

# Sample

October 12, 2021

```
[1]: import pandas as pd
from copy import deepcopy
import matplotlib.pyplot as plt
import re
import numpy as np
from matplotlib import gridspec
import matplotlib
```

## 1 Helper functions

These are borrowed from the Convert.ipynb file.

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

```
[3]: df = pd.read_excel('../Dataset/dataset.xlsx',header=3,index_col=1)
df = df.drop('Unnamed: 0',axis=1).T#.reset_index().rename({'Building_
↳Identifier':'index','index':'Building Identifier'},axis=1)
df = df[df.index.str.contains('0')]
```

```
[4]: df[[c for c in df.columns if 'kg' in c]] = df[[c for c in df.columns if 'kg' in
↳c]].astype('float')
```

```
[5]: df = pd.concat([df[headings].groupby(lambda x: x.split('.')[0],axis=0).
↳max(),df[[c for c in df.columns if 'kg' in c]].groupby(lambda x: x.split('.')[
↳'])[0],axis=0).mean(numeric_only=True)],axis=1)
```

```
[6]: name_conversion = pd.read_csv('name_conversion.csv')
building_name_conversion = pd.read_csv('building_type_name_conversion.csv')
```

```
[7]: building_name_map = {k['Building Code']:k['Building Type'] for _,k in
↳building_name_conversion.iterrows()}
```

```
[8]: name_map = {k.Code:k.Category for _,k in name_conversion.iterrows()}
```

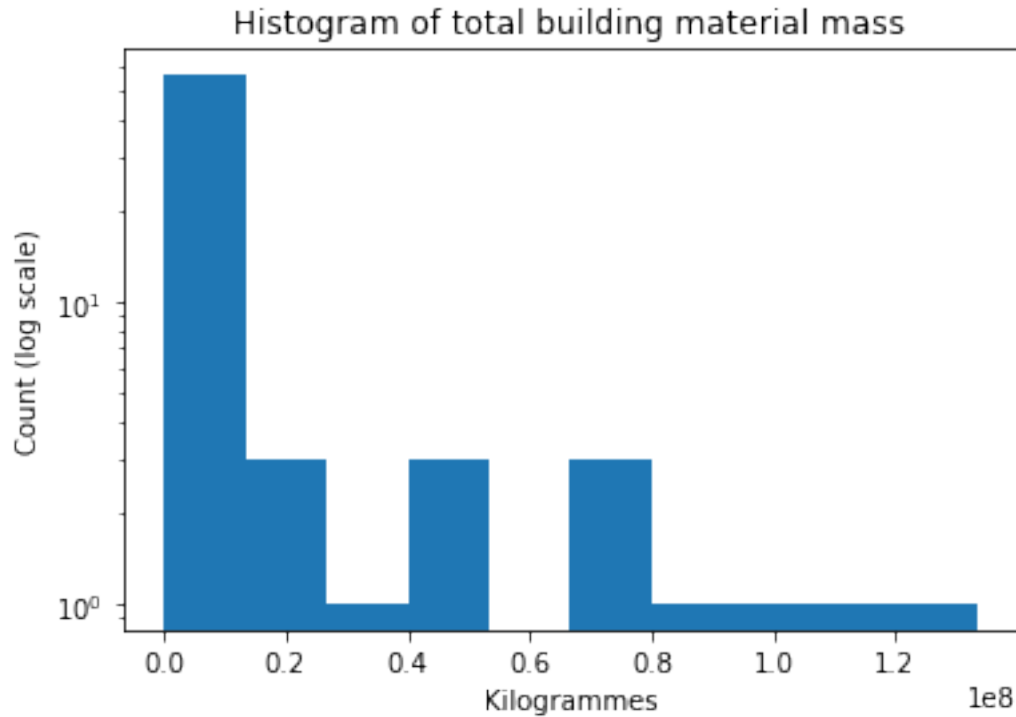
```
[9]: additional_categories_map = {v:k for k,v in {
    'Continuous Footings':'OCF',
    'Foundation Walls':'OFW',
    'Spread Footings':'OSF',
    'Column Piers':'OCP',
    'Columns Supporting Floors':'CSF',
    'Floor Girders and Beams':'FGB',
    'Floor Trusses':'OFT',
    'Floor Joists':'OFJ',
    'Columns Supporting Roofs':'CSR',
    'Roof Girders and Beams':'RGB',
    'Roof Trusses':'ORT',
    'Roof Joists':'ORJ',
    'Parking Bumpers':'OPB',
    'Precast Concrete Stair Treads':'PCS',
    'Roof Curbs':'ORC',
    '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',
    'Rainwater Storage Tanks':'RST',
    'Gray Water Tanks':'GWT'}.items()
}

additional_categories_map['OFT'] = 'Floor Trusses'
```

## 2 1. Plot sample figures

Here we plot building material mass.

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



### 3 2. Investigate a specific material

In this example, we select only columns that match the MasterFormat code for Concrete. Then, we aggregate based on Level 2 UniFormat code.

```
[11]: cols = [d for d in df.columns if ('_03' in d or '_04 22' in d) and not '_03 20' in d]
```

```
[12]: f = lambda x: re.split('[_\\.\\ ]',x)[1][0:3]
concrete_df = pd.concat([df[headings],df[cols].groupby(f,axis=1).sum()],axis=1).
    rename(columns=name_map)
```

```
[13]: concrete_df
```

```
[13]: Building Identifier Country City Quality / Stage of Data Construction Date \
001 CA TOR 00IFC 2021
002 CA TOR 00IFC 2021
003 CA TOR 00IFC 2021
004 CA TOR 00IFC 2021
005 CA TOR 00IFC 2011
.. ...
066 CA TOR 00IFT 2020
067 CA TOR 00IFC 2019
```

068	CA	TOR	OIFBP	2021
069	CA	TOR	00IFC	2020
070	CA	TOR	00IFC	2021

Building Identifier	Building Type	Gross Floor Area	Foundations \
001	SND	521.18	1.710150e+05
002	SND	389.24	1.082862e+05
003	SND	411.64	1.911912e+05
004	SND	269.56	6.739916e+04
005	OFF	11248.00	1.278753e+06
..	...	...	...
066	MIX	95769.00	1.566786e+07
067	LNW	131.00	2.657254e+04
068	LNW	71.00	2.721844e+04
069	LNW	98.00	2.846246e+04
070	LNW	131.00	3.930037e+03

Building Identifier	Subgrade Enclosures	Slabs-On-Grade \
001	0.000	6.751475e+04
002	0.000	3.578757e+04
003	0.000	3.254672e+04
004	0.000	1.618022e+04
005	1027239.110	6.846302e+05
..	...	...
066	3484448.795	1.395514e+06
067	0.000	2.924485e+04
068	0.000	2.206696e+04
069	0.000	1.264324e+04
070	0.000	1.647018e+04

Building Identifier	Substructure Interior	Water And Gas Mitigation \
001	0.000000e+00	0.0
002	0.000000e+00	0.0
003	0.000000e+00	0.0
004	0.000000e+00	0.0
005	7.359709e+05	0.0
..	...	...
066	1.368861e+07	0.0
067	0.000000e+00	0.0
068	0.000000e+00	0.0
069	0.000000e+00	0.0
070	0.000000e+00	0.0

Building Identifier	Substructure Related Activities	Superstructure \
001	0.0	1.949675e+03
002	0.0	1.409585e+03
003	0.0	1.562240e+02

004	0.0	2.269760e+01
005	0.0	7.126901e+06
..	...	...
066	0.0	5.737703e+07
067	0.0	0.000000e+00
068	0.0	0.000000e+00
069	0.0	0.000000e+00
070	0.0	0.000000e+00

Building Identifier	Exterior Vertical Enclosures	\
001	0.00	
002	0.00	
003	0.00	
004	0.00	
005	311760.72	
..	...	
066	71331.23	
067	0.00	
068	0.00	
069	0.00	
070	0.00	

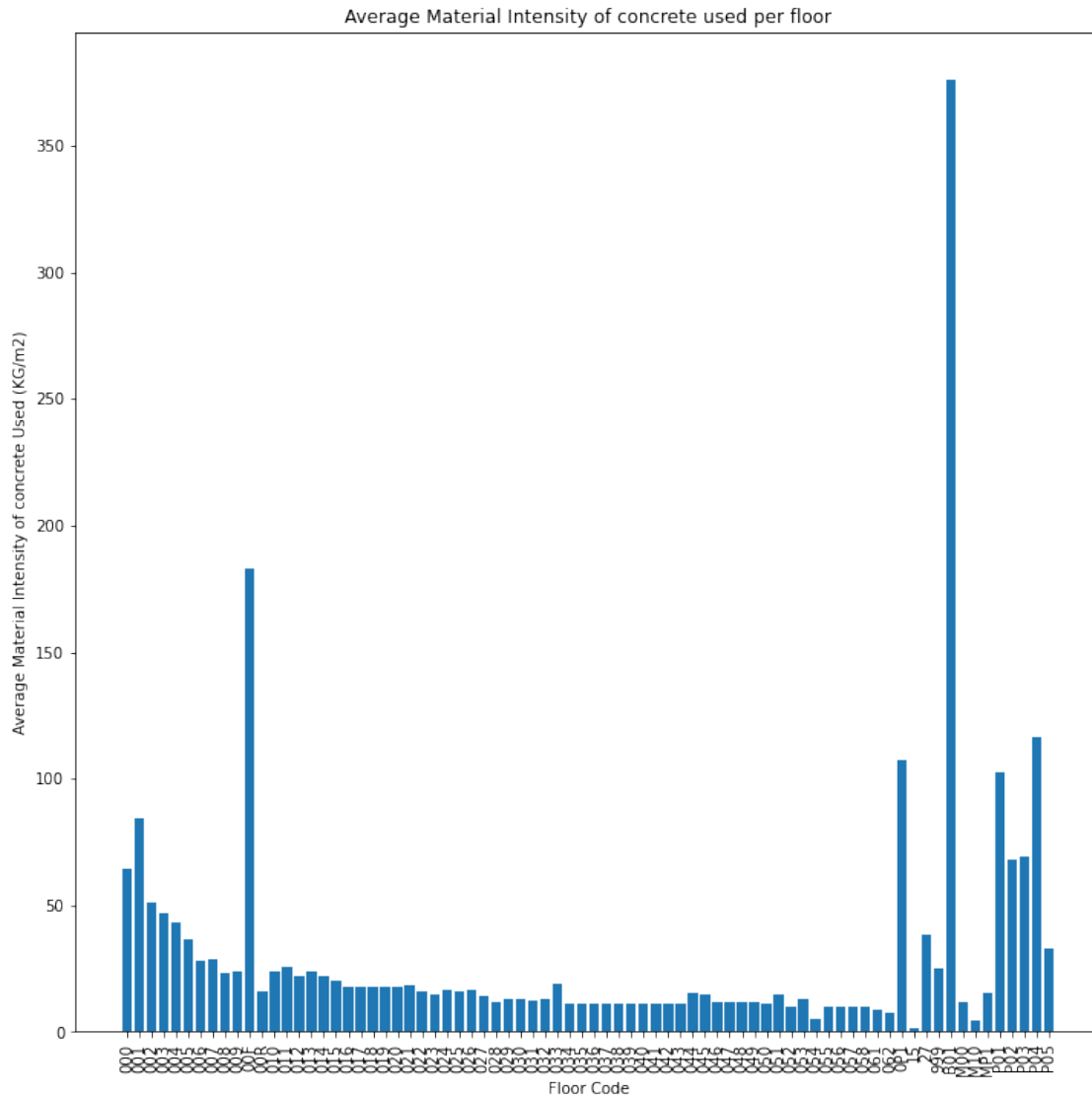
Building Identifier	Exterior Horizontal Enclosures	Interior Construction	\
001	0.0	0.000000e+00	
002	0.0	0.000000e+00	
003	0.0	0.000000e+00	
004	0.0	0.000000e+00	
005	552.0	1.175564e+06	
..	...	...	
066	0.0	1.463901e+07	
067	0.0	0.000000e+00	
068	0.0	0.000000e+00	
069	0.0	0.000000e+00	
070	0.0	0.000000e+00	

