In [5]:
```
# Install and import necessary packages
!pip install pandas scipy statsmodels openpyxl
import pandas as pd
import numpy as np
from scipy import stats
import statsmodels.api as sm
import matplotlib.pyplot as plt
```

```
Requirement already satisfied: pandas in c:\users\nihar\anaconda3\lib\site-packages (2.1.4)
Requirement already satisfied: scipy in c:\users\nihar\anaconda3\lib\site-packages (1.11.4)
Requirement already satisfied: statsmodels in c:\users\nihar\anaconda3\lib\site-packages (0.14.0)
Requirement already satisfied: openpyxl in c:\users\nihar\anaconda3\lib\site-packages (3.0.10)
Requirement already satisfied: numpy<2,>=1.23.2 in c:\users\nihar\anaconda3\lib\site-packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\nihar\anaconda3\lib\site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\nihar\anaconda3\lib\site-packages (from pandas) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\nihar\anaconda3\lib\site-packages (from pandas) (2023.3)
Requirement already satisfied: patsy>=0.5.2 in c:\users\nihar\anaconda3\lib\site-packages (from statsmodels) (0.5.3)
Requirement already satisfied: packaging>=21.3 in c:\users\nihar\anaconda3\lib\site-packages (from statsmodels) (23.1)
Requirement already satisfied: et_xmlfile in c:\users\nihar\anaconda3\lib\site-packages (from openpyxl) (1.1.0)
Requirement already satisfied: six in c:\users\nihar\anaconda3\lib\site-packages (from patsy>=0.5.2->statsmodels) (1.16.0)
```

In [7]:
```
# Load the dataset
ipl_data = pd.read_csv(
    "C:/Users/nihar/OneDrive/Desktop/Bootcamp/SCMA 632/Assignments/Atb/IPL_ball_by_ball_updated till 2024.csv",
    low_memory=False
)
```

In [11]:
```
# Display column names to verify them
print(ipl_data.columns)
```

```
Index(['Match id', 'Date', 'Season', 'Batting team', 'Bowling team',
       'Innings No', 'Ball No', 'Bowler', 'Striker', 'Non Striker',
       'runs_scored', 'extras', 'type of extras', 'score', 'score/wicket',
       'wicket_confirmation', 'wicket_type', 'fielders_involved',
       'Player Out'],
      dtype='object')
```

In [15]:
```
# Aggregate data season-wise, batsman-wise, and bowler-wise
ipl_summary = ipl_data.groupby(['Season', 'Match id', 'Striker', 'Bowler']).agg(
    total_runs=('runs_scored', 'sum'),
    total_wickets=('wicket_confirmation', 'sum')
).reset_index()
```

In [17]:
```
# Top three run-getters and top three wicket-takers per season
top_players_per_season = ipl_summary.groupby('Season').apply(
    lambda x: pd.DataFrame({
        'top_run_getters': [x.nlargest(3, 'total_runs')],
        'top_wicket_takers': [x.nlargest(3, 'total_wickets')]
    })
).reset_index()
```

In [19]:
```
# Filter data for the last three IPL tournaments
last_three_seasons = ipl_data[ipl_data['Season'].isin(ipl_data['Season'].unique()[-3:])]
```

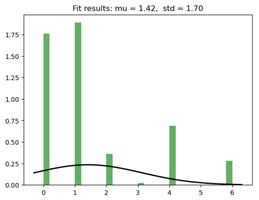
In [21]:
```
# Get top three batsmen based on total runs
top_batsmen = last_three_seasons.groupby('Striker')['runs_scored'].sum().nlargest(3).reset_index()
```

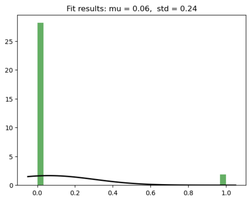
In [23]:
```
# Get top three bowlers based on total wickets
top_bowlers = last_three_seasons.groupby('Bowler')['wicket_confirmation'].sum().nlargest(3).reset_index()
```

In [25]:
```
# Extract the data for top batsmen and bowlers
top_batsmen_runs = last_three_seasons[last_three_seasons['Striker'].isin(top_batsmen['Striker'])]['runs_scored']
top_bowlers_wickets = last_three_seasons[last_three_seasons['Bowler'].isin(top_bowlers['Bowler'])]['wicket_confirmation']
```

