

# Bioestatística – Gráficos

Eric Kauati Saito

2022-06-21

## Observações

Lembrem que tudo que estiver depois de # é um comentário, o programa não vai ler essa parte. Recomendo utilizar para que vocês lembrem o que está sendo feito, e até para explicar BREVEMENTE o que vocês estão fazendo e por que, dessa forma se vocês abrirem o programa para trabalhar no dia seguinte vocês não vão se esquecer de tudo que fizeram.

Lembrem que vocês precisam dar 'Run' em cada linha, ou se quiserem marcar mais de uma linha e apertar no botão 'Run' vai rodar todas as linhas marcadas.

Alguns atalhos:

"ctrl" + "l" Limpa o console

"ctrl" + "enter" Roda a linha, ou as linhas marcadas (pode ser utilizado no lugar do botão 'Run')

## Carregar dados

Essa parte do código é onde vamos importar nossos dados para o programa. Para isso vamos utilizar a função:

```
read.table().
```

Ao utilizar essa função precisamos atribuir uma variável a ela. Ou seja, onde vamos guardar os nossos dados que estamos importando do arquivo ".csv". Dentro do parantêses é onde vamos colocar nossos parâmetros necessários para utilizar a função.

Nesse caso precisamos dizer onde e qual é o arquivo que estamos tentando importar. Para isso vamos fazer da seguinte forma:

```
file = "arquivo"
```

Perceba que o está entre "" ou ", isso indica que ele é um texto e não um número ou uma variável, como o nome do arquivo é um texto, vamos usar "". Veja que precisamos dizer o local onde o arquivo está salvo. Se vocês estão com dificuldade em como escreve esse local, aperta com botão direito do mouse em cima do arquivo e vai em propriedades, ali vai ter uma região escrita 'local'. (Se o seu computador estiver em português). Obs.: Cuidado ao colocar o local, vejam no exemplo que eu precisei colocar mais um '\'. E cuidado também que no final do nome do arquivo vocês precisam colocar o tipo do arquivo, no meu caso foi '.csv', mas poderíamos ter outros formatos como '.xls' que também é um formato de arquivo de excel.

Além disso, se vocês lembram da aula, precisamos informar a função como ele separa os valores dentro dele (Geralmente separam com ',' ou com ';'). Para isso vamos escrever da seguinte forma:

```
sep = ";" ou sep = ","
```

E por último, precisamos dizer para a função que as colunas da nossa tabela tem nomes (Ex.: sistólica, diastólica). Para isso:





```
header = TRUE
```

```
dados = read.table(file="D:\\Macaes Disciplinas\\2022\\R\\dados_pratica.csv",  
sep=";", header=TRUE)
```

Veja que depois de cada parâmetro tem uma vírgula e vejam também que todos os parâmetros estão dentro dos parênteses.

A variável 'dados' poderia ter sido qualquer outro nome, tentem colocar um nome que seja mais adequado e fácil de vocês lembrarem (Ex.: diastolica). Não coloquem acento no nome da variável, e cuidado se está maiúscula ou minúscula as letras, o programa entende como variáveis diferentes:

```
dados != Dados
```

 dados	59 obs. of 12 variables	
 Dados	59 obs. of 12 variables	

Talvez o arquivo de dados que vocês pegaram é muito grande e vocês não pretendem utilizar todos. Nesse exemplo vocês podem ver que dados tem 59 observações e 12 variáveis. Se pretendem trabalhar com menos observações, uma forma de fazer isso:

```
dados_reduzidos = dados[1:40,] # Pega as observações: 1 até 40
```

Um pequeno detalhe, percebam que tem uma ',' depois do número. Se por algum motivo vocês não queriam pegar a primeira observação vocês poderiam ter feito:

```
dados_reduzidos = dados[2:40,]
```

Vejam que diferente do anterior, vocês teriam no final 39 observações.

## Trabalhando com os dados

Tentem abrir os dados, clique na variável que vocês criaram, na aba 'Environment'. Vocês vão ver que temos 12 colunas, cada um com uma informação referente ao voluntário (cada linha é um voluntário). Talvez não sejam importantes todas essas informações e estamos interessados em saber por exemplo a idade ou o nível de glicose.