Building Identifier	Conveying	Plumbing	Special Construction	\
001	0.000	0.0	0.000	
002	0.000	0.0	0.000	
003	0.000	0.0	0.000	
004	0.000	0.0	0.000	
005	0.000	0.0	0.000	
..	...	...	...	
066	8273703.915	0.0	711760.625	
067	0.000	0.0	0.000	
068	0.000	0.0	0.000	
069	0.000	0.0	0.000	
070	0.000	0.0	0.000	

Building Identifier	Site Improvements
001	0.0000
002	0.0000
003	0.0000
004	0.0000
005	169830.9495
..	...
066	0.0000
067	0.0000
068	0.0000
069	0.0000
070	0.0000

[70 rows x 20 columns]

```
[14]: 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.
to_draw = df[cols].groupby(grouping_function,axis=1).sum().replace(0,np.NaN).
      ↪div(df['Gross Floor Area'],axis='rows').mean()
plt.figure(figsize=(12,12))
plt.bar(to_draw.keys(), to_draw.values)
plt.xticks(rotation=90)
plt.title('Average Material Intensity of concrete used per floor')
plt.ylabel('Average Material Intensity of concrete Used (KG/m2)')
plt.xlabel('Floor Code');
```



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

```
[15]: f = lambda x: name_map[re.split('[_\\.\\ ]',x)[1][0:3]] #This function takes in a full column name and returns only the Level 3 MasterFormat code.
concrete_df = df[cols].groupby(f,axis=1).sum()
```

```
[16]: concrete_df.mean().sort_values(ascending=False)
```

```
[16]: Building Identifier
Superstructure          6.844172e+06
Substructure Interior   1.743091e+06
Interior Construction    1.692972e+06
```

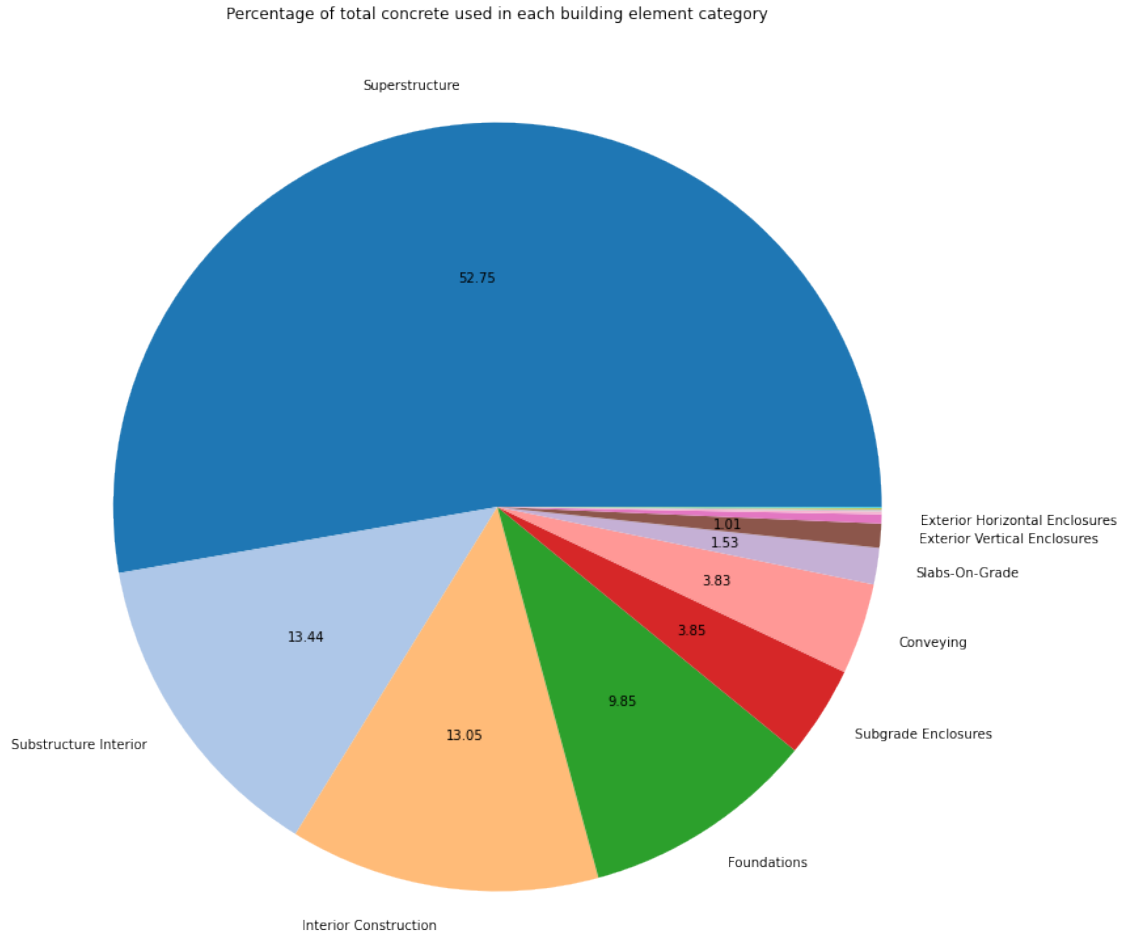
Foundations	1.277363e+06
Subgrade Enclosures	4.997662e+05
Conveying	4.970189e+05
Slabs-On-Grade	1.989609e+05
Exterior Vertical Enclosures	1.306903e+05
Exterior Horizontal Enclosures	5.030072e+04
Special Construction	1.543692e+04
Substructure Related Activities	1.208292e+04
Site Improvements	5.666442e+03
Plumbing	5.186825e+03
Water And Gas Mitigation	1.219826e+03

dtype: float64

### 3.1 Pie chart version A: on-pie chart labels for all > 1%

```
[17]: def my_autopct(pct):
        return ('%.2f' % pct) if pct > 1 else ''
to_plot = concrete_df.mean().sort_values(ascending=False)
to_plot.plot.pie(figsize=(12,12),colormap='tab20',autopct=my_autopct,labels=[k_
    ↳if v > 35000 else '' for k,v in to_plot.items()])
plt.ylabel('')
plt.title('Percentage of total concrete used in each building element_
    ↳category');
# plt.legend(loc='center left',bbox_to_anchor=(-0.20, 0.75));
plt.tight_layout();
```





### 3.2 Pie version B: external legend with slice labels

```
[18]: fig = plt.figure(figsize=(16,12))
gs = gridspec.GridSpec(2, 2, width_ratios=[3, 1])
ax0 = plt.subplot(gs[:,0])

def my_autopct(pct):
    return ('%.2f' % pct) if pct > 1 else ''
to_plot = concrete_df.mean().sort_values(ascending=False)
to_plot.plot.pie(ax=ax0,colormap='tab20',autopct=my_autopct,labeldistance=None)
plt.ylabel('')
plt.legend(loc='center left',bbox_to_anchor=(-0.20, 0.75));
plt.tight_layout();

ax1 = plt.subplot(gs[0,1])
f = lambda x: \
```

```

additional_categories_map[re.split('_\\.\\ ',x)[3]] \
if \
re.split('_\\.\\ ',x)[3] != '000' \
else \
name_map['.'].join(re.split('_\\.\\ ',x)[1:3]))

superstructure_df = df[[c for c in cols if 'B10' in c]].groupby(f,axis=1).sum()
to_plot = superstructure_df.mean().sort_values(ascending=False)
def my_autopct(pct):
    return ('%.2f' % ((pct * 0.4335))) if pct > 1 else ''
to_plot.plot.pie(ax=ax1,colormap='tab20b',autopct=my_autopct,labeldistance=None)
plt.ylabel('')
plt.legend(loc='center right',bbox_to_anchor=(1, -0.65));
plt.tight_layout();

transFigure = fig.transFigure.inverted()

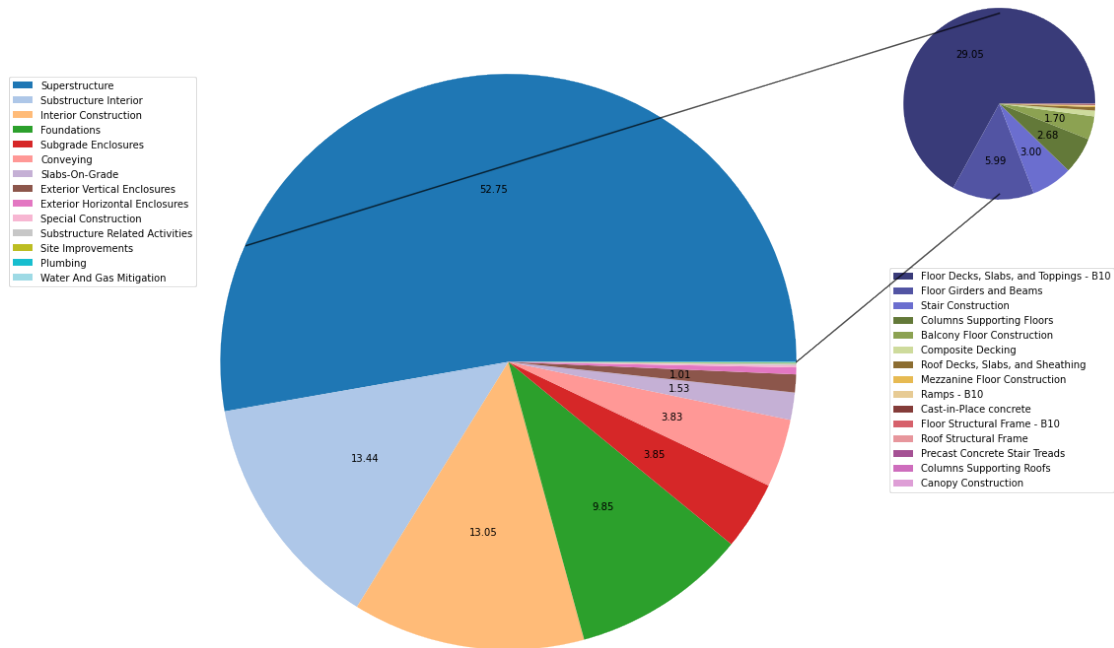
coord1a = transFigure.transform(ax0.transData.transform([1,0]))
coord2a = transFigure.transform(ax1.transData.transform([0,-0.72]))

coord1b = transFigure.transform(ax0.transData.transform([-0.91,0.35]))
coord2b = transFigure.transform(ax1.transData.transform([0,0.72]))

linea = matplotlib.lines.Line2D((coord1a[0],coord2a[0]),(coord1a[1],coord2a[1]),
                                transform=fig.transFigure,c='black',alpha=0.7)
lineb = matplotlib.lines.Line2D((coord1b[0],coord2b[0]),(coord1b[1],coord2b[1]),
                                transform=fig.transFigure,c='black',alpha=0.7)
fig.lines = linea,lineb,

plt.savefig('concrete_breakdown_pie.pdf')

