Lista 1: Revisão de Probabilidade e Introdução ao uso do R

FLP 0468/FLS 6183

Prof. Manoel Galdino
Gabriel Mardegan
Pedro Reis

Para entregar até: 06 de setembro de 2024

Introdução ao R

Esta primeira lista tem como objetivo revisar aspectos fundamentais de estatística e probabilidade, além de introduzir as alunas ao uso do software R. O envio dos resultados dos exercícios das listas, via Moodle, deverão conter: 1) um arquivo em formato ".Rmd", com o script completo, incluindo os códigos utilizados; 2) um arquivo em formato ".pdf" gerado no próprio R, através do RMarkdown (linguagem de marcação).

Para utilizar o R, é preciso instalar duas ferramentas. Primeiro, a linguagem de programação: R. E, em seguida, um programa/editor/interface que nos ajuda a digitar e organizar a nossa análise: RStudio.

1. Para baixar **R para Windows**: segue o link: https://cran.r-project.org/bin/windows/base/, clique no primeiro link, e executar. Para outros sistemas, veja: https://cran.r-project.org/bin/.

2. Para baixar o **RStudio Desktop** apropriado para o seu sistema e executar: https://posit.co/download/rstudio-desktop/#download

Observe que você nunca precisa abrir o R diretamente. Tudo pode ser feito no RStudio.

Para gerar os dois arquivos a serem enviados pelo Moodle, os scripts deverão ser estruturados através do RMarkdown. O RMarkdown é uma linguagem de marcação que possibilita a criação de vários tipos de documento (.html, .pdf, .docx) a partir do seu script em R. Para gerar um script com RMarkdown basta seguir este caminho a partir da barra de ferramentas:

"File" → "New File" → "RMarkdown"

Dê um nome para seu arquivo, como "lista1_gabriel", e assim você terá um script em formato ".Rmd", a partir do qual você pode gerar documentos em PDF através da função "Knit" (compilar/executar), que aparece como um ícone azul acima do seu script. Para gerar um PDF, entretanto, é preciso instalar a linguagem LaTeX no seu computador. *Para informações essenciais sobre como o relatório em PDF*, leiam: https://jonnyphillips.github.io/Analise_de_Dados_2022/introducao.html.

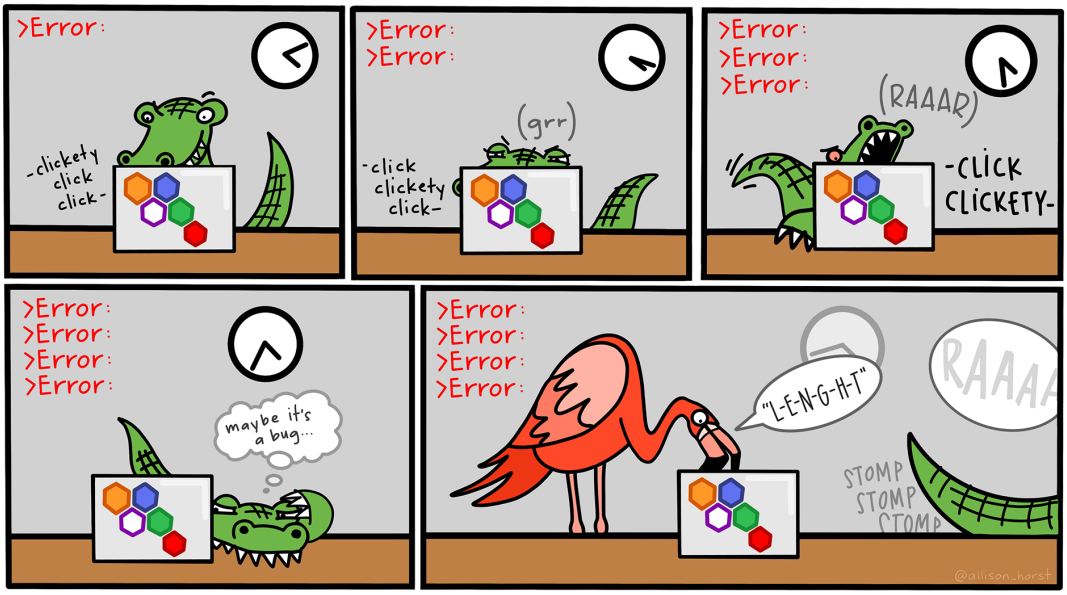
Como material de apoio, recomendamos os seguintes links:

- https://jonnyphillips.github.io/Ciencia_de_Dados/ (Material do curso ministrado pelo professor Jonathan Phillips no DCP/USP, em português, com as gravações das aulas)

- https://r4ds.hadley.nz/ (Livro *R for Data Science, 2nd edition*, em inglês)

Uma das principais funcionalidades do R é o trabalho com bases de dados, item indispensável para muitas áreas da ciência, incluindo as ciências sociais. Através do programa, é possível importar uma base de dados, limpá-la de modo a facilitar sua visualização, selecionar variáveis de interesse, produzir novas variáveis, gerar gráficos e tabelas, e também realizar operações, de modo a ampliar a compreensão sobre os dados e variáveis com os quais estamos trabalhando. É por esse caminho que chegaremos no tópico central deste curso: a regressão linear.

Antes de começarmos, recomendamos: *escreva você mesmo os códigos!* É a melhor forma de aprender R e se familiarizar com os códigos, mesmo se errarmos nas primeiras (não só primeiras!) tentativas. E *peça ajuda* para as pessoas: colegas, monitores, e professores.



Fonte: Allison Horst

Vamos começar a análise de dados no R usando o banco de dados `gapminder`, uma tabela com dados de expectativa de vida, população e PIB per capita por país e ano. Esse banco não está automaticamente embutido no R: baixemos o pacote `gapminder` usando a função `install.packages()` – só é preciso instalar uma vez o pacote de interesse. Após instalado, ativamos o pacote usando a função `library()`; é preciso rodar essa função no script toda vez que o conteúdo do pacote (funções/dados) for utilizado, de preferência em seu início.

```
install.packages("gapminder")
#install.packages("gapminder") # colocamos o jogo da velha no começo
                               # para não executar a função
                               # o jogo da velha # cria um comentário
library(gapminder)
```

Agora, temos acesso à base `gapminder`. Vejamos uma pequena amostra do banco: a função `head()` mostra as primeiras observações da base e o argumento `n =` mostra o número de observações a serem visualizadas. Observemos as primeiras 5 observações para termos uma ideia:

```r
head(gapminder, n = 5)
```

```
## # A tibble: 5 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.
```

O R sozinho oferece diversas funções para a análise de dados. Porém, hoje em dia, usamos um conjunto de pacotes que nos oferece novos jeitos de analisarmos os dados pelo R: o pacote (ou meta-pacote `tidyverse`). Esse pacote nos dá novas funções de limpeza, manipulação, e visualização de dados e torna essas tarefas mais intuitivas para nós.

```r
#install.packages("tidyverse")
library(tidyverse)
```

Uma das funções oferecidas pelo `tidyverse` é o `glimpse()`. Ela nos permite ter uma noção maior dos dados que temos.

```r
glimpse(gapminder)
```

```
## Rows: 1,704
## Columns: 6
## $ country   <fct> "Afghanistan", "Afghanistan", "Afghanistan", "Afghanist~
## $ continent <fct> Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, A~
## $ year      <int> 1952, 1957, 1962, 1967, 1972, 1977, 1982, 1987, 1992, 1~
## $ lifeExp   <dbl> 28.80, 30.33, 32.00, 34.02, 36.09, 38.44, 39.85, 40.82,~
## $ pop       <int> 8425333, 9240934, 10267083, 11537966, 13079460, 1488037~
## $ gdpPercap <dbl> 779.4, 820.9, 853.1, 836.2, 740.0, 786.1, 978.0, 852.4,~
```

O que a função nos mostra sobre a tabela `gapminder` é: o banco tem 1704 observações e 6 variáveis; a variável "country" (país) é uma variável categórica (`fct` ou `factor`); a variável "continent" (continente) é uma variável categórica (`fct` ou `factor`); a variável "year" (ano) é uma variável numérica discreta (`int` ou `integer`); a variável "lifeExp" (expectativa de vida) é uma variável numérica contínua (`dbl` ou `dbl`); a variável "pop" (população) é uma variável numérica discreta (`int` ou `integer`); e a variável "gdpPercap" (PIB per capita) é uma variável numérica contínua (`dbl` ou `dbl`).

