Code Decomposition and Modularity

Analytics Vidhya

Modularity

Modularity

- Modularity promotes bite-size code

Modularity

- Modularity promotes bite-size code
- Avoid complex nested constructs if possible (loops, if-else)

Modularity

- Modularity promotes bite-size code
- Avoid complex nested constructs if possible (loops, if-else)
- Each block should performs at-most one action

Modularity

- Modularity promotes bite-size code
- Avoid complex nested constructs if possible (loops, if-else)
- Each block should performs at-most one action
- Improves readability

Modularity

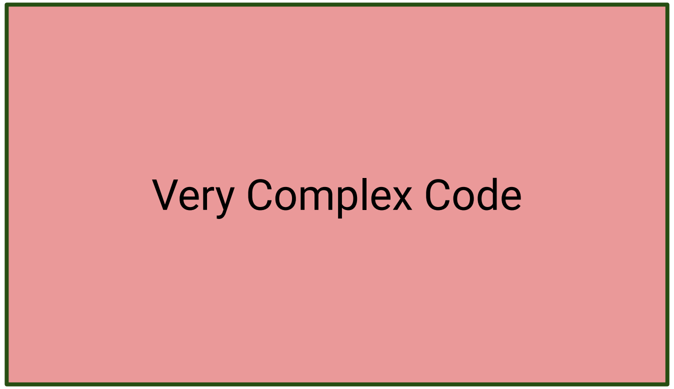
- Modularity promotes bite-size code
- Avoid complex nested constructs if possible (loops, if-else)
- Each block should performs at-most one action
- Improves readability
- Easier to resolve errors

Modularity

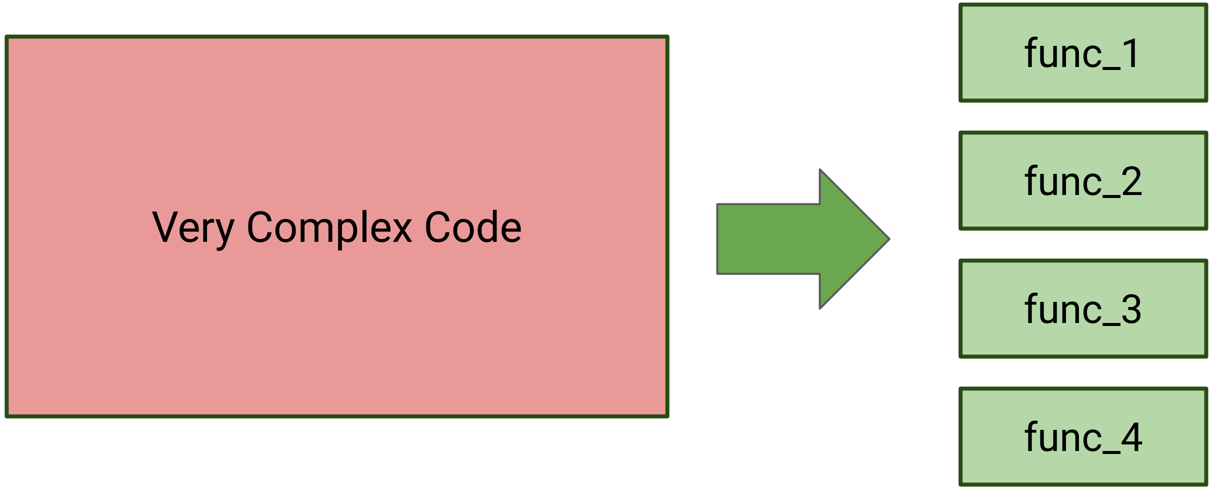
- Modularity promotes bite-size code
- Avoid complex nested constructs if possible (loops, if-else)
- Each block should performs at-most one action
- Improves readability
- Easier to resolve errors
- Easier to test and debug code

