ACCT 420: Advanced linear regression

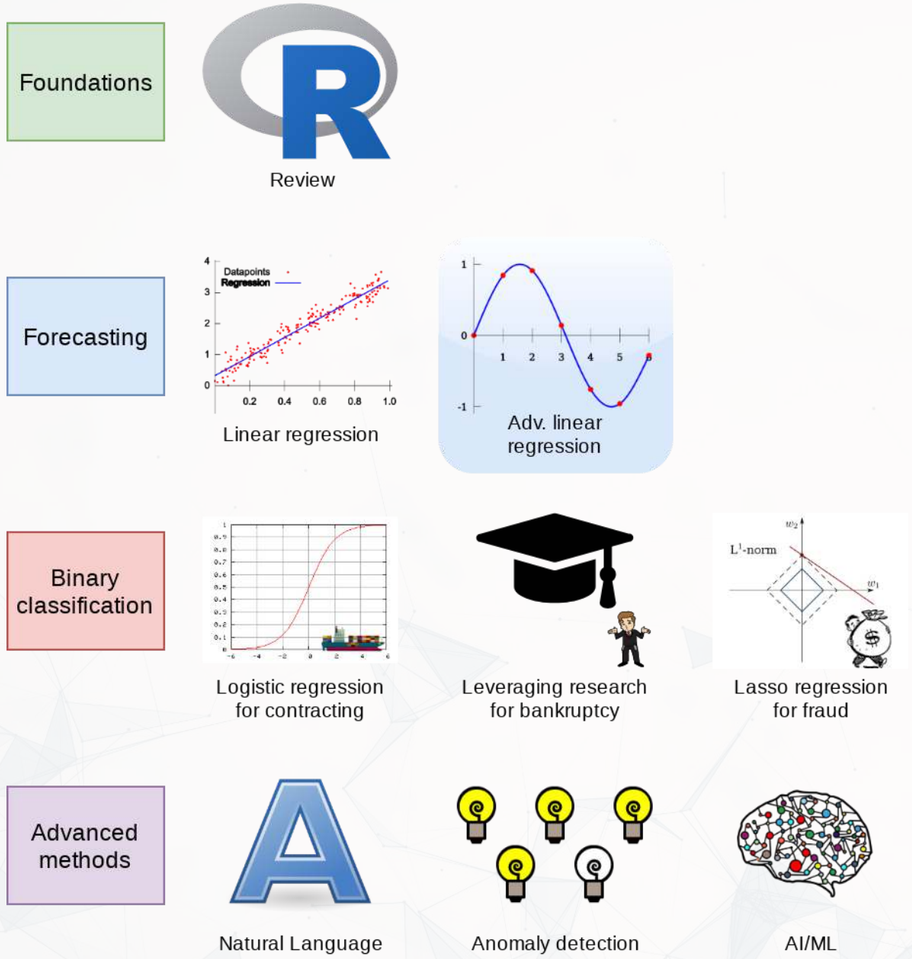
Dr. Richard M. Crowley

rcrowley@smu.edu.sg

https://rmc.link/

Front Matter

Learning objectives



- **Theory:**
  - Further understand stats treatments
    - Panel data
    - Fixed effects
    - Time (seasonality)
- **Application:**
  - Predicting revenue quarterly and weekly
- **Methodology:**
  - Univariate
  - Linear regression (OLS)
  - Visualization

Application: Quarterly retail revenue

The question

> How can we predict quarterly revenue for retail companies, leveraging our knowledge of such companies?

More specifically…

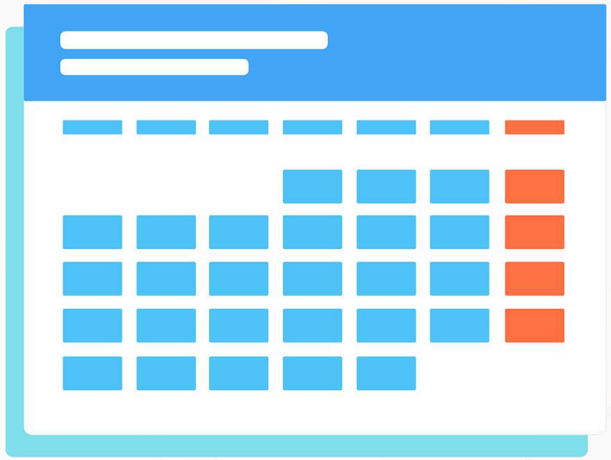
- Consider time dimensions
  - What matters:
    - Last quarter?
    - Last year?
    - Other time frames?
  - Cyclicality

Time and OLS

Time matters a lot for retail

How to capture time effects?

- Autoregression: our main focus today
  - Regress y_t on earlier value(s) of itself
    - Last quarter, last year, etc.
- Controlling for time directly in the model
  - We can do this using something called a *fixed effect*
    - This absorbs the *noise* in a model due to a particular grouping in the data



Noise in data

> Statistical noise is random error in the data

- Many sources of noise:
  - Factors we neglected to included in the model
  - Error in measurement
    - Accounting measurement!
  - Unexpected events / shocks
  - Groups within the data that behave differently

> Noise is OK, but the more we remove, the better!

Removing noise: Revisiting SG real estate revenue

- Different companies may behave slightly differently
  - Control for this using a *Fixed Effect* (ISIN uniquely identifies companies)

```r
forecast3.1 <-
  lm(revt_lead ~ revt + act + che + lct + dp + ebit + factor(isin),
     data=df_clean[df_clean$fic=="SGP",])
# n=15 to prevent outputting every fixed effect
library(broom)
print(tidy(forecast3.1), n=15)
```

```
# A tibble: 27 × 5
   term                      estimate std.error statistic  p.value
   <chr>                        <dbl>     <dbl>     <dbl>    <dbl>
 1 (Intercept)                -0.228    23.8      -0.00961 9.92e- 1
 2 revt                        0.585     0.0699    8.37    1.24e-15
 3 act                         0.220     0.0365    6.02    4.36e- 9
 4 che                        -0.0425    0.106    -0.399   6.90e- 1
 5 lct                         0.0771    0.0604    1.28    2.03e- 1
 6 dp                          0.622     0.885     0.703   4.83e- 1
 7 ebit                       -0.928     0.253    -3.66    2.85e- 4
 8 factor(isin)SG0581008505    2.12     34.6       0.0613  9.51e- 1
 9 factor(isin)SG1AE2000006   -1.69     41.1      -0.0410  9.67e- 1
10 factor(isin)SG1AG0000003   -3.58     33.9      -0.106   9.16e- 1
11 factor(isin)SG1BG1000000  -27.5      38.3      -0.716   4.74e- 1
12 factor(isin)SG1EE1000009   -5.62     36.5      -0.154   8.78e- 1
```

Removing noise: Singapore model

```
glance(forecast3.1)
```

```
# A tibble: 1 × 12
  r.squared adj.r.squared sigma statistic   p.value    df logLik   AIC   BIC
      <dbl>         <dbl> <dbl>     <dbl>     <dbl> <dbl>  <dbl> <dbl> <dbl>
1     0.894         0.887  121.      118. 9.14e-160    26 -2410. 4875. 4986.
# ℹ 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

```
anova(forecast3, forecast3.1, test="Chisq")
```

```
Analysis of Variance Table

Model 1: revt_lead ~ revt + act + che + lct + dp + ebit
Model 2: revt_lead ~ revt + act + che + lct + dp + ebit + factor(isin)
  Res.Df     RSS Df Sum of Sq Pr(>Chi)
1    383 5806559
2    363 5311530 20    495029  0.02729 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

> This only a bit better. Why? There was another source of noise within Singapore real estate companies

