UNIVERSITEIT VAN AMSTERDAM

Kansrekening en Statistiek

LAB-4

Authors:
Abe Wiersma

29 april 2015

19 Exercise: Transforming multivariate rv's

$$X = (X_1, X_2)^T$$

$$\mu_{\vec{X}} = \begin{pmatrix} 1 \\ -1 \end{pmatrix} \ \mathit{and} \ \Sigma_{\vec{X}} = \begin{pmatrix} 1 & 0 \\ 0 & 4 \end{pmatrix}$$

$$E(\vec{Z}) = E\begin{pmatrix} Z_1 \\ Z_2 \end{pmatrix} = E\begin{pmatrix} X_1 + X_2 \\ X_1 - X_2 \end{pmatrix} = \begin{pmatrix} E(X_1 + X_2) \\ E(X_1 - X_2) \end{pmatrix} = \begin{pmatrix} E(X_1) + E(X_2) \\ E(X_1) - E(X_2) \end{pmatrix} = \begin{pmatrix} 1 + -1 \\ 1 - -1 \end{pmatrix}$$

$$E(\vec{Z}) = \begin{pmatrix} 0 \\ 2 \end{pmatrix}$$

$$Cov(\vec{Z}) = \begin{pmatrix} Cov(Z_1, Z_1) & Cov(Z_1, Z_2) \\ Cov(Z_2, Z_1) & Cov(Z_2, Z_2) \end{pmatrix}$$

$$= \begin{pmatrix} Var(X_1 + X_2) & Cov(X_1 + X_2, X_1 - X_2) \\ Cov(X_1 + X_2, X_1 - X_2) & Var(X_1 - X_2) \end{pmatrix}$$

$$= \begin{pmatrix} Var(X_1 + X_2) & Cov(X_1, X_1 - X_2) + Cov(X_2, X_1 - X_2) \\ Cov(X_1, X_1 - X_2) + Cov(X_2, X_1 - X_2) & Var(X_1 - X_2) \end{pmatrix}$$

$$= \begin{pmatrix} Var(X_1 + X_2) & Cov(X_1, X_1) - Cov(X_2, X_2) \\ Cov(X_1, X_1) - Cov(X_2, X_2) & Var(X_1 - X_2) \end{pmatrix}$$

$$= \begin{pmatrix} E((X_1 + X_2)^2) - (E(X_1 + X_2))^2 & Var(X_1) - Var(X_2) \\ Var(X_1) - Var(X_2) & E((X_1 - X_2)^2) - (E(X_1 - X_2))^2 \end{pmatrix}$$

$$= \begin{pmatrix} E(X_1^2 + X_2^2 + 2X_1X_2) - E(Z_1)^2 & Var(X_1) - Var(X_2) \\ Var(X_1) - Var(X_2) & E(X_1^2 + X_2^2 - 2X_1X_2) - E(Z_2)^2 \end{pmatrix}$$

$$= \begin{pmatrix} E(X_1^2) + E(X_2^2) + 2E(X_1X_2) - E(Z_1)^2 & Var(X_1) - Var(X_2) \\ Var(X_1) - Var(X_2) & E(X_1^2) + E(X_2^2) - 2E(X_1X_2) - E(Z_2)^2 \end{pmatrix}$$

The variance of X_1 and X_2 can be read from the covariance matrix $\vec{X}$. The expected value of X_1 and X_2 can also be found above, furthermore the Covariance matrix $\Sigma_{\vec{X}}$ also shows that X_1 and X_2 are independent.

$$= \begin{pmatrix} E(X_1^2) + E(X_2^2) + 2(E(X_1)E(X_2)) - 0 & 1 - 4 \\ 1 - 4 & E(X_1^2) + E(X_2^2) - 2(E(X_1)E(X_2)) - 4 \end{pmatrix}$$

$$= \begin{pmatrix} Var(X_1) + E(X_1)^2 + Var(X_2) + E(X_2)^2 - 2 & -3 \\ -3 & Var(X_1) + E(X_1)^2 + Var(X_2) + E(X_2)^2 + 2 - 4 \end{pmatrix}$$

$$= \begin{pmatrix} 1 + 1 + 4 + 1 - 2 & -3 \\ -3 & 1 + 1 + 4 + 1 + 2 - 4 \end{pmatrix}$$

