

```
import numpy as np
```

```
# https://stats.stackexchange.com/questions/120179/generating-data-with-a-given-sample-covar
```

```
def gen_exact(mean=None, sigma=None, size=None, rng=None):
```

```
    if (mean is None or sigma is None or size is None or rng is None):  
        return None
```

```
    # Generate size cases
```

```
    # rng = np.random.RandomState(seed)
```

```
    X = rng.multivariate_normal(mean, sigma, size=size).T
```

```
    # Subtract the mean from each variable
```

```
    for n in range(X.shape[0]):
```

```
        X[n] = X[n] - X[n].mean()
```

```
    # Make each variable in X orthogonal to one another
```

```
    L_inv = np.linalg.cholesky(np.cov(X, bias = True))
```

```
    L_inv = np.linalg.inv(L_inv)
```

```
    X = np.dot(L_inv, X)
```

```
    # Rescale X to exactly match Sigma
```

```
    L = np.linalg.cholesky(sigma)
```

```
    X = np.dot(L, X)
```

```
    # Add the mean back into each variable
```

```
    for n in range(X.shape[0]):
```

```
        X[n] = X[n] + mean[n]
```

```
    # The covariance of the generated data should match Sigma
```

```
    cov = np.cov(X, bias = True)
```

```
    X = X.T
```

```
    return X
```

```
def gen_inexact(mean=None, sigma=None, size=None, rng=None):
```

```
    if (mean is None or sigma is None or size is None or rng is None):  
        return None
```

```
    # Generate size cases
```

```
    # rng = np.random.RandomState(seed)
```

```
    X = rng.multivariate_normal(mean, sigma, size=size).T
```

```
    # Make each variable in X orthogonal to one another
```

```
    L_inv = np.linalg.cholesky(np.cov(X, bias = True))
```

```
    L_inv = np.linalg.inv(L_inv)
```

```
X = np.dot(L_inv, X)

# Rescale X to exactly match Sigma
L = np.linalg.cholesky(sigma)
X = np.dot(L, X)

# The covariance of the generated data should match Sigma
cov = np.cov(X, bias = True)
X = X.T
return X
```