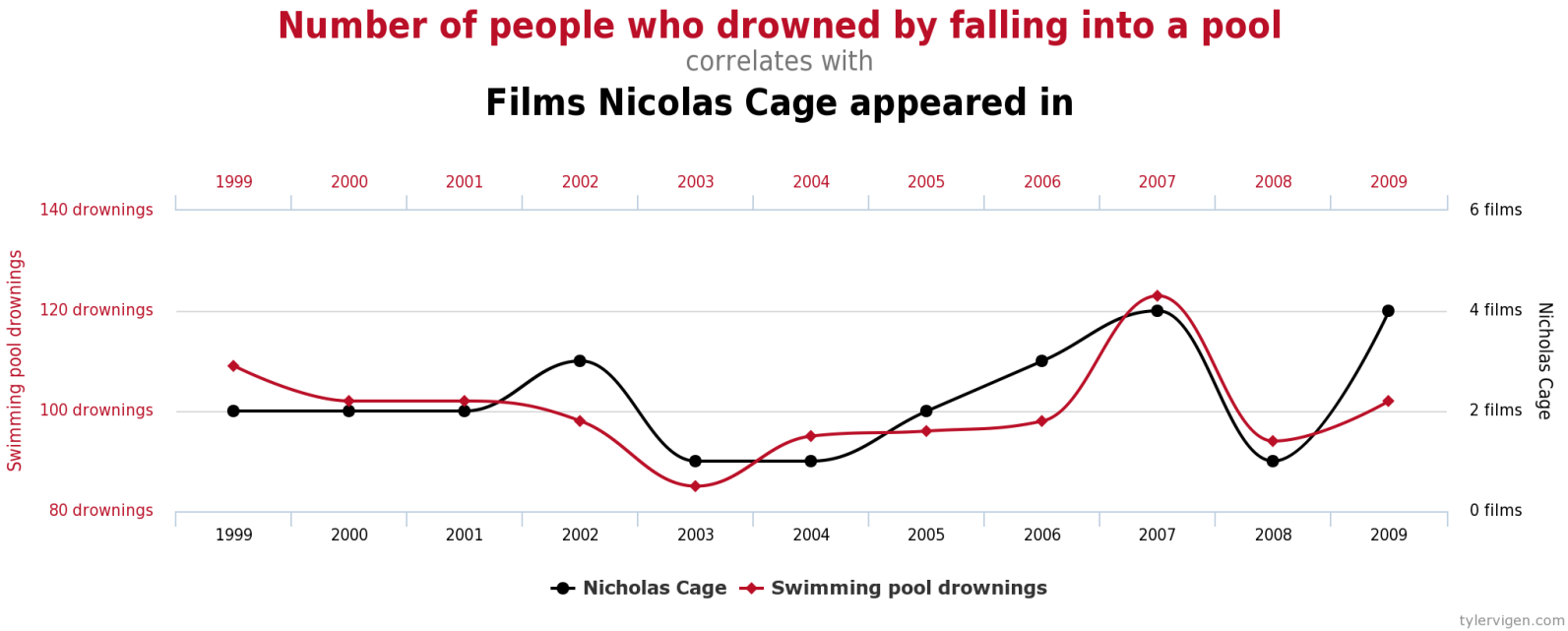
ORIE 3120: Practical Tools for OR, DS, and ML

Causality

Professor Frazier

2022

Correlation is not causation



https://www.tylervigen.com/spurious-correlations

Correlation is not causation

- Suppose we have an outcome Y we want to improve and a covariate X we can change
- We fit the regression $Y = \beta_0 + \beta_1 X + \epsilon$
- The fitted $\hat{\beta}_1$ is non-zero with a p-value < 0.05
- This shows Y and X are *correlated*
- It also shows that knowing X can help to *predict* Y
- This does *not* necessarily mean that changing X will change Y, i.e., it does not mean that Y and X have a *causal* relationship

Sources of correlation other than causation

- overfitting
- hidden confounders
- unclear directonality

Outline

Unclear Directionality

Hidden Confounders

Overfitting

- When there is a causal relationship between Y and X, it isn't always clear the direction
- Examples:
  - Incarceration and income
  - Depression and alcoholism
  - Homelessness and unemployment

Outline

Unclear Directionality

Hidden Confounders

Overfitting

Hidden Confounders

- Y is the outcome
- X is the predictor of interest
- Z is a "hidden confounder" — a random variable that influences both X and Y
- (X and Z can be vectors, but for now we'll consider them to be scalars)

Hidden Confounder Example

- Y = health (e.g., life expectancy)
- X = organic food consumption
- Z = income (or health consciousness, age, etc.)

