STA130 Notes.

Winston Liang

Week 1

1. **Import Pandas!**

```
import pandas as pd # aliasing usage is pd.read_csv()
```

2. **read_csv**

- pd.read_csv(”path/url!”, encoding="ISO-8859-1”)
- ~ You need to link it to a “raw” .csv file. (e.g. raw.githubusercontent.com/...., instead of just github.io)

3. **Missingness**

```
# Suppose we have a dataframe:
df = pd.DataFrame({
    'A': [1, 2, np.nan, 4],
    'B': [np.nan, 2, 3, 4],
    'C': [1, np.nan, np.nan, 4]
})
```

Number of missing rows

- df.isna().sum

  A: 1

  B: 2

  C: 3
  dtype: int64

The dataframe in TRUE/FALSE values if its NA

- df.isna()

  A B C

  0 F T F

  1 F F T

  2 T F T

  3 F F F

Total Missing values: df.isna().sum.sum

- Output: 6

No. of missing values in each row

- df.isna().sum(axis=1)

No. of **rows** with missing values

- df.isna().any(axis=1).sum()
- it is summing the number of rows, not the number of missing data in a row

Drop ROWS

- df.dropna(inplace=True)
- df.dropna(how='all', inplace=True) ⇒ drop if have all missing values
- df.dropna(thresh=2, inplace=True) ⇒ Drop if has 2 or more missing values.

4. Variables and Observations

- **Observations**
  - **single records / ROWS**
- **Variables**
  - **columns**
    - Quantitative (Numerical like age)
    - Qualitative (Categorical like sex)

- df.shape ⇒ (821,30) #(row, col) #counts **all columns**
- df.columns ⇒ (list of names of column)
- df.rename(columns{'old': 'new'}, inplace=True)
- df.describe() ⇒ summarize all the **NUMERICAL columns**
- df.groupby('Column name').describe()
- del df['col']

Boolean Values and Coercion

df.isna() returns the boolean table whether it's missing.

df.isna().sum returns total number of missing values each columns

df.dtypes returns the datatypes (attribute)

df.astype() converts the datatypes

EE Questions:

Q1 Could you do a standard deviation calculation by hand?

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

Q2 Do you know how to index into a pandas DataFrame using df[...], df.iloc[...] and df.loc[...[or are you confused between these?

Let's break it down with simple examples to illustrate `.iloc` and `.loc`:

1. Using `.iloc` (Index-based selection)

- `.iloc` works with **integer-based indexing** (i.e., row and column positions). It uses the **positions of rows and columns** in the DataFrame.

Example:

```
import pandas as pd
url = "<https://raw.githubusercontent.com/KeithGalli/pandas/master/pokemon_data.csv>"
df = pd.read_csv(url)

# Example: Select rows 0 to 4 (5 rows) and all columns
df.iloc[0:5, :]
```

- This will select the **first 5 rows** and **all columns** (remember, Python is 0-indexed).
- Example: Select **first 5 rows** but only **columns 1 and 2**

```
df.iloc[0:5, 1:3]
df.iloc[1:4, [0,2]] # Returns Row 1 to 3, only with Column 1 and Column 3.
```

This selects the first 5 rows, and columns 1 and 2. The first column in Python is 0, so columns 1 and 2 are the **second and third columns** of the dataset.

2. **Using** `.loc` **(Label-based selection)**

- `.loc` allows you to use **row labels** and **column names** instead of integer positions. You can also use **boolean conditions** to filter data.

Example:

```
# Example: Select all rows where 'Type 1' is 'Fire', and only show 'Name', 'Attack', and 'Defense' columns
df.loc[df['Type 1'] == 'Fire', ['Name', 'Attack', 'Defense']]
```

- This will return all **Fire-type Pokémon** with only the columns **Name**, **Attack**, and **Defense**.

Another example combining multiple conditions:

```
# Example: Select Pokémon where 'Type 1' is 'Fire' and HP is greater than or equal to 100
df.loc[(df['Type 1'] == 'Fire') & (df['HP'] >= 100), ['Name', 'HP', 'Attack', 'Defense']]
```

- This filters Pokémon that are **Fire-type** and have **HP >= 100**, displaying only **Name**, **HP**, **Attack**, and **Defense** columns.

Key Differences:

1. `.iloc` is for **integer positions** (like selecting by row or column number).
2. `.loc` is for **label-based** selection (like selecting by column name) and works well with **boolean conditions**.
3. For df.loc[], you can use &, |, ~ (negation!) for boolean selection.
4. You can also do df.loc[0:2, "A":"B"], using integer slicing.

Week 2

1. python object types... *tuple*, *list*, *dict*
2. another key data type... *np.array* (and `np.random.choice`)
3. for loops... *for i in range(n):*

    a. *print()*
    b. *for x in some_list:*
    c. *for i,x in enumerate(some_list):*
4. logical flow control... *if*, *elif*, *else*

```
for index, value in enumerate(list):

...
```

Multinomials

Scipy.stats.multinomial

```
scipy.stats.multinomial(n = how many dice rolling at once, p = [1/6] * 6).rvs(size = how many times)
# rvs stands for random variables
# p = probability list must sum to 1

# Returns [1, 2, 1, 0, 1, 0] means 1 die showed 1, 2 dice showed 2, and so on
```

np.random.choice

```
import numpy as np

# Simulate rolling a six-sided die 10 times
rolls = np.random.choice([1, 2, 3, 4, 5, 6], size=10, p=[1/6]*6)
# Less flexible: CANNOT ROLL MULTIPLE DICE AT ONCE

# Returns [3, 6, 2, 1, 5] means the first die showed 3, the second die showed 6, etc.
```

np.random.choice can be sampled with replacement but not stats.

df.sample(n=num_rows, replace=True)

- we want it to be independent ⇒ so we use replacement = True (it's false by default in df!)

Sampling Distributions: NOT Skewed

Conditional Probability and Independence:

Independence: the outcome of one trial does not affect the outcome of another

- scipy.stats.multinomial is INDEPENDENT by default
- np.random.choice with 'replace=False' makes the outcomes **dependent (without replacement!)**

P(A | B) represents the **probability of A given B**

EE Questions:

Q1 Do you know how to do composite (multi-part) boolean selection using logical conditinals (with & | ~) to select data? Can you use tuple, list, and dict object types appropriately? How about using a for loop based on a list or not on a list using range(n) correctly with 0-based indexing?

Use loc. / iloc

Q2 Do you know what scipy.stats.multinomial and np.random.choice have in common (and what they don't), and what independence and conditional probability might (or might not have to do with them?)

#multinomial means multiple outcomes (binomial means two outcomes)

Week 3

Data types

1. Continuous
2. Discrete
3. Nominal (unordered descriptions: im a turtle snail butterfly...)
4. Ordinal (ordered descriptions: unhappy ok happy)
5. Binary (Only 2 mutually exclusive outcomes)

A. Bar Plots

1. Shows the frequency of CATEGORICAL data.

B. Histogram

1. Counts group of numbers within a range
2. Bin size (width):
    a. Too large: obscure important details about how data is distributed
    b. Too small: cluttered with many bars and hard to see overall pattern

```
import numpy as np
from scipy import stats
import plotly.graph_objects as go

# Generate a random dataset
np.random.seed(0) # Seed the random number generator for reproducibility
n = 500
data = stats.norm().rvs(size=n)

# Create a histogram with smaller bin size
fig1 = go.Figure(data=[go.Histogram(x=data, nbinsx=50)])
fig1.show()
```

C. Box plots and spread

- range, IQR, median
- Advantage: clear presence of skew
- Disadvantage: cannot represent multimodality (do not indicate amount of data)

D. A Normal Distributionn

- Symmetry. Central Tendency. Bell-shaped curve. Spread (SD). 68% within +- 1 SD. 95% within +- 2 SD.

E. Kernel Density Estimation (KDE) (NON-BINNED!)

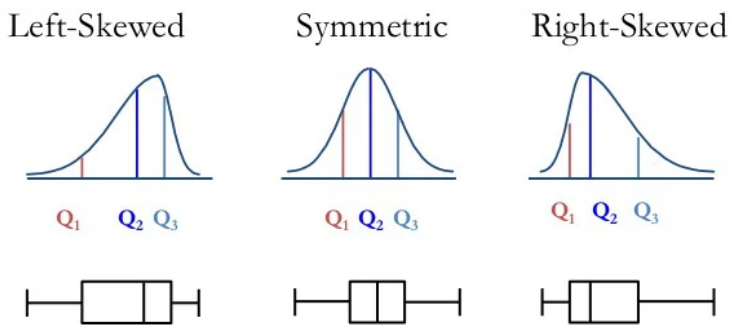
- A “local average of number of points”
- non-parametric estimation of the probability density function of a continuous random variable
- smooth curved function
- More ‘bins’ = Narrower bandwidth (the width of each kernel (point))
- Too few= oversimplifying / Too much = hard to grasp pattern
- Violin plot: just a mirrored image reflection of a KDE
  - Display multi-modal distributions which a box plot unable to visualize
  - Aesthetically pleasing than a histogram

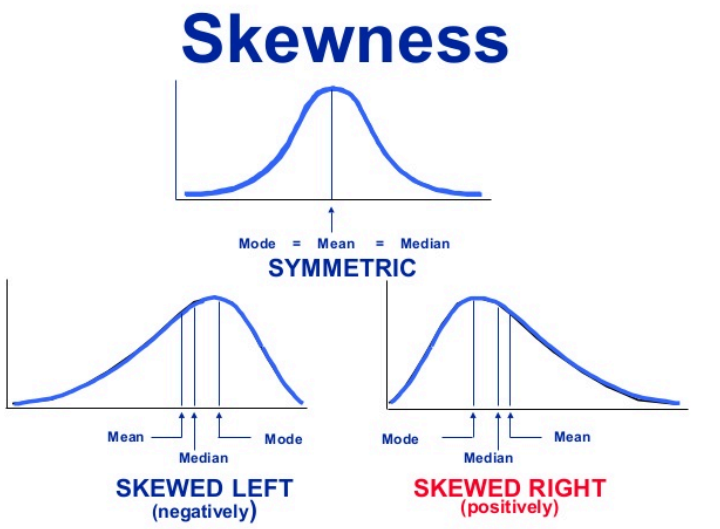
- **Legends** clarify what different elements in a plot represent.
- **Annotations** provide context and additional information about specific data points or trends.
- **Figure panels** organize multiple plots within a single figure to facilitate comparison and analysis.