```



We can produce a pie chart for a single building, also.

```
[19]: mf_codes = pd.read_csv('mf_name_conversion.csv')
```

```
[20]: tofind = [
    'Plain Steel Reinforcement Bars',
    'Reinforcement Bars',
    'Structural Steel Framing',
    'Fabric and Grid Reinforcing',
    'Metal Doors',
    'Metal Roof Panel',
    'Metal Stairs',
    'Metal Railings',
    'Steel Decking',
    'Steel Joist Framing',
    'Steel'
] #List of terms we are looking to identify in column names.

tokeep = [
    c for c in mf_codes.Title.values if any(t in c for t in tofind)
] #For each codes' corresponding in MasterFormat

steel_codes = mf_codes[mf_codes.Title.isin(tokeep)]
```

```
[21]: columns_to_keep = []
      for column in df.columns:
          if 'kg' in column:
              code = re.split('_',column)[2]
              for k,c in steel_codes.values:
                  if c in code:
                      columns_to_keep.append(column)

[22]: f = lambda x: mf_codes[mf_codes.Code == str.replace(re.split('_',x)[2], '00', '')].
      ↪split('.')[0]].values[0][0]
      steel_df = df[columns_to_keep].groupby(f,axis=1).sum()

[23]: (steel_df>0).sum(axis=1).sort_values()

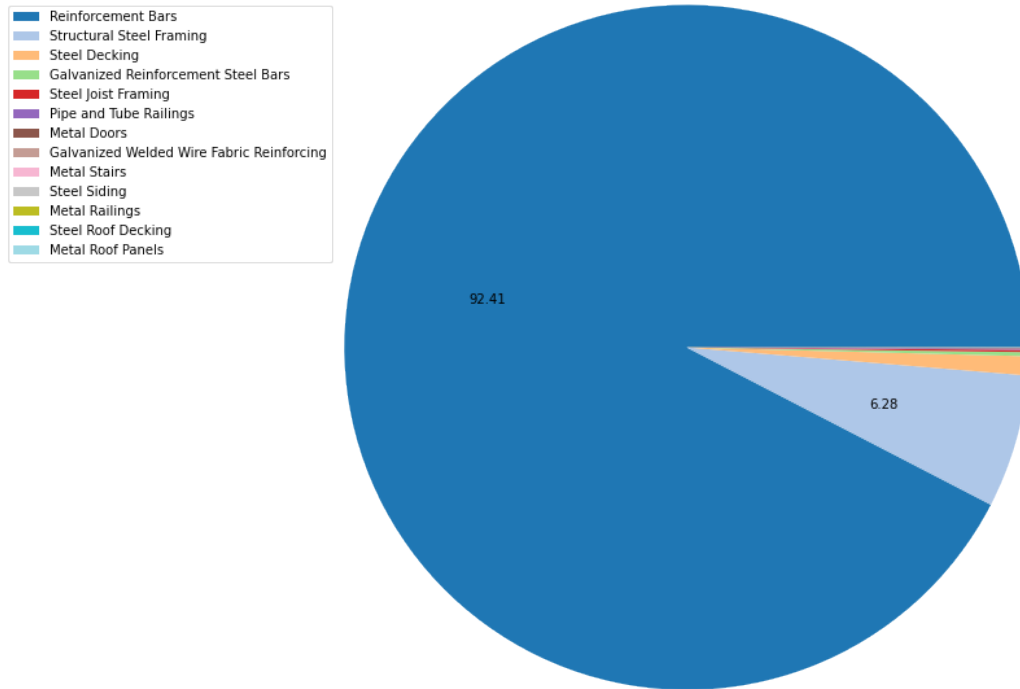
[23]: 035      1
      067      1
      066      1
      023      1
      036      1
      ..
      058      4
      049      4
      050      4
      020      4
      048      4
      Length: 70, dtype: int64

[24]: def my_autopct(pct):
      return ('%.2f' % (pct)) if pct > 1 else ''
      to_plot = steel_df.sum().sort_values(ascending=False)
      to_plot.plot.
      ↪pie(figsize=(12,12),colormap='tab20',autopct=my_autopct,labeldistance=None)
      plt.legend(loc='center left',bbox_to_anchor=(-0.30, 0.75));

      plt.ylabel('')
      plt.title(f'Types of steel use in all buildings in terms of MasterFormat_
      ↪categories');
      plt.tight_layout();

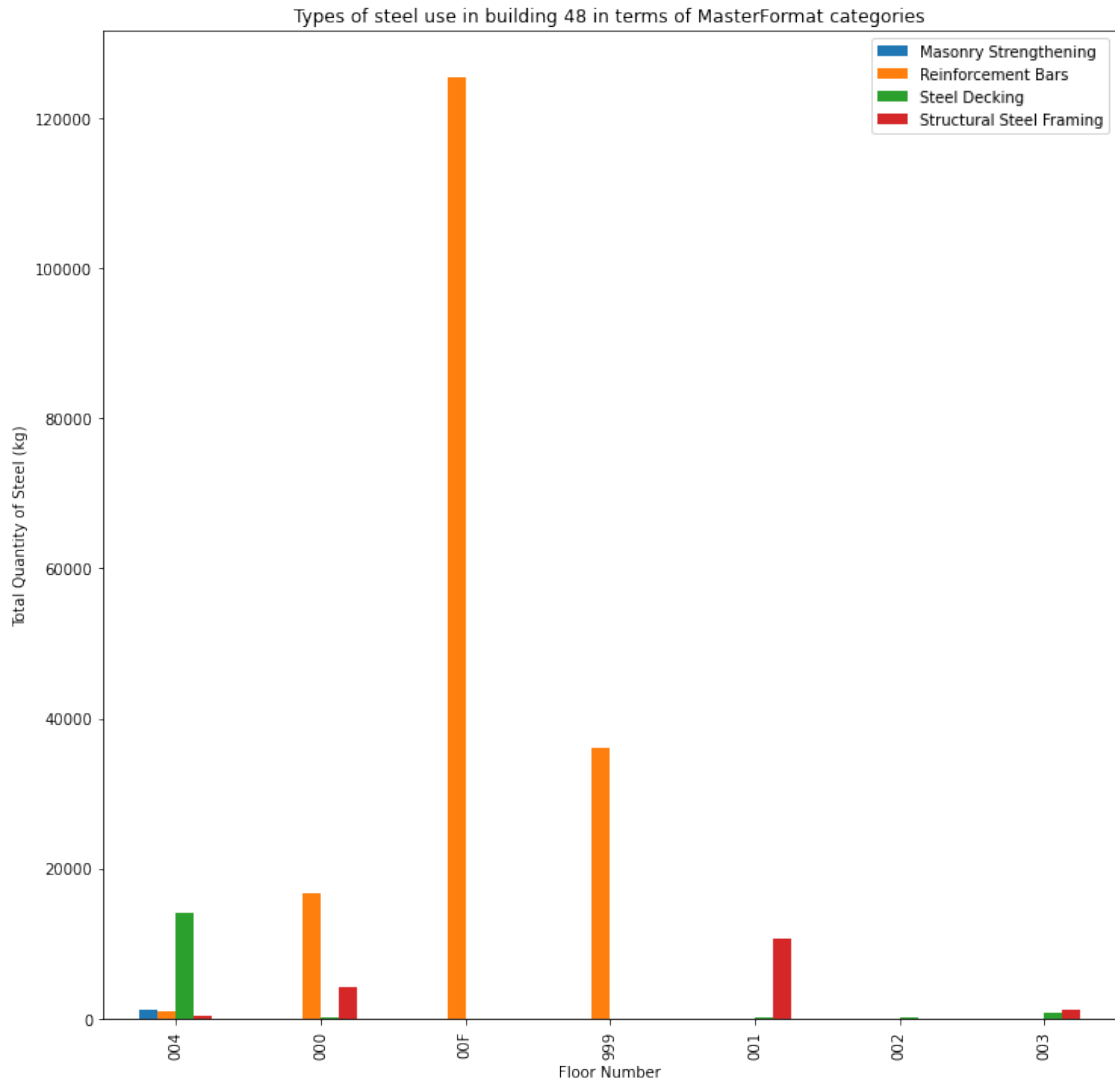
      plt.savefig('steel_composition_pie.pdf')
```

Types of steel use in all buildings in terms of MasterFormat categories



```
[25]: f = lambda x: mf_codes[mf_codes.Code.str.contains(str.replace(re.
    ↳split('_',x)[2], '00', '').split('.')[0].strip())].values[0][0] + '/' + x.
    ↳split('_')[0]
tdf = df[columns_to_keep].groupby(f,axis=1).sum().iloc[47,:]
tdf = tdf[tdf>0]
```

```
[26]: from collections import defaultdict
todf = defaultdict(dict)
for (a,b),c in zip(tdf.keys().str.split('/'),tdf.values):
    todf[a][b] = c
toplot = pd.DataFrame(todf)
toplot.plot.bar(figsize=(12,12));
plt.xlabel('Floor Number')
plt.ylabel('Total Quantity of Steel (kg)')
plt.title('Types of steel use in building 48 in terms of MasterFormat_
    ↳categories')
plt.savefig('bar_steel_onebuildingtype_byfloor.pdf')
```