Modular Code

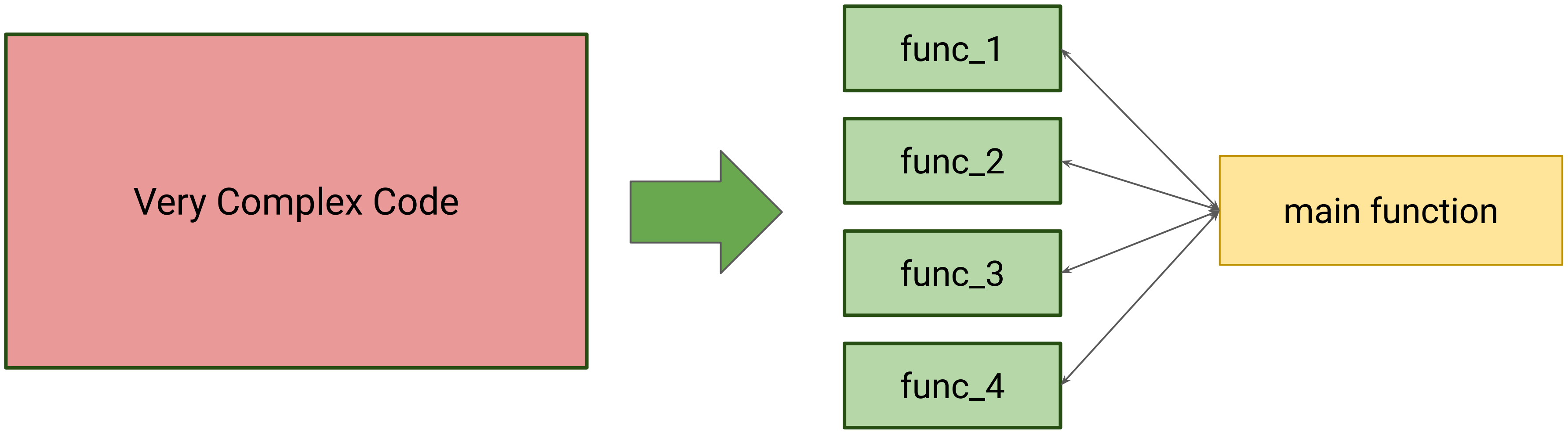
Modular Code



Modular Code

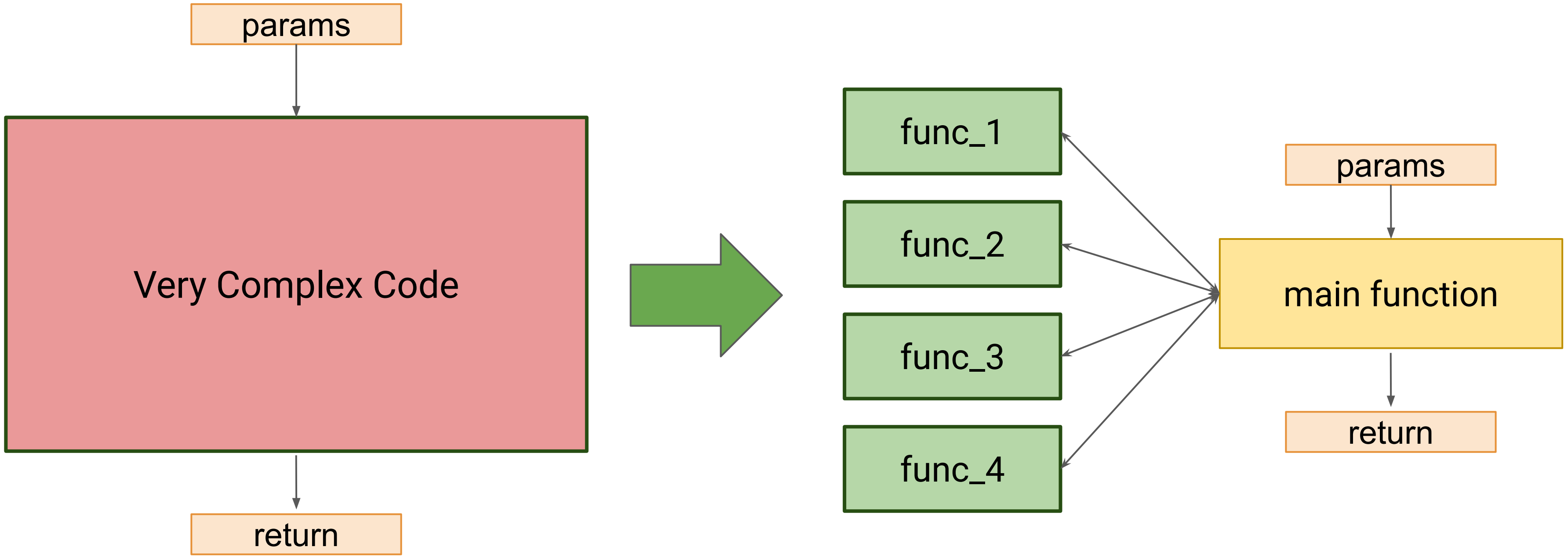


Modular Code



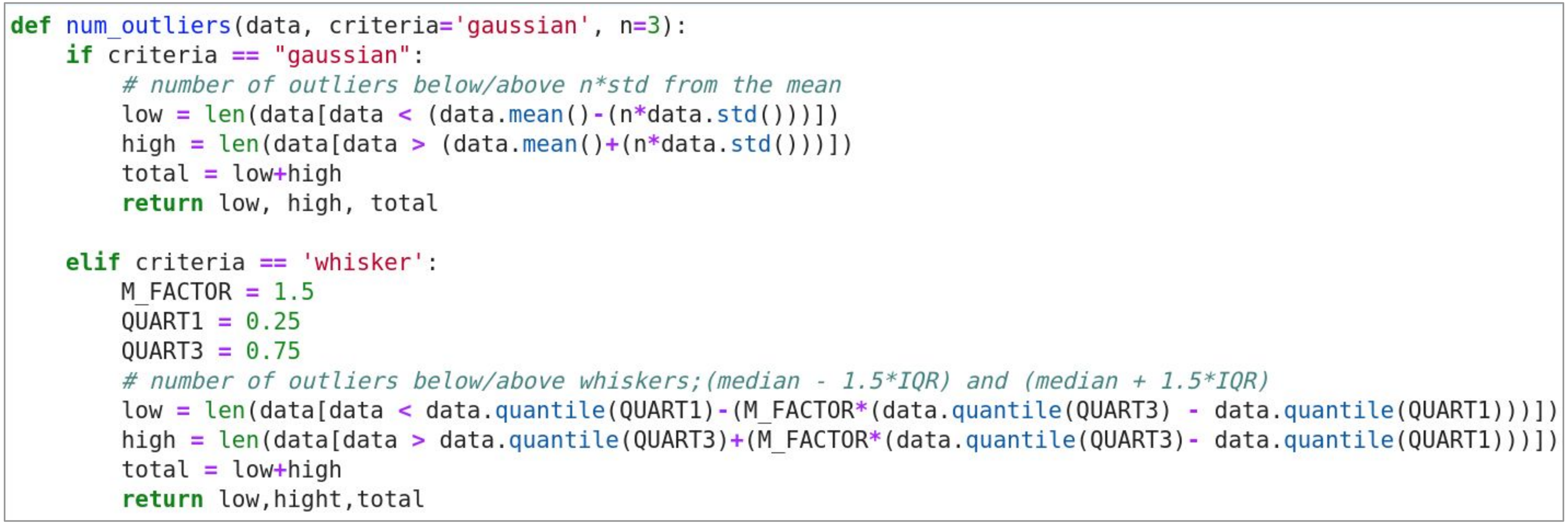
Analytics Vidhya

Modular Code



Analytics Vidhya

Decomposition of Code



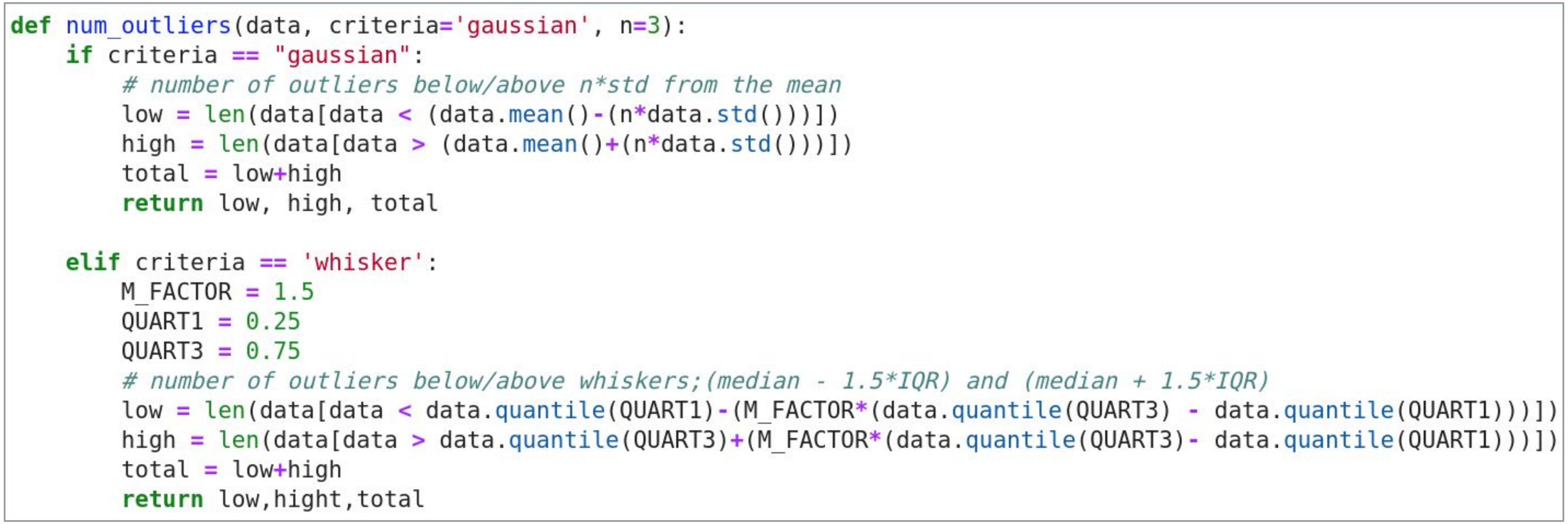
```python
def num_outliers(data, criteria='gaussian', n=3):
    if criteria == "gaussian":
        # number of outliers below/above n*std from the mean
        low = len(data[data < (data.mean()-(n*data.std()))])
        high = len(data[data > (data.mean()+(n*data.std()))])
        total = low+high
        return low, high, total

    elif criteria == 'whisker':
        M_FACTOR = 1.5
        QUART1 = 0.25
        QUART3 = 0.75
        # number of outliers below/above whiskers;(median - 1.5*IQR) and (median + 1.5*IQR)
        low = len(data[data < data.quantile(QUART1)-(M_FACTOR*(data.quantile(QUART3) - data.quantile(QUART1)))])
        high = len(data[data > data.quantile(QUART3)+(M_FACTOR*(data.quantile(QUART3)- data.quantile(QUART1)))])