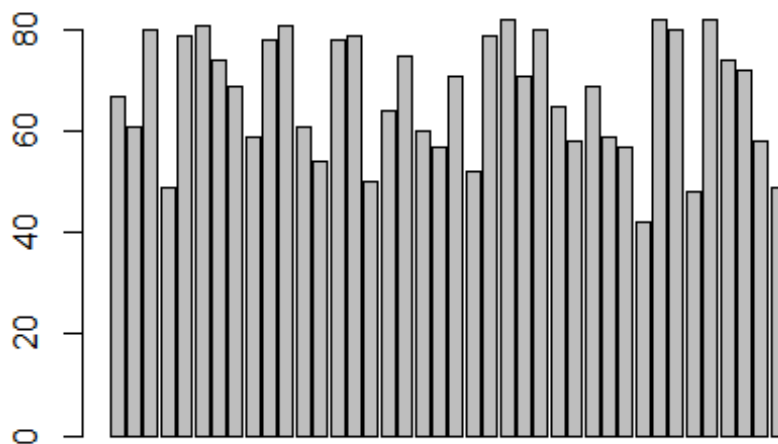


## Gráficos

### Gráfico de barras

Para fazer um gráfico de barras vamos utilizar a função `barplot()`.

```
barplot(idade)
```



Claramente não é o gráfico que estamos esperando, isso porque o programa está colocando em cada barra um voluntário e mostrando a idade dele (deve ter 40 barras, podem tentar contar e me avisam depois...).

Para corrigir isso vamos utilizar outra função: `table()`

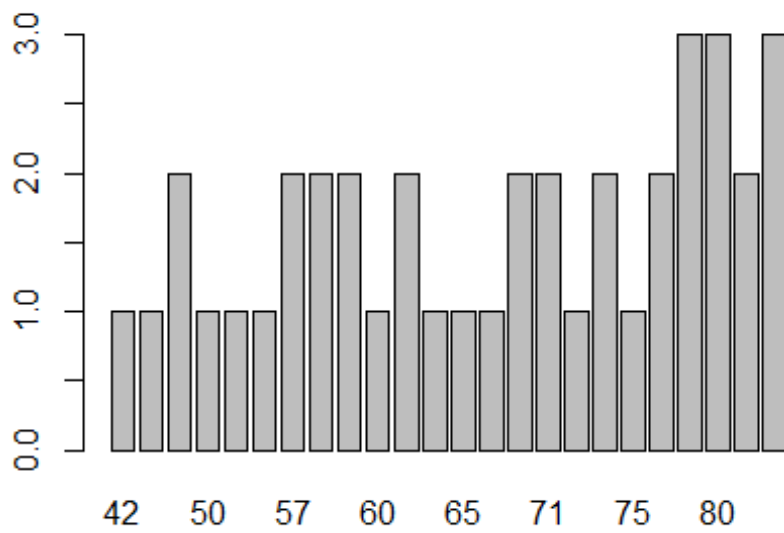
Essa função vai juntar todos que tiverem os mesmos valores:

```
table(idade)
```

```
## idade
## 42 48 49 50 52 54 57 58 59 60 61 64 65 67 69 71 72 74 75 78 79 80 81 82
##  1  1  2  1  1  1  2  2  2  1  2  1  1  1  2  2  1  2  1  2  3  3  2  3
```

