Aula 16 – Análise de Dados com R (Parte 1)

Desvendando Dados com R: Sua Ferramenta para Histórias Impactantes

Bem-vindo(a) à Aula 16 do Curso de Jornalismo de Dados! Sabemos que a rotina pode ser puxada, mas a paixão por transformar números em narrativas é um combustível poderoso. Nesta aula, você dará os primeiros passos em uma jornada que o(a) capacitará a extrair insights valiosos de grandes volumes de informação, uma habilidade cada vez mais requisitada no mercado de trabalho e essencial para quem busca se destacar.

Em um mundo onde a informação é abundante, mas a sabedoria é escassa, a capacidade de analisar dados de forma crítica e eficiente se torna um superpoder. O R, com sua robustez e flexibilidade, é a ferramenta que o(a) ajudará a empunhar esse poder. Não se preocupe se você nunca programou antes; nossa abordagem será prática, passo a passo, focando no "porquê" antes do "como".

Ao final desta aula, você não apenas terá uma compreensão sólida dos fundamentos da manipulação e sumarização de dados com R, mas também será capaz de aplicar essas técnicas para resolver problemas reais. Você aprenderá a selecionar informações relevantes, filtrar ruídos, organizar dados de forma lógica e criar novas variáveis que revelam padrões ocultos. Prepare-se para transformar conjuntos de dados complexos em histórias claras e impactantes.

Nesta jornada, vamos construir uma base sólida para sua **literacia de dados**, capacitando-o(a) a não apenas manipular, mas também a interpretar e questionar os dados de forma crítica. Começaremos com a configuração do ambiente, passaremos pelas funções essenciais do pacote dplyr e culminaremos em um exemplo prático de análise de despesas públicas, conectando a teoria diretamente à aplicação.

A Era dos Dados: Por Que R é Seu Melhor Aliado?

Explosão de Dados

Bilhões de informações geradas a cada segundo por transações, interações sociais, sensores e dispositivos conectados

Desafio Atual

Não é mais encontrar informação, mas dar sentido a ela, transformando-a em conhecimento útil e histórias relevantes

R como Solução

Ambiente completo para análise estatística e visualização, com comunidade ativa e vasta gama de pacotes

Vivemos em uma era de explosão de dados. A cada segundo, bilhões de informações são geradas por transações, interações sociais, sensores e dispositivos conectados. Para um jornalista, um pesquisador ou um profissional que busca se qualificar para concursos, essa avalanche de dados pode parecer intimidadora. O desafio não é mais encontrar informação, mas sim dar sentido a ela, transformando-a em conhecimento útil e histórias relevantes.

Imagine que você é um detetive em uma cena de crime digital, onde cada dado é uma pista. Sem as ferramentas certas, você se perderia no emaranhado de evidências. É aqui que o R entra em cena. Ele não é apenas uma linguagem de programação; é um ambiente completo para análise estatística e visualização de dados, amplamente utilizado por cientistas, pesquisadores e, cada vez mais, por jornalistas de dados. Sua comunidade ativa e a vasta gama de pacotes disponíveis o tornam uma escolha poderosa e versátil.

O R permite que você automatize tarefas repetitivas, realize análises complexas com poucas linhas de código e crie visualizações que comunicam suas descobertas de forma clara e persuasiva. Em um cenário onde a **automação e IA na coleta de dados** estão redefinindo o campo, dominar ferramentas como o R é fundamental. Ele o(a) prepara para lidar com dados coletados via web scraping ou APIs, identificando padrões que seriam invisíveis a olho nu.

Nesta aula, vamos focar no pacote dplyr, uma parte essencial do ecossistema tidyverse do R. Pense no dplyr como sua caixa de ferramentas suíça para manipular dados. Ele oferece um conjunto de funções intuitivas que simplificam as operações mais comuns, permitindo que você se concentre na lógica da análise, e não na complexidade da programação.

Preparando o Terreno: Instalando R e RStudio

01

Instalar o R

Baixe e instale o R a partir do site oficial do CRAN (Comprehensive R Archive Network)

02

Instalar o RStudio

Baixe e instale o RStudio Desktop (versão gratuita) do site do RStudio

03

Configurar o Ambiente

Abra o RStudio e familiarize-se com os quatro painéis principais

Antes de mergulharmos na análise de dados, precisamos garantir que você tenha as ferramentas certas instaladas em seu computador. Pense nisso como preparar a bancada de um chef antes de começar a cozinhar: ter os ingredientes e utensílios à mão faz toda a diferença para um processo eficiente e um resultado delicioso.