        total = low+high
        return low,hight,total
```

Analytics Vidhya

Decomposition of Code

Trying to do too much at once



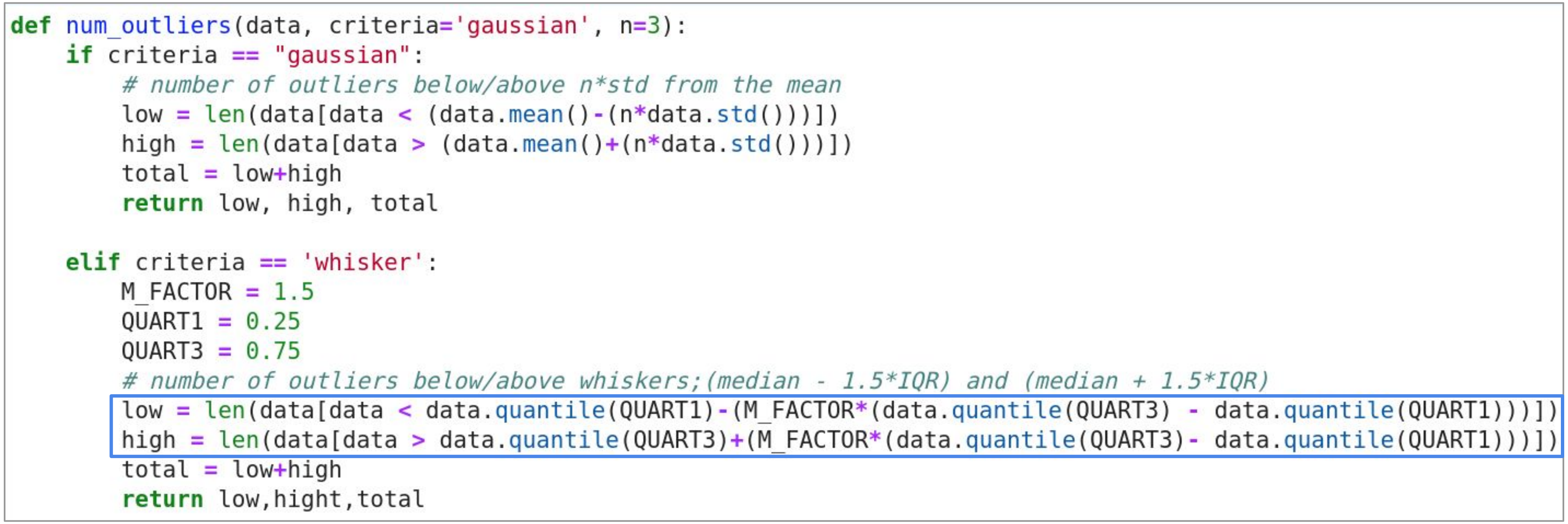
```python
def num_outliers(data, criteria='gaussian', n=3):
    if criteria == "gaussian":
        # number of outliers below/above n*std from the mean
        low = len(data[data < (data.mean()-(n*data.std()))])
        high = len(data[data > (data.mean()+(n*data.std()))])
        total = low+high
        return low, high, total

    elif criteria == 'whisker':
        M_FACTOR = 1.5
        QUART1 = 0.25
        QUART3 = 0.75
        # number of outliers below/above whiskers;(median - 1.5*IQR) and (median + 1.5*IQR)
        low = len(data[data < data.quantile(QUART1)-(M_FACTOR*(data.quantile(QUART3) - data.quantile(QUART1)))])
        high = len(data[data > data.quantile(QUART3)+(M_FACTOR*(data.quantile(QUART3)- data.quantile(QUART1)))])
        total = low+high
        return low,hight,total
```

