Crisis UK trusts and foundations mailing

Will Langdale

July 19, 2020

This is a quick project to use the Charity Commission's database of registered charities in England and Wales to create a mailing file. Note this project is not in any way affiliated with Crisis UK.

0. Contents

1. Task overview
    - Method
    - Disclaimer
2. Raw selection
3. Segmentation
    i) Charitable purpose
    ii) Expenditure
    iii) Nearest centre
4. Mailing file
    - Filtering the selection
    - Producing the outfile
5. Next steps
    - Accuracy of selection and missing data
    - Text mining and tf-idf
    - Trustees
    - Expenditure
    - Other languages
    - subno
6. Notes

1. Task overview

> In 2019 Crisis UK's income from charitable Trusts and Foundations (T&F) was £540k. To help grow this figure, the team have decided to put on a series of open days at their 11 Skylight Centres, and invite the trustees of interested charitable organisations to see the impact their gifts might have there.
>
> Your task is to build the file that will drive this mailing.

My plan is to select this mailing data from the Charity Commission's database that covers charities in England and Wales.

The eventual file will support a letter from the director of the local Skylight Centre inviting the trustees to their open day. It will include the following information for each charity:

- Name, address and email address of charity
- Expenditure and expenditure segment
- Charitable purpose segment
- Appropriate Skylight Centre to invite to, its distance, and centre signatory

Segmentation by expenditure will allow the T&F team to potentially ask very large charities for personal calls or meetings, rather than a collective open day. It will also help them to prioritise who the most important charities to ask might be, given my domain knowledge of Crisis is limited.

Segmentation by purpose will allow the T&F team to tailor their letter by the motivations of the charity to support. Perhaps they're specifically interested in helping the homeless, or perhaps it's that their concern for the poor might mean they also want to address homelessness.

Segmentation by Skylight Centre and distance will help the T&F team to tailor their letter by locale. Charities within a few miles might get a letter more focused on geographical benefits.

Method

1) Pull all charities whose way of working includes making grants to organisations into a dataframe
2) Add my segmentation scores one by one
3) Filter down to the people we actually want to contact
4) Create second dataframe of clean contact details
5) Join and export to .csv for delivery to client

Disclaimer

This project was done in R and this report produced in R Markdown. This project is in no way affiliated with Crisis UK, and is a purely academic task to help show and develop my skills with data. I chose Crisis because I think homelessness is one of the starkest failings of our society, and every day it continues to exist is an indictment of the systems that are supposed to make our society a safe, happy place to live.

2. Raw selection

Note some scripts for querying my database that include sensitive information have been quietly loaded, as has `tidyverse`, `geosphere`, `scales`, `rnaturalearth` and `stringr`.

First, we grab all charities who give grants to organisations (`class == 302`) and who are still operating.

```
ewtrusts <- runquery(sql = 'select ew_class.ewclass, ew_charity.*
                            from ew_charity
                            join ew_class on ew_charity.regno = ew_class.regno
                            where ew_charity.orgtype = \'R\' and ew_class.ewclass = \'302\';')
```

Then copy selection barebones to my propensity table.

```
ewtrusts_prop <- ewtrusts %>%
  select(regno, subno, name) %>%
  tibble()

ewtrusts_prop %>%
  print(n = 10)
```

```
## Warning: `...` is not empty.
##
## We detected these problematic arguments:
## * `needs_dots`
##
## These dots only exist to allow future extensions and should be empty.
## Did you misspecify an argument?
```

```
## # A tibble: 56,056 x 3
##      regno subno name
##      <int> <int> <chr>
##  1 1171136     0 THE WOOSNAM FOUNDATION
##  2 1171025     0 SYNEIDER MINISTRIES
##  3 1171029     0 SMILES TO TANZANIA CHARITY
##  4 1171030     0 THE GOOD SHEPHERD CHARITY
##  5 1171031     0 THE PAROCHIAL CHURCH COUNCIL OF THE ECCLESIASTICAL PARISH OF A~
##  6 1171035     0 DEEP GLOBAL
##  7 1171039     0 TADCASTER VOLUNTEER CARS & SERVICES ASSOCIATION
##  8 1171046     0 BRIMPSFIELD MUSIC SOCIETY
##  9 1171098     0 HQR LONDON CHARITABLE FOUNDATION
## 10 1171099     0 NORTH YARD COMMUNITY  TRUST CIO
## # ... with 56,046 more rows
```

3. Segmentation

We're going to slowly build these variables and add them to `ewtrusts_prop` as we go.

i) Charitable purpose

There are two ways of approaching this. One is to look at the `objects` table and use text analysis to create a propensity score for how well each charity's object matches Crisis UK's. I spent quite a bit of time on this, but didn't hone it enough to be useful. See the next steps section for more information.

The other is to look at the charity's declaration of what it does in the `class` table and segment this way. It's much blunter, but serviceable. This is the approach we're taking here.

We start by grabbing all the charities with aims shared with Crisis.

```
sharedaims <- runquery(sql = 'select regno, ewclass
                              from ew_class
                              where ew_class.ewclass in (\'101\', \'102\', \'105\', \'107\', \'113\');'
```

101 General, and 105 Poverty are particularly interesting but 107 is the standout – Housing.

```
sharedaims_asktype <- sharedaims %>%
  filter(ewclass == '105' | ewclass == '107' | ewclass == '101') %>%
  mutate(ewclass = gsub('105', 'Poverty', ewclass)) %>%
  mutate(ewclass = gsub('107', 'Housing', ewclass)) %>%
  mutate(ewclass = gsub('101', 'General', ewclass)) %>%
  pivot_wider(names_from = ewclass,
              values_from = ewclass) %>%
  unite(col = asktype,
        Poverty:Housing,
```

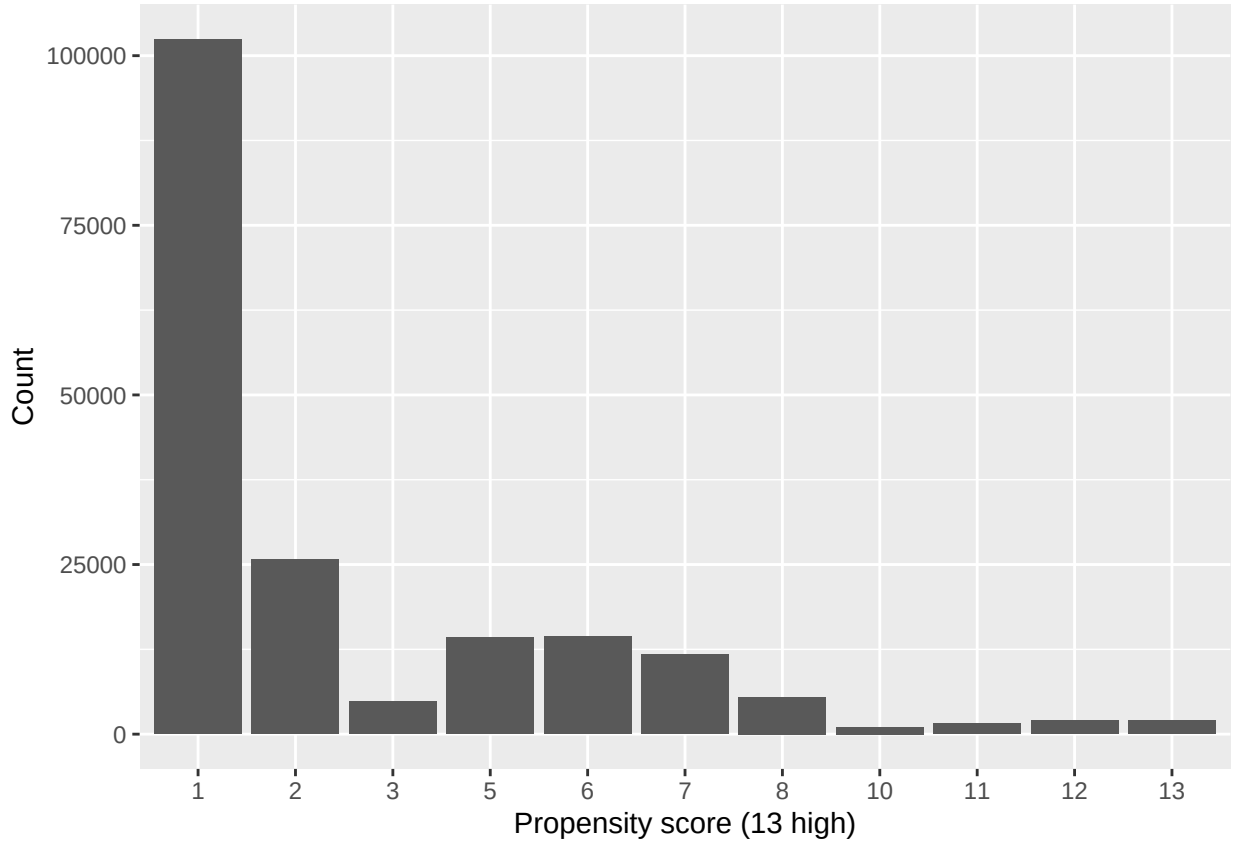
```
        sep = ' & ',
        na.rm = TRUE) %>%
  mutate(asktype = na_if(asktype, '')) %>%
  mutate(asktype = coalesce(asktype, General)) %>%
  select(-General)

sharedaims_asktype %>%
  count(asktype) %>%
  arrange(-n)
```

```
## Warning: `...` is not empty.
##
## We detected these problematic arguments:
## * `needs_dots`
##
## These dots only exist to allow future extensions and should be empty.
## Did you misspecify an argument?
```

```
## # A tibble: 4 x 2
##   asktype               n
##   <chr>             <int>
## 1 General           53680
## 2 Poverty           40145
## 3 Poverty & Housing  6748
## 4 Housing            5613
```

