STA2002 Assignment 2

Chen Ang (118010009)

1

(a)

	x^D	x^S	y^D	y^S
Sample mean	54.26	54.27	47.83	47.84
Sample variance	281.07	281.20	725.52	725.24

(b)

$$\hat{\alpha} = 47.83, \hat{\beta} = -0.10$$
$$\hat{a} = 47.84, \hat{\beta} = -0.10$$

(c)

For dinosaur, we have

$$\begin{aligned} \hat{\alpha} &= 47.83, n^D = 142, t_{0.025}(140) \approx z_{0.025} = 1.96 \\ \hat{\beta} &= -0.10, S_R^D = 35.68 \\ \sum_i (x_i^D - \bar{x}^D)^2 &= 39630.87 \end{aligned}$$

Therefore a 95% C.I. for α is

$$\hat{\alpha} \pm t_{0.025}(n-2) S_R^D \sqrt{1/n^D} = \boxed{47.83 \pm 5.87}$$

For β it is

$$\hat{\beta} \pm t_{0.025}(n-2) S_R^D \sqrt{\frac{1}{\sum_i (x_i^D - \bar{x}^D)^2}} = \boxed{-0.10 \pm 0.35}$$

For star, we have

$$\begin{aligned} \hat{a} &= 47.84, n^S = 142, t_{0.025}(140) \approx z_{0.025} = 1.96 \\ \hat{\beta} &= -0.10, S_R^S = 35.50 \\ \sum_j (x_j^S - \bar{x}^S)^2 &= 39648.92 \end{aligned}$$

Therefore a 95% C.I. for α is

$$\hat{a} \pm t_{0.025}(n-2) S_R^S \sqrt{1/n^S} = \boxed{47.84 \pm 5.84}$$

For β it is

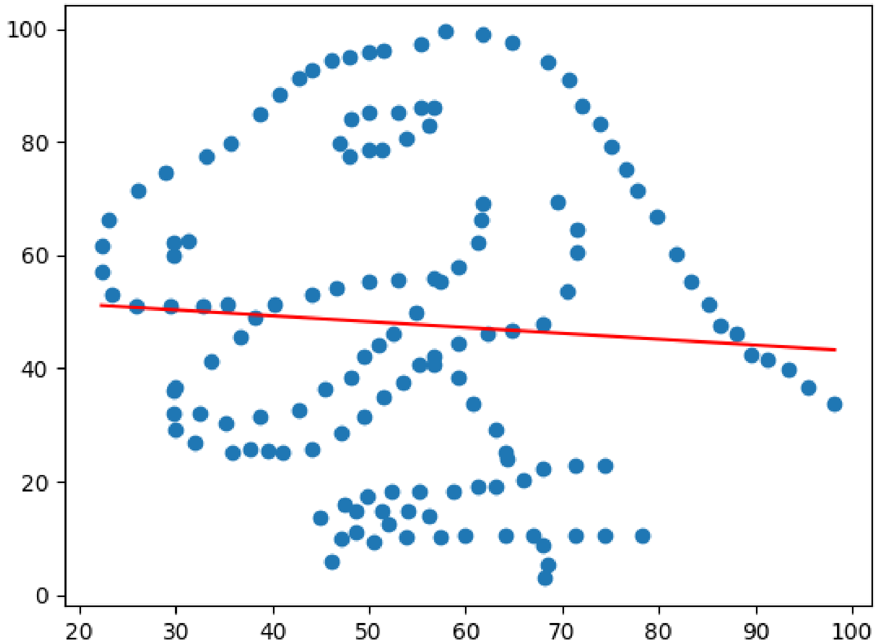
$$\hat{b} \pm t_{0.025}(n-2) S_R^S \sqrt{\frac{1}{\sum_i (x_i^S - \bar{x}^S)^2}} = \boxed{-0.10 \pm 0.35}$$

(d)

