Assignment A3 (Part-C)

```
import pandas as pd
import numpy as np

# Load data (replace with your actual data loading method)
df_nss = pd.read_csv('NSSO68.csv')
df_nss
```

```
<ipython-input-47-58ab53b0f2b4>:5: DtypeWarning: Columns (1) have mixed types. Specify dtype option on import or set low_memory=Fals
  df_nss = pd.read_csv('NSSO68.csv')
```

	slno	grp	Round_Centre	FSU_number	Round	Schedule_Number	Sample	Sector	state	State_R
0	1	4099999999999999926524952937758 72.0	1	41000	68	10	1	2	24	
1	2	4099999999999999926524952937758 72.0	1	41000	68	10	1	2	24	
2	3	4099999999999999926524952937758 72.0	1	41000	68	10	1	2	24	
3	4	4099999999999999926524952937758 72.0	1	41000	68	10	1	2	24	
4	5	4099999999999999926524952937758 72.0	1	41000	68	10	1	2	24	
...	...	...	...	...	...	...	...	...	...	...
101657	101658	7999999999999999708717035972 1984.0	1	79998	68	10	1	1	1	
101658	101659	7999999999999999708717035972 1984.0	1	79998	68	10	1	1	1	
101659	101660	7999999999999999708717035972 1984.0	1	79998	68	10	1	1	1	
101660	101661	7999999999999999708717035972 1984.0	1	79998	68	10	1	1	1	
101661	101662	7999999999999999708717035972 1984.0	1	79998	68	10	1	1	1	

101662 rows × 384 columns

```
# Data preparation (assuming similar variables as in R example)
df_ap = df_nss[df_nss['state_1'] == 'AP']

vars = ["Sector", "hhdsz", "Religion", "Social_Group", "MPCE_URP", "Sex", "Age",
        "Marital_Status", "Education", "chicken_q", "chicken_v"]

df_ap_p = df_ap[vars]

# Calculate price, handling potential division by zero
df_ap_p['price'] = np.where(df_ap_p['chicken_q'] != 0, df_ap_p['chicken_v'] / df_ap_p['chicken_q'], 0)

df_ap_p[['price']]
```

```
<ipython-input-49-be4f1ac297ea>:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  df_ap_p['price'] = np.where(df_ap_p['chicken_q'] != 0, df_ap_p['chicken_v'] / df_ap_p['chicken_q'], 0)
```

	price
6777	110.0
6778	110.0
6779	130.0
6780	110.0
6781	110.0
...	...
79719	130.0
79720	0.0
79721	0.0
79722	130.0
79723	0.0

6899 rows × 1 columns

```
from scipy.stats import describe

# Summary statistics
summary_stats = describe(df_ap_p['chicken_q'])
print(f"Summary Statistics for chicken_q:\n{summary_stats}")
```

```
Summary Statistics for chicken_q:
DescribeResult(nobs=6899, minmax=(0.0, 13.66666667), mean=0.30174615543948397, variance=0.12746968207875722, skewness=9.985607644666
DescribeResult(nobs=6899, minmax=(0.0, 13.66666667), mean=0.30174615543948397, variance=0.12746968207875722,
skewness=9.985607644660892, kurtosis=301.93321205798736)
```

```
import statsmodels.api as sm

# Linear regression using statsmodels
# Drop rows with NaN or inf in any column
df_ap_p_clean = df_ap_p.replace([np.inf, -np.inf], np.nan).dropna()

X = df_ap_p_clean[['hhdsz', 'Religion', 'MPCE_URP', 'Sex', 'Age', 'Marital_Status', 'Education', 'price']]
y = df_ap_p_clean['chicken_q']
X = sm.add_constant(X)  # Adding a constant (intercept) term
model = sm.OLS(y, X).fit()
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              chicken_q   R-squared:                       0.288
Model:                            OLS   Adj. R-squared:                  0.287
Method:                 Least Squares   F-statistic:                     348.9
Date:                Mon, 01 Jul 2024   Prob (F-statistic):               0.00
Time:                        15:16:04   Log-Likelihood:                -1510.1
No. Observations:                6899   AIC:                             3038.
Df Residuals:                    6890   BIC:                             3100.
Df Model:                           8
Covariance Type:            nonrobust
==============================================================================
                     coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const              0.0284      0.027      1.063      0.288      -0.024       0.081
hhdsz             -0.0255      0.002    -11.671      0.000      -0.030      -0.021
Religion           0.0112      0.008      1.414      0.157      -0.004       0.027
MPCE_URP        3.925e-05   1.92e-06     20.489      0.000    3.55e-05     4.3e-05
Sex               -0.0372      0.014     -2.582      0.010      -0.065      -0.009
Age             2.556e-05      0.000      0.083      0.934      -0.001       0.001
Marital_Status     0.0312      0.012      2.676      0.007       0.008       0.054
Education         -0.0056      0.001     -5.475      0.000      -0.008      -0.004
price              0.0032   6.85e-05     46.505      0.000       0.003       0.003
==============================================================================
Omnibus:                    12975.245   Durbin-Watson:                   1.861
Prob(Omnibus):                  0.000   Jarque-Bera (JB):         71245755.240
Skew:                          13.741   Prob(JB):                         0.00
Kurtosis:                     500.083   Cond. No.                     2.27e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.27e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

