The Mean

Nos cursos anteriores aprendemos um fluxo de como coletar dados, entender sua estrutura, organizar de maneira a poder ilustrar e analisar esses dados.

Nesse módulo vamos explorar a fundo maneiras de reduzir dados a um único valor que represente as características desse dado. Como por exemplo a média, mediana, moda e etc.

Além disso, vamos analisar a distribuição dos dados, pois um conjunto de dados [3,3,3,3] e [0,3,6,3] tem impactos diferentes e através do cálculo de desvio padrão, conseguimos avaliar essa distribuição.

Pra começar vamos falar da média.

Média Aritmédia, ou apenas Média

Imagine que gostaríamos de reduzir uma distribuição de números variados em uma única métrica que leve em consideração todos os números igualmente. Uma forma é somando todos eles e dividindo pela quantidade de números somados.

```
lista <- c(0,1,4,7,8,10)

0+1+4+7+8+10 # = 30
```

```
## [1] 30
```

```
30/6 #6 representa a quantidade de números somados
```

```
## [1] 5
```

```
#resulta em 5 a média, podemos calcular diretamente através da função
mean(lista)
```

```
## [1] 5
```

Para esse exemplo, temos bons motivos para acreditar que a média foi representativa para a distribuição.

- nosso menor valor é 0 e o maior 10, e a média 5 está bem no centro entre ambos.
- se somarmos a distância dos valores que estão acima da média, contra os que estão abaixo da média, a distância é a mesma.
- os valores estão próximos à média de forma justa, o 4 está a uma unidade da média, enquanto que os extremos 0 e 10 estão a 5 unidades de distâcia.

Vamos olhar esse outro cenário

```
lista  <-  c(0,2,3,3,3,4,13)

# média 4, mas o meio entre 0 e 13 é 6.5, então não está centralizada
media <- mean(lista)
media
```

```
## [1] 4
```

```
#conferindo se a distância dos valores acima e abaixo comparado a média são iguais

valores_acima <- lista[lista > media]
valores_abaixo <- lista[lista < media]

distancia_acima <- valores_acima - media
distancia_abaixo <- media - valores_abaixo

#as ditâncias são iguais
distancia_acima  %>% sum()
```

```
## [1] 9
```

```
distancia_abaixo %>% sum()
```

```
## [1] 9
```

Outra forma de calcular as distâncias

```
distancias <- lista - media
distancias
```

```
## [1] -4 -2 -1 -1 -1  0  9
```

```r
#a soma é zero pois são equivalentes
sum(distancias)
```

```
## [1] 0
```

Apesar do caso acima não ter a média centralizada, os valores existentes na lista tem a mesma distância até a média. São 9 unidades de distância de cada lado da média.

Nós podemos dizer então que a média é um valor localizado num ponto da distribuição onde a soma das distância dos valores abaixo da média são as mesmas dos valores acima da média.

Para ilustrar o comportamento da média, imagine uma balança parecida com uma gangorra com pesos em cada lado, a média é o ponto entre esses dois pesos que melhor equilibra a balança. Se um lado tiver um peso muito maior que o outro, a média estará mais próxima desse peso para equilibrar. Se ambos pesos forem semelhantes, a média vai ser um ponto mais centralizado.

Figura 1: Balança/Gangorra

Geração de amostras para estressar o teste da média

Vamos agora gerar amostras que vão nos apoiar no decorrer do exercício

Já aprendemos em outro momento no curso sobre a função `sample`. Agora vamos usar a função `sample.int` para gerar números aleatórios. É necessário informar qual o valor máximo que pode ser atingido, e o tamanho máximo da lista.

```r
set.seed(1) #conseguimos reproduzir o resultado dessa forma
lista <- sample.int(n = 15, size = 7)
lista
```

```
## [1]  9  4  7  1  2 13 11
```

Se for de interesse criar uma matriz ou dataset com um conjunto de valores, podemos replicar esse código com a função `replicate`

```
matriz <- replicate(n=5, expr = sample.int(n = 15, size = 7))
matriz
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]   11   14    9   12   15
## [2,]    3   10   14    1    4
## [3,]    1    7    5    4   12
## [4,]    5    9   13    3   14
## [5,]   12    5    2    6   10
## [6,]   10   11   10   10    9
## [7,]    6   12   15   11    7
```

Vamos confirmar se de fato a soma das distâncias da média é sempre igual a zero. Com uma função vamos gerar uma amostra aleatória, calcular a média dessa amostra, calcular a distância de cada valor em relação a média, somar as distâncias e comparar se o resultado é zero. Se para todos os testes a soma for zero, o resultado é TRUE. Sendo assim no teste abaixo com 5000 repetições, teremos 5000 TRUEs, ou uma variável com resultado de 5000

```
checkDist <- function(){
  sample <- sample.int(n=1000,size=10)
    mean <- mean(sample)

    distance <- sample - mean
    round(sum(distance)) == 0
}
#Teste Função
checkDist()
```

```
## [1] TRUE
```

```
#replicando o teste para ser executado 5 mil vezes
sample <- sum(replicate(n=5000,expr = checkDist()))
sample
```

```
## [1] 5000
```