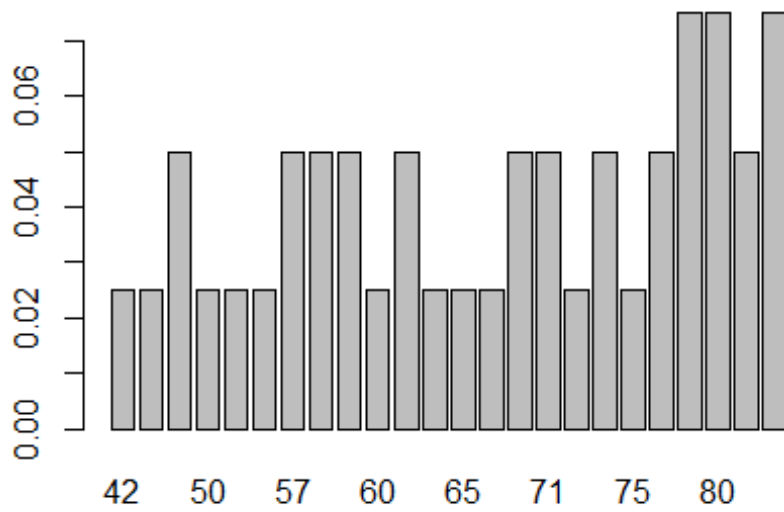
Mas como talvez só usaremos isso uma vez, não precisamos ter uma nova variável para isso, então podemos utilizar diretamente dentro da função `barplot()`.

```
barplot(table(idade))
```



Pergunta: O que temos no eixo x e no eixo y?

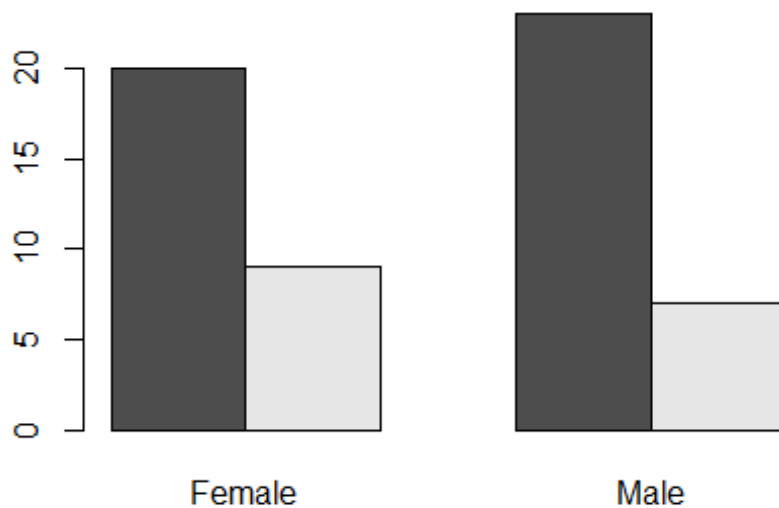
```
barplot(table(idade)/length(idade))
```



E agora, o que temos no eixo x e no eixo y? Obs.: `length(idade)` é o número de observações em idade

De repente, estamos interessados em fazer gráficos de barras agrupados. Por exemplo, se estamos interessados em apresentar o número de mulheres com hipertensão e aqueles que não tem comparado com o número de homens com hipertensão.

```
barplot(table(dados$hypertension, dados$gender), beside = TRUE) #beside = TRUE, coloca a barra do nº de pessoas hipertensas e pessoas não hipertensas lado a lado
```



## Histograma

Vamos tentar fazer o histograma do nível de glicose. Utilizando a função:

```
hist()
```

```
hist(glicose)
```



Como vimos em aula, talvez esse não seja a melhor divisão de faixa de valores, podemos mudar o parâmetro 'breaks' da função para mudar isso.

```
hist(glicose, breaks = 6)
```



Tentem modificar o parâmetro `breaks` e veja como ele muda o histograma.

Se vocês perceberem, no eixo y temos a frequência absoluta. Muitas vezes queremos ver a frequência relativa. Para isso faremos da seguinte forma:



```
h <- hist(glicose, breaks = 6, plot = FALSE)
h$counts=h$counts/sum(h$counts)*100
plot(h)
```



Não se assustem. Podem copiar essa parte do código e utilizar, só modifiquem o nome da variável que vocês colocaram e o número de 'breaks'.

Agora temos nosso histograma em frequência relativa (%).

