LIS - Everything Counts - Assignment 1

The trees of London

https://github.com/nicc/lis-stats-1

Nic Young

Data is sourced from the Greater London Authority list of maintained trees.

Each row represents a single tree planted in the city of London. There are 817,150 total records, but we have cleaned this to leave 227,020. Note that, whilst this dataset ostensibly represents the entire population, we should treat this as a sample due to gaps in the data. The data set was last updated in July 2021.

Available variables:

variable	description
`objectid`	record identifier.
`borough`	The borough of London in which the tree resides.
`maintainer`	The entity responsible for maintenance of the tree.
`gla_tree_name`	The display name used for navigating the Greater London Authority tree map.
`tree_name`	Unsure. This appears to be an uncleaned version of gla_tree_name.
`taxon_name`	Botanical name. This is the most specific identifier of the "same" tree across records.
`common_name`	Common name.
`age`	This variable is messy and not reliable. See age_group below.
`age_group`	This classifies each tree as being in a specific age range. Options are: 'Young (0-15)', 'Early mature (16-30)', 'Mature (31-80)', 'Over mature (81-150)', and 'Veteran (over 150)'. Some are undefined.
`heigh_m`	Height in meters. This variable is sparsely populated and inconsistent in format.
`spread_m`	Canopy spread in meters. This variable is sparsely populated.
`canopy_spread_group`	This classifies each tree as being in a specific range of canopy spread (in meters). Options are: '00 to 05m', '05

variable	description
	to 10m', '10 to 15m', and '15 to 20m'. Some are undefined.
diameter_at_breast_height	Diameter at breast height in meters. This variable is sparsely populated.
dbh_group	This classifies each tree as being in a specific range of diameter at breast height (in meters). Options are: '21 to 40cm', '41 to 70cm', '11 to 20cm', 'Upto 10cm', and '70cm+'. This variable is sparsely populated.
longitude	The longitudinal location of the tree.
latitude	The latitudinal location of the tree.
condition	The condition of the tree. Options are: 'Reasonable', 'Good', 'Poor', and 'Dead'. Data only available for Kingston Upon Thames
load_data	The date the tree was first recorded.
updated	The date the tree was last updated.

Exploratory analysis

Let's load the data and clean it...

In [1]:
```
import warnings
warnings.filterwarnings("ignore")

import numpy as np
import pandas as pd

# load the data
data_file = './data/Borough_tree_list_2021July.csv'
raw = pd.read_csv(data_file)

# remove records that are not identified as species (this includes tree stum
data = raw[~raw['taxon_name'].str.lower().str.startswith('zz', na=True)].sor

# strip leading and trailing whitespace from age group values
data['age_group'] = data['age_group'].str.strip()

# remove NaN age_group values
data = data[data['age_group'].notna()]

# remove 'Undefined' age_group values
data = data[data['age_group'] != 'Undefined']

# remove 'Out' from boroughs
data = data[data['borough'] != 'Out']
```

Let's look at the data

How many of each species is present in each borough?

In [2]:
```
# group by borough and species as a matrix
species_matrix = data.groupby(['borough', 'taxon_name']).size().unstack(fill
species_matrix.head(50)
```

Out[2]:

taxon_name	Abies	Abies alba	Abies grandis	Abies koreana	Abies lasiocarpa	Abies nordmanniana	Abies procera	Ac
borough								
Barking and Dagenham	0	0	0	0	0	0	0	
Barnet	0	0	0	0	0	0	0	
Bexley	0	0	0	0	0	0	0	
Brent	0	0	0	0	0	0	0	
Bromley	0	0	0	0	0	0	0	
Camden	0	0	0	0	0	0	0	
City	0	0	1	0	0	0	0	
Croydon	0	0	0	0	0	0	0	
Ealing	0	0	0	0	0	0	0	
Enfield	0	0	0	0	0	0	0	
Greenwich	0	0	0	0	0	0	0	
Hackney	0	0	0	0	0	0	0	
Hammersmith and Fulham	0	0	0	0	0	0	0	
Haringey	0	0	0	0	0	0	0	
Havering	0	0	0	0	0	0	0	
Hillingdon	0	0	0	1	0	0	0	
Hounslow	0	0	0	0	0	0	0	
Islington	0	0	0	0	0	0	0	
Kensington and Chelsea	0	0	0	0	0	0	0	
Kingston upon Thames	0	0	15	0	0	0	0	
Lambeth	0	0	0	0	0	0	0	
Lewisham	24	0	0	0	0	0	0	
Merton	0	0	0	0	0	0	0	
Newham	1	0	0	0	0	0	0	
Redbridge	2	0	2	3	0	1	1	
Richmond	0	0	0	0	0	0	0	
Southwark	0	10	2	0	2	1	0	

taxon_name	Abies	Abies alba	Abies grandis	Abies koreana	Abies lasiocarpa	Abies nordmanniana	Abies procera	Ac
borough								
Sutton	0	0	0	0	0	0	0	
Tower Hamlets	0	0	0	0	0	0	0	
Waltham Forest	0	0	0	0	0	0	0	
Wandsworth	0	0	0	0	0	0	0	
Westminster	0	0	0	0	0	0	0	

32 rows × 528 columns

That's looking a bit sparse. Let's sense check a very common species...

In [3]:
```
# sycamore and wild cherry
selected_species = ['Acer pseudoplatanus', 'Prunus avium']
print(species_matrix[selected_species])
```

taxon_name	Acer pseudoplatanus	Prunus avium
borough		
Barking and Dagenham	85	76
Barnet	61	201
Bexley	23	84
Brent	8	17
Bromley	35	2
Camden	228	146
City	7	11
Croydon	29	26
Ealing	33	38
Enfield	32	18
Greenwich	13	20
Hackney	2	26
Hammersmith and Fulham	0	0
Haringey	15	15
Havering	27	46
Hillingdon	823	541
Hounslow	117	119
Islington	8	5
Kensington and Chelsea	3	45
Kingston upon Thames	13	208
Lambeth	8	1
Lewisham	2085	131
Merton	14	43
Newham	740	472
Redbridge	1894	634
Richmond	16	29
Southwark	3475	1892
Sutton	57	45
Tower Hamlets	48	39
Waltham Forest	50	24
Wandsworth	25	2
Westminster	1	0

Okay that's good.

Let's try to visualise diversity across boroughs. It's going to be a big plot...