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

```
[27]: concrete_df = pd.concat([df[headings[1:]], df[cols].groupby(f,axis=1).
    ↳sum()),axis=1)
concrete_df.groupby('Construction Date').mean()
```

```
[27]: Building Identifier  Gross Floor Area  Cast Decks and Underlayment/002  \
Construction Date
1913                    161.080000                    0.0
1917                    199.930000                    0.0
1969                    373.605000                    0.0
1988                   21934.000000                    0.0
2007                    73600.000000                    0.0
2009                    73083.000000                    0.0
```

2011	11282.500000	54943.2
2016	26841.666667	0.0
2017	35280.510000	0.0
2018	43365.090000	0.0
2019	107.050000	0.0
2020	10236.270000	0.0
2021	427.277895	0.0
2025	112537.000000	0.0

Building Identifier Cast Decks and Underlayment/003 \

Construction Date

1913	0.0
1917	0.0
1969	0.0
1988	0.0
2007	0.0
2009	0.0
2011	65145.6
2016	0.0
2017	0.0
2018	0.0
2019	0.0
2020	0.0
2021	0.0
2025	0.0

Building Identifier Cast Decks and Underlayment/999 \

Construction Date

1913	0.000000e+00
1917	0.000000e+00
1969	0.000000e+00
1988	0.000000e+00
2007	1.329816e+06
2009	0.000000e+00
2011	0.000000e+00
2016	0.000000e+00
2017	2.587372e+04
2018	0.000000e+00
2019	0.000000e+00
2020	0.000000e+00
2021	0.000000e+00
2025	0.000000e+00

Building Identifier Cast-in-Place Concrete/000 Cast-in-Place Concrete/001 \

Construction Date

1913	0.000000e+00	0.000000e+00
1917	0.000000e+00	0.000000e+00

1969	0.000000e+00	0.000000e+00
1988	3.999773e+06	1.435583e+06
2007	0.000000e+00	0.000000e+00
2009	0.000000e+00	0.000000e+00
2011	0.000000e+00	0.000000e+00
2016	0.000000e+00	0.000000e+00
2017	0.000000e+00	0.000000e+00
2018	0.000000e+00	0.000000e+00
2019	0.000000e+00	0.000000e+00
2020	0.000000e+00	0.000000e+00
2021	0.000000e+00	0.000000e+00
2025	0.000000e+00	0.000000e+00

Building Identifier Construction Date	Cast-in-Place Concrete/002	Cast-in-Place Concrete/003 \
1913	0.000000e+00	0.000000e+00
1917	0.000000e+00	0.000000e+00
1969	0.000000e+00	0.000000e+00
1988	1.502795e+06	1.423554e+06
2007	0.000000e+00	0.000000e+00
2009	0.000000e+00	0.000000e+00
2011	0.000000e+00	0.000000e+00
2016	0.000000e+00	0.000000e+00
2017	0.000000e+00	0.000000e+00
2018	0.000000e+00	0.000000e+00
2019	0.000000e+00	0.000000e+00
2020	0.000000e+00	0.000000e+00
2021	0.000000e+00	0.000000e+00
2025	0.000000e+00	0.000000e+00

Building Identifier Construction Date	Cast-in-Place Concrete/004	Cast-in-Place Concrete/005 \
1913	0.000000e+00	0.000000
1917	0.000000e+00	0.000000
1969	0.000000e+00	0.000000
1988	1.318964e+06	788129.689933
2007	0.000000e+00	0.000000
2009	0.000000e+00	0.000000
2011	0.000000e+00	0.000000
2016	0.000000e+00	0.000000
2017	0.000000e+00	0.000000
2018	0.000000e+00	0.000000
2019	0.000000e+00	0.000000
2020	0.000000e+00	0.000000
2021	0.000000e+00	0.000000
2025	0.000000e+00	0.000000



Building Identifier	...	Structural Concrete/999	Structural Concrete/B01	\
Construction Date	...			
1913	...	0.0	64035.190000	
1917	...	0.0	114018.460000	
1969	...	0.0	132278.015000	
1988	...	0.0	0.000000	
2007	...	0.0	0.000000	
2009	...	0.0	0.000000	
2011	...	0.0	0.000000	
2016	...	156360.0	0.000000	
2017	...	205476.0	0.000000	
2018	...	593112.0	0.000000	
2019	...	0.0	47353.684083	
2020	...	34108.8	98902.934000	
2021	...	0.0	156066.475284	
2025	...	847704.0	0.000000	

Building Identifier	Structural Concrete/M00	Structural Concrete/M10	\
Construction Date			
1913	0.0	0.0	
1917	0.0	0.0	
1969	0.0	0.0	
1988	0.0	0.0	
2007	0.0	0.0	
2009	0.0	0.0	
2011	0.0	0.0	
2016	141136.0	0.0	
2017	0.0	0.0	
2018	633824.0	0.0	
2019	0.0	0.0	
2020	65894.4	0.0	
2021	0.0	0.0	
2025	0.0	391968.0	

Building Identifier	Structural Concrete/MP1	Structural Concrete/P01	\
Construction Date			
1913	0.0	0.0	
1917	0.0	0.0	
1969	0.0	0.0	
1988	0.0	0.0	
2007	0.0	0.0	
2009	0.0	0.0	
2011	0.0	0.0	
2016	0.0	1471112.0	
2017	0.0	2764302.0	
2018	0.0	2899816.0	
2019	0.0	0.0	

2020	0.0	610046.4
2021	0.0	0.0
2025	1405272.0	7396368.0

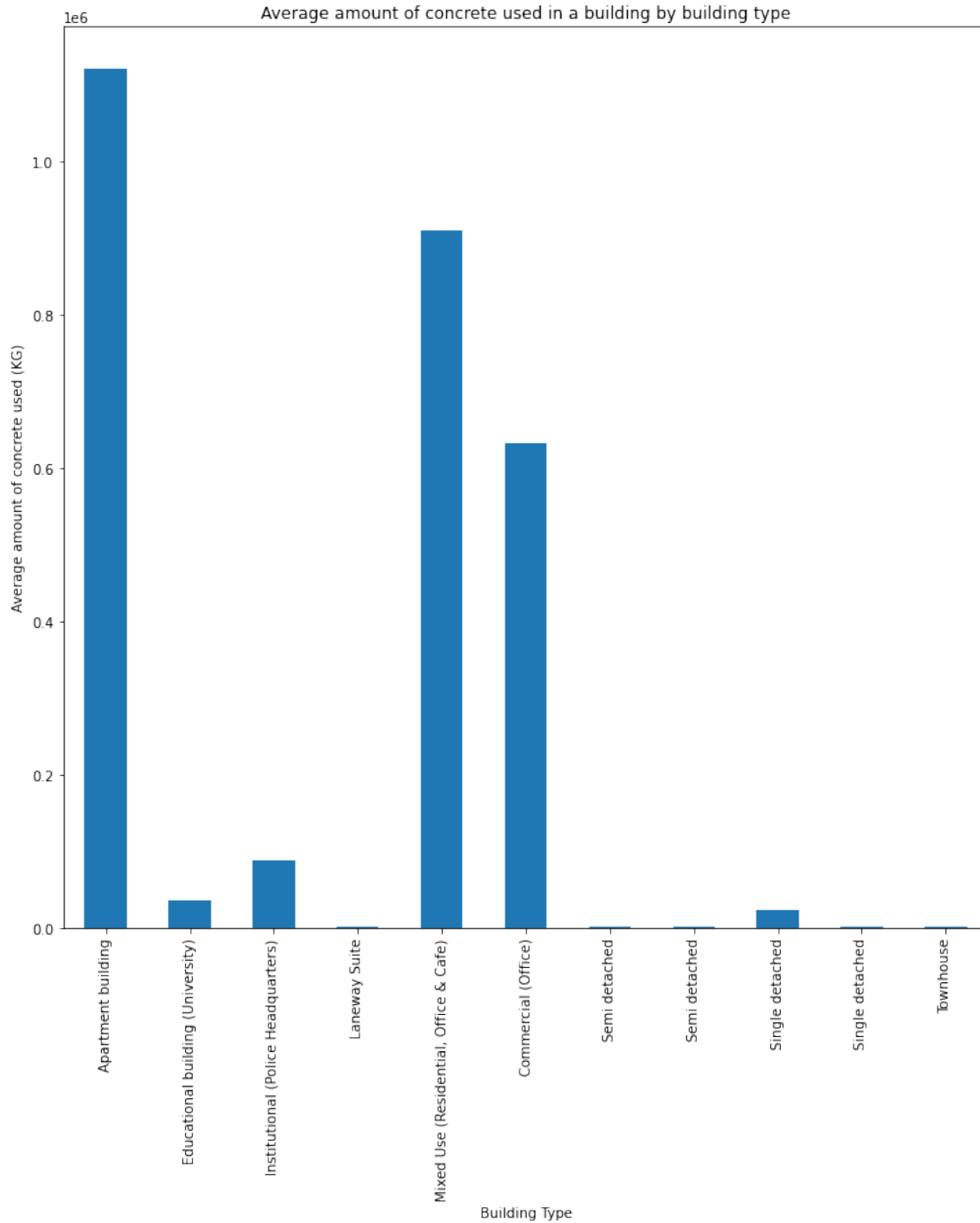
Building Identifier	Structural Concrete/P02	Structural Concrete/P03 \
Construction Date		
1913	0.0	0.0
1917	0.0	0.0
1969	0.0	0.0
1988	0.0	0.0
2007	0.0	0.0
2009	0.0	0.0
2011	0.0	0.0
2016	1143352.0	1064296.0
2017	2067108.0	2037768.0
2018	2405792.0	1837944.0
2019	0.0	0.0
2020	468100.8	466708.8
2021	0.0	0.0
2025	5522424.0	4559496.0

Building Identifier	Structural Concrete/P04	Structural Concrete/P05
Construction Date		
1913	0.0	0.0
1917	0.0	0.0
1969	0.0	0.0
1988	0.0	0.0
2007	0.0	0.0
2009	0.0	0.0
2011	0.0	0.0
2016	6087984.0	0.0
2017	1602108.0	609738.0
2018	2728856.0	0.0
2019	0.0	0.0
2020	1820392.8	0.0
2021	0.0	0.0
2025	6789888.0	0.0

[14 rows x 322 columns]

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

```
[28]: concrete_df.groupby('Building Type').sum().mean(axis=1).
      ↪ rename(index=building_name_map).plot(kind='bar',figsize=(12,12))
plt.ylabel('Average amount of concrete used (KG)')
plt.title('Average amount of concrete used in a building by building type');
```



## 4 3. Uncertainty by Building Type

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

```
[29]: 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])

[30]: from collections import Counter

[31]: for k,v in uncertainty_level.items():
      uncertainty_level[k] = Counter(v) #Construct a Counter object per building_
      ↪type

[32]: uncertainty_level

[32]: {'SND': Counter({'3': 626, '2': 1582, '5': 284}),
      'OFF': Counter({'2': 491, '4': 307}),
      'APB': Counter({'2': 1844, '3': 1, '4': 1601}),
      'SMR': Counter({'2': 20, '3': 26, '5': 8}),
      'SNR': Counter({'2': 55, '3': 70, '5': 52}),
      'SMD': Counter({'2': 167, '3': 34, '5': 19}),
      'EDU': Counter({'2': 91, '4': 24, '3': 6}),
      'INS': Counter({'4': 77, '2': 89, '3': 1}),
      'MIX': Counter({'2': 1262, '4': 1047}),
      'LNW': Counter({'3': 92, '2': 287, '5': 21}),
      'TWN': Counter({'2': 58, '4': 6})}
```

Next, we aggregate columns by the purpose of the material and uncertainty combined, and report the average by building type.

