# Feature Engineering I & Variable Selection

Hendrik Orem, Ph.D., with thanks to Jameson Watts

### Agenda

1. Review of Homework 1
2. Feature Engineering I
3. Dinner Break
4. The Caret framework
5. Vocabulary

### Setup

```
knitr::opts_chunk$set(echo = TRUE, message = FALSE, warning = FALSE)
library(tidyverse)
wine = read_rds("../resources/wine.rds")
```

## Basic feature engineering

### Exercise (20m)

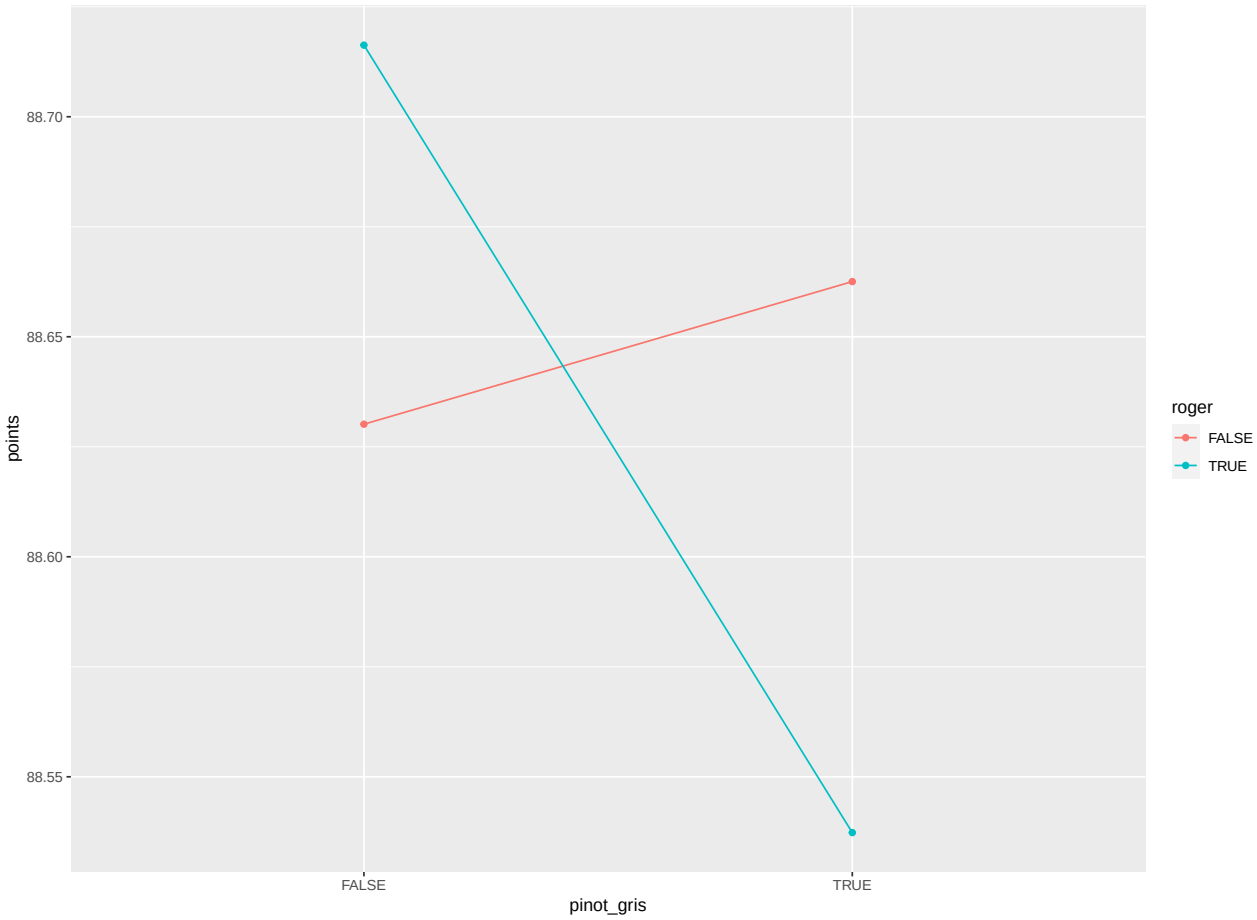
1. Gather in groups
2. Identify 3 "interesting" features of the wine dataset
3. **Bonus** Identify the wine variety (or varieties) that Roger Voss seems to dislike compared to the other critics

### Categorical vs. Continuous Variables

- What is a categorical variable?
- What is a continuous variable?
- Why do we need to "look" at the data before modeling it?