Analytics Vidhya

Decomposition of Code

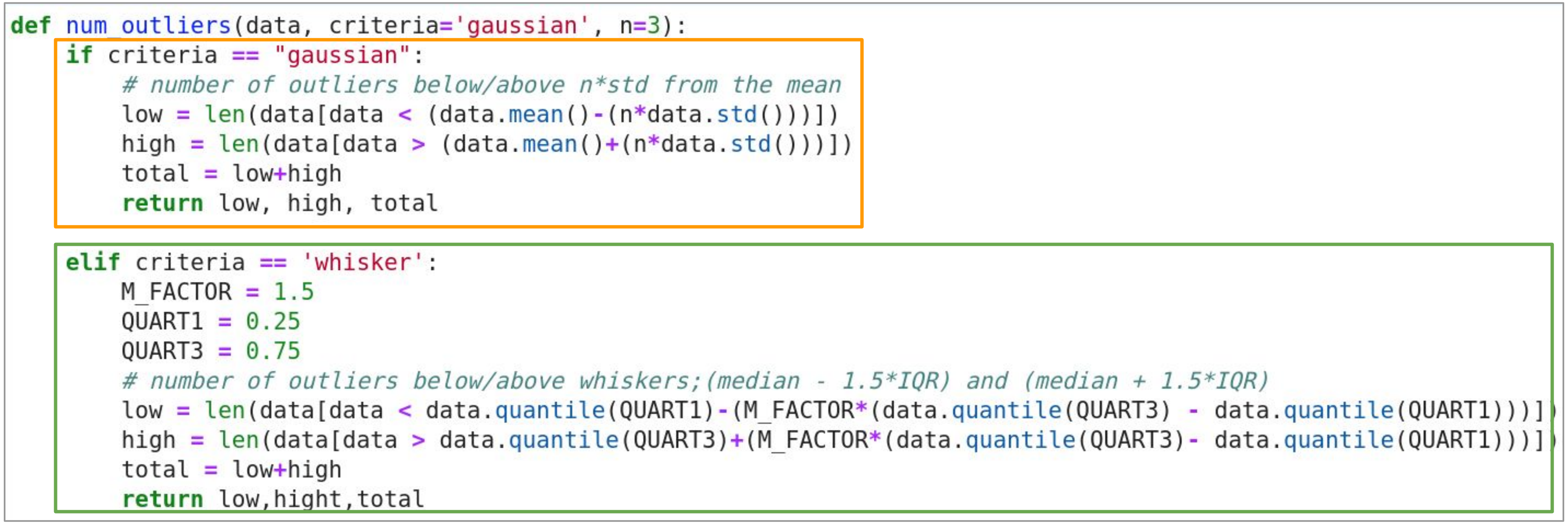
Trying to do too much at once



```python
def num_outliers(data, criteria='gaussian', n=3):
    if criteria == "gaussian":
        # number of outliers below/above n*std from the mean
        low = len(data[data < (data.mean()-(n*data.std()))])
        high = len(data[data > (data.mean()+(n*data.std()))])
        total = low+high
        return low, high, total

    elif criteria == 'whisker':
        M_FACTOR = 1.5
        QUART1 = 0.25
        QUART3 = 0.75
        # number of outliers below/above whiskers;(median - 1.5*IQR) and (median + 1.5*IQR)
        low = len(data[data < data.quantile(QUART1)-(M_FACTOR*(data.quantile(QUART3) - data.quantile(QUART1)))])
        high = len(data[data > data.quantile(QUART3)+(M_FACTOR*(data.quantile(QUART3)- data.quantile(QUART1)))])
        total = low+high
        return low,hight,total
```

