In [26]:
```
#matplotlibに日本語フォント適用
!pip install japanize-matplotlib
import japanize_matplotlib
```

```
Requirement already satisfied: japanize-matplotlib in /usr/local/lib/python3.7/dist-packages (1.1.3)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (from japanize-matplotlib) (3.2.2)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->japa
nize-matplotlib) (2.4.7)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib->japanize-matplotlib) (0.10.0)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->japanize-matplotlib) (2.
8.1)
Requirement already satisfied: numpy>=1.11 in /usr/local/lib/python3.7/dist-packages (from matplotlib->japanize-matplotlib) (1.19.5)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->japanize-matplotlib) (1.3.
1)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from cycler>=0.10->matplotlib->japanize-matplotlib) (1.15.
0)
```

Pythonによるデータ解析入門p70 ~ p89より抜粋

# 第３章

統計的な手法を使った多変量の分析

～相関分析・回帰分析・主成分分析・因子分析～

## 3.1相関分析と回帰分析

### 3.1.1 相関分析

２つの変数の間に関係性（相関）があるか、またどのような関係性があるのかを分析しようというのが相関分析です。

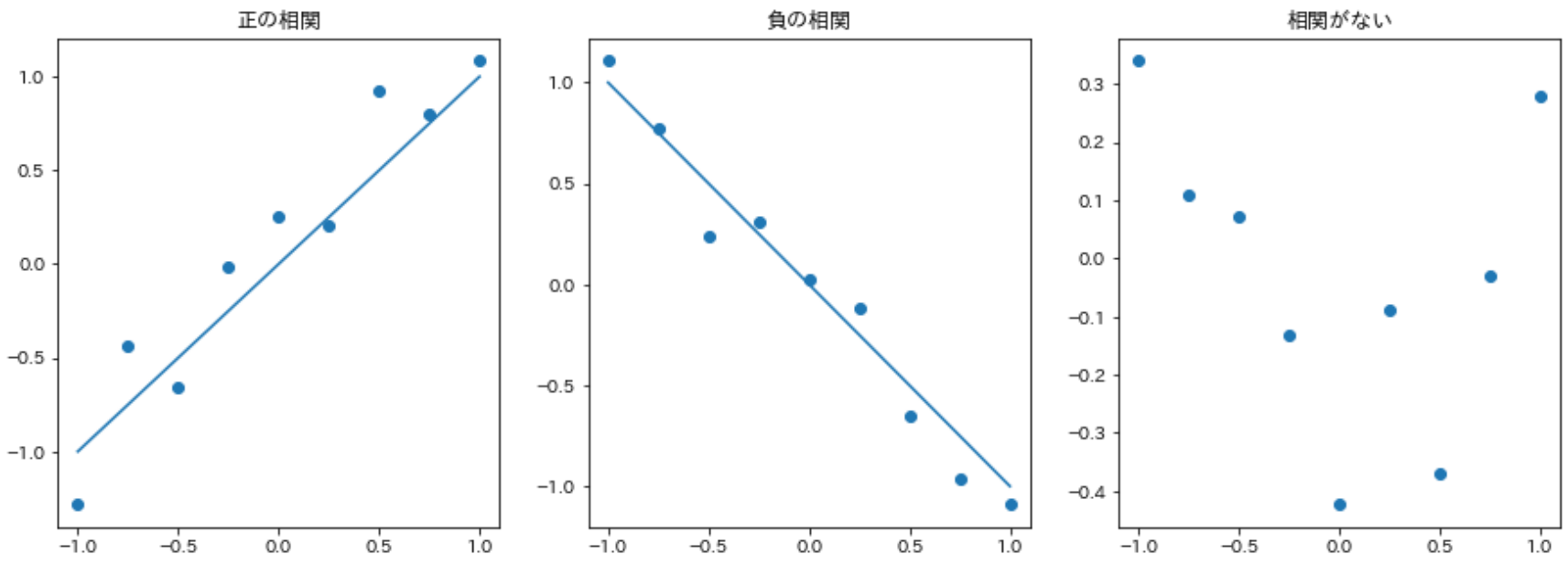
相関関係は２つの変数をx軸、y軸にとってそれぞれのデータ点をプロットした**散布図**を描くと、視覚的にも感じ取ることができます。たとえば、xの値が大きければyの値も大きく、xの値が小さければyの値も小さいという関係が成り立つときは、グラフ上の点は、左下から右上に連なる、右上がりの線上に固まってきます。このような右上がりの関係を**正の相関**と呼びます。逆に、xの値が増えるとyの値が減るという関係にあるときは右下がりの線上グラフ上の点が集まってきますが、このような右下がりの関係を**負の相関**と呼びます。また、xの値とyの値が無関係であれば、グラフ全体に点がばらまかれた状況となり、このような分布の仕方を**相関がない**（無相関）と呼びます。