Vemos que a unidade de medida dessa tabela é país-ano. Quantos países-anos há por continente? Podemos responder a essa pergunta combinando duas funções do `tidyverse`: i) `group_by()`, que agrupa os nossos dados pelas categorias da variável (ou variáveis) escolhida; isto é, se agruparmos por continente e quisermos calcular a média depois, o cálculo será feito por continente e não na base inteira. ii) `tally()`, que conta o número de observações. Combinadas, contaremos o número de observações por categoria.

```r
gapminder %>%
    group_by(continent) %>%
    tally() %>%
    ungroup() # boas práticas de programação!

## # A tibble: 5 x 2
##   continent     n
##   <fct>     <int>
## 1 Africa      624
## 2 Americas    300
## 3 Asia        396
## 4 Europe      360
## 5 Oceania      24
```

Antes de discutirmos os resultados, precisamos nos perguntar: o que são esses %>%? Por que não colocamos a nossa tabela dentro dessas funções?

Como dissemos, o `tidyverse` facilita o trabalho de análise de dados. %>% se chama "pipe"e transforma o nosso fluxo de funções numa receita: pegue isso e faça isso, depois faça isso, e depois isso. Com o pipe, não precisamos colocar nossa tabela dentro das funções; colocamos a nossa tabela em evidência no código e, com os pipes, executamos nossas funções de interesse em sequência. Assim, pegamos a tabela `gapminder` e agrupamos por "continent"; agrupado por essa variável, contamos o número de observações; após contarmos, desagrupe.

Se não tivéssemos usado o pipe, teríamos que executar essas mesmas funções da seguinte maneira:

```r
# forma sem %>%
ungroup(tally(group_by(gapminder, continent)))
# ocupa menos linhas, mas...
# ...não é muito intuitivo assim, né?
```

O que o resultado desse conjunto de funções nos mostra sobre a amostra é que a África possui 624 países-anos, a América possui 300 países-anos, a Ásia possui 396 países-anos, a Europa 360 países-anos, e a Oceania 24.

Agora, suponhamos que queremos saber o PIB per capita médio apenas no continente americano? Usamos a função `filter()` para manter apenas observações que satisfaçam a condição lógica que inserirmos dentro dessa função: nesse caso, mantenha apenas as observações cujo continente é a América (`continent == "Americas"`) – repare que usamos dois sinais de igual (=) pois estabelecemos uma condição lógica entre os dois lados (variável e categoria).

Após filtrarmos, usamos a função `summarise()`, que nos permite executar funções de medida-resumo sobre nossa tabela e criar uma nova tabela com isso.

```r
gapminder %>%
    filter(continent == "Americas") %>%
    summarise(mean_gdppc = mean(gdpPercap)) # um nome para a medida-resumo
```

```
## # A tibble: 1 x 1
##   mean_gdppc
##        <dbl>
## 1      7136.
```

Vemos que o PIB per capita médio no continente americano (lembremos que a unidade de medida é país-ano, não só país) é 7136,0 dólares.

Podemos usar múltiplas condições na função `filter()`. Suponhamos que queremos ver a expectativa de vida média apenas no continente asiático e apenas para países-anos cujas populações excedam 2 milhões de habitantes.

```
gapminder %>%
    filter(
        continent == "Asia",
        pop > 2000000 # ou 2e6
    ) %>%
    summarise(media_expec_vida = mean(lifeExp))

## # A tibble: 1 x 1
##   media_expec_vida
##              <dbl>
## 1             60.2
```

E se quisermos criar novas variáveis e renomear as colunas existentes? Podemos usar a função `mutate()` para criar novas variáveis (ou modificar as existentes) e a função `rename()` para renomear as variáveis. Vamos fazer o seguinte: vamos pegar a tabela original; depois, vamos renomear todas as variáveis por algo mais intuitivo para nós e no nosso idioma; depois de renomearmos as variáveis, vamos criar a variável "pib" (PIB per capita vezes o tamanho da população) e depois criar a variável "log_pib_percap", onde pegamos a variável original de PIB per capita e calculamos seu logaritmo natural – isso é muito útil para variáveis com distribuições assimétricas, como veremos adiante.

Bom, sabemos como fazer essas transformações. Mas, queremos usar essa tabela transformada para outras coisas no futuro. O que fazer? Precisamos guardar essa tabela transformada em um novo objeto (que daremos o nome de `gapminder_novo`). Para guardarmos nossas transformações em um novo objeto, escrevemos o nome do nosso novo objeto primeiro, escrevemos o operador de atribuição `<-` (atalho: `Alt + -`), e tudo que escrevermos depois desse operador será guardado no nosso novo objeto.

```
gapminder_novo <- # colocar a tabela modificada em um novo objeto
    gapminder %>%
    rename( # a regra é "nome novo" = "nome antigo"
        "populacao"      = "pop",
        "pib_per_capita" = "gdpPercap",
        "ano"            = "year",
        "pais"           = "country", # sem acento para facilitar
```

```
        "continente"     = "continent",
        "expec_vida"     = "lifeExp"
    ) %>%
    mutate(
        pib            = pib_per_capita * populacao, # var 1 vezes var 2
        log_pib_percap = log(pib_per_capita) # logaritmo natural
    )

head(gapminder_novo) # sem o argumento n = , a função imprime 6 observações
```

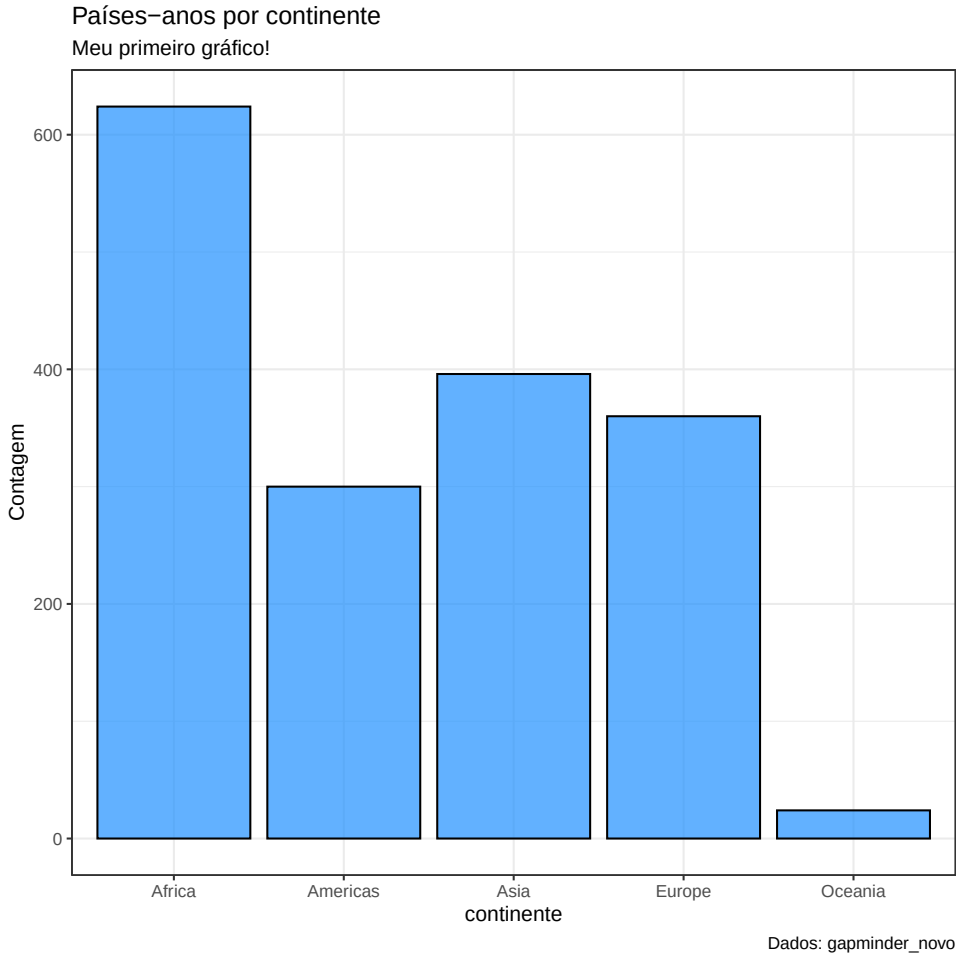
```
## # A tibble: 6 x 8
##   pais        continente   ano expec_vida populacao pib_per_capita      pib
##   <fct>       <fct>      <int>      <dbl>     <int>          <dbl>    <dbl>
## 1 Afghanistan Asia        1952       28.8   8425333           779.  6.57e 9
## 2 Afghanistan Asia        1957       30.3   9240934           821.  7.59e 9
## 3 Afghanistan Asia        1962       32.0  10267083           853.  8.76e 9
## 4 Afghanistan Asia        1967       34.0  11537966           836.  9.65e 9
## 5 Afghanistan Asia        1972       36.1  13079460           740.  9.68e 9
## 6 Afghanistan Asia        1977       38.4  14880372           786.  1.17e10
## # i 1 more variable: log_pib_percap <dbl>
```

Veremos um pouco mais sobre R na revisão de probabilidade e nos exercícios, mas, para finalizar a introdução ao R nessa lista, vamos ver como visualizar gráficos no estilo `tidyverse`: usaremos o pacote `ggplot2`, que usa o dialeto *Grammar of Graphics* para criar visualizações. O que é isso? Para criarmos um gráfico, criaremos "camadas" de funções geométricas (isto é, os tipos de gráficos e componentes que inseriremos na nossa visualização).

