split_equal_sum

December 18, 2019

Developed by David Requena. 12/18/2019.

GitHub: @davidrequena

Contact: drequena@rockefeller.edu / d.requena.a@gmail.com

Please cite the repository if used.

Link: https://github.com/davidrequena/split_equal_sum

[1]:
```
from pandas import read_csv
import numpy as np
import random
from math import ceil
```

[2]:
```
#--------#
# INPUTS #
#--------#

# The input file is a CSV table with two columns:
# A first column with unique identifiers per sample, named 'uid'.
# A second column with the number of elements per sample, named 'n_elements'.

# In this particular example, we have a list of people, each one has an 'uid'.
# And 'n_elements' is the number of articles each person has published.
# We want to split this list into 'N' exclusive subgroups with equal or
# nearly similar total number of articles per group.

filename = 'uid_npapers.csv'

# We are going to distribute the dataset into 'N' even groups
N = 8

# Number of simulations
S = 100000

# Setting the random seed
R = 2019
```

[3]:
```
# Reading and visualizing the file
spb_db = read_csv(filename, header = 0)
print(spb_db)
```

```
          uid  n_elements
0    ruid_001           0
1    ruid_002           3
2    ruid_003           7
3    ruid_004           0
4    ruid_005           2
..        …          …
216  ruid_217           0
217  ruid_218           0
218  ruid_219           0
219  ruid_220          23
220  ruid_221           6

[221 rows x 2 columns]
```