相関の図 <!--

# 後でpicのみ張っておく

import random x = np.linspace(-1,1,9)

y1 = [i + (random.random()-0.5) for i in x] y2 = [-1 *i + (random.random()-0.5) for i in x] y3 = [(random.random()-0.5) for i in x] fig, ax = plt.subplots(1, 3, figsize = (15,5)) ax[0].scatter(x,y1) ax[0].plot(x,x) ax[0].set_title("正の相関") ax[1].scatter(x,y2) ax[1].plot(x,-1*x) ax[1].set_title("負の相関") ax[2].scatter(x,y3) ax[2].set_title("相関がない") print("相関係数は",np.corrcoef(x, y)[0, 1].round(4)) -->




2016年の東京における日最高気温の月平均（気象庁） http://www.data.jma.go.jp/obd/stats/etrn/view/monthly_s3.php?prec_no=44&block_no=47662&view=a2

2016年の１世帯当たりアイスクリーム支出金額（一般社団法人日本アイスクリーム協会） https://www.icecream.or.jp/biz/data/expenditures.html

相関係数

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

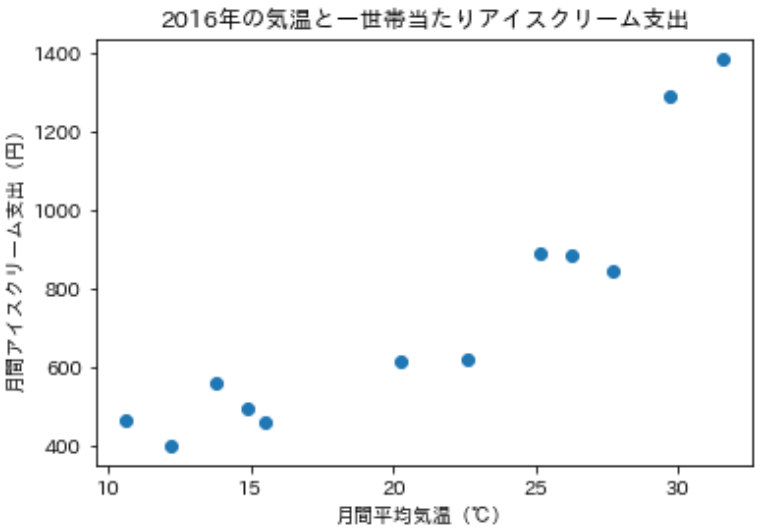
$$-1 \geqq r \geqq 1$$

- rが 1 に近ければ正の相関
- 0のときは相関がない状態
- -1に近ければ負の相関

In [27]:

```
# リスト3-1　アイスクリームの支出と気温の相関係数
# -*- coding: utf-8 -*-
import numpy as np
import matplotlib.pyplot as plt
# 2016年　一世帯当たりアイスクリーム支出金額（一般社団法人日本アイスクリーム協会）
#   https://www.icecream.or.jp/data/expenditures.html
icecream = [[1, 464], [2, 397], [3, 493], [4, 617], [5, 890], [6, 883],
    [7, 1292], [8, 1387], [9, 843], [10, 621], [11, 459], [12, 561]]
# 2016年　月別平均気温（気象庁）
#   http://www.data.jma.go.jp/obd/stats/etrn/view/monthly_s3.php?
#                                  prec_no=44&block_no=47662&view=a2
temperature = [[1, 10.6], [2, 12.2], [3, 14.9], [4, 20.3], [5, 25.2], [6, 26.3],
    [7, 29.7], [8, 31.6], [9, 27.7], [10, 22.6], [11, 15.5], [12, 13.8]]
x = np.array([u[1] for u in temperature])
y = np.array([u[1] for u in icecream])
print('correlation coefficient', np.corrcoef(x, y)[0, 1].round(4))
# グラフを描く
plt.scatter(x, y)
plt.title('2016年の気温と一世帯当たりアイスクリーム支出')
plt.xlabel('月間平均気温（℃）')
plt.ylabel('月間アイスクリーム支出（円）')
plt.show()
```

```
correlation coefficient 0.9105
```




Pythonで相関係数を計算する方法

- 定義の式を式のとおりPythonで計算する
- Numpyの `corrcoeff()` 関数で計算する
- 3.1.2節で詳しく見る回帰分析のためのパッケージを使って、相関係数も出す