Decomposition of Code

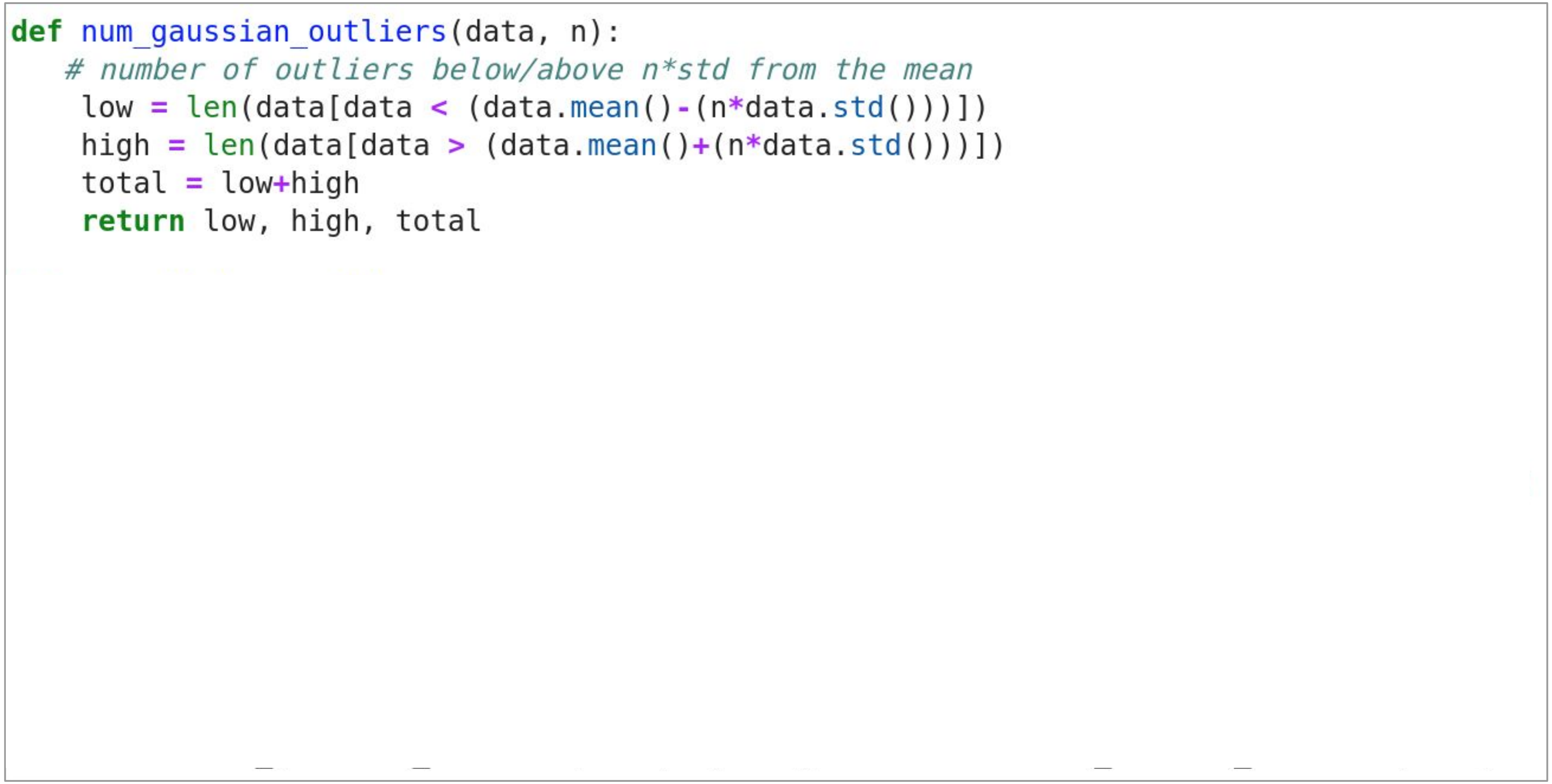


```python
def num_outliers(data, criteria='gaussian', n=3):
    if criteria == "gaussian":
        # number of outliers below/above n*std from the mean
        low = len(data[data < (data.mean()-(n*data.std()))])
        high = len(data[data > (data.mean()+(n*data.std()))])
        total = low+high
        return low, high, total

    elif criteria == 'whisker':
        M_FACTOR = 1.5
        QUART1 = 0.25
        QUART3 = 0.75
        # number of outliers below/above whiskers;(median - 1.5*IQR) and (median + 1.5*IQR)
        low = len(data[data < data.quantile(QUART1)-(M_FACTOR*(data.quantile(QUART3) - data.quantile(QUART1)))])
        high = len(data[data > data.quantile(QUART3)+(M_FACTOR*(data.quantile(QUART3)- data.quantile(QUART1)))])
        total = low+high
        return low,hight,total
```

