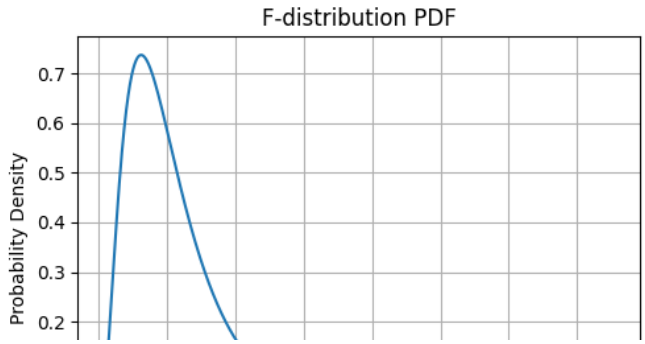
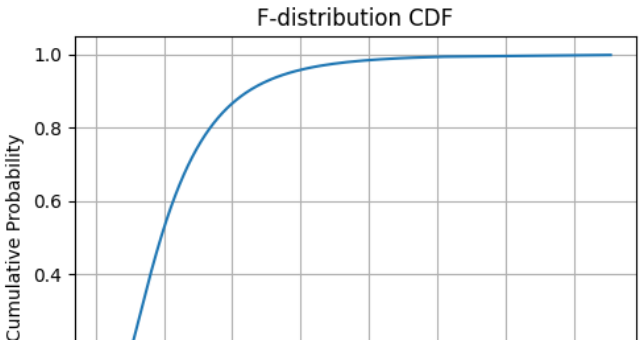
```
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt

# Create the data for two groups
group1 = np.random.rand(25)
group2 = np.random.rand(20)

# Calculate the sample variances
#variance1 = np.var(group1, ddof=1)
#variance2 = np.var(group2, ddof=1)

variance1 = np.var(group1)
variance2 = np.var(group2)

# Calculate the F-statistic
f_value = variance1 / variance2

# Calculate the degrees of freedom
df1 = len(group1) - 1
df2 = len(group2) - 1

# Calculate the p-value
p_value = stats.f.cdf(f_value, df1, df2)

# Print the results
print('Degree of freedom 1:',df1)
print('Degree of freedom 2:',df2)
print("F-statistic:", f_value)
print("p-value:", p_value)
```

```
Degree of freedom 1: 24
Degree of freedom 2: 19
F-statistic: 0.7178078934812079
p-value: 0.21926755402589274
```

```
# Set the degrees of freedom
df1 = 7
df2 = 13

# Generate random samples from chi-square distributions
sample1 = np.random.chisquare(df1, size=1000)
sample2 = np.random.chisquare(df2, size=1000)

# Calculate the F-statistic
f_value = (sample1 / df1) / (sample2 / df2)

# Sort the f-statistic for better distribution plot
f_value = np.sort(f_value)
# Calculate the PDF of the F-distribution
pdf = stats.f.pdf(f_value, df1, df2)

# Calculate the CDF of the F-distribution
cdf = stats.f.cdf(f_value, df1, df2)

# Calculate the corresponding p-value
p_value = 1 - cdf


# Plot the PDF and CDF of the F-distribution
plt.figure(figsize=(10, 4))
plt.subplot(1, 2, 1)
plt.plot(f_value, pdf)
plt.title('F-distribution PDF')
plt.xlabel('x')
plt.grid(True)

plt.ylabel('Probability Density')

plt.subplot(1, 2, 2)
plt.plot(f_value, cdf)
plt.title('F-distribution CDF')
plt.xlabel('x')
plt.ylabel('Cumulative Probability')
plt.grid(True)
plt.tight_layout()
plt.show()
```


F-distribution PDF
Probability Density
0.7
0.6
0.5
0.4
0.3
0.2


F-distribution CDF
Cumulative Probability
1.0
0.8
0.6
0.4

```
# Generate 25 samples
x = np.random.rand(25)

# Randomly group the data into 10 groups
num_groups = 5
group_labels = np.random.randint(0, num_groups, size=len(x))

# Calculate the group means
group_means = []
for i in range(num_groups):
    group_means.append(np.mean(x[group_labels == i]))

# Calculate the overall mean
overall_mean = np.mean(x)

# Calculate the sum of squares between groups
SSB = np.sum([len(x[group_labels == i]) * (group_means[i] - overall_mean)**2 for i in ran

# Calculate the degrees of freedom between groups
df_between = num_groups - 1
# Calculate the degrees of freedom with in groups
df_within = len(x)-num_groups

# Calculate the mean square between groups
MSB = SSB / df_between

# Calculate the sum of squares within groups
SSW = 0
for i in range(num_groups):
    group_samples = x[group_labels == i]
    SSW += np.sum((group_samples - group_means[i])**2)

MSW = SSW / df_within

# Calculate the F-value
F_value = MSB / MSW

# Degree of Freedom
print('Degree of Freedom between groups',df_between)
print('Degree of Freedom within groups',df_within)

# Print the F-value
print("F-value:", F_value)

# Set the significance level
alpha = 0.05

# Calculate the F-value using Percent point function (inverse of cdf)
f_critical = stats.f.ppf(1 - alpha, df_between, df_within)

# Print the F-critical
print("F-critical:", f_critical)

# Check the hypothesis
```

```
if F_value > f_critical:
    print("Reject the null hypothesis")
else:
    print("Fail to reject the null hypothesis")
```

```
Degree of Freedom between groups 4
Degree of Freedom within groups 20
F-value: 2.766556989383863
F-critical: 2.8660814020156584
Fail to reject the null hypothesis
```

```
import pandas as pd
import statsmodels.api as sm

np.random.seed(9876789)

nsample = 100
x= np.linspace(0,10,100)
X = np.column_stack((x,x**2))
beta = np.array([1,0.1,10])
e= np.random.normal(size=nsample)
```

```
X = sm.add_constant(X)
y = np.dot(X, beta) + e
```

```
model =sm.OLS(y,x)
results =model.fit()
print(results.summary())
```

```
                                 OLS Regression Results
=======================================================================================
Dep. Variable:                      y   R-squared (uncentered):                   0.9
Model:                            OLS   Adj. R-squared (uncentered):              0.9
Method:                 Least Squares   F-statistic:                              150
Date:                Thu, 21 Dec 2023   Prob (F-statistic):                    1.12e-
Time:                        07:28:41   Log-Likelihood:                         -613.
No. Observations:                 100   AIC:                                      123
Df Residuals:                      99   BIC:                                      123
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
x1            75.6143      1.949     38.790      0.000      71.746      79.482
==============================================================================
Omnibus:                       18.608   Durbin-Watson:                   0.003
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               22.939
Skew:                           1.161   Prob(JB):                     1.04e-05
Kurtosis:                       3.336   Cond. No.                         1.00
==============================================================================

Notes:
```

```
[1] R² is computed without centering (uncentered) since the model does not contain a
[2] Standard Errors assume that the covariance matrix of the errors is correctly spec
```

```
nsample = 50
sig = 0.5
x = np.linspace(0, 20, nsample)
X = np.column_stack((x, np.sin(x), (x-5) ** 2, np.ones(nsample)))
beta = [0.5, 0.5, -0.02, 5.0]

y_true = np.dot(X, beta)
y = y_true + sig * np.random.normal(size=nsample)


res =sm.OLS(y, X).fit()
print(res.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.933
Model:                            OLS   Adj. R-squared:                  0.928
```