Está quase completo, precisamos mudar o nome do eixo x e do eixo y do gráfico e o título. Para isso vamos ter que mudar alguns parâmetros da nossa função (esses parâmetros serão iguais para qualquer gráfico):

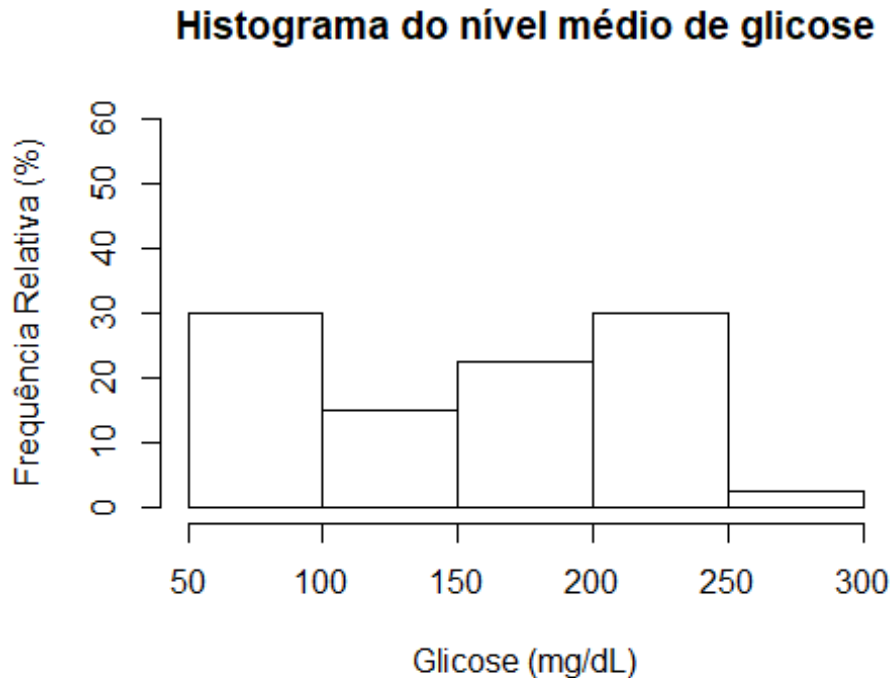
main = "Título"

xlab = "nome do eixo x"

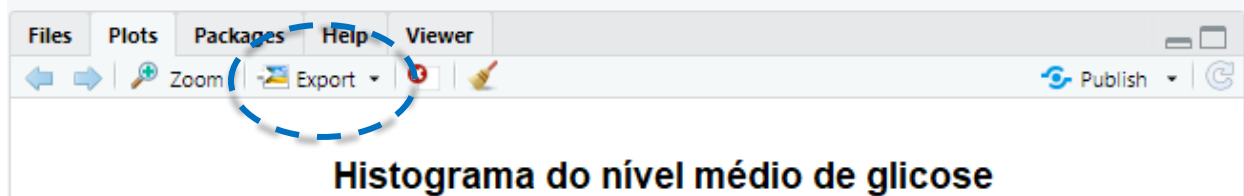
ylab = "nome do eixo y"

ylim = c(0,60) # Muda os limites no eixo y do nosso gráfico, tentem mudar os valores e vejam o que acontece.

```
h <- hist(glicose, breaks = 6, plot = FALSE)
h$counts=h$counts/sum(h$counts)*100
plot(h,
      ylim = c(0, 60),
      main = "Histograma do nível médio de glicose",
      xlab="Glicose (mg/dL)",
      ylab = "Frequência Relativa (%)")
)
```



Pronto, se está tudo certo e é esse o gráfico que iremos utilizar, podemos apertar o botão 'Export' na aba 'Plots' e salvar como imagem (ou PDF).