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

[34]: by_function_df.groupby('Building Type').mean().rename(index=building_name_map).
      ↪drop(['Construction Date'],axis=1).round(2)
```

```
[34]: Building Identifier      Gross Floor Area  Interiors/2 \
      Building Type
      Apartment building      39160.26      5624203.35
      Educational building (University)      7901.00      480382.15
      Institutional (Police Headquarters)      21934.00      1295281.75
      Laneway Suite      128.88      0.00
```

Mixed Use (Residential, Office & Cafe)	80760.42	12716484.57
Commercial (Office)	52643.67	9898215.44
Semi detached	248.84	0.00
Semi detached	199.93	0.00
Single detached	478.40	0.00
Single detached	302.76	0.00
Townhouse	3566.00	0.00

Building Identifier	Interiors/3	Interiors/4	Services/2	\
Building Type				
Apartment building	0.00	171337.00	1529274.0	
Educational building (University)	3096.66	14080.27	0.0	
Institutional (Police Headquarters)	0.00	40860.46	0.0	
Laneway Suite	0.00	0.00	0.0	
Mixed Use (Residential, Office & Cafe)	0.00	370412.46	7268736.0	
Commercial (Office)	0.00	285637.96	0.0	
Semi detached	0.00	0.00	0.0	
Semi detached	0.00	0.00	0.0	
Single detached	0.00	0.00	0.0	
Single detached	68.77	0.00	0.0	
Townhouse	0.00	0.00	0.0	

Building Identifier	Services/4	Shell/2	Shell/3	\
Building Type				
Apartment building	50074.69	20886862.84	0.00	
Educational building (University)	0.00	1520252.59	834695.64	
Institutional (Police Headquarters)	0.00	17371405.92	0.00	
Laneway Suite	0.00	0.00	0.00	
Mixed Use (Residential, Office & Cafe)	237801.46	51743951.21	0.00	
Commercial (Office)	0.00	43308969.36	0.00	
Semi detached	0.00	1866.95	5.41	
Semi detached	0.00	0.00	40.11	
Single detached	0.00	1549.49	22.18	
Single detached	0.00	2504.95	6.65	
Townhouse	0.00	0.00	0.00	

Building Identifier	Shell/4	Shell/5	Sitework/2	\
Building Type				
Apartment building	761128.06	0.00	14493.0	
Educational building (University)	7713.03	0.00	0.0	
Institutional (Police Headquarters)	656655.11	0.00	0.0	
Laneway Suite	0.00	0.00	0.0	
Mixed Use (Residential, Office & Cafe)	1909970.03	0.00	0.0	
Commercial (Office)	1621345.80	0.00	89288.0	
Semi detached	0.00	0.00	0.0	
Semi detached	0.00	0.00	0.0	
Single detached	0.00	0.93	0.0	

Single detached	0.00	0.00	0.0
Townhouse	0.00	0.00	0.0

Building Identifier	Sitework/4 \
Building Type	
Apartment building	474.04
Educational building (University)	0.00
Institutional (Police Headquarters)	0.00
Laneway Suite	0.00
Mixed Use (Residential, Office & Cafe)	0.00
Commercial (Office)	3016.86
Semi detached	0.00
Semi detached	0.00
Single detached	0.00
Single detached	0.00
Townhouse	0.00

Building Identifier	Special Construction And Demolition/2 \
Building Type	
Apartment building	37698.0
Educational building (University)	0.0
Institutional (Police Headquarters)	0.0
Laneway Suite	0.0
Mixed Use (Residential, Office & Cafe)	249760.0
Commercial (Office)	0.0
Semi detached	0.0
Semi detached	0.0
Single detached	0.0
Single detached	0.0
Townhouse	0.0

Building Identifier	Special Construction And Demolition/4 \
Building Type	
Apartment building	339.88
Educational building (University)	0.00
Institutional (Police Headquarters)	0.00
Laneway Suite	0.00
Mixed Use (Residential, Office & Cafe)	9000.47
Commercial (Office)	0.00
Semi detached	0.00
Semi detached	0.00
Single detached	0.00
Single detached	0.00
Townhouse	0.00

Building Identifier	Substructure/2 Substructure/3 \
Building Type	

Apartment building	15661850.24	109212.00
Educational building (University)	2793438.68	0.00
Institutional (Police Headquarters)	8890567.75	0.00
Laneway Suite	48858.34	2104.66
Mixed Use (Residential, Office & Cafe)	22801051.57	0.00
Commercial (Office)	12411535.27	0.00
Semi detached	97751.05	7.78
Semi detached	110261.75	8921.68
Single detached	181911.50	5413.20
Single detached	93196.84	19429.34
Townhouse	534318.99	0.00

Building Identifier	Substructure/4	Substructure/5
Building Type		
Apartment building	365922.73	0.00
Educational building (University)	91853.12	0.00
Institutional (Police Headquarters)	239579.15	0.00
Laneway Suite	0.00	0.65
Mixed Use (Residential, Office & Cafe)	645320.97	0.00
Commercial (Office)	354767.84	0.00
Semi detached	0.00	6.93
Semi detached	0.00	0.00
Single detached	0.00	38.46
Single detached	0.00	0.00
Townhouse	14428.09	0.00

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

```
[35]: f = lambda x: x.split('_')[-1].split('.')[0] #Select only the uncertainty score.
print('Average amount of material used per building, by year and uncertainty_
→score (%)')
result = pd.concat([df['Construction Date'],df[[c for c in df.columns if 'kg'_
→in c]].groupby(f,axis=1).sum()],axis=1).groupby('Construction Date').mean()
for k,v in result.iterrows():
    result.loc[k,:] = v/v.sum()
display(result.round(2))
```

Average amount of material used per building, by year and uncertainty score (%)

	2	3	4	5
Construction Date				
1913	0.85	0.08	0.00	0.07
1917	0.75	0.14	0.00	0.11
1969	0.50	0.37	0.00	0.13
1988	0.97	0.00	0.03	0.00
2007	0.97	0.00	0.03	0.00
2009	0.97	0.00	0.03	0.00

2011	0.94	0.03	0.03	0.00
2016	0.96	0.02	0.03	0.00
2017	0.97	0.00	0.03	0.00
2018	0.97	0.00	0.03	0.00
2019	0.98	0.02	0.00	0.00
2020	0.97	0.00	0.03	0.00
2021	0.78	0.09	0.00	0.13
2025	0.97	0.00	0.03	0.00

## 5 4. Material Intensity

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

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

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