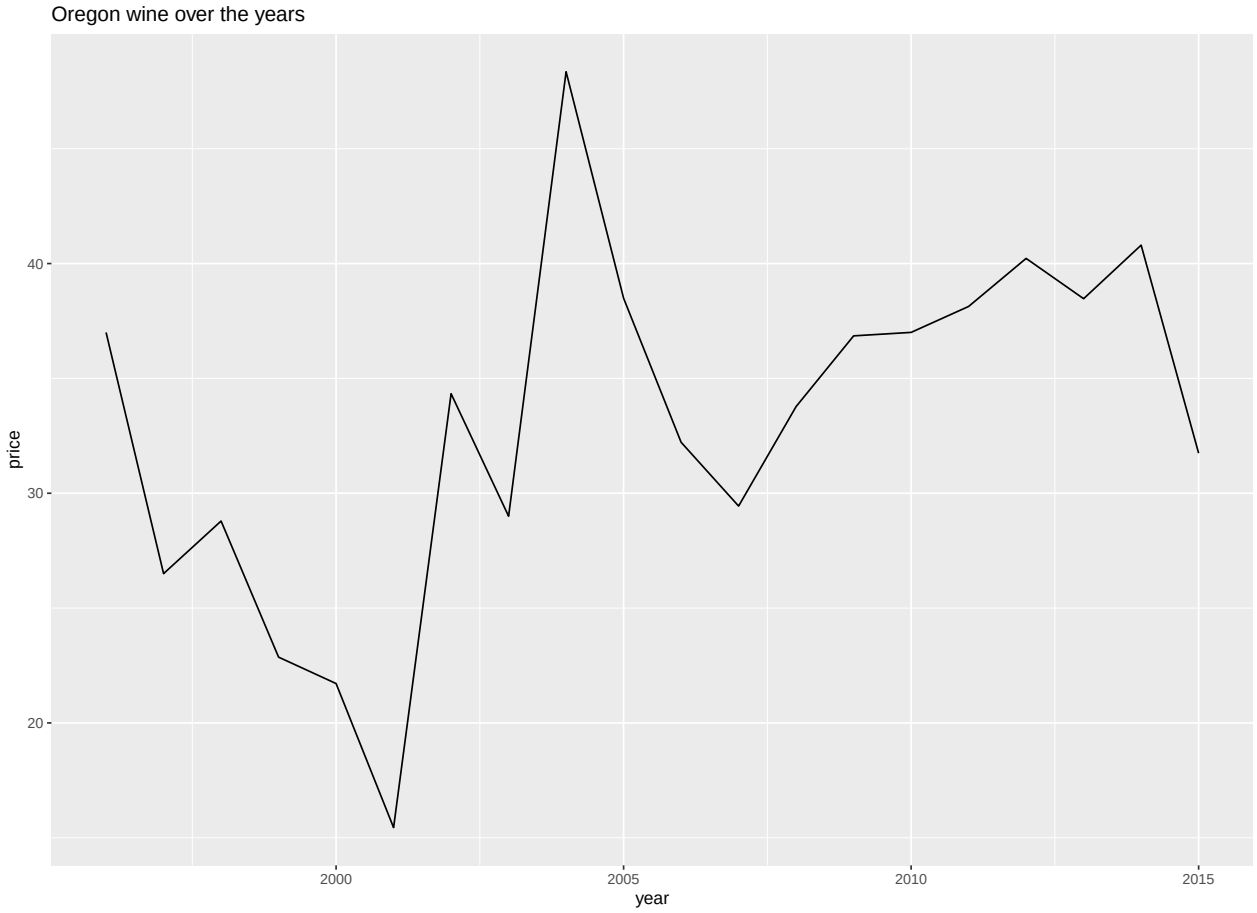
### Categorical Example 1

```
wine %>%
  mutate(roger=taster_name=="Roger Voss") %>%
  mutate(pinot_gris=variety=="Pinot Gris") %>%
  drop_na(roger) %>%
  group_by(roger, pinot_gris) %>%
  summarise(points = mean(points)) %>%
  ggplot() +
  aes(x = pinot_gris, y = points, color = roger) +
  geom_line(aes(group = roger)) +
  geom_point()
```



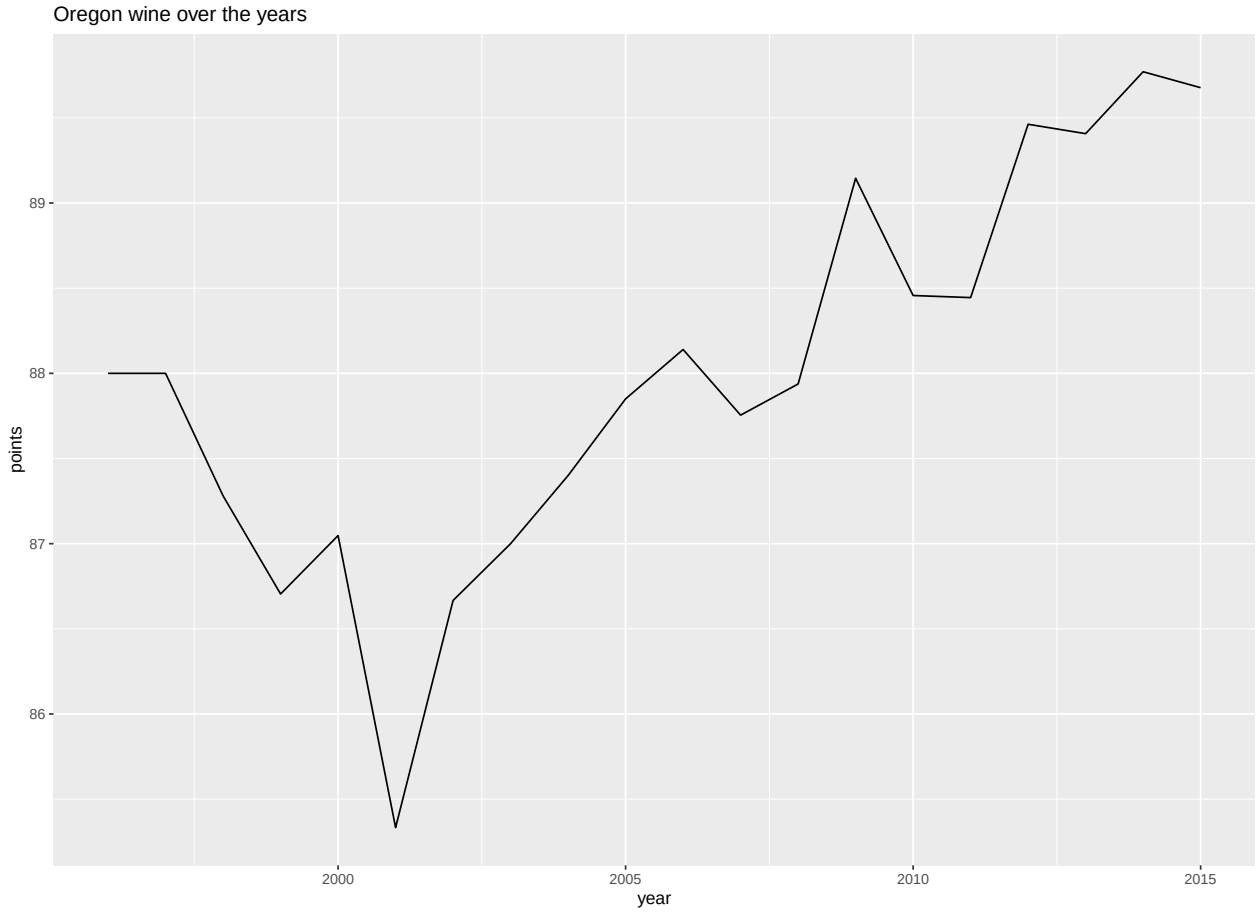

## Categorical Example 2

```
wine %>%
  filter(province=="Oregon") %>%
  group_by(year) %>%
  summarise(price=mean(price)) %>%
  ggplot(aes(year,price))+
  geom_line()+
  labs(title = "Oregon wine over the years")
```




## Categorical Example 2

```
wine %>%
  filter(province=="Oregon") %>%
  group_by(year) %>%
  summarise(points=mean(points)) %>%
  ggplot(aes(year,points))+
  geom_line()+
  labs(title = "Oregon wine over the years")
```




## Exercise (15 min)

1. Group by winery and year to find the average score and number of reviews per winery per year.
2. Calculate the year-on-year change in score for each winery.
3. How might you use this in prediction? What kind of problem might it help with?

