Sample

March 31, 2021

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import re
```

1 Helper functions

These are borrowed from the `Convert.ipynb` file.

[2]:
```
headings = ['Building Identifier',
            'Country',
             'City',
             'Quality / Stage of Data',
             'Construction Date',
             'Building Type',
             'Gross Floor Area']
```

[3]:
```
df = pd.read_excel('../Dataset/dataset.xlsx',header=1,usecols='B:DWO')
```

[4]:
```
mapper = pd.read_excel('../Conversion/Mapping material names_20210324.
↪xlsx',header=2,usecols='B:U').replace(r'\n','', regex=True)
```

[5]:
```
additional_categories_map = {v:k for k,v in {
    'Continuous Footings':'0CF',
    'Foundation Walls':'0FW',
    'Spread Footings':'0SF',
    'Column Piers':'0CP',
    'Columns Supporting Floors':'CSF',
    'Floor Girders and Beams':'FGB',
    'Floor Trusses':'0FT',
    'Floor Joists':'0FJ',
    'Columns Supporting Roofs':'CSR',
    'Roof Girders and Beams':'RGB',
    'Roof Trusses':'0RT',
    'Roof Joists':'0RJ',
    'Parking Bumpers':'0PB',
    'Precast Concrete Stair Treads':'PCS',
    'Roof Curbs':'0RC',
```

```
    'Exterior Wall Construction':'EWC',
    'Composite Decking':'CPD',
    'Cast-in-Place concrete':'CIC',
    'Floor Structural Frame':'FSF',
    'Associated Metal Fabrications':'AMF',
    'Floor Construction Supplementary Components':'FCS',
    'Roof Construction Supplementary Components':'RCS',
    'Residential Elevators':'ORE',
    'Vegetated Low-Slope Roofing':'VLR',
    'Swimming Pools':'SWP',
    'Excavation Soil Anchors':'ESA',
    'Floor Trusses':'FTS',
    'Roof Window and Skylight Performance':'RWS'}.items()
}

additional_categories_map['0FT'] = 'Floor Trusses'
```

[6]:
```
def get_material_name(l):
    try:
        split = re.split('[_\.\ ]',l) #Split up the code into its requisite␣
↪parts
        result = mapper[mapper['Unnamed: 7'] == split[1]+'.'+split[2]] #Filter␣
↪by Level 4 Master Format
        if len(result) == 0:
            result = mapper #If that code does not exist in the table, reset
        if len(result) == 1:
            return result['Mapping Table'].values[0] #If it maps to exactly one␣
↪value, return that. We do this check after every step
        if split[3] != '000': #Check if there is an additional code, and if so␣
↪filter by that
            result = result[result['Level 5\n'] ==␣
↪additional_categories_map[split[3]]]
            if len(result) == 1:
                return result['Mapping Table'].values[0]

        #Now filter by UniFormat.
        #Filter only by the level of UniFormat present. If the code is XX 00␣
↪00, for example, then we only have Level 1.
        if int(split[5]) == 0:
            result = result[result['Unnamed: 12'] == f'{split[4]} 00 00']
            if len(result) == 1:
                return result['Mapping Table'].values[0]
        elif int(split[6]) == 0:
            result = result[(result['Unnamed: 14'] == f'{split[4]} {split[5]}␣
↪00') | (result['Unnamed: 16'] == f'{split[4]} {split[5]} 00')]
            if len(result) == 1:
```

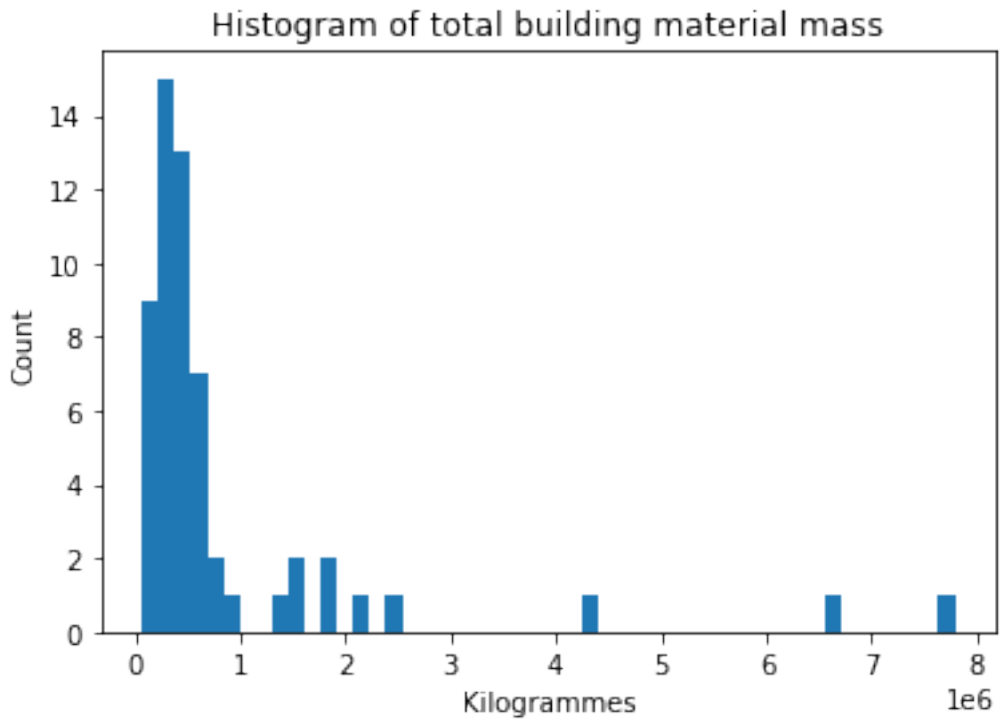
```
                return result['Mapping Table'].values[0]
        else:
            result = result[result['Unnamed: 18'] == f'{split[4]} {split[5]}␣
↪{split[6]}']
            if len(result) == 1:
                return result['Mapping Table'].values[0]

