201700949 설재혁

In [124]:

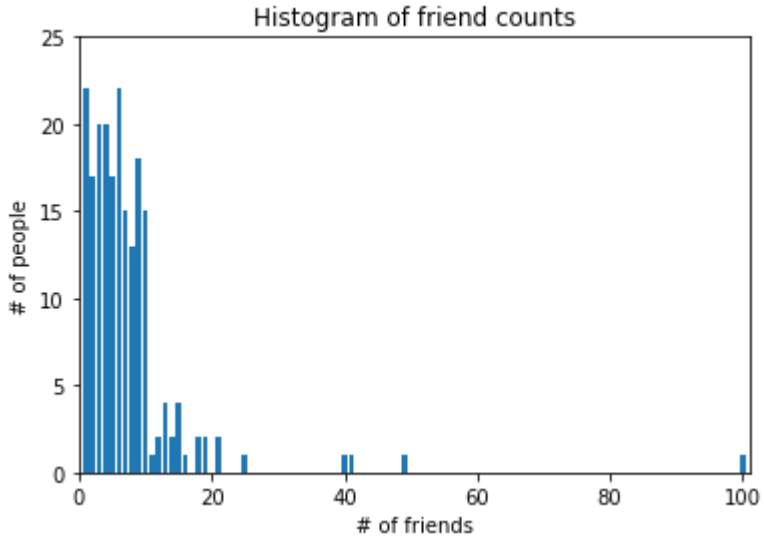
```
def sum_of_squares(v):  # linear_algebra.py 코드에서 임포트 하는 대신 여기에 코딩
    """v_1 * v_1 + … + v_n * v_n"""
    return dot(v, v)

def dot(v, w):           # linear_algebra.py 코드에서 임포트 하는 대신 여기에 코딩
    """v_1 * w_1 + … + v_n * w_n"""
    return sum(v_i * w_i for v_i, w_i in zip(v, w))

from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

from collections import Counter
#from linear_algebra import sum_of_squares, dot # linear_algebra.py 코드에서 임포트
import math
import numpy as np
import matplotlib as plt
%pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib

/Users/seoljaehyeok/opt/anaconda3/lib/python3.8/site-packages/IPytho
n/core/magics/pylab.py:159: UserWarning: pylab import has clobbered
these variables: ['plt', 'mean', 'f', 'dot', 'quantile', 'median']
`%matplotlib` prevents importing * from pylab and numpy
  warn("pylab import has clobbered these variables: %s"  % clobbered
+
```

친구 수 히스토그램

In [9]:

```
num_friends = [100,49,41,40,25,21,21,19,19,18,18,16,15,15,15,15,14,14,13,13,13,1
3,12,12,11,10,10,10,10,10,10,10,10,10,10,10,10,10,10,9,9,9,9,9,9,9,9,9,9,9,
9,9,9,9,9,9,8,8,8,8,8,8,8,8,8,8,8,8,8,7,7,7,7,7,7,7,7,7,7,7,7,7,7,6,6,6,6,6,6,
6,6,6,6,6,6,6,6,6,6,6,6,6,6,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,4,4,4,4,4,4,4,
4,4,4,4,4,4,4,4,4,4,4,4,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,2,2,2,2,2,2,2,
2,2,2,2,2,2,2,2,2,2,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1]

def make_friend_count_histogram(plt):
    friend_counts = Counter(num_friends)
    xs = range(101)
    ys = [friend_counts[x] for x in xs]
    plt.bar(xs,ys)
    plt.axis([0,101,0,25])
    plt.title('Histogram of friend counts')
    plt.xlabel('# of friends')
    plt.ylabel('# of people')
    plt.show()

make_friend_count_histogram(plt)
```