- Suppose we fit the regression $Y = \beta_0 + \beta_1 X + \epsilon$,
- Suppose changing X would cause no change in Y. The regression may still find a stat-sig non-zero $\hat{\beta}_1$
- This is because Z causes Y to be correlated with X
- If there is a causal impact of X on Y, $\hat{\beta}_1$ will mis-estimate its magnitude

Demo: causality.ipynb (Hidden Confounders: Examples 1 and 2)

Two Solutions

- Include the confounder(s) Z into the regression
- Run a randomized controlled trial (also called an A/B test)

Demo: causality.ipynb (Hidden Confounders: Solutions)

Solutions: Pros and Cons

- Include the confounder(s) Z into the regression
  - Pro: Don't need new data
  - Con: You never know for sure if the answer is right because (1) it's hard to identify all of the confounders; (2) including a confounder but not fitting a flexible enough model can also mis-estimate the causal impact.
- Run a randomized controlled trial (also called an A/B test)
  - Pro: Estimates of causal impacts are correct (up to statistical error, which can be quantified)
  - Con: Expensive, might be impossible (e.g., if running the experiment would be unethical)

Including all confounders is hard

Regular consumers of organic food are most likely to be **female**, **health-conscious**, **physically active**, and in the higher brackets of **education** and **income** than their non-organic consuming counterparts [7,8]. They are also more likely to have a **higher ratio of plant to animal foods** . . . [and] an **increased wholefood dietary intake**, . . .

Diet composition between organic and non-organic consumers may, therefore, be quite different.

Vigar et al. 2020 "A Systematic Review of Organic Versus Conventional Food Consumption: Is There a Measurable Benefit on Human Health?"

Natural Experiments

- Smoking ban in Helena MT. Outcome: heart attacks. Sargent et al. 2005
- Vietnam War draft. Outcome: lifetime earnings. Angrist 1990
- Computer glitch hiding the surge heatmap for Uber drivers on Android. Outcome: movement toward high-surge areas Lu et al. 2018

Outline

Unclear Directionality

Hidden Confounders

Overfitting

Overfitting

- Suppose the number of predictors (p) is almost as large as the number of datapoints (n)
- There are many combinations of coefficients $\beta_0, \ldots, \beta_p$
- When n is small, one of these combinations can produce $\hat{Y}$ that correlates well with Y just by chance
- This is called "overfitting"
- When we have overfit, changing the X_j to increase $\hat{Y}$ will change Y by much less, or by nothing at all
- Also, if we gather new data, our predictions will be much worse than they were on the training data

Overfitting Example

```
# Generate n data points, each with p predictors
df = pd.DataFrame()
n = 150
p = 100
for j in range(p):
    xp = 'x'+str(j)
    df[xp] = np.random.randn(n)
df = sm.add_constant(df)

# y does not depend on the predictors
y = np.random.randn(n)
```

Overfitting Example

	const	x0	x1	x2	x3	x4	x5	x6	x7	x8	...
0	1.0	1.949386	-0.235541	0.561186	1.291321	1.152173	2.339091	-0.598028	-1.956971	-0.620413	...
1	1.0	-0.734335	0.425791	-1.694647	0.087672	0.141258	-0.880642	-0.350737	-0.606976	-0.578668	...
2	1.0	2.482734	0.292133	0.644090	-0.254376	2.238506	0.273161	0.969626	-0.749776	0.335687	...
3	1.0	-0.742045	0.478984	-0.730207	-1.558961	-0.056479	-0.085716	-0.705976	0.752218	-0.493057	...
4	1.0	-0.042689	-1.911862	-0.235362	0.957435	0.957367	-0.124206	0.020149	-0.526945	0.559360	...