        #If we couldn't find it, or there is an unspecified edge case, return␣
↪None.
        if len(result) == 0:
            return None

        #If there are multiple results but they all map to the same material,␣
↪return that material.
        if all(element == result['Mapping Table'].values[0] for element in␣
↪result['Mapping Table'].values):
            return result['Mapping Table'].values[0]
        else:
            return None
    except:
        return None
```

2 1. Plot sample figures

Here we plot building material mass, and volume histograms.

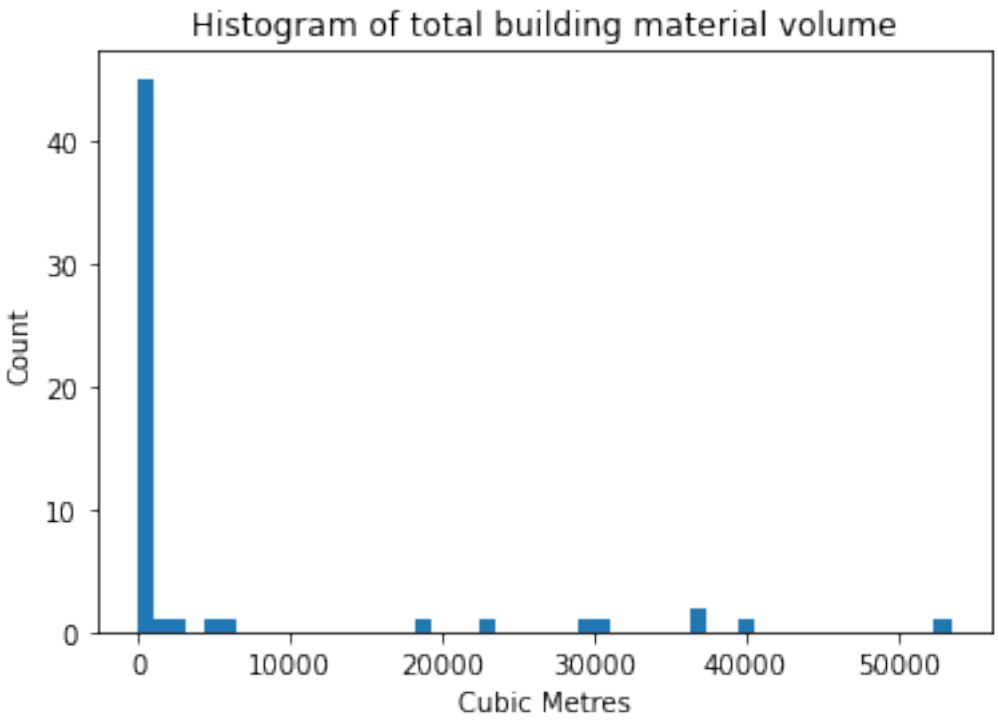
[7]:
```
plt.hist(df[[c for c in df.columns if 'kg' in c]].sum(axis=1),bins=50);
plt.title('Histogram of total building material mass')
plt.xlabel('Kilogrammes')
plt.ylabel('Count');
```

[7]:
```
Text(0, 0.5, 'Count')
```



[8]:
```
plt.hist(df[[c for c in df.columns if 'm3' in c]].sum(axis=1),bins=50);
plt.title('Histogram of total building material volume')
plt.xlabel('Cubic Metres')
plt.ylabel('Count');
```

[8]: Text(0, 0.5, 'Count')



3 2. Investigate a specific material

In this example, we use the helper function `get_material_name()` to select columns which match `steel`. Then, we calculate the average amount of steel used by floor, produce a table of values by Level 3 MasterFormat only, and calculate the average values for these by year in the dataset.

[9]:
```
material = 'steel'
cols = []
for column in df.columns[7:]: #Iterate through columns that represent materials
    if get_material_name(column) == 'steel' and 'kg' in column: #If that column␣
 ↪represents steel and is a mass value:
        cols.append(column) #Append to cols
```

[10]:
```
steel_df = df[df.columns[1:7].to_list() + cols].fillna(0) #Select only the␣
 ↪heading columns and the columns related to steel
```

[11]:
```
grouping_function = lambda x: x.split('_')[0] #This function takes in a full␣
 ↪column name, like "000_G2010.20.000_03 00 00.00_m3_1", and returns only the␣
 ↪floor.
steel_df[cols].groupby(grouping_function,axis=1).sum().mean()
```

[11]:
```
000     9723.396120
001     5793.423194
```

002	5537.649070
003	3468.436813
004	1067.126362
005	1674.052936
006	1692.734624
007	2929.281477
008	2036.739192
009	1328.666273
00F	36029.486850
00R	285.259726
010	1017.699219
011	1070.051550
012	169.526162
013	199.462368
014	122.536579
015	113.883947
016	137.662193
017	127.333246
018	99.919211
019	131.161842
020	131.161842
021	467.557719
022	91.721053
023	91.610877
024	95.219123
025	94.792193
026	191.878333
027	2013.988158
028	95.205351
029	81.708860
030	81.571140
031	85.509912
032	91.542018
033	314.808702
034	9.172105
035	9.764298
036	10.439123
037	10.163684
038	10.439123
039	10.163684
040	10.439123
041	10.163684
042	10.439123
043	15.603596
044	74.740263
0P1	859.013105
0P2	569.529895

