Ambientação ao R

Fernando de Pol Mayer (LEG/DEST/UFPR)
2022-02-01 (última atualização 2022-01-31)

Programa

Ambientação ao R

- Instalação e configuração do ambiente R
- Instalação e tour pela RStudio IDE
- Outras IDEs para trabalhar com R
- Instruções e comentários e uso REPL
- Atribuição, espaço de trabalho e diretório de trabalho
- Arquivos da linguagem R (.R, .Rprofile, .Rhistory, .RData, etc)
- Instalação e inspeção de pacotes
- Acesso à documentação

O que é e instalação

O que é o R

Um dialeto do S

- *Ambiente* estatístico para análise de dados e produção de gráficos
- Uma completa linguagem de programação:
  - Interpretada (contrário de compilada)
  - Orientada a objetos: **Tudo no R é um objeto...**
- Livre distribuição (código-aberto)
- Versão atual: 4.1.2 "Bird Hippie" (2021-11-01)
  - Ciclo de lançamentos: 6 meses (versões menores), 1 ano (versões maiores)

Histórico

- 1980: Linguagem S: desenvolvida por R. Becker, J. Chambers e A. Wilks (AT&T Bell Laboratories)
- 1980: Versão comercial: S-Plus (Insightful Corporation)
- 1996: Versão livre: R desenvolvido por R. Ihaka e R. Gentleman (Universidade de Auckland)
- 1997: R Development Core Team
- Hoje: 20 desenvolvedores principais e muitos outros colaboradores em todo o mundo

> *... to turn ideas into software, quickly and faithfully.*
>
> — John M. Chambers

Características

O R fornece amplo ferramental de estatística além de:

- Inúmeros recursos gráficos.
- Extensível: coleção de mais de 18 mil **pacotes** oficiais no CRAN.
- Interoperabilidade & Interconectividade: **APIs/drivers** para outros softwares/linguagens.
- Multiplataforma: Linux, Mac OS, Windows, Android.
- Escalabilidade e código compilado: C e Fortran.
- Boa parte do código fonte do *R base* está em C e Fortran.

Instalação e uso

Instalação

- **Linux**: http://cran-r.c3sl.ufpr.br/bin/linux/
  - Cada distribuição tem sua particularidade
  - Muito possivelmente já está no sistema de pacotes (apt, pacman, etc)
  - Não tem nenhuma interface próprio, abrir no terminal
- **Mac OS X**: http://cran-r.c3sl.ufpr.br/bin/macosx/
  - `.pkg` para macOS 10.13 (High Sierra) ou posterior
  - Possui uma interface básica própria
- **Windows**: http://cran-r.c3sl.ufpr.br/bin/windows/
  - Instalar o `.exe` da distribuição "base"
  - Possui uma interface básica própria

Serviços na nuvem

- **RStudio Cloud** https://rstudio.cloud/
  - Necessário criar conta
  - Versão gratuita possui várias limitações
- **Google Colab** https://colab.to/r
  - Jupyter Notebook (igual ao Python)

Editores para o R

Interagindo com o computador

O que significa este ícone?

- É um documento do Microsoft Excel?
- **Não**, é um arquivo de **texto pleno**, separado por vírgulas (CSV *comma separated values*)
- De fato, o nome do arquivo é `final.csv` e não `final`
- O Excel pode sim abrir este arquivo... assim como milhares de outros programas!

O que está acontecendo?