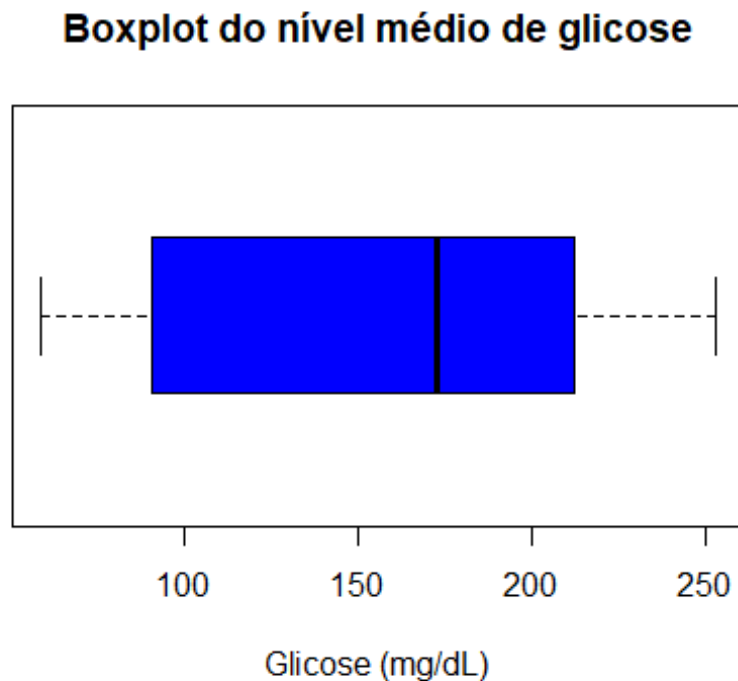


Obs.: O gráfico de barras agrupados feito anteriormente tem os eixos y “errados”, como vocês corrigiriam?

## Boxplot

Vamos tentar gerar o gráfico de boxplot. Para simplificar, vou pular direto para o gráfico já mudando os parâmetros para mudar os títulos:

```
boxplot(glicose,  
        horizontal = TRUE,  
        main = "Boxplot do nível médio de glicose",  
        xlab="Glicose (mg/dL)",  
        col = "blue"  
        )
```



Vejam que agora temos um novo parâmetro:

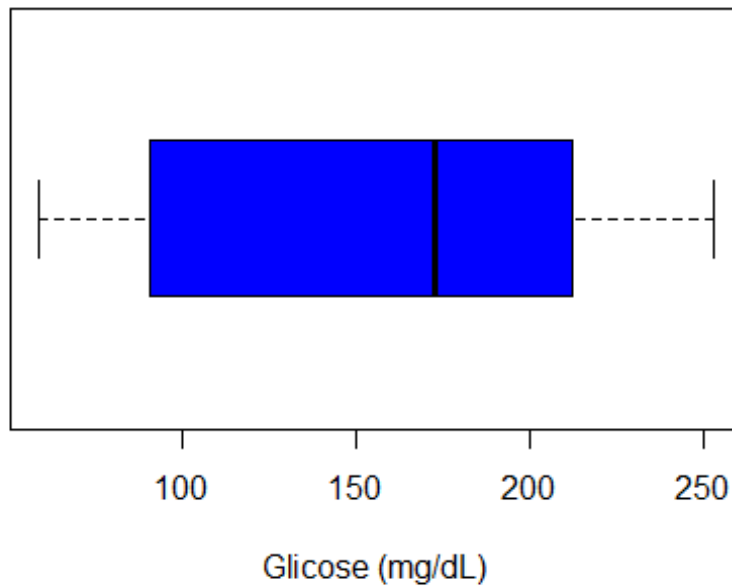
`col = "cor"`

Podemos utilizar ele para mudar a cor do nosso boxplot.

Vejam que no começo do programa nós criamos a variável `glicose` (`glicose = dados_reduzidos$avg_glucose_level`). Fizemos isso porque íamos trabalhar bastante utilizando esse dado de nível de glicose. Se você sabe que vai usar poucas vezes (ou até uma única vez), ou já estão mais acostumados com o programa, não era necessário criar essa variável, vocês podiam ter pulado essa etapa e utilizado diretamente na função. Exemplo:

```
boxplot(dados_reduzidos$avg_glucose_level,  
        horizontal = TRUE,  
        main = "Boxplot do nível médio de glicose",  
        xlab="Glicose (mg/dL)",  
        col = "blue"  
        )
```