Skew and Multimodality

Skew left (negative): Because outliers at left → Mean is more affected than median → Mean is more left than median

Skew Right (positive): Because outliers at right → Mean is more affected than median → Mean is more right than median





41

Log Transformation

- extremely right-skewed → normal scale
- pulls long tail of larger values closer to bulk smaller values
- make meaning of statistics like mean/sd more interpretable

Bar plot vs Histogram: categorical vs numerical

Box plot vs Histogram: no/have **sample size** + **multimodality** display

Box plot vs Violin Plot (KDE): combines box plot and KDE.

Violin Plot (KDE) vs Probability Density Function: approximation of PDF by smoothing out the data points VS probability of a random variable taken a given value

Here's a table summarizing the differences between various chart types, drawing on our conversation history and the provided sources:

Chart Type	Data Type	Purpose	Strengths	Limitations	Relation to PDF
Bar Plot	Categorical	Shows the frequency of different categories.	Simple and effective for comparing the prevalence of different categories.	Not suitable for numerical data.	Not directly related to PDF.
Histogram	Numerical	Displays the distribution of numerical data by showing the count of data points within specified ranges (bins).	Provides a detailed view of data spread, effectively reveals sample size and multimodality.	Can be sensitive to bin width choice, potentially obscuring underlying patterns.	The shape of a histogram can be thought of as an empirical approximation of the PDF.
Box Plot	Numerical	Summarizes data spread and central tendency using quartiles, median, and potential outliers.	Offers a concise way to compare distributions, particularly useful for identifying skewness and outliers.	Doesn't show the full distribution shape, may miss multimodality, and doesn't inherently display sample size.	Not directly related to PDF, but can offer hints about skewness which is a property of PDFs.
Violin Plot	Numerical	Visualizes the distribution of	Combines a box plot's summary with KDE's	Still an estimation of the PDF, may be	The KDE within the violin plot serves as a

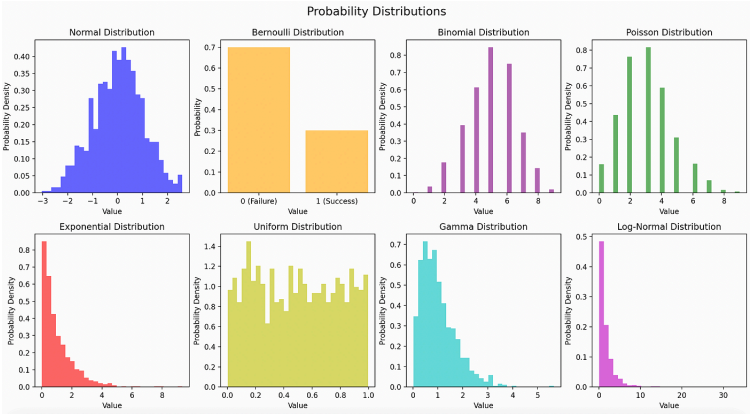
		numerical data using a kernel density estimation (KDE), reflecting the data's probability density.	smooth distribution representation, effectively displaying multimodality. Often considered more aesthetically pleasing than histograms.	influenced by KDE parameters, and could potentially obscure outliers compared to box plots.	visual approximation of the PDF.
PDF	**N/A**	**Not a chart type, but a theoretical mathematical function.** Describes the likelihood of a continuous random variable taking on specific values.	Provides a precise mathematical description of the probability distribution for a continuous variable.	Not a visualization tool itself, requires mathematical understanding.	**The PDF is the theoretical probability distribution that KDEs and violin plots attempt to visualize based on observed data.**

- *matplotlib*, is whats installed, pandas plots using it, everyone hates the syntax but its stablished
- *plotly*, super nice interactive plots, still cautious about it because its created by a for profit organization *gasp!*
- *seaborn*, its nice nice syntax, weird stats plots if you find the one you need.

- **Population (distribution) →→ (independent) Sample →→ Mean (statistic)**

Populations and Distributions

- Population: 'all' of a group
- Distribution: all possible data points



Sampling

- Sampling = choosing a subset from a population
  - has to be representative for ESTIMATING the characteristic of parameters of population
  - Statistical inference = **estimating population parameters** based on sample
  - Should use **independent** samples to avoid **bias**

Statistics Estimate Parameters

- Population Mean
- Population Standard Deviation

Sample Statistics

- Sample mean distribution
- Sample SD (Sample ERROR of Mean!)

EE Questions:

EE Q1: What's the difference between a bar plot and a histogram, a histogram and a box plot, a box plot and a violin plot (made with a kernel density estimator), and violin plot (made with a kernel density estimator) and a probability density function?

EE Q2 Can you fit a normal distribution to a sample of data using python? If so, how? Could you use the '.fit()' method of a distribution object from 'scipy.stats' to fit any distribution to any data set and visualize the result?

- loc (location) specifies the MEAN.
- scale specifies the SD.
- size specify the NUMBER of random samples.

```
np.random.seed(0)  # For reproducibility
data = np.random.normal(loc=0, scale=1, size=1000)  # Generate 1000 samples from a normal distribution

# Fit a normal distribution to the data
mu, std = stats.norm.fit(data)  # mu is the mean, std is the standard deviation
# It means storing the best fit MEAN and Standard Deviation into these two variables
```

Week 4 Bootstrapping

Simulation

- exercise of repeating a sampling process large no. of times (LARGER n = SMALLER VARIABILITY)

Simulation of a population (using Normal Distribution)

- sampling a single observation from a Normal distribution for n = 10000 times

FAQS:
1. Why is `number_of_simulations` introduced and why is it different than `n`?

- controls the number of times we REPEAT the sampling process. n is the SIZE (how big the sample is).

2. What is the effect of appending the `.mean()` **method** onto `normal_distribution_object.rvs(size=n)` inside the simulation `for` loop?

- calculates the MEAN of each simulated sample

  ⇒ Variability of sample statistics / means (instead of individual datapoints)

  ⇒ Creating a distribution of SAMPLE MEANS

3. Are the histograms from the two simulations equivalent? If not, why are the differences and what is causing them?

- Individual datapoints drawn from normal distribution of ONE SAMPLE(n)
- Sample means distribution of MANY SAMPLES (n, number of simulations)
- Usually sample means distribution is LESS SPREAD OUT / VARIABLE

Sample Mean Distribution: Variability/Uncertainty

- The statistic (like a sample mean) is being 'a sample' from the "distribution of the statistic".

Standard Error (SD/sqrt(n))

$$s_{\bar{x}} = \frac{s}{\sqrt{n}}$$

- = STANDARD DEVIATION of the sampling DISTRIBUTION of SAMPLED MEANS
- It refers to the variability of the sample mean relative to the true population mean (represents the precision of the sample mean as an estimate of the population mean)
- So, if the sample is made up of independent data points, then the larger the sample size n, the smaller standard error (so the smaller the "standard deviation of the sampling distribution of the sample mean" is), and thus the more "precisely" the sample mean x estimates the population mean μ.
- Standard deviation only refers to a sample or population in other sense.

Standard Deviation (SD)	Standard Error of Mean (SEM)
Variability among individual data points	Variability of SAMPLE MEAN (as estimate of population mean)
No effect from Sample size n	Decreases when sample size increase

Creating a Sampling Distribution:

1. Create a population using RVS
2. Pick many samples with a size n
3. Create a sampling distribution of sample mean (symmetric for large n)

Bootstrapping (get the job done using what you have available)

1. Pretend sample is the population (indepenent + large enough = sufficiently representative of the population)
2. Resampling with replacement
3. Bootstrapped Sampling Distribution of the Sample Mean

- Sample size n should be the same. (Similar variability with same sample size)
- replace = True (draw independent resamples)

TLDR:

Bootstrapping is a statistical approach to understanding **variability/uncertainty** of a **statistic**.

It is based on **resampling with replacement** from a **sample** to create many **simulated bootstrap samples** (of the **original sample size**), and understanding the behavior of the **statistic** across these **bootstrapped samples**.

Bootstrapping in this manner can be used to **simulate** the **sampling distribution** of a **statistic** (like the **sample mean**) which is what allows us to characterize **variability/uncertainty** of our **sample statistic estimators**.

The steps in bootstrapping are as follows.

1. Take an initial sample from the population.
2. **Resample with replacement** to create many **simulated bootstrapped samples** (of the same size as the **actual original sample**).
3. Calculate the **statistic** of interest (such as the **sample mean**) for each **bootstrapped sample**.
4. Create the **bootstrapped sampling distribution** of the **statistic** which indeed characterizes the **variability/uncertainty** of the **statistic** at the **sample size** of the **actual original sample**.
5. Use the **bootstrapped sampling distribution** to make statistical **inferences** about **population parameters**.

Estimation VS Inference

- Estimation is a POINT. E.g. Sample mean estimate population mean.
- Inference is based on distribution. E.g. Sample distribution inferences population parameters (is it TRUE / likely?)

Confidence Intervals

- There's a 95% chance this **confidence interval** does "**capture**" the actual true population parameter value.
- Or, we are 95% confident that the **interval CAPTURES** the population parameter
- chance that a new independent sample will fall into
- Confidence interval is NOT estimation but INFERENCE.
  - Estimation is a point estimate (e.g. the mean height is 173cm)
  - Inference allows us to UNDERSTAND UNCERTAINTY.
- Why do we need confidence intervals?
  - Confidence intervals provide more info than point estimates (like ONE sample mean): it reflects the INHERENT UNCERTAINTY in the estimation, while offering a PLAUSIBLE RANGE.

```
np.quantile(simulated_bootstrap_means, [0.025, 0.975])
```

EE Q1 Do you know why there is variability / uncertainty in statistics and what sample size has to do with it?

- There is variability because of the inherent dispersion during the random sampling process.
- The greater the sample size, the smaller the variability of the sample error of mean.

EE Questions

EE Q2 Can you sensibly use the words “bootstrapping”, “sample”, “statistic”, “population”, and “parameters” together in a sentence with little to no problems or concerns?

EE Q3: You would NEVER EVER NOT EVER POSSIBLY EVEN CONSIDER SAYING, “There’s a 95% chance the parameter is in this confidence interval”, right? Right??

Week 5

xi s refers to the sample values.

$\bar{x}$ refers to the sample average / statistic.

μ refers to the actual population parameter.

$\mu 0$ refers to the hypothesized population parameter (the null hypothesis).

Null and Alternative Hypotheses

- hypothesis made. statistic is used to provide inference about parameter. used to consider plausibility of hypothesis on the parameter

A Null hypothesis... (μ = parameter! μ0 = parameter under null hypothesis!)

