



Estatística para Ciência de Dados

Estruturas de dados e extração/importação de dados para análise descritiva

Fatec 2025

Importando arquivos de uma pasta

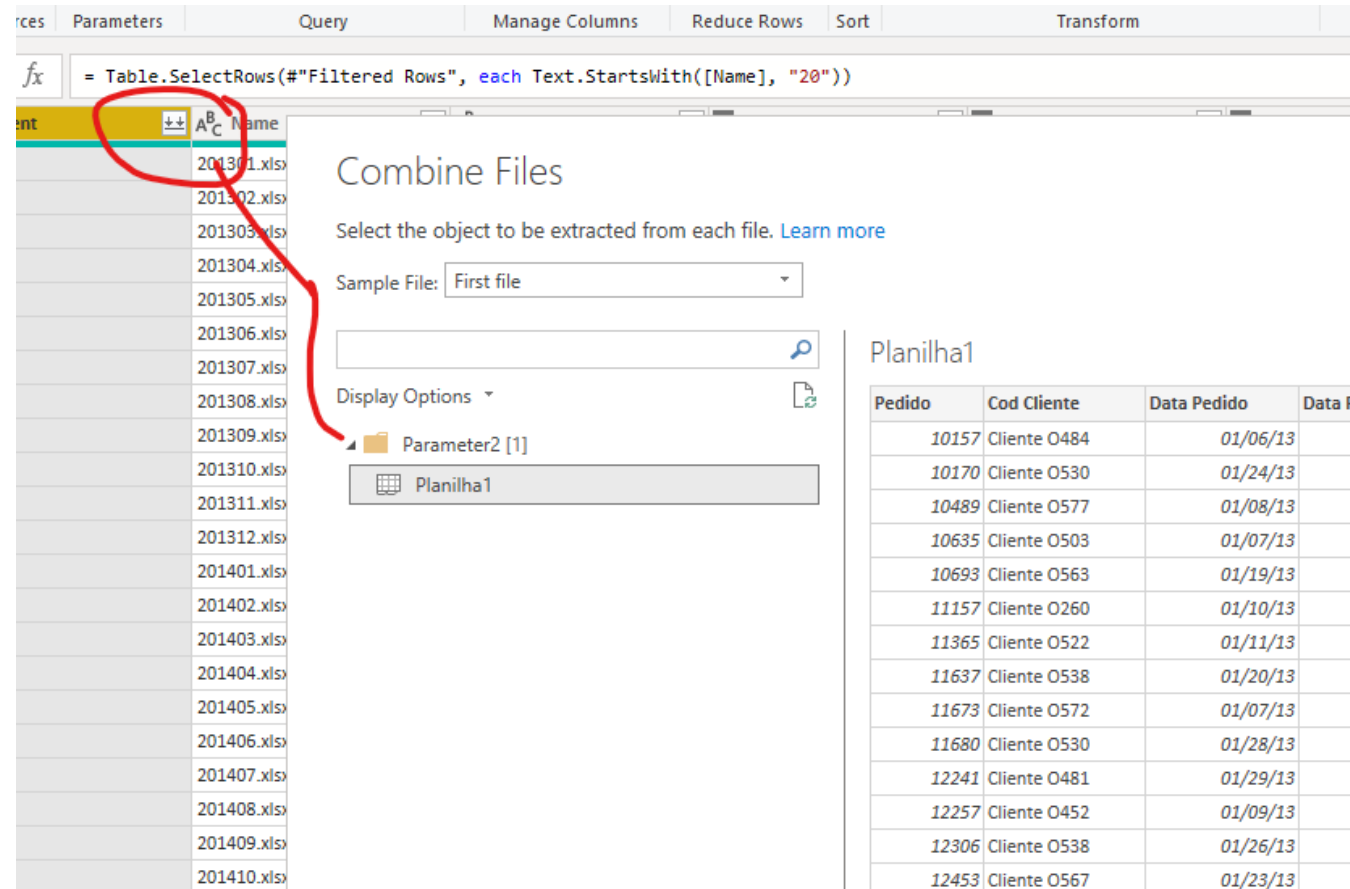
Muitos arquivos de Excel com a mesma estrutura mas que apresentam recortes de tempo ou valores diferentes

