Using Land Registry Data to Estimate Market Value of Property by Area

This analysis uses the publicly available data from the Land Registry to estimate a reasonable market value for each property type in each area. The conclusion is that the data are sufficient to set fairly broad, but useful, boundaries for transactions at 'fair market value' in most areas.

Data Preparation

To begin with we load the libraries required for the analysis.

```
library(tidyr)
library(dplyr)
library(data.table)
library(readr)
library(purrr)
library(lubridate)
library(stringr)
```

Now we create a function that will check whether the price paid data are present locally. If so it will read that into memory as `full_data`. If not it will download the price paid data and create the `full_data` object.

```
# Function to download (if necessary) price paid data and prepare for analysis
create_full_data <- function() {

    make_url <- function(year) {
        paste0("http://prod.publicdata.landregistry.gov.uk.s3-website-eu-west-1.amazonaws.com/pp-",
               year,
               ".txt")
    }

    # Set years parameter to allow for easily changing the scope of the data we get
    years <- seq(2014, 2017)

    # Create the urls and download the data
    urls <- map_chr(years, make_url)
    if(!dir.exists("data/")) {
        dir.create("data/")
    }
    setwd("data")
    walk(urls, function(x) {
        if(!file.exists(basename(x))) {
            download.file(x, basename(x), method = "wget")
        }
    })

    # Generate form for the filenames
    make_filename <- function(year) {
        paste("pp-",
              year,
              ".txt",
```

```r
             sep = "")
    }

    filenames <- basename(urls)

    # Read the files into memory, then bind the data frames by row and delete the
    # list of data frames to save memory
    datalist <- map(filenames,
                    function(x) fread(x,
                                      sep = ",",
                                      header = F,
                                      col.names = c('tuid', 'price',
                                                    'date_of_transfer',
                                                    'postcde', 'prop_typ',
                                                    'old_new', 'duration',
                                                    'paon', 'saon', 'street',
                                                    'locality', 'town',
                                                    'district', 'county',
                                                    'ppd_type',
                                                    'rec_status')))

    full_data <- rbindlist(datalist)

    rm(datalist)

    # Clean up the data: change certain columns to factors, set names for columns,
    # and reorder them.
    full_data[
        ,
        prop_typ := as.factor(prop_typ)
        ][
            ,
            c("outcde", "incde") := tstrsplit(postcde, " ", fixed = TRUE)
        ][
            ,
            date_of_transfer := as_date(date_of_transfer)
        ][
            ,
            year := year(date_of_transfer)
        ][
            ,
            tax_year := ifelse(month(date_of_transfer) >= 4 & day(date_of_transfer) >= 6,
                               year + 1,
                               year)
        ]

    # Write out full_data to rds to save reprocessing
    setwd('..')
    saveRDS(full_data, './data/full_data.rds')
}
```

The next step is to run that function, checking first that the required object is not already in memory.

NB. These extra steps are to prevent unnecessary downloading of the data or creation of the data objects, as they are rather large.

```
if(!exists('full_data')) {
    ifelse(file.exists('data/full_data.rds'),
           invisible(full_data <- as.data.table(readRDS('data/full_data.rds'))),
           create_full_data())
}
```

Data Munging

Now apply grouping and summarising functions to the data to get what we need: entries for each combination of year, property type, and postcode area/outcode (e.g. M33, UB6).

```
# Group the data by outcode, year, and property type, then summarise with a few
# key stats
if(!exists('by_outcde_yr_typ')) {
    ifelse(file.exists('/data/by_outcde_yr_typ.rds'),
           by_outcde_yr_typ <- as.data.table(readRDS('/data/by_outcde_yr_typ.rds')),
           by_outcde_yr_typ <- full_data[,
                                         .(.N,
                                           avg_price = mean(price),
                                           median = quantile(price, .50),
                                           sd = sd(price),
                                           q25 = quantile(price, .25)),
                                         keyby = .(outcde, prop_typ, tax_year)][
                                             tax_year %in% 2015:2017
                                         ])
}
```

Save the summarised table to file to speed up the process when re-running.

```
# Write out the data.table to rds
if(!file.exists('data/by_outcde_yr_typ.rds')) {
    saveRDS(by_outcde_yr_typ, 'data/by_outcde_yr_typ.rds')
}
```

Run a quick check for data quality.

```
# Check how many groups have at least 10 data points
paste0(round((100 * nrow(by_outcde_yr_typ %>%
                     filter(N > 10)) / nrow(by_outcde_yr_typ))),
       "% of outcode, year, property type groups have at least 10 data points.")
```

```
## [1] "80% of outcode, year, property type groups have at least 10 data points."
```

The next step is to compare our list to the reference list of postcode areas, to ensure that we have entries for every possible combination (since any groups with no transactions will be missing from our list).

