Aula 3 - Tidyverse e Estatística Descritiva

João Paulo Lazzarini Cyrino

Agosto de 2020

Introdução

Nesta aula vamos abordar algumas noções sobre estatística descritiva e manipulação de dados utilizando o pacote *tidyverse*.

tidyverse é um pacote de bibliotecas para R muito utilizado atualmente. Ele é especialmente útil por tornar a sintaxe mais elegante e forçar dados organizados e consistentes. Trata-se de um pacote que tem várias bibliotecas dentro, como *dplyr* (para manipulação de dados) e *ggplot2* (para gráficos). Nesta aula abordaremos algumas funções dessas duas bibliotecas.

Para instalar *tidyverse* você pode ir no menu Tools > Install Packages... do RStudio. Você também pode simplesmente digitar no console: `install.packages("tidyverse")`. Uma vez instalado, você carrega as bibliotecas *dplyr* e *ggplot2* no seu ambiente R.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
```

Também precisaremos carregar os dados para hoje. Você pode encontrar esses dados na própria pasta da aula, no arquivo *wals.xlsx*. Siga o mesmo procedimento já explicado para carregar arquivos de excel em R.

Esses dados são referentes aos estudos presentes no projeto WALS. Temos o nome de cada estudo, a área em que eles se dão (Morfologia, Fonologia, etc.) e o número de línguas comparadas em cada estudo. Apesar de não servir muito para propósitos de insights sobre tipologia ou linguística, esses dados são interessantes para trabalhar estatística descritiva.

Tome um tempo para observar os dados:

```
glimpse(wals)
```

```
## Rows: 192
## Columns: 3
## $ name      <chr> "'Want' Complement Subjects", "'When' Clauses", "Absence ...
## $ languages <dbl> 283, 174, 567, 168, 124, 10, 190, 172, 380, 194, 183, 261...
## $ area      <chr> "Complex Sentences", "Complex Sentences", "Phonology", "N...
```

Funções de estatística descritiva

Em estatística temos uma série de funções para descrever os dados. Você provavelmente já trabalhou com *médias*, *desvios padrões*, *variância*, *medianas*, etc. Vamos aqui explicar brevemente como calcular essas medidas e para quê usá-las.

Média, Proporção e Mediana

A média é uma medida de centralidade, calculada pela soma dos valores de um conjunto de dados divido pelo número de dados do conjunto. A fórmula da média amostral segue abaixo:

$$X' = \frac{\sum_{i=1}^{n} x_i}{n}$$

Nos nossos dados da tabela *wals* podemos calcular a média de línguas utilizadas nos estudos. A variável *languages* tem esse número para cada estudo.

```
mean(wals$languages)
```

```
## [1] 398.2552
```

Para variáveis categóricas, como por exemplo *area*, podemos calcular a proporção de cada uma. Por exemplo, a proporção de estudos feitos sobre ordem de palavras ("Word Order"):

```
mean(wals$area == "Word Order")
```

```
## [1] 0.28125
```

Vemos que cerca de 28,12% dos estudos do WALS tem relação com ordem de palavras.

Voltando às variáveis numéricas, outra medida de centralidade bastante comum é a *mediana*. Esta nos dá o valor central de um conjunto de dados e é bastante útil quando os dados são dispersos.