I used these aims to build a propensity score that weights Poverty and Housing highly, but ultimately I think the named categories will suffice.

```
sharedaims_score <- sharedaims %>%
  mutate(weight = case_when(ewclass == '105' | ewclass == '107' ~ 5, TRUE ~ 1)) %>%
  group_by(regno) %>%
  summarise(count = sum(weight))

ggplot(sharedaims_score,
       aes(x = factor(count))) +
  geom_bar() +
  labs(x = 'Propensity score (13 high)',
       y = 'Count')
```



And finally add to the propensity frame.

```
sharedaims <- sharedaims_score %>%
  left_join(sharedaims_asktype) %>%
  mutate(propensity_aims = round((count / 13) * 10, 2))

ewtrusts_prop <- ewtrusts_prop %>%
  left_join(select(sharedaims, regno, asktype, propensity_aims))
```

ii) Expenditure

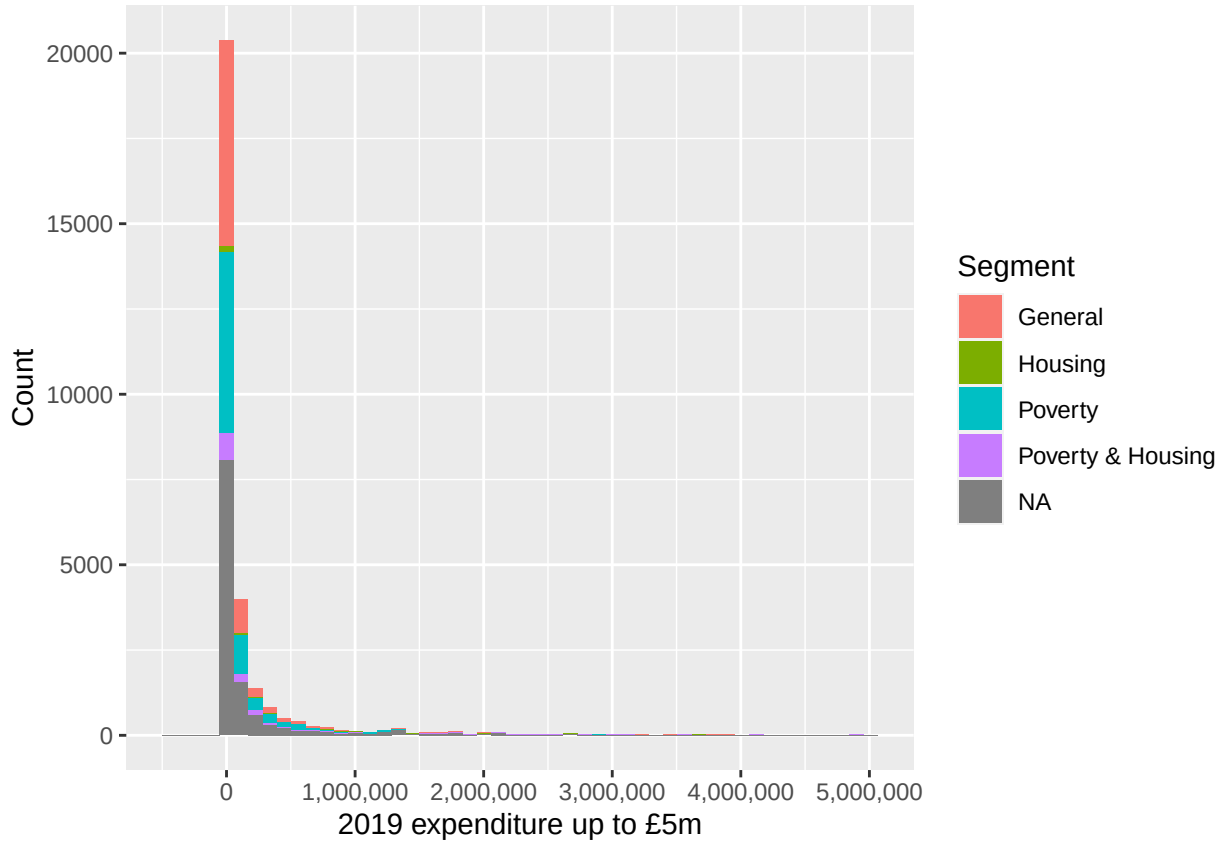
We'll use 2019 expenditure to check that the trust is actively making gifts, then look at the size of their expenditure to get an idea of their size. We're using this because lots of charities are slow, COVID-19 has slowed them more, and FY1920 data isn't as up to date as it could be right now.

Start by grabbing FY1819 expenditure and adding to the propensity dataframe.

```
endof19exp <- runquery(sql = 'select f.regno, f.fyend, f.expend
                              from ew_financial f
                              inner join(
                                select regno, max(fyend) fyend
                                from ew_financial
                                where fyend >= \'2019-01-01\' and fyend <= \'2019-12-31\' and expend is
                                group by regno
                              ) f2 on f.regno = f2.regno and f.fyend = f2.fyend;')
```

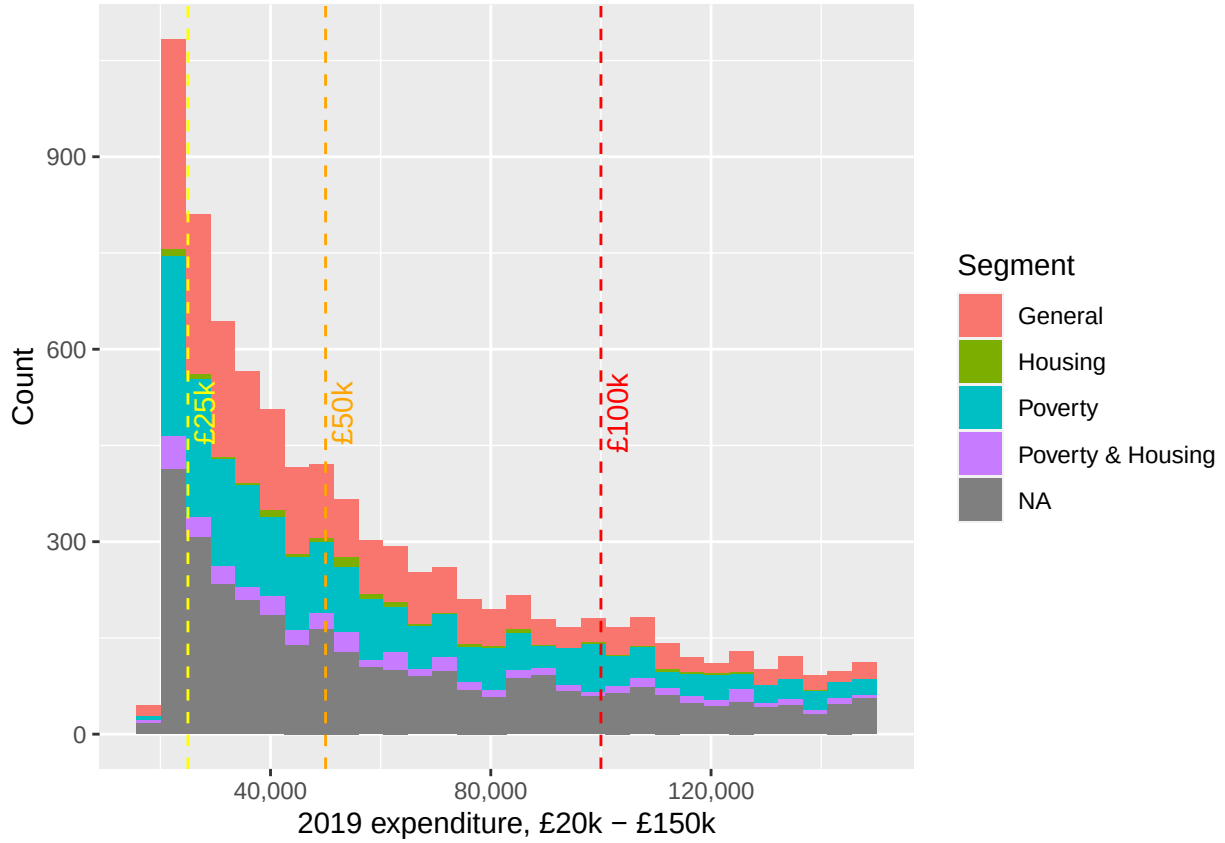
```
colnames(endof19exp)[3] <- 'expend2019'

ewtrusts_prop <- ewtrusts_prop %>%
  left_join(select(endof19exp, regno, expend2019))
```

