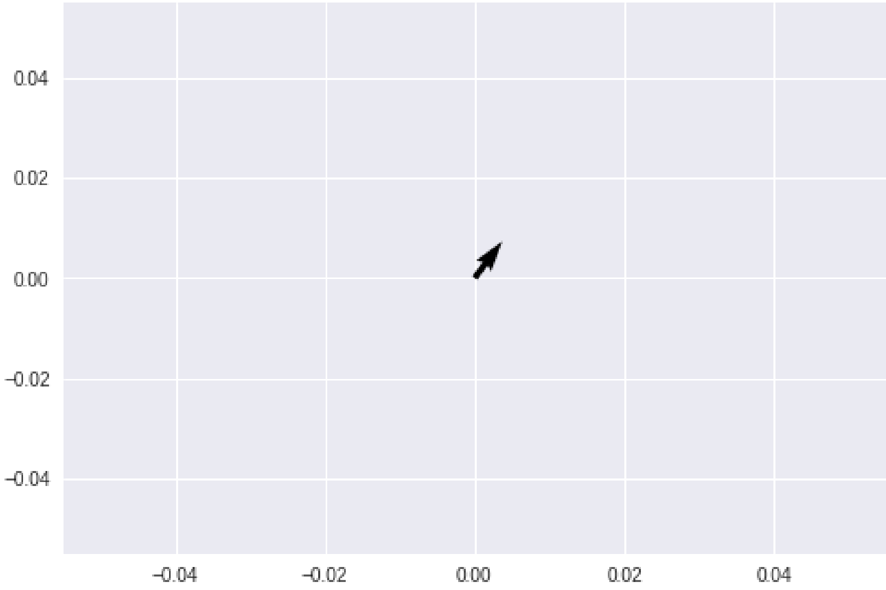
In [0]:

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

Vector plotting

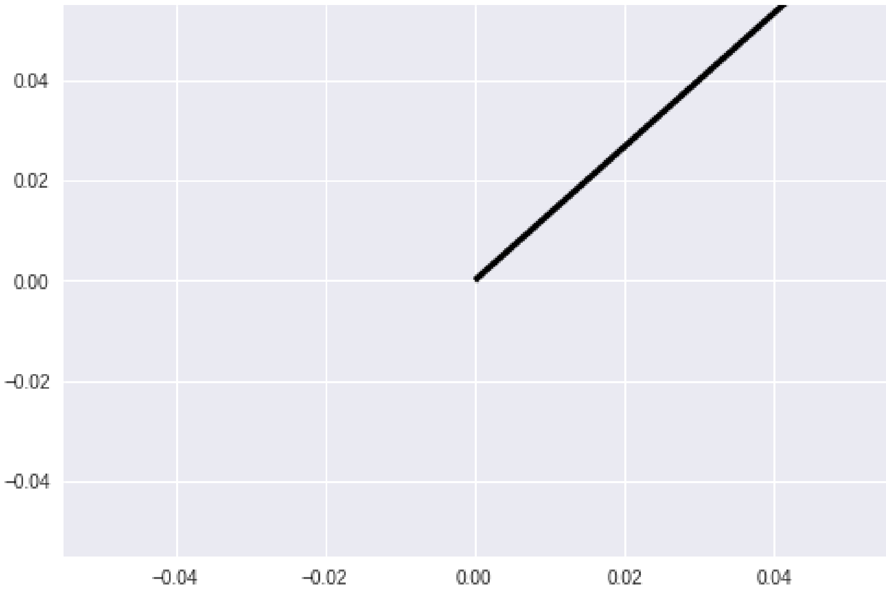
In [2]:

```
plt.quiver(0,0,3,4)
plt.show()
```