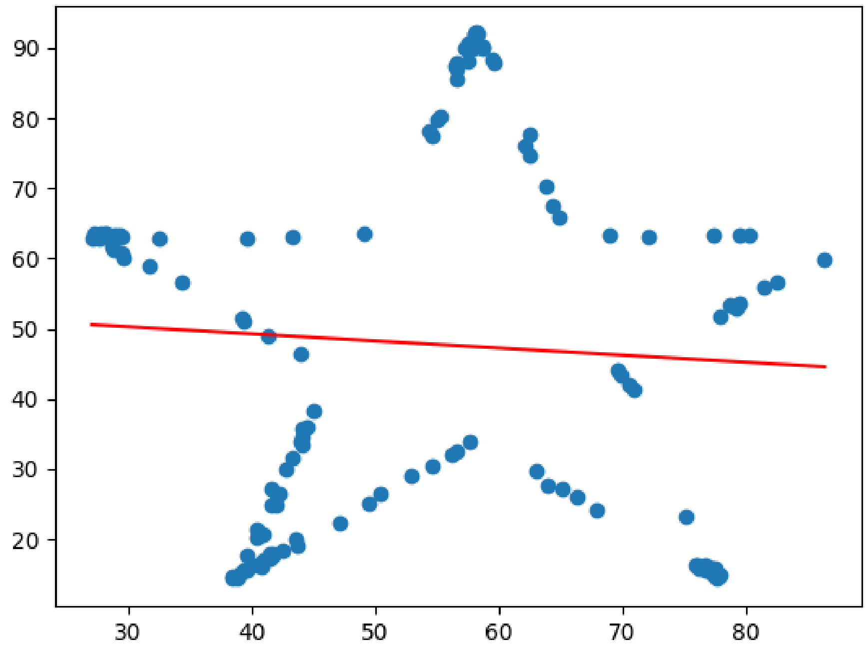
They are almost identical in terms of those statistics.

(e)

Dinosaur:



Star:



(f)

Simple linear regression model is indeed simple in that it fails to capture the complicated geometries of two datasets (that is, it cannot distinguish a dinosaur from a star, or even from a genuine straight line). This is because the simple linear regression model is based on the a-prior assumption that the samples are linearly distributed with some gaussian noise - we didn't do anything to validate this assumption! The example tells us we should not blindly apply the regression. Rather, we should always look at the scatter plot first for a gut feeling of what the distribution might be (or carry out some statistical tests for linearity).

Python Code:

```
import csv
import os, sys
import matplotlib.pyplot as plt
import numpy as np

class LinReg:
    def __init__(self, name=None, file=None, reg=True):
        self.name = name
        self.xs = self.ys = []
        self.size = 0
        self.x_mean = self.y_mean = None
        self.x_S2 = self.y_S2 = None
        self.a = self.b = self.sigma2 = self.SR2 = None
        self.a_max_err = self.b_max_err = None
        if file:
            self.load_data(file)
            if reg: self.regress()
```

```
    def load_data(self, file):
        self.__init__(self.name)
        with open(file, newline='') as f:
            reader = csv.reader(f)
            data = list(reader)[1::]
            self.size = len(data)
            self.xs = [float(x) for [x, y] in data]
            self.ys = [float(y) for [x, y] in data]
            self.x_mean = sum(self.xs) / self.size
            self.y_mean = sum(self.ys) / self.size
            self.x_S2 = sum((x - self.x_mean) ** 2 for x in
self.xs) / (self.size - 1)
            self.y_S2 = sum((y - self.y_mean) ** 2 for y in
self.ys) / (self.size - 1)

    def regress(self, confidence=0.95):
        xy_sum = sum(x * y for x, y in zip(self.xs, self.ys))
        x2_sum = sum(x ** 2 for x in self.xs)
        y2_sum = sum(y ** 2 for y in self.ys)
        num = xy_sum - self.size * self.x_mean * self.y_mean
        den = x2_sum - self.size * self.x_mean ** 2
        x_SE = (self.size - 1) * self.x_S2
        t = 1.96 # should ideally be a t-quantile function,
estimated by z_0.025

        self.a = self.y_mean
        self.b = num / den
        self.sigma2 = y2_sum / self.size - self.y_mean ** 2 - \
                      self.b * (xy_sum / self.size + self.x_mean *
self.y_mean)
        self.SR2 = self.sigma2 * self.size / (self.size - 2)
        self.a_max_err = t * (self.SR2 / self.size) ** .5
        self.b_max_err = t * (self.SR2 / x_SE) ** .5

        self.print_stat()

    def plot(self):
        plt.scatter(self.xs, self.ys)
        if (self.a and self.b):
            X = np.arange(min(self.xs), max(self.xs), 0.01)
            Y = self.a + self.b * (X - self.x_mean)
            plt.plot(X, Y, 'r')
        plt.show()

    def print_stat(self):
        print("{} (n = {})".format(self.name, self.size))
        print("{:-^60}".format(" statistics"))
        print("{:<13}{:<13}{:<13}".format("", "x", "y"))
        print("{:<13}{:<13.2f}{:<13.2f}".format("mean",
self.x_mean, self.y_mean))
        print("{:<13}{:<13.2f}{:<13.2f}\n".format("sample var",
self.x_S2, self.y_S2))
```

```
        print("{:-^60}".format(" regression "))
        print("{:<13}{:<13}{:<13}{:<13}{:<13}".format("", "a^",
"b^", "σ2^", "SR2"))
        print("{:<13}{:<13.2f}{:<13.2f}{:<13.2f}{:
<13.2f}".format("estimator", self.a, self.b, self.sigma2,
self.SR2))
        print("{:<13}{:<13.2f}{:<13.2f}{:<13.2}{:
<13.2}\n".format("max error", self.a_max_err, self.b_max_err, "",
""))

dinosaur = LinReg("Dinosaur", os.path.join(sys.path[0], 'D.csv'))
star = LinReg("Star", os.path.join(sys.path[0], 'S.csv'))
dinosaur.plot()
star.plot()
```

