Anaconda

Boas-vindas à aula sobre o uso do **Anaconda** para gerenciar seus pacotes e ambientes ao trabalhar com Python. Com o Anaconda, é simples instalar os pacotes que você usará com frequência ao analisar dados. Você também o usará para criar ambientes virtuais que tornam a tarefa de trabalhar em diversos projetos simultaneamente muito mais simples. O Anaconda simplificou meu fluxo de trabalho e resolveu diversas questões que apareciam quando eu precisava lidar com diversos pacotes e versões do Python ao mesmo tempo.

O Anaconda é, na verdade, a distribuição de software que contém o conda, o Python e mais de 150 pacotes científicos e suas dependências. A aplicação conda é o gerenciador de pacotes e ambientes. O Anaconda é um download relativamente pesado (~500 MB), porque vem com os pacotes mais comuns de *data science* do Python. Se você não precisa de todos os pacotes ou quer economizar no volume de dados baixados/salvos, existe também o **Miniconda**, uma distribuição menor, que inclui apenas o conda e o Python. Ainda é possível instalar qualquer um dos pacotes disponíveis usando o conda, mas eles não vêm nessa distribuição.

OConda é um programa que você usará apenas na linha de comando, então, caso não esteja confortável com isso, dê uma olhada neste **tutorial para linha de comando do Windows** ou em nosso curso para OSX/Linux de **Linha de comando Linux nível básico**.

Você provavelmente já tem o Python instalado e deve estar se perguntando o porquê de fazer tudo isso. Em primeiro lugar, como o Anaconda vem com um monte de pacotes de *data science*, você já estará preparado para trabalhar com dados. Segundo, usar o conda para gerenciar seus pacotes e ambientes reduzirá o número de problemas futuros decorrentes das diversas bibliotecas que você usará.

Gerenciando pacotes

Instalando o NumPy com o conda

Gerenciadores de pacotes são utilizados para instalar bibliotecas e outros programas em seu computador. Você provavelmente já conhece o pip, que é o gerenciador de pacotes padrão das bibliotecas Python. O conda é parecido com o pip, exceto pelo fato de que os pacotes disponíveis são mais focados em *data science*, enquanto o pip é mais generalista. No entanto, o conda *não* é específico para o Python, tal como o pip; ele também pode instalar pacotes que não são do Python. Trata-se de um gerenciador de pacotes para *qualquer* software. Dito isso, nem todas as bibliotecas Python estão disponíveis para a distribuição Anaconda e para o conda. Você também pode (e de fato, irá) utilizar o pip junto ao conda para instalar pacotes.

O conda instala pacotes pré-compilados. Por exemplo, a distribuição Anaconda contém as bibliotecas Numpy, Scipy e Scikit-learn compiladas com a **biblioteca MKL**, o que acelera diversas operações matemáticas. Os pacotes são mantidos por colaboradores da distribuição, o que significa que nem sempre as últimas versões estão disponíveis automaticamente. Mas, graças ao fato de que alguém precisou montar os pacotes para diversos sistemas, eles tendem a ser mais estáveis (e, por isso, mais convenientes).

Ambientes

Criando um ambiente com o conda

Além de administrar pacotes, o Conda também gerencia ambientes virtuais, similar ao **virtualenv** e ao **pyenv**, outros gerenciadores de ambientes famosos.

Ambientes permitem que você separe e isole pacotes que estão sendo utilizados para projetos diferentes. Códigos que dependem de versões diferentes de uma mesma biblioteca serão utilizados com frequência. Por exemplo, é possível ter código que use aspectos novos do Numpy ou então códigos que usem aspectos antigos que foram removidos das versões novas. É praticamente impossível ter duas versões do Numpy instaladas ao mesmo tempo. Em vez disso, cada versão do Numpy deve estar presente em um ambiente e, então, o projeto que precisa de uma versão será feito em determinado ambiente.

Essa questão também surge quando se lida com Python 2 e Python 3. É possível trabalhar com código antigo que não funciona em Python 3 ou código novo que não roda em Python 2. Ter ambas as versões instaladas evita muita confusão e bugs. É muito melhor ter ambientes separados.

Você também pode exportar a lista de pacotes de um ambiente para um arquivo e, então, incluir esse arquivo no código. Isso permite que outras pessoas consigam instalar com facilidade todas as dependências para seu código rodar. O pip tem uma funcionalidade parecida com o comando `pip freeze > requirements.txt`.

Para onde seguiremos a partir daqui

A seguir, detalharei o uso do Anaconda. Primeiro cobriremos a instalação, depois, o uso do gerenciador de pacotes para, por fim, gerenciar e criar ambientes.

PRÓXIMO