```
0P3     1080.440947
0P4     4120.500807
0P5      424.547281
999     8593.565847
B01      975.846839
M00      306.215105
M01      226.528965
P01     5795.620553
P02       34.495105
P03     3049.679053
dtype: float64
```

Now, we will aggregate to Level 3 MasterFormat codes, and display these values for the first three entries.

[12]:
```
f = lambda x: re.split('[_\.\ ]',x)[1] #This function takes in a full column␣
↪name and returns only the Level 3 MasterFormat code.
steel_general_df = pd.concat([steel_df[headings[1:]],steel_df[cols].
↪groupby(f,axis=1).sum()],axis=1)
```

[13]:
```
steel_general_df.head(3)
```

[13]:
```
  Country City Quality / Stage of Data  Construction Date Building Type  \
0      CA  TOR                   00IFC              2021           SND
1      CA  TOR                   00IFC              2021           SND
2      CA  TOR                   00IFC              2021           SND

   Gross Floor Area    A1010  A1020     A4010    A4020  …  B2010  B2050  \
0            521.18  182.801    0.0  571.6172  33.4976  …    0.0    0.0
1            389.24    0.000    0.0    0.0000  27.1446  …    0.0    0.0
2            411.64  522.636    0.0    0.0000  15.0979  …    0.0    0.0

   B2070     B3010  B3060  C1030  C1090  D1010  G2010  G2060
0    0.0  163.5038    0.0    0.0    0.0    0.0    0.0    0.0
1    0.0    0.0000    0.0    0.0    0.0    0.0    0.0    0.0
2    0.0    0.0000    0.0    0.0    0.0    0.0    0.0    0.0