Vamos ter mais concretude com um exemplo: queremos um gráfico de barras que nos mostre a quantidade de países-anos por continente. Sabemos essa informação pela combinação `group_by()` e `tally()` acima, mas queremos uma imagem disso. Vamos usar a tabela transformada que guardamos anteriormente.

```
gapminder_novo %>%
    ggplot(aes(x = continente)) + # preste atenção no + para funções do ggplot2
    geom_bar( # gráfico de barras
        alpha = 0.7, # transparência
        colour = "black", # cor nas bordas
        fill = "dodgerblue1" # cor no interior
    ) +
    labs( # nomes dos eixos e título
        y = "Contagem",
        title = "Países-anos por continente",
        subtitle = "Meu primeiro gráfico!",
        caption = "Dados: gapminder_novo"
```

```
) +
theme_bw() # estética geral do gráfico
```



O que fizemos aqui?

- Primeiro, pegamos a nossa tabela nova.
- Depois, começamos com a função `ggplot()` para dizer que faremos um gráfico de agora em diante.
- Dentro da função `ggplot()`, inserimos a função `aes()` para dizer que o gráfico que fizermos dirá respeito às variáveis que inserirmos nessa função. No caso, queremos só a variável "continente" no eixo x.
- Agora, queremos um gráfico de barras. Usamos a função `geom_bar()`. Dentro dessa função, podemos mudar algumas configurações: por exemplo, a transparência das barras (`alpha =` , que vai de 0 a 1); a cor das bordas das barras (`colour =` , ou `color =`); e a cor do interior das barras (`fill =`).
- Até aqui, já criamos o gráfico de barras. Não precisamos fazer mais nada. Mas, queremos mudar mais umas coisas: vamos mudar o nome do eixo y, colocar um título, um subtítulo, e uma anotação. Para isso, usamos a função `labs()` – reparemos que como não mexemos no nome do eixo x, ele usa o nome da variável da tabela analisada.

- Por fim, vamos mudar a estética geral do gráfico, torná-lo mais "publicável": uma estética de preferência de um dos monitores é a `theme_bw()`, mas há outras – veremos outras no futuro.
- Quando finalizamos, não colocamos um sinal de adição depois da última função.

Revisão de Probabilidade

Probabilidade

Probabilidade é uma linguagem. Uma linguagem formal, que possui determinadas regras lógicas, para fazer sentido da nossa incerteza sobre o mundo, sobre o resultado de eventos *aleatórios*.

Esses eventos (chamemos esses "eventos" de *resultados*) são "aleatórios" não porque são imprevisíveis, mas porque se originaram de um *processo gerador* (aleatório), o qual não conhecemos, e que gera resultados possíveis e mutuamente exclusivos, cada um com uma chance de ocorrência associada.

A *probabilidade* de um resultado é a proporção de vezes que aquele evento ocorreu (dentre todos os resultados possíveis), no longo prazo. Por sua vez, o *espaço amostral* é o conjunto de todos os possíveis resultados. Um *evento* é um subconjunto do espaço amostral, logo, pode possuir um ou mais resultados.

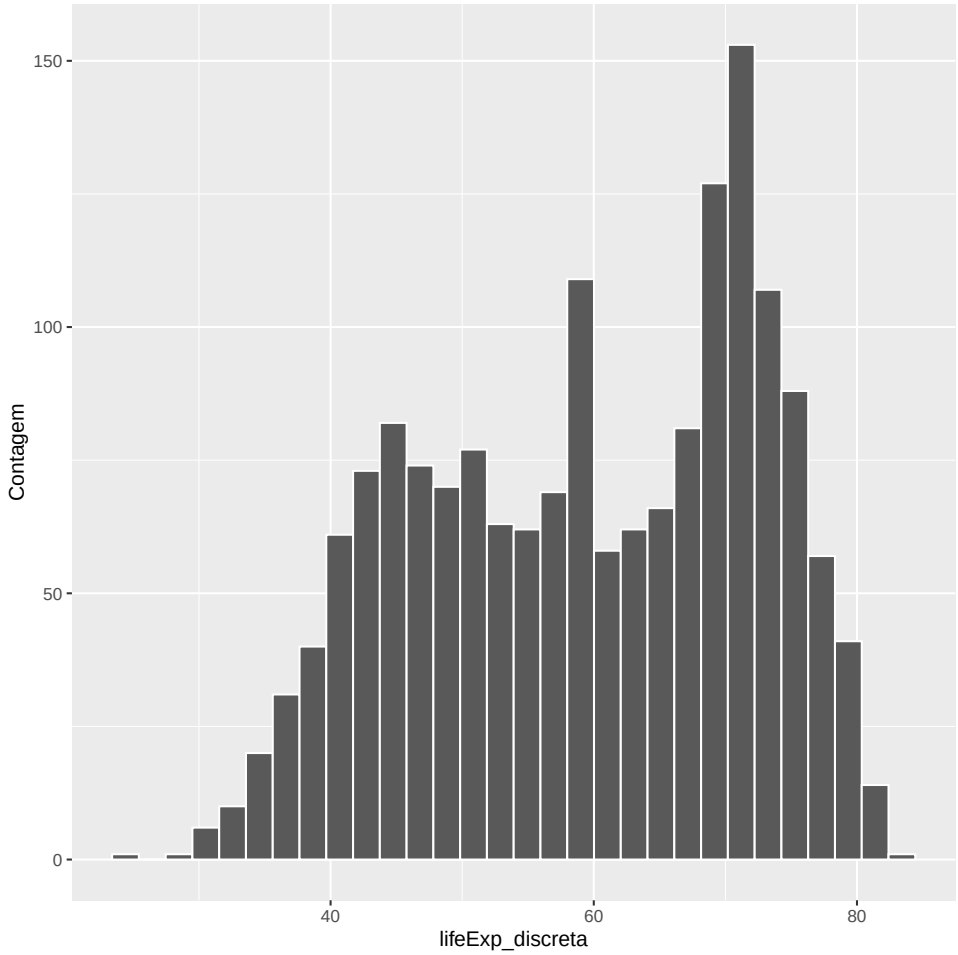
Uma variável aleatória

Uma *variável aleatória* (ou v.a.) é um resumo numérico de um resultado aleatório. Algumas variáveis aleatórias são *discretas*: essas assumem um conjunto discreto de valores, como $\{0, 1, 2, ...\}$. Outras são *contínuas*: essas assumem um contínuo não-contável de valores possíveis.

- Para *variáveis discretas*: a *função massa de probabilidade* ou *distribuição de probabilidade* de uma variável aleatória discreta é a lista de todos os valores possíveis da variável e probabilidade de ocorrência de cada valor. Em variáveis discretas, podemos computar a probabilidade de um evento singular a partir de sua distribuição de probabilidade. A *distribuição acumulada de probabilidade* ou *função de distribuição acumulada* é a probabilidade de que a variável aleatória é menor ou igual a um valor específico.
- Para *variáveis contínuas*: a *função densidade de probabilidade* de uma variável aleatória contínua, por lidar com um contínuo virtualmente infinito de valores, deve ser tratada com intervalos definidos de valores para descobrir a probabilidade de que a variável se encontra dentro desse intervalo. A *distribuição acumulada de probabilidade* ou *função de distribuição acumulada* para variáveis contínuas é definida da mesma forma que para variáveis discretas.

