GGR376 - Lab 1 - Tutorial

Regression

Open a new R Script in RStudio

Scripts must be saved with the .R extension

```
library(tidyverse)
```

Reading Data

Files on your hard drive:

- Navigate to: https://raw.githubusercontent.com/tidyverse/readr/master/inst/extdata/mtcars.csv
- Download the file with "Save As"
  - Note where the file is saved

```
# Read in our csv file
mtcars <- read_csv("**actual_file_location**/mtcars.csv")

# View the data
mtcars
```

Files from the internet

Read in files stored on the internet using the url.

```
mtcars_url <- read_csv(
  "https://raw.githubusercontent.com/tidyverse/readr/master/inst/extdata/mtcars.csv")
```

View the data using RStudio's View()

```
View(mtcars_url)
```

Read the help file for the read_csv() function, using the ? or help() function.

```
?read_csv
# or
help(read_csv)
```

Examine the environment panel in RStudio

Remove both versions of the mtcars data

```
remove(mtcars, mtcars_url)
```

Regression Exercise

The TA will walk you through the process of regression and spatial regression. This tutorial was inspired by Roger Bivand's Spatial Regression examples on rspatial.org. Your assignment will require you to produce a similar work using Canadian Census Data.

The data for this tutorial is stored in three files on a GitHub account. It is California house price data from the 2000 Census, and the files include:

- Average house price per census tract file
- Census tract attributes file
- Census tract polygon file.

They all share a GEOID variable, which can be used to join the files.

Merging Tables

Read in the housing price file and the attributes file.

```
house_prices <- read_csv(
  "https://raw.githubusercontent.com/gisUTM/GGR376/master/Lab_1/Tutorial/house_prices.csv")
house_attributes <- read_csv(
  "https://raw.githubusercontent.com/gisUTM/GGR376/master/Lab_1/Tutorial/house_attributes.csv",
  guess_max = 9500)
```

The tables need to be joined. The dplyr package includes a functions for joining data.frames/tibbles. See the Data Wrangling Cheat Sheet for visual examples: https://www.rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf

- dplyr::left_join(), Join matching rows from b to a.
  - NAs possible
- dplyr::right_join(), Join matching rows from a to b.
  - NAs possible
- dplyr::inner_join(), Join data, retain only rows in both sets
  - No NAs, rows excluded that create a NA
- dplyr::full_join(), Join data, retain all values, all rows.
  - NAs possible

Rows with missing values will need to be excluded in the regression model. Use an inner_join() so no NAs will be introduced.

```
california_cts <- inner_join(house_prices, house_attributes, by = "GEOID")
california_cts
```

Another file contains attribute definitions. Read in and store as a tibble.

```
variable_attributes <- read_csv(
  "https://raw.githubusercontent.com/gisUTM/GGR376/master/Lab_1/Tutorial/Attribute_Definitions.txt")
variable_attributes
```

Data Management

These data are at the census tract level of geography. There are 7,049 census tracts. The regression model will focus on San Diego County.

1. Split the label variable into three variables:
   1. CT
   2. County
   3. State
2. Remove any extra spaces before or adter a county name, which can occur when splitting strings.
3. Filter County == “San Diego County”

```
san_diego <- california_cts%>%
  separate(label, sep = ",", into = c("CT", "County", "State"))%>% # Split
  mutate(County = str_trim(County, side = "both"))%>% # Remove
  filter(County == "San Diego County") # Filter
```

```
san_diego
```

Modify the count variables

Create ratio variables from count variables. Many variables are only dependent on CT population, which won't be helpful in the linear regression model.

```
san_diego_ratios <- san_diego%>%
  # Proportion of mobile homes to total units
  mutate(pMobileHom = nMobileHom/nhousingUn)%>%
  mutate(pBadPlumbi = nBadPlumbi/nhousingUn)%>%
  mutate(pBadKitche = nBadKitche/nhousingUn)
```

Potential Predictors

The tutorial will focus on seven predictor variables:

- yearBuilt
- nRooms
- nBedrooms
- MedianAge
- pMobileHom
- pBadPlumbi
- pBadKitche

Subset the variables we will use in the model

```
# Include GEOID and houseValue
sd_houses <- san_diego_ratios %>%
  select(GEOID, houseValue, yearBuilt, nRooms, nBedrooms, MedianAge, pMobileHom,
         pBadPlumbi, pBadKitche)
```

Review Data Values: e.g. zeros where data should have values.

```
sd_houses%>%
  summary()

# Notice houseValue, yearBuilt, nRooms, nBedrooms
```

Filter out rows with 0 values for either houseValue, yearBuilt, nRooms, nBedrooms

```
sd_houses <- sd_houses %>%
  filter(yearBuilt > 0 , houseValue > 0 , nRooms > 0, nBedrooms >0 )

# Check the summary
sd_houses%>%
  summary
```

Graphical Analysis Pre-Check

- Scatter plot: Check for linear relationships between predictors and dependent
- Box plot: Outlier check
- Histogram: Check variables for normal distributions

Each plot type will be demonstrated. Then you need to complete the remainder. Base R plots are fine for variable review.

Including mapping = aes(x = yearBuilt, y = houseValue in the ggplot() function removes the need to specify the x and y for each geom added.

Scatter Plot ggplot2()

```
ggplot(data = sd_houses, mapping = aes(x = yearBuilt, y = houseValue))+
  geom_point()+
  geom_smooth(method = "lm", se = F)
```

Scatter Plot Base R

```
plot(sd_houses$yearBuilt, sd_houses$houseValue)
# To include the regression line, we use abline()
abline(lm(houseValue~yearBuilt, data = sd_houses))
```

Box plot ggplot2()

```
ggplot(data = sd_houses)+
  geom_boxplot( mapping = aes(x = "", y = nBedrooms)) # Note: x = ""
# In this example the value is not extreme, it is fine to leave it.
```

Box plot base r

```
boxplot(sd_houses$nBedrooms) # Much easier to work with base R here.
```

Histogram in ggplot2

```
ggplot(sd_houses)+
  geom_histogram(mapping = aes(x = houseValue))
# Warning can be ignored because this plot is temporary
```

Histogram in base r

```
hist(sd_houses$houseValue)
```

Normality test - shaprio.test()

The null hypothesis of this test is that the data are normal.

```
sd_houses%>%
  # We can ignore GEOID in this test
  select(-GEOID)%>%

  # Map allows us to apply a function to each variable in our tibble
  map(hist)
```

Data Transformations

First data transformation for houseValue

```
sd_trans <- sd_houses%>%
  mutate(logValue = log(houseValue+1)) # We add 1 to each value so we don't get -Inf
# Chain on addition mutates with the pipe
```

Chained mutations

```
sd_trans <- sd_houses%>%
  mutate(logValue = log(houseValue+1))%>%
  mutate(logMobile = log(pMobileHom+1))%>%
  mutate(logPlumb = log(pBadPlumbi+1))%>%
  mutate(logKitchen = log(pBadKitche+1))%>%
  select(-houseValue, -pMobileHom, -pBadPlumbi, -pBadKitche)
```

Correlation Matrix

```
cor_mat <- cor(sd_trans%>%
                 select(-GEOID)) # Exclude as it will not be used in calculations.
corrplot::corrplot(cor_mat, cl.pos = "b", tl.pos = "d")
```

Variable Selection

In the tutorial a manual approach to variable selection will be applied. - Begin with the strongest predictor variable based on the correlation plot, which appears to be median age. - Add variables and check how they affect the model.

```
model_1 <- lm(logValue ~ MedianAge, data = sd_trans)
```

Take a look at the model.

```
model_1
```

Use the summary function

```
summary(model_1)
```

The predictor variable, MedianAge, is statistically significant and is the the overall model. The adjusted R-Squared is 0.36, thus the model's explains 36% of the variation in the dependent variable.

Recall Assumptions of the model

1. The mean of the residuals is zero
2. Homoscedasticity of residuals or equal variance
3. Multicollinearity
4. The independent variables and residuals are uncorrelated
5. The variability in X values is positive
6. The number of observations must be greater than number of independent
7. Normality of residuals
8. No auto-correlation of residuals

Without any work we know we have satisfied: 5. The variability in X values is positive + Which is based on when we looked at the ranges of all our our variables.

6. The number of observations must be greater than number of independent
    - We have hundreds of observations (rows) in our dataset.

The mean of the residuals is zero

```
# Mean of the residuals is zero
mean(model_1$residuals)
```

R by default uses scientific notation. To change this run the following command.

```
options(scipen=999)
```

Now call the function.

```
mean(model_1$residuals)
```

To set it back

```
options(scipen=0)
mean(model_1$residuals)
```

Remove the scientific notation and re-run the summary for model_1

```
options(scipen=999)
# Go back and run the summary of your model_1
```

Homoscedasticity of residuals or equal variance

Check for a flat line in the first and third plots.

```
par(mfrow=c(2,2)) # set 2 rows and 2 column plot layout in base R
plot(model_1)
```

Oobservations 402, 405 and 471 are causing some issues in the model. Take a look at their values using the slice function.

```
sd_trans%>%
  # Slice the two rows using a vector c(402, 471)
  slice(c(402, 471, 405))
```

Recall from the boxplot of MedianAge that these are extreme values. Remove them from the model input, as it is okay in this scenario to exclude area of very old houses.

```
sd_trans_sub <- sd_trans%>%
  # We can use a minus sign to include everything in the slice except those rows
  slice(-c(402, 471, 405))
```

Fit the model with the subset data.

```
model_2 <- lm(logValue ~ MedianAge, sd_trans_sub) # Note the name change
summary(model_2)
```

Plot the assesment graphics

```
par(mfrow=c(2,2)) # set 2 rows and 2 column plot layout
plot(model_2)
```

Add a second variable to the model. - nBedrooms

```
model_3 <- lm(logValue ~ MedianAge + nBedrooms, data = sd_trans_sub)
summary(model_3) # The Adj.R.SQ increased by ~0.04
```

Check that we didn't introduce heteroscedasticity.

```
par(mfrow=c(2,2)) # set 2 rows and 2 column plot layout
plot(model_2)
```

Let's try adding one more variable.

```
model_4 <- lm(logValue ~ MedianAge + nBedrooms + logKitchen, data = sd_trans_sub)

summary(model_4)
```

No increase to the Adj.R.SQ - It is not staitically significant

Add nRooms

```
model_5 <- lm(logValue ~ MedianAge + nBedrooms + nRooms, data = sd_trans_sub)

summary(model_5)
```

- nBedrooms is no longer significant
  - Multicollinearity is occuring because its cofficient sign flipped.
  - Confirm multicollinearity with car::vif()

```
library(car)
vif(model_5)
```

Spend ten minutes testing additional models

- summary()
- plot()

Continue with model_3

```
summary(model_3)
```

Continue Assumption Checks

Multicollinearity

```
vif(model_3)
```

Values are less than 4, we can move on.

Correlation between residuals and independent variables

```
# Test for significant correlation between residuals and MedianAge
cor.test(model_3$residuals, sd_trans_sub$MedianAge)

# Test for significant correlation between residuals and nBedrooms
cor.test(model_3$residuals, sd_trans_sub$nBedrooms)
```

- No issues of correlation, p-value > 0.05

Normality of residuals

- Can check with the normal Q-Q plot, histogram, or shapiro.test(model_3$residuals).
- Given the large sample size, it is possible to relax this assumption
  - i.e. we do not need to be extremely strict.

```
hist(model_3$residuals)
```

This is an acceptable amount of deviation from normal given the large sample size.

Auto-correlation of residuals

Geographical data often suffers from spatial auto-correlation of the residuals.

The steps for checking for spatial auto-correlation include:

1. Import spatial boundaries
   - data.frame is not geographical
2. Join data and residuals
3. Moran's I calculation for residuals
4. If spatial-autocorrelation occurs, use a spatial-error or spatial-lag model

Spatial Data in R

Spatial data can be stored in spatial data classes from the package.

- SpatialPoints or SpatialPointsDataFrame
- SpatialLines or SpatialLinesDataFrame
- SpatialPolygons or SpatialPolygonsDataFrame

For each of these, the first class is only able to store geometry. The second class, Spatial*DataFrame allows variables along with geometry.

```
library(sp)
?SpatialPolygonsDataFrame
```

The polygon data is stored on the GitHub page and can be opened using: - rgdal::readOGR()

```
library(rgdal)
?readOGR
```

Data are stored in a geoJSON

```
california_polygons <- readOGR(
  "https://raw.githubusercontent.com/gisUTM/GGR376/master/Lab_1/Tutorial/house_polygons.geojson")
```

Check the class (type) of the variable california_polygons

```
class(california_polygons)
```

Plot the spatial boundaries

```
plot(california_polygons)
```

Take a look at the data. It contains one variable GEOID, which can be used in joining the data and residuals.

```
california_polygons@data
```

- Create a dataframe with the model residuals and the model input data

```
model_3_dataframe <- cbind(sd_trans_sub, residuals = model_3$residuals)
model_3_dataframe
```

- Join the new dataframe to the sp object using sp::merge()

```
california_spdf <- merge(california_polygons, model_3_dataframe, by.x = "GEOID", all.x = FALSE)
```

Map the residuals

```
library(latticeExtra)
spplot(california_spdf, "residuals")
```

This is a ugly map. We have colour Brewer schemes in R

```
display.brewer.all()
```

Create a colour palette to be used in our map with 7 colours (6 breaks)

```
col_palette <- brewer.pal(n = 7, name = "BrBG")

col_palette
```

Generate the map

```
spplot(california_spdf, "residuals",
       col.regions = col_palette, # The colour palette you just created
       cuts = 6, # Cuts is the number of colours - 1
       col = "transparent") # This sets the border colour, try changing to "black"
```

Spatial auto-correlation is indicated by this plot.

Confirm Spatial Auto-Correlation with Moran's I

- Moran's I requires spatial weights
  - Points: most often distances
  - Polygons: Neighbours or distances

Create a list of polygons that share borders with spdep::poly2nb() - It would be possible to add connections in this list - E.g. areas that do not physically touch by are connect by a bridge across a bay.

```
library(spdep)
california_nb <- poly2nb(california_spdf)
```

Plot neighbour connections

```
par(mai=c(0,0,0,0))
plot(california_spdf)
plot(california_nb, coordinates(california_spdf), col='red', lwd=2, add=TRUE)
```

It is congested on the west coast, but this demonstrates the relationships stored in the neighbours list.

Moran's I Calculation

1. The neighbours list must be changed to weights based on the number of neighbours.
   - If a polygon has 5 neighbours, each weight is 0.2.
2. moran.test()
   - sensitive to the form of the graph of neighbour relationships and other factors
   - best to check results with those of moran.mc()
3. moran.mc()

Convert Neigbours List to Spatial Weights

```
california_listw <- nb2listw(california_nb)

print("View the neighbours for polygon 20")
california_listw$neighbours[[20]]

print("View the weights for polygon 20")
california_listw$weights[[20]]
```

Run the two tests for Moran's I

```
moran.test(california_spdf$residuals, california_listw)
moran.mc(california_spdf$residuals, california_listw, 999)
```

Spatial Regression

1. Spatial Lag
2. Spatial Error

Spatial Lag Model

spdep::larsarlm()

```
# Spatial Lag
lag_model = lagsarlm(logValue ~ MedianAge + nBedrooms,
                     data = california_spdf,
                     listw = california_listw)
```

Get more details with the summary function

```
summary(lag_model)
```

Check the residuals for spatial auto-correlation.

- Add residuals from lag model to SpatialPolygonDataFrame

```
california_spdf$lagResids <- residuals(lag_model)
moran.mc(california_spdf$lagResids, california_listw, 999)
```

Spatial Error Model

spdep::errorsarlm()

```
error_model = errorsarlm(logValue ~ MedianAge + nBedrooms,
                     data = california_spdf,
                     listw = california_listw)
```

Check the error model residuals for spatial auto-correlation.

```
# Add error model residuals to SpatialPolygonDataFrame
california_spdf$errorResids <- residuals(error_model)

# Moran's I
moran.mc(california_spdf$errorResids, california_listw, 999)
```

One last plot of the residuals

```
spplot(california_spdf, "errorResids",
       col.regions = col_palette, # The colour palette you just created
       cuts = 6, # Cuts is the number of colours - 1
       col = "transparent")
```

Lagrange Multiplier diagnostics

You would run this test prior to selecting the error or lag model.

```
summary(lm.LMtests(model_3, california_listw, test="all"))
```

In this example, it is not very helpful as they are both significant. A theoretical underpinning should be applied to select the model.