[3 rows x 26 columns]
```

We can also calculate the average for each Level 3 MasterFormat code by year of construction:

[14]:
```
steel_general_df.groupby('Construction Date').mean()
```

[14]:
```
                   Gross Floor Area          A1010        A1020  \
Construction Date
1913                     161.080000       0.000000       0.0000
1917                     199.930000       0.000000       0.0000
1969                     373.605000       0.000000       0.0000
```

1988	21934.000000	134033.498513	0.0000
2007	73600.000000	0.000000	244138.1400
2009	73083.000000	202831.440000	0.0000
2011	11282.500000	20097.177500	22038.8750
2016	30345.000000	7123.286250	256540.0660
2017	39392.013333	21271.223747	210904.2151
2018	29040.423333	29656.776667	228291.6590
2020	529.510000	837.089200	0.0000
2021	451.422000	990.563554	0.0000

Construction Date	A4010	A4020	A4040	A5010	\
1913	96.325400	0.000000	0.000000	0.000000	
1917	0.000000	20.818800	0.000000	0.000000	
1969	0.000000	98.436400	0.000000	0.000000	
1988	54829.743735	0.000000	0.000000	1235.905423	
2007	150141.926000	0.000000	0.000000	0.000000	
2009	128379.999000	0.000000	0.000000	0.000000	
2011	38548.367000	0.000000	360.315000	1671.340000	
2016	28069.480500	0.000000	517.236500	8969.017500	
2017	30701.309915	0.000000	680.269612	8357.110000	
2018	6067.265000	0.000000	0.000000	25720.001667	
2020	143.213200	58.670900	0.000000	594.014000	
2021	337.313286	112.766049	0.000000	57.623263	

Construction Date	A6010	B1010	B1020	…	\
				…	
1913	0.000000	0.000000	0.000000	…	
1917	0.000000	0.000000	0.000000	…	
1969	0.000000	0.000000	0.000000	…	
1988	0.000000	259.573171	0.000000	…	
2007	0.000000	65657.800000	4498.000000	…	
2009	0.000000	155524.200000	127481.444506	…	
2011	0.000000	185403.503350	0.000000	…	
2016	0.000000	16048.100000	0.000000	…	
2017	0.000000	285336.159133	2272.634333	…	
2018	359.837894	8281.158667	2951.329000	…	
2020	0.000000	791.986800	0.000000	…	
2021	0.000000	183.398457	0.000000	…	

Construction Date	B2010	B2050	B2070	B3010	B3060	\
1913	0.000000	0.000000	0.000	0.000000	0.000	
1917	0.000000	0.000000	0.000	0.000000	0.000	
1969	0.000000	0.000000	0.000	0.000000	0.000	
1988	10078.408608	0.000000	0.000	0.000000	0.000	
2007	0.000000	0.000000	0.000	0.000000	0.000	

```
2009                2125.780000    0.000000  177182.000   0.000000     0.000
2011                3010.789500    0.000000       0.000   0.000000     0.000
2016                   0.000000    0.000000       0.000   0.000000     0.000
2017                2267.603333    0.000000       0.000   0.000000     0.000
2018                 361.100000    0.000000     474.744   0.000000  1798.362
2020                   0.000000  266.537200       0.000   0.000000     0.000
2021                   0.000000   52.570857       0.000  18.768566     0.000

                        C1030         C1090         D1010      G2010  \
Construction Date
1913                 0.000000      0.000000      0.000000     0.0000
1917                 0.000000      0.000000      0.000000     0.0000
1969                 0.000000      0.000000      0.000000     0.0000
1988                 0.000000      0.000000    668.292683     0.0000
2007                 0.000000  33330.000000  15851.600000     0.0000
2009                 0.000000      0.000000  13919.200000  7047.2590
2011                 0.000000      0.000000      0.000000  2129.4695
2016                 0.000000      0.000000      0.000000     0.0000
2017                 0.000000      0.000000      0.000000     0.0000
2018                 0.000000   1098.978454      0.000000     0.0000
2020                 0.000000      0.000000      0.000000     0.0000
2021                39.517714      0.000000      0.000000     0.0000

                         G2060
Construction Date
1913                  0.000000
1917                  0.000000
1969                  0.000000
1988                  0.000000
2007                  0.000000
2009                  0.000000
2011               3397.480000
2016                  0.000000
2017               1264.111667
2018                  0.000000
2020                  0.000000
2021                  0.000000

[12 rows x 21 columns]
```

We can get the average amount of steel in KG used per building type:

[15]:
```
steel_general_df.groupby('Building Type').sum().mean(axis=1)
```

[15]:
```
Building Type
APB    107688.992252
EDU     56688.647637
```

```
INS     10796.235456
MIX     16932.120205
OFF     97247.792600
ROW       927.811000
SMD       439.309182
SND      7954.707846
dtype: float64
```

[16]:
```
f = lambda x: re.split('[_\.\ ]',x)[1][0:3] #From a full code, return only the use code and uncertainty code.
pd.concat([df[headings[1:]],df[cols].groupby(f,axis=1).sum()],axis=1).groupby('Building Type').mean()
```

[16]:

Building Type	Construction Date	Gross Floor Area	A10	\
APB	2015.80	45113.208000	313712.342400	
EDU	2016.50	7901.000000	137559.628021	
INS	1988.00	21934.000000	134033.498513	
MIX	2018.00	33975.250000	198095.110000	
OFF	2009.00	52643.666667	162387.978333	
ROW	2018.00	1961.020000	0.000000	
SMD	1994.75	236.615000	148.803500	
SND	2015.60	465.227000	956.498910	

Building Type	A40	A50	A60	B10	\
APB	28267.818600	18204.180000	118.378000	61664.079600	
EDU	26121.268789	0.000000	0.000000	449976.727200	
INS	54829.743735	1235.905423	0.000000	12749.331064	
MIX	6271.365000	31593.110000	487.623683	97607.035000	
OFF	108668.753667	299.346667	0.000000	299745.711069	
ROW	0.000000	0.000000	0.000000	16432.822000	
SMD	36.032000	0.000000	0.000000	0.000000	
SND	423.302155	124.672105	0.000000	259.472000	

Building Type	B20	B30	C10	D10	\
APB	2514.058200	1079.017200	384.216909	0.000000	
EDU	0.000000	0.000000	0.000000	0.000000	
INS	10078.408608	0.000000	0.000000	668.292683	
MIX	1083.300000	0.000000	1375.850817	0.000000	
OFF	60328.703333	0.000000	11110.000000	9923.600000	
ROW	0.000000	0.000000	0.000000	0.000000	
SMD	0.000000	0.000000	0.000000	0.000000	
SND	79.316650	16.422495	34.578000	0.000000	

Building Type	G20