## Year-on-Year Change Example

```
wine %>%
  group_by(winery, year) %>%
  summarize(avg_score=mean(points), num_reviews=n_distinct(id)) %>%
  select(year, winery, num_reviews, avg_score) %>%
  arrange(winery, year) %>%
  mutate(score_change = avg_score - lag(avg_score)) %>%
  View()
```

## Encoding categorical features: few dummies

```
library(fastDummies)
wine %>%
  select(taster_name) %>%
  dummy_cols() %>%
  select(1:4) %>%
  head()
```

| taster_name | taster_name_Alexander Peartree | taster_name_Anna Lee C. Iijima | taster_name_Anne Krebiehl MW |
|---|---|---|---|
| Roger Voss | 0 | 0 | 0 |
| Paul Gregutt | 0 | 0 | 0 |
| Alexander Peartree | 1 | 0 | 0 |
| Paul Gregutt | 0 | 0 | 0 |
| Michael Schachner | 0 | 0 | 0 |
| Kerin O'Keefe | 0 | 0 | 0 |

## Encoding categorical features: many dummies

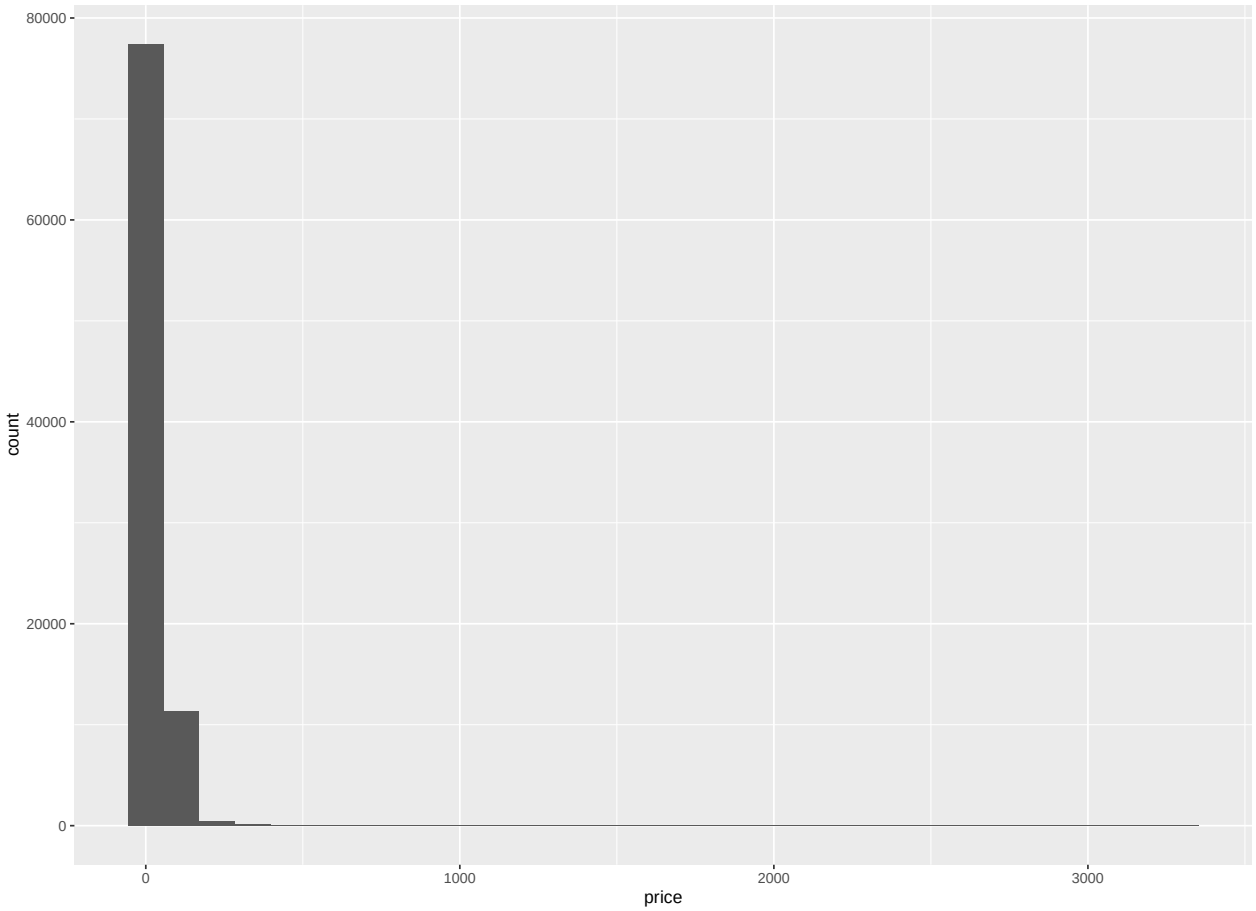
```
wine %>%
  select(variety) %>%
  mutate(variety=fct_lump(variety,4)) %>%
  dummy_cols() %>%
  head()
```

| variety | variety_Cabernet Sauvignon | variety_Chardonnay | variety_Pinot Noir | variety_Red Blend | variety_Other |
|---|---|---|---|---|---|
| Other | 0 | 0 | 0 | 0 | 1 |
| Other | 0 | 0 | 0 | 0 | 1 |
| Other | 0 | 0 | 0 | 0 | 1 |
| Pinot Noir | 0 | 0 | 1 | 0 | 0 |
| Other | 0 | 0 | 0 | 0 | 1 |
| Other | 0 | 0 | 0 | 0 | 1 |

