Módulo | Análise de Dados: Controle de Versão I

Caderno de **Aula**
Professor André Perez

Tópicos

1. Introdução ao Git;
2. Introdução ao GitHub;
3. Criando um repositório no GitHub.

Aulas

1. Introdução ao Git

No mercado é muito comum trabalhar em equipes (mistas ou não) de analistas de dados, cientistas de dados, engenheiros de dados, especialistas de negócio, etc. Neste contexto, ao se utilizar programação como ferramenta de trabalho, é preciso garantir que:

- os **códigos** estejam disponíveis em algum **repositório** externo;
- que haja um **histórico de alterações** dos **códigos**;
- etc.

O git é um software de controle de versão distribuído de **arquivos**. Ele vem pré instalado em muitos sistemas operacionais mas você pode encontrar instruções de instalação para Linux/MacOS/Windows neste link. Nas máquinas virtuais do Google Colab o `git` já vem instalado.

In [ ]:
```
!git --version
```

O `git` utiliza o conceito de repositório (diretório **local** com uma pasta `.git`) para organizar projetos. Dentro do repositório, modificações em arquivos podem ser:

- visualizadas (`git status`);
- adicionadas (`git add`);
- salvas (`git commit`);
- etc.

Exemplo:

In [ ]:
```
!mkdir ./projeto
!touch ./projeto/hello.py
```

In [ ]:
```
%cd /content/projeto/
```

In [ ]:
```
!pwd
```

In [ ]:
```
!python3 hello.py
```

In [ ]:
```
# criar um repositório
!git init
```

In [ ]:
```
# visualizar o estado do repositório
!git status
```

In [ ]:
```
# adicionar modificações
!git add hello.py
!git status
```

In [ ]:
```
# visualizar o estado do repositório
!git status
```

In [ ]:
```
# adicionar modificações
!git add hello.py
!git status
```

In [ ]:
```
# salvar as alterações
!git commit -m "arquivo hello.py criado"
!git status
```

In [ ]:
```
%cd /content/
```

O `git` possui seu próprio jargão e seu uso é feito através da linha de comando (`bash` no Linux/MacOS e `cmd` no Windows) e por isso ele é conhecido por ser difícil de se aprender.

In [ ]:
```
!git --help
```

2. Introdução ao GitHub

O `git` utiliza o conceito de repositório (diretório **local** com uma pasta `.git`) para organizar projetos. Existem diversas empresas e organizações que oferecem versões **remotas** (online) de repositórios:

- GitHub;
- GitLab;
- Bitbucket.

Para fazer o download de um projeto, basta utilizar o comando `git clone`. Como examplo, vamos fazer o download do projeto *Awesome Public Datasets* (link):

In [ ]:
```
!git clone https://github.com/awesomedata/awesome-public-datasets
```

3. Criando um repositório no GitHub

> Conta

- Criar uma conta no GitHub.
- Criar um *personal access token*.
- Configurar a conexão entre o `git` local com o `git` remoto (GitHub):

In [ ]:
```
import os

username = "andre-marcos-perez"
os.environ["GITHUB_USER"] = username

!git config --global user.name "${GITHUB_USER}"
```

In [ ]:
```
import os
from getpass import getpass

usermail = getpass()
os.environ["GITHUB_MAIL"] = usermail

!git config --global user.email "${GITHUB_MAIL}"
```

In [ ]:
```
import os
from getpass import getpass

usertoken = getpass()
os.environ["GITHUB_TOKEN"] = usertoken
```

> Projeto

- Criar o projeto **remoto**.
- Fazer o download do projeto **remoto** na máquina **local**.

In [ ]:
```
!git clone https://${GITHUB_USER}:${GITHUB_TOKEN}@github.com/andre-marcos-per
```

In [ ]:
```
%cd /content/da-ebac/
```

- Criar um arquivo e salvar as aterações no repositório **local**.

In [ ]:
```
!touch hello.py
!git status
```

In [ ]:
```
!git add hello.py
!git status
```

In [ ]:
```
!git commit -m "arquivo hello.py criado"
!git status
```

- Enviar as alterações para o repositório **remoto**.

In [ ]:
```
!git push origin main
!git status
```