Exercise1_bc

October 31, 2023

[27]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

[28]:
```
mean_a0 = 1.0
mean_a1 = 1.0
sigma_a0 = 0.2
sigma_a1 = 0.2
rho = -0.8
```

[29]:
```
def y(x,a0,a1):
    return a0+a1*x
```

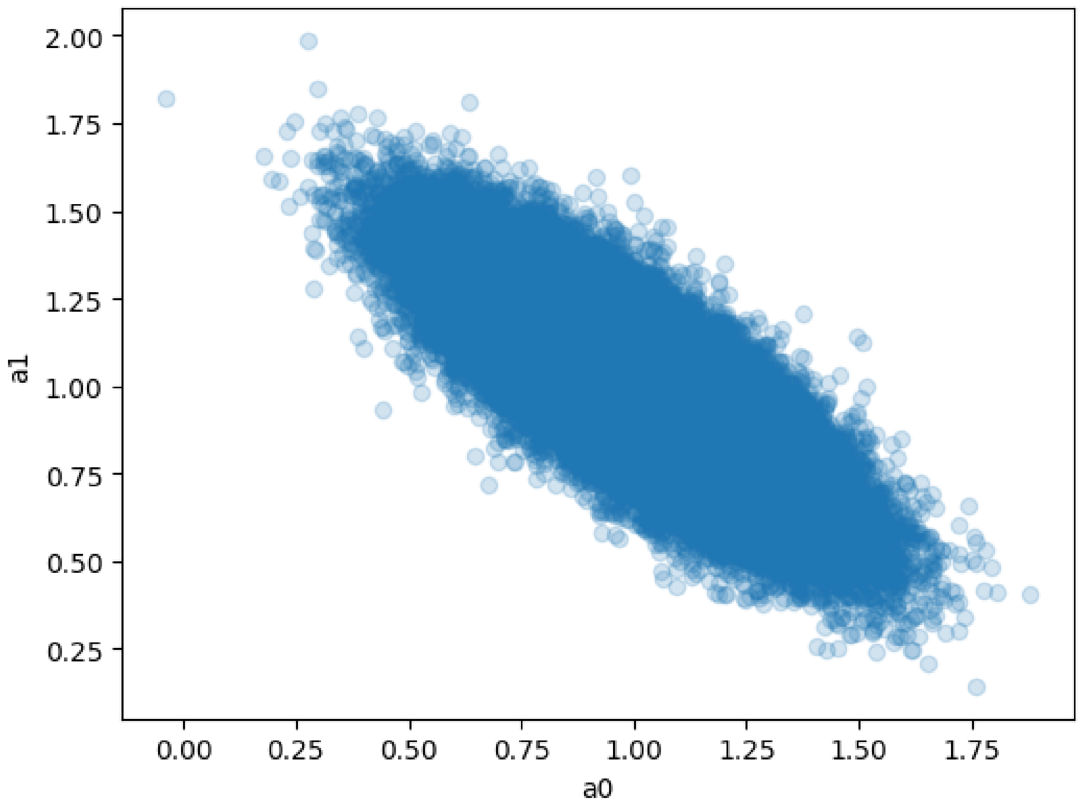
0.1 (b) Determine the result numerically with a Monte Carlo simulation. Visualise the parameters a0 and a1 in a scatter plot

[37]:
```
rng = np.random.default_rng(42)
cov = [[sigma_a0**2,           rho*sigma_a0*sigma_a1],
       [rho*sigma_a0*sigma_a1,  sigma_a1**2]]
```

[44]:
```
# 100000 simulations
a = rng.multivariate_normal(mean = [mean_a0, mean_a1], cov=cov, size=100000)
```

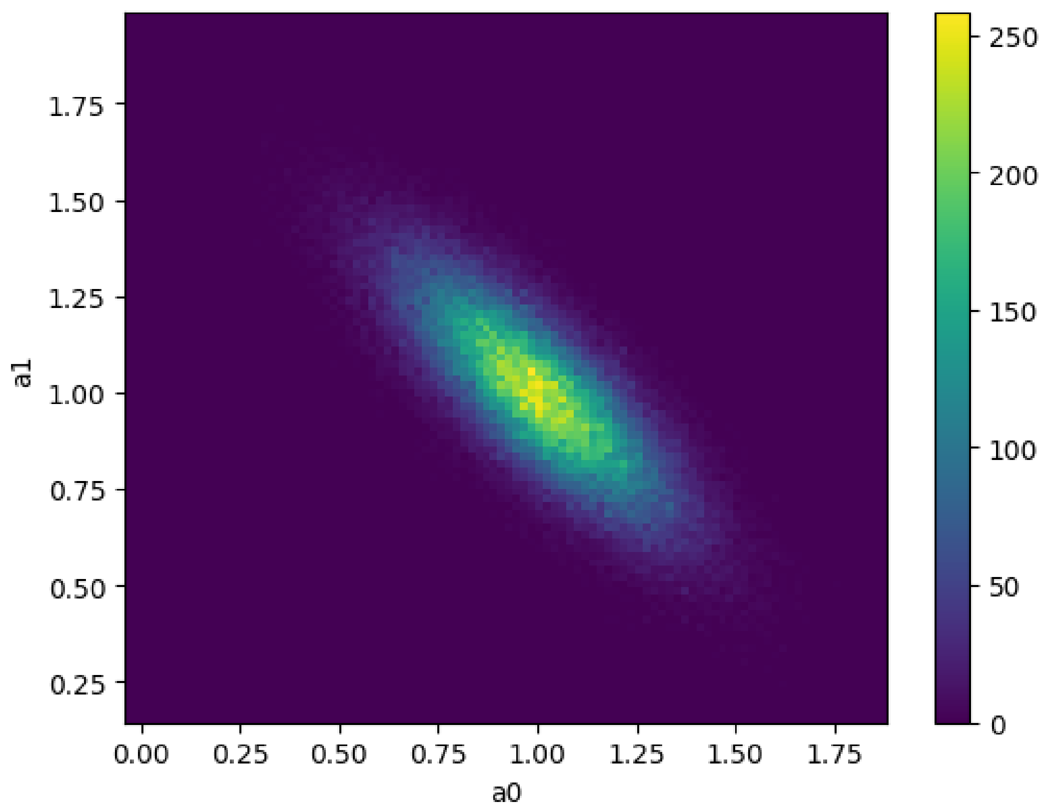
[39]:
```
plt.scatter(a[:,0],a[:,1],alpha=0.2)
plt.xlabel('a0')
plt.ylabel('a1')
```

[39]:
```
Text(0, 0.5, 'a1')
```



[40]:
```
plt.hist2d(a[:,0],a[:,1],bins=100)
plt.xlabel('a0')
plt.ylabel('a1')
plt.colorbar()
```

[40]: <matplotlib.colorbar.Colorbar at 0x7f5d39846260>



0.2 (c) Determine the predictions y (mean and standard deviation) for fixed x=-3,0,3 numerically as well as analytically and compare them.

[34]:
```
def sigma_y_ana(x):
    return np.sqrt(sigma_a0**2 + x**2*sigma_a1**2 + 2*x*sigma_a0*sigma_a1*rho)
```

[41]:
```
for x in [-3,0,3]:
    print(f'y(x={x})={y(x,mean_a0,mean_a1):.5f} \t analytisch␣
 ↪sigma_y(x={x})={sigma_y_ana(x):.5f}')
    y_ = y(x,a[:,0],a[:,1])
    print(f'y(x={x})={np.mean(y_):.5f} \t numerical sigma_y (x={x})={np.std(y_):
 ↪.5f}\n')
```

```
y(x=-3)=-2.00000         analytisch sigma_y(x=-3)=0.76942
y(x=-3)=-1.99841         numerical sigma_y (x=-3)=0.76836

y(x=0)=1.00000   analytisch sigma_y(x=0)=0.20000
y(x=0)=1.00046   numerical sigma_y (x=0)=0.20003

y(x=3)=4.00000   analytisch sigma_y(x=3)=0.45607
y(x=3)=3.99934   numerical sigma_y (x=3)=0.45620
```

[45]:
```
# 1000 simulations
a = rng.multivariate_normal(mean = [mean_a0, mean_a1], cov=cov, size=1000)
for x in [-3,0,3]:
    print(f'y(x={x})={y(x,mean_a0,mean_a1):.5f} \t analytisch␣
 ↪sigma_y(x={x})={sigma_y_ana(x):.5f}')
    y_ = y(x,a[:,0],a[:,1])
    print(f'y(x={x})={np.mean(y_):.5f} \t numerical sigma_y (x={x})={np.std(y_):
 ↪.5f}\n')
```

```
y(x=-3)=-2.00000         analytisch sigma_y(x=-3)=0.76942
y(x=-3)=-1.98456         numerical sigma_y (x=-3)=0.77488

y(x=0)=1.00000   analytisch sigma_y(x=0)=0.20000
y(x=0)=1.00463   numerical sigma_y (x=0)=0.19543

y(x=3)=4.00000   analytisch sigma_y(x=3)=0.45607
y(x=3)=3.99382   numerical sigma_y (x=3)=0.46734
```

0.2.1 The quality of the numerical solution depends heavily on the size of the Monte Carlo simulation. One should consider how precise the results are meant to be and whether an analytical solution is feasible.

[ ]:
```
```