O R é o motor por trás de tudo, a linguagem de programação em si. Já o RStudio é o ambiente de desenvolvimento integrado (IDE) que torna a interação com o R muito mais amigável e produtiva. Ele oferece uma interface visual com diferentes painéis para seu código, console, variáveis e gráficos, facilitando a organização e o fluxo de trabalho. É como ter um painel de controle completo para sua análise.

Ordem Importante: O RStudio precisa do R para funcionar, então sempre instale o R primeiro!

Este é o seu ponto de partida para explorar o universo dos dados. Com o R e o RStudio configurados, você tem em mãos um laboratório completo para experimentar, testar hipóteses e descobrir insights. Não se preocupe em memorizar todos os comandos agora; o importante é se familiarizar com o ambiente e entender que cada ferramenta tem um propósito específico para facilitar sua jornada de análise.

dplyr: Sua Caixa de Ferramentas para Manipulação de Dados

Caixa de Ferramentas

O dplyr oferece um conjunto de "verbos" ou funções que correspondem a operações comuns de manipulação de dados

Sintaxe Clara

Permite encadear múltiplas operações de forma lógica, como se você estivesse contando uma história para seus dados

Assistente Pessoal

Entende suas intenções e executa as tarefas de forma rápida e precisa, liberando você para focar na estratégia

Quando você trabalha com dados, raramente eles vêm prontos para a análise. É como receber uma caixa de peças de LEGO misturadas: você precisa selecionar as peças certas, remover as desnecessárias, organizá-las e, talvez, até montar algumas novas antes de construir seu modelo final. No mundo do R, o pacote dplyr é a sua caixa de ferramentas para fazer exatamente isso, de forma eficiente e elegante.

O dplyr faz parte do tidyverse, uma coleção de pacotes do R projetada para tornar a ciência de dados mais intuitiva e produtiva. Ele se baseia em um conjunto de "verbos" ou funções que correspondem a operações comuns de manipulação de dados. A beleza do dplyr reside na sua sintaxe clara e consistente, que permite encadear múltiplas operações de forma lógica, como se você estivesse contando uma história para seus dados.

Nesta seção, vamos explorar os quatro verbos fundamentais do dplyr: **select()**, **filter()**, **arrange()** e **mutate()**. Cada um deles tem um papel específico, mas juntos, eles formam a espinha dorsal da manipulação de dados eficaz. Dominar essas funções é o primeiro passo para transformar dados brutos em informações estruturadas e prontas para revelar suas histórias.

select(): Escolhendo as Colunas Certas para Sua Análise

Imagine que você está lendo um jornal e se depara com uma matéria enorme, cheia de detalhes. Para entender a essência, você naturalmente foca nos parágrafos e informações mais relevantes, ignorando o que é secundário para o seu objetivo. Com dados, a lógica é a mesma. Muitas vezes, um conjunto de dados vem com dezenas ou centenas de colunas, mas apenas algumas são realmente importantes para a pergunta que você quer responder.

> **Analogia:** É como ter um holofote que ilumina apenas as informações que você precisa

É aí que entra a função **select()** do dplyr. Ela permite que você escolha quais colunas (ou variáveis) deseja manter em seu conjunto de dados, descartando as que não são necessárias. Isso não apenas simplifica a visualização dos dados, mas também melhora o desempenho da sua análise, especialmente com grandes volumes de informação.

```
# Exemplo prático de select()
# Carregando o pacote dplyr
library(dplyr)

# Criando um dataframe de exemplo
despesas <- data.frame(
  id_despesa = 1:5,
  nome_orgao = c("Educação", "Saúde", "Saúde", "Transporte", "Educação"),
  valor = c(15000, 22000, 18000, 50000, 12000),
  data = as.Date(c("2024-01-15", "2024-01-20", "2024-02-01",
                   "2024-02-10", "2024-02-15")),
  categoria = c("Material", "Medicamentos", "Equipamento",
                "Infraestrutura", "Pessoal"),
  observacoes = c("Compra de livros", "Vacinas", "Tomógrafo",
                  "Asfalto", "Salários")
)

# Selecionando apenas as colunas 'nome_orgao', 'valor' e 'data'
despesas_simplificadas <- despesas %>%
  select(nome_orgao, valor, data)

print(despesas_simplificadas)
```

Neste exemplo, despesas_simplificadas agora contém apenas as três colunas que nos interessam, tornando a análise mais focada. Essa é uma etapa crucial para qualquer projeto de jornalismo de dados, pois evita a sobrecarga de informação e direciona o olhar para o que realmente importa.