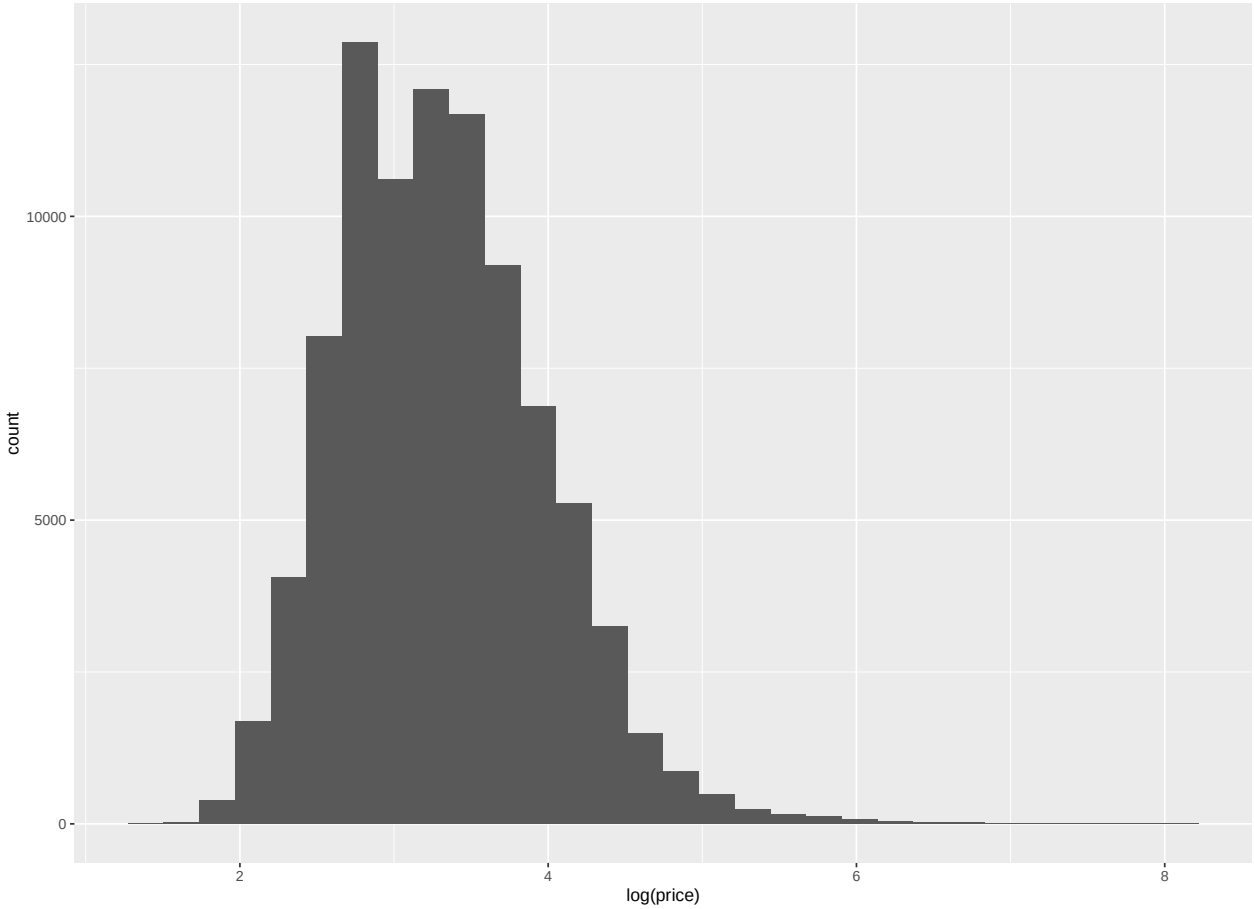
## Other types of engineered categorical features...

- Words or phrases in text
- A given time period
- An arbitrary numerical cut-off
- Demographic variables
- Etc.

## What about numerical features?



## Take the natural log




## Engineering numeric features: Standardizing

- mean-centering $x - \bar{x}$
- scaling: $x/std(x)$

. . . allows for common scale across variables. Also helps reduce bias when interactions are included (i.e. eliminates variance inflation).

And there are many other transformations that you can read about.

A few more to mention:

- YoY, QoQ, etc. (absolute and percent)
- log
- polynomial transforms
- lags!

wine %>% mutate_at("points", list(normalized = ~(scale(.) %>% as.vector))) %>% View()

## Interaction effects

This chapter has a good overview of interactions.

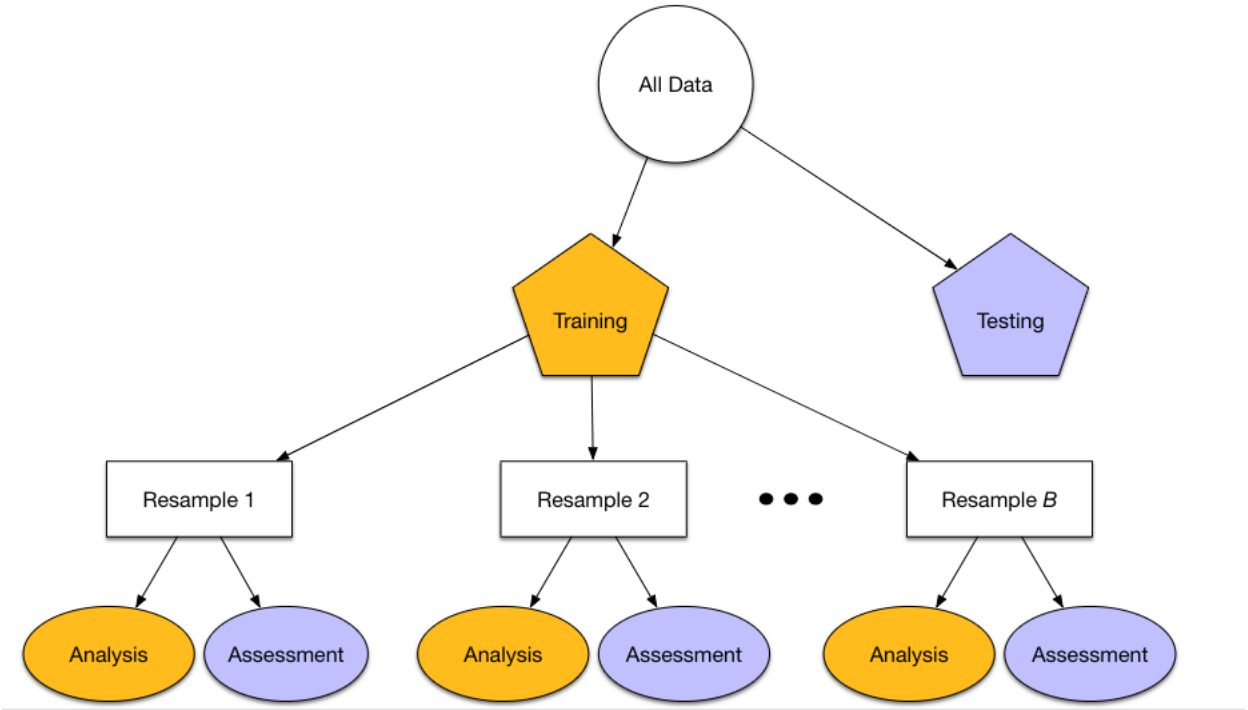
- start with domain knowledge
- use visualizations