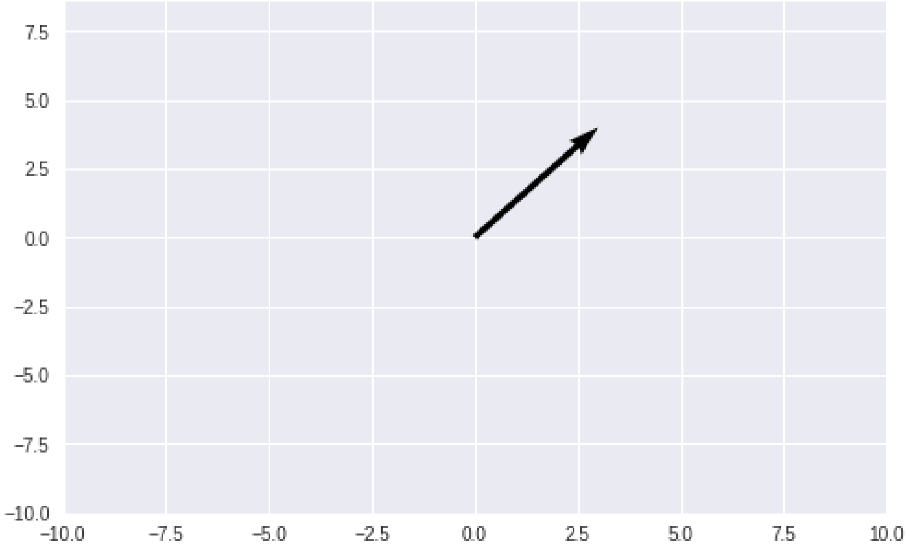
In [3]:

```
plt.quiver(0,0,3,4, scale_units='xy', angles='xy', scale=1)
plt.show()
```



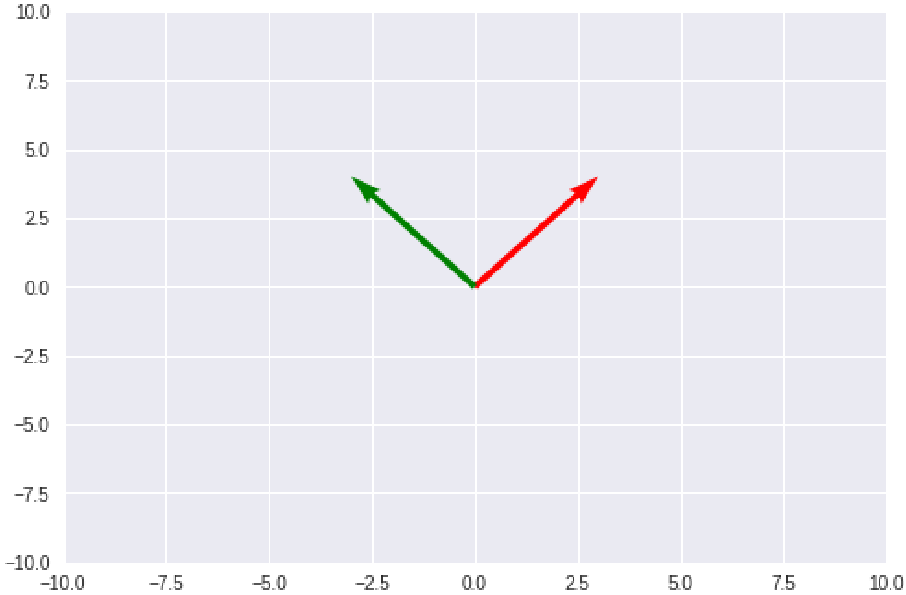
In [4]:

```
plt.quiver(0,0,3,4, scale_units='xy', angles='xy', scale=1)
plt.xlim(-10,10)
plt.ylim(-10,10)
plt.show()
```



In [6]:

```
plt.quiver(0,0,3,4, scale_units='xy', angles='xy', scale=1, color='r')
plt.quiver(0,0,-3,4, scale_units='xy', angles='xy', scale=1, color='g')
plt.xlim(-10,10)
plt.ylim(-10,10)
plt.show()
```