- H0: there is no affect

  ⇒ H0 : μ = 0 (if μ means the change in average values, beyond scope of STA130, calledpaired t-test)

- H0: p = 0.5 (e.g. μ0 = 0.5)

Alternative hypothesis...

- HA : H0 is FALSE.

The 3 characteristics of an experiment under H0:

1. Sample Size
2. Number of Simulations
3. Hypothesized value of parameter

Why Simulation?

- To find the **variability / standard error of mean** ⇒ influences the strength of evidence we use against null hypothesis
- Sample size n plays a HUGE role.

```
import numpy as np
from scipy import stats
import plotly.express as px

# Set the parameters
n_coinflips_per_sample = 100  # Number of coin flips per sample
n_sample_simulations = 10000  # Number of samples to simulate
simulated_proportion_coinflips_heads = np.zeros(n_sample_simulations)
H0_p = 0.5

# Simulate the samples
for i in range(n_sample_simulations):
    # Simulate the coin flips
    n_coinflips = np.random.choice(['heads','tails'], p=[H0_p,1-H0_p], size=n_coinflips_per_sample)
    simulated_proportion_coinflips_heads[i] = (n_coinflips=='heads').mean()
    # or just use `proportion_coinflips_heads[i] =
    #             np.random.choice([0,1], p=[H0_p,1-H0_p], size=n_coinflips_per_sample).mean()`

# or instead of the `for` loop, this could be done as
# simulated_coinflips = stats.binom(n=1, p=0.5, size=(n_samples, n_coinflips_per_sample))
# simulated_proportion_coinflips_heads = simulated_coinflips(axis=1)

# Create the histogram using plotly.express
fig = px.histogram(simulated_proportion_coinflips_heads, nbins=30,
                   title='Sampling Distribution of Proportion of Heads (Fair Coin)',
                   labels={'value': 'Proportion of Heads', 'count': 'Frequency'})
fig.update_traces(name='Simulated Proportions')
fig.show()
```

p-values

- the **probability that a statistic is as or more extreme than the observed statistic if the null hypothesis is true** (statistic is the summary figure of one sample) / the **strength of evidence**
- provides **INFERENCE** about **POPULATION PARAMETER**

- p-value small → observed statistic not plausible under null → evidence in sample data disagrees with H0
  - So if the p-value is small, then the observed test statistic is very strange relative to the null hypothesis
    This means the data is very unusual if the null hypothesis is true, so it's probably more likely that the null hypothesis is fals
- either
  - fail to reject H0
  - reject H0 (when p is less than **α-significance level)**

- Rejecting a hypothesis at the α=0.05 means that **5% samples would incorrectly reject the null hypothesis when in fact it was true.** (NOT p VALUE!)

One-tailed VS Two-tailed Hypothesis Testing

$$H_0: p \leq 0.5 \text{ and } H_A: H_0 \text{ is false} \quad \text{or} \quad H_0: p \geq 0.5 \text{ and } H_A: H_0 \text{ is false}$$

One-tailed: p-value smaller as “as or more extreme” only considered in one-side, but same sample size.

Type I and Type II Errors

Type I Error (REJ)

- Reject null hypothesis when it’s actually true (FALSE POSTIVE)
- probability of Type I error = **alpha** α-significance level **threshold**

Type II Error (FAIL REJ)

- Fail to reject null hypothesis when it’s actually false
- probability of Type II error = **beta** β (out of scope)

Decision	Null Hypothesis is True	Null Hypothesis is False
Reject Null	**Type I Error (with chance α)**	Correct Decision
Fail to Reject	Correct Decision	**Type II Error (with chance β)**

1. **Formulate the Null Hypothesis (H_0):** This hypothesis often simply states that there is no effect or no difference.
2. **Collect Data and Perform a Test:** Use confidence intervals or p-values to make a decision regarding the null hypothesis.
3. **Make a Decision:** Communicate the nature of your decision based on a characterization of statistical evidence of your analysis.
4. **Alternative Hypotheses (H_1) are simple:** They're just "Not H_0" which is pretty boring and why confidence intervals are interesting.

The Reproducibility Crisis

- people do not know how to interpret the numeric value of p-value
- scientific findings hard to reproduce
  1. Use the p-value table
  2. Use Confidence Intervals instead

EE Questions

EE Q1 What's the difference between the sampling distribution of a statistic under the null hypothesis VS the bootstrapped sampling distribution of a statistic VS variance?

- First sampling distribution is sampled under a hypothesis
- Second is sampled under one sample that we assume as the population, with replacement and with same sample size
- Variance is the variability of the data points in one sample

EE Q2 What's a p-value, "alpha" α-significance level, and strength of evidence when it comes to statistical hypothesis testing?

1. p-value is the probability that a simulated statistic is as or more extreme than our observed test statistic under the null hypothesis
2. α-significance level is the thresholds that are used to determine the strength of evidence based on the p-values
3. Strength of evidence is the likelihood that we would do a successful inference to estimate the true population parameter

EE Q3 Can you correctly interpret all those things of the previous question? Like, how confidence intervals need to be interpreted carefully?

- the confidence that the interval CAPTURES the parameter
- Confidence intervals provide more info than point estimates (like ONE sample mean): it reflects the INHERENT UNCERTAINTY in the estimation, while offering a PLAUSIBLE RANGE.

Week 7ate9

```
# The ONLY code you will need
import statsmodels.api as sm
import statsmodels.formula.api as smf
import numpy as np
import pandas

import seaborn as sns
penguins = sns.load_dataset('penguins')
penguins = penguins.dropna()

mod = smf.ols(formula='bill_length_mm ~ island', data=penguins)
# mod = smf.ols(formula = 'Y~X', dataframe = df)
# mod = smf.ols(formula = 'Y~C(X)', dataframe = df)
# mod = smf.ols(formula = 'Y ~ X : Z, dataframe = df) (INTERACTIVE TERM)
# mod = smf.ols(formula = 'Y ~ X * Z, dataframe = df) (means Y~ X+Z + X:Z)
# Interactive terms: usually is beta(X times Z), where Z an indicator variable


res = mod.fit()

print(res.summary())
```

Correlation Assiociation (is NOT CAUSATION)

Correlation rx,y = Cor(X,Y) is within range 1 to -1.

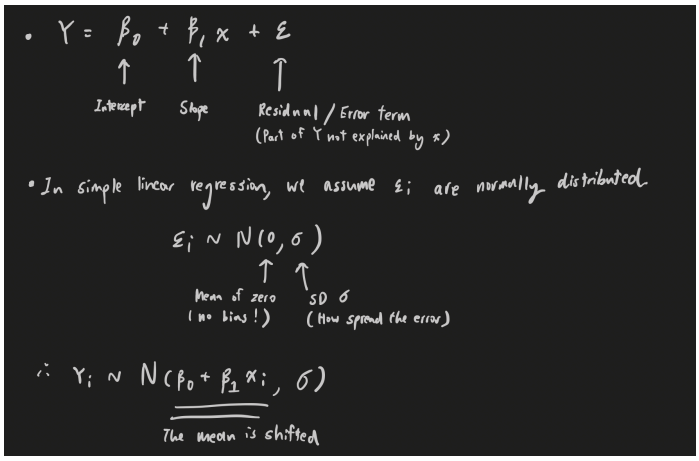
- 0 means NO linear relationship
- Variables: are they proxies or the real causes of what's happening?
- Genes: real cause; Parent height: a proxy
- Linear regression only works if there indeed is linear associaton

Simple Linear Regression is Just a Normal Distribution

$$Y_i \sim \mathcal{N}\left(\beta_0 + \beta_1 x_i, \sigma\right) \quad \longleftarrow \text{here we think of } Y_i \text{ as the r.v.} \tag{1a}$$

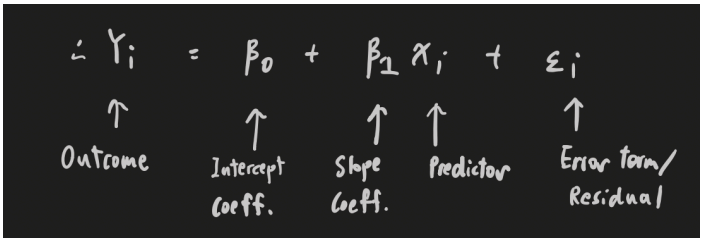
$$Y_i = \beta_0 + \beta_1 x_i + \epsilon_i \quad \text{where} \quad \epsilon_i \sim \mathcal{N}\left(0, \sigma\right) \quad \longleftarrow \text{here we think of } \epsilon_i \text{ as the r.v.} \tag{2a}$$

-



Central Limit Theorem: sum of independent, small random variables tend to be normally distributed

- A larger slope generally strengthen correlation between X Y.
- A small SD in error term N(0, SD) results in stronger corrleation between X Y.



Remember ERROR is for theoretical mode, RESIDUAL is for estimated model

Assumptions:

1. errors are normally distributed
2. errors are homoscedastic (SD of error term does not change as function of x. It is always constant.)
3. linear form is true in the sense that the above remain true so that then behavior of Yi outcomes are determined by the linear equation

4. errors are statistically independent and unbiased, mean of error distribution is 0
5. predictor xi is measured without error

TUT/HW Topics

statsmodels

- scipy.stats
  - **Distributional** and **hypothesis testing**
- statsmodels
  - Combines scipy.stats distributional and hypothesis testing for fitting linear regression models

```
import statsmodels.formula.api as smf
```

smf.ols

(statsmodels.formula.api.ordinary.least.squares)

- specify a linear regression model in pandas df object...

```
linear_specificiation = “y ~ x” # the R-Style formula (Outcome on the left, Predictors on the right)
model_data_specification = smf.ols(linear_specification, data=df)
```

$$Y_i \sim \mathcal{N}(\beta_0 + \beta_1 x_i, \sigma) \quad \text{or} \quad Y_i = \beta_0 + \beta_1 x_i + \epsilon_i, \quad \text{where} \quad \epsilon_i \sim \mathcal{N}(0, \sigma)$$

- x and y would then be the names of columns in df.

```
import pandas as pd
import statsmodels.formula.api as smf

# Simplified version of Galton's data: this dataset is small and simplified for illustrative purposes
# The actual dataset used by Galton was much larger and included more precise measurements and other variables; ho
wever,
# this should give you a sense of the type of data Galton worked with
francis_galton_like_data = {
    'parent_height': [70.5, 68.0, 65.5, 64.0, 63.5, 66.5, 67.0, 67.5, 68.0, 70.0,
                      71.5, 69.5, 64.5, 67.0, 68.5, 69.0, 66.0, 65.0, 67.5, 64.0],
    'child_height': [68.0, 66.5, 65.0, 64.0, 63.0, 65.5, 67.0, 67.5, 68.0, 70.0,
                     70.5, 69.5, 63.5, 66.0, 68.5, 69.0, 66.0, 64.5, 67.5, 63.5]
}
francis_galton_df = pd.DataFrame(francis_galton_like_data)

linear_specification = "child_height ~ parent_height"  # not `linear_specification = "y~x"`
model_data_specification = smf.ols(linear_specification, data=francis_galton_df)
```

Quoting

```
#linear_specification = 'Q("child\'s height") ~ Q("parents average height")' # or
#linear_specification = "`child\'s height` ~ `parents average height`"
```

Fitting Models

After using smf.ols... use x = smf.ols(linear_specification, data=df).fit()

```
data_fitted_model = model_data_specification.fit()  # estimate model coefficients and perform related calculations
data_fitted_model.params  # the estimated model coefficients beta0 and beta1 (in the case of simple linear regression)

#Output:
Intercept       3.023613
parent_height    0.947526
dtype: float64
```

Estimations of the parameter (true value) from the fitted model:

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$$

is used to estimate the theoretical model.