Another way to do fixed effects

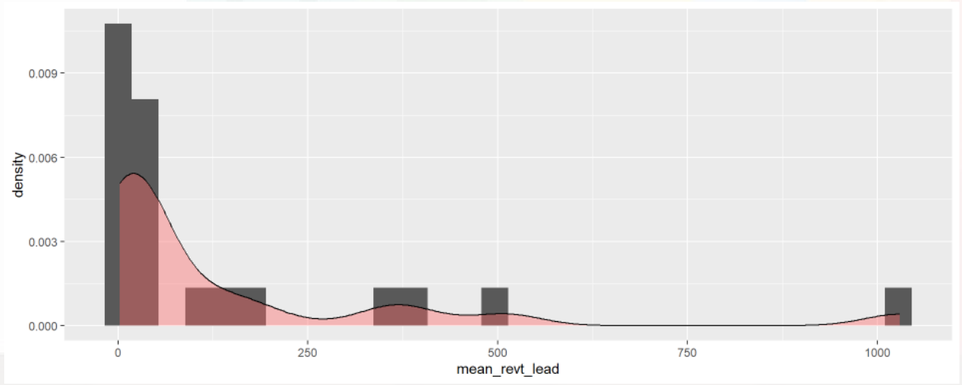
- The library fixest has `feols()`: **f**ixed **e**ffects **OLS**
  - Better for complex models
  - Extremely efficient computationally

```r
library(fixest)
forecast3.2 <-
  feols(revt_lead ~ revt + act + che + lct + dp + ebit | isin,
        data=df_clean[df_clean$fic=="SGP",])
summary(forecast3.2)
```

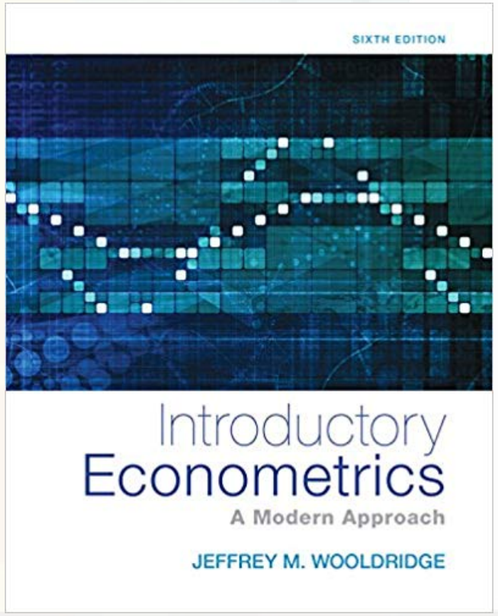
```
OLS estimation, Dep. Var.: revt_lead
Observations: 390
Fixed-effects: isin: 21
Standard-errors: Clustered (isin)
      Estimate Std. Error   t value  Pr(>|t|)
revt  0.584999   0.265386  2.204336 0.0393790 *
act   0.219649   0.057738  3.804256 0.0011114 **
che  -0.042455   0.193770 -0.219103 0.8287907
lct   0.077060   0.151119  0.509926 0.6156849
dp    0.621635   1.014534  0.612729 0.5469600
ebit -0.927741   0.474333 -1.955886 0.0645925 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
RMSE: 116.7     Adj. R2: 0.88664
              Within R2: 0.747629
```

Why exactly would we use fixed effects?

- Fixed effects are used when the average of $\hat{y}$ varies by some group in our data
  - In our problem, the average revenue of each firm is different
- Fixed effects absorb this difference



- Further reading:
  - Introductory Econometrics by Jeffrey M. Wooldridge



Data preparation

The data

- From quarterly reports
- Two sets of firms:
  - US “Hypermarkets & Super Centers” (GICS: 30101040)
  - US “Multiline Retail” (GICS: 255030)
- Data from Compustat - Capital IQ > North America - Daily > Fundamentals Quarterly

Formalization

1. Question
   - How can we predict quarterly revenue for large retail companies?
2. Hypothesis (just the alternative ones)
   1. Current quarter revenue helps predict next quarter revenue
   2. 3 quarters ago revenue helps predict next quarter revenue (Year-over-year)
   3. Different quarters exhibit different patterns (seasonality)
   4. A long-run autoregressive model helps predict next quarter revenue
3. Prediction
   - Use OLS for all the above, t-tests for coefficients
   - Hold out sample: 2018-2023

Variable generation

```r
library(tidyverse)  # As always
library(plotly)  # interactive graphs
library(lubridate)  # import some sensible date functions

# Generate quarter over quarter growth "revtq_gr"
df <- df %>% group_by(gvkey) %>% mutate(revtq_gr=revtq / lag(revtq) - 1) %>% ungroup()

# Generate year-over-year growth "revtq_yoy"
df <- df %>% group_by(gvkey) %>% mutate(revtq_yoy=revtq / lag(revtq, 4) - 1) %>% ungroup()

# Generate first difference "revtq_d"
df <- df %>% group_by(gvkey) %>% mutate(revtq_d=revtq - lag(revtq)) %>% ungroup()

# Generate a proper date
# Date was YYYY-MM-DD, can be converted from text to date easily
df$date <- ymd(df$datadate)  # From lubridate
df$qtr <- quarter(df$date)   # From lubridate
```

- Use mutate for variables using lags
- `ymd()` from lubridate is handy to convert any date listing year, then month, then day.
  - It also has `ydm()`, `mdy()`, `myd()`, `dmy()` and `dym()`
  - It can handle quarters, times, and date-times as well
  - Cheat sheet

Base R Date manipulation

- `as.Date()` can take a date formatted as “YYYY/MM/DD” and convert to a proper date value
  - You can convert other date types using the `format=` argument
    - i.e., “DD.MM.YYYY” is format code “%d.%m.%Y”
    - Full list of date codes

Example output

conm	date	revtq	revtq_gr	revtq_yoy	revtq_d
ALLIED STORES	1962-04-30	156.5	NA	NA	NA
ALLIED STORES	1962-07-31	161.9	0.0345048	NA	5.4
ALLIED STORES	1962-10-31	176.9	0.0926498	NA	15.0
ALLIED STORES	1963-01-31	275.5	0.5573770	NA	98.6
ALLIED STORES	1963-04-30	171.1	-0.3789474	0.0932907	-104.4
ALLIED STORES	1963-07-31	182.2	0.0648743	0.1253860	11.1

```
# A tibble: 6 × 3
  conm          date       datadate
  <chr>         <date>     <chr>
1 ALLIED STORES 1962-04-30 1962-04-30
2 ALLIED STORES 1962-07-31 1962-07-31
3 ALLIED STORES 1962-10-31 1962-10-31
4 ALLIED STORES 1963-01-31 1963-01-31
5 ALLIED STORES 1963-04-30 1963-04-30
6 ALLIED STORES 1963-07-31 1963-07-31
```

Creating 8 quarters (2 years) of lags

- Eight is easy to do by copying and pasting code, but…
  - What if we wanted 100 lags?
    - That would be a lot of copying
      - The code for this is in the appendix
- Three solutions, though none is straightforward
  1. Use sapply and a function to reassemble output into a data frame
     - We will ignore this – option 2 is always better
  2. Use purrr to loop into a data frame
  3. Use advanced dplyr programming with quosures
- To compare the methods, we'll make a copy of the data frame

```
# Make a copy of our data frame to compare later
df2 <- df
```

Method 2: Map with Purrr

```r
# Approach #2: Mixing purrr with dplyr across()
library(purrr)

multi_lag_map <- function(df, lags, var, postfix="") {
  new_columns <- map(lags,
    function(x) df %>%
      group_by(gvkey) %>%
      transmute(across(all_of(var),
                       .fns = list(~ lag(., x)),
                       .names = paste0('{col}', postfix, x) ) ) %>%
      ungroup() %>%
      select(-gvkey)
      )
  cbind(df, list_cbind(new_columns))
}
```

- `map()` is the key function. It takes a vector and iterates over it applying a specified function, returning a list of data frames
- `transmute()`: like mutate except it extracts the column it makes automatically
- `cbind_list()`: takes a list of data frames and combines them into 1 data frame
- `cbind()` is a base R function to combine

Method 3: Advanced programming

```r
# Approach #3: Advanced programming using quosures...
library(rlang)
multi_lag <- function(df, lags, var, postfix="") {
  var <- enquo(var)
  quosures <- map(lags, ~quo(lag(!!var, !!.x))) %>%
    set_names(paste0(quo_text(var), postfix, lags))
  return(ungroup(mutate(group_by(df, gvkey), !!!quosures)))
}
```

- `setNames()`: allows for storing a value and name simultaneously

⚠ Quosures are beyond the scope of the course. However, they are very useful if:

1. You need to do non-standard calculations at scale
2. You are programming a library.

Comparing the methods

2. Using purrr + dplyr

```r
df <- multi_lag_map(df, 1:8, 'revtq', "_l")   # Generate lags "revtq_l#"
df <- multi_lag_map(df, 1:8, 'revtq_gr')      # Generate changes "revtq_gr#"
df <- multi_lag_map(df, 1:8, 'revtq_yoy')     # Generate year-over-year changes "revtq_yoy#"
df <- multi_lag_map(df, 1:8, 'revtq_d')       # Generate first differences "revtq_d#"
```

3. Using advanced programming techniques

```r
df2 <- multi_lag(df2, 1:8, revtq, "_l")   # Generate lags "revtq_l#"
df2 <- multi_lag(df2, 1:8, revtq_gr)      # Generate changes "revtq_gr#"
df2 <- multi_lag(df2, 1:8, revtq_yoy)     # Generate year-over-year changes "revtq_yoy#"
df2 <- multi_lag(df2, 1:8, revtq_d)       # Generate first differences "revtq_d#"
```

- Verify that the output is the same

```r
all(df==df2, na.rm=T)
```

```
[1] TRUE
```

Example output

conm	date	revtq	revtq_l1	revtq_l2	revtq_l3	revtq_l4
ALLIED STORES	1962-04-30	156.5	NA	NA	NA	NA
ALLIED STORES	1962-07-31	161.9	156.5	NA	NA	NA
ALLIED STORES	1962-10-31	176.9	161.9	156.5	NA	NA
ALLIED STORES	1963-01-31	275.5	176.9	161.9	156.5	NA
ALLIED STORES	1963-04-30	171.1	275.5	176.9	161.9	156.5
ALLIED STORES	1963-07-31	182.2	171.1	275.5	176.9	161.9

Clean and split into training and testing

```r
# Clean the data: Replace NaN, Inf, and -Inf with NA
df <- df %>%
  mutate(across(where(is.numeric), ~replace(., !is.finite(.), NA)))

# Split into training and testing data
# Training data: We'll use data released before 2018
train <- filter(df, year(date) < 2018)

# Testing data: We'll use data released 2018 through 2023
test <- filter(df, year(date) >= 2018)
```

- Same cleaning function as last week:
  - Replaces all `NaN`, `Inf`, and `-Inf` with `NA`
- `year()` comes from lubridate

Univariate stats

Univariate stats

- To get a better grasp on the problem, looking at univariate stats can help
  - Summary stats (using `summary()`)
  - Correlations using `cor()`
  - Plots using your preferred package such as ggplot2

```
summary(df[,c("revtq","revtq_gr","revtq_yoy", "revtq_d","qtr")])
```

```
     revtq              revtq_gr          revtq_yoy           revtq_d
Min.   :     -0.26   Min.   :-1.2452   Min.   :-24.0000   Min.   :-24325.206
1st Qu.:     68.43   1st Qu.:-0.1043   1st Qu.:  0.0038   1st Qu.:   -20.256
Median :    317.78   Median : 0.0494   Median :  0.0759   Median :     4.723
Mean   :   2942.73   Mean   : 0.0802   Mean   :  0.2677   Mean   :    44.860
3rd Qu.:   1525.00   3rd Qu.: 0.2051   3rd Qu.:  0.1677   3rd Qu.:    61.913
Max.   :164048.00   Max.   :38.0000   Max.   :468.3333   Max.   : 29410.000
NA's   :613         NA's   :1090      NA's   :1381       NA's   :1055
      qtr
Min.   :1.000
1st Qu.:2.000
Median :2.000
Mean   :2.498
3rd Qu.:3.000
Max.   :4.000
```

ggplot2 for visualization

- The next slides will use some custom functions using ggplot2
- ggplot2 has an odd syntax:
  - It doesn't use pipes (%>%), but instead adds everything together (+)

```r
library(ggplot2)  # or tidyverse -- it's part of tidyverse
df %>%
  ggplot(aes(y=var_for_y_axis, x=var_for_x_axis)) +
  geom_point()  # scatterplot
```

- `aes()` is for aesthetics – how the chart is set up
- Other useful aesthetics:
  - `group=` to set groups to list in the legend. Not needed if using the below though
  - `color=` to set color by some grouping variable. Put `factor()` around the variable if you want discrete groups, otherwise it will do a color scale (light to dark)
  - `shape=` to set shapes for points – see here for a list

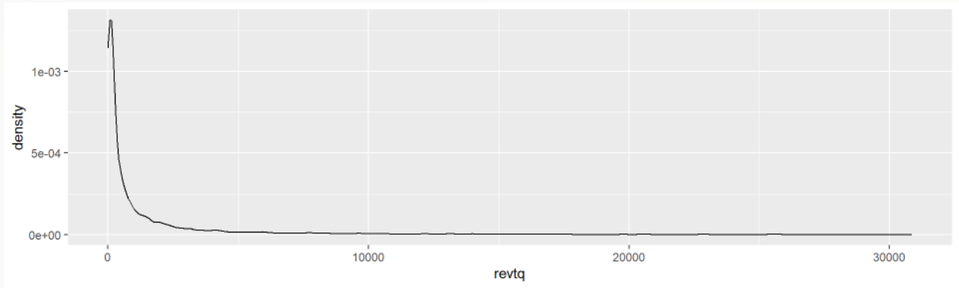
ggplot2 for visualization

```r
library(ggplot2)  # or tidyverse -- it's part of tidyverse
df %>%
  ggplot(aes(y=var_for_y_axis, x=var_for_x_axis)) +
  geom_point()  # scatterplot
```

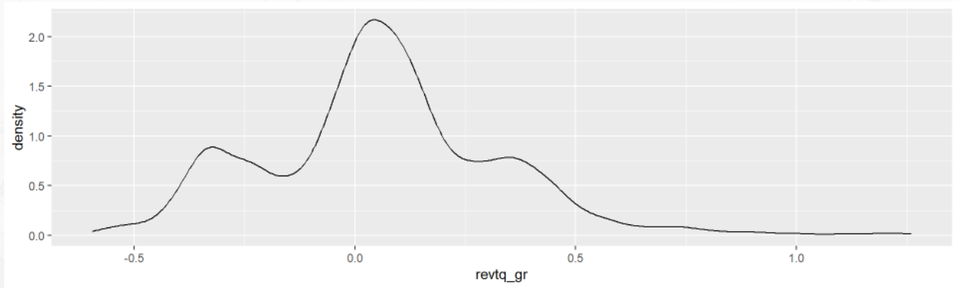
- `geom` stands for geometry – any shapes, lines, etc. start with `geom`
- Other useful geoms:
  - `geom_line()`: makes a line chart
  - `geom_bar()`: makes a bar chart – y is the height, x is the category
  - `geom_smooth(method="lm")`: Adds a linear regression into the chart
  - `geom_abline(slope=1)`: Adds a 45° line
- Add `xlab("Label text here")` to change the x-axis label
- Add `ylab("Label text here")` to change the y-axis label
- Add `ggtitle("Title text here")` to add a title
- Plenty more details in the Data Visualization Cheat Sheet

Plotting: Distribution of revenue

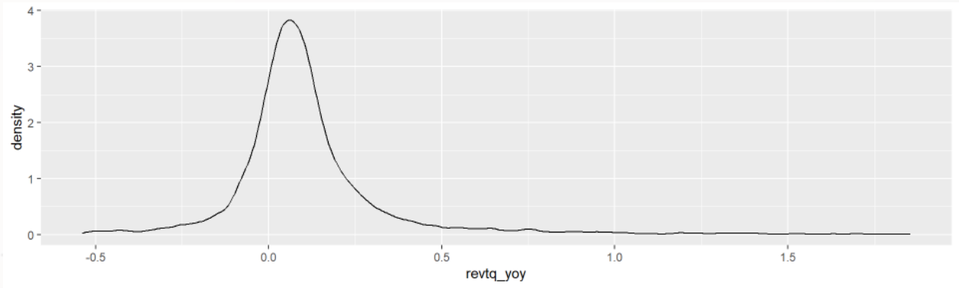
1. Revenue



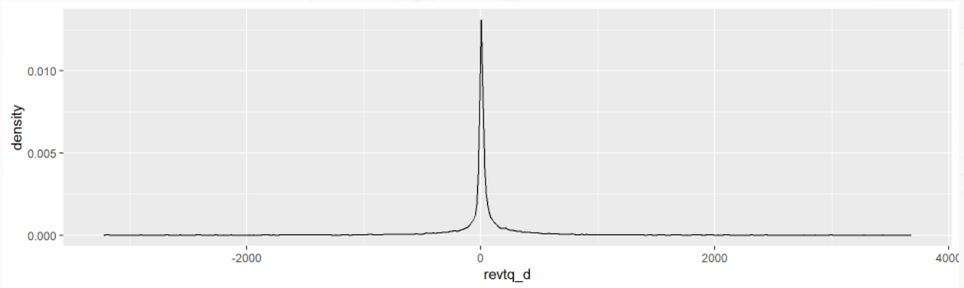
2. Quarterly growth



3. Year-over-year growth



4. First difference

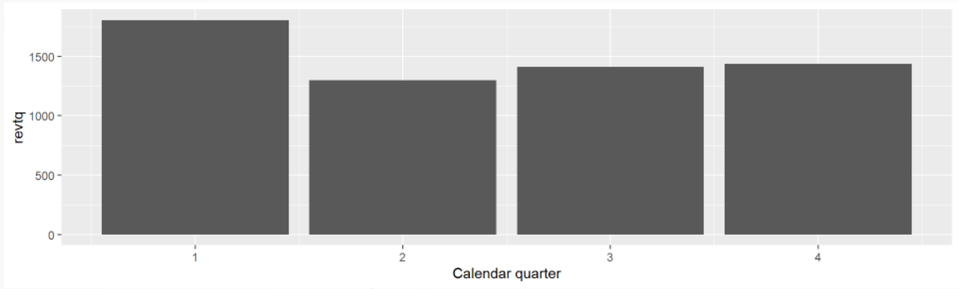


What do we learn from these graphs?

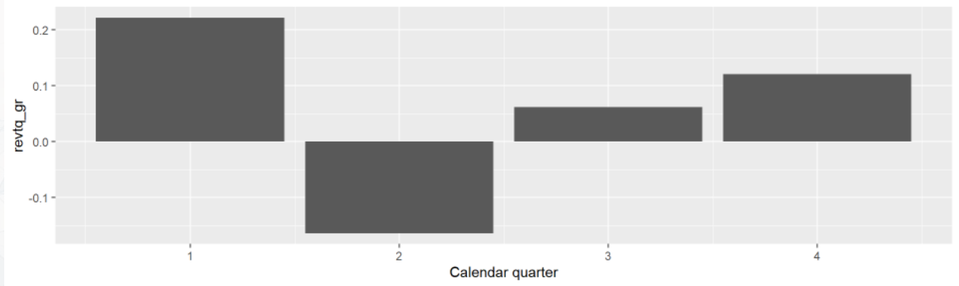
1. Revenue
    -
    -
2. Quarterly growth
    -
    -
3. Year-over-year growth
    -
    -
4. First difference
    -
    -

Plotting: Mean revenue (y) by quarter (x)

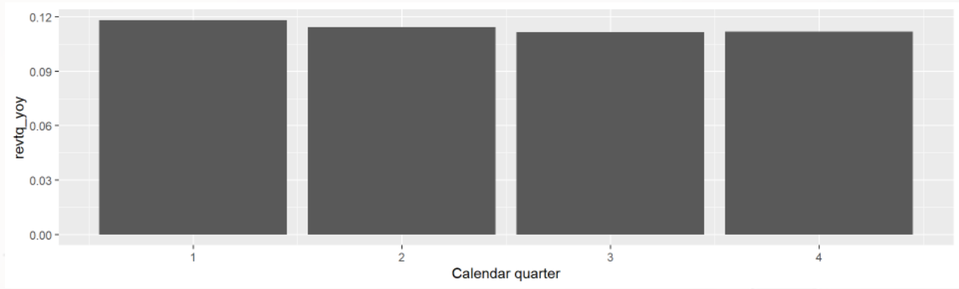
1. Revenue



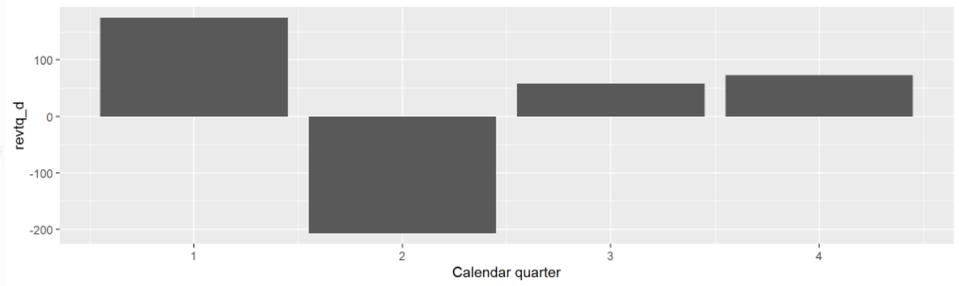
2. Quarterly growth



3. Year-over-year growth



4. First difference

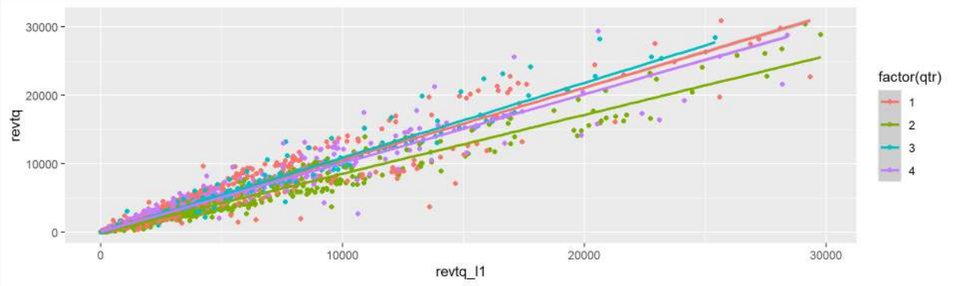


What do we learn from these graphs?

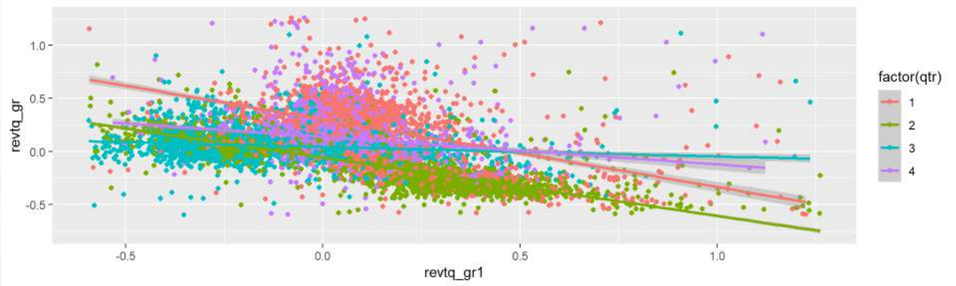
1. Revenue
    -
    -
2. Quarterly growth
    -
    -
3. Year-over-year growth
    -
    -
4. First difference
    -
    -

Plotting: Revenue (y) vs lag (x) by quarter

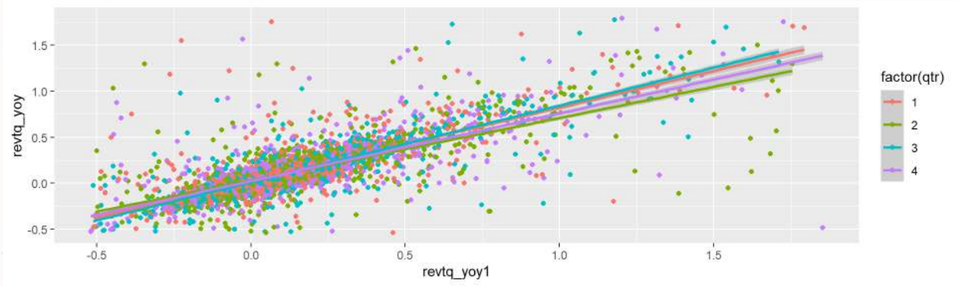
1. Revenue



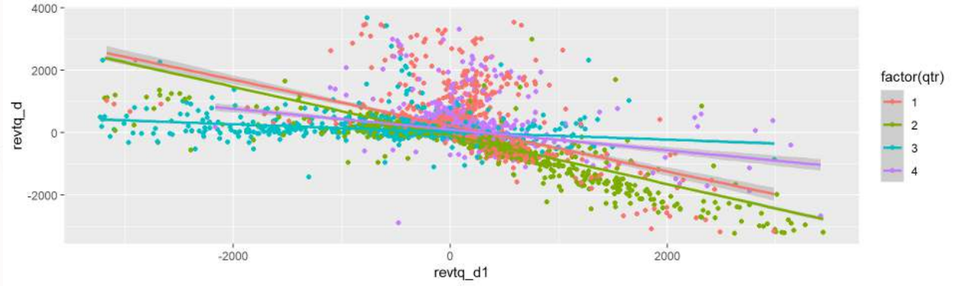
2. Quarterly growth



3. Year-over-year growth



4. First difference



What do we learn from these graphs?

1. Revenue
   - Revenue is really linear! But each quarter has a distinct linear relation.
2. Quarterly growth
   - All over the place. Each quarter appears to have a different pattern though. Quarters will matter.
3. Year-over-year growth
   - Linear but noisy.
4. First difference
   - Again, all over the place. Each quarter appears to have a different pattern though. Quarters will matter.

Correlation matrices

```
cor(train[,c("revtq","revtq_l1","revtq_l2","revtq_l3", "revtq_l4")],
    use="complete.obs")
```

```
             revtq  revtq_l1  revtq_l2  revtq_l3  revtq_l4
revtq    1.0000000 0.9916945 0.9934665 0.9904917 0.9970908
revtq_l1 0.9916945 1.0000000 0.9917766 0.9936673 0.9902467
revtq_l2 0.9934665 0.9917766 1.0000000 0.9917388 0.9931696
revtq_l3 0.9904917 0.9936673 0.9917388 1.0000000 0.9911274
revtq_l4 0.9970908 0.9902467 0.9931696 0.9911274 1.0000000
```

```
cor(train[,c("revtq_gr","revtq_gr1","revtq_gr2","revtq_gr3", "revtq_gr4")],
    use="complete.obs")
```

```
            revtq_gr   revtq_gr1   revtq_gr2   revtq_gr3   revtq_gr4
revtq_gr   1.0000000 -0.30650646  0.13222914 -0.13376729  0.39769716
revtq_gr1 -0.3065065  1.00000000 -0.29829311  0.06717413 -0.12452303
revtq_gr2  0.1322291 -0.29829311  1.00000000 -0.22696363  0.04106885
revtq_gr3 -0.1337673  0.06717413 -0.22696363  1.00000000 -0.03141264
revtq_gr4  0.3976972 -0.12452303  0.04106885 -0.03141264  1.00000000
```

> Retail revenue has **high** autocorrelation! There is a concern for multicolinearity. Revenue growth is less autocorrelated and oscillates.

Correlation matrices

```
cor(train[,c("revtq_yoy","revtq_yoy1","revtq_yoy2","revtq_yoy3", "revtq_yoy4")],
    use="complete.obs")
```

```
            revtq_yoy revtq_yoy1 revtq_yoy2 revtq_yoy3  revtq_yoy4
revtq_yoy  1.000000000  0.7519807 0.02477479 0.01730051 0.008416565
revtq_yoy1 0.751980703  1.0000000 0.51028046 0.41010043 0.152340656
revtq_yoy2 0.024774792  0.5102805 1.00000000 0.79026409 0.318107988
revtq_yoy3 0.017300513  0.4101004 0.79026409 1.00000000 0.605670844
revtq_yoy4 0.008416565  0.1523407 0.31810799 0.60567084 1.000000000
```

```
cor(train[,c("revtq_d","revtq_d1","revtq_d2","revtq_d3", "revtq_d4")],
    use="complete.obs")
```

```
            revtq_d   revtq_d1   revtq_d2   revtq_d3   revtq_d4
revtq_d   1.0000000 -0.5894501  0.2995031 -0.5900719  0.9280353
revtq_d1 -0.5894501  1.0000000 -0.6027336  0.3021009 -0.5729021
revtq_d2  0.2995031 -0.6027336  1.0000000 -0.6083765  0.2936155
revtq_d3 -0.5900719  0.3021009 -0.6083765  1.0000000 -0.5840817
revtq_d4  0.9280353 -0.5729021  0.2936155 -0.5840817  1.0000000
```

> Year over year change fixes the multicollinearity issue. First difference oscillates like quarter over quarter growth.

R Practice

- This practice will look at predicting Walmart's quarterly revenue using:
  - 1 lag
  - Cyclicality
- Practice using:
  - `lm()`
  - ggplot2
- Do the exercises in today's practice file
  - R Practice
  - Short link: rmc.link/420r3

> In this practice, we'll work through an easier variant of today's problem

Forecasting

The models

1. 1 Quarter lag: We saw a very strong linear pattern here earlier (*AR(1) model*)

```r
mod1 <- lm(revtq ~ revtq_l1, data=train)
```

2. Quarter and year lag: Year-over-year seemed pretty constant

```r
mod2 <- lm(revtq ~ revtq_l1 + revtq_l4, data=train)
```

3. 2 years of lags: Other lags could also help us predict (*AR(8) model*)

```r
mod3 <- lm(revtq ~ revtq_l1 + revtq_l2 + revtq_l3 + revtq_l4 + revtq_l5 +
             revtq_l6 + revtq_l7 + revtq_l8, data=train)
```

Quarter lag

```
summary(mod1)
```

```
Call:
lm(formula = revtq ~ revtq_l1, data = train)

Residuals:
     Min       1Q   Median       3Q      Max 
-24450.0    -41.5    -18.7     30.4  16547.7 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) 22.956320  13.212844   1.737   0.0823 .  
revtq_l1     1.003163   0.001412 710.620   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1224 on 9070 degrees of freedom
```

Quarter and year lag

```
summary(mod2)
```

```
Call:
lm(formula = revtq ~ revtq_l1 + revtq_l4, data = train)

Residuals:
     Min       1Q   Median       3Q      Max 
-20059.2    -39.8    -25.8      2.8  14911.4 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) 27.783189   7.387569   3.761  0.00017 ***
revtq_l1     0.231282   0.005493  42.102  < 2e-16 ***
revtq_l4     0.809231   0.005693 142.147  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

2 years of lags

```
summary(mod3)
```

```
Call:
lm(formula = revtq ~ revtq_l1 + revtq_l2 + revtq_l3 + revtq_l4 +
    revtq_l5 + revtq_l6 + revtq_l7 + revtq_l8, data = train)

Residuals:
    Min      1Q  Median      3Q     Max
-5251.2   -17.2    -7.3     6.3  6589.8

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  7.54050    4.09176   1.843 0.065389 .
revtq_l1     0.87920    0.01176  74.774  < 2e-16 ***
revtq_l2     0.07123    0.01558   4.572 4.91e-06 ***
revtq_l3    -0.03127    0.01507  -2.075 0.037990 *
revtq_l4     0.08224    0.01141   7.205 < 2e-16 ***
```

Testing out of sample

- RMSE: Root mean square Error
- RMSE is very affected by outliers, and a bad choice for noisy data where you are OK with missing a few outliers here and there
  - Doubling error *quadruples* the penalty

```r
rmse <- function(v1, v2) {
  sqrt(mean((v1 - v2)^2, na.rm=T))
}
```

- MAE: Mean absolute error
- MAE is measures average accuracy with no weighting
  - Doubling error *doubles* the penalty

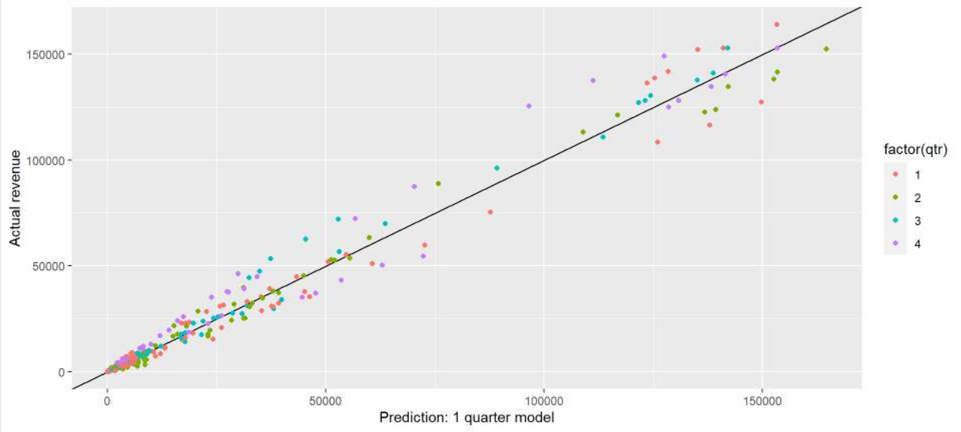
```r
mae <- function(v1, v2) {
  mean(abs(v1-v2), na.rm=T)
}
```

> Both are commonly used for evaluating OLS out of sample

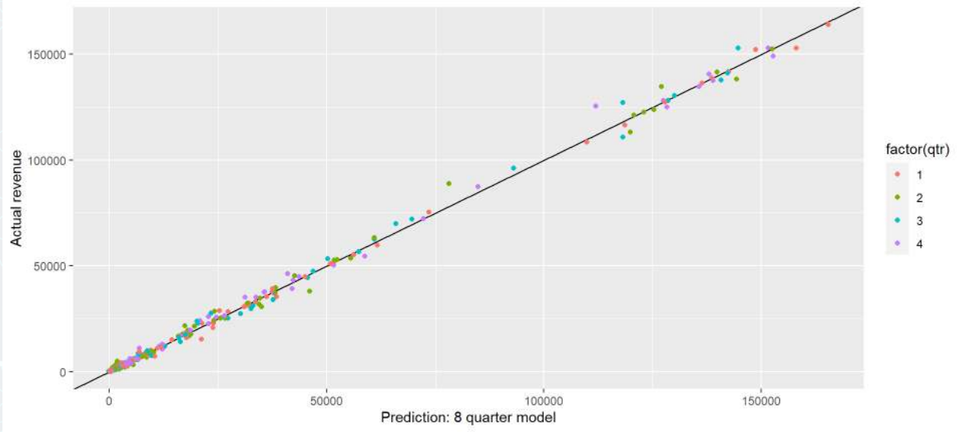
Testing out of sample: Lags of revenue

	adj_r_sq	rmse_in	mae_in	rmse_out	mae_out
1 quarter	0.9823367	1223.8606	345.02693	3463.412	1240.2024
1 and 4 quarters	0.9950640	664.4809	185.46272	2408.481	823.1251
8 quarter	0.9987348	350.6589	96.40259	1273.640	486.7373

1 quarter model



8 quarter model

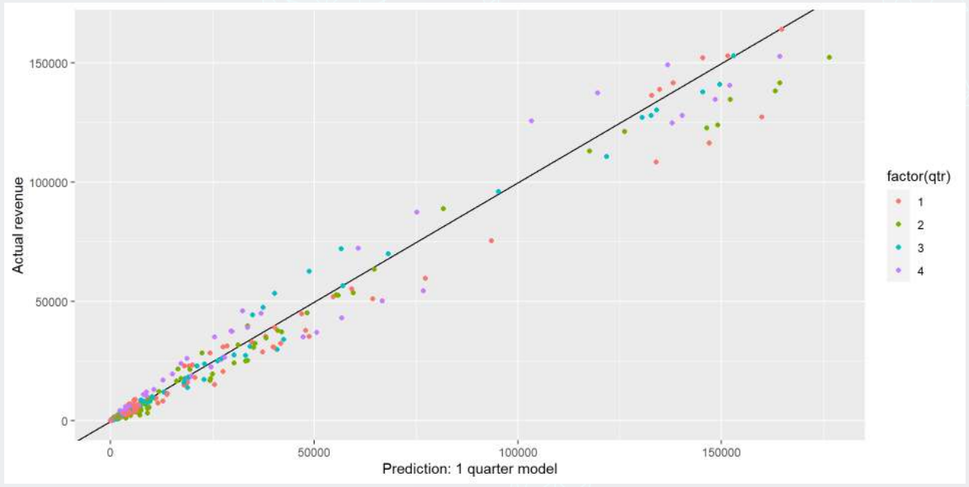


What about for revenue growth?

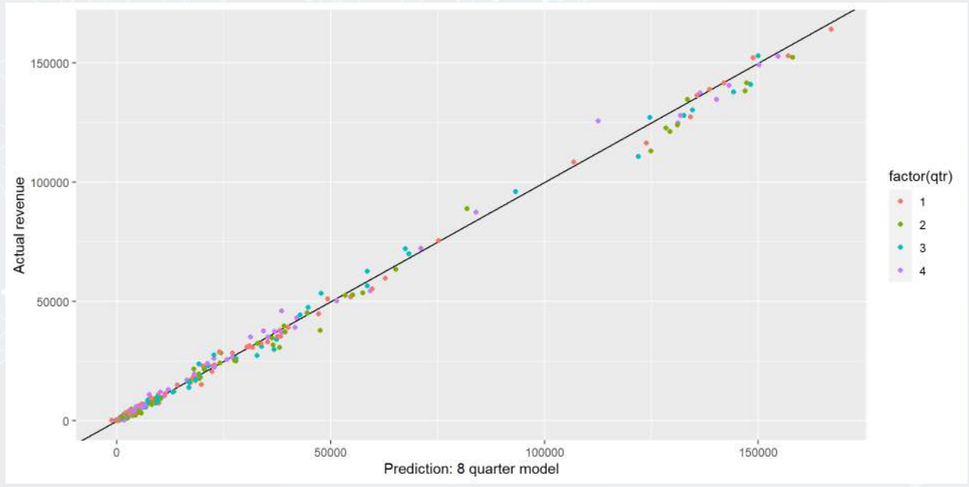
$$revt_t = (1 + growth_t) \times revt_{t-1}$$

	adj_r_sq	rmse_in	mae_in	rmse_out	mae_out
1 quarter	0.0014511	1351.3353	357.1677	3873.697	1371.667
1 and 4 quarters	0.0963806	965.4291	266.4816	2781.420	1036.653
8 quarter	0.6209886	493.8014	143.0223	1585.755	639.204

1 quarter model



8 quarter model

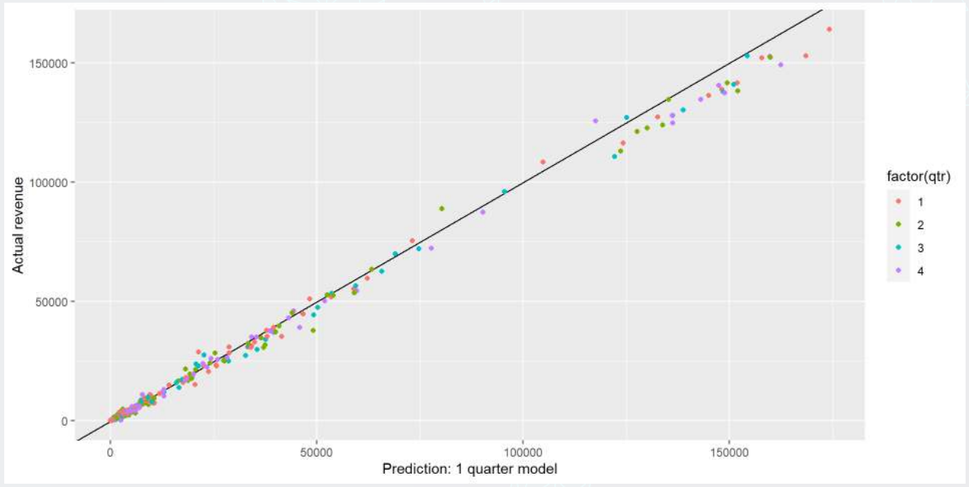


What about for YoY revenue growth?

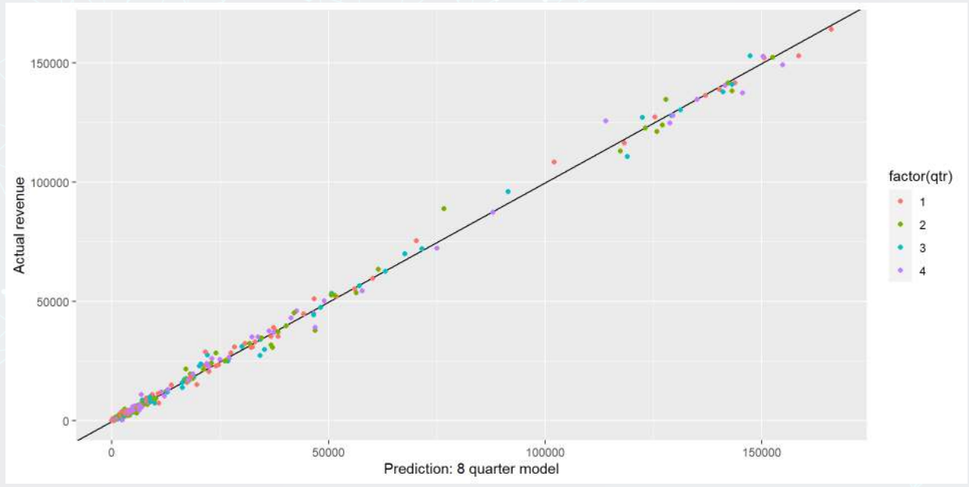
$$revt_t = (1 + yoy\_growth_t) \times revt_{t-4}$$

	adj_r_sq	rmse_in	mae_in	rmse_out	mae_out
1 quarter	0.0819290	705.1382	169.8824	2029.457	764.2961
1 and 4 quarters	0.3181392	2468.5552	609.4654	8200.987	3060.8517
8 quarter	0.5870339	373.8323	101.0739	1424.345	569.5131

1 quarter model



8 quarter model

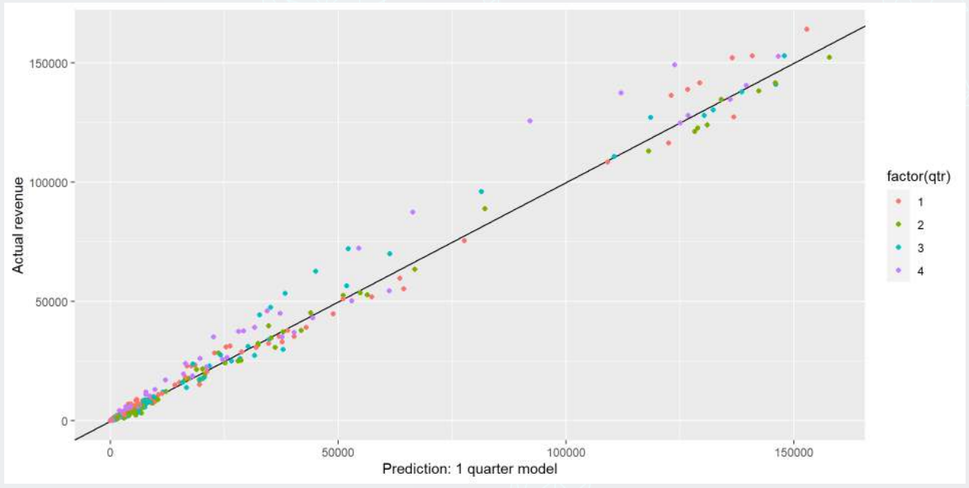


What about for first difference?

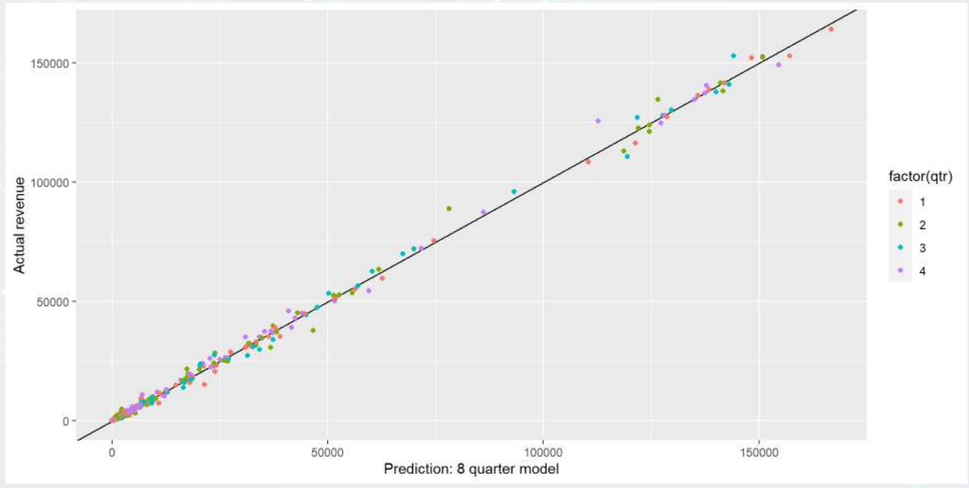
$$revt_t = change_t + revt_{t-1}$$

	adj_r_sq	rmse_in	mae_in	rmse_out	mae_out
1 quarter	0.3258854	982.1141	311.6347	2928.851	997.6321
1 and 4 quarters	0.8661368	450.5154	112.2122	1284.958	492.3839
8 quarter	0.9309542	337.1684	97.3112	1273.836	488.1175

1 quarter model



8 quarter model



Takeaways

1. The first difference model works a bit better than the revenue model at predicting next quarter revenue
   - It also doesn’t suffer (as much) from multicollinearity
     - This is why time series analysis is often done on first differences
       - Or second differences (difference in differences)
2. The other models perform pretty well as well
3. Extra lags generally seems helpful when accounting for cyclicality
4. Regressing by quarter helps a bit, particularly with revenue growth

Case: Advanced revenue prediction

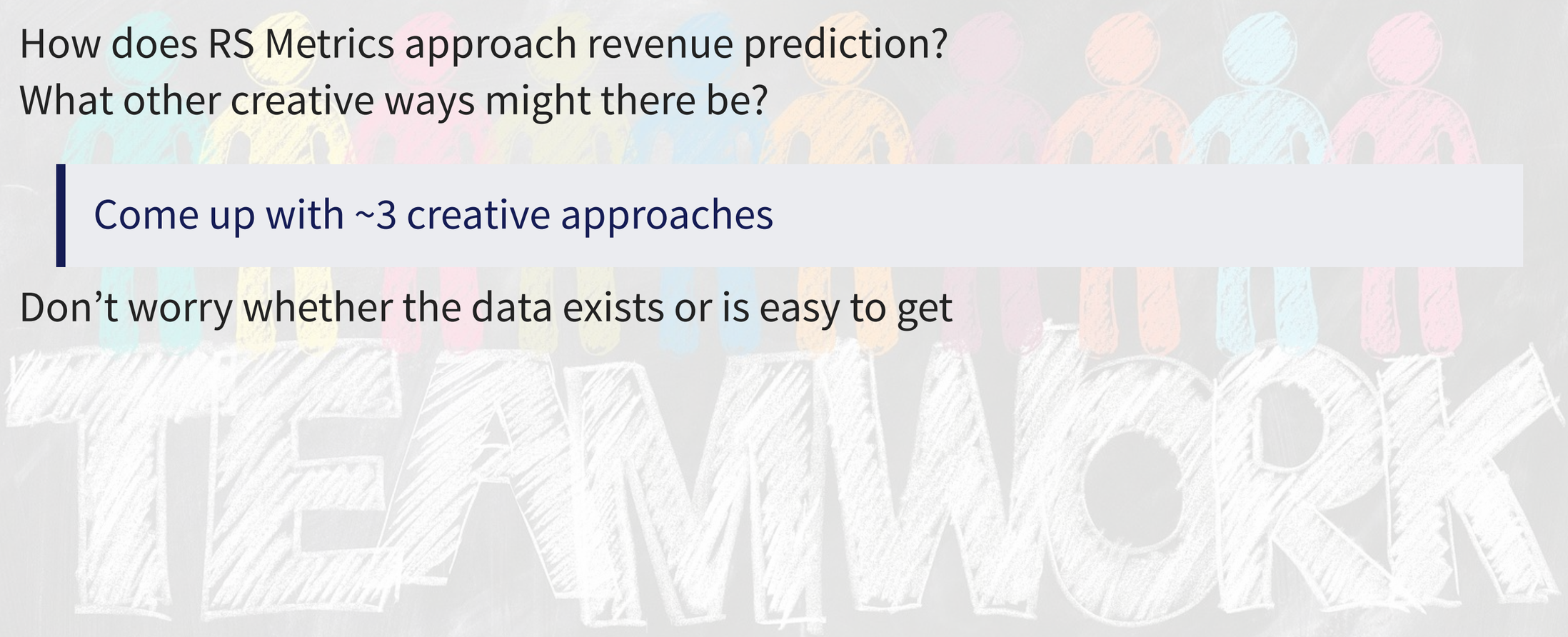
RS Metrics' approach

How they do it: rmc.link/420class3

- How does RS Metrics approach revenue prediction?
- What other creative ways might there be?

> Come up with ~3 creative approaches

- Don't worry whether the data exists or is easy to get



End Matter

Wrap up

- For next week:
  - First individual assignment is due!
    - Finish by next class
    - Submit on eLearn
  - Datacamp
    - Practice a bit more to keep up to date
      - Using R more will make it more natural
- Survey on the class session at this QR code:

Packages used for these slides

- broom
- DT
- downlit
- fixest
- kableExtra
- knitr
- lubridate
- plotly
- purrr
- quarto
- revealjs
- rlang
- tidyverse

Custom code

```r
# Brute force code for variable generation of quarterly data lags
df <- df %>%
  group_by(gvkey) %>%
  mutate(revtq_lag1=lag(revtq), revtq_lag2=lag(revtq, 2),
         revtq_lag3=lag(revtq, 3), revtq_lag4=lag(revtq, 4),
         revtq_lag5=lag(revtq, 5), revtq_lag6=lag(revtq, 6),
         revtq_lag7=lag(revtq, 7), revtq_lag8=lag(revtq, 8),
         revtq_lag9=lag(revtq, 9), revtq_gr=revtq / revtq_lag1 - 1,
         revtq_gr1=lag(revtq_gr), revtq_gr2=lag(revtq_gr, 2),
         revtq_gr3=lag(revtq_gr, 3), revtq_gr4=lag(revtq_gr, 4),
         revtq_gr5=lag(revtq_gr, 5), revtq_gr6=lag(revtq_gr, 6),
         revtq_gr7=lag(revtq_gr, 7), revtq_gr8=lag(revtq_gr, 8),
         revtq_yoy=revtq / revtq_lag4 - 1, revtq_yoy1=lag(revtq_yoy),
         revtq_yoy2=lag(revtq_yoy, 2), revtq_yoy3=lag(revtq_yoy, 3),
         revtq_yoy4=lag(revtq_yoy, 4), revtq_yoy5=lag(revtq_yoy, 5),
         revtq_yoy6=lag(revtq_yoy, 6), revtq_yoy7=lag(revtq_yoy, 7),
         revtq_yoy8=lag(revtq_yoy, 8), revtq_d=revtq - revtq_l1,
         revtq_d1=lag(revtq_d), revtq_d2=lag(revtq_d, 2),
         revtq_d3=lag(revtq_d, 3), revtq_d4=lag(revtq_d, 4),
         revtq_d5=lag(revtq_d, 5), revtq_d6=lag(revtq_d, 6),
         revtq_d7=lag(revtq_d, 7), revtq_d8=lag(revtq_d, 8)) %>%
  ungroup()
```

Custom code

```r
# Custom html table for small data frames
library(knitr)
library(kableExtra)
html_df <- function(text, cols=NULL, col1=FALSE, full=F) {
  if(!length(cols)) {
    cols=colnames(text)
  }
  if(!col1) {
    kable(text,"html", col.names = cols, align = c("l",rep('c',length(cols)-1))) %>%
      kable_styling(bootstrap_options = c("striped","hover"), full_width=full)
  } else {
    kable(text,"html", col.names = cols, align = c("l",rep('c',length(cols)-1))) %>%
      kable_styling(bootstrap_options = c("striped","hover"), full_width=full) %>%
      column_spec(1,bold=T)
  }
}
```

Custom code

```r
# Density plot for 1st to 99th percentile of both columns
plt_bar <- function(df,var) {
  df %>%
    filter(eval(parse(text=var)) < quantile(eval(parse(text=var)),0.99, na.rm=TRUE),
           eval(parse(text=var)) > quantile(eval(parse(text=var)),0.01, na.rm=TRUE)) %>%
    ggplot(aes(y=eval(parse(text=var)), x=qtr)) +
    geom_bar(stat = "summary", fun.y = "mean") + xlab(var)
}
```

```r
# Scatter plot with lag for 1st to 99th percentile of data
plt_sct <- function(df,var1, var2) {
  df %>%
    filter(eval(parse(text=var1)) < quantile(eval(parse(text=var1)),0.99, na.rm=TRUE),
           eval(parse(text=var2)) < quantile(eval(parse(text=var2)),0.99, na.rm=TRUE),
           eval(parse(text=var1)) > quantile(eval(parse(text=var1)),0.01, na.rm=TRUE),
           eval(parse(text=var2)) > quantile(eval(parse(text=var2)),0.01, na.rm=TRUE)) %>%
    ggplot(aes(y=eval(parse(text=var1)), x=eval(parse(text=var2)), color=factor(qtr))) +
    geom_point() + geom_smooth(method = "lm") + ylab(var1) + xlab(var2)
}
```

```r
# Calculating various in and out of sample statistics
models <- list(mod1,mod2,mod3)
model_names <- c("1 quarter", "1 and 4 quarters", "8 quarter")

df_test <- data.frame(adj_r_sq=sapply(models, function(x)summary(x)[["adj.r.squared"]]),
                      rmse_in=sapply(models, function(x)rmse(train$revtq, predict(x,train))),
                      mae_in=sapply(models, function(x)mae(train$revtq, predict(x,train))),
                      rmse_out=sapply(models, function(x)rmse(test$revtq, predict(x,test))),
                      mae_out=sapply(models, function(x)mae(test$revtq, predict(x,test))))
rownames(df_test) <- model_names
html_df(df_test)  # Custom function using knitr and kableExtra
```