Fórmula matemática

Para representar matematicamente as fórmulas da média aqui no código, podemos usufruir de funcionalidades do RMarkdown, para mais detalhes clique **neste link**

A média de uma população pode ser representada da seguinte maneira, usando a letra grega abaixo com a pronúncia “miu”.

$$\mu = \frac{x_1 + x_2 + ...x_N}{N}$$

Então para uma população com os seguintes 7 valores [0,2,3,3,3,4,13], a representação ficaria assim:

$$\mu = \frac{0 + 2 + 3 + 3 + 3 + 4 + 13}{7} = \frac{28}{7} = 4$$

Agora quando queremos calcular a média para uma parte dessa população, em outras palavras, queremos calcular a média para uma amostra, temos que usar outra representação, nesse caso o “x barra”, o n nesse caso fica minúsculo também.

$$\overline{x} = \frac{x_1 + x_2 + ...x_n}{n}$$

Para uma amostra de 3 valores [2,3,4], essa seria a representação:

$$\overline{x} = \frac{2 + 3 + 4}{3} = \frac{9}{3} = 3$$

Existem variações dessas representações:

	População	Amostra
Média	μ	$\overline{x}, \overline{x}_n, \overline{X}, M$
Qtde valores	N	n

Uma forma mais enxuta para representar o cálculo da média de uma população seria assim:

$$\mu = \sum_{i=1}^{N} x_i$$

Essa letra grega parecida com a letra E se chama sigma e ela representa a soma de uma sequência de números. É como se fosse uma estrutura de loop for, onde a cada

iteração *i* vai assumir um número diferente da sequência. *i* = 1 define que a soma iniciará em 1 e irá incrementar 1 a cada iteração. E finalizará quando atingir *N*. Então para uma sequência de 3 números [2,4,6], a soma de x_i vai começar em x_1 que é o primeiro número da sequência nesse caso o 2, e vai somar com o próximo x_2 que representa o 4, assim por diante até atingir *N* que nesse caso é 3, são 3 iterações.

Podemos representar essas fórmulas com o seguinte código:

```
populacao <- c(10, 5, 12)

calcula_media  <- function(populacao){
  N <- length(populacao)
  soma_populacao = 0
  for ( i in 1:N) {
      soma_populacao <- soma_populacao + populacao[i]
  }

  media <- soma_populacao / N
}

media <- calcula_media(populacao)
media
```

```
## [1] 9
```

Então recapitulando... Notação de como representar média da população passo a passo

$$\mu = \frac{x_1 + x_2 + ...x_N}{N} = \frac{\sum X}{N} = \frac{\sum_{i=1}^{N} x_i}{N}$$

Já a média da amostra

$$\overline{x} = \frac{x_1 + x_2 + ...x_n}{n} = \frac{\sum X}{n} = \frac{\sum_{i=1}^{n} x_i}{n}$$

Calculando Média em grandes conjuntos de dados

Agora vamos falar mais sobre a média usando dados reais, nessa base temos 2930 linhas com 82 colunas contendo informações de características de casas vendidas entre 2006 e 2010 na cidade Ames (estado de Iowa nos EUA).

Esse foi um trabalho feito pelo professor Dean DeCock, publicado neste artigo e os detalhes sobre as informações presentes na base estão neste link

