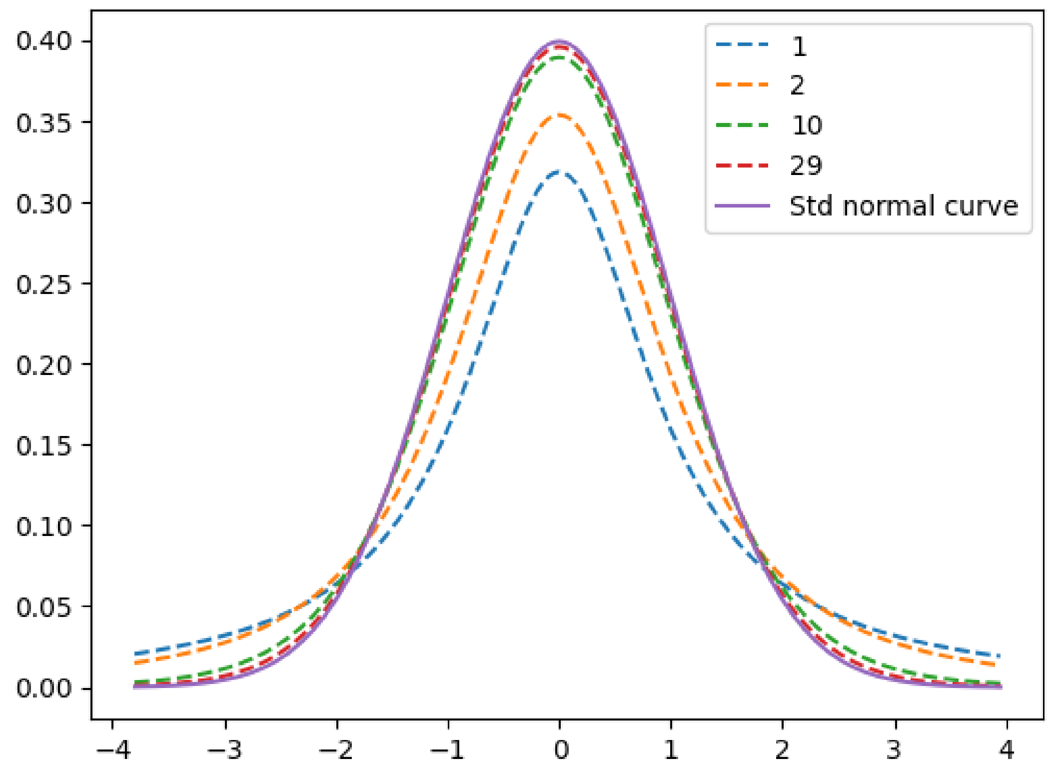
studentstdist

November 1, 2024

[1]:
```python
# Enable inline plotting for Jupyter notebooks
%matplotlib inline
# Import necessary libraries
import matplotlib.pyplot as plt # For creating plots
from scipy.stats import t, norm # For statistical functions related to␣
↪t-distribution and normal distribution
import numpy as np # For numerical operations
import pandas as pd # For data manipulation (not used in this snippet)
# Create an array of values from -3.8 to 4, with increments of 1/20
x = np.arange(-3.8, 4, 1/20) # This serves as the x-axis values for the plots
# Loop through different degrees of freedom for the t-distribution
for i in [1, 2, 10, 29]: # List of degrees of freedom to plot
    # Plotting the t-distribution curves (PDF gives probability density␣
↪function)
    plt.plot(x, t.pdf(x, i), '--', label=i) # Dashed lines for different␣
↪t-distribution curves
# Plotting the standard normal curve
plt.plot(x, norm.pdf(x), label='Std normal curve') # Solid line for the␣
↪standard normal distribution
# Add legend to the upper right of the plot
plt.legend(loc='upper right') # Show the legend with labels for each curve
plt.show() # Display the plot
# Calculate and print the complement of the cumulative distribution function␣
↪(CDF) for the t-distribution
print("1 - cdf gives :", 1 - t.cdf(1.59, 2)) # Tail probability for␣
↪t-distribution with 2 degrees of freedom
print('same as :', t.sf(1.59, 2)) # Calculate the survival function (SF), which␣
↪is equivalent to 1 - CDF
# Calculate and print the tail probabilities for the standard normal␣
↪distribution
print(1 - norm.cdf(2), norm.sf(2)) # Tail probability for the standard normal␣
↪distribution at z = 2
```

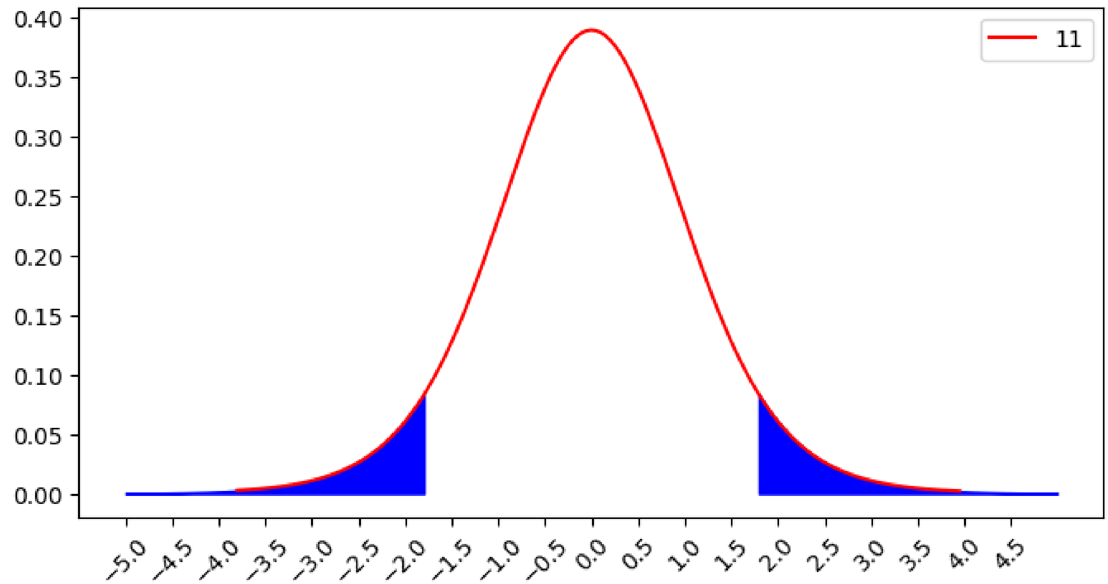



```
1 - cdf gives : 0.12639805893063705
same as : 0.12639805893063707
0.022750131948179209 0.0227501319481792
```

[2]:

```python
# Define a function to plot the t-distribution with shaded areas for the␣
↪critical region
def t_table(n, alpha):
    # Calculate the critical value (t-score) for the given alpha level
    s = t.ppf(alpha / 2, n - 1) # PPF is the percent point function (inverse of␣
↪CDF)
    # Set up the figure size for the plot
    plt.figure(figsize=(8, 4))
    # Plot the t-distribution curve for n-1 degrees of freedom
    plt.plot(x, t.pdf(x, n - 1), color='red', label=n - 1) # Red curve for␣
↪t-distribution
    # Calculate the ranges for the areas to be shaded
    section1 = np.arange(-5, s, 1/20.) # Range from -5 to the critical value s
    section2 = np.arange(-s, 5, 1/20.) # Range from -s to 5
    # Fill the areas under the t-distribution curve
    plt.fill_between(section1, t.pdf(section1, n - 1), color='blue') # Fill␣
↪left critical region
```

```python
    plt.fill_between(section2, t.pdf(section2, n - 1), color='blue') # Fill right critical region
    # Set x-ticks for better readability
    plt.xticks(np.arange(-5, 5, 0.5), rotation=45) # Set ticks and rotate for clarity
    plt.legend(loc='upper right') # Show the legend for the plot
    plt.show() # Display the plot
# Call the t_table function with sample size and significance level
t_table(12, 0.1) # Sample size of 12 and alpha level of 0.1
```



[3]:

```python
# Create an array of values from -7 to 8, with increments of 1/20
x = np.arange(-7, 8, 1/20)
# Define a function to plot the confidence interval
def ci(t_score, n):
    # Set up the figure size for the plot
    plt.figure(figsize=(8, 4))
    # Calculate the area under the t-distribution curve for the confidence interval
    area = t.cdf(t_score, n - 1) - t.cdf(-t_score, n - 1) # Area between -t_score and +t_score
    print('Confidence Level', area * 100) # Print the confidence level as a percentage
    # Plot the t-distribution curve for n-1 degrees of freedom
    plt.plot(x, t.pdf(x, n - 1), color='red', label=n - 1) # Red curve for t-distribution
    # Define the range for the shaded area (confidence interval)
```

```
    section = np.arange(-t_score, t_score, 1/20.) # Range from -t_score to
 ↪+t_score
    # Fill the area between -t_score and +t_score
    plt.fill_between(section, t.pdf(section, n - 1)) # Fill the confidence
 ↪interval area
    # Set x-ticks for better readability
    plt.xticks(np.arange(-6, 7, 0.5), rotation=45) # Set ticks and rotate for
 ↪clarity
    plt.legend(loc='upper right') # Show the legend for the plot
    plt.show() # Display the plot
# Call the ci function with a specific t-score and sample size
ci(5.841, 4) # t-score of 5.841 and sample size of 4
```

```
Confidence Level 99.00004355246759
```