MODELOS Y SIMULACION

En la clase de hoy entrenaremos modelos de regresion lineal (simple y multiple) en base a lo visto en la clase del 17 de Agosto. Pero antes repasaremos algunas cuestiones logísticas:

- Los horarios de Office Hours se pueden reservar por aqui (https://calendly.com/tomas-nozica/15min)
- El libro de esta materia es An Introduction to Statistical Learning (Editorial Springer)
- Al finalizar esta clase recibiran la practica de esta semana. No se entregan pero recomendamos que las hagan y toda consulta es bienvenida.

Conocer el entorno: Correlacion y Causalidad

Antes de comenzar con la clase de hoy vamos a hablar de "Correlacion Espuria" -Spurious Correlation en ingles-

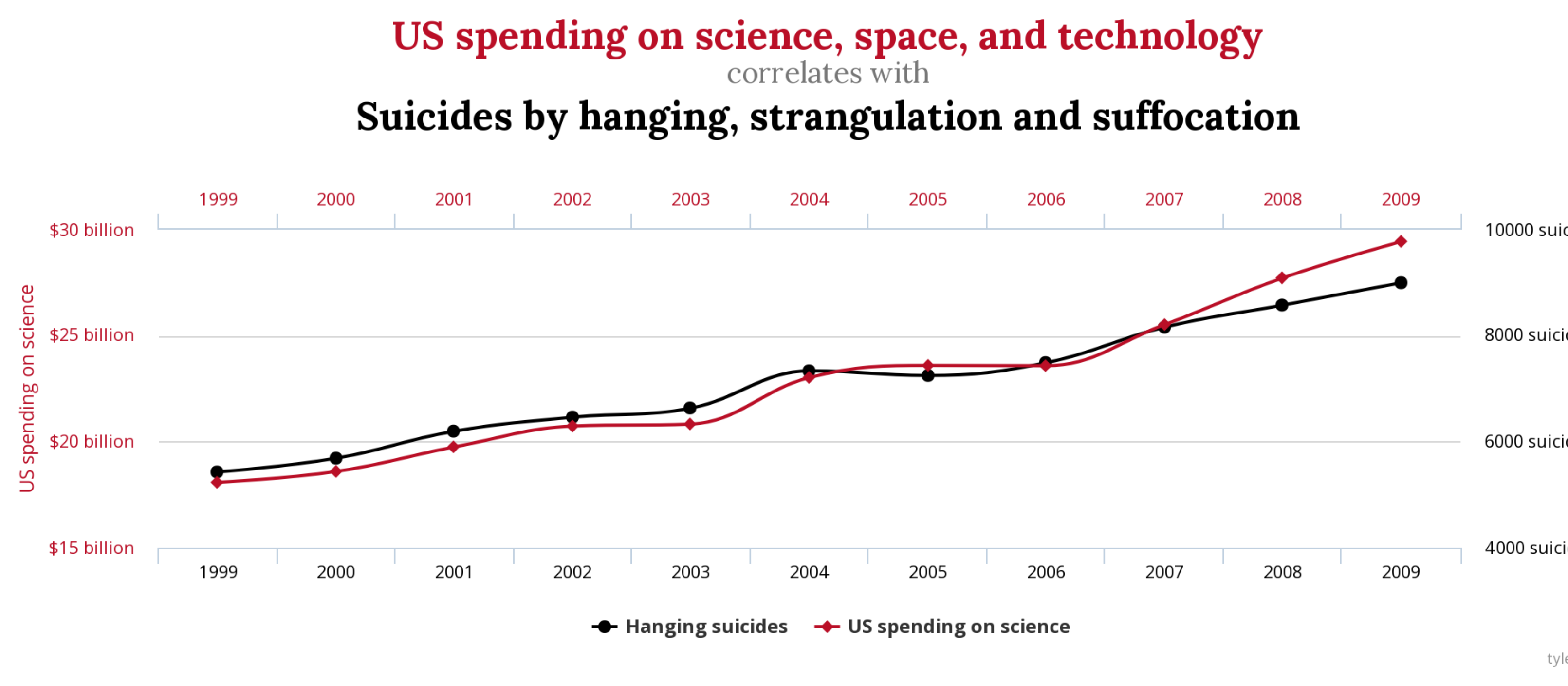
¿Qué significa Spurious Correlation?

