

Cadernos de Scripts - módulo 1

Daniel Pagotto e Fernanda Arantes (LAPEI-UFG)

21/06/2021

Nivelando conhecimentos sobre R

Para começar nosso curso, vamos aplicar operações básicas com o R. Insira os códigos e rode (ctrl + Enter em cima da linha) para ver os resultados.

```
# Operações básicas  
5 + 5
```

```
## [1] 10
```

```
10 - 6
```

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

```
10*2
```

```
## [1] 20
```

```
5/2
```

```
## [1] 2.5
```

```
#potência  
5**2
```

```
## [1] 25
```

```
# raiz quadrada  
sqrt(16)
```

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

```
# veja que os parênteses são usados para estabelecer uma ordem de operações,  
# igual aprendemos na matemática  
5*(50-45)
```

```
## [1] 25
```

Operações de atribuição

Atribuições são muito importantes! Usamos o “<-” para atribuir o resultado de uma operação a uma variável. É como se eu estivesse falando assim: “R, realize essa operação e guarde em um objeto chamado x”

```
#Atribuições
x <- 5 + 5
y <- 10 - 16
a <- 9
soma <- a + x
nome <- "daniel"
certo <- TRUE
```

Operações básicas

Vamos fazer um programinha para calcular meu índice de massa corpórea (IMC).

```
pesoDaniel <- 79
alturaDaniel <- 1.78
imcDaniel <- pesoDaniel/alturaDaniel**2
```

Agora calcule o IMC de todas as pessoas.

Nome	Peso	Altura
Alice	65	1.60
Gilmar	95	1.78
Cecília	75	1.80
Bianca	77	1.68
Valentina	80	1.72
Augusto	68	1.65

Não existe um jeito mais fácil de calcular?

Vetores

Você até pode calcular o IMC de cada um individualmente.

Mas vou apresentar uma forma de resolver - existem várias formas, usando função, loops. Mas vamos usar um tipo de objeto chamado **vetor**. O vetor é um conjunto unidimensional de objetos de um mesmo tipo (ex.: números, palavras).

Traduzindo... imagina uma tabela de excel formada por várias colunas. Uma das colunas é a idade e está expressa em número. Pronto, um vetor é como se fosse uma coluna com valores de um mesmo tipo.

```
# trabalhando com vetores. Basta colocar um c e abrir parênteses
pesos <- c(65, 95, 75, 77, 80, 68)
alturas <- c(1.60, 1.78, 1.80, 1.68, 1.72, 1.65)
imc <- pesos/alturas**2
imc
```

```
## [1] 25.39062 29.98359 23.14815 27.28175 27.04164 24.97704
```

Consigo arredondar os valores?

Consegue! Só usar a função `round`.

```
# dica, a função help serve para você pedir uma ajudinha para o R e ver o que a
# função faz
help(round)

round(imc, 2)
```

```
## [1] 25.39 29.98 23.15 27.28 27.04 24.98
```

```
imc <- round(imc, 2) #estou sobrescrevendo um vetor
                    # arredondado sobre ele mesmo
imc
```

```
## [1] 25.39 29.98 23.15 27.28 27.04 24.98
```

Matrizes

As **matrizes** possuem uma estrutura tabular, com linhas e colunas. Porém, semelhante ao vetor, todos os objetos devem ser de um mesmo tipo (ex.: tudo número, tudo caracter).

```
Matriz<-cbind(pesos,alturas,imc)
Matriz
```

```
##      pesos alturas  imc
## [1,]    65    1.60 25.39
## [2,]    95    1.78 29.98
## [3,]    75    1.80 23.15
## [4,]    77    1.68 27.28
## [5,]    80    1.72 27.04
## [6,]    68    1.65 24.98
```

Veja que tem uma aparência de tabela. Mas daqui em diante trabalharemos com outra estrutura chamada **dataframe**. Essa estrutura tem formato tabular e ainda permite que os objetos tenham tipos diferentes, ou seja, posso ter uma coluna numérica, outra no formato data, outra no formato de caracteres e assim por diante.

Existe um tipo de estrutura de dados chamado **lista** muito importante também. Mas entrar nele é assunto para um curso de R intermediário.

```
rownames(Matriz) <- c("Alice","Gilmar","Cecilia","Bianca","Valentina","Augusto")
Matriz
```

```
##      pesos alturas  imc
## Alice      65    1.60 25.39
## Gilmar     95    1.78 29.98
## Cecilia    75    1.80 23.15
## Bianca     77    1.68 27.28
## Valentina  80    1.72 27.04
## Augusto    68    1.65 24.98
```

Introduzindo funções para manipulação de dados

Aqui vamos começar a falar sobre manipulação de dados usando o pacote `dplyr`. É importante instalar os pacotes `dplyr` e `gapminder`, caso já não tenha feito, usando a função `install.packages("dplyr")`, assim como o pacote `gapminder`, que vai prover a base de dados para o curso.

Inspecionando o dataframe

```
library(gapminder)
library(dplyr)
basePaíses <- gapminder
```

```
# inspecionando a estrutura da base
str(basePaíses)
```

```
## tibble [1,704 x 6] (S3: tbl_df/tbl/data.frame)
## $ country : Factor w/ 142 levels "Afghanistan",...: 1 1 1 1 1 1 1 1 1 1 ...
## $ continent: Factor w/ 5 levels "Africa","Americas",...: 3 3 3 3 3 3 3 3 3 3 ...
## $ year      : int [1:1704] 1952 1957 1962 1967 1972 1977 1982 1987 1992 1997 ...
## $ lifeExp   : num [1:1704] 28.8 30.3 32 34 36.1 ...
## $ pop       : int [1:1704] 8425333 9240934 10267083 11537966 13079460 14880372 12881816 13867957 163...
## $ gdpPercap: num [1:1704] 779 821 853 836 740 ...
```

```
# inspecionando as 6 primeiras observações
head(basePaíses)
```

```
## # A tibble: 6 x 6
##   country    continent  year lifeExp      pop gdpPercap
##   <fct>      <fct>      <int>  <dbl>    <int>    <dbl>
## 1 Afghanistan Asia      1952   28.8  8425333    779.
## 2 Afghanistan Asia      1957   30.3  9240934    821.
## 3 Afghanistan Asia      1962   32.0 10267083    853.
## 4 Afghanistan Asia      1967   34.0 11537966    836.
## 5 Afghanistan Asia      1972   36.1 13079460    740.
## 6 Afghanistan Asia      1977   38.4 14880372    786.
```

```
# inspecionando as 10 últimas observações
tail(basePaíses, n = 10)
```

```
## # A tibble: 10 x 6
##   country    continent  year lifeExp      pop gdpPercap
##   <fct>      <fct>      <int>  <dbl>    <int>    <dbl>
## 1 Zimbabwe Africa      1962   52.4  4277736    527.
## 2 Zimbabwe Africa      1967   54.0  4995432    570.
## 3 Zimbabwe Africa      1972   55.6  5861135    799.
## 4 Zimbabwe Africa      1977   57.7  6642107    686.
## 5 Zimbabwe Africa      1982   60.4  7636524    789.
## 6 Zimbabwe Africa      1987   62.4  9216418    706.
## 7 Zimbabwe Africa      1992   60.4 10704340    693.
## 8 Zimbabwe Africa      1997   46.8 11404948    792.
## 9 Zimbabwe Africa      2002   40.0 11926563    672.
## 10 Zimbabwe Africa      2007   43.5 12311143    470.
```

```
# estatísticas descritivas da base
summary(basePaíses)
```

```
##           country      continent      year      lifeExp
## Afghanistan: 12 Africa :624 Min. :1952 Min. :23.60
## Albania : 12 Americas:300 1st Qu.:1966 1st Qu.:48.20
## Algeria : 12 Asia :396 Median :1980 Median :60.71
## Angola : 12 Europe :360 Mean :1980 Mean :59.47
## Argentina : 12 Oceania : 24 3rd Qu.:1993 3rd Qu.:70.85
## Australia : 12 Max. :2007 Max. :82.60
## (Other) :1632
##      pop      gdpPercap
## Min. :6.001e+04 Min. : 241.2
## 1st Qu.:2.794e+06 1st Qu.: 1202.1
## Median :7.024e+06 Median : 3531.8
## Mean :2.960e+07 Mean : 7215.3
## 3rd Qu.:1.959e+07 3rd Qu.: 9325.5
## Max. :1.319e+09 Max. :113523.1
##
```

```
# Acessando uma variável da base
head(basePaíses$continent,20)
```

```
## [1] Asia Asia Asia Asia Asia Asia Asia Asia Asia Asia
## [11] Asia Asia Europe Europe Europe Europe Europe Europe Europe
## Levels: Africa Americas Asia Europe Oceania
```

```
# Acessando elementos únicos
unique(basePaíses$year)
```

```
## [1] 1952 1957 1962 1967 1972 1977 1982 1987 1992 1997 2002 2007
```

```
# Verificando a média do vetor de expectativa de vida
mean(basePaíses$lifeExp)
```

```
## [1] 59.47444
```

```
# A função glimpse permite dar uma olhadinha nos dados. Parece muito com a
# função str(), mas é do pacote dplyr
```

```
glimpse(basePaíses)
```

```
## Rows: 1,704
## Columns: 6
## $ country <fct> "Afghanistan", "Afghanistan", "Afghanistan", "Afghanistan", ~
## $ continent <fct> Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, ~
## $ year <int> 1952, 1957, 1962, 1967, 1972, 1977, 1982, 1987, 1992, 1997, ~
## $ lifeExp <dbl> 28.801, 30.332, 31.997, 34.020, 36.088, 38.438, 39.854, 40.8~
## $ pop <int> 8425333, 9240934, 10267083, 11537966, 13079460, 14880372, 12~
## $ gdpPercap <dbl> 779.4453, 820.8530, 853.1007, 836.1971, 739.9811, 786.1134, ~
```

Abrindo um parêntese

Veja que o glimpse mostra o tipo das variáveis (int, fct, dbl). Existem vários tipos de variáveis no R e compreendê-los é importante, pois determinam possíveis visualizações e técnicas de análise. Apresentei três tipos importantes:

- dbl (double) - valores numéricos e contínuos (“quebrados”)
- int (integer) - valores numéricos discretos (“inteiros”)
- fct (factor) - uma palavra.

Se alguém a série Stranger Things, da Netflix, vai lembrar o nome dessa menininha: Eleven. Nesse caso, Eleven é um nome, portanto, representaria um factor.

11 (int)

11.05 (dbl)

2011-01-11 (date)



“11”
(fct)

Função filter

Função para filtrar observações de uma base conforme algumas condições. Nesse caso, países cujo continente corresponde à Ásia.

```
basePaíses %>%  
  filter(continent == "Asia")
```

```
## # A tibble: 396 x 6  
##   country    continent  year lifeExp      pop gdpPercap  
##   <fct>      <fct>    <int>  <dbl>    <int>   <dbl>  
## 1 Afghanistan Asia      1952   28.8  8425333    779.  
## 2 Afghanistan Asia      1957   30.3  9240934    821.  
## 3 Afghanistan Asia      1962   32.0 10267083    853.  
## 4 Afghanistan Asia      1967   34.0 11537966    836.  
## 5 Afghanistan Asia      1972   36.1 13079460    740.
```

```
## 6 Afghanistan Asia      1977      38.4 14880372      786.
## 7 Afghanistan Asia      1982      39.9 12881816      978.
## 8 Afghanistan Asia      1987      40.8 13867957      852.
## 9 Afghanistan Asia      1992      41.7 16317921      649.
## 10 Afghanistan Asia     1997      41.8 22227415      635.
## # ... with 386 more rows
```

```
basePaíses %>%
  filter(continent == "Americas" & year>1990)
```

```
## # A tibble: 100 x 6
##   country    continent  year lifeExp      pop gdpPercap
##   <fct>      <fct>    <int>  <dbl>    <int>    <dbl>
## 1 Argentina Americas   1992    71.9  33958947    9308.
## 2 Argentina Americas   1997    73.3  36203463   10967.
## 3 Argentina Americas   2002    74.3  38331121    8798.
## 4 Argentina Americas   2007    75.3  40301927   12779.
## 5 Bolivia    Americas   1992    60.0   6893451    2962.
## 6 Bolivia    Americas   1997    62.0   7693188    3326.
## 7 Bolivia    Americas   2002    63.9   8445134    3413.
## 8 Bolivia    Americas   2007    65.6   9119152    3822.
## 9 Brazil     Americas   1992    67.1 155975974    6950.
## 10 Brazil     Americas   1997    69.4 168546719    7958.
## # ... with 90 more rows
```