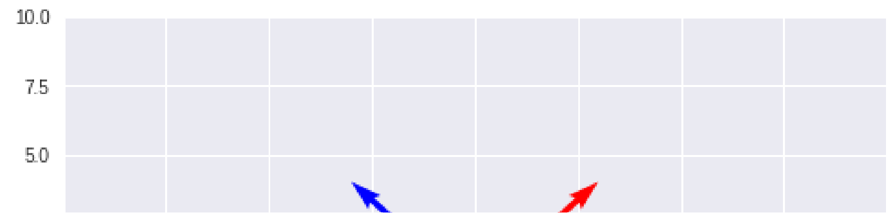


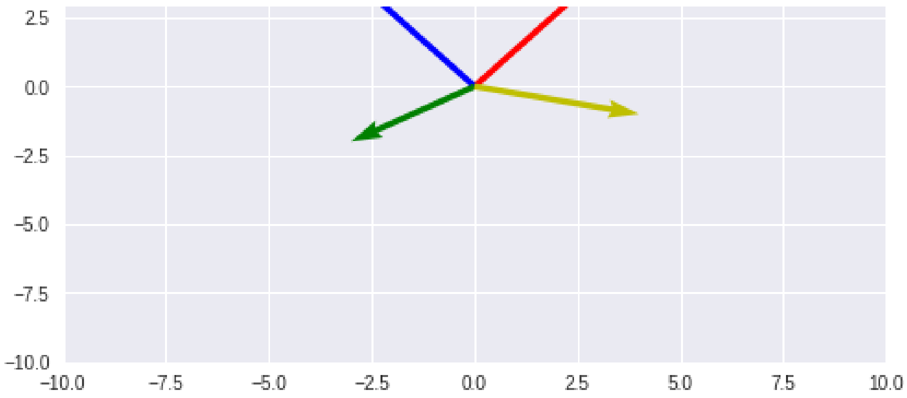
In [0]:

```
def plot_vectors(vecs):
  colors = ['r', 'b', 'g', 'y']
  i = 0
  for vec in vecs:
    plt.quiver(vec[0], vec[1], vec[2], vec[3], scale_units='xy', angles='xy', scale=1, c
olor=colors[i%len(colors)])
    i += 1
  plt.xlim(-10,10)
  plt.ylim(-10,10)
  plt.show()
```