[4]:
```
spb_sorted = spb_db.sort_values(by = ['n_elements'], ascending = False)
spb_np = np.asarray(spb_sorted)
print(spb_np.shape)
print(spb_np[:,0])
print(spb_np[:,1])
```

```
(221, 2)
['ruid_143' 'ruid_119' 'ruid_181' 'ruid_042' 'ruid_220' 'ruid_176'
 'ruid_106' 'ruid_174' 'ruid_169' 'ruid_165' 'ruid_210' 'ruid_065'
 'ruid_063' 'ruid_139' 'ruid_211' 'ruid_148' 'ruid_101' 'ruid_029'
 'ruid_172' 'ruid_214' 'ruid_086' 'ruid_151' 'ruid_203' 'ruid_179'
 'ruid_023' 'ruid_173' 'ruid_015' 'ruid_003' 'ruid_107' 'ruid_056'
 'ruid_091' 'ruid_221' 'ruid_182' 'ruid_202' 'ruid_072' 'ruid_216'
 'ruid_197' 'ruid_194' 'ruid_062' 'ruid_080' 'ruid_175' 'ruid_028'
 'ruid_039' 'ruid_135' 'ruid_212' 'ruid_158' 'ruid_011' 'ruid_200'
 'ruid_187' 'ruid_051' 'ruid_018' 'ruid_066' 'ruid_201' 'ruid_090'
 'ruid_112' 'ruid_002' 'ruid_070' 'ruid_060' 'ruid_032' 'ruid_057'
 'ruid_022' 'ruid_058' 'ruid_196' 'ruid_054' 'ruid_138' 'ruid_026'
 'ruid_145' 'ruid_020' 'ruid_021' 'ruid_131' 'ruid_190' 'ruid_193'
 'ruid_096' 'ruid_098' 'ruid_108' 'ruid_155' 'ruid_153' 'ruid_134'
 'ruid_142' 'ruid_113' 'ruid_087' 'ruid_111' 'ruid_030' 'ruid_053'
 'ruid_033' 'ruid_075' 'ruid_005' 'ruid_041' 'ruid_048' 'ruid_067'
 'ruid_036' 'ruid_031' 'ruid_180' 'ruid_132' 'ruid_034' 'ruid_083'
 'ruid_171' 'ruid_141' 'ruid_147' 'ruid_150' 'ruid_046' 'ruid_084'
 'ruid_164' 'ruid_156' 'ruid_170' 'ruid_188' 'ruid_043' 'ruid_205'
 'ruid_103' 'ruid_204' 'ruid_081' 'ruid_009' 'ruid_008' 'ruid_213'
 'ruid_037' 'ruid_209' 'ruid_038' 'ruid_007' 'ruid_040' 'ruid_168'
 'ruid_167' 'ruid_035' 'ruid_006' 'ruid_166' 'ruid_215' 'ruid_217'
 'ruid_218' 'ruid_219' 'ruid_163' 'ruid_162' 'ruid_004' 'ruid_161'
```

```
 'ruid_208' 'ruid_013' 'ruid_207' 'ruid_014' 'ruid_199' 'ruid_198'
 'ruid_016' 'ruid_017' 'ruid_012' 'ruid_195' 'ruid_019' 'ruid_024'
 'ruid_192' 'ruid_191' 'ruid_025' 'ruid_189' 'ruid_160' 'ruid_027'
 'ruid_186' 'ruid_185' 'ruid_184' 'ruid_183' 'ruid_010' 'ruid_178'
 'ruid_206' 'ruid_177' 'ruid_152' 'ruid_159' 'ruid_100' 'ruid_116'
 'ruid_115' 'ruid_114' 'ruid_068' 'ruid_069' 'ruid_110' 'ruid_109'
 'ruid_071' 'ruid_073' 'ruid_074' 'ruid_105' 'ruid_104' 'ruid_102'
 'ruid_099' 'ruid_157' 'ruid_076' 'ruid_097' 'ruid_077' 'ruid_095'
 'ruid_094' 'ruid_093' 'ruid_092' 'ruid_078' 'ruid_079' 'ruid_089'
 'ruid_088' 'ruid_082' 'ruid_085' 'ruid_117' 'ruid_118' 'ruid_064'
 'ruid_120' 'ruid_044' 'ruid_154' 'ruid_045' 'ruid_149' 'ruid_047'
 'ruid_146' 'ruid_144' 'ruid_049' 'ruid_050' 'ruid_140' 'ruid_052'
 'ruid_137' 'ruid_136' 'ruid_055' 'ruid_059' 'ruid_133' 'ruid_061'
 'ruid_130' 'ruid_129' 'ruid_128' 'ruid_127' 'ruid_126' 'ruid_125'
 'ruid_124' 'ruid_123' 'ruid_122' 'ruid_121' 'ruid_001']
[115 90 40 24 23 22 20 20 15 15 15 13 12 12 12 11 10 10 10 10 10 10 9 8 7
 7 7 7 6 6 6 6 6 6 5 5 5 5 5 5 5 5 5 4 4 4 4 4 4 4 4 4 4 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0]
```

[5]:
```
# Calculate the number of people with at least 1 article
one_or_more = sum([1 for i in spb_db['n_elements'] if i > 0])
print(one_or_more)
```

```
113
```