### Boxplot do nível médio de glicose

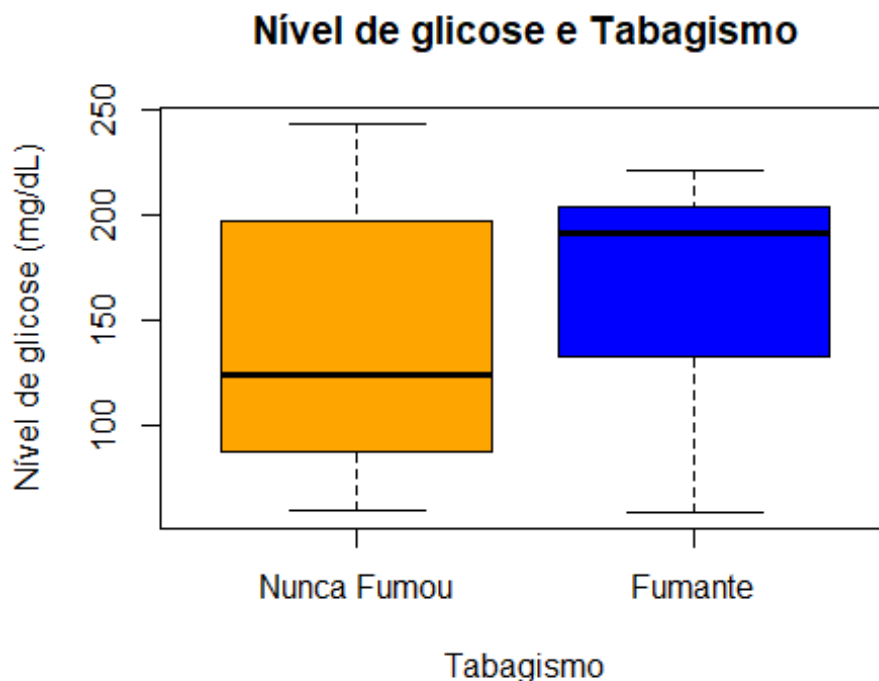


Vejam que no lugar da variável 'glicose', eu coloquei direto `dados_reduzidos$avg_glucose_level`.

Muito provavelmente estamos interessados em fazer uma comparação entre boxplots.

```
n_fumante = dados[dados$smoking_status=='never smoked',] #Pessoas que nunca fumaram
fumante = dados[dados$smoking_status=='smokes',] #Pessoas que fumam

boxplot(n_fumante$avg_glucose_level, fumante$avg_glucose_level,
        names = c('Nunca Fumou', 'Fumante'),
        col = c('orange', 'blue'),
        main = "Nível de glicose e Tabagismo",
        ylab = "Nível de glicose (mg/dL)",
        xlab= "Tabagismo")
```



Nesse caso, apresentamos o boxplot do nível médio de glicose das pessoas que nunca fumaram e das pessoas que fumam.

E como separamos esses dados? Abram novamente os dados e vejam a coluna 'smoking\_status', podemos ver que existem 4 opções de resposta nessa pesquisa: 'formerly smoked', 'never smoked', 'smokes', 'Unknown'. Como estamos interessados nas pessoas que fumam e nas pessoas que nunca fumaram, estamos interessados em 'never smoked' e 'smokes'. Para separar os dados e escolher apenas os 'never smoked' podemos fazer da seguinte forma:

```
dados[dados$smoking_status=='never smoked',]
```

`dados$smoking_status=='never smoked'` -> Verifica se está escrito 'never smoked' na coluna 'smoking\_status', se sim vai mostrar TRUE (verdadeiro) e caso contrário FALSE (falso).

```
dados$smoking_status=='never smoked'
```

```
## [1] FALSE TRUE TRUE FALSE TRUE FALSE TRUE TRUE FALSE FALSE TRUE FALSE
## [13] FALSE FALSE TRUE TRUE FALSE FALSE TRUE FALSE FALSE TRUE TRUE FALSE
## [25] FALSE TRUE FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE TRUE FALSE
## [37] FALSE FALSE FALSE TRUE FALSE TRUE FALSE FALSE FALSE TRUE FALSE FALSE
## [49] TRUE FALSE FALSE FALSE TRUE TRUE TRUE FALSE TRUE FALSE TRUE
```

Ao colocar esse resultado dentro dos colchetes na nossa variável 'dados', vamos estar escolhendo apenas aqueles que são TRUE, ou seja, apenas aqueles que no 'smoking\_status' está escrito 'never smoked'. É a mesma lógica de quando nós criamos a função dados\_reduzidos com apenas 40 observações lá no começo do programa (dados[1:40,]). Depois de criada a variável 'n\_fumante', abram e vejam o que tem lá dentro, e verifiquem a aba 'smoking status', vão ver só vai ter pessoas que nunca fumaram.

