

### 3.10 Multivariate Probability and Correlation



Thus far, we have considered probability density and mass functions (PDFs and PMFs) of only one random variable. But, of course, often we measure multiple random variables  $X_1, X_2, \dots, X_n$  during a single event, meaning a measurement consists of determining values  $x_1, x_2, \dots, x_n$  of these random variables.

We can consider an  $n$ -tuple of continuous random variables to form a sample space  $\Omega = \mathbb{R}^n$  on which a multivariate function  $f: \mathbb{R}^n \rightarrow \mathbb{R}$ , called the **joint PDF** assigns a probability density to each outcome  $x \in \mathbb{R}^n$ . The joint PDF must be greater than or equal to zero for all  $x \in \mathbb{R}^n$ , the multiple integral over  $\Omega$  must be unity, and the multiple integral over a subset of the sample space  $A \subset \Omega$  is the probability of the event  $A$ .

We can consider an  $n$ -tuple of discrete random variables to form a sample space  $\mathbb{N}_0^n$  on which a multivariate function  $f: \mathbb{N}_0^n \rightarrow \mathbb{R}$ , called the **joint PMF** assigns a probability to each outcome  $x \in \mathbb{N}_0^n$ . The joint PMF must be greater than or equal to zero for all  $x \in \mathbb{N}_0^n$ , the multiple sum over  $\Omega$  must be unity, and the multiple sum over a subset of the sample space  $A \subset \Omega$  is the probability of the event  $A$ .

#### Example 3.12

Let's visualize multivariate PDFs by plotting a bivariate gaussian using the `scipy.stats` function `multivariate_normal`

We proceed in Python. First, load packages:

```
import numpy as np
from scipy.stats import multivariate_normal
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
```

Define the mean and covariance matrix for a bivariate Gaussian distribution.

```
mu = [10, 20] # Mean
Sigma = [[1, 0], [0, 0.2]] # Covariance matrix
```

Generate grid points as input for the PDF.

```
x1_a = np.linspace(mu[0] - 5 * np.sqrt(Sigma[0][0]), mu[0] + 5 * np.sqrt(Sigma[0][0]))
x2_a = np.linspace(mu[1] - 5 * np.sqrt(Sigma[1][1]), mu[1] + 5 * np.sqrt(Sigma[1][1]))
```

Create a meshgrid.

```
X1, X2 = np.meshgrid(x1_a, x2_a)
```

Calculate the PDF.

```
pos = np.dstack((X1, X2))
rv = multivariate_normal(mu, Sigma)
f = rv.pdf(pos)
```

Plot the PDF.

```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
p = ax.plot_surface(X1, X2, f, cmap='copper')
ax.set_xlabel(r'$x_1$', fontsize=12)
ax.set_ylabel(r'$x_2$', fontsize=12)
ax.set_zlabel(r'$f(x_1, x_2)$', fontsize=12)
plt.show()
```

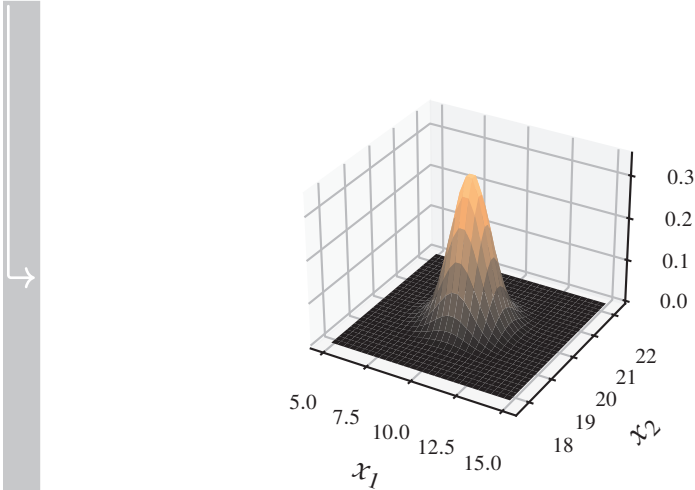


Figure 3.15. Bivariate Gaussian PDF.