This data is heavily frontloaded to about £250k, with a very, very long tail. The top two expenditures are £1.2bn and £1.3bn – The British Council and the Wellcome Trust. They're so big I actually couldn't store them as a 32-bit integer in the database!



Let's take a look at some of those lower values. There's not much point in going to charities who give away less than £20k a year, so let's look from there to £150k.

It seems there's some clear places it might be useful to cut the data.



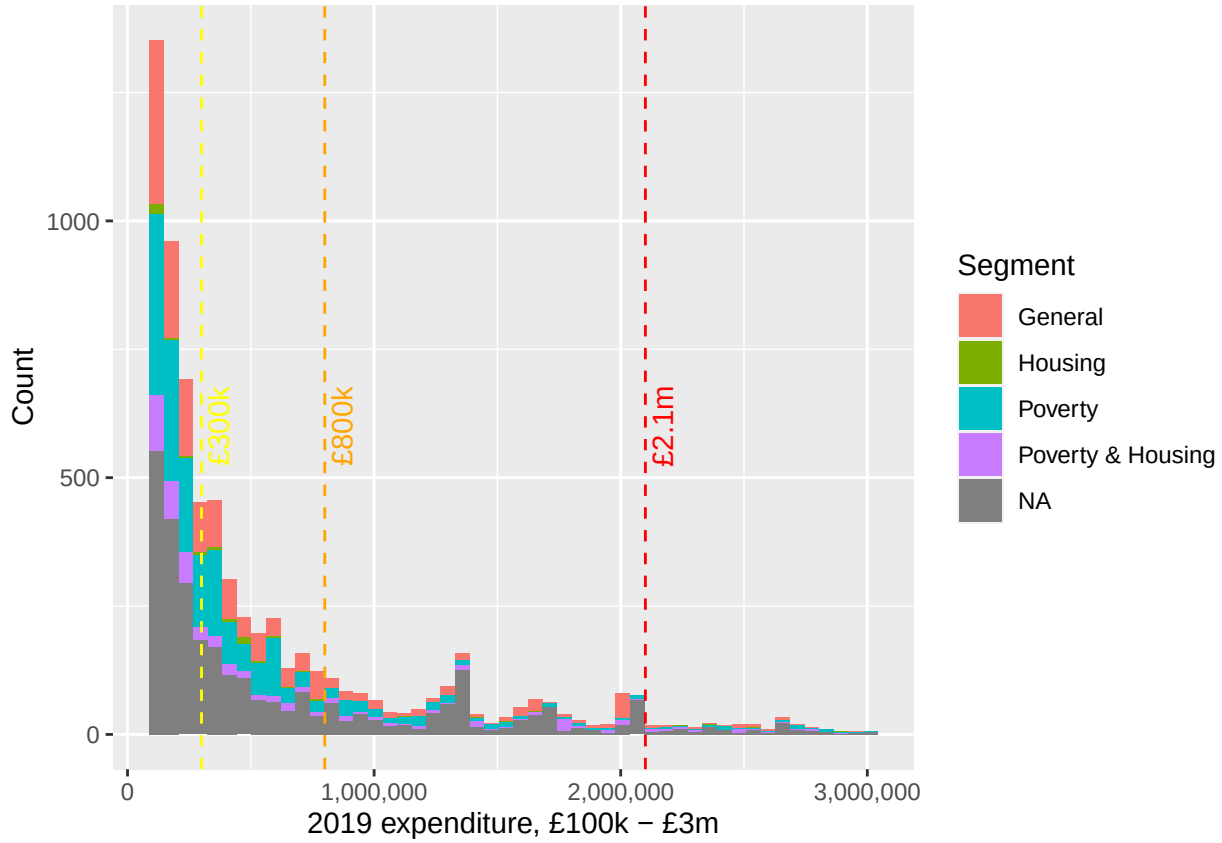
Let's look at some counts.

```
ewtrusts_prop %>%
  mutate(expendcat = cut(expend2019,
                         breaks = c(25000, 50000, 100000, Inf),
                         labels = c('low', 'mid', 'high'))) %>%
  count(expendcat, asktype) %>%
  pivot_wider(names_from = expendcat,
              values_from = n)
```

```
## Warning: `...` is not empty.
##
## We detected these problematic arguments:
## * `needs_dots`
##
## These dots only exist to allow future extensions and should be empty.
## Did you misspecify an argument?
```

```
## # A tibble: 5 x 5
##   asktype             low   mid  high  `NA`
##   <chr>             <int> <int> <int> <int>
## 1 General             992   733  1743 10478
## 2 Housing              32    58   150   483
## 3 Poverty             818   788  2150 10745
## 4 Poverty & Housing   145   172   634  1850
## 5 <NA>               1145   999  3877 18064
```

That's a lot in the high section! We should look at resegmenting. Let's plot from 100k expenditure upwards. Again, there seem to be some logical places to cut this data.



Now the counts, this time including our slices across both plots.

```
ewtrusts_prop %>%
  mutate(expendcat = cut(expend2019,
                         breaks = c(25000, 50000, 100000, 300000, 800000, 2100000, Inf),
                         labels = c('vsml', 'sml', 'med', 'hmed', 'hgh', 'vhgh'))) %>%
  count(expendcat, asktype) %>%
  pivot_wider(names_from = expendcat,
              values_from = n)
```

```
## Warning: `...` is not empty.
##
## We detected these problematic arguments:
## * `needs_dots`
##
## These dots only exist to allow future extensions and should be empty.
## Did you misspecify an argument?
```

```
## # A tibble: 5 x 8
##   asktype            vsml   sml   med  hmed   hgh  vhgh  `NA`
##   <chr>             <int> <int> <int> <int> <int> <int> <int>
```

```
## 1 General              992   733   724   464   268   287 10478
## 2 Housing               32    58    31    41     9    69   483
## 3 Poverty              818   788   903   607   235   405 10745
## 4 Poverty & Housing    145   172   258   115   117   144  1850
## 5 <NA>                1145   999  1387   755   696  1039 18064
```

This looks like a much more even distribution of expenditures that reflects the lay of the histograms. By adding these segments to the propensity and layering with `asktype`, the T&F team will be able to, say, contact small charities with an interest in Poverty and Housing to build local alliances, yet also open their mailing up to bigger charities that have more General ends, and pockets deep enough to meet them. This would hopefully increase their mailing ROI.

```
ewtrusts_prop <- ewtrusts_prop %>%
  mutate(expendcat = cut(expend2019,
                         breaks = c(25000, 50000, 100000, 300000, 800000, 2100000, Inf),
                         labels = c('vsml', 'sml', 'med', 'hmed', 'hgh', 'vhgh')))
```

iii) Nearest Centre

This mailing is essentially all about geography. We need to know we're asking trusts with a vested interest in each Skylight Centre to help support it. That could be the centre serving a similar community to the trust, or that the trust itself is geographically nearby.

There are two ways of approaching this. The first is to look at the postcode of the trust and calculate the closest Skylight Centre to them. The second, and preferred, is to look at their declared geographical area of benefit and see if a Skylight Centre is operating in one of them.