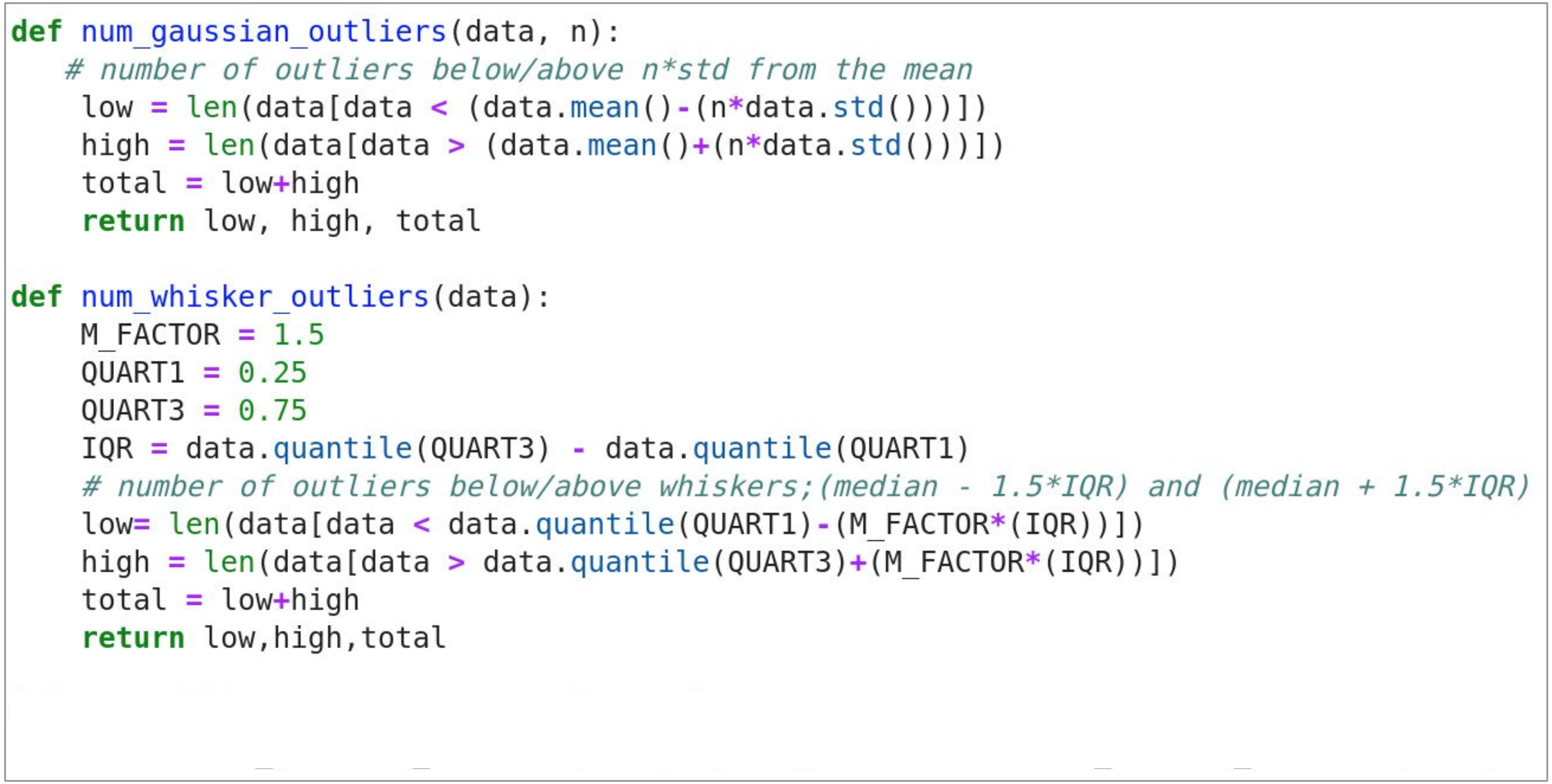
Analytics Vidhya

Modular Code



```python
def num_gaussian_outliers(data, n):
    # number of outliers below/above n*std from the mean
    low = len(data[data < (data.mean()-(n*data.std()))])
    high = len(data[data > (data.mean()+(n*data.std()))])
    total = low+high
    return low, high, total
```

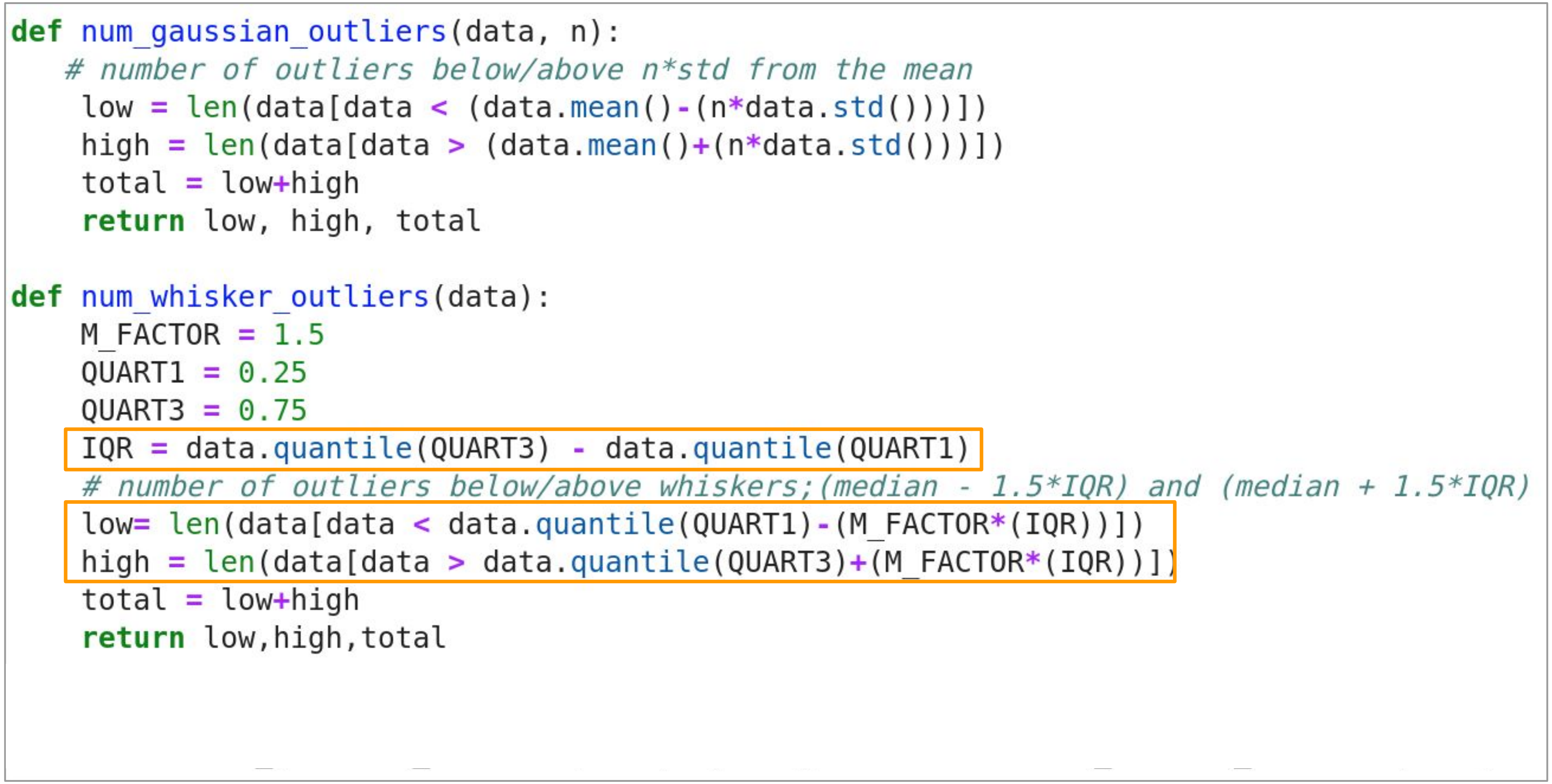
Modular Code



```python
def num_gaussian_outliers(data, n):
   # number of outliers below/above n*std from the mean
    low = len(data[data < (data.mean()-(n*data.std()))])
    high = len(data[data > (data.mean()+(n*data.std()))])
    total = low+high
    return low, high, total

def num_whisker_outliers(data):
    M_FACTOR = 1.5
    QUART1 = 0.25
    QUART3 = 0.75
    IQR = data.quantile(QUART3) - data.quantile(QUART1)
    # number of outliers below/above whiskers;(median - 1.5*IQR) and (median + 1.5*IQR)
    low= len(data[data < data.quantile(QUART1)-(M_FACTOR*(IQR))])
    high = len(data[data > data.quantile(QUART3)+(M_FACTOR*(IQR))])
    total = low+high
    return low,high,total
```