Nesse caso, precisamos de todas observações, com exceção do continente Oceania. Veja como fica:

```
# != diferente
basePaíses %>%
  filter(continent != "Oceania")
```

```
## # A tibble: 1,680 x 6
##   country    continent  year lifeExp      pop gdpPercap
##   <fct>      <fct>    <int>  <dbl>    <int>    <dbl>
## 1 Afghanistan Asia      1952    28.8   8425333     779.
## 2 Afghanistan Asia      1957    30.3   9240934     821.
## 3 Afghanistan Asia      1962    32.0  10267083     853.
## 4 Afghanistan Asia      1967    34.0  11537966     836.
## 5 Afghanistan Asia      1972    36.1  13079460     740.
## 6 Afghanistan Asia      1977    38.4  14880372     786.
## 7 Afghanistan Asia      1982    39.9  12881816     978.
## 8 Afghanistan Asia      1987    40.8  13867957     852.
## 9 Afghanistan Asia      1992    41.7  16317921     649.
## 10 Afghanistan Asia     1997    41.8  22227415     635.
## # ... with 1,670 more rows
```

```
# Você pode armazenar sua consulta em outro objeto
baseAsia <- basePaíses %>%
  filter(continent == "Asia")
```

Função select

A função `select` é muito útil para você selecionar apenas as variáveis que precisa trabalhar. O PSED (base que usaremos no módulo 3) tem mais de 8000 colunas. É prudente usar a função `select` para separar apenas o que você precisa depois que tiver estudado os dados.

```
basePaíses %>%  
  select(year, country, gdpPercap)
```

```
## # A tibble: 1,704 x 3  
##   year country      gdpPercap  
##   <int> <fct>      <dbl>  
## 1  1952 Afghanistan    779.  
## 2  1957 Afghanistan    821.  
## 3  1962 Afghanistan    853.  
## 4  1967 Afghanistan    836.  
## 5  1972 Afghanistan    740.  
## 6  1977 Afghanistan    786.  
## 7  1982 Afghanistan    978.  
## 8  1987 Afghanistan    852.  
## 9  1992 Afghanistan    649.  
## 10 1997 Afghanistan    635.  
## # ... with 1,694 more rows
```

Quando usamos o `-` pegamos qualquer variável menos aquela de interesse, nesse caso `lifeExp`.

```
basePaíses %>%  
  select(-lifeExp)
```

```
## # A tibble: 1,704 x 5  
##   country      continent year      pop gdpPercap  
##   <fct>      <fct>      <int>   <int>   <dbl>  
## 1 Afghanistan Asia      1952  8425333    779.  
## 2 Afghanistan Asia      1957  9240934    821.  
## 3 Afghanistan Asia      1962 10267083    853.  
## 4 Afghanistan Asia      1967 11537966    836.  
## 5 Afghanistan Asia      1972 13079460    740.  
## 6 Afghanistan Asia      1977 14880372    786.  
## 7 Afghanistan Asia      1982 12881816    978.  
## 8 Afghanistan Asia      1987 13867957    852.  
## 9 Afghanistan Asia      1992 16317921    649.  
## 10 Afghanistan Asia      1997 22227415    635.  
## # ... with 1,694 more rows
```

Função select + filter

Olha que legal, as funções conversam entre si! Então eu posso fazer dois procedimentos - filtrar e depois selecionar - em um só conjunto de comandos.

```
basePaíses %>%  
  filter(continent == "Americas" & year>1990) %>%  
  select(year, country, gdpPercap)
```



```
## # A tibble: 100 x 3
##   year country  gdpPercap
##   <int> <fct>      <dbl>
## 1  1992 Argentina    9308.
## 2  1997 Argentina   10967.
## 3  2002 Argentina    8798.
## 4  2007 Argentina   12779.
## 5  1992 Bolivia     2962.
## 6  1997 Bolivia     3326.
## 7  2002 Bolivia     3413.
## 8  2007 Bolivia     3822.
## 9  1992 Brazil      6950.
## 10 1997 Brazil      7958.
## # ... with 90 more rows
```

Função mutate

A função `mutate` serve para criar uma nova variável. Nesse primeiro exemplo, criei uma variável chamada GDP (PIB), que é resultado da multiplicação entre as variáveis `gdpPercap` (PIB per capita) e `pop` (população).

```
basePaíses <- basePaíses %>%
  mutate(GDP = gdpPercap * pop)

basePorte <- basePaíses %>%
  filter(year == 1992) %>%
  mutate(porte = if_else(pop > median(pop), "G", "P"))

head(basePorte)
```

```
## # A tibble: 6 x 8
##   country continent year lifeExp      pop gdpPercap      GDP porte
##   <fct>      <fct>   <int>   <dbl>   <int>   <dbl>   <dbl> <chr>
## 1 Afghanistan Asia     1992   41.7 16317921    649. 10595901589. G
## 2 Albania     Europe   1992   71.6  3326498    2497.  8307722183. P
## 3 Algeria     Africa   1992   67.7 26298373    5023. 132102425043. G
## 4 Angola      Africa   1992   40.6  8735988    2628.  22956828370. G
## 5 Argentina   Americas 1992   71.9 33958947    9308. 316104097627. G
## 6 Australia   Oceania  1992   77.6 17481977   23425. 409511234952. G
```

Funções group_by e summarize

A função `group_by` geralmente é aplicada associada a outra função, nesse caso, vamos usar associada ao `summarize`.

```
basePaíses %>%
  group_by(country) %>%
  summarize(meanLE=mean(lifeExp), meanPop=mean(pop), meanGpc=mean(gdpPercap))
```

```
## # A tibble: 142 x 4
##   country      meanLE meanPop meanGpc
```

```
##      <fct>          <dbl>      <dbl>      <dbl>
##  1 Afghanistan    37.5 15823715.    803.
##  2 Albania         68.4 2580249.    3255.
##  3 Algeria         59.0 19875406.   4426.
##  4 Angola          37.9 7309390.    3607.
##  5 Argentina       69.1 28602240.   8956.
##  6 Australia       74.7 14649312.  19981.
##  7 Austria         73.1 7583298.   20412.
##  8 Bahrain         65.6 373913.    18078.
##  9 Bangladesh      49.8 90755395.    818.
## 10 Belgium        73.6 9725119.   19901.
## # ... with 132 more rows
```

Veja como ela funciona: vamos agrupar todas as observações cuja variável *country* for a mesma e, em sequência, aplicar um cálculo de média sobre o agrupamento.

country	continent	year	lifeExp	pop	gdpPercap
Afghanistan	Asia	1952	28,801	8425333	779,4453
Afghanistan	Asia	1957	30,332	9240934	820,853
Afghanistan	Asia	1962	31,997	10267083	853,1007
Afghanistan	Asia	1967	34,02	11537966	836,1971
Afghanistan	Asia	1972	36,088	13079460	739,9811
Afghanistan	Asia	1977	38,438	14880372	786,1134
Afghanistan	Asia	1982	39,854	12881816	978,0114
Afghanistan	Asia	1987	40,822	13867957	852,3959
Afghanistan	Asia	1992	41,674	16317921	649,3414
Afghanistan	Asia	1997	41,763	22227415	635,3414
Afghanistan	Asia	2002	42,129	25268405	726,7341
Afghanistan	Asia	2007	43,828	31889923	974,5803
Albania	Europe	1952	55,23	1282697	1601,056
Albania	Europe	1957	59,28	1476505	1942,284
Albania	Europe	1962	64,82	1728137	2312,889
Albania	Europe	1967	66,22	1984060	2760,197
Albania	Europe	1972	67,69	2263554	3313,422
Albania	Europe	1977	68,93	2509048	3533,004
Albania	Europe	1982	70,42	2780097	3630,881
Albania	Europe	1987	72	3075321	3738,933
Albania	Europe	1992	71,581	3326498	2497,438
Albania	Europe	1997	72,95	3428038	3193,055
Albania	Europe	2002	75,651	3508512	4604,212
Albania	Europe	2007	76,423	3600523	5937,03
Algeria	Africa	1952	43,077	9279525	2449,008
Algeria	Africa	1957	45,685	10270856	3013,976
Algeria	Africa	1962	48,303	11000948	2550,817
Algeria	Africa	1967	51,407	12760499	3246,992
Algeria	Africa	1972	54,518	14760787	4182,664
Algeria	Africa	1977	58,014	17152804	4910,417
Algeria	Africa	1982	61,368	20033753	5745,16
Algeria	Africa	1987	65,799	23254956	5681,359
Algeria	Africa	1992	67,744	26298373	5023,217
Algeria	Africa	1997	69,152	29072015	4797,295
...

group_by(country) +
summarise(media)

Country	lifeExp	pop	gdpPercap
Afghanistan	37,47	15823715	802,67
Albania	68,43	2580249	3255,367
Algeria	59,03	19875406	4426,026
...

Veja que posso agrupar usando mais de uma variável. Nesse caso, agrupamos todos as observações iguais de continente e ano.

```
basePaises %>%
  group_by(continent,year) %>%
  summarize(meanLE=mean(lifeExp),meanPop=mean(pop),meanGpc=mean(gdpPercap))
```

`summarise()` has grouped output by 'continent'. You can override using the `.groups` argument.

```
## # A tibble: 60 x 5
## # Groups:   continent [5]
##   continent year meanLE meanPop meanGpc
##   <fct>      <int> <dbl>    <dbl>    <dbl>
```

```
## 1 Africa      1952   39.1  4570010.   1253.
## 2 Africa      1957   41.3  5093033.   1385.
## 3 Africa      1962   43.3  5702247.   1598.
## 4 Africa      1967   45.3  6447875.   2050.
## 5 Africa      1972   47.5  7305376.   2340.
## 6 Africa      1977   49.6  8328097.   2586.
## 7 Africa      1982   51.6  9602857.   2482.
## 8 Africa      1987   53.3 11054502.   2283.
## 9 Africa      1992   53.6 12674645.   2282.
## 10 Africa     1997   53.6 14304480.   2379.
## # ... with 50 more rows
```

Funções top_n e arrange

A função `top_n` serve para selecionar os `n` maiores valores que desejar. Já a `arrange` permite ordenar em ordem decrescente ou ascendente (padrão).

```
basePaíses %>%
  filter(year == 2007) %>%
  top_n(5,pop) %>%
  arrange(desc(pop))
```

```
## # A tibble: 5 x 7
##   country      continent year lifeExp      pop gdpPercap      GDP
##   <fct>        <fct>    <int>   <dbl>    <int>    <dbl>    <dbl>
## 1 China        Asia      2007   73.0 1318683096  4959.  6.54e12
## 2 India        Asia      2007   64.7 1110396331  2452.  2.72e12
## 3 United States Americas  2007   78.2 301139947  42952.  1.29e13
## 4 Indonesia    Asia      2007   70.6 223547000  3541.  7.92e11
## 5 Brazil        Americas  2007   72.4 190010647  9066.  1.72e12
```

As próximas funções de manipulação são os joins, que servem para juntar duas bases. São funções importantíssimas. Vamos falar somente sobre dois tipos de joins (`left_join` e `inner_join`). Existem outros, mas esses são os que mais usamos no cotidiano. Para isso, vamos usar duas bases: o Índice de Cidades Empreendedoras (ICE), desenvolvido pela ENAP e Endeavor, e a base MUNIC do IBGE.

Base ICE



Tabela ICE 2020

Confira a seguir a tabela completa do Índice de Cidades Empreendedoras de 2020.

ICE 2020

Ambiente Regulatório

Infraestrutura

Mercado

Acesso Capital

Inovacao

Capital Humano

Cultura Empreendedora

Fonte: Índice de Cidades empreendedoras

Base MUNIC

BRASIL

CORONAVIRUS (COVID-19)

Simplifique!

Participe

Acesso à informação

Legislação

Canais

IBGE
Instituto Brasileiro de Geografia e Estatística

Institucional

Próximas Divulgações

Biblioteca

Respondendo

Estadísticas

Geociências

Cidades e Estados

Agência de Notícias

Nossos sites

Acesso à Informação

> Estatísticas > Sociais > Saúde

Pesquisa de Informações Básicas Municipais - MUNIC

O que é

Edições

2019

Sobre a publicação
Principais resultados
Tabelas
Conceitos e métodos
Publicações

Suplementos

Downloads

Informações técnicas

O que é

Efetua, periodicamente, um levantamento pormenorizado de informações sobre a estrutura, a dinâmica e o funcionamento das instituições públicas municipais, tendo como unidade de investigação o município e, como informante principal, a prefeitura, por meio dos diversos setores que a compõem.

A Pesquisa de Informações Básicas Municipais - MUNIC teve início em 1999, extensiva à totalidade dos municípios do País. Os temas e questões abordados em seu questionário básico são levantados regularmente e visam responder às necessidades de informação da sociedade e do Estado brasileiro, com vistas à consolidação de uma base de dados estatísticos e cadastrais atualizados e que proporcionem um conjunto relevante de indicadores de avaliação e monitoramento dos quadros institucional e administrativo das municipalidades. Usualmente, a MUNIC traz um caderno suplementar que contempla temas específicos, de forma mais detalhada, em um esforço permanente de atualização da pesquisa. A partir de 2005, o bloco sobre características básicas dos gestores, presente no questionário básico, passou a ser investigado a cada quadriênio, sempre nos anos que marcam o início das administrações eleitas no ano anterior.