In [13]:

```
plot_vectors([(0,0,3,4), (0,0,-3,4), (0,0,-3,-2), (0,0,4,-1)])
```





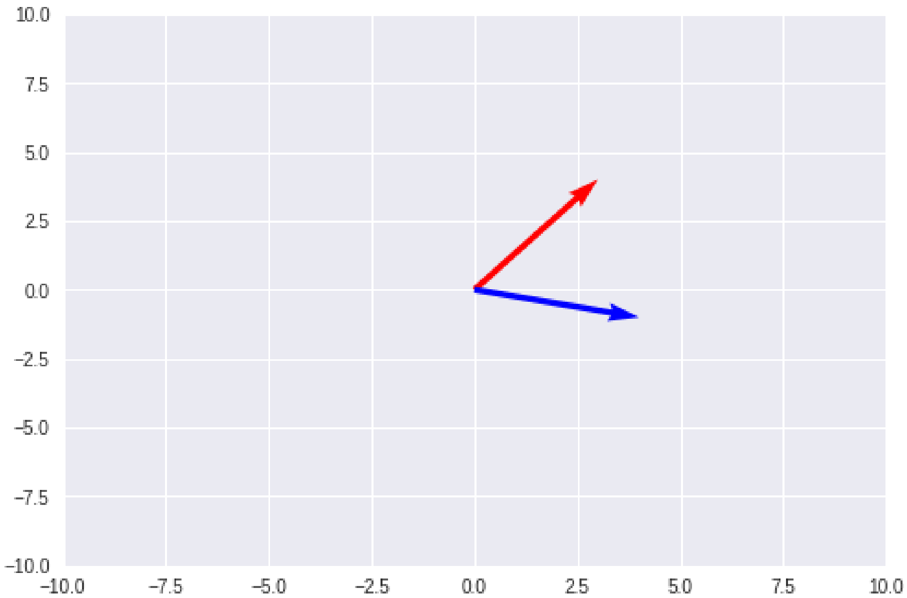
Vector addition and subtraction

In [0]:

```
vecs = [np.asarray([0,0,3,4]), np.asarray([0,0,-3,4]), np.asarray([0,0,-3,-2]), np.asarr
ay([0,0,4,-1])]
```

In [29]:

```
plot_vectors([vecs[0], vecs[3]])
```



In [31]:

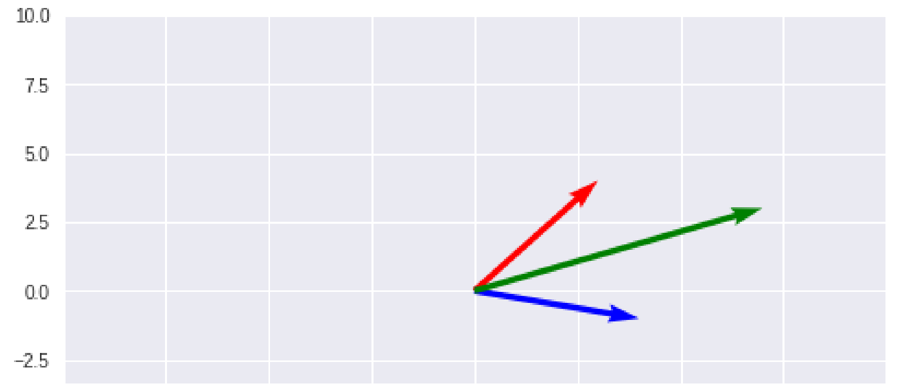
```
vecs[0] + vecs[3]
```

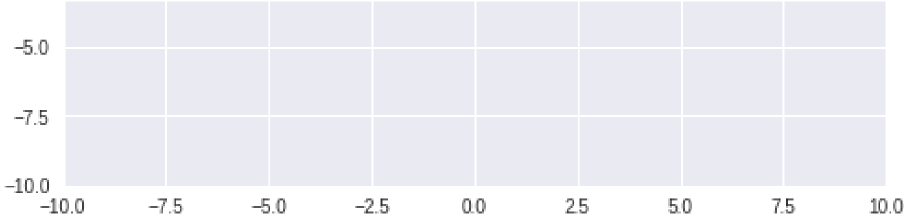
Out[31]:

```
array([0, 0, 7, 3])
```

In [33]:

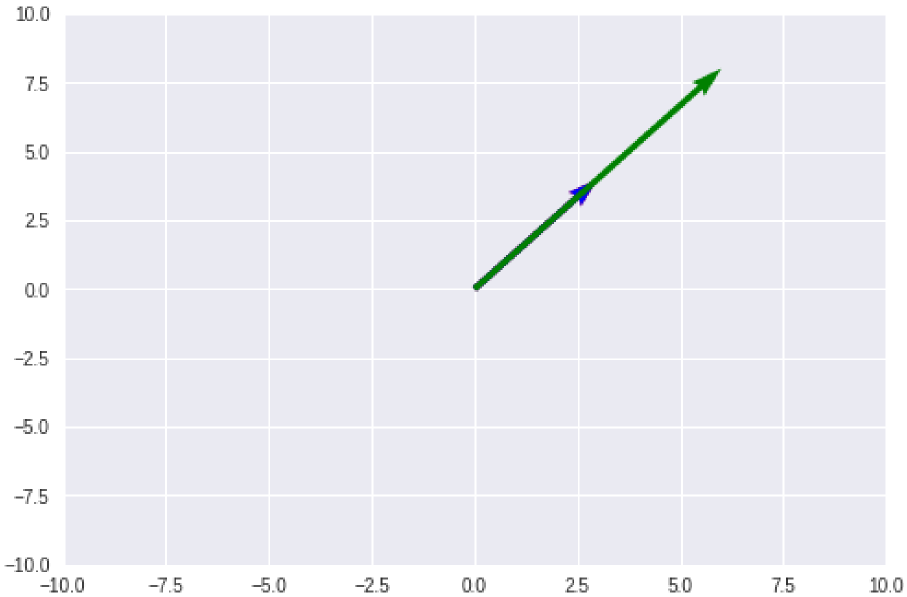
```
plot_vectors([vecs[0], vecs[3], vecs[0] + vecs[3]])
```





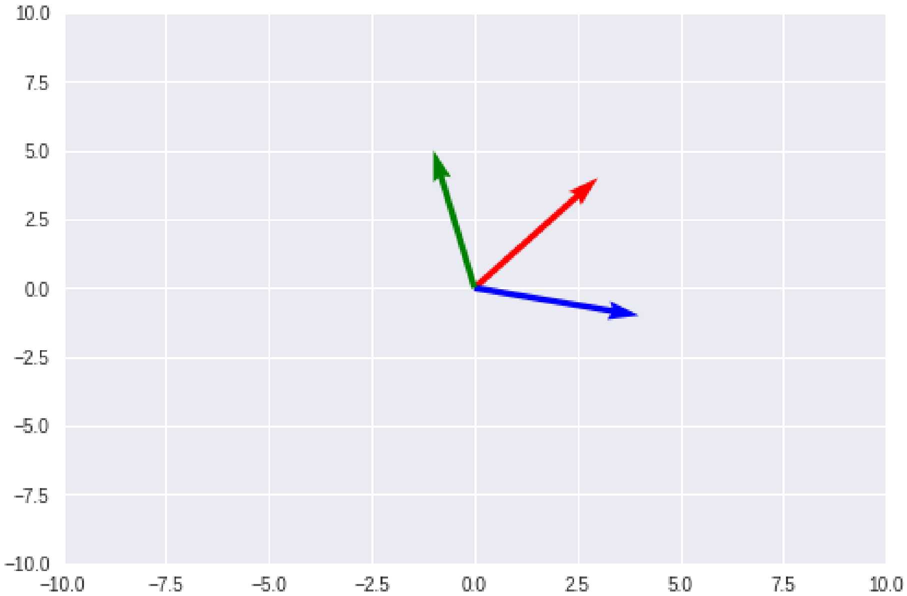
In [36]:

```
plot_vectors([vecs[0], vecs[0], vecs[0] + vecs[0]])
```



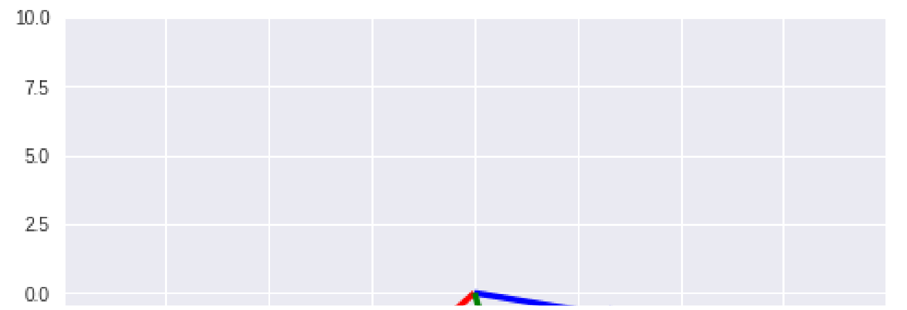
In [37]:

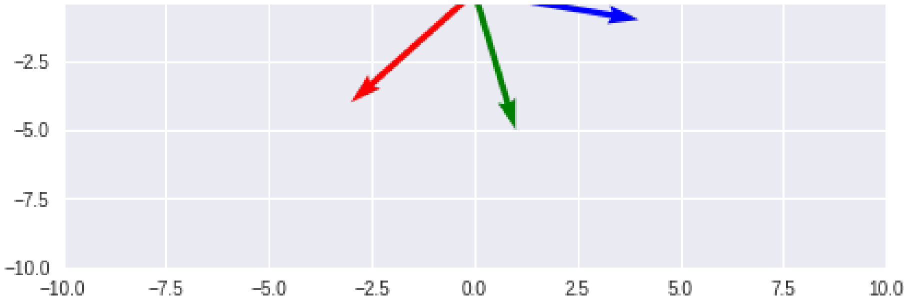
```
plot_vectors([vecs[0], vecs[3], vecs[0] - vecs[3]])
```



In [39]:

```
plot_vectors([-vecs[0], vecs[3], - vecs[0] + (vecs[3])])
```





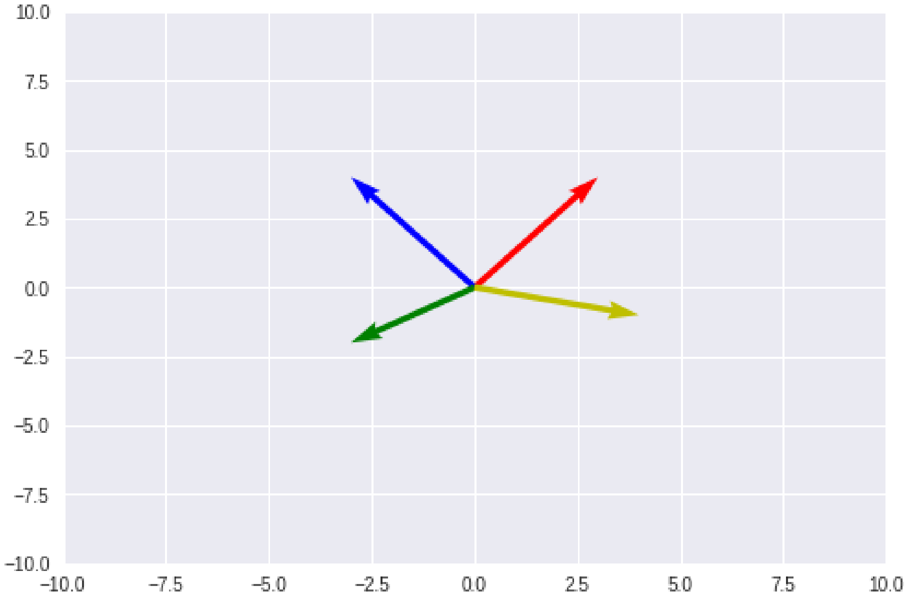
Vector dot product

In [0]:

```
vecs = [np.asarray([0,0,5,4]), np.asarray([0,0,-3,4]), np.asarray([0,0,-3,-2]), np.asarr
ay([0,0,4,-1])]
```

In [4]:

```
plot_vectors(vecs)
```



In [0]:

```
a = np.asarray([5, 4])
b = np.asarray([-3, -2])
```

$$\vec{a}\cdot\vec{b} = |\vec{a}||\vec{b}|\cos(\theta) = a_x b_x + a_y b_y$$

In [0]:

```
a_dot_b = np.dot(a, b)
```

In [18]:

```
print(a_dot_b)
```

```
-23
```

$$a_b = |\vec{a}|\cos(\theta) = |\vec{a}|\frac{\vec{a}\cdot\vec{b}}{|\vec{a}||\vec{b}|} = \frac{\vec{a}\cdot\vec{b}}{|\vec{b}|}$$

In [0]:

In [0]:

```
a_b = np.dot(a, b)/np.linalg.norm(b)
```

In [9]:

```
print(a_b)
```

```
1.9402850002906638
```

$$\overrightarrow{a_b} = a_b\hat{b}$$
$$= a_b\frac{\vec{b}}{|\vec{b}|}$$

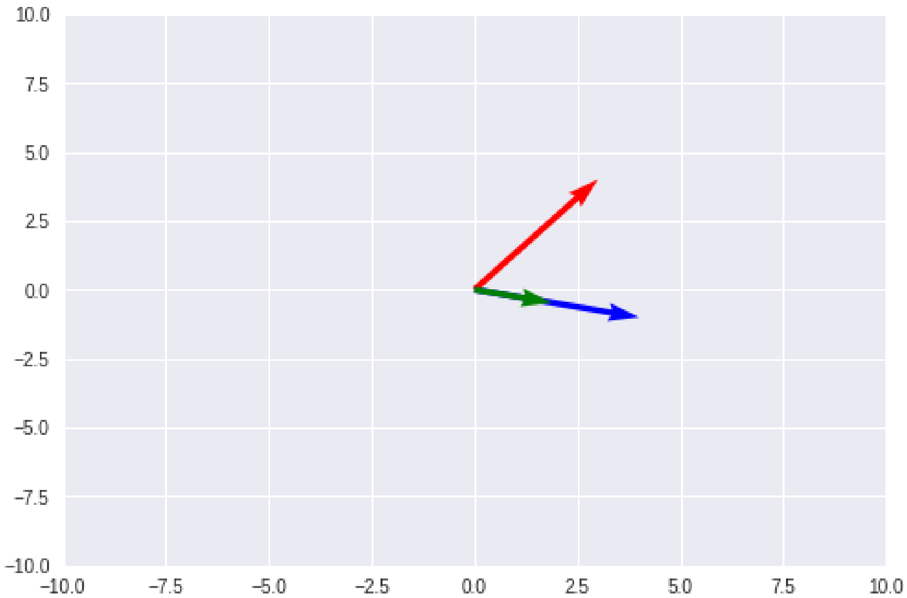
In [10]:

```
vec_a_b = (a_b/np.linalg.norm(b))*b
print(vec_a_b)
```

```
[ 1.88235294 -0.47058824]
```

In [11]:

```
plot_vectors([np.asarray([0,0,3,4]), np.asarray([0,0,4,-1]), np.asarray([0, 0, 1.8823529
4, -0.47058824])])
```



Linear combination

$$\vec{c} = w_1\vec{a}$$
$$+ w_2\vec{b}$$

In [0]:

```
def plot_linear_combination(a, b, w1, w2):

  plt.quiver(0,0,a[0],a[1], scale_units='xy', angles='xy', scale=1, color='r')
  plt.quiver(0,0,b[0],b[1], scale_units='xy', angles='xy', scale=1, color='b')

  c = w1 * a + w2 * b

  plt.quiver(0,0,c[0],c[1], scale_units='xy', angles='xy', scale=1, color='g')

  plt.xlim(-10,10)
  plt.ylim(-10,10)
  plt.show()
```

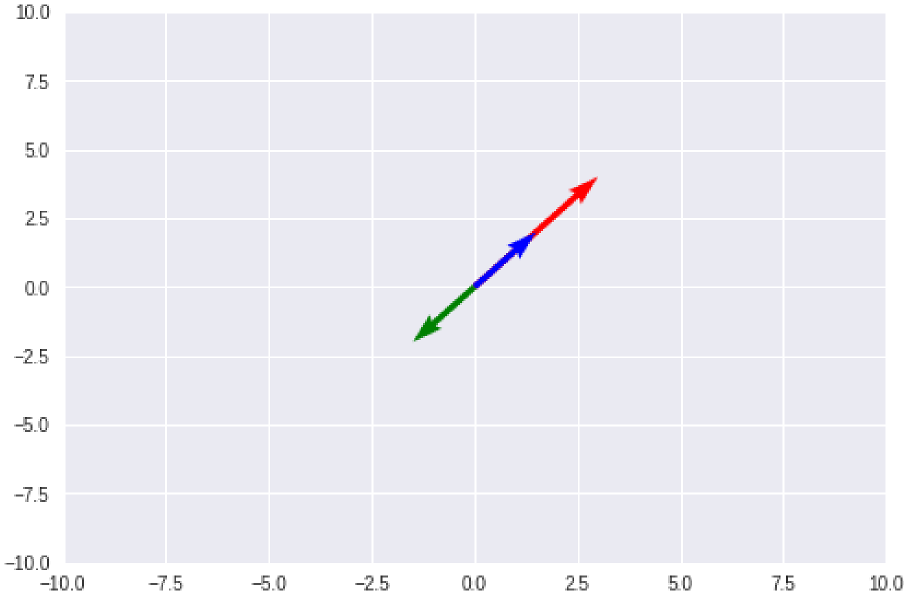
In [0]:

In [0]:

```
a = np.asarray([3, 4])
b = np.asarray([1.5, 2])
```

In [38]:

```
plot_linear_combination(a, b, -1, 1)
```

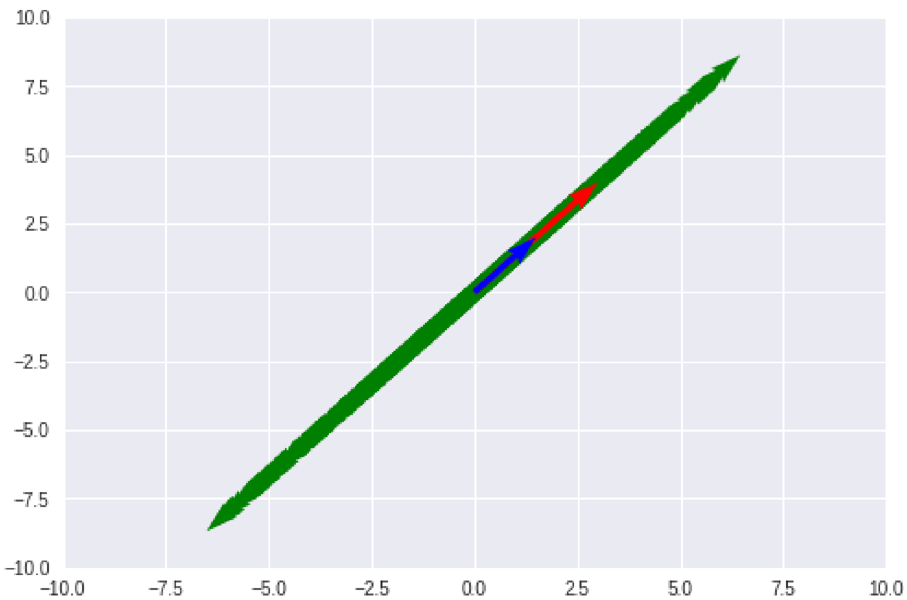


In [0]:

```
def plot_span(a, b):

  for i in range(1000):
    w1 = (np.random.random(1) - 0.5) * 3
    w2 = (np.random.random(1) - 0.5) * 3
    c = w1 * a + w2 * b
    plt.quiver(0,0,c[0],c[1], scale_units='xy', angles='xy', scale=1, color='g')

  plt.quiver(0,0,a[0],a[1], scale_units='xy', angles='xy', scale=1, color='r')
  plt.quiver(0,0,b[0],b[1], scale_units='xy', angles='xy', scale=1, color='b')

  plt.xlim(-10,10)
  plt.ylim(-10,10)
  plt.show()
```

In [40]:

```
plot_span(a, b)
```



In [0]:

In [ ]:

```

```