```
Building Type
APB             758.467000
EDU               0.000000
INS               0.000000
MIX               0.000000
OFF            6033.719333
ROW               0.000000
SMD               0.000000
SND               0.000000
```

4 3. Uncertainty by Building Type

In this section, we look at the uncertainty code associated with each column. We collect these by building type and then report the number of each value per type of building.

[17]:
```
uncertainty_level = {}
for k,v in df.iterrows():
    #Initialise empty lists for each building type as they occur
    if v['Building Type'] not in uncertainty_level.keys():
        uncertainty_level[v['Building Type']] = []
    #Append the uncertainty value for each column that is non-NaN
    for key in v[~v.isna()].keys()[7:]:
        uncertainty_level[v['Building Type']].append(key.split('_')[-1])
```

[18]:
```
from collections import Counter
```

[19]:
```
for k,v in uncertainty_level.items():
    uncertainty_level[k] = Counter(v) #Construct a Counter object per building␣
 ↪type
```

[20]:
```
uncertainty_level
```

[20]:
```
{'SND': Counter({'1': 1812,
          '2': 731,
          '4': 357,
          '1.1': 1088,
          '4.1': 204,
          '2.1': 314}),
 'OFF': Counter({'1': 494, '3': 307, '1.1': 109, '3.1': 307}),
 'APB': Counter({'1': 1167, '2': 1, '3': 985, '1.1': 298, '3.1': 312}),
 'SMD': Counter({'1': 204, '2': 61, '4': 27, '1.1': 107, '2.1': 9, '4.1': 10}),
 'EDU': Counter({'1': 93, '3': 24, '1.1': 38, '3.1': 24, '2': 6}),
 'INS': Counter({'1': 90, '3': 77, '2': 1, '1.1': 90, '3.1': 77, '2.1': 1}),
 'ROW': Counter({'1': 15, '3': 5, '1.1': 14, '3.1': 5}),
 'MIX': Counter({'1': 364, '3': 276, '1.1': 287})}
```

Next, we aggregate columns by use code and uncertainty combined, and report the average by

building type.

[21]:
```
f = lambda x: re.split('[_\.\ ]',x)[1][0] + x.split('_')[-1] #From a full code,␣
↪return only the use code and uncertainty code.
by_function_df = pd.concat([df[headings[1:]],df[cols].groupby(f,axis=1).
↪sum()],axis=1)
```

[22]:
```
by_function_df.groupby('Building Type').mean()
```

[22]:

Building Type	Construction Date	Gross Floor Area	A1	A1.1	A2	A2.1	A3	A3.1	A4	A4.1	…	B4.1	C1	C1.1	C2	C2.1	D1	D1.1	G1	G3
APB	2015.80	45113.208000	231737.663200	1985.845200	0.000000	0.00000	96188.909000	30390.301600	0.000000	0.000000	…	0.000000	192.108455	192.108455	0.000	0.000	0.000000	0.000000	225.295	533.172000
EDU	2016.50	7901.000000	0.000000	0.000000	0.000000	0.00000	74976.547506	88704.349305	0.000000	0.000000	…	0.000000	0.000000	0.000000	0.000	0.000	0.000000	0.000000	0.000	0.000000
INS	1988.00	21934.000000	0.000000	0.000000	0.000000	0.00000	92557.312757	97541.834914	0.000000	0.000000	…	0.000000	0.000000	0.000000	0.000	0.000	334.146341	334.146341	0.000	0.000000
MIX	2018.00	33975.250000	151968.510000	0.000000	0.000000	0.00000	84478.698683	0.000000	0.000000	0.000000	…	0.000000	1375.850817	0.000000						
OFF	2009.00	52643.666667	0.000000	0.000000	0.000000	0.00000	127794.205833	143561.872833	0.000000	0.000000	…	0.000000	5555.000000	5555.000000						
ROW	2018.00	1961.020000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000	…	0.000000	0.000000	0.000000						
SMD	1994.75	236.615000	82.653250	82.653250	11.036450	0.00000	0.000000	0.000000	4.246275	4.246275	…	0.000000	0.000000	0.000000						
SND	2015.60	465.227000	676.023563	676.023563	68.474865	43.26537	0.000000	0.000000	20.342905	20.342905	…	6.686572	0.000000	0.000000						