Obter dados/Pasta

- Dentro da pasta existem 91 arquivos com dados de vendas de 2013 a 2020
- Use **combinar e transformar dados**
- Todos os arquivos serão combinados através de uma função
- A pasta “consulta de exemplo” armazena esta função e não pode ser apagada
- Crie um gráfico de colunas para visualizar os dados dos 91 arquivos

Importando apenas arquivos desejados da pasta

- Na pasta “relatórios exportados desejados” existem arquivos .txt
 - Sendo assim, não é possível fazer a importação automática
 - Clique direto em **Transformar dados**
 - Filtre pela coluna **Extensão**, mantendo apenas os arquivos .XLSX
 - Na coluna **Nome** filtre apenas os arquivos cujo nome **começa com “20”**
 - Após o filtro, combine os arquivos clicando em “Combinar”



The screenshot shows the Power Query 'Combine Files' dialog box. The 'Sample File' dropdown is set to 'First file'. The 'Display Options' section shows 'Parameter2 [1]' and 'Planilha1'. A red circle highlights the 'Planilha1' option. The background shows a list of files with extensions like .xlsx and .xls.

Planilha1

Pedido	Cod Cliente	Data Pedido	Data F
10157	Cliente O484	01/06/13	
10170	Cliente O530	01/24/13	
10489	Cliente O577	01/08/13	
10635	Cliente O503	01/07/13	
10693	Cliente O563	01/19/13	
11157	Cliente O260	01/10/13	
11365	Cliente O522	01/11/13	
11637	Cliente O538	01/20/13	
11673	Cliente O572	01/07/13	
11680	Cliente O530	01/28/13	
12241	Cliente O481	01/29/13	
12257	Cliente O452	01/09/13	
12306	Cliente O538	01/26/13	
12453	Cliente O567	01/23/13	

Importando dados de PDF

- Vá até **Página inicial/obter dados/mais/PDF**
- Importe o arquivo “relatório gerencial”
- Os principais elementos serão tabelas e páginas
- Selecione **table004** e pressione **Transformar dados**
 - Formate colunas como número decimal, remova as colunas desnecessárias e renomeie a tabela

Queries [1] <

Table004 (Page 2)

fx = Table.TransformColumnTypes(#"Promoted Headers",{{"VARIÇÕES PATRIMONIAIS DIMINUTIVAS", type text}, {"NE", Int64.Type}, {"2018 (R\$ mil)", type text}, {"2017 (R\$ mil)", type text}})

A ^B _C VARIÇÕES PATRIMONIAIS DIMINUTIVAS	1 ² ₃ NE	A ^B _C 2018 (R\$ mil)	A ^B _C 2017 (R\$ mil)
1 Pessoal e Encargos	15	188.453,98	185.179,34
2 Remuneração a Pessoal	null	153.348,72	149.453,32
3 Encargos Patronais	null	28.712,18	29.177,23
4 Benefícios a Pessoal	null	6.314,44	6.502,44
5 Outras Var. Patrimoniais Diminutivas - Pessoal e Encargos	null	78,64	46,35
6 Benefícios Previdenciários e Assistenciais	16	5.100,43	4.635,23
7 Aposentadorias e Reformas	null	4.417,88	3.933,21

Importando arquivos de um banco de dados

A conexão com diferentes banco de dados possui requisitos parecidos: é necessário informar o caminho ou nome do servidor e as credenciais de acesso

Após escolher a conexão específica com o banco de dados, o nome do servidor é solicitado

São duas opções de conectividade:

- Importar – carrega os dados no PBI (até 10Gb)
- Direct Query – fornece apenas a conexão com o banco e não carrega os dados no PBI