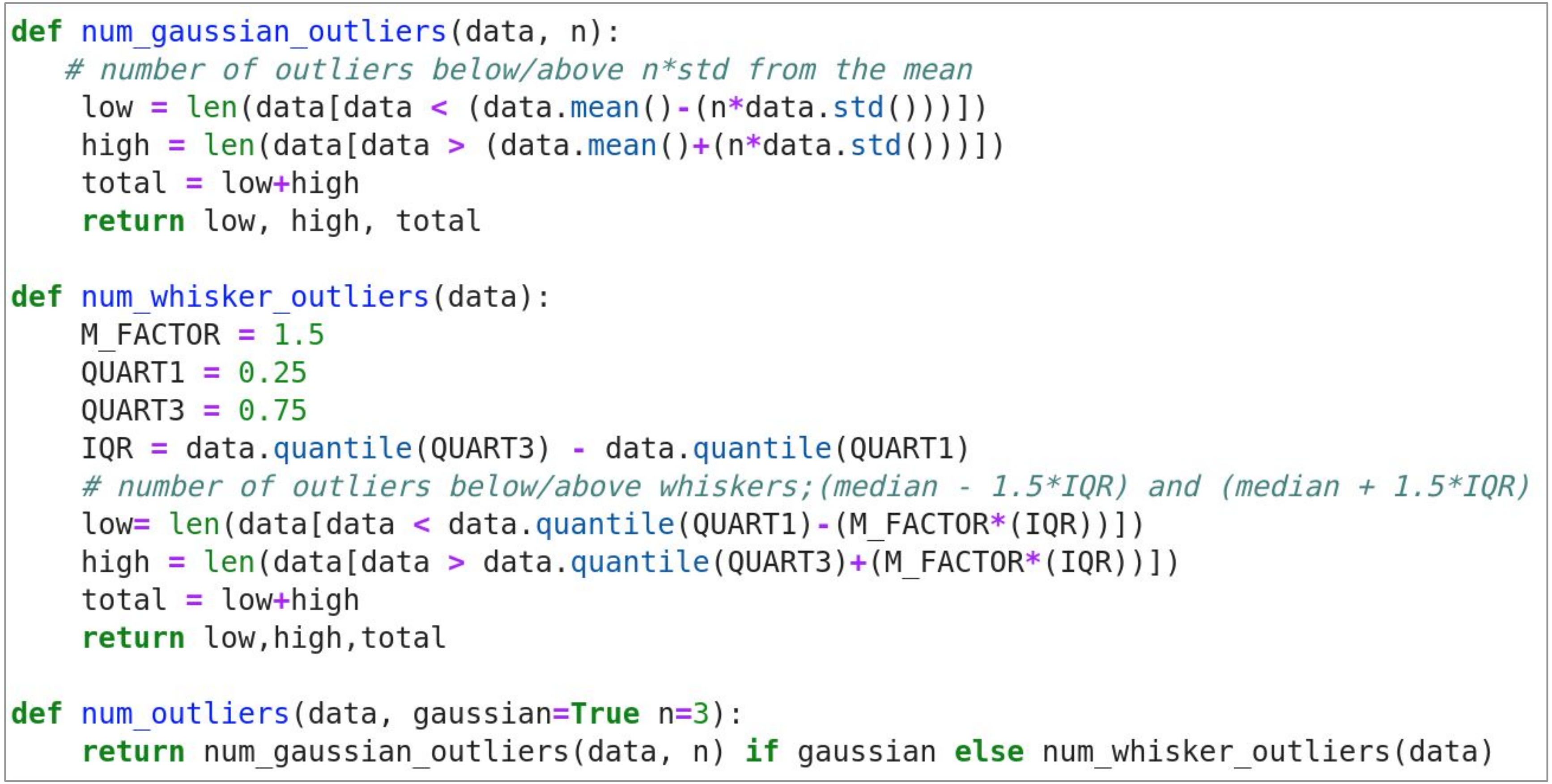
Modular Code



```
def num_gaussian_outliers(data, n):
   # number of outliers below/above n*std from the mean
    low = len(data[data < (data.mean()-(n*data.std()))])
    high = len(data[data > (data.mean()+(n*data.std()))])
    total = low+high
    return low, high, total

def num_whisker_outliers(data):
    M_FACTOR = 1.5
    QUART1 = 0.25
    QUART3 = 0.75
    IQR = data.quantile(QUART3) - data.quantile(QUART1)
    # number of outliers below/above whiskers;(median - 1.5*IQR) and (median + 1.5*IQR)
    low= len(data[data < data.quantile(QUART1)-(M_FACTOR*(IQR))])
    high = len(data[data > data.quantile(QUART3)+(M_FACTOR*(IQR))])
    total = low+high
    return low,high,total
```