- 3-way interactions exist, but are rare and sometimes hard to explain

### Dinner (and virtual high fives)

## The 'caret' package

### Philosophy




### Types of resampling

- V-fold Cross-Validation
- Monte Carlo Cross-Validation
- The Bootstrap

### Typical setup

```
library(caret)
wino <- wine %>% ## look ma, engineered features!
  mutate(fr=(country=="France")) %>%
  mutate(cab=str_detect(variety,"Cabernet")) %>%
  mutate(lprice=log(price)) %>%
  drop_na(fr, cab) %>%
  select(lprice, points, fr, cab)

wine_index <- createDataPartition(wino$lprice, p = 0.8, list = FALSE)
wino_tr <- wino[ wine_index, ]
wino_te <- wino[-wine_index, ]

control <- trainControl(method="repeatedcv", number=5, repeats=3)
m1 <- train(lprice ~ .,
```

```
                data = wino_tr,
                method = "lm",
                trControl = control)
```

Follow this link for the full documentation on caret.

### RMSE outputs

```
print(m1$resample)
```

```
##         RMSE  Rsquared       MAE   Resample
## 1  0.5070809 0.3922505 0.4018716 Fold1.Rep1
## 2  0.5125308 0.3994230 0.4027681 Fold2.Rep1
## 3  0.5151348 0.3859931 0.4041974 Fold3.Rep1
## 4  0.5119503 0.3987446 0.4032244 Fold4.Rep1
## 5  0.5065800 0.3990311 0.4005815 Fold5.Rep1
## 6  0.5109854 0.3872156 0.4030652 Fold1.Rep2
## 7  0.5137343 0.3943372 0.4050433 Fold2.Rep2
## 8  0.5055354 0.4004718 0.3974456 Fold3.Rep2
## 9  0.5072796 0.3984587 0.4000922 Fold4.Rep2
## 10 0.5157122 0.3952686 0.4069972 Fold5.Rep2
## 11 0.5114088 0.3962774 0.4030923 Fold1.Rep3
## 12 0.5079097 0.3921353 0.4003821 Fold2.Rep3
## 13 0.5118545 0.3936632 0.4055349 Fold3.Rep3
## 14 0.5104556 0.3938121 0.4019634 Fold4.Rep3
## 15 0.5116723 0.3997528 0.4016708 Fold5.Rep3
```

### Train vs. test

```
m1
```

```
## Linear Regression
##
## 71603 samples
##     3 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 3 times)
## Summary of sample sizes: 57282, 57282, 57282, 57283, 57283, 57282, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   0.510655  0.3951223  0.4025287
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

### Train vs. test

```
wine_pred <- predict(m1, wino_te)
postResample(pred=wine_pred, obs = wino_te$lprice)
```

```
##      RMSE  Rsquared       MAE
## 0.5109661 0.3970851 0.4017492
```

### Exercise (30-40 minutes)

1. Gather in groups
2. Create 5-10 new features (in addition to points)
3. Create training and test data
4. Use your new predictors to train a linear regression model for log(price)
5. Report RMSE on test set and the cross-validated score.
6. Keep tweaking/engineering new features to lower the RMSE.
7. Should you focus on the CV score or the test score when tweaking to optimize score?
8. Does it make a difference if you use standardized points instead of just points?

## Feature selection

### Stepwise selection is bad

Harrell (2015) provides a comprehensive indictment of the method that can be encapsulated by the statement:

> **"... if this procedure had just been proposed as a statistical method, it would most likely be rejected because it violates every principle of statistical estimation and hypothesis testing."**

Reference: Harrell, F. 2015. Regression Modeling Strategies. Springer.

### Engineer 9 features

```
wino <- wine %>%
  mutate(country=fct_lump(country,4)) %>%
  mutate(variety=fct_lump(variety,4)) %>%
  mutate(lprice=log(price)) %>%
  select(lprice, points, country, variety) %>%
  drop_na(.)

wino <- dummy_cols(wino, remove_selected_columns = T) %>%
  select(-country_Other, -variety_Other) %>%
  rename_all(funs(tolower(.))) %>%
  rename_all(funs(str_replace_all(., "-", "_"))) %>%
  rename_all(funs(str_replace_all(., " ", "_")))

head(wino) %>%
  select(1:7)
```

| lprice | points | country_france | country_italy | country_spain | country_us | variety_cabernet_sauvignon |
|---|---|---|---|---|---|---|
| 2.708050 | 87 | 0 | 0 | 0 | 0 | 0 |
| 2.639057 | 87 | 0 | 0 | 0 | 1 | 0 |
| 2.564949 | 87 | 0 | 0 | 0 | 1 | 0 |
| 4.174387 | 87 | 0 | 0 | 0 | 1 | 0 |
| 2.708050 | 87 | 0 | 0 | 1 | 0 | 0 |
| 2.772589 | 87 | 0 | 1 | 0 | 0 | 0 |

### Basic Model