Output

```
Dinosaur (n = 142)
------------------------ statistics------------------------
             x            y
mean         54.26        47.83
sample var   281.07       725.52

------------------------ regression ------------------------
             a^           b^           σ2^          SR2
estimator    47.83        -0.10        1255.12      1273.05
max error    5.87         0.35

Star (n = 142)
------------------------ statistics------------------------
             x            y
mean         54.27        47.84
sample var   281.20       725.24

------------------------ regression ------------------------
             a^           b^           σ2^          SR2
estimator    47.84        -0.10        1242.28      1260.03
max error    5.84         0.35
```

2

Denote the outcome of each coin flip by $X_1, \cdots X_n \overset{\text{i.i.d.}}{\sim} \text{Bernoulli}(p)$.

(a)

If n is large enough by CLT we have

$$\hat{p} = \bar{X} = \frac{\sum_{i=1}^{n} X_i}{n} \overset{\text{approx}}{\sim} N\left(p, \frac{p(1-p)}{n}\right) \approx N\left(p, \frac{\hat{p}(1-\hat{p})}{n}\right)$$

Hence a 90% C.I. for p can be approximated by

$$\hat{p} \pm z_{0.05}\sqrt{\frac{\hat{p}(1-\hat{p})}{n}}$$

In our example

$$\hat{p} = \frac{159}{314} = 0.5064, n = 314$$

Plugging in $z_{0.05} = 1.645$, we obtain the approximated 90% C.I. for p :

$$0.5064 \pm 0.0464$$

(b)

Keeping $\hat{p} = 50.64\%$, we want the maximum error at

$$z_{0.05}\sqrt{\frac{\hat{p}(1-\hat{p})}{n}} = 0.01$$

Therefore

$$\begin{aligned} n &= \hat{p}(1-\hat{p})\left(\frac{0.01}{z_{0.05}}\right)^{-2} \\ &= 6764 \end{aligned}$$

(c)

That is

$$\hat{p} \pm z_{\alpha/2}\sqrt{\frac{\hat{p}(1-\hat{p})}{314}} = 0.445 \pm 0.055$$

Therefore

$$\begin{aligned} \hat{p} &= 0.445 \\ z_{\alpha/2} &= 0.055\sqrt{\frac{314}{0.445(1-0.445)}} \\ &= 1.9411 \\ \implies \alpha &= 0.0524 \end{aligned}$$

Thus my friend is using $100(1-\alpha)\% \approx 95\%$ confidence level.

3

(a)

We have

$$\bar{X} = \frac{\sum_{i=1}^{n} X_i}{n} \sim N(\mu_X, \sigma_X^2/n)$$
$$\bar{Y} = \frac{\sum_{j=1}^{m} Y_j}{m} \sim N(\mu_Y, \sigma_Y^2/m)$$

which are independent. Hence

$$\bar{X} - \bar{Y} \sim N(\mu_X - \mu_Y, \sigma_X^2/n + \sigma_Y^2/m)$$
$$\frac{\bar{X} - \bar{Y} - (\mu_X - \mu_Y)}{\sqrt{\sigma_X^2/n + \sigma_Y^2/m}} \sim N(0,1)$$

It follows that

$$\begin{aligned} 1 - \alpha &= P\left(-z_{\alpha/2} \leq \frac{\bar{X} - \bar{Y} - (\mu_X - \mu_Y)}{\sqrt{\sigma_X^2/n + \sigma_Y^2/m}} \leq z_{\alpha/2}\right) \\ &= P\left(\bar{X} - \bar{Y} - \epsilon \leq \mu_X - \mu_Y \leq \bar{X} - \bar{Y} + \epsilon\right) \end{aligned}$$

where $\epsilon = z_{\alpha/2}\sqrt{\sigma_X^2/n + \sigma_Y^2/m}$.