[6]:
```
# Then, we have to select the closest multiple of N above or equal to␣
 ↪one_or_more
t = N * ceil(one_or_more / N)

top_pool = spb_np[:t]
print(top_pool.shape)
print(top_pool[:,0])
print(top_pool[:,1])
```

```
(120, 2)
['ruid_143' 'ruid_119' 'ruid_181' 'ruid_042' 'ruid_220' 'ruid_176'
 'ruid_106' 'ruid_174' 'ruid_169' 'ruid_165' 'ruid_210' 'ruid_065'
 'ruid_063' 'ruid_139' 'ruid_211' 'ruid_148' 'ruid_101' 'ruid_029'
 'ruid_172' 'ruid_214' 'ruid_086' 'ruid_151' 'ruid_203' 'ruid_179'
 'ruid_023' 'ruid_173' 'ruid_015' 'ruid_003' 'ruid_107' 'ruid_056'
 'ruid_091' 'ruid_221' 'ruid_182' 'ruid_202' 'ruid_072' 'ruid_216'
 'ruid_197' 'ruid_194' 'ruid_062' 'ruid_080' 'ruid_175' 'ruid_028'
 'ruid_039' 'ruid_135' 'ruid_212' 'ruid_158' 'ruid_011' 'ruid_200'
 'ruid_187' 'ruid_051' 'ruid_018' 'ruid_066' 'ruid_201' 'ruid_090'
```

```
 'ruid_112' 'ruid_002' 'ruid_070' 'ruid_060' 'ruid_032' 'ruid_057'
 'ruid_022' 'ruid_058' 'ruid_196' 'ruid_054' 'ruid_138' 'ruid_026'
 'ruid_145' 'ruid_020' 'ruid_021' 'ruid_131' 'ruid_190' 'ruid_193'
 'ruid_096' 'ruid_098' 'ruid_108' 'ruid_155' 'ruid_153' 'ruid_134'
 'ruid_142' 'ruid_113' 'ruid_087' 'ruid_111' 'ruid_030' 'ruid_053'
 'ruid_033' 'ruid_075' 'ruid_005' 'ruid_041' 'ruid_048' 'ruid_067'
 'ruid_036' 'ruid_031' 'ruid_180' 'ruid_132' 'ruid_034' 'ruid_083'
 'ruid_171' 'ruid_141' 'ruid_147' 'ruid_150' 'ruid_046' 'ruid_084'
 'ruid_164' 'ruid_156' 'ruid_170' 'ruid_188' 'ruid_043' 'ruid_205'
 'ruid_103' 'ruid_204' 'ruid_081' 'ruid_009' 'ruid_008' 'ruid_213'
 'ruid_037' 'ruid_209' 'ruid_038' 'ruid_007' 'ruid_040' 'ruid_168']
[115 90 40 24 23 22 20 20 15 15 15 13 12 12 12 11 10 10 10 10 10 10 9 8 7
 7 7 7 6 6 6 6 6 6 5 5 5 5 5 5 5 5 5 4 4 4 4 4 4 4 4 4 4 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0]
```

[7]:
```python
# Split a list into buckets using a random seed
def random_split_buckets(df, nb, seed):

    # Get a copy of the list of IDs to not alter the original
    L = np.copy(df[:,0])

    random.Random(seed).shuffle(L)

    # Split the list into 'nb' buckets of size 'sb'
    sb = int(len(L) / nb)

    buckets = [L[sb*i : sb*(i+1)] for i in range(nb)]

    # Calculate the sum in each bucket
    sums_buckets = [sum(df[np.isin(df[:,0], b), 1]) for b in buckets]

    return buckets, sums_buckets


# Find the most even split
def best_distribution(df, nb, n_iter):

    std_iterations = []

    for s in range(n_iter):

        # Split into buckets
        buckets, sums_buckets = random_split_buckets(df, nb, s)

        # Calculate and store the Standard Deviation
        std_iterations.append(np.std(sums_buckets))
```

```
    # Select the best iteration
    bit = np.argmin(std_iterations)

    return random_split_buckets(df, nb, bit)
```