기본 통계치

In [11]:

```
num_points = len(num_friends)                    # 204
largest_value = max(num_friends)                 # 100
smallest_value = min(num_friends)                # 1
sorted_values = sorted(num_friends)
smallest_value = sorted_values[0]                  # 1
second_smallest_value = sorted_values[1]    # 1
second_largest_value = sorted_values[-2]    # 49

print(num_points)
print(largest_value)
print(smallest_value)
print(sorted_values)
print(smallest_value)
print(second_smallest_value)
print(second_largest_value)
```

```
204
100
1
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6,
6, 6, 6, 6, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 8, 8, 8, 8,
8, 8, 8, 8, 8, 8, 8, 8, 8, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9,
9, 9, 9, 9, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
10, 11, 12, 12, 13, 13, 13, 13, 14, 14, 15, 15, 15, 15, 16, 18, 18,
19, 19, 21, 21, 25, 40, 41, 49, 100]
1
1
49
```

중심 경향성 : 평균(average)

In [13]:

```
def mean(x):
    return sum(x)/len(x)

mean(num_friends)

# Numpy version
np.mean(num_friends)
```

Out[13]:

```
7.333333333333333
```

Out[13]:

```
7.333333333333333
```

중심 경향성 : 중앙값(median)

In [16]:

```
def median(x):
    """finds the 'middle-most' value of v"""
    n = len(x)
    sorted_x = sorted(x)
    midpoint = n // 2

    if n % 2 == 1:
        # if odd, return the middle value
        return sorted_x[midpoint]
    else:
        # if even, return the average of the middle values
        lo = midpoint - 1
        hi = midpoint
        return (sorted_x[lo] + sorted_x[hi]) / 2

median(num_friends)

# Numpy version
np.median(num_friends)
```

Out[16]:

```
6.0
```

Out[16]:

```
6.0
```

분위(Quantile)

- 중앙값을 일반화한 개념으로 특정 백분위보다 낮은 분위에 속하는 데이터를 의미합니다
- 중앙값은 상위 50%의 데이터보다 작은 값을 의미합니다

In [17]:

```
def quantile(x, p):
    """returns the pth-percentile value in x"""
    p_index = int(p * len(x))
    return sorted(x)[p_index]

for i in range(0, 100, 25):
    print("%.2f Percentage value" % (i*0.01) , quantile(num_friends, i * 0.01))

# Numpy version
np.percentile(num_friends, [i for i in range(0,100,25)])
```

```
0.00 Percentage value 1
0.25 Percentage value 3
0.50 Percentage value 6
0.75 Percentage value 9
```

Out[17]:

```
array([1., 3., 6., 9.])
```

최빈값(mode)

- 데이터에서 가장 많이 나오는 값

In [21]:

```
def mode(x):
    """returns a list, might be more than one mode"""
    counts = Counter(x)
    max_count = max(counts.values())
    return [x_i for x_i, count in counts.items()
            if count == max_count]

mode(num_friends)
```

Out[21]:

```
[6, 1]
```

산포도 : 범위

- 데이터가 얼마나 퍼져있는지를 나타내는 지표
- 범위는 가장 큰 값과 가장 작은 값의 차이

In [22]:

```
# "range" already means something in Python, so we'll use a different name
def data_range(x):
    return max(x) - min(x)

data_range(num_friends)

np.max(num_friends) - np.min(num_friends)
```

Out[22]:

```
99
```

Out[22]:

```
99
```

분산(Variance)

- 데이터들이 기대값(평균)으로부터 얼마나 떨어진 곳에 분포하는지를 가늠하는 숫자
- 분산은 편차의 제곱의 평균을 계산하므로 단위는 기존 단위의 제곱이 됩니다

In [23]:

```
# Mean - value

def de_mean(x):
    """translate x by subtracting its mean (so the result has mean 0)"""
    x_bar = mean(x)
    return [x_i - x_bar for x_i in x]

def variance(x):
    """assumes x has at least two elements"""
    n = len(x)
    deviations = de_mean(x)
    return sum_of_squares(deviations) / (n - 1)

variance(num_friends)

%timeit variance(num_friends)
%timeit np.var(num_friends) # 일반적인 분산 연산도 numpy가 빠름
```

Out[23]:

```
81.54351395730706

116 µs ± 3.35 µs per loop (mean ± std. dev. of 7 runs, 10000 loops ea
ch)
33.3 µs ± 1.21 µs per loop (mean ± std. dev. of 7 runs, 10000 loops e
ach)
```

