Linear Regression

Linear Regression

- Predict the value of a variable depending on the value of another variable
- Simplest form: Calculate slope (m) and intersection (x) to fit equation $y = m * x + c$
- Tools and libraries:
  - Scipy.polyfit()
  - Numpy.polyfit( )
  - numpy.linalg.lstsq()
  - Scipy.stats.linregress
  - statsmodels
  - Seaborn regplot (plot only)

Linear Regression - Dataset

Linear Regression - Dataset

In [4]:
```
df = pd.read_csv('data/cafe.csv', parse_dates=['date'])
df.head(3)
```

Out[4]:

	date	temperature	sold_icecream	sold_cups_coffee	sold_coke
0	2018-06-29	28	40	57	44
1	2018-06-30	25	36	61	19
2	2018-07-01	31	45	53	15

Linear Regression - Dataset

In [4]:
```
df = pd.read_csv('data/cafe.csv', parse_dates=['date'])
df.head(3)
```

Out[4]:

	date	temperature	sold_icecream	sold_cups_coffee	sold_coke
0	2018-06-29	28	40	57	44
1	2018-06-30	25	36	61	19
2	2018-07-01	31	45	53	15

In [5]:
```
df.corr()
```

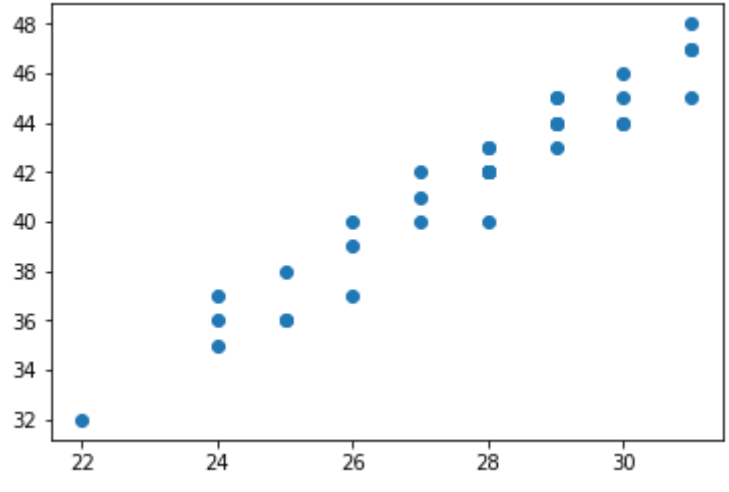
Out[5]:

	temperature	sold_icecream	sold_cups_coffee	sold_coke
temperature	1.000000	0.966549	-0.932512	0.002587
sold_icecream	0.966549	1.000000	-0.934670	-0.002490
sold_cups_coffee	-0.932512	-0.934670	1.000000	0.093498
sold_coke	0.002587	-0.002490	0.093498	1.000000

