Process Appraisal District - 2019

These lines specify whether or not we want to cache the results. Ideally, we don't want to run everything over and over again, so once we have the code running like we want, we can set cache to TRUE here.

```
knitr::opts_chunk$set(cache = TRUE)
```

General notes on this script

This script reformats files obtained from the Hays County Appraisal District to the common format used for import into the web database. The PDF created by this file should be saved as a record of the process used to create the database.

This script might need to be modified each year in order to accomodate changes to the files obtained from the appraisal district.

File output

Properties - [year]*properties_export*[date].csv

This is our main file for importing properties into the database.

When we check to see if a property already exists in the database, we'll convert this string to lowercase and remove all punctuation.

The desired format for this file is:

address ref_id property_name longitude latitude value owner_name owner_address owner_address_2 owner_address_3 owner_city owner_state owner_zip ownership_percent year

The address_canonical field should be composed as follows:

address_street, address_city, address_state address_zip fields

Notes for 2019

This script was prepared by Daniel Carter using the 2019 Certified Property Data Export Files available from https://www.hayscad.com/data-downloads/. These files are archived on the web server at rent_data/static/rent_data/data/2019.

Settings

This section sets up needed variables such as references to the file for import. Because we are using R Notebooks to process data, all files live in the working directory for this project. (Currently this directory lives on Daniel's laptop.)

Load the tidyverse library and set a variable with the name of the files to import. Those file should be stored in the same directory as this script. These

```
library("tidyverse")
```

```
## Registered S3 methods overwritten by 'ggplot2':
##   method         from
##   [.quosures     rlang
##   c.quosures     rlang
##   print.quosures rlang
```

```
## -- Attaching packages ---------------------------------------------------------------------------
```

```
## v ggplot2 3.1.1     v purrr   0.3.2
## v tibble  2.1.1     v dplyr   0.8.1
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library("readxl")
import_file_land = "2019_PropertyDataExport735288.txt"
import_file_properties = "2019_PropertyDataExport735286.txt"
import_file_owners = "2019_PropertyDataExport735287.txt"
```

Set the year, which is used to name output files.

```
year = "2019"
```

Process Properties

Read in the import file:

```
original_land = read_csv(import_file_land)
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   QuickRefID = col_character(),
##   PropertyNumber = col_character(),
##   LandType = col_character(),
##   Description = col_character(),
##   StateCode = col_character(),
##   ApprMethod = col_character(),
##   AgFlag = col_character(),
##   EffDepth = col_logical()
## )
```

```
## See spec(...) for full column specifications.
```

```
## Warning: 44 parsing failures.
##   row      col           expected     actual                                file
##  6168 EffDepth 1/0/T/F/TRUE/FALSE 115.000000 '2019_PropertyDataExport735288.txt'
##  6504 EffDepth 1/0/T/F/TRUE/FALSE 115.000000 '2019_PropertyDataExport735288.txt'
## 14411 EffDepth 1/0/T/F/TRUE/FALSE 48.000000  '2019_PropertyDataExport735288.txt'
## 15822 EffDepth 1/0/T/F/TRUE/FALSE 115.000000 '2019_PropertyDataExport735288.txt'
## 16320 EffDepth 1/0/T/F/TRUE/FALSE 208.000000 '2019_PropertyDataExport735288.txt'
## ..... ........ .................. .......... ................................
```

```
## See problems(...) for more details.
```

```
original_properties = read_csv(import_file_properties)
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   RecordType = col_double(),
##   PropertyID = col_double(),
##   LegalAcres = col_double(),
##   SubLotRange = col_logical(),
##   SubSection = col_logical(),
##   SubUnit = col_logical(),
##   LeaseNumber = col_logical(),
##   CurrMarketValue = col_double(),
##   CurrAssessedValue = col_double(),
##   CurrLandValue = col_double(),
##   CurrImprovmentValue = col_double(),
##   CurrAgValue = col_double(),
##   MarketValue = col_double(),
##   AssessedValue = col_double(),
##   LandValue = col_double(),
##   ImprovmentValue = col_double(),
##   AgValue = col_double(),
##   SquareFootage = col_double(),
##   SitusPostDirectional = col_logical(),
##   SitusZip = col_double()
## )
## See spec(...) for full column specifications.
```

```
## Warning: 706 parsing failures.
##  row                  col               expected actual                                file
## 1211 SitusZip             no trailing characters  -7964 '2019_PropertyDataExport735286.txt'
## 1472 SitusPostDirectional 1/0/T/F/TRUE/FALSE      S     '2019_PropertyDataExport735286.txt'
## 1744 SitusPostDirectional 1/0/T/F/TRUE/FALSE      N     '2019_PropertyDataExport735286.txt'
## 2433 SitusZip             no trailing characters  -1440 '2019_PropertyDataExport735286.txt'
## 2546 SitusPostDirectional 1/0/T/F/TRUE/FALSE      N     '2019_PropertyDataExport735286.txt'
## .... ..................... ...................... ...... ...................................
## See problems(...) for more details.
```

Reduce land to B1 designation. We're merging everything that has B1 for either StateCode or LandType, because this catches mixed use complexes and some other oddities.

```
original_state_b1 = original_land[original_land$StateCode == "B1",]
original_land_b1 = original_land[original_land$LandType == "B1",]
original_land_b1 = unique(rbind(original_state_b1, original_land_b1))
```

Merge B1 land with property information:

```
properties = merge(original_land_b1, original_properties, by.x = "PropertyID", by.y = "PropertyID")
```

At this point we have to make some corrections to the CAD data. There are 29 properties at 705 W River Rd., but only 20 match land records, meaning that when we reduce our properties to B1 land designation, we lose nine properties. The following step goes ahead and merges all the properties at that location.

