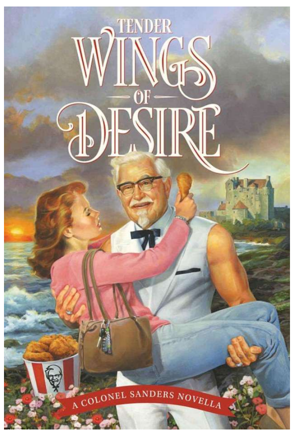
Gasoline Submarkets in Western Canada

A Love Story

A Torrid Tale

- Gasoline Prices, R and Python are not very romantic…

Disclaimer: This is 100% tongue in cheek I plan on using both R and Python in the future.



What’s so romantic?

- Manage expectations…
  - The bar is low
- Still a better love story than Twilight
- *Wings of Desire* quality
  - Yes that is Colonel Sanders
- NB, I haven’t ready either of these

Agenda

- Over view of economics and market structure
- Explain data gathering and wrangling methods
- Supervised unsupervised training
- Results
- Why R is great
- Lame jokes and overused memes…

Hypothesis

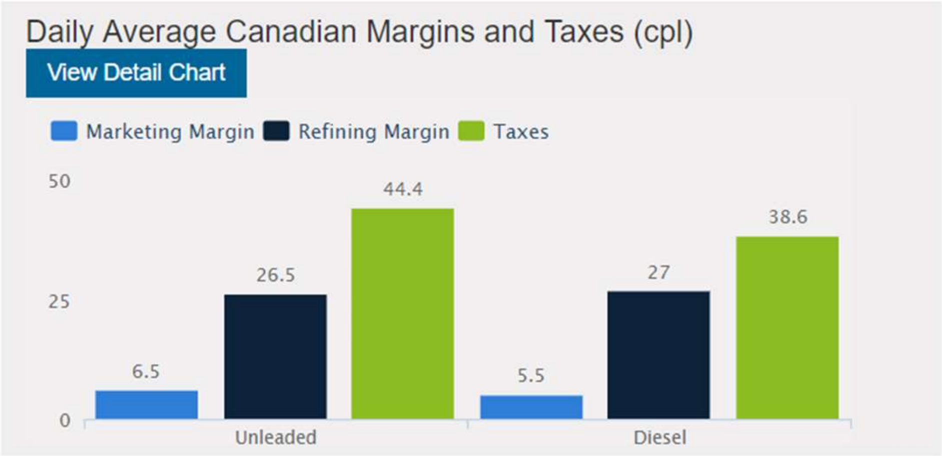
- Retail markets have distinct sub regions that react differently to economic forces. This implies that the optimal price is not set at a city level but at a neighborhood level.
- If this is true, retailers are leaving significant value on the table.

Economic Background

- Like all great romances a strong economic background can go a long way explaining what’s going on.
- $\pi(p) = (p - c) * q(p)$
- $q(p) = l - (p_{own} * \alpha) - ((p_{own} - p_{competition}) * \beta)$ Where $\beta > \alpha$
- Therefore given a price increase a firm will lose volume in general, but the price difference to their competitors is more important.
- Gasoline prices are highly dependant on other sites close by.

Market Structure

- Gas prices are made up by:
  - Tax
  - Crude/refining/transportation costs
  - Retail margin
- Tax is set municipally
- Refining margin is set globally
- Retail margin is set locally



Updated on Fri Apr 12, 2019 (refiner margin is from previous weekday)

Kent Fuel Group

Methods

The romance begins…

- Gathering
- Cleaning
- Un-supervised learning
- Supervised learning
- Cross Validation

Web Scraping

Less than romantic

- https://www.gasbuddy.com/home?search=calgary&fuel=1
- Python
    - Selenium
    - Beautiful Soup
    - Blood sweat and tears
- AWS
    - Serverless (Faild due to Pandas)
    - EC2, RDS
    - Cron jobs
    - So many failures

Code Review

The ugly side of the relationship

- https://github.com/kaiserxc/DATA698_Capstone_Project/blob/master/scraper_app/scraper.py
- Please don’t laugh at my very janky scraping.
- Why not R?
  - Rvest, RSelenium
  - About the same difficulty
  - Personal Preference, I was ignoring how much I loved R

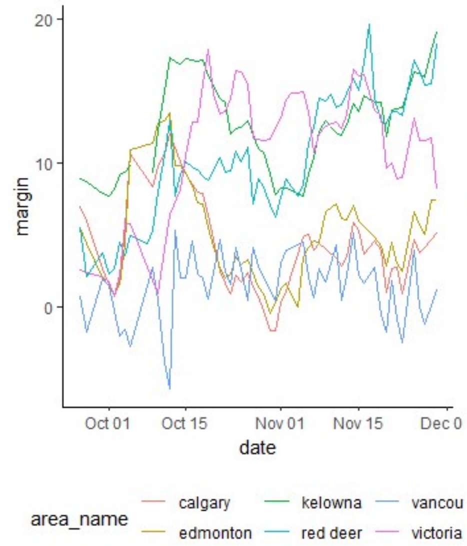
Relationship Counselling

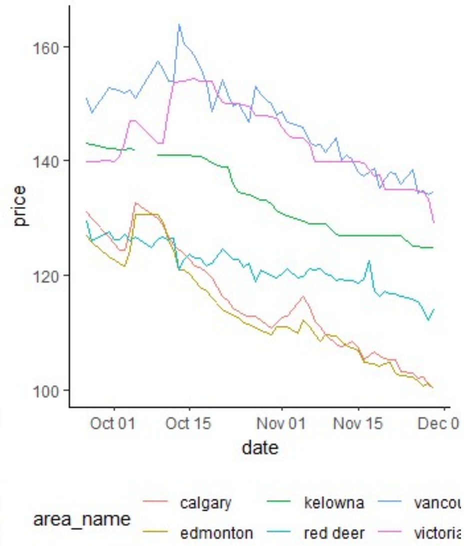
- Painful Web Scraping
- AWS Failures
  - The relationship fell apart because of poor communication
  - Set up server failure notifications
  - Reset server after failure (making up after a fight)

Sorry for the overused tropes and stretched metaphors…

Time Series Creation

- Pandas
  - Re-index to hourly data
  - Forward and backfill 72 hours
  - Tried time series clustering…
    - Relationship issues
- R’s XTS
  - Functionally similar but I just wasn’t ready for it.





The Breakup

The devil is in the details

- The issue:
  - Python has some time series clustering functionality
  - Difficult, unintuitive existing functions
  - Need to implement my own functions

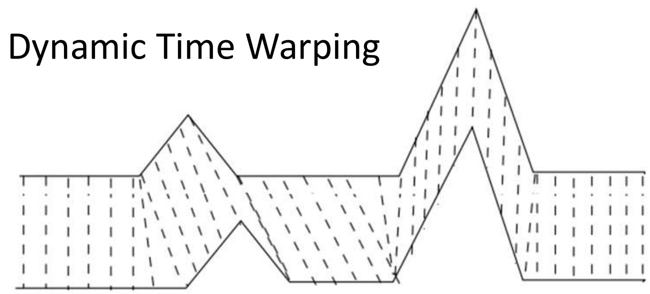
- The solution:
  - TSClust
  - data.table

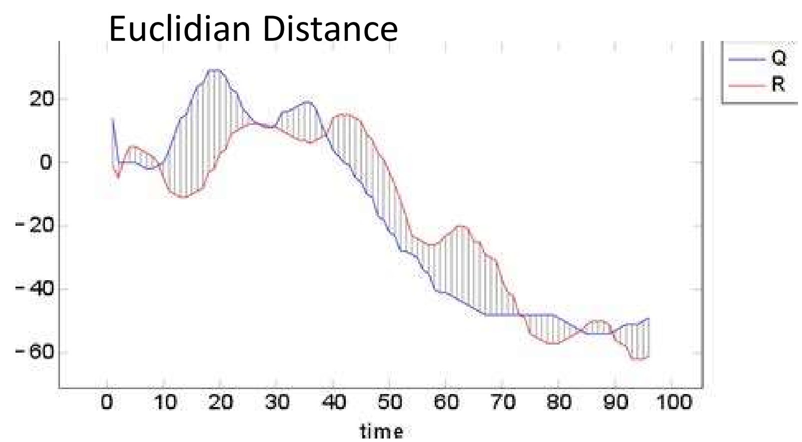
Honeymoon

- data.table is amazingly concise
- ggplot has all the best graphics
- TSClust has all the functionality one could wish for time series clustering

Time Series Clustering…

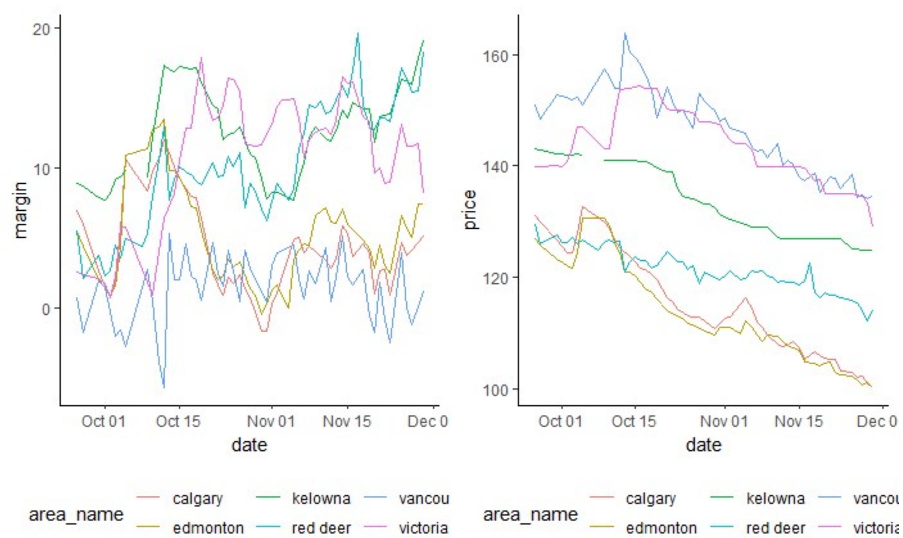
- I started using R because of DTW
  - Great for Audio
  - Poor for clustering markets
  - Super Expensive (out of budget)
- Euclidian Distance catches market changes better
  - Cheap
- Correlation has similar benefits





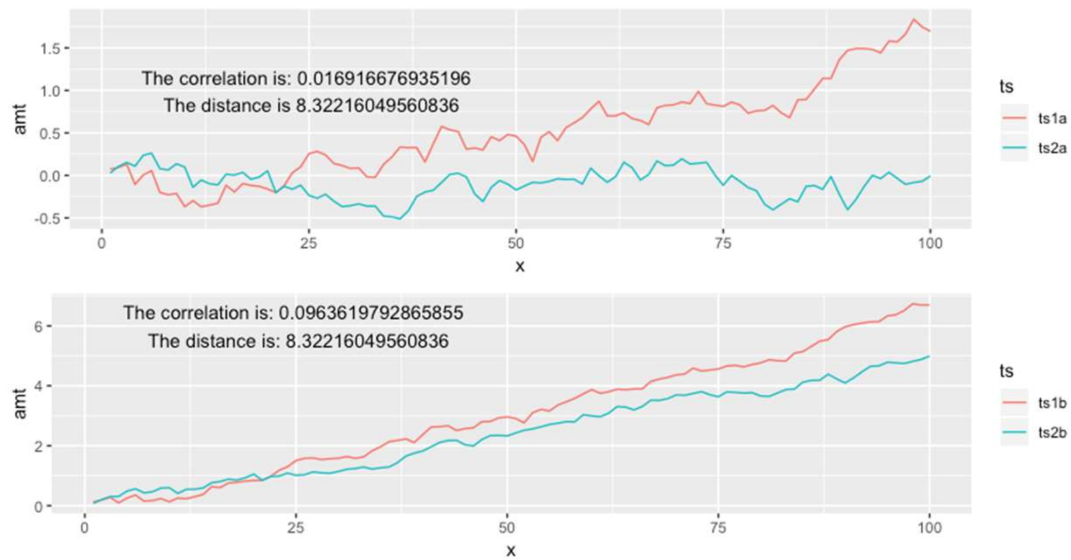
Margin vs. Price

- Correlation is higher with Prices than Margin
- Difficult to see underlying relation with trend
- Non-adjusted prices is playing on easy
  - Differences between cities could overwhelm submarket differences



Correlation vs Euclidian Distance

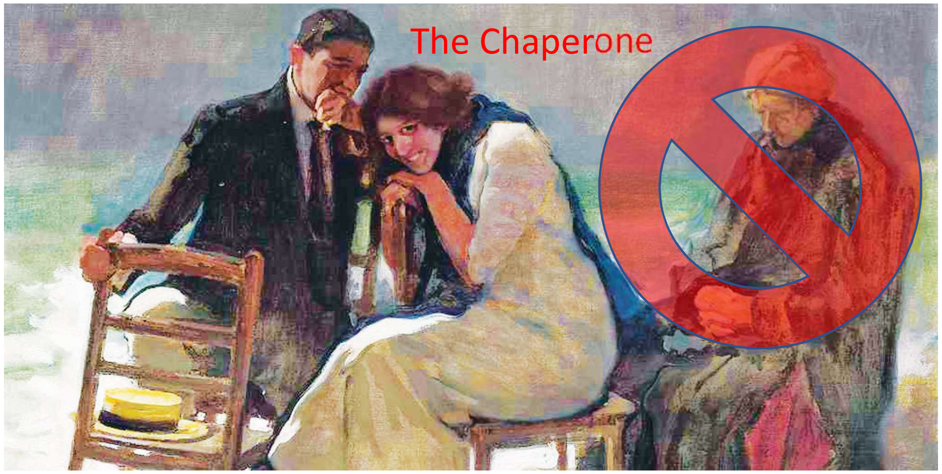
- Correlation can be affected by a trend
- Euclidian distance doesn't care about the trend
- I used both raw and 1 day rolling average
- Both values had tax and refining margin taken out still



Unsupervised Learning

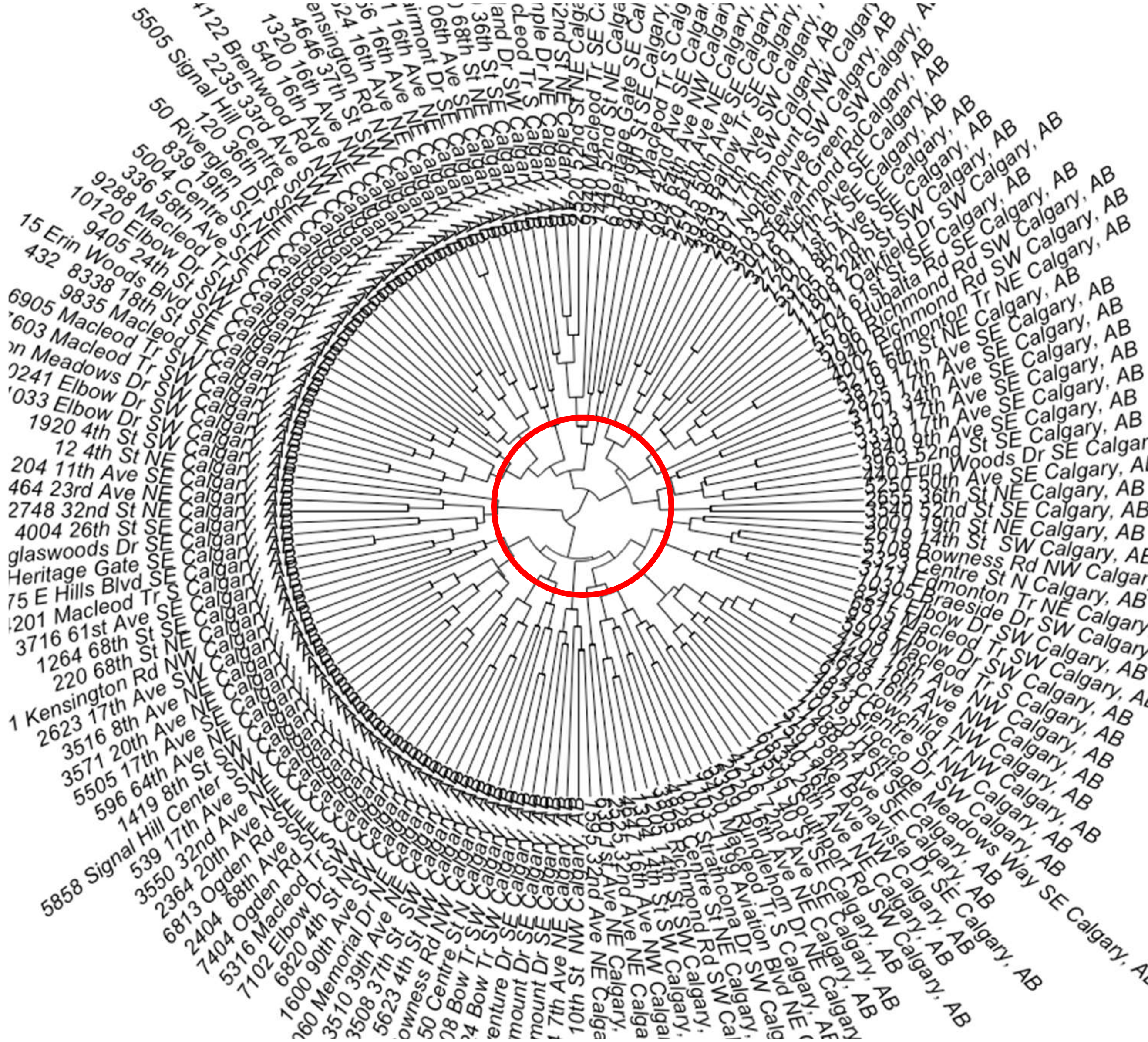
Who wants a chaperone?

- Cluster by Corr and Euclid
- Dendrograms were used to cluster individual sites
- Clusters were labeled at an a arbitrary height



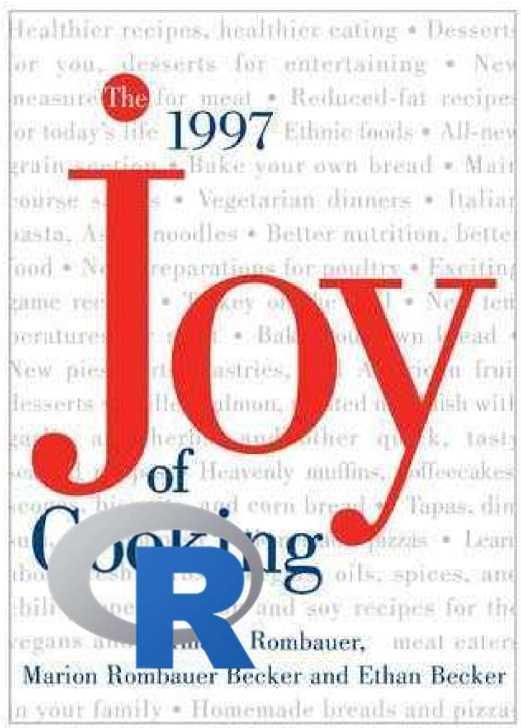
Dendrogram

- Calgary area dendrogram
- Hight cutoff to group similar sites
- Note similar street addresses clustered together



The Joy of R

Why I fell in love again.

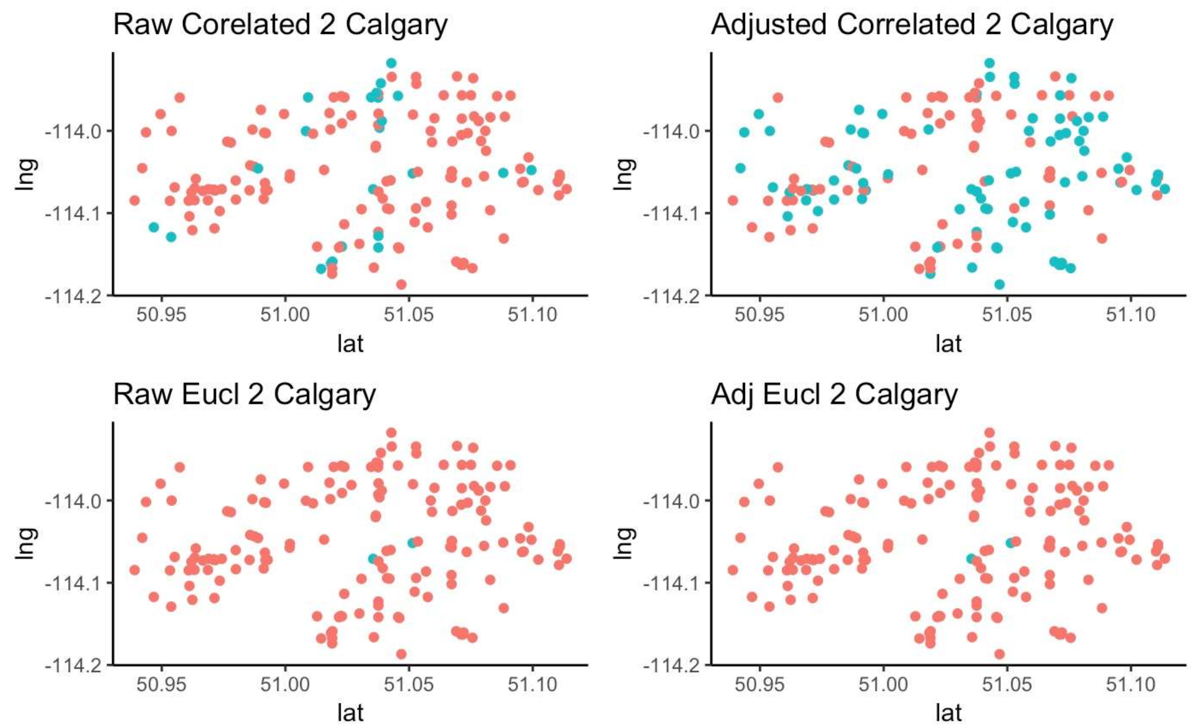



```
cluster_getter <- function(tb, method =
'COR'){
  tb <- ts_maker(tb) #Makes time series
  h <- diss(tb, METHOD = method)
  c <- hclust(h)
  return(c)
}
```

This concise syntax and wonderful library are why I'm such a big fan of R

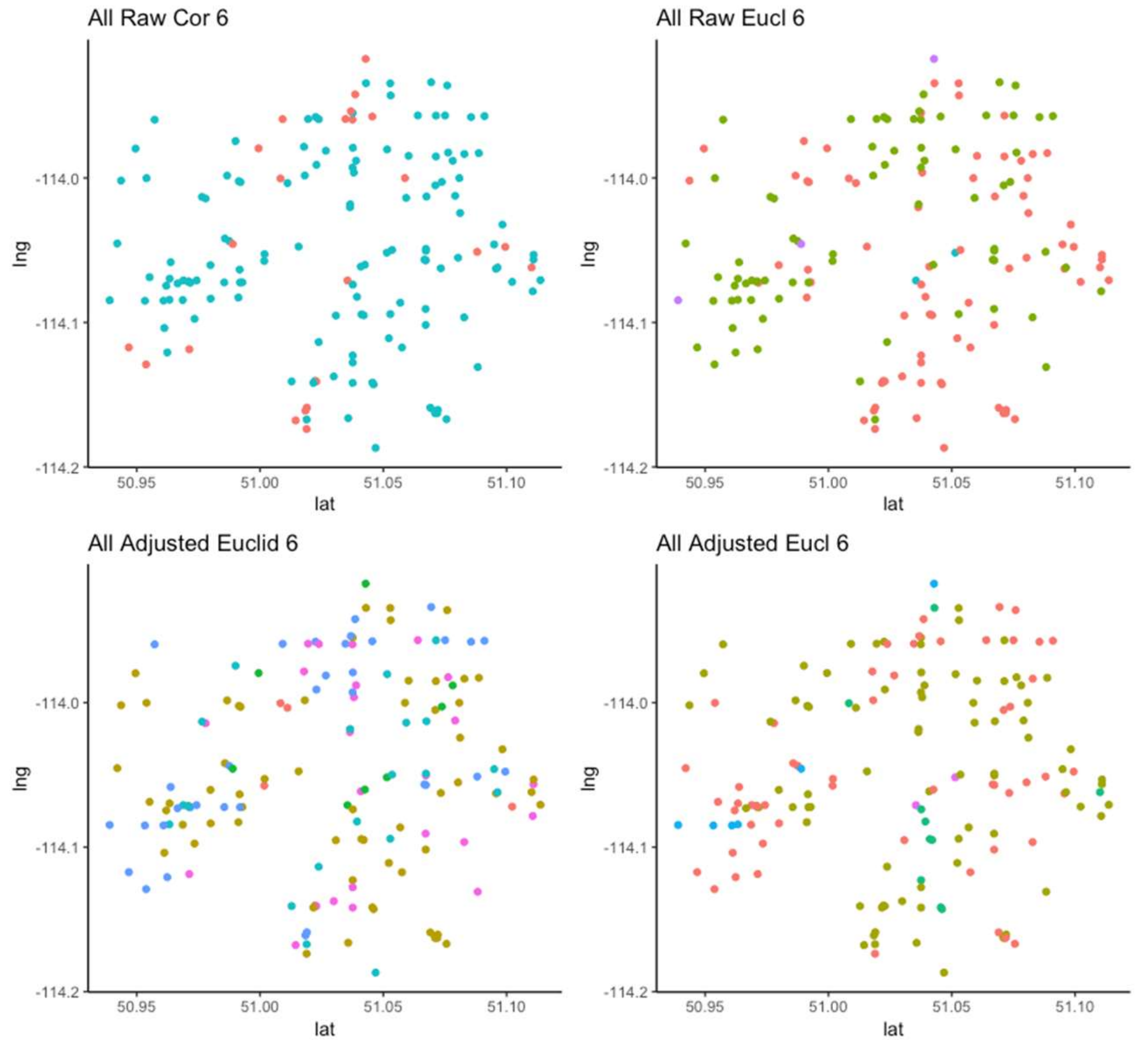
Supervised Learning

- Latitude and Longitude as independent variables
- Cluster ID as dependant variable



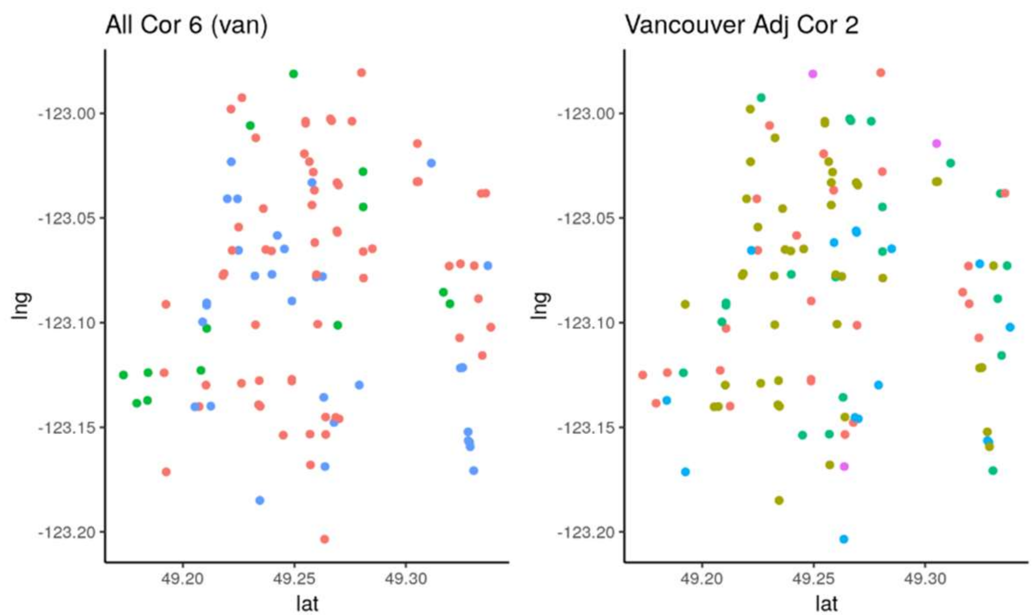
6 Categories

- Still no super distinct areas



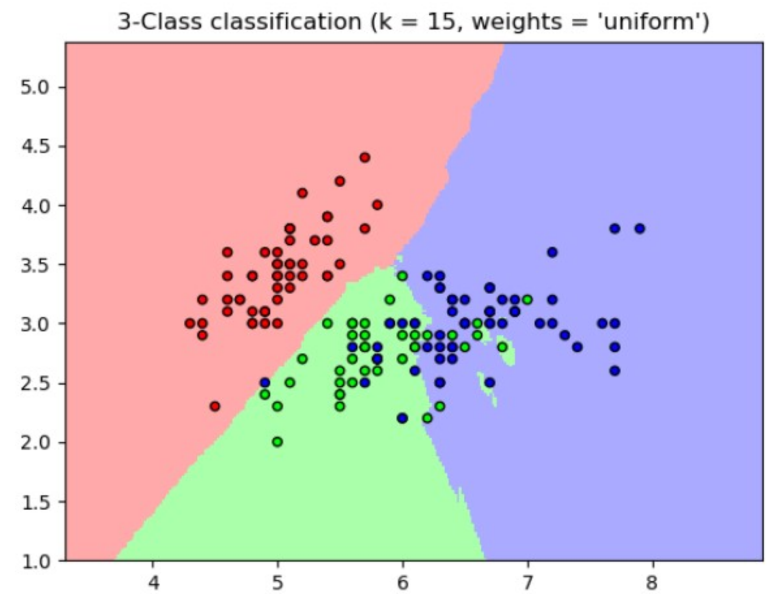
KNN Clustering in Vancouver

- Similar results to Calgary
- Potentially interesting clusters



KNN Models

- Uses n independent variables to predict labels of the dependant variable
- Uses the n-nearest points to vote on a variable
- The goal: Map like this over each city



KNN Code

- I wrote a function to train, test and score multiple models, speeding up development time
- It also returned the best number of neighbors
- Map functions could solve this more elegantly

```r
knn_maker <- function(dt){
  # Lable Must be first in dataset with lat, lng following
  lab_col <- names(dt)[1]
  in_train <- createDataPartition(y = dt[,as.factor(get(lab_col))], p = 0.7
  train_dt <- dt[in_train]
  test_dt <- dt[!in_train]
  # CV
  trControl <- trainControl(method  = "cv",
                            number  = 10)
  fit <- train(as.formula(paste0(lab_col, "~ .")),
               method     = "knn",
               tuneGrid   = expand.grid(k = 1:20),
               trControl  = trControl,
               metric     = "Accuracy",
               data       = train_dt)
  result_list <- list()
  preds <- predict(fit, newdata = test_dt)
  # List gen
  actuals <- as.factor(test_dt[, get(lab_col)])
  #conf_mat <- confusionMatrix(preds, actuals)
  try(accuracy_table <- table(preds, actuals))
  train_labs <- predict(fit, newdata = train_dt)
  train_labs <- train_dt[, train_preds := train_labs]
  result_list[['preds']] <- preds
  # try(result_list[['conf_mat']] <- conf_mat)
  try(result_list[['accuracy_table']] <- accuracy_table)
  result_list[['train_labs']] <- train_labs
  result_list[['k']] <- as.integer(c(fit$bestTune))
  result_list[['fit']] <- fit
  return(result_list)

}
```

Results

- Best model used Raw Euclidean Distance
- 76% Accuracy is “good” for a test set
- Naive models would predict only 1/6 accuracy
- A better baseline using the majority for a city has a 63% accuracy
- Out of the 95% CI -- Significant
- Still disappointing

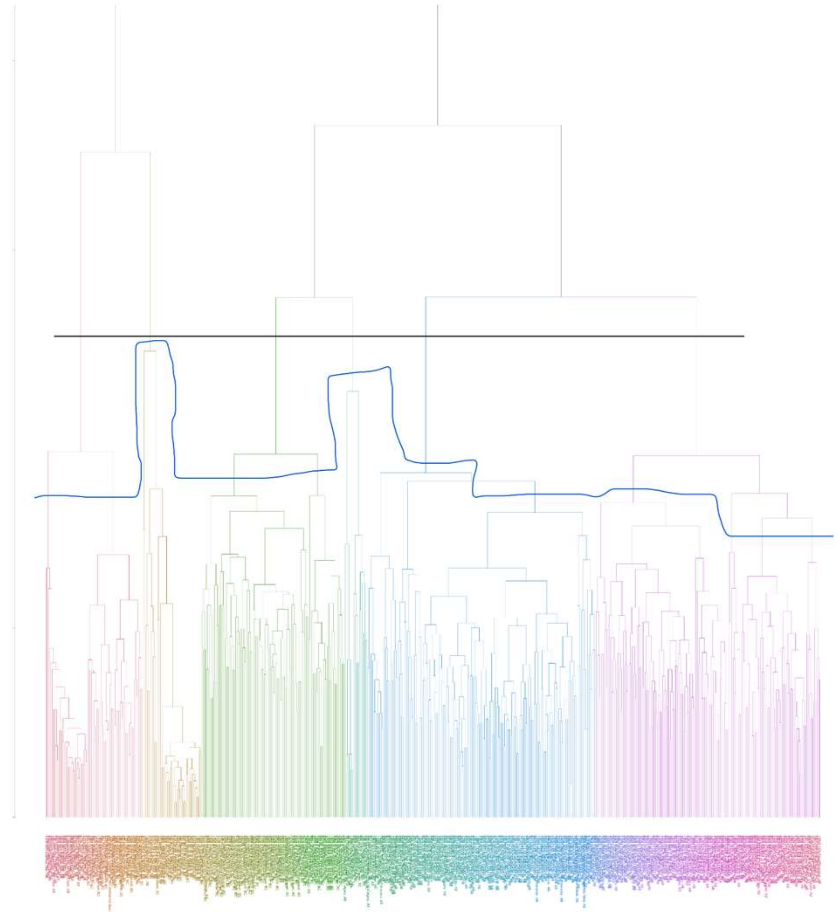
Table 2: KNN Classification Results for Euclidean Raw

actuals preds	$Cluster_1$	$Cluster_2$	$Cluster_3$	$Cluster_4$	$Cluster_5$
$Cluster_1$	32	0	0	0	0
$Cluster_2$	0	20	0	0	1
$Cluster_3$	0	0	29	18	2
$Cluster_4$	0	0	17	28	0
$Cluster_5$	0	0	0	0	10
$Cluster_6$	0	0	1	0	0

```
              Accuracy : 0.761
                95% CI : (0.687, 0.825)
   No Information Rate : 0.2956
P-Value [Acc > NIR] : < 2.2e-16
```

Relationship issues

- Low fidelity of scraped data
  - Not enough granularity on price changes
- Submarkets could be transient due to different pricing strategies over time
- Different cluster levels →



Disappointing Results But…

- More seriously:
  - R is Great
  - I'm happy this project connected me with it again

Questions?

Please keep in mind I did this several months ago so my memory might be a bit weak.