```
wine_index <- createDataPartition(wino$lprice, p = 0.8, list = FALSE)
wino_tr <- wino[ wine_index, ]
```

```
wino_te <- wino[-wine_index, ]

m2 <- train(lprice ~ .,
                data = wino_tr,
                method = "lm",
                trControl = control)
```

### Results (train)

```
m2
```

```
## Linear Regression
##
## 71603 samples
##     9 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 3 times)
## Summary of sample sizes: 57281, 57283, 57283, 57283, 57282, 57282, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.4886725  0.4468541  0.3796762
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

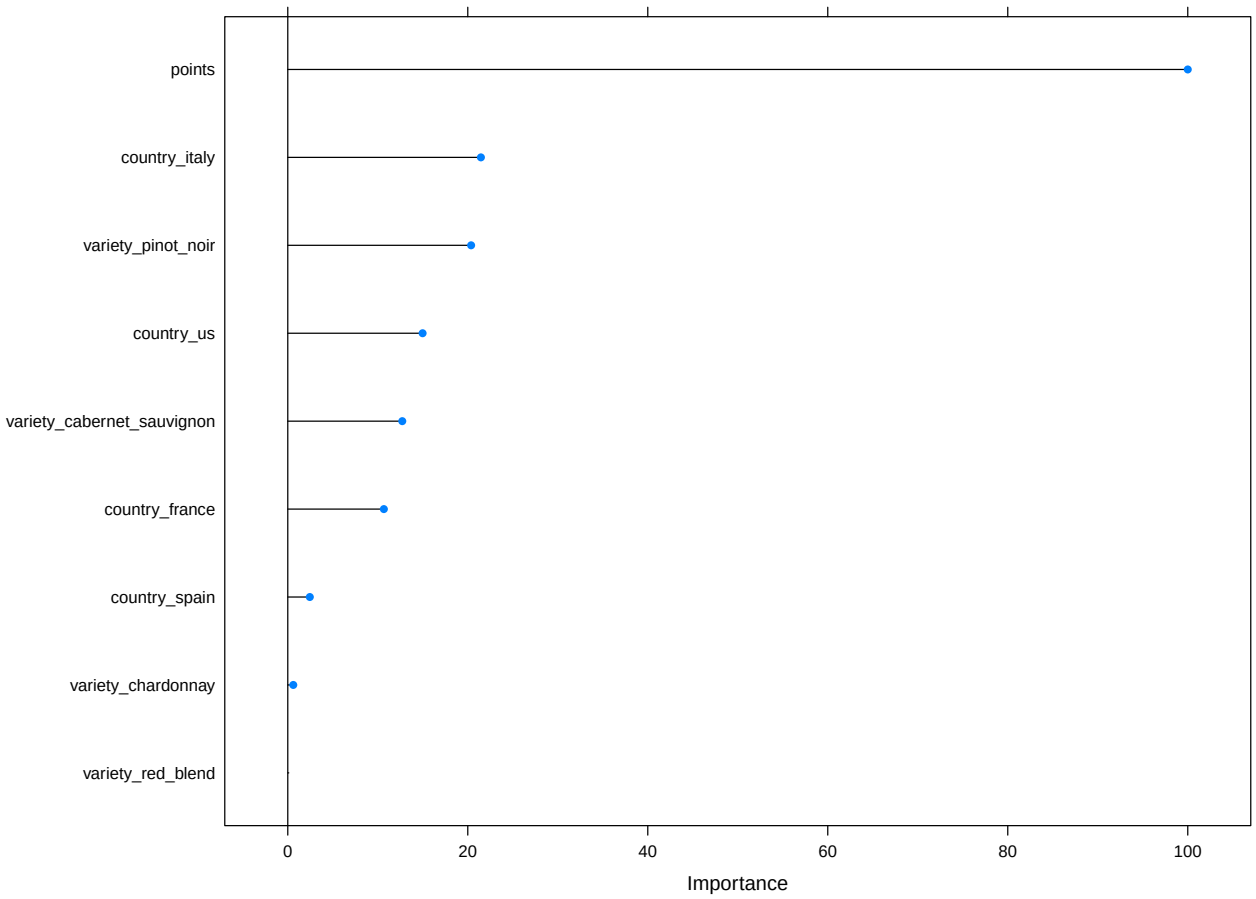
### Results (test)

```
wine_pred <- predict(m2, wino_te)
postResample(pred=wine_pred, obs = wino_te$lprice)
```

```
##      RMSE  Rsquared       MAE
## 0.4892389 0.4442430 0.3811357
```

### Variable Importance (depends on model used)

```
# estimate variable importance
importance <- varImp(m2, scale=TRUE)
# plot importance
plot(importance)
```




## Variable Importance (linear regression)

- Each coefficient in a linear model has a standard error, which is a measure of how certain we are about that coefficient based on the data.
- For the t-statistic, we are asking how confident we are that the coefficient is different from 0: divide the coefficient by the standard error.
- If the standard error is "small" relative to the coefficient, then we have a "big" t-statistic, hence high feature importance.

## Recursive feature elimination

**Algorithm 2:** Recursive feature elimination incorporating resampling

2.1 **for** *Each Resampling Iteration* **do**
2.2 Partition data into training and test/hold–back set via resampling
2.3 Tune/train the model on the training set using all predictors
2.4 Predict the held–back samples
2.5 Calculate variable importance or rankings
2.6 **for** *Each subset size* $S_i$, $i = 1 \ldots S$ **do**
2.7 Keep the $S_i$ most important variables
2.8 [Optional] Pre–process the data
2.9 Tune/train the model on the training set using $S_i$ predictors
2.10 Predict the held–back samples
2.11 [Optional] Recalculate the rankings for each predictor
2.12 **end**
2.13 **end**
2.14 Calculate the performance profile over the $S_i$ using the held–back samples
2.15 Determine the appropriate number of predictors
2.16 Estimate the final list of predictors to keep in the final model
2.17 Fit the final model based on the optimal $S_i$ using the original training set

