Programming with Data

Session 6: Forecasting Walmart Sales

Dr. Wang Jiwei
Master of Professional Accounting

Case: Walmart Store Sales Forecasting

The question

> How can we predict weekly departmental revenue for Walmart, leveraging our knowledge of Walmart, its business, and some limited historical information

- Check out the Kaggle competition
- Predict weekly for 115,064 (Store, Department, Week) tuples
  - From 2012-11-02 to 2013-07-26: test dataset
- Using [incomplete] weekly revenue data from 2010-02-05 to 2012-11-01
  - By department (some weeks missing for some departments): training dataset

More specifically...

- Consider time dimensions
  - What matters:
    - Time of the year?
    - Holidays?
    - Do different stores or departments behave differently?
- Wrinkles:
  - Walmart won't give us weekly sales in the test data
    - But they'll tell us how well the algorithm performs when we submit the forecasts to Kaggle
  - We can't use past week sales for prediction because we won't have it for most of the prediction in the testing data...

Load data and packages

```
library(tidyverse)  # we'll extensively use dplyr here
library(lubridate)  # Great for simple date functions
library(broom) # Display regression results in a tidy way
weekly <- read.csv("Data/Session_6_WMT_train.csv")
weekly.test <- read.csv("Data/Session_6_WMT_test.csv")
weekly.features <- read.csv("Data/Session_6_WMT_features.csv")
weekly.stores <- read.csv("Data/Session_6_WMT_stores.csv")
```

- `weekly` is our training data
- `weekly.test` is our testing data -- no `Weekly_Sales` column
- `weekly.features` is general information about (week, store) pairs
    - Temperature, pricing, etc.
- `weekly.stores` is general information about each store

The data

- Revenue by week for each department of each of 45 stores
  - Department is just a number between 1 and 99
  - Date of that week
  - If the week is considered a holiday for sales purposes
    - Super Bowl (first Sunday in February), Labor Day (first Monday in September), Black Friday (fourth Friday of November), Christmas
- Store data:
  - Which store the data is for, 1 to 45
  - Store type (A, B, or C)
  - Store size
- Other data, by week and location:
  - Temperature, gas price, markdown, CPI, Unemployment, Holidays

The training data

```
## Rows: 421,570
## Columns: 5
## $ Store        <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
## $ Dept         <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
## $ Date         <chr> "2010-02-05", "2010-02-12", "2010-02-19", "2010-02-26", "~
## $ Weekly_Sales <dbl> 24924.50, 46039.49, 41595.55, 19403.54, 21827.90, 21043.3~
## $ IsHoliday    <lgl> FALSE, TRUE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FA~
```

```
##   Store Dept       Date Weekly_Sales IsHoliday
## 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
```

```
##      Store          Dept           Date            Weekly_Sales
##  Min.   : 1.0   Min.   : 1.00   Length:421570      Min.   : -4989
##  1st Qu.:11.0   1st Qu.:18.00   Class :character   1st Qu.:  2080
##  Median :22.0   Median :37.00   Mode  :character   Median :  7612
##  Mean   :22.2   Mean   :44.26                      Mean   : 15981
##  3rd Qu.:33.0   3rd Qu.:74.00                      3rd Qu.: 20206
##  Max.   :45.0   Max.   :99.00                      Max.   :693099
##  IsHoliday
##  Mode :logical
##  FALSE:391909
##  TRUE :29661
##
##
##
```

Walmart's evaluation metric

- Walmart uses MAE (mean absolute error), but with a twist:
  - They care more about holidays, so any error on holidays has **5 times** the penalty
  - They call this WMAE, for *weighted* mean absolute error

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

- n is the number of test data points
- $\hat{y}_i$ is your prediction
- y_i is the actual sales
- w_i is 5 on holidays and 1 otherwise

```r
# Construct a function in R to calculate WMAE
wmae <- function(actual, predicted, holidays) {
  sum(abs(actual - predicted) * (holidays * 4 + 1), na.rm = TRUE) /
    (length(actual) + 4 * sum(holidays))
}
```

Before we get started...

- The data isn't very clean:
  - Markdowns are given by 5 separate variables instead of 1
  - Date is text format instead of a date
  - CPI and unemployment data are missing in around a third of the training data
  - There are some (week, store, department) groups missing from our training data!
- Some features to add:
  - Year
  - Week
  - A unique ID for tracking: (store-department-week) tuples
  - The ID Walmart requests we use for submissions: "1_1_2012-11-02"
  - Average sales by (store, department)
  - Average sales by (week, store, department)

Data cleaning

```r
preprocess_data <- function(df) {
  # Merge the data together (Pulled data from outside of function -- "scoping")
  # https://bookdown.org/rdpeng/rprogdatascience/scoping-rules-of-r.html
  df <- left_join(df, weekly.stores)
  # last col 'isHoliday' is already in train data, join the first 11 col only.
  df <- left_join(df, weekly.features[ , 1:11])
  # I am not sure what exactly the five markdowns represent
  # All missing markdowns will be assigned to 0 and record the last non-missing
  df$markdown <- 0
  df[!is.na(df$MarkDown1), ]$markdown <- df[!is.na(df$MarkDown1), ]$MarkDown1
  df[!is.na(df$MarkDown2), ]$markdown <- df[!is.na(df$MarkDown2), ]$MarkDown2
  df[!is.na(df$MarkDown3), ]$markdown <- df[!is.na(df$MarkDown3), ]$MarkDown3
  df[!is.na(df$MarkDown4), ]$markdown <- df[!is.na(df$MarkDown4), ]$MarkDown4
  df[!is.na(df$MarkDown5), ]$markdown <- df[!is.na(df$MarkDown5), ]$MarkDown5
  # Fix dates and add useful time variables
  df$date <- as.Date(df$Date)
  df$week <- week(df$date)
  df$year <- year(df$date)
  df
}
```

```r
df <- preprocess_data(weekly)
df[df$Weekly_Sales < 0, ]$Weekly_Sales <- 0
df_test <- preprocess_data(weekly.test)
```

> Model may perform better without using markdown

What this looks like

```
df[91:94, ] %>%
  select(Store, date, markdown, MarkDown3, MarkDown4, MarkDown5) %>%
  html_df()
```

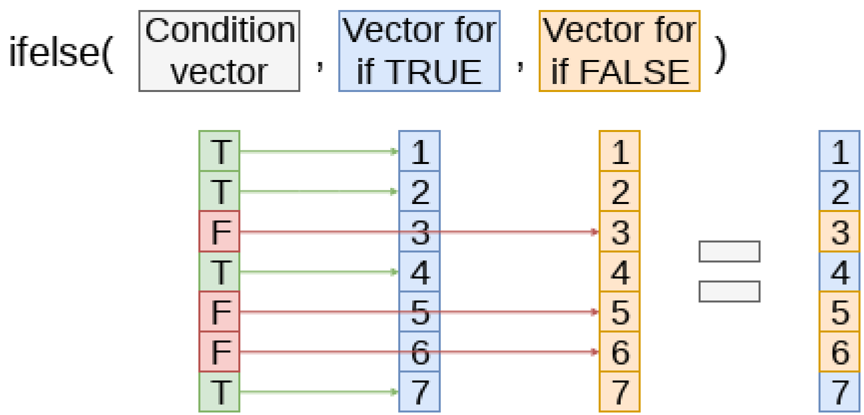
	Store	date	markdown	MarkDown3	MarkDown4	MarkDown5
91	1	2011-10-28	0.00	NA	NA	NA
92	1	2011-11-04	0.00	NA	NA	NA
93	1	2011-11-11	6551.42	215.07	2406.62	6551.42
94	1	2011-11-18	5988.57	51.98	427.39	5988.57

```
df[1:2, ] %>% select(date, week, year) %>% html_df()
```

date	week	year
2010-02-05	6	2010
2010-02-12	7	2010

Cleaning: Missing CPI and Unemployment

```r
# Fill in missing CPI and Unemployment data
df_test <- df_test %>%
  group_by(Store, year) %>%
  mutate(CPI = ifelse(is.na(CPI), mean(CPI, na.rm = T), CPI),
         Unemployment = ifelse(is.na(Unemployment),
                               mean(Unemployment, na.rm = T),
                               Unemployment)) %>%
  ungroup()
```