filter(): Refinando Seus Dados para Encontrar o Que Importa

Depois de selecionar as colunas relevantes, o próximo passo é focar nas linhas (ou observações) que realmente importam para sua análise. Imagine que você está procurando por um livro específico em uma biblioteca enorme. Você não vai olhar todos os livros; em vez disso, você filtra por gênero, autor ou título. Nos dados, a função **filter()** do dplyr faz exatamente isso: ela permite que você selecione subconjuntos de linhas com base em condições específicas.

filter() é como ter um filtro de café: ele retém o pó (os dados irrelevantes) e deixa passar apenas o líquido (os dados que você precisa). Você pode usar uma ou várias condições lógicas para refinar seu conjunto de dados. Quer ver apenas as despesas de um órgão específico? Ou talvez só as despesas acima de um determinado valor? O filter() é a ferramenta perfeita para essas tarefas.

```
# Exemplo prático de filter()
# Filtrando despesas do órgão "Saúde"
despesas_saude <- despesas %>%
  filter(nome_orgao == "Saúde")

# Filtrando despesas com valor superior a 20000
despesas_altas <- despesas %>%
  filter(valor > 20000)

# Filtrando despesas da "Educação" com valor superior a 10000
despesas_educacao_altas <- despesas %>%
  filter(nome_orgao == "Educação", valor > 10000)
```

A combinação de select() e filter() já oferece um poder enorme para começar a explorar seus dados de forma direcionada, transformando um mar de informações em um lago navegável.

arrange(): Organizando a Informação para Revelar Padrões

Depois de selecionar as colunas e filtrar as linhas, muitas vezes precisamos organizar os dados para facilitar a visualização e a identificação de padrões. Pense em uma lista telefônica: ela não seria muito útil se os nomes não estivessem em ordem alfabética. Da mesma forma, ordenar seus dados pode revelar tendências, os maiores ou menores valores, ou a sequência cronológica de eventos.

A função **arrange()** do dplyr é a sua ferramenta para ordenar as linhas do seu conjunto de dados. Você pode organizar os dados em ordem crescente ou decrescente, com base em uma ou mais colunas. É como organizar os livros em uma estante por autor, título ou data de publicação, tornando muito mais fácil encontrar o que você procura e perceber a estrutura da coleção.

1 Ordem Crescente (Padrão)

Do menor para o maior valor

2 Ordem Decrescente

Use desc() para inverter a ordem

3 Múltiplas Colunas

Ordene por várias variáveis simultaneamente

```
# Exemplo prático de arrange()
# Ordenando despesas pelo valor (crescente)
despesas_ordenadas_valor_asc <- despesas %>%
 arrange(valor)

# Ordenando despesas pelo valor (decrescente)
despesas_ordenadas_valor_desc <- despesas %>%
 arrange(desc(valor))

# Ordenando por nome_orgao e depois por valor (decrescente)
despesas_ordenadas_multiplas <- despesas %>%
 arrange(nome_orgao, desc(valor))
```

mutate(): Criando Novas Perspectivas com Variáveis Derivadas

Cálculos

Percentuais, médias, proporções

Categorias

Classificações baseadas em condições

Combinações

Unir informações de várias colunas

Transformações

Aplicar funções matemáticas

Às vezes, os dados brutos não contêm todas as informações que você precisa. É como ter os ingredientes de uma receita, mas precisar combiná-los para criar um novo sabor ou um prato diferente. A função **mutate()** do dplyr permite que você crie novas colunas (variáveis) em seu conjunto de dados, baseadas em cálculos ou transformações de colunas existentes.

mutate() é a sua ferramenta para enriquecer seus dados, adicionando novas perspectivas que não estavam lá originalmente. A criação de novas variáveis é um pilar da análise de dados. Ela permite que você vá além do que está explicitamente nos dados e comece a construir métricas mais significativas.

```
# Exemplo prático de mutate()
# Criando uma nova coluna 'valor_em_milhares'
despesas_com_milhares <- despesas %>%
  mutate(valor_em_milhares = valor / 1000)

# Criando uma coluna 'mes_despesa' a partir da coluna 'data'
despesas_com_mes <- despesas %>%
  mutate(mes_despesa = format(data, "%Y-%m"))

# Criando uma coluna 'tipo_gasto' baseado no valor
despesas_com_tipo_gasto <- despesas %>%
  mutate(tipo_gasto = ifelse(valor > 20000, "Alto", "Baixo"))
```

Com mutate(), você não está apenas observando os dados; você está ativamente moldando-os para extrair mais significado. Essa capacidade de criar novas informações a partir das existentes é um diferencial para qualquer análise aprofundada.