O separador da base são tabs, é um arquivo do tipo TSV (tab-separated value), são basicamente espaços. Poderíamos usar a função read.csv e informar o parâmetro sep= "\t" que funcionaria da mesma forma.

```
base <- read_tsv("https://s3.amazonaws.com/dq-content/444/AmesHousing.txt")
```

```
## Rows: 2930 Columns: 82
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (45): PID, MS SubClass, MS Zoning, Street, Alley, Lot Shape, Land Contou...
## dbl (37): Order, Lot Frontage, Lot Area, Overall Qual, Overall Cond, Year Bu...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
glimpse(base)
```

```
## Rows: 2,930
## Columns: 82
## $ Order             <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 1~
## $ PID               <chr> "0526301100", "0526350040", "0526351010", "052635303~
## $ `MS SubClass`     <chr> "020", "020", "020", "020", "060", "060", "120", "12~
## $ `MS Zoning`       <chr> "RL", "RH", "RL", "RL", "RL", "RL", "RL", "RL", "RL"~
## $ `Lot Frontage`    <dbl> 141, 80, 81, 93, 74, 78, 41, 43, 39, 60, 75, NA, 63,~
## $ `Lot Area`        <dbl> 31770, 11622, 14267, 11160, 13830, 9978, 4920, 5005,~
## $ Street            <chr> "Pave", "Pave", "Pave", "Pave", "Pave", "Pave", "Pav~
## $ Alley             <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ `Lot Shape`       <chr> "IR1", "Reg", "IR1", "Reg", "IR1", "IR1", "Reg", "IR~
## $ `Land Contour`    <chr> "Lvl", "Lvl", "Lvl", "Lvl", "Lvl", "Lvl", "Lvl", "HL~
## $ Utilities         <chr> "AllPub", "AllPub", "AllPub", "AllPub", "AllPub", "A~
## $ `Lot Config`      <chr> "Corner", "Inside", "Corner", "Corner", "Inside", "I~
## $ `Land Slope`      <chr> "Gtl", "Gtl", "Gtl", "Gtl", "Gtl", "Gtl", "Gtl", "Gt~
## $ Neighborhood      <chr> "NAmes", "NAmes", "NAmes", "NAmes", "Gilbert", "Gilb~
## $ `Condition 1`     <chr> "Norm", "Feedr", "Norm", "Norm", "Norm", "Norm", "No~
## $ `Condition 2`     <chr> "Norm", "Norm", "Norm", "Norm", "Norm", "Norm", "Nor~
## $ `Bldg Type`       <chr> "1Fam", "1Fam", "1Fam", "1Fam", "1Fam", "1Fam", "Twn~
## $ `House Style`     <chr> "1Story", "1Story", "1Story", "1Story", "2Story", "2~
## $ `Overall Qual`    <dbl> 6, 5, 6, 7, 5, 6, 8, 8, 8, 7, 6, 6, 6, 7, 8, 8, 8, 9~
## $ `Overall Cond`    <dbl> 5, 6, 6, 5, 5, 6, 5, 5, 5, 5, 5, 7, 5, 5, 5, 5, 7, 2~
## $ `Year Built`      <dbl> 1960, 1961, 1958, 1968, 1997, 1998, 2001, 1992, 1995~
## $ `Year Remod/Add`  <dbl> 1960, 1961, 1958, 1968, 1998, 1998, 2001, 1992, 1996~
## $ `Roof Style`      <chr> "Hip", "Gable", "Hip", "Hip", "Gable", "Gable", "Gab~
```

```
## $ `Roof Matl`       <chr> "CompShg", "CompShg", "CompShg", "CompShg", "CompShg~
## $ `Exterior 1st`    <chr> "BrkFace", "VinylSd", "Wd Sdng", "BrkFace", "VinylSd~
## $ `Exterior 2nd`    <chr> "Plywood", "VinylSd", "Wd Sdng", "BrkFace", "VinylSd~
## $ `Mas Vnr Type`    <chr> "Stone", "None", "BrkFace", "None", "None", "BrkFace~
## $ `Mas Vnr Area`    <dbl> 112, 0, 108, 0, 0, 20, 0, 0, 0, 0, 0, 0, 0, 0, 0, 60~
## $ `Exter Qual`      <chr> "TA", "TA", "TA", "Gd", "TA", "TA", "Gd", "Gd", "Gd"~
## $ `Exter Cond`      <chr> "TA", "TA", "TA", "TA", "TA", "TA", "TA", "TA", "TA"~
## $ Foundation        <chr> "CBlock", "CBlock", "CBlock", "CBlock", "PConc", "PC~
## $ `Bsmt Qual`       <chr> "TA", "TA", "TA", "TA", "Gd", "TA", "Gd", "Gd", "Gd"~
## $ `Bsmt Cond`       <chr> "Gd", "TA", "TA", "TA", "TA", "TA", "TA", "TA", "TA"~
## $ `Bsmt Exposure`   <chr> "Gd", "No", "No", "No", "No", "No", "Mn", "No", "No"~
## $ `BsmtFin Type 1`  <chr> "BLQ", "Rec", "ALQ", "ALQ", "GLQ", "GLQ", "GLQ", "AL~
## $ `BsmtFin SF 1`    <dbl> 639, 468, 923, 1065, 791, 602, 616, 263, 1180, 0, 0,~
## $ `BsmtFin Type 2`  <chr> "Unf", "LwQ", "Unf", "Unf", "Unf", "Unf", "Unf", "Un~
## $ `BsmtFin SF 2`    <dbl> 0, 144, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1120, 0,~
## $ `Bsmt Unf SF`     <dbl> 441, 270, 406, 1045, 137, 324, 722, 1017, 415, 994, ~
## $ `Total Bsmt SF`   <dbl> 1080, 882, 1329, 2110, 928, 926, 1338, 1280, 1595, 9~
## $ Heating           <chr> "GasA", "GasA", "GasA", "GasA", "GasA", "GasA", "Gas~
## $ `Heating QC`      <chr> "Fa", "TA", "TA", "Ex", "Gd", "Ex", "Ex", "Ex", "Ex"~
## $ `Central Air`     <chr> "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y~
## $ Electrical        <chr> "SBrkr", "SBrkr", "SBrkr", "SBrkr", "SBrkr", "SBrkr"~
## $ `1st Flr SF`      <dbl> 1656, 896, 1329, 2110, 928, 926, 1338, 1280, 1616, 1~
## $ `2nd Flr SF`      <dbl> 0, 0, 0, 0, 701, 678, 0, 0, 0, 776, 892, 0, 676, 0, ~
## $ `Low Qual Fin SF` <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ `Gr Liv Area`     <dbl> 1656, 896, 1329, 2110, 1629, 1604, 1338, 1280, 1616,~
## $ `Bsmt Full Bath`  <dbl> 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1~
## $ `Bsmt Half Bath`  <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ `Full Bath`       <dbl> 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 3, 2, 1~
## $ `Half Bath`       <dbl> 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1~
## $ `Bedroom AbvGr`   <dbl> 3, 2, 3, 3, 3, 3, 2, 2, 2, 3, 3, 3, 3, 2, 1, 4, 4, 1~
## $ `Kitchen AbvGr`   <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
## $ `Kitchen Qual`    <chr> "TA", "TA", "Gd", "Ex", "TA", "Gd", "Gd", "Gd", "Gd"~
## $ `TotRms AbvGrd`   <dbl> 7, 5, 6, 8, 6, 7, 6, 5, 5, 7, 7, 6, 7, 5, 4, 12, 8, ~
## $ Functional        <chr> "Typ", "Typ", "Typ", "Typ", "Typ", "Typ", "Typ", "Ty~
## $ Fireplaces        <dbl> 2, 0, 0, 2, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1~
## $ `Fireplace Qu`    <chr> "Gd", NA, NA, "TA", "TA", "Gd", NA, NA, "TA", "TA", ~
## $ `Garage Type`     <chr> "Attchd", "Attchd", "Attchd", "Attchd", "Attchd", "A~
## $ `Garage Yr Blt`   <dbl> 1960, 1961, 1958, 1968, 1997, 1998, 2001, 1992, 1995~
## $ `Garage Finish`   <chr> "Fin", "Unf", "Unf", "Fin", "Fin", "Fin", "Fin", "RF~
## $ `Garage Cars`     <dbl> 2, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 2, 3~
## $ `Garage Area`     <dbl> 528, 730, 312, 522, 482, 470, 582, 506, 608, 442, 44~
## $ `Garage Qual`     <chr> "TA", "TA", "TA", "TA", "TA", "TA", "TA", "TA", "TA"~
## $ `Garage Cond`     <chr> "TA", "TA", "TA", "TA", "TA", "TA", "TA", "TA", "TA"~
## $ `Paved Drive`     <chr> "P", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y~
```

```
## $ `Wood Deck SF`    <dbl> 210, 140, 393, 0, 212, 360, 0, 0, 237, 140, 157, 483~
## $ `Open Porch SF`   <dbl> 62, 0, 36, 0, 34, 36, 0, 82, 152, 60, 84, 21, 75, 0,~
## $ `Enclosed Porch`  <dbl> 0, 0, 0, 0, 0, 0, 170, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ `3Ssn Porch`      <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ `Screen Porch`    <dbl> 0, 120, 0, 0, 0, 0, 0, 144, 0, 0, 0, 0, 0, 0, 140, 2~
## $ `Pool Area`       <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ `Pool QC`         <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ Fence             <chr> NA, "MnPrv", NA, NA, "MnPrv", NA, NA, NA, NA, NA, NA~
## $ `Misc Feature`    <chr> NA, NA, "Gar2", NA, NA, NA, NA, NA, NA, NA, NA, "She~
## $ `Misc Val`        <dbl> 0, 0, 12500, 0, 0, 0, 0, 0, 0, 0, 0, 500, 0, 0, 0, 0~
## $ `Mo Sold`         <dbl> 5, 6, 6, 4, 3, 6, 4, 1, 3, 6, 4, 3, 5, 2, 6, 6, 6, 6~
## $ `Yr Sold`         <dbl> 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010, 2010~
## $ `Sale Type`       <chr> "WD", "WD", "WD", "WD", "WD", "WD", "WD", "WD", "WD"~
## $ `Sale Condition`  <chr> "Normal", "Normal", "Normal", "Normal", "Normal", "N~
## $ SalePrice         <dbl> 215000, 105000, 172000, 244000, 189900, 195500, 2135~
```

Agora vamos dizer que estamos interessados em entender a distribuição do preço de venda das casas, coluna SalePrice. Com a função summary já conseguimos ter uma boa noção do comportamento.

```
base$SalePrice %>% summary(digits = 8)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   12789  129500  160000  180796  213500  755000
```

Aqui já podemos ver que o menor valor de uma casa nessa base foi de $12,789, enquanto que o maior valor atingiu $755,00, já a média ficou em $180,796.

Para calcular apenas a média poderíamos usar a função mean

```
base$SalePrice %>% mean()
```

```
## [1] 180796.1
```

Usando a função criada durante o exercício chegamos no mesmo resultado.

```
media <- calcula_media(base$SalePrice)
media
```

```
## [1] 180796.1
```

Estimando a média da população com base na amostra

Na maioria das vezes buscamos responder perguntas durante uma análise que diz respeito à população como um todo, porém muitas vezes trabalhamos apenas com uma parte dessa população, uma amostra. E o que podemos esperar é que calculando a média dessa amostra ela seja próxima à amostra da população.

Nesse caso pode acontecer 3 situações.

- de fato a média da amostra é igual à população ($\overline{x} = \mu$)
- a média da amostra superestima a média da população ($\overline{x} > \mu$)
- a média da amostra subestima a média da população ($\overline{x} < \mu$)

Nos casos que a média da população vs amostra não são iguais, temos erros de amostragem. O erro de amostragem é uma diferença entre o parâmetro da população e a estatística da amostra. μ é o parâmetro e $\overline{x}$ é a estatística.

Sendo assim o erro amostral é a diferença entre essas duas informações:

$$sampling\ error = \mu - \overline{x}$$

A ideia é sempre diminuir esse erro amostral, e tem 2 fatores que influenciam isso:

- **a representatividade da amostra**: quanto mais representativo, quanto mais a amostra tiver características da população, menor o erro amostral
- **o tamanho da amostra**: quanto maior o tamanho da amostra, mais chances de obter uma amostra representativa

Vamos seguir assumindo que nossa base é a população completa com propósito de exercitar essa questão. E assim a partir dessa população vamos criar uma série de amostras e comparar num gráfico o comportamento dessa média.

A ideia é tentar ilustrar como o tamanho da amostra afeta o comportamento da média, então vamos amostrar tamanhos diferentes começando em tamanhos bem pequenos até chegar no tamanho da população.

```
parametro <-  mean(base$SalePrice)

#vamos começar com amostras de 5 registros
#e ir incrementando de 29 em 29 até ter 100 tamanhos diferentes
tamanho_amostras <- seq(5,by=29, length.out=100)
tamanho_amostras
```

```
##   [1]    5   34   63   92  121  150  179  208  237  266  295  324  353  382  411
##  [16]  440  469  498  527  556  585  614  643  672  701  730  759  788  817  846
##  [31]  875  904  933  962  991 1020 1049 1078 1107 1136 1165 1194 1223 1252 1281
##  [46] 1310 1339 1368 1397 1426 1455 1484 1513 1542 1571 1600 1629 1658 1687 1716
##  [61] 1745 1774 1803 1832 1861 1890 1919 1948 1977 2006 2035 2064 2093 2122 2151
##  [76] 2180 2209 2238 2267 2296 2325 2354 2383 2412 2441 2470 2499 2528 2557 2586
##  [91] 2615 2644 2673 2702 2731 2760 2789 2818 2847 2876
```

```
#para cada amostra vamos calcular a média estatística
#e comparar com o parâmertro
#assim encontramos o erro amostral para cada amostra
erros_amostrais <- map_dbl(tamanho_amostras,
                           ~ parametro - mean(sample(base$SalePrice,size=.)))
erros_amostrais
```

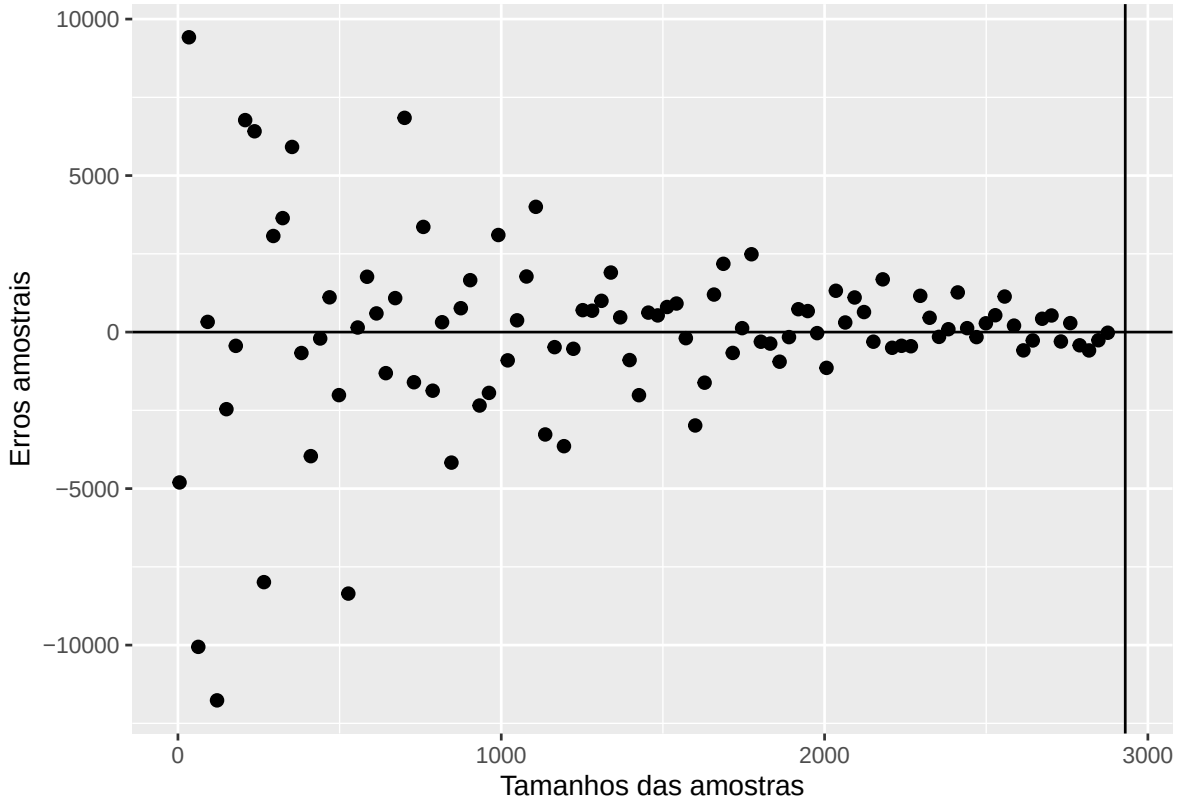
```
##   [1]  -4803.93993   9418.73654 -10056.44787    325.62529 -11765.90687
##   [6]  -2463.06660   -439.84496   6772.19468   6415.05585  -7988.84219
##  [11]   3069.83634   3640.76686   5914.68613   -668.58653  -3963.50198
##  [16]   -205.36266   1110.90868  -2014.31744  -8354.55853    146.43237
##  [21]   1767.76947    596.06333  -1312.40961   1085.27435   6844.09858
##  [26]  -1602.37829   3359.95994  -1872.60998    315.17145  -4170.40447
##  [31]    762.89664   1658.21936  -2346.48656  -1942.82455   3101.02576
##  [36]   -903.18895    376.13156   1775.66953   4001.48825  -3270.69257
##  [41]   -481.51332  -3645.16606   -533.88433    700.51374    682.49410
##  [46]    998.48755   1901.87261    470.80276   -894.20550  -2016.63488
##  [51]    622.18584    532.54592    801.31387    911.97252   -194.24547
##  [56]  -2983.54243  -1616.48075   1198.79891   2181.10571   -665.77327
##  [61]    125.60505   2486.61024   -308.16678   -368.29037   -946.21935
##  [66]   -161.04840    730.64371    671.14631    -33.20093  -1143.28589
##  [71]   1320.08955    306.79747   1105.59089    639.81737   -307.20446
##  [76]   1684.93713   -501.48000   -437.97925   -451.91611   1158.85754
##  [81]    457.47254   -152.77086     91.41760   1267.09282    125.62787
##  [86]   -160.60147    280.92762    538.20208   1135.54071    206.87484
##  [91]   -584.32540   -267.66686    425.63358    530.42794   -300.83338
##  [96]    286.31876   -422.25259   -586.07230   -261.13768    -21.20940
```

```
df <- tibble(x = tamanho_amostras,
             y = erros_amostrais)
df
```

```
## # A tibble: 100 x 2
##        x       y
```

```
##    <dbl>   <dbl>
##  1     5  -4804.
##  2    34   9419.
##  3    63 -10056.
##  4    92    326.
##  5   121 -11766.
##  6   150  -2463.
##  7   179   -440.
##  8   208   6772.
##  9   237   6415.
## 10   266  -7989.
## # i 90 more rows
```

```
#vamos criar um scatter plot, um gráfico de pontos,
#onde cada ponto representa o erro amostral daquela amostra
# no eixo x vamos ver a evolução desse erro amostral do menor tamanho de amostra
#até chegar no maior tamanho que mais se aproxima da população original
#vamos criar uma linha no eixo x que representa o tamanho da população como base
# e no eixo y vamos deixar uma linha no valor 0 que
#seria a amostra "perfeita" sem erros
df %>%
  ggplot(aes(x = tamanho_amostras,y = erros_amostrais))+
  geom_point(size=2)+
  geom_hline(yintercept=0)+
  geom_vline(xintercept=2930)+
  labs(x = "Tamanhos das amostras",
       y = "Erros amostrais")
```



Com o gráfico acima conseguimos evidenciar como o tamanho da amostra influencia diretamente na representatividade da amostra, e quanto mais representativa menor o erro amostral, pois é muito próximo das características originais da população. Há sim exceções, vemos no gráfico de alguns casos de amostras com tamanhos menores que conseguem um bom resultado no erro amostral.

Então o quanto podemos confiar em amostra pequenas? Quais as chances de amostras pequenas acertarem a média da população? Podemos fazer um teste apenas com amostras pequenas e jogar o resultado num histograma.

```
set.seed(1)
library(scales)
```

```
##
## Attaching package: 'scales'
```

```
## The following object is masked from 'package:purrr':
##
##     discard
```

```
## The following object is masked from 'package:readr':
```

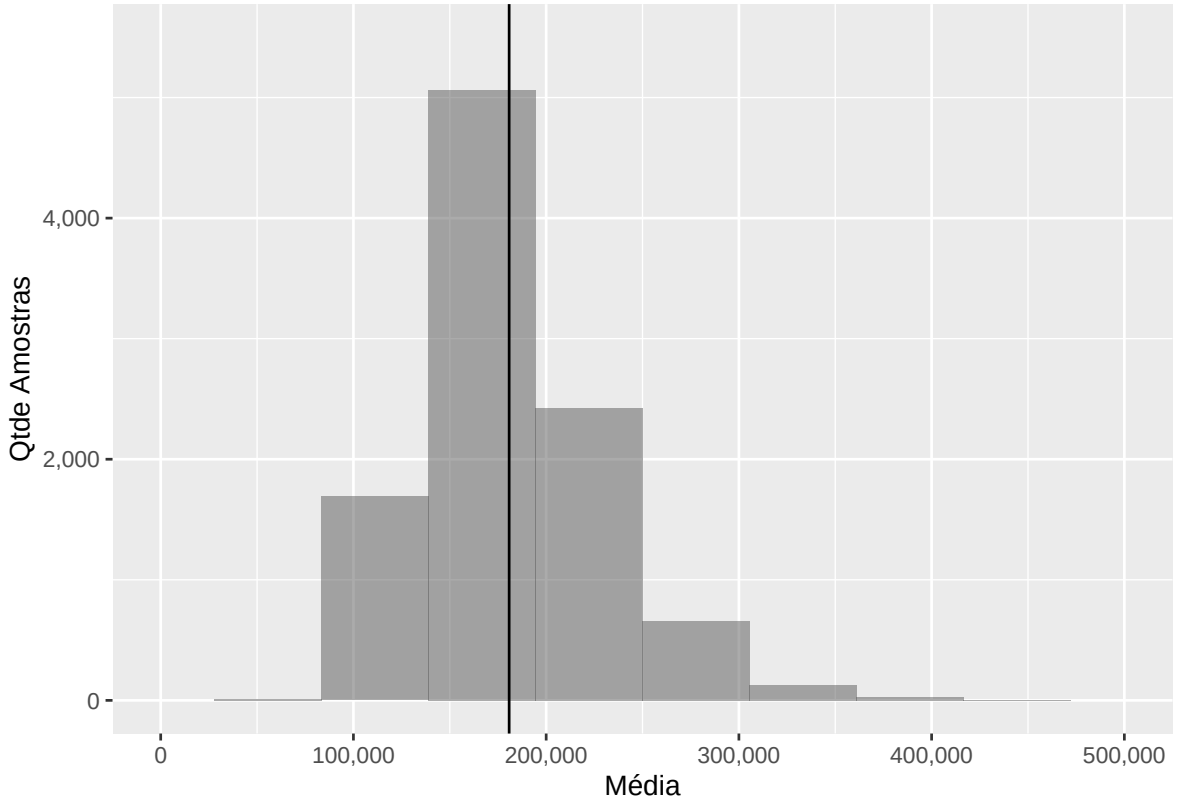
```
##
##     col_factor
```

```
media_amostras <- replicate(n=10000,
                            expr = sample(base$SalePrice,3) %>% mean) %>% tibble()
media_amostras %>% head()
```

```
## # A tibble: 6 x 1
##         .
##     <dbl>
## 1 148667.
## 2 179733.
## 3 150667.
## 4 139500
## 5 215826.
## 6 282387.
```

```
media_amostras %>%
  ggplot(aes(x=.))+
  geom_histogram(bins = 10, position = "identity", alpha=0.5) +
  geom_vline(aes(xintercept = mean(base$SalePrice))) +
  scale_x_continuous(labels = comma, lim=c(0, 500000))+
  scale_y_continuous(labels = comma, lim=c(0, 5500))+
  xlab("Média")+
  ylab("Qtde Amostras")
```

```
## Warning: Removed 2 rows containing missing values (`geom_bar()`).
```



Com o resultado do gráfico acima, é possível notar que mesmo selecionando uma amostra muito pequena, com 3 registros, das 10000 amostras criadas, a grande maioria fica com um resultado próximo à média da população.

Aumentando a quantidade de registros em cada amostra para 100 por exemplo, o resultado fica ainda melhor.

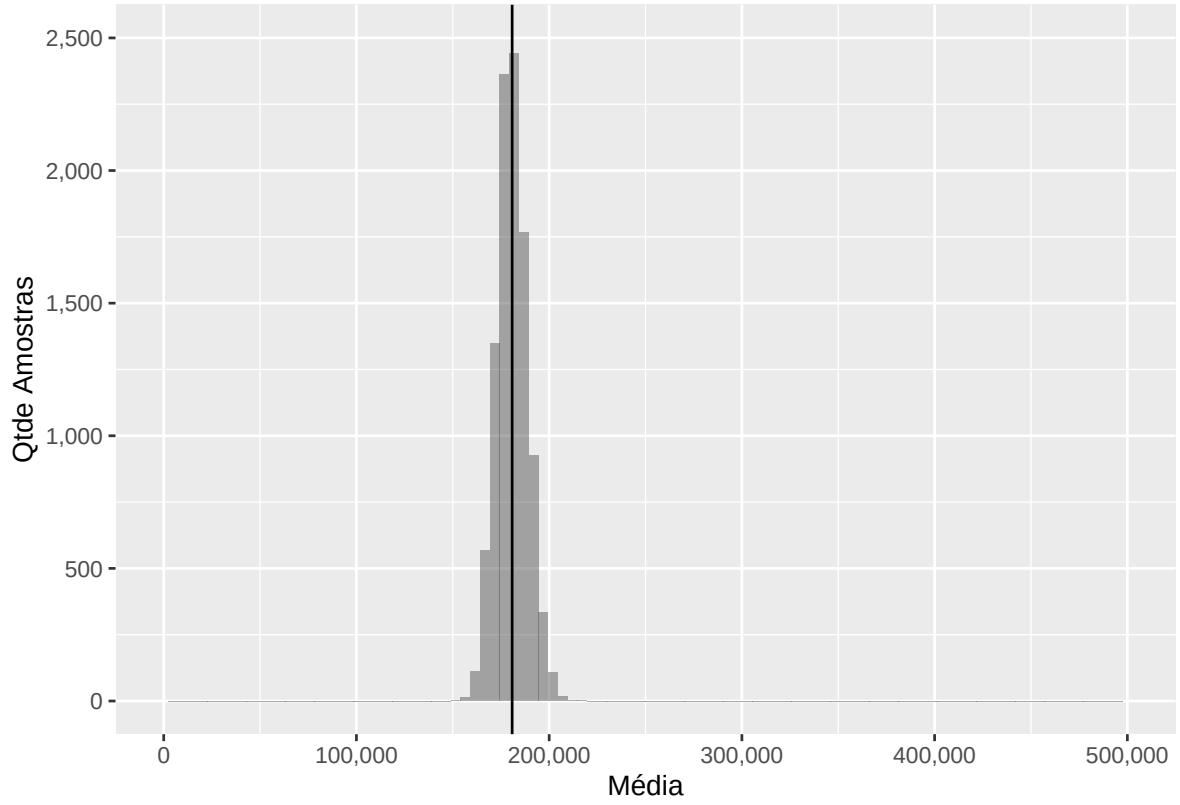
```
set.seed(1)
library(scales)

media_amostras <- replicate(n=10000,
                            expr = sample(base$SalePrice,100) %>% mean) %>% tibble()
media_amostras %>% head()
```

```
## # A tibble: 6 x 1
##         .
##     <dbl>
## 1 189092.
## 2 192530.
## 3 182741.
## 4 181951.
```

```
## 5 185087.
## 6 187398.
```

```
media_amostras %>%
  ggplot(aes(x=.))+
  geom_histogram(bins = 100, position = "identity", alpha=0.5) +
  geom_vline(aes(xintercept = mean(base$SalePrice))) +
  scale_x_continuous(labels = comma, lim=c(0, 500000))+
  scale_y_continuous(labels = comma, lim=c(0, 2500))+
  xlab("Média")+
  ylab("Qtde Amostras")
```

```
## Warning: Removed 2 rows containing missing values (`geom_bar()`).
```



A Média das amostras usada como indicador de viés

Se a gente gerar todas as possibilidades de amostra que uma população permite, calcular a média dessas amostras, e calcular a média dessas médias, vamos observar que o resultado é igual à média da população.

Para exemplificar, vamos criar um exemplo super simples, uma população de 3 valores e amostras de 2 valores.

População : [3,7,2]

```
populacao  <-  c(3, 7, 2)

amostras <- list(c(3,7),
                 c(3,2),
                 c(7,2),
                 c(7,3),
                 c(2,3),
                 c(2,7))

media_amostras <- map_dbl(amostras, function(x) mean(x))

media_populacao<- mean(populacao)
media_populacao
```

```
## [1] 4
```

```
media_da_media_amostras <- mean(media_amostras)
media_da_media_amostras
```

```
## [1] 4
```

```
sem_vies <- (media_populacao == media_da_media_amostras)

sem_vies
```

```
## [1] TRUE
```