```
MIX              0.000   0.000     0.000000     0.000000    0.000     0.000000
OFF              0.000   0.000  4961.800000  4961.800000    0.000  2872.053333
ROW              0.000   0.000     0.000000     0.000000    0.000     0.000000
SMD              0.000   0.000     0.000000     0.000000    0.000     0.000000
SND             17.289  17.289     0.000000     0.000000    0.000     0.000000

                   G3.1
Building Type
APB               0.000
EDU               0.000
INS               0.000
MIX               0.000
OFF            3161.666
ROW               0.000
SMD               0.000
SND               0.000

[8 rows x 27 columns]
```

Next, we report the total amount of material falling under each uncertainty code by year of construction.

[23]:
```
f = lambda x: x.split('_')[-1] #Select only the uncertainty code.
pd.concat([df[headings[1:]],df[cols].groupby(f,axis=1).sum()],axis=1).
↪groupby('Construction Date').mean()
```

[23]:
```
                    Gross Floor Area              1            1.1           2  \
Construction Date
1913                      161.080000      48.162700      48.162700    0.000000
1917                      199.930000       0.000000       0.000000   20.818800
1969                      373.605000       0.000000       0.000000   98.436400
1988                    21934.000000     463.932927     463.932927    0.000000
2007                    73600.000000   59668.700000   59668.700000    0.000000
2009                    73083.000000  237053.422253  237053.422253    0.000000
2011                    11282.500000   93514.931675   93514.931675    0.000000
2016                    30345.000000  133494.550000    8024.050000    0.000000
2017                    39392.013333  316381.280567  142282.354567    0.000000
2018                    29040.423333  190121.507697    6725.995424    0.000000
2020                      529.510000    1076.002180    1076.002180  157.941140
2021                      451.422000     758.911369     758.911369  133.872774

                          2.1             3            3.1           4  \
Construction Date
1913                 0.000000        0.000000       0.000000    0.000000
1917                 0.000000        0.000000       0.000000    0.000000
1969                 0.000000        0.000000       0.000000    0.000000
1988                 0.000000   103273.679739  109393.634433    0.000000
```

```
2007                 0.000000  276886.770000  298443.498000    0.000000
2009                 0.000000  155250.235000  185134.243000    0.000000
2011                 0.000000   45065.083750   49359.975750    0.000000
2016                 0.000000  122269.048750   81174.024000    0.000000
2017                 0.000000  126400.375837    7595.330870    0.000000
2018                 0.000000  122584.599561   63667.058667    0.000000
2020               133.268600       0.000000       0.000000  124.148600
2021               113.545023       0.000000       0.000000   13.640606
```