표준편차와 사분위간 분위

- 분산 대신 원래 데이터의 단위와 같은 단위를 가지는 표준편차를 더 많이 이용
- 범위와 표준편차는 이상치(outlier)에 민감하게 반응
- 이상치에 덜 민감하게 반응하는 사분위간 분위는 상위 25%와 하위 25%에 해당하는 값의 차이를 계산

In [24]:

```
def standard_deviation(x):
    return math.sqrt(variance(x))

standard_deviation(num_friends)

np.std(num_friends, dtype=np.float64)

def interquartile_range(x):
    return quantile(x, 0.75) - quantile(x, 0.25)

interquartile_range(num_friends)
```

Out[24]:

```
9.030144736232474
```

Out[24]:

```
9.007984838446012
```

Out[24]:

```
6
```

공분산(Covariance)

- 분산(variance)은 하나의 변수가 평균에서 얼마나 멀리 떨어져 있는지 계산
- 공분산(covariance)은 두 변수가 각각의 평균에서 얼마나 멀리 떨어져 있는지 계산
- 공분산이 양수이면 x 값이 커질수록 y 값도 커짐을 의미하고, 공분산이 음수이면 x값이 커질수록 y 값은 작아짐을 의미

In [25]:

```
daily_minutes = [1,68.77,51.25,52.08,38.36,44.54,57.13,51.4,41.42,31.22,34.76,5
4.01,38.79,47.59,49.1,27.66,41.03,36.73,48.65,28.12,46.62,35.57,32.98,35,26.07,2
3.77,39.73,40.57,31.65,31.21,36.32,20.45,21.93,26.02,27.34,23.49,46.94,30.5,33.8
,24.23,21.4,27.94,32.24,40.57,25.07,19.42,22.39,18.42,46.96,23.72,26.41,26.97,3
6.76,40.32,35.02,29.47,30.2,31,38.11,38.18,36.31,21.03,30.86,36.07,28.66,29.08,3
7.28,15.28,24.17,22.31,30.17,25.53,19.85,35.37,44.6,17.23,13.47,26.33,35.02,32.0
9,24.81,19.33,28.77,24.26,31.98,25.73,24.86,16.28,34.51,15.23,39.72,40.8,26.06,3
5.76,34.76,16.13,44.04,18.03,19.65,32.62,35.59,39.43,14.18,35.24,40.13,41.82,35.
45,36.07,43.67,24.61,20.9,21.9,18.79,27.61,27.21,26.61,29.77,20.59,27.53,13.82,3
3.2,25,33.1,36.65,18.63,14.87,22.2,36.81,25.53,24.62,26.25,18.21,28.08,19.42,29.
79,32.8,35.99,28.32,27.79,35.88,29.06,36.28,14.1,36.63,37.49,26.9,18.58,38.48,2
4.48,18.95,33.55,14.24,29.04,32.51,25.63,22.22,19,32.73,15.16,13.9,27.2,32.01,2
9.27,33,13.74,20.42,27.32,18.23,35.35,28.48,9.08,24.62,20.12,35.26,19.92,31.02,1
6.49,12.16,30.7,31.22,34.65,13.13,27.51,33.2,31.57,14.1,33.42,17.44,10.12,24.42,
9.82,23.39,30.93,15.03,21.67,31.09,33.29,22.61,26.89,23.48,8.38,27.81,32.35,23.8
4]

def covariance(x, y):
    n = len(x)
    return dot(de_mean(x), de_mean(y)) / (n - 1)

covariance(num_friends, daily_minutes)

np.cov(num_friends,daily_minutes)
```

Out[25]:

```
22.42543513957307
```

Out[25]:

```
array([[ 81.54351396,  22.42543514],
       [ 22.42543514, 100.78589895]])
```

상관관계(Correlation)

- 공분산의 절대값이 크다고 상관관계가 더 크다고 판단하기 어렵다
- 상관관계(correlation)는 공분산에서 각각의 표준편차를 나눠준다
- 상관관계는 단위가 없으며 항상 -1 ~ 1 사이의 값을 가진다