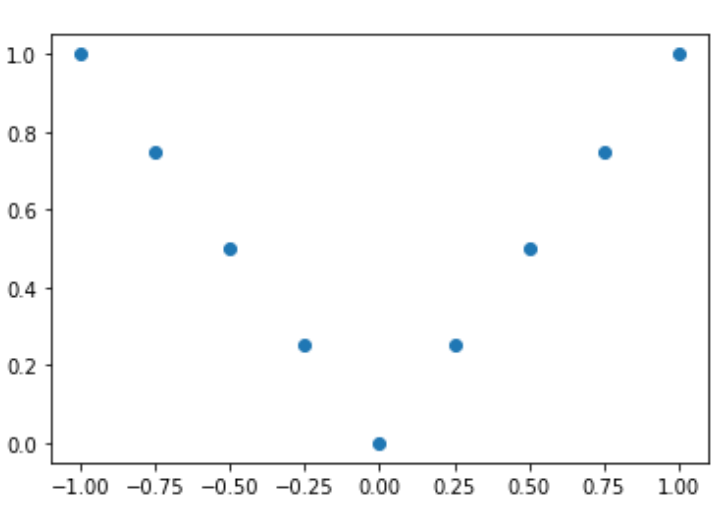
相関係数は便利ですが、直線ではないような強い関連性がある場合に、相関係数が小さい値になってしまうことがあります。

V字型分布の図 <!--

# 後でpicのみ張っておく

x = np.linspace(-1,1,9) y = [-1*i if i<0 else i for i in np.linspace(-1,1,9)] fig = plt.figure() plt.scatter(x,y) print("相関係数は",np.corrcoef(x, y)[0, 1].round(4)) fig.savefig("img.png")

-->




また、相関係数は$x, y$が量的データであることが前提です。質的データ（カテゴリデータ）について値の間隔に意味がない尺度では、厳密には相関関係が意味をなしません。（それでも、順序尺度で同じ係数を計算することはあります。）

## 3.1.2 回帰分析

入力と出力の関係性を式にする分析。通常よく使われるのは１次関数。

$$f(x) = bx + a$$

データとの誤差の２乗和が最小になるような引き方をする。

$$\text{データとの誤差} = y_i - (bx_i + a)$$

$$L = \sum_{i=1}^{n}(y_i - (bx_i + a))^2$$

aとbを偏微分する。

$$\begin{cases} \frac{\delta L}{\delta a} = \sum(-2)・(y_i - (bx_i + a)) \\ \frac{\delta L}{\delta b} = \sum(-2x_i)・(y_i - (bx_i + a)) \end{cases}$$

最小値を求めるために0とおく。

$$\begin{cases} na + (\sum_{i=1}^{n} x_i)b = \sum_{i=1}^{n} y_i \\ (\sum_{i=1}^{n} x_i)a + (\sum_{i=1}^{n} {x_i}^2)b = \sum_{i=1}^{n} x_i y_i \end{cases}$$

上の式より、

$$a = \frac{\sum_{i=1}^{n} y_i}{n} - b\frac{(\sum_{i=1}^{n} x_i)}{n}$$

$$a = \bar{y} - b\bar{x}$$

下の式に代入して、

$$(\sum_{i=1}^{n} x_i)(\bar{y} - b\bar{x}) + (\sum_{i=1}^{n} {x_i}^2)b = \sum_{i=1}^{n} x_i y_i$$

$$(\sum_{i=1}^{n} x_i)\bar{y} - (\sum_{i=1}^{n} x_i)b\bar{x} + (\sum_{i=1}^{n} {x_i}^2)b = \sum_{i=1}^{n} x_i y_i$$

$$n\frac{(\sum_{i=1}^{n} x_i)}{n}\bar{y} - n\frac{(\sum_{i=1}^{n} x_i)}{n}\bar{x}b + (\sum_{i=1}^{n} {x_i}^2)b = \sum_{i=1}^{n} x_i y_i$$

$$n\bar{x}\bar{y} - n\bar{x}^2 b + (\sum_{i=1}^{n} {x_i}^2)b = \sum_{i=1}^{n} x_i y_i$$

$$(\sum_{i=1}^{n} {x_i}^2 - n\bar{x}^2)b = \sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y}$$

$$b = \frac{\sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y}}{\sum_{i=1}^{n} {x_i}^2 - n\bar{x}^2}$$

平均気温を**説明変数(入力x)**、アイスクリームの支出を**目的変数(出力$f(x)$)**としたとき、

$a = -107.1$、$b = 40.70$であるため、

$$y = 40.70x + 107.1$$

これを**回帰方程式（回帰直線）**と呼ぶ。

**決定係数**

回帰直線の当てはまりの良さの尺度。

１に近いと当てはまりが良く、０に近いと当てはまりが良くない。

$R^2 = 1 - \frac{\text{回帰式による推定値の分散}}{\text{標本値の分散}}$

$f(x_i)$・・・予測値

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - f(x_i))^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

線形回帰（１次関数）の場合は$r^2 = R^2$となる。

Pythonプログラムで回帰直線を求めるパッケージ

- SciPy `stats` モジュールの `linregress()`
- scikit-learnの `linear_model` モジュールの `LinearRegression()`
- StatsModelsパッケージの `api.OLS` モジュール

## SciPyのlinregress

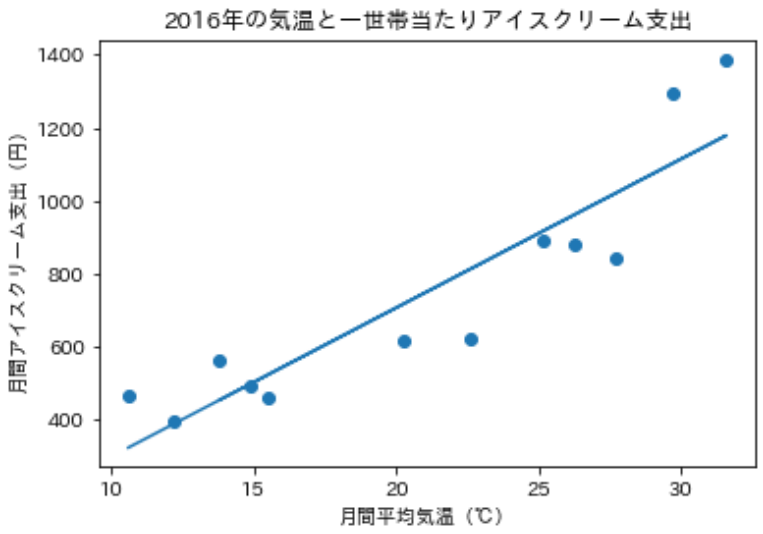
相関係数のp値まで出してくれますが、単回帰（説明変数が１つだけ）で、重回帰分析は扱えません。

In [28]:

```
# リスト3-2　アイスクリーム支出と気温から回帰方程式を求める（SciPyのlinregressを使用）
# -*- coding: utf-8 -*-
# アイスクリーム支出の回帰分析　stats.linregressを用いる
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats
# 2016年　一世帯当たりアイスクリーム支出金額（一般社団法人日本アイスクリーム協会）
#   https://www.icecream.or.jp/data/expenditures.html
icecream = [[1, 464], [2, 397], [3, 493], [4, 617], [5, 890], [6, 883], ¥
```

```
    [7, 1292], [8, 1387], [9, 843], [10, 621], [11, 459], [12, 561]]
# 2016年　月別平均気温（気象庁）
#   http://www.data.jma.go.jp/obd/stats/etrn/view/monthly_s3.php?
#                                prec_no=44&block_no=47662&view=a2
temperature = [[1, 10.6], [2, 12.2], [3, 14.9], [4, 20.3], [5, 25.2], [6, 26.3], ¥
    [7, 29.7], [8, 31.6], [9, 27.7], [10, 22.6], [11, 15.5], [12, 13.8]]
x = np.array([u[1] for u in temperature])
y = np.array([u[1] for u in icecream])
result = scipy.stats.linregress(x, y)
print('傾き=', result.slope.round(4), '切片=', result.intercept.round(4), ¥
    '信頼係数=', result.rvalue.round(4), 'p値=', result.pvalue.round(4), ¥
    '標準誤差=', result.stderr.round(4))
# グラフを描く
b = result.slope
a = result.intercept
plt.plot(x, [b * u + a for u in x])  # predict(X)はXに対応した回帰直線上のyの値を返す
plt.scatter(x, y)
plt.title('2016年の気温と一世帯当たりアイスクリーム支出')
plt.xlabel('月間平均気温（℃）')
plt.ylabel('月間アイスクリーム支出（円）')
plt.show()
```

```
傾き= 40.7016 切片= -107.0571 信頼係数= 0.9105 p値= 0.0 標準誤差= 5.8471
```




## scikit-learnのlinear_model

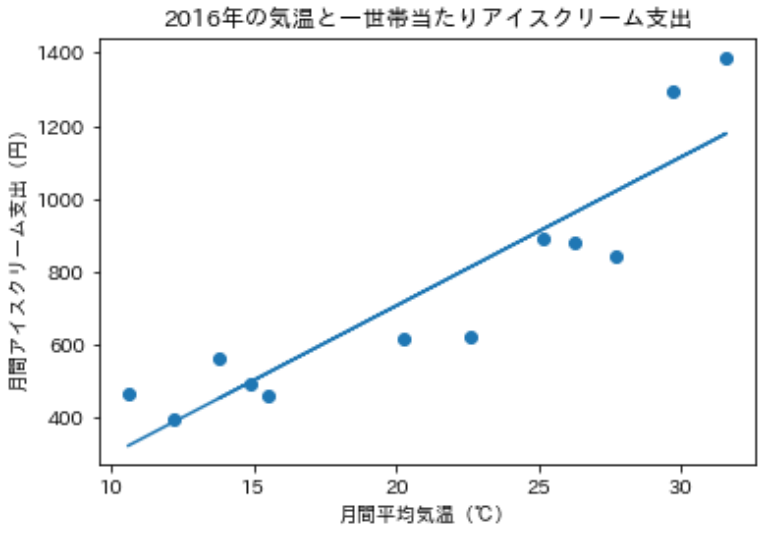
回帰直線yの値を `model.predict(x)` としてmodelに任せる形で計算しています。

In [29]:

```
# リスト3-3　アイスクリーム支出と気温から回帰方程式を求める
#   （scikit-learnのlinear_modelを使用）
# -*- coding: utf-8 -*-
# アイスクリーム支出の回帰分析　sklearnのlinear_modelを用いる
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import linear_model  # scikit-learnのlinear_modelを使って回帰分析
# 2016年　一世帯当たりアイスクリーム支出金額（一般社団法人日本アイスクリーム協会）
#   https://www.icecream.or.jp/data/expenditures.html
icecream = [[1, 464], [2, 397], [3, 493], [4, 617], [5, 890], [6, 883], ¥
    [7, 1292], [8, 1387], [9, 843], [10, 621], [11, 459], [12, 561]]
# 2016年　月別平均気温（気象庁）
#    http://www.data.jma.go.jp/obd/stats/etrn/view/monthly_s3.php?
#                                    prec_no=44&block_no=47662&view=a2
temperature = [[1, 10.6], [2, 12.2], [3, 14.9], [4, 20.3], [5, 25.2], [6, 26.3], ¥
    [7, 29.7], [8, 31.6], [9, 27.7], [10, 22.6], [11, 15.5], [12, 13.8]]
X = pd.DataFrame([u[1] for u in temperature])
Y = pd.DataFrame([u[1] for u in icecream])
model = linear_model.LinearRegression()
results = model.fit(X, Y)
print('a', model.coef_[0][0], 'b', model.intercept_[0])

# グラフを描く
plt.plot(X.values, model.predict(X))  # predict(X)はXに対応した回帰直線上のyの値を返す
plt.scatter(X, Y)
plt.title('2016年の気温と一世帯当たりアイスクリーム支出')
plt.xlabel('月間平均気温（℃）')
plt.ylabel('月間アイスクリーム支出（円）')
plt.show()
```

```
a 40.70161776241745 b -107.05709064244411
```



## StatsModelsのOLS

OLSの詳細： http://www.statsmodels.org/dev/generated/statsmodels.regression.linear_model.OLS.html

回帰分析全体の説明： http://www.statsmodels.org/dev/generated/statsmodels.regression.linear_model.RegressionResults.html

OLS固有の記述： http://www.statsmodels.org/dev/generated/statsmodels.regression.linear_model.OLSResults.html

`sm.OLS` のインスタンス `model` を作るときに、 `sm.add_constant(x)` によってxを１欄増やして定数１を入れています。 `add_constant()` について詳しくはマニュアルを参照： http://www.statsmodels.org/dev/generated/statsmodels.tools.tools.add_constant.html

In [30]:
```
# リスト3-4　アイスクリーム支出と気温から回帰方程式を求める（StatsModelsのOLSを使用）
# -*- coding: utf-8 -*-
# アイスクリーム支出の回帰分析　StatsModelsのOLSを用いる
import numpy as np
import pandas as pd
import statsmodels.api as sm  # 回帰分析はStatsModelsパッケージを利用する
icecream = [[1, 464], [2, 397], [3, 493], [4, 617], [5, 890], [6, 883], ¥
    [7, 1292], [8, 1387], [9, 843], [10, 621], [11, 459], [12, 561]]
temperature = [[1, 10.6], [2, 12.2], [3, 14.9], [4, 20.3], [5, 25.2], [6, 26.3], ¥
    [7, 29.7], [8, 31.6], [9, 27.7], [10, 22.6], [11, 15.5], [12, 13.8]]
x = np.array([u[1] for u in temperature])
y = np.array([u[1] for u in icecream])
# 切片計算のため、xに定数列を1列加えてからモデルを作る
model = sm.OLS(y, sm.add_constant(x))
results = model.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.829
Model:                            OLS   Adj. R-squared:                  0.812
Method:                 Least Squares   F-statistic:                     48.46
Date:                Mon, 10 May 2021   Prob (F-statistic):           3.89e-05
Time:                        21:27:47   Log-Likelihood:                -75.369
No. Observations:                  12   AIC:                             154.7
Df Residuals:                      10   BIC:                             155.7
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       -107.0571    128.673     -0.832      0.425    -393.759     179.645
x1            40.7016      5.847      6.961      0.000      27.674      53.730
==============================================================================
Omnibus:                        1.129   Durbin-Watson:                   1.509
Prob(Omnibus):                  0.569   Jarque-Bera (JB):                0.744
Skew:                           0.204   Prob(JB):                        0.689
Kurtosis:                       1.850   Cond. No.                         69.4
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
/usr/local/lib/python3.7/dist-packages/scipy/stats/stats.py:1535: UserWarning: kurtosistest only valid for n>=20 ... continuing anyway, n=12
  "anyway, n=%i" % int(n))
```

summaryの見方は重回帰分析で説明します。

In [31]:
```
print('p-values¥n', results.pvalues)
b, a = results.params
print('a', a.round(4), 'b', b.round(4))
```

```
p-values
 [4.24828131e-01 3.89478187e-05]
a 40.7016 b -107.0571
```

pvaluesは、（t検定に基づく）p値。説明変数として意味の無い(係数がゼロである)確率です。小さければ小さいほど有意になります。

## 3.1.3 重回帰分析

重回帰分析は、説明変数が多次元の場合の回帰分析です。説明変数$x$がベクトルになっているところが違うだけで、後は単回帰分析と同じ原理です。

ボストンデータセットとは

> ボストンの506地区について、下記13属性値と住宅平均価格を表にしたものです。ここでは、13の属性値から住宅価格が決まるというモデルを仮定して、１次式モデルの係数を推定します。

| columns | 説明 |
|---|---|
| CRIM | 町ごとの人口１人当たりの犯罪率 |
| ZN | 宅地の比率。25000平方フィート以上のゾーンで数えた値 |
| INDUS | 町ごとの非小売業の面積比 |
| CHAS | チャールズ川へ道がつながっているか |
| NOX | NO_x濃度 |
| RM | 在宅当たり部屋数 |
| AGE | 1940年以前に建てられた、所有者が住む建物の割合 |
| DIS | ボストンの５つの雇用中心からの距離 |
| RAD | 放射状幹線道路からの距離 |

| columns | 説明 |
|---|---|
| TAX | 固定資産税率 |
| PTRATIO | 町ごとの教師当たりの生徒数 |
| B | Bkは町ごとの黒人の比率 |
| LSTAT | 低階層人口の比率% |
| MEDV | 所有者が住む住宅の価値の中央値（target） |

In [32]:

```
# リスト3-5  scikit-learnのlinear_modelを使った重回帰分析の例（ボストンの住宅価格）
# -*- coding: utf-8 -*-
# scikit-learn linear_modelを使ったボストン住宅価格の線形回帰
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import linear_model, datasets
dset = datasets.load_boston()
print(dset.DESCR) #データに付属しているdescriptionに書かれている
```

```
.. _boston_dataset:

Boston house prices dataset
---------------------------

**Data Set Characteristics:**

    :Number of Instances: 506

    :Number of Attributes: 13 numeric/categorical predictive. Median Value (attribute 14) is usually the target.

    :Attribute Information (in order):
        - CRIM     per capita crime rate by town
        - ZN       proportion of residential land zoned for lots over 25,000 sq.ft.
        - INDUS    proportion of non-retail business acres per town
        - CHAS     Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
        - NOX      nitric oxides concentration (parts per 10 million)
        - RM       average number of rooms per dwelling
        - AGE      proportion of owner-occupied units built prior to 1940
        - DIS      weighted distances to five Boston employment centres
        - RAD      index of accessibility to radial highways
        - TAX      full-value property-tax rate per $10,000
        - PTRATIO  pupil-teacher ratio by town
        - B        1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
        - LSTAT    % lower status of the population
        - MEDV     Median value of owner-occupied homes in $1000's

    :Missing Attribute Values: None

    :Creator: Harrison, D. and Rubinfeld, D.L.

This is a copy of UCI ML housing dataset.
https://archive.ics.uci.edu/ml/machine-learning-databases/housing/


This dataset was taken from the StatLib library which is maintained at Carnegie Mellon University.

The Boston house-price data of Harrison, D. and Rubinfeld, D.L. 'Hedonic
prices and the demand for clean air', J. Environ. Economics & Management,
vol.5, 81-102, 1978.   Used in Belsley, Kuh & Welsch, 'Regression diagnostics
...', Wiley, 1980.   N.B. Various transformations are used in the table on
pages 244-261 of the latter.

The Boston house-price data has been used in many machine learning papers that address regression
problems.

.. topic:: References

   - Belsley, Kuh & Welsch, 'Regression diagnostics: Identifying Influential Data and Sources of Collinearity', Wiley, 1980. 244-261.
   - Quinlan,R. (1993). Combining Instance-Based and Model-Based Learning. In Proceedings on the Tenth International Conference of Machi
ne Learning, 236-243, University of Massachusetts, Amherst. Morgan Kaufmann.
```

In [33]:

```
boston = pd.DataFrame(dset.data)    #説明変数
boston.columns = dset.feature_names
target = pd.DataFrame(dset.target)  #目的変数

model = linear_model.LinearRegression()
model.fit(boston, target)
# 偏回帰係数
print(pd.DataFrame({"Name":boston.columns,¥
    "Coefficients":model.coef_[0]}).sort_values(by='Coefficients').round(4) )
# 切片（誤差）
print('intercept', model.intercept_[0].round(4))
```

```
        Name  Coefficients
4        NOX       -17.7666
7        DIS        -1.4756
10   PTRATIO        -0.9527
12     LSTAT        -0.5248
0       CRIM        -0.1080
9        TAX        -0.0123
6        AGE         0.0007
11         B         0.0093
2      INDUS         0.0206
1         ZN         0.0464
8        RAD         0.3060
3       CHAS         2.6867
5         RM         3.8099
intercept 36.4595
```

説明変数**RM**（1住居当たりの部屋数）が最も強く住宅価格を押し上げる要因になっており、**CHAS**（チャールズ川に面した地域）が上から2番目となっています。ただし**CHAS**変数は1か0の2値なので、注意が必要です。他方、**NOX**（$NO_x$の汚染）が最も強いマイナス要因になっており、かなり下がっ

て**DIS**（ボストンの５つの雇用中心からの距離）、**PTRATIO**（教師に対する生徒の比率）、**LSTAT**（低階層人口の比率）となっていることがわかります。

さらに、StatsModelsパッケージのOLS（Ordinary Least Squares）モジュールを使って重回帰分析を行うと、さまざまな追加情報を得ることができます。

In [34]:
```
#sample3-1-3.py
import statsmodels
import statsmodels.api as sm
# statmodels.OLS
# examples: http://www.statsmodels.org/dev/examples/notebooks/generated/ols.html
# manual --- http://www.statsmodels.org/dev/regression.html#module-reference
# OLSREsultsのmanual --- http://www.statsmodels.org/dev/generated/statsmodels.¥
#     regression.linear_model.OLSResults.html¥
#     #statsmodels.regression.linear_model.OLSResults

model3 = sm.OLS(target, sm.add_constant(boston))
result3 = model3.fit()
print(result3.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      0   R-squared:                       0.741
Model:                            OLS   Adj. R-squared:                  0.734
Method:                 Least Squares   F-statistic:                     108.1
Date:                Mon, 10 May 2021   Prob (F-statistic):          6.72e-135
Time:                        21:27:48   Log-Likelihood:                -1498.8
No. Observations:                 506   AIC:                             3026.
Df Residuals:                     492   BIC:                             3085.
Df Model:                          13
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         36.4595      5.103      7.144      0.000      26.432      46.487
CRIM          -0.1080      0.033     -3.287      0.001      -0.173      -0.043
ZN             0.0464      0.014      3.382      0.001       0.019       0.073
INDUS          0.0206      0.061      0.334      0.738      -0.100       0.141
CHAS           2.6867      0.862      3.118      0.002       0.994       4.380
NOX          -17.7666      3.820     -4.651      0.000     -25.272     -10.262
RM             3.8099      0.418      9.116      0.000       2.989       4.631
AGE            0.0007      0.013      0.052      0.958      -0.025       0.027
DIS           -1.4756      0.199     -7.398      0.000      -1.867      -1.084
RAD            0.3060      0.066      4.613      0.000       0.176       0.436
TAX           -0.0123      0.004     -3.280      0.001      -0.020      -0.005
PTRATIO       -0.9527      0.131     -7.283      0.000      -1.210      -0.696
B              0.0093      0.003      3.467      0.001       0.004       0.015
LSTAT         -0.5248      0.051    -10.347      0.000      -0.624      -0.425
==============================================================================
Omnibus:                      178.041   Durbin-Watson:                   1.078
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              783.126
Skew:                           1.521   Prob(JB):                    8.84e-171
Kurtosis:                       8.281   Cond. No.                     1.51e+04
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.51e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

詳細な読み方は

http://www.statsmodels.org/dev/generated/statsmodels.regression.linear_model.OLSResults.html#statsmodels.regression.linear_model.OLSResults を参照してください。

***R-squared***が決定係数の$R^2$の値であり、0.741を得ているので、この回帰モデルはそれなりに実データを説明できているといえます。

表になっている部分では、***coef***が回帰式のそれぞれの説明変数における係数で、そのなかで***const***は切片です。また、***std_err***が標準誤差、***t***はt値、***P>|t|***はp値、***[0.025 0.975]***は信頼区間を表示しています。

In [35]:
```
print(result3.pvalues)
```

```
const      3.283438e-12
CRIM       1.086810e-03
ZN         7.781097e-04
INDUS      7.382881e-01
CHAS       1.925030e-03
NOX        4.245644e-06
RM         1.979441e-18
AGE        9.582293e-01
DIS        6.013491e-13
RAD        5.070529e-06
TAX        1.111637e-03
PTRATIO    1.308835e-12
B          5.728592e-04
LSTAT      7.776912e-23
dtype: float64
```

p値が極端に大きいのは、**INDUS**（商業用途の面積の比率）が0.735、**AGE**（1940年以前に建てられた割合）が0.955で、これらはいずれも係数が信頼できないことになります。一方、上記で取り上げた回帰式の係数の値が際立っている説明変数RM、**CHAS**、**NOX**などは、いずれも信頼性があると考えられるp値になっています。

結果のsummaryの中に***AIC***（Akaike's Information Criterion：赤池情報量基準）と***BIC***（Bayesian Information Criterion：ベイズ情報量基準）が出ています。これらはいずれもモデルの選択に使われる指標で、「小さいほどよく適合している」といわれています。簡単にいうと、モデルに多数の項を含めるとサンプル点ではよくフィットするようになりますが、他方でいわゆる過適合（オーバーフィッティング）が起こるので、データの量と適合度のバランスをとることが求められます。つまり、データの適合具合に項の多さのペナルティを加味した値と考えられます。

重回帰分析に特徴的な問題として、異なる説明変数の間に線形（依存）関係が生じてしまう**多重共線性**があります。多重共線性があると回帰係数が求められない、もしくは回帰係数の分散が大きくなって正しく求められない（誤差が大きい）といったことが起こります。このような説明変数を検出するには、基本的に説明変数のすべてのペアに対して相関係数を計算し、一定以上の相関がある（正負を含めて）場合には多重共線性を避けるためにそ

の説明変数を外すことが考えられます。また、分散拡大係数（VIF:Variance Inflation Factor）を計算してその値によって除外の要否を判断することもできます。たとえば変数aとbの間のVIFは、ab間の相関係数を$r_{ab}$とすると

$$VIF_{ab} = \frac{1}{1 - r_{ab}^2}$$

で定義されるので、対象とする説明変数が他の変数と依存関係がない（直行している）ときはVIFが1となり、そうでないときには1より大きくなります。

一般にVIFの値が10を超えると、依存関係が強いために適切に重回帰分析ができないといわれています。StatsModelsでは、`statsmodels.stats.outliers_influence.variance_inflation_factor` を使ってVIFを計算することができます。上記のボストン住宅価格の例で `model3` にフィットした後、次の処理を行ってみました。

ドキュメントはhttp://www.statsmodels.org/devel/generated/statsmodels.stats.outliers_influence.variance_inflation_factor.html

In [36]:
```
from statsmodels.stats.outliers_influence import *

num_cols = model3.exog.shape[1]  # 説明変数の列数
vifs = [variance_inflation_factor(model3.exog, i)
for i in range(0, num_cols)]
pdv = pd.DataFrame(vifs, index=model3.exog_names, columns=["VIF"])
print(pdv)
```

```
                VIF
const    585.265238
CRIM       1.792192
ZN         2.298758
INDUS      3.991596
CHAS       1.073995
NOX        4.393720
RM         1.933744
AGE        3.100826
DIS        3.955945
RAD        7.484496
TAX        9.008554
PTRATIO    1.799084
B          1.348521
LSTAT      2.941491
```

constに対するVIFは説明変数に関係ないので除いて考えるとして、その他の説明変数のなかでは、**TAX**=9.01が10に近い他、**RAD**=7.48も5を超えています。どちらも10を超えていないのですが、要注意ということになるでしょう。

しかし、幸いどちらも回帰係数が小さく、あまり価格に影響していないと思われるので、この場合は気にすることはないと思われます。