```
                         4.1
Construction Date
1913                 0.000000
1917                 0.000000
1969                 0.000000
1988                 0.000000
2007                 0.000000
2009                 0.000000
2011                 0.000000
2016                 0.000000
2017                 0.000000
2018                 0.000000
2020               124.148600
2021                13.640606
```

5 4. Material Intensity

We can easily calculate material intensity by dividing columns which are measured in kilograms by the `Gross Floor Area`:

[24]:
```
kilogram_columns = [d for d in df.columns if 'kg' in d]
df_mi = df[kilogram_columns].div(df['Gross Floor Area'],axis=0)
```

[25]:
```
f = lambda x: re.split('[_\.\ ]',x)[1][0:3]
pd.concat([df[headings[1:]],df_mi[kilogram_columns].groupby(f,axis=1).
 ↪sum()],axis=1)[df['Building Type'] == 'SND']
```

[25]:
```
   Country City Quality / Stage of Data  Construction Date Building Type  \
0       CA  TOR                   00IFC              2021           SND
1       CA  TOR                   00IFC              2021           SND
2       CA  TOR                   00IFC              2021           SND
3       CA  TOR                   00IFC              2021           SND
6       CA  TOR                   00IFC              2021           SND
7       CA  TOR                   00IFC              2021           SND
8       CA  TOR                   00IFC              2021           SND
9       CA  TOR                   00IFC              2021           SND
12      CA  TOR                   00IFC              2021           SND
13      CA  TOR                   00IFC              2021           SND
```

14	CA	TOR	00IFC	2021	SND
15	CA	TOR	00IFC	2021	SND
16	CA	TOR	00IFC	1969	SND
17	CA	TOR	00IFC	1969	SND
18	CA	TOR	00IFC	2021	SND
19	CA	TOR	00IFC	2021	SND
20	CA	TOR	00IFC	2020	SND
21	CA	TOR	00IFC	2021	SND
22	CA	TOR	00IFC	2021	SND
24	CA	TOR	00IFC	2021	SND
25	CA	TOR	00IFC	2021	SND
27	CA	TOR	00IFC	2021	SND
28	CA	TOR	00IFC	2021	SND
30	CA	TOR	00IFC	2021	SND
31	CA	TOR	00IFC	2021	SND
32	CA	TOR	00IFC	2020	SND
34	CA	TOR	00IFC	2021	SND
35	CA	TOR	00IFC	2021	SND
36	CA	TOR	00IFC	2021	SND
37	CA	TOR	00IFC	2020	SND
38	CA	TOR	00IFC	2021	SND
40	CA	TOR	00IFC	2021	SND
41	CA	TOR	00IFC	1913	SND
42	CA	TOR	00IFC	2021	SND
43	CA	TOR	00IFC	2021	SND
44	CA	TOR	00IFC	2021	SND
45	CA	TOR	00IFC	2021	SND
46	CA	TOR	00IFC	2021	SND
48	CA	TOR	00IFC	2020	SND
49	CA	TOR	00IFC	2021	SND

	Gross Floor Area	A10	A20	A40	A50	…	A90	\
0	521.18	353.958084	14.438812	323.952856	0.000000	…	0.0	
1	389.24	281.318698	13.374339	194.232091	0.000000	…	0.0	
2	411.64	465.097017	19.208236	218.629213	0.000000	…	0.0	
3	269.56	258.361801	6.543260	128.098456	0.000000	…	0.0	
6	445.99	301.393384	16.469983	179.786278	0.108904	…	0.0	
7	438.45	270.947699	7.767302	277.432676	0.000000	…	0.0	
8	714.07	276.917123	15.274836	317.786761	0.000000	…	0.0	
9	343.24	285.386581	16.513088	141.281528	0.000000	…	0.0	
12	226.89	265.332998	5.559963	136.637311	1.871224	…	0.0	
13	611.73	344.014507	10.923807	211.850660	0.000000	…	0.0	
14	343.44	424.099610	15.616982	132.692813	0.000000	…	0.0	
15	613.38	351.176047	6.672007	209.540477	0.934876	…	0.0	
16	413.72	224.634608	9.729092	166.704176	0.000000	…	0.0	
17	333.49	355.746799	11.950137	196.595229	0.000000	…	0.0	
18	178.38	380.256408	0.000000	223.398638	0.000000	…	0.0	

19	323.80	151.150500	8.404137	161.509749	0.000000	…	0.0
20	837.56	318.446436	9.835182	146.453834	0.000000	…	0.0
21	587.86	428.797751	13.634641	307.141806	0.000000	…	0.0
22	568.21	259.885070	8.759483	280.260170	0.000000	…	0.0
24	294.84	262.791586	13.534551	162.155700	0.000000	…	0.0
25	496.77	256.167921	16.603660	296.424095	0.156035	…	0.0
27	643.30	164.379820	0.000000	154.144741	0.193518	…	0.0
28	701.61	269.790747	20.967024	205.862491	0.000000	…	0.0
30	378.70	417.101590	10.259954	231.374434	0.660343	…	0.0