In [27]:
```
# Function to fit and plot distribution
def fit_and_plot_distribution(data):
    fit = stats.norm.fit(data)
    plt.hist(data, bins=20, density=True, alpha=0.6, color='g')
    xmin, xmax = plt.xlim()
    x = np.linspace(xmin, xmax, 100)
    p = stats.norm.pdf(x, *fit)
    plt.plot(x, p, 'k', linewidth=2)
    title = "Fit results: mu = %.2f,  std = %.2f" % (fit[0], fit[1])
    plt.title(title)
    plt.show()
    return fit
```

In [29]:
```
# Fit distributions for top batsmen and bowlers
fit_batsmen = fit_and_plot_distribution(top_batsmen_runs)
fit_bowlers = fit_and_plot_distribution(top_bowlers_wickets)
```





In [31]:
```
# Print summary of fitted distributions
print(f"Top Batsmen Distribution: mu = {fit_batsmen[0]}, std = {fit_batsmen[1]}")
print(f"Top Bowlers Distribution: mu = {fit_bowlers[0]}, std = {fit_bowlers[1]}")
```

```
Top Batsmen Distribution: mu = 1.4200776815055826, std = 1.7012772912760035
Top Bowlers Distribution: mu = 0.06190643686240922, std = 0.24098553885547225
```

In [33]:
```
# Filter data for SP Narine
narine_data = ipl_data[(ipl_data['Striker'] == 'SP Narine') | (ipl_data['Bowler'] == 'SP Narine')]
```

In [35]:
```
# Load the salary dataset
salary_data = pd.read_excel("C://Users//nihar//OneDrive//Desktop//Bootcamp//SCMA 632//Assignments//Atb//IPL SALARIES 2024.xlsx")
```

In [37]:
```
# Filter the salary data for SP Narine
narine_salary = salary_data[salary_data['Player'].str.contains("Sunil Narine", na=False)]
```

In [39]:
```
# Manually create the salary data for SP Narine
seasons = [2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2021, 2022, 2023, 2024]
salaries = [600] * len(seasons)
narine_salary = pd.DataFrame({
    'Player': ['Sunil Narine'] * len(seasons),
    'Season': seasons,
    'Salary': salaries
})
```

In [41]:
```
# Summarize performance metrics for SP Narine
narine_performance = narine_data.groupby('Season').agg(
    total_runs=('runs_scored', lambda x: x[narine_data['Striker'] == 'SP Narine'].sum()),
    total_wickets=('wicket_confirmation', lambda x: x[narine_data['Bowler'] == 'SP Narine'].sum())
).reset_index()
```

In [43]:
```
# Clean the Season column to retain only numeric values and then convert to numeric
narine_salary['Season'] = pd.to_numeric(narine_salary['Season'], errors='coerce')
narine_performance['Season'] = pd.to_numeric(narine_performance['Season'], errors='coerce')
```

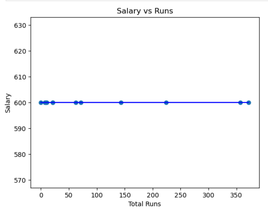
In [45]:
```
# Join the summarized performance metrics with the salary data
narine_performance = narine_performance.merge(narine_salary, on='Season', how='left')

# Remove rows with NA values in the narine_performance data frame
narine_performance = narine_performance.dropna(subset=['Player', 'Salary'])
```

In [47]:
```
# Fit a linear model to find the relationship between performance and salary
fit_model_runs = sm.OLS(narine_performance['Salary'], sm.add_constant(narine_performance['total_runs'])).fit()
fit_model_wickets = sm.OLS(narine_performance['Salary'], sm.add_constant(narine_performance['total_wickets'])).fit()
```

