(bloco/fator\_a) + fator\_b + fator\_a:fator\_b)  
summary(mod)

Os mesmos resultados podem ser conseguidos por meio da função psub2.dbc do pacote ExpDes.pt.

psub2.dbc(fator\_a, fator\_b, bloco, y, quali = c(TRUE,TRUE), mcomp = "tukey")