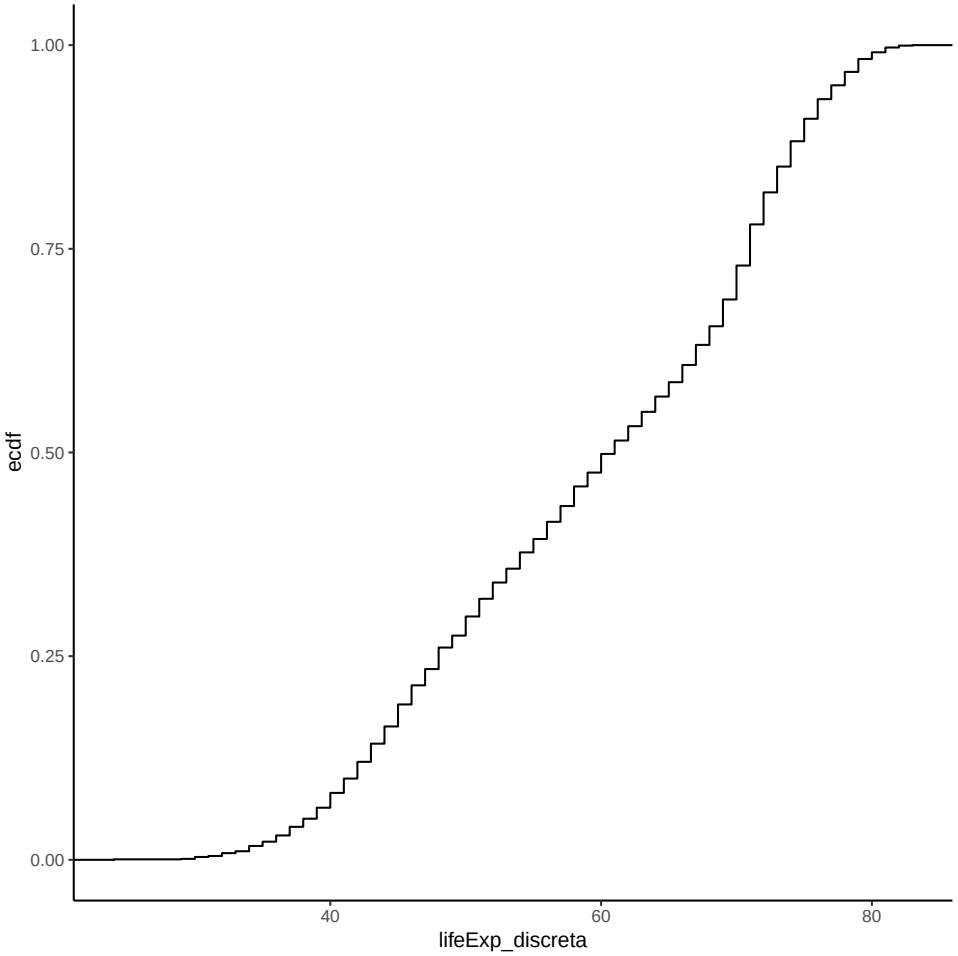
Vejamos a implementação desses conceitos no R, voltando ao exemplo do banco `gapminder`. Abaixo, criamos uma variável numérica discreta: expectativa de vida de cada país-ano arredondada sem decimais, com isso a variável se torna "contável". Para visualizarmos sua massa, um gráfico apropriado é um histograma.

```r
gapminder %>%
    mutate(lifeExp_discreta = round(lifeExp, 0)) %>% # arredondar
    ggplot(aes(x = lifeExp_discreta)) +
    labs(y = "Contagem") + # título do eixo y
    geom_histogram(colour = "white") # bordas brancas
```



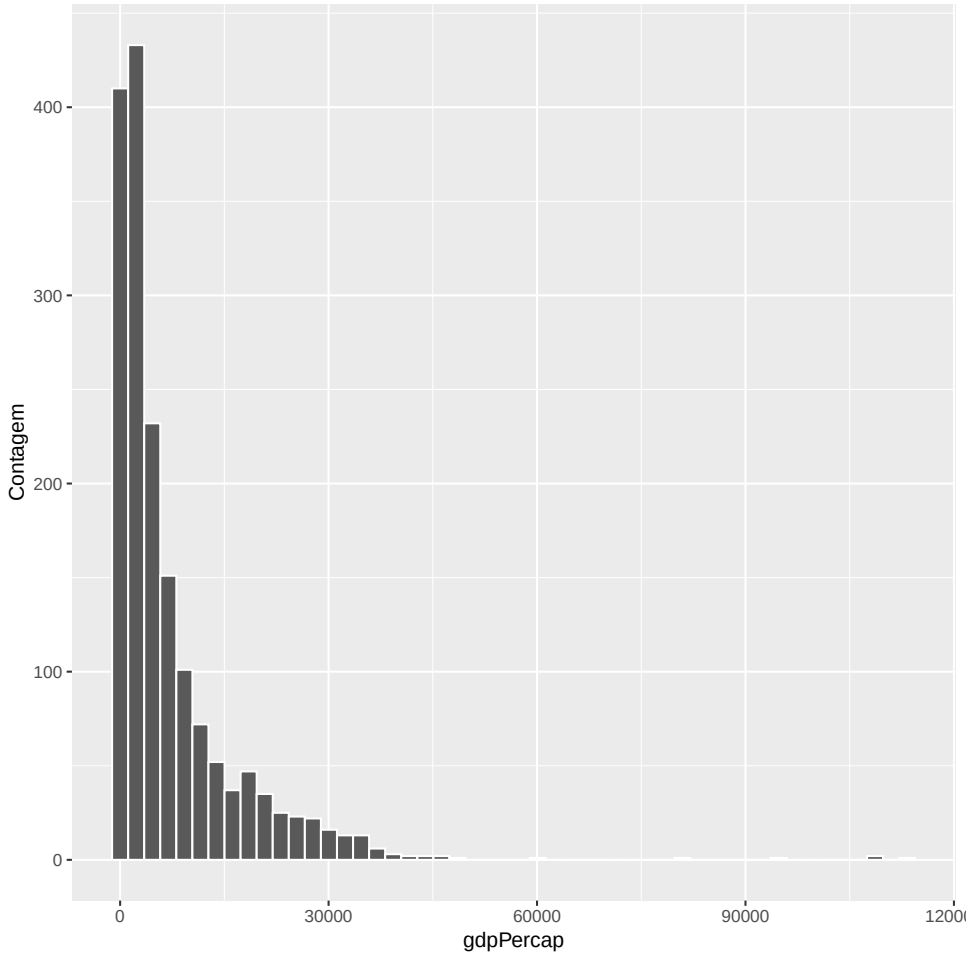
Podemos criar visualizar a função de distribuição acumulada empírica dessa variável. Podemos reparar como, por ser uma variável discreta, a função se movimenta por esses "degraus".

```r
gapminder %>%
    mutate(lifeExp_discreta = round(lifeExp, 0)) %>% # arredondar
    ggplot(aes(x = lifeExp_discreta)) +
    stat_ecdf() + # distribuição acumulada empírica
    theme_classic() # outra estética
```