$$= \begin{pmatrix} 5 & -3 \\ -3 & 5 \end{pmatrix}$$

You can conclude Z_1 and Z_2 are dependent cause $Cov(Z_1, Z_2) \neq 0$

20 Exercise: Correlation vs Dependence

$$X = (X_1, X_2)^T$$

$$\mu_{\vec{X}} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \; and \; \Sigma_{\vec{X}} = \begin{pmatrix} 1 & 0 \\ 0 & c \end{pmatrix}$$

$$\mu_{\vec{Z}} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$Cov(\vec{Z}) = \begin{pmatrix} Cov(Z_1, Z_1) & Cov(Z_1, Z_2) \\ Cov(Z_2, Z_1) & Cov(Z_2, Z_2) \end{pmatrix}$$

$$\sigma_{1,1} = Var(X_1) + E(X_1)^2 + Var(X_2) + E(X_2)^2 + 2(E(X_1)E(X_2)) - E(Z_1)^2$$

$$\sigma_{1,2} = Var(X_1) - Var(X_2)$$

$$\sigma_{2,1} = Var(X_1) - Var(X_2)$$

$$\sigma_{2,2} = Var(X_1) + E(X_1)^2 + Var(X_2) + E(X_2)^2 - 2(E(X_1)E(X_2)) - E(Z_2)^2$$

$$\sigma_{1,1} = 1 + 0^2 + c + 0^2 + 2(0 * 0) - 0^2$$

$$\sigma_{1,2} = 1 - c$$

$$\sigma_{2,1} = 1 - c$$

$$\sigma_{2,2} = 1 + 0^2 + c + 0^2 - 2(0 * 0) - 0^2$$

$$Cov(\vec{Z}) = \begin{pmatrix} 1 + c & 1 - c \\ 1 - c & 1 + c \end{pmatrix}$$

$$Corr(\vec{Z}) = \begin{pmatrix} 1 & \frac{1-c}{\sqrt{(1+c)(1+c)}} \\ \frac{1-c}{\sqrt{(1+c)(1+c)}} & 1 \end{pmatrix}$$

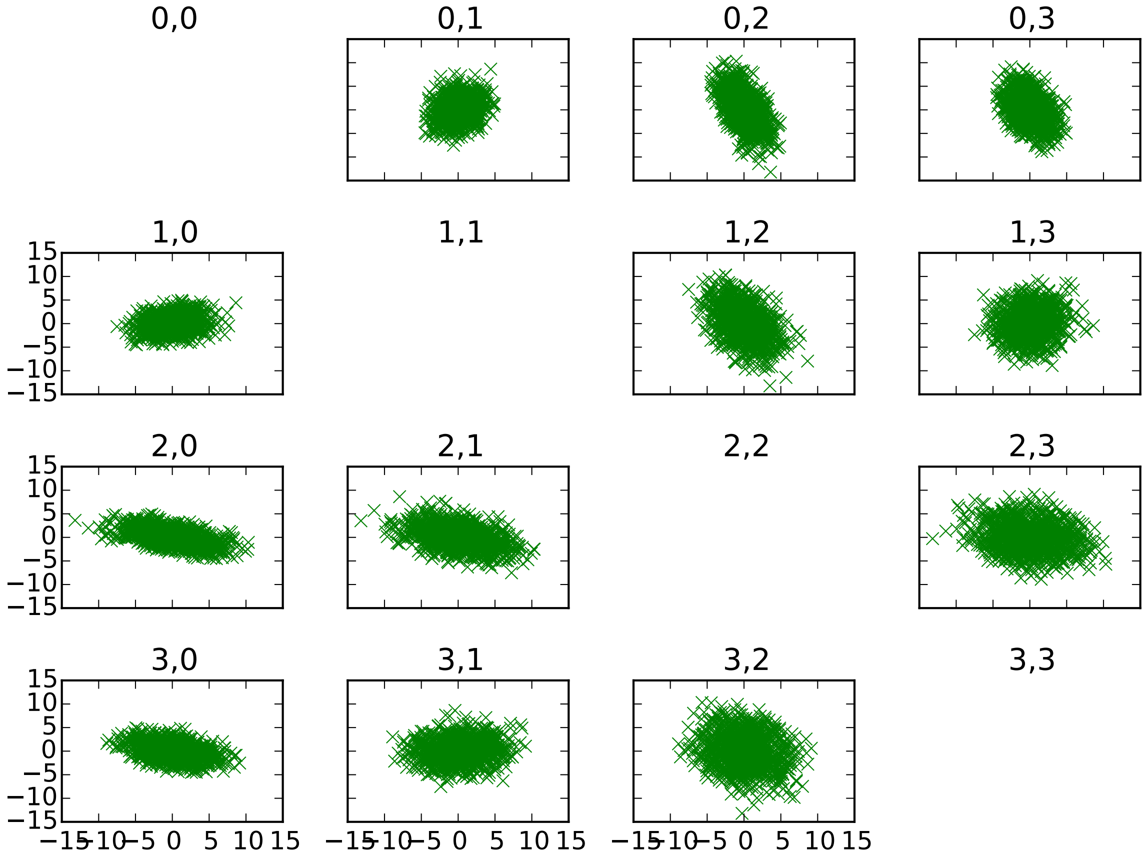
Because $c > 0$ You can leave out the sqrt and the square.