[8]:
```
# Get buckets of people with articles, evenly distributed
buckets_art, sums_art = best_distribution(top_pool, N, S)

print(sums_art)

for i in range(len(buckets_art)):
    print('Bucket with articles ' + str(i) + ':')
    print(buckets_art[i])
```

```
[99, 99, 140, 68, 128, 87, 100, 78]
Bucket with articles 0:
['ruid_220' 'ruid_150' 'ruid_193' 'ruid_032' 'ruid_084' 'ruid_212'
 'ruid_051' 'ruid_054' 'ruid_108' 'ruid_046' 'ruid_080' 'ruid_063'
 'ruid_176' 'ruid_148' 'ruid_072']
Bucket with articles 1:
['ruid_057' 'ruid_203' 'ruid_053' 'ruid_213' 'ruid_058' 'ruid_091'
 'ruid_023' 'ruid_139' 'ruid_021' 'ruid_011' 'ruid_187' 'ruid_096'
 'ruid_156' 'ruid_070' 'ruid_181']
Bucket with articles 2:
['ruid_145' 'ruid_066' 'ruid_143' 'ruid_103' 'ruid_060' 'ruid_209'
 'ruid_043' 'ruid_037' 'ruid_007' 'ruid_075' 'ruid_147' 'ruid_048'
 'ruid_180' 'ruid_194' 'ruid_087']
Bucket with articles 3:
['ruid_171' 'ruid_015' 'ruid_101' 'ruid_210' 'ruid_033' 'ruid_040'
 'ruid_211' 'ruid_216' 'ruid_182' 'ruid_090' 'ruid_081' 'ruid_132'
 'ruid_005' 'ruid_034' 'ruid_067']
Bucket with articles 4:
['ruid_170' 'ruid_214' 'ruid_158' 'ruid_020' 'ruid_008' 'ruid_113'
 'ruid_135' 'ruid_031' 'ruid_200' 'ruid_168' 'ruid_026' 'ruid_041'
 'ruid_098' 'ruid_119' 'ruid_083']
Bucket with articles 5:
['ruid_169' 'ruid_151' 'ruid_018' 'ruid_173' 'ruid_009' 'ruid_065'
 'ruid_165' 'ruid_153' 'ruid_056' 'ruid_164' 'ruid_155' 'ruid_197'
 'ruid_131' 'ruid_142' 'ruid_134']
Bucket with articles 6:
['ruid_205' 'ruid_138' 'ruid_062' 'ruid_022' 'ruid_029' 'ruid_086'
 'ruid_042' 'ruid_002' 'ruid_039' 'ruid_036' 'ruid_030' 'ruid_221'
 'ruid_174' 'ruid_038' 'ruid_107']
Bucket with articles 7:
['ruid_106' 'ruid_204' 'ruid_112' 'ruid_202' 'ruid_190' 'ruid_188'
 'ruid_003' 'ruid_196' 'ruid_179' 'ruid_111' 'ruid_141' 'ruid_172'
```

```
 'ruid_175' 'ruid_201' 'ruid_028']
```

[9]:
```
# Calculate the maximum multiple of N contained in the rows,
# to evenly distribute them into the N buckets
T = N * (len(spb_np) // N)
print(T)

# Now select people with no articles until row N:
bottom_pool = spb_np[t:T]
print(bottom_pool.shape)
print(bottom_pool[:,0])
print(bottom_pool[:,1])
```

```
216
(96, 2)
['ruid_167' 'ruid_035' 'ruid_006' 'ruid_166' 'ruid_215' 'ruid_217'
 'ruid_218' 'ruid_219' 'ruid_163' 'ruid_162' 'ruid_004' 'ruid_161'
 'ruid_208' 'ruid_013' 'ruid_207' 'ruid_014' 'ruid_199' 'ruid_198'
 'ruid_016' 'ruid_017' 'ruid_012' 'ruid_195' 'ruid_019' 'ruid_024'
 'ruid_192' 'ruid_191' 'ruid_025' 'ruid_189' 'ruid_160' 'ruid_027'
 'ruid_186' 'ruid_185' 'ruid_184' 'ruid_183' 'ruid_010' 'ruid_178'
 'ruid_206' 'ruid_177' 'ruid_152' 'ruid_159' 'ruid_100' 'ruid_116'
 'ruid_115' 'ruid_114' 'ruid_068' 'ruid_069' 'ruid_110' 'ruid_109'
 'ruid_071' 'ruid_073' 'ruid_074' 'ruid_105' 'ruid_104' 'ruid_102'
 'ruid_099' 'ruid_157' 'ruid_076' 'ruid_097' 'ruid_077' 'ruid_095'
 'ruid_094' 'ruid_093' 'ruid_092' 'ruid_078' 'ruid_079' 'ruid_089'
 'ruid_088' 'ruid_082' 'ruid_085' 'ruid_117' 'ruid_118' 'ruid_064'
 'ruid_120' 'ruid_044' 'ruid_154' 'ruid_045' 'ruid_149' 'ruid_047'
 'ruid_146' 'ruid_144' 'ruid_049' 'ruid_050' 'ruid_140' 'ruid_052'
 'ruid_137' 'ruid_136' 'ruid_055' 'ruid_059' 'ruid_133' 'ruid_061'
 'ruid_130' 'ruid_129' 'ruid_128' 'ruid_127' 'ruid_126' 'ruid_125']
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
```