Therefore a two-sided $100(1-\alpha)\%$ C.I. for $\mu_X - \mu_Y$ is given by

$$\bar{x} - \bar{y} \pm z_{\alpha/2}\sqrt{\sigma_X^2/n + \sigma_Y^2/m}$$

(b)

We need

$$\begin{aligned} n &= \arg\min_{n,m \in \mathbb{N}^2} \epsilon \quad \text{s.t.} \quad n + m = 6000 \\ &= \arg\min_{n,m \in \mathbb{N}^2} z_{\alpha/2}\sqrt{\sigma_X^2/n + \sigma_Y^2/m} \quad \text{s.t.} \quad n + m = 6000 \\ &= \arg\min_{n,m \in \mathbb{N}^2} \sigma_X^2/n + \sigma_Y^2/m \quad \text{s.t.} \quad n + m = 6000 \\ &= \arg\min_{n \in \{0,1,\cdots,6000\}} \sigma_X^2/n + \sigma_Y^2/(6000 - n) \\ &= \arg\min_{n \in \{0,1,\cdots,6000\}} 2500/n + 900/(6000 - n) \\ &= 3750 \end{aligned}$$

samples from Company X

4

Sample size $n = 12$.

(a)

A $100(1-\alpha)\%$ C.I. for μ is given by

$$\begin{aligned}\bar{x} \pm z_{\alpha/2}(\sigma/\sqrt{n}) &= 41.83 \pm z_{\alpha/2}(11/\sqrt{12}) \\ &= 41.83 \pm z_{\alpha/2} \cdot 3.175\end{aligned}$$

To obtain a 90% C.I., we set $\alpha = 0.1$, $z_{\alpha/2} = 1.645$, which gives

$$41.83 \pm 5.22$$

(b)

Set $\alpha = 0.05$, $z_{\alpha/2} = 1.96$. Thus a 95% C.I. for μ is given by

$$41.83 \pm 6.22$$

Set instead $\alpha = 0.01$, $z_{\alpha/2} = 2.576$. This yields a 99% C.I. for μ :

$$41.83 \pm 8.18$$

(c)

Without the knowledge of σ^2, we estimate it by S^2 and reach a $100(1-\alpha)\%$ C.I.:

$$\begin{aligned}\bar{x} \pm t_{\alpha/2}(n-1)(s/\sqrt{n}) &= 41.83 \pm t_{\alpha/2}(11)(11.8/\sqrt{12}) \\ &= 41.83 \pm t_{\alpha/2}(11) \cdot 3.406\end{aligned}$$

Set $\alpha = 0.1, t_{\alpha/2}(11) = 1.796$. The $100(1-\alpha)\%$ C.I. is given by

$$41.83 \pm 6.12$$

5

(a)

Consider

$$U := \sum_{i=1}^{n} \left(\frac{X_i - \mu}{\sigma} \right)^2 = \sum_{i=1}^{n} Z_i^2 = \sum_{i=1}^{n} C_i$$

where $Z_i \overset{\text{i.i.d}}{\sim} N(0,1), C_i \overset{\text{i.i.d}}{\sim} \chi^2(1)$.

Due to additivity of i.i.d. Chi-square RVs,

$$U \sim \chi^2(n)$$

Hence

$$
\begin{aligned}
1-\alpha &= P\left(\chi^2_{1-\alpha/2}(n) \le U \le \chi^2_{\alpha/2}(n)\right) \\
&= P\left(\chi^2_{1-\alpha/2}(n) \le \sum_{i=1}^{n}\left(\frac{X_i-\mu}{\sigma}\right)^2 \le \chi^2_{\alpha/2}(n)\right) \\
&= P\left(\chi^2_{1-\alpha/2}(n) \le \frac{1}{\sigma^2}\sum_{i=1}^{n}(X_i-\mu)^2 \le \chi^2_{\alpha/2}(n)\right) \\
&= P\left(\frac{\sum_{i=1}^{n}(X_i-\mu)^2}{\chi^2_{\alpha/2}(n)} \le \sigma^2 \le \frac{\sum_{i=1}^{n}(X_i-\mu)^2}{\chi^2_{1-\alpha/2}(n)}\right)
\end{aligned}
$$