- Visualization: px.scatter(df, x="", y="", title="", trendline="ols")

Plotly: Generation

Generation of Theoretical Line and Fitted Line

```
# Generate best-fit line based on the data
import plotly.express as px
px.scatter(df, x='x',  y='Y', color='Data',
        trendline='ols', title='Y vs. x')

# Generates line based on THEORETICAL TRUE VALUES
import plotly.graph_objects as go
fig = go.Figure()
fig.add_trace(go.Scatter(x=x, y=Y, mode='markers', name='Data'))
```

▼ Explanation

Q1 line: this line is generated solely by the intercept and slope coefficient of the theoretical linear regression model that we used to generate the sample data. Therefore, this line is fixed and are not influenced by the data. We may see this as the 'true' model.

Q2 line: this line is based on the simulated dataset using the same intercept and slope coefficient, together with a random error term which is normally distributed by assumption. The error term introduces variability due random sampling, thus the fitted regression line would likely not be identical to the theoretical regression line we used to generate the data.

Each different sample would yield a slightly different fitted line due to sampling variability. Thus, we generate different fitted lines based on different samples, and they basically never overlap completely with the theoretical line.

.fittedvalues

- fitted values
  - are not proper predictions

```
data_fitted_model.fittedvalues  # model "predicted" *fitted values*, which for the current example is equivalent to
# y_hat = data_fitted_model.params[0] + data_fitted_model.params[1] * francis_galton_df['parent_height']

Output:
0     69.824213
1    67.455397
2     65.086582
3     63.665292
4     63.191529
5     66.034108
6     66.507871
7     66.981634
8     67.455397
...
dtype: float64
```

.rsquared variation proortion explained

- `data_fitted_model.rsquared ⇒ 0.9161191969033332`
- proportion of variation explained by the model.
- R^2 = 0.7 means 70% of variation in dependent outcome yi is explained by the model.

.resid residuals and assumption diagnostics

```
data_fitted_model.resid

Output:
0    -1.824213
1    -0.955397
2    -0.086582
3     0.334708
4    -0.191529
5    -0.534108
dtype: float64
```

Testing "On Average" Linear Association

- .summary() method

```
data_fitted_model.summary()  # generates a detailed report of statistical estimates and related information; but, ...
data_fitted_model.summary().tables[1]  # what we need for *hypothesis testing* is contained in its `.tables[1]` attribute
```

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	3.0236	4.540	0.666	0.514	-6.515	12.563
parent_height	0.9475	0.068	14.021	0.000	0.806	1.090

- P > |t| are p-values for a null hypothesis that the true coefficient value is zero.
- $H0 : \beta 1=0$ where $\beta 1$ is the hypothesized parameter value which the associated statistic $\beta \hat{1}$ estimates
- Small p-value rejects null hypothesis, suggesting a linear association 'on average' between the outcome and predictor variable

Two unpaired samples group comparisons (HYPOTHESIS TESTING USING P-VALUE)

Permutation Testing

- Label shuffling: Simulation of sampling distribution under null! Then, estimate the p-value of the actually observed statistic relative to this sampling distribution.
- If null hypothesis is true, then label shuffling does not matter as there is no difference between the two populations.

```
display(pd.DataFrame(pokeaman[ fire | water ].groupby('Type 1')['Sp. Atk'].mean()))
print(pokeaman[ fire | water ].groupby('Type 1')['Sp. Atk'].mean().diff().values[1])
# -14.168269230769226 (Actual observed difference in mean)
```

```
#Label Shuffling under null with Bootstrap simulation!
(Remember, same sample size n, replace = True!)
import numpy as np

groups4racism = pokeaman[ fire | water ].copy()

# Set parameters for bootstrap
n_bootstraps = 1000  # Number of bootstrap samples
sample_size = len(groups4racism)  # Sample size matches the original dataset
label_permutation_mean_differences = np.zeros(n_bootstraps)

for i in range(n_bootstraps):
    groups4racism['Shuffled Pokeaman Race'] = groups4racism['Type 1'].sample(n=sample_size, replace=True).values
    label_permutation_mean_differences[i] = \
        groups4racism.groupby('Shuffled Pokeaman Race')['Sp. Atk'].mean().diff().values[1]
```

```
# Show the histogram of 10000 boostrapped means.
# Calculate the p-value.

fig = px.histogram(pd.DataFrame({"label_permutation_mean_differences": label_permutation_mean_differences}), nbins
=30,
                           title="Mean Difference Sampling under SHUFFLED Pokeaman Race")

mean_differene_statistic = groups4racism.groupby('Type 1')['Sp. Atk'].mean().diff().values[1]

fig.add_vline(x=mean_differene_statistic, line_dash="dash", line_color="red",
              annotation_text=f".<br><br>Shuffled Statistic >= Observed Statistic: {mean_differene_statistic:.2f}",
```

```
            annotation_position="top right")
fig.add_vline(x=-mean_differene_statistic, line_dash="dash", line_color="red",
            annotation_text=f"Shuffled Statistic <= Observed Statistic: {-mean_differene_statistic:.2f}<br><br>.",
            annotation_position="top left")
fig.show()  # USE `fig.show(renderer="png")` FOR ALL GitHub and MarkUs SUBMISSIONS

print("p-value",
     (abs(label_permutation_mean_differences) >= abs(mean_differene_statistic)).sum()/n_bootstraps)
```

Two unpaired sample bootstrapping (CONFIDENCE INTERVALS FOR INFERENCE)

```
within_group_bootstrapped_mean_differences = np.zeros(n_bootstraps)
for i in range(n_bootstraps):
    double_bootstrap = \
        groups4racism.groupby("Type 1")[["Type 1","Sp. Atk"]].sample(frac=1, replace=True)
    # Bootstrapping the two columns separately :))

    within_group_bootstrapped_mean_differences[i] = \
        double_bootstrap.groupby('Type 1')["Sp. Atk"].mean().diff().values[1]
    # Store the difference of the two bootstrapped columns :))

np.quantile(within_group_bootstrapped_mean_differences, [0.05,0.95])

# array([-22.14708104,  -6.1709478 ])
```

- each sample is bootstrapped side-by-side (randomized so even if it “matches” with another sample it doesn’t matter as it is random!)
- Mean difference statistic (distribution) is obtained :0
- Generate the “double” bootstrapped confidence interval

Summary:

- Statistical methods are based on
    - A. Sampling distribution of a statistic under a null hypothesis (p-values)
    - B. One sampling distribution of the sample statistic (to provide inference about the population parameter, confidence intervals!)

Indicator and contrast linear regression

$Y_i = \beta_0 + \beta_1 \times 1_{[\text{group}]}(k_i) + \epsilon_i \quad \text{where } 1_{[\text{group}]}(k_i) \text{ will become the value of 1 if “} k_i = \text{group”} \text{ is } \texttt{True}\text{, and will othe}$

- 1[group](ki) means If ki == group, then return 1.
- Variable ki doesn’t have to be binary.
- Indicator variable 1[group](ki) is a binary variable.
- The contrast β1 captures the difference between the two groups.

- The contrast $\beta 1$ captures the difference between the two groups.

Notes for week10 and week 11 classification

- Logistic regression is NOT regression it is classification Regression ESTIMATES continuous variable. Logistic regression outcome is either 0, 1.
- The difference between multiple linear regression and logistic regression: ONLY smf.ols and smf.logit, and the outcome variable is either continuous and binary

```
import statsmodels.formula.api as smf

formula = '''
I((income=='>50K').astype(int)) ~ scale(age) + I(scale(age)**2) + I(scale(age)**3)
                        + C(education, Treatment(reference='HS-grad'))
'''
logreg = smf.logit(formula, data=train) # THE ONLY CHANGE! You just change ols to logit hahahha
logreg_fit = logreg.fit()
logreg_fit.summary()
```

Pseudo R-squared: regression

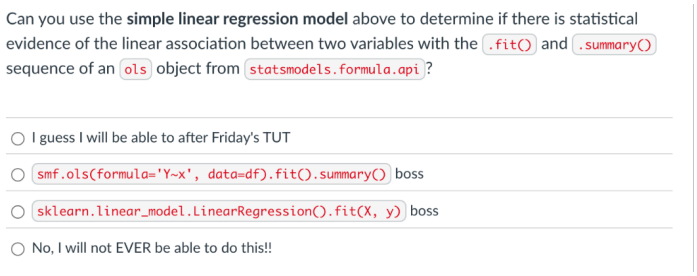
EE Q2 Can you name each element of the following expressions, and explain why they are equivalent and what it would mean to put “hats” on the coefficient parameters?

$$Y_i = \beta_0 + \beta_1 x_i + \epsilon_i \quad \text{where} \quad \epsilon_i \sim \mathcal{N}\left(0, \sigma^2\right)$$
$$Y_i \sim \mathcal{N}\left(\beta_0 + \beta_1 x_i, \sigma^2\right)$$

- Putting hats means ESTIMATED. Without hats means TRUE PARAMETER.

- You shift the Yi into a normal distribution.
- Line two shows that the deterministic part is now the mean of the normal distribution.