In [26]:

```
def correlation(x, y):
    stdev_x = standard_deviation(x)
    stdev_y = standard_deviation(y)
    if stdev_x > 0 and stdev_y > 0:
        return covariance(x, y) / stdev_x / stdev_y
    else:
        return 0 # if no variation, correlation is zero

correlation(num_friends, daily_minutes)

np.corrcoef(num_friends, daily_minutes)

plt.plot(num_friends, daily_minutes, 'ro')
plt.axis([0,max(num_friends)+10,0,max(daily_minutes) +10 ])
plt.show()
```

Out[26]:

```
0.2473695736647823
```

Out[26]:

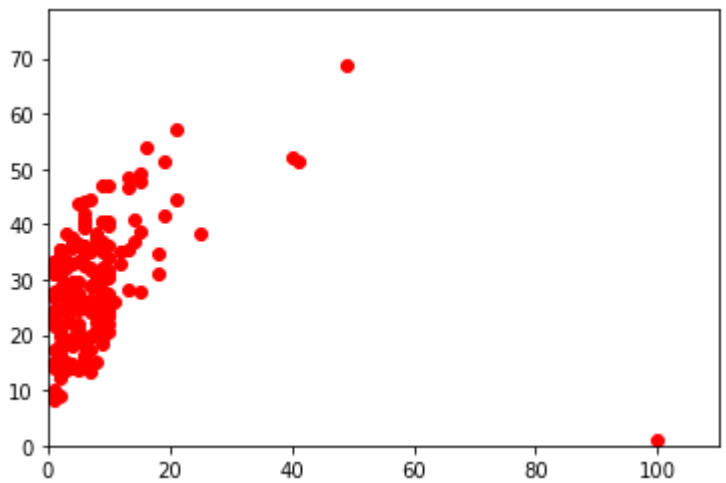
```
array([[1.        , 0.24736957],
       [0.24736957, 1.        ]])
```

Out[26]:

```
[<matplotlib.lines.Line2D at 0x7fdc853c1370>]
```

Out[26]:

```
(0.0, 110.0, 0.0, 78.77)
```



이상치를 제거하면 더 강력한 상관관계를 볼 수 있다

- 예를 들어, 100명의 친구가 있지만 하루에 1분만 사이트를 이용하는 사용자는 이상치다.

In [27]:

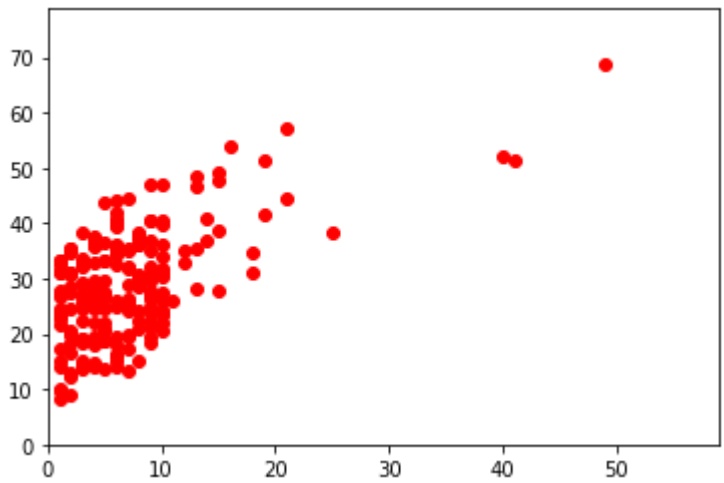
```
outlier = num_friends.index(100) # index of outlier

num_friends_good = [x
                    for i, x in enumerate(num_friends)
                    if i != outlier]

daily_minutes_good = [x
                      for i, x in enumerate(daily_minutes)
                      if i != outlier]

plt.plot(num_friends_good, daily_minutes_good, 'ro')
plt.axis([0,max(num_friends_good)+10,0,max(daily_minutes_good) +10 ])
plt.show()
```

