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

Hamed Bastan-Hagh

Contents

1 Overview **1**

2 Pre-Analysis **1**
2.1 Data Loading and Preparation 1
2.2 Data Munging 3

3 Property Price By Type and Location **4**
3.1 Categorical Variables 4
3.2 Data Completeness 5
3.3 Dispersion of Price Paid Data 10
3.4 Creating the Output Product 12

1 Overview

This analysis uses the publicly available price paid data from the Land Registry to determine whether it is possible 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. A more useful approach may be to consider changes in property value, and to use comparisons of actual to expected price changes as a risk marker. Each of these approaches is summarised below.

Further work will be done to analyse the distribution of house prices, and whether this allows for useful application of the expected values and variance in any profiling applications.

2 Pre-Analysis

2.1 Data Loading and Preparation

To begin with we load the libraries required.

```r
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. The contents of this function show how the data will be structured (e.g. variable names and types).

```
create_full_data_all_years <- function() {
    make_url <- function(year) {
        paste0("http://prod2.publicdata.landregistry.gov.uk.s3-website-eu-west-1.amazonaws.com/pp-",
               year,
               ".txt")
    }

    years <- seq(1995, 2017)

    urls <- map_chr(years, make_url)
    if(!dir.exists("data/")) {
        dir.create("data/")
    }
    setwd("data")
    walk(urls, function(x) {
        if(!file.exists(paste0("data/", basename(x)))) {
            download.file(x, paste0("data/", basename(x)), method = "curl")
        }
    })

    make_filepaths <- function(year) {
        paste0("data/pp-",
               year,
               ".txt")
    }

    filepaths <- make_filepaths(years)

    datalist <- map(filepaths,
                    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)

    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 := case_when(
                            month(date_of_transfer) > 4 ~ year + 1,
                            month(date_of_transfer) < 4 ~ year,
                            day(date_of_transfer) >= 6  ~ year + 1,
                            TRUE                        ~ year)
                        ]

    write_rds(full_data, './data/full_data_all_years.rds')

    rm(datalist)
}
```

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. This step will take some time even if the `.rds` file is saved locally.

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

2.2 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_all_yrs.rds'),
           by_outcde_yr_typ <- read_rds('data/by_outcde_yr_typ_all_yrs.rds'),
           by_outcde_yr_typ <- full_data[,
                                         .(.N,
                                           avg_price = mean(price),
                                           median = quantile(price, .50),
                                           sd = sd(price),
                                           q05 = quantile(price, .05),
                                           q10 = quantile(price, .10),
                                           q15 = quantile(price, .15),
                                           q20 = quantile(price, .20),
                                           q25 = quantile(price, .25),
                                           q30 = quantile(price, .30),
                                           q35 = quantile(price, .35),
                                           q40 = quantile(price, .40),
                                           q45 = quantile(price, .45),
                                           q55 = quantile(price, .55),
                                           q60 = quantile(price, .60),
                                           q65 = quantile(price, .65),
                                           q70 = quantile(price, .70),
                                           q75 = quantile(price, .75),
                                           q80 = quantile(price, .80),
                                           q85 = quantile(price, .85),
                                           q90 = quantile(price, .90),
                                           q95 = quantile(price, .95)),
                                         keyby = .(outcde, prop_typ, tax_year)])
}
```

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_all_yrs.rds')) {
    write_rds(by_outcde_yr_typ, 'data/by_outcde_yr_typ_all_yrs.rds')
}
```

3 Property Price By Type and Location

Our first aim is to determine whether we have sufficient data to give a meaningful expected value for a given property transaction. There are at least three elements to consider with this:

- What categorical variables are available to us for segmenting the data?
- Do we have sufficient observations in each of those segments for reliable inference?
- Does the dispersion of the data points in each segment allow us to set reasonable error bounds for the expected value?

Each will be addressed in turn.

3.1 Categorical Variables

Let's consider the variables in the full price paid data.