```
median(wals$languages)
```

```
## [1] 257
```

Dispersão: Variância e Desvio Padrão

Embora média e mediana nos digam mais ou menos onde se centralizam os dados, apenas com essas medidas não conseguimos saber o quanto os dados podem se distanciar desse centro. Para isso temos as medidas de *variância* e *desvio padrão*. A variância da amostra (simbolizada por s^2) é calculada subtraindo a média de cada valor e elevando esse total ao quadrado. A soma disso é dividida pelo total de dados menos 1. Como na fórmula abaixo:

$$s^2 = \frac{\sum_{i=1}^{n} (x_i - X')^2}{n-1}$$

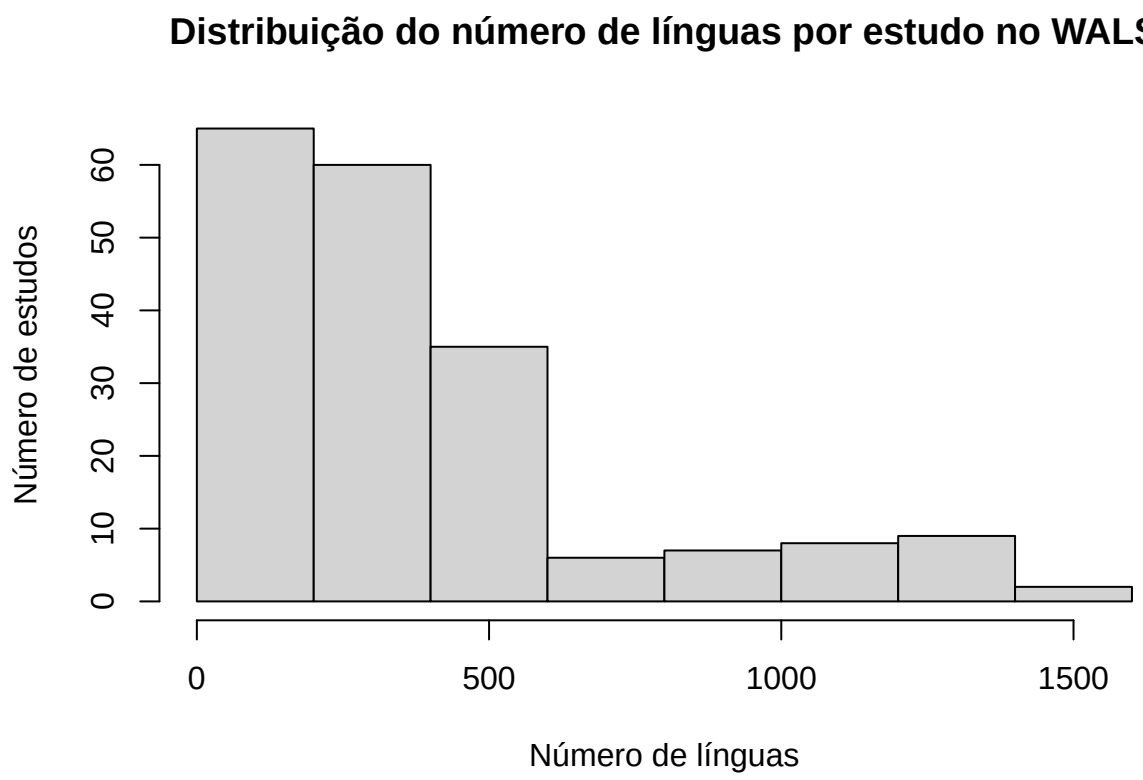
Podemos calcular a variância da variável *languages* utilizando o seguinte código em R:

```
var(wals$languages)
```

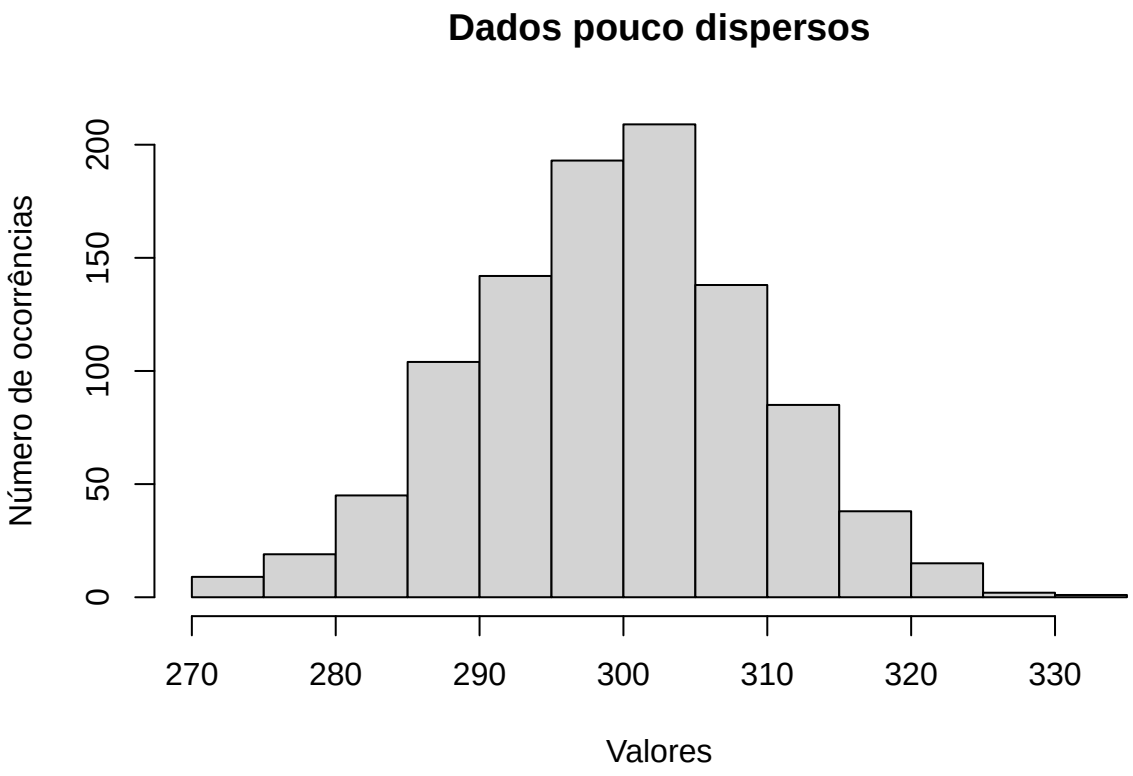
```
## [1] 122320.2
```

Como variância é uma medida ao quadrado, ela pode ser bastante grande em dados dispersos. Os dados que temos para *languages* por exemplo, são bastante dispersos. Aqui temos um histograma deles, o tipo de gráfico que mede a distribuição dos dados:

```r
hist(wals$languages,
     main="Distribuição do número de línguas por estudo no WALS",
     xlab="Número de línguas",
     ylab="Número de estudos")
```



Dados mais concentrados na média costumam ter o histograma com um formato bastante específico, como o visto abaixo:



O *desvio padrão* é a raiz quadrada da *variância* e é calculado da seguinte forma:

```
sd(wals$languages)
```

```
## [1] 349.7431
```

Sintaxe Tidy

Anteriormente apresentamos as medidas de *média*, *proporção*, *mediana*, *variância* e *desvio padrão*. Isso, no entanto, foi apresentado usando as funções comuns do R. Aqui introduzimos a sintaxe da biblioteca *dplyr*, que traz o padrão *tidy(verse)*.

Essa sintaxe que vamos aprender aqui é bastante útil para manipular tabelas, extrair informações, etc.

Criando uma tabela que sumariza os dados

A partir da tabela *wals*, podemos criar uma tabela que traz dados de centralidade e dispersão da variável *languages*. Abaixo criamos essa tabela e a salvamos na variável *sumario.languages*:

```
sumario.languages <- wals %>%
  summarise(
    media = mean(languages),
    mediana = median(languages),
    variancia = var(languages),
    desvio.padrao = sd(languages)
  )
```

Podemos visualizar essa tabela simplesmente chamando nossa variável *sumario.languages*:

```
sumario.languages
```

```
## # A tibble: 1 x 4
##   media mediana variancia desvio.padrao
##   <dbl>   <dbl>     <dbl>         <dbl>
## 1  398.     257   122320.          350.
```

O código segue um pouco do pensamento: vamos tomar a tabela wals e, a partir dela, criamos uma tabela com a média, mediana, variância e desvio padrão da variável *languages*. O operador `%>%` indica que a função a seguir estará lidando com dados de *wals*. A função `summarise` cria uma nova tabela com dados que descrevem a tabela *wals*, como média, desvio padrão, etc.

Podemos incrementar a tabela colocando também a proporção de alguns elementos da variável *area*:

```
sumario.languages <- wals %>%
  summarise(
    media = mean(languages),
    mediana = median(languages),
    variancia = var(languages),
    desvio.padrao = sd(languages),
    prop.ordem.palavras = mean(area=="Word Order"),
    prop.morfologia = mean(area=="Morphology")
  )
sumario.languages
```

```
## # A tibble: 1 x 6
##   media mediana variancia desvio.padrao prop.ordem.palavras prop.morfologia
##   <dbl>   <dbl>     <dbl>         <dbl>               <dbl>           <dbl>
## 1  398.     257   122320.          350.               0.281          0.0625
```

Agrupando dados

Vimos anteriormente que, por conta da grande variância, não parece haver um padrão geral no número de línguas utilizadas para cada estudo do WALS. Mas podemos nos perguntar se um padrão não aparece quando observamos a média, variância e desvio padrão do número línguas por area.

Podemos agrupar os dados em termos de uma variável, no caso *area*, da seguinte forma:

```
sumario.areas <- wals %>%
  group_by(area)
sumario.areas
```

```
## # A tibble: 192 x 3
## # Groups:   area [11]
##    name                                          languages area
##    <chr>                                             <dbl> <chr>
##  1 'Want' Complement Subjects                          283 Complex Sentences
##  2 'When' Clauses                                      174 Complex Sentences
##  3 Absence of Common Consonants                        567 Phonology
##  4 Action Nominal Constructions                        168 Nominal Syntax
##  5 Adjectives without Nouns                            124 Nominal Syntax
##  6 Adjoined relative clauses                            10 Word Order
##  7 Alignment of Case Marking of Full Noun Phrases      190 Simple Clauses
##  8 Alignment of Case Marking of Pronouns               172 Simple Clauses
##  9 Alignment of Verbal Person Marking                  380 Simple Clauses
## 10 Antipassive Constructions                           194 Simple Clauses
## # ... with 182 more rows
```

Apenas a função `group_by` não nos apresenta muita coisa. Porém, o grande poder dela é quando associamos seu resultado à função `summarise`. Note a utilização do operador `%>%` novamente, indicando que estamos trabalhando com os dados de wals agrupados por *area*:

```
sumario.areas <- wals %>%
  group_by(area) %>%
  summarise(media.linguas = mean(languages),
            variancia.linguas = var(languages),
            desv.padrao.linguas = sd(languages))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
sumario.areas
```