In [4]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb

plt.figure(figsize=(450, 15))

sb.heatmap(species_matrix, cmap='YlGnBu', linewidths=.2)

plt.title('Distribution of Tree Species Across London Boroughs', fontsize=8)
plt.xlabel('Species')
plt.ylabel('Borough')

plt.show()
```

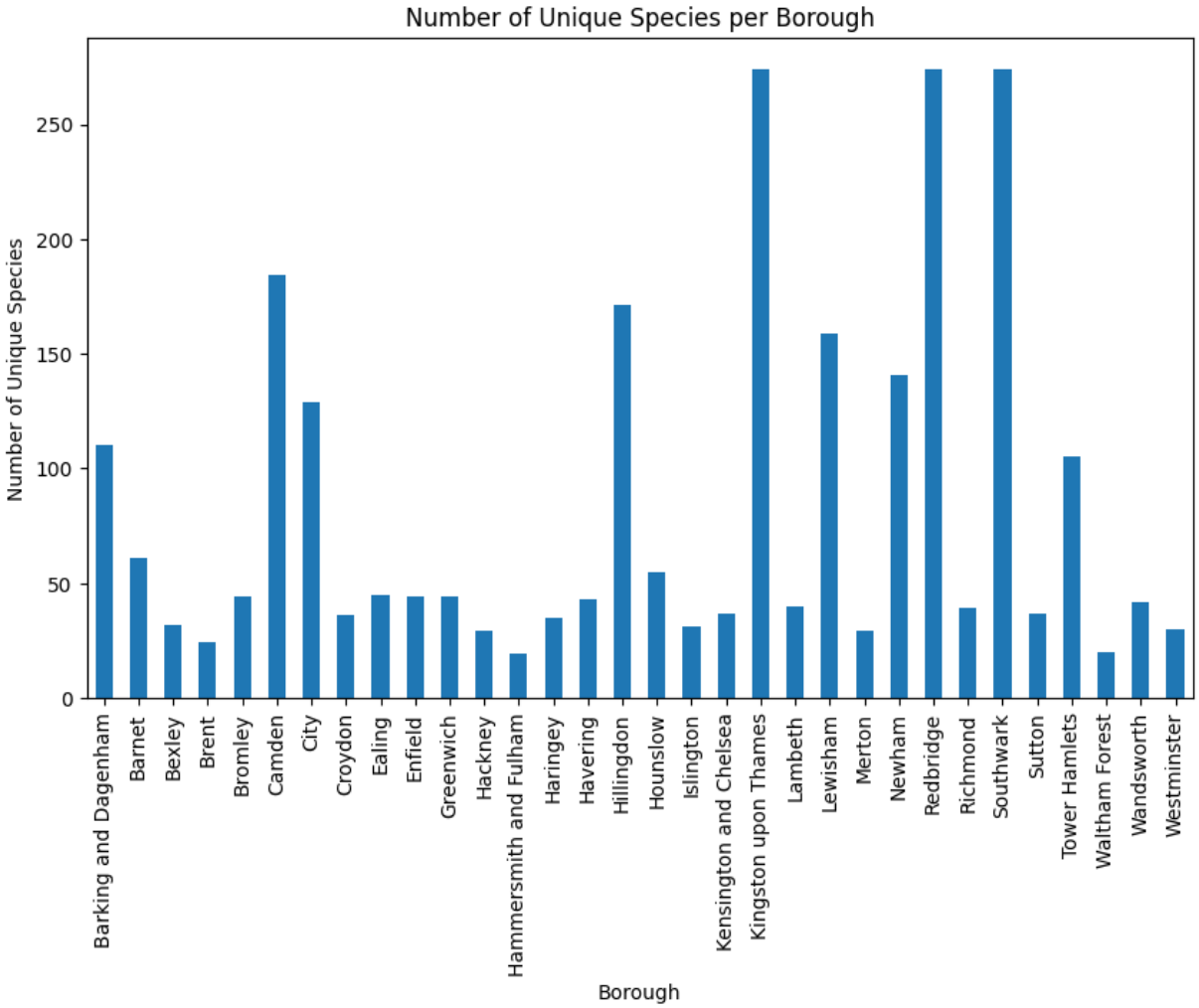
Okay that's a bit extreme. We have a few outliers and an otherwise fairly even distribution. It's also just way too much information. So it doesn't tell us much.

Let's try to reduce each borough to a value representing its diversity. Starting simple, we'll just count the number of unique species in each borough and plot that on a bar chart.

In [5]:
```
# group by borough and species
species_count = data.groupby('borough')['taxon_name'].nunique()
print(species_count)
```

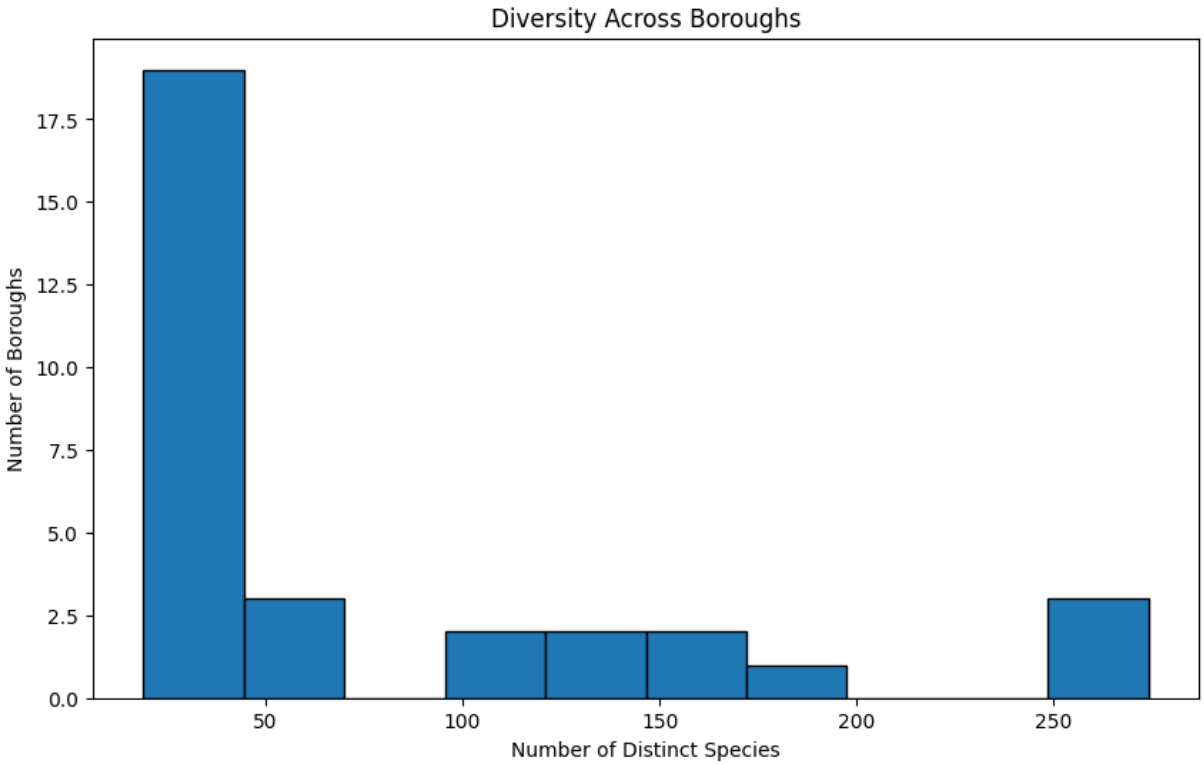
```
borough
Barking and Dagenham      110
Barnet                     61
Bexley                     32
Brent                      24
Bromley                    44
Camden                    184
City                      129
Croydon                    36
Ealing                     45
Enfield                    44
Greenwich                  44
Hackney                    29
Hammersmith and Fulham     19
Haringey                   35
Havering                   43
Hillingdon                171
Hounslow                   55
Islington                  31
Kensington and Chelsea     37
Kingston upon Thames      274
Lambeth                    40
Lewisham                  159
Merton                     29
Newham                    141
Redbridge                 274
Richmond                   39
Southwark                 274
Sutton                     37
Tower Hamlets             105
Waltham Forest             20
Wandsworth                 42
Westminster                30
Name: taxon_name, dtype: int64
```

In [6]:
```
# bar chart
species_count.plot(kind='bar', figsize=(10, 6))