(b)

Consider

$$
W := \sum_{i=1}^{n}\left(\frac{X_i-\bar{X}}{\sigma}\right)^2 = \frac{(n-1)S^2}{\sigma^2} \sim \chi^2(n-1)
$$

Similarly

$$
\begin{aligned}
1-\alpha &= P\left(\chi^2_{1-\alpha/2}(n-1) \le W \le \chi^2_{\alpha/2}(n-1)\right) \\
&= P\left(\chi^2_{1-\alpha/2}(n-1) \le \sum_{i=1}^{n}\left(\frac{X_i-\bar{X}}{\sigma}\right)^2 \le \chi^2_{\alpha/2}(n-1)\right) \\
&= P\left(\chi^2_{1-\alpha/2}(n-1) \le \frac{1}{\sigma^2}\sum_{i=1}^{n}(X_i-\mu)^2 \le \chi^2_{\alpha/2}(n-1)\right) \\
&= P\left(\frac{\sum_{i=1}^{n}(X_i-\mu)^2}{\chi^2_{\alpha/2}(n-1)} \le \sigma^2 \le \frac{\sum_{i=1}^{n}(X_i-\mu)^2}{\chi^2_{1-\alpha/2}(n-1)}\right)
\end{aligned}
$$

(c)

By (b),

$$
\begin{aligned}
1-\alpha &= P\left(\frac{\sum_{i=1}^{n}(X_i-\mu)^2}{\chi^2_{\alpha/2}(n-1)} \le \sigma^2 \le \frac{\sum_{i=1}^{n}(X_i-\mu)^2}{\chi^2_{1-\alpha/2}(n-1)}\right) \\
&= P\left(\sqrt{\frac{\sum_{i=1}^{n}(X_i-\mu)^2}{\chi^2_{\alpha/2}(n-1)}} \le \sigma \le \sqrt{\frac{\sum_{i=1}^{n}(X_i-\mu)^2}{\chi^2_{1-\alpha/2}(n-1)}}\right)
\end{aligned}
$$

6

Let

$$
f(\alpha,\beta) := \sum_{i=1}^{n} w_i(y_i-\alpha-\beta x_i)^2
$$

FONC:

$$
\begin{aligned}
\nabla_\alpha f(\alpha,\beta) &= \sum_{i=1}^{n} 2w_i(y_i-\alpha-\beta x_i)(-1) = 0 \\
\nabla_\beta f(\alpha,\beta) &= \sum_{i=1}^{n} 2w_i(y_i-\alpha-\beta x_i)(-x_i) = 0
\end{aligned}
$$

Hence

$$\sum_i w_i(y_i - \alpha - \beta x_i) = \sum_i w_i y_i - \alpha \sum_i w_i - \beta \sum_i w_i x_i = 0$$
$$\sum_i w_i x_i(y_i - \alpha - \beta x_i) = \sum_i w_i y_i x_i - \alpha \sum_i w_i x_i - \beta \sum_i w_i x_i^2 = 0$$

Writing in correspondence,

$$A - B\alpha - C\beta = 0$$
$$D - C\alpha - E\beta = 0$$

The system yields

$$\alpha = \frac{A}{B} - \frac{C}{B}\beta$$
$$\beta = \frac{BD - AC}{BE - C^2}$$

where $A = \sum_i w_i y_i, B = \sum_i w_i, C = \sum_i w_i x_i, D = \sum_i w_i y_i x_i, E = \sum_i w_i x_i^2$, which is to be shown.

7

(a)

$$\bar{x} = 3055.91, \bar{y} = 3317.91$$

(b)

A $100(1-\alpha)\%$ C.I. for $\mu_X - \mu_Y$ is

$$\bar{X} - \bar{Y} \pm t_{\alpha/2}(n+m-2) S_p \sqrt{\frac{1}{n} + \frac{1}{m}}$$

where

$$S_p^2 = \frac{(n-1)S_X^2 + (m-1)S_Y^2}{n+m-2}$$

is the pooled estimator of σ^2.