```
## # A tibble: 11 x 4
##    area                media.linguas variancia.linguas desv.padrao.linguas
##    <chr>                       <dbl>             <dbl>               <dbl>
##  1 Complex Sentences           179               3136                56
##  2 Lexicon                     232.             28001.              167.
##  3 Morphology                  276.             51209.              226.
##  4 Nominal Categories          353.             62584.              250.
##  5 Nominal Syntax              212.              3787.               61.5
##  6 Other                        74.5             9384.               96.9
##  7 Phonology                   491.             18739.              137.
##  8 Sign Languages               36.5               4.5                2.12
##  9 Simple Clauses              353.             59133.              243.
## 10 Verbal Categories           384.            109947.              332.
## 11 Word Order                  563.            253903.              504.
```

A tabela nos mostra que, ainda assim, existe bastante variância dentro das áreas. Isso significa que os estudos usam números de línguas bastante diversos entre si.

Algo interessante de se colocar nesse sumário são também os dados de proporção e contagem de estudos em cada área. Para contar o número de estudos em cada grupo, utilizamos a função `n()` dentro de summarize. Ela, por si só, já nos dá o número de estudos de cada área. Para ter a proporção, basta dividir `n()/nrow(wals)`, ou seja, a contagem de cada grupo pelo número de linhas (de dados) total da tabela wals. Vamos ver isso na prática:

```
sumario.areas <- wals %>%
  group_by(area) %>%
  summarise(cont.area = n(),
            prop.area = n()/nrow(wals),
            media.linguas = mean(languages),
            variancia.linguas = var(languages),
            desv.padrao.linguas = sd(languages))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
sumario.areas
```

```
## # A tibble: 11 x 6
##    area      cont.area prop.area media.linguas variancia.lingu~ desv.padrao.lin~
##    <chr>         <int>     <dbl>         <dbl>            <dbl>            <dbl>
##  1 Complex ~         6    0.0312         179              3136              56
##  2 Lexicon          14    0.0729         232.            28001.            167.
##  3 Morpholo~        12    0.0625         276.            51209.            226.
##  4 Nominal ~        29    0.151          353.            62584.            250.
##  5 Nominal ~         8    0.0417         212.             3787.             61.5
```

```
##  6 Other             2    0.0104          74.5          9384.            96.9
##  7 Phonology        20    0.104          491.          18739.           137.
##  8 Sign Lan~         2    0.0104          36.5             4.5            2.12
##  9 Simple C~        26    0.135          353.          59133.           243.
## 10 Verbal C~        19    0.0990         384.         109947.           332.
## 11 Word Ord~        54    0.281          563.         253903.           504.
```

Alterando tabelas

Outra função bastante poderosa é a função `mutate`, que permite adicionar ou alterar colunas de uma tabela. Por exemplo, podemos arredondar para cima a média, variância e desvio padrão de nossa tabela *sumario.areas*:

```
sumario.areas.arred <- sumario.areas %>%
  mutate(media.linguas = ceiling(media.linguas),
         variancia.linguas = ceiling(variancia.linguas),
         desv.padrao.linguas = ceiling(desv.padrao.linguas))
sumario.areas.arred
```

```
## # A tibble: 11 x 6
##    area      cont.area prop.area media.linguas variancia.lingu~ desv.padrao.lin~
##    <chr>         <int>     <dbl>         <dbl>            <dbl>            <dbl>
##  1 Complex ~         6    0.0312           179             3136               56
##  2 Lexicon          14    0.0729           233            28001              168
##  3 Morpholo~        12    0.0625           276            51209              227
##  4 Nominal ~        29    0.151            353            62585              251
##  5 Nominal ~         8    0.0417           212             3787               62
##  6 Other             2    0.0104            75             9385               97
##  7 Phonology        20    0.104            491            18739              137
##  8 Sign Lan~         2    0.0104            37                5                3
##  9 Simple C~        26    0.135            353            59134              244
## 10 Verbal C~        19    0.0990           384           109947              332
## 11 Word Ord~        54    0.281            564           253904              504
```

Perceba que, para fazer o arredondamento simplesmente aplicamos a função `ceiling` para cada variável da tabela e salvamos ela na variável de mesmo nome, sobrescrevendo-a. Isso faz com que alteremos a tabela.

Podemos criar uma nova variável também. Por exemplo, é comum às vezes representar proporções (probabilidades, na verdade) com o logaritmo positivo. Podemos criar uma nova variável que dá o logaritmo positivo de cada proporção utilizando a formula `-log2(prop.area)` dentro de mutate.