The plot shows the PDF of a bivariate Gaussian distribution. Pretty neat, right?

### 3.10.1 Marginal Probability

The **marginal PDF** of a multivariate PDF is the PDF of some subspace of  $\Omega$  after one or more variables have been “integrated out,” such that a fewer number of random variables remain. Of course, these marginal PDFs must have the same properties of any PDF, such as integrating to unity.

#### Example 3.13

Let’s demonstrate this by numerically integrating over  $x_2$  in the bivariate Gaussian, above.

Continuing from where we left off, let’s integrate.

```
f1 = np.trapz(f.T, x2_a, axis=1) # Trapezoidal integration
```

Let’s plot the marginal PDF.

```

fig, ax = plt.subplots()
ax.plot(x1_a, f1, linewidth=2)
ax.set_xlabel(r'$x_1$')
ax.set_ylabel(r'$g(x_1)=\int_{-\infty}^{\infty} f(x_1,x_2) d x_2$')
plt.show()
↳ [matplotlib.lines.Line2D at 0x1680f0bd0]
↳ Text(1, 0, '$x_1$')
↳ Text(0, 0.5, '$g(x_1)=\int_{-\infty}^{\infty} f(x_1,x_2) d x_2$')

```

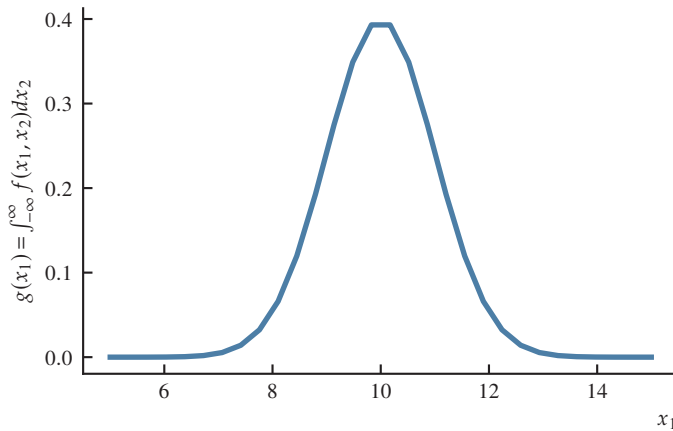


Figure 3.16. Marginal PDF of a bivariate Gaussian distribution.

We should probably verify that this integrates to one.

```

integral_value = np.trapz(f1, x1_a)
print(f'integral over x_1 = {integral_value:.7f}')

```

```

| integral over x_1 = 0.9999986

```

Not bad.

### 3.10.2 Covariance

Very often, especially in machine learning applications, the question about two random variables  $X$  and  $Y$  is: how do they co-vary? That is what is their **covariance**, defined as

$$\begin{aligned}\text{Cov}[X, Y] &\equiv E((X - \mu_X)(Y - \mu_Y)) \\ &= E(XY) - \mu_X \mu_Y.\end{aligned}$$

Note that when  $X = Y$ , the covariance is just the variance. When a covariance is large and positive, it is an indication that the random variables are *strongly correlated*. When it is large and negative, they are *strongly anti-correlated*. Zero covariance means the variables are *uncorrelated*. In fact, **correlation** is defined as

$$\text{Cor}[X, Y] = \frac{\text{Cov}[X, Y]}{\sqrt{\text{Var}[X] \text{Var}[Y]}}.$$

This is essentially the covariance “normalized” to the interval  $[-1, 1]$ .

**3.10.2.1 Sample Covariance** As with the other statistics we’ve considered, covariance can be estimated from measurement. The estimate, called the **sample covariance**  $q_{XY}$ , of random variables  $X$  and  $Y$  with sample size  $N$  is given by

$$q_{XY} = \frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{X})(y_i - \bar{Y}).$$

**3.10.2.2 Multivariate Covariance** With  $n$  random variables  $X_i$ , one can compute the covariance of each pair. It is common practice to define an  $n \times n$  matrix of covariances called the **covariance matrix**  $\Sigma$  such that each pair’s covariance

$$\text{Cov}[X_i, X_j]$$

appears in its row-column combination (making it symmetric), as shown below.