> Apply the (store, year)'s average CPI and average Unemployment to missing data

Cleaning: Adding IDs

- Build a unique ID
  - Since store, week and department are all 2 digits, make a 6 digit number with 2 digits for each
    - `sswwdd`
- Build Walmart's requested ID for submissions
  - `ss_dd_YYYY-MM-DD`

```
# Unique IDs in the data
df$id <- df$Store *10000 + df$week * 100 + df$Dept
df_test$id <- df_test$Store *10000 + df_test$week * 100 + df_test$Dept

# Unique ID and factor building
swd <- c(df$id, df_test$id)  # Pool all IDs
swd <- unique(swd)  # Only keep unique elements
swd <- data.frame(id = swd)  # Make a data frame
swd$swd <- factor(swd$id)  # Extract factors for using later

# Add unique factors to data -- ensures same factors for both data sets
df <- left_join(df, swd)
df_test <- left_join(df_test, swd)
```

```
df_test$Id <- paste0(df_test$Store, '_', df_test$Dept, "_", df_test$date)
```

What the IDs look like

```r
# id: numerical
# swd: factor
# Id: character
html_df(df_test[c(20000, 40000, 60000),
                c("Store", "week", "Dept", "id", "swd", "Id")])
```

Store	week	Dept	id	swd	Id
8	27	33	82733	82733	8_33_2013-07-05
15	46	91	154691	154691	15_91_2012-11-16
23	52	25	235225	235225	23_25_2012-12-28

Add in (store, department) average sales

```r
# Calculate average sales by store-dept
df <- df %>%
  group_by(Store, Dept) %>%
  mutate(store_avg = mean(Weekly_Sales, rm.na = T)) %>%
  ungroup()
# Select the first average sales data for each store-dept
df_sa <- df %>%
  group_by(Store, Dept) %>%
  slice(1) %>% # Select rows by position
  select(Store, Dept, store_avg) %>%
  ungroup()
# Distribute the store-dept average sales to the testing data
df_test <- left_join(df_test, df_sa)
```

```
## Joining, by = c("Store", "Dept")
```

```r
# 36 observations have messed up department codes -- ignore (set to 0)
df_test[is.na(df_test$store_avg), ]$store_avg  <- 0

# Calculate multipliers based on store_avg (and removing NaN and Inf)
df$Weekly_mult <- df$Weekly_Sales / df$store_avg
df[!is.finite(df$Weekly_mult), ]$Weekly_mult <- NA
```

Add in (week, store, dept) average sales

```
# Calculate mean by week-store-dept and distribute to df_test
df <- df %>%
  group_by(Store, Dept, week) %>%
  mutate(naive_mean = mean(Weekly_Sales, rm.na = T)) %>%
  ungroup()
df_wm <- df %>%
  group_by(Store, Dept, week) %>%
  slice(1) %>%
  ungroup() %>%
  select(Store, Dept, week, naive_mean)
df_test <- df_test %>% arrange(Store, Dept, week)
df_test <- left_join(df_test, df_wm)
```

```
## Joining, by = c("Store", "Dept", "week")
```

ISSUE: New (week, store, dept) groups

- This is in our testing data!
  - So we'll need to predict out groups we haven't observed at all

```
table(is.na(df_test$naive_mean))
```

```
##
##  FALSE   TRUE
## 113827   1237
```

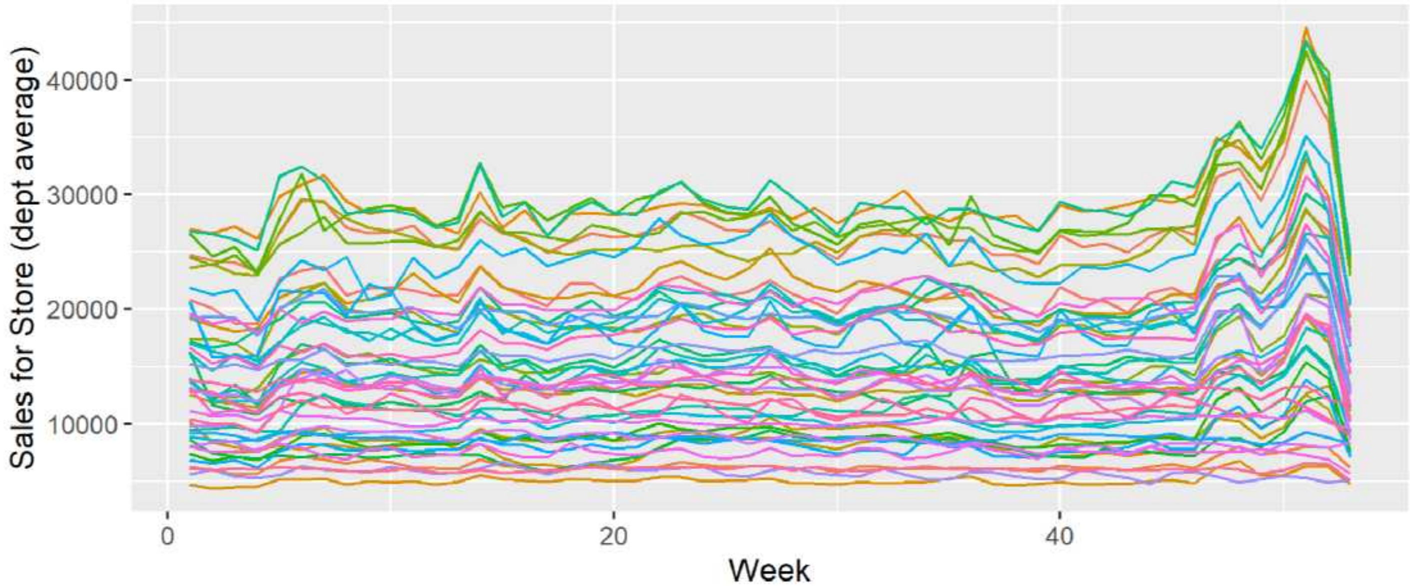
- Fix: Fill with 1 or 2 lags where possible using `ifelse()` and `lag()`
- Fix: Fill with 1 or 2 leads where possible using `ifelse()` and `lead()`
- Fill with `store_avg` when the above fail
- Code is available in the code file -- a bunch of code like:

```
df_test <- df_test %>%
  arrange(Store, Dept, date) %>%
  group_by(Store, Dept) %>%
  mutate(naive_mean=ifelse(is.na(naive_mean), lag(naive_mean), naive_mean)) %>%
  ungroup()
```

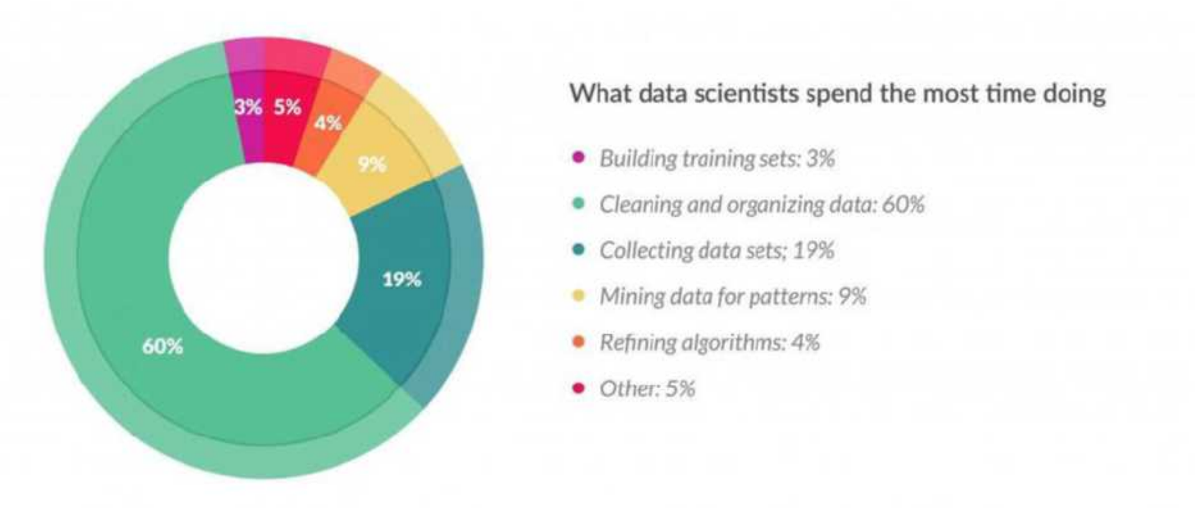
Cleaning is done

- Data is in order
  - No missing values where data is needed
  - Needed values created

```
df %>%
  group_by(week, Store) %>%
  mutate(sales = mean(Weekly_Sales)) %>% slice(1) %>% ungroup() %>%
  ggplot(aes(y = sales, x = week, color = factor(Store))) +
  geom_line() + xlab("Week") + ylab("Sales for Store (dept average)") +
  theme(legend.position = "none") # remove the plot legend
```



How much time on data prep?



The Survey

Feature engineering techniques

> There are many ways to prepare data. You may read the following articles for a summary of typical feature engineering techniques. We will apply more techniques in future topics.

Fundamental Techniques of Feature Engineering for Machine Learning

The Hitchhiker's Guide to Feature Extraction

Tackling the problem

First try

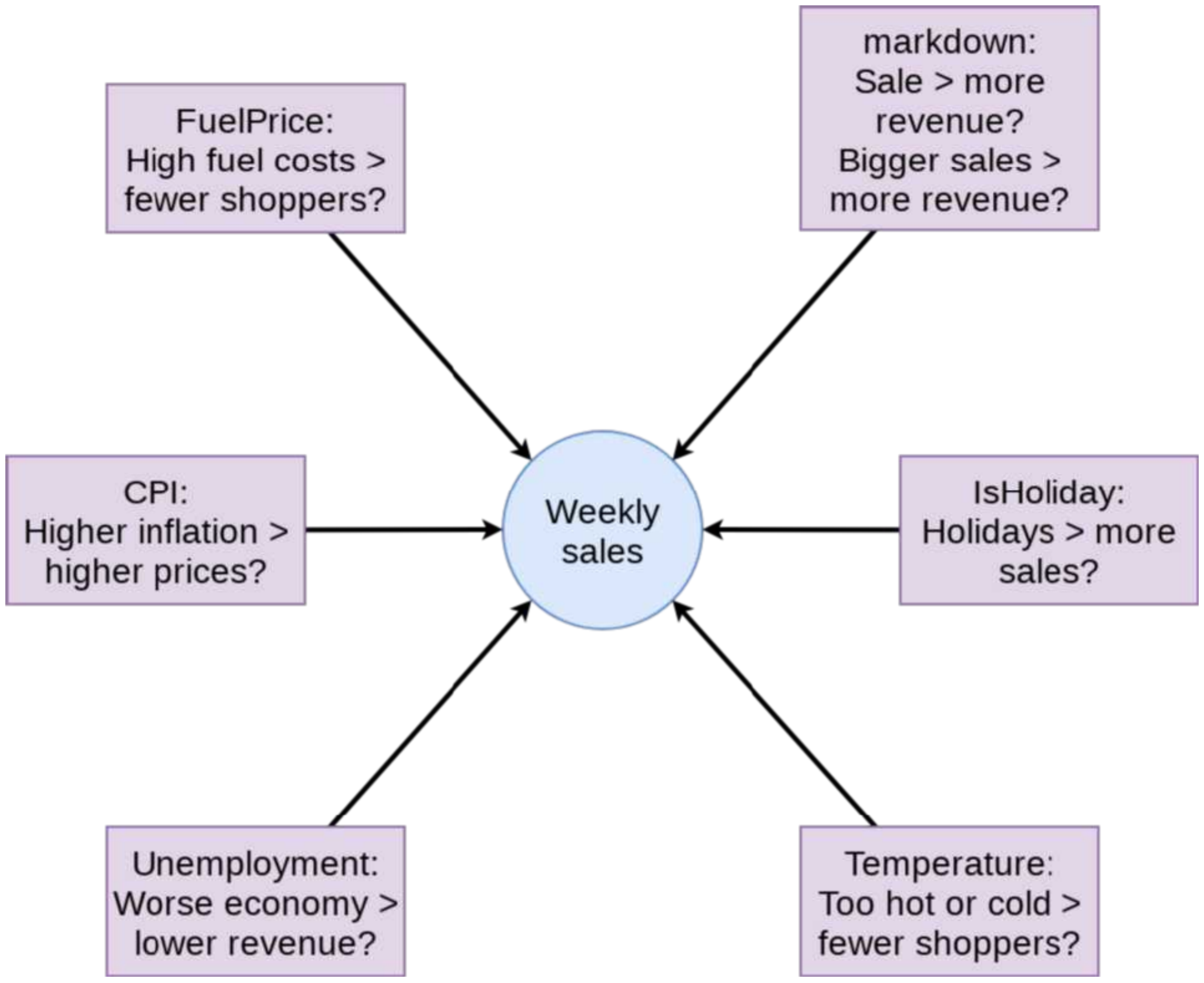
- Ideal: Use last week to predict next week!

  - No data for testing...

- First instinct: try to use a linear regression to solve this

  - We have this

What to put in the model?



First model

```
mod1 <- lm(Weekly_mult ~ factor(IsHoliday) + factor(markdown > 0) +
                         markdown + Temperature +
                         Fuel_Price + CPI + Unemployment,
           data = df)
tidy(mod1)
```

```
## # A tibble: 8 x 5
##   term                       estimate   std.error statistic   p.value
##   <chr>                         <dbl>       <dbl>     <dbl>     <dbl>
## 1 (Intercept)                1.25        0.0100       125.   0       
## 2 factor(IsHoliday)TRUE      0.0597      0.00337       17.7  2.00e- 70
## 3 factor(markdown > 0)TRUE   0.0486      0.00240       20.3  3.42e- 91
## 4 markdown                   0.000000697 0.000000237    2.94 3.32e-  3
## 5 Temperature               -0.000832    0.0000490    -17.0  1.16e- 64
## 6 Fuel_Price                -0.0721      0.00223      -32.3  1.23e-228
## 7 CPI                       -0.0000842   0.0000241     -3.50 4.67e-  4
## 8 Unemployment               0.00406     0.000494       8.22 1.97e- 16
```

```
glance(mod1)
```

```
## # A tibble: 1 x 12
##   r.squared adj.r.squared sigma statistic p.value    df   logLik     AIC     BIC
##       <dbl>         <dbl> <dbl>     <dbl>   <dbl> <dbl>    <dbl>   <dbl>   <dbl>
## 1   0.00556       0.00554 0.549      337.       0     7 -345649. 691317. 691415.
## # ... with 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

Prep submission and in-sample WMAE

```
# Out of sample result
df_test$Weekly_mult <- predict(mod1, df_test)
df_test$Weekly_Sales <- df_test$Weekly_mult * df_test$store_avg

# Required to submit a csv of Id and Weekly_Sales
write.csv(df_test[ , c("Id", "Weekly_Sales")], "WMT_linear.csv",
          row.names = FALSE)

# track
df_test$WS_linear <- df_test$Weekly_Sales

# Check in sample WMAE
df$WS_linear <- predict(mod1, df) * df$store_avg
w <- wmae(actual = df$Weekly_Sales, predicted = df$WS_linear,
          holidays = df$IsHoliday)
names(w) <- "Linear"
wmaes <- c(w)
wmaes
```

```
##   Linear
## 3040.644
```

Performance for linear model

Your most recent submission

Name	Submitted	Wait time	Execution time	Score
WMT_linear.csv	just now	1 seconds	1 seconds	4954.44928

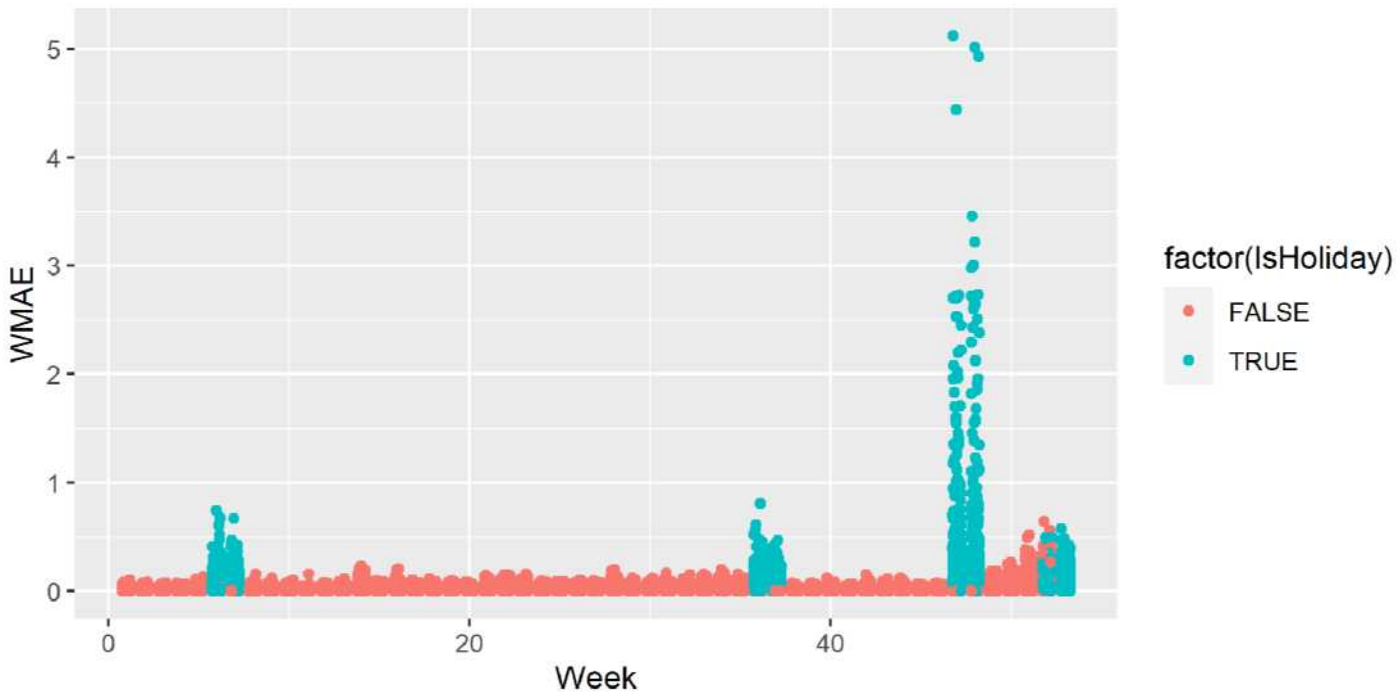
Complete

Jump to your position on the leaderboard

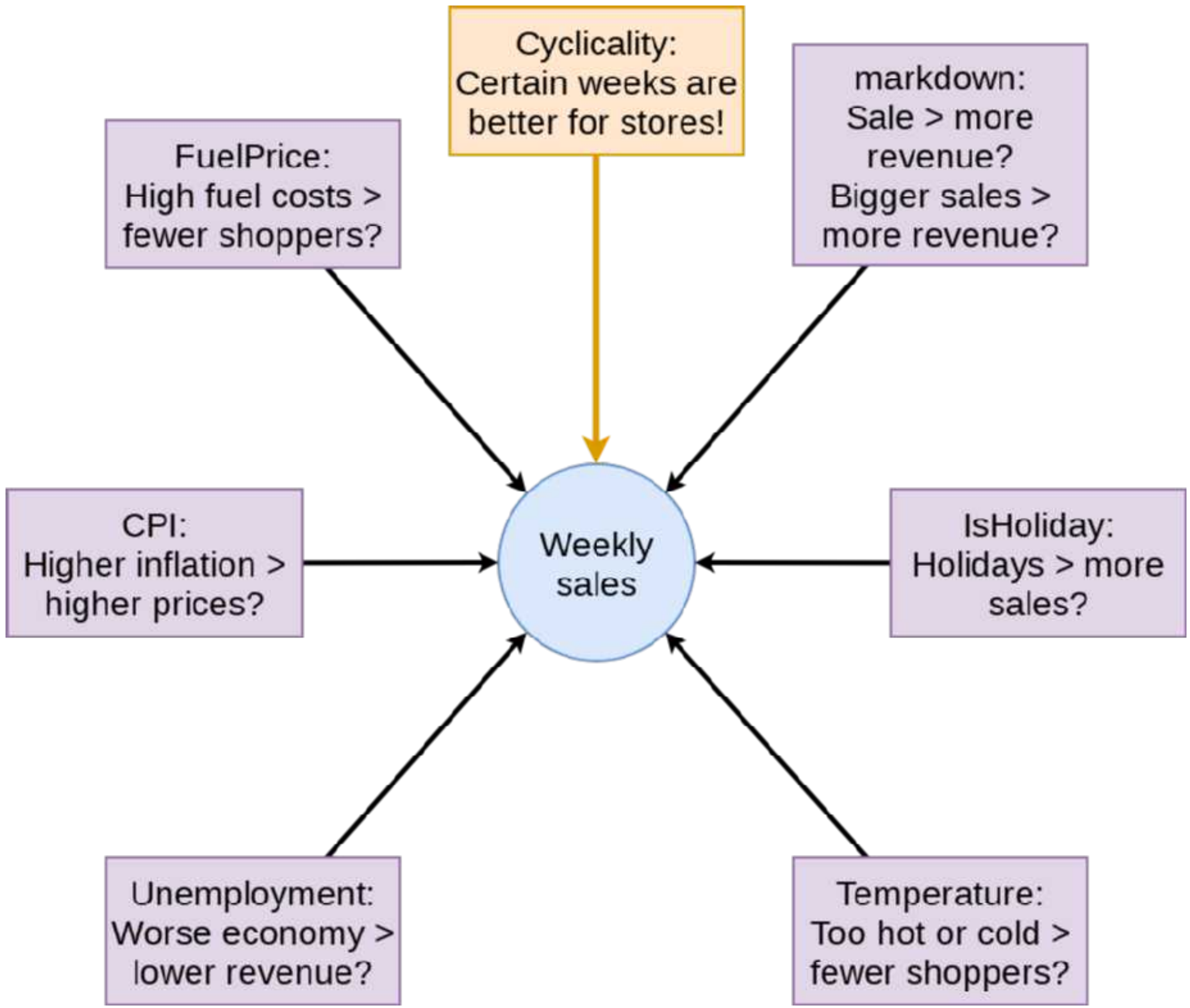
428	▾1	Bill Szaroletta, P.E.	4949.29906	2	5y
429	▴1	Kalanand Mishra	4961.02377	9	5y
430	▴3	TBarker	4970.66978	8	5y
431	▾3	Yogesh Bhalerao	4972.22957	1	5y

Visualizing in-sample WMAE

```
# compute WMAE for each obs
wmae_obs <- function(actual, predicted, holidays) {
  abs(actual - predicted) * (holidays * 4 + 1) /
    (length(actual) + 4 * sum(holidays))
}
df$wmaes  <- wmae_obs(actual = df$Weekly_Sales, predicted = df$WS_linear,
                      holidays = df$IsHoliday)
ggplot(data = df, aes(y = wmaes, x = week, color = factor(IsHoliday))) +
  geom_jitter(width = 0.25) + xlab("Week") + ylab("WMAE")
```



Back to the drawing board...



Second model: Including week

```r
mod2 <- lm(Weekly_mult ~ factor(week) + factor(IsHoliday) + factor(markdown>0) +
             markdown + Temperature + Fuel_Price + CPI + Unemployment, data=df)
tidy(mod2)
```

```
## # A tibble: 60 x 5
##    term            estimate std.error statistic   p.value
##    <chr>              <dbl>     <dbl>     <dbl>     <dbl>
##  1 (Intercept)       1.01      0.0119      84.6 0
##  2 factor(week)2    -0.0604    0.00982     -6.16 7.48e- 10
##  3 factor(week)3    -0.0668    0.00983     -6.80 1.05e- 11
##  4 factor(week)4    -0.0911    0.00983     -9.27 1.93e- 20
##  5 factor(week)5     0.0432    0.00981      4.41 1.06e-  5
##  6 factor(week)6     0.166     0.00953     17.4  5.68e- 68
##  7 factor(week)7     0.227     0.00910     25.0  8.90e-138
##  8 factor(week)8     0.101     0.00896     11.3  1.09e- 29
##  9 factor(week)9     0.0722    0.00897      8.05 8.15e- 16
## 10 factor(week)10    0.0830    0.00899      9.23 2.63e- 20
## # ... with 50 more rows
```

```r
glance(mod2)
```

```
## # A tibble: 1 x 12
##   r.squared adj.r.squared sigma statistic p.value    df   logLik     AIC     BIC
##       <dbl>         <dbl> <dbl>     <dbl>   <dbl> <dbl>    <dbl>   <dbl>   <dbl>
## 1    0.0642        0.0640 0.533      490.       0    59 -332843. 665808. 666476.
## # ... with 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

Prep submission and in-sample WMAE

```r
# Out of sample result
df_test$Weekly_mult <- predict(mod2, df_test)
df_test$Weekly_Sales <- df_test$Weekly_mult * df_test$store_avg

# Required to submit a csv of Id and Weekly_Sales
write.csv(df_test[ , c("Id", "Weekly_Sales")], "WMT_linear2.csv",
          row.names = FALSE)

# track
df_test$WS_linear2 <- df_test$Weekly_Sales

# Check in sample WMAE
df$WS_linear2 <- predict(mod2, df) * df$store_avg
w <- wmae(actual = df$Weekly_Sales, predicted = df$WS_linear2,
          holidays = df$IsHoliday)
names(w) <- "Linear 2"
wmaes <- c(wmaes, w)
wmaes
```

```
##   Linear Linear 2
## 3040.644 3208.144
```

Performance for linear model 2

Your most recent submission

Name	Submitted	Wait time	Execution time	Score
WMT_linear2.csv	10 minutes ago	97 seconds	1 seconds	5540.29197

Complete

Jump to your position on the leaderboard ▾

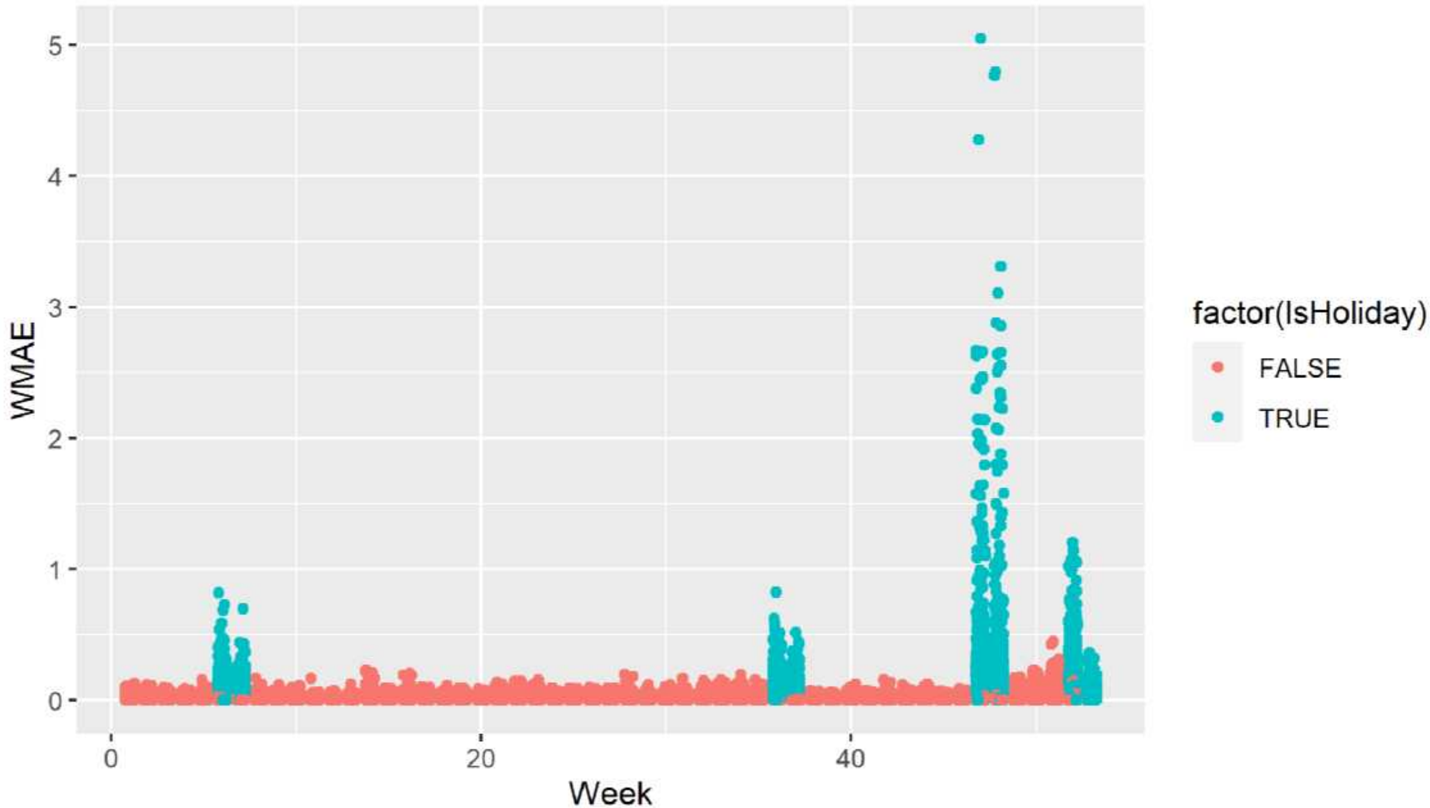
465	▲3	Bullet Bill	5514.16117	25	5y
466	—	Jesus Fernandez-Bes	5547.45068	12	5y
467	▼3	Carmine Genovese	5553.17509	8	5y
468	▲4	27685	5694.66116	5	5y

```
wmaes_out
```

```
##   Linear Linear 2
##   4954.4   5540.3
```

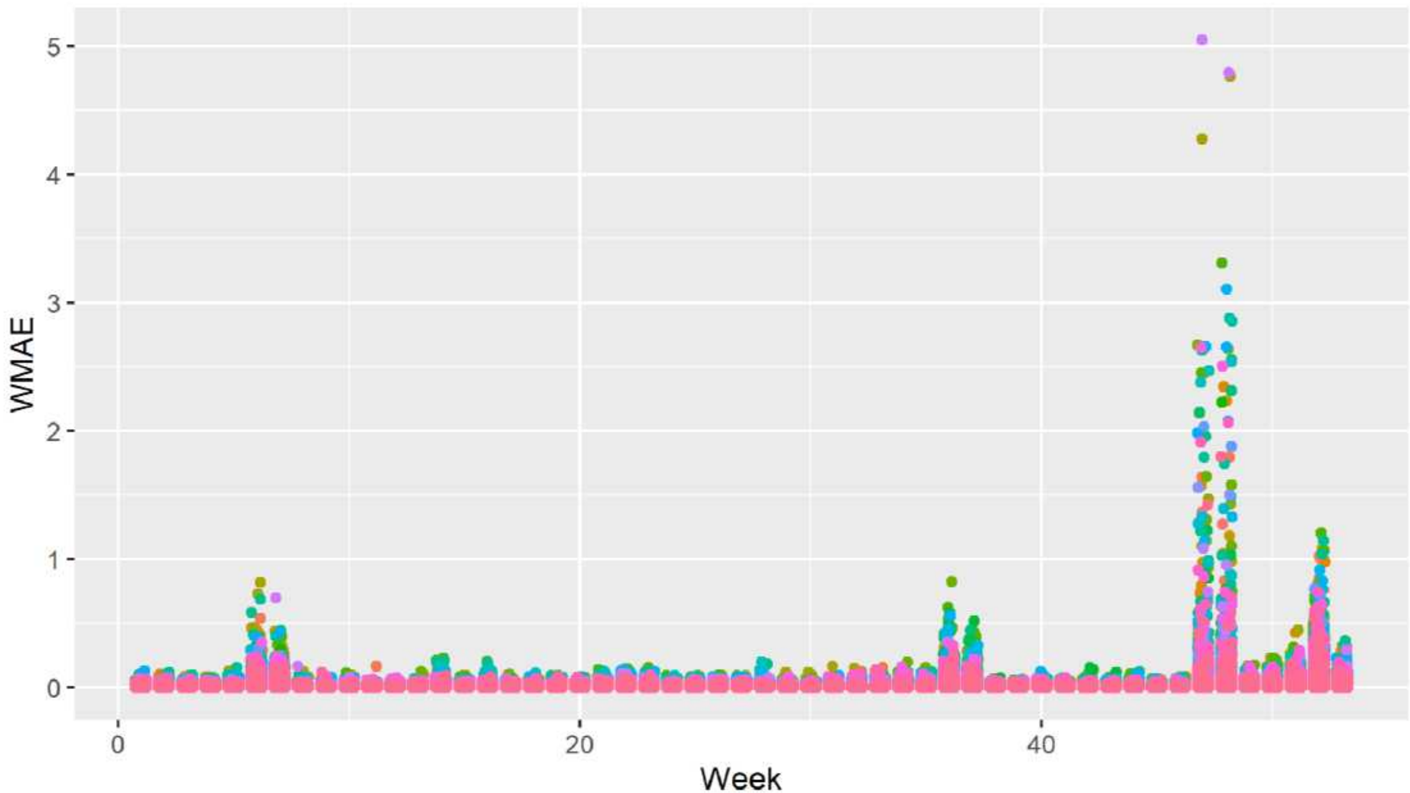
Visualizing in-sample WMAE

```r
df$wmaes  <- wmae_obs(actual = df$Weekly_Sales, predicted = df$WS_linear2,
                      holidays = df$IsHoliday)
ggplot(data=df, aes(y = wmaes, x = week,  color = factor(IsHoliday))) +
  geom_jitter(width = 0.25) + xlab("Week") + ylab("WMAE")
```



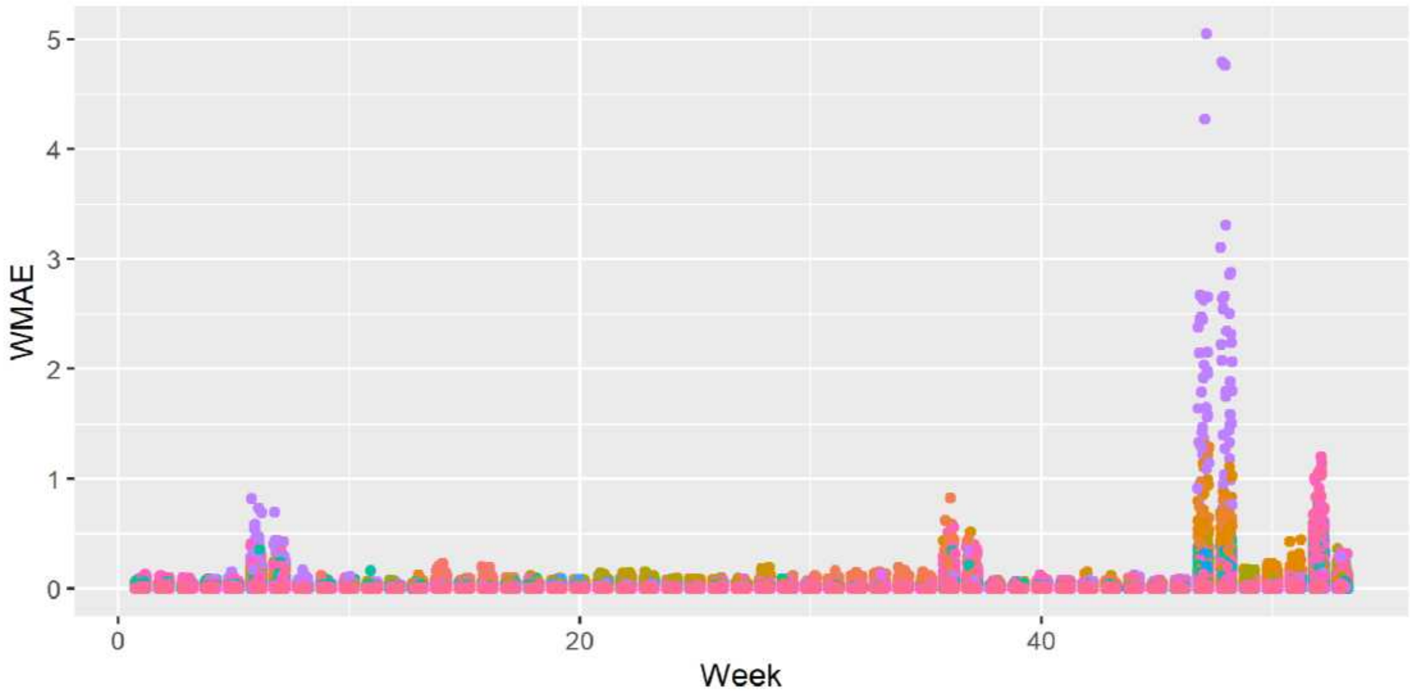
Visualizing in-sample WMAE by Store

```
ggplot(data=df, aes(y = wmae_obs(Weekly_Sales, WS_linear2, IsHoliday),
                    x = week, color = factor(Store))) +
  geom_jitter(width = 0.25) + xlab("Week") + ylab("WMAE") +
  theme(legend.position = "none")
```

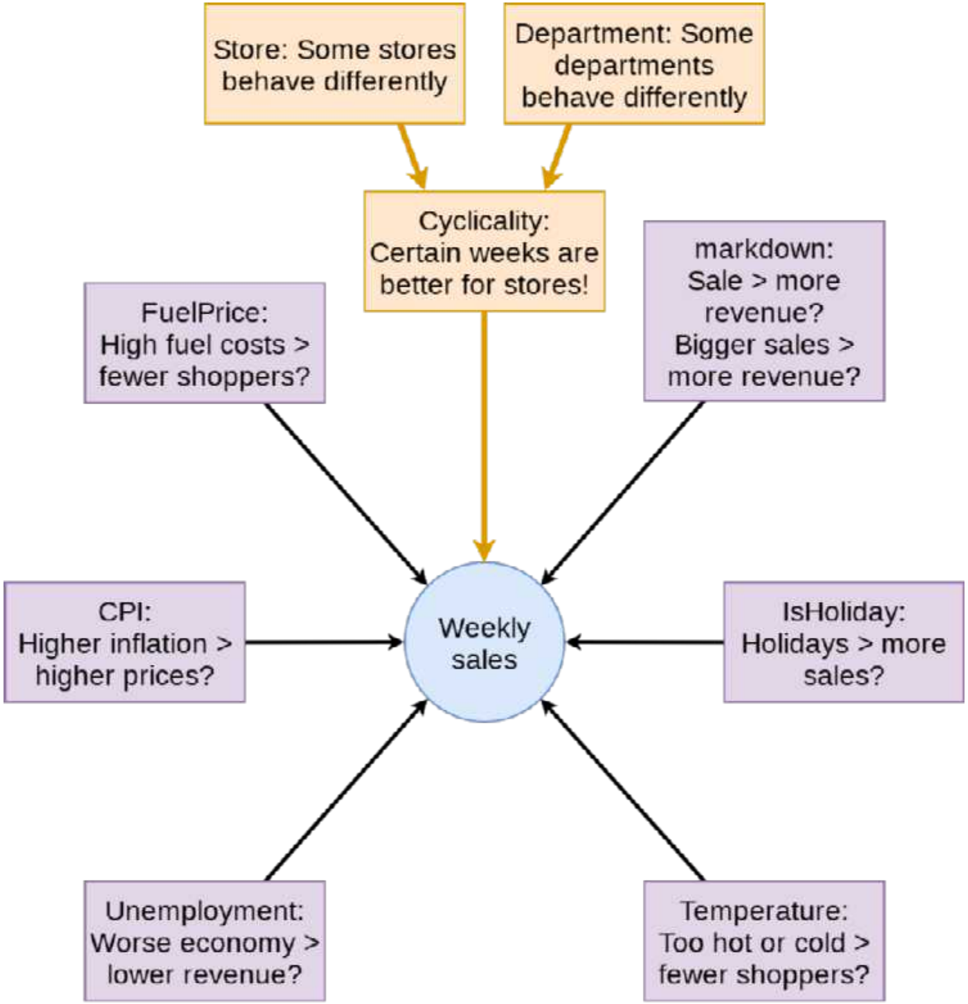


Visualizing in-sample WMAE by Dept

```
ggplot(data = df, aes(y = wmae_obs(actual = Weekly_Sales,
                                   predicted = WS_linear2,
                                   holidays = IsHoliday),
                      x = week, color = factor(Dept))) +
  geom_jitter(width = 0.25) + xlab("Week") + ylab("WMAE") +
  theme(legend.position = "none")
```



Back to the drawing board...



Third model: Including week x Store x Dept

```r
mod3 <- lm(Weekly_mult ~ factor(week):factor(Store):factor(Dept) +
             factor(IsHoliday) + factor(markdown>0) + markdown + Temperature +
             Fuel_Price + CPI + Unemployment, data = df)
## Error: cannot allocate vector of size 606.8Gb
```

- ...

Third model: Including week x Store x Dept

- Use `package:fixest`'s `feols()` -- it's really more efficient!

```
library(fixest)
mod3 <- feols(Weekly_mult ~ markdown + Temperature + Fuel_Price + CPI +
               Unemployment | swd, data = df) # now you know why create swd
tidy(mod3)
```

```
## # A tibble: 5 x 5
##   term             estimate   std.error statistic   p.value
##   <chr>               <dbl>       <dbl>     <dbl>     <dbl>
## 1 markdown     -0.00000122 0.000000220    -5.56  2.63e-  8
## 2 Temperature   0.00130    0.000163        7.95  1.90e- 15
## 3 Fuel_Price   -0.0532     0.00226       -23.5   2.20e-122
## 4 CPI           0.000190   0.000366        0.518 6.04e-  1
## 5 Unemployment -0.0291     0.00136       -21.3   8.12e-101
```

```
glance(mod3)
```

```
## # A tibble: 1 x 9
##   r.squared adj.r.squared within.r.squared pseudo.r.squared sigma   nobs     AIC
##       <dbl>         <dbl>            <dbl>            <dbl> <dbl>  <int>   <dbl>
## 1     0.708         0.526          0.00373               NA 0.379 421551 498237.
## # ... with 2 more variables: BIC <dbl>, logLik <dbl>
```

Prep submission and in-sample WMAE

```r
ıt of sample result
ot sure why there are NA prediction output although all predictors have no missing d
:est$Weekly_mult <- predict(mod3, df_test)
:est$Weekly_Sales <- df_test$Weekly_mult * df_test$store_avg

equired to submit a csv of Id and Weekly_Sales
:e.csv(df_test[ , c("Id", "Weekly_Sales")], "WMT_FE.csv",
      row.names = FALSE)

ack
:est$WS_FE <- ifelse(is.na(df_test$Weekly_Sales), 0, df_test$Weekly_Sales)

ıeck in sample WMAE
IS_FE <- predict(mod3, df) * df$store_avg
· wmae(actual = df$Weekly_Sales, predicted = df$WS_FE,
      holidays = df$IsHoliday)
es(w) <- "FE"
es <- c(wmaes, w)
es
```

```
##   Linear Linear 2       FE
## 3040.644 3208.144 1551.232
```

The general predict() function

- `predict()` is a generic function for predictions from the results of various model fitting functions.
- The function invokes particular methods which depend on the class of the first argument.
- For example, if the first argument is an object from the `lm()` model, `predict()` will call the `predict.lm()` function
- Typically model functions have been defined such as `predict.lm()` and `predict.glm()`
- `predcit.fixest()` is defined in the `fixest` package
- You may replace the `predict()` with `predict.fixest()` and get same results.
- Refer the manual here

Performance for FE model

Your most recent submission

Name	Submitted	Wait time	Execution time	Score
WMT_FE.csv	just now	1 seconds	1 seconds	3357.88481

Complete

Jump to your position on the leaderboard

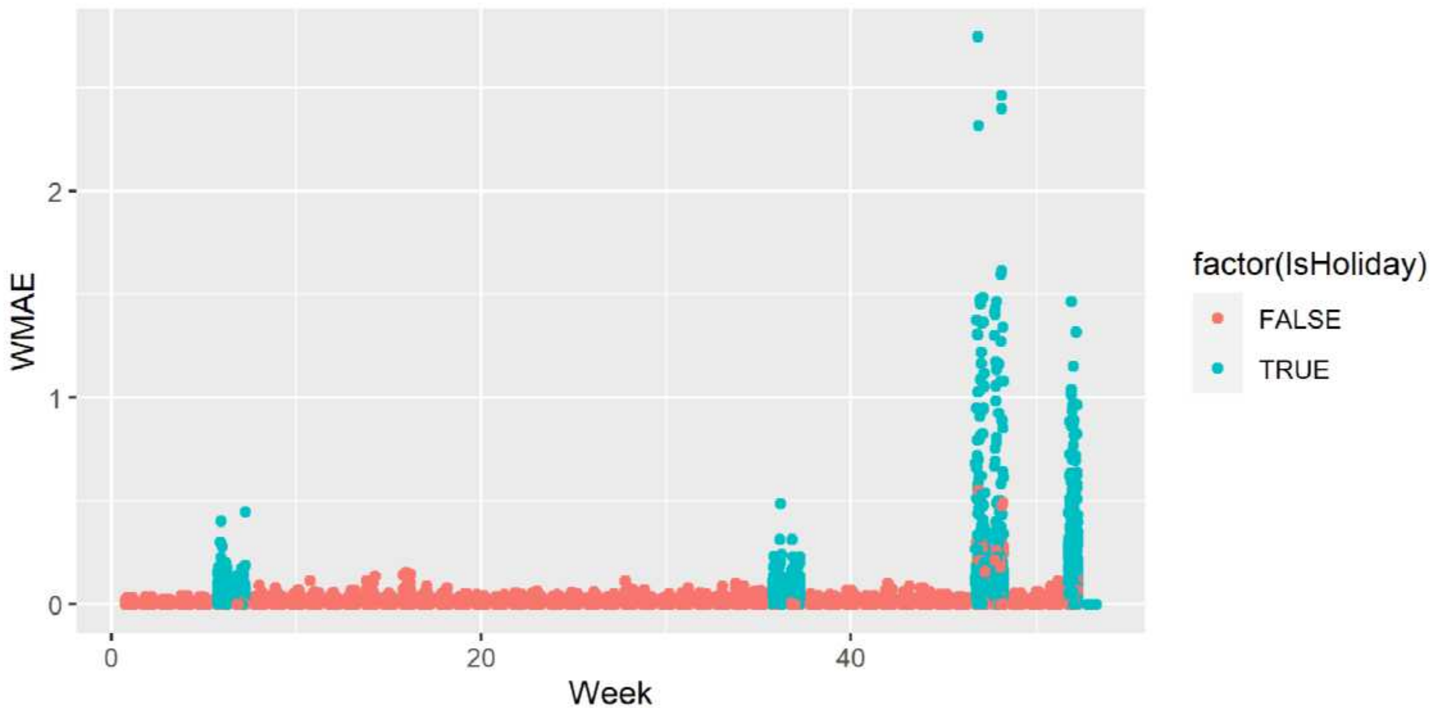
264	—	Sandeep	3349.90154	26	5y
265	▲ 13	Satya Prakash	3364.07150	23	5y
266	▲ 5	Prashant Kumar	3365.02867	8	5y
267	▼ 10	Gautam Gogoi	3370.85784	38	5y

```
wmaes_out
```

```
##    Linear Linear 2       FE
##    4954.4   5540.3   3357.9
```

Visualizing in-sample WMAE

```r
df$wmaes  <- wmae_obs(actual = df$Weekly_Sales, predicted = df$WS_FE,
                      holidays = df$IsHoliday)
ggplot(data=df, aes(y = wmaes,
                    x = week,
                    color = factor(IsHoliday))) +
  geom_jitter(width = 0.25) + xlab("Week") + ylab("WMAE")
```



Problems with the data

> Super Bowl: 12-Feb-10, 11-Feb-11, 10-Feb-12, 8-Feb-13 Labor Day: 10-Sep-10, 9-Sep-11, 7-Sep-12, 6-Sep-13 Thanksgiving: 26-Nov-10, 25-Nov-11, 23-Nov-12, 29-Nov-13 Christmas: 31-Dec-10, 30-Dec-11, 28-Dec-12, 27-Dec-13

1. The holidays are not always on the same week (the last indicates the week in the testing data)
   - The Super Bowl is in weeks 7, 7, 6 and *6*
   - Labor day isn't in our *testing data* at all!
   - Black Friday is in weeks 48, 47, and *47*
   - Christmas is in weeks 53, 52, and *52*
   - Manually adjust the data for these differences
2. Yearly growth -- we aren't capturing it, since we have such a small time span
   - We can manually adjust the data for this

> Code is in the code file -- a lot of `package:dplyr`

Performance overall

Your most recent submission

Name	Submitted	Wait time	Execution time	Score
WMT_FE_shift.csv	just now	1 seconds	1 seconds	3249.12698

Complete

Jump to your position on the leaderboard

240	▲ 9	RG50	3247.76071	13	5y
241	▲ 15	Will West	3248.16860	15	5y
242	▼ 2	Ugly Duckling	3264.66376	19	5y
243	▼ 2	Chiranjeev	3266.39474	3	5y

```
wmaes_out
```

```
##      Linear   Linear 2         FE Shifted FE
##      4954.4     5540.3     3357.9     3249.1
```

Performance overall

Your most recent submission

Name	Submitted	Wait time	Execution time	Score
WMT_naivemean.csv	just now	1 seconds	1 seconds	3167.99329

Complete

Jump to your position on the leaderboard

219	▲ 11	jong	3165.17441	20	4y
220	▲ 13	abhirup mallik	3168.04232	4	4y
221	▲ 2	KaggleBob	3170.86773	19	4y
222	▲ 2	pythonomic	3172.02059	13	4y

```
wmaes_out
```

```
##      Linear    Linear 2          FE Shifted FE Naive Mean
##     4954.40     5540.30     3357.90     3249.10     3167.99
```

Performance overall

Your most recent submission

Name	Submitted	Wait time	Execution time	Score
WMT_ens.csv	just now	1 seconds	1 seconds	3173.32504

Complete

Jump to your position on the leaderboard

220	▲ 2	KaggleBob	3170.86773	19	7y
221	▲ 2	pythonomic	3172.02059	13	7y
222	▼ 12	Vyassa Baratham	3172.93938	21	7y
223	▲ 8	Sriram Kovil	3191.36644	15	7y

```
wmaes_out
```

```
##      Linear   Linear 2         FE Shifted FE Naive Mean   Ensemble
##     4954.40    5540.30    3357.90    3249.10    3167.99    3173.30
```

This was a real problem!

- Walmart provided this data back in 2014 as part of a recruiting exercise
  - Details here
  - Discussion of first place entry
    - Code for first place entry
  - Discussion of second place entry
- This is what the group project will be like
  - Each group tackling a data problem which is hosted on Kaggle or Tianchi
  - You will have training data but testing data will be withheld
  - You will need to submit to Kaggle/Tianchi for model evaluation

Project deliverables

1. Submission to Kaggle/Tianchi
    - For model evaluation purpose
2. Submission to me: A .rmd (and .html + .pdf) file including:
    - The integrated code chunks
    - Main points and findings
    - Exploratory analysis of the data used
    - Your model development, implementation, evaluation, and refinement
    - A conclusion on how well your group did and what you learned
    - No zipped file please
3. A group presentation in the last session
    - A presentation slides (.rmd or .pptx) shall also be submitted
    - All members to present
4. If files > 50M, please submit through a shared folder using OneDrive or Google Drive. Keep all folder structure with all files and data, and make sure I can reproduce your code without any changes.

Ethics

- Kaggle 1st place winner cheated, $10,000 prize declared irrecoverable

Summary of Session 6

For next week

- Try to replicate the code
- You should have completed exploring your project data
- Continue your Datacamp career track
- Logistic regression for classification problems

R Coding Style Guide

Style is subjective and arbitrary but it is important to follow a generally accepted style if you want to share code with others. I suggest the The tidyverse style guide which is also adopted by Google with some modification

- Highlights of **the tidyverse style guide**:
  - *File names*: end with .R
  - *Identifiers*: variable_name, function_name, try not to use "." as it is reserved by Base R's S3 objects
  - *Line length*: 80 characters
  - *Indentation*: two spaces, no tabs (RStudio by default converts tabs to spaces and you may change under global options)
  - *Spacing*: x = 0, not x=0, no space before a comma, but always place one after a comma
  - *Curly braces {}*: first on same line, last on own line
  - *Assignment*: use `<-`, not = nor `->`
  - *Semicolon(;)*: don't use, I used once for the interest of space
  - *return()*: Use explicit returns in functions: default function return is the last evaluated expression
  - *File paths*: use relative file path "../../filename.csv" rather than absolute path "C:/mydata/filename.csv". Backslash needs \\

R packages used in this slide

This slide was prepared on 2021-09-07 from Session_6s_Kaggle.Rmd with R version 4.1.1 (2021-08-10) Kick Things on Windows 10 x64 build 18362 😄.

The attached packages used in this slide are:

```
##     fixest      broom  lubridate    forcats    stringr      dplyr      purrr
##    "0.9.0"    "0.7.9"   "1.7.10"    "0.5.1"    "1.4.0"    "1.0.7"    "0.3.4"
##      readr      tidyr     tibble    ggplot2  tidyverse kableExtra      knitr
##    "2.0.1"    "1.1.3"    "3.1.3"    "3.3.5"    "1.3.1"    "1.3.4"     "1.33"
```