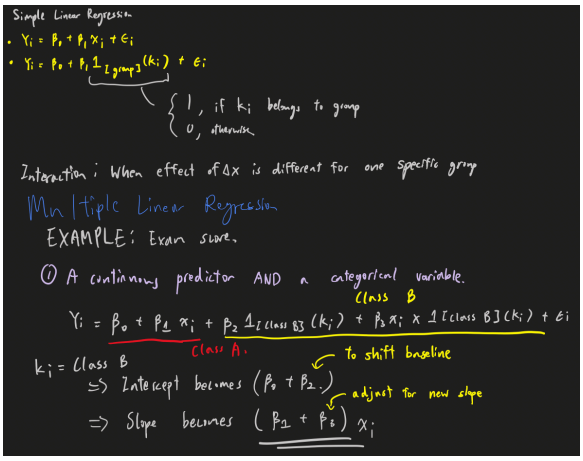
EE Q3. Can you name each element of the following expressions, and explain why they are equivalent and what it would mean to put “hats” on the coefficient parameters?

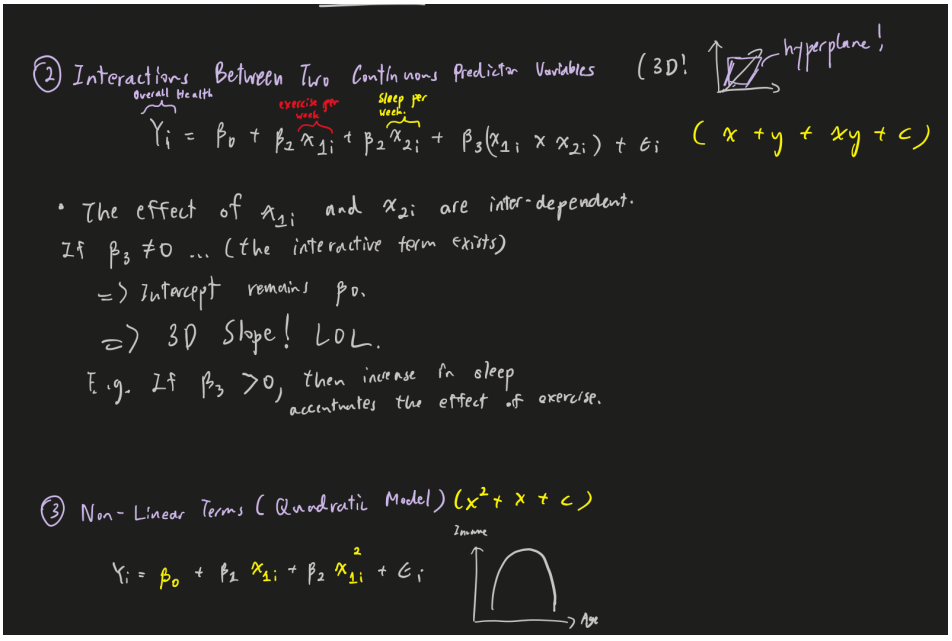


- p-value

- Statsmodel smf.ols: PROVIDES STATISTICAL EVIDENCE (p-values, coefficient significance)
- Sklearn: PREDICTION AND MACHINE LEARNING (No statistical evidence)

Week 10 Multiple Linear Regression





④ Categoricals. (No continuous x_i predictors)

$$Y_i = \beta_0 + \beta_1 1_{[\text{Class B}]}(k_i) + \beta_2 1_{[\text{Class C}]}(k_i) + \cdots + \beta_k 1_{[\text{Class D}]}(k_i) + \epsilon_i$$

β_0: Base Line

- When k_i = Class B, $Y_i = \beta_0 + \beta_1$.
  When k_i = Class C, $Y_i = \beta_0 + \beta_2$,
- p-value; against null for 'no difference' (i.e. Class A = Class B)
- Confidence Interval: For INFERENCE

Model Fitting

3 Steps

1. model = smf.ols ('Y ~ X', data) * Defined model, but NO calculations yet (model object that holds the specific regression structure)
2. result = model.fit() * Calculate the results in a new object
3. result.summary

Now Hypothesis:

H_0: $\beta_k = 0$ within the context of full model (for all k i)

Evidence-based Model Building (Use when you have few predictors)

- 1. Forward Selection (Building Up)
  - 1. Start with NO predictors
  2. Add predictor β_k with smallest p-value.
  3. Refit and check p-values of the included predictors.
  4. Continue adding predictors as long as statistical significant (p-value < 0.05)
  5. Stop when no more $p < 0.05$

  Reason to check: p-values of one predictor can change in the model, after adding other predictors (interaction → more statistically significant)

Recap: Assumptions

1. Linearity
2. Independence (errors are independent)
3. Homoscedasticity (constant residual variance)
4. Normality (residuals normally distributed)
5. No Multicollinearity (predictors with that highly correlated each other)

2. Backward Elimination (Pruning Down) (Use with many predictors)
   1. Start with all predictors
   2. Fit the model, examine p-value for each predictor
   3. Remove predictor with highest p-value
   4. Refit after each removal (check if p-value for predictor changes)
   5. Repeat until all predictors p-value < 0.05

Avoids... (As predictors with significant evidence are included)

- Overfitting: too many predictors → Perform well on training → Poorly on new, unseen data. (Capture random fluctuations in training data, don't generalize to new data)
- Underfitting: too few predictors → doesn't capture important relationship (too restricted) in data.

Performance-based Model Building.

1. Train-Test Framework.
   - Training data (used to estimate the model's coefficients) ← "Fitting"
   - Testing data (model never seen before)

2. In-Sample VS Out-of-Sample Performance
   - In-sample performance
     - ▶ R^2 score on Training data.
   - Out-of-sample performance
     - ▶ R^2 score on real-world data.

3. Model Generalization
   - well-generalized = make accurate predictions on new, unseen data.

4. train_test_split
   - Use train_test_split to split data into testing/training data

Complexity, Multicollinearity and Generalizability.

①
- Model Complexity: adding more predictors → more complex
- Model Generalizability: how well it can predict unseen data.
  - Complex model may be too tailored to traning data
  - Simpler model may less prone to overfitting

② Overfitting / Risk of Adding Predictors
- can fit training data better
- ↓ capture random noise rather than true underlying patterns
- ▷ Learns specific quirks (in training set) instead of general relationships

③ Multicollinearity

- 2 highly correlated predictors
  - → make it difficult to determine which predictor is truly driving the outcome
  - ☹ Increase uncertainty in coefficient estimation (unstable coefficient estimates)
  - ☹ Significantly undermine validity of your inference
  - ☺ doesn't necessarily harm predictive power of model

④ Understanding R^2 and Adding Predictors

- R^2 measures proportion of variation explained by predictors in model
- Adding more predictors NEVER DECREASE R^2.
  - → we are providing model with more information.
  - → Irrelevant predictors → boost R^2 artificially
    - → Not improving generalizability.

⑤ Out-of-sample R^2 and the Train-Test Framework

→ Adding irrelevant predictors...

↑ Increase in-sample R^2

↓ Decrease out-of-sample R^2

⑥ Diagnosing Overcomplexity & Multicollinearity

- Check the no. of predictors
- Check multicollinearity (out of scope)

Logistic Regression

- Outcome Y is binary (0 or 1)
- Link function (the logit function)
  → transforms output of linear model into value between $[0, 1]$.
- $\log \text{odds} = \log\left(\frac{p}{1-p}\right) = \text{logit}(p)$

  ↑ the ratio of an event occur : does not occur

  $p : 1-p$ → odds
- logit(p) is used transform probabilities (between 0 to 1) to -inf to inf.

Odds $= \frac{p}{1-p} = \frac{0.7}{1-0.7} = 2.33$ (Event is 2.33 likely to occur than not to occur)

log odds	
Greater than 0	probability of event > 0.5
Smaller than 0	probability of event < 0.5
Equal to 0	probability of event $= 0.5$

Coefficient in Logistic Regression.

- Attack coefficient $= 0.0172$
- One-unit increase in 'Attack' => log odds of it being Fire increase by 0.0172.

Increase in $e^{0.0172} \approx 1.017$ (odds)

∴ Odds increase by 1.7% (for each unit increase in Attack)

$$p = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k)}}$$

$-(\beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k)$ = logit(p)

$$\text{logit}(p) = -3.2644 + 0.0172 \times \text{Attack} + 4.3478 \times \text{Legendary}$$

(Sub 70) (Sub 1)

$$= 2.2874$$

$$p = \frac{1}{1 + e^{-2.2874}} = 0.897 \rightarrow 89.9\%.$$

Binary Indicator Variables and Cat2Bin Approach in Logistic Regression

- We need to make the outcome variable into a binary variable before feeding it to the logistic regression model.

  e.g. `df['Converted01'] = (df['Type 1'] == 'Fire').astype(int)`

  - If type 1 = fire, assign 1
  - If type 1 = others, assign 0.

- Create binary indicators directly using 'R-styled formula'

  `I(Q("Type 2") == "None")` (only works for predictor)

- Why not include every category? ✓
  - multicollinearity

  Type == Fire => 1
  Type ≠ Fire => 0

Extra...

Beyond Binary Outcomes: Multinomial and Ordinal Logistic Regression

1. Multinomial Logistic Regression (MNLogit)
2. Ordinal Regression.

} Concise understanding!

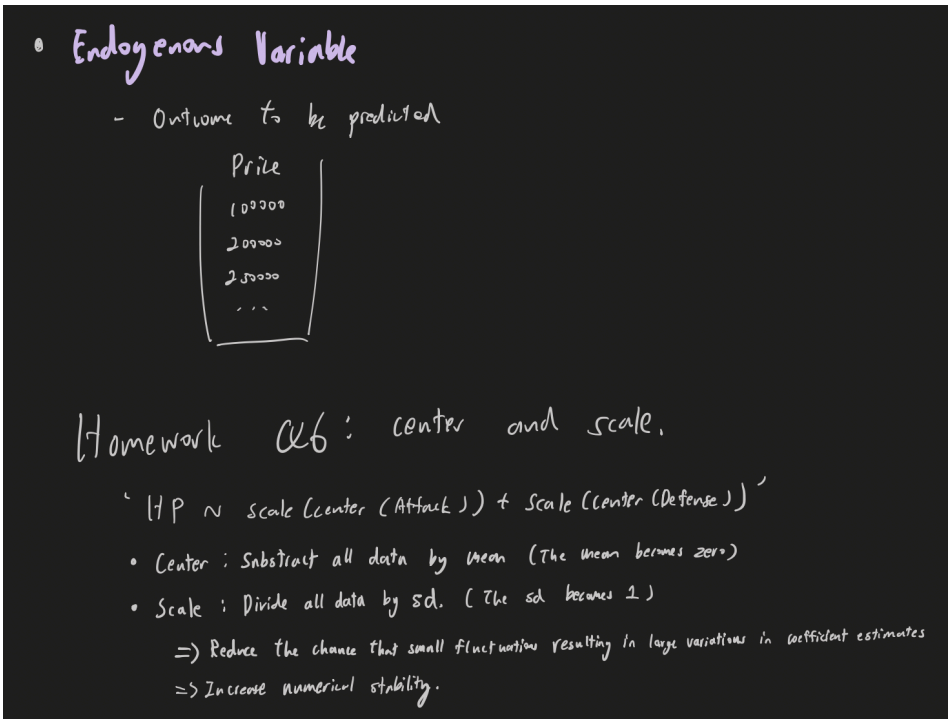
- Design Matrix (Exog)
  - the table of input to make predictions

Size (sqft)	No. Rooms
1000	1
1500	2
2000	3
2500	4

Automatically generated when we do `model = smf.ols('Y~X', data)`

✅ We use design matrix and endog vector to fit our model.

- Then after training our model, we can use our values from design matrix to predict back the endog values.



EE Q2.

Do you even know what $Y_i = \beta_0 + \beta_1 \times 1_{[\text{group}]}(k_i) + \epsilon_i$ is? And WHY EVEN it's different than $Y_i = \beta_0 + \beta_1 \times x_i + \epsilon_i$?? And what in the bloody hell $\mathcal{N}(0, \sigma)$ even has to do with ANY of this???

- Regression with categorical values.
- It is different because the second linear form is regression with continuous values.
- There's the error term which is a normal distribution.

EE Q3. Three methods to compare independent group (as opposed "paired samples") work?

- Group comparisions
  - Permutation tests
  - Bootstrapping
  - Indicator variables and contrast linear regression

EE Q2 Do you know what happens when you have continuous and binary, or a non-binary categorical, or a continuous and categorical, or even more variables, or even interactions between some of your variables in a multivariate regression model?

- The regression line splits

EE Q3 Can you start using model building to start exploring and searching through the course project data logistic regression?

- smf.logit

Week 11 Classification Decision Trees

Classification Decision Trees

- predict the value of an outcome based on sequential application of rules relative specific predictor variables
- 2 Types: Classification and Regression Decision Trees
- Multiple Linear Regression: Predict a continuous outcome
- Logistic Regression: predict a binary categorical outcome (l**inear models** (Y~X) for general categorical outcomes are possible) (IT IS CLASSIFICATION NOT REGRESSION)
- Classification
  - Binary (typically referred)
  - Multi-class

scikit-learn versus statsmodels

- statsmodel
  - Statistical Model functionality: tools for Hypothesis testing and Inference (not in scikit-learn)
- scikit-learn
  - Machine Learning: only concerns the raw predictive performance of a model

Machine Learning

- High Multicollinearity: reduce model ability to provide precise statistical inference, but does not always imply predictive performance drop
- Multicollinearity → Model Complexity → Overfitting
- try to use incredibly complex models, but REGULARIZE them.
- Regularization
  - Main Goal: Regularize the tree to a model complexity that appropriately finds generalizable associations without overfitting
  - limit expressibility by constraining model fit.
  - e.g. limit the magnitudes of estimated coefficient parameters

Linear Regression — Machine Learning:

```
from sklearn import datasets
import pandas as pd
import statsmodels.formula.api as smf
from sklearn.linear_model import LinearRegression

cancer_data = datasets.load_breast_cancer()
cancer_df = pd.DataFrame(data=cancer_data.data,
                         columns=cancer_data.feature_names)

MLR = smf.ols("Q('mean area') ~ Q('mean fractal dimension') + Q('smoothness error')",
              data=cancer_df)
MLR_fit = MLR.fit()  # Fit the mulitple lienar regression model (MLR)
display(MLR_fit.summary())  # Hypothesis Testing and Inference
```

```
MLR2 = LinearRegression()
MLR2_fit = MLR2.fit(X=cancer_df[['mean fractal dimension','smoothness error']],
                    y=cancer_df['mean area'])
print(MLR2.intercept_, MLR2.coef_)  # NEITHER Hypothesis Testing NOR Inference
# Both estimated coefficients agree
# Both predictions agree
print(MLR_fit.predict(cancer_df[['mean fractal dimension','smoothness error']].iloc[:2,:]),
      MLR2_fit.predict(cancer_df[['mean fractal dimension','smoothness error']].iloc[:2,:]))
```

Classification Decision Tree — Machine Learning:

```
from sklearn.tree import DecisionTreeClassifier, plot_tree
import matplotlib.pyplot as plt
import numpy as np

# Make Benign 0 and Malignant 1
# JUST RUN THIS ONCE!!
cancer_data.target = 1-cancer_data.target
cancer_data.target_names = np.array(['Benign\n(negative)','Malignant\n(positive)'])

# Increase for more Model Complexity
Model_Complexity_Tuning_Parameter = 5
# Decrease to for more Regularization

# Initialize and train the Decision Tree classifier
clf = DecisionTreeClassifier(max_depth=Model_Complexity_Tuning_Parameter, random_state=42)
clf.fit(X=cancer_df, y=cancer_data.target)
# The outcome has changed compared to the earlier multiple linear regression model...

plt.figure(figsize=(12, 4), dpi=200)
plot_tree(clf, feature_names=cancer_data.feature_names.tolist(),
          class_names=cancer_data.target_names.tolist(),
          filled=True, rounded=True)
plt.title("Decision Tree (Depth = "+str(Model_Complexity_Tuning_Parameter)+")")
plt.show()
```

- Nature of relationship : decision rules in each node
- No direct mathematical expression that determines the prediction
- require traversal down a tree. based on number of decision rules made at each node of the tree
- Since each decision rule explains some amount of outcome → determine % of explanation of outcomes that should be attributable to each predictor (Like the coefficient in linear regression!)

```
import plotly.express as px

feature_importance_df = pd.DataFrame({
    'Feature': cancer_data.feature_names.tolist(),
    'Importance': clf.feature_importances_
}).sort_values(by='Importance', ascending=False).reset_index()
```

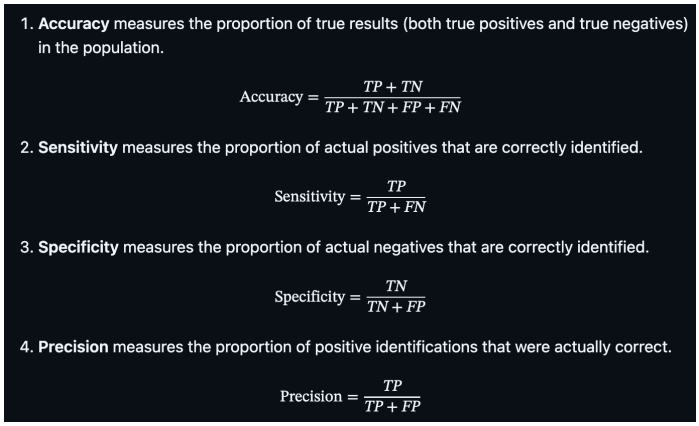
```
fig = px.bar(feature_importance_df[:15], y='Feature', x='Importance',
             title='Feature Importance')
fig.show()

# #https://stackoverflow.com/questions/52771328/plotly-chart-not-showing-in-jupyter-notebook
# import plotly.offline as pyo
# # Set notebook mode to work in offline
# pyo.init_notebook_mode()
```

Confusion Matrices

	Predicted **Postive**	Predicted **Negative**
Actual Positive	True **Positive**	False **Negative**
Actual Negative	False **Postive**	True **Negative**

Metrics

1. Accuracy: correct predictions / all
2. Sensitivity: correct positives / all actual positives (Also called “Recall”)
3. Specificity: correct negatives / all actual negatives
4. Precision: correct positives / all actual guessed positives



- False Positive Rate = Type I error α
- False Negative Rate = Type II error β

In Sample Versus Out of Sample

- I.e. the Train-test framework
- In-sample: the data used to fit the model
- Out of sample: the data unseen
    1. Use Panda’s .sample method
    2. Use train_test_split from sk_learn

```
from sklearn.model_selecton import train_test_split
import numpy as np

np.random.seed(130)
training_indices = df.sample(frac=0.5, replace=False).index #take randomly half of the sample!
testing_indices = df.index[~cancer_df.index.isin(training_indices)]
print(training_indices)
print(testing_indices)

np.random.seed(130)
train,test = train_test_split(df, train_size=0.5)
print(train.index)
print(test.index)
```

- Try the code below to determine what level of model complexity leads to the best out of sample predictive performance (outofsample)
-

```
import seaborn as sns
from sklearn.metrics import confusion_matrix

# Increase for more Model Complexity
Model_Complexity_Tuning_Parameter = 2
# Decrease to for more Regularization

# Initialize and train the Decision Tree classifier
clf = DecisionTreeClassifier(max_depth=Model_Complexity_Tuning_Parameter, random_state=42)
clf.fit(X=cancer_df.iloc[training_indices, :], y=cancer_data.target[training_indices])
# The outcome has changed compared to the earlier multiple linear regression model...

plt.figure(figsize=(12, 4), dpi=200)
plot_tree(clf, feature_names=cancer_data.feature_names.tolist(),
          class_names=cancer_data.target_names.tolist(),
          filled=True, rounded=True)
plt.title("Decision Tree (Depth = "+str(Model_Complexity_Tuning_Parameter)+")")
plt.show()

# Predict on the test set
y_pred = clf.predict(cancer_df.iloc[testing_indices, :])

# Generate the confusion matrix
conf_matrix = confusion_matrix(cancer_data.target[testing_indices], y_pred)

# Get the target names for 'benign' and 'malignant'
target_names = cancer_data.target_names.tolist()

# Set up a confusion matrix with proper labels using target names
plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
            xticklabels=[f'Predicted {label}' for label in target_names],
            yticklabels=[f'Actually {label}' for label in target_names])
```

```
plt.title('Confusion Matrix (Depth = '+str(Model_Complexity_Tuning_Parameter)+')')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')

# Add custom labels for FP, FN, TP, and TN
plt.text(0.5, 0.1, "True Negative (TN)", fontsize=12, color='red', ha='center', va='center')
plt.text(1.5, 0.1, "False Positive (FP)", fontsize=12, color='red', ha='center', va='center')
plt.text(0.5, 1.1, "False Negative (FN)", fontsize=12, color='red', ha='center', va='center')
plt.text(1.5, 1.1, "True Positive (TP)", fontsize=12, color='red', ha='center', va='center')

plt.show()
```

TUT/HW/LEC Extension

Model Fitting: Decision Trees Construction

- Exhaustive search of every possible decision rule (finding the optimal rule to sequentially grow the Decision Tree)
- “Greedy Search Algorithm”
- Regression: Mean Squared Error, with complexity penalty or stopping rule
- Classification: Gini Impurity / Shannon Entropy, with complexity penalty or stopping rule

Model Complexity and Machine Learning

- Machine Leanring: extends statistical models with REGULARIZATION TUNING PARAMETERS
- Allows greatly improved our of sample performance over traditional statistical modelling

Traditional Statistical Method	Machine Learning
Interpretability of Linear form	Raw predictive peformance

- Decision trees are incredibly complex models
  - Large no. of decision rules → increasingly deep decision trees → out of sample accruacy metric becomes super overfit (incredible complexity potential of the Decision Tree)
- Regularization Tuning Parameters for Decision Trees:
- `max_depth`
- `min_samples_split`
- `min_samples_leaf`
- `min_weight_fraction_leaf`
- `max_features`
- `max_leaf_nodes`
- `min_impurity_decrease`
- `ccp_alpha`
  - ccp_alpha: measures the complexity of a fitted Decision Tree → chooses a final Decision Tree as tradeoff between complexity and in sample predictive performance of the Decision Tree

    - High ccp_alpha: Simple models (more penalty to complexity)
    - Low ccp_alpha: Complex models (less penalty to complexity)
    - “dial in”: tuning a dial on radio to find the best position
- In terms of raw predictive performance, deeper Decision Tree may still be preferred

Random Forest

- hundreds of different decision trees using bootstrapped samples, average the predictions from all Decision Trees
- No linear model can even get close to matching the predictive performance of Random Forest. (Capable fof capturing complex and nonlinear patterns..)
- Models are built by algorithms rather than explicitly specified equations (The data guide the model-building process, rather than imposing rigid assumptions)

Feature	Decision Tree	Random Forest
Model Type	Single tree	Ensemble of trees
Overfitting	Prone to overfitting	Reduced due to averaging
Data Randomization	None	Bootstrapped samples of data
Feature Selection	Considers all features for splits	Considers a random subset of features for splits
Prediction	Single tree prediction	Aggregated predictions from multiple trees
Speed	Faster for small datasets	Slower due to multiple trees
Robustness	Less robust to noisy data	More robust to noise and outliers
Interpretability	Easy to interpret	Harder to interpret (ensemble model)

Prediction

- .predict(x) methods (can be used when the models have been fit)
- Applies to Regression / Logistic Regression (Classification) / Decision Tree / Random Forest

ROC Curves

- Receiver Operating Characteristic Curve

y = True Positive Rate (估啱）

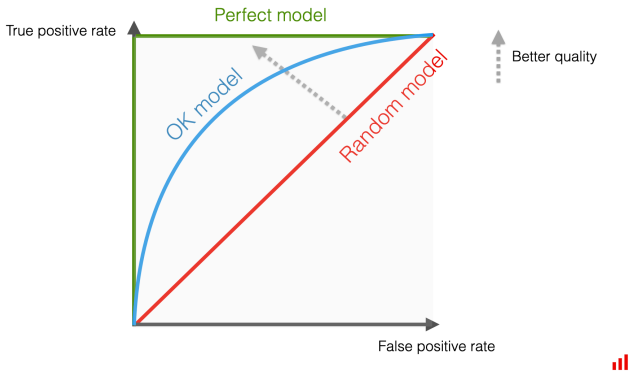
x = False Positive Rate （估錯）

Area Under the Curve (AUC):

AUC = 0.5 means RANDOM MODEL

AUC = 1.0 means PERFECT MODEL

AUC < 0.5 means WORSE than random



- Binary classifcation models output **probabilities** ⇒ Predict 0/1 based on probability larger or smaller than 0.5 (the THRESHOLD determines if the model predict 0 or 1)
  - (Logistic Regression, Binary Classification Decision Trees, Binary Classificaiton Random Forests)
- What if we don't use 0.5 as the threshold?
  - Larger than 0.5: fewer positive predictions (True Positive Rate Decrease)
  - Smaller than 0.5: more positive predictions (False Positive Rate Increase)

Partial Dependency Plots

- Relationship between outcomes and predictor variables

EE Q2 Can you make predictions using a Classification Decision Tree? Can you make a Confusion Matrix using a Classification Decision Tree?

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.model_selection import train_test_split
# import graphviz # if you want nicer tree visualisations

# Load your data into a pandas DataFrame
# Make sure your data is cleaned and preprocessed (no missing values, etc.)
data = pd.read_csv('your_data.csv')

# Split your data into features (X) and target variable (y)
# Choose the predictor variables relevant to your problem
X = data[['predictor_variable_1', 'predictor_variable_2', ...]]
y = data['target_variable']

# Split the data into training and testing sets (e.g., 80/20 split)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=130)

# Create a Decision Tree Classifier object
# Set hyperparameters like max_depth to control model complexity
clf = DecisionTreeClassifier(max_depth=4, random_state=130)
# Use random_state for reproducibility
```

```
# Train the classifier on the training data
clf.fit(X_train, y_train)

# Visualise the decision tree (optional, but helpful for understanding)
plot_tree(clf, feature_names=X.columns, class_names=['class_0', 'class_1'], filled=True)
# plt.show() # to display the plot

# Make predictions on the test data
y_pred = clf.predict(X_test)

# Evaluate the model's performance (e.g., using a confusion matrix)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)

# Explore feature importances (optional, to understand feature contributions)
print(clf.feature_importances_)
```

EE Q3 Are you wielder of Model Complexity, Machine Learning, and Thresholding?

- More predictors, more complex
- Machine learning...
- Threshold (If logistic regression gives 0.4, threshold is 0.5, then predict 0)

EE Q2 Do you know how to threshold probability predictions to change confusion matrices? Nd can you do so in meaningful ways relative to different metrics that might be of interest?

- As above

EE

Essay Writing

7*: Boxplot vs Histogram

1. Histogram shows more actual **shape** but boxplot simpler
2. Boxplot are easier to **compare** with each other
3. Boxplot cannot show **modality** like histogram
4. Boxplots does not show **data size** info
5. Histogram not helpful with **too many/less bins**
6. Both shows outliers (NOT a benefit)

7. KDE is the alternative (gives a smooth line)

8* p-value

- probability of getting a test statistic as or more extreme than the observed test statistic, assuming null hypothesis to be true

9*: relationship of parameters, population distributions, empirical distributions, samples, statistics, and sampling distributions (of statistical inference)

1. Parameters: characterize population features
2. Sample drawn from population
3. Empirical Distribution visualized with histogram, converges to population distribution shape with increasing sample size
4. Statistics are calculations of sample data that estimate parameters
5. Uncertainty of the estimation is characterized by variability of the sampling distribution from the process of sampling
6. ⇒ Confidence interval giving uncertainty characterization, evidence against null hypothesis

10*: What does it mean by 95% confidence that ... and why not same as saying 95% chance ... and is it sensible to talk about probability after a confidence interval is constructed?

1. 95% CI procedure construct an interval that does **actually capture the actual true parameter value 95% of the time** (theoretically)
2. A chance associated with **confidence interval procedure** is different than saying a chance with a **parameter**
3. probability of success after constructing is either 0 / 1 (either work or not work)

11*: 2 Ways we can influence the length of a confidence interval? Which is more useful?

1. Confidence interval (reduce → narrower)
2. Sample size (larger n → less variability in sampling distribution → distribution is closer to center)

18* Provide python pseudocode to simulate a p-value which approximates the “two sided” theoretical p-value for the binomial distribution considered in the previous few problems for an observed test statistic of 0 misses, and comment on how the simulation size influences the approximation accuracy.

```
import numpy as np; from scipy import stats
simulation_repititions = 10000
simulated_misses = np.array([0.0[*simulation_repetitions, dtype=float)
for r in range(simulation_repetitions):
    simulated_misses[r] = stats.binom(p=0.3).rvs(10)
(abs(simulated_misses-3) >= abs(0-3)).sum()/simulation_repetitions
```

- which is more accurately approximated by counting the number of successes we create samples from the sampling distribution

19 Give python psuedocode specifying reasonable population distribution models for each of the folowing, then briefly desscribe a general parametric method that could be used test the hypothesized parameters of these specifications and a general nonparametric method that could be used to provide statistical inference(estimation) of the parameters of these populations

- Height of students at UofT
- The number of correct answers on a quiz with 10 questions
- An unfair six-sided die with sides that don't have equal probability of being rolled

1. stats.norm(mean 1.75 meters, sd 1 metres)
2. stats.binom(p=0.5, n=10)
3. np.random.choice([1,2,3,4,5,6], p=[1/6,1/3,1/12,1/12,1/6,1/6])
4. parametric hypothesis test: these can define assumed null hypothesis and sampled to create sample distribution under the null
5. Nonparametric inference: bootstrapping from the sample to produce a bootstrapped confidence interval

20*. For contexts in which we are interested in population means, and we have samples x1 and x2 each stored as an np.array, write the null hypotheses and corresponding observed test statistic calculations for hypothesis testing involving one sample, two sample, and paired sample contexts.

One Sample: H0: mu = mu0 with test stat x1.mean()

Two Sample: H0: mu1 = mu2 with test stat x1.mean() - x2.mean()

Paired: H0: mu1 = mu2 with test stat (x1-x2).mean()

21.

. Suppose you have a sample of size $n = 2$ numbers $x_{(1)} < x_{(2)}$. Assuming sample quantiles defined as

$$\texttt{quantiles}(x_{(i)}) = \frac{\left(\frac{1}{n}\sum_{j=1}^{n} 1_{[x_{(j)}<x_{(i)}]}(x_{(i)})\right) + \left(\frac{1}{n}\sum_{j=1}^{n} 1_{[x_{(j)}\leq x_{(i)}]}(x_{(i)})\right)}{2}$$

draw a boxplot of this sample and explicitly note the rationale for your chosen treatment of whiskers and outliers in your visualization. Then draw the bootstrap sampling distribution of the median of this sample assuming an extremely large number of bootstrap samples (on a relative proportional scale as opposed to an absolute count scale), and annotate this figure with visual lines indicating the exactly precise numerical locations of the end points of the 50% confidence interval (which are the values that 50% of the bootstrapped sampling distribution are equal to or between). Report the length of this 50% confidence interval, and label your x and y axes and their numeric scales for both figures

1. Box plot: x1 to x2, Median in the middle (x1+x2/2)
2. Explanation: there is no whisker (just the box) as no data out htere
3. Sampling distribution: 25% at x1, 50% of the data at (x1+x2)/2, 25% at x2
4. 50% Confidence interval is length 0
5. Interval is (x1+x2/2, x1+x2/2)

22*. Explain what survivor bias means for UofT by discussing its possible role in differences in study habits that might exist between a sample of first year students currently enrolled at Uoft as opposed to a sample of fourth year students currently enrolled at UofT

1. study habits might be better in fourth year students sample as they have succeed and persisted at UofT
2. may include some students who don't have good study habits which would lead to them not be representative of the study behaviors that are present among the students who have succeeded in remaining at uoft into ther 4th year at uoft

32*. Giving statistical evidence of differences in protest sizes between cities using this model specification? Describe next steps to follow up on the analysis above and which comparisons you could address statistically with this current model specification.

1. use statsmodel to produce p-value and coefficient of the model
2. each ocomapres a city against Paris based on a null hypothesis that there's no difference in protest size on average between cities
3. gives stronger evidence of a rpotest sisze difference for each city versus paris the smaller the p-value

33. Give a possible confounder variable tha might go some way in explaining the difference in the average number of protesters in the different cities.

- city size, media attention...

34. Give the null hypotehsis for statistically analyzing each coefficient of the model

34. The statistical model currently under consideration can be specified as

$$y_i = \beta_0 + \overbrace{\left[\sum_{\substack{\text{except}\\ \text{Paris}}}^{\substack{\text{c in}\\ \text{cities}}} \beta_c 1_{[c]}(\text{city}_i)\right]}^{E[y_i|\text{city}_i]} + \epsilon_i$$

Write the *null hypotheses* for statistically analyzing each of the *coefficients* of this model specification.
1 point, and lose 1/4 point if β_0 is missing $H_0 : \beta_c = 0$ for c in cities and $H_0 : \beta_0 = 0$]

35. What would it mean in practical terms if all these null hypotheses were together simultaneously true? And how about if we asked this same question but without including the null hypotheses for beta0?

1. That there are no protests in any of these cities
2. All the cities have the same average number of protesters as Paris

42. Use the plot here compared to the previous plot to explain the conclusion of the previous problem

- Paris has large outliers in the original protest size scale aare pulling up the average in that model, but the log transformation seems to greatly relatively reduces these outliers, so the average in Paris for the log transformed outcome is lower. On the log transformed sacle, the averages in Marseille and Nantes are visually believable as being higher than Paris

43*. Why reliability and generalizability of an analysis based on 110 cities of greater concern than analysis based on 8 cities?

1. less data in smaller cities ⇒ more uncertainty in their estimation and declining accuracy reliability
2. small sample size means more opportunity we get a poor estimate ⇒ overfitting mistake

44. Describe a machine learning approach to determining the number of French cities that we would be willing to predict average protest size for using this data, and discuss the relative drawbacks of this compared to the drawbacks of a statistical hypothesis testing based approach.

1. train/test framework → we cold evaluate R2 in testing data set

2. Choose combinations of cities for which the model has the best overall out of sample performance
3. there might not be enough data to support a 80/20 train test framework though.

54. Explain why a false negative is worse mistake than a false positive in predicting heart attack risk.

- A false negative means heart attack risk was predicted to be Low Risk, but if the risk was in fact High Risk, then this would go unnoticed with an false negative prediction

55. Execution of a convicted person. False positive means the person is falsely accused to have committed the crime and is executed.

56. Draw a graph with Age on the x-axis and Weight on the y-axis, and partition the space according to the decision tree of the previous problem, labelling the regions of High Risk and Low Risk or annotating the figure to indicate the role of smoking in this distinction where appropriate.

...

57. Draw a confusion matrix such that the total number of data points is n=100 and the sensitivity is 80%, specificity is 80% and precision is 80%.

	true 1	true 0
pred 1	40 (TP)	10 (FP)
pred 0	10 (FN)	40 (TN)

58. How we should change the threshold if we were more concerned about false negative errors so that having a higher sensitivity was desirable.

- Lowering threshold → more positive (higher sensitivity, lower specificity)
- Precision can be same, increased, decreasead.

59. wtf

59. Consider the following recidivism classification matrices (of convicted criminals tendency to reoffend):

Social group A	Recidivates	Does NOT recidivate
Predicted to recidivate	59	13
Predicted to NOT recidivate	13	15

Social group B	Recidivates	Does NOT recidivate
Predicted to recidivate	17	13
Predicted to NOT recidivate	13	57

What is an argument against using *demographic parity*, making an equal proportion of positive predictions across both social groups (A and B), as a measure for fairness of a predictive algorithm?

2 points

(1) It fails to take into account the base rate – the actual rate at which people recidivate. Whether someone recidivates should play some role in whether they are predicted to recidivate.
(1) If we must predict more recidivatism [and act on that prediction accordingly in some harsh/authoritarian manner] for one group because the other group has higher recidivatism [or vice-versa], then it seems we are not fair to the people in the group who don't have as much recidivatism on average.

Essentially saying the two groups are different and can overpredict group B or underpredict group A if we force the positive prediction proportion to be the same, despite being discriminatory.

SUPER NOTES. Concepts

1	Coercion.	Coercion means changing data types.
2	Pandas	We cannot tell what data type passed based on variable names
3	Pandas	df[df.Event==”400 Meter Sprint”].groupby(”Medal”).count()
4	Pandas	You can use df[(df.Medal != “G”) & (...)] for filtering :) (using round brakets0
5	Data types	Qualitative (categorical), ordinal vs nominal, discrete...
6	Sampling	... be more careful with the question
7*	Visualization	Histogram vs Boxplot pros and cons and alternative
8*	p-value	the definition of p-value
9*	Sampling	elationship of parameters, population distributions, empirical distributions, samples, statistics, and sampling distributions (of statistical inference)
10*	Confidence Intervals	What does it mean by 95% confidence that ... and why not same as saying 95% chance ... and is it sensible to talk about probability after a confidence interval is constructed?
11*	Confidence Intervals	2 ways to influence length of CI, and which one useful more?
12	Bootstrapping	Fundamental assummption: presuming the sample can be used in place of the population
13	Numpy random	np.random.choice([”a”,”b”,”c”,”d”,”e”,”f”], replace=True, size=4) → 4 distinct values
14	p-values	Two-sided p-value calculation
15	p-values	One-sided p-value calculation
16	p-values	the strength of evidence it provides against H0 depends on the denominator of n = 10
17	Sampling	Variability of theoretical sampling is a function of sample size The accuracy of the estimation of the sampling distribution is a function of the simulation size
18	Coding: simulate p-value	
19	Coding: distribution model	
23	Test statistic	An efficacy x mean = 95% is the ratio of the absolute difference between the probabilities of laboratory-confirmed Covid-19 in the placebo and BNT173b2 groups divided by the maximum of the two probabilities.
24	Alternative Hypothesis	
25	p-value	Remember consider the other side too
26	Chance	
27	Chance	
28	Conclusion on data	... depends very rare event, may suffer from insufficient data
29	Relative risk vs Absolute Risk	the actual probability of the event the ratio of absolute risk of two groups (how much likely is in one group compared to others)

		⇒ Relative risk has increasingly widespread impacts as absolute risk increases
30	Prediction	Remember, Paris is missing because it is used as the intercept value!
31	Prediction	
36	error term assumption	
37	log transformation	
38	Homoskedasticity	means not a difference in the variability of the error terms across the different prediction levelw
39	Assumption of iid	iid: independently and identically distributed Not applicable for protests, since it would be unreasonable if the protests were driven by underlying social dynamics, such as growing social movement with each protest attracting new participants or new counter-protests.
40	p-value	p-value ANOVA model: theoretical parametric...
41	point estimate	do corresponding point estimates of the coefficient appear to agree? No, for example protest size in Marseille and Nantes relative to Paris changes between models
45	MLP	indiciator variables (3 options, just need 2 since we got one intercept)
46	linear form	y ~ C(x)*z: would model linear relationship between z and y uniquely within each of the groups
47	p-values	we don't know if parallel lines means p-value big or small for the variables
48	Design matrix	Only feed variables to the design matrix, not addition or interaction terms because we know it is parallel (y = beta0 + beta a 1[a](x) + betaz z. Same slope for different x categories.
49	Machine Learning	Machine learning vs Statistics: focus on performance evaluation as opposed to evidence and uncertainty characterization
50	Decision Tree	How deep would it be if we can only split into two?
51	False Positive	
52	Decision Tree	
53	Machine Learning	Regularization parameters: Leaf node size prior to proposed split, leaf node sizes that would result from a split, max tree depth, number of nodes in the tree (all of these options actually)
60	Equalized odds fariness criterion	both groups have equal proportions of FP and FN equally calibrated means both groups the chance a positive prediction is correct equals the true chance the observation is positive (same for negative)
61	Equalized odds + equally calibrated	the base rates must be correspondingly the same in group A and group B, because if this is so then having the same number of false positives and false negatives will then additionaly imply equal calibration

Midterm Tricky Questions

3	Inference	Inference: Quantifying uncertainty about a parameter
4	Inference	Confidence intervals are better - it quantifies uncertainty, as well as make decision about a null hypothesis like hyp test.
13	Chatbot	repeatedly randomly picking each subsequent word in the response proportionally to the conditional probabilities of words that are likely to follow the generated words so far
22	Inference	Inference: C as we find the confidence interval
24	Confidence Intervals	Narrower preferred ⇒ More INFORMATIVE (imagine what does 100% CI tells us?)
25	Confidence Intervals	How to find actual true coverage rate? through simulatin if we're willing to assume the samples population
27	Observation	we prefer independence
28	Free marks	
39		
40	Confidence Intervals	Bootstrap confidence interval estimate the Uncertainty regarding a sample baed estimate of the true value of the population parameter
45	KDE vs PDF	PDF is the relative frequency of infinitely many (as opposed to n from KDE) samples from a population as a density
46	KDE histogram	KDE AUC is 1. If normalziing histogram to provide a discrete approx of PDF in bar heights of 1,2,3,4, then how wide must the bins of histogram be? Ans: 0.1! Because we want the area under the curve to be 1.
54	Sampling	reflects the sampling distribution... we need a parametric option, which is only C (the coin flip one), since the other ones are non-parametric.\
55	Sampling	A.... proportion of red is farther from 0.5 than the proportion of reds observed in that of the game in quewstion