```
sumario.areas.plog <- sumario.areas %>%
  mutate(plog.area = -log2(prop.area))
sumario.areas.plog
```

```
## # A tibble: 11 x 7
##    area  cont.area prop.area media.linguas variancia.lingu~ desv.padrao.lin~
##    <chr>     <int>     <dbl>         <dbl>            <dbl>            <dbl>
##  1 Comp~         6    0.0312         179              3136               56
##  2 Lexi~        14    0.0729         232.            28001.             167.
##  3 Morp~        12    0.0625         276.            51209.             226.
##  4 Nomi~        29    0.151          353.            62584.             250.
##  5 Nomi~         8    0.0417         212.             3787.              61.5
##  6 Other         2    0.0104          74.5            9384.              96.9
##  7 Phon~        20    0.104          491.            18739.             137.
##  8 Sign~         2    0.0104          36.5               4.5              2.12
##  9 Simp~        26    0.135          353.            59133.             243.
## 10 Verb~        19    0.0990         384.           109947.             332.
```

```
## 11 Word~        54    0.281          563.          253903.            504.
## # ... with 1 more variable: plog.area <dbl>
```

Infelizmente nossa tabela `sumario.linguas.plog` ficou muito grande e não conseguimos visualizar a nova variável *plog.area*. Vamos enxugar essa nova tabela selecionando para ela apenas as colunas relativas a *area*. Para isso usamos a função `select`:

```
sumario.areas.plog <- sumario.areas %>%
  select(cont.area, prop.area) %>%
  mutate(plog.area = -log2(prop.area))
sumario.areas.plog
```

```
## # A tibble: 11 x 3
##    cont.area prop.area plog.area
##        <int>     <dbl>     <dbl>
##  1         6    0.0312      5
##  2        14    0.0729      3.78
##  3        12    0.0625      4
##  4        29    0.151       2.73
##  5         8    0.0417      4.58
##  6         2    0.0104      6.58
##  7        20    0.104       3.26
##  8         2    0.0104      6.58
##  9        26    0.135       2.88
## 10        19    0.0990      3.34
## 11        54    0.281       1.83
```

Resumindo:

Vimos as principais funções do pacote *dplyr*: `summarise`, `group_by`, `mutate` e `select`. Há muito mais coisas para se aprender sobre esse pacote e toda a sintaxe *tidy*, mas isso vem com a experiência.

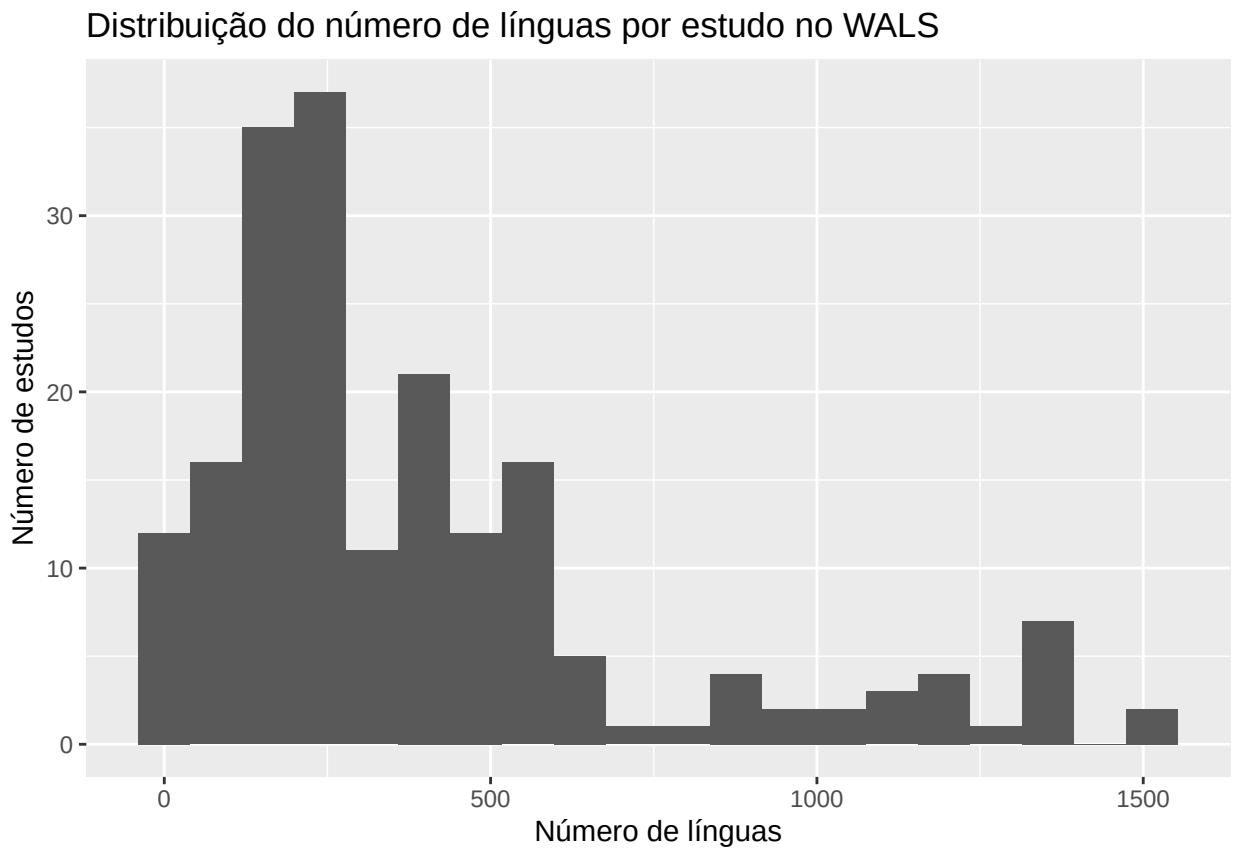
No seu uso normal da linguagem R, você pode contar com os cheatsheets (guias ilustrados) disponibilizados pelos mantenedores da biblioteca *tidyverse*. Nos materiais da aula você encontra uma cheatsheet para o pacote *dplyr* e outra para o *ggplot2*.

Visualizando dados com ggplot2

R já vem com uma série de gráficos que podemos plotar. No entanto, as opções disponíveis não tem toda a flexibilidade que às vezes é necessária. Por essa razão, damos preferência por utilizar o pacote *ggplot2*.

Abaixo plotamos o histograma da variável *languages* utilizando ggplot:

```
ggplot(wals, aes(x=languages)) +
  geom_histogram(bins=20) +
  labs(title="Distribuição do número de línguas por estudo no WALS",
       x="Número de línguas",
       y="Número de estudos")
```

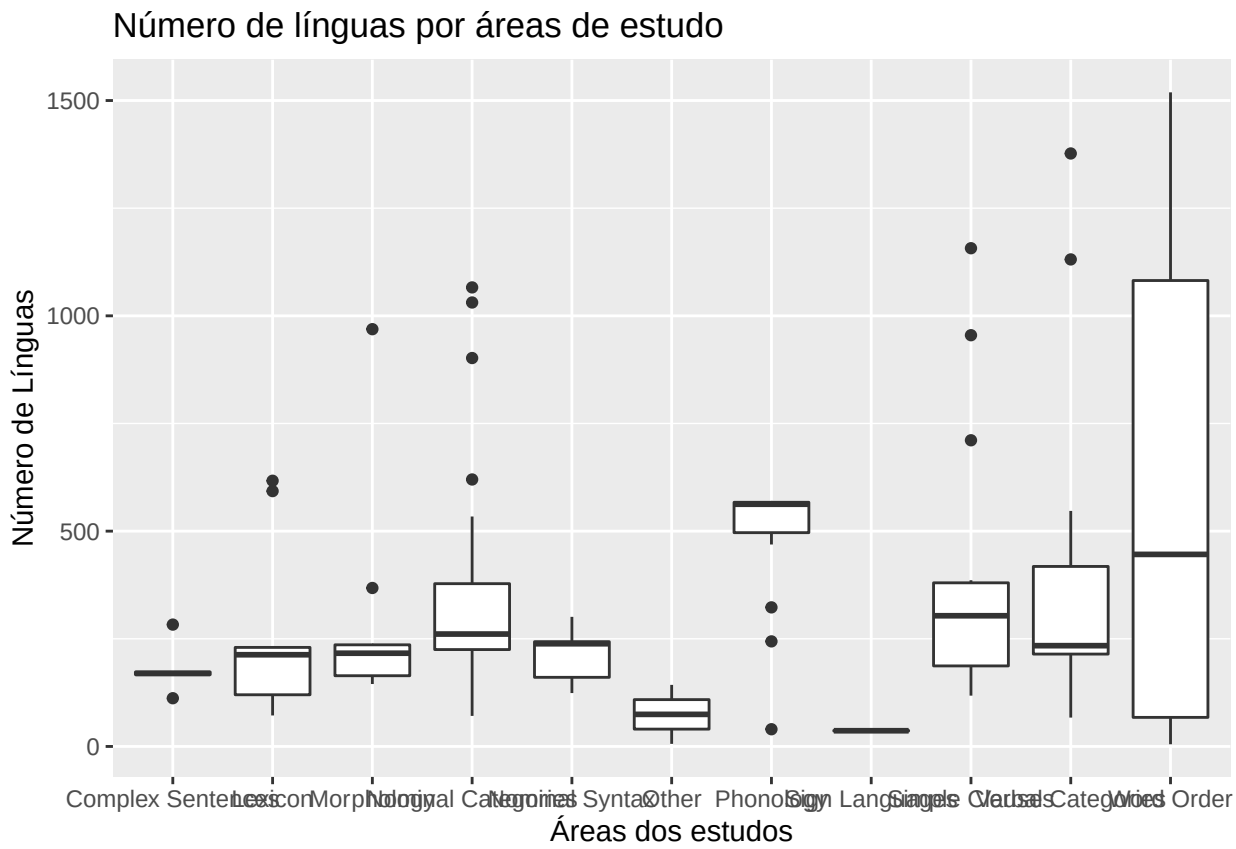


Na primeira linha chamamos a função `ggplot`, que pede a tabela de onde os dados vem (no caso, *wals*) e a função `aes`, em que colocamos os dados que serão plotados. No caso, um histograma é uma visualização de uma única variável. Por essa razão, colocamos essa variável no eixo x do gráfico.