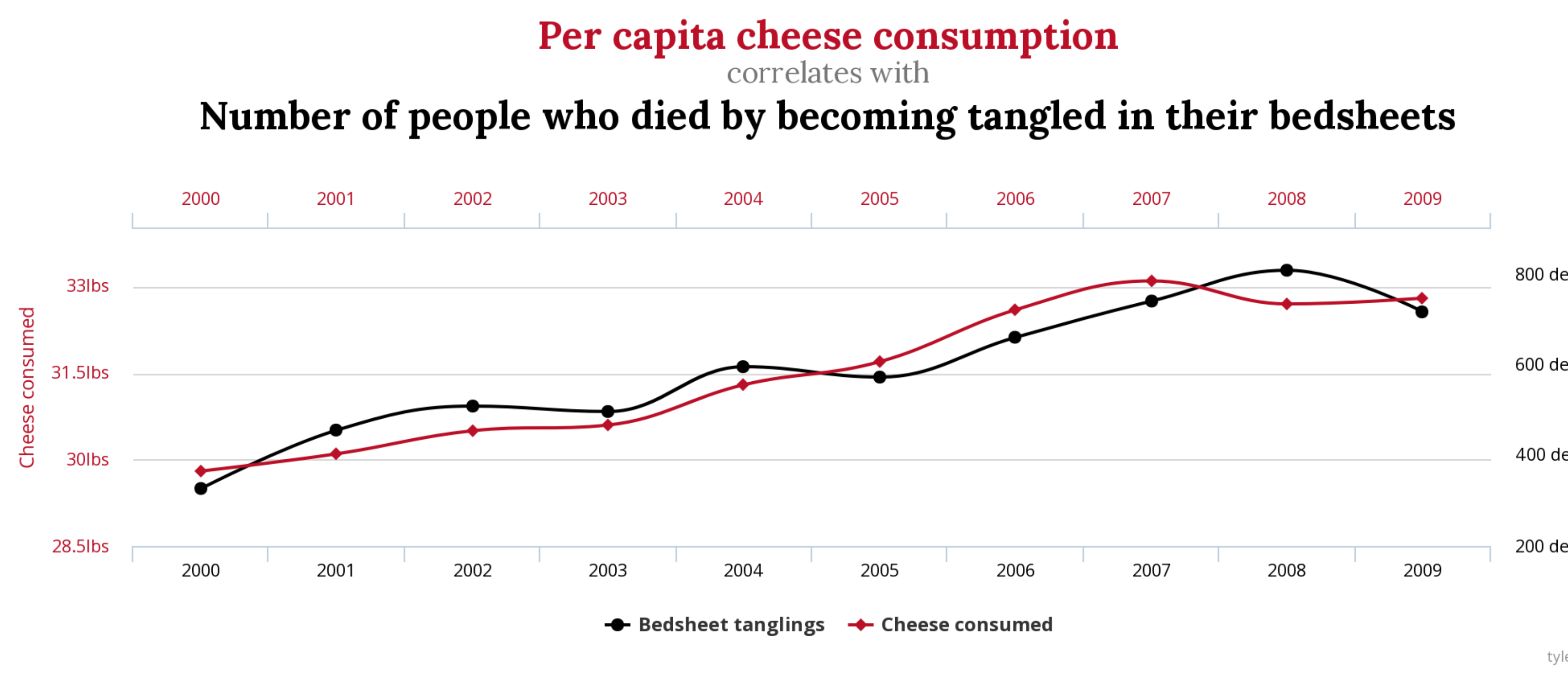
Segun Yashvi Patel en un articulo [1] publicado en Kaggle, en estadística, la correlacion espuria refiere a la conexion entre dos variables que aparentan causalidad cuando en realidad no es asi. Las relaciones espurias frecuentemente dan la apariencia que una variable afecta a otra.