Recapitulando os Verbos dplyr: Uma Orquestra de Dados

Até agora, exploramos individualmente os quatro verbos essenciais do dplyr: select(), filter(), arrange() e mutate(). Cada um deles tem um papel específico, mas o verdadeiro poder do dplyr reside na capacidade de combiná-los em uma sequência lógica, como os instrumentos de uma orquestra que, juntos, criam uma sinfonia.

Pense no operador `%>%` (lido como "pipe" ou "então") como o maestro dessa orquestra. Ele permite que você encadeie operações, passando o resultado de uma função diretamente para a próxima.

Verbo	Função	Aplicação	Exemplo
select()	Escolhe colunas	Redução de dimensionalidade	select(coluna1, coluna2)
filter()	Seleciona linhas	Filtragem condicional	filter(coluna > 100)
arrange()	Ordena linhas	Organização de dados	arrange(desc(coluna))
mutate()	Cria/modifica colunas	Transformação de variáveis	mutate(nova = antiga * 2)

Essa abordagem de encadeamento de operações é uma das características mais elogiadas do tidyverse, pois reflete a forma como pensamos sobre a manipulação de dados: "pegue estes dados, então faça isso, então faça aquilo". Isso não só torna seu código mais legível, mas também mais robusto e menos propenso a erros.

A Jornada da Análise: group_by() e summarise()

01

Dados Individuais

Análise linha por linha, coluna por coluna

02

Agrupamento

Definir categorias para análise coletiva

03

Sumarização

Calcular métricas agregadas por grupo

04

Insights

Padrões e tendências revelados

Até agora, aprendemos a manipular dados em um nível individual de linhas e colunas. Mas e se quisermos entender tendências gerais, médias, somas ou contagens para diferentes categorias? Imagine que você tem uma lista de vendas de uma loja e quer saber o total vendido por cada vendedor, ou o produto mais vendido em cada categoria. Analisar cada venda individualmente seria exaustivo.

É aqui que entram as funções **group_by()** e **summarise()** do dplyr. Elas são a dupla dinâmica para a agregação de dados, permitindo que você resuma informações em um nível mais alto. Pense nelas como a capacidade de ver a floresta, e não apenas as árvores.

> A combinação de group_by() e summarise() transforma dados detalhados em resumos concisos que revelam padrões e insights importantes.

Nesta seção, vamos desvendar como essas duas funções trabalham em conjunto para agregar seus dados. Primeiro, group_by() define os grupos nos quais a sumarização será aplicada. Em seguida, summarise() executa os cálculos desejados dentro de cada um desses grupos.

group_by(): Agrupando para Entender o Coletivo

Para entender o comportamento de um grupo, primeiro precisamos definir quais são esses grupos. A função **group_by()** do dplyr faz exatamente isso: ela divide seu conjunto de dados em grupos lógicos com base em uma ou mais variáveis categóricas. É como separar uma pilha de cartas de baralho por naipe ou por número antes de contar quantos ases você tem em cada naipe.

Quando você aplica group_by(), o R não muda a aparência do seu dataframe imediatamente, mas ele "marca" os dados internamente. Todas as operações subsequentes que você realizar (especialmente com summarise()) serão aplicadas separadamente a cada um desses grupos.

Importante: group_by() prepara os dados para operações de sumarização, mas não altera a visualização do dataframe

```
# Exemplo prático de group_by()
# Agrupando as despesas por 'nome_orgao'
despesas_agrupadas_orgao <- despesas %>%
  group_by(nome_orgao)

# O resultado não muda a visualização do dataframe,
# mas prepara-o para operações de sumarização
print(despesas_agrupadas_orgao)
```

A beleza do group_by() é que ele estabelece o contexto para a sua análise agregada. Sem ele, qualquer sumarização seria feita sobre o conjunto de dados inteiro, perdendo a nuance e a capacidade de comparar diferentes segmentos. Ele é o primeiro passo para transformar dados brutos em insights segmentados.

summarise(): Resumindo o Essencial de Cada Grupo

Soma (sum)

Total de valores em cada grupo

Média (mean)

Valor médio por grupo

Contagem (n)

Número de observações

Min/Max

Valores extremos em cada grupo

Uma vez que seus dados estão agrupados com group_by(), a função **summarise()** (ou summarize(), ambas funcionam) entra em ação para calcular métricas de resumo para cada um desses grupos. É como ter um relatório de vendas onde, após agrupar por vendedor, você calcula o "total de vendas", a "média de vendas por transação" e o "número de clientes atendidos" para cada um.