Out[27]:

```
[<matplotlib.lines.Line2D at 0x7fdc84244340>]
```

Out[27]:

```
(0.0, 59.0, 0.0, 78.77)
```



Lab6 (1)

In [218]:

```
import pandas

df = pandas.read_csv('height-weight.csv')
maleIndex = df[df['Gender'] == 'Male'].index # 남자인 경우 index == 0~4999
femaleIndex = df[df['Gender'] == 'Female'].index # 여자인 경우 index == 5000~9999

df.loc[maleIndex, "color"] = 'b' # 남자인 경우로 슬라이싱 하고 파랑색으로 지정
df.loc[femaleIndex, "color"] = 'r' # 여자인 경우로 슬라이싱 하고 빨간색으로 지정

# scatter_plot = df.plot.scatter(x='Height', y='Weight', s=1, c=df['color'], lab
el=df['Height'])
# scatter_plot.plot()
plt.scatter(df['Height'], df['Weight'], s=1, c=df['color'])
# plt.legend(loc=2)
plt.title('Height-Weight Plotting by Seol Jae hyeok')
plt.xlabel('Height in Inches')
plt.ylabel('Weight in Lbs')
plt.show()
```

Out[218]:

```
<matplotlib.collections.PathCollection at 0x7fdc0b2abdc0>
```

Out[218]:

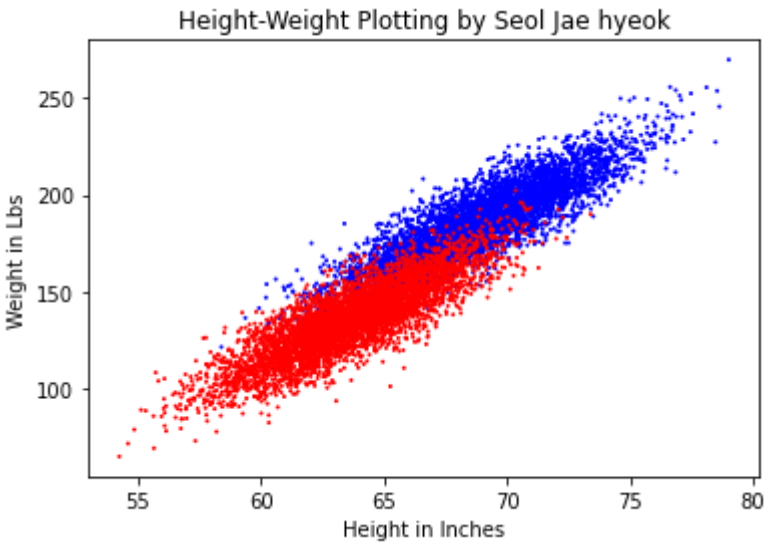
```
Text(0.5, 1.0, 'Height-Weight Plotting by Seol Jae hyeok')
```

Out[218]:

```
Text(0.5, 0, 'Height in Inches')
```

Out[218]:

```
Text(0, 0.5, 'Weight in Lbs')
```



(2) 평균 mean

In [244]:

```
# 남자, 여자 데이터 구분하여 저장
is_Male = df['Gender'] == 'Male'
is_Female = df['Gender'] == 'Female'
male = df[is_Male]
female = df[is_Female]

# 평균값
print('남자 키 평균 값: ',np.mean(male['Height'])) # 남자 키 평균
print('여자 키 평균 값: ',np.mean(female['Height'])) # 여자 키 평균
print('남자 몸무게 평균 값: ',np.mean(male['Weight'])) # 남자 몸무게 평균
print('여자 몸무게 평균 값: ',np.mean(female['Weight'])) # 여자 몸무게 평균
```

```
남자 키 평균 값:  69.02634590621737
여자 키 평균 값:  63.708773603424916
남자 몸무게 평균 값:  187.0206206581929
여자 몸무게 평균 값:  135.8600930074687
```