```
from scipy.optimize import minimize

# Tobit regression using scipy.optimize for handling censored data
def tobit_log_likelihood(params, y, X):
    beta = params[:-2]
    sigma = np.exp(params[-2])
    gamma = params[-1]

    mu = np.dot(X, beta)
    cens_residuals = (y - mu) / sigma
    ll = np.sum(np.log(sigma) + gamma * cens_residuals - np.log(1 + np.exp(gamma * cens_residuals)))

    return -ll

# Initial guess for parameters
initial_params = np.zeros(X.shape[1] + 2)
initial_params[-2] = np.log(np.std(y))
initial_params[-1] = 1.0

# Minimize negative log-likelihood
results = minimize(tobit_log_likelihood, initial_params, args=(y, X))

# Extract estimated coefficients and standard errors
```

```
beta_est = results.x[:-2]
sigma_est = np.exp(results.x[-2])
gamma_est = results.x[-1]

# Print Tobit regression results
print("\nTobit Regression Results:")
print(f"Beta (coefficients):\n{beta_est}")
print(f"Sigma (standard deviation of residuals):\n{sigma_est}")
print(f"Gamma (Tobit parameter):\n{gamma_est}")
```

```
/usr/local/lib/python3.10/dist-packages/pandas/core/arraylike.py:396: RuntimeWarning: overflow encountered in exp
  result = getattr(ufunc, method)(*inputs, **kwargs)
/usr/local/lib/python3.10/dist-packages/scipy/optimize/_numdiff.py:576: RuntimeWarning: invalid value encountered in subtract
  df = fun(x) - f0
/usr/local/lib/python3.10/dist-packages/pandas/core/arraylike.py:396: RuntimeWarning: overflow encountered in exp
  result = getattr(ufunc, method)(*inputs, **kwargs)
<ipython-input-45-e900b554f2ef>:6: RuntimeWarning: overflow encountered in exp
  sigma = np.exp(params[-2])
/usr/local/lib/python3.10/dist-packages/scipy/optimize/_numdiff.py:576: RuntimeWarning: invalid value encountered in subtract
  df = fun(x) - f0

Tobit Regression Results:
Beta (coefficients):
[-0.13269726  0.06723805 -0.1390201   0.00028822 -0.18811097 -0.09154083
 -0.24325422  0.05288843 -0.00597459]
Sigma (standard deviation of residuals):
86.06524486156626
Gamma (Tobit parameter):
5.115603295006487
<ipython-input-45-e900b554f2ef>:6: RuntimeWarning: overflow encountered in exp
  sigma = np.exp(params[-2])
/usr/local/lib/python3.10/dist-packages/scipy/optimize/_numdiff.py:576: RuntimeWarning: invalid value encountered in subtract
  df = fun(x) - f0
```