summarise() permite que você crie novas colunas que contêm os resultados de funções de agregação, como sum() (soma), mean() (média), median() (mediana), min() (mínimo), max() (máximo), n() (contagem de linhas), sd() (desvio padrão), entre outras.

```
# Exemplo prático de summarise()
# Agrupando por 'nome_orgao' e sumarizando métricas
resumo_por_orgao <- despesas %>%
  group_by(nome_orgao) %>%
  summarise(
    total_gasto = sum(valor),
    numero_despesas = n(),
    media_despesa = mean(valor)
  )

print(resumo_por_orgao)
```

Neste exemplo, transformamos uma lista detalhada de despesas em um resumo conciso que nos diz o gasto total, o número de despesas e a média por despesa para cada órgão. Essa é uma informação muito mais digerível e útil para um jornalista de dados que busca entender a alocação de recursos.

Combinando group_by() e summarise(): O Poder da Agregação

A verdadeira magia acontece quando group_by() e summarise() são usados em conjunto, encadeados pelo operador %>%. Essa combinação permite que você responda a perguntas complexas sobre seus dados de forma elegante e eficiente. É como ter um assistente que não só organiza seus documentos por categoria, mas também calcula automaticamente o total de páginas em cada uma, ou a data do documento mais recente.

```
# Exemplo prático: Combinando group_by() e summarise()
# Calcular métricas completas por órgão
resumo_orgao_completo <- despesas %>%
  group_by(nome_orgao) %>%
  summarise(
    total_gasto = sum(valor),
    numero_despesas = n(),
    media_por_despesa = mean(valor)
  ) %>%
  arrange(desc(total_gasto)) # Ordenar pelo total gasto

print(resumo_orgao_completo)
```

Neste código, primeiro agrupamos por nome_orgao. Em seguida, para cada grupo (cada órgão), calculamos a soma dos valores (total_gasto), a contagem de despesas (numero_despesas) e a média dos valores (media_por_despesa). Por fim, ordenamos o resultado para ver qual órgão teve o maior gasto total.

Essa capacidade de agregar dados é fundamental para qualquer análise que busque ir além do detalhe e entender as tendências macro. É a ponte entre os dados brutos e os insights acionáveis, permitindo que você construa narrativas baseadas em evidências sólidas.

Exemplo Prático: Analisando Dados de Despesas Públicas – Preparando o Terreno

Cenário Real

Simulação de análise de despesas públicas para jornalismo investigativo

Literacia de Dados

Interpretação crítica das informações encontradas

Ética e Transparência

Responsabilidade ao lidar com informações públicas

Agora que você conhece os verbos essenciais do dplyr e a dupla group_by()/summarise(), é hora de aplicar esse conhecimento em um cenário real. Vamos simular a análise de um conjunto de dados de despesas públicas, um tema de grande interesse para o jornalismo de dados e para a fiscalização da gestão pública.

Imagine que você é um jornalista investigativo e recebeu um arquivo com milhares de registros de gastos de um governo local. Sua missão é identificar padrões, os maiores gastos, e talvez até mesmo possíveis irregularidades. O primeiro passo é carregar esses dados no R e dar uma olhada inicial para entender sua estrutura.

```
# Carregando os pacotes necessários
library(dplyr)
library(lubridate) # Para trabalhar com datas

# Criando um dataframe de exemplo de despesas públicas
set.seed(123) # Para reprodutibilidade
despesas_publicas <- data.frame(
  id_despesa = 1:100,
  orgao = sample(c("Secretaria de Educação", "Secretaria de Saúde",
                   "Secretaria de Transporte", "Secretaria de Cultura"),
                 100, replace = TRUE),
  categoria = sample(c("Material de Consumo", "Serviços Terceirizados",
                       "Obras e Infraestrutura", "Pessoal", "Equipamentos"),
                     100, replace = TRUE),
  valor = round(runif(100, min = 500, max = 150000), 2),
  data_emissao = sample(seq(as.Date("2023-01-01"),
                            as.Date("2024-12-31"), by = "day"),
                        100, replace = TRUE),
  fornecedor = sample(c("Empresa A", "Empresa B", "Empresa C",
                        "Empresa D", "Empresa E", "Empresa F"),
                      100, replace = TRUE),
  status = sample(c("Pago", "Pendente", "Cancelado"), 100, replace = TRUE)
)

# Visualizando as primeiras linhas e a estrutura dos dados
print(head(despesas_publicas))
print(str(despesas_publicas))
```

Com esses dados carregados, temos nosso ponto de partida. A função head() nos mostra as primeiras linhas, e str() (structure) nos dá um resumo dos tipos de dados em cada coluna. Essa "primeira olhada" é vital para qualquer análise, pois nos ajuda a entender o que temos em mãos antes de começar a manipular.