$$\Sigma = \begin{bmatrix} \text{Cov}[X_1, X_1] & \text{Cov}[X_1, X_2] & \cdots & \text{Cov}[X_1, X_n] \\ \text{Cov}[X_2, X_1] & \text{Cov}[X_2, X_2] & & \text{Cov}[X_2, X_n] \\ \vdots & & \ddots & \vdots \\ \text{Cov}[X_n, X_1] & \text{Cov}[X_n, X_2] & \cdots & \text{Cov}[X_n, X_n] \end{bmatrix}$$

The multivariate **sample covariance matrix**  $Q$  is the same as above, but with sample covariances  $q_{X_i X_j}$ .

Both covariance matrices have correlation analogs.

### Example 3.14

Let's use a dataset from the Scikit-Learn package with multivariate data on the attributes of wine. Compute the sample covariance and correlation matrices. Plot variables pairwise and color them with the corresponding correlation.

Load the necessary libraries.

```
import numpy as np
from sklearn.datasets import load_wine
import matplotlib.pyplot as plt
from matplotlib.colors import Normalize
import matplotlib.cm as cm
```

Load the dataset and print the feature names.

```
data = load_wine()
print(f"Features: {data.feature_names}")

Features: ['alcohol', 'malic_acid', 'ash', 'alcalinity_of_ash',
↪ 'magnesium', 'total_phenols', 'flavanoids',
↪ 'nonflavanoid_phenols', 'proanthocyanins', 'color_intensity',
↪ 'hue', 'od280/od315_of_diluted_wines', 'proline']
```

Select a list of features to analyze and select the corresponding data.

```
features = [
    'alcohol', 'malic_acid', 'ash', 'magnesium',
    'total_phenols', 'flavanoids'
]
X = data.data[
    :, [data.feature_names.index(f) for f in features]
]
```

Compute the sample covariance and correlation matrices.

```
cov = np.cov(X.T) # Covariance matrix
cor = np.corrcoef(X.T) # Correlation matrix (normalized covariance)
print(f"Covariance matrix:\n{cov}")
print(f"Correlation matrix:\n{cor}")
```

```

Covariance matrix:
[[ 6.59062328e-01  8.56113090e-02  4.71151590e-02  3.13987812e+00
   1.46887218e-01  1.92033222e-01]
 [ 8.56113090e-02  1.24801540e+00  5.02770393e-02 -8.70779534e-01
 -2.34337723e-01 -4.58630366e-01]
 [ 4.71151590e-02  5.02770393e-02  7.52646353e-02  1.12293658e+00
  2.21455913e-02  3.15347299e-02]
 [ 3.13987812e+00 -8.70779534e-01  1.12293658e+00  2.03989335e+02
  1.91646988e+00  2.79308703e+00]
 [ 1.46887218e-01 -2.34337723e-01  2.21455913e-02  1.91646988e+00
  3.91689535e-01  5.40470422e-01]
 [ 1.92033222e-01 -4.58630366e-01  3.15347299e-02  2.79308703e+00
  5.40470422e-01  9.97718673e-01]]

Correlation matrix:
[[ 1.          0.09439694  0.2115446  0.27079823  0.28910112
  ↪ 0.23681493]
 [ 0.09439694  1.          0.16404547 -0.0545751 -0.335167
  ↪ -0.41100659]
 [ 0.2115446  0.16404547  1.          0.28658669  0.12897954
  ↪ 0.11507728]
 [ 0.27079823 -0.0545751  0.28658669  1.          0.21440123
  ↪ 0.19578377]
 [ 0.28910112 -0.335167  0.12897954  0.21440123  1.
  ↪ 0.8645635 ]
 [ 0.23681493 -0.41100659  0.11507728  0.19578377  0.8645635  1.
  ↪ ]]