Since $n+m$ is large, we may use the approximation

$$\bar{X} - \bar{Y} \pm z_{\alpha/2} S_p \sqrt{\frac{1}{n} + \frac{1}{m}}$$

With $\alpha = 0.05$, plug in

$$n = 13391, m = 5672$$
$$\bar{X} = \bar{x} = 3055.91, \bar{Y} = \bar{y} = 3317.91$$
$$z_{\alpha/2} = 1.96, S_p = 1985.65$$

we have the pooled t-interval

$$-262.00 \pm 61.66$$

(c)

A $100(1-\alpha)\%$ Welch's t-interval for $\mu_X - \mu_Y$ is

$$\bar{X} - \bar{Y} \pm t_{\alpha/2}(r)\sqrt{\frac{S_X^2}{n} + \frac{S_Y^2}{m}}$$

where

$$r = \left\lfloor \frac{\left(\frac{S_X^2}{n} + \frac{S_Y^2}{m}\right)^2}{\frac{1}{n-1}\left(\frac{S_X^2}{n}\right)^2 + \frac{1}{m-1}\left(\frac{S_Y^2}{m}\right)^2} \right\rfloor$$

Since n, m are both large, r is also large. Hence we may use the approximation

$$\bar{X} - \bar{Y} \pm z_{\alpha/2}\sqrt{\frac{S_X^2}{n} + \frac{S_Y^2}{m}}$$

Plugging in the data, we obtain

$$-262.00 \pm 61.60$$

(d)

No. Instead, the data shows at 95% confidence level that there are *more* cars when the weather is "Rain".

Python Code

```
import csv
import os, sys
import matplotlib.pyplot as plt
import numpy as np

class Traffic:
    def __init__(self, file=None):
        self.xs = []    # traffic when clear
        self.n = 0
        self.x_mean = None
        self.x_S2 = None

        self.ys = []    # traffic when rain
        self.m = 0
        self.y_mean = None
        self.y_S2 = None

        self.Sp2 = None
        self.pooled_max_err = None
        self.welch_max_err = None

        if file:
            self.load_data(file)
```

```python
    def load_data(self, file):
        self.__init__()
        with open(file, newline='') as f:
            reader = csv.reader(f)
            data = list(reader)[1::]
            self.xs = [float(d[-1]) for d in data if d[5] == 'Clear']
            self.ys = [float(d[-1]) for d in data if d[5] == 'Rain']
            self.n, self.m = len(self.xs), len(self.ys)
            self.x_mean = sum(self.xs) / self.n
            self.y_mean = sum(self.ys) / self.m
            self.x_S2 = sum((x - self.x_mean) ** 2 for x in self.xs) / (self.n - 1)
            self.y_S2 = sum((y - self.y_mean) ** 2 for y in self.ys) / (self.m - 1)
            num = (self.n - 1) * self.x_S2 + (self.m - 1) * self.y_S2
            den = self.n + self.m - 2
            self.Sp2 = num / den
            self.pooled_max_err = 1.96 * self.Sp2 ** .5 * (1 / self.n + 1 / self.m) ** .5
            self.welch_max_err = 1.96 * (self.x_S2 / self.n + self.y_S2 / self.m) ** .5

    def plot(self):
        plt.scatter(self.xs, self.ys)
        if (self.a and self.b):
            X = np.arange(min(self.xs), max(self.xs), 0.01)
            Y = self.a + self.b * (X - self.x_mean)
            plt.plot(X, Y, 'r')
        plt.show()

    def print_stat(self):
        print("{:-^60}".format(" statistics"))
        print("{:<13}{:<13}{:<13}".format("", "x (clear)", "y (rain)"))
        print("{:<13}{:<13}{:<13}".format("size", self.n, self.m))
        print("{:<13}{:<13.2f}{:<13.2f}".format("mean", self.x_mean, self.y_mean))
        print("{:<13}{:<13.2f}{:<13.2f}".format("sample var", self.x_S2, self.y_S2))
        print("{:<13}{:<13.2f}({:<.2f})".format("Sp2 (Sp)", self.Sp2, self.Sp2 ** .5))
        print("{:<13}{:<13.2f}".format("pooled error", self.pooled_max_err))
        print("{:<13}{:<13.2f}\n".format("welch error", self.welch_max_err))

traffic = Traffic(os.path.join(sys.path[0], 'traffic.csv'))
traffic.print_stat()
```

Output

```
------------------------- statistics--------------------------
             x (clear)     y (rain)
size         13391         5672
mean         3055.91       3317.91
sample var   3948572.02    3929230.64
sp2 (Sp)     3942817.60    (1985.65)
pooled error 61.66
welch error  61.60
```