Постановка задачи

Требуется

In [3]:
```
from scipy.stats import multivariate_normal
import matplotlib.pyplot as plt
import numpy as np
```

In [16]:
```
n = 1000

m1 = np.array([1, 2, 3])
m2 = np.array([2, 5, -1])

s = np.array([[2, 0, 1],
              [0, 3, 0.5],
              [1, 0.5, 4]])
s_ = (s+0.5)*1.5
```

In [62]:
```
print(m2 @ s @ m2)
S = 0

for i in range(3):
    for j in range(3):
        S += s[i, j]*m2[i]*m2[j]

print(S)
```

```
78.0
78.0
```

In [70]:
```
class Research:
    def __init__(self, mean1, mean2, cov, q1, size, test_set=None):
        self.sample1 = []
        self.sample2 = []
        self.m1 = mean1
        self.m2 = mean2
        self.cov = cov
        self.q1 = q1
        self.q2 = 1 - q1
        self.n1 = int(size*q1)
        self.n2 = size - self.n1
        self.test_set = test_set

        self.m1_ = []
        self.m2_ = []
        self.cov_ = []
        self.alpha = []

        self.mz1_ = 0
        self.mz2_ = 0
        self.z_var = 0

    def generate_sample(self):
        self.sample1 = multivariate_normal(mean=self.m1, cov=self.cov).rvs(self.n1)
        self.sample2 = multivariate_normal(mean=self.m2, cov=self.cov).rvs(self.n2)

    def calc_estimates(self):
        self.m1_ = np.mean(self.sample1, axis = 0)
        self.m2_ = np.mean(self.sample2, axis=0)

        self.cov_ = (np.cov(self.sample1.T) + np.cov(self.sample2.T)) * 0.5

        self.alpha = np.linalg.inv(self.cov_) @ (self.m1_-self.m2_)

        self.mz1_ = np.dot(self.alpha, self.m1_)
        self.mz2_ = np.dot(self.alpha, self.m2_)
        self.z_var = self.alpha @ self.cov_ @ self.alpha

    def makhalanobis(self, unbiased=False):
        makh = abs(self.mz1_-self.mz2_) / self.z_var
        if unbiased:
            p = len(self.m1)
            makh = ((self.n1 + self.n2 - p - 3)/
                    (self.n1 + self.n2 - 2)*makh**2 - p*(1/self.n1 + 1/self.n2))**0.5
        return makh

    def predict(self, test_set=None):
        if not test_set:
            test_set = self.test_set
        if not test_set:
            print("Warning!\n\tTest set not specified!")
            return
```

```
  File "C:\Users\nikit\AppData\Local\Temp/ipykernel_14628/4048743989.py", line 49
    def predict(self):
                      ^
IndentationError: expected an indented block
```

In [68]:
```
R = Research(m1, m2, s, q1, n, n)
R.generate_sample()
R.n1
```

Out[68]:
```
500
```

In [69]:
```
R.calc_estimates()
```

```
10.481341029647895
10.481341029647895
```

In [22]:
```
np.linalg.inv(s) @ (m1-m2)
```

Out[22]:
```
array([-1.23170732, -1.24390244,  1.46341463])
```

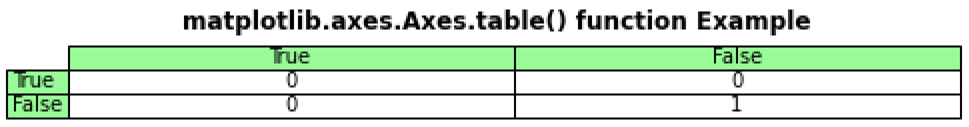
In [1]:
```
def makhalanobis(a, b, s):
    return abs(a-b)/s
```

In [14]:
```
val1 = ["True", "False"]
val2 = ["True", "False"]
val3 = [[str(c*r) for c in range(2)] for r in range(2)]

fig, ax = plt.subplots(1, 1, figsize=(8, 2))
ax.set_axis_off()
table = ax.table(
    cellText = val3,
    rowLabels = val2,
    colLabels = val1,
    rowColours =["palegreen"] * 2,
    colColours =["palegreen"] * 2,
    cellLoc ='center',
    loc ='upper left',
    fontsize=10)

ax.set_title('matplotlib.axes.Axes.table() function Example',
             fontweight ="bold")

plt.show()
```



matplotlib.axes.Axes.table() function Example

	True	False
True	0	0
False	0	1

In [2]:

```
h = 10
```

True\Predict	1	2
1	{{h}}	283
2	{{h}}	283

In [ ]:

```
```