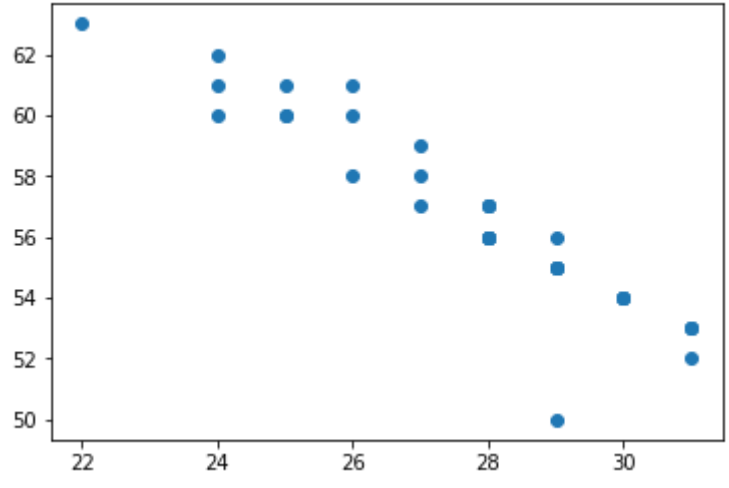
Scatter Plots

Scatter Plots

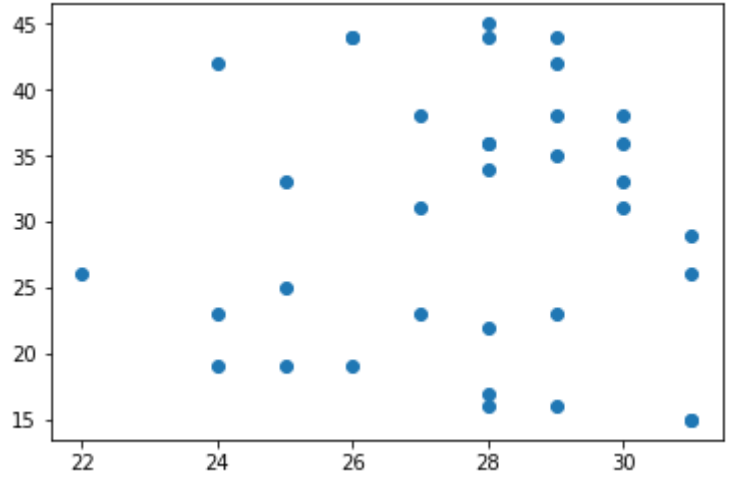
In [5]:
```
temperature = df['temperature']
sold_icecream = df['sold_icecream']
plt.scatter(temperature, sold_icecream);
```



In [7]:
```
sold_coffee = df['sold_cups_coffee']
plt.scatter(temperature, sold_coffee);
```



In [6]:
```
sold_coke = df['sold_coke']
plt.scatter(temperature, sold_coke);
```



Linear Regression with NumPy

- numpy.polyfit()
- Least squares polynomial fit
- For simple linear regression we search for a polynomial of degree 1

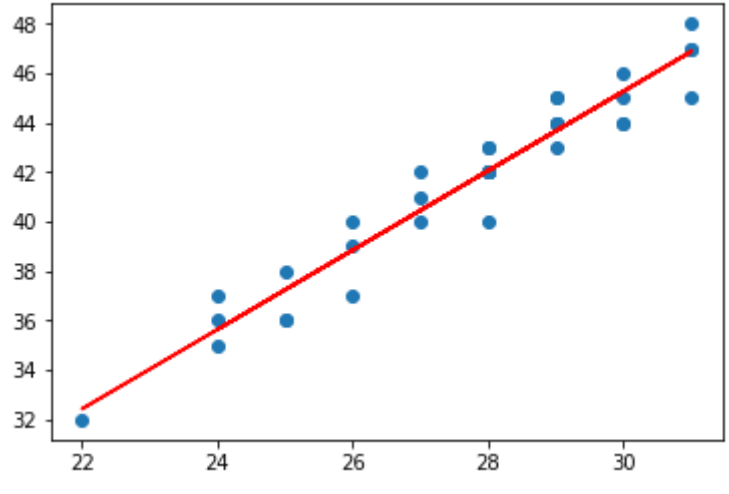
Linear Regression with NumPy

- numpy.polyfit()
- Least squares polynomial fit
- For simple linear regression we search for a polynomial of degree 1

In [9]:
```
m, c = np.polyfit(temperature, sold_icecream, 1)
print(f'm={m}, c={c}')

plt.scatter(temperature, sold_icecream);
plt.plot(temperature, m * temperature + c, 'r');
```

```
m=1.60646687697161, c=-2.9220820189274543
```



Linear Regression with NumPy

- Module numpy.linalg

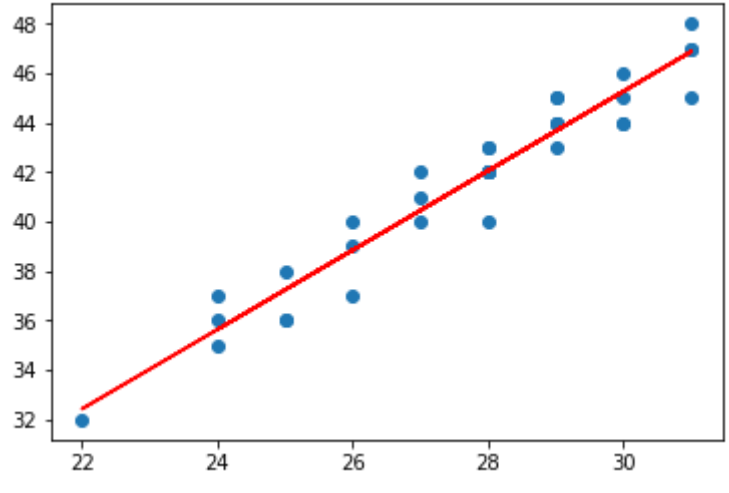
Linear Regression with NumPy

- Module numpy.linalg

In [12]:
```
a = np.vstack([temperature, np.ones(len(temperature))]).T
m, c = np.linalg.lstsq(a, sold_icecream, rcond=None)[0]
print(f'm={m}, c={c}')

plt.scatter(temperature, sold_icecream);
plt.plot(temperature, m*temperature + c, 'r');
```

```
m=1.6064668769716095, c=-2.9220820189274526
```