```

Plot the data pairings with color corresponding to the correlation matrix.

```
fig, ax = plt.subplots(cor.shape[0], cor.shape[1], figsize=(10, 10))
norm = Normalize(vmin=-1, vmax=1)
cmap = cm.coolwarm
scatter = np.empty(cor.shape, dtype=object)
for i in range(cor.shape[0]):
    for j in range(cor.shape[1]):
        scatter[i, j] = ax[i, j].scatter(
            X[:, i], X[:, j],
            c=cor[i, j] * np.ones(X.shape[0]), cmap=cmap, norm=norm,
            s=0.5 # Point size
        )
    if i == cor.shape[0] - 1:
        ax[i, j].set_xlabel(
            features[j].replace("_", " "), rotation=45, ha='right')
    if j == 0:
        ax[i, j].set_ylabel(
            features[i].replace("_", " "), rotation=0, ha='right')
    ax[i, j].set_xticks([])
    ax[i, j].set_yticks([])
plt.tight_layout()
cbar = fig.colorbar(scatter[0, 0], ax=ax, orientation='horizontal')
plt.show()
```

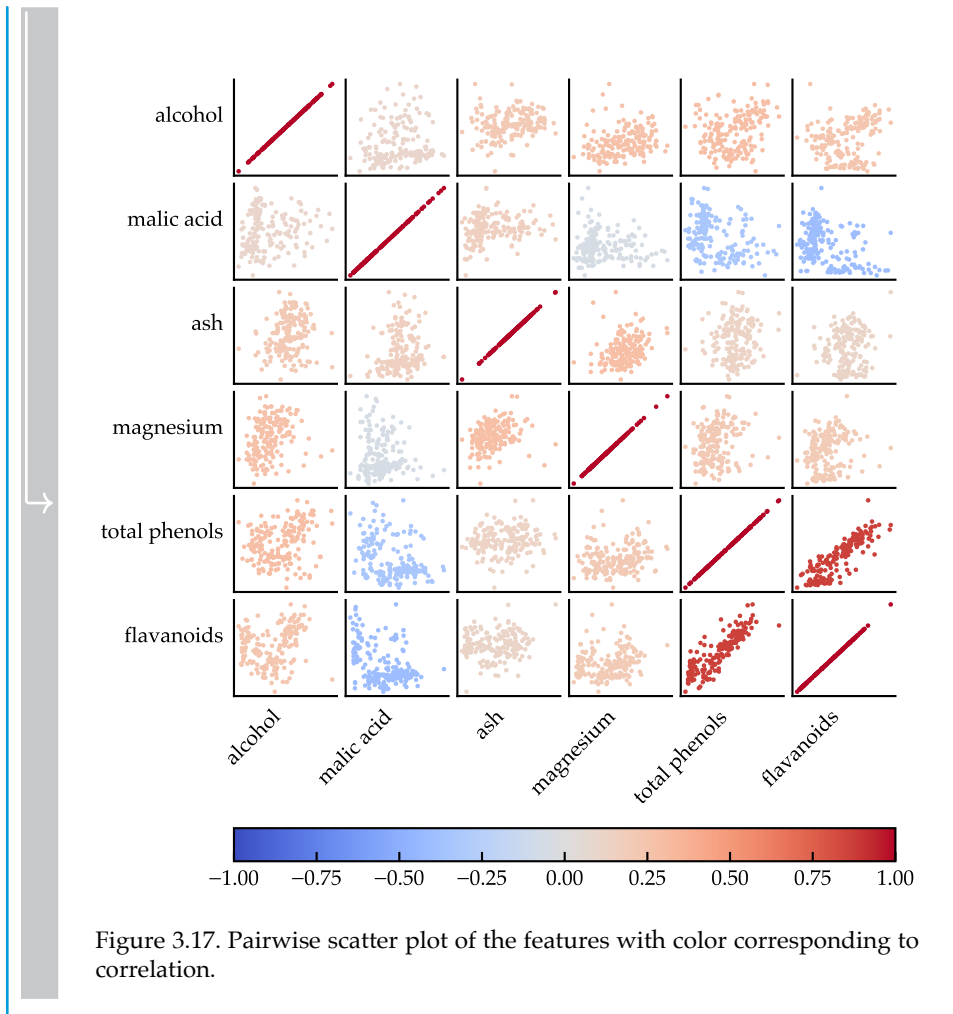


Figure 3.17. Pairwise scatter plot of the features with color corresponding to correlation.