Exemplo Prático: Despesas Públicas – Explorando com select() e filter()

Com nossos dados de despesas públicas carregados, o próximo passo é começar a explorá-los de forma mais direcionada. Lembra-se do detetive? Ele não olha todas as pistas ao mesmo tempo. Ele seleciona as mais promissoras e descarta as irrelevantes. É exatamente isso que faremos com select() e filter().

Vamos supor que, como jornalista, você está interessado(a) em investigar os gastos da "Secretaria de Saúde" e quer focar apenas nas colunas que indicam o "órgão", a "categoria", o "valor" e a "data de emissão". Além disso, você quer excluir despesas que foram "Canceladas", pois elas não representam um gasto efetivo.

```
# Exemplo prático: select() e filter() em despesas públicas
# Selecionar colunas e filtrar despesas da Secretaria de Saúde
despesas_saude_filtradas <- despesas_publicas %>%
  select(orgao, categoria, valor, data_emissao, status) %>%
  filter(orgao == "Secretaria de Saúde", status != "Cancelado")

# Visualizando as primeiras linhas do resultado
print(head(despesas_saude_filtradas))

# Quantas despesas da saúde foram filtradas?
print(paste("Número de despesas da Saúde (não canceladas):",
            nrow(despesas_saude_filtradas)))
```

Dica de Investigação: Sempre exclua dados cancelados ou inválidos para ter uma visão real dos gastos efetivos

Neste trecho de código, primeiro usamos select() para focar nas colunas que nos interessam. Em seguida, encadeamos com filter() para restringir os dados apenas às despesas da "Secretaria de Saúde" que não foram "Canceladas". O resultado é um conjunto de dados muito mais limpo e focado, pronto para análises mais profundas.

Essa etapa é crucial para garantir que sua análise seja relevante e que você não esteja perdendo tempo com dados que não contribuem para sua pergunta principal. É a arte de refinar o ruído para encontrar o sinal.

Exemplo Prático: Despesas Públicas – Transformando com mutate() e arrange()

Extrair Datas

Criar colunas de ano e mês para análises temporais

Ordenar Valores

Identificar as maiores despesas para investigação

Gerar Insights

Preparar dados para análises mais profundas

Com os dados da Secretaria de Saúde já selecionados e filtrados, podemos agora enriquecê-los e organizá-los para extrair mais insights. Lembra-se de mutate() para criar novas variáveis e arrange() para ordenar? Vamos aplicá-los para ver o que mais podemos descobrir.

Suponha que você queira calcular o mês e o ano de cada despesa para futuras análises temporais. Além disso, você quer identificar as 5 maiores despesas da Secretaria de Saúde para uma investigação mais aprofundada.

```
# Exemplo prático: mutate() e arrange() em despesas públicas
# Adicionar colunas de ano e mês, e ordenar pelas maiores despesas
despesas_saude_analise <- despesas_publicas %>%
  select(orgao, categoria, valor, data_emissao, status) %>%
  filter(orgao == "Secretaria de Saúde", status != "Cancelado") %>%
  mutate(
    ano_despesa = year(data_emissao), # Extrai o ano da data
    mes_despesa = month(data_emissao, label = TRUE) # Extrai o mês (com nome)
  ) %>%
  arrange(desc(valor)) # Ordena do maior para o menor valor

# Visualizando as 10 maiores despesas da Secretaria de Saúde
print(head(despesas_saude_analise, 10))
```

Neste código, após selecionar e filtrar, usamos mutate() para criar duas novas colunas: ano_despesa e mes_despesa, extraindo essas informações da coluna data_emissao. Isso é incrivelmente útil para análises de séries temporais. Em seguida, usamos arrange(desc(valor)) para ordenar o dataframe, colocando as maiores despesas no topo.

A capacidade de criar novas variáveis e de organizar os dados de forma significativa é o que permite que você vá além da superfície e comece a construir uma narrativa mais rica. Agora, temos um conjunto de dados da Secretaria de Saúde, focado, enriquecido e ordenado, pronto para a sumarização.