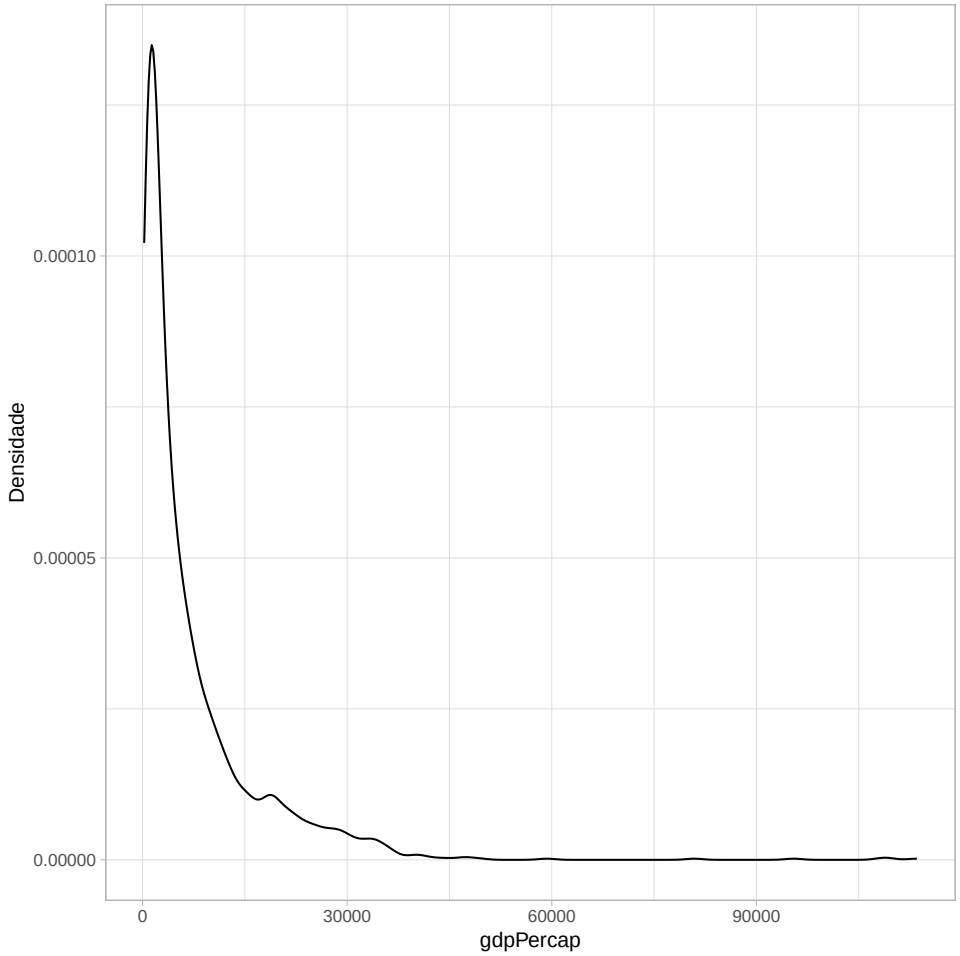
No banco `gapminder`, temos uma variável contínua importante: PIB per capita ("gdpPercap"). Para variáveis contínuas, podemos usar histogramas também.

```
gapminder %>%
    ggplot(aes(x = gdpPercap)) +
    labs(y = "Contagem") + # título do eixo y
    geom_histogram(
        bins = 50, # aumentar o número de barras
        colour = "white" # bordas brancas
    )
```



Mas, muitas vezes é mais apropriado usar uma versão mais suavizada: um gráfico de densidade.

```
gapminder %>%
    ggplot(aes(x = gdpPercap)) +
    labs(y = "Densidade") + # título do eixo y
    geom_density() + # gráfico de densidade kernel
    theme_light() # mais outra estética
```



É notável observarmos nesse gráfico que, na nossa amostra, há uma grande concentração de países-anos com renda baixa. A distribuição de renda é bastante assimétrica – uma observação assim, porém, é muito casual: precisamos de números para justificar essas observações.

Esperança, variância, assimetria e curtose

Esperança

A *esperança* ou valor esperado de uma v.a. Y, denotada $E(Y)$ ou μ_Y, é o valor médio, no longo prazo, da v.a. ao longo de muitos experimentos (ocorrências) repetidos. O valor esperado de uma v.a. discreta é calculado como a média ponderada dos possíveis resultados daquela variável, onde os pesos são as probabilidades de cada resultado.

$$\mathbb{E}[Y] = \sum_y y \times \Pr(Y = y) \tag{1}$$

A esperança de uma v.a. contínua, por sua vez, também é a média ponderada pela probabilidade dos possíveis resultados:

$$\mathbb{E}[Y] = \int_y y \cdot f(y)\, dy \tag{2}$$

A esperança é uma quantidade teórica e desconhecida que buscamos identificar a partir da nossa amostra. A melhor "forma matemática"[1] para identificar a esperança de uma v.a. é a média aritmé-

[1] O termo adequado é "estimador", mas esse será um assunto para o futuro.

tica.

$$\bar{Y} = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{3}$$

Vamos calcular a média da variável "lifeExp" no nosso banco `gapminder`.

```
mean(gapminder$lifeExp)

## [1] 59.47
```

Vemos que a expectativa de vida média na nossa amostra de países-anos é 59,47 anos.

Variância

A *variância* de uma v.a. Y, denotada $\text{var}(Y)$ ou σ^2_Y é o valor esperado do quadrado do desvio de Y em relação a sua média. Formalmente:

$$\begin{aligned}
\text{var}(Y) &= \mathbb{E}\left[(Y - \mathbb{E}(Y))^2\right] \\
&= \mathbb{E}\left[Y^2 - 2Y\mathbb{E}(Y) + [\mathbb{E}(Y)]^2\right] \\
&= \mathbb{E}[Y^2] - 2 \times \mathbb{E}[Y\mathbb{E}(Y)] + \mathbb{E}[[\mathbb{E}(Y)]^2] \\
&= \mathbb{E}[Y^2] - 2 \times \mathbb{E}(Y) \times \mathbb{E}(Y) + [\mathbb{E}(Y)]^2 \\
&= \mathbb{E}[Y^2] - 2[\mathbb{E}(Y)]^2 + [\mathbb{E}(Y)]^2 \\
&= \mathbb{E}[Y^2] - [\mathbb{E}(Y)]^2
\end{aligned} \tag{4}$$

É uma medida de dispersão da distribuição de probabilidade da v.a. Porém, como a unidade de medida da variância envolve o quadrado de Y, sua interpretação é complicada. Por isso, é comum mensurarmos a dispersão mediante o *desvio padrão*, a raiz quadrada da variância: $\sqrt{\sigma^2_Y} = \sigma_Y$.

Variância ou desvio padrão são quantidades populacionais desconhecidas por nós. Por isso, precisamos estimá-las mediante uma forma adequada: variância amostral ou desvio padrão amostral.

$$\hat{\sigma}^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{X})^2 \qquad \text{(Variância amostral)} \tag{5}$$

$$\hat{\sigma} = \sqrt{\hat{\sigma}^2} \qquad \text{(Desvio padrão amostral)} \tag{6}$$

Por que o denominador $n-1$? Por uma prova simples, vemos que, para pequenas amostras, usar o denominador n erra sistematicamente a variância populacional; o $n-1$ corrige isso.

As funções `var()` e `sd()` no R usam a fórmulas das variância e desvio padrão amostrais. Apliquemos isso para a variável de expectativa de vida no banco `gapminder`.

```
var(gapminder$lifeExp)

## [1] 166.9

sd(gapminder$lifeExp) # ou sqrt(var(gapminder$lifeExp))

## [1] 12.92
```

Como o desvio padrão está na unidade de medida da nossa variável, faz mais sentido falarmos de desvio padrão em conjunto com nossa média, para vermos a dispersão da nossa variável em torno da média. Podemos incluir as duas informações em uma função só usando o `summarise()`.

```
gapminder %>%
    summarise(
        mean_lifeExp = mean(lifeExp),
        sd_lifeExp   = sd(lifeExp)
    )

## # A tibble: 1 x 2
##   mean_lifeExp sd_lifeExp
##          <dbl>      <dbl>
## 1         59.5       12.9
```

Assimetria

A *assimetria* da distribuição de uma v.a. Y descreve quanto essa distribuição desvia da simetria:

$$\text{Assimetria} = \frac{\mathbb{E}\left[\left(Y-\mu_Y\right)^3\right]}{\sigma_Y^3} \tag{7}$$

Em uma distribuição simétrica, um valor de Y um tanto acima de sua média possui a mesma probabilidade de ocorrência que um valor de Y o mesmo tanto abaixo de sua média – assim, a assimetria de uma distribuição simétrica é 0. Quando a distribuição é assimétrica, esse equilíbrio entre valores um tanto acima e um tanto abaixo da média não acontece – com isso, a assimetria é diferente de zero. Quando a distribuição possui uma cauda longa à esquerda, a assimetria é negativa; quando a distribuição possui uma cauda longa à direita, a assimetria é positiva.