```
[37]: Building Identifier City Quality / Stage of Data Construction Date \
001 TOR 00IFC 2021
002 TOR 00IFC 2021
003 TOR 00IFC 2021
004 TOR 00IFC 2021
007 TOR 00IFC 2021
008 TOR 00IFC 2021
009 TOR 00IFC 2021
010 TOR 00IFC 2021
013 TOR 00IFC 2021
014 TOR 00IFC 2021
015 TOR 00IFC 2021
016 TOR 00IFC 2021
019 TOR 00IFC 2021
020 TOR 00IFC 2021
021 TOR 00IFC 2020
022 TOR 00IFC 2021
023 TOR 00IFC 2021
025 TOR 00IFC 2021
026 TOR 00IFC 2021
028 TOR 00IFC 2021
029 TOR 00IFC 2021
031 TOR 00IFC 2021
032 TOR 00IFC 2021
```



033	TOR	00IFC	2020
035	TOR	00IFC	2021
036	TOR	00IFC	2021
037	TOR	00IFC	2021
038	TOR	00IFC	2020
039	TOR	00IFC	2021
041	TOR	00IFC	2021
043	TOR	00IFC	2021
044	TOR	00IFC	2021
045	TOR	00IFC	2021
046	TOR	00IFC	2021
047	TOR	00IFC	2021
049	TOR	00IFC	2020
050	TOR	00IFC	2021

Building Identifier	Building Type	Gross Floor Area	Conveying \
001	SND	521.18	0.0
002	SND	389.24	0.0
003	SND	411.64	0.0
004	SND	269.56	0.0
007	SND	445.99	0.0
008	SND	438.45	0.0
009	SND	714.07	0.0
010	SND	343.24	0.0
013	SND	226.89	0.0
014	SND	611.73	0.0
015	SND	343.44	0.0
016	SND	613.38	0.0
019	SND	178.38	0.0
020	SND	323.80	0.0
021	SND	837.56	0.0
022	SND	587.86	0.0
023	SND	568.21	0.0
025	SND	294.84	0.0
026	SND	496.77	0.0
028	SND	643.30	0.0
029	SND	701.61	0.0
031	SND	378.70	0.0
032	SND	324.16	0.0
033	SND	533.53	0.0
035	SND	423.03	0.0
036	SND	328.16	0.0
037	SND	421.59	0.0
038	SND	628.59	0.0
039	SND	464.51	0.0
041	SND	346.14	0.0
043	SND	891.97	0.0

044	SND	525.61	0.0
045	SND	502.87	0.0
046	SND	379.18	0.0
047	SND	549.65	0.0
049	SND	393.82	0.0
050	SND	648.14	0.0

Building Identifier Exterior Horizontal Enclosures \

001	11.137992
002	5.461939
003	3.786074
004	6.503479
007	11.933511
008	12.707195
009	12.865930
010	4.300619
013	12.424245
014	5.140200
015	6.494467
016	13.090524
019	9.782438
020	9.824569
021	13.521848
022	6.949783
023	12.754287
025	3.650542
026	5.352985
028	11.769043
029	11.799093
031	5.522739
032	5.361174
033	8.494907
035	11.102019
036	10.234937
037	12.223172
038	10.408758
039	4.118745
041	11.787081
043	10.710312
044	18.918490
045	6.014586
046	6.169302
047	11.310711
049	16.116861
050	9.684756

Building Identifier Exterior Vertical Enclosures Foundations \

001	136.939623	335.649367
002	69.018253	281.318698
003	101.450370	464.462195
004	188.215196	255.359136
007	61.325975	295.116668
008	130.552921	269.468463
009	104.310510	276.917123
010	210.632241	283.893850
013	186.668275	261.874926
014	102.332008	343.714248
015	147.104280	424.099610
016	156.986570	298.537712
019	112.523711	371.149916
020	186.570501	148.769711
021	91.689386	317.583491
022	94.557055	428.185321
023	83.789887	255.012975
025	127.856507	261.274626
026	89.883144	251.725837
028	83.949693	156.365248
029	53.418023	266.164355
031	164.214896	403.602589
032	190.512918	377.853541
033	68.518430	309.062696
035	154.072547	243.607664
036	184.202156	388.744353
037	158.716507	424.443503
038	136.076590	369.744859
039	151.068033	412.845205
041	146.479339	287.564257
043	213.677214	245.205806
044	109.529933	498.010299
045	91.481074	278.679758
046	172.418003	391.303861
047	127.866168	266.468237
049	140.069509	188.980245
050	131.118584	347.187490

Building Identifier	Interior Construction	Interior Finishes	Plumbing \
001	16.482129	6.202080	0.0
002	12.248343	4.491260	0.0
003	15.931829	3.030369	0.0
004	4.574132	2.920482	0.0
007	19.773909	4.539900	0.0
008	10.683759	4.767511	0.0
009	18.937583	4.898301	0.0
010	17.891930	6.753884	0.0

013	17.256393	4.154604	0.0
014	13.258982	5.577869	0.0
015	18.195449	5.729880	0.0
016	17.589067	5.763898	0.0
019	19.638502	7.549843	0.0
020	18.186467	3.384055	0.0
021	17.799752	5.017694	0.0
022	19.088554	4.710543	0.0
023	23.268519	5.714419	0.0
025	20.047035	3.601363	0.0
026	14.370613	4.321980	0.0
028	16.010229	5.765195	0.0
029	23.078653	5.728781	0.0
031	19.181898	7.221059	0.0
032	24.166732	4.906090	0.0
033	34.027695	4.971297	0.0
035	16.390809	3.227528	0.0
036	7.854953	1.765491	0.0
037	16.125050	3.247311	0.0
038	16.271010	4.180593	0.0
039	15.108900	5.465049	0.0
041	19.523228	5.764737	0.0
043	20.691791	5.194042	0.0
044	19.155639	5.835201	0.0
045	22.485115	2.978621	0.0
046	16.651076	4.323340	0.0
047	20.753973	4.819176	0.0
049	22.332639	7.801305	0.0
050	23.995586	3.705203	0.0

Building Identifier	Site Improvements	Slabs-On-Grade	Special Construction \
001	0.0	273.972401	0.0
002	0.0	192.874465	0.0
003	0.0	170.733356	0.0
004	0.0	124.186526	0.0
007	0.0	153.061618	0.0
008	0.0	211.910108	0.0
009	0.0	266.709576	0.0
010	0.0	138.510228	0.0
013	0.0	129.263543	0.0
014	0.0	165.513154	0.0
015	0.0	129.532248	0.0
016	0.0	166.414337	0.0
019	0.0	223.398638	0.0
020	0.0	158.178114	0.0
021	0.0	143.282268	0.0
022	0.0	237.918968	0.0

023	0.0	199.364347	0.0
025	0.0	131.174185	0.0
026	0.0	242.284758	0.0
028	0.0	152.407914	0.0
029	0.0	169.419640	0.0
031	0.0	179.868896	0.0
032	0.0	132.696247	0.0
033	0.0	135.390288	0.0
035	0.0	147.458950	0.0
036	0.0	128.887840	0.0
037	0.0	147.225241	0.0
038	0.0	186.334547	0.0
039	0.0	145.273403	0.0
041	0.0	139.821081	0.0
043	0.0	138.994603	0.0
044	0.0	139.646277	0.0
045	0.0	182.059329	0.0
046	0.0	158.446049	0.0
047	0.0	154.805714	0.0
049	0.0	198.860705	0.0
050	0.0	199.209464	0.0

Building Identifier	Subgrade Enclosures	Substructure Interior \
001	9.652903	7.521547
002	6.851955	11.871041
003	11.298572	8.277288
004	4.351465	20.070275
007	9.478642	5.575509
008	4.218921	1.817270
009	8.902623	25.192687
010	9.601245	7.744759
013	3.818403	9.532825
014	7.722754	6.168162
015	9.135529	5.601240
016	4.868508	9.004152
019	0.000000	8.758309
020	4.617006	11.946436
021	7.131170	8.875410
022	7.959752	9.098153
023	6.339651	11.209887
025	7.469048	3.895085
026	9.448689	4.154656
028	0.000000	11.506782
029	11.919460	8.789598
031	7.509119	10.575300
032	5.073992	8.309600
033	8.867868	13.435344

035	0.000000	10.013415
036	4.762839	19.086997
037	9.538939	12.833857
038	6.039206	7.143042
039	9.071017	12.485838
041	7.568785	12.011677
043	4.540919	10.725241
044	6.720435	8.275280
045	6.092739	10.878686
046	9.489156	13.750663
047	6.042229	8.345960
049	6.057127	5.861907
050	7.221222	8.240307

Building Identifier	Substructure Related Activities	Superstructure \
001	0.0	30.228003
002	0.0	26.271523
003	0.0	23.756286
004	0.0	30.396721
007	0.0	39.906513
008	0.0	39.907474
009	0.0	38.291591
010	0.0	35.370538
013	0.0	35.355314
014	0.0	33.388004
015	0.0	39.370016
016	0.0	40.958564
019	0.0	63.006044
020	0.0	36.597047
021	0.0	28.734226
022	0.0	37.457583
023	0.0	36.265538
025	0.0	30.389475
026	0.0	43.728928
028	0.0	35.393414
029	0.0	39.408113
031	0.0	82.392236
032	0.0	46.380703
033	0.0	25.469871
035	0.0	35.666107
036	0.0	49.284461
037	0.0	34.035382
038	0.0	47.065025
039	0.0	37.921434
041	0.0	27.740220
043	0.0	29.045531
044	0.0	33.265489

045	0.0	37.265275
046	0.0	46.860447
047	0.0	31.152827
049	0.0	49.899420
050	0.0	38.021046

Building Identifier	Water And Gas Mitigation
---------------------	--------------------------

001	0.0
002	0.0
003	0.0
004	0.0
007	0.0
008	0.0
009	0.0
010	0.0
013	0.0
014	0.0
015	0.0
016	0.0
019	0.0
020	0.0
021	0.0
022	0.0
023	0.0
025	0.0
026	0.0
028	0.0
029	0.0
031	0.0
032	0.0
033	0.0
035	0.0
036	0.0
037	0.0
038	0.0
039	0.0
041	0.0
043	0.0
044	0.0
045	0.0
046	0.0
047	0.0
049	0.0
050	0.0

```
[38]: master_format_convert = {v:k for k,v in {
      'Concrete':'03',
```

```

'Masonry':'04',
'Metals':'05',
'WoodPlasticsAndComposites':'06',
'ThermalAndMoistureProtection':'07',
'Finishes':'09',
'Openings':'08',
'Earthwork':'31',
'ExteriorImprovements':'32'
}.items() }

```

```

[39]: f = lambda x: master_format_convert[re.split('_\.\ ',x)[4]]
      toplot = pd.concat([df[headings[1:]],df_mi[kilogram_columns].groupby(f,axis=1).
      ↪sum()),axis=1).sort_values(['Building Type'])

```

```

[40]: building_type_map = dict(building_name_conversion[['Building Code','Type']].
      ↪values)

      toplot['Building Type'] = toplot['Building Type'].replace(building_type_map)
      toplot = toplot.sort_values('Building Type')

```

```

[41]: set(df['Building Type'].values)

```

```

[41]: {'APB', 'EDU', 'INS', 'LNW', 'MIX', 'OFF', 'SMD', 'SMR', 'SND', 'SNR', 'TWN'}

```

```

[42]: fig, ax = plt.subplots(figsize=(10,7))

      cols = toplot.columns[5:]
      margin_bottom = np.zeros(len(toplot))

      cmap = plt.get_cmap('tab10')