Exemplo Prático: Despesas Públicas – Sumarizando Insights com group_by() e summarise()

Chegamos à etapa final da nossa análise exploratória com R: a sumarização. Com os dados da Secretaria de Saúde já preparados, vamos usar group_by() e summarise() para responder a perguntas como: "Qual categoria de despesa mais contribui para o gasto total da Saúde?" ou "Qual a média de gastos por mês?".

Essas perguntas nos ajudam a identificar os principais focos de despesa e a entender a distribuição dos gastos ao longo do tempo. Para um jornalista, essas informações são ouro, pois podem apontar para áreas que merecem uma investigação mais aprofundada ou para políticas públicas que precisam de mais atenção.

```
# 1. Sumarizar por categoria na Secretaria de Saúde
resumo_por_categoria_saude <- despesas_publicas %>%
  select(orgao, categoria, valor, data_emissao, status) %>%
  filter(orgao == "Secretaria de Saúde", status != "Cancelado") %>%
  group_by(categoria) %>%
  summarise(
    total_gasto_categoria = sum(valor),
    numero_despesas_categoria = n(),
    media_despesa_categoria = mean(valor)
  ) %>%
  arrange(desc(total_gasto_categoria))

print(resumo_por_categoria_saude)

# 2. Sumarizar por mês e ano na Secretaria de Saúde
resumo_por_mes_ano_saude <- despesas_publicas %>%
  select(orgao, categoria, valor, data_emissao, status) %>%
  filter(orgao == "Secretaria de Saúde", status != "Cancelado") %>%
  mutate(ano_mes_despesa = format(data_emissao, "%Y-%m")) %>%
  group_by(ano_mes_despesa) %>%
  summarise(
    total_gasto_mes = sum(valor),
    numero_despesas_mes = n()
  ) %>%
  arrange(ano_mes_despesa)

print(resumo_por_mes_ano_saude)
```

> Com esses resumos, podemos ver rapidamente que "Obras e Infraestrutura" e "Serviços Terceirizados" são as categorias que mais consomem recursos na Secretaria de Saúde, e como os gastos se distribuem ao longo dos meses.

Essas são as "manchetes" que seus dados estão contando, prontas para serem aprofundadas e transformadas em uma reportagem impactante.

Ética e Transparência na Análise de Dados: Um Pilar Essencial

Ao manipular e analisar dados, especialmente em contextos como o jornalismo de dados ou a avaliação de políticas públicas, a **ética e a transparência** não são apenas boas práticas; são pilares fundamentais. A capacidade de usar ferramentas poderosas como o R vem com a responsabilidade de garantir que a análise seja justa, imparcial e que os resultados sejam apresentados de forma clara e honesta.

1 Documente seu código

Comente cada etapa da sua análise no R. Isso permite que outros (e você mesmo no futuro) entendam exatamente o que foi feito.

2 Seja transparente sobre as fontes

Sempre cite a origem dos dados e qualquer limitação conhecida.

3 Explicite suas escolhas

Se você filtrou dados, criou novas variáveis ou fez suposições, deixe isso claro. Por que você escolheu aquelas colunas? Por que excluiu certas linhas?

4 Considere o contexto

Dados raramente falam por si só. Apresente seus achados dentro de um contexto mais amplo, evitando conclusões precipitadas.

5 Verifique a validade

Certifique-se de que suas transformações e cálculos estão corretos e que os resultados fazem sentido.

A **literacia de dados** não é apenas sobre saber usar as ferramentas, mas também sobre a capacidade de questionar os dados, as fontes e as metodologias. Ao praticar a ética e a transparência, você não só fortalece a credibilidade da sua análise, mas também contribui para um ecossistema de informação mais confiável e responsável.

O Futuro da Análise: Automação e IA na Coleta de Dados

Web Scraping

Extração automatizada de dados de websites, eliminando a necessidade de cópia manual

APIs

Interfaces para comunicação direta com sistemas, obtendo dados em tempo real

Inteligência Artificial

Identificação de padrões complexos, previsões e insights que seriam difíceis de perceber

A jornada que você iniciou com o R é apenas o começo. O campo da análise de dados está em constante evolução, impulsionado por avanços em **automação e inteligência artificial (IA)**. Essas tecnologias estão redefinindo como coletamos, processamos e interpretamos dados, abrindo novas fronteiras para o jornalismo de dados e para a pesquisa em geral.

Imagine não precisar mais copiar e colar dados manualmente de sites, ou esperar por relatórios que demoram a ser publicados. A automação, através de técnicas como **web scraping** e o uso de **APIs (Application Programming Interfaces)**, permite que você colete dados em larga escala, de forma programática e contínua.

