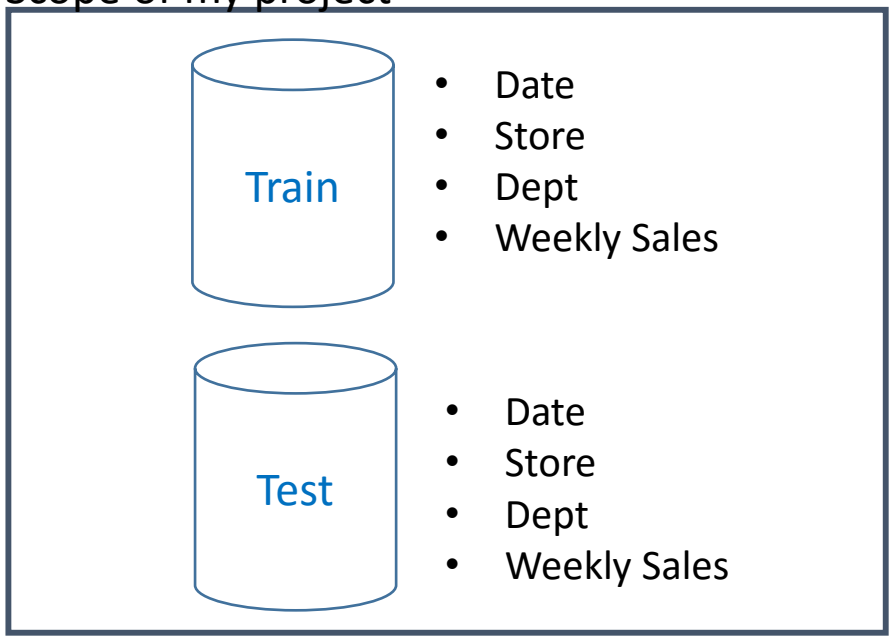
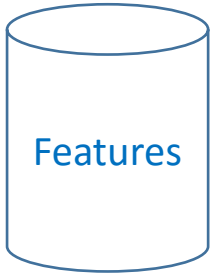
Forecasting Walmart Store Sales using Time series analysis

Daniel Tan

Project Background

Scope of my project





- In 2014, Walmart held a Kaggle competition to challenge Kagglers to build an accurate prediction of future sales based on historical data
- The original data set included weekly sales data for each of Walmart's Stores and Departments, and additional features. For example, Temperature, fuel price etc.
- To scope my project down, I focused on analysing the effect of Dates on Store Sales