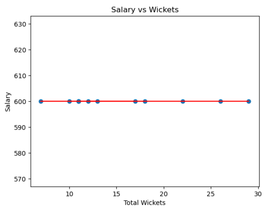
In [49]:
```
# Summary of the models
print(fit_model_runs.summary())
print(fit_model_wickets.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                 Salary   R-squared:                       -inf
Model:                            OLS   Adj. R-squared:                  -inf
Method:                 Least Squares   F-statistic:                   -10.00
Date:                Wed, 19 Jun 2024   Prob (F-statistic):             1.00
Time:                        00:44:32   Log-Likelihood:                340.64
No. Observations:                  12   AIC:                           -677.3
Df Residuals:                      10   BIC:                           -676.3
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        600.0000   4.66e-14   1.29e+16      0.000     600.000     600.000
total_runs     0   2.74e-16          0      1.000   -6.11e-16    6.11e-16
==============================================================================
Omnibus:                        nan   Durbin-Watson:                   0.000
Prob(Omnibus):                  nan   Jarque-Bera (JB):                nan
Skew:                           nan   Prob(JB):                        nan
Kurtosis:                       nan   Cond. No.                         220.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
                            OLS Regression Results
==============================================================================
Dep. Variable:                 Salary   R-squared:                       -inf
Model:                            OLS   Adj. R-squared:                  -inf
Method:                 Least Squares   F-statistic:                   -10.00
Date:                Wed, 19 Jun 2024   Prob (F-statistic):             1.00
Time:                        00:44:32   Log-Likelihood:                331.50
No. Observations:                  12   AIC:                           -659.0
Df Residuals:                      10   BIC:                           -658.0
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        600.0000   1.97e-13   3.05e+15      0.000     600.000     600.000
total_wickets  1.421e-14   1.16e-14      1.221      0.250   -1.17e-14    4.01e-14
==============================================================================
Omnibus:                        1.903   Durbin-Watson:                   0.164
Prob(Omnibus):                  0.386   Jarque-Bera (JB):                1.215
Skew:                          -0.743   Prob(JB):                        0.545
Kurtosis:                       2.530   Cond. No.                         43.5
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
C:\Users\nihar\anaconda3\Lib\site-packages\statsmodels\stats\stattools.py:125: RuntimeWarning: Precision loss occurred in moment calculation due to catastrophic cancellation. This occurs when the data are nearly identical. Results may be unreliable.
  skew = stats.skew(resids, axis=axis)
C:\Users\nihar\anaconda3\Lib\site-packages\statsmodels\stats\stattools.py:126: RuntimeWarning: Precision loss occurred in moment calculation due to catastrophic cancellation. This occurs when the data are nearly identical. Results may be unreliable.
  kurtosis = 3 + stats.kurtosis(resids, axis=axis)
C:\Users\nihar\anaconda3\Lib\site-packages\scipy\stats\_stats_py.py:1606: RuntimeWarning: Precision loss occurred in moment calculation due to catastrophic cancellation. This occurs when the data are nearly identical. Results may be unreliable.
  b2 = skew(a, axis)
C:\Users\nihar\anaconda3\Lib\site-packages\scipy\stats\_stats_py.py:1806: UserWarning: kurtosistest only valid for n>=20 ... continuing anyway, n=12
  warnings.warn("kurtosistest only valid for n>=20 ... continuing "
C:\Users\nihar\anaconda3\Lib\site-packages\scipy\stats\_stats_py.py:1808: RuntimeWarning: Precision loss occurred in moment calculation due to catastrophic cancellation. This occurs when the data are nearly identical. Results may be unreliable.
  b2 = kurtosis(a, axis, fisher=False)
C:\Users\nihar\anaconda3\Lib\site-packages\statsmodels\regression\linear_model.py:1781: RuntimeWarning: divide by zero encountered in scalar divide
  return 1 - self.ssr/self.centered_tss
C:\Users\nihar\anaconda3\Lib\site-packages\scipy\stats\_stats_py.py:1806: UserWarning: kurtosistest only valid for n>=20 ... continuing anyway, n=12
  warnings.warn("kurtosistest only valid for n>=20 ... continuing "
C:\Users\nihar\anaconda3\Lib\site-packages\statsmodels\regression\linear_model.py:1781: RuntimeWarning: divide by zero encountered in scalar divide
  return 1 - self.ssr/self.centered_tss
```

In [51]:
```
# Plot the relationship
plt.scatter(narine_performance['total_runs'], narine_performance['Salary'])
plt.plot(narine_performance['total_runs'], fit_model_runs.predict(sm.add_constant(narine_performance['total_runs'])), color='blue')
plt.title('Salary vs Runs')
plt.xlabel('Total Runs')
plt.ylabel('Salary')
plt.show()
```



In [53]:
```
plt.scatter(narine_performance['total_wickets'], narine_performance['Salary'])
plt.plot(narine_performance['total_wickets'], fit_model_wickets.predict(sm.add_constant(narine_performance['total_wickets'])), color='red')
plt.title('Salary vs Wickets')
plt.xlabel('Total Wickets')
plt.ylabel('Salary')
plt.show()
```


Salary vs Wickets
630
620
610
600
590
580
570
Salary
10
15
20
25
30
Total Wickets