      for num, col in enumerate(cols):
          values = toplot[col].values

          toplot[col].plot.bar(x='Year',y='Value', ax=ax, stacked=True,
                              bottom = margin_bottom, color=cmap(num),
          ↪label=col)
          margin_bottom += values
      plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
      plt.ylabel('Material Intensity (kg/m^2)')
      plt.xlabel('Building ID ')
      ax2 = ax.twinx()
      ax2.set_xlim(0, len(toplot))
      ax2.set_xticks([k for k,v in enumerate(toplot['Building Type'].values) if v !=
      ↪toplot['Building Type'].values[k-1] or k==0])
      for tick in ax2.get_xticklabels():
          tick.set_rotation(90)

```

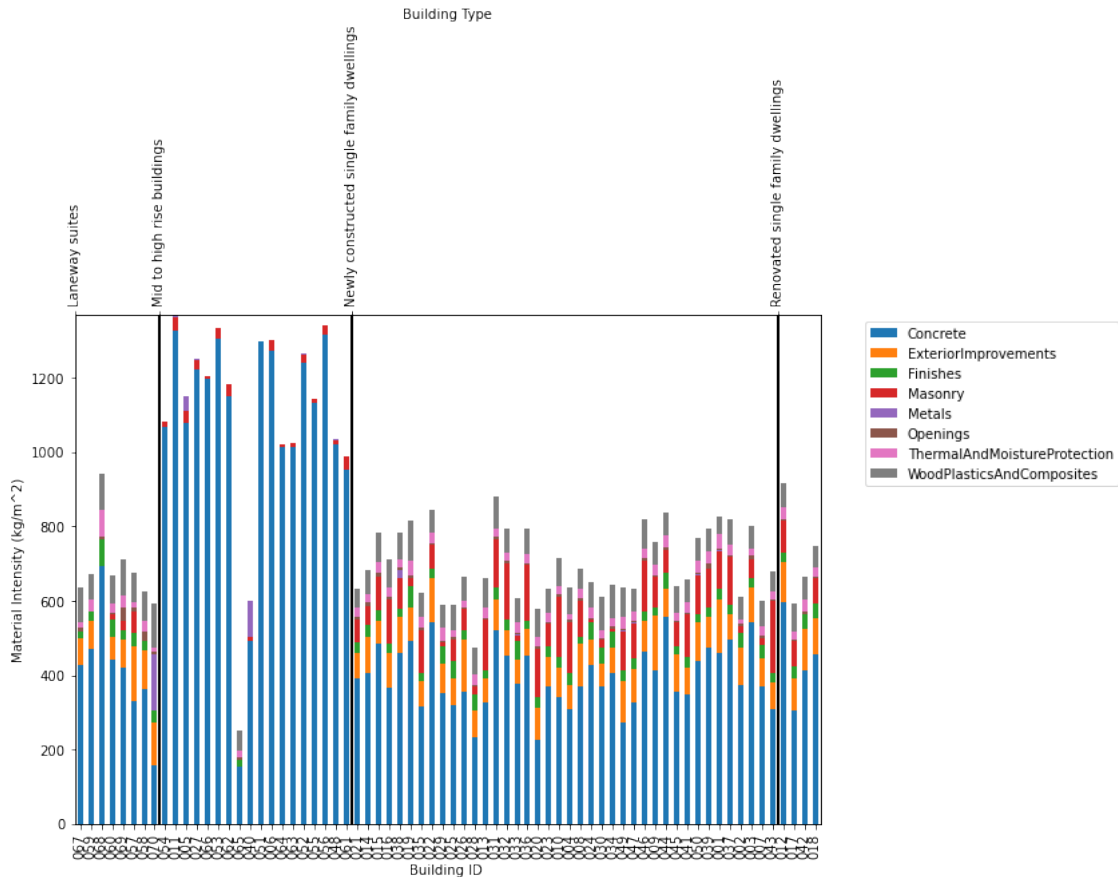


```

ax2.set_xticklabels([v for k,v in enumerate(toplot['Building Type'].values) if
↳v != toplot['Building Type'].values[k-1] or k==0])
ax2.set_xlabel("Building Type")
plt.grid(color='black',linewidth=2)

plt.show()

```



```
[43]: toplot['Total MI'] = toplot.iloc[:,5:].sum(axis=1)
```

```

[44]: print('Mean Material Intensity:')
display(toplot.groupby('Building Type').mean().iloc[:,1:].round(2))
print('Std Dev Material Intensity:')
display(toplot.groupby('Building Type').std().iloc[:,1:].round(2))

```

Mean Material Intensity:

Building Identifier	Gross Floor Area	Concrete \
Building Type		
Laneway suites	128.88	412.96
Mid to high rise buildings	41933.13	1070.18

Newly constructed single family dwellings	461.18	396.71
Renovated single family dwellings	277.06	442.97

Building Identifier	ExteriorImprovements	Finishes	\
Building Type			
Laneway suites	81.80	35.26	
Mid to high rise buildings	0.00	0.99	
Newly constructed single family dwellings	86.16	31.17	
Renovated single family dwellings	100.30	33.64	

Building Identifier	Masonry	Metals	Openings	\
Building Type				
Laneway suites	11.68	18.65	12.42	
Mid to high rise buildings	18.83	8.57	0.39	
Newly constructed single family dwellings	83.77	0.96	5.99	
Renovated single family dwellings	55.31	0.74	5.84	

Building Identifier	ThermalAndMoistureProtection	\
Building Type		
Laneway suites	28.81	
Mid to high rise buildings	0.97	
Newly constructed single family dwellings	25.63	
Renovated single family dwellings	26.98	

Building Identifier	WoodPlasticsAndComposites	Total MI
Building Type		
Laneway suites	89.25	690.83
Mid to high rise buildings	3.02	1102.96
Newly constructed single family dwellings	68.82	699.22
Renovated single family dwellings	64.59	730.36

Std Dev Material Intensity:

Building Identifier	Gross Floor Area	Concrete	\
Building Type			
Laneway suites	50.64	150.56	
Mid to high rise buildings	32439.52	302.20	
Newly constructed single family dwellings	168.17	82.14	
Renovated single family dwellings	117.28	120.26	

Building Identifier	ExteriorImprovements	Finishes	\
Building Type			
Laneway suites	43.53	17.14	
Mid to high rise buildings	0.00	4.22	
Newly constructed single family dwellings	22.30	9.40	
Renovated single family dwellings	12.94	6.38	

Building Identifier	Masonry	Metals	Openings	\
Building Type				

Laneway suites	20.51	52.54	11.58
Mid to high rise buildings	11.42	23.39	1.66
Newly constructed single family dwellings	49.26	3.35	2.21
Renovated single family dwellings	37.88	0.86	1.43

Building Identifier	ThermalAndMoistureProtection \
Building Type	
Laneway suites	19.63
Mid to high rise buildings	4.13
Newly constructed single family dwellings	6.14
Renovated single family dwellings	5.44

Building Identifier	WoodPlasticsAndComposites	Total MI
Building Type		
Laneway suites	17.69	107.85
Mid to high rise buildings	12.82	280.73
Newly constructed single family dwellings	11.58	95.96
Renovated single family dwellings	6.55	140.02

```
[45]: df_mi = df[kilogram_columns].div(df['Gross Floor Area'],axis=0)
```

```
[46]: df_mi = df[kilogram_columns].div(df['Gross Floor Area'],axis=0)
df_mi = df_mi.div(df_mi.sum(axis=1),axis=0) * 100
f = lambda x: name_map[re.split('_\.\. ',x)[1][0]]
toplot = pd.concat([df[headings[1:]],df_mi[kilogram_columns].groupby(f,axis=1).
    ↳sum(),axis=1).sort_values('Building Type')
toplot['Building Type'] = toplot['Building Type'].replace(building_type_map)
toplot = toplot.sort_values('Building Type')
fig, ax = plt.subplots(figsize=(10,7))

cols = toplot.columns[5:]
margin_bottom = np.zeros(len(toplot))

cmap = plt.get_cmap('tab10')

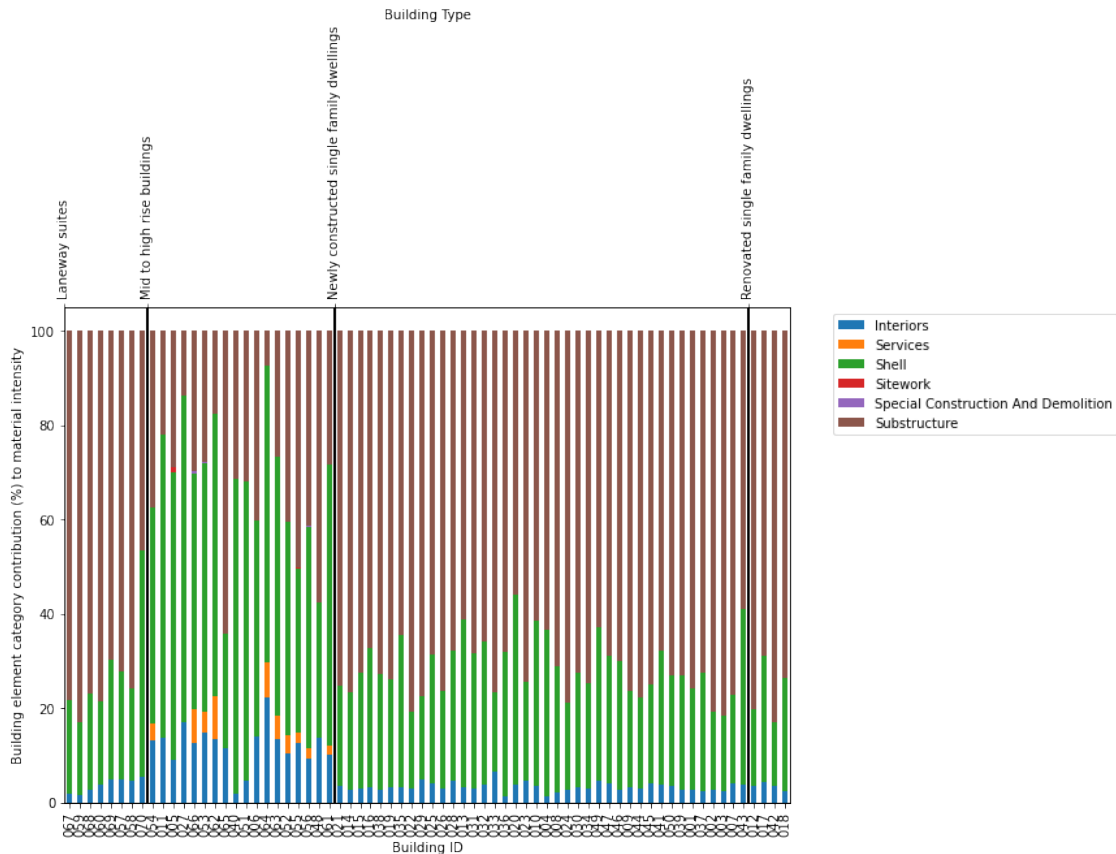
for num, col in enumerate(cols):
    values = toplot[col].values

    toplot[col].plot.bar(x='Year',y='Value', ax=ax, stacked=True,
        bottom = margin_bottom, color=cmap(num),
    ↳label=col)
    margin_bottom += values
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.xlabel('Building ID')
plt.ylabel('Building element category contribution (%) to material intensity')
ax2 = ax.twinx()
```

```

ax2.set_xlim(0, len(toplot))
ax2.set_xticks([k for k,v in enumerate(toplot['Building Type'].values) if v != '']
↳ toplot['Building Type'].values[k-1] or k==0])
for tick in ax2.get_xticklabels():
    tick.set_rotation(90)
ax2.set_xticklabels([v for k,v in enumerate(toplot['Building Type'].values) if v
↳ v != toplot['Building Type'].values[k-1] or k==0])
ax2.set_xlabel("Building Type")
plt.grid(color='black',linewidth=2)
plt.show()

```



```

[47]: f = lambda x: name_map[re.split('_\.\\ ',x)[1][0]] + '/' + re.split('_\.\\ 
↳ ',x)[-1]
toplot = df_mi[kilogram_columns].groupby(f,axis=1).sum()

```