plt.title('Number of Unique Species per Borough')
plt.xlabel('Borough')
plt.ylabel('Number of Unique Species')

plt.show()
```



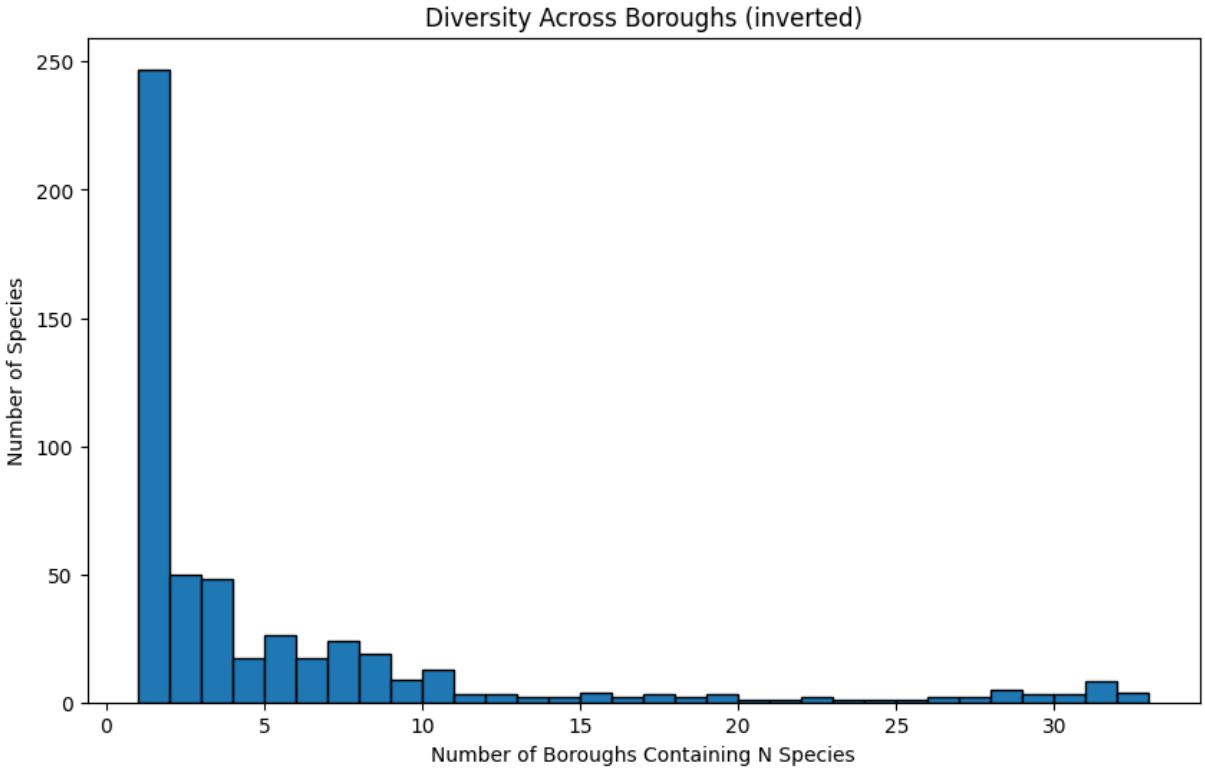
How is that distributed?

In [7]:
```
# histogram of the number of species per borough
plt.figure(figsize=(10, 6))
plt.hist(species_count, bins=10, edgecolor='black')

plt.title('Diversity Across Boroughs')
plt.xlabel('Number of Distinct Species')
plt.ylabel('Number of Boroughs')

plt.show()
```



What if we inverted that?

In [8]:
```
# count the number of boroughs each species is found in
species_per_borough_count = data.groupby('taxon_name')['borough'].nunique()

# histogram showing the number of species by number of boroughs
plt.figure(figsize=(10, 6))
plt.hist(species_per_borough_count, bins=range(1, species_per_borough_count.

plt.title('Diversity Across Boroughs (inverted)')
plt.xlabel('Number of Boroughs Containing N Species')
plt.ylabel('Number of Species')