In this analysis the postcode lists have already been downloaded from the Ordnance Survey in `.csv` format, and stored in a directory called `OS_data`.

```
# Combine the Ordnance Survey postcode list csv files to get full list of all
# UK outcodes.
fils <- list.files('./data/OS_data',
                   pattern = '.csv$',
                   full.names = TRUE)

pcdes <- rbindlist(lapply(fils,
```

```
                          fread))[,
                                  .(pcde = paste(str_extract(V1,
                                                             '^[A-z]{1,2}\\d{1,2}[A-z]?'),
                                                 str_extract(V1,
                                                             '\\d[A-z]{2}$')),
                                    outcde = str_extract(V1,
                                                         '^[A-z]{1,2}\\d{1,2}[A-z]?'),
                                    incde = str_extract(V1,
                                                        '\\d[A-z]{2}$'))]
```

Now test that the postcodes all fit the required format.

```
# Test whether all pcdes have required format
min(str_detect(pcdes$pcde, "[A-z]{1,2}\\d{1,2}[A-z]? \\d[A-z]{2}")) == 1
```

```
## [1] TRUE
```

```
# Test whether all outcdes have required format
min(str_detect(pcdes$outcde, "[A-z]{1,2}\\d{1,2}[A-z]?")) == 1
```

```
## [1] TRUE
```

We only need the outcodes, so we can reduce the size of the object dramatically. Importantly we add a row for a blank string, to match to those sales in the price paid data with no geographical location.

Then we cross join that table with all possible years and property types.

```
# Drop cols and duplicates from pcdes to save memory
outcdes <- rbindlist(list(list(''),
                          unique(pcdes[, .(outcde)])))

# Prepare all_outcdes for binding with year and type data
outcdes_yrs_typs <- CJ(outcde = outcdes$outcde,
                       tax_year = 2014:2017,
                       prop_typ = c('D', 'F', 'O', 'S', 'T'))
```

We now join the outcodes to the price paid data, and add a logical column to flag Scottish postcodes. Due to the separation of Scottish property taxes we will remove those areas from the analysis.

The two `sapply()` operations at the end of this section test for missing values in the combined dataset.

```
# Left Join full list of outcodes with the price paid data, see which have no
# data. Add logical for Scottish pcdes.
scot_area_cdes <- c('AB', 'DD', 'DG', 'EH', 'FK', 'G', 'HS', 'IV',
                    'KA', 'KW', 'KY', 'ML', 'PA', 'PH', 'TD', 'ZE')
scot_outcde_regx <- paste0('^',
                           scot_area_cdes,
                           '[0-9]{1,2}',
                           collapse = '|')
setkey(outcdes_yrs_typs, outcde, prop_typ, tax_year)
setkey(by_outcde_yr_typ, outcde, prop_typ, tax_year)
all_ppdata <- merge(outcdes_yrs_typs, by_outcde_yr_typ, all.x = T)[
    ,
    scottish := (grepl(scot_outcde_regx, outcde))
]
sapply(all_ppdata, function(x) {max(is.na(x))})
```

```
##    outcde  prop_typ  tax_year         N avg_price    median        sd
```

```
##         0         0         0         1         1         1         1
##       q25  scottish
##         1         0
```

```r
sapply(all_ppdata, function(x) {sum(is.na(x))})
```

```
##    outcde  prop_typ  tax_year         N avg_price    median        sd
##         0         0         0     24677     24677     24677     25531
##       q25  scottish
##     24677         0
```

Now we separate the Scottish data, leaving an object `main_ppdata` that we can use for analysis. We also impute zeroes for the `NA` entries in the `N` field, which will make our analysis easier.

```r
# Separate Scottish data
main_ppdata <- all_ppdata[scottish == FALSE]
main_ppdata$N[is.na(main_ppdata$N)] <- 0L
scot_ppdata <- all_ppdata[scottish == TRUE]
```