## Using recursive feature elimination in caret

```
x <- select(wino_tr,-lprice)
y <- wino_tr$lprice

control <- rfeControl(functions=rfFuncs, method="cv", number=2)
# run the RFE algorithm
results <- rfe(x, y, sizes=c(1:3), rfeControl=control)
# summarize the results
print(results)
# list the chosen features
predictors(results)
# plot the results
plot(results, type=c("g", "o"))
```

## Feature selection in practice

1. Raw data —feature engineering—>
2. Lots and lots of features! —feature selection—>
3. Shortlist of features (based on metric) —expert input—>
4. Shortlist of features II —DS judgement—>
5. Finalist models —stakeholders—>
6. Production model

# Vocabulary

## Key Terms

- Feature Engineering
- Categorical Feature
- Continuous Feature
- Dummy
- Interaction
- Caret
- Model
- Resampling
- Training Data vs. Test Data
- Variable Importance

# Linear Regressions again

## 5 Assumptions of Linear Regression

- Linear regressions have a well-developed statistical theory.
- This brings perks like confidence intervals on predictions.
- It also has "costs" in that assumptions need to be satisfied.

## 5 Assumptions of Linear Regression

1. Linearity - the dependent variable is a linear combination of the features.

- This is less of big deal than it might seem! If y is actually quadratic in x, then y is linear in x^2! That's feature engineering.

## 5 Assumptions of Linear Regression

2. Constant variance / homoscedasticity - the variance of the errors do not depend on the values of the features.

- It's important (for linear regressions) that you don't make bigger prediction errors for some values of x than for others.

## 5 Assumptions of Linear Regression

3. Normality - the errors should be independent and normally distributed.

- If you imagine a scatter plot of target variable value and residual (model error), it should look like white noise.

## 5 Assumptions of Linear Regression

4. Lack of perfect multicollinearity - none of your predictors should be a perfect linear combination of the others.

- This can happen if you over-engineer features but in my experience this is less of a concern. You'll see an error that your coefficient matrix is singular or something.

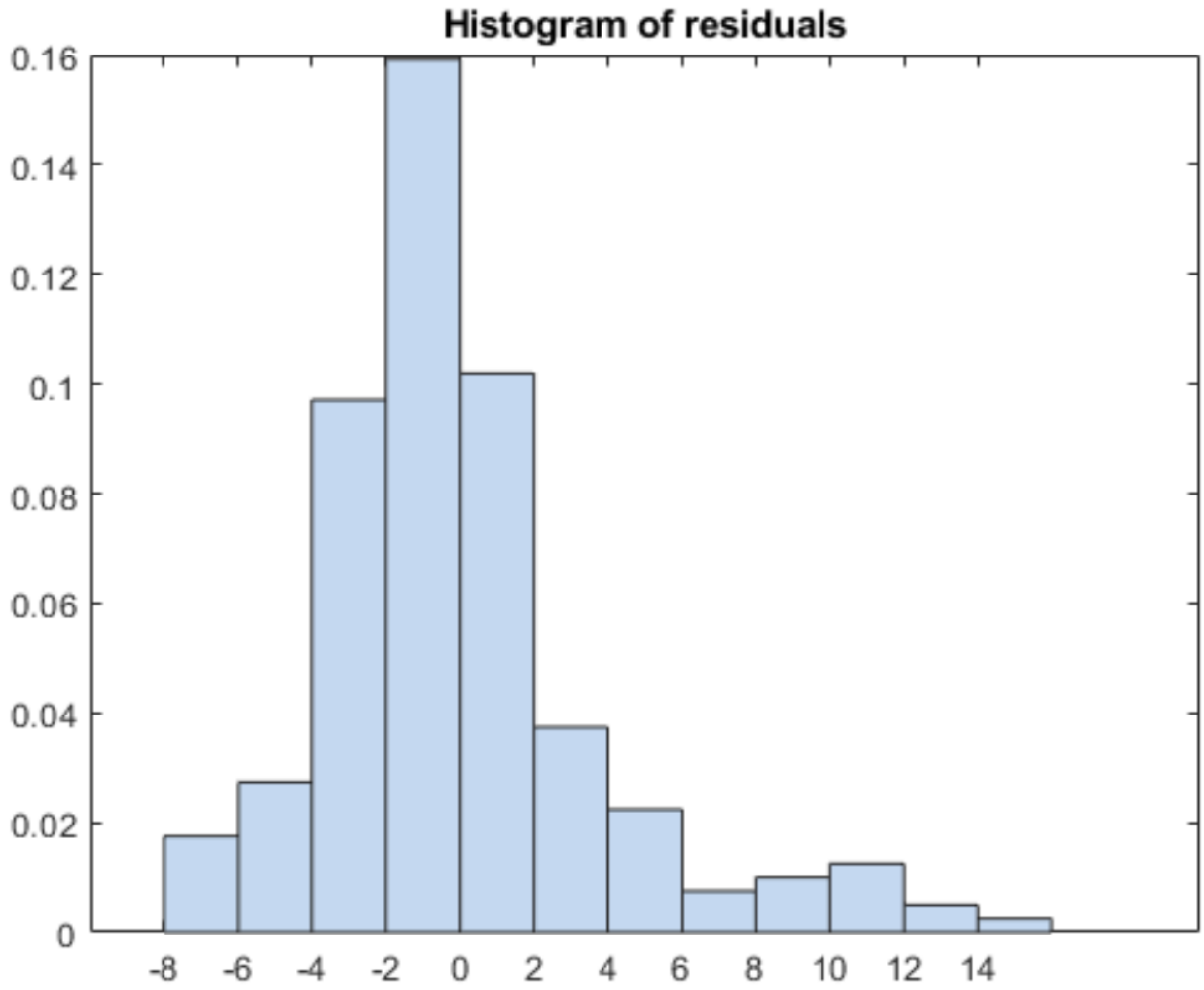
## 5 Assumptions of Linear Regression

5. Exogeneity - model errors should be independent of the values of the features.

- In particular, errors should have mean zero. It's always good to look at a histogram of your residuals (see also normality).