plt.show()
```

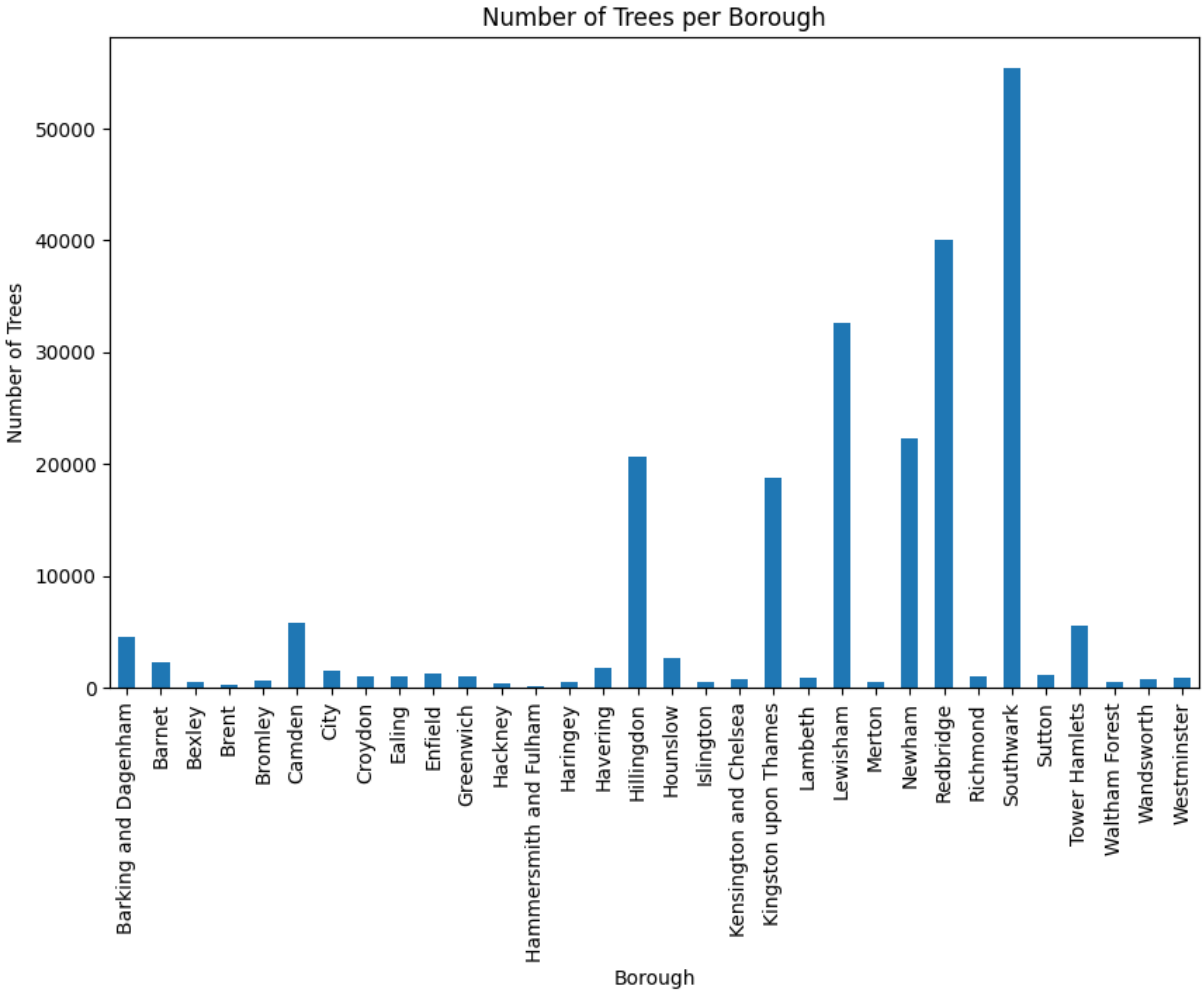


Okay what about sheer number of trees?

In [9]:
```
# count the records per borough
tree_count = data.groupby('borough').size()
print(tree_count)
```

```
borough
Barking and Dagenham       4574
Barnet                     2250
Bexley                      471
Brent                       259
Bromley                     672
Camden                     5801
City                       1544
Croydon                     954
Ealing                      941
Enfield                    1262
Greenwich                   941
Hackney                     427
Hammersmith and Fulham      161
Haringey                    512
Havering                   1717
Hillingdon                20598
Hounslow                   2586
Islington                   506
Kensington and Chelsea      787
Kingston upon Thames      18702
Lambeth                     879
Lewisham                  32628
Merton                      516
Newham                    22230
Redbridge                 39996
Richmond                    991
Southwark                 55375
Sutton                     1070
Tower Hamlets              5552
Waltham Forest              493
Wandsworth                  704
Westminster                 921
dtype: int64
```

In [10]:
```
# bar chart
tree_count.plot(kind='bar', figsize=(10, 6))

plt.title('Number of Trees per Borough')
plt.xlabel('Borough')
plt.ylabel('Number of Trees')

plt.show()
```

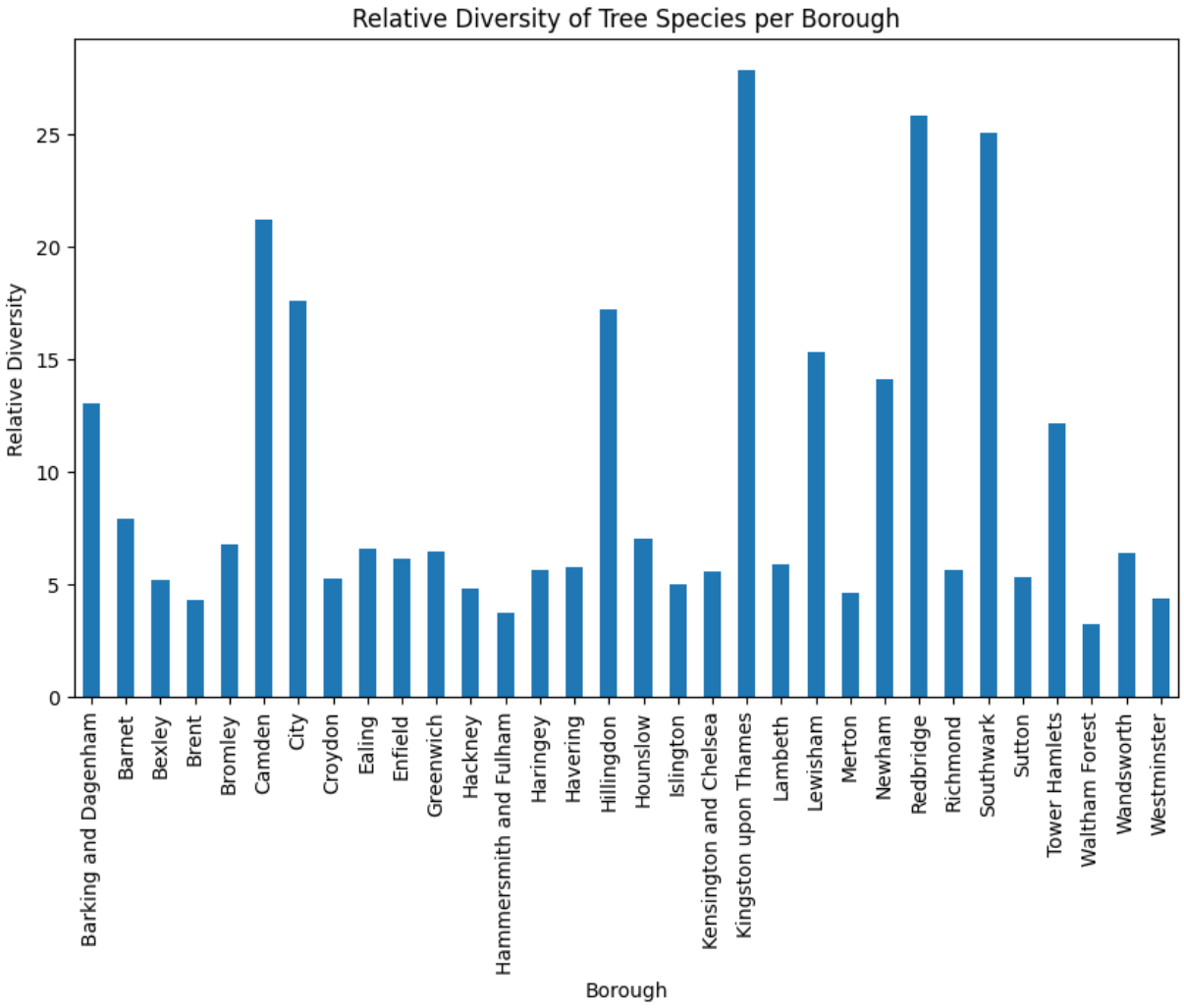


Okay the number of distinct species is obviously skewed by the variance in sheer number of trees. What if we ascribe a simple, relative diversity score to normalise for totals?

In [11]:
```
# unique species / log(total trees)
relative_diversity = species_count / np.log(tree_count)
print(relative_diversity)
```

```
borough
Barking and Dagenham      13.051510
Barnet                     7.902900
Bexley                     5.199145
Brent                      4.319011
Bromley                    6.758564
Camden                    21.232928
City                      17.569829
Croydon                    5.247306
Ealing                     6.572276
Enfield                    6.162074
Greenwich                  6.426225
Hackney                    4.788020
Hammersmith and Fulham     3.739124
Haringey                   5.610481
Havering                   5.773103
Hillingdon                17.215431
Hounslow                   6.999354
Islington                  4.978691
Kensington and Chelsea     5.548700
Kingston upon Thames      27.855760
Lambeth                    5.900763
Lewisham                  15.298868
Merton                     4.642892
Newham                    14.087043
Redbridge                 25.857510
Richmond                   5.653227
Southwark                 25.087248
Sutton                     5.304345
Tower Hamlets             12.178271
Waltham Forest             3.225542
Wandsworth                 6.405585
Westminster                4.395308
dtype: float64
```

In [12]:
```
# bar chart
relative_diversity.plot(kind='bar', figsize=(10, 6))

plt.title('Relative Diversity of Tree Species per Borough')
plt.xlabel('Borough')
plt.ylabel('Relative Diversity')

plt.show()
```



Okay that's more interesting.

Now let's look at a different facet of the data: age groups. We'll show the percentage of each age group per borough as a 100% stacked bar chart...

In [13]:
```
# number of trees in a given age group by borough, as matrix
age_group_count = data.groupby(['borough', 'age_group']).size().unstack(fill
print(age_group_count)
```

age_group	Early mature (16–30)	Mature (31–80)	\
borough			
Barking and Dagenham	61	757	
Barnet	789	498	
Bexley	285	168	
Brent	32	18	
Bromley	265	362	
Camden	118	2506	
City	497	586	
Croydon	310	507	
Ealing	272	92	
Enfield	267	221	
Greenwich	426	358	
Hackney	186	114	
Hammersmith and Fulham	37	52	
Haringey	143	153	
Havering	832	131	
Hillingdon	18639	192	
Hounslow	874	693	
Islington	283	115	
Kensington and Chelsea	290	255	
Kingston upon Thames	6639	4135	
Lambeth	340	261	
Lewisham	9375	20272	
Merton	201	178	
Newham	2079	2373	
Redbridge	12177	8033	
Richmond	197	499	
Southwark	17091	17213	
Sutton	442	444	
Tower Hamlets	311	2912	
Waltham Forest	144	22	
Wandsworth	337	142	
Westminster	259	472	

age_group	Over mature (81–150)	Veteran (over 150)	Young (0–1
5)			
borough			
Barking and Dagenham	22	0	37
34			
Barnet	3	6	9
54			
Bexley	1	0	
17			
Brent	0	0	2
09			
Bromley	6	1	
38			
Camden	38	1781	13
58			
City	0	0	4
61			
Croydon	0	0	1
37			
Ealing	4	0	5
73			

```
Enfield                          0                0           7
74
Greenwich                        4                0           1
53
Hackney                          2                0           1
25
Hammersmith and Fulham           0                0
72
Haringey                         1                0           2
15
Havering                         4                4           7
46
Hillingdon                       1                0          17
66
Hounslow                         3                2          10
14
Islington                        0                0           1
08
Kensington and Chelsea           0                0           2
42
Kingston upon Thames          1206                0          67
22
Lambeth                          3                0           2
75
Lewisham                      1372                0          16
09
Merton                           3                0           1
34
Newham                          32                1         177
45
Redbridge                     4209              283         152
94
Richmond                        17                0           2
78
Southwark                     3019              138         179
14
Sutton                           5                0           1
79
Tower Hamlets                    1                0          23
28
Waltham Forest                   0                0           3
27
Wandsworth                       1                0           2
24
Westminster                      0                0           1
90
```

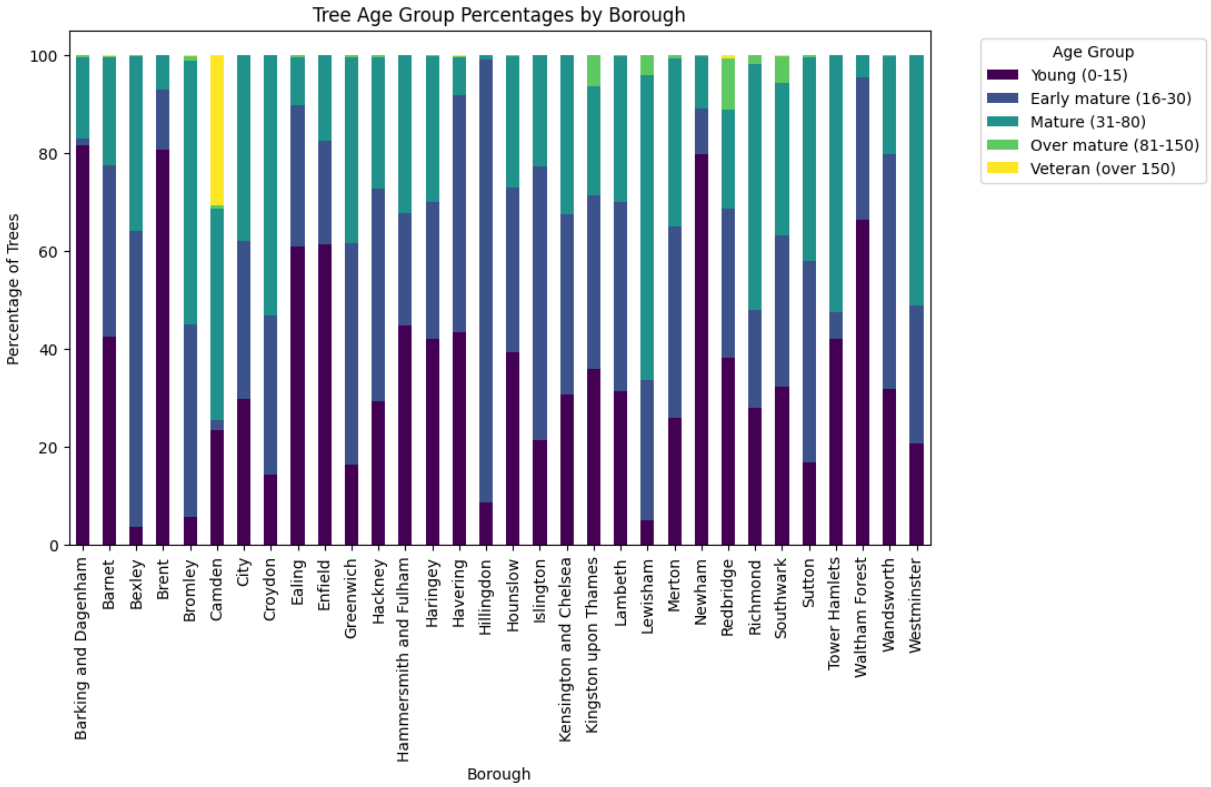
In [14]:
```
# normalize by total trees in each borough to get percentages
age_group_percentages = age_group_count.div(age_group_count.sum(axis=1), axi

# reorder columns
age_group_order = ['Young (0-15)', 'Early mature (16-30)', 'Mature (31-80)',
age_group_percentages = age_group_percentages[age_group_order]

print(age_group_percentages)
```

age_group \ borough	Young (0-15)	Early mature (16-30)	Mature (31-80)
Barking and Dagenham	81.635330	1.333625	16.550066
Barnet	42.400000	35.066667	22.133333
Bexley	3.609342	60.509554	35.668790
Brent	80.694981	12.355212	6.949807
Bromley	5.654762	39.434524	53.869048
Camden	23.409757	2.034132	43.199448
City	29.857513	32.189119	37.953368
Croydon	14.360587	32.494759	53.144654
Ealing	60.892667	28.905420	9.776833
Enfield	61.331220	21.156894	17.511886
Greenwich	16.259299	45.270988	38.044633
Hackney	29.274005	43.559719	26.697892
Hammersmith and Fulham	44.720497	22.981366	32.298137
Haringey	41.992188	27.929688	29.882812
Havering	43.447874	48.456610	7.629586
Hillingdon	8.573648	90.489368	0.932129
Hounslow	39.211137	33.797370	26.798144
Islington	21.343874	55.928854	22.727273
Kensington and Chelsea	30.749682	36.848793	32.401525
Kingston upon Thames	35.942680	35.498877	22.109935
Lambeth	31.285552	38.680319	29.692833
Lewisham	4.931347	28.732990	62.130685
Merton	25.968992	38.953488	34.496124
Newham	79.824561	9.352227	10.674764
Redbridge	38.238824	30.445545	20.084508
Richmond	28.052472	19.878910	50.353179
Southwark	32.350339	30.864108	31.084424
Sutton	16.728972	41.308411	41.495327
Tower Hamlets	41.930836	5.601585	52.449568
Waltham Forest	66.328600	29.208925	4.462475
Wandsworth	31.818182	47.869318	20.170455
Westminster	20.629750	28.121607	51.248643

age_group borough	Over mature (81-150)	Veteran (over 150)
Barking and Dagenham	0.480979	0.000000
Barnet	0.133333	0.266667
Bexley	0.212314	0.000000
Brent	0.000000	0.000000
Bromley	0.892857	0.148810
Camden	0.655059	30.701603
City	0.000000	0.000000
Croydon	0.000000	0.000000
Ealing	0.425080	0.000000
Enfield	0.000000	0.000000
Greenwich	0.425080	0.000000
Hackney	0.468384	0.000000
Hammersmith and Fulham	0.000000	0.000000
Haringey	0.195312	0.000000
Havering	0.232964	0.232964
Hillingdon	0.004855	0.000000
Hounslow	0.116009	0.077340
Islington	0.000000	0.000000

Kensington and Chelsea	0.000000	0.000000
Kingston upon Thames	6.448508	0.000000
Lambeth	0.341297	0.000000
Lewisham	4.204977	0.000000
Merton	0.581395	0.000000
Newham	0.143950	0.004498
Redbridge	10.523552	0.707571
Richmond	1.715439	0.000000
Southwark	5.451919	0.249210
Sutton	0.467290	0.000000
Tower Hamlets	0.018012	0.000000
Waltham Forest	0.000000	0.000000
Wandsworth	0.142045	0.000000
Westminster	0.000000	0.000000

In [15]:
```
# 100% stacked bar chart
age_group_percentages.plot(kind='bar', stacked=True, figsize=(10, 6), colorm

plt.title('Tree Age Group Percentages by Borough')
plt.xlabel('Borough')
plt.ylabel('Percentage of Trees')
plt.legend(title='Age Group', bbox_to_anchor=(1.05, 1), loc='upper left')

plt.show()
```



We have some interesting data in Kingston Upon Thames that is not available for other boroughs. Let's look at that too...

First up: tree condition...

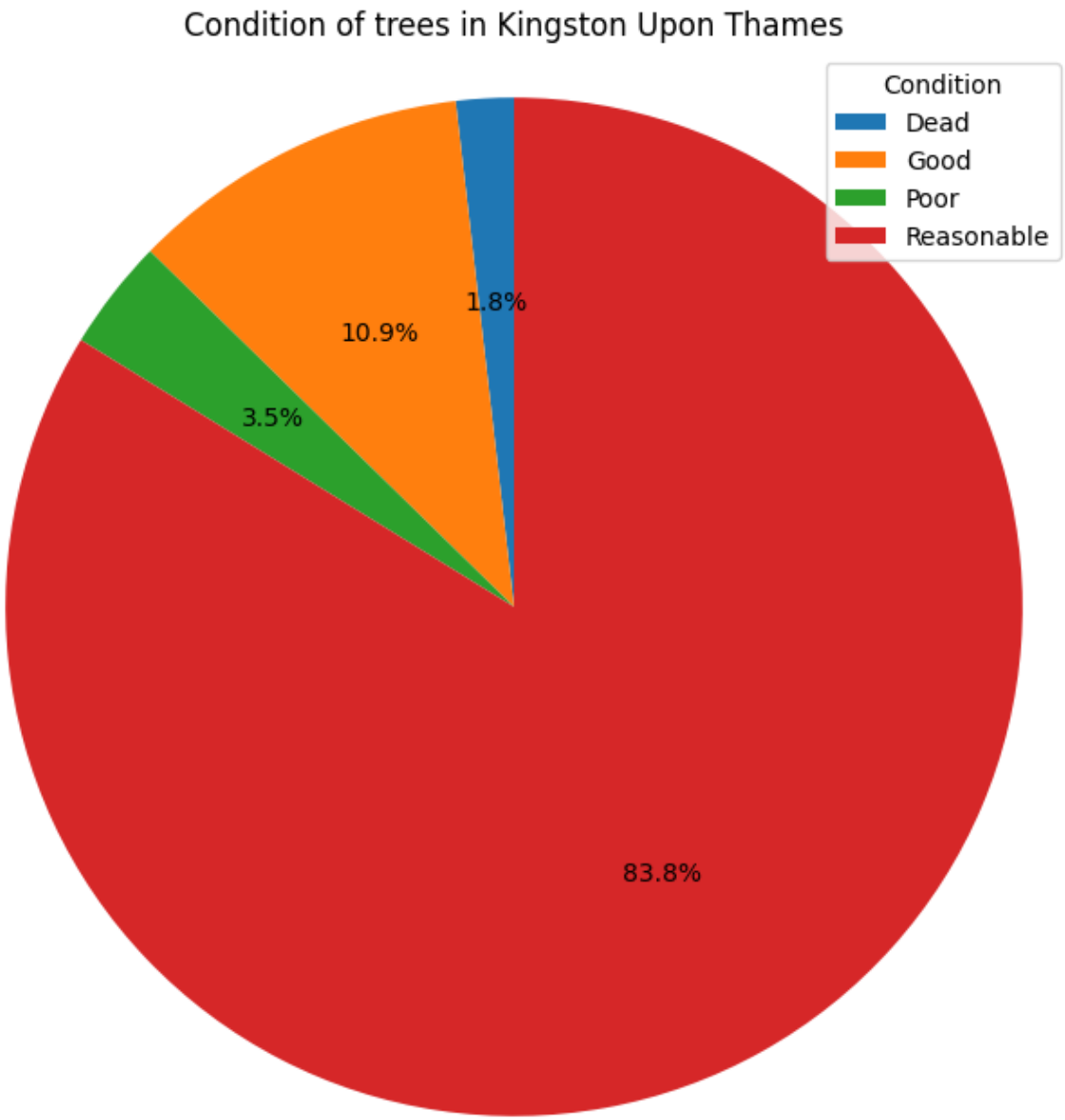
In [16]:
```
# filter for rows that have the condition property, group by borough and con
```

```
condition_count = data[['borough', 'condition']].groupby(['borough', 'condit
print(condition_count)
```

```
borough               condition
Kingston upon Thames  Dead            333
                      Good            1974
                      Poor             641
                      Reasonable     15227
dtype: int64
```

A pie chart should suffice here...

In [17]:
```
# pie chart
plt.figure(figsize=(8, 8))
wedges, texts, autotexts = plt.pie(condition_count, autopct='%1.1f%%', start

plt.legend(wedges, [c for b, c in condition_count.index], title="Condition")
plt.title('Condition of trees in Kingston Upon Thames')
plt.axis('equal')

plt.show()
```



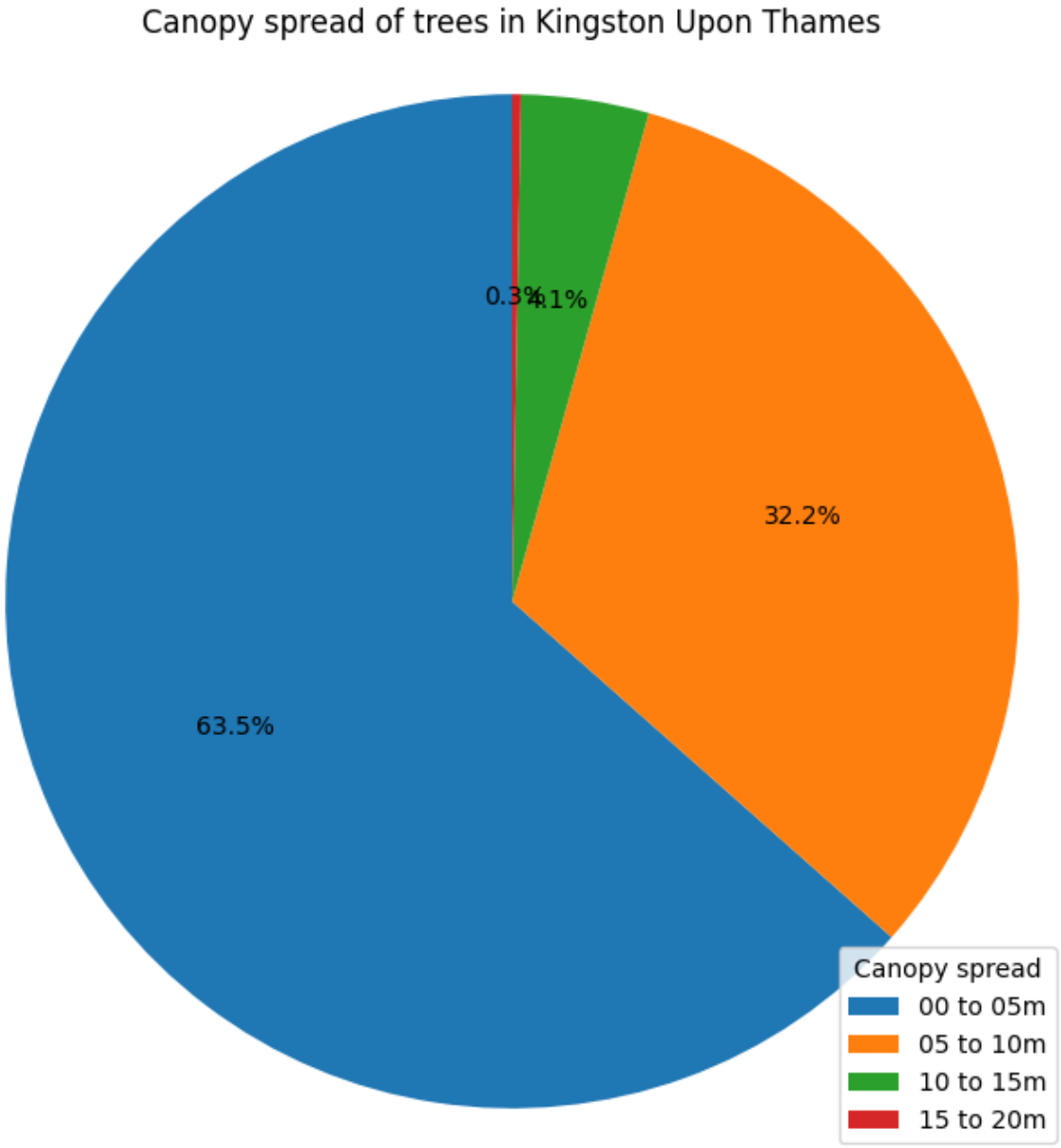
Next up: canopy spread group...

In [18]:
```
# filter for rows that have a canopy spread group, group by borough and cano
canopy_spread_count = data[['borough', 'canopy_spread_group']].groupby(['bor
print(canopy_spread_count)
```

```
borough              canopy_spread_group
Kingston upon Thames  00 to 05m             11537
                      05 to 10m              5847
                      10 to 15m               740
                      15 to 20m                51
dtype: int64
```

In [19]:
```
# pie chart again
plt.figure(figsize=(8, 8))
wedges, texts, autotexts = plt.pie(canopy_spread_count, autopct='%1.1f%%', s

plt.legend(wedges, [c for b, c in canopy_spread_count.index], title="Canopy
```

```
plt.title('Canopy spread of trees in Kingston Upon Thames')
plt.axis('equal')

plt.show()
```



Let's see this broken down by age...

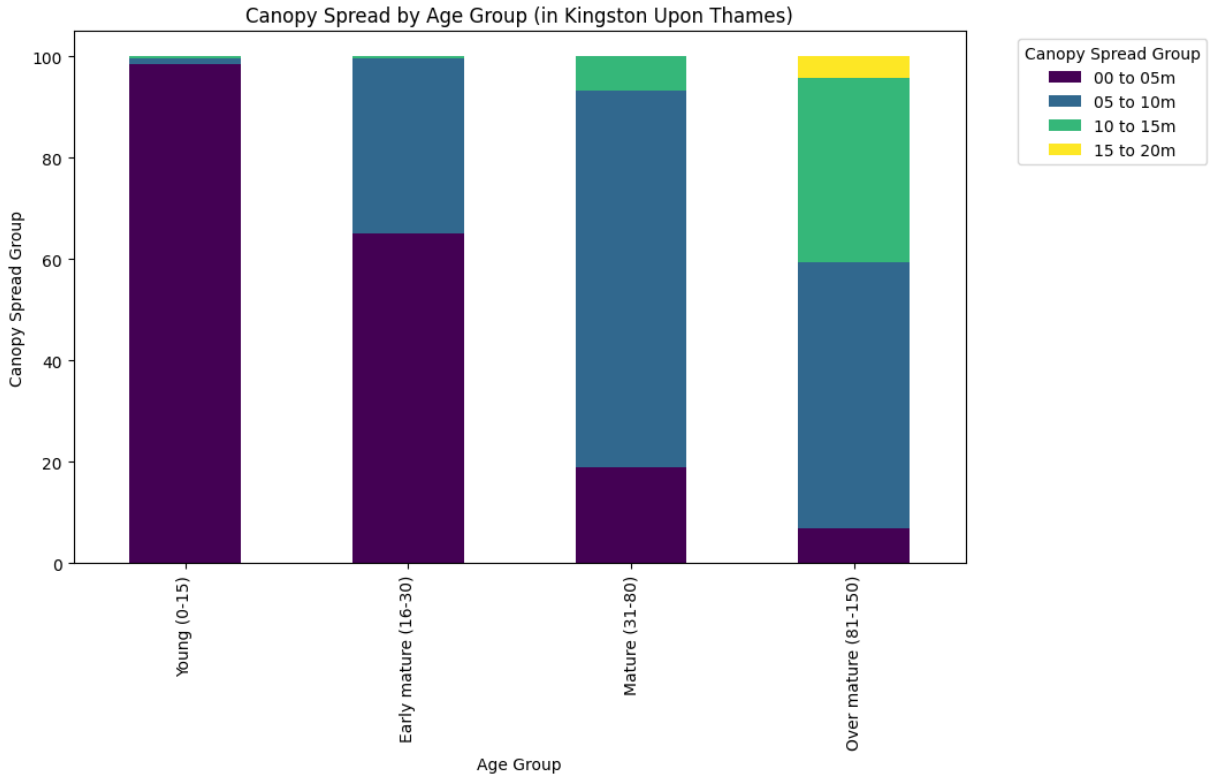
In [20]:
```
# group by age group and canopy spread group, in a matrix
canopy_spread_by_age_count = data.groupby(['age_group', 'canopy_spread_group
print(canopy_spread_by_age_count)
```

canopy_spread_group	00 to 05m	05 to 10m	10 to 15m	15 to 20m
age_group				
Early mature (16-30)	4174	2227	17	1
Mature (31-80)	740	2919	264	0
Over mature (81-150)	84	629	437	50
Young (0-15)	6539	72	22	0

In [21]:
```
# normalize by total trees in each age group to get percentages
canopy_spread_group_percentages = canopy_spread_by_age_count.div(canopy_spre

# reorder the bar stacks to be incremental
canopy_spread_group_order = ['00 to 05m', '05 to 10m', '10 to 15m', '15 to 2
canopy_spread_group_percentages = canopy_spread_group_percentages[canopy_spr

# reorder the columns to be incremental
age_group_order = ['Young (0-15)', 'Early mature (16-30)', 'Mature (31-80)',
canopy_spread_group_percentages = canopy_spread_group_percentages.reindex(ag

print(canopy_spread_group_percentages)
```

```
canopy_spread_group   00 to 05m  05 to 10m  10 to 15m  15 to 20m
age_group
Young (0-15)          98.582843   1.085482   0.331675   0.000000
Early mature (16-30)  65.025705  34.693878   0.264839   0.015579
Mature (31-80)        18.863115  74.407341   6.729544   0.000000
Over mature (81-150)   7.000000  52.416667  36.416667   4.166667
```

In [22]:
```
# 100% stacked bar chart
canopy_spread_group_percentages.plot(kind='bar', stacked=True, figsize=(10,

plt.title('Canopy Spread by Age Group (in Kingston Upon Thames)')
plt.xlabel('Age Group')
plt.ylabel('Canopy Spread Group')
plt.legend(title='Canopy Spread Group', bbox_to_anchor=(1.05, 1), loc='upper

plt.show()
```



This checks out I guess. Older == bigger :D

I think I'll stop here. Next up for assignment 2 will be to interrogate how well this sample represents the population, whether we can form any hypotheses, and whether we can verify them if so.