Cleaning, Filtering and Reshaping the data

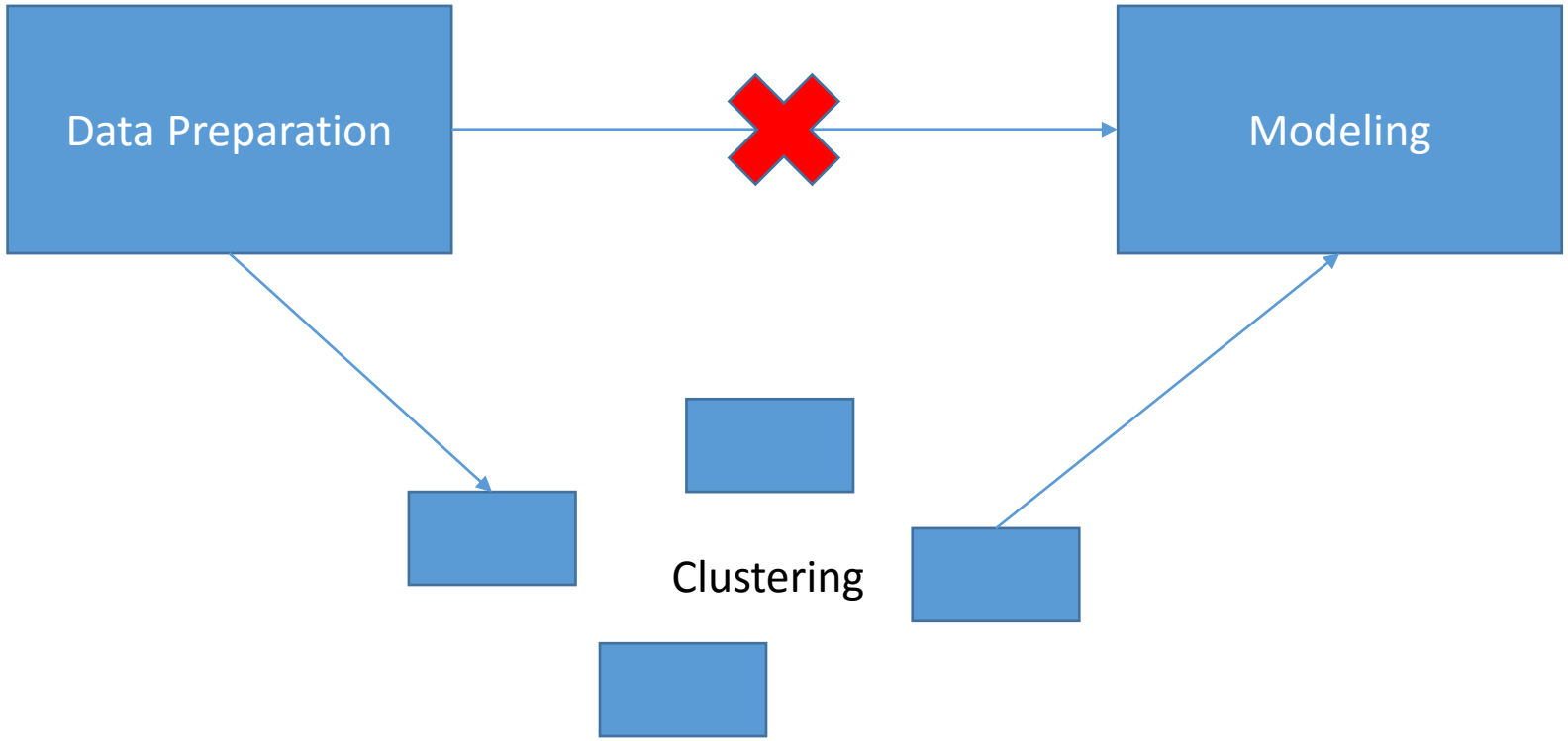
```
# Snippet of train data
train
```

```
## Source: local data frame [421,570 x 5]
##
##     Store   Dept       Date Weekly_Sales IsHoliday
##    (fctr) (fctr)     (date)        (dbl)     (lgl)
## 1       1      1 2010-02-05     24924.50     FALSE
## 2       1      1 2010-02-12     46039.49      TRUE
## 3       1      1 2010-02-19     41595.55     FALSE
## 4       1      1 2010-02-26     19403.54     FALSE
## 5       1      1 2010-03-05     21827.90     FALSE
## 6       1      1 2010-03-12     21043.39     FALSE
## 7       1      1 2010-03-19     22136.64     FALSE
## 8       1      1 2010-03-26     26229.21     FALSE
## 9       1      1 2010-04-02     57258.43     FALSE
## 10      1      1 2010-04-09     42960.91     FALSE
```

	1_1 (dbl)	1_10 (dbl)	1_11 (dbl)	1_12 (dbl)	1_13 (dbl)	1_14 (dbl)	1_16 (dbl)	1_17 (dbl)	1_18 (dbl)	1_19 (dbl)
1	24924.50	30721.50	24213.18	8449.54	41969.29	19466.91	10217.55	13223.76	4729.50	1947.05
2	46039.49	31494.77	21760.75	8654.07	36476.40	18129.02	11873.89	13403.66	19006.50	1490.79
3	41595.55	29634.13	18706.21	9165.98	37857.68	17491.36	13855.54	13485.61	17623.72	1722.17
4	19403.54	27921.96	17306.61	9015.37	37467.32	16118.26	12881.02	10667.06	545.02	1655.32
5	21827.90	33299.27	19082.90	10239.06	40423.95	18268.78	17129.81	12657.65	634.61	1556.65
6	21043.39	28208.00	17864.32	12386.15	35833.07	15331.75	23766.82	12793.17	1275.08	1742.14
7	22136.64	33731.81	19738.42	12917.55	36807.21	12738.78	41742.21	11097.80	2303.36	1881.91
8	26229.21	31406.96	17592.13	11865.53	35431.73	12832.47	26679.52	8900.01	4037.16	2153.61
9	57258.43	31794.04	21762.46	12033.50	38104.53	14333.75	46060.69	9572.23	17916.41	1514.32
10	42960.91	32486.28	22186.81	10109.00	37638.44	12874.07	52976.67	8159.34	8668.06	1880.64