Importar x Direct Query

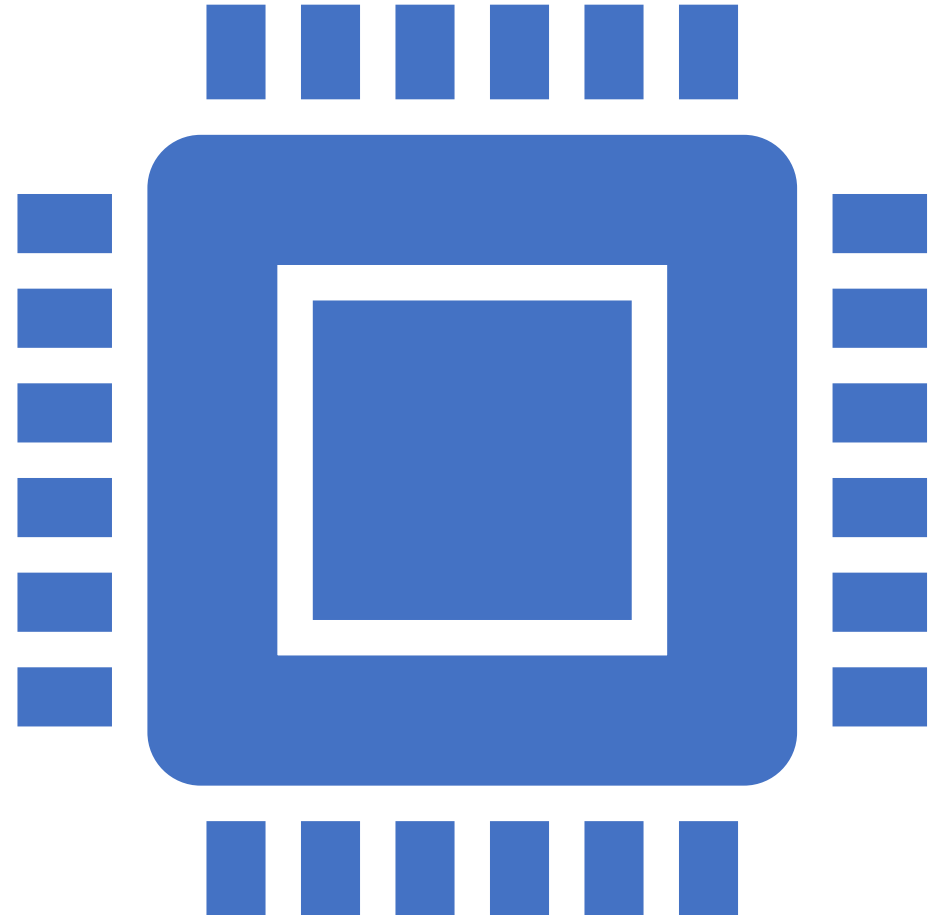
Sempre prefira Importar (por exemplo, questões de desempenho)

Use direct Query quando

- Dados da fonte são muito grandes
- Alterações frequentes necessárias
- Soberania dos dados – DQ usa as credenciais de segurança do banco
- A fonte é multidimensional e contém medidas – usando DQ é possível realizar o tratamento dessas medidas

Importando dados da web

- Uma das conexões mais utilizadas
- Acesse os dados do portal:
<http://www.yahii.com.br/dolar.html>
- Em seguida, transforme as colunas em linhas, utilizando as opções do Editor de Consultas.



Importando dados de uma api

- APIs (Application Programming Interface) permitem conectar aplicações e seus dados com usuários interessados em extrair e utilizar essas informações
- Os dados geralmente estão em formato JSON ou XML
- Acesse: <https://servicodados.ibge.gov.br/api/v1/localidades/estados>
 - Converta para tabela, expanda o objeto Record clicando nas setas ao lado do nome da coluna e
 - desmarque a opção “Usar o nome da coluna original como prefixo”.
- Acesse: <https://servicodados.ibge.gov.br/api/v1/localidades/estados/33|35/municipios>
 - Filtro: Somente estado do Rio e São Paulo

Importando dados de arquivos texto formatado

- Use o arquivo “relatório exportado.txt”