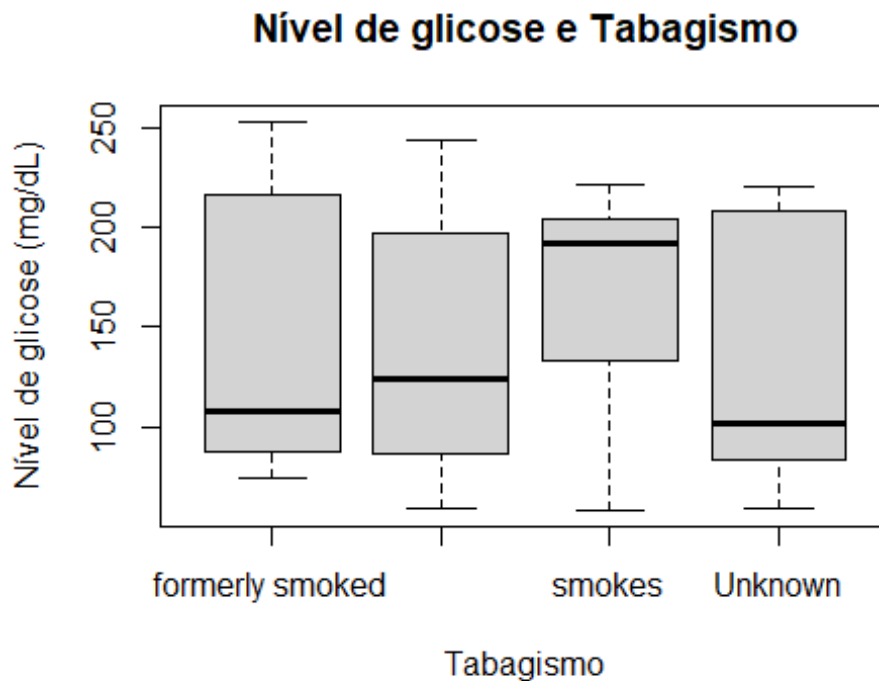
Ok, agora vamos voltar ao boxplot:

```
boxplot(n_fumante$avg_glucose_level, fumante$avg_glucose_level, names = c('Nunca Fumou', 'Fumante'), col = c('orange', 'blue'), main = "Nível de glicose e Tabagismo", ylab = "Nível de glicose (mg/dL)", xlab= "Tabagismo")
```

Como estávamos interessado no nível de glicose, novamente utilizamos o artifício \$avg\_glucose\_level. Na função boxplot colocamos as duas variáveis que estamos interessados, n\_fumante\$avg\_glucose\_level e fumante\$avg\_glucose\_level, se fosse necessário poderíamos colocar mais variáveis. Vejam que podemos colocar o nome e a cor diferente para cada um dos boxplot, para isso precisamos utilizar c(), por exemplo: c('cor1','cor2').

Se estamos interessados em ver todas as opções de 'smoking\_status' podemos fazer de uma forma mais simples, sem precisar criar uma nova variável e separar os dados para cada opção, da seguinte forma:

```
boxplot(dados$avg_glucose_level ~ dados$smoking_status,  
        main = "Nível de glicose e Tabagismo",  
        ylab = "Nível de glicose (mg/dL)",  
        xlab= "Tabagismo")
```



Podemos entender 'dados\$avg\_glucose\_level ~ dados\$smoking\_status' como sendo "estamos vendo o nível de glicose (avg\_glucose\_level) para cada categoria de tabagismo (smoking\_status)". Se queremos mudar o nome e a cor de cada um desses boxplot, como podemos fazer? Tentem fazer.