```
str(full_data)
```

```
## Classes 'data.table' and 'data.frame':   22969808 obs. of  20 variables:
##  $ tuid            : chr  "{5BBE9CB3-6332-4EB0-9CD3-8737CEA4A65A}" "{20E2441A-0F16-
49AB-97D4-8737E62A5D93}" "{D893EE64-4464-44B5-B01B-8E62403ED83C}" "{F9F753A8-E56A-4ECC-
9927-8E626A471A92}" ...
##  $ price           : int  42000 95000 74950 43500 63000 29995 105000 121250 43000 128500 ...
##  $ date_of_transfer: Date, format: "1995-12-21" "1995-03-03" ...
##  $ postcde         : chr  "NE4 9DN" "RM16 4UR" "CW10 9ES" "TS23 3LA" ...
##  $ prop_typ        : Factor w/ 5 levels "D","F","O","S",..: 4 4 1 4 4 4 1 2 5 5 ...
##  $ old_new         : chr  "N" "N" "Y" "N" ...
##  $ duration        : chr  "F" "F" "F" "F" ...
##  $ paon            : chr  "8" "30" "15" "19" ...
##  $ saon            : chr  "" "" "" "" ...
##  $ street          : chr  "MATFEN PLACE" "HEATH ROAD" "SHROPSHIRE CLOSE" "SLEDMERE CLOSE" ...
##  $ locality        : chr  "FENHAM" "GRAYS" "MIDDLEWICH" "BILLINGHAM" ...
##  $ town            : chr  "NEWCASTLE UPON TYNE" "GRAYS" "MIDDLEWICH" "BILLINGHAM" ...
##  $ district        : chr  "NEWCASTLE UPON TYNE" "THURROCK" "CONGLETON" "STOCKTON-ON-
TEES" ...
##  $ county          : chr  "TYNE AND WEAR" "THURROCK" "CHESHIRE" "STOCKTON-ON-TEES" ...
##  $ ppd_type        : chr  "A" "A" "A" "A" ...
##  $ rec_status      : chr  "A" "A" "A" "A" ...
##  $ outcde          : chr  "NE4" "RM16" "CW10" "TS23" ...
##  $ incde           : chr  "9DN" "4UR" "9ES" "3LA" ...
##  $ year            : num  1995 1995 1995 1995 1995 ...
##  $ tax_year        : num  1996 1995 1996 1996 1996 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

For this analysis we will use three variables to segment the data:

- **postcode area** (or outcode);
- **property type**: Detached, Flat, Semi-Detached, Terraced, Other;
- **tax_year**.

It might be possible to do a more detailed analysis of location using geo data, based on the full postcode, but for the purposes of this work we will cluster properties based on their postcode area and consider it as a discrete variable.

Segmenting by larger areas (e.g. outcodes starting with "M" or "SN") seems unlikely to give useful data as there is so much variation within these.

3.2 Data Completeness

We have our three categories for the analysis, so we start by checking what proportion of the tax year-property type-outcode groups have a reasonable number of observations, say 10.

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

```
## [1] "84% of outcode, year, property type groups have at least 10 data points."
```

However we cannot be sure that every possible combination of tax year-property type-outcode is present in our data. So we compare our list to the reference list of postcode areas from the Ordnance Survey, to ensure that we have entries for every possible combination.

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`. These data are available from the Ordnance Survey's Open Data website for free download.

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

# 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 = 1995:2018,
                       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 main 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 <- by_outcde_yr_typ[outcdes_yrs_typs][
    ,
    scottish := (grepl(scot_outcde_regx, outcde))
]

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

```
##       outcde     prop_typ     tax_year            N    avg_price
##            0            0            0            1            1
## median_price  price_stdev    price_var          q05          q10
##            1            1            1            1            1
##          q15          q20          q25          q30          q35
##            1            1            1            1            1
##          q40          q45          q55          q60          q65
##            1            1            1            1            1
##          q70          q75          q80          q85          q90
##            1            1            1            1            1
##          q95     scottish
##            1            0
```

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

```
##       outcde     prop_typ     tax_year            N    avg_price
##            0            0            0       124115       124115
## median_price  price_stdev    price_var          q05          q10
##       124115       129728       129728       124115       124115
##          q15          q20          q25          q30          q35
##       124115       124115       124115       124115       124115
##          q40          q45          q55          q60          q65
##       124115       124115       124115       124115       124115
##          q70          q75          q80          q85          q90
##       124115       124115       124115       124115       124115
##          q95     scottish
##       124115            0
```