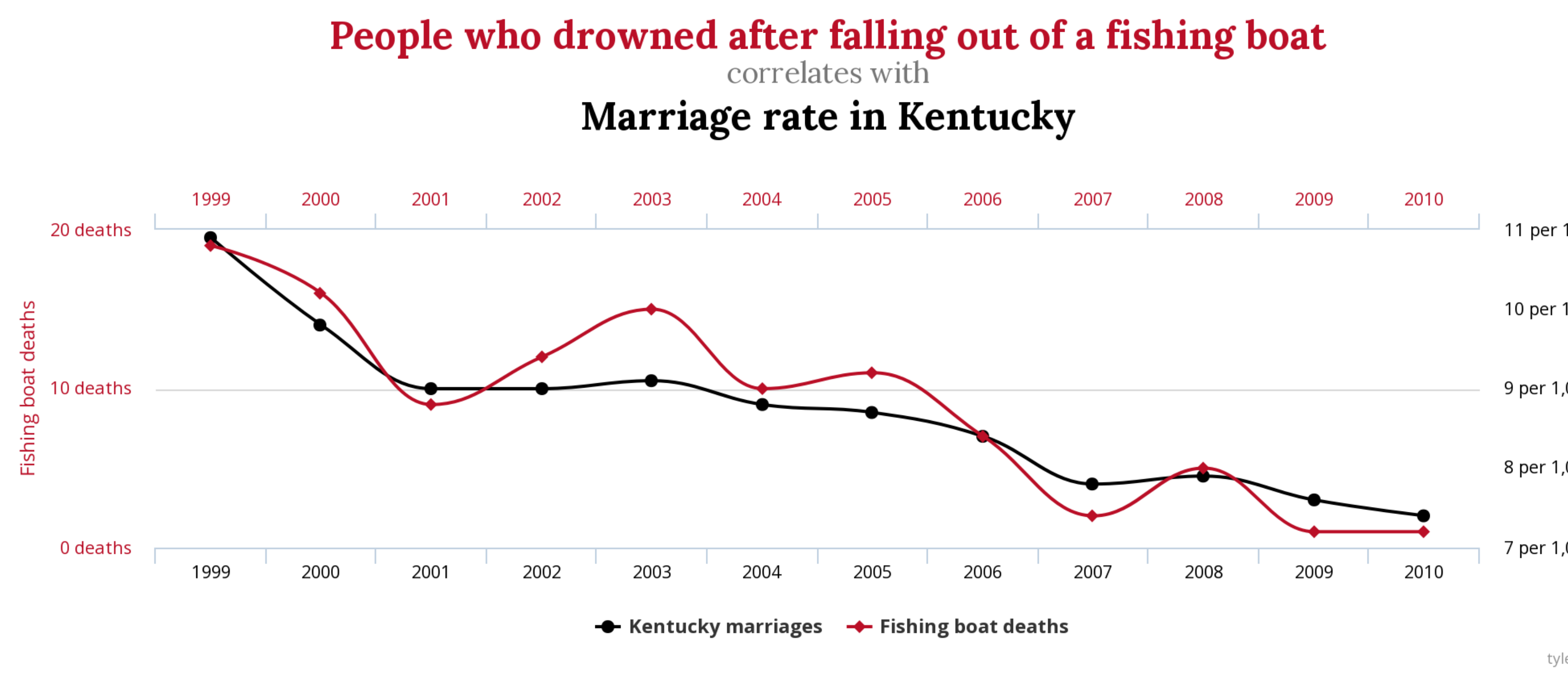
[1]: https://www.kaggle.com/getting-started/167037

Algunos Ejemplos

fuente: http://www.tylervigen.com/spurious-correlations







Debemos tener en claro que al armar nuestros modelos, no podemos apresurar nuestras conclusiones simplemente por tener una correlacion entre las variables. Existen maneras de prevenir esto y es un tema vigente de estudio. En el 2020, en la ICML (International Conference of Machine Learning) se presento un paper de investigadores de la Universidad de Stanford donde se presenta esta tematica. Pueden leerlo aqui: https://arxiv.org/abs/2007.06661

Regresion Lineal Simple

Vamos a empezar con lo que verdaderamente nos trae a esta practica: Modelos de Regresion.

Comenzaremos con un modelo de Regresion Lineal Simple. Consideraciones:

- Trabajaremos con un Dataset que incluye datos de las propiedades de Boston
- La libreria a usar es scikit-learn (https://jmlr.csail.mit.edu/papers/volume12/pedregosa11a/pedregosa11a.pdf)
- Usaremos matplot para visualizar la data.

In [ ]:
```
#Importamos la libreria sklearn, una libreria sobre machine learning en Python.
#De estas librerias traemos datasets,que son datos que vienen con la libreria
#para fines educativos. Traeremos el modelo de regresion lineal que vamos a entrenar.
from sklearn import datasets, linear_model
#De la libreria sklearn.metrics traeremos aqullas formulas que nos permiten
#evaluar la precision de las predicciones
from sklearn.metrics import mean_squared_error, r2_score
```

Nuestro dataset sera un conjunto de datos sobre viviendas en Boston. (https://scikit-learn.org/stable/datasets/toy_dataset.html#boston-dataset) Una cosa a tener en cuenta: sklearn viene con los datasets de Boston cargados y su manipulacion para entrenar el modelo es trivial. Pero nosotros no vamos a hacer las cosas tan a alto nivel. Vamos a usar lo aprendido anteriormente para que manipulen los datos desde 0. Por ellos vamos a construir nosotros nuestro DF con panda

