STA 3180 Statistical Modelling: Markov Chain Monte Carlo

Start of Code

1. Write a program to simulate a Markov Chain Monte Carlo (MCMC) algorithm using Python.

```
import numpy as np

# Define the transition probability matrix
P = np.array([[0.7, 0.3], [0.4, 0.6]])

# Initialize the state vector
x = np.array([1, 0])

# Define the number of iterations
n_iter = 1000

# Run the MCMC algorithm
for i in range(n_iter):
        x = np.dot(x, P)

# Print the final state vector
print(x)
```

End of Code

Start of Code

2. Write a program to calculate the posterior distribution of a Markov Chain Monte Carlo (MCMC) algorithm using Python.

```
import numpy as np

# Define the transition probability matrix
P = np.array([[0.7, 0.3], [0.4, 0.6]])

# Initialize the state vector
x = np.array([1, 0])

# Define the number of iterations
n_iter = 1000

# Initialize the posterior distribution
post_dist = np.zeros(n_iter)

# Run the MCMC algorithm
for i in range(n_iter):
        x = np.dot(x, P)
        post_dist[i] = x[0]

# Print the posterior distribution
print(post_dist)
```

End of Code

Start of Code

3. Write a program to calculate the mean and variance of a Markov Chain Monte Carlo (MCMC) algorithm using Python.

```
import numpy as np

# Define the transition probability matrix
P = np.array([[0.7, 0.3], [0.4, 0.6]])

# Initialize the state vector
x = np.array([1, 0])

# Define the number of iterations
n_iter = 1000

# Initialize the posterior distribution
post_dist = np.zeros(n_iter)

# Run the MCMC algorithm
for i in range(n_iter):
        x = np.dot(x, P)
        post_dist[i] = x[0]

# Calculate the mean and variance
mean = np.mean(post_dist)
var = np.var(post_dist)

# Print the mean and variance
print("Mean:", mean)
print("Variance:", var)
```

End of Code

Start of Code

4. Write a program to calculate the autocorrelation of a Markov Chain Monte Carlo (MCMC) algorithm using Python.

```
import numpy as np

# Define the transition probability matrix
P = np.array([[0.7, 0.3], [0.4, 0.6]])

# Initialize the state vector
x = np.array([1, 0])

# Define the number of iterations
n_iter = 1000

# Initialize the posterior distribution
post_dist = np.zeros(n_iter)

# Run the MCMC algorithm
for i in range(n_iter):
        x = np.dot(x, P)
```

```
        post_dist[i] = x[0]
# Calculate the autocorrelation
autocorr = np.correlate(post_dist, post_dist, mode='full')
# Print the autocorrelation
print(autocorr)
End of Code
```

Start of Code

5. Write a program to calculate the acceptance rate of a Markov Chain Monte Carlo (MCMC) algorithm using Python.

```
import numpy as np
# Define the transition probability matrix
P = np.array([[0.7, 0.3], [0.4, 0.6]])
# Initialize the state vector
x = np.array([1, 0])
# Define the number of iterations
n_iter = 1000
# Initialize the acceptance rate
accept_rate = 0
# Run the MCMC algorithm
for i in range(n_iter):
        x_new = np.dot(x, P)
        if np.random.rand() < np.min(x_new/x):
                x = x_new
                accept_rate += 1
# Print the acceptance rate
print("Acceptance rate:", accept_rate/n_iter)
End of Code
```