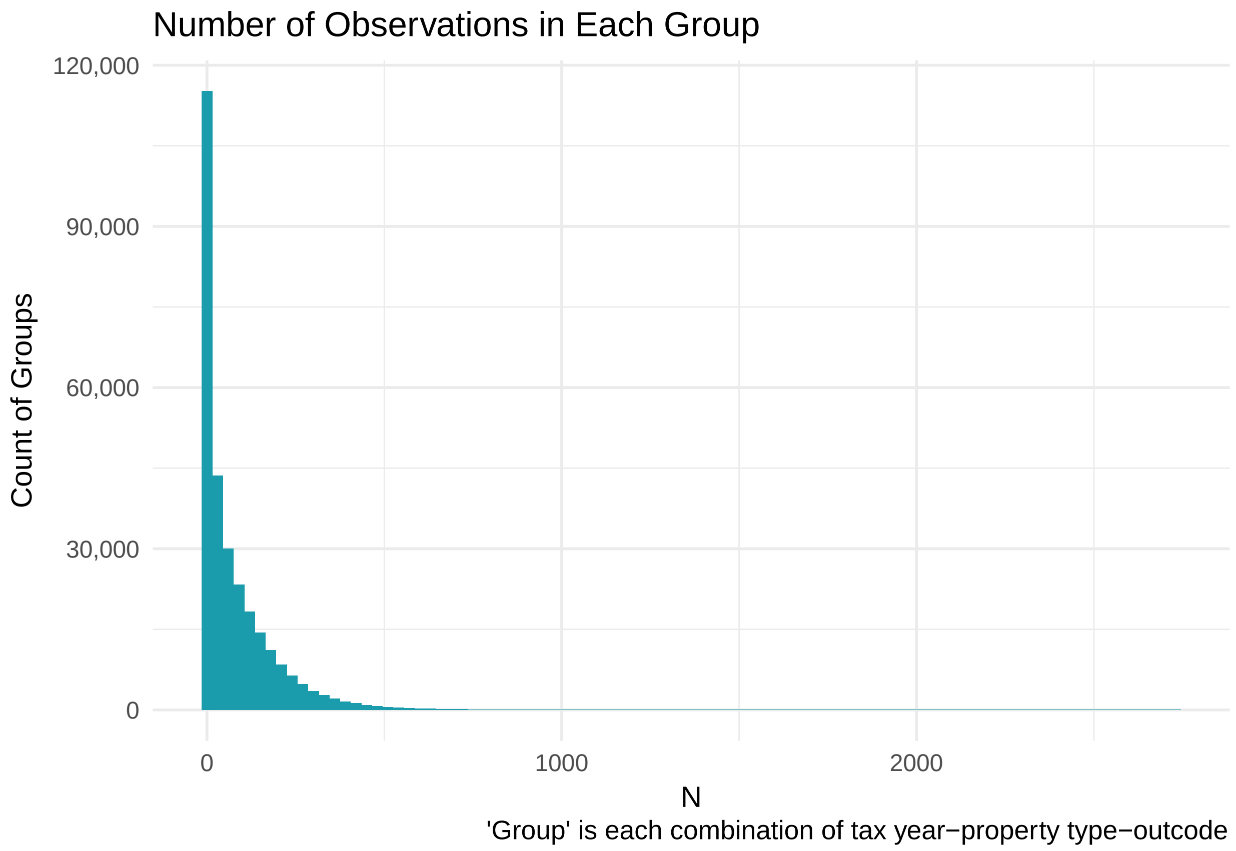
Now we separate the Scottish data, leaving an object `main_ppdata` that we can use for analysis. We also impute zeroes for the `NA` entries in the `N` field, which will make our analysis easier.

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

For which outcode-year-type groups do we have sufficient data for reliable inferences?

We can visualise this to begin with.

```
library(ggplot2)
library(ggthemes)
library(scales)
ggplot(main_ppdata, aes(N)) +
    geom_histogram(binwidth = 30, fill = "#1b9cad") +
    theme_minimal() +
    ggtitle("Number of Observations in Each Group") +
    labs(x = "N", y = "Count of Groups",
         caption = "'Group' is each combination of tax year-property type-outcode") +
    scale_y_continuous(label = comma)
```



```
    scale_x_continuous(breaks = seq(500, 2500, by = 500))
```

```
## <ScaleContinuousPosition>
##  Range:
##  Limits:    0 --    1
```

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 189530
## 2:     TRUE   FALSE  31866
## 3:     TRUE    TRUE  69724
```

So of the 291120 groups in our dataset, 101590 have 1-9 observations; 0 have none. That means 189530 of the groups have at least 10 observations, or 65%.

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             139555
## 2:    TRUE           22792924
```

This is more promising: only 1% of property transactions in the tax years 1995-2017 took place in groups with fewer than 10 data points. We can test this with a higher threshold for 'good data' of n = 20.

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

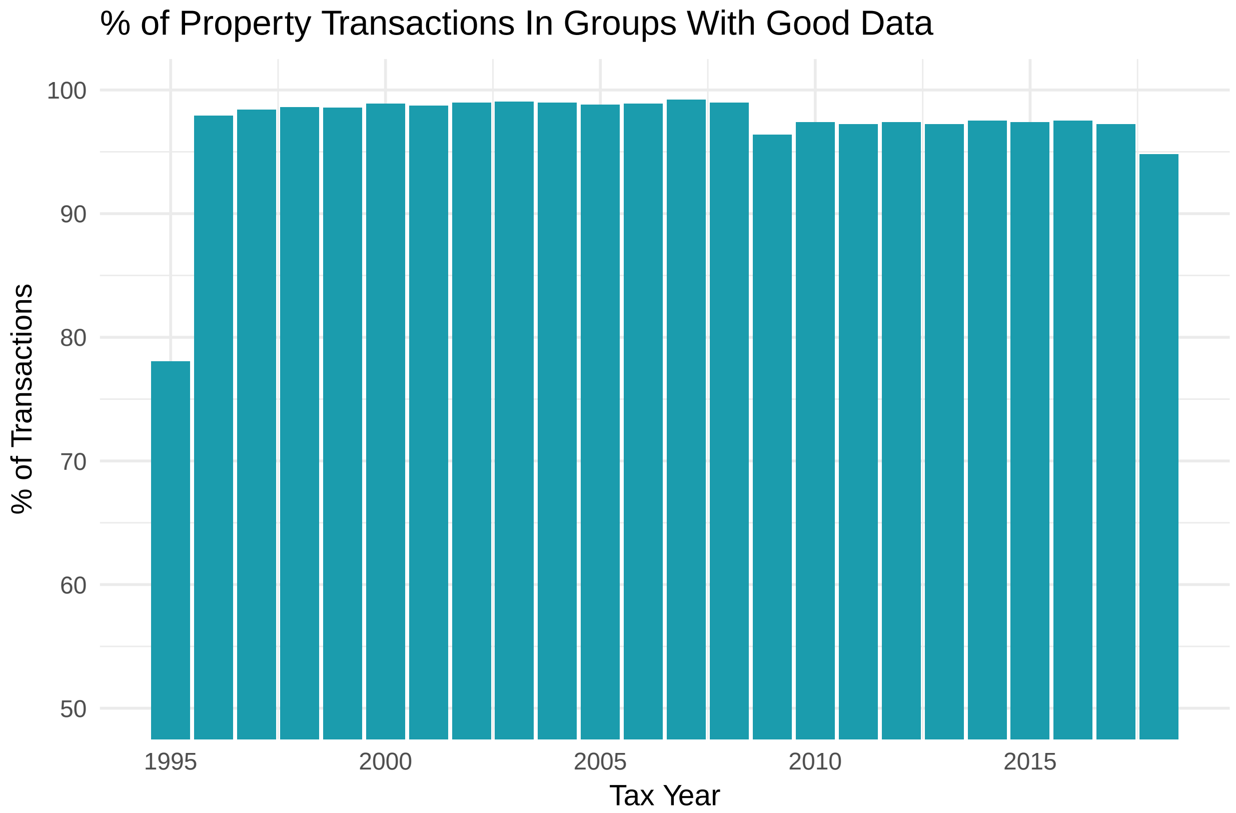
```
##    N >= 20 total_transactions
## 1:   FALSE             443340
## 2:    TRUE           22489139
```

2% of transactions are in groups that fall below this increased threshold. We can also check whether this remains consistent year-on-year.

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

good_data_totals <- main_ppdata[,
                                good_data := (N >= 20)][
                                    ,
                                    .(transactions = sum(N)),
                                    by = .(tax_year, good_data)
                                    ][
                                        good_data == TRUE
                                        ]
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', fill = "#1b9cad") +
    coord_cartesian(ylim = c(50, 100)) +
    ggtitle("% of Property Transactions In Groups With Good Data") +
    labs(x = "Tax Year", y = "% of Transactions",
         caption = "NB. Good data defined as those groups with 20 or more data points.") +
    theme_minimal()
```



% of Property Transactions In Groups With Good Data

NB. Good data defined as those groups with 20 or more data points.

This implies that the proportions are fairly consistent over time, and are above 78% in all years with all but two years above 96%.

NB. The two years with much lower totals are those for which we have incomplete data, because the Land Registry source data is based on calendar years. This further strengthens our confidence in the data quality.

3.3 Dispersion of Price Paid Data

Given that we have reasonable quality of data, we can now consider whether the data points themselves are useful. The starting point for this is to consider a specific example: what are the prices for terraced houses in West Kensington (W14) in the 2017 tax year?

```
by_outcde_yr_typ[prop_typ == "T" & tax_year == 2017 & outcde == "W14",
                 .(avg_price, median_price, price_stdev, q25, q75)]
```

```
##    avg_price median_price price_stdev     q25     q75
## 1:   2738202      2160000     3495526 1300000 2750000
```

The standard deviation of 3,495,526 is significantly more than the mean price of 2,738,202. In total there are 7,052 of 222,248 groups where the standard deviation is greater than the mean. And there are 41,083 groups where the standard deviation is greater than one half of the mean. These are crude measures, but they indicate that simply comparing a property transaction to the average will be insufficient in many parts of the country. We now consider trends over time as a means of augmenting this and determining how a property price compares to others in its area.

3.3.1 Trends over time

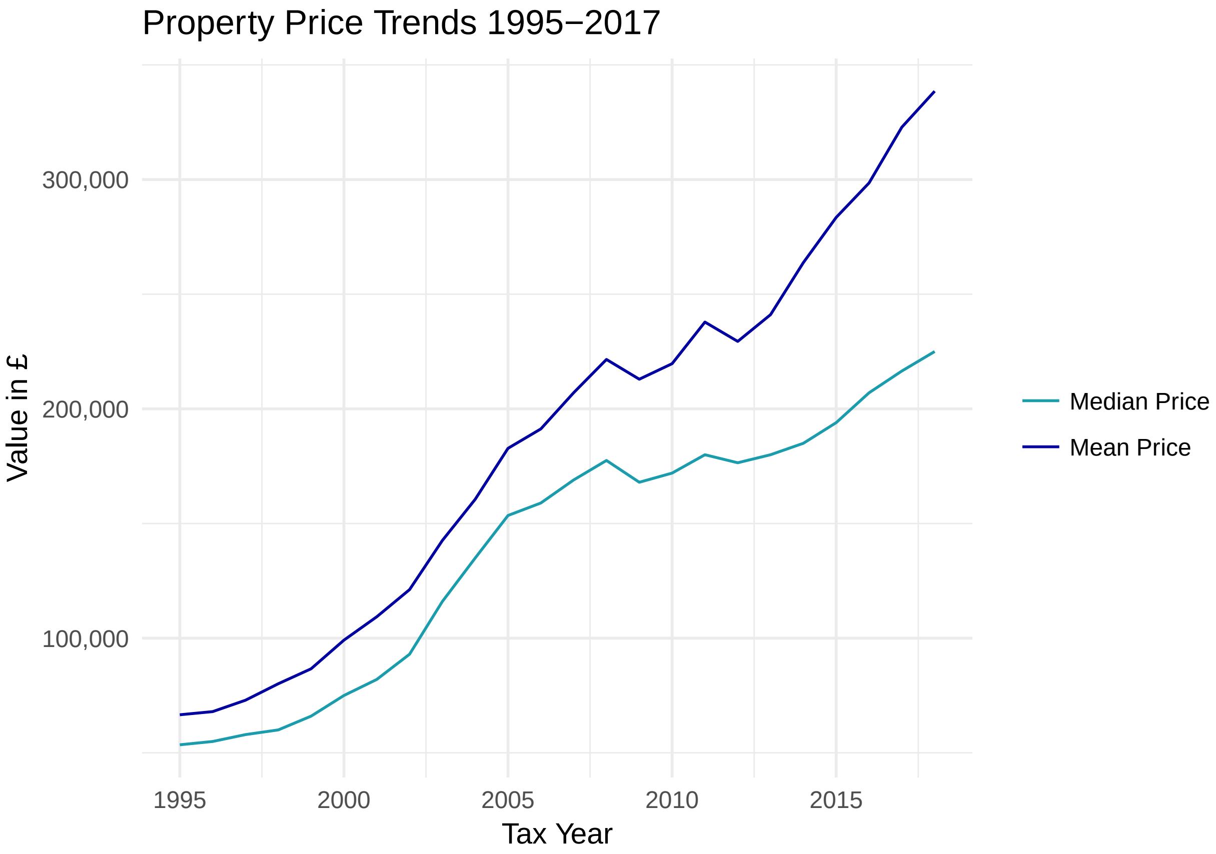
We need to consider whether the within-year trends seem to be consistent between years. Remember that we are considering three levels in our grouping:

- tax year;
- property type;
- area (i.e. postal outcode).

We can examine each of these individually, and then combine them and consider overall effects.

3.3.1.1 Year On Year Trends

```
melt(full_data[,
          .("Median Price" = quantile(price, .5),
            "Mean Price" = round(mean(price, na.rm = TRUE))),
          by = tax_year],
     id.vars = "tax_year",
     measure_vars = c("med_price", "avg_price")) %>%
    ggplot(., aes(x = tax_year, y = value, colour = variable)) +
    geom_line() +
    scale_y_continuous(label = comma) +
    theme_minimal() +
    theme(legend.title = element_blank()) +
    ggtitle("Property Price Trends 1995-2017") +
    labs(x = "Tax Year", y = "Value in £") +
    scale_colour_manual(values = c("#1b9cad", "#0000a0"))
```



Property prices are broadly increasing over time, with mean and median prices diverging. This indicates that the data are increasingly right-skewed, which tallies with anecdotal evidence about prices of property in London over recent years. (Although recent articles suggest that house prices in London are remaining static or shrinking in some areas, this is not yet reflected in the data.)

There are two useful elements to include in the output from this dataset.

1. An indicative price for any property transaction based on type, outcode, and tax year. (This will need to be considered along with measures of variance to place the expected price in a band.)
2. A measure of the year-to-year change in average price for each type, in each outcode. Where we have a previous transaction price for a given property this will allow us to compare the empirical and expected change in value: a large difference between these may be a useful risk indicator.

3.4 Creating the Output Product

The datasets created below are house prices indexes that can be used to compare with new transaction data. These will be most useful where we have a second data point for the same property from a previous transaction, so that we can:

1. compare the price directly to the averages for its group, and
2. compare the change over time against the group average.

Using both of these should give us a clearer idea of which property transactions are outliers.

```
output_table <- copy(main_ppdata)
output_table[,
             sd_mean_ratio := price_stdev / avg_price][
                 ,
                 paste0("diff_",
                        1:23) := lapply(1:23,
                                        function(x) {
                                            avg_price / shift(avg_price, n = x)
                                    }),
             by = .(outcde, prop_typ)
             ][
                 ,
                 scottish := NULL
             ]
```

Create the output files.

```
if(!file.exists("outputs/output_table_xlformat.csv")) {
    write_excel_csv(output_table, "outputs/output_table_xlformat.csv")
}
if(!file.exists("outputs/output_table_pipe.txt")) {
    write_delim(output_table, "outputs/output_table_pipe.txt", delim = "|")
}
if(!file.exists("outputs/output_table.csv")) {
    write_csv(output_table, "outputs/output_table.csv")
}
if(!file.exists("data/output_table.rds")) {
    write_rds(output_table, "data/output_table.rds")
}
```