Conceito	Âmbito/Aplicação	Base/Origem	Exemplo
Web Scraping	Extração de dados de websites	Automação de navegação e parsing HTML	Coletar preços de produtos de e-commerce
APIs	Interface para comunicação entre sistemas	Padrões de requisição/resposta (JSON/XML)	Obter dados de redes sociais, governos
IA na Coleta	Otimização e identificação de padrões	Machine Learning, PLN	Filtrar notícias relevantes automaticamente

Para o jornalista de dados, isso significa mais tempo para focar na narrativa e na interpretação crítica, em vez de gastar horas em tarefas repetitivas de coleta e limpeza. O R é a sua porta de entrada para esse futuro, capacitando-o(a) a não apenas consumir, mas também a criar e inovar com dados.

Consolidação: Seus Primeiros Passos no Universo R

select()

Escolher as colunas certas

filter()

Filtrar as linhas relevantes

arrange()

Organizar seus dados

mutate()

Criar novas variáveis

group_by()

Agrupar dados para análise coletiva

summarise()

Sumarizar informações para extrair insights

Chegamos ao fim da nossa primeira aula sobre Análise de Dados com R. Você deu um passo gigante ao desvendar os fundamentos da manipulação e sumarização de dados, habilidades que são a base para qualquer projeto de jornalismo de dados ou análise crítica. Vimos como o R e o RStudio formam um ambiente poderoso, e como os verbos do dplyr são suas ferramentas essenciais para transformar dados brutos em informações significativas.

Em prática: Comece a aplicar esses conceitos em pequenos conjuntos de dados. Baixe dados abertos de governos, de portais de notícias ou até mesmo crie seus próprios dados de teste. A prática é a chave para a fluência.

Lembre-se de que cada linha de código é uma pergunta que você faz aos seus dados, e cada resultado é uma parte da história que eles têm a contar.

Autoavaliação

Questões Objetivas

1

Qual a principal função do operador %>% (pipe) no dplyr?

- a) Criar novas colunas a partir de existentes.
- b) Filtrar linhas com base em condições lógicas.
- c) Encadear múltiplas operações, passando o resultado de uma para a próxima.
- d) Ordenar as linhas de um dataframe em ordem crescente ou decrescente.

2

Para calcular o total_vendas por regiao, qual sequência usar?

Você tem um dataframe vendas com as colunas produto, regiao e valor.

- a) summarise() seguido de group_by().
- b) filter() seguido de summarise().
- c) group_by() seguido de summarise().
- d) mutate() seguido de arrange().

3

Função para criar lucro_liquido = receita - custo?

- a) select()
- b) filter()
- c) arrange()
- d) mutate()

4

Para remover despesas com status == "Cancelado"?

- a) select()
- b) filter()
- c) arrange()
- d) mutate()

Questão Discursiva

Questão: Explique a importância da **literacia de dados** e da **ética e transparência** ao utilizar ferramentas como o R para analisar dados em um contexto de jornalismo de dados.

Gabarito

Questão 1

c) Encadear múltiplas operações, passando o resultado de uma para a próxima.

Questão 2

c) group_by() seguido de summarise().

Questão 3

d) mutate().

Questão 4

b) filter().

Resposta Discursiva Esperada

> A literacia de dados é crucial para que o jornalista não apenas manipule os dados, mas também os interprete criticamente, questionando sua origem, metodologia e possíveis vieses. A ética e a transparência, por sua vez, garantem que a análise seja imparcial, que as fontes sejam citadas, as escolhas metodológicas explicitadas e os resultados apresentados de forma honesta, construindo credibilidade e evitando a desinformação.

Próxima Aula: Aula 17 – Fundamentos da Visualização de Dados

Transformando Insights em Visualizações Impactantes

Na próxima aula, daremos um passo adiante, transformando os insights que você extraiu dos dados em visualizações impactantes. Aprenderemos os fundamentos da visualização de dados, explorando como gráficos e tabelas bem construídos podem contar histórias de forma ainda mais poderosa e acessível. Prepare-se para dar vida aos seus números!

Recursos Adicionais

R for Data Science

Livro online gratuito e abrangente sobre o tidyverse (Hadley Wickham & Garrett Grolemund)

Documentação oficial do dplyr

Para detalhes sobre todas as funções e seus argumentos

Coursera/edX

Cursos de Introdução ao R para aprofundar seus conhecimentos em programação R

NOTA IMPORTANTE: As informações regulatórias/legais/técnicas desta aula estão atualizadas até 2025. Consulte sempre fontes oficiais para verificar alterações.