8.Git

October 27, 2019

1 Git

Este notebook apresenta os seguintes tópicos:

Outra fonte de informações de um repositório de software é o repositório do sistema de controle de versões.

Pelo controle de versões, conseguimos ter acesso a todos os arquivos de todas as versões, todas as mensagens de commit, branches, e colaboradores.

Nesta parte do minicurso, faremos a mineração dessas informações.

No caso do Git, ao clonar um repositório, ficamos com uma cópia local do que está lá. Portanto, começamos a mineração com um clone e não precisamos de nenhum proxy.

[1]:
```
import pdffallback
output = !git clone https://github.com/gems-uff/sapos
pdffallback.show(output)
```

```
Cloning into 'sapos'…
```

Com o repositório clonado, podemos usar comandos do git para extrair informações.

[2]:
```
%cd sapos
```

```
/home/joao/projects/tutorial/pdf/source/sapos
```

[3]:
```
!git branch -r
```

```
  origin/HEAD -> origin/master
  origin/bugfixes
  origin/hotfixes
  origin/master
  origin/reports
```

Essas informações também podem ser obtidas para tratarmos usando variáveis do Python.

A seguir tentamos descobrir qual é o commit de cada um desses branches.

[4]:
```
git_branch_output = !git branch -r
branches = [
    branch.strip().split(' ')[0].split('/')[1]
    for branch in git_branch_output
]
branches
```

[4]:
```
['HEAD', 'bugfixes', 'hotfixes', 'master', 'reports']
```

[5]:
```
branch_commit = {}
for branch in branches:
    __ = !git checkout $branch
    commit = !git show --pretty=format:"%h" --no-patch
    branch_commit[branch] = commit
__ = !git checkout master
branch_commit
```

[5]:
```
{'HEAD': ['d3e4307'],
 'bugfixes': ['4d22176'],
 'hotfixes': ['d3e4307'],
 'master': ['d3e4307'],
 'reports': ['b294935']}
```

Usamos `__ = !...` para evitar a exibição do output do comando de sistema. O IPython imprime o output quando bang expressions são usadas isoladas e retorna o output quando elas são usadas em atribuições.

Note que apenas o branch `reports` e `bugfixes` está em um commit diferente.

1.1 Exercicio 9

Faça a mesma operação para obter o código dos commits de tags e salve na variável `tag_commit`.

[6]:
```
tags = !git tag
...
```

Agora vamos agrupar as tags por versões minor e ordenar as versões patch.

[7]:
```
from itertools import groupby
groups = groupby(tags, lambda x: x.rsplit(".", 1)[0])
minor_tags = {}
for minor, elements in groups:
    minor_tags[minor] = sorted(
        elements,
        key=lambda x: int(x.split('-')[0].split('.')[-1])
    )
```

```
minor_tags['4.3']
```

[7]:
```
['4.3.0',
 '4.3.1',
 '4.3.2',
 '4.3.3',
 '4.3.4',
 '4.3.5',
 '4.3.6',
 '4.3.7',
 '4.3.8',
 '4.3.9',
 '4.3.10',
 '4.3.11',
 '4.3.12',
 '4.3.13',
 '4.3.14']
```

Fazendo o mesmo para agrupar versões major.

[8]:
```
groups = groupby(minor_tags, lambda x: x.rsplit(".", 1)[0])
major_tags = {}
for major, elements in groups:
    major_tags[major] = sorted(
        elements,
        key=lambda x: int(x.split('-')[0].split('.')[-1])
    )
major_tags['4']
```

[8]:
```
['4.0', '4.1', '4.2', '4.3', '4.4']
```

Com isso, podemos escolher versões major (e.g., 3 e 4) e obter a última versão patch para cada minor delas.

[9]:
```
last_patch_for_v3v4 = {
    minor: minor_tags[minor][-1]
    for minor in major_tags['3'] + major_tags['4']
}
last_patch_for_v3v4
```

[9]:
```
{'3.0': '3.0.0',
 '3.1': '3.1.0',
 '3.2': '3.2.1',
 '3.3': '3.3.7',
 '4.0': '4.0.4',
 '4.1': '4.1.1',
 '4.2': '4.2.0',
 '4.3': '4.3.14',
```

```
 '4.4': '4.4.27'}
```

Agora queremos ver a evolução de linhas de código para as versões selecionadas. Para isso, vamos percorrer o dicionário fazendo checkout de cada versão, carregar o número de linhas usando `cloc` e parsear o resultado para extrair as colunas para construir linhas de uma tabela.

[10]:
```
from collections import defaultdict
columns = {"id"}
rows = []

for minor, tag in last_patch_for_v3v4.items():
    __ = !git checkout $tag
    lines = !cloc .
    filtered_lines = lines[lines.index("-" * 79) + 3:]
    commit_result = defaultdict(int)
    commit_result["id"] = minor
    for line in filtered_lines:
        if not line.startswith("-"):
            split = line.split()
            language = split[0]
            commit_result[language + "_files"] = int(split[1])
            commit_result[language + "_blank"] = int(split[2])
            commit_result[language + "_comment"] = int(split[3])
            commit_result[language + "_code"] = int(split[4])
            columns |= {
                language + "_files", language + "_blank",
                language + "_comment", language + "_code"
            }
    rows.append(commit_result)
```

1.2 Pandas

Podemos usar `pandas` para construir a tabela a partir da lista de dicionários.

[11]:
```
import pandas as pd
df = pd.DataFrame(rows)
df
```

[11]:

	id	Ruby_files	Ruby_blank	Ruby_comment	Ruby_code	HTML_files	\
0	3.0	336	1880	1362	19147	108	
1	3.1	336	1937	1363	19359	108	
2	3.2	364	2437	1385	21739	108	
3	3.3	396	2749	1527	23319	109	
4	4.0	447	2965	1653	24719	109	
5	4.1	449	2977	1667	24759	109	
6	4.2	449	2978	1667	24767	109	

```
7  4.3         505        3221          2100      25564           5
8  4.4         508        3186          2042      25765           4

   HTML_blank  HTML_comment  HTML_code  Sass_files  …  XML_comment  \
0       10891            22      15740           3  …          0.0
1       10891            22      15740           3  …          0.0
2       10891            22      15740           8  …          0.0
3       10901            22      15887           8  …          0.0
4       10903            22      15799           8  …          0.0
5       10903            22      15799           8  …          0.0
6       10903            22      15799           8  …          0.0
7          28            22        227           9  …          NaN
8          16            22        168           9  …          NaN

   XML_code  SUM:_files  SUM:_blank  SUM:_comment  SUM:_code  \
0       9.0         508       13520          1525      38849
1       9.0         508       13582          1526      39079
2       9.0         546       14249          1643      42458
3       9.0         583       14604          1790      44486
4       9.0         645       14903          1966      46478
5       9.0         647       14917          1980      46532
6       9.0         647       14918          1980      46540
7       NaN         594        4237          2418      31506
8       NaN         600        4192          2321      31740

   CoffeeScript_files  CoffeeScript_blank  CoffeeScript_comment  \
0                 NaN                 NaN                   NaN
1                 NaN                 NaN                   NaN
2                 NaN                 NaN                   NaN
3                 NaN                 NaN                   NaN
4                 NaN                 NaN                   NaN
5                 NaN                 NaN                   NaN
6                 NaN                 NaN                   NaN
7                 1.0                 0.0                   3.0
8                 1.0                 0.0                   3.0

   CoffeeScript_code
0                NaN
1                NaN
2                NaN
3                NaN
4                NaN
5                NaN
6                NaN
7                0.0
8                0.0
```

```
[9 rows x 45 columns]
```

O pandas permite descrever a tabela com o método `.describe()`.

[12]:
```
df.describe()
```

[12]:
```
       Ruby_files   Ruby_blank  Ruby_comment     Ruby_code  HTML_files  \
count    9.000000     9.000000      9.000000      9.000000    9.000000
mean   421.111111  2703.333333   1640.666667  23237.555556   85.444444
std     66.325795   507.187589    275.371113   2566.055587   45.894202
min    336.000000  1880.000000   1362.000000  19147.000000    4.000000
25%    364.000000  2437.000000   1385.000000  21739.000000  108.000000
50%    447.000000  2965.000000   1653.000000  24719.000000  108.000000
75%    449.000000  2978.000000   1667.000000  24767.000000  109.000000
max    508.000000  3221.000000   2100.000000  25765.000000  109.000000

         HTML_blank  HTML_comment     HTML_code  Sass_files  Sass_blank  …  \
count      9.000000           9.0      9.000000    9.000000    9.000000  …
mean    8480.777778          22.0  12322.111111    7.111111  319.666667  …
std     4795.680160           0.0   6874.179268    2.368778   23.690715  …
min       16.000000          22.0    168.000000    3.000000  277.000000  …
25%    10891.000000          22.0  15740.000000    8.000000  315.000000  …
50%    10891.000000          22.0  15740.000000    8.000000  333.000000  …
75%    10903.000000          22.0  15799.000000    8.000000  334.000000  …
max    10903.000000          22.0  15887.000000    9.000000  336.000000  …

       XML_comment  XML_code  SUM:_files    SUM:_blank  SUM:_comment  \
count          7.0       7.0    9.000000      9.000000      9.000000
mean           0.0       9.0  586.444444  12124.666667   1905.444444
std            0.0       0.0   55.854971   4516.526874    320.371703
min            0.0       9.0  508.000000   4192.000000   1525.000000
25%            0.0       9.0  546.000000  13520.000000   1643.000000
50%            0.0       9.0  594.000000  14249.000000   1966.000000
75%            0.0       9.0  645.000000  14903.000000   1980.000000
max            0.0       9.0  647.000000  14918.000000   2418.000000

          SUM:_code  CoffeeScript_files  CoffeeScript_blank  \
count      9.000000                 2.0                 2.0
mean   40852.000000                 1.0                 0.0
std     6016.483711                 0.0                 0.0
min    31506.000000                 1.0                 0.0
25%    38849.000000                 1.0                 0.0
50%    42458.000000                 1.0                 0.0
75%    46478.000000                 1.0                 0.0
max    46540.000000                 1.0                 0.0

       CoffeeScript_comment  CoffeeScript_code
```

```
count                  2.0               2.0
mean                   3.0               0.0
std                    0.0               0.0
min                    3.0               0.0
25%                    3.0               0.0
50%                    3.0               0.0
75%                    3.0               0.0
max                    3.0               0.0

[8 rows x 44 columns]
```

Além disso, é possível fazer seleções nos dados.

[13]:
```
df[df["Ruby_code"] > 25000]
```

[13]:
```
    id  Ruby_files  Ruby_blank  Ruby_comment  Ruby_code  HTML_files  \
7  4.3         505        3221          2100      25564           5
8  4.4         508        3186          2042      25765           4

   HTML_blank  HTML_comment  HTML_code  Sass_files  …  XML_comment  \
7          28            22        227           9  …          NaN
8          16            22        168           9  …          NaN

   XML_code  SUM:_files  SUM:_blank  SUM:_comment  SUM:_code  \
7       NaN         594        4237          2418      31506
8       NaN         600        4192          2321      31740

   CoffeeScript_files  CoffeeScript_blank  CoffeeScript_comment  \
7                 1.0                 0.0                   3.0
8                 1.0                 0.0                   3.0

   CoffeeScript_code
7                0.0
8                0.0

[2 rows x 45 columns]
```

1.3 Exercicio 10

Selecione as versões que usam CoffeeScript e as versões que não usam XML.

[14]:
```
with_coffee = ...
with_coffee
```

[14]:
```
Ellipsis
```

[15]:
```
without_xml = ...
without_xml
```

[15]:
```
Ellipsis
```

Além de selecionar linhas, podemos selecionar colunas.

[16]:
```
columns = ['SUM:_files', 'SUM:_blank', 'SUM:_comment', 'SUM:_code']
ndf = df[columns]
ndf
```

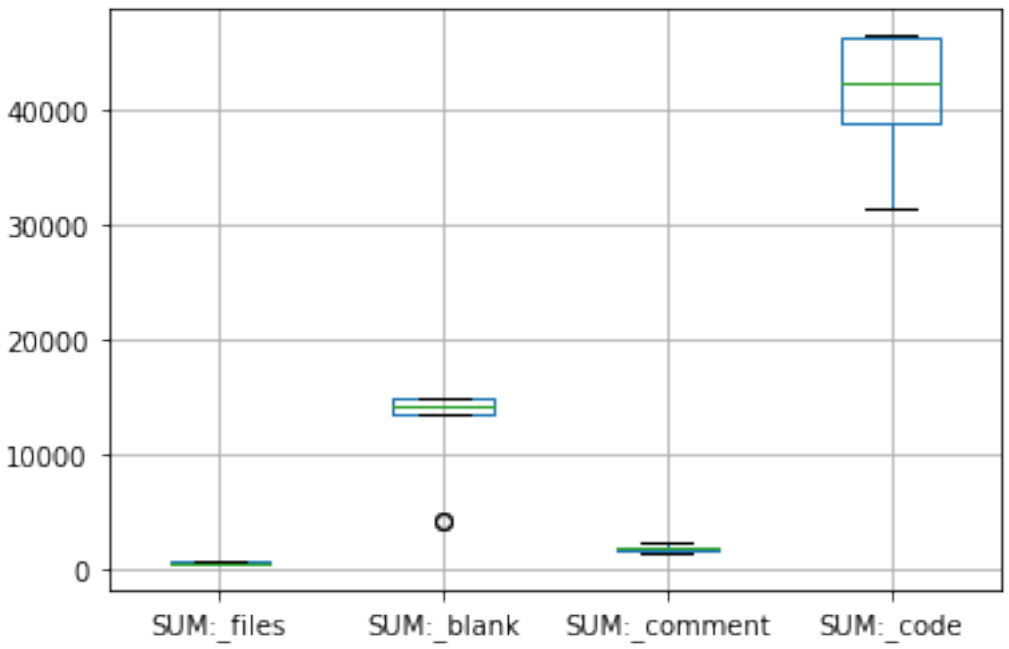
[16]:

	SUM:_files	SUM:_blank	SUM:_comment	SUM:_code
0	508	13520	1525	38849
1	508	13582	1526	39079
2	546	14249	1643	42458
3	583	14604	1790	44486
4	645	14903	1966	46478
5	647	14917	1980	46532
6	647	14918	1980	46540
7	594	4237	2418	31506
8	600	4192	2321	31740

O pandas também oferece algumas funções que facilitam a geração de gráficos.

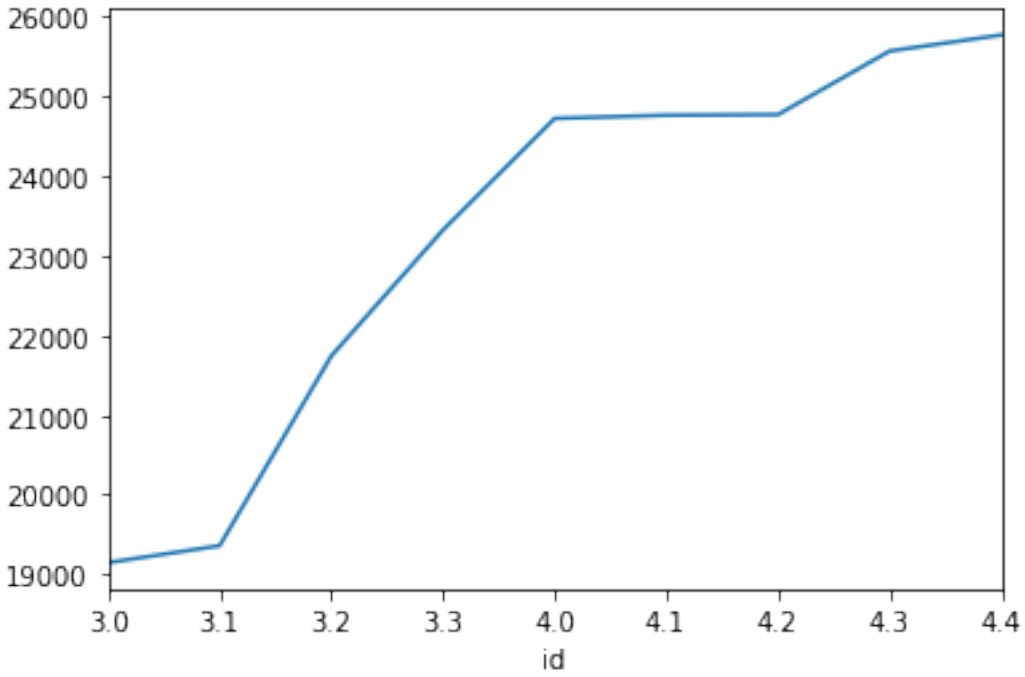
[17]:
```
%matplotlib inline
ndf.boxplot()
```

[17]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4c722ee5c0>
```



[18]:
```
df.set_index("id")["Ruby_code"].plot()
```

[18]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4c71a5abe0>
```



É possível aplicar operações em colunas e criar novas colunas.

[19]:
```
df.loc[:, "tag"] = df["id"].apply(lambda minor: last_patch_for_v3v4[minor])
```

[20]:
```
df
```

[20]:

	id	Ruby_files	Ruby_blank	Ruby_comment	Ruby_code	HTML_files	\
0	3.0	336	1880	1362	19147	108	
1	3.1	336	1937	1363	19359	108	
2	3.2	364	2437	1385	21739	108	
3	3.3	396	2749	1527	23319	109	
4	4.0	447	2965	1653	24719	109	
5	4.1	449	2977	1667	24759	109	
6	4.2	449	2978	1667	24767	109	
7	4.3	505	3221	2100	25564	5	
8	4.4	508	3186	2042	25765	4	

	HTML_blank	HTML_comment	HTML_code	Sass_files	…	XML_code	SUM:_files	\

```
0       10891            22      15740           3  …       9.0         508
1       10891            22      15740           3  …       9.0         508
2       10891            22      15740           8  …       9.0         546
3       10901            22      15887           8  …       9.0         583
4       10903            22      15799           8  …       9.0         645
5       10903            22      15799           8  …       9.0         647
6       10903            22      15799           8  …       9.0         647
7          28            22        227           9  …       NaN         594
8          16            22        168           9  …       NaN         600

   SUM:_blank  SUM:_comment  SUM:_code  CoffeeScript_files  \
0       13520          1525      38849                 NaN
1       13582          1526      39079                 NaN
2       14249          1643      42458                 NaN
3       14604          1790      44486                 NaN
4       14903          1966      46478                 NaN
5       14917          1980      46532                 NaN
6       14918          1980      46540                 NaN
7        4237          2418      31506                 1.0
8        4192          2321      31740                 1.0

   CoffeeScript_blank  CoffeeScript_comment  CoffeeScript_code     tag
0                 NaN                   NaN                NaN   3.0.0
1                 NaN                   NaN                NaN   3.1.0
2                 NaN                   NaN                NaN   3.2.1
3                 NaN                   NaN                NaN   3.3.7
4                 NaN                   NaN                NaN   4.0.4
5                 NaN                   NaN                NaN   4.1.1
6                 NaN                   NaN                NaN   4.2.0
7                 0.0                   3.0                0.0  4.3.14
8                 0.0                   3.0                0.0  4.4.27

[9 rows x 46 columns]
```

Existem muitas outras operações que podem ser vistas na documentação: https://pandas.pydata.org/pandas-docs/stable/.

Continua: 9.Pygit2.pdf