A pesquisa fornece informações variadas sobre a gestão pública municipal, incluindo a legislação vigente e os instrumentos de planejamento existentes nessa esfera da administração, especialmente aqueles discriminados no Estatuto da Cidade e que, junto com o Plano Diretor, têm por meta regular o uso e a ocupação do solo urbano; organização das prefeituras; composição do quadro de pessoal por vínculo empregatício das prefeituras, tanto na administração direta quanto na indireta; recursos financeiros utilizados

Fonte: MUNIC

Baixando as bases direto do meu github

```
munic <- read.csv("https://raw.githubusercontent.com/danielppagotto/R_empreendedorismo1/main/arquivos%20munic.csv")
ice <- read.csv("https://raw.githubusercontent.com/danielppagotto/R_empreendedorismo1/main/arquivos%20ice.csv")
```

Inspecionando as bases

```
glimpse(munic)
```

```
## Rows: 195
## Columns: 10
## $ X.U.FEFF.id      <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, ~
## $ CodMun          <int> 1100155, 1100189, 1100205, 1100254, 1100262, 1200385, ~
## $ reducao_iptu     <chr> "Não", "Sim", "Não", "Não", "Sim", "Sim", "Não", "Não"~
## $ isencao_iptu     <chr> "Sim", "Sim", "Não", "Não", "Sim", "Não", "Não", "Sim"~
## $ reducao_issqn    <chr> "Não", "Não", "Sim", "Não", "Não", "Não", "Não", "Não"~
## $ isencao_issqn    <chr> "Não", "Não", "Não", "Não", "Não", "Não", "Sim", "Não"~
## $ isencao_taxas    <chr> "Não", "Não", "Não", "Não", "Sim", "Sim", "Não", "Não"~
## $ cessao_terrenos  <chr> "Sim", "Sim", "Sim", "Não", "Não", "Não", "Não", "Não"~
## $ doacao_terrenos  <chr> "Não", "Sim", "Não", "Não", "Sim", "Não", "Não", "Não"~
## $ outros          <chr> "Não", "Não", "Não", "Sim", "Não", "Não", "Não", "Não"~
```

```
glimpse(ice)
```

```
## Rows: 100
## Columns: 11
## $ X.U.FEFF.cidade  <chr> "São Paulo", "Florianópolis", "Osasco", "Vitória"~
## $ UF              <chr> "SP", "SC", "SP", "ES", "DF", "SP", "SP", "SP", ~
## $ cod_ibge         <int> 3550308, 4205407, 3534401, 3205309, 5300108, 354~
## $ ICE              <dbl> 9.51, 8.12, 7.94, 7.91, 7.58, 7.54, 7.52, 7.46, ~
## $ ambiente_regulatorio <dbl> 7.97, 6.60, 7.01, 8.90, 4.14, 7.14, 6.63, 6.39, ~
## $ infraestrutura   <dbl> 9.76, 7.15, 6.93, 5.65, 7.35, 6.71, 8.13, 7.48, ~
## $ mercado          <dbl> 7.70, 6.19, 7.55, 6.20, 8.30, 8.13, 7.18, 8.64, ~
## $ acesso_capital   <dbl> 11.33, 8.43, 10.06, 7.32, 6.82, 5.74, 6.11, 6.04~
## $ inovacao         <dbl> 7.22, 8.52, 6.27, 7.26, 6.39, 7.56, 7.67, 6.52, ~
## $ capital_humano    <dbl> 5.82, 8.88, 5.94, 8.05, 6.56, 6.98, 6.40, 7.31, ~
## $ cultura_empreendedora <dbl> 5.54, 5.26, 5.96, 4.33, 3.95, 5.95, 5.90, 6.46, ~
```

Vamos juntar as bases por meio das colunas cod_cidade e cod_ibge.

Um ponto **muito importante**: ambas variáveis devem ser do mesmo tipo. No caso, ambas estão como `int`. Caso uma fosse `factor` e a outra `int`, o join geraria uma mensagem de erro.

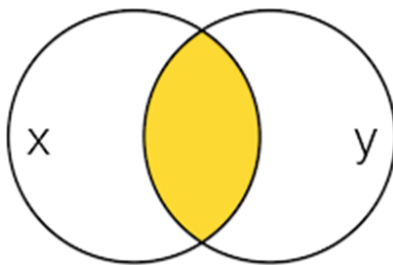
Vamos supor que o `cod_ibge` da base `ice` estive em formato `factor` (lembra da Eleven do Stranger Things). Para transformá-lo para o formato `int` seria necessário aplicar o seguinte comando.

```
ice$cod_ibge <- as.integer(ice$cod_ibge)
```

inner_join

Cidade	UF	cod_ibge	ICE	...	cod_cidade	reducao_ipu	isencao_ipu	...
São Paulo	SP	3550308	9.51	...	1100155	Não	Sim	...
Florianópolis	SC	4205407	8.12	...	1100189	Sim	Sim	...
Osasco	SP	3534401	7.94	...	1100205	Não	Não	...
...

inner_join

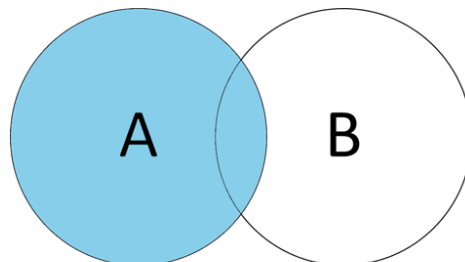


```
inner <- munic %>%  
  inner_join(ice, by = c("CodMun"="cod_ibge"))
```

left_join

Cidade	UF	cod_ibge	ICE	...	cod_cidade	reducao_ipu	isencao_ipu	...
São Paulo	SP	3550308	9.51	...	1100155	Não	Sim	...
Florianópolis	SC	4205407	8.12	...	1100189	Sim	Sim	...
Osasco	SP	3534401	7.94	...	1100205	Não	Não	...
...

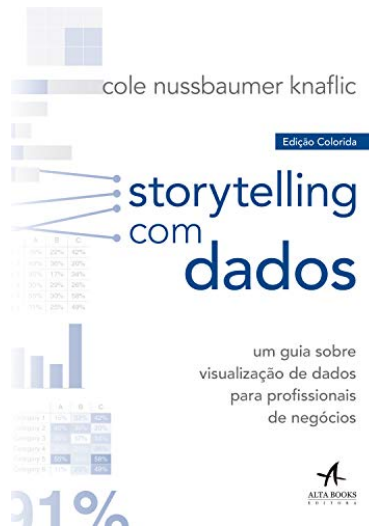
left_join



```
left <- munic %>%
  left_join(ice, by = c("CodMun"="cod_ibge"))
```

Introduzindo funções para visualização de dados

Dica de leitura



Preparando nosso ambiente

Antes de começar, vamos chamar alguns pacotes e preparar uma base que usaremos! Caso ainda não tenha algum dos pacotes abaixo, terá que instalar usando o comando `install.packages("nome do pacote")`.

```
library(ggplot2)
library(dplyr)
library(gapminder)

base <- gapminder %>%
  filter(year == 2007)

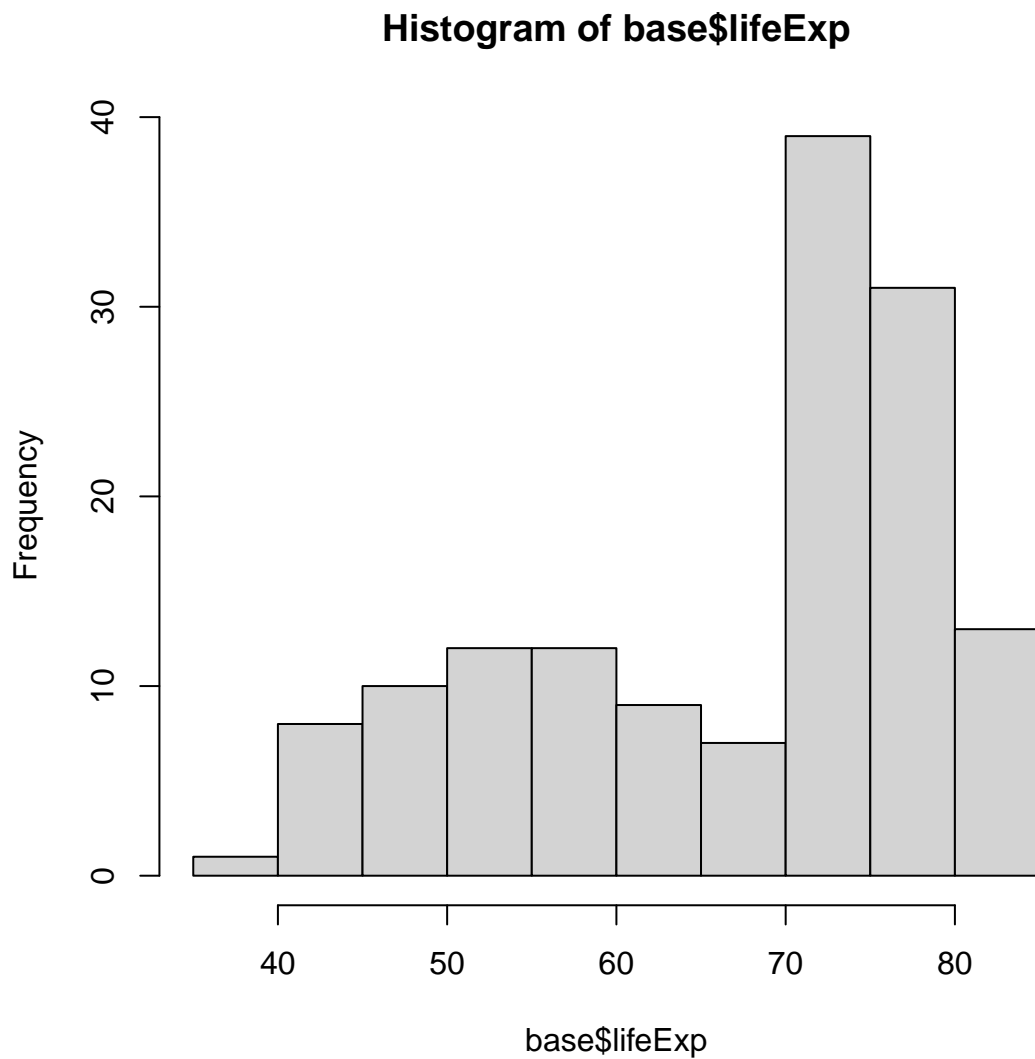
glimpse(base)
```

```
## Rows: 142
## Columns: 6
## $ country   <fct> "Afghanistan", "Albania", "Algeria", "Angola", "Argentina", ~
## $ continent <fct> Asia, Europe, Africa, Africa, Americas, Oceania, Europe, Asi~
## $ year      <int> 2007, 2007, 2007, 2007, 2007, 2007, 2007, 2007, 2007, 2007, ~
## $ lifeExp   <dbl> 43.828, 76.423, 72.301, 42.731, 75.320, 81.235, 79.829, 75.6~
## $ pop       <int> 31889923, 3600523, 33333216, 12420476, 40301927, 20434176, 8~
## $ gdpPercap <dbl> 974.5803, 5937.0295, 6223.3675, 4797.2313, 12779.3796, 34435~
```

GGPlot2 - a base de tudo

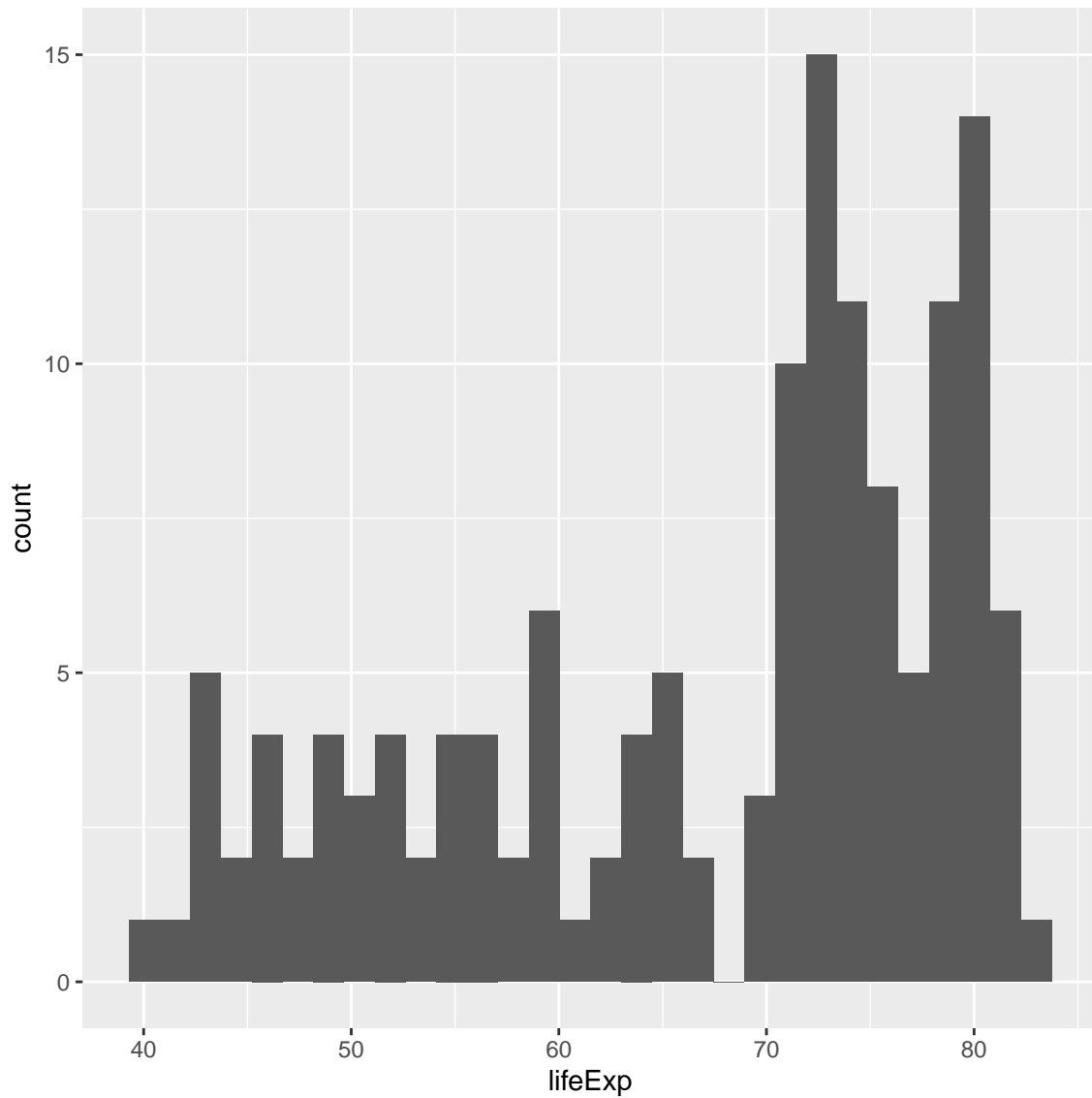
O R por si só possui funções para gerar gráficos, porém o ggplot2 é um pacote que fornece um conjunto bem extenso de possibilidades

```
hist(base$lifeExp)
```



Vamos criar um histograma sobre a variável expectativa de vida.

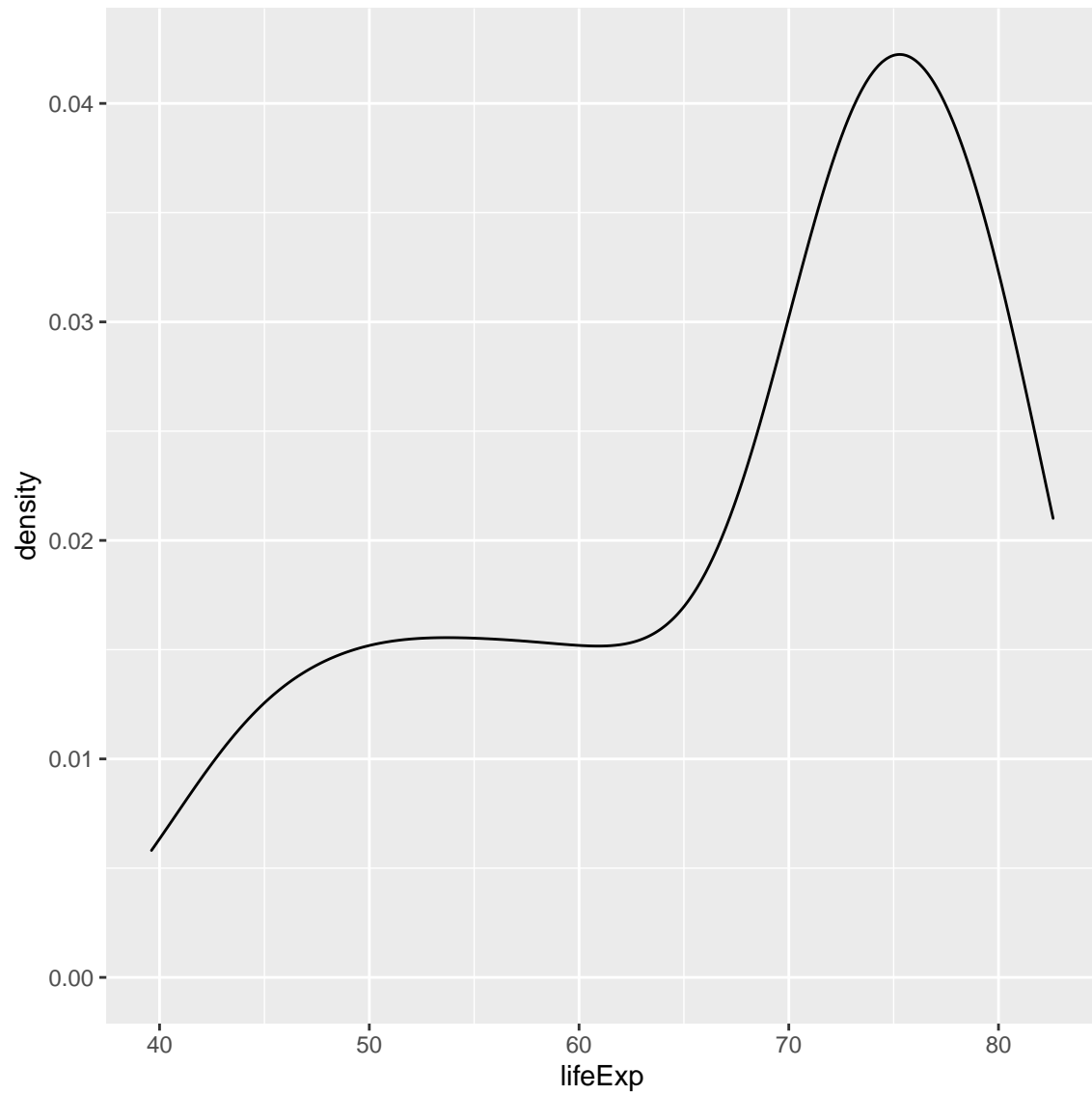
```
ggplot(base, aes(x = lifeExp)) + geom_histogram()
```

Densidade

Vamos criar um gráfico de densidade sobre a variável expectativa de vida. Veja outra forma de usar a função ggplot.

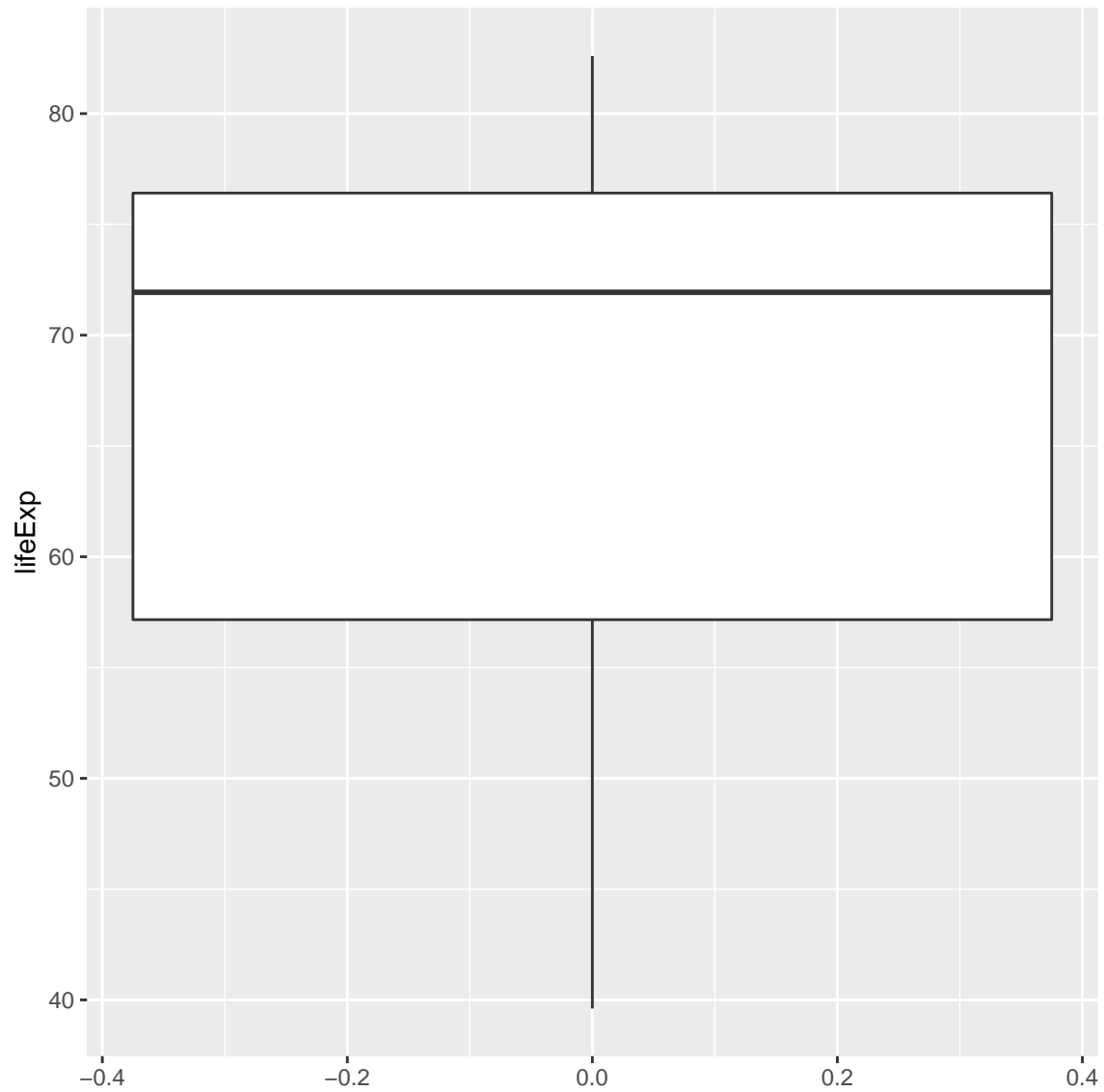
```
base %>%  
  ggplot(aes(x = lifeExp)) + geom_density()
```



Boxplot

Vamos criar um boxplot sobre a variável expectativa de vida.

```
base %>%  
  ggplot(aes(y = lifeExp)) + geom_boxplot()
```



Vamos criar outro boxplot sobre a variável expectativa de vida.

```
base %>%  
  ggplot(aes(x = continent, y = lifeExp)) + geom_boxplot()
```

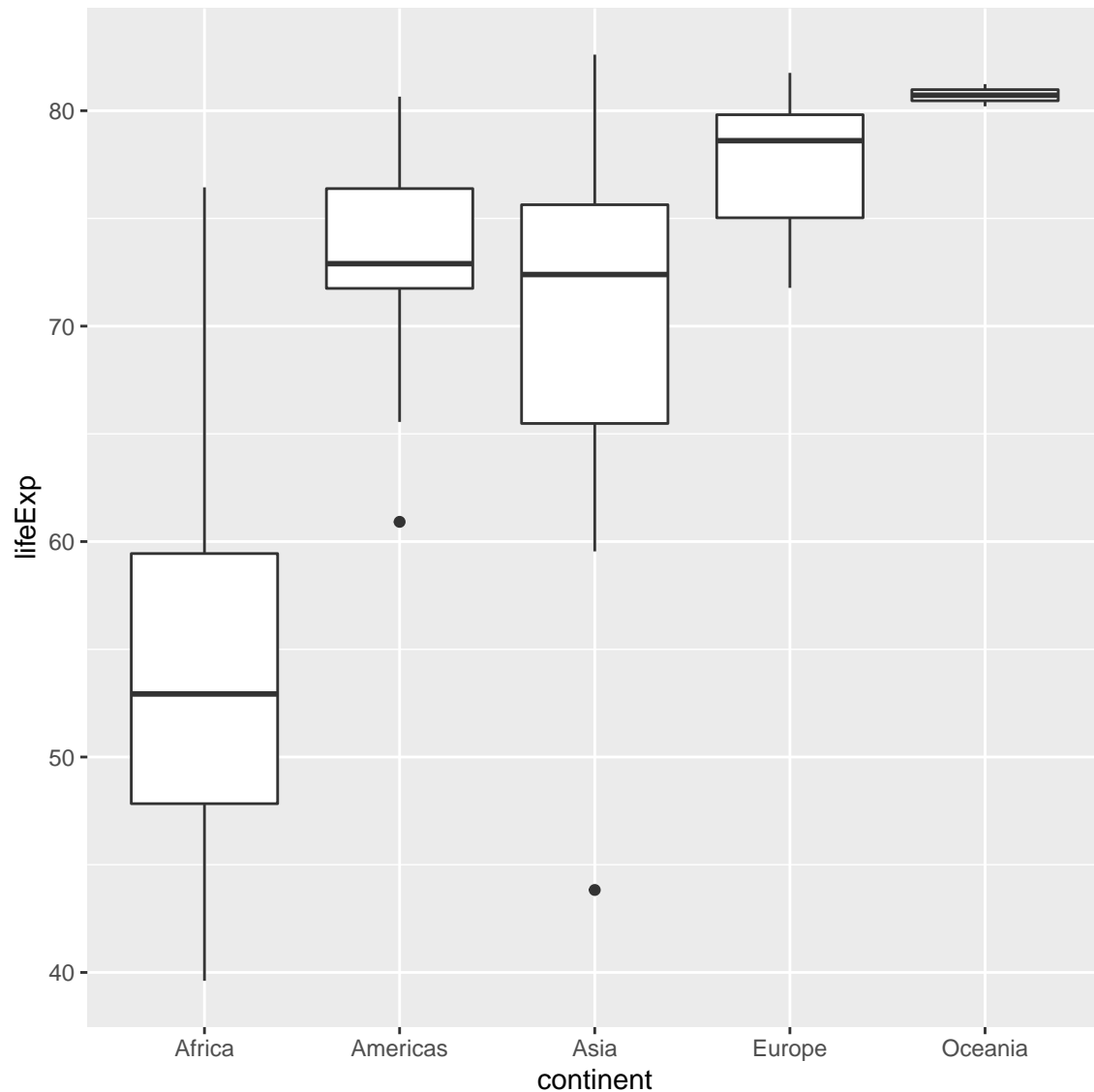
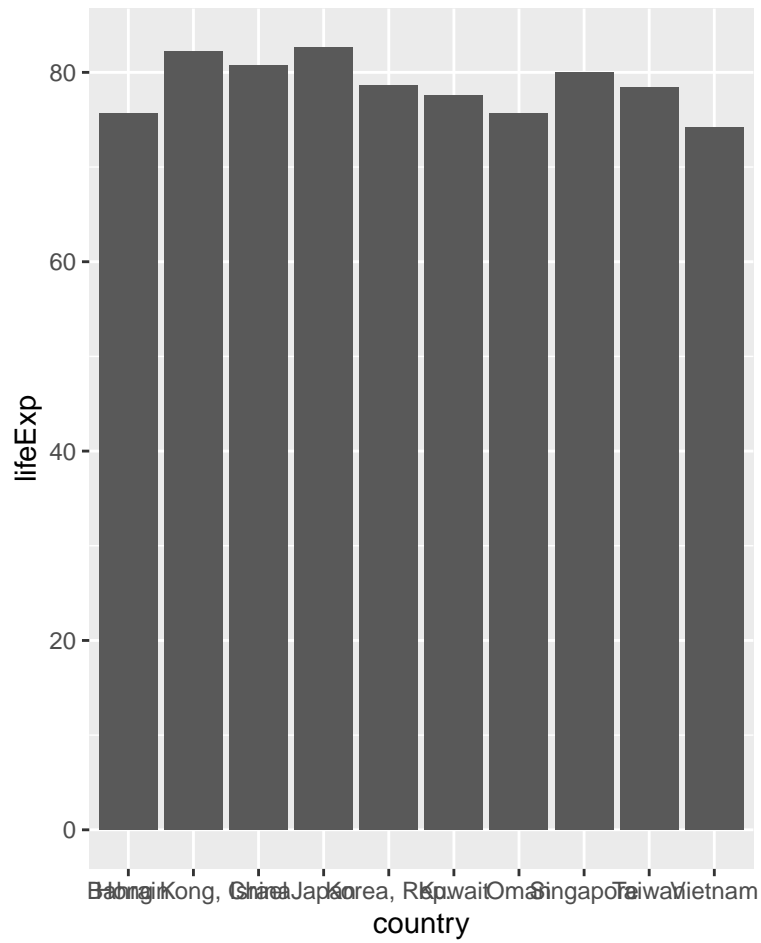


Gráfico de colunas

Vamos criar um gráfico de barras da variável expectativa de vida. Nesse caso, vamos manter os 10 países da Ásia com maior taxa de expectativa de vida.

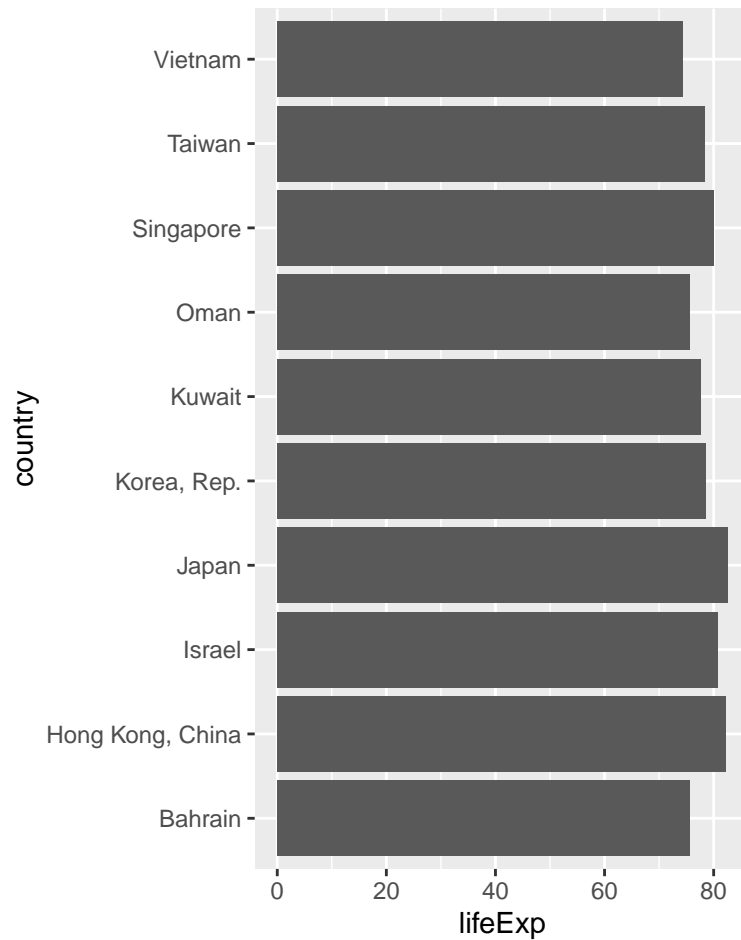
```
base %>%
  filter(continent == "Asia") %>%
  top_n(n = 10, wt = lifeExp) %>%
  ggplot(aes(x = country, y = lifeExp)) + geom_col()
```



Coord_flip

Veja como melhorar a visualização com um simples argumento:

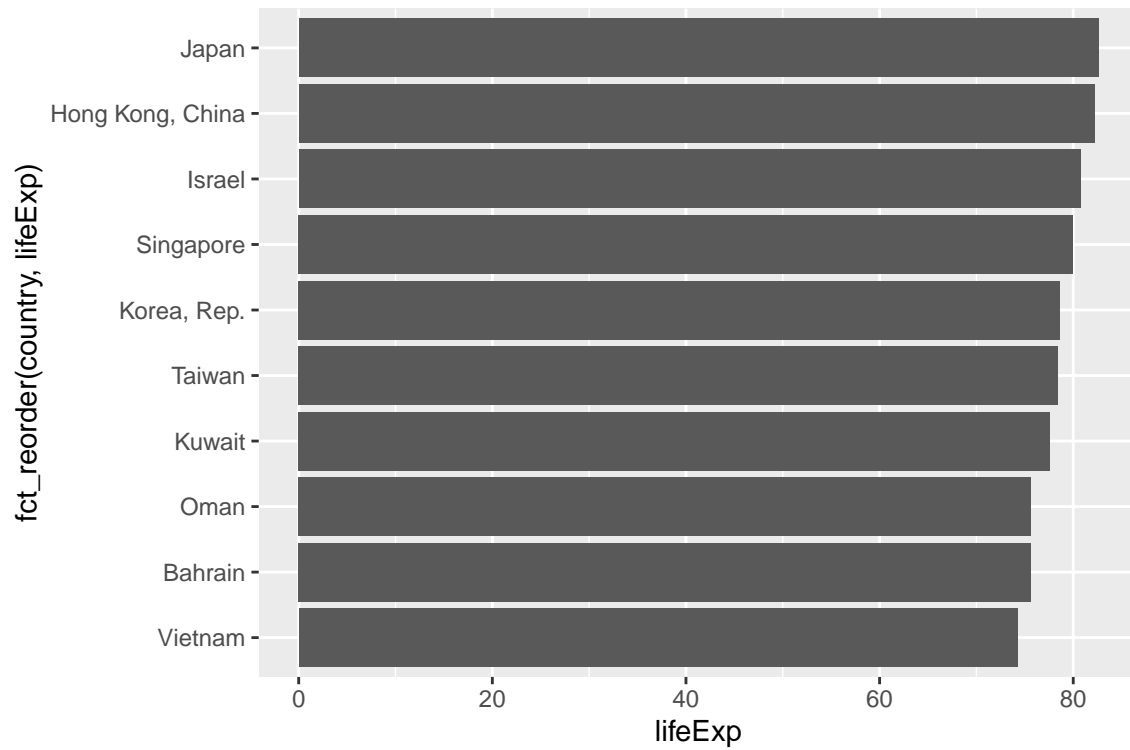
```
base %>%
  filter(continent == "Asia") %>%
  top_n(n = 10, wt = lifeExp) %>%
  ggplot(aes(x = country, y = lifeExp)) + geom_col() + coord_flip()
```



Mas consigo deixar em ordem?

Sim, para isso, vamos usar uma função do pacote forcats.

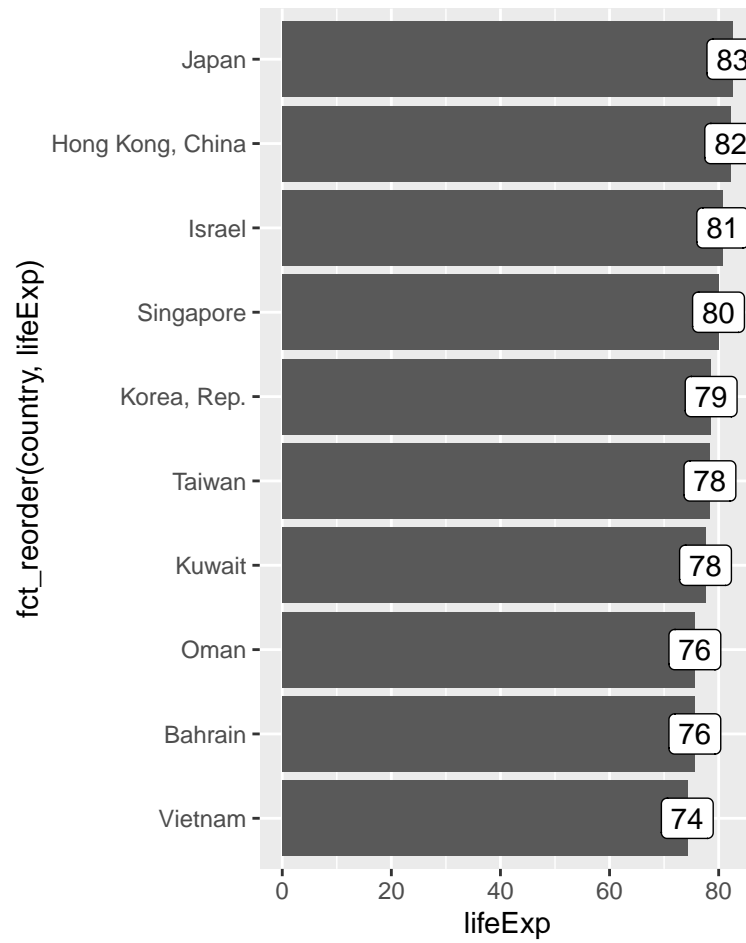
```
library(forcats)
base %>%
  filter(continent == "Asia") %>%
  top_n(n = 10, wt = lifeExp) %>%
  ggplot(aes(x = fct_reorder(country, lifeExp), y = lifeExp)) + geom_col() + coord_flip()
```



Labels

Adicionando labels

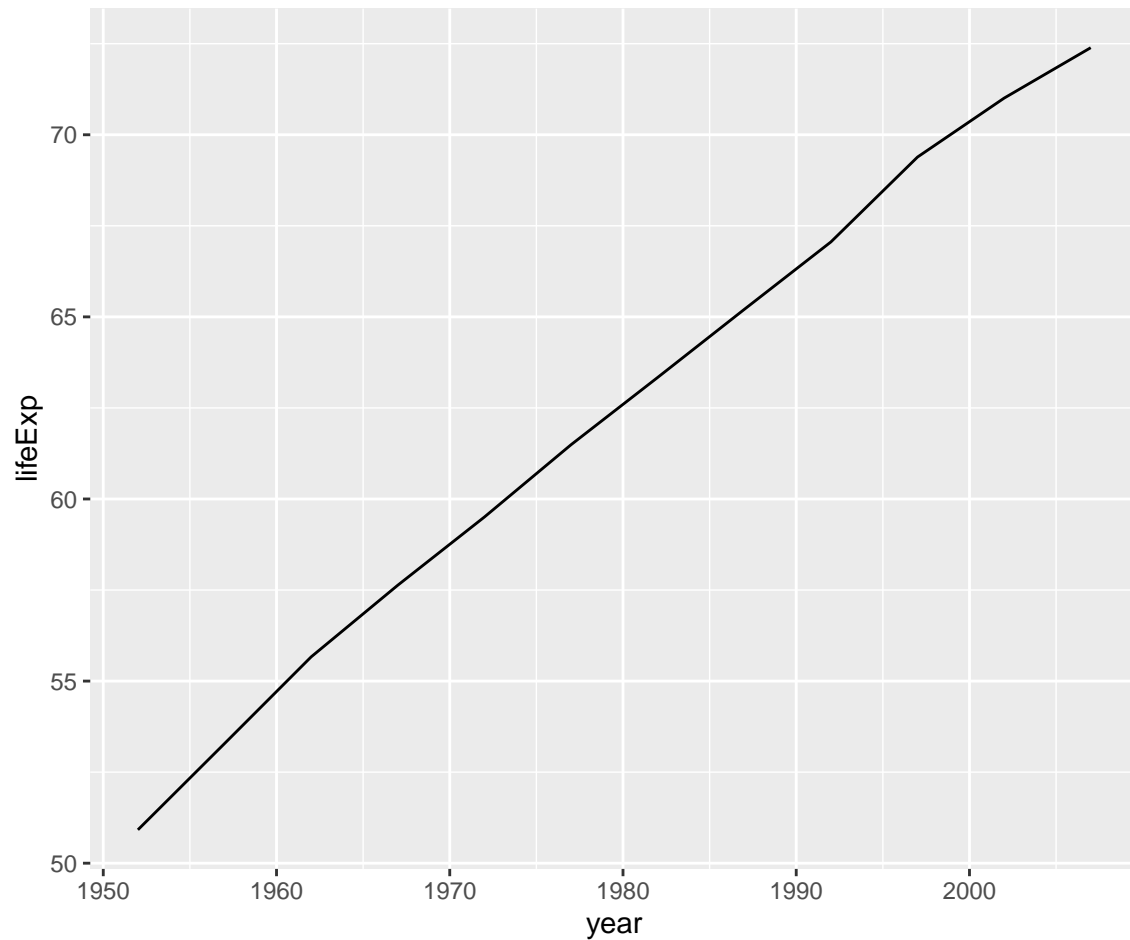
```
base %>%
  filter(continent == "Asia") %>%
  top_n(n = 10, wt = lifeExp) %>%
  ggplot(aes(x = fct_reorder(country, lifeExp), y = lifeExp)) + geom_col() +
  geom_label(aes(label=round(lifeExp))) + coord_flip()
```



Linhas

Para fazer esse gráfico, vamos usar a base original, gapminder, e filtrar apenas o Brasil.

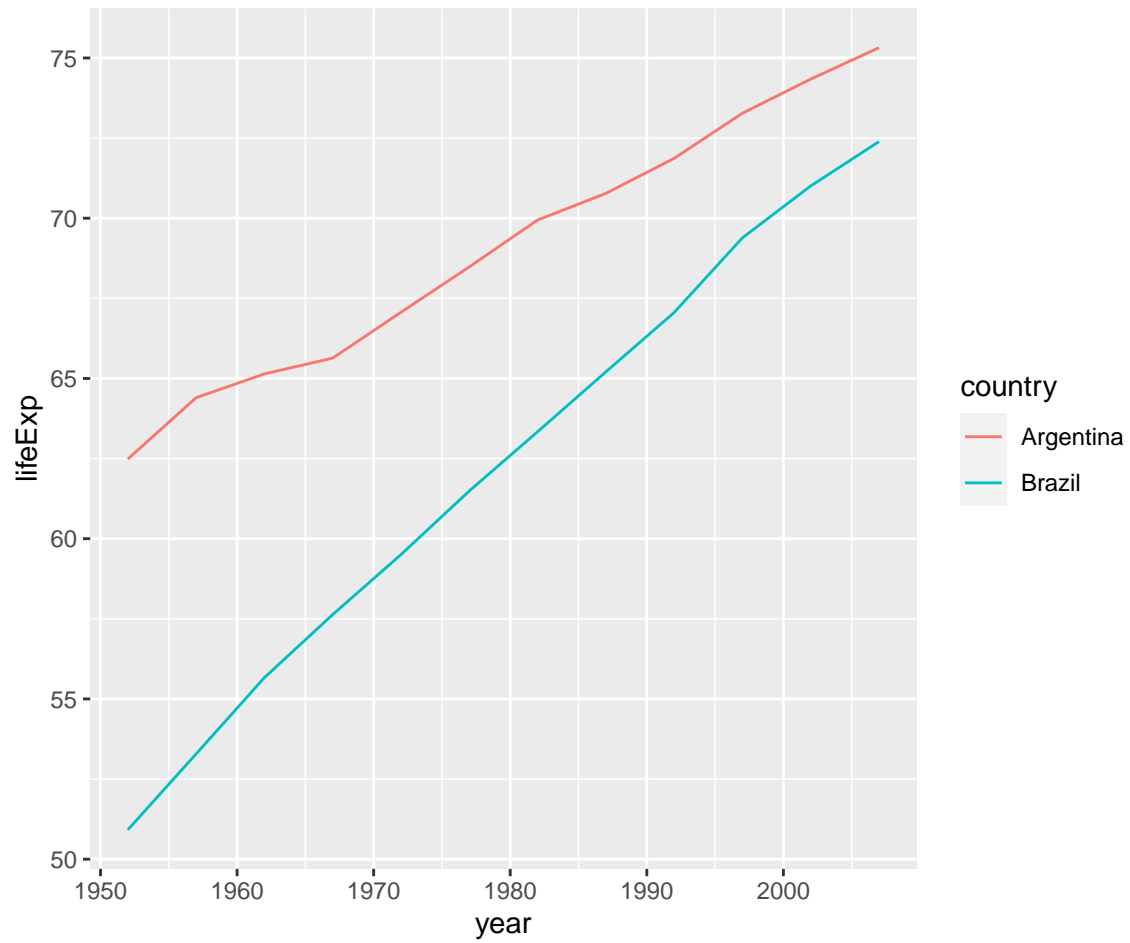
```
gapminder %>%  
  filter(country == "Brazil") %>%  
  ggplot(aes(x = year, y = lifeExp)) + geom_line()
```

Argumento col

Veja que legal é o argumento `col` dentro do `aes`! Nesse caso pegamos Brasil e Argentina. Olha como estamos filtrando... Primeiro criamos um vetor com o nome dos países e depois aplicamos o filter usando o `%in%`.

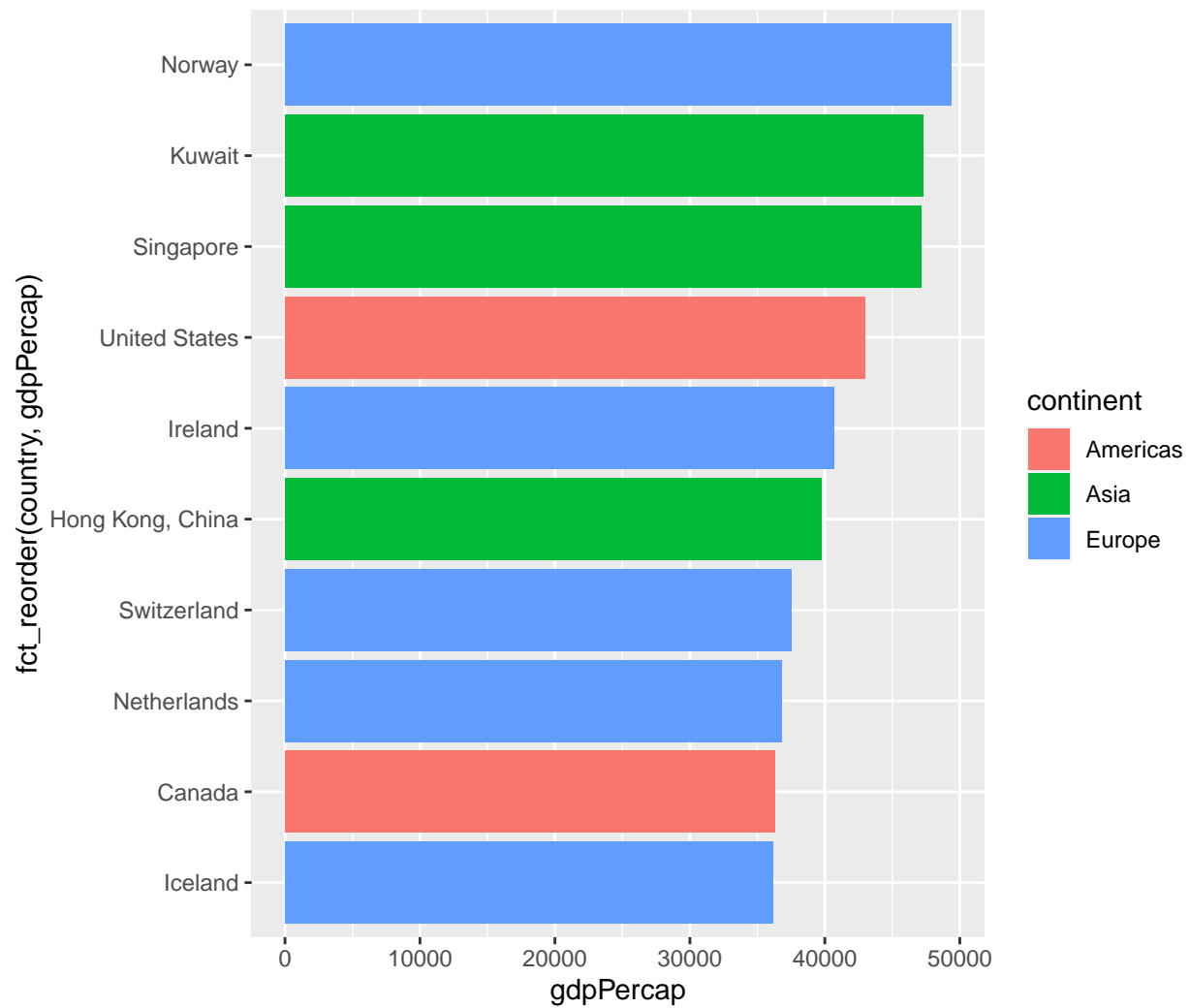
```
países <- c("Brazil", "Argentina")
gapminder %>%
  filter(country %in% países) %>%
  ggplot(aes(x = year, y = lifeExp, col = country)) + geom_line()
```



Usando argumento fill

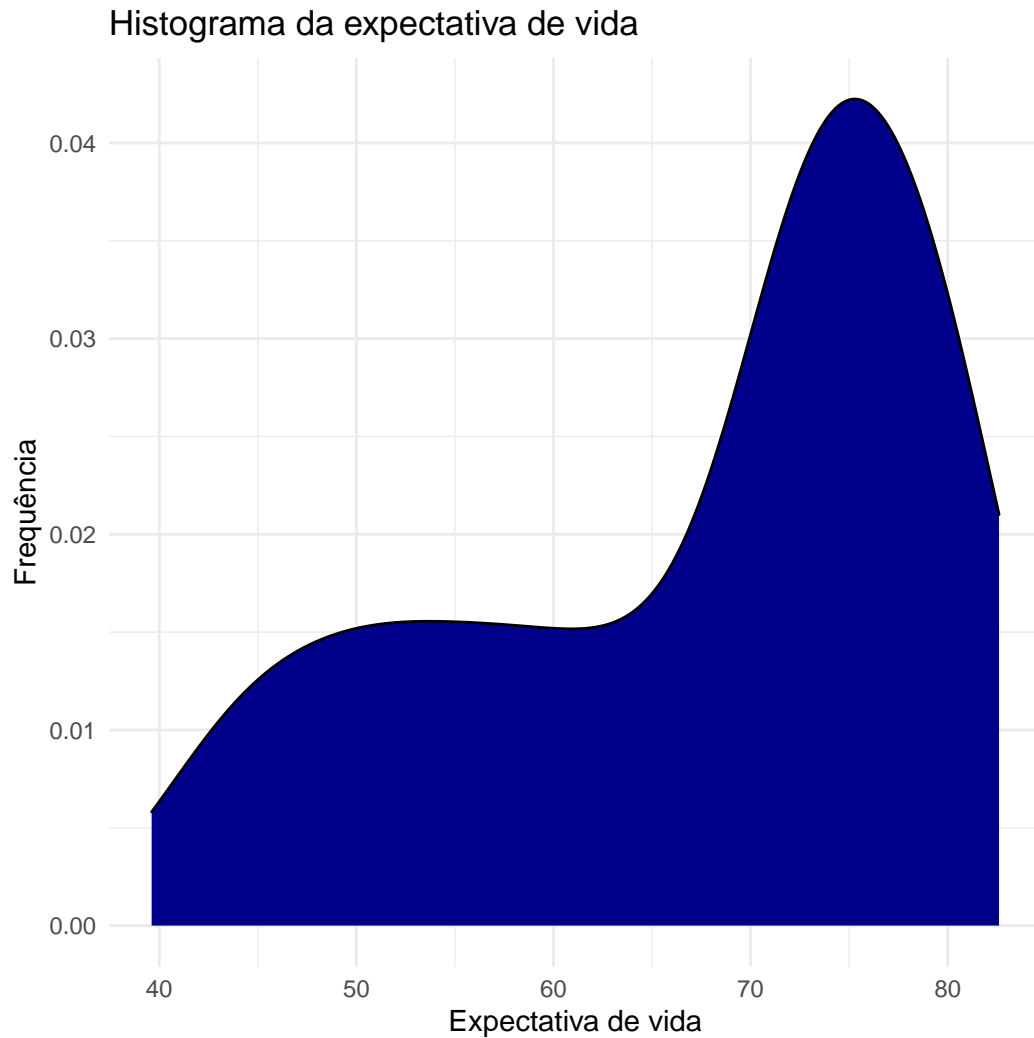
Vamos ver como funciona o argumento `fill`.

```
base %>%  
  top_n(10) %>%  
  ggplot(aes(x = fct_reorder(country,gdpPercap), y=gdpPercap, fill = continent)) + geom_col() + coord_f
```



Podemos customizar mais elementos?

```
ggplot(base,aes(x=lifeExp)) + geom_density(fill="darkblue") +
  labs(title = "Histograma da expectativa de vida",
        x = "Expectativa de vida", y = "Frequência") + theme_minimal()
```

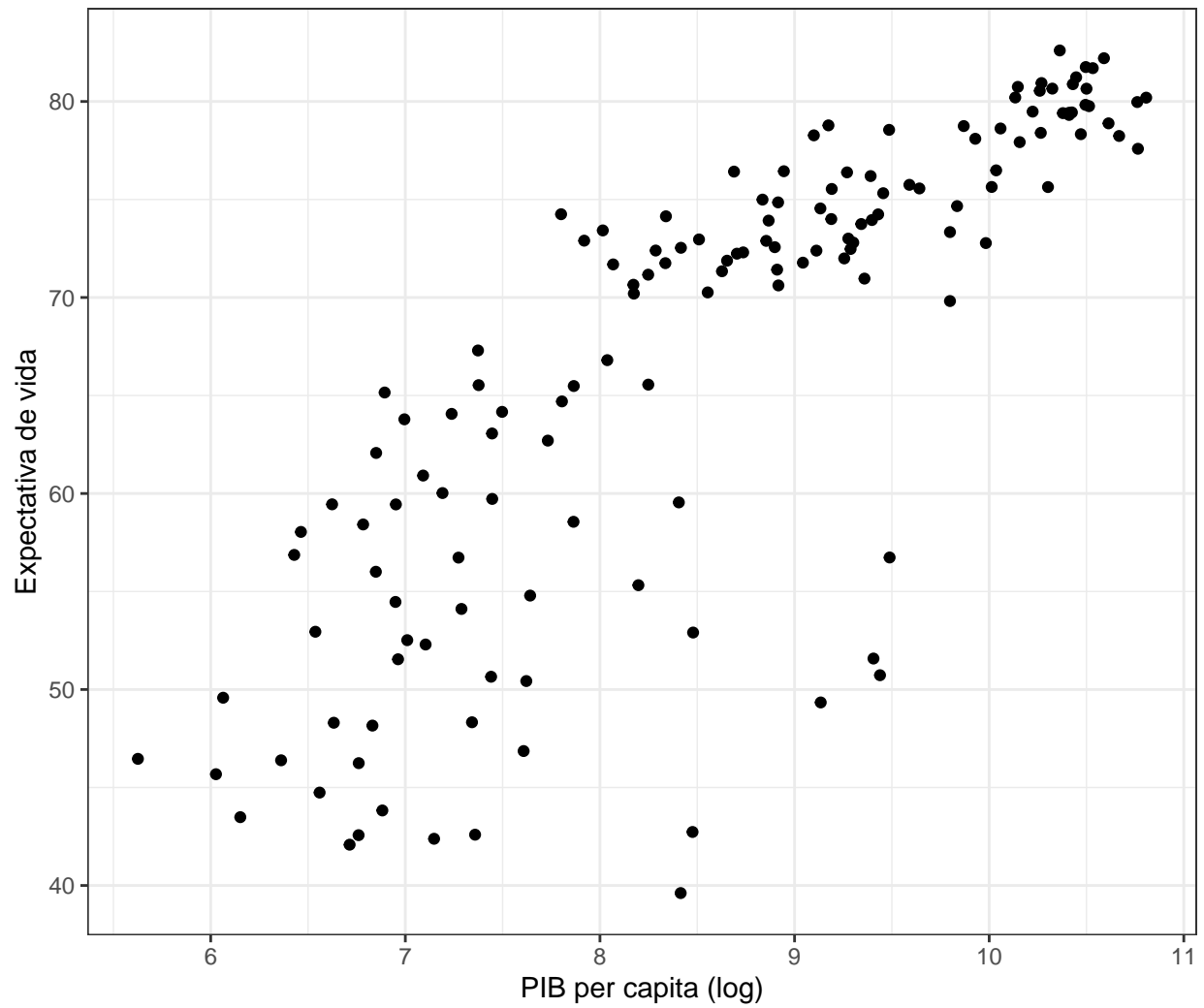


Dá para customizar uma muitos outros elementos: cores de fundo, de linhas, tamanho de letras, posição da legenda e assim por diante.

Vamos trabalhar com duas variáveis

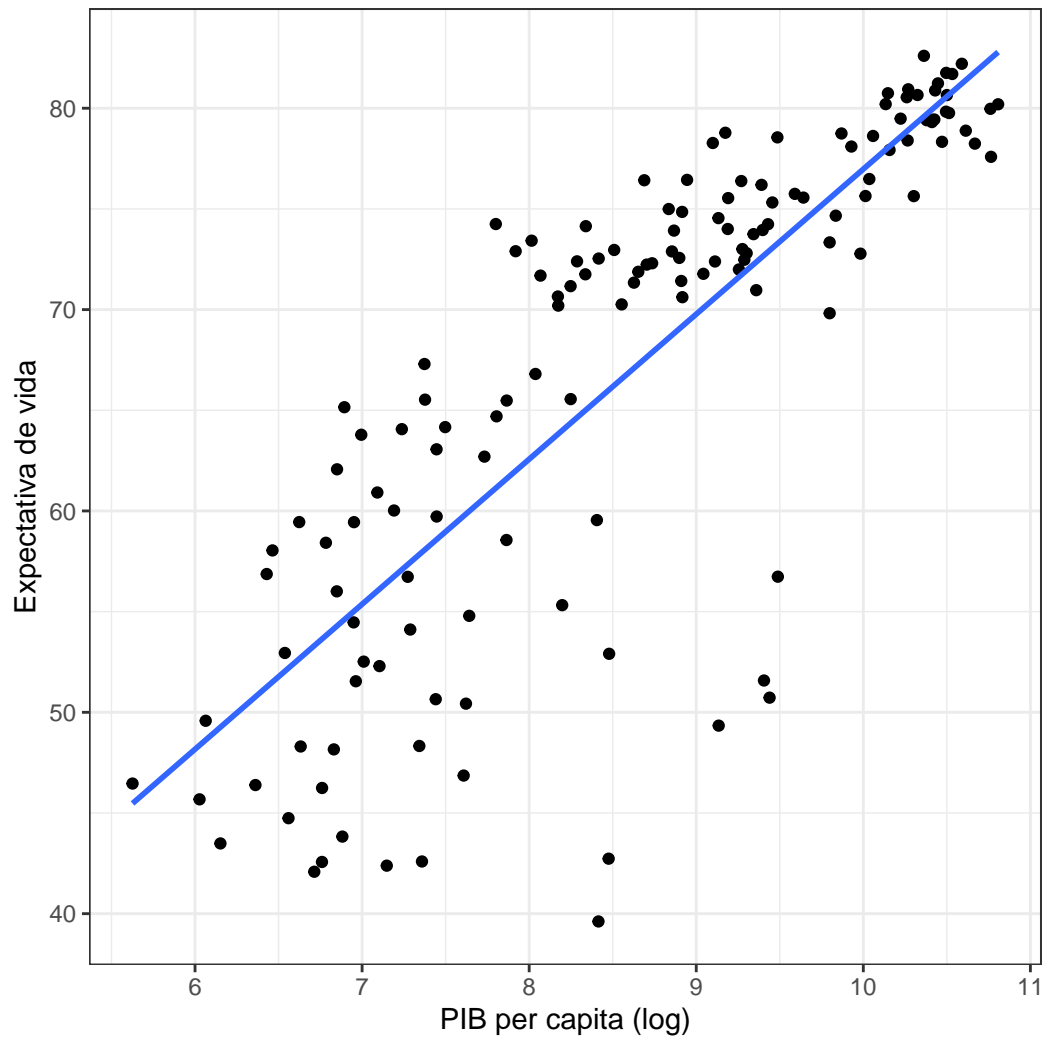
Vamos relacionar duas variáveis: PIB per capita e expectativa de vida

```
ggplot(base,aes(x=log(gdpPercap),y=lifeExp)) + geom_point() +  
  labs(x = "PIB per capita (log)", y = "Expectativa de vida") + theme_bw()
```



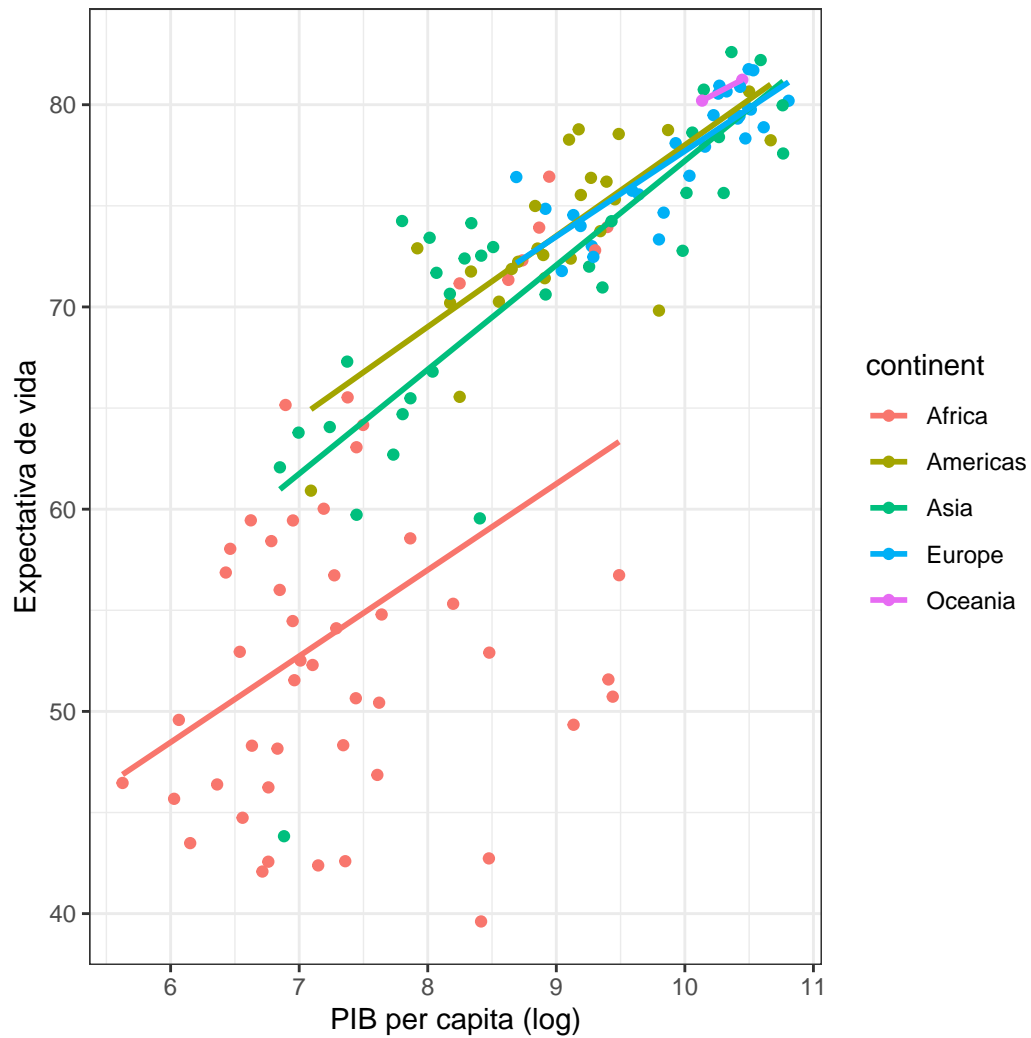
Mais uma camada

```
ggplot(base,aes(y=lifeExp, x=log(gdpPercap))) + geom_point() +  
  geom_smooth(method = "lm", se=FALSE) +  
  labs(x = "PIB per capita (log)",  
       y = "Expectativa de vida") + theme_bw()
```



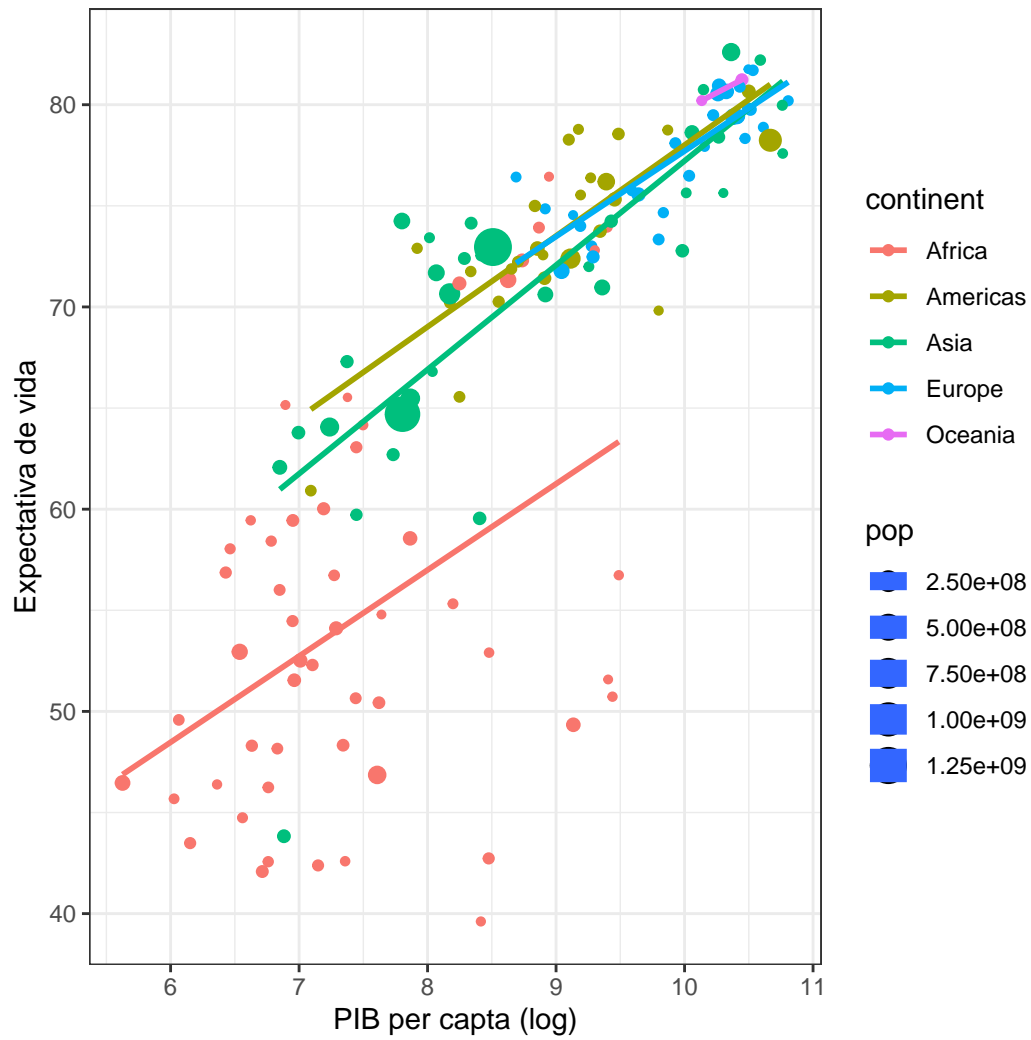
Argumento col

```
ggplot(base,aes(y=lifeExp, x=log(gdpPercap), col=continent)) + geom_point() +  
  geom_smooth(method = "lm", se=FALSE) +  
  labs(x = "PIB per capita (log)",  
       y = "Expectativa de vida") + theme_bw()
```



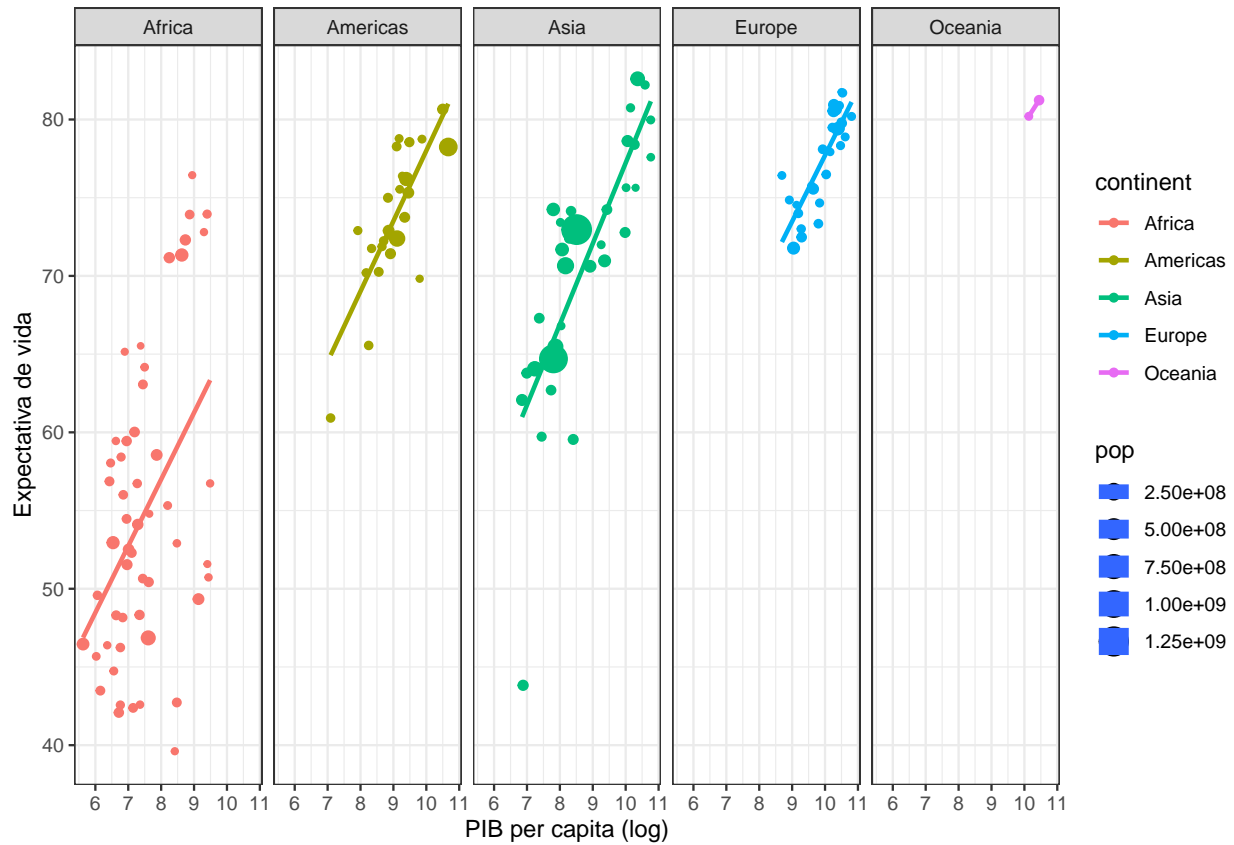
Argumento size

```
ggplot(base,aes(y=lifeExp, x=log(gdpPercap), col=continent,size = pop)) + geom_point() +
  geom_smooth(method = "lm", se=FALSE) +
  labs(x = "PIB per capta (log)",
       y = "Expectativa de vida") + theme_bw()
```



Facet

```
ggplot(base,aes(x=log(gdpPercap), y=lifeExp, col=continent,
                size = pop)) + geom_point() + geom_smooth(method = "lm", se=FALSE) +
  labs(x = "PIB per capita (log)", y = "Expectativa de vida") + theme_bw() +
  facet_grid(~continent)
```

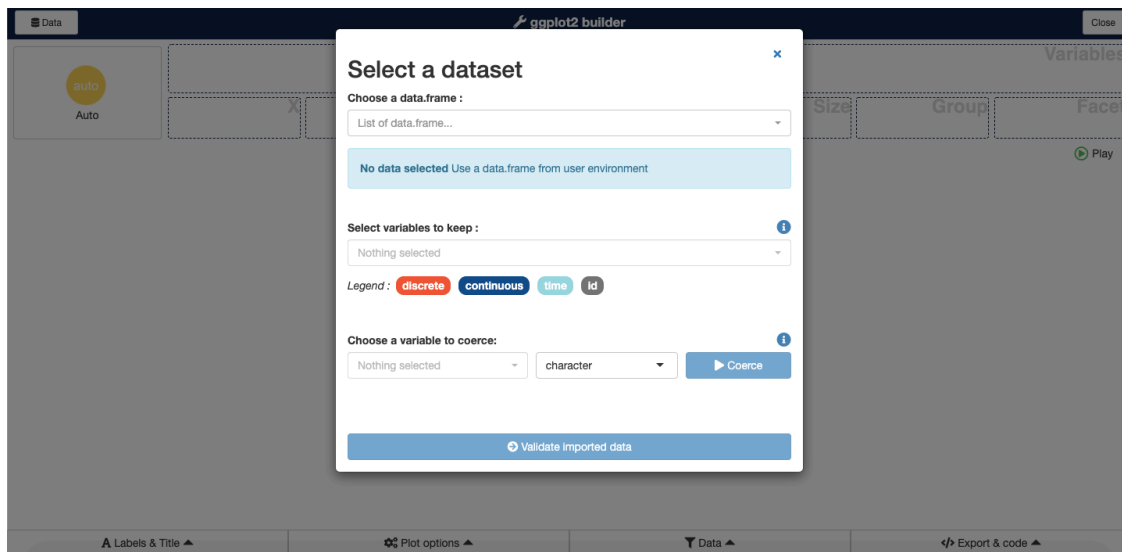



Criando gráficos com Esquisse

O esquisse permite visualizar gráficos no estilo ggplot2 usando point n' click. É uma ótima ferramenta para “estudar” o ggplot2.

```
library(esquisse)

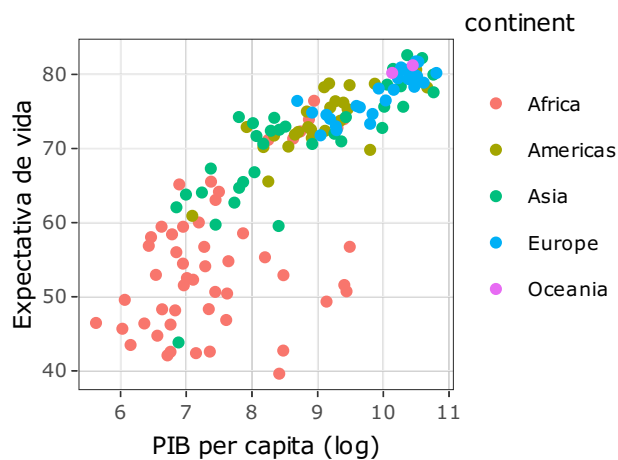
esquisser(viewer = "browser")
```



Gerando gráficos interativos

Vamos resgatar um gráfico que fizemos lá atrás.

```
library(plotly)
grafico <- ggplot(base,aes(y=lifeExp, x=log(gdpPercap), col=continent)) + geom_point() +
  labs(x = "PIB per capita (log)",
       y = "Expectativa de vida") + theme_bw()
ggplotly(grafico)
```



Para aprender mais

Ao clicar será direcionad@ para a página indicada

- [GGPlot2 cheat sheet](#)
- [GGPlot2 - LAPEI](#)
- [Plotly](#)
- [Blog Datanovia](#)
- [Datavis with R](#)
- [Live Curso-R sobre extensões do ggplot2](#)
- [The R graph gallery](#)
- [Documentação dplyr](#)
- [Documentação ggplot2](#)

Obrigado

Se encontrou algum erro ou tem alguma sugestão de melhoria pode entrar em contato. Se foi útil também avise! Ah... e pode passar adiante. Quanto mais gente tendo acesso a esse conhecimento melhor!

Daniel Pagotto | danielppagotto@gmail.com | [LinkedIn](#)

LAPEI - UFG | lapeiufg@gmail.com. | No instagram: [lapeiufg](#)