(3) 중앙값 median

In [245]:

```
print('남자 키 중앙 값: ',np.median(male['Height'])) # 남자 키 중앙값
print('여자 키 중앙 값: ',np.median(female['Height'])) # 여자 키 중앙값
print('남자 몸무게 중앙 값: ',np.median(male['Weight'])) # 남자 몸무게 중앙값
print('여자 몸무게 중앙 값: ',np.median(female['Weight'])) # 여자 몸무게 중앙값
```

```
남자 키 중앙 값:  69.02770850939555
여자 키 중앙 값:  63.7309238591475
남자 몸무게 중앙 값:  187.033546088862
여자 몸무게 중앙 값:  136.11758297008498
```

(4) 분위 Quantile

In [247]:

```
print('남자 키 분위 값: ',np.percentile(male['Height'],[i for i in range(0,100,25
)])) # 남자 키 분위
print('여자 키 분위 값: ',np.percentile(female['Height'],[i for i in range(0,100,25
)])) # 여자 키 분위
print('남자 몸무게 분위 값: ',np.percentile(male['Weight'],[i for i in range(0,100,25
)])) # 남자 몸무게 분위
print('여자 몸무게 분위 값: ',np.percentile(female['Weight'],[i for i in range(0,100,
25)])) # 여자 몸무게 분위
```

```
남자 키 분위 값:  [58.40690493 67.17467907 69.02770851 70.98874363]
여자 키 분위 값:  [54.26313333 61.89444149 63.73092386 65.56356518]
남자 몸무게 분위 값:  [112.90293945 173.88776733 187.03354609 200.3578018
]
여자 몸무게 분위 값:  [ 64.70012671 122.93409617 136.11758297 148.8109262
6]
```

(5) 최빈값 mode

In [290]:

```
parsedMaleHeight = []
parsedMaleWeight = []
parsedFemaleHeight = []
parsedFemaleWeight = []

for MH,MW,FH,FW in zip(male["Height"], male['Weight'], female['Height'], female[
'Weight']):
    parsedMaleHeight.append(int(MH))
    parsedMaleWeight.append(int(MW))
    parsedFemaleHeight.append(int(FH))
    parsedFemaleWeight.append(int(FW))

print('남자 키 최빈값: ',mode(parsedMaleHeight)) # 남자 키 최빈값
print('여자 키 최빈값: ',mode(parsedFemaleHeight)) # 여자 키 최빈값
print('남자 몸무게 최빈값: ',mode(parsedMaleWeight)) # 남자 몸무게 최빈값
print('여자 몸무게 최빈값: ',mode(parsedFemaleWeight)) # 여자 몸무게 최빈값
```

```
남자 키 최빈값:  [69]
여자 키 최빈값:  [63]
남자 몸무게 최빈값:  [192]
여자 몸무게 최빈값:  [137]
```

(6) 산포도 range

In [291]:

```
print('남자 키 산포도: ',np.max(male['Height'])-np.min(male['Height'])) # 남자 키 산포
도
print('여자 키 산포도: ',np.max(female['Height'])-np.min(female['Height'])) # 여자 키
산포도
print('남자 몸무게 산포도: ',np.max(male['Weight'])-np.min(male['Weight'])) # 남자 몸무
게 산포도
print('여자 몸무게 산포도: ',np.max(female['Weight'])-np.min(female['Weight'])) # 여자
몸무게 산포도
```

```
남자 키 산포도:  20.59183741463979
여자 키 산포도:  19.126452540972608
남자 몸무게 산포도:  157.086759057288
여자 몸무게 산포도:  137.53708702680598
```

(7) 분산 Variance

In [292]:

```
print('남자 키 분산: ',np.var(male['Height'])) # 남자 키 분산
print('여자 키 분산: ',np.var(female['Height'])) # 여자 키 분산
print('남자 몸무게 분산: ',np.var(male['Weight'])) # 남자 몸무게 분산
print('여자 몸무게 분산: ',np.var(female['Weight'])) # 여자 몸무게 분산
```

```
남자 키 분산:  8.19720348386999
여자 키 분산:  7.268493504171401
남자 몸무게 분산:  391.21581520128217
여자 몸무게 분산:  361.7819105481172
```