A assimetria amostral, por sua vez, é:

$$\widehat{\text{Assimetria}} = \frac{1}{(n-1)\cdot\hat{\sigma}^3}\sum_{i=1}^{n}(x_i-\bar{X})^3 \tag{8}$$

Voltemos ao banco `gapminder`, mas examinemos também uma variável que sabemos, visualmente, ser assimétrica: PIB per capita.

```
#install.packages("e1071")
library(e1071) # pacote necessário para funções de assimetria e curtose

gapminder %>%
    summarise(
        skewness_lifeExp    = skewness(lifeExp),
        skewness_gdpPercap  = skewness(gdpPercap)
    )
```

```
## # A tibble: 1 x 2
##   skewness_lifeExp skewness_gdpPercap
##              <dbl>              <dbl>
## 1           -0.252               3.84
```

Vemos como a variável "lifeExp" possui uma leve assimetria negativa, mas a variável "gdpPercap" possui uma alta assimetria positiva – isto é, há poucos países-anos muito ricos na nossa amostra.

Curtose

A *curtose* da distribuição é uma medida de quanta "massa" (ou área) há em suas caudas – os intervalos menos prováveis da distribuição; isto é, é uma medida de quanto da variância de Y é resultante de valores extremos – os quais são chamados de *valores atípicos* (*outliers*).

$$\text{Curtose} = \frac{\mathbb{E}\left[\left(Y - \mu_Y\right)^4\right]}{\sigma_Y^4} \tag{9}$$

Quanto maior a curtose, mais massa há nas caudas da distribuição, com isso, mais prováveis são os valores atípicos. A distribuição Normal, uma distribuição simétrica em torno de sua média, tem uma curtose igual a 3. Uma distribuição com curtose maior que 3 é uma distribuição com alta curtose, e uma distribuição menor que é uma distribuição com baixa curtose.

A curtose amostral é:

$$\widehat{\text{Curtose}} = \frac{1}{(n-1)\cdot\hat{\sigma}^4}\sum_{i=1}^{n}(x_i - \bar{X})^4 \tag{10}$$

Como exemplo, voltemos às variáveis de expectativa de vida e de PIB per capita.

```
gapminder %>%
    summarise(
        kurt_lifeExp    = kurtosis(lifeExp),
        kurt_gdpPercap  = kurtosis(gdpPercap)
    )

## # A tibble: 1 x 2
##   kurt_lifeExp kurt_gdpPercap
##          <dbl>          <dbl>
## 1        -1.13           27.4
```

Vemos que a variável "lifeExp" possui baixa curtose, mas a variável "gdpPercap" possui altíssima curtose – os histogramas que vimos anteriormente dão sentido a essas informações.

Duas variáveis aleatórias

Distribuição de probabilidade conjunta

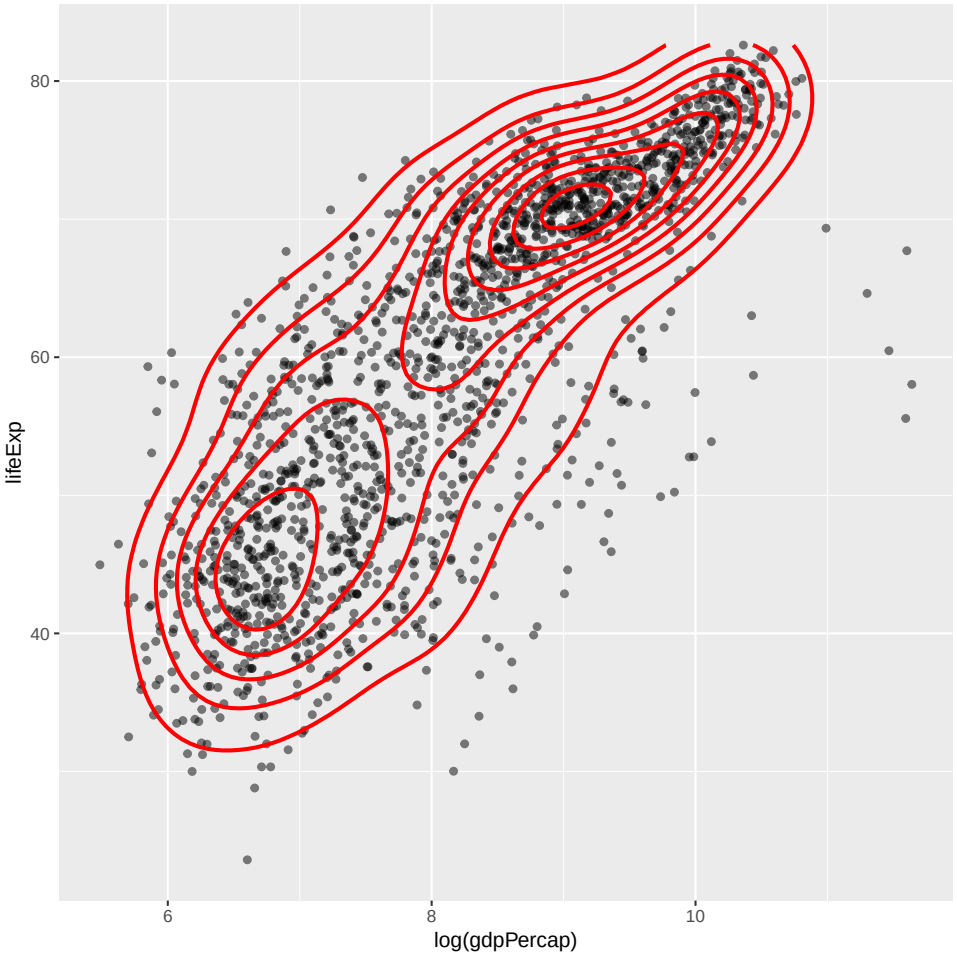
Agora, envolvendo duas variáveis aleatórias discretas (X e Y), sua *distribuição de probabilidade conjunta* é a probabilidade de que essas v.a.s assumem, simultaneamente, certos valores como x e y –

"simultaneamente" não porque aquelas variáveis $X = x$ e $X = y$ aconteceram ao mesmo tempo, mas porque, na coleta de dados, essas variáveis se mostraram no mesmo espaço e tempo definidos. A distribuição conjunta pode ser escrita como:

$$\Pr\big(X = x, Y = y\big) \tag{11}$$

Empiricamente, podemos visualizar a distribuição conjunta de duas variáveis aleatórias. Peguemos duas variáveis contínuas, para simplificar: expectativa de vida e PIB per capita – transformamos em logaritmo natural a variável de PIB per capita para facilitar a visualização do gráfico. Um gráfico de dispersão é apropriado para essa tarefa, mas também podemos usar um gráfico de densidade para duas variáveis.

```
gapminder %>%
    ggplot(aes(
        x = log(gdpPercap), # transformação em log natural
        y = lifeExp
        )
    ) +
    geom_point( # gráfico de dispersão (pontos)
        alpha = 0.5 # transparência dos pontos
    ) + # podemos adicionar outro tipo de gráfico...
    geom_density_2d( # gráfico de densidade em duas dimensões
        colour = "red", # cor dos contornos
        linewidth = 1 # contornos mais espessos
    ) # não esqueça: podemos usar separadamente um ou outro tipo de gráfico
```



Pensemos esse gráfico como uma montanha vista de cima: os contornos indicam onde os valores estão concentrados; quanto mais contornos e mais próximos os contornos entre si, maior concentração, como o pico de uma montanha. Primeiro, vemos uma relação positiva entre PIB per capita (em log) e expectativa de vida. Segundo, há uma grande concentração de observações nos maiores valores dessa distribuição conjunta.