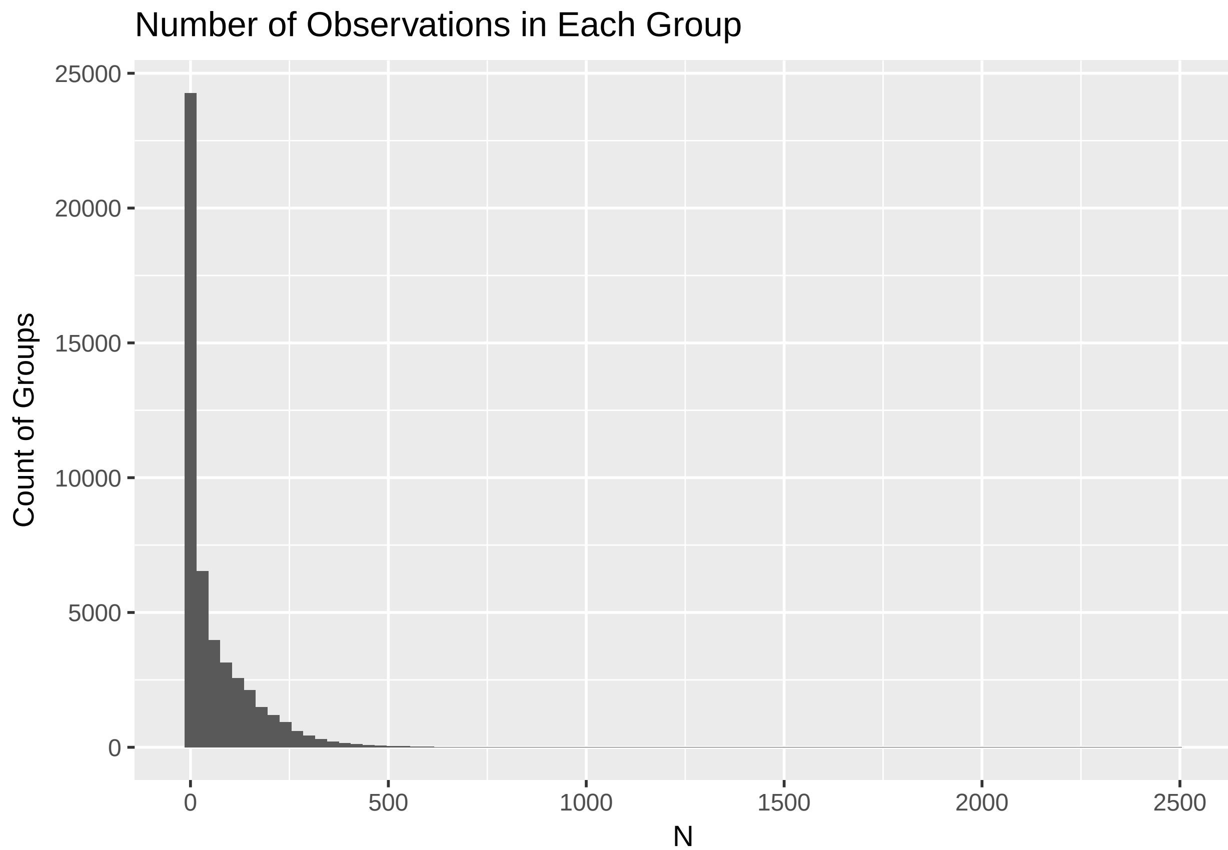
Analysis

Data Quantity

An important question to answer is: for which outcode-year-type groups do we have sufficient data to be able to draw reliable inferences?

We can visualise this to begin with.

```r
library(ggplot2)
library(ggthemes)
library(hrbrthemes)
extrafont::loadfonts()
ggplot(main_ppdata, aes(N)) +
    geom_histogram(binwidth = 30) +
    # theme_ipsum() +
    ggtitle("Number of Observations in Each Group") +
    labs(x = "N", y = "Count of Groups")
```



This doesn't look promising, but there is a long tail here that may not be captured clearly in the histogram.

```
main_ppdata[,
            .N,
            keyby = .(low_data = (N < 10 & !is.na(N)),
                      no_data = (N == 0))
            ]
```

```
##    low_data no_data     N
## 1:    FALSE   FALSE 26787
## 2:     TRUE   FALSE  6126
## 3:     TRUE    TRUE 15607
```

So of the 48520 groups in our dataset, 21733 have 1-9 observations; 0 have none. That means 26787 of the groups have at least 10 observations, or 55%.

This is a little better, but seems to imply that the usefulness of our data will be confined by less than three-quarters of the country. However this does not take account of the clustering of property transactions. What proportion of property transactions occur in groups with 10 or more observations?

```
main_ppdata[,
            .(total_transactions = sum(N)),
            by = .(N >= 10)]
```

```
##    N >= 10 total_transactions
## 1:   FALSE              27978
## 2:    TRUE            2932164
```

This is more promising: only 1% of property transactions in the tax years 2014-2017 took place in groups with fewer than 10 data points. We can test this with a higher threshold for 'good data' of n = 20.

```r
main_ppdata[,
            .(total_transactions = sum(N)),
            by = .(N >= 20)]
```

```
##    N >= 20 total_transactions
## 1:   FALSE              82005
## 2:    TRUE            2878137
```

3% of transactions are in groups that fall below this increased threshold. We can also check whether this remains consistent year-on-year.

```r
(data_totals <- main_ppdata[,
                            .(total_transactions = sum(N)),
                            by = tax_year])
```