(8) 표준편차

In [293]:

```
print('남자 키 표준편차: ',np.std(male['Height'])) # 남자 키 표준편차
print('여자 키 표준편차: ',np.std(female['Height'])) # 여자 키 표준편차
print('남자 몸무게 표준편차: ',np.std(male['Weight'])) # 남자 몸무게 표준편차
print('여자 몸무게 표준편차: ',np.std(female['Weight'])) # 여자 몸무게 표준편차
```

```
남자 키 표준편차:  2.8630758781195427
여자 키 표준편차:  2.6960143738807107
남자 몸무게 표준편차:  19.779176302396472
여자 몸무게 표준편차:  19.020565463416624
```

(9) 공분산 Covariance

In [294]:

```
print('남자 여자 키 공분산: ',np.cov(male['Height'], female['Height'])) # 남자 키 표준편차
print('남자 여자 몸무게 공분산: ',np.cov(male['Weight'], female['Weight'])) # 남자 몸무게 표준편차
```

```
남자 여자 키 공분산:  [[ 8.19884325 -0.25129107]
 [-0.25129107  7.26994749]]
남자 여자 몸무게 공분산:  [[391.29407402  -7.37151287]
 [ -7.37151287 361.8542814 ]]
```

(10) 상관관계 Correlation

In [295]:

```
print('남자의 키와 몸무게이 상관관계: ',np.corrcoef(male['Height'], male['Weight'])) #
 남자의 키와 몸무게의 상관관계
print('여자의 키와 몸무게의 상관관계: ',np.corrcoef(female['Height'], female['Weight']))
# 여자의 키와 몸무게의 상관관계

plt.scatter(male['Height'], male['Weight'], s=1, color='blue')
plt.axis([0,max(male['Height'])+10,0,max(male['Weight']) +10 ])
plt.scatter(female['Height'], female['Weight'], s=1, color='red')
plt.axis([0,max(female['Height'])+10,0,max(female['Weight']) +10 ])
plt.show()
```

```
남자의 키와 몸무게이 상관관계:  [[1.         0.86297885]
 [0.86297885 1.        ]]
여자의 키와 몸무게의 상관관계:  [[1.         0.84960859]
 [0.84960859 1.        ]]
```

Out[295]:

```
<matplotlib.collections.PathCollection at 0x7fdc14762820>
```

Out[295]:

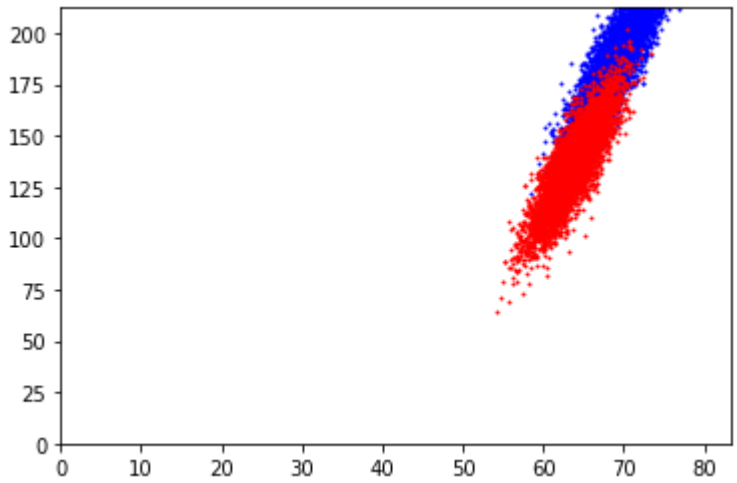
```
(0.0, 88.99874234638959, 0.0, 279.989698505106)
```

Out[295]:

```
<matplotlib.collections.PathCollection at 0x7fdc14762ac0>
```

Out[295]:

```
(0.0, 83.38958586606971, 0.0, 212.237213739559)
```



- 산점도가 우상향하는 모양이기 때문에 남자 여자 모두 키가 클수록 몸무게가 많이 나가는 것을 확인할 수 있다.

201700949 설재혁

In [ ]: