Frequency Distributions

Ao coletar dados, nós os estruturamos, entendemos como medí-los, os transformamos e etc. Mas o objetivo real é extrair algum valor deles, usar na tomada de decisões e não somente coletar por coletar.

Pode ser que queremos analisar os dados para:

- Descrever um fenômeno sobre o mundo (Ciência)
- Tomar melhor decisões (indústria)
- Aprimorar sistemas (engenharia)
- Descrever diferentes aspectos da sociedade (jornalismo), e etc

Estamos num mundo em que cada vez temos mais e mais dados e o volume é tão imenso que se torna incompreensível olhar cada informação individualmente. Para isso, temos que simplificar os dados, pois apenas analisa-los numa tabela diminui nossa capacidade de entender a medida que a quantidade de observações aumenta.

Nessa base de dados que estamos lidando com 143 observações, apesar de não ser uma base extremamente grande, já é complicada de "dar uma olhada" para extrair qualquer insight.

Explorando dados do WNBA

O dataset pode ser acessado através deste link.

E o glossário dos termos neste link.

```
wnba <- read_csv("WNBA_Stats.csv")
```

```
## Rows: 143 Columns: 32
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): Name, Team, Pos, Birth_Place, Birthdate, College, Experience
## dbl (25): Height, Weight, BMI, Age, Games Played, MIN, FGM, FGA, FG%, 15:00,...
##
## 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.
```

Tabelas de Frequência de Distribuição

Uma forma de simplificar é criando tabelas de frequência de distribuição, que é nada mais que contar quantas vezes um valor único ocorre na base de dados.

Na esquerda ficam os valores únicos, aqui por exemplo o tipo de posição de cada jogadora, e na direita quantas vezes se repete (frequência), quantas jogadoras são Guards, ou Fowards, etc. e assim entender como a base está distribuida, fazer comparações e analisar os resultados.

```
wnba %>%
  group_by(Pos) %>%
  summarise(qtd = n())
```

```
## # A tibble: 5 x 2
##   Pos     qtd
##   <chr> <int>
## 1 C        25
## 2 F        33
## 3 F/C      12
## 4 G        60
## 5 G/F      13
```

Uma outra forma de ver essa informação é usando a função `table` Apesar de uma alternativa útil e simples, a abordagem com group_by pode ser muito mais completa e flexivel de usar.

```
tabela_freq <- table(wnba$Pos)
tabela_freq
```

```
##
##   C   F F/C   G G/F
##  25  33  12  60  13
```

E para estruturar em formato de tabela:

```
tabela_freq %>% as.data.frame()
```

```
##   Var1 Freq
## 1    C   25
## 2    F   33
## 3  F/C   12
## 4    G   60
## 5  G/F   13
```

Ainda pensando em simplificar a análise, podemos ordenar a informação e já descobrir qual é a posição que concentra maior quantidade de jogadoras.

```
wnba %>%
  group_by(Pos) %>%
  summarise(qtd = n()) %>%
  arrange(desc(qtd))
```

```
## # A tibble: 5 x 2
##   Pos     qtd
##   <chr> <int>
## 1 G        60
## 2 F        33
## 3 C        25
## 4 G/F      13
## 5 F/C      12
```

Para ordenar informações categóricas que possuem uma ordem lógica não necessariamente alfabética, podemos usar **fatores**.

Por exemplo se criarmos uma categoria que indique se a jogadora é baixa ou alta, em ordem alfabética alta seria a primeira informação, mas pensando no significado da informação baixo deveria ser a primeira informação a aparecer.

```
wnba <- wnba %>%
  mutate(Height_labels = case_when(
    Height <= 170 ~ "baixa",
    Height > 170 & Height <= 180 ~ "media",
    Height > 180 ~ "alta"
  ))

wnba %>%
  group_by(Height_labels) %>%
  summarise(qtd = n()) %>%
  arrange(Height_labels)
```

```
## # A tibble: 3 x 2
##   Height_labels   qtd
##   <chr>         <int>
## 1 alta             92
## 2 baixa             9
## 3 media            42
```

Com o uso de fatores:

```
ordem <- c("baixa", "media", "alta")

wnba %>%
  group_by(Height_labels) %>%
  summarise(qtd = n()) %>%
  arrange(factor(Height_labels,
                 level = ordem))
```

```
## # A tibble: 3 x 2
##   Height_labels   qtd
##   <chr>         <int>
## 1 baixa             9
## 2 media            42
## 3 alta             92
```

Voltando na informação de Posição das Jogadoras, olhar a frequência absoluta, a quantidade exata de jogadoras em cada posição, pode não ser tão intuitivo. Calcular o percentual ou a proporção de quantas jogadoras tem em cada posição em relação ao total de jogadoras pode trazer uma noção melhor de distribuição das jogadoras.

```
wnba %>%
  group_by(Pos) %>%
  summarise(qtd = n()) %>%
  arrange(desc(qtd)) %>%
  mutate(prop = qtd/nrow(wnba),
         perc = qtd/nrow(wnba)*100)
```

```
## # A tibble: 5 x 4
##   Pos     qtd   prop  perc
##   <chr> <int>  <dbl> <dbl>
## 1 G        60 0.420  42.0
## 2 F        33 0.231  23.1
## 3 C        25 0.175  17.5
## 4 G/F      13 0.0909  9.09
## 5 F/C      12 0.0839  8.39
```

A proporção é um numéro de 0 a 1, enquanto o percentual está numa escala de 0 a 100%. Podemos entender que cerca de 40% da base é formada por jogadoras da posição Guards, quase metade das observações.

Podemos também olhar uma condição específica para calcular essa proporção e percentual. No exemplo abaixo verificamos quantas jogadoras tem até 23 anos de idade e qual é percentual que isso representa.

```
wnba %>%
  filter(Age <= 23) %>%
  summarise(qtd = n()) %>%
  arrange(desc(qtd)) %>%
  mutate(prop = qtd/nrow(wnba),
         perc = qtd/nrow(wnba)*100)
```

```
## # A tibble: 1 x 3
##     qtd  prop  perc
##   <int> <dbl> <dbl>
## 1    27 0.189  18.9
```

De forma ainda mais simples, podemos fazer dessa forma:

```
mean(wnba$Age <= 23)
```

```
## [1] 0.1888112
```

```
mean(wnba$Age <= 23) * 100
```

```
## [1] 18.88112
```

Percentile Rank

Percentil é uma medida usada para dividir uma amostra de valores em 100 partes (daí a similaridade com a palavra Percentual). O cálculo que realizamos acima nos diz que 18% da amostra da base são jogadoras com até 23 anos, e portanto, nos ajuda a compreender a distribuição das informações na base.

Ainda assim temos uma função para fazer esse cálculo de forma ainda mais simplificada. A função `summary` traz já num único resultado e menor e maior valor, a mediana e a média, e o primeiro e terceiro quarter.

Quarter é nada mais que quartos, a base foi dividida em 4 partes. O primeiro quarter é 24, então significa que 25% da base tem até 24 anos.

```r
summary(wnba$Age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   21.00   24.00   27.00   27.08   30.00   36.00
```

Outra forma de enxergar somente os percentuais é com a função `quantile` que por padrão também divide a base em quarters.

```r
quantile(wnba$Age)
```

```
##   0%  25%  50%  75% 100%
##   21   24   27   30   36
```

O que notamos são 3 pontos de cortes que separam os 4 quartos. A idade 24 é o que separa os percentis de 0 a 25% dos percentis de 25 a 50%. Da mesma forma acontece com os pontos de corte 27 e 30.

Essa função é flexível e nos permite determinar os percentuais em que queremos os pontos de corte.

```r
quantile(wnba$Age,
         probs = c(0,0.1,0.25,0.30,0.66,0.72,0.89,1))
```

```
##   0%  10%  25%  30%  66%  72%  89% 100%
## 21.0 23.0 24.0 24.6 28.0 29.0 32.0 36.0
```

Um percentil não tem uma definição padronizada, e dependendo da função o resultado pode ser um pouco diferente. Por exemplo logo abaixo o cálculo do percentil até a idade de 24 anos resulta em 30% sendo que nos testes acima o resultado foi 25%. Isso depende muito da concentração e variação dos dados na base em análise.

```r
mean(wnba$Age <= 24)
```

```
## [1] 0.3006993
```

Dependendo da situação pode ser interessante olhar as observações no detalhe e calcular os percentis para cada uma delas. No exemplo abaixo notamos que todas jogadoras com 23 anos tem a distribuição acumulada de 18%, ou em outras palavras, 18% da base tem até 23 anos.

```r
wnba %>%
  mutate(cume_dist_age = cume_dist(Age)) %>%
  select(Name, Age,cume_dist_age) %>%
  head()
```

```
## # A tibble: 6 x 3
##   Name              Age cume_dist_age
##   <chr>           <dbl>         <dbl>
## 1 Aerial Powers      23        0.189
## 2 Alana Beard        35        0.993
## 3 Alex Bentley       26        0.490
## 4 Alex Montgomery    28        0.678
## 5 Alexis Jones       23        0.189
## 6 Alexis Peterson    22        0.0839
```

Granularidade dos dados

Dependendo da informação numérica e continua que estamos trabalhando, a granularidade dos dados pode ser tão alta que a abordagem que adotamos até agora não permitiria analisar muito bem essa distribuição.

```r
wnba %>%
  group_by(Weight) %>%
  summarize(Freq = n()) %>%
  head(15)
```

```
## # A tibble: 15 x 2
##    Weight  Freq
##     <dbl> <int>
##  1     55     1
##  2     57     1
##  3     58     1
##  4     59     2
##  5     62     1
##  6     63     3
##  7     64     5
##  8     65     4
##  9     66     8
## 10     67     1
## 11     68     2
## 12     69     2
```

```
## 13     70     3
## 14     71     2
## 15     73     6
```

Sendo assim pode fazer mais sentido transformar a informação contínua em categórica através da criação de faixas de intervalos.

```
wnba <- wnba %>%
  mutate(weight_categories = cut(Weight, breaks = 10, dig.lab = 4))

wnba %>%
  group_by(weight_categories) %>%
  summarise(Freq = n())
```

```
## # A tibble: 11 x 2
##    weight_categories  Freq
##    <fct>             <int>
##  1 (54.94,60.8]          5
##  2 (60.8,66.6]          21
##  3 (66.6,72.4]          10
##  4 (72.4,78.2]          33
##  5 (78.2,84]            31
##  6 (84,89.8]            24
##  7 (89.8,95.6]          10
##  8 (95.6,101.4]          3
##  9 (101.4,107.2]         2
## 10 (107.2,113.1]         3
## 11 <NA>                  1
```

Com a função `cut` é possível dividir a distribuição das informações de forma padronizada. Acima dividimos a base em 10 partes iguais (breaks). Portanto o valor (54.94,60.8] é um intervalo que abrange valores de 54.94 a 60.8, sendo que o parênteses "(" indica que o valor 54.94 não está incluso nessa faixa, ou seja o intervalo se inicia na verdade em 54.95. Enquanto que o conchetes "]" indica que o valor 60.8 está incluso nessa faixa, mas não na faixa seguinte (60.8,66.6] que inicia com parênteses.

Podemos ver que os pontos de corte tem 5,8 de diferença. Portanto se somar 54.94 + 5.8 é aproximadamente 60.8 e assim por diante. O parâmetro dig.lab serve para informar quantos dígitos decimais desejamos exibir, se o valor tiver mais casas decimais do que informado no parâmetro a visualização é convertida em notação científica.

O que montamos se trata de uma **tabela de distribuição de frequências com classe** (grouped frequency distribution table em inglês)

É uma alternativa para analisar dados com alta granularidade.

Perda de informação

Quando realizamos esse agrupamento dos dados é natural ter perda de informação, podemos aumentar a granularidade dos dados para diminuir o efeito dessa perda, mas com cuidado para evitar aumentar muito o tamanho da tabela e tornar difícil a compreensão dos dados da mesma forma que é difícil analisar a tabela original sem nenhum agrupamento.

Usando 2 quebras fica difícil de entender se as 100 jogadoras com peso entre 54 e 84 estão mais concentradas no menor ou no maior peso do intervalo.

```
wnba <- wnba %>%
  mutate(weight_categories = cut(Weight, breaks = 2, dig.lab = 4))

wnba %>%
  group_by(weight_categories) %>%
  summarize(Freq = n())
```

```
## # A tibble: 3 x 2
##   weight_categories  Freq
##   <fct>             <int>
## 1 (54.94,84]          100
## 2 (84,113.1]           42
## 3 <NA>                  1
```

Muitas quebras fica dificil de analisar e chegar numa conclusão.

```
wnba <- wnba %>%
  mutate(weight_categories = cut(Weight, breaks = 30, dig.lab = 4))

wnba %>%
  group_by(weight_categories) %>%
  summarize(Freq = n())
```

```
## # A tibble: 26 x 2
##    weight_categories  Freq
##    <fct>             <int>
```

```
##  1 (54.94,56.93]         1
##  2 (56.93,58.87]         2
##  3 (58.87,60.8]          2
##  4 (60.8,62.73]          1
##  5 (62.73,64.67]         8
##  6 (64.67,66.6]         12
##  7 (66.6,68.53]          3
##  8 (68.53,70.47]         5
##  9 (70.47,72.4]          2
## 10 (72.4,74.33]         10
## # i 16 more rows
```