In [ ]:
```
#Antes de comenzar, recomiendo leer las referencias del dataset
import pandas as pd
boston = pd.read_csv('boston.csv')
#Veamos los primeros elementos
boston.head()
#Nuestro dataset tiene 13 variables explicativas (CRIM, ZN, INDUS, CHAS, ..., LSTAT)
#La variable objetivo es MEDV y significa Median value of owner-occupied homes in $1000’s
```

In [ ]:
```
var_explicativas = boston[['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD',
'TAX', 'PTRATIO', 'B', 'LSTAT']]
var_explicativas
```

In [ ]:
```
var_objetivo = boston[['MEDV']]
var_objetivo
```

In [ ]:
```
import matplotlib.pyplot as plt
```

¿Que les parece que hace este loop?

In [ ]:
```
variables = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX', 'PTRATIO',
'B', 'LSTAT']
for var in variables:
    x = boston[[var]]
    plt.scatter(x, var_objetivo)
    plt.xlabel(var)
    plt.ylabel("MEDV")
    plt.show()
```

Entrenemos un modelo de regresion lineal simple. Nuestro objetivo es predecir el precio de una vivienda en funcion de LSTAT.

In [ ]:
```
#Paso 1: Cargamos el modelo
regr = linear_model.LinearRegression()
```

In [ ]:
```
#La variable explicativa sera la columna Rooms
var_explicativa = boston[['LSTAT']]
#Paso 2: (es clave) entrenar el modelo en base a los datos fuente
regr.fit(var_explicativa, var_objetivo)
```

In [ ]:
```
#Veamos cual es el beta 1 de este modelo. (la pendiente)
beta_1 = regr.coef_
beta_1
```

Pregunta: ¿Tiene sentido que la pendiente sea negativa?

In [ ]:
```
beta_0 = regr.intercept_
beta_0
#absurdo porque nunca van a existir 0 habitaciones
```

In [ ]:
```
prediccion_precios = regr.predict(var_explicativa)
prediccion_precios
```

In [ ]:
```
#Grafiquemos la linea de regresion y la dispersion de datos
plt.scatter(var_explicativa, var_objetivo)
plt.plot(var_explicativa, prediccion_precios, 'r')
plt.xlabel("LSTAT")
plt.ylabel("MEDV")
```

In [ ]:
```
import numpy as np
precio = np.transpose(var_objetivo)
```

In [ ]:
```
#Calculemos el error cuadrado medio
mse = 0
for i in range(0, precio.size):
    mse += np.power(precio[i] - prediccion_precios[i], 2)
mse = mse / precio.size
mse
```

In [ ]:
```
#Cuanto es el MSE o Error Cuadrado Medio
mean_squared_error(var_objetivo, prediccion_precios)
```

In [ ]:
```
#Cuanto es el R2
r2_score(var_objetivo, prediccion_precios)
```

Regresion Lineal Multiple

Ahora entrenaremos un modelo de regresion que se explique por dos variables: LSTAT y RM. Para ello utilizaremos tambien regresion lineal

In [ ]:
```
#En este caso elegiremos dos columnas como variables explicativas
var_explicativa = boston[['LSTAT', 'RM']]
var_explicativa
```

In [ ]:
```
ax = plt.figure().add_subplot(projection='3d')
ax.scatter(boston['LSTAT'], boston['RM'], boston['MEDV'])

ax.set_xlabel('LSTAT')
ax.set_ylabel('ROOMS')
ax.set_zlabel('MEDV')
```

In [ ]:
```
#Entrenamos el modelo, la var_objetivo sigue siendo MEDV (declarada previamente)
regr.fit(var_explicativa, var_objetivo)
```

In [ ]:
```
#Veamos cuales son los coeficientes Beta 1 y Beta 2
regr.coef_
```

In [ ]:
```
#Y ahora cual es Beta 0
regr.intercept_
```

In [ ]:
```
#Veamos que resultados da el modelo
prediccion_prec = regr.predict(var_explicativa)
```

In [ ]:
```
#Calculemos el error cuadrado medio
mse = 0
for i in range(0, precio.size):
    mse += np.power(precio[i] - prediccion_precios[i], 2)
mse = mse / precio.size
mse
```

In [ ]:
```
mean_squared_error(var_objetivo, prediccion_precios)
```

In [ ]:
```
#Cuanto es el R2
r2_score(var_objetivo, prediccion_precios)
```

Eso seria todo por hoy! Que no es poco... Ya estan listos para hacer la practica 1