Importando dados com R

```
[111] #Bibliotecas
library(ggplot2)

#setwd('C:/Dados')
names <- c("horario", "temp", "vento", "umid", "sensa")
con <- url("https://ic.unicamp.br/~zanoni/cepagri/cepagri.csv")
cepagri <- read.table(con, header = FALSE, fill = TRUE, sep = ";", col.names = names)
head(cepagri)
tail(cepagri)
```

Tratamento dos dados

```
[112] class(cepagri) ##f verificando a classe  
typeof(cepagri) ##f verificando o tipo de dado
```

↔
'data.frame'
'list'

```
##### Tratamento dos dados #####  
|  
# Convertendo o tipo dos campos Temp e Horário  
sapply(cepagri, class)  
# Temp  
cepagri$temp <- as.numeric(cepagri$temp)  
class(cepagri$temp)  
# Horário  
cepagri$horario <- as.POSIXct(cepagri$horario, format = '%d/%m/%Y-%H:%M', tz="America/Sao_Paulo")  
class(cepagri$horario)
```

Filtrando os dados

```
totalcepagri <- nrow(cepagri) # Total de linhas =
print(totalcepagri)
# Filtrando o período de 01/01/2015 a 31/12/2019
cepagri <- cepagri[cepagri$horario >= "2025-01-01" & cepagri$horario < "2025-03-26", ]

# Verifica se veio algum dado
if (nrow(cepagri) == 0) {
  message("Nenhum dado disponível para o período de 2025.")
} else {
  print(summary(filtro))
}
periodoleitura <- nrow(cepagri)
print(periodoleitura) # Total de linhas =
```

```
Totalexcluidas <- (totalcepagri- periodoleitura) # total de linhas =
print(Totalexcluidas)
```

[1] 558512

```
head(cepagri) # somente dados do filtro
tail(cepagri)
```

Visualização dos dados

```
# Criar duas colunas: ano e mês
```

```
cepagri$horario <- as.POSIXlt(cepagri$horario)
cepagri$ano      <- unclass(cepagri$horario)$year + 1900
cepagri$mes      <- unclass(cepagri$horario)$mon + 1
```

```
# verificando os dados de temperatura através de um grafico
```

```
novadata = as.POSIXct(cepagri$horario)
gtm18 <- ggplot(cepagri, aes(x = novadata, y= temp))
gtm18 <- gtm18 + geom_point()
gtm18 <- gtm18 +geom_smooth()
gtm18
```

Análise de dados

```
##### Análise de Dados #####
```

```
summary(cepagri)
med <- mean(cepagri$temp, na.rm = TRUE) #média de temperatura
md <- median(cepagri$temp, na.rm = TRUE) #mediana
sd(cepagri$temp, na.rm = TRUE) #desvio padrao
var(cepagri$temp, na.rm = TRUE) # variancia
mt <- max(cepagri$temp, na.rm = TRUE) # valor máximo
mint <- min(cepagri$temp, na.rm = TRUE) # valor mínimo
quantile(cepagri$temp, na.rm = TRUE) # gerando os quartis
```

Visualização dos dados

```
plot(cepagri$temp, xlab = "período de medição", ylab = "Temperatura", main = "Temperaturas em Campinas (2025)")
lines(cepagri$temp) # plotando os dados em linha
abline(med, 0, col = "red") # indicador da média
abline(md, 0, col = "blue") # indicador da mediana
abline(mt, 0, col = "purple") # indicador do máximo
abline(mint, 0, col = "purple3") # indicador do mínimo
quartis <- quantile(cepagri$temp, na.rm = TRUE) # armazenando os quartis
abline(quartis[[2]], 0, col = "green1") # plotando o 2.o quartil
abline(quartis[[4]], 0, col = "green1") # plotando o 4.o quartil
amplitude <- quartis[[4]] - quartis[[2]] # calculando a amplitude
limsup <- mean(cepagri$temp, na.rm = TRUE) + 1.5 * amplitude #interpolacao para encontrar limites
liminf <- mean(cepagri$temp, na.rm = TRUE) - 1.5 * amplitude #interpolacao para encontrar limites
abline(limsup, 0, col = "red3") # plotando o limite superior
abline(liminf, 0, col = "red3") # plotando o limite inferior
```