Task2

December 2, 2018

```
In [49]: import pandas as pd
         import numpy as np
```

Считаем наши данные и посмотрим на несколько первых:

```
In [50]: data = pd.read_csv("Regression.csv", header=None, names=["distance"])
         data.head()
```

Out[50]:

	distance
0	14.7978
1	29.0651
2	43.3372
3	57.5955
4	70.1335

Для удобства сделаем из них numpy-массив:

```
In [51]: values = np.array(data.values, dtype=float)
```

Для использования линейной регрессионной модели перепишем нашу систему в таком виде:

$$X_0 = \beta_1 + \epsilon_0 X_1 - X_0 = \beta_2 + \epsilon_1 X_2 - X_1 = \beta_2 + \epsilon_2 \dots X_n - X_{n-1} = \beta_2 + \epsilon_n$$

Тогда в нашей моделе $X = (X_0, X_1 - X_0, \dots, X_n - X_{n-1})^T, \theta = (\beta_1, \beta_2)^T$,

$$z = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ \dots & \\ 0 & 1 \end{pmatrix}$$

Создадим вектор X и матрицу z:

```
In [52]: X = np.append(np.array(values[0]), values[1:] - np.roll(values, 1)[1:])
```

```
In [53]: Z = np.array([[0, 1] if i > 0 else [1, 0] for i in range(len(X))])
```

На семинаре было доказано, что оценки методом наименьших квадратов:

$$\hat{\theta} = \begin{pmatrix} X_0 \\ \frac{X_n - X_0}{n} \end{pmatrix}$$

Посчитаем данный вектор:

```
In [54]: coeffs = np.array([values[0][0], (values[-1] - values[0])[0] / (len(X) - 1)])
```

```
In [55]: print("beta_1 = ", coeffs[0])
         print("beta_2 = ", coeffs[1])
```

```
beta_1 =  14.7978
beta_2 =  14.050086986986987
```

Для несмещенной оценки $\hat{\sigma^2}$ получаем:

$$\hat{\sigma^2} = \frac{1}{n-1}\left|\left|X - z\hat{\theta}\right|\right|^2$$

```
In [56]: sigma = np.linalg.norm(X - Z @ coeffs) ** 2 / (len(X) - 1)
```

```
In [57]: print("sigma^2 = ", sigma)
```

```
sigma^2 =  2.315951410461287
```

Так как $\epsilon_i \sim N(0, \sigma^2)$ - ошибка приращения расстояния, а $\epsilon_i^t = \frac{\epsilon_i}{\beta_2}$ - ошибка отсчета времени, то оценка дисперсии отсчета времени равна $\frac{\hat{\sigma^2}}{\beta_2^2}$. Посчитаем ее:

```
In [58]: variance = sigma / (coeffs[1] ** 2)
```

```
In [59]: print("variance = ", variance)
```

```
variance =  0.011731982793957325
```

Вывод: мы смогли, преобразовав задачу, свести ее к линейной моделе и посчитать оценки наименьших квадратов для начального расстояния и скорости, а также оценить дисперсию отсчета времени.