#Sensitive Analysis Code

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.integrate import solve_ivp
from scipy.stats.qmc import LatinHypercube
from scipy.stats import rankdata
import pandas as pd

# Define the epidemic model
def epidemic_model(t, y, beta_pu, beta_mu, beta_pv, beta_mv, e1, e2, mu_p,
mu_m, m, Cv, A):
    S, V, Ipu, Imu, Ipv, Imv, Rs, Rv, x = y

    dS_dt = -x * S - beta_pu * (S - x * S) * (Ipu + Ipv) - beta_mu * (S -
x * S) * (Imu + Imv)
    dV_dt = x * S - beta_pv * (V - e1 * V) * (Ipu + Ipv) - beta_mv * (V -
e2 * V) * (Imu + Imv)
    dIpu_dt = beta_pu * (S - x * S) * (Ipu + Ipv) - mu_p * Ipu
    dImu_dt = beta_mu * (S - x * S) * (Imu + Imv) - mu_m * Imu
    dIpv_dt = beta_pv * (V - e1 * V) * (Ipu + Ipv) - mu_p * Ipv
    dImv_dt = beta_mv * (V - e2 * V) * (Imu + Imv) - mu_m * Imv
    dRs_dt = mu_p * Ipu + mu_m * Imu
    dRv_dt = mu_p * Ipv + mu_m * Imv
    dx_dt  = m * x * (1-x) * (-Cv * V + A + Ipu + Imu + Ipv + Imv)

    return [dS_dt, dV_dt, dIpu_dt, dImu_dt, dIpv_dt, dImv_dt, dRs_dt,
dRv_dt, dx_dt]


# Parameters for LHS sampling
param_ranges = {
    'beta_pu': [0.1, 0.5],
    'beta_mu': [0.1, 0.5],
    'beta_pv': [0.1, 0.5],
    'beta_mv': [0.1, 0.5],
    'e1': [0.5, 0.95],
    'e2': [0.5, 0.95],
    'mu_p': [0.01, 0.05],
    'mu_m': [0.01, 0.05],
    'm': [0.01, 0.05],
    'Cv': [0.1, 0.9],
    'A': [0.1, 1]
}
```

```
n_samples = 1000

# Latin Hypercube Sampling
lhs = LatinHypercube(d=len(param_ranges))
sample = lhs.random(n_samples)

# Scale the samples according to the parameter ranges
scaled_samples = []
for i, (param, bounds) in enumerate(param_ranges.items()):
    lower, upper = bounds
    scaled_samples.append(lower + (upper - lower) * sample[:, i])

scaled_samples = np.array(scaled_samples).T

# Initial conditions
initial_conditions = [0.99, 0.01, 0.001, 0.001, 0, 0, 0, 0, 0.01]
t_span = [0, 160]  # Simulation over 160 days
t_eval = np.linspace(t_span[0], t_span[1], 500)

# Function to calculate peak infected
def calculate_peak_infected(sol):
    I_total = sol.y[2] + sol.y[3] + sol.y[4] + sol.y[5]  # Ipu + Imu + Ipv
+ Imv
    return np.max(I_total)

# Storage for outputs
peak_infected = []

# Run the model for each LHS sample
for params in scaled_samples:
    beta_pu, beta_mu, beta_pv, beta_mv, e1, e2, mu_p, mu_m, m, Cv, A =
params
    sol = solve_ivp(epidemic_model, t_span, initial_conditions,
args=(beta_pu, beta_mu, beta_pv, beta_mv, e1, e2, mu_p, mu_m, m, Cv, A),
t_eval=t_eval)

    # Get peak infected
    peak_infected.append(calculate_peak_infected(sol))

peak_infected = np.array(peak_infected)

# Create DataFrame for scatter plots and PRCC
df = pd.DataFrame(scaled_samples, columns=param_ranges.keys())
df['peak_infected'] = peak_infected
```

```
# Save the DataFrame to an Excel file
output_filename = "Case_1_with_game.xlsx"
df.to_excel(output_filename, index=False)
print(f"Scatter data saved to {output_filename}")

# Scatter plots with colors and lines using Seaborn
colors = ['orange', 'green', 'olive', 'blue', 'purple', 'magenta', 'red',
'brown', 'blueviolet','navy','teal']

for i, param in enumerate(param_ranges.keys()):
    plt.figure(figsize=(8, 6))

    # Scatter plot with Seaborn and trend line
    sns.regplot(x=df[param], y=df['peak_infected'], scatter_kws={'color':
colors[i], 's': 50, 'alpha': 0.6, 'marker': 's', 'edgecolor': 'black' },
                line_kws={"color": "black", "alpha": 0.7, "lw": 2},
ci=None)

    plt.xlabel(param)
    plt.ylabel("Peak Infected")
    plt.show()

# PRCC Analysis
ranked_df = df.apply(rankdata)
prcc_results = ranked_df.corrwith(ranked_df['peak_infected'])
print(prcc_results)

# Compute PRCC using Spearman rank correlation and store p-values
prcc = {}
pvals = {}
for param in ranked_df.columns:
    rho, pval = spearmanr(ranked_df[param], df["peak_infected"])
    pvals[param] = pval
print(pvals)

# Plot PRCC results
plt.figure(figsize=(8, 6))
prcc_results[:-1].plot(kind='bar', color=colors)
plt.title('PRCC Sensitive Analysis on Peak Infected(Case 1: With using
Game)')
plt.ylabel('PRCC Value')
plt.show()
```