

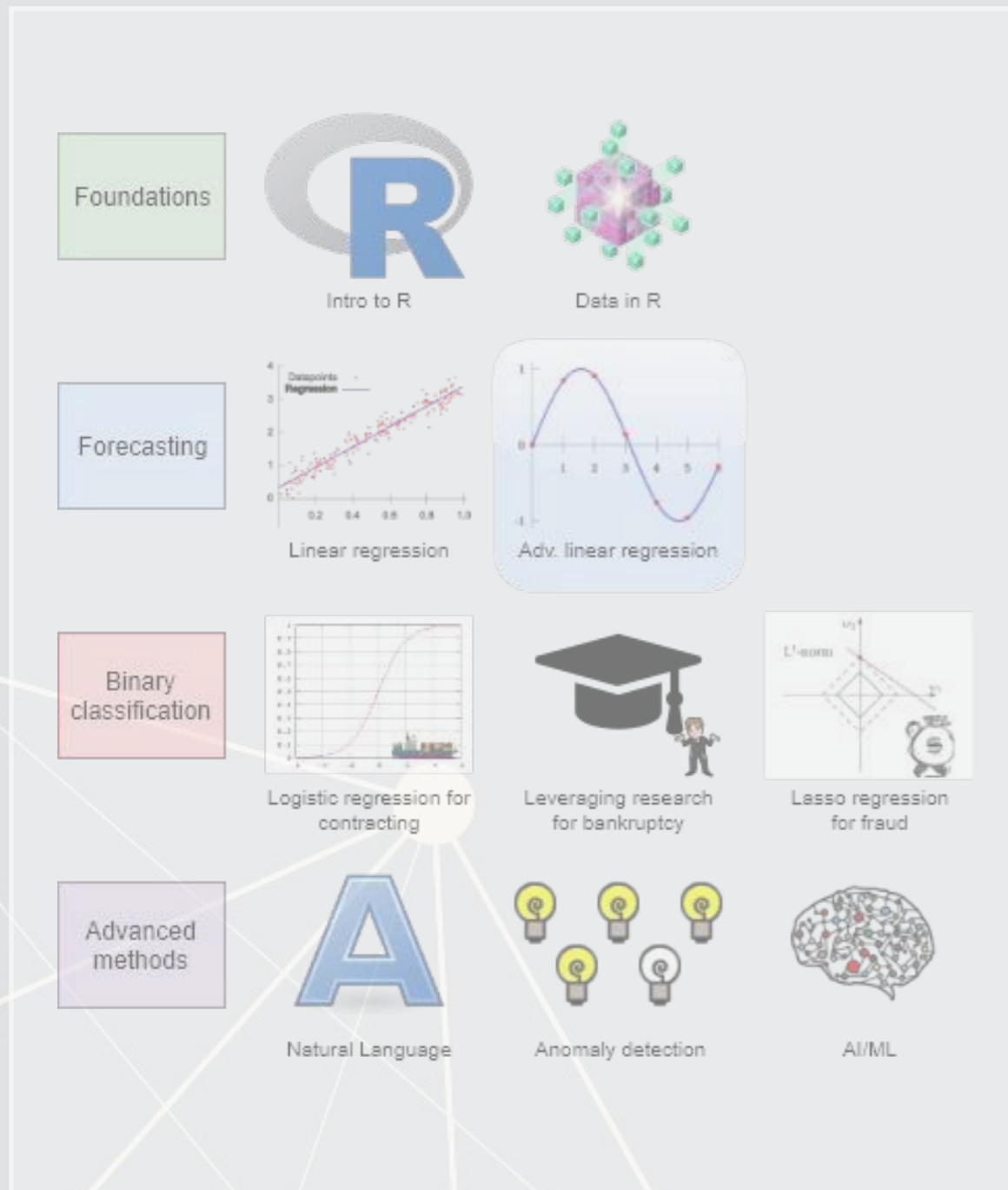
ACCT 420: Advanced linear regression

Session 4

Dr. Richard M. Crowley

Front matter

Learning objectives



- **Theory:**
 - Further understand:
 - Statistics
 - Causation
 - Data
 - Time
- **Application:**
 - Predicting revenue quarterly and weekly
- **Methodology:**
 - Univariate
 - Linear regression (OLS)
 - Visualization

Datacamp

- Explore on your own
- No specific required class this week

Based on your feedback...

- To help with replicating slides, each week I will release:
 1. A code file that can directly replicate everything in the slides
 2. The data files used, where allowable.
 - I may occasionally use proprietary data that I cannot distribute as is – those will not be distributed
- To help with coding
 1. I have released a practice on mutate and ggplot
 2. We will go back to having in class R practices when new concepts are included
- To help with statistics
 1. We will go over some statistics foundations today

Assignments for this course

- Based on feedback received today, I may host extra office hours on Wednesday

Quick survey: rnc.link/420hw1

Statistics Foundations

Frequentist statistics

A specific test is one of an infinite number of replications

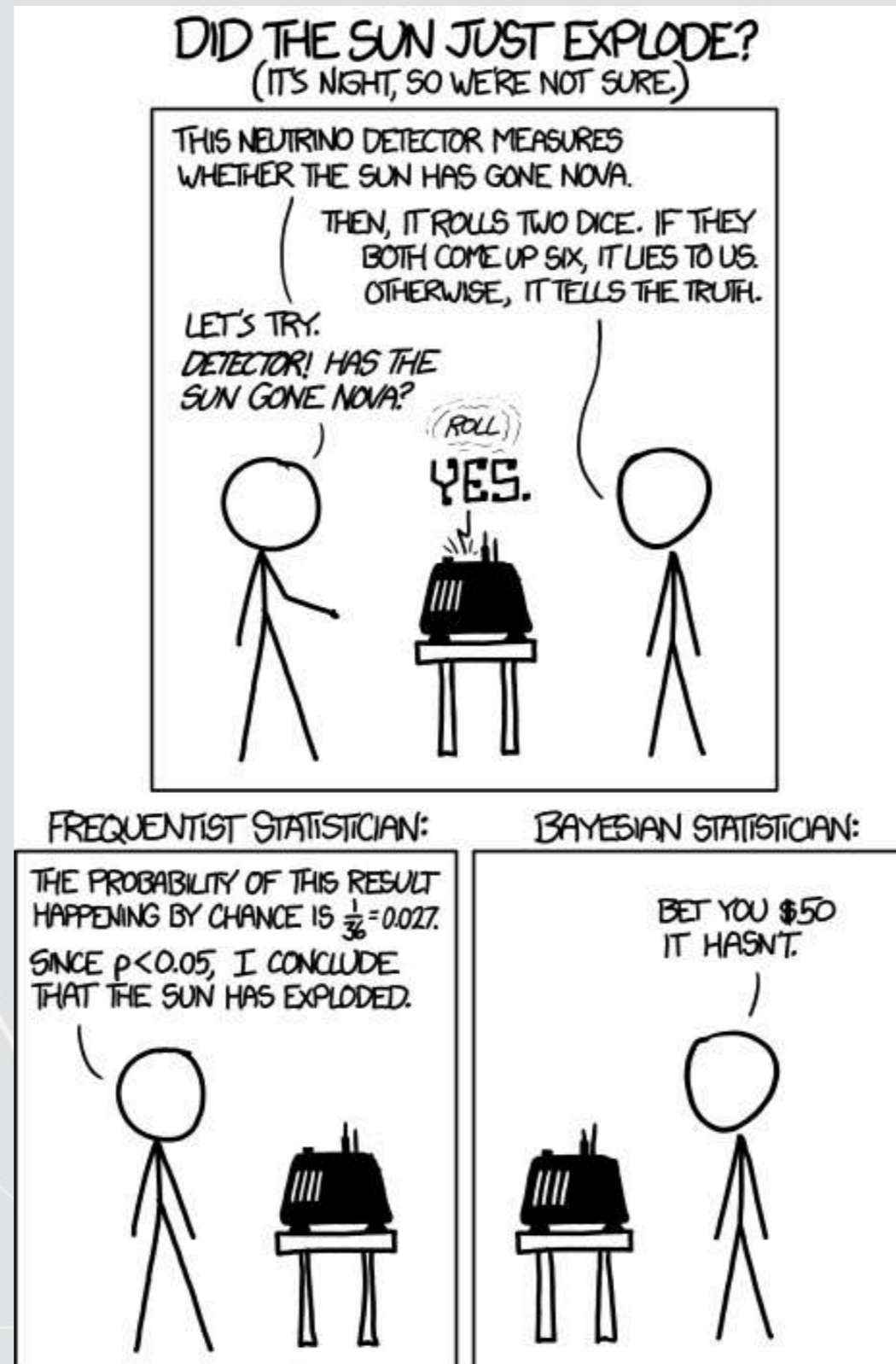
- The “correct” answer should occur most frequently, i.e., with a high probability
- Focus on true vs false
- Treat unknowns as fixed constants to figure out
 - Not random quantities
- Where it’s used
 - Classical statistics methods
 - Like OLS

Bayesian statistics

Focus on distributions and beliefs

- Prior distribution – what is believed before the experiment
- Posterior distribution: an updated belief of the distribution due to the experiment
- Derive distributions of parameters
- Where it's used:
 - Many machine learning methods
 - Bayesian updating acts as the learning
 - Bayesian statistics

Frequentist vs Bayesian methods



Frequentist perspective: Repeat the test

```
detector <- function() {  
  dice <- sample(1:6, size=2, replace=TRUE)  
  if (sum(dice) == 12) {  
    "exploded"  
  } else {  
    "still there"  
  }  
}
```

```
experiment <- replicate(1000,detector())  
# p value  
paste("p-value: ",  
      sum(experiment == "still there") / 1000,  
      "-- Reject H_A that sun exploded")
```

```
## [1] "p-value: 0.962 -- Reject H_A that sun exploded"
```

Frequentist: The sun didn't explode

Bayes perspective: Bayes rule

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

- A : The sun exploded
- B : The detector said it exploded
- $P(A)$: Really, really small. Say, ~ 0 .
- $P(B)$: $\frac{1}{6} \times \frac{1}{6} = \frac{1}{36}$
- $P(B|A)$: $\frac{35}{36}$

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} = \frac{\frac{35}{36} \times \sim 0}{\frac{1}{36}} = 35 \times \sim 0 \approx 0$$

Bayesian: The sun didn't explode

What analytics typically relies on

- Regression approaches
 - Most often done in a frequentist manner
 - Can be done in a Bayesian manner as well
- Artificial Intelligence
 - Often frequentist
 - Sometimes neither – “It just works”
- Machine learning
 - Sometimes Bayesian, sometime frequentist
 - We’ll see both

We will use both to some extent – for our purposes, we will not debate the merits of either school of thought, but use tools derived from both

Confusion from frequentist approaches

- Possible contradictions:
 - F test says the model is good yet nothing is statistically significant
 - Individual p -values are good yet the model isn't
 - One measure says the model is good yet another doesn't

There are many ways to measure a model, each with their own merits. They don't always agree, and it's on us to pick a reasonable measure.

Frequentist approaches to things

Hypotheses

- H_0 : The status quo is correct
 - Your proposed model doesn't work
- H_A : The model you are proposing works
- Frequentist statistics can never directly support H_0 !
 - Only can fail to find support for H_A
- Even if our p -value is 1, we can't say that the results prove the null hypothesis!

OLS terminology

- y : The output in our model
- \hat{y} : The *estimated* output in our model
- x_i : An input in our model
- \hat{x}_i : An *estimated* input in our model
- $\hat{\cdot}$: Something *estimated*
- α : A constant, the expected value of y when all x_i are 0
- β_i : A coefficient on an input to our model
- ε : The error term
 - This is also the *residual* from the regression
 - What's left if you take actual y minus the model prediction

Regression

- Regression (like OLS) has the following assumptions
 1. The data is generated following some model
 - E.g., a linear model
 - Next week, a logistic model
 2. The data conforms to some statistical properties as required by the test
 3. The model coefficients are something to precisely determine
 - I.e., the coefficients are constants
 4. p -values provide a measure of the chance of an error in a particular aspect of the model
 - For instance, the p -value on β_1 in $y = \alpha + \beta_1 x_1 + \varepsilon$ essentially gives the probability that the sign of β_1 is wrong

OLS Statistical properties

$$y = \alpha + \beta_1 x_1 + \beta_2 x_2 + \dots + \varepsilon$$

$$\hat{y} = \alpha + \beta_1 \hat{x}_1 + \beta_2 \hat{x}_2 + \dots + \hat{\varepsilon}$$

1. There should be a *linear* relationship between y and each x_i
 - I.e., y is [approximated by] a constant multiple of each x_i
 - Otherwise we **shouldn't** use a *linear* regression
2. Each \hat{x}_i is normally distributed
 - Not so important with larger data sets, but a good to adhere to
3. Each observation is independent
 - We'll violate this one for the sake of *causality*
4. Homoskedasticity: Variance in errors is constant
 - This is important
5. Not too much multicollinearity
 - Each \hat{x}_i should be relatively independent from the others
 - Some is OK

Practical implications

Models designed under a frequentist approach can only answer the question of “does this matter?”

- Is this a problem?

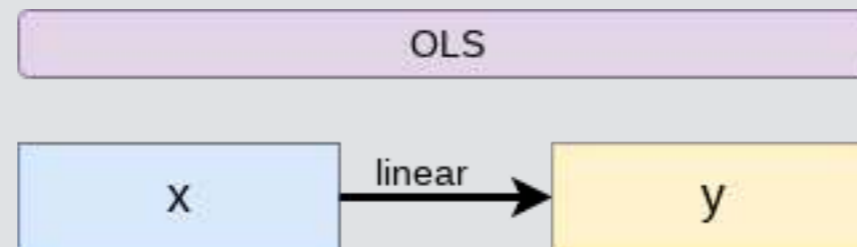
Linear model implementation

What exactly is a linear model?

- Anything OLS is linear
- Many transformations can be recast to linear
 - Ex.: $\log(y) = \alpha + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1^2 + \beta_4 x_1 \cdot x_2$
 - This is the same as $y' = \alpha + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4$
where:
 - $y' = \log(y)$
 - $x_3 = x_1^2$
 - $x_4 = x_1 \cdot x_2$

Linear models are *very* flexible

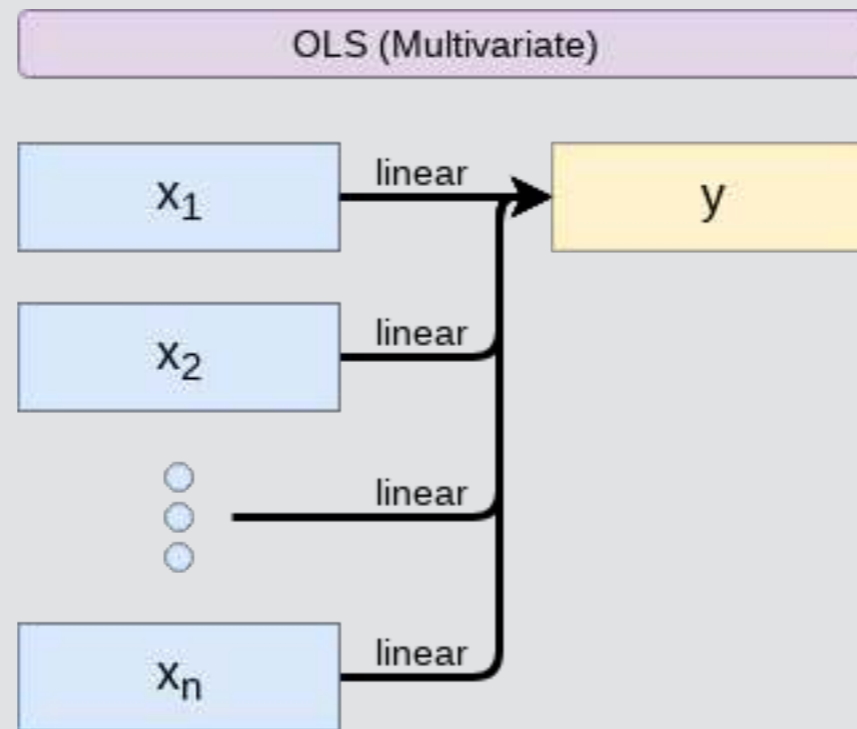
Mental model of OLS: 1 input



Simple OLS measures a simple linear relationship between an input and an output

- E.g.: Our first regression last week: Revenue on assets

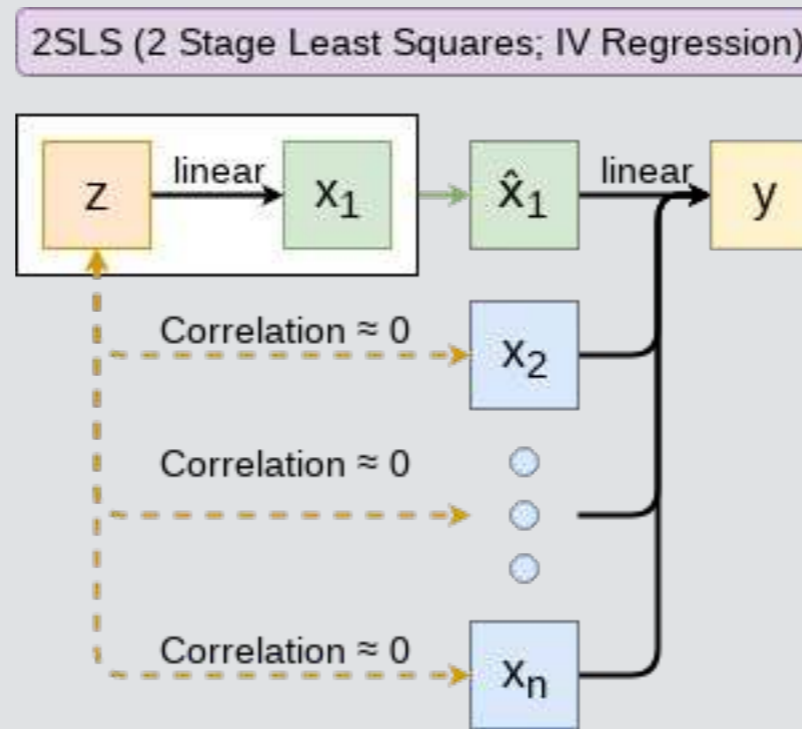
Mental model of OLS: Multiple inputs



OLS measures simple linear relationships between a set of inputs and one output

- E.g.: Our main models last week: Future revenue regressed on multiple accounting and macro variables

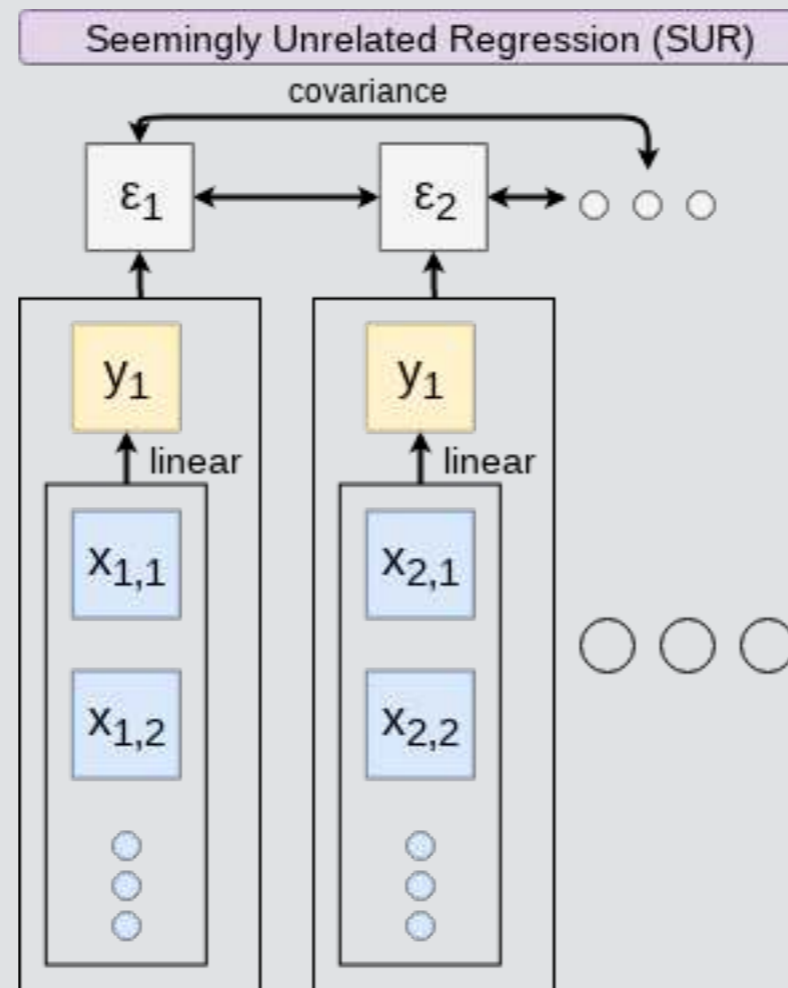
Other linear models: IV Regression (2SLS)



IV/2SLS models linear relationships where the effect of some x_i on y may be confounded by outside factors.

- E.g.: Modeling the effect of management pay duration (like bond duration) on firms' choice to issue earnings forecasts
- Instrument with CEO tenure (Cheng, Cho, and Kim 2015)

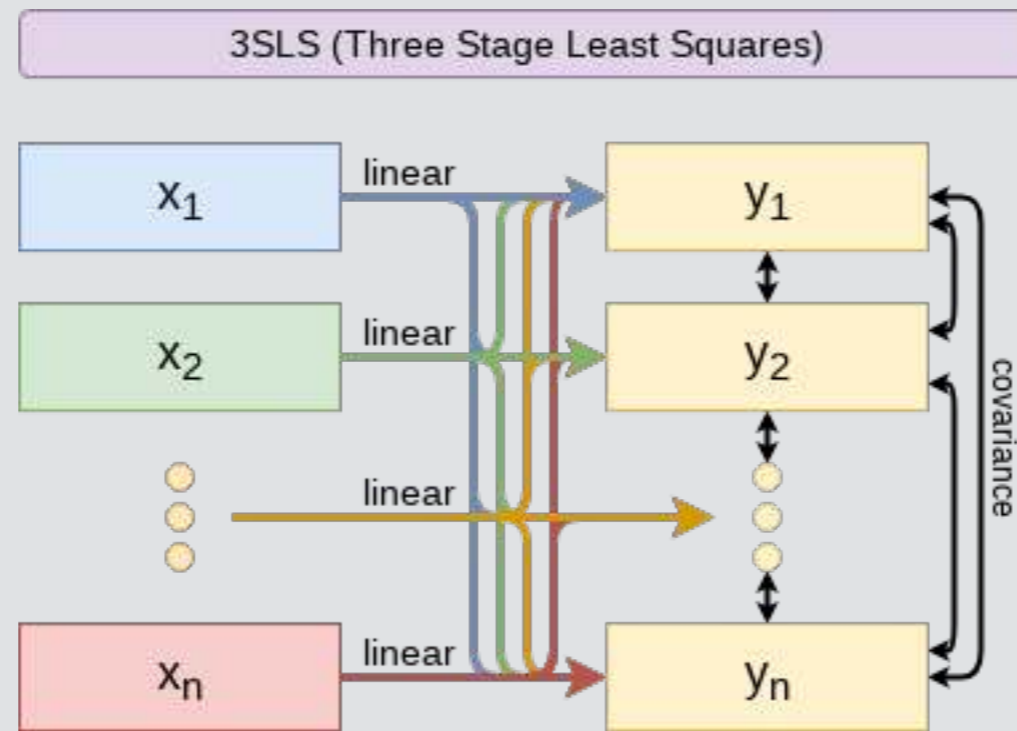
Other linear models: SUR



SUR models systems with related error terms

- E.g.: Modeling both revenue and earnings simultaneously

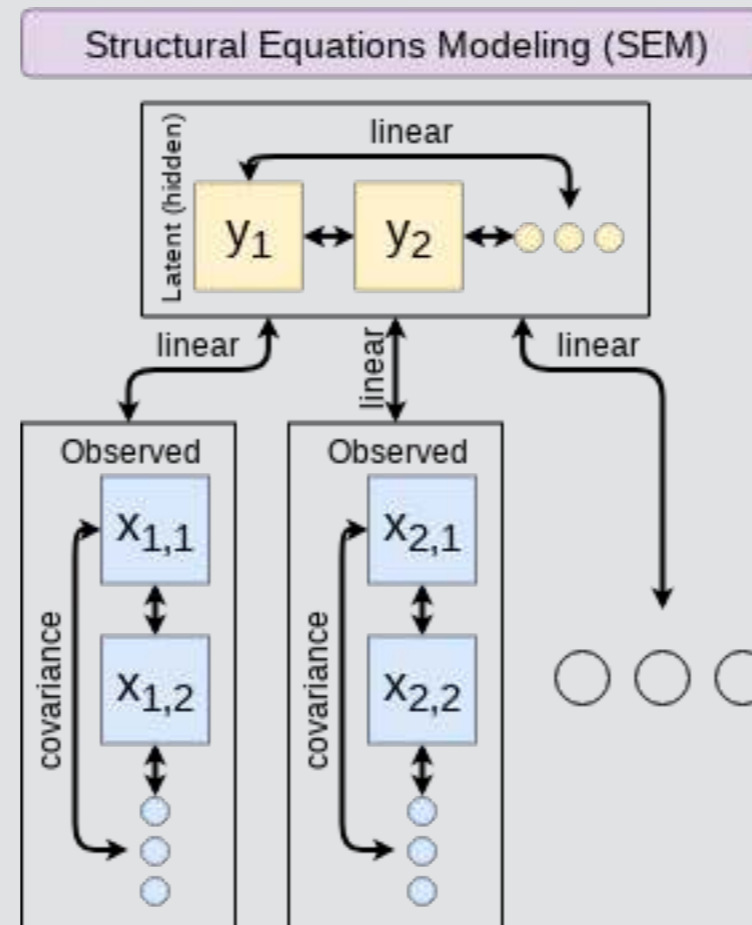
Other linear models: 3SLS



3SLS models systems of equations with *related outputs*

- E.g.: Modeling both stock return, volatility, and volume simultaneously

Other linear models: SEM



SEM can model abstract and multi-level relationships

- E.g.: Showing that organizational commitment leads to higher job satisfaction, not the other way around (Poznanski and Bline 1999)

Modeling choices: Model selection

Pick what fits your problem!

- For forecasting a quantity
 - Usually some sort of linear model regressed using OLS
 - The other model types mentioned are great for simultaneous forecasting of multiple outputs
- For forecasting a binary outcome
 - Usually logit or a related model (we'll start this next week)
- For forensics:
 - Usually logit or a related model

Modeling choices: Variable selection

- The options:
 1. Use your own knowledge to select variables
 2. Use a selection model to automate it

Own knowledge

- Build a model based on your knowledge of the problem and situation
- This is generally better
 - The result should be more interpretable
 - For prediction, you should know relationships better than most algorithms



Modeling choices: Automated selection

- Traditional methods include:
 - Forward selection: Start with nothing and add variables with the most contribution to Adj R^2 until it stops going up
 - Backward selection: Start with all inputs and remove variables with the worst (negative) contribution to Adj R^2 until it stops going up
 - Stepwise selection: Like forward selection, but drops non-significant predictors
- Newer methods:
 - Lasso and Elastic Net based models
 - Optimize with high penalties for complexity (i.e., # of inputs)
 - We will discuss these in week 6



The overfitting problem

Or: Why do we like simpler models so much?

- Overfitting happens when a model fits in-sample data *too well...*
 - To the point where it also models any idiosyncrasies or errors in the data
 - This harms prediction performance
 - Directly harming our forecasts

An overfitted model works really well on its own data, and quite poorly on new data

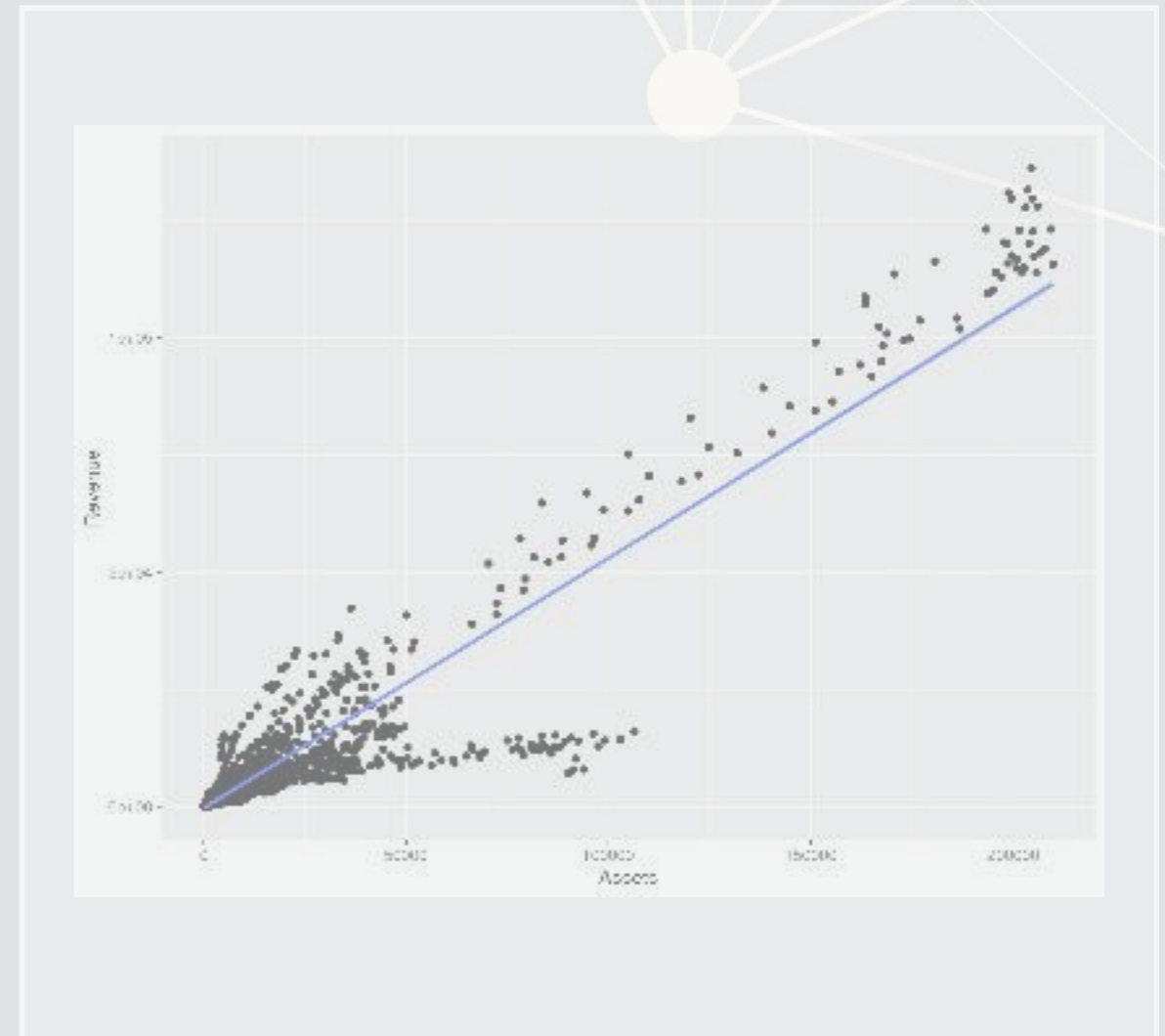
Statistical tests and Interpretation

Coefficients

- In OLS:

$$\beta_i$$

- A change in x_i by 1 leads to a change in y by β_i
- Essentially, the slope between x and y
- The blue line in the chart is the regression line for $\hat{Revenue} = \alpha + \beta_i \hat{Assets}$ for retail firms since 1960



P-values

- p -values tell us the probability that an individual result is due to random chance

“The P value is defined as the probability under the assumption of no effect or no difference (null hypothesis), of obtaining a result equal to or more extreme than what was actually observed.””

– Dahiru 2008

- These are very useful, particularly for a frequentist approach
- First used in the 1700s, but popularized by Ronald Fisher in the 1920s and 1930s

P-values: Rule of thumb

- If $p < 0.05$ and the coefficient matches our mental model, we can consider this as supporting our model
 - If $p < 0.05$ but the coefficient is opposite, then it is suggesting a problem with our model
 - If $p > 0.10$, it is rejecting the alternative hypothesis
- If $0.05 < p < 0.10$ it depends...
 - For a small dataset or a complex problem, we can use 0.10 as a cutoff
 - For a huge dataset or a simple problem, we should use 0.05

One vs two tailed tests

- Best practice:
 - Use a two tailed test
- Second best practice:
 - If you use a 1-tailed test, use a p-value cutoff of 0.025 or 0.05
 - This is equivalent to the best practice, just roundabout
- Common but semi-inappropriate:
 - Use a two tailed test with cutoffs of 0.05 or 0.10 because your hypothesis is directional

R^2

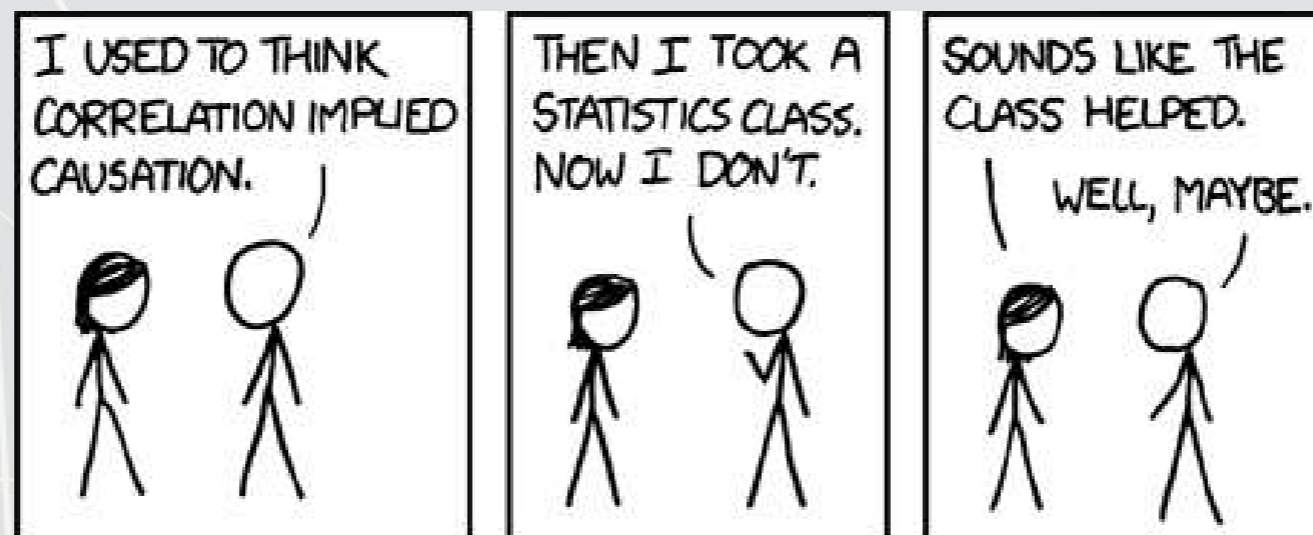
- $R^2 = \text{Explained variation} / \text{Total variation}$
- Variation = difference in the observed output variable from its own mean
- A high R^2 indicates that the model fits the data very well
- A low R^2 indicates that the model is missing much of the variation in the output
- R^2 is technically a *biased* estimator
- Adjusted R^2 downweights R^2 and makes it unbiased
 - $R^2_{Adj} = PR^2 + 1 - P$
 - Where $P = \frac{n-1}{n-p-1}$
 - n is the number of observations
 - p is the number of inputs in the model

Causality

What is causality?

$$A \rightarrow B$$

- Causality is *A causing B*
 - This means more than *A* and *B* are correlated
- I.e., If *A* changes, *B* changes. But *B* changing doesn't mean *A* changed
 - Unless *B* is 100% driven by *A*
- Very difficult to determine, particularly for events that happen [almost] simultaneously
- Examples of correlations that aren't causation



Time and causality

$$A \rightarrow B \text{ or } A \leftarrow B?$$

$$A_t \rightarrow B_{t+1}$$

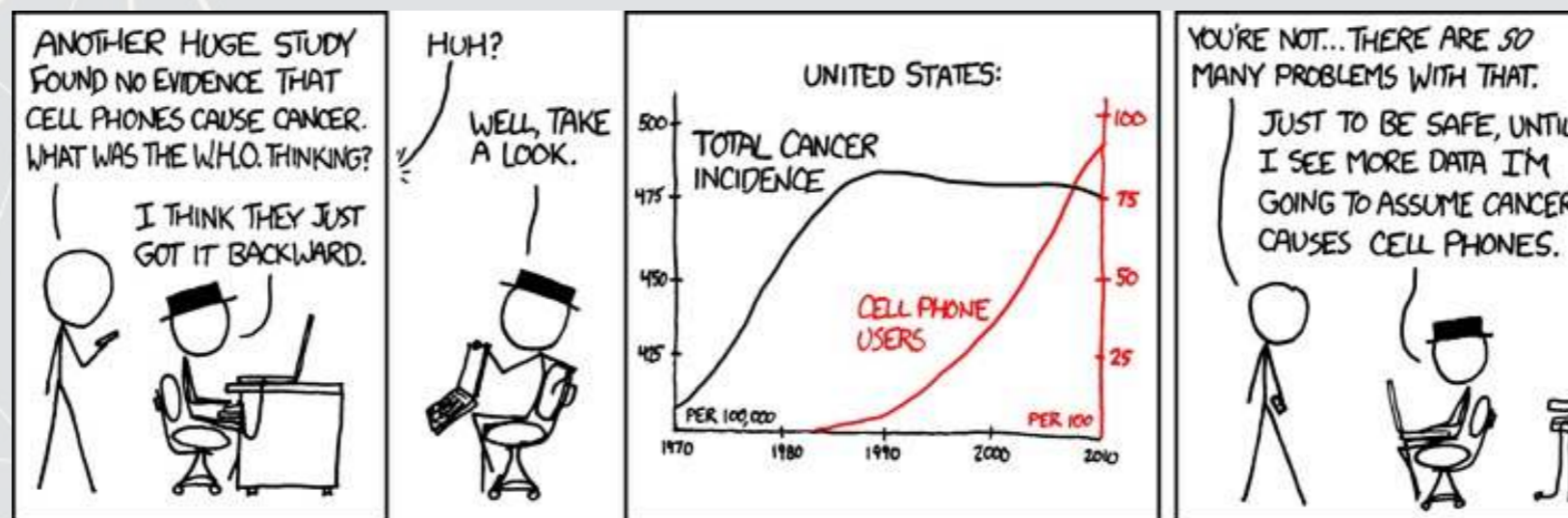
- If there is a separation in time, it's easier to say A caused B
 - Observe A , then see if B changes after
 - Conveniently, we have this structure when forecasting
 - Recall last week's model:

$$\text{Revenue}_{t+1} = \text{Revenue}_t + \dots$$

Time and causality break down

$$A_t \rightarrow B_{t+1}? \quad \text{OR} \quad C \rightarrow A_t \text{ and } C \rightarrow B_{t+1}?$$

- The above illustrates the *Correlated omitted variable problem*
 - A doesn't cause B ... Instead, some other force C causes both
 - Bane of social scientists everywhere
- This is less important for predictive analytics, as we care more about performance, but...
 - It can complicate interpreting your results
 - Figuring out C can help improve your model's predictions
 - So find C !



Today's application: Quarterly retail revenue

The question

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

- In aggregate
- By Store
- By department

More specifically...

- Consider time dimensions
 - What matters:
 - Last quarter?
 - Last year?
 - Other timeframes?
 - Cyclicity

Time and OLS

Time matters a lot for retail



Great Singapore Sale

How to capture time effects?

- Autoregression
 - Regress y_t on earlier value(s) of itself
 - Last quarter, last year, etc.
- Controlling for time directly in the model
 - Essentially the same as fixed effects last week



Quarterly revenue prediction

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: 2015-2017

Variable generation

```
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 YYMMDDs10: YYYY/MM/DD, can be converted from text to date easily
df$date <- as.Date(df$datadate) # Built in to R
```

- Use mutate for variables using lags
- `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 x 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
```

Create 8 quarters (2 years) of lags

```
# Custom Function to generate a series of lags
multi_lag <- function(df, lags, var, ext="") {
  lag_names <- paste0(var,ext,lags)
  lag_funs <- setNames(paste("dplyr::lag(.,",lags,")"), lag_names)
  df %>% group_by(gvkey) %>% mutate_at(vars(var), funs_(lag_funs)) %>% ungroup()
}
# Generate lags "revtq_l#"
df <- multi_lag(df, 1:8, "revtq", "_l")

# Generate changes "revtq_gr#"
df <- multi_lag(df, 1:8, "revtq_gr")

# Generate year-over-year changes "revtq_yoy#"
df <- multi_lag(df, 1:8, "revtq_yoy")

# Generate first differences "revtq_d#"
df <- multi_lag(df, 1:8, "revtq_d")

# Equivalent brute force code for this is in the appendix
```

- `paste0()`: creates a string vector by concatenating all inputs
- `paste()`: same as `paste0()`, but with spaces added in between
- `setNames()`: allows for storing a value and name simultaneously
- `mutate_at()`: is like `mutate` but with a list of functions

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

```
# Clean the data: Replace NaN, Inf, and -Inf with NA  
df <- df %>%  
  mutate_if(is.numeric, funs(replace(., !is.finite(.), NA)))
```

```
# Split into training and testing data  
# Training data: We'll use data released before 2015  
train <- filter(df, year(date) < 2015)
```

```
# Testing data: We'll use data released 2015 through 2018  
test <- filter(df, year(date) >= 2015)
```

- 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", "fqtr")])
```

```
##   revtq      revtq_gr      revtq_yoy
## Min.   : 0.00 Min.   :-1.0000 Min.   :-1.0000
## 1st Qu.: 64.46 1st Qu.: -0.1112 1st Qu.: 0.0077
## Median : 273.95 Median : 0.0505 Median : 0.0740
## Mean   : 2439.38 Mean   : 0.0650 Mean   : 0.1273
## 3rd Qu.: 1254.21 3rd Qu.: 0.2054 3rd Qu.: 0.1534
## Max.   :136267.00 Max.   :14.3333 Max.   :47.6600
## NA's   :367      NA's   :690      NA's   :940
##   revtq_d      fqtr
## Min.   :-24325.21 Min.   :1.000
## 1st Qu.: -19.33 1st Qu.:1.000
## Median :  4.30 Median :2.000
## Mean   :  22.66 Mean   :2.478
## 3rd Qu.:  55.02 3rd Qu.:3.000
## Max.   :15495.00 Max.   :4.000
## NA's   :663
```

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 (`+`)

```
library(ggplot2) # or tidyverse -- it's part of tidyverse
df %>%
  ggplot(aes(y=var_for_y_axis, x=var_for_y_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](#)

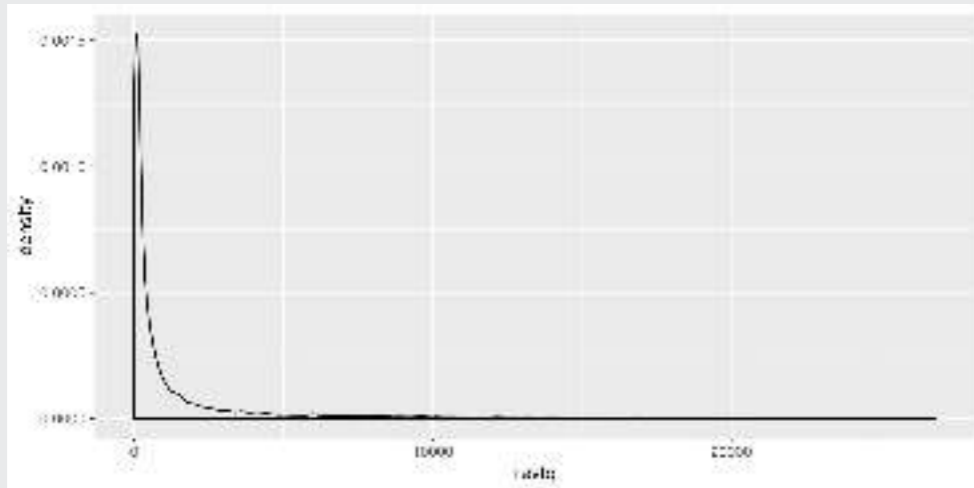
ggplot2 for visualization

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

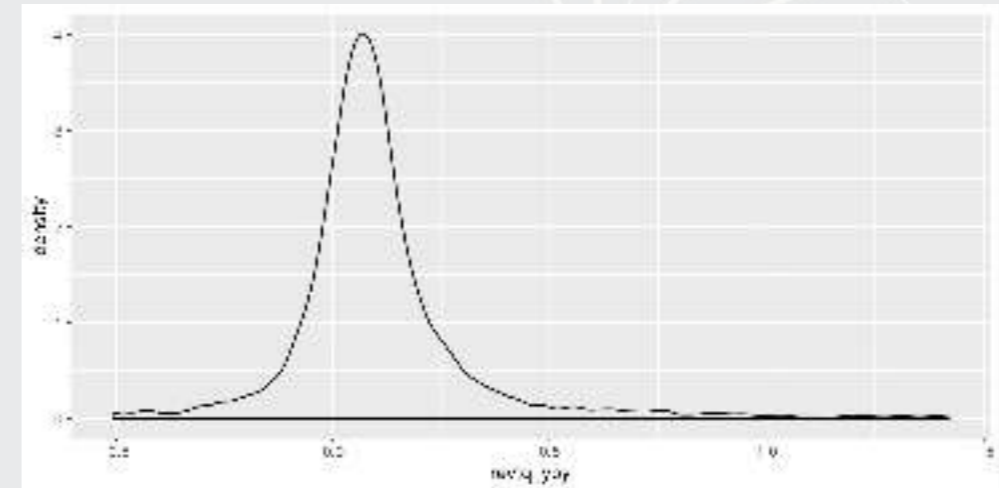
- geom stands for geometry – any shapes, lines, etc. start with geom
- Other useful geoms:
 - geom_line(): makes a line chart
 - geom_bar(): makes a bar chart – y is the height, x is the category
 - geom_smooth(method="lm"): Adds a linear regression into the chart
 - geom_abline(slope=1): Adds a 45° 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’](#) on eLearn

Plotting: Distribution of revenue

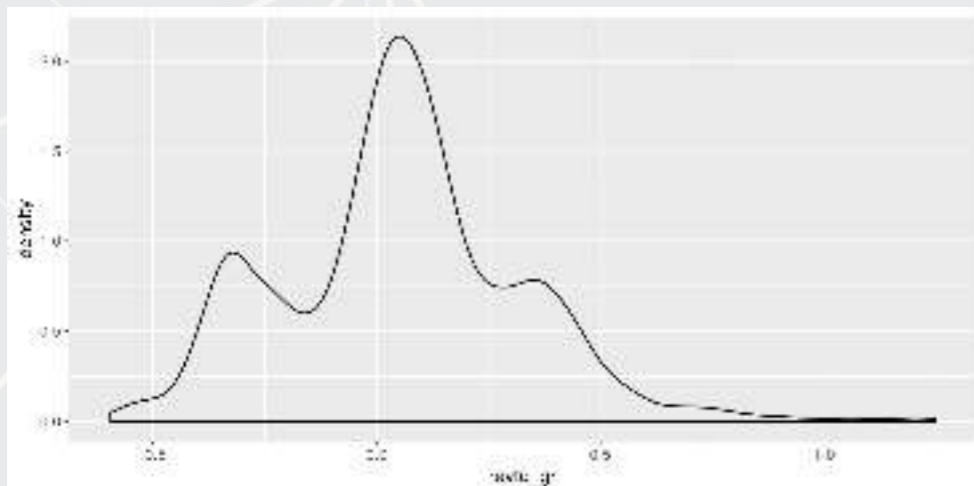
1. Revenue



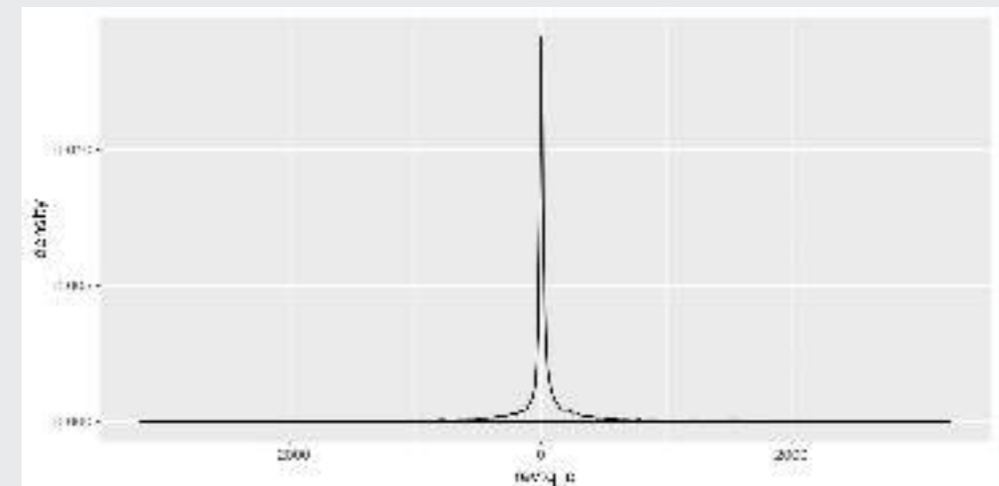
3. Year-over-year growth



2. Quarterly growth



4. First difference

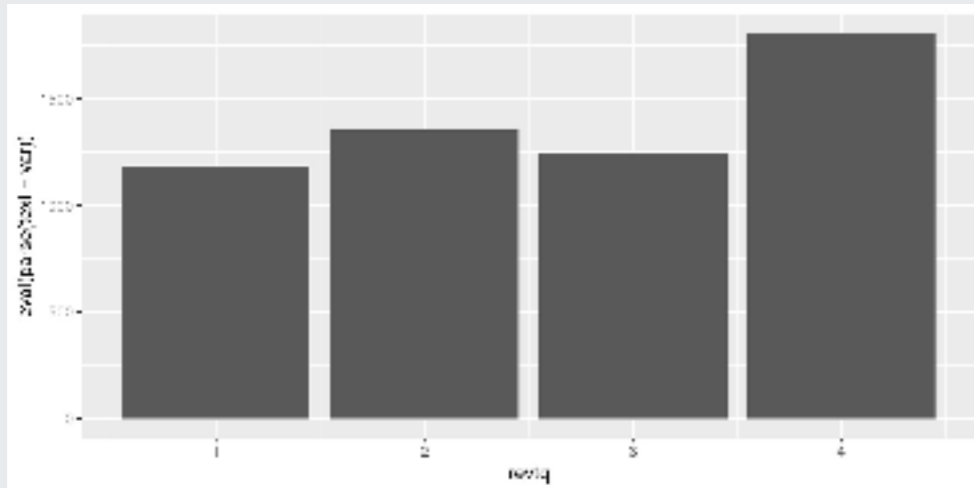


What do we learn from these graphs?

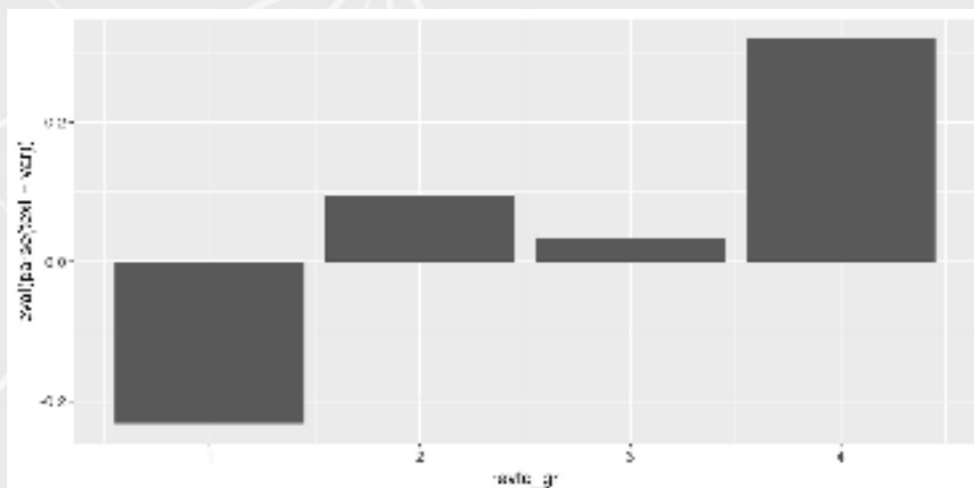
1. Revenue
 -
2. Quarterly growth
 -
3. Year-over-year growth
 -
4. First difference
 -

Plotting: Mean revenue by quarter

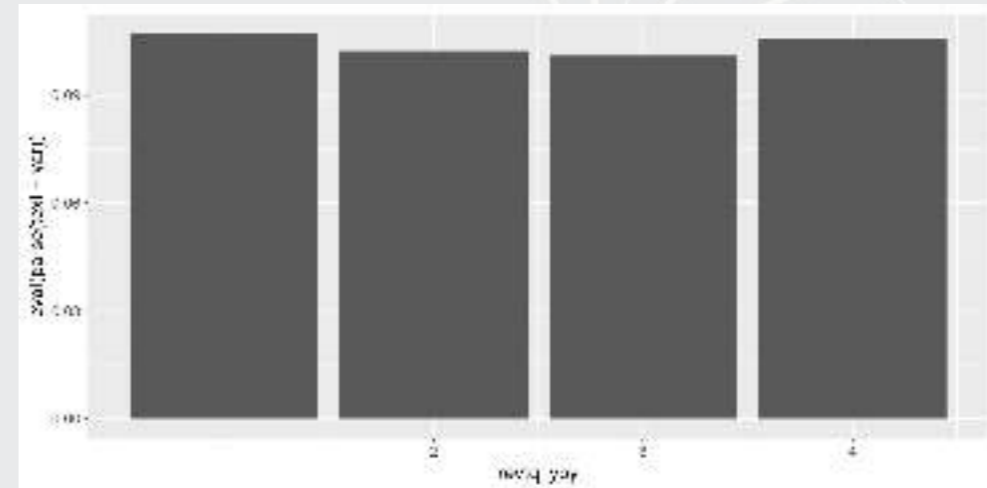
1. Revenue



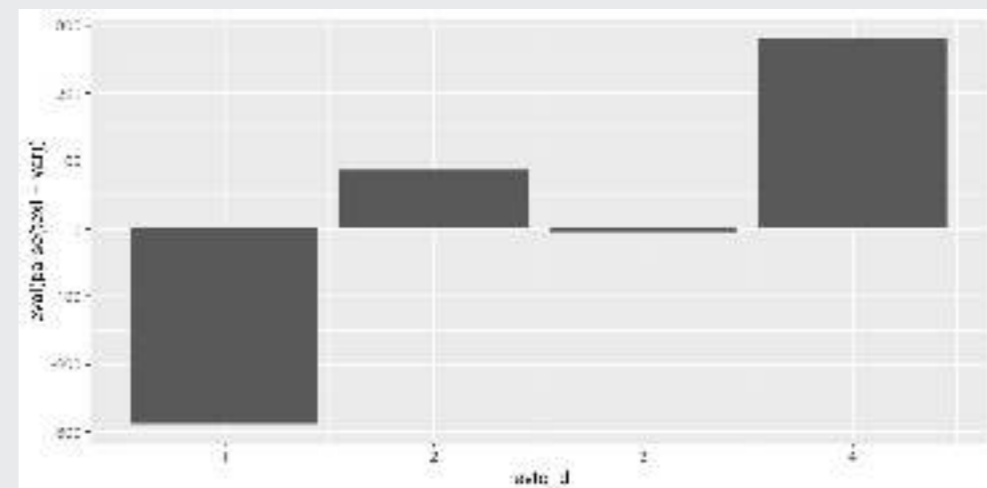
2. Quarterly growth



3. Year-over-year growth



4. First difference

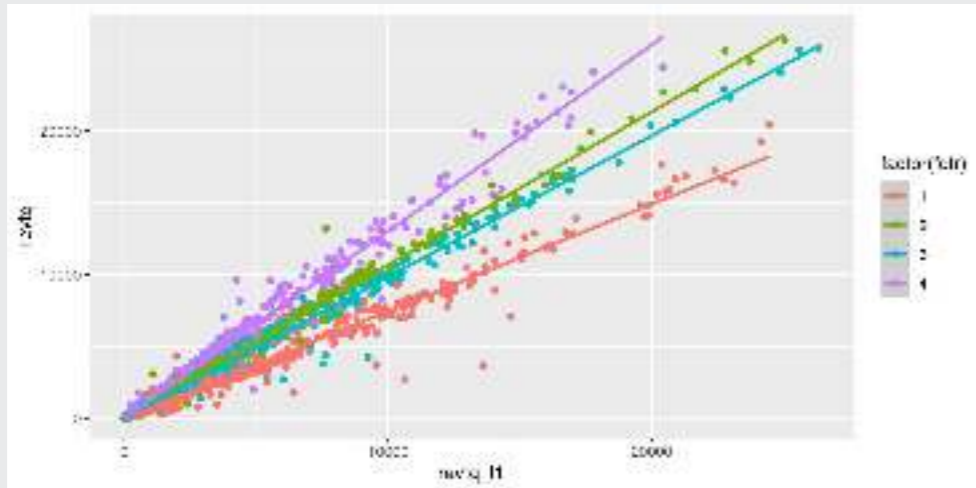


What do we learn from these graphs?

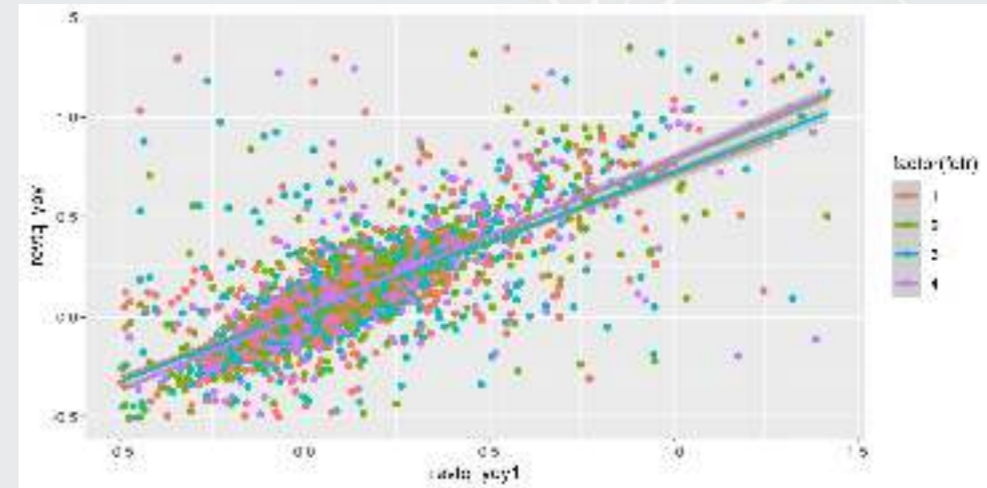
1. Revenue
 -
2. Quarterly growth
 -
3. Year-over-year growth
 -
4. First difference
 -

Plotting: Revenue vs lag by quarter

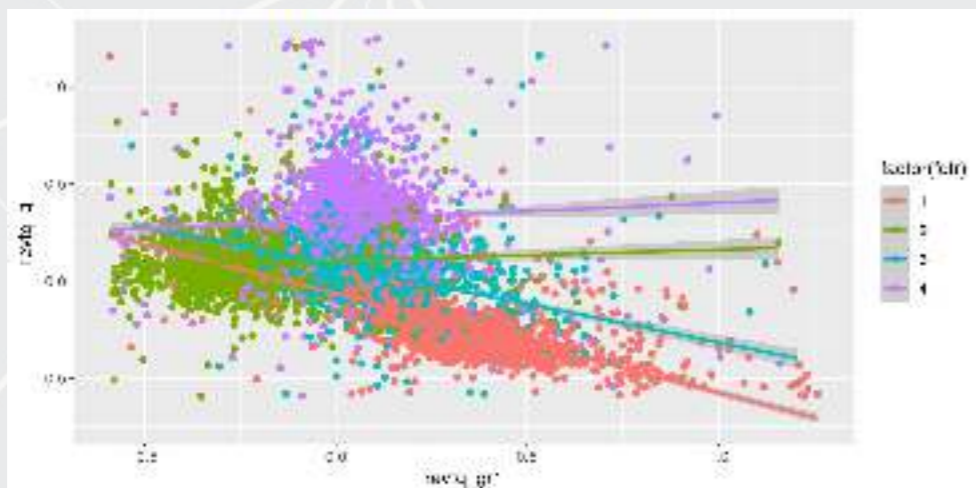
1. Revenue



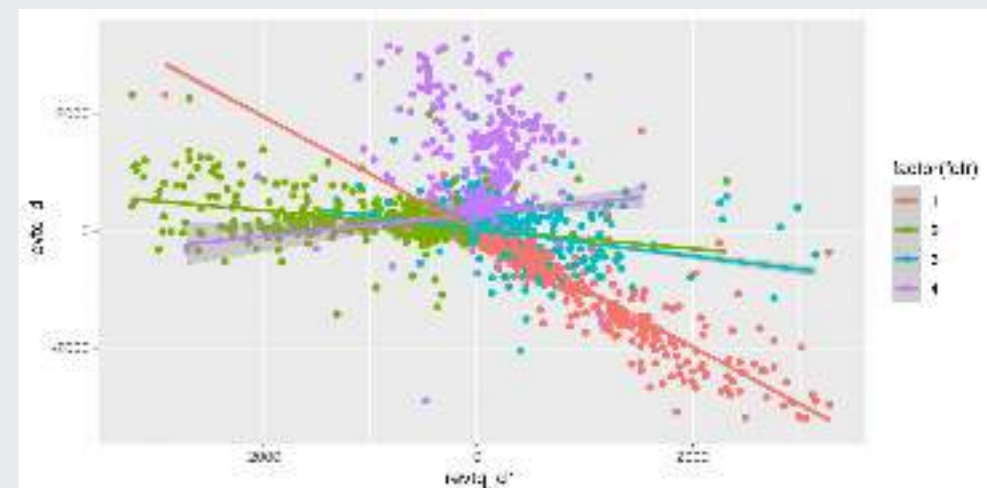
3. Year-over-year growth



2. Quarterly 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 0.9916167 0.9938489 0.9905522 0.9972735  
## revtq_l1 0.9916167 1.0000000 0.9914767 0.9936977 0.9898184  
## revtq_l2 0.9938489 0.9914767 1.0000000 0.9913489 0.9930152  
## revtq_l3 0.9905522 0.9936977 0.9913489 1.0000000 0.9906006  
## revtq_l4 0.9972735 0.9898184 0.9930152 0.9906006 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  1.00000000 -0.32291329 0.06299605 -0.22769442 0.64570015  
## revtq_gr1 -0.32291329 1.00000000 -0.31885020 0.06146805 -0.21923630  
## revtq_gr2 0.06299605 -0.31885020 1.00000000 -0.32795121 0.06775742  
## revtq_gr3 -0.22769442 0.06146805 -0.32795121 1.00000000 -0.31831023  
## revtq_gr4 0.64570015 -0.21923630 0.06775742 -0.31831023 1.00000000
```

Retail revenue has really high autocorrelation! Concern for multicollinearity. 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  1.0000000  0.6554179  0.4127263  0.4196003  0.1760055  
## revtq_yoy1 0.6554179  1.0000000  0.5751128  0.3665961  0.3515105  
## revtq_yoy2 0.4127263  0.5751128  1.0000000  0.5875643  0.3683539  
## revtq_yoy3 0.4196003  0.3665961  0.5875643  1.0000000  0.5668211  
## revtq_yoy4 0.1760055  0.3515105  0.3683539  0.5668211  1.0000000
```

```
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 -0.6181516  0.3309349 -0.6046998  0.9119911  
## revtq_d1 -0.6181516  1.0000000 -0.6155259  0.3343317 -0.5849841  
## revtq_d2  0.3309349 -0.6155259  1.0000000 -0.6191366  0.3165450  
## revtq_d3 -0.6046998  0.3343317 -0.6191366  1.0000000 -0.5864285  
## revtq_d4  0.9119911 -0.5849841  0.3165450 -0.5864285  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
 - Cyclicity
- Practice using:
 - `mutate()`
 - `lm()`
 - `ggplot2`
- Do the exercises in today's practice file
 - [R Practice](#)
 - Shortlink: rmc.link/420r4

Forecasting

1 period models

1. 1 Quarter lag

- We saw a very strong linear pattern here earlier

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

2. Quarter and year lag

- Year-over-year seemed pretty constant

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

3. 2 years of lags

- Other lags could also help us predict

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

4. 2 years of lags, by observation quarter

- Take into account cyclicity observed in bar charts

```
mod4 <- lm(revtq ~ (revtq_l1 + revtq_l2 + revtq_l3 + revtq_l4 +  
                 revtq_l5 + revtq_l6 + revtq_l7 + revtq_l8):factor(fqtr),  
           data=train)
```

Quarter lag

```
summary(mod1)
```

```
##  
## Call:  
## lm(formula = revtq ~ revtq_l1, data = train)  
##  
## Residuals:  
##      Min       1Q   Median       3Q      Max   
## -24438.7  -34.0  -11.7   34.6  15200.5   
##  
## Coefficients:  
##              Estimate Std. Error t value Pr(>|t|)      
## (Intercept) 15.639975  13.514877   1.157   0.247      
## revtq_l1    1.003038   0.001556 644.462 <2e-16 ***  
## ---  
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1  
##  
## Residual standard error: 1152 on 7676 degrees of freedom  
## (662 observations deleted due to missingness)  
## Multiple R-squared:  0.9819, Adjusted R-squared:  0.9819   
## F-statistic: 4.153e+05 on 1 and 7676 DF, p-value: < 2.2e-16
```


Quarter and year lag

`summary(mod2)`

```
##
## Call:
## lm(formula = revtq ~ revtq_l1 + revtq_l4, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -20245.7  -18.4   -4.4   19.1  9120.8
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5.444986   7.145633   0.762   0.446
## revtq_l1     0.231759   0.005610  41.312 <2e-16 ***
## revtq_l4     0.815570   0.005858 139.227 <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 592.1 on 7274 degrees of freedom
## (1063 observations deleted due to missingness)
## Multiple R-squared:  0.9954, Adjusted R-squared:  0.9954
## F-statistic: 7.94e+05 on 2 and 7274 DF, p-value: < 2.2e-16
```

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
## -5005.6  -12.9   -3.7    9.3  5876.3
##
## Coefficients:
##           Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.02478   4.37003   0.921  0.3571
## revtq_l1     0.77379   0.01229  62.972 < 2e-16 ***
## revtq_l2     0.10497   0.01565   6.707 2.16e-11 ***
## revtq_l3    -0.03091   0.01538  -2.010  0.0445 *
## revtq_l4     0.91982   0.01213  75.800 < 2e-16 ***
## revtq_l5    -0.76459   0.01324 -57.749 < 2e-16 ***
## revtq_l6    -0.08080   0.01634  -4.945 7.80e-07 ***
## revtq_l7     0.01146   0.01594   0.719  0.4721
## revtq_l8     0.07924   0.01209   6.554 6.03e-11 ***
##
.....
```

2 years of lags, by observation quarter

summary(mod4)

```
##
## Call:
## lm(formula = revtq ~ (revtq_l1 + revtq_l2 + revtq_l3 + revtq_l4 +
##   revtq_l5 + revtq_l6 + revtq_l7 + revtq_l8):factor(fqtr),
##   data = train)
##
## Residuals:
##   Min     1Q   Median     3Q    Max
## -6066.6  -13.9    0.1   15.1  4941.1
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -0.201107  4.004046  -0.050  0.959944
## revtq_l1:factor(fqtr)1  0.488584  0.021734  22.480 < 2e-16 ***
## revtq_l1:factor(fqtr)2  1.130563  0.023017  49.120 < 2e-16 ***
## revtq_l1:factor(fqtr)3  0.774983  0.028727  26.977 < 2e-16 ***
## revtq_l1:factor(fqtr)4  0.977353  0.026888  36.349 < 2e-16 ***
## revtq_l2:factor(fqtr)1  0.258024  0.035136   7.344 2.33e-13 ***
## revtq_l2:factor(fqtr)2 -0.100284  0.024664  -4.066 4.84e-05 ***
## revtq_l2:factor(fqtr)3  0.212954  0.039698   5.364 8.40e-08 ***
```

Testing out of sample

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

```
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* that part of the error

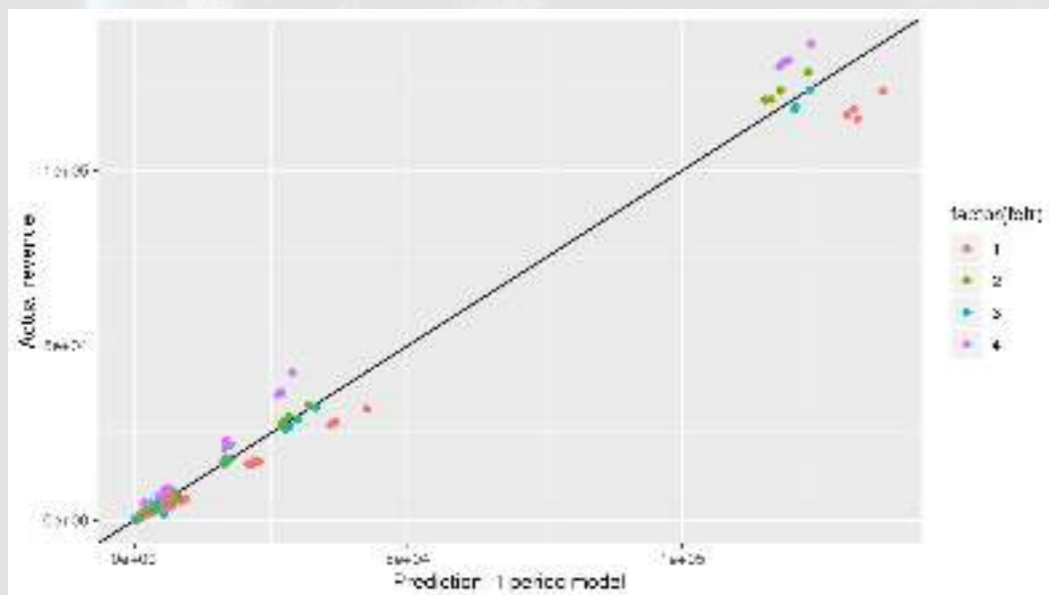
```
mae <- function(v1, v2) {  
  mean(abs(v1-v2), na.rm=T)  
}
```

Both are commonly used for evaluating OLS out of sample

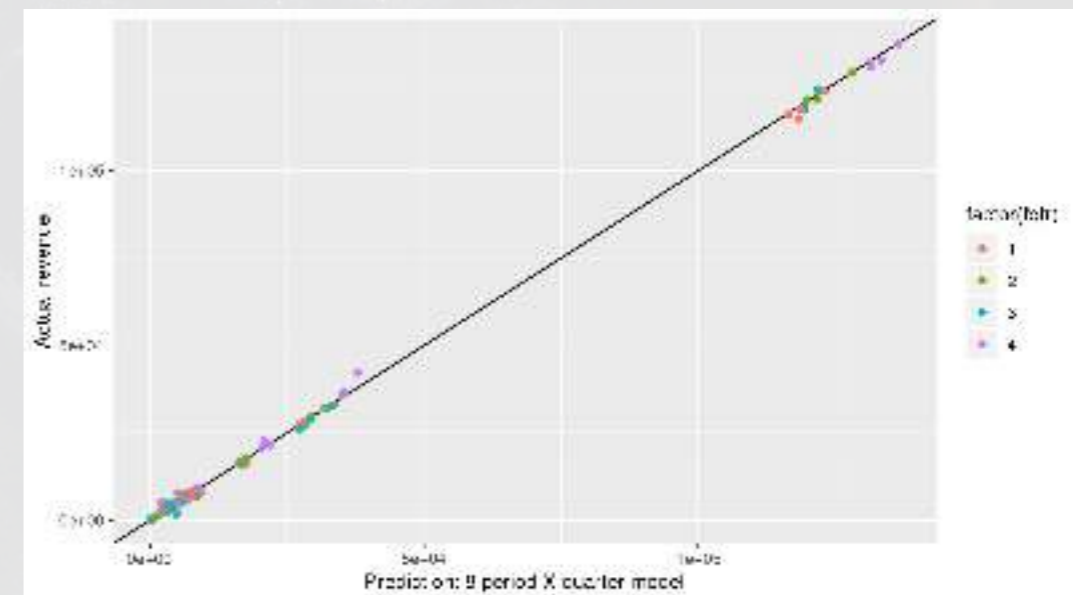
Testing out of sample

	adj_r_sq	rmse_in	mae_in	rmse_out	mae_out
1 period	0.9818514	1151.3535	322.73819	2947.3619	1252.5196
1 and 4 periods	0.9954393	591.9500	156.20811	1400.3841	643.9823
8 periods	0.9985643	345.8053	94.91083	677.6218	340.8236
8 periods w/ quarters	0.9989231	298.9557	91.28056	645.5415	324.9395

1 quarter model



8 period model, by quarter

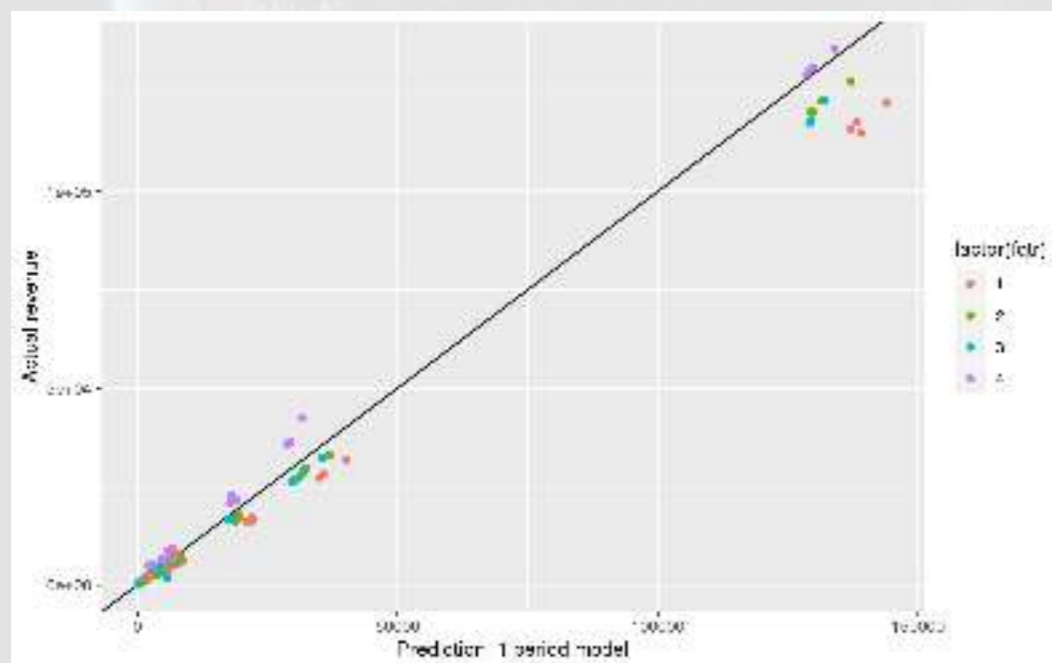


What about for revenue growth?

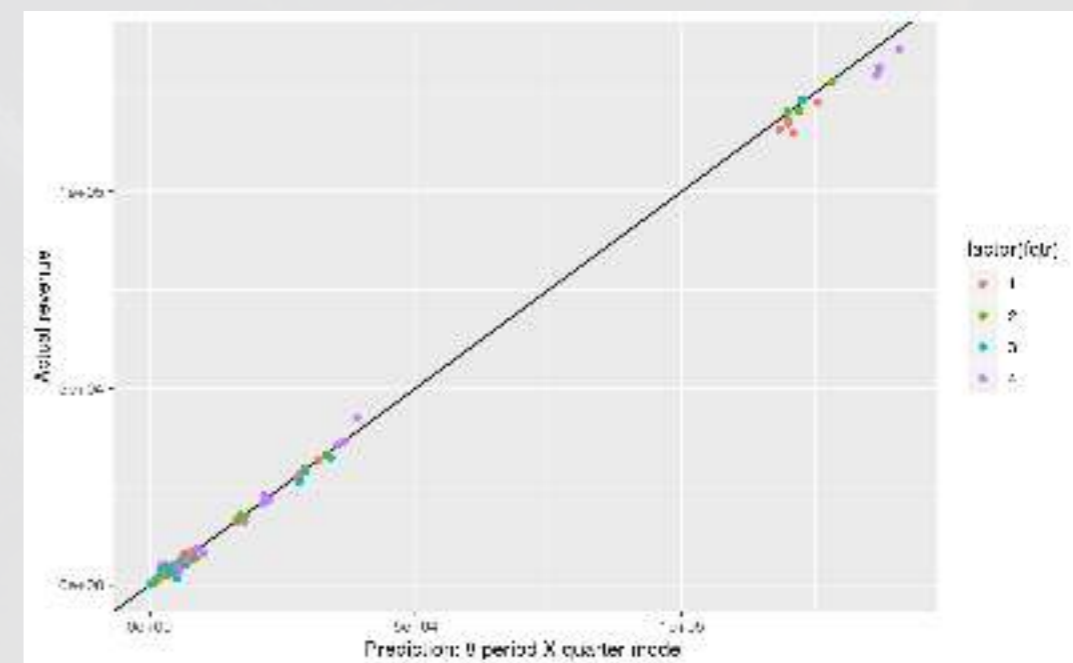
Backing out a revenue prediction, $rev_t = (1 + growth_t) \times rev_{t-1}$

	adj_r_sq	rmse_in	mae_in	rmse_out	mae_out
1 period	0.0910390	1106.3730	308.48331	3374.728	1397.6541
1 and 4 periods	0.4398456	530.6444	154.15086	1447.035	679.3536
8 periods	0.6761666	456.2551	123.34075	1254.201	584.9709
8 periods w/ quarters	0.7758834	378.4082	98.45751	1015.971	436.1522

1 quarter model



8 period model, by quarter

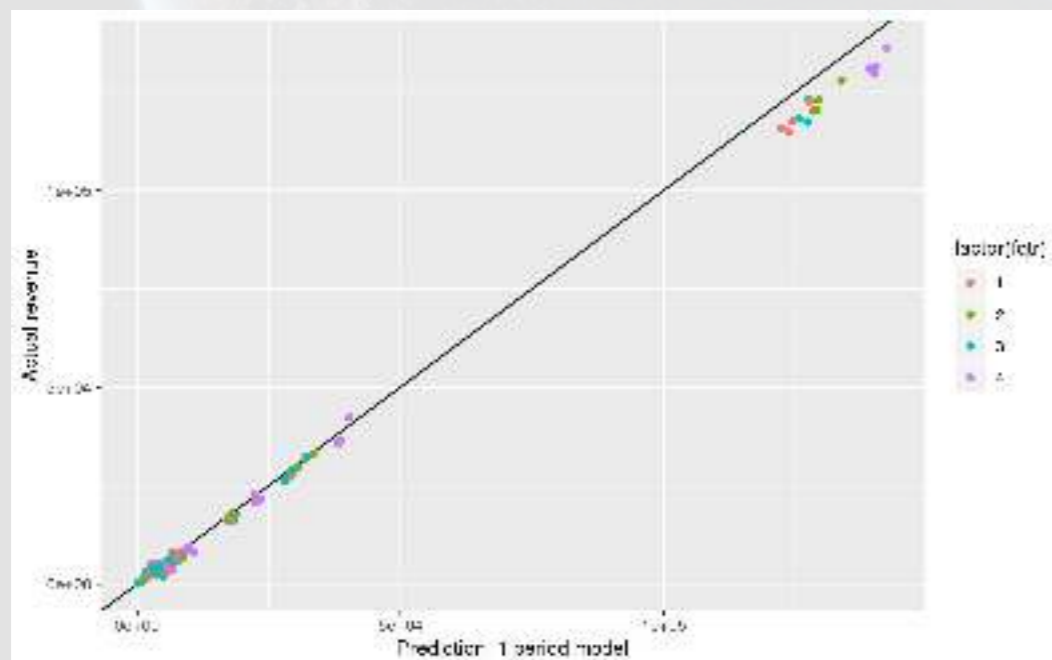


What about for YoY revenue growth?

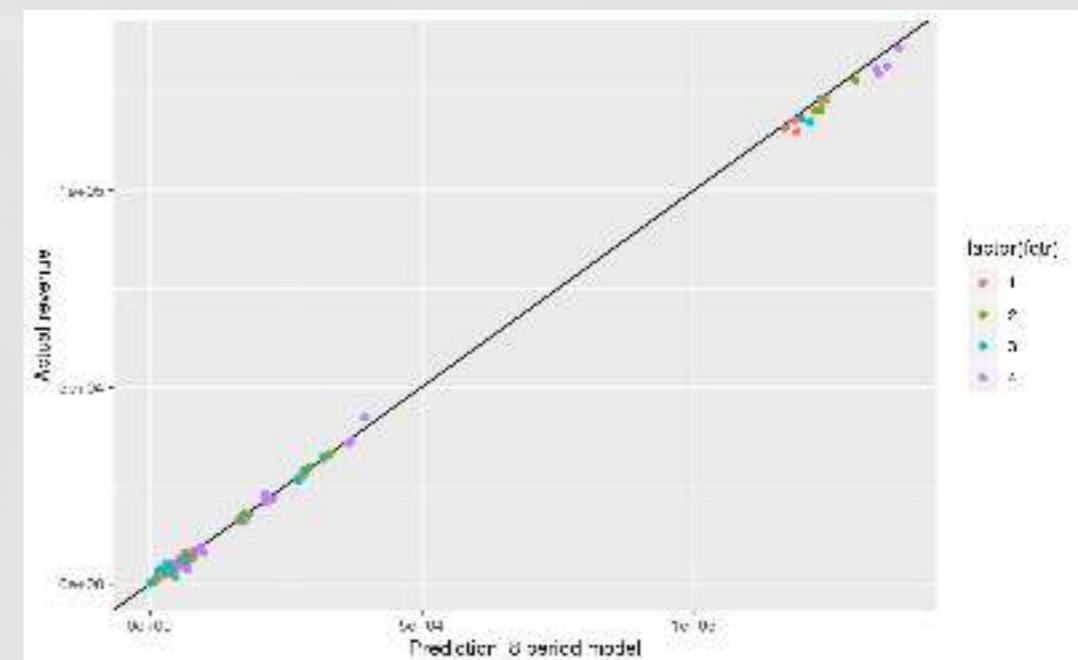
Backing out a revenue prediction,
 $rev_t = (1 + yoy_growth_t) \times rev_{t-4}$

	adj_r_sq	rmse_in	mae_in	rmse_out	mae_out
1 period	0.4370372	513.3264	129.2309	1867.4957	798.0327
1 and 4 periods	0.5392281	487.6441	126.6012	1677.4003	731.2841
8 periods	0.5398870	384.2923	101.0104	822.0065	403.5445
8 periods w/ quarters	0.1563169	714.4285	195.3204	1231.8436	617.2989

1 quarter model



8 period model

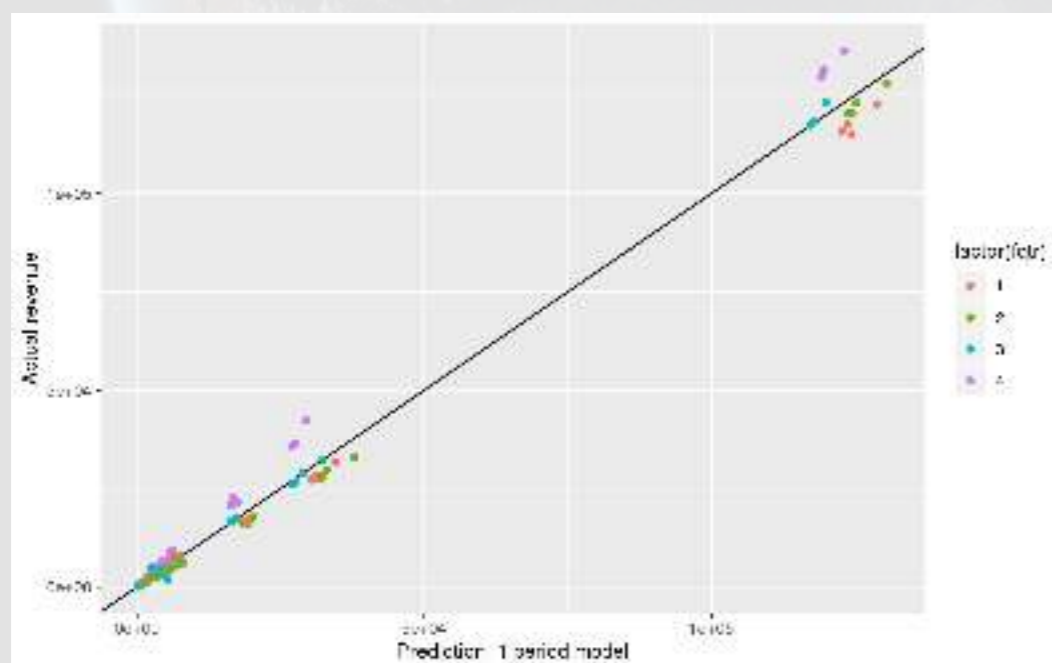


What about for first difference?

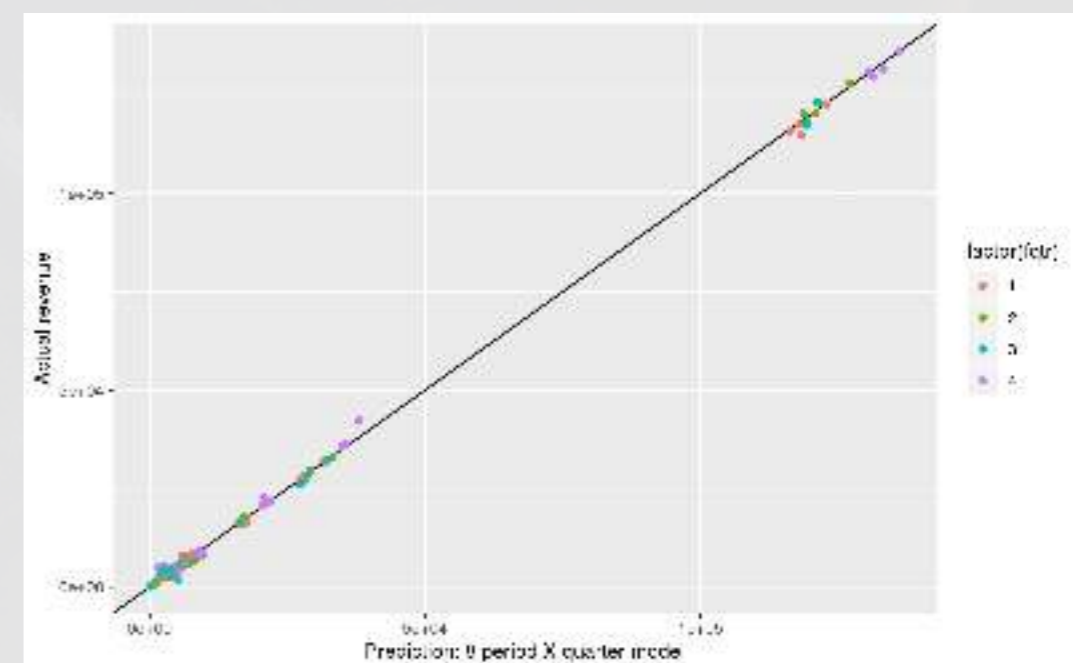
Backing out a revenue prediction, $rev_t = change_t + rev_{t-1}$

	adj_r_sq	rmse_in	mae_in	rmse_out	mae_out
1 period	0.3532044	896.7969	287.77940	2252.7605	1022.0960
1 and 4 periods	0.8425348	454.8651	115.52694	734.8120	377.5281
8 periods	0.9220849	333.0054	95.95924	651.4967	320.0567
8 periods w/ quarters	0.9397434	292.3102	86.95563	659.4412	319.7305

1 quarter model



8 period model, by quarter



Takeaways

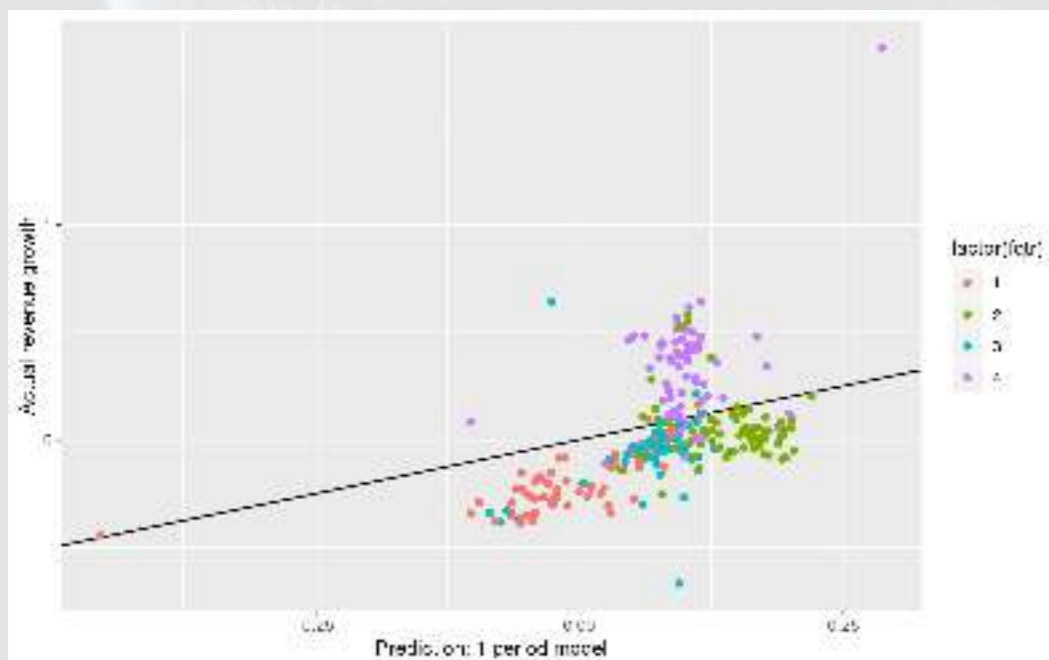
1. The first difference model works about as well as the revenue model at predicting next quarter revenue
 - From earlier, it doesn't suffer (as much) from multicollinearity either
 - This is why time series analysis is often done on first differences
 - Or second differences (difference in differences)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

What about for revenue growth?

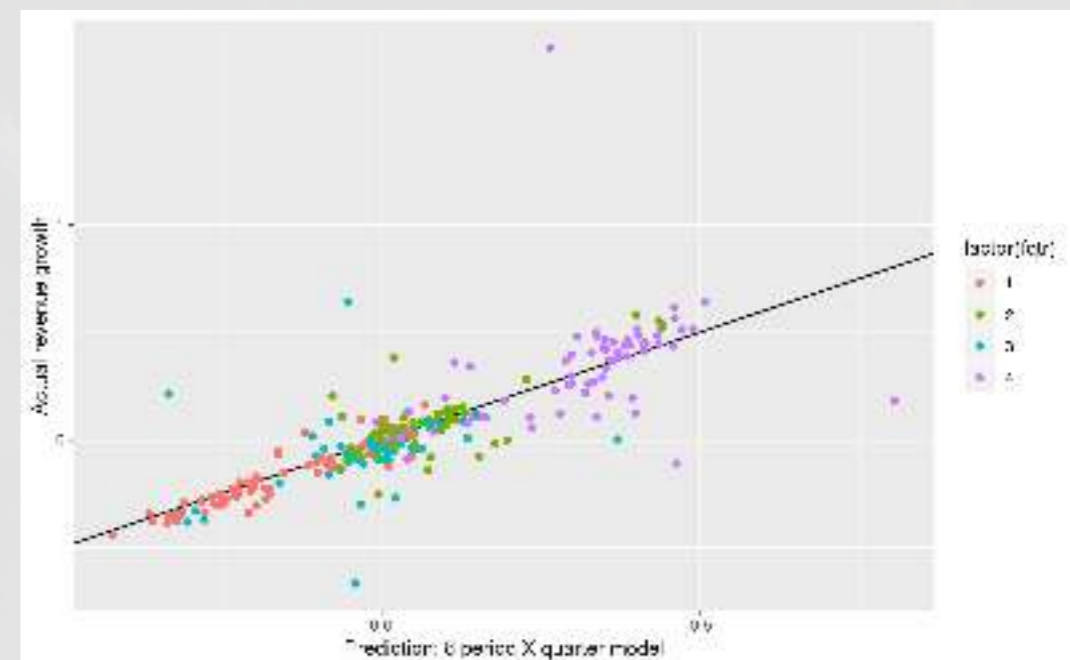
Predicting quarter over quarter revenue growth itself

	adj_r_sq	rmse_in	mae_in	rmse_out	mae_out
1 period	0.0910390	0.3509269	0.2105219	0.2257396	0.1750580
1 and 4 periods	0.4398456	0.2681899	0.1132003	0.1597771	0.0998087
8 periods	0.6761666	0.1761825	0.0867347	0.1545298	0.0845826
8 periods w/ quarters	0.7758834	0.1462979	0.0765792	0.1459460	0.0703554

1 quarter model



8 period model, by quarter

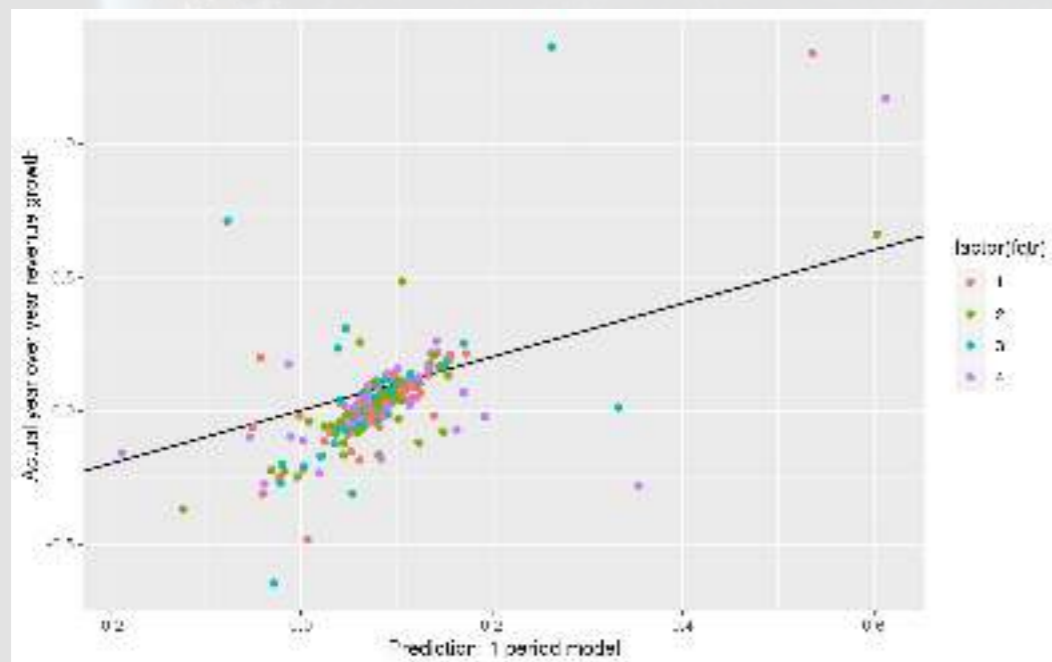


What about for YoY revenue growth?

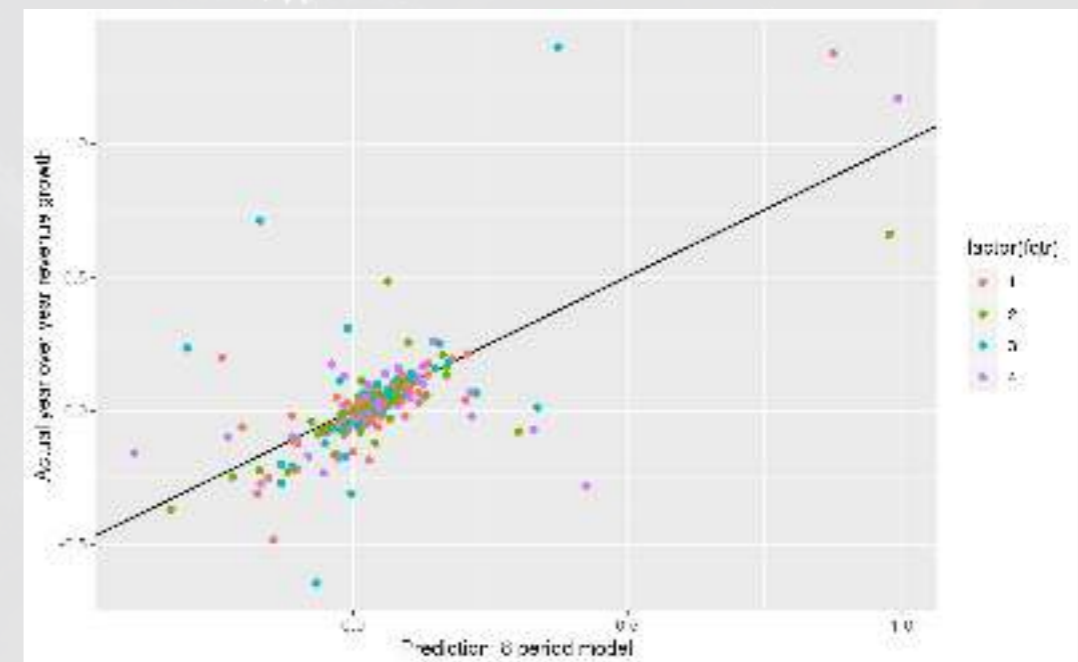
Predicting YoY revenue growth itself

	adj_r_sq	rmse_in	mae_in	rmse_out	mae_out
1 period	0.4370372	0.3116645	0.1114610	0.1515638	0.0942544
1 and 4 periods	0.5392281	0.2451749	0.1015699	0.1498755	0.0896079
8 periods	0.5398870	0.1928940	0.0764447	0.1346238	0.0658011
8 periods w/ quarters	0.1563169	0.3006075	0.1402156	0.1841025	0.0963205

1 quarter model



8 period model

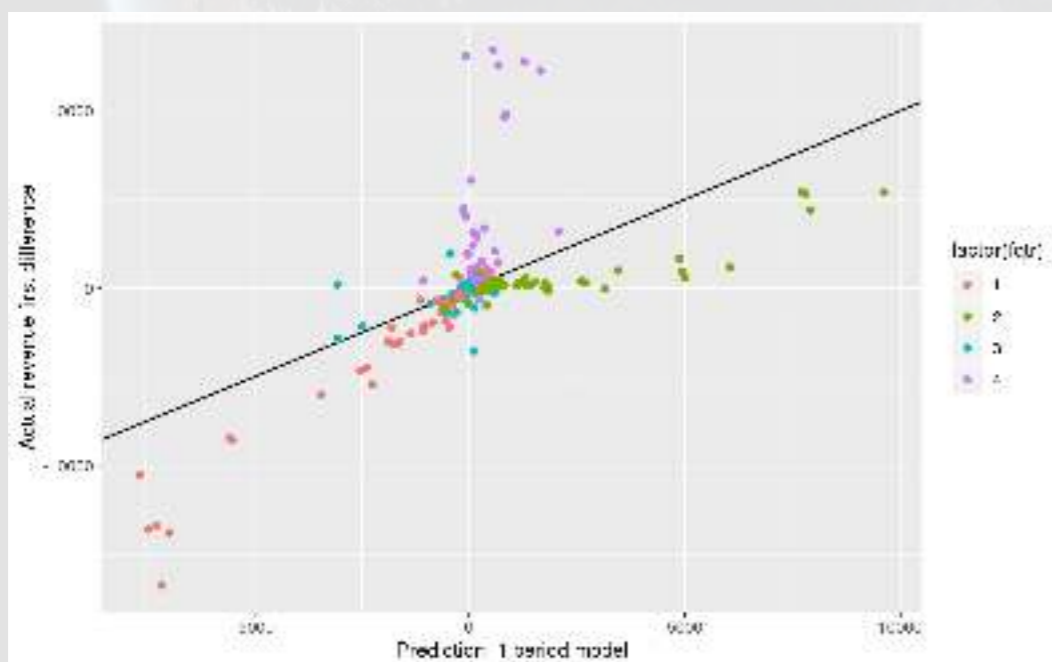


What about for first difference?

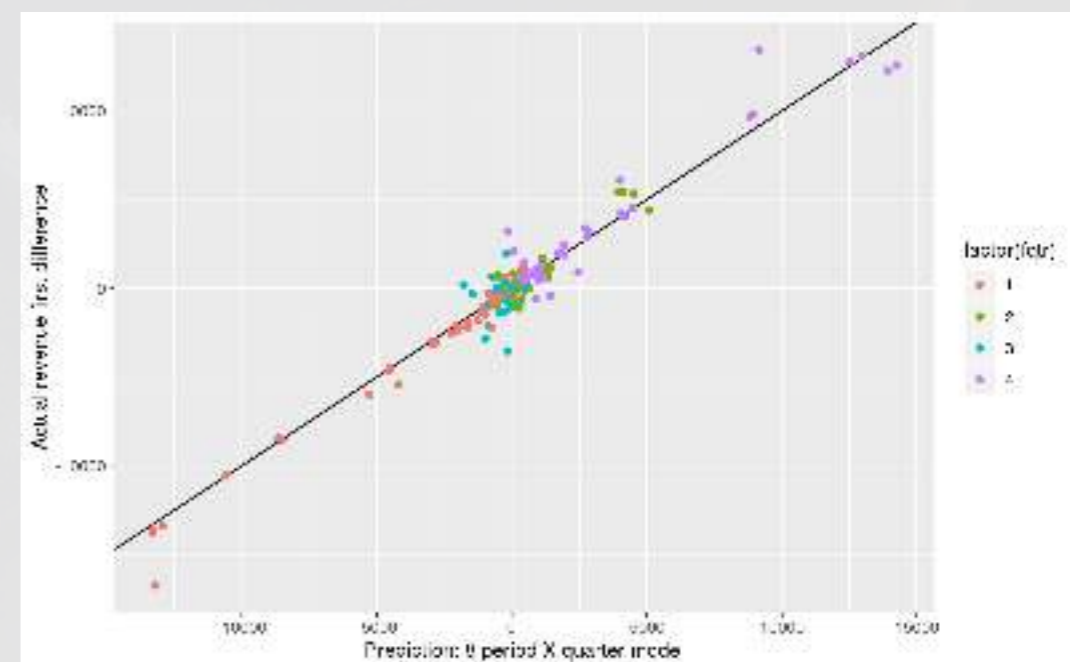
Predicting first difference in revenue itself

	adj_r_sq	rmse_in	mae_in	rmse_out	mae_out
1 period	0.3532044	896.7969	287.77940	2252.7605	1022.0960
1 and 4 periods	0.8425348	454.8651	115.52694	734.8120	377.5281
8 periods	0.9220849	333.0054	95.95924	651.4967	320.0567
8 periods w/ quarters	0.9397434	292.3102	86.95563	659.4412	319.7305

1 quarter model



8 period model, by quarter



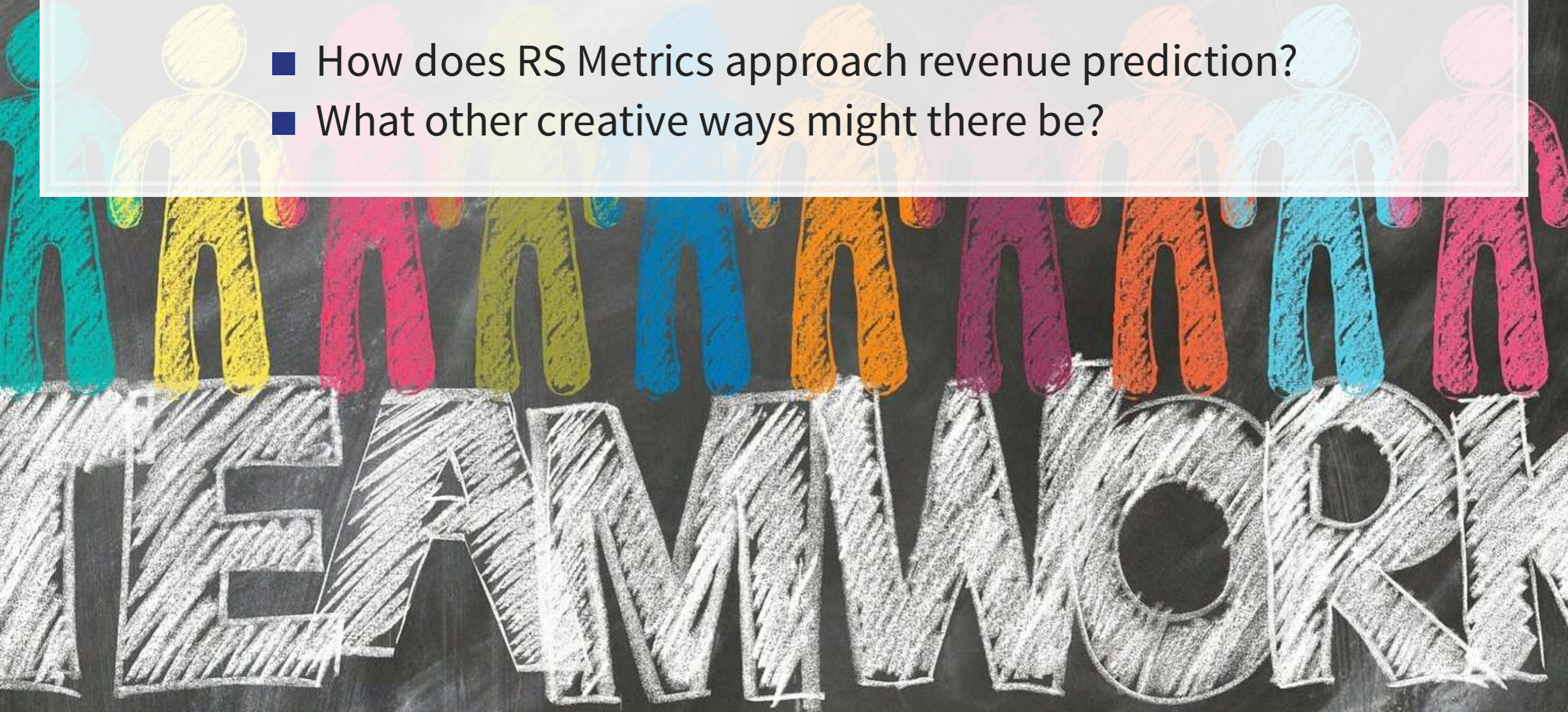
Case: Advanced revenue prediction



RS Metrics' approach

Read the press release: rnc.link/420class4

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



Weekly revenue prediction

Shifted to week 5



End matter



For next week

- For next week:
 - First individual assignment
 - Finish by the end of Thursday
 - Submit on eLearn
 - Datacamp
 - Practice a bit more to keep up to date
 - Using R more will make it more natural

Packages used for these slides

- kableExtra
- knitr
- lubridate
- magrittr
- revealjs
- tidyverse

Custom code

```
# 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 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

```
# These functions are a bit ugly, but can construct many charts quickly
# eval(parse(text=var)) is just a way to convert the string name to a variable reference
# Density plot for 1st to 99th percentile of data
plt_dist <- 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(x=eval(parse(text=var)))) +
    geom_density() + xlab(var)
}
```

```
# 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=fqtr)) +
    geom_bar(stat = "summary", fun.y = "mean") + xlab(var)
}
```

```
# 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(fqtr))) +
    geom_point() + geom_smooth(method = "lm") + ylab(var1) + xlab(var2)
}
```

```
# Calculating various in and out of sample statistics
models <- list(mod1,mod2,mod3,mod4)
model_names <- c("1 period", "1 and 4 period", "8 periods", "8 periods w/ quarters")

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
```