5 rows × 101 columns

Overfitting Example

```
# Fit linear regression
model = sm.OLS(y,df).fit()
model.summary()
```

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.711
Model:	OLS	**Adj. R-squared:**	0.120
Method:	Least Squares	**F-statistic:**	1.203
Date:	Thu, 14 Apr 2022	**Prob (F-statistic):**	0.238
Time:	22:32:22	**Log-Likelihood:**	-114.88
No. Observations:	150	**AIC:**	431.8
Df Residuals:	49	**BIC:**	735.8
Df Model:	100		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.3030	0.135	2.237	0.030	0.031	0.575
x0	-0.0048	0.132	-0.036	0.971	-0.269	0.260
x1	0.0662	0.140	0.472	0.639	-0.215	0.348
x2	0.0616	0.137	0.451	0.654	-0.213	0.336
x3	-0.0228	0.119	-0.191	0.849	-0.262	0.217
x4	-0.1373	0.126	-1.087	0.282	-0.391	0.116
x5	0.0008	0.145	0.006	0.995	-0.291	0.292
x6	-0.3196	0.163	-1.965	0.055	-0.646	0.007
x7	0.0251	0.122	0.205	0.838	-0.220	0.270
x8	-0.2568	0.137	-1.868	0.068	-0.533	0.019
x9	-0.1760	0.140	-1.255	0.215	-0.458	0.106
x10	0.1783	0.127	1.402	0.167	-0.077	0.434

Overfitting Example

- The R^2 is 0.7
- There are 6 features with statistically significant effects on the outcome (i.e., with p-values below 0.05)

```
model.rsquared
```

```
0.710649460411463
```

```
model.pvalues[model.pvalues<0.05]
```

```
const    0.029858
x18      0.003734
x22      0.009725
x30      0.028338
x36      0.025108
x80      0.017284
x84      0.009233
dtype: float64
```

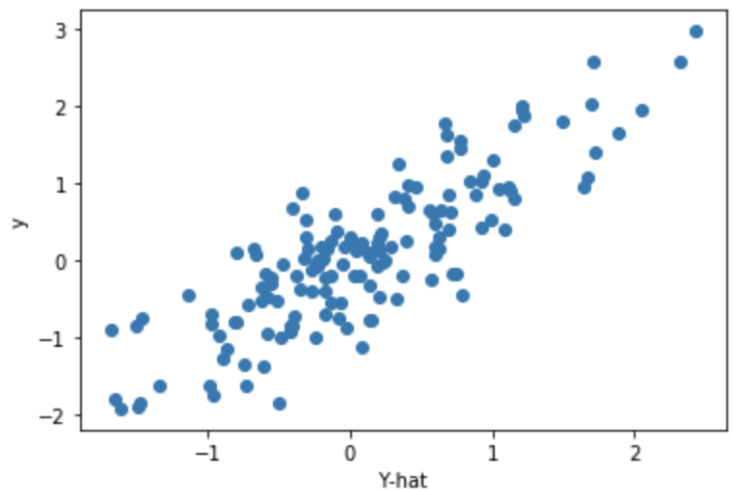
- It *looks* like the covariates predict the outcomes
- But the code generating the outcomes ignored the covariates

Overfitting Example

- The outcomes (Y_i) are correlated with our predictions ($\hat{Y}_i$) in the training data

```
plt.scatter(model.predict(df),y)
plt.ylabel('y')
plt.xlabel('Y-hat')
```

```
Text(0.5, 0, 'Y-hat')
```



- Again, it *looks* like the covariates predict the outcomes
- But the code generating the outcomes ignored the covariates

Overfitting Solution

- Split dataset randomly into training and test
- Train on the training set
- Evaluate the model and the correlation of $\hat{Y}$ with Y on the test set

Overfitting Solution, in the Example

Split data into training and test

```
X_train, X_test, y_train, y_test = train_test_split(
...      df, y, test_size=0.5, random_state=1)
```

Train on the training data only

```
model = sm.OLS(y_train,X_train).fit()
```

Overfitting Solution, in the Example

- The (overfitted) model performs badly on the test data, evaluated via sum of squares
- The predictive model fitted on the training data has a larger sum of squares than a simple prediction of 0

```
np.sum((model.predict(X_test)-y_test)**2) # Sum of squares
```

```
242.45401938254386
```

```
np.sum(y_test**2) # Sum of squares for predicting 0
```

```
77.35967704914083
```

Overfitting Solution, in the Example

- The (overfitted) model performs badly on the test data, when evaluated via a scatterplot
- The scatterplot shows the prediction and outcome have low correlation, on the test data