- A quick look through **train** reveals that there are a total of 45 unique store numbers from 1 to 45. And, a total of 81 unique dept numbers from 1 to 99. This creates 3331 unique store-dept pairs. I.e. 3331 time-series, each with a frequency of 143 weeks
- The data also stacked each time series on top of one another
- It required filtering and reshaping, to move it into a form that extracted each time series into a single column (143 weeks x 3331 time series)

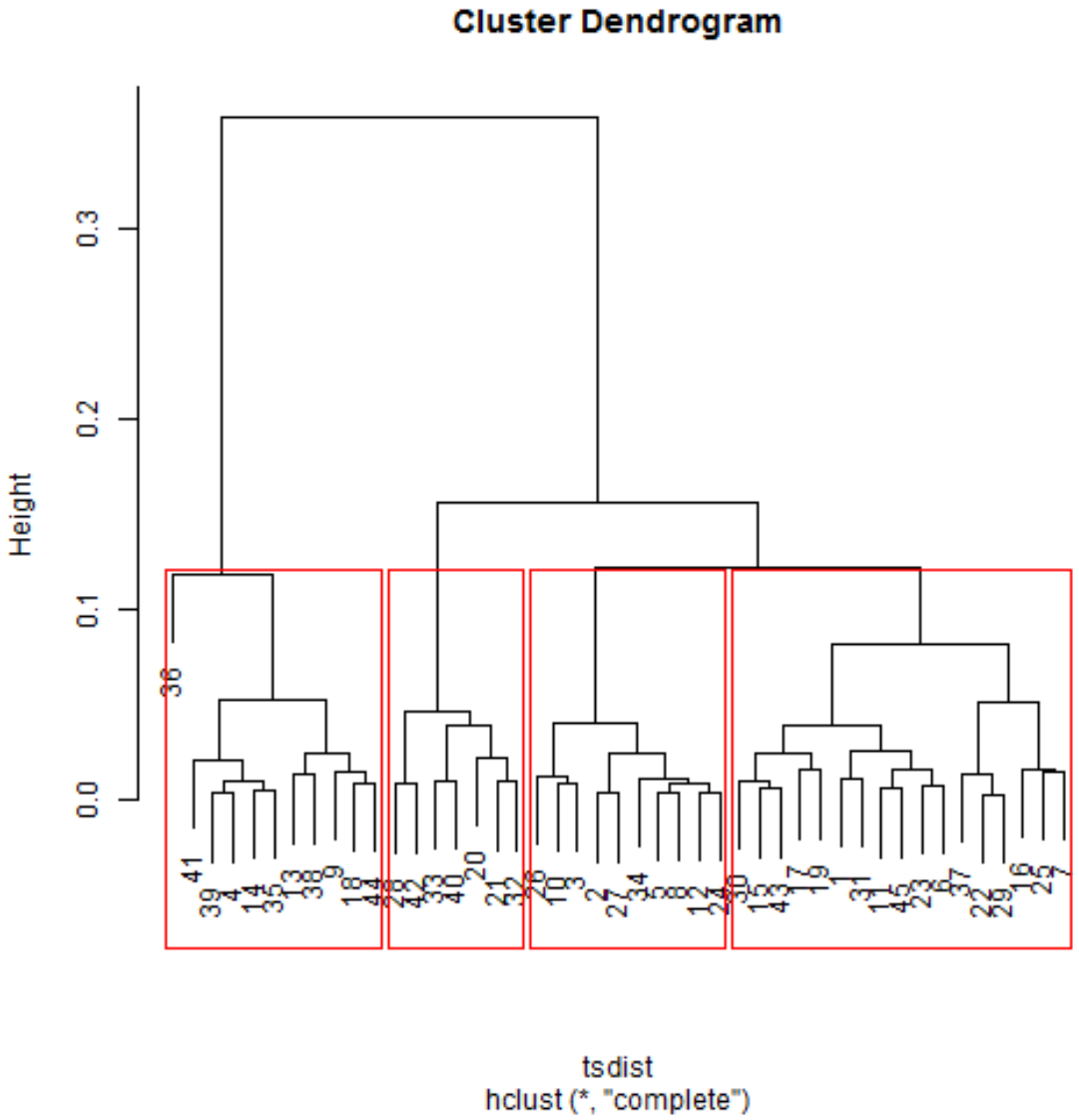
Why I decided to cluster my time series



- My initial intention was to create an individual model for each time series.
- However, I soon found that this approach, which would require looped iterations of a 3331 x 3331 matrix, was too taxing for my computer.
- Therefore, my next step was to figure out how to scope the 3331 time series into manageable clusters.

Clustering using the TSclust library

- First, I aggregated the time series at a store level (143 weeks x 45 stores)
- Next, using the TSclust library, I applied an Autocorrelation (ACF) based dissimilarity calculation to my store matrix.
- This calculation performs a weighted Euclidean distance between 2 time-series, and the resulting output distance can be used as a measure for clustering.
- I ran the distance matrix through hierarchical clustering, and plotted out the dendrogram.
- Upon visual inspection of the dendrogram, I decide to cluster the stores into 4 clusters.



Building an ARIMA model for forecasting

$$\text{ARIMA}\ \underbrace{(p,d,q)}_{\text{Non-seasonal part of the model}}\ \underbrace{(P,D,Q)_m}_{\text{Seasonal part of the model}}$$

- The ARIMA model is a combination of Autoregressive models (AR), Moving average models (MA) and can be extended to non-stationary time series through Integration(I).
- The AR part of ARIMA indicates that the evolving variable of interest is regressed on its own lagged (i.e., prior) values.
- The MA part indicates that the regression error is actually a linear combination error terms whose values occurred contemporaneously and at various times in the past.
- The I (for "integrated") indicates that the data values have been replaced with the difference between their values and the previous values
- The purpose of each of these features is to make the model fit the data as well as possible.
- ARIMA can also be extended to include parameters to determine both trend and seasonal patterns.

Testing for stationarity using the ADF test

- A stationary time series is one whose properties do not depend on the time at which the series is observed
- Before I began to build the ARIMA model, I first tested for stationarity using the Augmented Dickey-Fuller (ADF) test

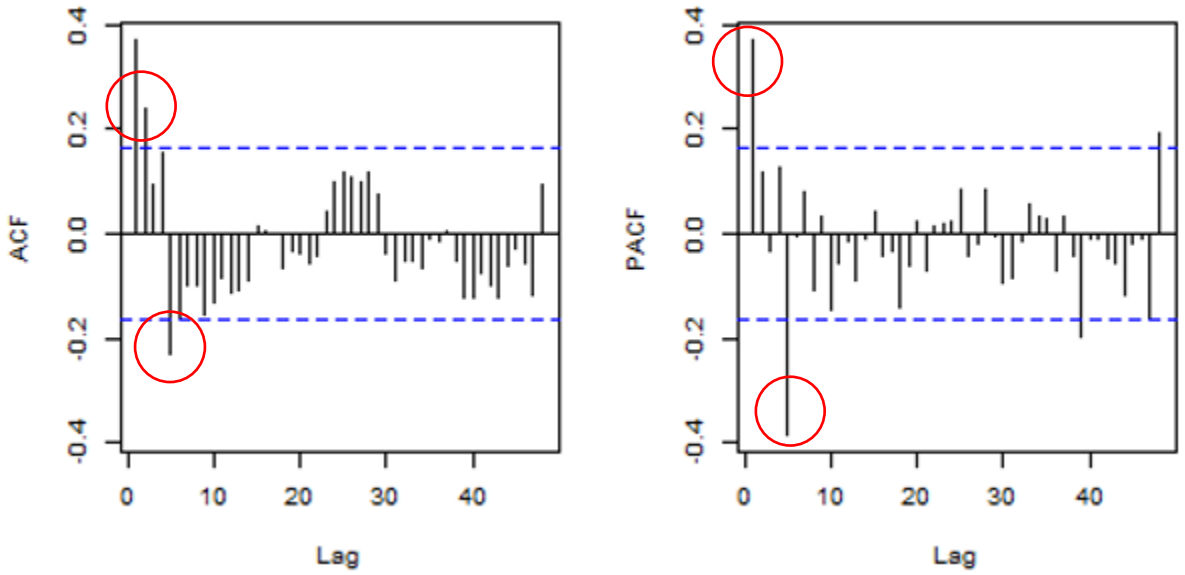
```
#Test for stationarity by performing ADF test
adf.test(cluster1.ts, alternative='stationary')
```

```
## Warning in adf.test(cluster1.ts, alternative = "stationary"): p-value
## smaller than printed p-value
```

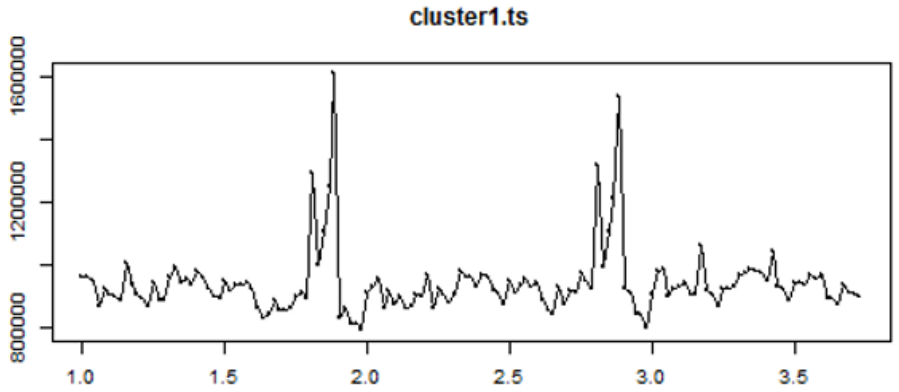
```
##
##  Augmented Dickey-Fuller Test
##
## data:  cluster1.ts
## Dickey-Fuller = -5.279, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary
```

- The results of the test suggest that this time series is stationary, and no differencing is required.

Estimating and calculating appropriate AR and MA coefficients



- ACF is a bar chart of the coefficients of correlation between a time series and lags of itself
- PACF is a plot of the partial correlation coefficients between the series and lags of itself
- From the plot, the PACF and ACF lag orders whose values which cross the confidence boundaries, are candidates for the AR and MA coefficients respectively.
- The ACF plot suggests lag order 1, 2 and 5 are suitable candidates for the MA coefficient (q).
- he PACF plot suggests lag order 1 and 5 are suitable candidates for the AR coefficient (p).



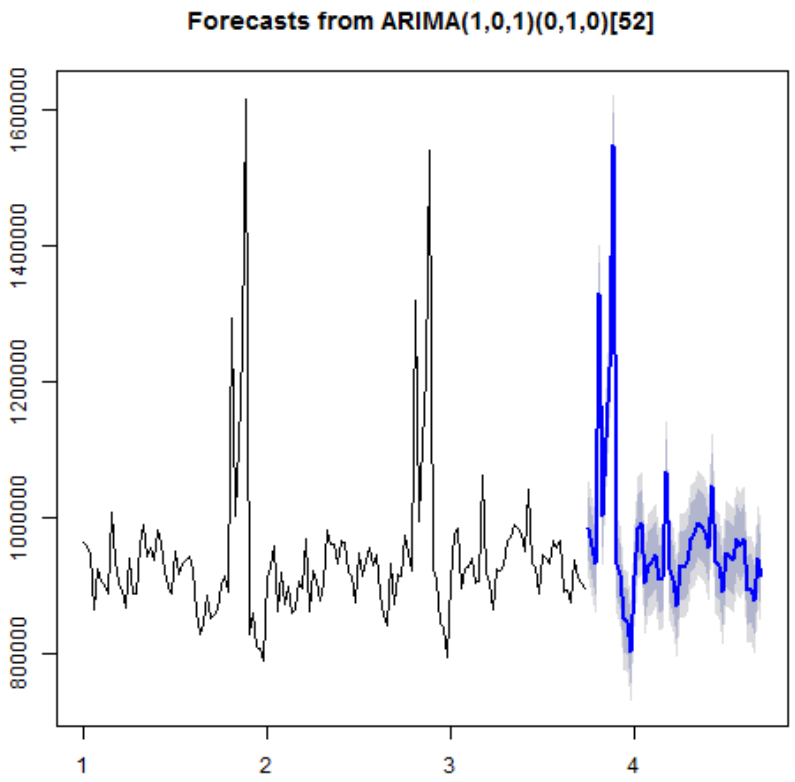
- The plot of the time series shows a clear seasonal pattern
- To account for this seasonality in my ARIMA model, I will apply a seasonal differencing for a period of 52 weeks.
- i.e.

$$(P, D, Q)_m = (0,1,0)[52]$$

Using the ARIMA model to forecast future sales values

```
Arima(cluster1.ts,order=c(1,0,1), seasonal = list(order = c(0,1,0), period = 52), include.mean = FALSE)
```

```
## Series: cluster1.ts
## ARIMA(1,0,1)(0,1,0)[52]
##
## Coefficients:
##          ar1      ma1
##       0.9733  -0.8749
## s.e.  0.0295   0.0627
##
## sigma^2 estimated as 1.326e+09:  log likelihood=-1084.13
## AIC=2174.26   AICc=2174.54   BIC=2181.79
```



- Based on the candidates for AR & MA, I loop through my candidates to find the combination of p, d and q that will result in the lowest Akaike's Information Criterion (AIC) score.
- AIC is a likelihood estimation on the measure of how accurate the ARIMA model fits with the data.

- The best ARIMA model is then plotted as a forecast to show the expected sales for the next h=50 weeks.
- The dark and light shaded areas represent the 80% and 95% prediction intervals.

Evaluating the accuracy of the forecasts

Local testing

- Splitting my train data set (143 weeks), into my own mini-train(120 weeks) and mini-test(23 weeks) data set for local testing.

Mean Absolute Percentage Error (MAPE)

$$\left(\frac{1}{n}\sum\frac{|Actual - Forecast|}{|Actual|}\right)*100$$

Results

```
##   Cluster     MAPE
## 1       1 5.837927
## 2       2 5.824512
## 3       3 5.570019
## 4       4 6.833386
```

The MAPE is roughly 5-6%, which is satisfactory for a small, local test.

Testing on the Kaggle Platform

- Test full train data by uploading on for a ranked score on public leader board
- Benchmark ARIMA model against 2 simpler models:
  - **Seasonal Naive**: A simple forecast that uses last year's sales for predicting next year's sales
  - **Linear Regression Model**: Computes a forecast using linear regression and seasonal dummy variables

Weighted Mean Absolute Error (WMAE)

$$\text{WMAE} = \frac{1}{\sum w_i}\sum_{i=1}^{n} w_i|y_i - \hat{y}_i|$$

w = 5 if the week is a holiday week, 1 otherwise

Results

65	↓4	statlearning	2839.58433	125	Mon, 05 May 2014 14:39:34 (-0h)
-		**DanielTan**	**2845.22845**	-	**Tue, 16 Aug 2016 14:31:46** Post-Deadline

Post-Deadline Entry
If you would have submitted this entry during the competition, you would have been around here on the leaderboard.

Conclusion

Recap of this project:
1. I took Walmart's Kaggle competition data and, filtered and reshaped it into separate time series
2. I took a measure of the similarity of these time series to each other, and clustered them into 4 individual time series
3. For each time series, I applied exploration and analysis to fit an appropriate ARIMA model
4. I tested the accuracy of each ARIMA model against a small sample of local data, and the official test data provided on Kaggle

Problems encountered	**Solution**
1. Dataset was too large to be computed on my system	Used time series clustering to scope down the size of the data in a way that minimizes confidence loss
2. I was unprepared for the technical complexity of clustering and modeling time series data accurately	Adopted a "Done is better than perfect" mentality, and constantly sought out advice on intelligent ways to scope my problem into something more managable
3. Actual test data was hidden behind Kaggle	Split my train data into smaller subsets of train and test data that could be used to measure accuracy on my local system

Acknowledgements

People:

1. Anirban Ghosh (Springboard mentor)

2. William TJVI

3. Peh Shu Ming

Resources:

1. Forecasting: Principles and Practice

2. David Thaler’s winning entry for the actual Kaggle competition

3. Duke University's notes on ARIMA order selection

4. A little book of R for time series

5. Time series clustering from AusDM R and Data Mining Workshop

Please head over to https://github.com/ddanieltan/Kaggle_Walmart/blob/master/summary.md for links to these resources