```

[48]: df_mi = df[kilogram_columns].div(df['Gross Floor Area'],axis=0)
df_mi = df_mi.div(df_mi.sum(axis=1),axis=0)
f = lambda x: name_map[re.split('_\.\\ ',x)[1][0]] + '/' + re.split('_\.\\ 
↳ ',x)[-1]
toplot = df_mi[kilogram_columns].groupby(f,axis=1).sum()

```

```

for i in range(1,6):
    toplot[f'Total/{i}'] = 0
for k,v in toplot.iteritems():
    toplot[f'Total/{k.split("/") [1]}'] += v
toplot_out = deepcopy(toplot)
for k,v in toplot.iteritems():
    toplot_out[k] = (v/toplot[[c for c in toplot.columns if k.split('/')[0] in_
    ↳c]].sum(axis=1)) * int(k.split('/')[1])
f = lambda x: x.split('/')[0]
toplot_out = pd.concat([df['Building Type'], toplot_out.groupby(f,axis=1).
    ↳sum()],axis=1).sort_values('Building Type')
toplot_out = toplot_out.reset_index()
toplot_out['index'] = toplot_out['index'].astype('int') + 1
toplot_out['index'] = toplot_out['index'].astype('str')

```

```

[49]: # toplot_out = toplot_out[toplot_out['Building Type'].isin(types_to_keep)]
toplot_out['Building Type'] = toplot_out['Building Type'].
    ↳replace(building_type_map)
toplot_out = toplot_out.sort_values('Building Type')

```

```

[50]: from matplotlib.lines import Line2D
fig, ax = plt.subplots(figsize=(7,15))
ax.set_xlim(1,5)
ax.set_ylim(0,len(toplot_out))
# ax.set_yticks(toplot_out['index'])
handles = []
for v,m,c in_
    ↳[(('Interiors','o','blue'),('Shell','X','green'),('Total','*','red'),('Substructure','s','or
    ↳
        ax.scatter(x=toplot_out[v].values,y=toplot_out['index'].values, marker=m,
        ↳color=c, s=75)
        handles.append(
            Line2D([0], [0], marker=m, color='w', label=v,
                markerfacecolor=c, markersize=15)
        )
plt.legend(handles=handles,bbox_to_anchor=(1.05, 0.5), loc='lower left')
plt.ylabel('Building Identifier')
plt.xlabel('Weighted Uncertainty Level')
plt.grid()
ax2 = ax.twinx()
ax2.set_ylim(0, len(toplot_out))
ax2.set_yticks([k-1.5 for k,v in enumerate(toplot_out['Building Type'].values)
    ↳if v != toplot_out['Building Type'].values[k-1] or k==0])
# for tick in ax2.get_yticklabels():
#     tick.set_rotation(90)

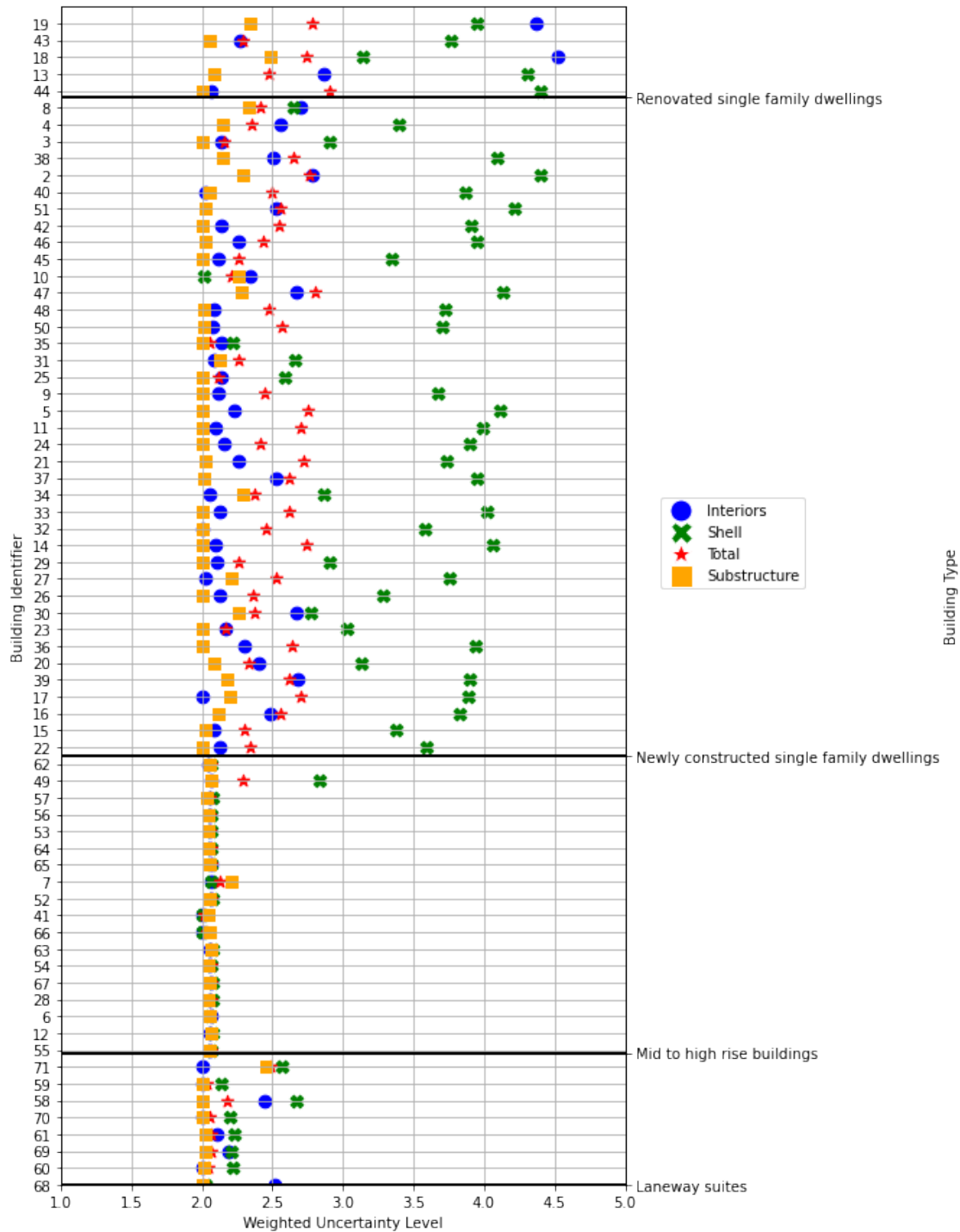
```

```

ax2.set_yticklabels([v for k,v in enumerate(toplot_out['Building Type'].values)
↳if v != toplot_out['Building Type'].values[k-1] or k==0])
ax2.set_ylabel("Building Type")

plt.grid(color='black',linewidth=2)

```



```
[51]: toplot_out
```

```
[51]:
```

	index	Building Type	Interiors	Services	\
18	68	Laneway suites	2.520976	0.0	
17	60	Laneway suites	2.000000	0.0	
16	69	Laneway suites	2.190950	0.0	
15	61	Laneway suites	2.106514	0.0	
11	70	Laneway suites	2.005075	0.0	
..	...	...	...	...	
34	44	Newly constructed single family dwellings	2.065774	0.0	
28	13	Renovated single family dwellings	2.868511	0.0	
66	18	Renovated single family dwellings	4.523878	0.0	
67	43	Renovated single family dwellings	2.275307	0.0	
68	19	Renovated single family dwellings	4.371953	0.0	
		Shell	Sitework	Special Construction And Demolition	Substructure \
18	2.025651	0.0		0.0	2.000000
17	2.222478	0.0		0.0	2.009786
16	2.206190	0.0		0.0	2.023037
15	2.229146	0.0		0.0	2.024653
11	2.200763	0.0		0.0	2.000000
..	...	...		...	...
34	4.396133	0.0		0.0	2.005594
28	4.306551	0.0		0.0	2.082720
66	3.139931	0.0		0.0	2.480406
67	3.763229	0.0		0.0	2.056058
68	3.946027	0.0		0.0	2.342662
		Total			
18	2.015289				
17	2.042591				
16	2.064759				
15	2.063625				
11	2.051110				
..	...				
34	2.900427				
28	2.475825				
66	2.744405				
67	2.294809				
68	2.777552				

```
[70 rows x 9 columns]
```

## 6 Additional Characteristics

### 6.1 1. Count number of floors in a given building based on position relative to the ground.

```
[52]: from collections import Counter
import re
import eeweather
BUILDING_ID = '043' #As an example, select building 043
building_data = df.loc[BUILDING_ID]
```

```
[53]: seen = set()
c = Counter()
for k,v in building_data.items():
    floor = k.split('_')[0]
    if floor in seen or v!=v or 'kg' not in k:
        continue
    seen.add(floor)
for x in seen:
    parts = re.split('([A-Z])',x)
    parts = [p for p in parts if p!='']
    parts = [int(p) if p.isdigit() else p for p in parts]
    if 'B' in parts:
        c.update(['Basement'])
    elif 'R' in parts:
        c.update(['Roof'])
    elif 0 in parts:
        c.update(['Ground'])
    else:
        c.update(['Above Ground'])
print(f'Floors relative to ground for building {BUILDING_ID}:')
for k,v in c.items():
    print(f'{k}: {v} floor(s)')
```

Floors relative to ground for building 043:

Above Ground: 2 floor(s)

Basement: 1 floor(s)

Ground: 1 floor(s)

Roof: 1 floor(s)

### 6.2 2. Get climate conditions for a given building

This code retrieves local climate zones for a given building.

```
[54]: from geopy.geocoders import Nominatim
locator = Nominatim(user_agent="ConstructionDataset")
name_map = {
    'TOR': 'Toronto',
```



```

        'WIN': 'Winnipeg',
        'NEW': 'New York',
        'RIC': 'Richmond',
        'MIS': 'Mississauga'
    }
    location = locator.geocode(f'{name_map[building_data.City]}, {building_data.
        ↳Country}')

```

```

[55]: ranked_stations = eewweather.rank_stations(location.latitude,location.longitude)
ranked_stations = ranked_stations[~ranked_stations.iecc_climate_zone.isnull()]
station, warnings = eewweather.select_station(ranked_stations)

```

```

[56]: print(f'Climate zones for building {BUILDING_ID}:')
for k,v in station.climate_zones.items():
    if v is None:
        continue
    print(f'{k}: {v}')

```

```

Climate zones for building 043:
iecc_climate_zone: 5
iecc_moisture_regime: A
ba_climate_zone: Cold

```

```

[ ]:

```