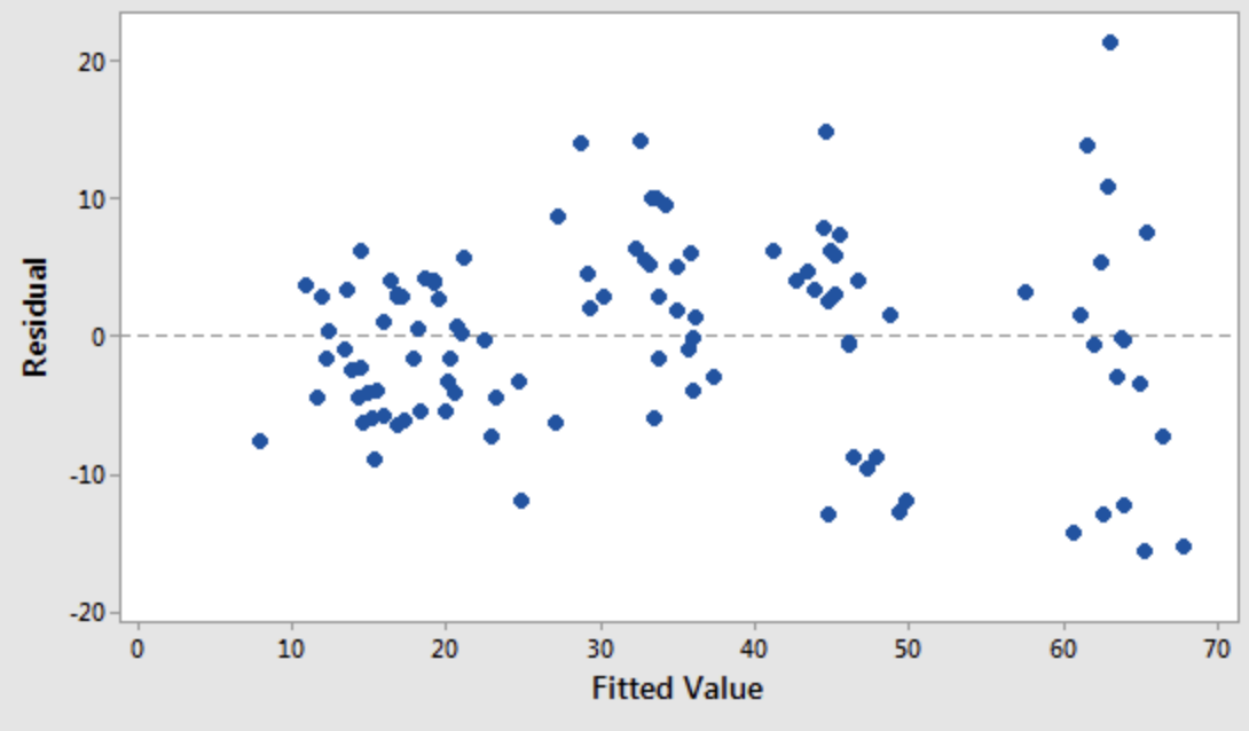
## 5 Assumptions of Linear Regression: testing

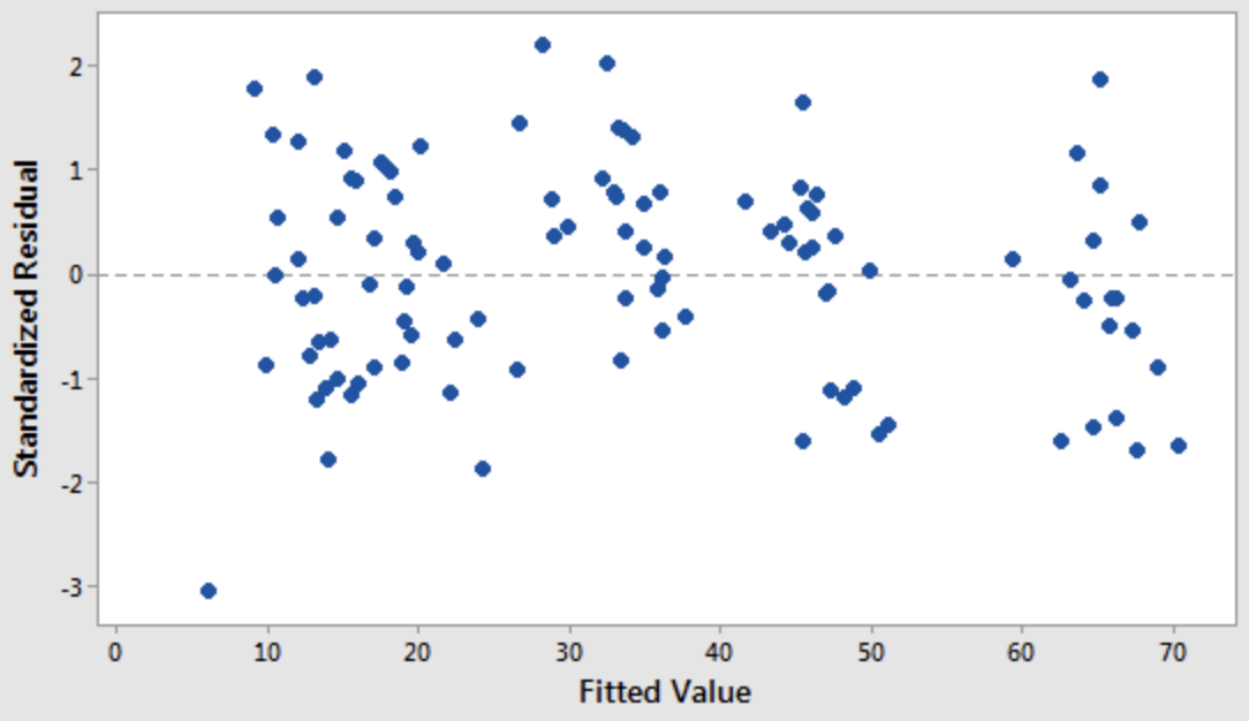
- First and foremost people look at a statistical test to determine whether the errors are normally distributed, like Shapiro-Wilk (also, plot them).




## 5 Assumptions of Linear Regression: testing

- Second I would always look at fitted value vs. residual to check homoscedasticity.




- For more, see for example https://people.duke.edu/~rnau/testing.htm