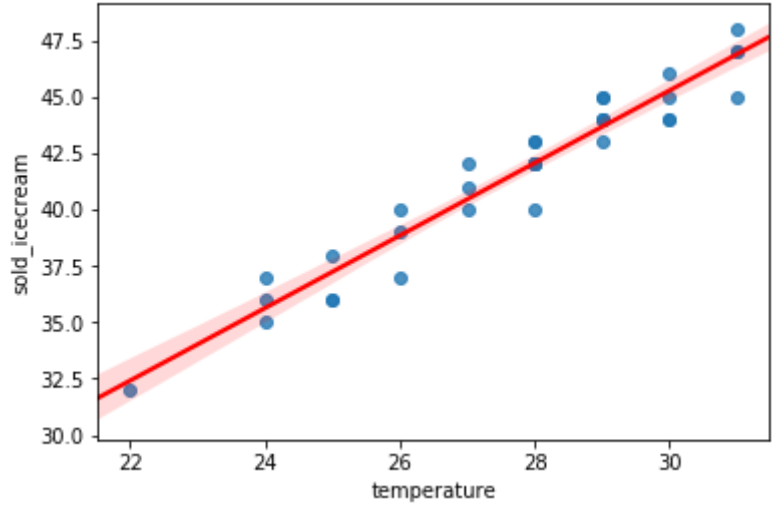
Linear Regression with Seaborn

- Plots the regression line and the confidence interval
- There is no way to get back the calculated coefficients and stats

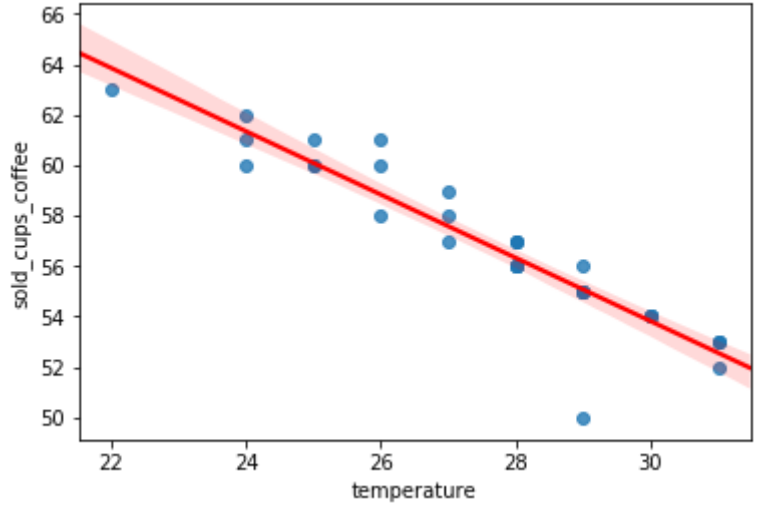
Linear Regression with Seaborn

- Plots the regression line and the confidence interval
- There is no way to get back the calculated coefficients and stats

In [10]:
```
sns.regplot(temperature, sold_icecream, line_kws={'color':'r'});
```



In [11]:
```
sns.regplot(temperature, sold_coffee, line_kws={'color':'r'});
```



Complete Regression Analysis process

- Draw a scatter plot of the independent variable versus the dependent variable
- Calculate the correlation coefficient
- Calculate the regression equation
- Conduct an analysis of variance
- Calculate the confidence intervals
- Make a prediction

Exercise 9

- Load the Rossmann sales data
- Analyse how sales might be related to the number of customers
  - Calculate and visualize the linear regression using different tools
  - Hint: Work with a sample of the data if calculation takes too long
- Load the Tips example data set
  - `tips = sns.load_dataset('tips')`
- Find out how the tip might be related to the total bill amount