[10]:
```
# Split the bottom_pool randomly into buckets:
buckets_zeros, sums_zeros = random_split_buckets(bottom_pool, N, R)

for i in range(len(buckets_zeros)):
    print('Bucket of zeros ' + str(i) + ':')
    print(buckets_zeros[i])
```

```
Bucket of zeros 0:
['ruid_068' 'ruid_061' 'ruid_178' 'ruid_183' 'ruid_049' 'ruid_095'
 'ruid_198' 'ruid_044' 'ruid_160' 'ruid_157' 'ruid_163' 'ruid_073']
Bucket of zeros 1:
['ruid_097' 'ruid_120' 'ruid_206' 'ruid_207' 'ruid_192' 'ruid_064'
 'ruid_059' 'ruid_199' 'ruid_019' 'ruid_099' 'ruid_055' 'ruid_118']
```

```
Bucket of zeros 2:
['ruid_217' 'ruid_167' 'ruid_004' 'ruid_069' 'ruid_024' 'ruid_074'
 'ruid_047' 'ruid_014' 'ruid_035' 'ruid_195' 'ruid_208' 'ruid_184']
Bucket of zeros 3:
['ruid_125' 'ruid_089' 'ruid_218' 'ruid_105' 'ruid_082' 'ruid_110'
 'ruid_071' 'ruid_117' 'ruid_144' 'ruid_159' 'ruid_016' 'ruid_189']
Bucket of zeros 4:
['ruid_152' 'ruid_027' 'ruid_079' 'ruid_136' 'ruid_088' 'ruid_191'
 'ruid_045' 'ruid_092' 'ruid_115' 'ruid_127' 'ruid_006' 'ruid_219']
Bucket of zeros 5:
['ruid_161' 'ruid_094' 'ruid_166' 'ruid_137' 'ruid_076' 'ruid_130'
 'ruid_128' 'ruid_085' 'ruid_013' 'ruid_093' 'ruid_025' 'ruid_010']
Bucket of zeros 6:
['ruid_114' 'ruid_154' 'ruid_109' 'ruid_215' 'ruid_116' 'ruid_162'
 'ruid_077' 'ruid_102' 'ruid_186' 'ruid_146' 'ruid_104' 'ruid_177']
Bucket of zeros 7:
['ruid_133' 'ruid_149' 'ruid_050' 'ruid_100' 'ruid_140' 'ruid_129'
 'ruid_052' 'ruid_126' 'ruid_012' 'ruid_078' 'ruid_185' 'ruid_017']
```