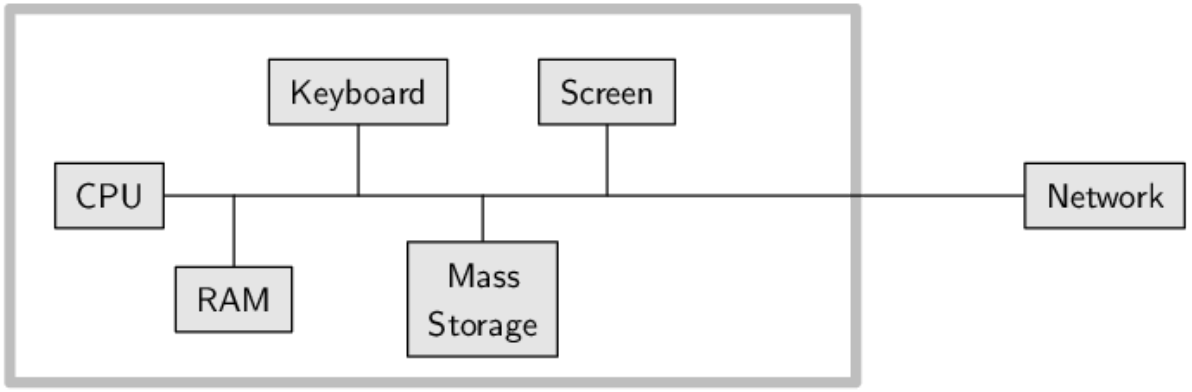
- O computador (leia-se, nesse caso, o sistema operacional Windows) "proteje" o usuário dos detalhes sujos
- Isso é ruim? **Sim!**
- O usuário se acostuma com o computador ditando as regras
- É importante lembrar que é você quem deve dizer o que o computador deve fazer

O que deve acontecer?

- Para a maioria dos usuários, a interação com o computador se limita a clicar em links, selecionar menus e caixas de diálogo
- O problema com essa abordagem é que parece que o usuário é controlado pelo computador
- A verdade deve ser o oposto!
- É o usuário que possui o controle e deve dizer para o computador exatamente o que fazer
- Escrever código ainda tem a vantagem de deixar registrado tudo o que foi feito

Editores de texto

Uma linguagem de programação nos permite interagir não só com outros *softwares*, mas também com o nosso *hardware*



Uma característica importante de códigos de programação é que eles são em **texto puro**, por isso precisamos de um bom **editor de textos**

Características de um bom editor:

- **Identação automática**
- **Complementação de parênteses**
- **Destaque de sintaxe** (*syntax highlighting*)
- **Numeração de linhas**
- **Auto completar comandos**

Editores para o R

- Interface padrão no macOS e Windows
- Tinn-R (Windows)
- Vim-R-plugin (Linux)
- Gedit-R-plugin (Linux)
- Rstudio: mais utilizado atualmente
- Emacs + ESS: altamente recomendado

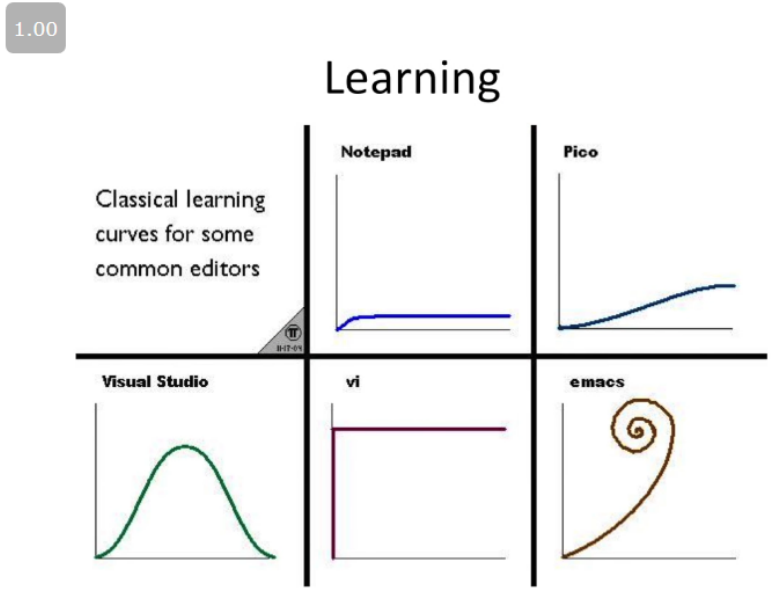
RStudio

- IDE (*Integrated Development Environment*)
- Desenvolvido especificamente para o R
- **Necessário instalar o R antes**
- Possui diversas "facilidades"
- Desenvolvido em JavaScript
  - Multi-plataforma
  - Pode ser "pesado" na memória