Modular Code

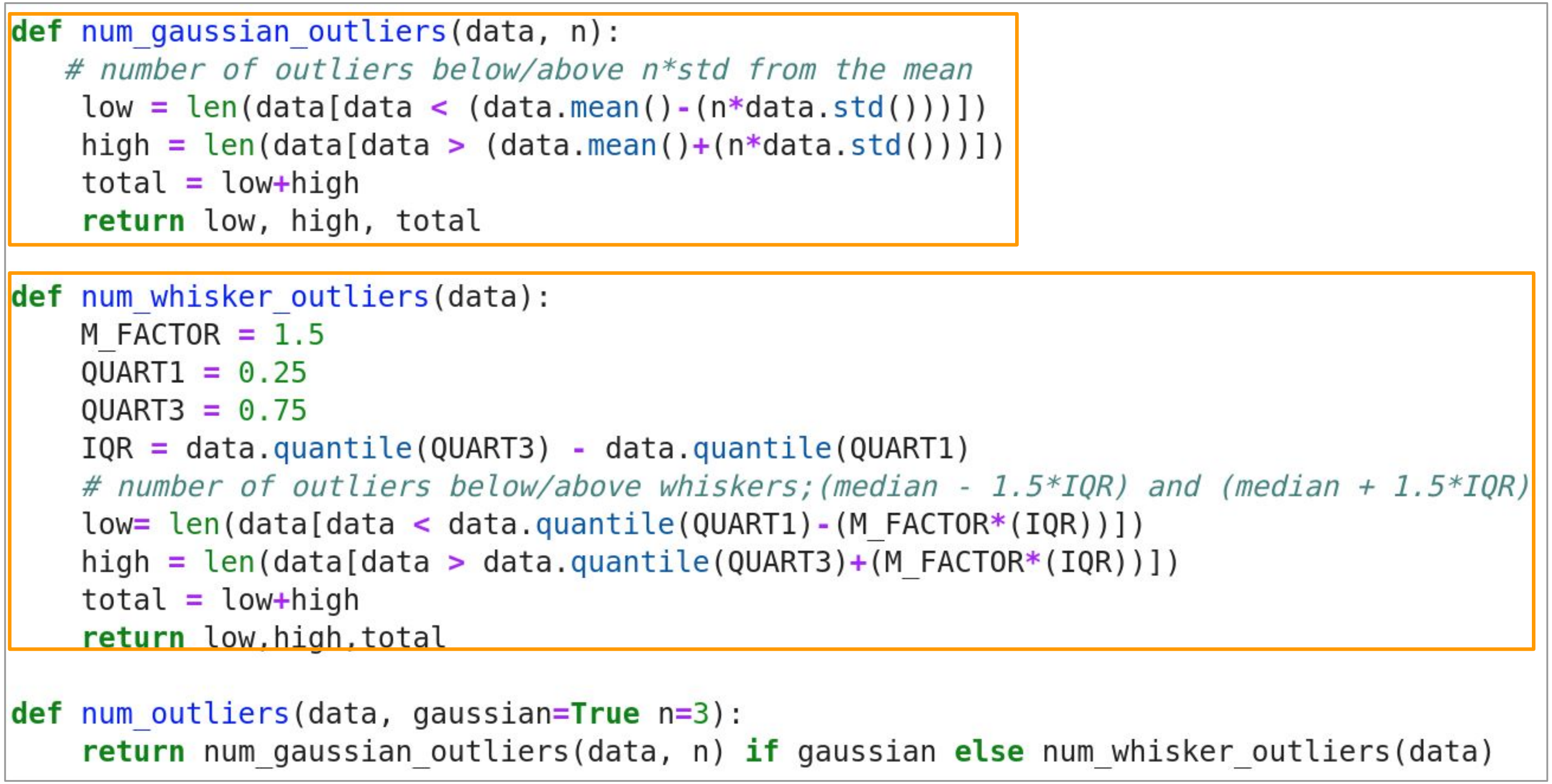


```python
def num_gaussian_outliers(data, n):
   # number of outliers below/above n*std from the mean
    low = len(data[data < (data.mean()-(n*data.std()))])
    high = len(data[data > (data.mean()+(n*data.std()))])
    total = low+high
    return low, high, total

def num_whisker_outliers(data):
    M_FACTOR = 1.5
    QUART1 = 0.25
    QUART3 = 0.75
    IQR = data.quantile(QUART3) - data.quantile(QUART1)
    # number of outliers below/above whiskers;(median - 1.5*IQR) and (median + 1.5*IQR)
    low= len(data[data < data.quantile(QUART1)-(M_FACTOR*(IQR))])
    high = len(data[data > data.quantile(QUART3)+(M_FACTOR*(IQR))])
    total = low+high
    return low,high,total

def num_outliers(data, gaussian=True n=3):
    return num_gaussian_outliers(data, n) if gaussian else num_whisker_outliers(data)
```

Modular Code



```
def num_gaussian_outliers(data, n):
   # number of outliers below/above n*std from the mean
    low = len(data[data < (data.mean()-(n*data.std()))])
    high = len(data[data > (data.mean()+(n*data.std()))])
    total = low+high
    return low, high, total

def num_whisker_outliers(data):
    M_FACTOR = 1.5
    QUART1 = 0.25
    QUART3 = 0.75
    IQR = data.quantile(QUART3) - data.quantile(QUART1)
    # number of outliers below/above whiskers;(median - 1.5*IQR) and (median + 1.5*IQR)
    low= len(data[data < data.quantile(QUART1)-(M_FACTOR*(IQR))])
    high = len(data[data > data.quantile(QUART3)+(M_FACTOR*(IQR))])
    total = low+high
    return low,high,total

def num_outliers(data, gaussian=True n=3):
    return num_gaussian_outliers(data, n) if gaussian else num_whisker_outliers(data)
```

Thank You