### 3.10.3 Conditional Probability and Dependence

*Independent variables are uncorrelated. However, uncorrelated variables may or may not be independent. Therefore, we cannot use correlation alone as a test for independence. For instance, for random variables  $X$  and  $Y$ , where  $X$  has some even distribution and  $Y = X^2$ , clearly the variables are dependent. However, they are also uncorrelated (due to symmetry).*



### Example 3.15

Using a uniform distribution  $U(-1, 1)$ , show that  $X$  and  $Y$  are uncorrelated (but dependent) with  $Y = X^2$  with some sampling. We compute the correlation for different sample sizes.

Load the necessary libraries.

```
import numpy as np
import matplotlib.pyplot as plt
```

Generate the data for  $x$  and  $y$ .

```
N_a = np.round(np.linspace(10, 500, 100)).astype(int) # Sample sizes
qc_a = np.full(N_a.shape, np.nan) # Correlation initialization
np.random.seed(6) # Seed for reproducibility
x_a = -1 + 2 * np.random.rand(max(N_a)) # Uniform random numbers
y_a = x_a ** 2 #  $y = x^2$ 
```

Calculate the cross-correlation.

```
for i in range(len(N_a)):
    q = np.cov(x_a[:N_a[i]], y_a[:N_a[i]])
    qc = np.corrcoef(x_a[:N_a[i]], y_a[:N_a[i]])
    qc_a[i] = qc[0, 1] # "cross" correlation
```

Plot the absolute cross correlation as a function of sample size.

```
fig, ax = plt.subplots()
p, = ax.plot(N_a, np.abs(qc_a), linewidth=2)
ax.set_xlabel(r'Sample size $N$')
ax.set_ylabel(r'Absolute sample correlation')
ax.set_ylim(bottom=0)
plt.show()
```

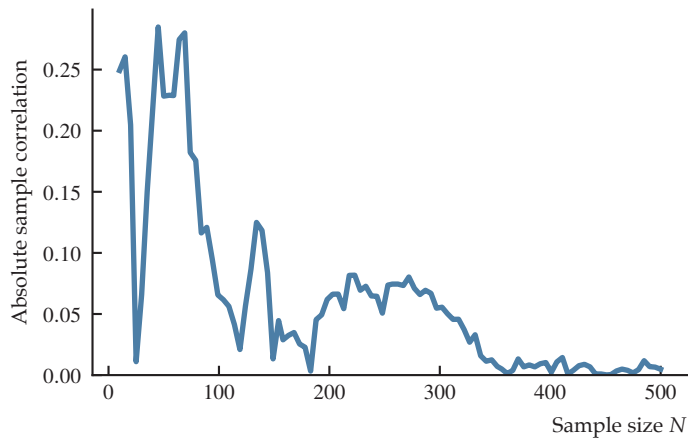



Figure 3.18. Correlation between  $x$  and  $y$  as a function of sample size.


The absolute values of the correlations are shown in the figure. Note that we should probably average several such curves to estimate how the correlation would drop off with  $N$ , but the single curve describes our understanding that the correlation, in fact, approaches zero in the large-sample limit.

---

### 3.11 Problems



**Problem 3.1**  **GRAIN** Several physical processes can be modeled with a *random walk*: a process of iteratively changing a quantity by some random amount. Infinitely many variations are possible, but common factors of variation include probability distribution, step size, dimensionality (e.g. one-dimensional, two-dimensional, etc.), and coordinate system. Graphical representations of these walks can be beautiful. Develop a computer program that generates random walks and corresponding graphics. Do it well and call it art because it is.

**Problem 3.2**  **FREE** Consider the defective spring problem from example 3.4. One way to improve the probability of a true positive test (i.e., the sensitivity) is to add a second test for which a positive event is called  $C$ . Again assuming that the sensitivity and specificity are equal for tests  $B$  and  $C$ , and that the sensitivity of test  $B$  is  $P(B|A) = 0.995$  what is the required sensitivity for test  $C$ ? Clearly state any assumptions.