[11]:
```
# Select the remaining rows without articles:
remaining_pool = spb_np[T:]
nr = len(remaining_pool)
print(nr)

# Now add this rows into randomly selected buckets of zeros
rdm = random.sample(range(N), nr)
print(rdm)

for i in range(nr):
    buckets_zeros[rdm[i]] = np.concatenate((buckets_zeros[rdm[i]],␣
 ↪[remaining_pool[i,0]]))

for i in range(len(buckets_zeros)):
    print('Bucket of zeros ' + str(i) + ':')
    print(buckets_zeros[i])
```

```
5
[3, 5, 4, 1, 6]
Bucket of zeros 0:
['ruid_068' 'ruid_061' 'ruid_178' 'ruid_183' 'ruid_049' 'ruid_095'
 'ruid_198' 'ruid_044' 'ruid_160' 'ruid_157' 'ruid_163' 'ruid_073']
Bucket of zeros 1:
['ruid_097' 'ruid_120' 'ruid_206' 'ruid_207' 'ruid_192' 'ruid_064'
 'ruid_059' 'ruid_199' 'ruid_019' 'ruid_099' 'ruid_055' 'ruid_118'
 'ruid_121']
Bucket of zeros 2:
['ruid_217' 'ruid_167' 'ruid_004' 'ruid_069' 'ruid_024' 'ruid_074'
 'ruid_047' 'ruid_014' 'ruid_035' 'ruid_195' 'ruid_208' 'ruid_184']
```

```
Bucket of zeros 3:
['ruid_125' 'ruid_089' 'ruid_218' 'ruid_105' 'ruid_082' 'ruid_110'
 'ruid_071' 'ruid_117' 'ruid_144' 'ruid_159' 'ruid_016' 'ruid_189'
 'ruid_124']
Bucket of zeros 4:
['ruid_152' 'ruid_027' 'ruid_079' 'ruid_136' 'ruid_088' 'ruid_191'
 'ruid_045' 'ruid_092' 'ruid_115' 'ruid_127' 'ruid_006' 'ruid_219'
 'ruid_122']
Bucket of zeros 5:
['ruid_161' 'ruid_094' 'ruid_166' 'ruid_137' 'ruid_076' 'ruid_130'
 'ruid_128' 'ruid_085' 'ruid_013' 'ruid_093' 'ruid_025' 'ruid_010'
 'ruid_123']
Bucket of zeros 6:
['ruid_114' 'ruid_154' 'ruid_109' 'ruid_215' 'ruid_116' 'ruid_162'
 'ruid_077' 'ruid_102' 'ruid_186' 'ruid_146' 'ruid_104' 'ruid_177'
 'ruid_001']
Bucket of zeros 7:
['ruid_133' 'ruid_149' 'ruid_050' 'ruid_100' 'ruid_140' 'ruid_129'
 'ruid_052' 'ruid_126' 'ruid_012' 'ruid_078' 'ruid_185' 'ruid_017']
```