RStudio $\neq$ R

Emacs

- Criado em 1976 para ser um editor de texto de uso geral do projeto GNU
- Para integrar o R é necessário instalar o ESS (*Emacs Speaks Statistics*)
- Não possui "apelo visual" (em termos de interface)
- Desenvolvido em Lisp
  - Multi-plataforma
  - É programável (altamente customizável)

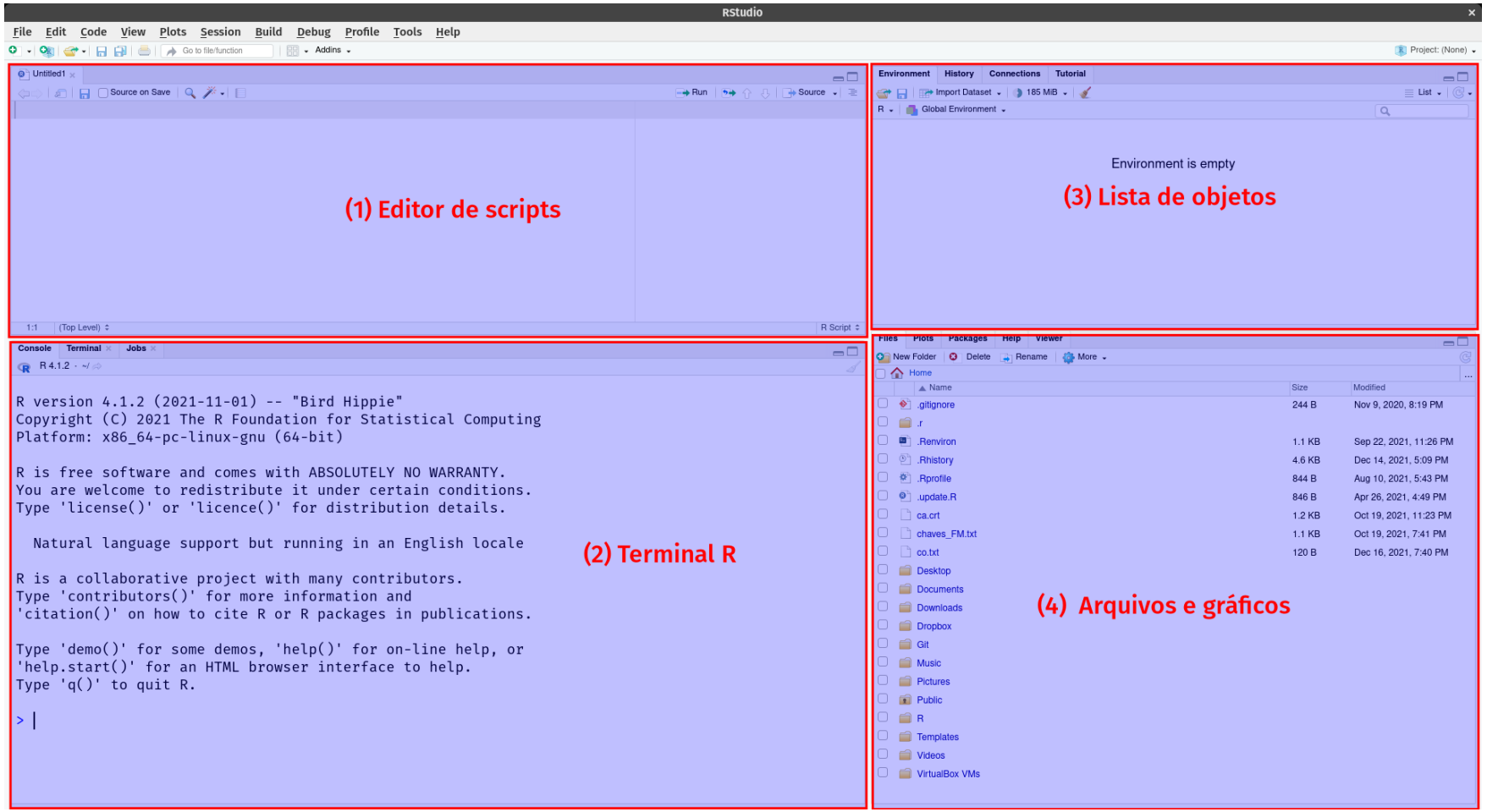


RStudio IDE

Detalhes

- A RStudio Desktop IDE é a IDE mais popular para a linguagem R.
- Mantida pela **RStudio**: https://www.rstudio.com/.
- Tem vesões para servidor com acesso web: RStudio Server.
- Instalar a versão RStudio Desktop

RStudio IDE


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
(1) Editor de scripts
Environment is empty
(3) Lista de objetos
Console Terminal Jobs
R version 4.1.2 (2021-11-01) -- "Bird Hippie"
Copyright (C) 2021 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
Natural language support but running in an English locale
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
(2) Terminal R
(4) Arquivos e gráficos

RStudio IDE

- RStudio Desktop IDE é muito fácil de aprender.
- Já vem toda preparada para trabalhar com todas tarefas envolvendo R:
  - Análise de dados com scripts.
  - Conexão com bancos de dados.
  - Elaboração de documentos dinâmicos.
  - Desenvolvimento de aplicações web.
  - Atuação em projetos com versionamento.
  - Desenvolvimento de pacotes R.
  - Integração R e Python.

Dicas

- Domine os **atalhos de teclado** para maior produtividade.
- Trabalhe sempre com **sessões novas**.
- Para projetos grandes, use o **recurso de projetos**.

Usando o R

Usando pela primeira vez

Configurando um diretório de trabalho

1. Abra sua interface

2. Configure um diretório de trabalho

- O diretório de trabalho é uma pasta onde o R será direcionado