By combining these two methods, we should get a good idea of the best place to invite the trustees to visit, and to potentially make a gift to support.

Postcode distance We need the postcodes of both the trusts and the Skylight Centres, and the latitude and longitude of each.

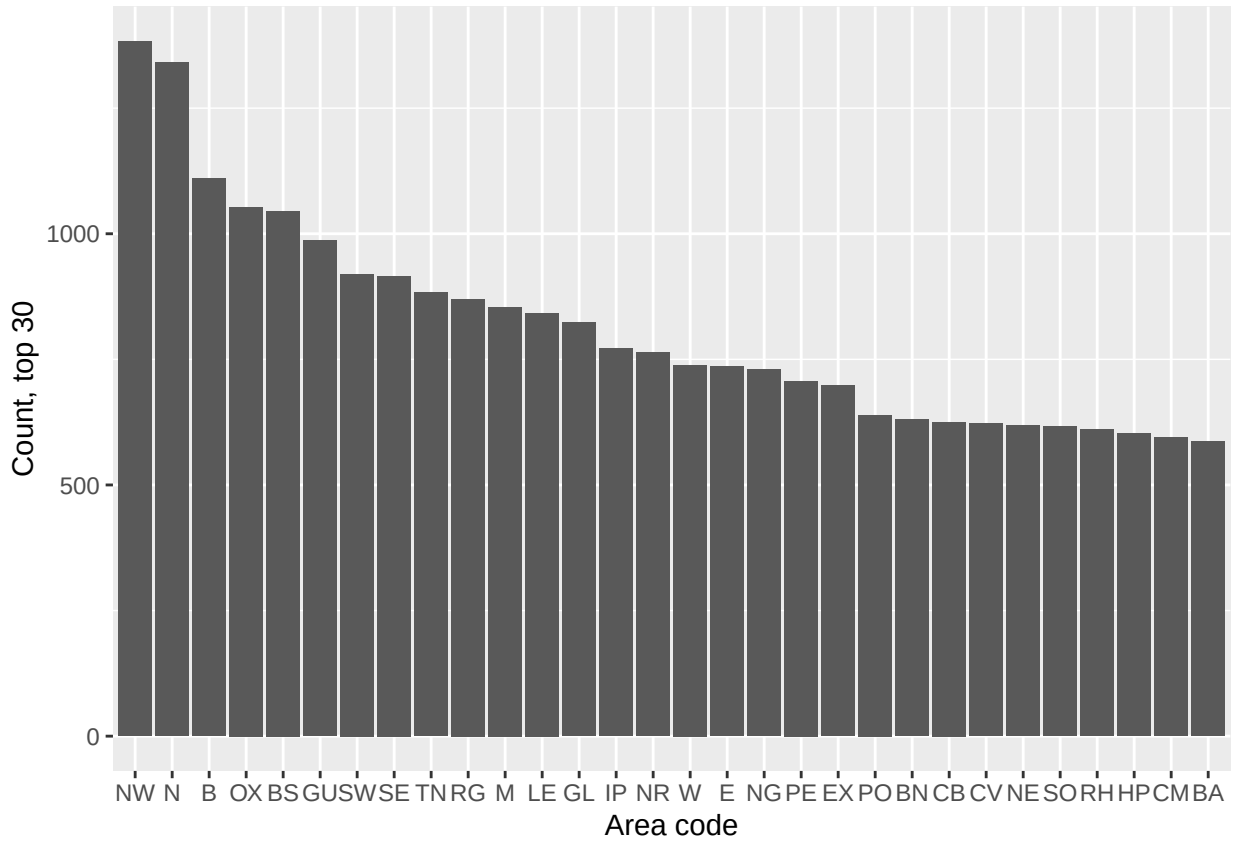
Luckily, the ONS produces a quarterly lat/long lookup of all UK postcodes. `ocode` and `icode` represents the two halves of the postcode. The idea was that if lookup on the whole code failed because the data was dirty, just using the first half, the `ocode`, would suffice for distances. It turned out this split wasn't necessary.

```
ukpostcodes <- read.csv('.\\downloads\\postcodes\\NSPL_MAY_2020_UK.csv')
ukpostcodes <- ukpostcodes %>%
  select(pcd, lat, long) %>%
  mutate(pcd2 = pcd) %>%
  separate(pcd2, c('ocode', 'icode'), sep = -3) %>%
  mutate(pcd = trimws(pcd)) %>%
  mutate(ocode = trimws(ocode)) %>%
  select(pcd, ocode, icode, lat, long) %>%
  mutate_at(vars(pcd, ocode, icode), as.factor)
```

Next, we take the trust postcodes from ewtrust, clean them, and join the lat/long.

```
trustpostcodes <- ewtrusts %>%
  select(regno, subno, postcode) %>%
  mutate(postcode = gsub('!', '1', postcode, fixed = TRUE)) %>%
  mutate(postcode = gsub('"', '2', postcode, fixed = TRUE)) %>%
```

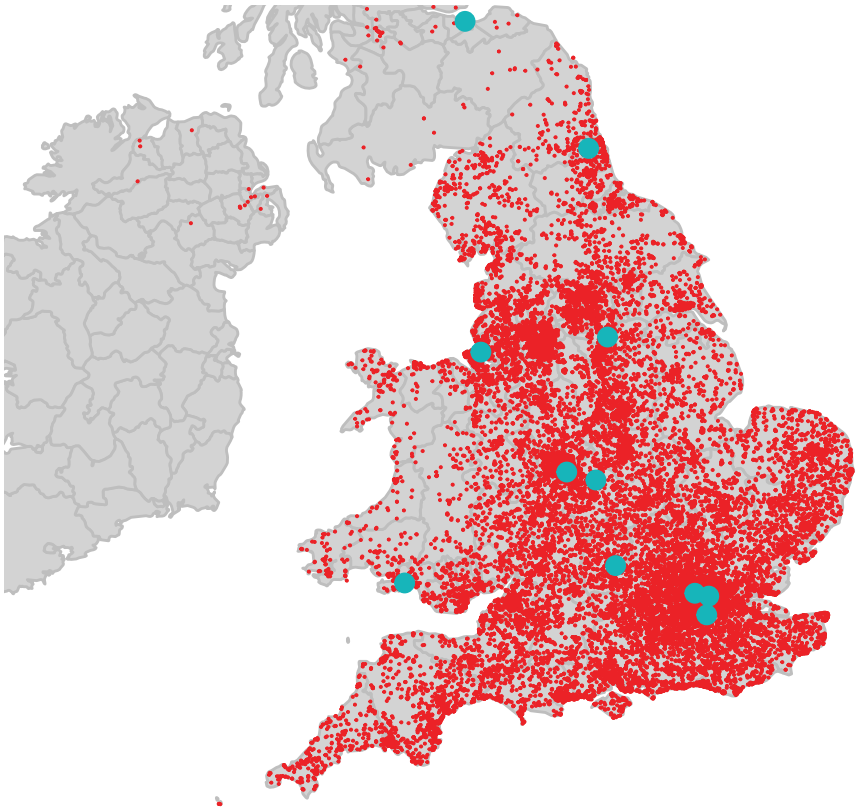
```
  mutate(postcode = gsub('£', '3', postcode, fixed = TRUE)) %>%
  mutate(postcode = gsub('$', '4', postcode, fixed = TRUE)) %>%
  mutate(postcode = gsub('%', '5', postcode, fixed = TRUE)) %>%
  mutate(postcode = gsub('^', '6', postcode, fixed = TRUE)) %>%
  mutate(postcode = gsub('&', '7', postcode, fixed = TRUE)) %>%
  mutate(postcode = gsub('*', '8', postcode, fixed = TRUE)) %>%
  mutate(postcode = gsub('(', '9', postcode, fixed = TRUE)) %>%
  mutate(postcode = gsub(')', '0', postcode, fixed = TRUE)) %>%
  mutate(pcd = postcode) %>%
  separate(pcd, c('ocode', 'icode'), sep = -3) %>%
  mutate(postcode = trimws(postcode)) %>%
  mutate(ocode = trimws(ocode)) %>%
  select(regno, subno, postcode, ocode, icode)

trustpostcodes <- trustpostcodes %>%
  left_join(select(ukpostcodes, ocode, icode, lat, long))

trustpostcodes %>%
  mutate(area = gsub('[[:digit:]]+', '', ocode)) %>%
  filter(!is.na(area)) %>%
  count(area) %>%
  top_n(30, wt = n) %>%
  ggplot(aes(x = reorder(area, -n),
             y = n)) +
  geom_col() +
  labs(x = 'Area code',
       y = 'Count, top 30')
```