$$Corr(\vec{Z}) = \begin{pmatrix} 1 & \frac{1-c}{1+c} \\ \frac{1-c}{1+c} & 1 \end{pmatrix}$$

For every $c > 1$ there is a correlation, because of non-zero $p_{2,1}$ and $p_{1,2}$.

21 Exercise: Generating & Visualizing Samples from N(μ, Σ)

```
import pylab as plt

n = 1000
mu = [[0],
      [0],
      [0],
      [0]]
Sigma = [[3.01602775, 1.02746769, -3.60224613, -2.08792829],
         [1.02746769, 5.65146472, -3.98616664, 0.48723704],
         [-3.60224613, -3.98616664, 13.04508284, -1.59255406],
         [-2.08792829, 0.48723704, -1.59255406, 8.28742469]]

d, U = plt.eig(Sigma)  # Sigma = U L Ut
L = plt.diagflat(d)
A = plt.dot(U, plt.sqrt(L))  # required transform matrix

X = plt.randn(4, n)  # 4xn matrix with each element ~ N(0,1)
Y = plt.dot(A, X) + plt.tile(mu, n)  # 4xn each column vector ~N(mu,Sigma)

f, axarr = plt.subplots(4, 4, sharex=True, sharey=True)
for i in range(0, len(Y)):
    for j in range(0, len(Y)):
        if(i == j):
            axarr[i][j].set_title(str(i) + ',' + str(j))
            axarr[i][j].axis('off')
            continue

        axarr[i][j].plot(Y[i], Y[j], 'xg')
        axarr[i][j].set_title(str(i) + ',' + str(j))

plt.setp([a.get_xticklabels() for a in axarr[0, :]], visible=False)
plt.setp([a.get_yticklabels() for a in axarr[:, 1]], visible=False)
plt.tight_layout()
# scatter plot of N(mu,Sigma) distribution
plt.savefig('scatter.pdf')
plt.clf()
```


0,0
0,1
0,2
0,3
1,0
1,1
1,2
1,3
2,0
2,1
2,2
2,3
3,0
3,1
3,2
3,3
15
10
5
0
−5
−10
−15
−15 −10 −5 0 5 10 15

22 Exercise: Estimating mean($\vec{x}$) and covariance matrix (S)

- Below you can find the Maximum Likelihood estimator of the covariance matrix from a sample of n observations.

$$\widehat{\mathbf{\Sigma}} = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i - \overline{\mathbf{x}})(\mathbf{x}_i - \overline{\mathbf{x}})^T$$

  It was easy to see that when the program was run with more draws the relative accuracy of the estimated mean and covariance matrix increased by differing less of the original μ and Σ.

-
```
import pylab as plt

n = 1000
mu = [[0],
      [0],
      [0],
      [0]]
Sigma = [[3.01602775, 1.02746769, -3.60224613, -2.08792829],
         [1.02746769, 5.65146472, -3.98616664, 0.48723704],
         [-3.60224613, -3.98616664, 13.04508284, -1.59255406],
         [-2.08792829, 0.48723704, -1.59255406, 8.28742469]]

d, U = plt.eig(Sigma)  # Sigma = U L Ut
L = plt.diagflat(d)
A = plt.dot(U, plt.sqrt(L))  # Required transform matrix.

X = plt.randn(4, n)  # 4*n matrix with each element ~ N(0,1)
# 4*n each column vector ~N(mu,Sigma), random draws from distribution.
Y = plt.dot(A, X) + plt.tile(mu, n)

Ybar = [[avg] for avg in plt.mean(Y, 1)]  # Mean along the 1 axis.
Yzm = Y - plt.tile(Ybar, n)  # Subtract mean from each column.
# Estimator for covariance matrix.
S = plt.dot(Yzm, plt.transpose(Yzm)) / n - 1

print(Ybar, S)
```