Unique case: 705 W River Rd.

One issue here is that we have to assume that the ownership is the same for all the properties at this address, which it currently is. This is something we will need to check and verify each year.

First we make a table with just the properties we want to merge. We reduce that table to the first row and update the current market value.

```
tmp_705_w_river_rd = properties[grepl('705 w river rd', properties$Situs, ignore.case = TRUE),]
merged_705_w_river_rd = tmp_705_w_river_rd[1,]
merged_705_w_river_rd$CurrMarketValue = sum(tmp_705_w_river_rd$CurrMarketValue)
```

Now we remove all the properties at that address and add in our new merged row.

```
properties = properties[!grepl('705 w river rd', properties$Situs, ignore.case = TRUE),]
properties = rbind(properties, merged_705_w_river_rd)
```

Unique case: Stone Brook and La Vista

These are retirement homes, so we're getting rid of them

```
properties = properties[!grepl('R93286', properties$QuickRefID.y, ignore.case = TRUE),]
properties = properties[!grepl('R93223', properties$QuickRefID.y, ignore.case = TRUE),]
```

Unique case: 600 comanche

This is property owned by the university – no complex.

```
properties = properties[!grepl('R21257', properties$QuickRefID.y, ignore.case = TRUE),]
```

Unique case: 800 N LBJ

We also need to merge 800 N LBJ and 800 N LBJ & Forest Dr – same as above, except we need to match on the name instead of the address, so if Treehouse Apartments ever opens a new location, we'll need to update this.

```
tmp_800_n_lbj = properties[grepl('treehouse apts', properties$SitusLocation, ignore.case = TRUE),]
merged_800_n_lbj = tmp_800_n_lbj[1,]
merged_800_n_lbj$CurrMarketValue = sum(tmp_800_n_lbj$CurrMarketValue)
merged_800_n_lbj$Situs = "800 N LBJ DR, SAN MARCOS, TX 78666"

properties = properties[!grepl('treehouse apts', properties$SitusLocation, ignore.case = TRUE),]
properties = rbind(properties, merged_800_n_lbj)
```

Unique case: W Hutchinson

This is part of Sanctuary Lofts, 350 North Street. We're doing this a little differently, because matching on W Hutchinson seems like a bad idea. So, we're using the property ID, which we assume doesn't change, bu we'll need to keep an eye on this.

```
properties[grepl('treehouse apts', properties$SitusLocation, ignore.case = TRUE),]$CurrMarketValue = pr

properties = properties[properties$PropertyID != 84811,]
```

Unique case: VILLAS AT WILLOW SPRINGS

Need to combine this with its parking lot, which just has the address S IH 35.

```
properties[grepl('1506 S IH 35', properties$Situs, ignore.case = TRUE),]$CurrMarketValue = properties[g

properties = properties[properties$PropertyID != 118256,]
```

Unique cases – not apartments

Removing these based on manual inspection:

```
### Zeta Tau Alpha sorority
properties = properties[!grepl('102 MOSSCLIFF CIR', properties$Situs, ignore.case = TRUE),]

### No improvements on land
properties = properties[!grepl('1408 MARLETON ST', properties$Situs, ignore.case = TRUE),]

### Alpha Zeta sorority?
properties = properties[!grepl('428 W HUTCHISON ST', properties$Situs, ignore.case = TRUE),]

### Vacant lot?
properties = properties[!grepl('745 RIVER RD', properties$Situs, ignore.case = TRUE),]
```

1271 Sadler

```
properties[grepl('1271 SADLER', properties$Situs, ignore.case = TRUE),]$SitusLocation = "SADLER HOUSE A
properties[grepl('1271 SADLER', properties$Situs, ignore.case = TRUE),]$Situs = "1271 SADLER DR, SAN MA
```

Rename complexes without names

```
properties[grepl('417 N COMANCHE ST', properties$Situs, ignore.case = TRUE),]$SitusLocation = "POINTE S

properties[grepl('755 RIVER RD', properties$Situs, ignore.case = TRUE),]$SitusLocation = "RIVER ROAD AP
```

End unique cases

Reduce to columns

```
properties = properties[,c("QuickRefID.x","Situs","SitusLocation","PropertyID","PropertyNumber.x","Desc
```

Remove properties not in 78666:

```
properties = properties[grepl("^.* 78666", properties$Situs),]
```

Some properties might not have an address – kick those out to a separate table to examine later:

```
properties_no_address = properties[is.na(properties$Situs),]
properties = properties[!is.na(properties$Situs),]
```

Geocoding

Merge coordinates with properties table:

```
addresses_lon_lat = as.data.frame(cbind(unique_addresses, unique_lat_lon$lon, unique_lat_lon$lat))
colnames(addresses_lon_lat) = c("address","lon","lat")

addresses_lon_lat$address = as.character(addresses_lon_lat$address)
addresses_lon_lat$lon = as.numeric(as.character(addresses_lon_lat$lon))
addresses_lon_lat$lat = as.numeric(as.character(addresses_lon_lat$lat))

properties = merge(properties, addresses_lon_lat, by.y ="address", by.x ="Situs" )
```

Merge in Owners

Read in the owners and merge with the properties.

```
original_owners = read_csv(import_file_owners)
```

```
## Parsed with column specification:
## cols(
##   RecordType = col_double(),
##   PropertyID = col_double(),
##   QuickRefID = col_character(),
##   PropertyNumber = col_character(),
##   OwnerID = col_character(),
##   OwnerQuickRefID = col_character(),
##   OwnerPropertyNumber = col_character(),
##   OwnerName = col_character(),
##   Address1 = col_character(),
##   Address2 = col_character(),
##   Address3 = col_character(),
##   City = col_character(),
##   State = col_character(),
##   Zip = col_character(),
##   OwnershipPercent = col_double(),
##   ConfidentialOwner = col_logical(),
##   ExemptionList = col_character(),
##   HSCapAdj = col_double(),
##   CurrHSCapAdj = col_double()
## )
```

```
owners_properties = merge(properties, original_owners, by.x = "PropertyNumber.x", by.y = "PropertyNumbe
```

At this point, there might be rows that are exact duplicates. We assume there are errors in the data, so we just delete the duplicate rows.

```
owners_properties = unique(owners_properties)
```

Special Case: Copper Beech

Copper Beech has two owners listed, but we're pretty sure they're the same, so we're just going to set the owner to COPPER BEECH TOWNHOME COMMUNITIES. The next step will merge these and take the first owner address.

```
owners_properties[grepl("1701 MILL ST",owners_properties$Situs, ignore.case = TRUE),]$OwnerName = "COPPI
owners_properties[grepl("1701 MILL ST",owners_properties$Situs, ignore.case = TRUE),]$SitusLocation = "
```

End Special Case: Copper Beech

At this stage, we'll also go ahead and reduce the columns to just what we need. Note that we assume here that all of the multiple properties to be merged have a single owner. In the report section, there's a flag for whether or not any properties seem to violate this.

```
library(dplyr)

multiple_owners_multiple_properties = owners_properties %>% group_by(Situs) %>% filter (n() > 1)
multiple_owners_flag = FALSE
if ( length(unique(multiple_owners_multiple_properties$OwnerName)) != length(unique(multiple_owners_mult
  multiple_owners_flag = TRUE
  }

owners_properties = owners_properties %>% group_by(Situs) %>% summarise(
  ref_id = first(QuickRefID.x),
  SitusLocation = first(SitusLocation),
  CurrMarketValue = sum(CurrMarketValue),
  lon = first(lon),
  lat = first(lat),
  OwnerName = first(OwnerName),
  Address1 = first(Address1),
  Address2 = first(Address2),
  Address3 = first(Address3),
  City = first(City),
  State = first(State),
  Zip = first(Zip),
  OwnershipPercent = first(OwnershipPercent)
  )
```

Export Files

Export a csv of properties to upload into the web system.

```
properties_export = owners_properties[,c("Situs","SitusLocation","lon","lat","CurrMarketValue", "OwnerNa
properties_export$year = year
colnames(properties_export) = c("address_canonical","property_name","longitude","latitude","value","owne
properties_export = properties_export[!is.na(properties_export$latitude),]
properties_filename = paste(year,"_properties_export_",as.Date(Sys.Date()),".csv",collapse="", sep="")
write.csv(properties_export, properties_filename, row.names = FALSE)
```

Also export a csv of unique addresses to use in filtering.

```
unique_addresses = unique(unique_addresses)
unique_addresses_simple = unlist(lapply(unique_addresses, function(x) { unlist(strsplit(x, ",", fixed=TR

unique_address_filename = paste(year,"_unique_property_addresses",as.Date(Sys.Date()),".csv",collapse="
unique_addresses = cbind(unique_addresses, unique_addresses_simple)
colnames(unique_addresses) = c("address_full","address_simple")
write.csv(unique_addresses, unique_address_filename, row.names = FALSE)
```

Sample 50

```
sample_filename = paste(year,"_sample_50_",as.Date(Sys.Date()),".csv",collapse="", sep="")
for_sample = owners_properties[owners_properties$CurrMarketValue >= 1000000,]
write.csv( for_sample[sample(nrow(for_sample), 50),], sample_filename, row.names = FALSE)
```

for_sample = owners_properties[owners_properties$CurrMarketValue >= 1000000,]

Report

Property records imported: 134

Unique properties: 125

Final reduced properties: 165

Properties without addresses: 0

Properties without coordinates: 0

Check on combined properties that might have multiple owners? TRUE