Then we build a dataframe of Skylight Centre locations and lat/long.

```
skylightpostcodes <- tibble(
  id = 1:11,
  centre_pc = c('Birmingham', 'Brent', 'Coventry & Warwickshire', 'Croydon',
                'Edinburgh', 'London', 'Merseyside', 'Newcastle',
                'Oxford', 'South Wales', 'South Yorkshire'),
  director_pc = c('Matt Green', 'Atara Fridler', 'Kieran Lyons', 'Tania Marsh',
                  'Grant Campbell', 'James Hickman', 'Steve Harding', 'Andrew Burnip',
                  'Kate Cocker', 'Karen Grunhut', 'Mandy Carlson'),
  postcode = c('B9 4AE', 'NW10 4AJ', 'CV1 4LY', 'CR0 1RG',
                  'EH8 8DT', 'E1 6LT', 'L1 5BD', 'NE1 2AF',
                  'OX1 2AQ', 'SA1 5JQ', 'S63 5BD'),
  ocode = c('B9', 'NW10', 'CV1', 'CR0',
              'EH8', 'E1', 'L1', 'NE1',
              'OX1', 'SA1', 'S63'),
  icode = c('4AE', '4AJ', '4LY', '1RG',
              '8DT', '6LT', '5BD', '2AF',
              '2AQ', '5JQ', '5BD'),
  lat = c(52.4751465, 51.5402633, 52.4136524, 51.3734977,
          55.951425, 51.5181011, 53.4001155, 54.9715911,
          51.7535395, 51.619402, 53.5165338),
  long = c(-1.8850323, -0.2528802, -1.5136714, -0.1013623,
            -3.1806497, -0.0752281, -2.9789889, -1.6060048,
            -1.2621555, -3.950593, -1.3653272)
)
```

Skylight Centres are in blue, trusts are in red.



We then cross join the centre locations with the charity locations to get a very, very big dataframe of about 11m rows. We then compute distance using `distGeo` and convert to km.

```
trustpostcodes_dist <- trustpostcodes %>%
  drop_na() %>%
  full_join(skylightpostcodes, by = character())

trustpostcodes_dist <- trustpostcodes_dist %>%
  filter(long.x != 0) %>%
  rowwise() %>%
  mutate(distance_km = distm(x = c(long.x, lat.x), y = c(long.y, lat.y), fun = distGeo)) %>%
  tibble() %>%
  mutate(distance_km = distance_km / 1000)
```

Finally, we take the closest one for each charity and join it to the propensity frame.

```
trustpostcodes_topdist <- trustpostcodes_dist %>%
  select(regno, subno, centre_pc, director_pc, distance_km) %>%
  group_by(regno, subno) %>%
  top_n(-1, wt = distance_km) %>%
  ungroup()

ewtrusts_prop <- ewtrusts_prop %>%
  left_join(trustpostcodes_topdist)
```

Area of benefit Not all trusts are located where they work. A charity based in Cambridge might help people living in London, for example. Luckily, these areas of benefit are declared, and when the closest Skylight Centre doesn't match our postcode analysis, this will be the preferred method of identifying the best place to invite the trustees to.

Areas of benefit are defined in the database as:

- A, for the whole nation or country
- B, for local authorities
- C, for metropolitan areas
- D, for places outside the UK

We want to suppress all Ds, then use Bs and Cs to identify an appropriate Centre.

First we collect the areas defined as B and C, which are the local areas charities are saying they operate in. We also want the corresponding B and C codes for the Skylight Centres.

```
ew_aoo_c <- runquery(sql = 'select aootype, aookey, aooname, aoosort, master
                            from ew_aoo_ref
                            where aootype = \'C\';')

ew_aoo_b <- runquery(sql = 'select aootype, aookey, aooname, aoosort, master
                            from ew_aoo_ref
                            where aootype = \'B\';')

ew_aoo <- runquery(sql = 'select regno, aootype, aookey, master from ew_charity_aoo
                          where aootype = \'B\' or aootype = \'C\';')

skylight_aoocodes <- tibble(
  id = 1:11,
  centre_aoo = c('Birmingham', 'Brent', 'Coventry & Warwickshire', 'Croydon',
                 'Edinburgh', 'London', 'Merseyside', 'Newcastle',
                 'Oxford', 'South Wales', 'South Yorkshire'),
  director_aoo = c('Matt Green', 'Atara Fridler', 'Kieran Lyons', 'Tania Marsh',
                   'Grant Campbell', 'James Hickman', 'Steve Harding', 'Andrew Burnip',
                   'Kate Cocker', 'Karen Grunhut', 'Mandy Carlson'),
  bcodes = c(64, 10, 65, 15,
             NA, 26, 51, 60,
             126, 164, 57),
  ccodes = c(57, 20, 57, 20,
             NA, 20, 33, 52,
             NA, NA, 48),
)
```

Because we can make the argument that a Skylight Centre in Oxford will help people as far away as, say, Swindon, let's make sure any charity working in a local authority that's part of a metropolitan area that contains a centre gets picked up by that centre.

We'll do this by adding the relevant C reference to any charity with a B reference that's part of a C.

```
ew_aoo <- ew_aoo %>%
  tibble() %>%
  filter(aootype == 'B' & !is.na(master)) %>%
  left_join(ew_aoo_c, by = c('master' = 'aookey')) %>%
```

```
  select(regno, aootype.y, master) %>%
  mutate(aootype = aootype.y, aookey = master) %>%
  select(regno, aootype, aookey) %>%
  mutate(master = NA) %>%
  mutate(master = as.integer(master)) %>%
  rbind(ew_aoo) %>%
  distinct()
```

The next bit was tricky. Basically I slowly filtered down starting with the area of benefits that had an obvious, unambiguous centre associated with them. I then tried to clear up ambiguities in this order of preference:

- **Group A** have an unambiguous centre in their area of operation
  - Group B are charities with more than one
- **Group C** are charities where one of the area of operations matches the closest centre by postcode
  - Group D are what's left over from the above. All have a conflict in the centres in their area of operation
- **Group E** resolves the conflict if it comes from local vs regional, prioritising local
- **Group F** resolves the conflict by choosing the centre closest by postcode (may not be closest overall, like Group C)
- **Group G** is everyone else. Decision tree is London, Birmingham, Newcastle, Merseyside, random. This is based on counts of number of conflicts

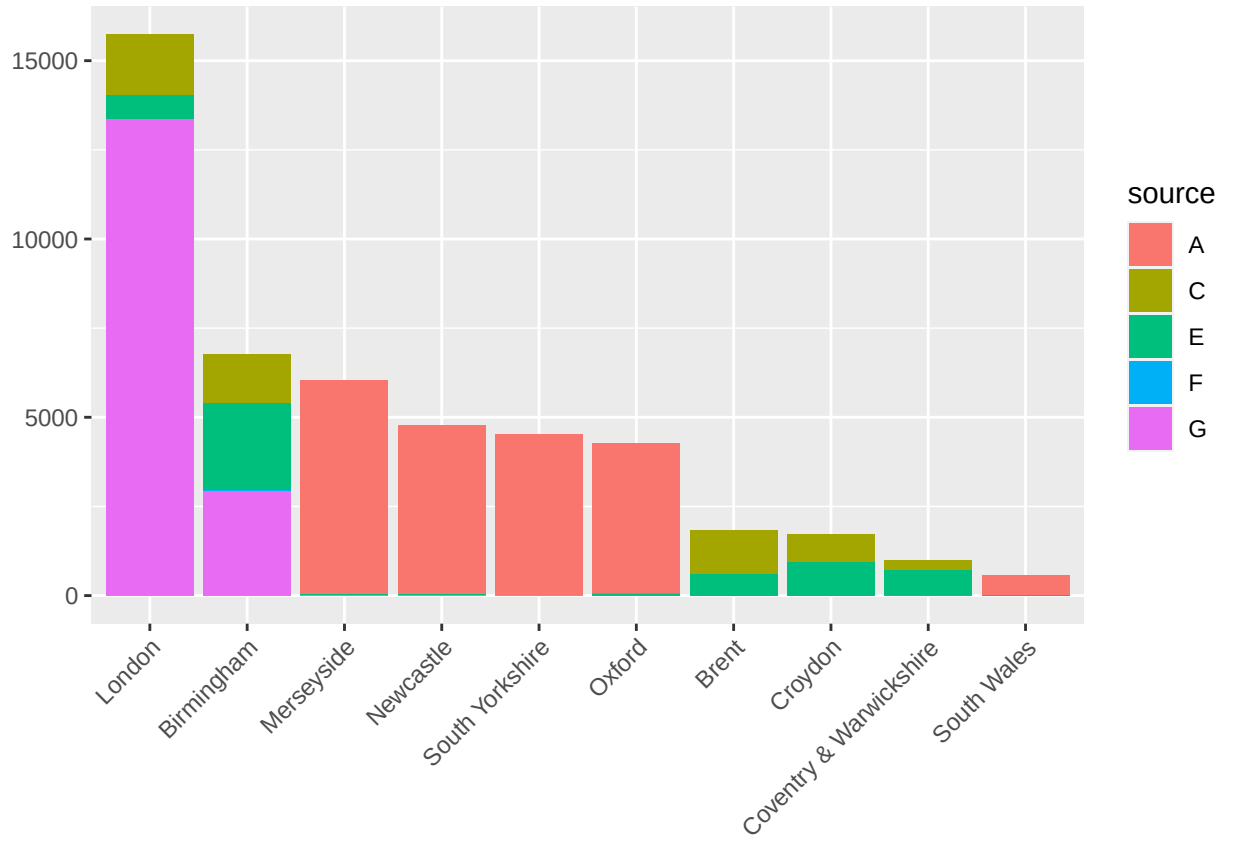
Bold groups are the ones that are will be added to the propensity grid, along with the group they were in. The others are helper groups.

```
ew_aoo_filtered_c <- ew_aoo %>%
  filter(aootype == 'C') %>%
  inner_join(skylight_aoocodes, by = c('aookey' = 'ccodes')) %>%
  select(-bcodes)

ew_aoo_filtered <- ew_aoo %>%
  filter(aootype == 'B') %>%
  inner_join(skylight_aoocodes, by = c('aookey' = 'bcodes')) %>%
  select(-ccodes) %>%
  rbind(ew_aoo_filtered_c)

ew_aoo_filtered_b <- ew_aoo_filtered %>%
  group_by(regno) %>%
  summarise(count = n_distinct(id)) %>%
  filter(count > 1) %>%
  select(regno) %>%
  inner_join(ew_aoo_filtered)

ew_aoo_filtered_a <- ew_aoo_filtered %>%
  anti_join(select(ew_aoo_filtered_b, regno))

ew_aoo_filtered_c <- ew_aoo_filtered_b %>%
  inner_join(ewtrusts_prop, by = c('regno' = 'regno', 'centre_aoo' = 'centre_pc')) %>%
  select(regno, aootype, aookey, master, id, centre_aoo, director_aoo)
```

```
ew_aoo_filtered_d <- ew_aoo_filtered_b %>%
  anti_join(ew_aoo_filtered_c, by = c('regno' = 'regno'))

ew_aoo_filtered_e <- ew_aoo_filtered_d %>%
  filter(aootype == 'B') %>%
  group_by(regno) %>%
  summarise(count = n_distinct(id)) %>%
  filter(count == '1') %>%
  inner_join(filter(ew_aoo_filtered_d, aootype == 'B')) %>%
  select(-count)

ew_aoo_filtered_f <- ew_aoo_filtered_d %>%
  anti_join(ew_aoo_filtered_e, by = c('regno' = 'regno')) %>%
  filter(aootype == 'B') %>%
  left_join(trustpostcodes_dist, by = c('regno' = 'regno', 'id' = 'id')) %>%
  group_by(regno) %>%
  select(regno, aootype, aookey, master, id, centre_aoo, director_aoo, distance_km) %>%
  top_n(-1, wt = distance_km) %>%
  select(-distance_km)

ew_aoo_filtered_g <- ew_aoo_filtered_d %>%
  anti_join(ew_aoo_filtered_e, by = c('regno' = 'regno')) %>%
  anti_join(ew_aoo_filtered_f, by = c('regno' = 'regno')) %>%
  mutate(rank = case_when(centre_aoo == 'London' ~ 1,
                          centre_aoo == 'Birmingham' ~ 2,
                          centre_aoo == 'Newcastle' ~ 3,
                          centre_aoo == 'Merseyside' ~ 4,
                          TRUE ~ 5)) %>%
  group_by(regno) %>%
  top_n(-1, wt = rank) %>%
  distinct(regno, .keep_all = TRUE) %>%
  select(-rank)
```

Joining into one single file to add.

```
ew_aoo_filtered_final <- mutate(ew_aoo_filtered_a, source = 'A') %>%
  rbind(mutate(ew_aoo_filtered_c, source = 'C')) %>%
  rbind(mutate(ew_aoo_filtered_e, source = 'E')) %>%
  rbind(mutate(ew_aoo_filtered_f, source = 'F')) %>%
  rbind(mutate(ew_aoo_filtered_g, source = 'G'))

ggplot(ew_aoo_filtered_final,
       aes(x = fct_infreq(centre_aoo),
           fill = source)) +
  geom_bar() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        axis.title.x = element_blank(),
        axis.title.y = element_blank())
```



Add to propensity dataframe.

```
ewtrusts_prop <- ewtrusts_prop %>%
  left_join(select(ew_aoo_filtered_final, regno, centre_aoo, director_aoo, source))
```

Finally, suppress charities with overseas interests.

I originally tried to accommodate charities with interests both overseas and in the UK, but it caused more problems than it got rid of. In the end it was safer to just remove any charity with an overseas interest, exclusive or otherwise.

```
ew_aoo_suppress <- runquery(sql = 'select regno from ew_charity_aoo
                                   where aootype = \'D\';')

ewtrusts_prop <- ewtrusts_prop %>%
  anti_join(ew_aoo_suppress)
```

4. Mailing file

With our locations and propensities in place, we're ready to produce a single mailing file for the T&F team to use. We need to filter out the charities we definitely don't want, then produce a file with the following columns:

- Name, address and email address of charity

- Expenditure and expenditure segment
- Charitable purpose segment
- Appropriate Skylight Centre to invite to, its distance, and centre signatory

Filtering the selection

Our huge file has lots of charities the T&F team are definitely not going to want to speak to. Here we get rid of:

- Subsidiary charities
- Charities who spent under £25k in 2019
- Charities who don't have a charitable objective of either General, Poverty or Housing
- Charities who are too far from a centre to be likely to support one. This is defined as:
  - Not having a centre in their area of benefit
  - AND having postcode that is more than 50km from their nearest centre

```
ewtrusts_prop_filt <- ewtrusts_prop %>%
  tibble() %>%
  filter(subno == 0) %>%
  filter(!is.na(expendcat)) %>%
  filter(!is.na(asktype)) %>%
  filter(!is.na(centre_aoo) | distance_km < 50) %>%
  distinct()
```

Count of centre vs ask type.

```
## Warning: `...` is not empty.
##
## We detected these problematic arguments:
## * `needs_dots`
##
## These dots only exist to allow future extensions and should be empty.
## Did you misspecify an argument?
```

```
## # A tibble: 11 x 5
##    centre_pc               General Poverty `Poverty & Housing` Housing
##    <chr>                     <int>   <int>                <int>   <int>
##  1 London                      502     414                   87      22
##  2 Brent                       306     292                   52      11
##  3 Merseyside                  169     172                   24       8
##  4 Croydon                     170      95                   17       4
##  5 Birmingham                  127      83                   28       7
##  6 Oxford                      114      77                   17       5
##  7 South Yorkshire             106      73                   19       3
##  8 Coventry & Warwickshire      55      47                   22       6
##  9 Newcastle                    37      51                   13       5
## 10 South Wales                  21       9                    2      NA
## 11 Edinburgh                     2      NA                   NA      NA
```

Count of centre vs charity expenditure segment.

```
## Warning: `...` is not empty.
##
## We detected these problematic arguments:
## * `needs_dots`
##
## These dots only exist to allow future extensions and should be empty.
## Did you misspecify an argument?
```

```
## # A tibble: 11 x 7
##    centre_pc                 med   sml  vsml  hmed   hgh  vhgh
##    <chr>                   <int> <int> <int> <int> <int> <int>
##  1 London                    249   223   198   178    92    85
##  2 Brent                     173   155   166    95    41    31
##  3 Merseyside                 97    77   112    46    25    16
##  4 Croydon                    65    77    87    31    16    10
##  5 Oxford                     43    63    69    22     8     8
##  6 Birmingham                 65    62    61    25    17    15
##  7 South Yorkshire            50    40    62    26    11    12
##  8 Coventry & Warwickshire    35    32    36    11     8     8
##  9 Newcastle                  29    16    22    23     5    11
## 10 South Wales                 7     4    10     5     4     2
## 11 Edinburgh                   1    NA    NA    NA     1    NA
```

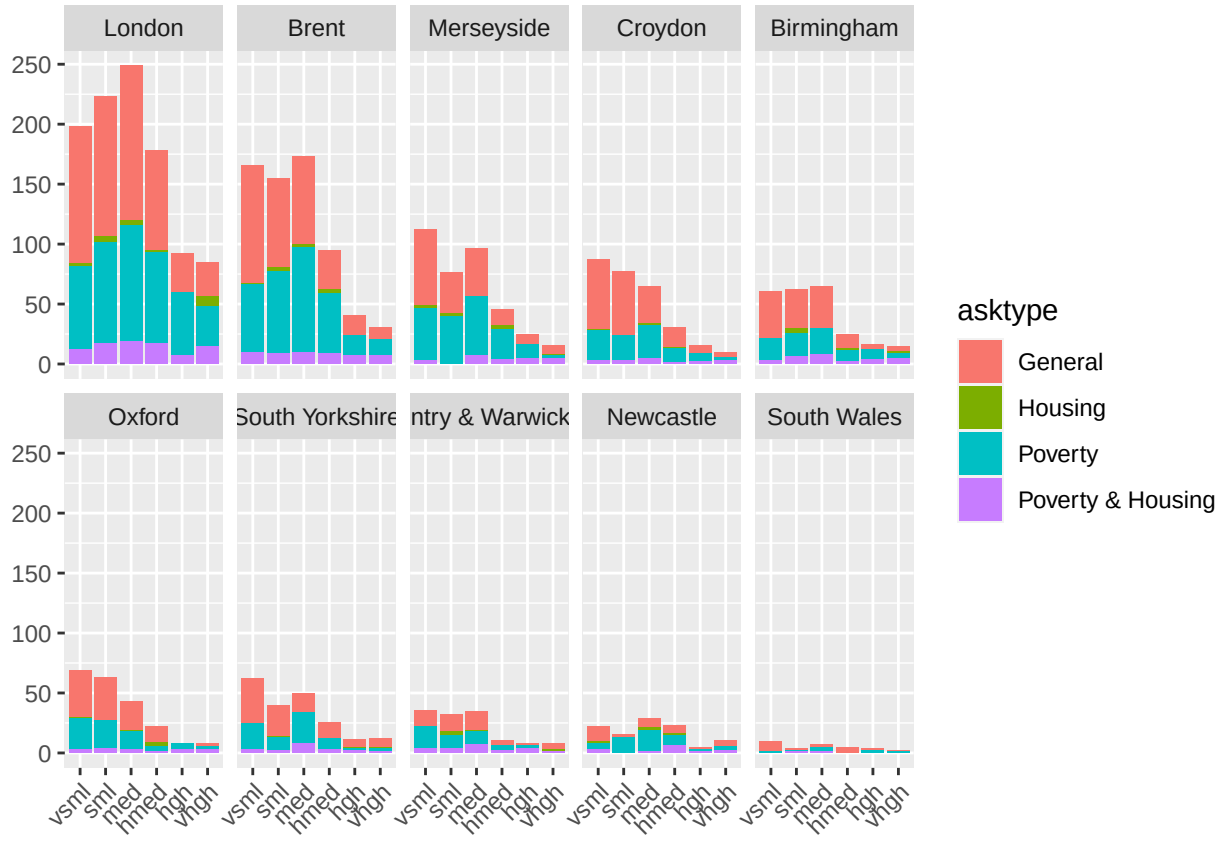
Count of ask type vs charity expenditure segment.

I actually think this is one of the most important plots for the T&F team to look at. Using this, we can begin thinking about who we might cut from the mailing in order to make it as efficient as possible.

```
## Warning: `...` is not empty.
##
## We detected these problematic arguments:
## * `needs_dots`
##
## These dots only exist to allow future extensions and should be empty.
## Did you misspecify an argument?
```

```
## # A tibble: 6 x 5
##   expendcat General Poverty `Poverty & Housing` Housing
##   <fct>       <int>   <int>               <int>   <int>
## 1 vsml          484     286                  44       9
## 2 sml           391     293                  46      19
## 3 med           375     357                  69      13
## 4 hmed          203     202                  45      14
## 5 hgh            83     110                  35       1
## 6 vhgh           74      67                  42      15
```

Finally, a visualisation of all three major propensity variables, showing distribution of charities invited to each centre and how large their expenditure in 2019 was. We could use this to be more discerning in who is invited to London versus Oxford, for example.



Final count is 3,277, though it's likely there'll be plenty of charities the T&F team will want to filter down even further to remove. See next steps for discussion.

Producing the outfile

We're going to create a fresh dataframe and start joining the information we want. Most of it has already been loaded elsewhere, it's just a case of shaping and mixing.

Variables from propensity data We can get `regno`, `name`, expenditure variables, purpose variables and the various Skylight Centres straight from the propensity dataframe.

Sadly names aren't recorded in an appropriate case anywhere in the database – a real issue for things like Crisis UK, where a simple title case conversion will render as 'Crisis Uk'. We do some basic formatting to clear up the most egregious examples. A more general solution is likely possible, but too time consuming right now.

We also get rid of anything in brackets. This is a subjective decision, but from a spot check it makes the names read much more like a human has examined and edited each one.

```
ewtrusts_mailout <- ewtrusts_prop_filt %>%
  select(-subno) %>%
  mutate(name = str_replace(name, '\\(.*\\)', '')) %>%
  mutate(name = str_replace(name, '\\s+', ' ')) %>%
  mutate(name = str_to_title(name)) %>%
  mutate(name = str_replace(name, '\\bUk\\b', 'UK')) %>%
```

```
  mutate(name = str_replace(name, '\\bCio\\b', 'CIO')) %>%
  mutate(name = trimws(name))
```

Address We can very easily get this from our original SQL query. It just needs some quick formatting.

I've chosen to follow the database layout of `add1:add5` plus `postcode`. Reshaping this into one field would be trivial using `unite` if the team needed it. Trickier would be pulling out the mailing town, but the approach I'd take is to lookup postal town on the ONS data using `postcode`, then use `stringr` to remove it from the address fields.

```
ewtrusts_adds <- ewtrusts %>%
  filter(subno == 0) %>%
  select(regno, add1, add2, add3, add4, add5, postcode) %>%
  filter(!is.na(postcode))

ewemails <- runquery(sql = 'select regno, email from ew_main_charity
                            where email is not null;')

ewtrusts_adds <- ewtrusts_adds %>%
  left_join(ewemails)

ewtrusts_adds <- ewtrusts_adds %>%
  mutate(across(add1:add5, str_to_title)) %>%
  mutate(email = str_to_lower(email)) %>%
  mutate(postcode = str_to_upper(postcode)) %>%
  mutate(postcode = gsub('!', '1', postcode, fixed = TRUE)) %>%
  mutate(postcode = gsub('"', '2', postcode, fixed = TRUE)) %>%
  mutate(postcode = gsub('£', '3', postcode, fixed = TRUE)) %>%
  mutate(postcode = gsub('$', '4', postcode, fixed = TRUE)) %>%
  mutate(postcode = gsub('%', '5', postcode, fixed = TRUE)) %>%
  mutate(postcode = gsub('^', '6', postcode, fixed = TRUE)) %>%
  mutate(postcode = gsub('&', '7', postcode, fixed = TRUE)) %>%
  mutate(postcode = gsub('*', '8', postcode, fixed = TRUE)) %>%
  mutate(postcode = gsub('(', '9', postcode, fixed = TRUE)) %>%
  mutate(postcode = gsub(')', '0', postcode, fixed = TRUE))

ewtrusts_mailout <- ewtrusts_mailout %>%
  left_join(ewtrusts_adds)
```

Export to csv With all our variables in order and contact details attached, it's time to turn this into a file the T&F team can use.

The 3.2k outfile has all the information they need to decide how many charities they want to invite, where they might invite them to, who will invite them, and how to target to the charities who are most likely to accept their invitation.

