hash_distribution

April 14, 2021

1 2A.algo - Hash et distribution

Une fonction de hash a pour propriété statistiques de transformer une distribution quelconque en distribution uniforme. C'est pour cela que beaucoup d'algorithmes utilisent ce type de fonction avant tout traitement pour répartir les données de manières uniformes plutôt que d'utiliser une variable une colonne telle quelle.

[1]:
```
%matplotlib inline
import matplotlib.pyplot as plt
from jyquickhelper import add_notebook_menu
add_notebook_menu()
```

[1]:
```
<IPython.core.display.HTML object>
```

1.1 Récupérer un fichier wikipédia

[2]:
```
from mlstatpy.data.wikipedia import download_pageviews
import os
from datetime import datetime

download_pageviews(datetime(2018,2,1), folder=".")
```

[2]:
```
'.\\pageviews-20180201-000000'
```

On ne garde que les pages françaises.

[3]:
```
with open("pageviews-20180201-000000", "r", encoding="utf-8") as f:
    fr = filter(lambda line: line.startswith("fr "), f)
    with open("pageviews-20180201-000000.fr.txt", "w", encoding="utf-8") as g:
        for line in fr:
            g.write(line)
```

[4]:
```
import pandas
df = pandas.read_csv("pageviews-20180201-000000.fr.txt", encoding="utf-8", sep=" ",␣
↪header=None)
df.columns="country page impressions _".split()
df = df[["page", "impressions"]]
df = df.sort_values("impressions", ascending=False)
print(df.shape)
df.head()
```

```
(136440, 2)
```

[4]:
	page	impressions
131064	Wikipédia:Accueil_principal	10868
115817	Sp?cial:Search	8031
116771	Spécial:Recherche	2815
95020	Patrick_Dils	841
44847	France	668

Les données sont biaisées car les pages non démandées par les utilisateurs sur cett date ne sont pas répertoriées mais cela ne nuit pas à la démonstration faite ci-dessous.

1.2 Distribution volume, impressions par rapprt au premier caractère

[5]:
```
df["ch1"] = df["page"].apply(lambda r: r[0] if isinstance(r, str) else r)
df.head()
```

[5]:
	page	impressions	ch1
131064	Wikipédia:Accueil_principal	10868	W
115817	Sp?cial:Search	8031	S
116771	Spécial:Recherche	2815	S
95020	Patrick_Dils	841	P
44847	France	668	F

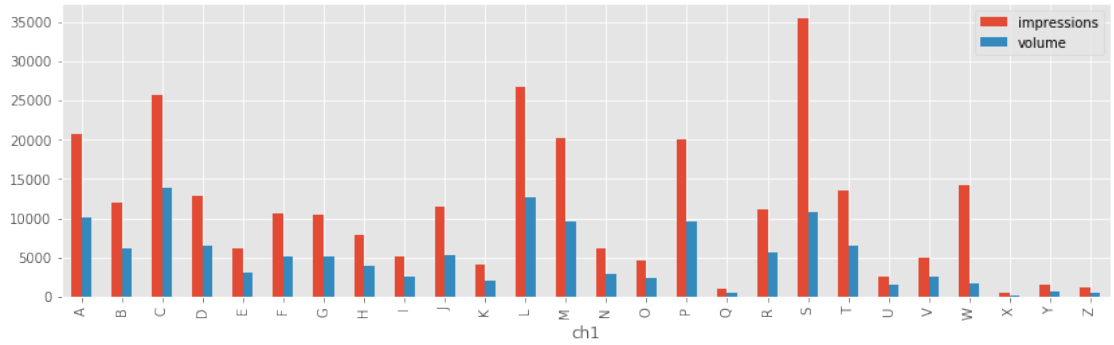
[6]:
```
co = df.copy()
co["volume"] = 1
gr = co.groupby("ch1", as_index=False).sum().sort_values("ch1")
gr.head()
```

[6]:
	ch1	impressions	volume
0	$	2	1
1	&	1	1
2	'	8	4
3	(	60	54
4	-	249	11

[7]:
```
gr[(gr["ch1"] >= "A") & (gr["ch1"] <= "Z")].plot(x="ch1", y=["impressions", "volume"],␣
↪kind="bar", figsize=(14,4))
```

```
c:\Python364_x64\lib\site-packages\pandas\plotting\_core.py:1716: UserWarning:
Pandas doesn't allow columns to be created via a new attribute name - see
https://pandas.pydata.org/pandas-docs/stable/indexing.html#attribute-access
  series.name = label
```

[7]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1ddc8d0eb70>
```



Il est possible de distribuer les impressions et les volumns sur plusieurs machines mais ce serait beaucoup plus simple si les volumes (volume de données) et les impressions (usage de ces données) suivent des distributions identiques.

1.3 Distribution après hashage

[8]:
```
import hashlib
def hash(text):
    md5 = hashlib.md5()
    md5.update(text.encode('utf-8'))
    return md5.hexdigest()

hash("France")
```

[8]:
```
'0309a6c666a7a803fdb9db95de71cf01'
```

[9]:
```
ha = co.copy()
ha["hash"] = ha["page"].apply(lambda r: hash(r) if isinstance(r, str) else r)
ha.head()
```

[9]:
```
                              page  impressions ch1  volume  \
131064  Wikipédia:Accueil_principal        10868   W       1
115817               Sp?cial:Search         8031   S       1
116771           Spécial:Recherche         2815   S       1
95020                 Patrick_Dils          841   P       1
44847                       France          668   F       1

                                  hash
131064  6cc68aa5234cb8c4e129d5b431eea95a
115817  e4c619292a0e11c8afba20eb4531cbf8
116771  6f86a30966129c5bf50147eb44c4e82c
95020   624f2274ebdbb30187e57d430c58990d
44847   0309a6c666a7a803fdb9db95de71cf01
```

[10]:
```
ha["ch2"] = ha["hash"].apply(lambda r: r[0] if isinstance(r, str) else r)
ha.head()
```

[10]:
```
                              page  impressions ch1  volume  \
131064  Wikipédia:Accueil_principal        10868   W       1
```

```
115817              Sp?cial:Search         8031   S       1
116771           Spécial:Recherche         2815   S       1
95020                 Patrick_Dils          841   P       1
44847                       France          668   F       1

                                  hash ch2
131064  6cc68aa5234cb8c4e129d5b431eea95a   6
115817  e4c619292a0e11c8afba20eb4531cbf8   e
116771  6f86a30966129c5bf50147eb44c4e82c   6
95020   624f2274ebdbb30187e57d430c58990d   6
44847   0309a6c666a7a803fdb9db95de71cf01   0
```

[11]:
```
gr = ha.groupby("ch2", as_index=False).sum().sort_values("ch2")
gr.head()
```

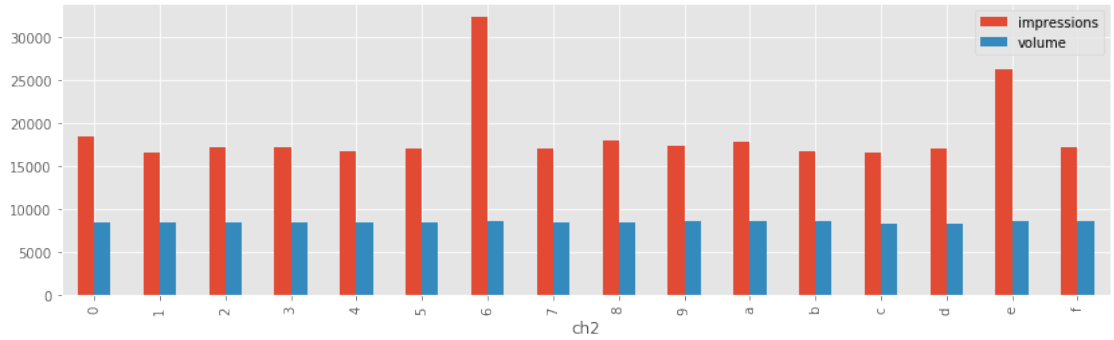
[11]:

	ch2	impressions	volume
0	0	18493	8531
1	1	16615	8513
2	2	17276	8514
3	3	17219	8547
4	4	16847	8468

[12]:
```
gr.plot(x="ch2", y=["impressions", "volume"], kind="bar", figsize=(14,4))
```

```
c:\Python364_x64\lib\site-packages\pandas\plotting\_core.py:1716: UserWarning:
Pandas doesn't allow columns to be created via a new attribute name - see
https://pandas.pydata.org/pandas-docs/stable/indexing.html#attribute-access
  series.name = label
```

[12]: `<matplotlib.axes._subplots.AxesSubplot at 0x1ddc748cc88>`



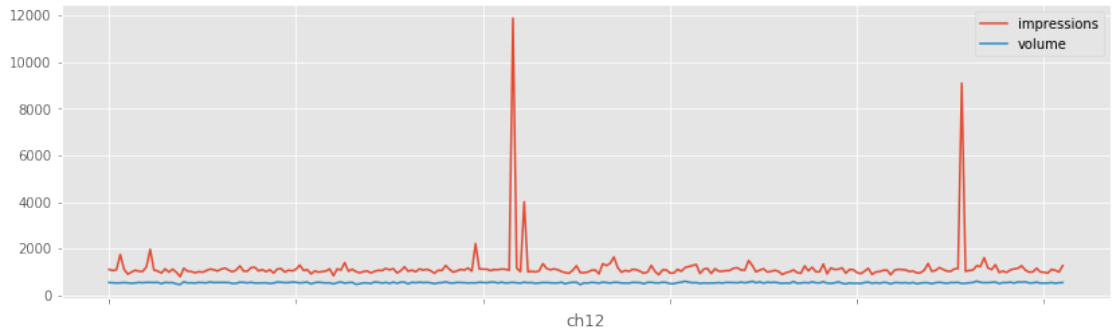
Après avoir appliqué une fonction de hashage, les deux distributions volumes et impressions sont presque uniforme par rapport au premier caractère du hash. Il reste un pic : il provient de la page wikipédia la plus demandée. Ces pages très demandées sont très souvent en très petit nombre et on implémentera une mécanisme de cache s'il s'agit d'un site web. En revanche, lors d'un calcul distribué sur Map / Reduce, un des problèmes consistera à traiter ces clés de distributions qui regroupent une part trop importante des données.
On recommence le même raisonnement en utilisant les deux premiers caractères du hash comme clé de distribution.

[13]:
```
def substr(s, size=2):
    if isinstance(s, str):
        if len(s) < size:
            return s
        else:
            return s[:size]
    else:
        return s

ha["ch12"] = ha["hash"].apply(lambda r: substr(r))
gr = ha.groupby("ch12", as_index=False).sum().sort_values("ch12")
gr.plot(x="ch12", y=["impressions", "volume"], figsize=(14,4))
```

```
c:\Python364_x64\lib\site-packages\pandas\plotting\_core.py:1716: UserWarning:
Pandas doesn't allow columns to be created via a new attribute name - see
https://pandas.pydata.org/pandas-docs/stable/indexing.html#attribute-access
  series.name = label
```

[13]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1ddc9e66ac8>
```



Dans ce cas précis, si le traitement appliqué aux données est d'un coût proportionnelle à la taille des données associées à chaque clé, cela signifie qu'une opération de type **reduce** aura un temps de traitement proportionnel au volume de données associée à la plus grande clé et non au nombre de machines puisque toutes les données d'une même clé sont envoyées sur la même machine.

[14]:
```
```