Distribuição de probabilidade marginal

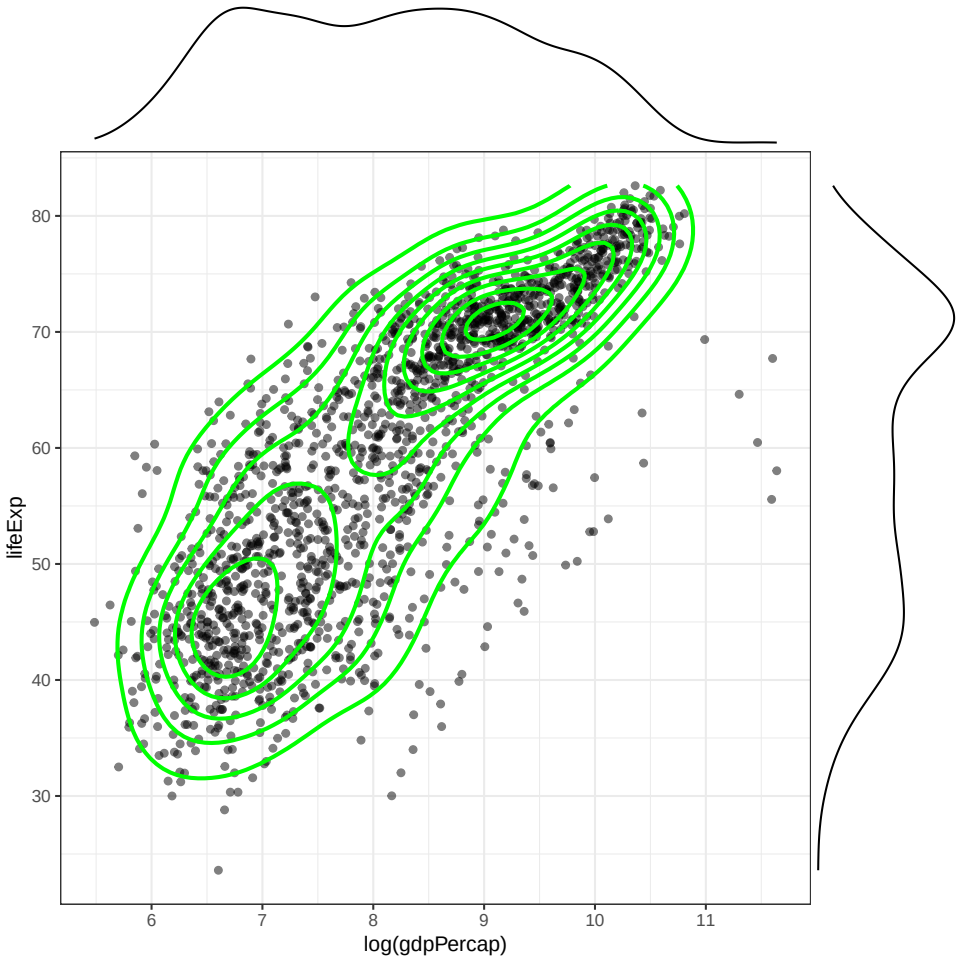
A *distribuição de probabilidade marginal* de uma v.a. Y é o mesmo que a distribuição de probabilidade; utilizamos "marginal" para distinguir a distribuição de Y em relação a sua distribuição conjunta com outra variável aleatória. Como computar a distribuição marginal de Y? Assumindo duas v.a.s discretas X e Y, a distribuição marginal de Y pode ser calculada a partir da distribuição conjunta de X e Y: adicionamos as probabilidades de todos os possíveis resultados nos quais Y assume um valor específico y, mas X pode assumir l valores diferentes. Formalmente:

$$\Pr\big(Y = y\big) = \sum_{i=1}^{l} \Pr\big(X = x_i, Y = y\big) \tag{12}$$

Podemos visualizar as distribuições marginais empiricamente no R de duas formas: i) Usar histogramas, boxplots, ou gráficos de densidade separadamente para cada variável, tal como fizemos anteriormente. ii) Combinadas com o gráfico de distribuição conjunta. Escolhamos a segunda opção.

```r
#install.packages("ggExtra")
library(ggExtra) # para visualizarmos as distribuições marginais
grafico_gap <- # guardar o gráfico em um objeto
    gapminder %>%
    ggplot(aes(
        x = log(gdpPercap), # transformação em log natural
        y = lifeExp # eixo y
        )
    ) +
    geom_point(alpha = 0.5) + # dispersão
    geom_density_2d( # densidade em duas dimensões
        colour = "green", # cor dos contornos
        linewidth = 1 # contornos mais espessos
    ) + # mais camadas... preste atenção no código e no resultado!
    theme_bw() +
    scale_x_continuous( # alterar configurações do eixo x
        breaks = scales::breaks_extended(n = 6) # mais números no eixo
    ) +
    scale_y_continuous( # alterar configurações do eixo y
        breaks = scales::breaks_extended(n = 6) # vem do pacote scales
    )

ggMarginal(grafico_gap, type = "density") # com as distribuições marginais
```



Distribuição de probabilidade condicional

A distribuição de probabilidade condicional é a (distribuição de) probabilidade de ocorrência de uma v.a. $Y = y$, sabendo que outra v.a. X assumiu um valor específico x – isto é, a probabilidade de Y dado que X ocorreu. Formalmente:

$$\Pr(Y = y \mid X = x) = \frac{\Pr\big(X = x, Y = y\big)}{\Pr\left(X = x\right)} \tag{13}$$

A probabilidade condicional é usada para descobrir a probabilidade de um evento não conhecido, a partir de um evento conhecido. Quando a ocorrência de Y está associada à ocorrência de X – isto é, quando essas variáveis são dependentes entre si –, significa que nossa informação sobre X nos permite saber algo sobre Y.

Em contrapartida, quando duas v.a.s X e Y são *independentes*, ou *independentemente distribuídas*, o nosso conhecimento sobre o valor de uma variável não nos dá qualquer informação sobre a ocorrência da outra variável. Especificamente, X e Y são independentes se a distribuição condicional de Y dado X for igual à distribuição marginal de Y. Formalmente:

$$\Pr\big(Y = y \mid X = x\big) = \Pr\big(Y = y\big) \quad \text{(X e Y são independentes)} \tag{14}$$

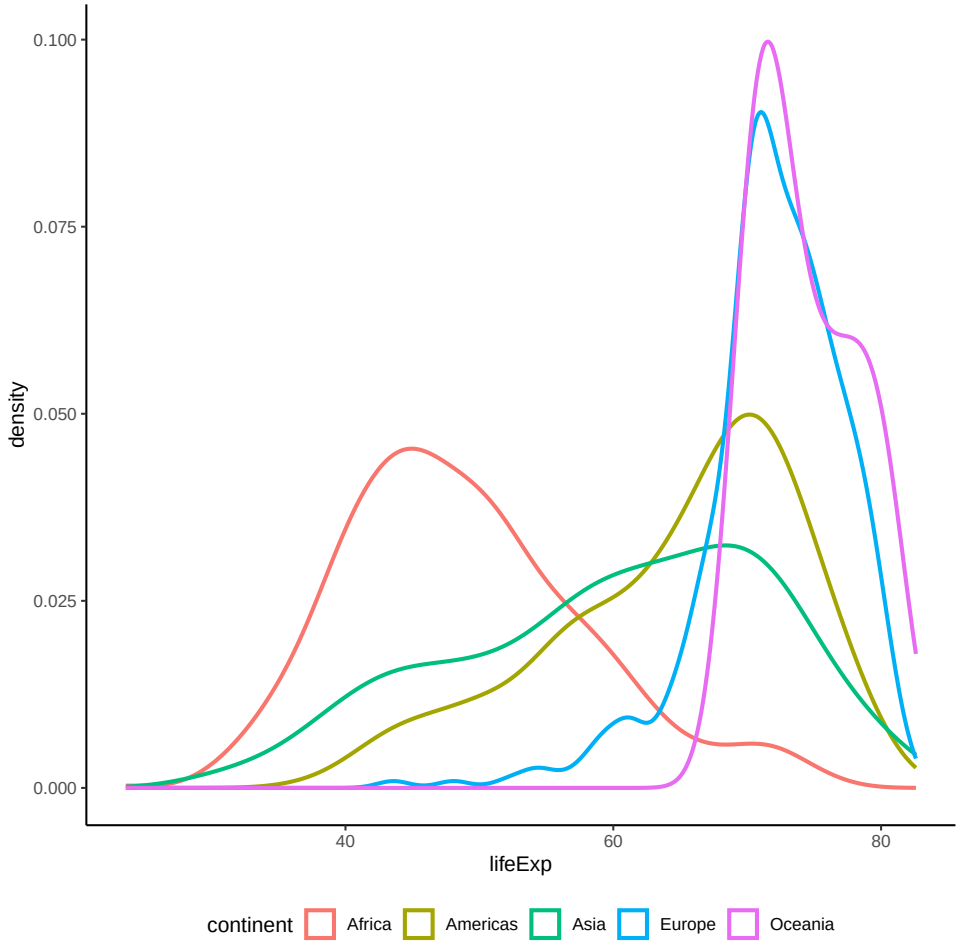
Se substituirmos a equação (14) pela (13), obtemos:

$$\begin{aligned} \frac{\Pr\big(X=x, Y=y\big)}{\Pr\left(X=x\right)} &= \Pr(Y=y) \\ \Pr\big(X=x, Y=y\big) &= \Pr(Y=y)\cdot\Pr\left(X=x\right) \end{aligned} \tag{15}$$

O que vemos aqui é que a distribuição conjunta de duas v.a.s independentes é o produto de suas distribuições marginais.