Depois de chamar ggplot, adicionamos (com o operador + de soma) as funções de geometria, que plotam o tipo de gráfico. No caso, temos a função `geom_histogram`, que plota um histograma dos dados. Podemos adicionar ainda a função `labs`, para acrescentar títulos do gráfico e dos eixos.

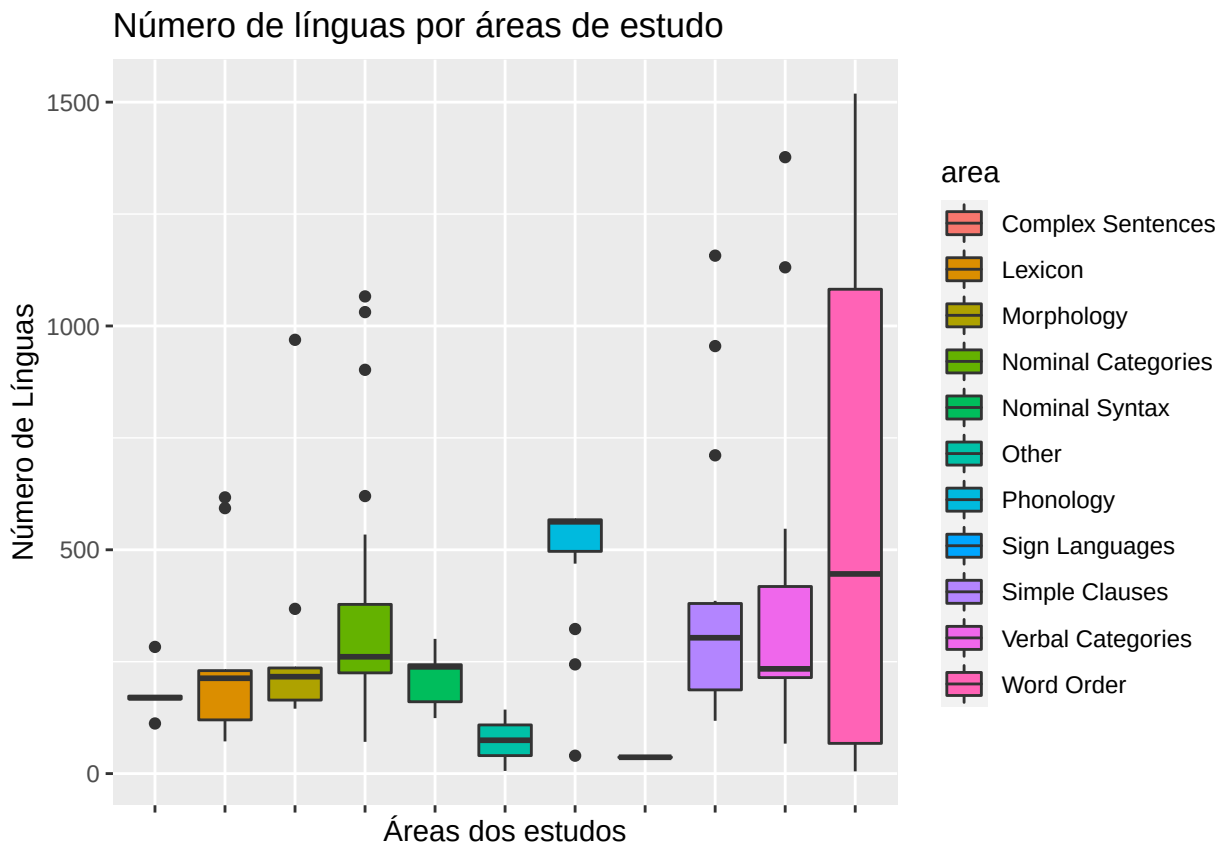
Abaixo plotamos um gráfico *boxplot* que relaciona a variável categórica *area* a *languages*:

```
ggplot(wals, aes(x=area, y=languages)) +
  geom_boxplot() +
  labs(title="Número de línguas por áreas de estudo",
       x="Áreas dos estudos",
       y="Número de Línguas")
```



O gráfico boxplot é interessante para mostrar como se distribuem as quantidades de línguas nos estudos de cada área. Infelizmente, o nome das áreas é muito grande e acaba ficando confuso na visualização. Para consertar isso, podemos adicionar cores e legenda na função `aes` e também retirar os rótulos no eixo x com alguns valores na função `theme`:

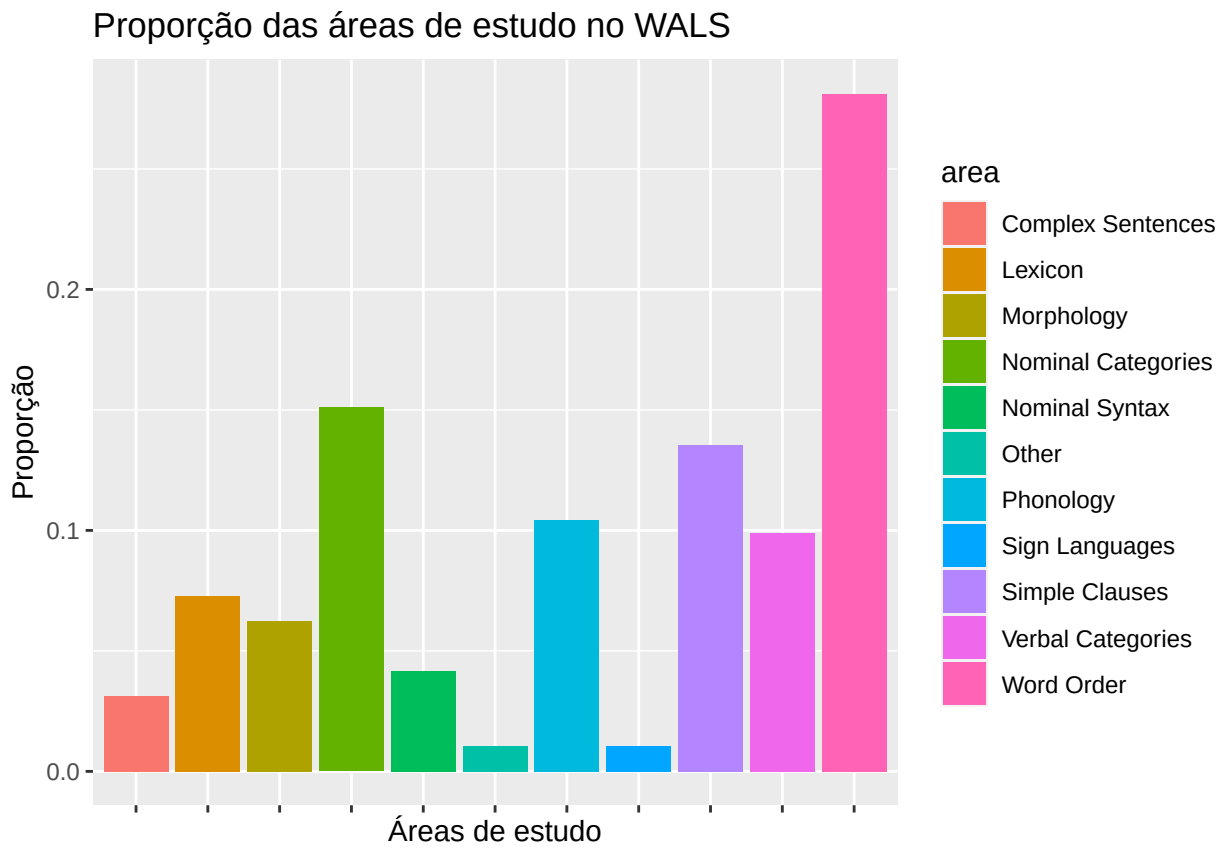
```
ggplot(wals, aes(x=area, y=languages, fill=area)) + # adiciona cores para as áreas
  geom_boxplot() +
  labs(title="Número de línguas por áreas de estudo",
       x="Áreas dos estudos",
       y="Número de Línguas") +
  theme(axis.text.x = element_blank()) # retira os rótulos do eixo x
```



Também podemos acomodar o gráfico ggplot em uma operação de manipulação de tabela. Abaixo construimos uma tabela que agrupa as áreas e dá a proporção de estudos em cada uma no *wals* e plota as proporções de cada área em um gráfico de barras:

```
wals %>%
  group_by(area) %>%
  summarise(prop = n()/nrow(wals)) %>%
  ggplot(aes(x=area, y=prop, fill=area)) +
  geom_bar(stat="identity") +
  labs(title="Proporção das áreas de estudo no WALS",
       x="Áreas de estudo",
       y="Proporção") +
  theme(axis.text.x = element_blank())
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```



Perceba-se que com o uso do operador `%>%` encadeamos todas as funções até a plotagem do gráfico.

Visualização de dados é algo que pode ser assunto de muitos livros, e a biblioteca *ggplot2* permite muitas opções e tem vários tipos de gráficos. Não é o escopo dessa aula se aprofundar nisso, mas apenas introduzir a sintaxe. Recomendamos consultar a documentação do ggplot (cheatsheet incluido na pasta dessa aula) e outros tutoriais. Outros gráficos que plotaremos ao longo deste curso serão, também, devidamente explicados.

Atividade

A partir dos dados da Aula 2, construa gráficos de barras mostrando cada categoria e sua respectiva proporção.