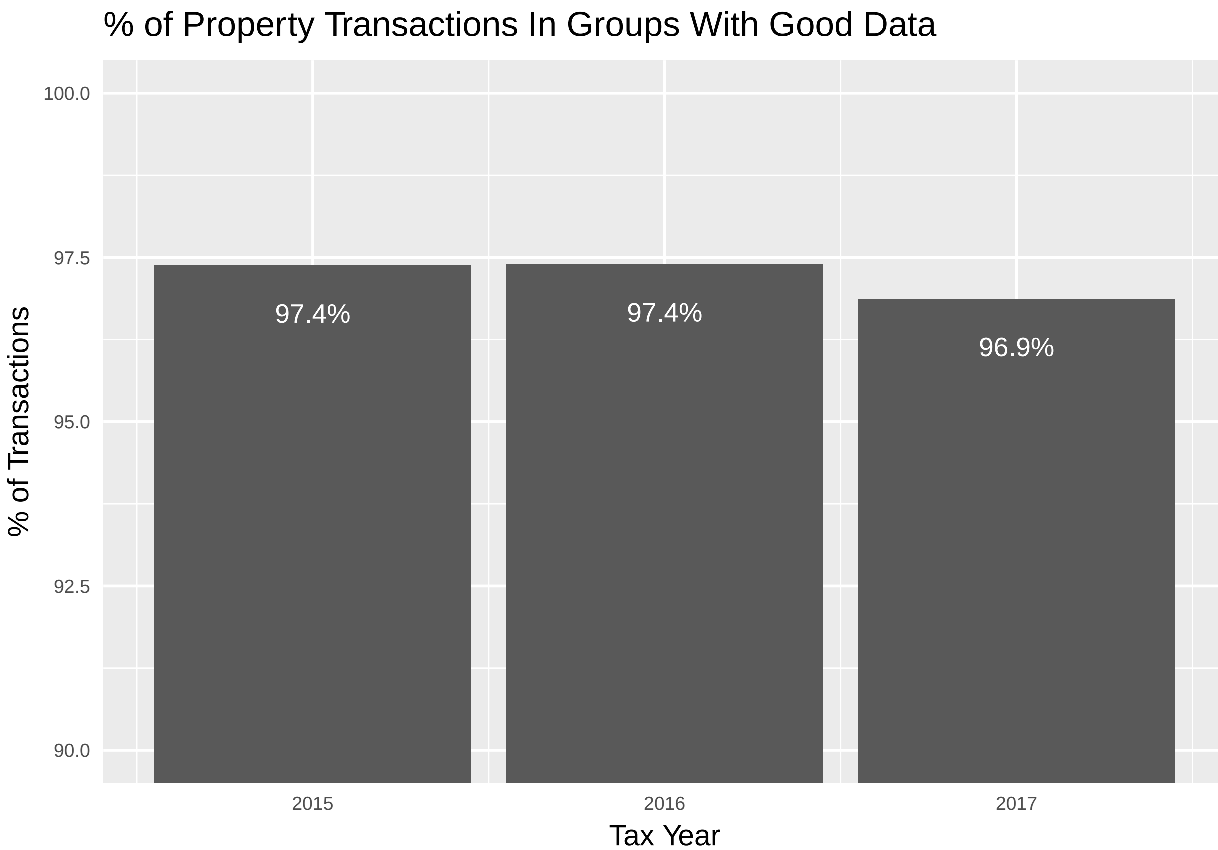
```
##    tax_year total_transactions
## 1:     2014                  0
## 2:     2015             968470
## 3:     2016            1078655
## 4:     2017             913017
```

```r
(good_data_totals <- main_ppdata[,
                                good_data := (N >= 20)][
                                    ,
                                    .(transactions = sum(N)),
                                    by = .(tax_year, good_data)
                                    ][
                                        good_data == TRUE
                                        ])
```

```
##    tax_year good_data transactions
## 1:     2015      TRUE       943097
## 2:     2016      TRUE      1050588
## 3:     2017      TRUE       884452
```

```r
setkey(good_data_totals, tax_year)
setkey(data_totals, tax_year)

data_totals[good_data_totals][,
                              .(good_data_pct = 100 * (transactions / total_transactions)),
                              by = tax_year] %>%
    ggplot(., aes(tax_year, good_data_pct)) +
    geom_bar(stat = 'identity') +
    coord_cartesian(ylim = c(90, 100)) +
    geom_text(aes(label = paste0(round(good_data_pct, 1), '%')),
              vjust = 3,
              colour = 'white',
              size = 3.5) +
    ggtitle("% of Property Transactions In Groups With Good Data") +
    labs(x = "Tax Year", y = "% of Transactions") +
    #theme_ipsum() +
    theme(plot.title=element_text(hjust=0)) +
    theme(axis.ticks=element_blank()) +
    theme(axis.text=element_text(size=7))
```



% of Property Transactions In Groups With Good Data
% of Transactions
100.0
97.5
95.0
92.5
90.0
97.4%
97.4%
96.9%
2015
2016
2017
Tax Year