[12]:
```
# Now merge the buckets with articles and the buckets without articles
final_buckets = [np.concatenate((buckets_art[i], buckets_zeros[i])) for i in␣
 ↪range(N)]

for i in range(len(final_buckets)):
    print('Final Bucket ' + str(i) + ':')
    print(final_buckets[i])
```

```
Final Bucket 0:
['ruid_220' 'ruid_150' 'ruid_193' 'ruid_032' 'ruid_084' 'ruid_212'
 'ruid_051' 'ruid_054' 'ruid_108' 'ruid_046' 'ruid_080' 'ruid_063'
 'ruid_176' 'ruid_148' 'ruid_072' 'ruid_068' 'ruid_061' 'ruid_178'
 'ruid_183' 'ruid_049' 'ruid_095' 'ruid_198' 'ruid_044' 'ruid_160'
 'ruid_157' 'ruid_163' 'ruid_073']
Final Bucket 1:
['ruid_057' 'ruid_203' 'ruid_053' 'ruid_213' 'ruid_058' 'ruid_091'
 'ruid_023' 'ruid_139' 'ruid_021' 'ruid_011' 'ruid_187' 'ruid_096'
 'ruid_156' 'ruid_070' 'ruid_181' 'ruid_097' 'ruid_120' 'ruid_206'
 'ruid_207' 'ruid_192' 'ruid_064' 'ruid_059' 'ruid_199' 'ruid_019'
 'ruid_099' 'ruid_055' 'ruid_118' 'ruid_121']
Final Bucket 2:
['ruid_145' 'ruid_066' 'ruid_143' 'ruid_103' 'ruid_060' 'ruid_209'
 'ruid_043' 'ruid_037' 'ruid_007' 'ruid_075' 'ruid_147' 'ruid_048'
 'ruid_180' 'ruid_194' 'ruid_087' 'ruid_217' 'ruid_167' 'ruid_004'
 'ruid_069' 'ruid_024' 'ruid_074' 'ruid_047' 'ruid_014' 'ruid_035'
 'ruid_195' 'ruid_208' 'ruid_184']
Final Bucket 3:
['ruid_171' 'ruid_015' 'ruid_101' 'ruid_210' 'ruid_033' 'ruid_040'
```

```
 'ruid_211' 'ruid_216' 'ruid_182' 'ruid_090' 'ruid_081' 'ruid_132'
 'ruid_005' 'ruid_034' 'ruid_067' 'ruid_125' 'ruid_089' 'ruid_218'
 'ruid_105' 'ruid_082' 'ruid_110' 'ruid_071' 'ruid_117' 'ruid_144'
 'ruid_159' 'ruid_016' 'ruid_189' 'ruid_124']
Final Bucket 4:
['ruid_170' 'ruid_214' 'ruid_158' 'ruid_020' 'ruid_008' 'ruid_113'
 'ruid_135' 'ruid_031' 'ruid_200' 'ruid_168' 'ruid_026' 'ruid_041'
 'ruid_098' 'ruid_119' 'ruid_083' 'ruid_152' 'ruid_027' 'ruid_079'
 'ruid_136' 'ruid_088' 'ruid_191' 'ruid_045' 'ruid_092' 'ruid_115'
 'ruid_127' 'ruid_006' 'ruid_219' 'ruid_122']
Final Bucket 5:
['ruid_169' 'ruid_151' 'ruid_018' 'ruid_173' 'ruid_009' 'ruid_065'
 'ruid_165' 'ruid_153' 'ruid_056' 'ruid_164' 'ruid_155' 'ruid_197'
 'ruid_131' 'ruid_142' 'ruid_134' 'ruid_161' 'ruid_094' 'ruid_166'
 'ruid_137' 'ruid_076' 'ruid_130' 'ruid_128' 'ruid_085' 'ruid_013'
 'ruid_093' 'ruid_025' 'ruid_010' 'ruid_123']
Final Bucket 6:
['ruid_205' 'ruid_138' 'ruid_062' 'ruid_022' 'ruid_029' 'ruid_086'
 'ruid_042' 'ruid_002' 'ruid_039' 'ruid_036' 'ruid_030' 'ruid_221'
 'ruid_174' 'ruid_038' 'ruid_107' 'ruid_114' 'ruid_154' 'ruid_109'
 'ruid_215' 'ruid_116' 'ruid_162' 'ruid_077' 'ruid_102' 'ruid_186'
 'ruid_146' 'ruid_104' 'ruid_177' 'ruid_001']
Final Bucket 7:
['ruid_106' 'ruid_204' 'ruid_112' 'ruid_202' 'ruid_190' 'ruid_188'
 'ruid_003' 'ruid_196' 'ruid_179' 'ruid_111' 'ruid_141' 'ruid_172'
 'ruid_175' 'ruid_201' 'ruid_028' 'ruid_133' 'ruid_149' 'ruid_050'
 'ruid_100' 'ruid_140' 'ruid_129' 'ruid_052' 'ruid_126' 'ruid_012'
 'ruid_078' 'ruid_185' 'ruid_017']
```

[13]:
```python
# Save the output:
with open('final_buckets.txt', 'w') as output:

    output.write(str(sums_art) + '\n')   # Save the sums per bucket

    # Save the elements of the buckets, connected by commas
    for i in range(N):
        line_bucket = ','.join(str(final_buckets[i])[1:-1].split())
        output.write(line_bucket + '\n')
```