31	324.16	385.909729	9.830264	163.544787	0.000000	…	0.0
32	533.53	313.166720	14.230290	163.397529	0.000000	…	0.0
34	423.03	243.607664	0.000000	153.019643	0.000000	…	0.0
35	328.16	396.879947	9.227271	156.998172	0.000000	…	0.0
36	421.59	425.772558	9.538939	147.225241	0.000000	…	0.0
37	628.59	385.687306	9.558155	214.893910	2.922499	…	0.0
38	464.51	414.319976	12.721251	211.159960	0.000000	…	0.0
40	346.14	289.830976	13.416815	174.360072	0.788831	…	0.0
41	161.08	346.479960	10.604605	212.185022	0.000000	…	0.0
42	891.97	247.987159	6.379470	167.233434	0.743619	…	0.0
43	525.61	501.351964	11.927482	169.380368	0.000000	…	0.0
44	502.87	278.679758	8.072675	199.009172	0.000000	…	0.0
45	379.18	400.408477	15.895027	162.621675	0.390219	…	0.0
46	549.65	276.863718	9.505033	184.664964	0.999793	…	0.0
48	393.82	194.293002	10.775143	252.664660	3.294658	…	0.0
49	648.14	360.590459	10.280731	255.218231	2.416208	…	0.0

	B10	B20	B30	C10	C20	D10	D20	F10	\
0	54.998131	147.811220	14.393646	29.836010	16.618827	0.0	0.0	0.0	
1	36.739564	133.423435	5.461939	41.974701	6.490936	0.0	0.0	0.0	
2	43.752969	182.905692	3.955589	35.166432	9.149811	0.0	0.0	0.0	
3	57.294905	370.711117	6.503479	36.241829	8.510443	0.0	0.0	0.0	
6	63.871222	114.888632	11.934602	34.740910	12.782125	0.0	0.0	0.0	
7	64.247009	255.228896	19.770004	19.607741	6.584780	0.0	0.0	0.0	
8	63.916950	206.174209	19.930097	82.669534	13.127789	0.0	0.0	0.0	
9	57.151395	251.349228	8.589688	44.584390	11.076655	0.0	0.0	0.0	
12	57.861266	233.301466	17.701736	45.636983	6.134611	0.0	0.0	0.0	
13	62.638873	186.629283	5.196340	33.144722	7.638991	0.0	0.0	0.0	
14	64.373435	172.208993	12.988933	34.085599	9.173841	0.0	0.0	0.0	
15	61.645384	227.511321	13.200796	47.641599	8.068881	0.0	0.0	0.0	
16	85.427092	185.208662	6.437864	52.273300	10.747684	0.0	0.0	0.0	
17	96.907849	196.916984	7.176775	37.636721	9.221026	0.0	0.0	0.0	
18	118.460418	209.154796	12.856830	49.366826	19.103711	0.0	0.0	0.0	
19	57.287124	234.534916	14.747402	45.872421	18.967307	0.0	0.0	0.0	
20	52.914078	168.631238	17.980621	46.116657	7.152371	0.0	0.0	0.0	
21	67.561056	167.079124	8.239013	48.773508	6.754074	0.0	0.0	0.0	
22	69.322434	107.054336	12.754287	61.481266	7.860492	0.0	0.0	0.0	
24	43.589376	191.979555	7.257634	42.419929	4.807604	0.0	0.0	0.0	

25	81.500414	125.515047	5.364168	33.695747	5.921358	0.0	0.0	0.0
27	48.465254	159.047231	11.769043	47.094195	8.492430	0.0	0.0	0.0
28	72.713045	89.563664	17.201826	47.085343	15.905247	0.0	0.0	0.0
30	122.228178	321.823739	5.581552	58.854052	11.176248	0.0	0.0	0.0
31	76.955896	214.582384	5.433643	57.892675	6.597902	0.0	0.0	0.0
32	50.634977	130.860537	9.149949	89.525185	7.103407	0.0	0.0	0.0
34	47.967391	186.690685	16.189200	47.087985	9.434697	0.0	0.0	0.0
35	82.711032	213.129635	10.235444	33.398187	5.648226	0.0	0.0	0.0
36	67.072054	211.004239	18.486244	42.087956	11.251282	0.0	0.0	0.0
37	89.561993	215.935364	14.163660	29.485083	11.399949	0.0	0.0	0.0
38	74.175085	181.326954	4.561602	50.413723	7.621364	0.0	0.0	0.0
40	53.083904	173.471707	15.817365	54.597851	7.916204	0.0	0.0	0.0
41	42.232507	68.518319	9.423899	52.429433	8.911150	0.0	0.0	0.0
42	54.816615	230.295800	14.334660	55.621415	7.577250	0.0	0.0	0.0
43	63.694023	203.994024	33.455200	47.750330	7.954358	0.0	0.0	0.0
44	64.397894	111.584536	6.113129	59.938314	9.128976	0.0	0.0	0.0
45	87.323301	202.330996	11.762340	41.173085	12.678865	0.0	0.0	0.0
46	52.893904	154.188851	15.992035	51.160466	6.701647	0.0	0.0	0.0
48	66.284358	170.668478	26.331975	45.166809	10.629628	0.0	0.0	0.0
49	70.897904	159.874639	13.455489	45.812261	10.178764	0.0	0.0	0.0

	G20
0	0.0
1	0.0
2	0.0
3	0.0
6	0.0
7	0.0
8	0.0
9	0.0
12	0.0
13	0.0
14	0.0
15	0.0
16	0.0
17	0.0
18	0.0
19	0.0
20	0.0
21	0.0
22	0.0
24	0.0
25	0.0
27	0.0
28	0.0
30	0.0
31	0.0

```
32  0.0
34  0.0
35  0.0
36  0.0
37  0.0
38  0.0
40  0.0
41  0.0
42  0.0
43  0.0
44  0.0
45  0.0
46  0.0
48  0.0
49  0.0

[40 rows x 21 columns]
```

[ ]: