Module_1:

Team Members:

Molly Kessenich, Christy Lee

Project Title:

atherosclerosis in Alzheimer's Disease

Project Goal:

This project seeks to understand how atherosclerosis relates to beta-amyloid and tau levels in the brain.

Disease Background:

Fill in information about 11 bullets: Antibodies targeting Abeta and Ttau was used to determine the number of amyloid beta plaques per unit area and number of Ptau positive neurons per unit area

Neuropathological examination during autopsy was used to determine severity of atherosclerosis

Antibodies targeting Abeta and Ttau was used to determine the number of amyloid beta plaques per unit area and number of Ptau positive neurons per unit area

- Prevalence & incidence
    - over 7.2 million americans are living with alzheimers
    - 1 in 9 people age 65 and older have alzheimers disease
    - lifetime risk for alzheims are age 45 is 1 in 5 for women and 1 in 10 for men
    - older black americans are twice as likely to have alzheimers or other dementials as older white americans
    - hispanic people are 0.5-1 times as likely to have alzheimers as white people
    - 110 in every 100000 people age 30-64 have it
    - 20000 americans have younger-onset demetia

  https://www.alz.org/alzheimers-dementia/facts-figures

- Economic burden

    - health and long-term care costs for people living with it are projected to reach 384 billion.
    - ~12 million americans provide unpaid care for people with alzheimers
    - in 2024, unpaid caregivers provided an estimate of more than 19 billion hours of care which values to more than $413 billion

  https://www.alz.org/alzheimers-dementia/facts-figures
- Risk factors (genetic, lifestyle)
    - there are certain genes that can be inherited that can increase a person's chance of getting alzheimers
    - people with downs syndrome have a higher risk of developing alzheimers disease
    - people who live a healthy lifestyle (especially midlife) are less likely to develop alzheimers
    - not smoking, not drinking too much alcohol, eating a healthy and balanced diet
    - keeping physically, mentally, and socially active
    - protecting the head from injuries
    - diabetes, stroke, heart problems can increase chances
    - high blood pressure, high cholesterol, obesity can increase chances
    - age related hearing loss and depression can increase chances

  https://www.alzheimers.org.uk/about-dementia/types-dementia/who-gets-alzheimers-disease

- Societal determinants
    - lower level of education = poorer brain health (chances of getting alzheimers is twice as high for people without a hs diploma and lowest for college graduates)
    - cognitive reserve (the level of knowledge and education 'banked' in early years) can protect and compensate for a decline in cognitive health in later years
    - built environment (physical space where we live, work, learn, age) can influence overall community health and invidual behaviors such as physical activity and healthy eating. Unsafe or unhealthy environments can negatively impact brain health
    - healthy community design (crosswalks, bike lanes, sidewalks) improve opportunities for exercise and community supports social isolation significantly increases a person's risk of premature dealth and alzheimers

  https://www.cdc.gov/alzheimers-dementia/php/sdoh/index.html

- Symptoms:
    - Memory loss – especially forgetting recent events, names, and conversations.
    - Difficulty with problem-solving and planning – trouble managing tasks, finances, or following multi-step instructions.

    - Language problems – struggling to find words, follow conversations, or express thoughts clearly.
    - Disorientation – confusion about time, place, or familiar environments.
    - Impaired judgment and decision-making – reduced ability to assess situations or make safe choices.
    - Difficulty with familiar tasks – such as cooking, driving, or using appliances.
    - Changes in mood or personality – withdrawal, irritability, anxiety, or depression.
    - Loss of initiative – reduced interest in social activities or hobbies.
    - Progressive decline in motor and functional abilities – later stages may involve difficulty walking, swallowing, or controlling bladder/bowel function.
- Diagnosis
    - Clinical assessment:
        - Medical history and physical exam, cognitive testing, behavioral testing, social and functional history
    - Imaging
        - PET scans/MRI to visualize the changes in the brain's structure and function
    - Biomarkers
        - Blood/cerebrospinal fluid tests to measure levels of amyloid beta and tau.
    - Genetic testing
    - Early diagnosis:
        - Blood tests and artificial intelligence-based algorithms help diagnose early-onset
    - Diagnostic Criteria:
        - For diagnosis, the National Institute on Aging-Alzheimer's Association (NIA-AA) 2021 Research Criteria for Alzheimer's Disease require a combination of cognitive that affects multiple domains, evidence of underlying brain pathology from imaging/biomarkers, other potential causes of cognitive decline are ruled out. https://my.clevelandclinic.org/health/diseases/9164-alzheimers-disease#:~:text=Communicate%20or%20talk%20less%20than,or%20use%20t
- Standard of care treatments (& reimbursement)
    - Standard care of treatments
        - Symptomatic therapies to help with daily functioning, behavior, and cognition, but not the course of the disease
            - Cholinesterase inhibitors
            - NMDA receptor antagonist
            - Non-drug therapies
        - Disease-modifying therapies (DMTs) to slow disease processes
            - Lecanemab - anti-amyloid, slows progression in early disease
            - Donanemab - anti-amyloid,
    - Reimbursement

        - Newer DMTs are expensive, so there is reimbursement from Medicare, Medicaid, and private insurers
            - Medicare: for early disease covers new Alzheimer's drug
- Disease progression & prognosis
    - Progression:
        - Silent preclinical brain changes (amyloid buildup, tau tangles, subtle cell loss) that can occur 10-20 years before diagnosis
        - Moves to mild cognitive impairment (forgetting recent events, misplacing items, struggling to find words)
        - Mild stage: memory loss, trouble with daily tasks, mood changes
        - Moderate stage: confusion, disorientation, behavioral changes, increasing dependence.
        - Severe stage/late stage: profound memory loss, loss of speech, complete dependence, complications (for example, infections).
    - Prognosis:
        - No cure; treatments help manage symptoms or slow the decline temporarily.
        - Faster progression with a younger onset (<65 years)
        - Most deaths result from complications (like pneumonia, malnutrition, or falls).
        - Better prognosis factors include good overall health, a strong support system, early diagnosis, and active management of other health conditions.
- Continuum of care providers
    - First contact = primary care physicians for initial care and diagnosis, neurologists, geriatricians (specialize in the care of older adults), geriatric psychologists (mental and emotional health of older adults), neuropsychologists (assessment of cognition and behavior), general psychologists/social workers/clinical psychologists, speech/physical/cognitive therapists, nurses, health home aids

https://www.alzheimers.gov/professionals/health-care-providers

- Biological mechanisms (anatomy, organ physiology, cell & molecular physiology)
    - Anatomical changes - brain atropy/visible shrinkage, begins in the hippocampus and the entorhinal cortex (memory formation), neurons die and disease spreads to the cerebral cortex
    - Physiological Processes - amyloid plaques: in the space outside neurons, fragments of beta-amyloid protein form plaques and interfere with brain cell communication; Tau proteins tangle/twist inside neurons and distruct the microtubule transport system --> cell damage and neural death; Transmission of neural signals is disrupted as neurons lose their synapses; Neurotransmitter's

like acetylcholine (memory and cognitive function) decrease
https://www.sciencedirect.com/science/article/pii/S156816372400299X#:~:text=De
https://www.nia.nih.gov/health/alzheimers-and-dementia/alzheimers-disease-fact-sheet#:~:text=about%20their%20experiences.-,How%20does%20Alzheimer's%20

- Clinical Trials/next-gen therapies
  - Current clinical trials are aimed at preventative treatments, earlier interventions, targeting of amyloid and tau proteins with new drugs, earlier detection (AI to detect voice changes), gene therapy, etc.
  - Next-gen research is aimed at gene therapy and boosting growth factors like BDNF for the brain, and investigating targets other than amyloid and tau - eg. neuroinflammation
  - https://pmc.ncbi.nlm.nih.gov/articles/PMC10544555/#:~:text=Abstract,brain%20da

Data-Set:

Antibodies targeting Abeta and Ttau was used to determine the number of amyloid beta plaques per unit area and number of Ptau positive neurons per unit area

Neuropathological examination during autopsy was used to determine severity of atherosclerosis

- how was it aquired

Postmortem brain tissue and donor metadata were obtained via the UW BioRepository and Integrated Neuropathology (BRaIN) laboratory from participants in the Kaiser Permanente Washington Health Research Institute ACT Study and the University of Washington ADRC. Samples for this study were provided by the RADC, Rush University Medical Center, Chicago. The study cohort was selected based solely on donor brains undergoing precision rapid procedure (optimized tissue collection, slicing and freezing) during an inclusion time period at the start of the SEA-AD study The data was taken from a cohort 33 male donors, 51 female donors, average age at death is 88 quantitative neuropathology was used to place donors along a disease pseudoprogression score. There were two disease phases: an early phase with a slow increase in pathology and a latter phase with an exponential increase in pathology Various methods of tissue imaging and measurements were used to collect the data. The data was analyzed in either python with scripts or libraries

- when was it acquired

The data was published on 14 October 2024

- Who was it acquired by

Postmortem brain tissue and donor metadata were obtained via the UW BioRepository and Integrated Neuropathology (BRaIN) laboratory from participants in the Kaiser Permanente Washington Health Research Institute ACT Study and the University of Washington ADRC. The scripts written to analyze the data are available by the Allen Institute The data and studies were conducted by the Allen Institute for Brain Science

- What are some potential places for error in collecting the data?

Tissue quality is one of the most prominent sources of error directly affecting sequencing and imaging. There may also be bias in neuropathology measurements (staining/antibody variability, tissue sectioning artifacts, or automated image analysis thresholds). Errors could arise from RNA degradation in RNA sequencing. MERFISH spatial transcriptomics may have errors with cell segmentation. Another error with the computational and analytical data could arise from statistical errors and data interpretation.

- Is all the data objective, or is some of the data subjective?

The majority of the data is very objective, and the means of data collection are quantitative, and the researchers used unbiased means of data collection. To collect data, quantitative neuropathological measurements, single-nucleus RNA sequencing (snRNA-seq), single-nucleus assay for transposase-accessible chromatin with sequencing (snATAC–seq), single-nucleus multiome (snMultiome), and cellularly resolved spatial transcriptomics (multiplexed error-robust fluorescence in situ hybridization (MERFISH) were used from the cohort of 84 aged donors. These methods are very objective as they give measurable data and facts from the donors.

- What are the units the data is measured in

Quantitative neuropathological used immunohistochemistry (IHC) staining and units of the number of positive cells or objects per unit area in cells/mm² or plaques/mm² along with object size (for example, plaque diameter in µm), and immunoreactivity intensity/percent positive cell area snRNA-seq used Unique Molecular Identifiers (UMIs) (counts per nucleus), gene counts per nucleus (number of genes detected), mapped reads, intronic/exonic proportions (%). snATAC-seq used accessible chromatin peaks per nucleus counts, peak lengths (base pairs), and peak similarity (Jaccard distance, dimensionless). snMultiome used the same units/measurements as snRNA-seq and snATAC-seq MERFISH used transcript counts/cell, spatial position (µm) measurements. The cognitive testing used standardized scores (CASI)

- What questions about alzheimers disease are you curious about answering, given the data you have available in these data sets? (each partner thinks of 3 questions)

Is there a correlation between race and the MOCA score? How does education play a role in alzheimers How does gender play a role in alzheimers Does head injury impact tau

or beta-amyloid How do tau and beta-amyloid development differ depending on the age on onset in a patient? How does atherosclerosis relate to beta-amyloid and tau levels in the brain? How beta-amyloid and tau levels impact pH levels in the brain?

- Question we want to pursue:

How does atherosclerosis relate to beta-amyloid and tau levels in the brain? We can also talk about how blood flow impacts alzheimers with this question

- we have no questions for the teaching team
- our data is unpaired
-

Data Analyis:

- first, the data was gathered (ttau, ptau, ABeta40, ABeta42, and atherosclerosis). For purposes of creating graphs and conducting a t-test, we worked with mainly the ABeta40 and ABeta42 data along with the atherosclerosis to notice their patterns.
- we wanted to combine our data from ABeta40 and ABeta42, normalize the values, and then take the average of them to combine the data (as suggested by our TA). The equation: (data-mean)/stdev was applied to all the data points in ABeta40 using the mean of ABeta40 and stdev of ABeta40. The same equation was also applied to all the data points in ABeta42 using the mean and stdev of ABeta42
- All the data from the strings (already in code given to us) and numbers was instantiated and sorted.
- The average of the normalized values of the ABeta40 and ABeta42 was taken and used to create a box and whisker plot where the atherosclerosis level (None, Mild, Moderate, Severe) were plotted on the x-axis and the normalized average ABeta level was plotted on the y-axis.
- The following code was written to run a one-way ANOVA test on normalized amyloid-beta and tau levels compared to atherosclerosis categories. Both p-values were > 0.05, so we failed to reject the null hypothesis and found no statistical differences for both amyloid-beta and tau levels across atherosclerosis categories.

The following code will print a box and whisker plot of the normalized ptau and ttau levels in relation to the severity level of atherosclerosis

In [ ]:
```
import csv
import warnings
import matplotlib.pyplot as plt
import statistics
from scipy import stats
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
```

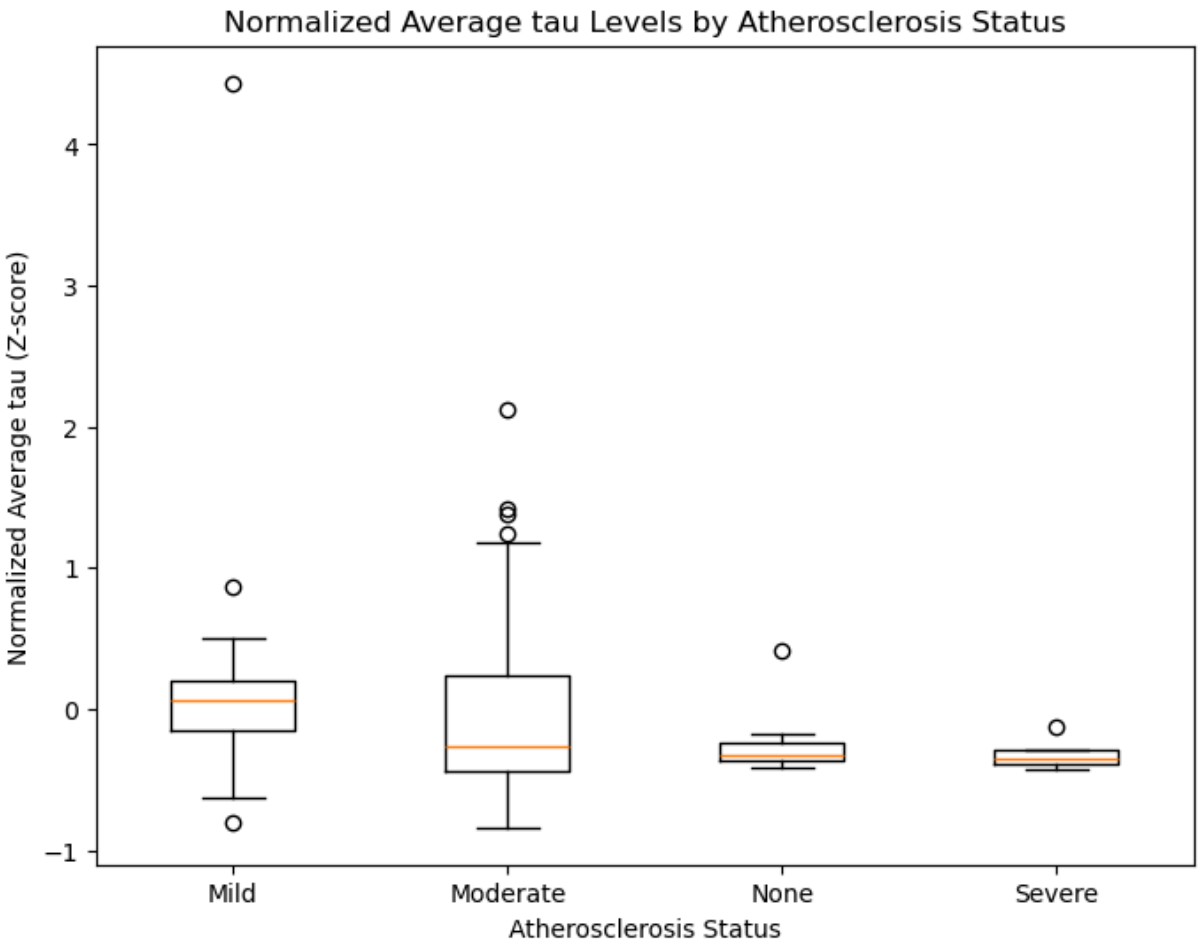
```
class Patient:

    all_patients = []
    death_age = []
    education_lvl = {}

    def __init__(self, DonorID, ABeta40: float , ABeta42: float, tTau: float
        self.DonorID = DonorID
        self.ABeta40 = ABeta40
        self.ABeta42 = ABeta42
        self.tTau = tTau
        self.pTau = pTau
        self.sex = None
        self.death_age = None
        self.ed_lvl = None
        self.cog_stat = None
        self.age_symp_on = None
        self.age_diag = None
        self.head_inj = None
        self.thal_score = None
        self.atherosclerosis = None
        # Add normalized values as attributes
        self.ttau_zscore = None
        self.ptau_zscore = None
        Patient.all_patients.append(self)

    def __repr__(self):
        return f"{self.DonorID} | sex: {self.sex} | ABeta40 {self.ABeta40} (

    def get_id(self):
        return self.DonorID

    def get_ABeta42(self):
        return self.ABeta42

    def get_thal(self):
        return self.thal_score

    def get_death_age(self):
        return self.death_age


    @classmethod
    def combine_data(cls, filename: str):
        with open(filename, encoding="utf8") as f:
            reader = csv.DictReader(f)
            rows_of_patients = list(reader)
            for row in range(len(rows_of_patients)):
                if Patient.all_patients[row].DonorID == rows_of_patients[row
                    if rows_of_patients[row]["Sex"] != "":
                        Patient.all_patients[row].sex = rows_of_patients[row

                    if rows_of_patients[row]["Age at Death"] != "":
                        Patient.all_patients[row].death_age = int(rows_of_pa
```

```
                    if rows_of_patients[row]["Highest level of education"] !
                        Patient.all_patients[row].ed_lvl = rows_of_patients[

                    if rows_of_patients[row]["Cognitive Status"] != "":
                        Patient.all_patients[row].cog_stat = rows_of_patient

                    if rows_of_patients[row]["Age of onset cognitive symptom
                        Patient.all_patients[row].age_symp_on = int(rows_of_

                    if rows_of_patients[row]["Age of Dementia diagnosis"] !=
                        Patient.all_patients[row].age_diag = int(rows_of_pat

                    if rows_of_patients[row]["Known head injury"] != "":
                        Patient.all_patients[row].head_inj = rows_of_patient

                    if rows_of_patients[row]["Thal"] != "":
                        Patient.all_patients[row].thal_score = int(rows_of_p

                    # atherosclerosis part
                    if rows_of_patients[row]["Atherosclerosis"] != "":
                        Patient.all_patients[row].atherosclerosis = rows_of_

                else:
                    warnings.warn("IDs do not match.")

    @classmethod
    def instantiate_from_csv(cls, filename: str, other_file: str):
        #open csv and create list of all rows
        with open(filename, encoding="utf8") as f:
            reader = csv.DictReader(f)
            rows_of_patients = list(reader)
            #for line in csv create object
            for row in rows_of_patients:
                Patient(
                    DonorID = row['Donor ID'],
                    ABeta40 = float(row['ABeta40 pg/ug']),
                    ABeta42 = float(row['ABeta42 pg/ug']),
                    tTau = float(row['tTAU pg/ug']),
                    pTau = float(row['pTAU pg/ug'])
                )
            Patient.all_patients.sort(key = Patient.get_id)
            Patient.combine_data(other_file)

    @classmethod
    def calculate_normalized_values(cls):
        """Calculate z-scores for ttau and ptau across all patients."""
        # Collect all ttau and ptau values
        ttau_values = [p.tTau for p in cls.all_patients if p.tTau is not Nor
        ptau_values = [p.pTau for p in cls.all_patients if p.pTau is not Nor

        # Calculate means and standard deviations
        ttau_mean = statistics.mean(ttau_values)
        ttau_std = statistics.stdev(ttau_values)
        ptau_mean = statistics.mean(ptau_values)
        ptau_std = statistics.stdev(ptau_values)
```

```
        # Calculate z-scores for each patient
        for patient in cls.all_patients:
            if patient.tTau is not None:
                patient.ttau_zscore = (patient.tTau - ttau_mean) / ttau_std
            if patient.pTau is not None:
                patient.ptau_zscore = (patient.pTau - ptau_mean) / ptau_std


#to plot astherosclerosis
    @classmethod
    def plot_atherosclerosis_tau(cls):
        athero_groups = {}

        for patient in cls.all_patients:
            if patient.atherosclerosis is not None and patient.atherosclero
                if patient.atherosclerosis not in athero_groups:
                    athero_groups[patient.atherosclerosis] = []
                # Calculate normalized average of ttau and ptau
                normalized_avg = (patient.ttau_zscore + patient.ptau_zscore)
                athero_groups[patient.atherosclerosis].append(normalized_avg

        print("Normalized Average tau (ttau + ptau)/2:", athero_groups)

        categories = list(athero_groups.keys())
        normalized_avg_data = [athero_groups[cat] for cat in categories]

        plt.figure(figsize=(8, 6))
        plt.boxplot(normalized_avg_data, tick_labels=categories)
        plt.xlabel("Atherosclerosis Status")
        plt.ylabel("Normalized Average tau (Z-score)")
        plt.title("Normalized Average tau Levels by Atherosclerosis Status")
        plt.show()

    @classmethod
    def _group_normalized_tau_avg_by_athero(cls):
        """
        Build {category: [ (Aβ40_z + Aβ42_z)/2, ... ]} and return (ordered_c
        """
        groups = {}
        for p in cls.all_patients:
            if p.atherosclerosis and p.ttau_zscore is not None and p.ptau_zs
                cat = cls._athero_label(p.atherosclerosis)
                if not cat:
                    continue
                norm_avg = (p.ttau_zscore + p.ptau_zscore) / 2
                groups.setdefault(cat, []).append(norm_avg)

        # keep a logical order if these exist, then append anything else
        ordered = [c for c in ["None", "Mild", "Moderate", "Severe"] if c in
        extras = sorted([c for c in groups.keys() if c not in ordered])
        ordered.extend(extras)
        return ordered, groups

if __name__ == "__main__":
    # Load data
    Patient.instantiate_from_csv('UpdatedLuminex.csv', 'UpdatedMetaData.csv'
```

```
# Calculate normalized values (z-scores)
Patient.calculate_normalized_values()

# Create atherosclerosis vs tau box plot
Patient.plot_atherosclerosis_tau()
```

```
Normalized Average tau (ttau + ptau)/2: {'Mild': [0.22861385426594893, -0.11
888047089114845, 0.5043422587204421, -0.8061235612761009, 0.354687930897724
3, 0.1578925761831415, 0.07517362353762501, 0.20927611936350618, 4.435541489
2514045, 0.8613581653214407, -0.09192402107917202, -0.26975392637818185, 0.0
4213672251912004, -0.14729036825260097, -0.6247326491282732, -0.129741422330
35493, -0.29472327315408076, -0.4721608563902231, 0.12038110245444789, 0.184
30010469902613, -0.1565877549493846, 0.1924829525653993], 'Moderate': [-0.42
542684899935873, -0.5543125923274691, 1.2476709925561766, -0.356900217972851
2, -0.526789897590817, -0.37775455597950036, -0.030464341251016464, -0.37407
73300206349, 1.418539636636767, -0.44292214811020425, -0.3214266571779508,
0.5954617119380355, 1.3778251209161443, -0.3546804269172149, -0.266206047211
3933, -0.01022958259776488, -0.6627588497000165, -0.06741151964406703, -0.27
76285017761185, -0.07601206922703291, -0.524653739877493, -0.540394049405277
7, 0.4395592694462096, 0.6326141420078532, 2.124804323143995, -0.25503171144
59015, -0.281944918600733, -0.25239182867034954, -0.17471733364236292, 0.142
44198352269782, -0.5739066274355598, -0.6961152051804005, -0.636373699033481
8, -0.6264281627625218, -0.35870076979132703, -0.8378369152910634, -0.685599
7940288444, 0.31714018491735163, -0.46532490362617585, -0.2889739073541411,
0.48193348125308555, 0.34489864272435244, 0.1574608669049436, 0.354340338644
1804, 0.4715666316152153, 0.0987718559828491, -0.014931079059828462, -0.3069
0766372390255, 0.11022370981236118, 1.1825278326970226, -0.3043249304897257
5], 'None': [-0.30925181676323177, -0.17202854541576812, 0.4105055155460163
3, -0.41626144065142956, -0.36469437045040787, -0.36034481939576957, -0.3232
8221265559937], 'Severe': [-0.36901908808041706, -0.33774034563260613, -0.43
26673624882311, -0.12770600875899768]}
```



The following code will create a box and whisker plot for the ABeta40 and ABeta42 levels in relation to the severity of atherosclerosis

In [ ]:
```
import csv
import warnings
import matplotlib.pyplot as plt
import statistics
from scipy import stats
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

class Patient:

    all_patients = []
    death_age = []
    education_lvl = {}

    def __init__(self, DonorID, ABeta40: float , ABeta42: float, tTau: float
        self.DonorID = DonorID
        self.ABeta40 = ABeta40
        self.ABeta42 = ABeta42
        self.tTau = tTau
        self.pTau = pTau
        self.sex = None
        self.death_age = None
        self.ed_lvl = None
```

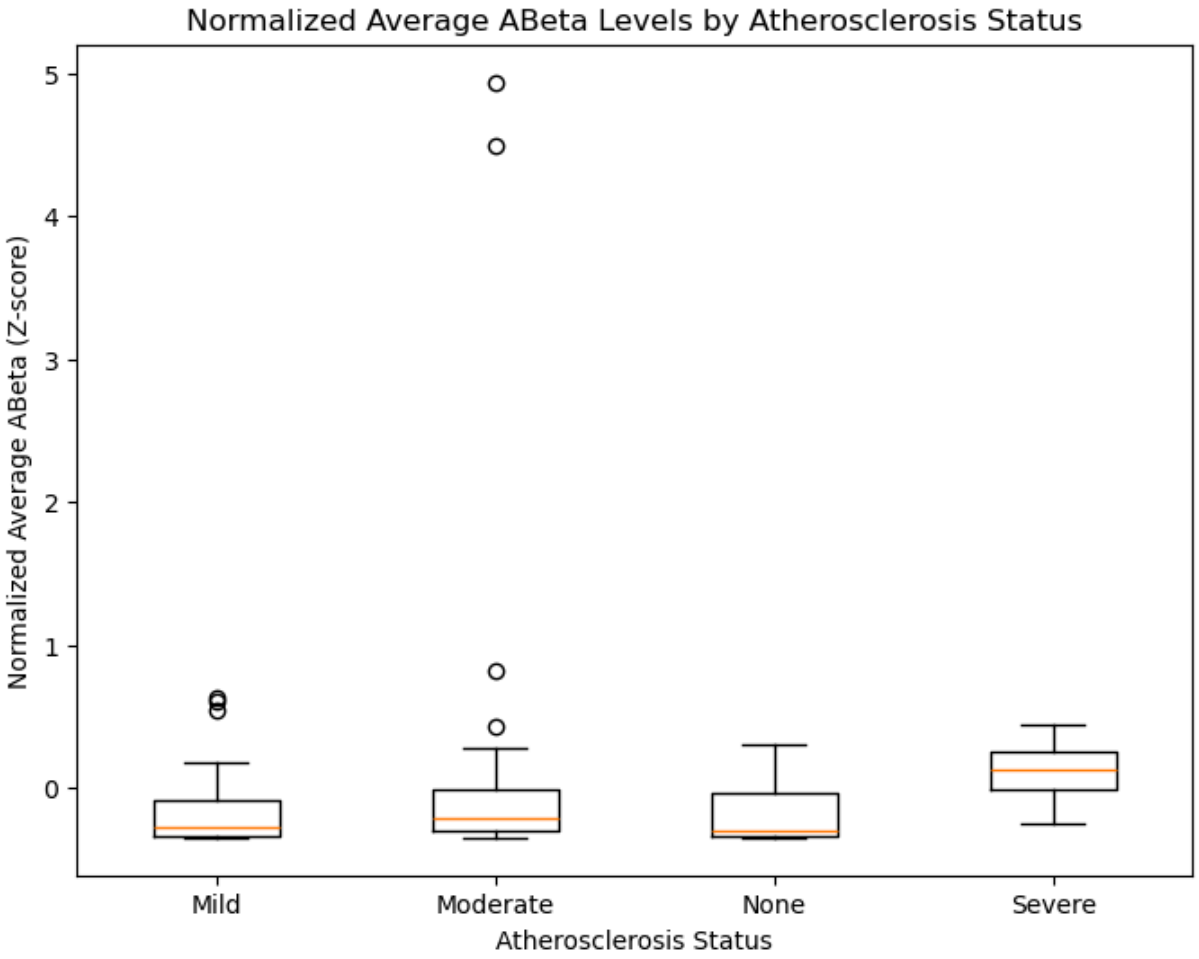
```
        self.cog_stat = None
        self.age_symp_on = None
        self.age_diag = None
        self.head_inj = None
        self.thal_score = None
        self.atherosclerosis = None
        # Add normalized values as attributes
        self.abeta40_zscore = None
        self.abeta42_zscore = None
        Patient.all_patients.append(self)

    def __repr__(self):
        return f"{self.DonorID} | sex: {self.sex} | ABeta40 {self.ABeta40} (

    def get_id(self):
        return self.DonorID

    def get_ABeta42(self):
        return self.ABeta42

    def get_thal(self):
        return self.thal_score

    def get_death_age(self):
        return self.death_age


    @classmethod
    def combine_data(cls, filename: str):
        with open(filename, encoding="utf8") as f:
            reader = csv.DictReader(f)
            rows_of_patients = list(reader)
            for row in range(len(rows_of_patients)):
                if Patient.all_patients[row].DonorID == rows_of_patients[row
                    if rows_of_patients[row]["Sex"] != "":
                        Patient.all_patients[row].sex = rows_of_patients[row

                    if rows_of_patients[row]["Age at Death"] != "":
                        Patient.all_patients[row].death_age = int(rows_of_pa

                    if rows_of_patients[row]["Highest level of education"] !
                        Patient.all_patients[row].ed_lvl = rows_of_patients[

                    if rows_of_patients[row]["Cognitive Status"] != "":
                        Patient.all_patients[row].cog_stat = rows_of_patient

                    if rows_of_patients[row]["Age of onset cognitive symptom
                        Patient.all_patients[row].age_symp_on = int(rows_of_

                    if rows_of_patients[row]["Age of Dementia diagnosis"] !=
                        Patient.all_patients[row].age_diag = int(rows_of_pat

                    if rows_of_patients[row]["Known head injury"] != "":
                        Patient.all_patients[row].head_inj = rows_of_patient

                    if rows_of_patients[row]["Thal"] != "":
```

```
                        Patient.all_patients[row].thal_score = int(rows_of_p

                    # atherosclerosis part
                    if rows_of_patients[row]["Atherosclerosis"] != "":
                        Patient.all_patients[row].atherosclerosis = rows_of_

                else:
                    warnings.warn("IDs do not match.")

    @classmethod
    def instantiate_from_csv(cls, filename: str, other_file: str):
        #open csv and create list of all rows
        with open(filename, encoding="utf8") as f:
            reader = csv.DictReader(f)
            rows_of_patients = list(reader)
            #for line in csv create object
            for row in rows_of_patients:
                Patient(
                    DonorID = row['Donor ID'],
                    ABeta40 = float(row['ABeta40 pg/ug']),
                    ABeta42 = float(row['ABeta42 pg/ug']),
                    tTau = float(row['tTAU pg/ug']),
                    pTau = float(row['pTAU pg/ug'])
                )
            Patient.all_patients.sort(key = Patient.get_id)
            Patient.combine_data(other_file)

    @classmethod
    def calculate_normalized_values(cls):
        """Calculate z-scores for ABeta40 and ABeta42 across all patients."""
        # Collect all ABeta40 and ABeta42 values
        abeta40_values = [p.ABeta40 for p in cls.all_patients if p.ABeta40 i
        abeta42_values = [p.ABeta42 for p in cls.all_patients if p.ABeta42 i

        # Calculate means and standard deviations
        abeta40_mean = statistics.mean(abeta40_values)
        abeta40_std = statistics.stdev(abeta40_values)
        abeta42_mean = statistics.mean(abeta42_values)
        abeta42_std = statistics.stdev(abeta42_values)

        # Calculate z-scores for each patient
        for patient in cls.all_patients:
            if patient.ABeta40 is not None:
                patient.abeta40_zscore = (patient.ABeta40 - abeta40_mean) /
            if patient.ABeta42 is not None:
                patient.abeta42_zscore = (patient.ABeta42 - abeta42_mean) /


#to plot astherosclerosis
    @classmethod
    def plot_atherosclerosis_abeta(cls):
        athero_groups = {}

        for patient in cls.all_patients:
            if patient.atherosclerosis is not None and patient.atherosclero
                if patient.atherosclerosis not in athero_groups:
```

```
                athero_groups[patient.atherosclerosis] = []
            # Calculate normalized average of ABeta40 and ABeta42
            normalized_avg = (patient.abeta40_zscore + patient.abeta42_z
            athero_groups[patient.atherosclerosis].append(normalized_avg

    print("Normalized Average ABeta (ABeta40 + ABeta42)/2:", athero_grou

    categories = list(athero_groups.keys())
    normalized_avg_data = [athero_groups[cat] for cat in categories]

    plt.figure(figsize=(8, 6))
    plt.boxplot(normalized_avg_data, tick_labels=categories)
    plt.xlabel("Atherosclerosis Status")
    plt.ylabel("Normalized Average ABeta (Z-score)")
    plt.title("Normalized Average ABeta Levels by Atherosclerosis Status
    plt.show()


#  one-way ANOVA Test
@classmethod
def _athero_label(cls, raw):
    """Normalize atherosclerosis labels to a consistent set."""
    if raw is None:
        return None
    s = str(raw).strip().lower()
    if s in {"none", "no", "0"}:
        return "None"
    if s in {"mild", "1"}:
        return "Mild"
    if s in {"moderate", "2"}:
        return "Moderate"
    if s in {"severe", "severe atherosclerosis", "3"}:
        return "Severe"
    # fallback: keep whatever text but make it pretty
    return s.title()

@classmethod
def _group_normalized_abeta_avg_by_athero(cls):
    """
    Build {category: [ (ABeta40_z + ABeta42_z)/2, ... ]} and return (ord
    """
    groups = {}
    for p in cls.all_patients:
        if p.atherosclerosis and p.abeta40_zscore is not None and p.abet
            cat = cls._athero_label(p.atherosclerosis)
            if not cat:
                continue
            norm_avg = (p.abeta40_zscore + p.abeta42_zscore) / 2
            groups.setdefault(cat, []).append(norm_avg)

    # keep a logical order if these exist, then append anything else
    ordered = [c for c in ["None", "Mild", "Moderate", "Severe"] if c in
    extras = sorted([c for c in groups.keys() if c not in ordered])
    ordered.extend(extras)
    return ordered, groups
```

```
if __name__ == "__main__":
    # Load data
    Patient.instantiate_from_csv('UpdatedLuminex.csv', 'UpdatedMetaData.csv'

    # Calculate normalized values (z-scores)
    Patient.calculate_normalized_values()

    # Create atherosclerosis vs ABeta box plot
    Patient.plot_atherosclerosis_abeta()
```

```
Normalized Average ABeta (ABeta40 + ABeta42)/2: {'Mild': [-0.342123679110013
4, -0.3357240333169674, 0.17669405835341526, -0.27192354320828604, -0.185300
8900397397, 0.5427135432441678, 0.004971526591620465, -0.08667874588218724,
-0.34444229228592577, 0.6011687073860029, 0.626955285786005, -0.333363125558
99417, -0.3108309224310804, -0.3435596053640201, -0.276048276129496, -0.2227
934711595548, -0.3179684010249769, -0.3444994984086457, -0.1012961476011489
5, -0.3434182959605542, -0.20854791901263434, -0.27460861040380735], 'Modera
te': [-0.3447769401821026, -0.01016701288573367, -0.10373823262304031, -0.26
2998551089927, -0.25053285918341933, -0.2670148219971483, 4.935296995966986,
-0.2477895667491072, -0.006583186272680465, -0.28117098274379015, -0.2480431
8958340812, -0.20383648559478995, 0.20314916409852002, 0.1295008980583179, -
0.3083334075906661, -0.34346689203383207, -0.009994912149982951, 0.102714461
50816758, -0.0693229575052104, -0.2508611746859628, -0.2993622402149506, 0.4
3514772059597395, 0.2822387326428123, 4.4909453153912375, -0.014033404197053
79, -0.3439716444800456, -0.34397199537052126, -0.34314410783173355, -0.3174
1071251217035, -0.2010824554672716, 0.030295769321124294, -0.213586241103378
48, -0.3098306559753046, -0.3451720813719924, -0.21944598309199725, -0.33690
69036244595, -0.34487793551476165, 0.8198669801924495, -0.07924780628001843,
-0.34427691829377594, -0.27928490567691566, 0.1964397562185763, -0.228541263
70496092, 0.21096703016142965, 0.13518607847573613, -0.08542326926140043, -
0.048163150902407886, -0.10402332878925838, -0.24498903537362265, -0.0815436
8839321344, -0.3450808321551466], 'None': [-0.2926127745601171, -0.031029112
86593002, 0.30685022244355076, -0.029154364409940063, -0.3448070957699031, -
0.3213562037309318, -0.3436752787640659], 'Severe': [0.4471443435080426, -0.
24367603672003513, 0.06117572638592338, 0.19201773984605827]}
```



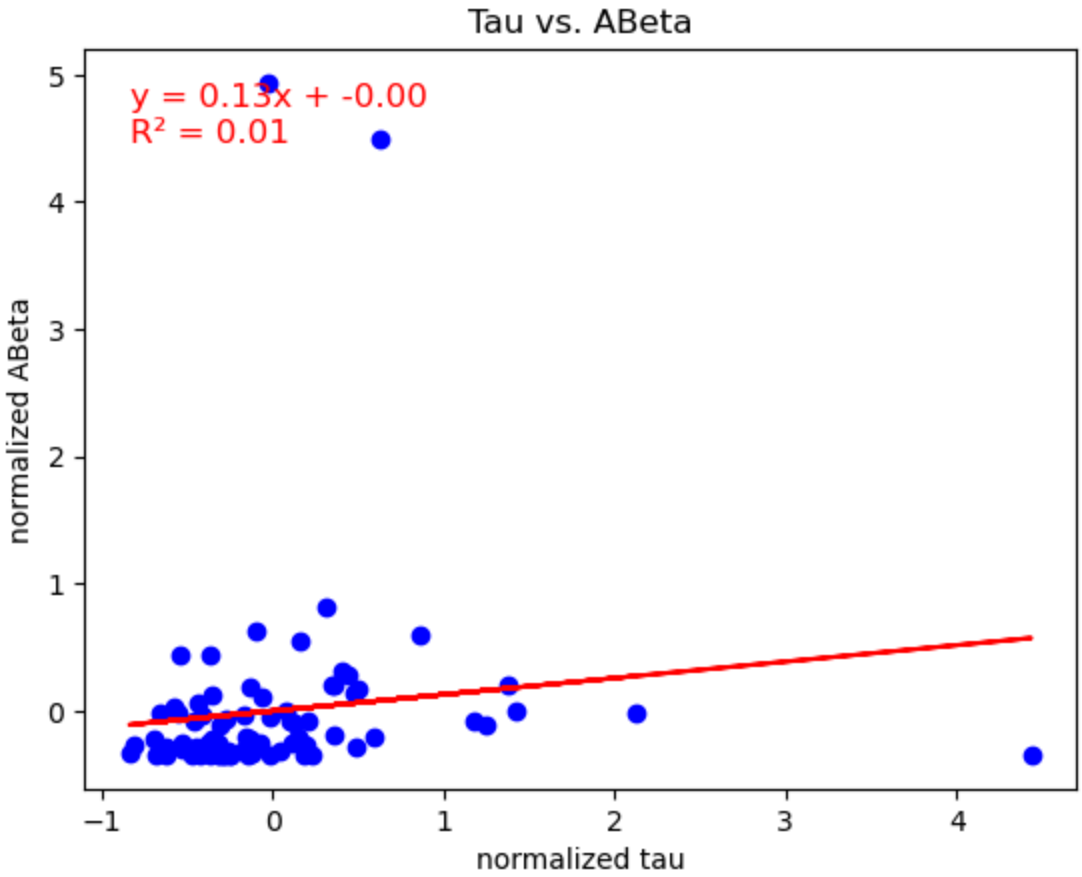
the following code creates a scatterplot of the normalized tau and ABeta levels

In [ ]:
```
import csv
import matplotlib.pyplot as plt
import statistics
import pandas as pd
from sklearn.linear_model import LinearRegression

class Patient:

    all_patients = []

    def __init__(self, DonorID, ABeta40: float, ABeta42: float, tTau: float,
        self.DonorID = DonorID
        self.ABeta40 = ABeta40
        self.ABeta42 = ABeta42
        self.tTau = tTau
        self.pTau = pTau
        Patient.all_patients.append(self)

    def get_id(self):
        return self.DonorID

    @classmethod
    def instantiate_from_csv(cls, filename: str):
        with open(filename, encoding="utf8") as f:
```

```
            reader = csv.DictReader(f)
            rows_of_patients = list(reader)
            for row in rows_of_patients:
                Patient(
                    DonorID = row['Donor ID'],
                    ABeta40 = float(row['ABeta40 pg/ug']),
                    ABeta42 = float(row['ABeta42 pg/ug']),
                    tTau = float(row['tTAU pg/ug']),
                    pTau = float(row['pTAU pg/ug'])
                )
            Patient.all_patients.sort(key = Patient.get_id)


# Main execution
if __name__ == "__main__":
    # Load data
    Patient.instantiate_from_csv('UpdatedLuminex.csv')

    ABeta42 = []
    ABeta40 = []
    ABeta_norm = []
    Ttau = []
    Ptau = []
    tau_norm = []

    # Collect all biomarker values
    for patient in Patient.all_patients:
        ABeta42.append(patient.ABeta42)
        ABeta40.append(patient.ABeta40)
        Ttau.append(patient.tTau)
        Ptau.append(patient.pTau)

    # Calculate means and standard deviations for all biomarkers
    mean_ABeta40 = statistics.mean(ABeta40)
    mean_ABeta42 = statistics.mean(ABeta42)
    mean_Ttau = statistics.mean(Ttau)
    mean_Ptau = statistics.mean(Ptau)

    stdev_ABeta40 = statistics.stdev(ABeta40)
    stdev_ABeta42 = statistics.stdev(ABeta42)
    stdev_Ttau = statistics.stdev(Ttau)
    stdev_Ptau = statistics.stdev(Ptau)

    # Normalize each biomarker and create combined normalized values
    for patient in Patient.all_patients:
        ABeta40_zscore = (patient.ABeta40 - mean_ABeta40) / stdev_ABeta40
        ABeta42_zscore = (patient.ABeta42 - mean_ABeta42) / stdev_ABeta42
        Ttau_zscore = (patient.tTau - mean_Ttau) / stdev_Ttau
        Ptau_zscore = (patient.pTau - mean_Ptau) / stdev_Ptau

        ABeta_combined = (ABeta40_zscore + ABeta42_zscore) / 2
        Tau_combined = (Ttau_zscore + Ptau_zscore) / 2

        ABeta_norm.append(ABeta_combined)
        tau_norm.append(Tau_combined)
```

```
    # Create a DataFrame
    df = pd.DataFrame({
        'tau': tau_norm,
        'ABeta': ABeta_norm
    })

    # Write to CSV
    df.to_csv('patient_data.csv', index=False)
    print("CSV file 'patient_data.csv' has been created.")

    # Read CSV for regression
    df = pd.read_csv("patient_data.csv")
    x = df["tau"].values.reshape(-1, 1)
    y = df["ABeta"].values

    model = LinearRegression()
    model.fit(x, y)
    slope = model.coef_[0]
    intercept = model.intercept_
    r2 = model.score(x, y)

    # Making the scatterplot
    plt.scatter(x, y, color='blue')
    plt.plot(x, model.predict(x), color="red")

    # Annotate equation
    equation = f"y = {slope:.2f}x + {intercept:.2f}\nR² = {r2:.2f}"
    plt.text(x.min(), y.max(), equation, color="red", fontsize=12, verticala

    # Annotate scatterplot with labels and title
    plt.xlabel("normalized tau")
    plt.ylabel("normalized ABeta")
    plt.title("Tau vs. ABeta")
    plt.show()
```

```
CSV file 'patient_data.csv' has been created.
```



the following code was written to conduct a t-test on the normalized values of "None" and "Severe" categories of atherosclerosis. We know an ANOVA should be used for comparing >2 categories but a t-test was written for practice.

In [ ]:
```
import csv
import statistics
from scipy import stats

class Patient:

    all_patients = []

    def __init__(self, DonorID, ABeta40: float, ABeta42: float, tTau: float,
        self.DonorID = DonorID
        self.ABeta40 = ABeta40
        self.ABeta42 = ABeta42
        self.tTau = tTau
        self.pTau = pTau
        self.atherosclerosis = None
        self.ABeta40_zscore = None
        self.ABeta42_zscore = None
        Patient.all_patients.append(self)

    def get_id(self):
        return self.DonorID

    @classmethod
```

```
    def normalize_biomarkers(cls):
        """
        Apply Z-score normalization to ABeta40 and ABeta42 biomarkers.
        """
        abeta40_values = [patient.ABeta40 for patient in cls.all_patients]
        abeta42_values = [patient.ABeta42 for patient in cls.all_patients]

        mean_abeta40 = statistics.mean(abeta40_values)
        stdev_abeta40 = statistics.stdev(abeta40_values)

        mean_abeta42 = statistics.mean(abeta42_values)
        stdev_abeta42 = statistics.stdev(abeta42_values)

        for patient in cls.all_patients:
            patient.ABeta40_zscore = (patient.ABeta40 - mean_abeta40) / stde
            patient.ABeta42_zscore = (patient.ABeta42 - mean_abeta42) / stde

    @classmethod
    def combine_data(cls, filename: str):
        with open(filename, encoding="utf8") as f:
            reader = csv.DictReader(f)
            rows_of_patients = list(reader)
            for row in range(len(rows_of_patients)):
                if Patient.all_patients[row].DonorID == rows_of_patients[row
                    if rows_of_patients[row]["Atherosclerosis"] != "":
                        Patient.all_patients[row].atherosclerosis = rows_of_

    @classmethod
    def instantiate_from_csv(cls, filename: str, other_file: str):
        with open(filename, encoding="utf8") as f:
            reader = csv.DictReader(f)
            rows_of_patients = list(reader)
            for row in rows_of_patients:
                Patient(
                    DonorID = row['Donor ID'],
                    ABeta40 = float(row['ABeta40 pg/ug']),
                    ABeta42 = float(row['ABeta42 pg/ug']),
                    tTau = float(row['tTAU pg/ug']),
                    pTau = float(row['pTAU pg/ug'])
                )
            Patient.all_patients.sort(key = Patient.get_id)
            Patient.combine_data(other_file)


# Main execution
if __name__ == "__main__":
    # Load data
    Patient.instantiate_from_csv('UpdatedLuminex.csv', 'UpdatedMetaData.csv'

    # Apply Z-score normalization
    Patient.normalize_biomarkers()

    # T-TEST where the difference between "none" and "severe" are compared
    print("\n" + "="*50)
    print("T-TEST RESULTS: None vs Severe Atherosclerosis")
    print("="*50)
```

```
none_group = []
severe_group = []

for patient in Patient.all_patients:
    if patient.atherosclerosis is not None and patient.atherosclerosis !
        if patient.ABeta40_zscore is not None and patient.ABeta42_zscore
            normalized_avg = (patient.ABeta40_zscore + patient.ABeta42_z

            athero_lower = patient.atherosclerosis.lower()
            if athero_lower in ['none', 'no', '0']:
                none_group.append(normalized_avg)
            elif athero_lower in ['severe', 'severe atherosclerosis', '3
                severe_group.append(normalized_avg)

print(f"None group: n = {len(none_group)}")
print(f"Severe group: n = {len(severe_group)}")

if len(none_group) > 0:
    print(f"None group mean: {statistics.mean(none_group):.3f}")
    print(f"None group std: {statistics.stdev(none_group):.3f}" if len(r

if len(severe_group) > 0:
    print(f"Severe group mean: {statistics.mean(severe_group):.3f}")
    print(f"Severe group std: {statistics.stdev(severe_group):.3f}" if l

if len(none_group) >= 1 and len(severe_group) >= 1:
    # Perform independent two-sample t-test
    t_stat, p_value = stats.ttest_ind(none_group, severe_group)

    print(f"\nT-statistic: {t_stat:.4f}")
    print(f"p-value: {p_value:.4f}")

    # Determine pass/fail based on a=0.05
    alpha = 0.05
    if p_value < alpha:
        result = "FAIL"
        print(f"\nResult: {result}")
        print("There is a statistically significant difference between N
    else:
        result = "PASS"
        print(f"\nResult: {result}")
        print("There is NO statistically significant difference between
```

```
==================================================
T-TEST RESULTS: None vs Severe Atherosclerosis
==================================================
None group: n = 7
Severe group: n = 4
None group mean: -0.151
None group std: 0.246
Severe group mean: 0.114
Severe group std: 0.287

T-statistic: -1.6234
p-value: 0.1390

Result: PASS
There is NO statistically significant difference between None and Severe gro
ups.
```

The following code was written to run a one-way ANOVA test on both normalized amyloid-beta and tau protein levels across atherosclerosis groups.

In [ ]:
```python
import csv
import matplotlib.pyplot as plt
import statistics
import pandas as pd
from sklearn.linear_model import LinearRegression
from scipy import stats

class Patient:

    all_patients = []

    def __init__(self, DonorID, ABeta40: float, ABeta42: float, tTau: float,
        self.DonorID = DonorID
        self.ABeta40 = ABeta40
        self.ABeta42 = ABeta42
        self.tTau = tTau
        self.pTau = pTau
        self.atherosclerosis = atherosclerosis
        self.ABeta40_zscore = None
        self.ABeta42_zscore = None
        Patient.all_patients.append(self)

    def get_id(self):
        return self.DonorID

    @classmethod
    def instantiate_from_csv(cls, luminex_file: str, metadata_file: str):
        # Read metadata first to get atherosclerosis info
        metadata = {}
        with open(metadata_file, encoding="utf8") as f:
            reader = csv.DictReader(f)
            for row in reader:
                donor_id = row.get('Donor ID') or row.get('DonorID')
                atherosclerosis = row.get('Atherosclerosis') or row.get('ath
                if donor_id:
```

```
                metadata[donor_id] = atherosclerosis

        # Read luminex data and create patients
        with open(luminex_file, encoding="utf8") as f:
            reader = csv.DictReader(f)
            rows_of_patients = list(reader)
            for row in rows_of_patients:
                donor_id = row['Donor ID']
                Patient(
                    DonorID=donor_id,
                    ABeta40=float(row['ABeta40 pg/ug']),
                    ABeta42=float(row['ABeta42 pg/ug']),
                    tTau=float(row['tTAU pg/ug']),
                    pTau=float(row['pTAU pg/ug']),
                    atherosclerosis=metadata.get(donor_id)
                )
            Patient.all_patients.sort(key=Patient.get_id)

    @classmethod
    def normalize_biomarkers(cls):
        """Calculate z-scores for ABeta40 and ABeta42"""
        # Get all values
        abeta40_values = [p.ABeta40 for p in cls.all_patients if p.ABeta40 i
        abeta42_values = [p.ABeta42 for p in cls.all_patients if p.ABeta42 i

        if not abeta40_values or not abeta42_values:
            print("Warning: No valid ABeta values found for normalization")
            return

        # Calculate means and standard deviations
        mean_abeta40 = statistics.mean(abeta40_values)
        stdev_abeta40 = statistics.stdev(abeta40_values)
        mean_abeta42 = statistics.mean(abeta42_values)
        stdev_abeta42 = statistics.stdev(abeta42_values)

        # Apply z-score normalization to each patient
        for p in cls.all_patients:
            if p.ABeta40 is not None and stdev_abeta40 > 0:
                p.ABeta40_zscore = (p.ABeta40 - mean_abeta40) / stdev_abeta4
            if p.ABeta42 is not None and stdev_abeta42 > 0:
                p.ABeta42_zscore = (p.ABeta42 - mean_abeta42) / stdev_abeta4

    @classmethod
    def _athero_label(cls, raw):
        """Normalize atherosclerosis labels to a consistent set."""
        if raw is None:
            return None
        s = str(raw).strip().lower()
        if s in {"none", "no", "0"}:
            return "None"
        if s in {"mild", "1"}:
            return "Mild"
        if s in {"moderate", "2"}:
            return "Moderate"
        if s in {"severe", "severe atherosclerosis", "3"}:
            return "Severe"
```

```
        # fallback: keep whatever text but make it pretty
        return s.title()

    @classmethod
    def _group_normalized_abeta_avg_by_athero(cls):
        """
        Build {category: [ (Aβ40_z + Aβ42_z)/2, ... ]} and return (ordered_c
        """
        groups = {}
        for p in cls.all_patients:
            if p.atherosclerosis and p.ABeta40_zscore is not None and p.ABet
                cat = cls._athero_label(p.atherosclerosis)
                if not cat:
                    continue
                norm_avg = (p.ABeta40_zscore + p.ABeta42_zscore) / 2
                groups.setdefault(cat, []).append(norm_avg)

        # keep a logical order if these exist, then append anything else
        ordered = [c for c in ["None", "Mild", "Moderate", "Severe"] if c in
        extras = sorted([c for c in groups.keys() if c not in ordered])
        ordered.extend(extras)
        return ordered, groups

    @classmethod
    def anova_atherosclerosis_beta(cls):
        """
        One-way ANOVA on normalized avg Aβ across atherosclerosis categories
        Prints F, p, dfs, eta^2; optionally runs Tukey HSD if statsmodels is
        """
        categories, groups = cls._group_normalized_abeta_avg_by_athero()
        samples = [groups[c] for c in categories]

        ns = [len(s) for s in samples]

        print("\n" + "="*60)
        print("ONE-WAY ANOVA: Normalized Avg Aβ by Atherosclerosis Category"
        print("="*60)
        for c, n in zip(categories, ns):
            if n > 1:
                mean = statistics.mean(groups[c])
                sd = statistics.stdev(groups[c])
                print(f"{c:>9}: n={n:>3} | mean={mean:>7.3f} | sd={sd:>7.3f}
            elif n == 1:
                print(f"{c:>9}: n=  1 | mean={groups[c][0]:>7.3f} | sd=   N/
            else:
                print(f"{c:>9}: n=  0")

        # need ≥2 non-empty groups
        nonempty = [s for s in samples if len(s) > 0]
        if len(nonempty) < 2:
            print("\nNot enough non-empty groups to run ANOVA.")
            return

        # One-way ANOVA
        F, p = stats.f_oneway(*nonempty)
```

```
        # degrees of freedom
        k_eff = len(nonempty)
        df_between = k_eff - 1
        df_within = sum(len(s) for s in nonempty) - k_eff

        # effect size (eta^2)
        overall = [x for s in nonempty for x in s]
        grand_mean = statistics.mean(overall)
        ss_between = sum(len(s) * (statistics.mean(s) - grand_mean) ** 2 for
        ss_within = sum(sum((x - statistics.mean(s)) ** 2 for x in s) for s
        ss_total = ss_between + ss_within
        eta_sq = ss_between / ss_total if ss_total > 0 else float('nan')

        print("\n--- ANOVA ---")
        print(f"F({df_between}, {df_within}) = {F:.4f},  p = {p:.6f}")
        print(f"Effect size (eta^2) = {eta_sq:.4f}")
        if p > 0.05:
            print("Result: FAILS to reject the null hypothesis.")
            print("There is NO statistically significant difference between

        else:
            print("Result: REJECTS the null hypothesis.")
            print("There IS a statistically significant difference between g
            # Optional: Tukey HSD post-hoc test if statsmodels is available
            try:
                from statsmodels.stats.multicomp import pairwise_tukeyhsd
                import numpy as np

                # Prepare data for Tukey HSD
                data = []
                labels = []
                for c in categories:
                    data.extend(groups[c])
                    labels.extend([c] * len(groups[c]))

                tukey = pairwise_tukeyhsd(endog=np.array(data), groups=np.ar
                print("\nTukey HSD post-hoc test results:")
                print(tukey)
            except ImportError:
                print("\nStatsmodels not available; skipping Tukey HSD post-

    @classmethod
    def _group_normalized_tau_avg_by_athero(cls):
        """
        Build {category: [ (tTau_z + pTau_z)/2, ... ]} and return (ordered_c
        """
        groups = {}
        # Calculate mean and stdev for tTau and pTau
        tTau_values = [p.tTau for p in cls.all_patients]
        pTau_values = [p.pTau for p in cls.all_patients]
        if not tTau_values or not pTau_values:
            return [], {}
        mean_tTau = statistics.mean(tTau_values)
        stdev_tTau = statistics.stdev(tTau_values)
        mean_pTau = statistics.mean(pTau_values)
        stdev_pTau = statistics.stdev(pTau_values)
```

```
        for p in cls.all_patients:
            if p.atherosclerosis and p.tTau is not None and p.pTau is not No
                cat = cls._athero_label(p.atherosclerosis)
                if not cat:
                    continue
                tTau_z = (p.tTau - mean_tTau) / stdev_tTau if stdev_tTau > 0
                pTau_z = (p.pTau - mean_pTau) / stdev_pTau if stdev_pTau > 0
                norm_avg = (tTau_z + pTau_z) / 2
                groups.setdefault(cat, []).append(norm_avg)
        ordered = [c for c in ["None", "Mild", "Moderate", "Severe"] if c in
        extras = sorted([c for c in groups.keys() if c not in ordered])
        ordered.extend(extras)
        return ordered, groups

    @classmethod
    def anova_tau(cls):
        """
        One-way ANOVA on normalized avg Tau across atherosclerosis categorie
        Prints F, p, dfs, eta^2; optionally runs Tukey HSD if statsmodels is
        """
        categories, groups = cls._group_normalized_tau_avg_by_athero()
        samples = [groups[c] for c in categories]

        ns = [len(s) for s in samples]

        print("\n" + "="*60)
        print("ONE-WAY ANOVA: Normalized Avg Tau by Atherosclerosis Category
        print("="*60)
        for c, n in zip(categories, ns):
            if n > 1:
                mean = statistics.mean(groups[c])
                sd = statistics.stdev(groups[c])
                print(f"{c:>9}: n={n:>3} | mean={mean:>7.3f} | sd={sd:>7.3f}
            elif n == 1:
                print(f"{c:>9}: n=  1 | mean={groups[c][0]:>7.3f} | sd=   N/
            else:
                print(f"{c:>9}: n=  0")

        # need ≥2 non-empty groups
        nonempty = [s for s in samples if len(s) > 0]
        if len(nonempty) < 2:
            print("\nNot enough non-empty groups to run ANOVA.")
            return

        # One-way ANOVA
        F, p = stats.f_oneway(*nonempty)

        # degrees of freedom
        k_eff = len(nonempty)
        df_between = k_eff - 1
        df_within = sum(len(s) for s in nonempty) - k_eff

        # effect size (eta^2)
        overall = [x for s in nonempty for x in s]
        grand_mean = statistics.mean(overall)
        ss_between = sum(len(s) * (statistics.mean(s) - grand_mean) ** 2 for
```

```
        ss_within = sum(sum((x - statistics.mean(s)) ** 2 for x in s) for s
        ss_total = ss_between + ss_within
        eta_sq = ss_between / ss_total if ss_total > 0 else float('nan')

        print("\n--- ANOVA ---")
        print(f"F({df_between}, {df_within}) = {F:.4f},  p = {p:.6f}")
        print(f"Effect size (eta^2) = {eta_sq:.4f}")

        if p > 0.05:
            print("Result: FAILS to reject the null hypothesis.")
            print("There is NO statistically significant difference between
        else:
            print("Result: REJECTS the null hypothesis.")
            print("There IS a statistically significant difference between g
            # Optional: Tukey HSD post-hoc test if statsmodels is available
            try:
                from statsmodels.stats.multicomp import pairwise_tukeyhsd
                import numpy as np

                # Prepare data for Tukey HSD
                data = []
                labels = []
                for c in categories:
                    data.extend(groups[c])
                    labels.extend([c] * len(groups[c]))

                tukey = pairwise_tukeyhsd(endog=np.array(data), groups=np.ar
                print("\nTukey HSD post-hoc test results:")
                print(tukey)
            except ImportError:
                print("\nStatsmodels not available; skipping Tukey HSD post-


# Main execution
if __name__ == "__main__":
    # Load data
    Patient.instantiate_from_csv('UpdatedLuminex.csv', 'UpdatedMetaData.csv'

    # Apply Z-score normalization
    Patient.normalize_biomarkers()

    # Calling ANOVA tests
    Patient.anova_atherosclerosis_beta()
    Patient.anova_tau()
    print("\n")
```

```
============================================================
ONE-WAY ANOVA: Normalized Avg Aβ by Atherosclerosis Category
============================================================
     None: n=  7 | mean= -0.151 | sd=  0.246
     Mild: n= 22 | mean= -0.122 | sd=  0.319
 Moderate: n= 51 | mean=  0.065 | sd=  0.978
   Severe: n=  4 | mean=  0.114 | sd=  0.287

--- ANOVA ---
F(3, 80) = 0.3965,  p = 0.755843
Effect size (eta^2) = 0.0147
Result: FAILS to reject the null hypothesis.
There is NO statistically significant difference between groups.

============================================================
ONE-WAY ANOVA: Normalized Avg Tau by Atherosclerosis Category
============================================================
     None: n=  7 | mean= -0.219 | sd=  0.288
     Mild: n= 22 | mean=  0.193 | sd=  1.016
 Moderate: n= 51 | mean= -0.028 | sd=  0.625
   Severe: n=  4 | mean= -0.317 | sd=  0.132

--- ANOVA ---
F(3, 80) = 1.0220,  p = 0.387422
Effect size (eta^2) = 0.0369
Result: FAILS to reject the null hypothesis.
There is NO statistically significant difference between groups.
```

When this code is run, a T-statistic value of -1.6234 is found and a P-value of 0.1390 is found. Because P>0.05, it can be concluded that the difference in "none" and "severe" ABeta values was not statistically significant

A concern we have about the project is the code's difficultly in running in Jupyter notebook. when all the code we have written above is copied and pasted into a separate file in the order that we have included it in the notebook, it is able to run but due to the setup of Jupyter notebook, some aspects of the code will not run unless a separate file is created elsewhere on vs code.

Verify and validate your analysis:

- Verification: • Both ANOVAs (for normalized Aβ and normalized tau) show no significant differences between atherosclerosis categories (p-values: 0.756 for Aβ, 0.387 for tau) • Intuitively, this makes sense, as although we did not see global changes in the protein levels for atherosclerosis, this is the case in the literature as atherosclerosis doesn't always increase bulk plaque/tangle load. Still, it does affect perivascular deposition and clearance. • Many studies suggest that the impact of atherosclerosis is more subtle and spatial — affecting how and where Aβ and tau

accumulate (along vessel walls or in perivascular spaces), rather than always increasing total levels.

- Validation: • Atherosclerosis is associated with amyloid and tau pathology via blood-brain barrier dysfunction in the hippocampus of aged human brains (Jin et al., 2025) o No significant difference in total parenchymal Aβ plaques between the no and severe atherosclerosis groups. o Increased vascular amyloid (CAA) and localized perivascular tau in severe atherosclerosis. • Impaired Aβ clearance: a potential link between atherosclerosis and Alzheimer's disease" (Gupta et al., 2015) o Atherosclerosis can impair clearance pathways for Aβ, especially perivascular drainage, rather than always increasing overall production o The connection is often through impaired clearance and localized vascular changes, not necessarily through large global increases in amyloid/tau burden across groups. • Does the validation data match your results? o Yes; no statistical significance in bulk difference in Aβ or tau between atherosclerosis categories is consistent with both studies and although there is a connection, it is not enough to show up in our data.

Conclusions and Ethical Implications:

- conclusion: Amyloid-beta and tau levels in the brain do not have an impact on the atherosclerosis development

Aβ p value: 0.755843 > 0.05, we fail to reject the null hypothesis and there is no statistical difference between groups Tau p-value: 0.387422 > 0.05, we fail to reject the null hypothesis and there is no statistical difference between groups

- Ethical implications:
  - If a person is found to have atherosclerosis, it cannot be directly assumed that the person will also have Alzheimer's disease
  - Different medications will have to be given to patients to target Aβ and tau levels and atherosclerosis
  - Patients and doctors can avoid unnecessary testing for amyloid/tau as markers of cardiovascular disease, preventing false hope or misdiagnosis.
  - Insurance agencies and public health policies cannot assume that if a person has increased levels of amyloid-beta and tau, they will also develop atherosclerosis

Limitations and Future Work:

- limitations: we had a limited amount of data to work with and only snapshots of values were taken (no longditual data)
- Future work:

- Determine if there are other proteins in the brain that could increase the speed of development atherosclerosis
- Collecting larger datasets with more longitudinal data to track if patients who show signs of atherosclerosis develop higher levels of tau and amyloid beta (and vice versa)

citations:

Alzheimer's Association. (2025). Alzheimer’s Disease Facts and Figures. Retrieved from Alzheimer’s Disease and Dementia website: https://www.alz.org/alzheimers-dementia/facts-figures Alzheimer's Society. (2023, April 13). Who Gets Alzheimer’s disease? Retrieved from Alzheimer’s Society website: https://www.alzheimers.org.uk/about-dementia/types-dementia/who-gets-alzheimers-disease Alzheimers.gov. (n.d.). Resources for Health Care Providers: Alzheimer’s and Related Dementias | National Institute on Aging. Retrieved from www.nia.nih.gov website: https://www.alzheimers.gov/professionals/health-care-providers CDC. (2024). Non-Medical Factors that Affect Alzheimer’s Disease and Related Dementias Risk. Retrieved from Alzheimer’s Disease and Dementia website: https://www.cdc.gov/alzheimers-dementia/php/sdoh/index.html Gupta, A., & Iadecola, C. (2015). Impaired AÎ2 clearance: a potential link between atherosclerosis and Alzheimerâ€TMs disease. Frontiers in Aging Neuroscience, 7. https://doi.org/10.3389/fnagi.2015.00115 Huang, L., Kuan, Y., Lin, H.-W., & Hu, C. (2023). Clinical Trials of New Drugs for Alzheimer disease: a 2020–2023 Update. Journal of Biomedical Science, 30(1). https://doi.org/10.1186/s12929-023-00976-6 Jin, Z., Liu, N., & Wei, H. (2025). Atherosclerosis is associated with amyloid and tau pathology via blood–brain barrier dysfunction in the hippocampus of aged human brains. PLOS One, 20(6), e0324652. https://doi.org/10.1371/journal.pone.0324652 National Institute on Aging. (2023). Alzheimer’s Disease Fact Sheet. Retrieved from National Institute on Aging website: https://www.nia.nih.gov/health/alzheimers-and-dementia/alzheimers-disease-fact-sheet Tryphena, K. P., Shukla, R., Khatri, D. K., & Vora, L. K. (2024). Pathogenesis, Diagnostics, and Therapeutics for Alzheimer’s Disease: Breaking the Memory Barrier. Ageing Research Reviews, 101, 102481–102481. https://doi.org/10.1016/j.arr.2024.102481