Statistical Analysis Using SciPy and Statsmodels

Statistical analysis involves summarizing data, testing hypotheses, and understanding relationships between variables. Python libraries like SciPy and Statsmodels provide tools for performing such analyses efficiently.

---

1. Install and Import Libraries

bash
Copy code

```bash
pip install scipy statsmodels
```

python
Copy code

```python
import numpy as np
import pandas as pd
from scipy import stats
import statsmodels.api as sm
import statsmodels.formula.api as smf
```

---

2. Descriptive Statistics

Basic Statistics

python
Copy code

```python
# Example data
data = [12, 15, 14, 10, 13, 14, 16, 11]

# Mean, median, mode
mean = np.mean(data)
median = np.median(data)
mode = stats.mode(data)
print("Mean:", mean)
print("Median:", median)
print("Mode:", mode)
```

Variance and Standard Deviation

python
Copy code

```python
variance = np.var(data, ddof=1)  # Sample variance
```

```
std_dev = np.std(data, ddof=1)   # Sample standard deviation
print("Variance:", variance)
print("Standard Deviation:", std_dev)
```

Percentiles
python
Copy code

```
percentiles = np.percentile(data, [25, 50, 75])  # Quartiles
print("25th, 50th, 75th Percentiles:", percentiles)
```

---

3. Hypothesis Testing

T-tests
One-Sample T-Test: Compare sample mean to a population mean.
python
Copy code

```
t_stat, p_value = stats.ttest_1samp(data, popmean=14)
print("T-Statistic:", t_stat, "P-Value:", p_value)
```

1.

Two-Sample T-Test: Compare means of two independent samples.
python
Copy code

```
data1 = [12, 15, 14, 10, 13, 14]
data2 = [14, 18, 17, 15, 16, 19]
t_stat, p_value = stats.ttest_ind(data1, data2)
print("T-Statistic:", t_stat, "P-Value:", p_value)
```

2.

Paired T-Test: Compare means of two related samples.
python
Copy code

```
t_stat, p_value = stats.ttest_rel(data1, data2)
print("T-Statistic:", t_stat, "P-Value:", p_value)
```

3.

Chi-Square Test

Test for independence or goodness of fit.

python
Copy code

```
# Contingency table
observed = np.array([[50, 30], [20, 40]])

# Chi-square test
chi2, p, dof, expected = stats.chi2_contingency(observed)
print("Chi-Square:", chi2, "P-Value:", p)
```

ANOVA (Analysis of Variance)

Compare means of three or more groups.

python
Copy code

```
group1 = [14, 15, 13, 16]
group2 = [22, 21, 19, 23]
group3 = [30, 29, 27, 31]

f_stat, p_value = stats.f_oneway(group1, group2, group3)
print("F-Statistic:", f_stat, "P-Value:", p_value)
```

---

4. Correlation and Regression

Correlation

python
Copy code

```
# Example data
x = [1, 2, 3, 4, 5]
y = [10, 20, 30, 40, 50]

# Pearson correlation
correlation, p_value = stats.pearsonr(x, y)
print("Pearson Correlation:", correlation)

# Spearman correlation
correlation, p_value = stats.spearmanr(x, y)
print("Spearman Correlation:", correlation)
```

Linear Regression

Using SciPy

python

Copy code

```
slope, intercept, r_value, p_value, std_err = stats.linregress(x, y)
print("Slope:", slope, "Intercept:", intercept)
```

1.

Using Statsmodels

python

Copy code

```
# Prepare data
X = sm.add_constant(x)  # Add intercept
model = sm.OLS(y, X).fit()
print(model.summary())
```

2.

---

5. Advanced Regression with Statsmodels

Multiple Linear Regression

python

Copy code

```
# Example DataFrame
df = pd.DataFrame({
    'y': [1, 2, 3, 4, 5],
    'x1': [10, 20, 30, 40, 50],
    'x2': [5, 4, 3, 2, 1]
})

# Define model
model = smf.ols('y ~ x1 + x2', data=df).fit()
print(model.summary())
```

Logistic Regression

python

Copy code

```
# Binary target variable
df['target'] = [0, 1, 0, 1, 0]

# Logistic regression
log_model = smf.logit('target ~ x1 + x2', data=df).fit()
```

```
print(log_model.summary())
```

---

6. Time Series Analysis

Autocorrelation

python

Copy code

```
from statsmodels.graphics.tsaplots import plot_acf

# Plot autocorrelation
plot_acf(data, lags=10)
```

ARIMA Model

python

Copy code

```
from statsmodels.tsa.arima.model import ARIMA

# Fit ARIMA model
model = ARIMA(data, order=(1, 1, 1))
model_fit = model.fit()
print(model_fit.summary())
```

---

7. Probability Distributions

Generate Random Data

python

Copy code

```
# Normal distribution
data = np.random.normal(loc=0, scale=1, size=1000)

# Uniform distribution
data = np.random.uniform(low=0, high=10, size=1000)
```

Fit and Test Distributions

python

Copy code

```
# Fit normal distribution
params = stats.norm.fit(data)
```

```
# Test goodness of fit
stat, p_value = stats.kstest(data, 'norm', args=params)
print("K-S Statistic:", stat, "P-Value:", p_value)
```

---

8. Visualization

Use `matplotlib` or `seaborn` for visualization of statistical results.

python
Copy code

```
import matplotlib.pyplot as plt
import seaborn as sns

# Boxplot
sns.boxplot(data=data)
plt.show()

# Histogram
sns.histplot(data=data, kde=True)
plt.show()
```