Como um exemplo simples de distribuições condicionais no R, peguemos duas variáveis no banco `gapminder`: "lifeExp" e `continent`. Qual é a distribuição da expectativa de vida dado o continente?

```
gapminder %>%
    ggplot(aes(
        x = lifeExp,
        colour = continent # uma cor por continente
        )
    ) +
    geom_density(linewidth = 1) +
    theme_classic() +
    theme(legend.position = "bottom") # posição da legenda
```



Esperança condicional

A *esperança condicional* de Y dado X, também chamada de *média condicional* de Y dado X, é o valor esperado (esperança ou média) da distribuição condicional de Y dado X. Se Y assume k valores $y_1, ..., y_k$, então, a média condicional de Y dado X é:

$$\mathbb{E}(Y \mid X = x) = \sum_{i=1}^{k} y_i \cdot \Pr\left(Y = y_i \mid X = x\right) \tag{16}$$

Ou seja, a esperança condicional de Y dado $X = x$ é simplesmente a média de Y quando $X = x$.

A esperança condicional será muito importante no futuro, pois é um conceito-chave na análise de regressão. Por ora, simplesmente calculemos a média condicional na nossa amostra de países-anos: sabemos a média incondicional da expectativa de vida. Mas, e a média da expectativa de vida dado o ano da amostra? No R, podemos combinar `group_by()` e `summarise()` para obtermos a média de "lifeExp", condicional por ano.

```
gapminder %>%
    group_by(year) %>% # agrupar por ano
    summarise(mean_lifeExp = mean(lifeExp)) # média condicional por ano
```

```
## # A tibble: 12 x 2
##     year mean_lifeExp
##    <int>        <dbl>
##  1  1952         49.1
##  2  1957         51.5
##  3  1962         53.6
##  4  1967         55.7
##  5  1972         57.6
##  6  1977         59.6
##  7  1982         61.5
##  8  1987         63.2
##  9  1992         64.2
## 10  1997         65.0
## 11  2002         65.7
## 12  2007         67.0
```

Exercícios

1 Importando a base de dados

Para esta lista de exercícios, iremos utilizar dados demográficos disponibilizados pelo IBGE, através do pacote `ribge`, que permite importar bases de dados diretamente para o R. Note que não se trata de um pacote oficial do CRAN (Comprehensive R Archive Network).

Para acessar o banco de dados a ser utilizado na lista, use os seguintes códigos dentro de um chunk (espaço para códigos no RMarkdown; pode ser criado através do atalho `Ctrl + Alt + i`:

```
#install.packages("devtools") # tire o comentário se precisar executar
library(devtools)
#install_github("tbrugz/ribge")
library(ribge)
```

Verifique na lista de pacotes se `ribge` foi instalado e ativado. A seguir, use o seguinte código para obter a base de dados de interesse:

```
pop2020 <- populacao_municipios(2020)
```

Perceba que usamos a função `populacao_municipios` do pacote `ribge` para conseguir a base de dados relativa ao ano de 2020, e atribuímos ao nome-objeto `pop2020`. Verifique no quadrante superior direito, na aba *environment*, se a base de dados `pop2020` foi criada. É possível clicar no objeto para abrir uma aba apenas com a base de dados, para melhor visualização.

A partir da tabela importada, responda às seguintes perguntas:

1. Qual é a unidade de análise desse banco de dados? Quantas observações estão listadas?
2. Quantas variáveis estão contidas nesse banco de dados?
3. Comente sobre outros aspectos da base de dados que considerar interessante, curioso ou que tenha gerado algum tipo de dúvida.

É pertinente limpar a base de dados para que ela contenha apenas as variáveis (colunas) que serão analisadas. Assim, sua visualização fica menos poluída, mais intuitiva, e poderemos focar apenas nos dados de interesse.

4. Utilize a função `select()` do pacote `dplyr` para estas três variáveis: “uf”, “nome_munic”, “populacao”.
5. Renomeie a variável “nome_munic” para “municipio”. Utilize a função `rename()`.
6. Por fim, transforme todos os caracteres dos nomes de município na coluna “municipio” em minúsculo. A função `mutate()` permite criar novas variáveis, e a função `tolower()` transforma todos as letras em minúsculas. Combine as duas para resolver o exercício.

2 Analisando o estado de São Paulo

Para este exercício e os posteriores, vamos analisar apenas dados relativos ao estado de São Paulo. Selecione apenas as observações referentes ao estado de São Paulo. Isso é equivalente a remover da base as observações relativas a outros estados brasileiros.

Para isso, é possível utilizar a função `filter()`, também do pacote `dplyr`, utilizando como argumento dentro do parênteses o tipo de observação que se deseja filtrar. No caso, os municípios cuja variável "uf" é "SP".

Observe novamente seu banco de dados, e responda às seguintes questões:

1. Quantos municípios há no estado de São Paulo?
2. Qual o menor município do estado para o ano de 2020? Quantos habitantes ele tinha?
3. Quantos municípios paulistas tem uma população maior que um milhão de habitantes? Liste-os, mencionando também a população de cada um.

3 Estatística descritiva dos dados

O R também permite calcular as estatísticas de interesse do nosso banco de dados. Vamos utilizar essas funções, boa parte delas da própria base do R, para conhecer as características do nosso dataframe. Para gerar um número único que resume uma variável em nossa tabela, usamos o verbo/função `summarise()` (ou `summarize()`, para o inglês americano), que cria uma nova tabela pequena que contém as estatísticas resumidas.

Para a variável "populacao", calcule:

1. Média – use a função `mean()`.
2. Mediana – use a função `median()`.
3. Desvio-padrão – use a função `sd()`.
4. Variância – use a função `var()`.

4 Trabalhando com gráficos

As estatísticas geradas no exercício anterior ajudam a sintetizar o nosso banco de dados. Vamos agora trabalhar com gráficos para melhor observar a distribuição dos nossos dados populacionais. Como mencionado em aula, o pacote `ggplot2` atualmente é o mais utilizado para essa finalidade. Não se esqueça de instalá-lo e ativá-lo.

1. Construa um gráfico de densidade (`geom_density()`) da variável população dos municípios paulistas. Extra: tente mudar a cor (argumento `colour = `), e preenchimento da curva (argumento `fill = `), e a transparência da curva (argumento `alpha = `) de seu gráfico.
2. Comente o gráfico gerado. O que você observa?
3. A partir da observação do gráfico, qual parece ser a medida mais adequada de tendência central da população: a média ou a mediana? Justifique sua resposta.

5 Gráfico de densidade de pequenos municípios

Vamos trabalhar agora com os municípios paulistas de população menor de 50.000 habitantes.

1. Crie um gráfico de densidade apenas para os municípios com menos de 50.000 habitantes. Utilize novamente a função `filter()`, e perceba que você não precisa criar um novo banco de dados, pois pode conectar todos os comandos através do "pipe" %>%.

2. Quantos são os municípios paulistas com menos de 50.000 habitantes? Qual a porcentagem dessas cidades em relação ao conjunto de municípios do estado? Tente fazer a conta pelo R.

3. Em comparação com o gráfico da questão anterior, o que você observa?

6 Trabalhando com escala logarítmica

Transformar dados em escala logarítmica permite melhor visualização de dados com valores muito altos e discrepantes, que tornam a distribuição dos dados assimétrica.

1. Crie novamente um gráfico de densidade para todos os municípios de São Paulo, mas agora utilize a variável de população em logaritmo. Um caminho possível é criar uma nova variável com a função `mutate()` e usar a função `log()` para transformar a variável em logaritmo natural.

2. Comente o gráfico de densidade, comparando com o gráfico do exercício 4.1.

7 Médias da população por estado

Para esta questão final, utilize novamente a base de dados original, com dados de todos os estados (para 2020).

1. Assim como foi feito para os municípios do estado de São Paulo, calcule a média da população para cada um dos estados brasileiros. Utilize as funções `group_by()` (pacote `dplyr`) para agrupar os estados (variável "uf"), além da função `summarise()` usada no exercício 3. Atribua esses comandos a um novo objeto.

2. Perceba que a tabela gerada não segue uma ordem crescente ou decrescente. Acrescente a função `arrange()` (pacote `dplyr`). Quais dos estados possuem maior e menor população média por município?