See next steps section (below) for how we might proceed from here.

```
write_csv(ewtrusts_mailout, '.\\analysis\\workbooks\\crisisselection_jul20_v1_outfile.csv')
```

5. Next steps

Next steps would likely be:

- Support the T&F team to understand my segmentation and propensities
- Help the T&F team slim down this selection so it suits their purposes
- Potentially return and calculate new propensities based on these discussions
- Iterate new selections until we're where we need to be

I also had some thoughts on the limitations and implications of this piece of work, which might produce some insight into the direction of the next (fictional) iteration of this selection.

Accuracy of selection and missing data

The elephant in the room with this selection is that I don't have Crisis' fundraising database. This would be a huge, huge boon to accomplishing this task because we'd be able to look at charities who had given in the past, when, how their gift size related to that year's expenditure, etc. We could start consiering some really powerful approaches, like using machine learning to perform a regression analysis that predicts, based on existing gifts, who in the database would likely be worth approaching.

With access to Crisis UK's database, using machine learning to find new prospects would be a very interesting next step.

I think my own selection is also just the start for my fictional T&F team. A 3k count is too high for this purpose, and their domain knowledge on what size of charity expenditure would make a mailing worthwhile would be first step to shrinking this to something manageable. Deciding how to approach the General `asktype` would also cut a lot of the dead weight, especially in smaller charities.

Some work on filtering out religious charities would also either require a better approach to text mining (more on that below), or some strident cutting of any religious charities, which could be done without too much effort.

Lastly, Scotland and Northern Ireland aren't included in the Charity Commission's database, and I haven't yet imported that data into my own version of it.

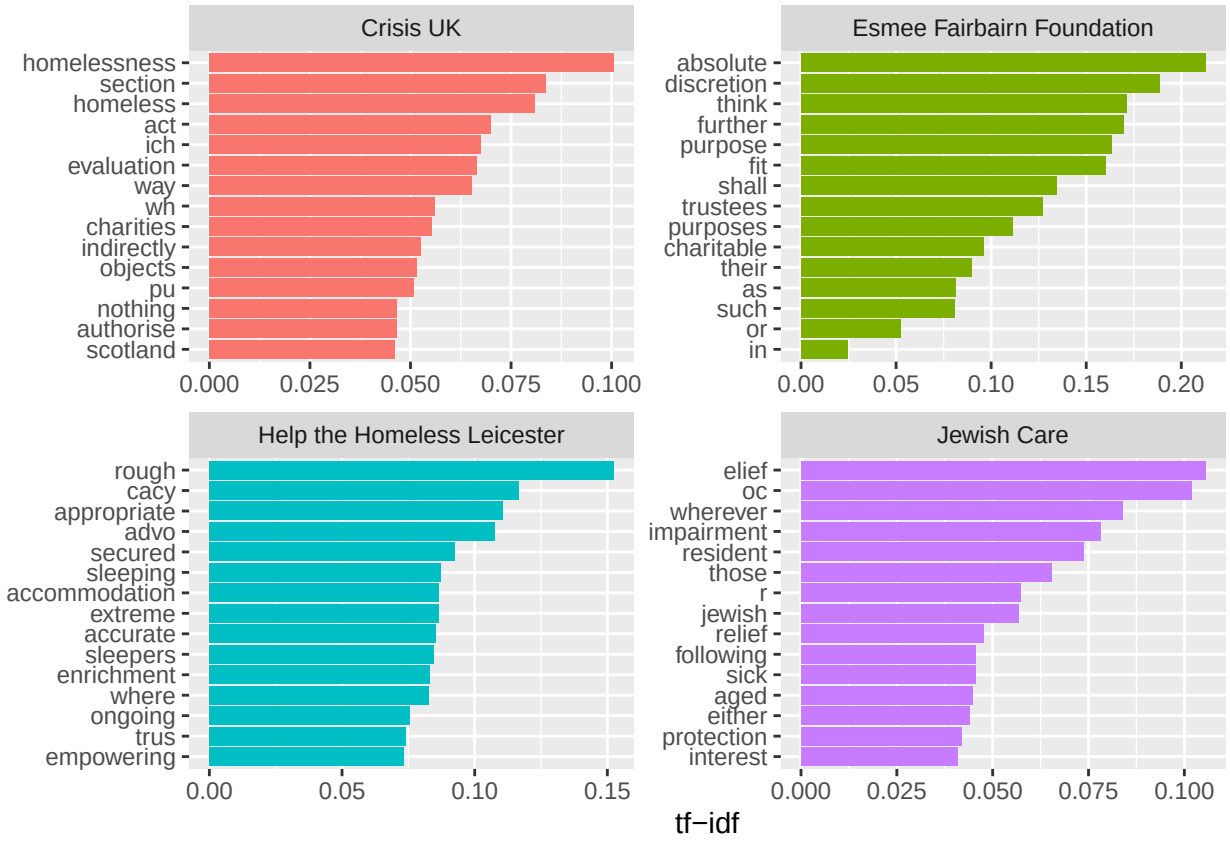
Including Scottish charities in the analysis would be a useful next step.

Text mining and tf-idf

`Tf-idf`, or text frequency-interdocument frequency, is a way of using an overall document corpus to quickly find the words (or n-grams) that make a particular document within that corpus unique. I tried using it to find commonalities between charitable objects. The idea was to help find charities with non-specific aims but with similar values to Crisis UK, without discounting things like religious charities who don't restrict themselves to their own faith. I ran into two problems.

Firstly, the objects corpus resists a straightforward calculation of a useful weighting. There's a lot of words around geography and charitable administration that aren't common enough for the analysis to suppress them, and aren't helpful for our task. They're essentially false positives. Furthermore the documents aren't of a reasonably uniform length, nor are long enough that varying length is no longer an issue.

Here, for example, is the top 15 tf-idf-ranked words for four charities, including Crisis UK. Crisis itself has some strange words creeping in, and other homeless charities like Help the Homeless Leicester don't even have the word homeless in their top 15. Furthermore, charities like Esmee Fairbairn, who are general enough and wealthy enough that they'd probably be worth asking, don't have aims long enough to be useful. Lastly, explicitly religious charities like Jewish Care might match on words around what the charity does, but it's hard to know which religious charities are interested only in supporting other faith charities, and which support wider causes.



Next steps could be creating a more customised, specialised stop file, or looking at n-grams instead of words.

Secondly, the corpus I'd actually like access to, 'aims and activities' (from the 'overview' tab on the Charity Commission website, see Crisis UK as an example) isn't actually in the database. This field is far more clean and uniform and would have been by far my preferred source.

Next steps could be a quick scrape to add this to my database.

Trustees

Because we have the data for trustees, it might be worth creating a network of which trustees existing supporting organisations have in common with other charities that might also be inclined to support. We could even ask the common link to be the letter signatory.

The database unfortunately doesn't contain a correspondent name (even though it's in the data definition), and using trustee data as our 'to' name would potentially increase our open and invite success rate. I didn't do it because it carries risks too – there's nothing in the data to show the Chair, and there's often too many trustees to invite every single one. Getting it wrong is a reputational risk, and a big one when working with this kind of targeted mailing to some wealthy, powerful organisations.

Expenditure

I used a very manual method to segment expenditure, constructing histograms then slicing by informed intuition. Other approaches could have distributed the segments evenly by count, or potentially even a

machine learning-based cluster analysis of the `part-b` table, where different types of income and expenditure are broken down.

One next step could be to perform all three methods, then compare to see which produces the most useful segmentation of the financial data. Domain knowledge from the T&F team would be particularly useful to working this out.

Other languages

There's features in the database to potentially accommodate contacting people in their preferred language if it's Welsh or English. This would likely make a difference to the propensity for charities to support the Skylight Centre in Swansea.

For now, as much I strongly believe in the importance of doing this, is was a layer of complication too far for my analysis.

subno

I realised quite late in the selection that `subno` is basically useless for my analysis, but so much of the code turns itself upside down to accommodate it that I decided it was easier to leave it in – for now. In the data definition half the tables don't even use `subno`, including really important things like area of benefit. It's basically safe to filter on `subno == 0` and strip it, and that would have saved me a lot of hassle.

One next step might be to pull `subno` out of the code entirely. But for now it's there, causing useless `unite` and `separate`s and other fun. You live and learn!

6. Notes

Below are some of the files and resources I used for reference, and to aid replication. Thanks very much for reading this far. Please feel free to get in touch at wpflangdale@gmail.com or visit my GitHub.

- Charity Commission's database export, including files and schemas
- Charity Commission database definitions
- Crisis UK on the database
- `tidytext` documentation, for using td-idf