- Todos os arquivos que serão importados ou exportados ficarão neste diretório

```
setwd("/home/mayer/cnj")
getwd()
[1] "/home/mayer/cnj"
```

- No RStudio, você pode usar o atalho: `Session > Set Working Directory > Choose Directory`

3. Abra um script e salve com a extensão `.R` ou `.r`

- No RStudio: `File > New File > R Script`

O R como uma calculadora

O símbolo `>` indica que o R está pronto para receber um comando:

```
> 2 * 2
```

```
[1] 4
```

O símbolo `>` muda para `+` se o comando estiver incompleto:

```
> 2 *
+ 2
```

```
[1] 4
```

Espaços entre os números não fazem diferença:

```
> 2 *          2
```

```
[1] 4
```

Instruções na linguagem R

Modos de uso

- Modo **CLI** (*Command Line Interface*)
  - Comandos são digitados direto no terminal
  - Mais utilizados para testes rápidos
  - Fica mais difícil acompanhar o que foi feito
- Modo **REPL** (*Read, Eval, Print, Loop*)
  - Script com instruções (receita).
  - Instruções avaliadas no console (interpretador).
  - Analista superviosiona o processo.
  - Salva/duplica/modifica o script conforme necessidade.
  - Para enviar comandos diretamente para o console, selecione-os e aperte `Ctrl + <Enter>`.
- Modo **Batch** (*lote*)
  - Script poder ser executado sem supervisão.
  - Usado em ambientes de produção ou simulação computacional.

Comentários e instruções

- Tudo que vem após `#` é um **comentário**
- Você deve utilizar para documentar o que o código faz

```
# Faz uma soma.
2 + 2
```

```
[1] 4
```

```
2 + 2  # esta linha será executada
```

```
[1] 4
```

```
## 2 + 2     esta linha não será executada

# Quantos segundos tem um dia?
24 * 60 * 60
```

```
[1] 86400
```

Instruções e comentários

Faça comentários relevantes

- Evite comentários óbvios.
- Evite comentários ambiguos.
- É melhor errar pelo excesso.

Instruções

- Podem ocupar uma única linha.
- Ocupar várias linhas.
- Uma linha pode ter várias instruções.

Recomendações

- Evite ultrapassar 72 ou 80 caracteres.
- Mantenha o código devidamente indentado (Ctrl + i).
- Evite muitas instruções em uma linha.

```
# Uma única linha.
2 + 2 + 7 + 5
```

```
[1] 16
```

```
# Em várias linhas.
2 +
    2 +
    7 +
    5
```

```
[1] 16
```

```
# Várias em uma linha (separados por ';')
2 + 2; 7 + 5
```

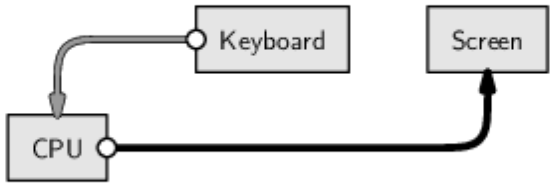
```
[1] 4
```

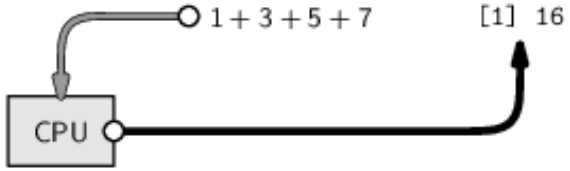
```
[1] 12
```

Para onde vão os resultados?

```
> 1 + 3 + 5 + 7
```

```
[1] 16
```





Note que o resultado é apenas mostrado na tela, nada é salvo na memória (por enquanto)

"Salvando" resultados ou criando objetos

```
## Dados de altura e peso de uma pessoa
altura <- 180
peso <- 83
## Para "ver"
altura
```

```
[1] 180
```

```
peso
```

```
[1] 83
```

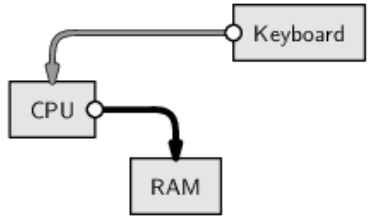
```
## Calcula o IMC
peso/altura^2
```

```
[1] 0.002561728
```

```
## Note que o comando acima não "salva" o resultado.
## Para isso deveríamos criar um novo objeto
imc <- peso/altura^2
imc
```

```
[1] 0.002561728
```

Quando criamos uma variável/objeto, ela fica armazenada **temporariamente** na memória RAM.



Para saber quais objetos estão criados, usamos a **função** `ls()`

```
ls()
```

```
[1] "altura" "imc"    "peso"
```

Para remover um ou mais objetos, usamos a **função** `rm()`

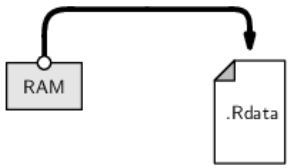
```
rm(altura, peso)
```

O símbolo `<-` é chamado de **operador de atribuição** (use `Alt` + `_` no RStudio)

"Salvando" resultados ou criando objetos

- Estes objetos ficam armazenados no chamado *workspace* do R
- O *workspace* consiste de tudo que for criado durante uma sessão do R, armazenado na memória RAM
- Para efetivamente salvar esses objetos, podemos armazenar esse *workspace* do R **em disco**, em um arquivo chamdo `.Rdata`

```
save.image()
```



- Quando o R é iniciado em um diretório com um arquivo `.Rdata`, as variáveis salvas são automaticamente carregadas
- No entanto, é sempre melhor salvar os dados e o **script**, assim é possível gerar os resultados novamente, sem salvar nada sem necessidade
- Veremos mais pra frente como **salvar objetos específicos**, por exemplo, resultados de uma análise que leva muito tempo para ser executada
- O mais importante é salvar o **código**, assim sabemos **como** chegamos a determinado resultado, e podemos recriá-lo depois

- Objetos podem ser reusados para criar outros (essa é a intenção).
- Objetos podem ser sobrescritos.
- Objetos podem ser apagados.

Finalizando o programa

- A qualquer momento durante uma sessão você pode usar o comando

```
> save.image()
```

- Alternativamente, você pode também salvar toda sua área de trabalho, clicando em `Workspace > Save As Default Workspace`. Este processo irá gerar dois arquivos:
  - `.Rdata`: contém todos os objetos criados durante uma sessão. Não é necessário (e nem recomendado) dar um nome antes do ponto. Dessa forma, a próxima vez que o programa for iniciado neste diretório, a área de trabalho será carregada automaticamente.
  - `.Rhistory`: um arquivo texto que contém todos os comandos que foram digitados no console.

Lembre-se: o mais importante é sempre salvar o **script** (`.R`)

Pacotes

Instalação e gerenciamento

Instalação de pacotes

Pacotes

- Pacotes são coleções de funções e conjuntos de dados organizados e documentados.
- O pacote contém código R e eventualmente códigos de outras linguagens.
- Pacotes podem depender de *libs* do sistema operacional (programas externos).

Formas de instalação

- Pacotes podem ser instalados de repositórios: CRAN, Bioconductor, MRAN, etc.
- De aquivos de instalação `*.tar.gz`.
- De repositórios Git: GitHub, GitLab, etc.
- Para mais, visite r-packages-guide.

```r
## Para instalar um pacote do repositório oficial
install.package("mvtnorm")

## Para carregar o pacote e usá-lo.
library(mvtnorm)

## Para ver o conteúdo dele.
ls("package:mvtnorm")

## Documentação do pacote.
help(package = "mvtnorm")

## Para ver onde foi instalado.
system.file(package = "mvtnorm")

# Os caminhos para endereços de instalação.
.libPaths()

# Para remover o pacote da sessão.
detach("package:mvtnorm", unload = TRUE)

# Funções relacionadas a pacotes.
apropos("package")
```

Acesso à documentação do R

Como aprender R sem sair do R

Documentação interna

A documentação do R

- Consiste de documentação de objetos e funções.
- Tutoriais chamados de vinhetas (*vignettes*).
- Existem funções específicas para a consulta destes.
- Pode-se procurar na web também.

```r
## Duas formas iguais de chamar a documentação.
?sum
help(sum)

## Procura por ocorrêncians de `tukey`.
help.search("sum")

## Objetos que batem com um termo.
apropos("tukey")

## Exibe as vinhetas de um pacote.
browseVignettes(package = "survival")

## Procura pelo termo no r-project.org.
RSiteSearch("spider plot")
```

Campos da documentação

Cabeçalho
Indica o pacote.

Título
Título da função.

Description
Descrição do que o objeto é/faz.

Usage
Como usar ou fazer a chamada.

Arguments
Quais os argumentos formais da função.

Value
O que a função retorna.

Details
Detalhes adicionais de implementação.

Note
Notas adicionais sobre uso e afins.

See Also
Referências para documentação relacionada.

References
Referências bibliográficas.

Authors
Autores da função.

Examples
Exemplos de uso.

Considerações Finais

Considerações Finais

- R é uma linguagem e ambiente para **computação estatística e gráficos**, livre e de código aberto.
- É extensível, escalável, interopera com várias outras linguagens/softwares.
- É cada vez mais popular na comunidade acadêmica.
- Tem sido adotado exponencialmente na comunidade profissional para atividades de análise de dados.
- O R pode ser empregado em várias etapas do processo de análise de dados:
  - Aquisição, importação e tratamentos dos dados.
  - Confecção de visualizações estáticas e dinâmicas (gráficos, mapas, etc).
  - Elaboração de relatórios dinâmicos/reproduzível.
  - Disponibilização de conteúdo via APIs e dashboards.
  - Ajuste de modelos estatísticos (diagnósticos, preditivos e prescritivos).
  - Uso de métodos de aprendizado de máquinas.