GNU

Lab Seminar: 2022. 07. 12

Linear Algebra and Probability & Statistics with Python Codes

Data Science from Scratch 2nd – Chapter 4, 5, 6

IDEALAB

Improving lives through learning

ChanKi Kim
School of Computer Science/Department of AI Convergence Engineering
Gyeongsang National University (GNU)

GNU 경상국립대학교 Gyeongsang National University

Contents

Additional Explanation of Last Seminar

- Python Encapsulation
  - 객체의 Data fields(속성)와 Methods(행위)를 하나로 묶음
  - 실제 구현 내용의 일부 은닉 가능

Additional Explanation of Last Seminar

- Python Encapsulation

```
class Chan_Encapsulation :
    def __init__(self, capsule1, capsule2, capsule3) :
        self.__capsule1 = capsule1
        self.__capsule2 = capsule2
        self.__capsule3 = capsule3

    def open_capsule1(self) :
        return self.__capsule1

    def __open_capsule2(self) :
        return self.__capsule2


real_capsule = Chan_Encapsulation('캡슐1', '캡슐2', '캡슐3')
print(real_capsule.open_capsule1())
print(help(Chan_Encapsulation))
print(real_capsule.__open_capsule2())
```

Additional Explanation of Last Seminar

- Python Encapsulation

```
캡슐1
Help on class Chan_Encapsulation in module __main__:

class Chan_Encapsulation(builtins.object)
 |  Chan_Encapsulation(capsule1, capsule2, capsule3)
 |
 |  Methods defined here:
 |
 |  __init__(self, capsule1, capsule2, capsule3)
 |      Initialize self.  See help(type(self)) for accurate signature.
 |
 |  open_capsule1(self)
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors defined here:
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)

None
```

Additional Explanation of Last Seminar

- Python Encapsulation

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Input In [1], in <cell line: 17>()
     15 print(real_capsule.open_capsule1())
     16 print(help(Chan_Encapsulation))
---> 17 print(real_capsule.__open_capsule2())

AttributeError: 'Chan_Encapsulation' object has no attribute '__open_capsule2'
```

Introduction

- Why should we learn Linear Algebra?
  - Machine Learning은 숫자를 이용해 복잡한 계산을 수행하는 것
  - 최소한의 코드로 규모가 큰 계산을 쉽게 PC에 지시 가능

$$k = ax_1 + bx_2 + c \quad \Longrightarrow \quad \begin{cases} k_1 = ax_1 + bx_2 + c_1 \\ k_2 = ay_1 + by_2 + c_2 \\ k_3 = az_1 + bz_2 + c_3 \\ k_4 = ap_1 + bp_2 + c_4 \\ k_5 = aq_1 + bq_2 + c_5 \end{cases}$$

Introduction

- Why should we learn Linear Algebra?

$$k = \begin{bmatrix} k_1 \\ k_2 \\ k_3 \\ k_4 \\ k_5 \end{bmatrix} \qquad A = \begin{bmatrix} x_1 & x_2 & c_1 \\ y_1 & y_2 & c_2 \\ z_1 & z_2 & c_3 \\ p_1 & p_2 & c_4 \\ q_1 & q_2 & c_5 \end{bmatrix} \qquad B = \begin{bmatrix} a \\ b \\ 1 \end{bmatrix}$$

$$k = AB$$

Introduction

- Why should we learn Probability & Statistics?
- Machine Learning의 궁극적인 목표는 예측을 수행하는 것
- 규모가 큰 데이터들을 확률과 통계를 활용하여 과거부터 축적되어온 데이터를 확률과 통계를 이용하여 분석하고 예측 가능
  - 대표적인 사례 : 구글에서 개발한 "사망 예측 시스템"

- Vector – Cross Product
  - 외적 : 두 벡터에 동시에 수직인 벡터를 구하는 방법
  - 내적 결과값 : 스칼라 vs 외적 결과값 : 벡터

$$x = (x_1, x_2, x_3)$$
$$y = (y_1, y_2, y_3)$$

$$x \times y = \left(\begin{vmatrix} x_2 & x_3 \\ y_2 & y_3 \end{vmatrix}, \begin{vmatrix} x_1 & x_3 \\ y_1 & y_3 \end{vmatrix}, \begin{vmatrix} x_1 & x_2 \\ y_1 & y_2 \end{vmatrix} \right)$$

- Vector – Cross Product

numpy.cross

```
numpy.cross(a, b, axisa=- 1, axisb=- 1, axisc=- 1, axis=None)
```

Return the cross product of two (arrays of) vectors.

Linear Algebra

- Vector – Cross Product

```
import numpy as np
import matplotlib.pyplot as plt

fig = plt.figure(figsize=(6,6))
ax = fig.add_subplot(1, 1, 1, projection='3d')

ax.set_xlabel("x", size = 12)
ax.set_ylabel("y", size = 12)
ax.set_zlabel("z", size = 12)

ax.set_xlim([-10.0, 10.0])
ax.set_ylim([-10.0, 10.0])
ax.set_zlim([-10.0, 10.0])

ax.set_xticks([-10.0, -5.0, 0.0, 5.0, 10.0])
ax.set_yticks([-10.0, -5.0, 0.0, 5.0, 10.0])
ax.set_zticks([-10.0, -5.0, 0.0, 5.0, 10.0])

start_point = np.array([0, 0, 0])
```

Linear Algebra

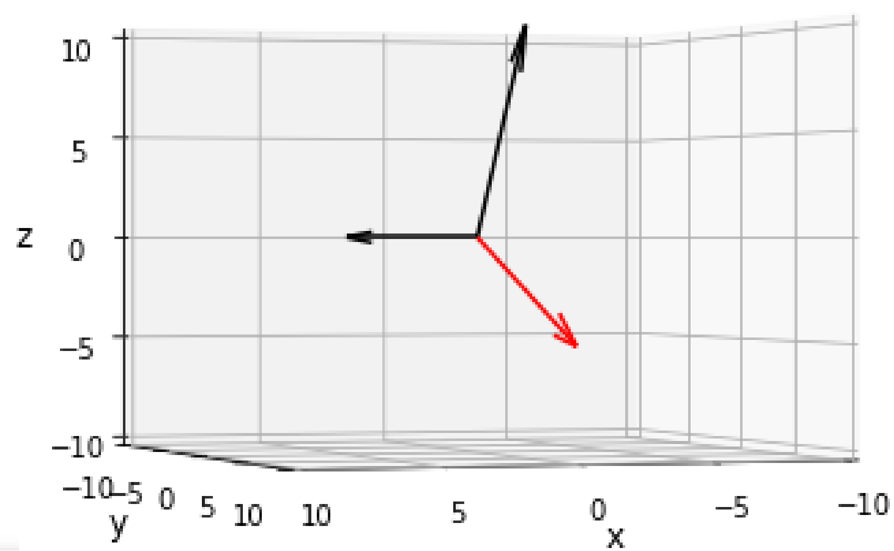
- Vector – Cross Product

```
a = np.array([0, 5, 10])
b = np.array([5, 0, 0])
c = np.cross(a, b)

ax.quiver(0, 0, 0, a[0], a[1], a[2], color='black', arrow_length_ratio=0.2)
ax.quiver(0, 0, 0, b[0], b[1], b[2], color='black', arrow_length_ratio=0.2)
ax.quiver(0, 0, 0, c[0]/5, c[1]/5, c[2]/5, color='red', arrow_length_ratio=0.2)

ax.view_init(elev=0, azim=70)

ax.set_axisbelow(True)

print(c)
print(c/5)
plt.show()
```

Linear Algebra

- Vector – Cross Product

```
[  0  50 -25]
[ 0. 10. -5.]
```



Linear Algebra

- Matrix
  - 연립 일차방정식 3x+4y=0, x+3y=0의 해를 구해보세요!

$$\begin{cases} 3x + 4y = 0 \\ x + 3y = 0 \end{cases} \qquad \begin{bmatrix} 3 & 4 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

- Matrix – Solution without Matrix

$$\begin{cases} 3x + 4y = 0 \\ x + 3y = 0 \end{cases} = \begin{cases} x + \dfrac{4}{3}y = 0 \\ x + 3y = 0 \end{cases}$$

$$\frac{5}{3}y = 0,\ y = 0 \qquad \therefore x = 0,\ y = 0$$

- Matrix – Solution with Matrix (1)

$$\begin{bmatrix} 3 & 4 & 0 \\ 1 & 3 & 0 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 3 & 0 \\ 3 & 4 & 0 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 3 & 0 \\ 0 & -5 & 0 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 3 & 0 \\ 0 & 1 & 0 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

$$\therefore x = 0, y = 0$$

- Matrix – Solution with Matrix (2)

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 3 & 4 \\ 1 & 3 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 3 & 4 \\ 1 & 3 \end{bmatrix} \xrightarrow[E_{12}(-3)]{E_{12}} \begin{bmatrix} 1 & 3 \\ 0 & -5 \end{bmatrix} \xrightarrow[E_{21}(-3)]{E_2(-\frac{1}{5})} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

- Matrix – Solution with Matrix (2)

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \xrightarrow[E_{12}(-3)]{E_{12}} \begin{bmatrix} 0 & 1 \\ 1 & -3 \end{bmatrix} \xrightarrow[E_{21}(-3)]{E_2(-\frac{1}{5})} \begin{bmatrix} \frac{3}{5} & \frac{-4}{5} \\ \frac{-1}{5} & \frac{3}{5} \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \frac{3}{5} & \frac{-4}{5} \\ \frac{-1}{5} & \frac{3}{5} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\therefore x = 0, y = 0$$

Linear Algebra

- Matrix – Solution without Matrix

```
# List를 활용해 연립일차방정식 풀기

from fractions import Fraction

def calc_value(p, q):
    if(a[0] / b[0]) == (a[1] / b[1]) == (a[2] / b[2]) :
        print('무수히 많은 해')
        return None, None

    elif (a[0] / b[0]) == (a[1] / b[1]) == (a[2] / b[2]):
        print('해가 존재하지 않음')
        return None, None
```

Linear Algebra

- Matrix – Solution without Matrix

```
else:
    if a[0] == 0:
        y = a[2] / a[1]
        x = (b[2] - b[1] * y) / b[0]

    elif b[0] == 0:
        y = b[2] / b[1]
        x = (a[2] - a[1] * y) / a[0]

    elif a[1] == 0:
        x = a[2] / a[0]
        y = Fraction(b[2] - b[0] * x, b[1])

    elif b[1] == 0:
        x = b[2] / b[0]
        y = a[2] - a[0] * x/a[1]

    else :
        a[0] = b[0]
        a[1] = a[1] * (b[0] / a[0])
        a[2] = a[2] * (b[0] / a[0])

        y = (a[2] - b[2])/(a[1] - b[1])
        x = (b[2] - (b[1] * y)) / b[0]

    return x, y
```

```
a = [3, 4, 0]
b = [1, 3, 0]

x, y = calc_value(a, b)
print(x, y)
```

```
0.0 0.0
```

Linear Algebra

- Matrix – Solution with Matrix
  - numpy.linalg.solve()

`linalg.solve(a, b)` # [source]

Solve a linear matrix equation, or system of linear scalar equations.

Computes the "exact" solution, *x*, of the well-determined, i.e., full rank, linear matrix equation *ax* = *b*.

Parameters: **a** : *(..., M, M) array_like*

Coefficient matrix.

b : *{(..., M,), (..., M, K)}, array_like*

Ordinate or "dependent variable" values.

Returns: **x** : *{(..., M,), (..., M, K)} ndarray*

Solution to the system a x = b. Returned shape is identical to *b*.

Linear Algebra

- Matrix – Solution with Matrix
  - numpy.linalg.inv()

`linalg.inv(a)` [source]

Compute the (multiplicative) inverse of a matrix.

Given a square matrix *a*, return the matrix *ainv* satisfying `dot(a, ainv) = dot(ainv, a) = eye(a.shape[0])`.

Parameters: a : *(..., M, M) array_like*

Matrix to be inverted.

Returns: ainv : *(..., M, M) ndarray or matrix*

(Multiplicative) inverse of the matrix *a*.

Linear Algebra

Matrix – Solution with Matrix

- numpy.dot()

```
numpy.dot(a, b, out=None)
```

Parameters: **a** : *array_like*

First argument.

b : *array_like*

Second argument.

out : *ndarray, optional*

Output argument. This must have the exact kind that would be returned if it was not used. In particular, it must have the right type, must be C-contiguous, and its dtype must be the dtype that would be returned for *dot(a,b)*. This is a performance feature. Therefore, if these conditions are not met, an exception is raised, instead of attempting to be flexible.

Returns: **output** : *ndarray*

Returns the dot product of *a* and *b*. If *a* and *b* are both scalars or both 1-D arrays then a scalar is returned; otherwise an array is returned. If *out* is given, then it is returned.

Linear Algebra

- Matrix – Solution with Matrix

```
import numpy as np

#행렬을 활용해 연립일차방정식 풀기
coef = np.array([[1,2], [2, 3]])
cons = np.array([[0], [0]])

value = np.linalg.solve(coef, cons)
print(value)
```

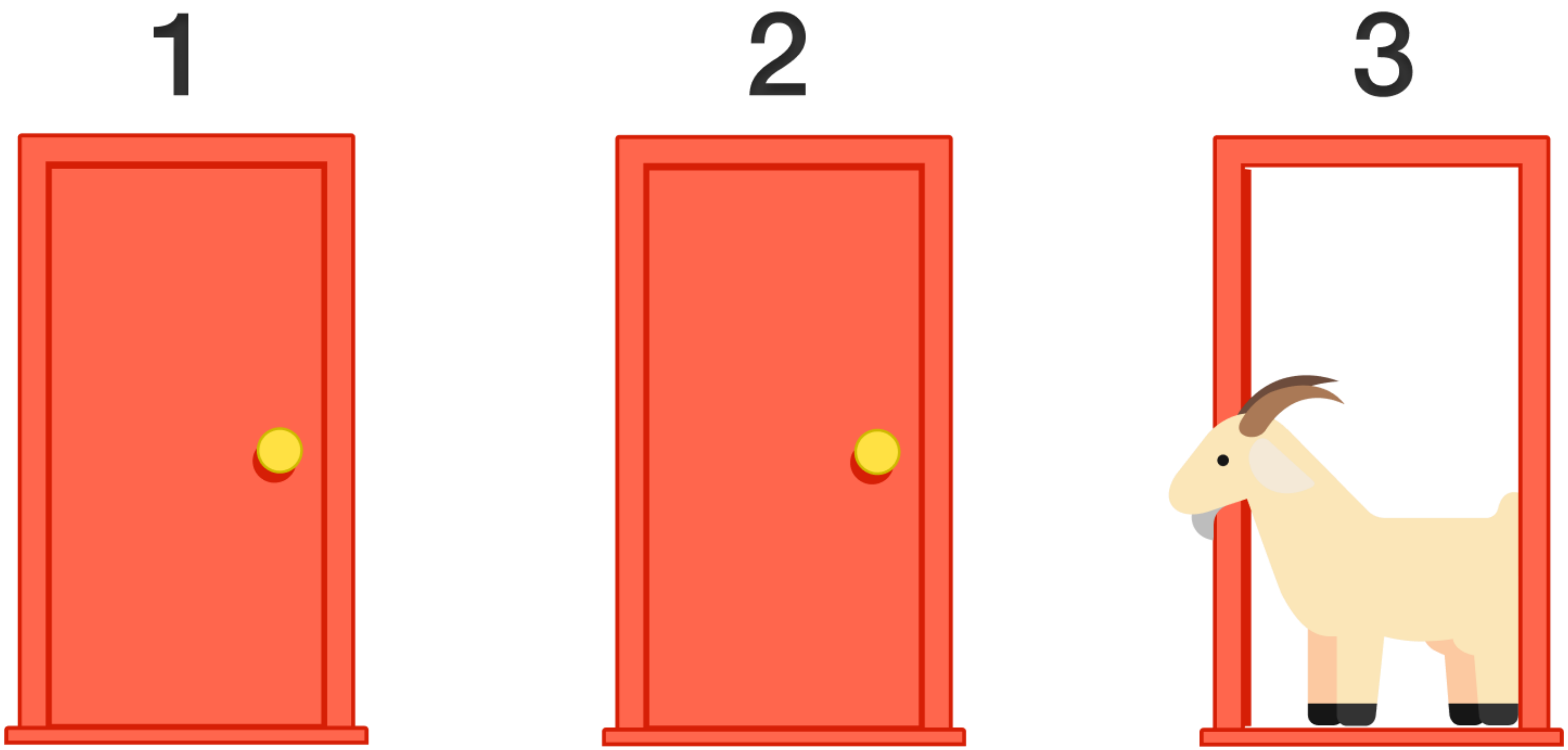
```
[[0.]
 [0.]]
```

```
# 역행렬을 활용해 연립일차방정식 풀기
value2 = np.linalg.inv(coef)
value2_dot = np.dot(value2, cons)
print(value2_dot)
```

```
[0. 0.]
```

Probability

- Monty Hall Problem



Monty Hall Problem

Suppose you're on a game show, and you're given the choice of three doors. Behind one door is a car, behind the others, goats. You pick a door, say #1, and the host, who knows what's behind the doors, opens another door, say #3, which has a goat. He says to you, "Do you want to pick door # 2?" Is it to your advantage to switch your choice of doors?

☞ 게임쇼에 나가서 세 개의 문을 선택했다고 가정해보세요. 한 문 뒤에는 차, 다른 문 뒤에는 염소들이 있습니다. 당신이 1번 문을 고르면, 문 뒤에 뭐가 있는지 아는 호스트는 염소가 있는 3번 문을 엽니다. 그는 당신에게 "2번 문을 고르고 싶으세요?"라고 말합니다. 당신이 선택한 문을 바꾸는 것이 당신에게 유리합니까?

Probability

- Monty Hall Problem
  - Premise
    - 사회자는 염소가 뒤에 있는 문을 임의로 선택합니다.
    - 사회자는 어느 문 뒤에 자동차가 존재하는지 알고 있습니다.
  - Rule
    - 문 3개 중 하나의 문 뒤에는 자동차가 있고, 나머지 문 뒤에는 염소가 있습니다. 참가자는 3개의 문 중 하나의 문을 골라 상품을 갖습니다.
    - 참가자가 문을 선택하면 선택하지 않은 문 중 염소가 뒤에 있는 문을 열어 염소를 보여줍니다.
    - 그리고 참가자가 처음으로 선택한 문 대신 다른 문으로 바꿀 수 있는 기회를 줍니다.

Probability

- Monty Hall Problem
  - Case 1) Not change the first selected door
    - 사건 X : 선택한 문을 제외한 두 문 중 호스트가 하나의 문을 여는 사건
    - 사건 X가 일어날 확률 : P(X) = 1/2

    - 사건 Y : 자동차가 처음으로 선택한 문 뒤에 있는 사건
    - 사건 Y가 일어날 확률 : P(Y) = 1/3

$$P(Y|X) = \frac{P(Y)P(X|Y)}{P(X)} = \frac{\frac{1}{3} \times \frac{1}{2}}{\frac{1}{2}} = \frac{1}{3}$$

Probability

- Monty Hall Problem
  - Case 2) Change the first selected door
    - 사건 X : 선택한 문을 제외한 두 문 중 호스트가 하나의 문을 여는 사건
    - 사건 X가 일어날 확률 : P(X) = 1/2

    - 사건 Y : 자동차가 처음으로 선택한 문 뒤에 있는 사건
    - 사건 Y가 일어날 확률 : P(Y) = 1/3

    - 사건 Z : 자동차가 처음 선택한 문과 호스트가 연 문을 제외한 문 뒤에 있을 때 호스트가 문을 여는 사건
    - 사건 Z가 일어날 확률 : P(Z) = P(X|Y) = 1

Probability

- Monty Hall Problem
  - Case 2) Change the first selected door

$$P(Y|X) = \frac{P(Y)P(X|Y)}{P(X)} = \frac{\frac{1}{3}\times 1}{\frac{1}{2}} = \frac{2}{3}$$

Probability

- Monty Hall Problem

```python
import numpy as np

def monty_hall_process(num) :

    car_cnt = 0
    goat_cnt = 0

    for a in range(num) :

        create_doors = [0, 0, 0]

        car_spot = np.random.randint(0,3)

        create_doors[car_spot] += 1

        first_choice = np.random.randint(0,3)

        if create_doors[first_choice] == 0 :
            create_doors[first_choice] += -1

        else :
            create_doors[first_choice] += 1

        create_doors.remove(0)

        if -1 in create_doors :
            car_cnt += 1

        else :
            goat_cnt += 1

    result = [car_cnt, goat_cnt, car_cnt/num]

    return result
```

Probability

- Monty Hall Problem

```python
a = f'성공한 횟수 : {result_a}번, 실패한 횟수 : {result_b}번 => 최종적으로 차를 고를 확률 : {result_c :.5f}%입니다!'
print(a)
```

```
성공한 횟수 : 66664367번, 실패한 횟수 : 33335633번 => 최종적으로 차를 고를 확률 : 66.66437%입니다!
```

Statistics

- Matplotlib & Seaborn Collaboration
  - Matplotlib과 Seaborn을 함께 사용 가능
    - 이 점을 이용한 각 종에 따른 펭귄의 무게와 물갈퀴의 길이 분석
      - Matplotlib과 Seaborn을 함께 사용할 때 발생하는 오류 발생 및 해결

Statistics

- Matplotlib & Seaborn Collaboration

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

penguin_data = sns.load_dataset('penguins')

fig_species = penguin_data["species"].unique()

fig, axes = plt.subplots(ncols=2, figsize=(10,5))
```

Statistics

- Matplotlib & Seaborn Collaboration

```
# 하나의 fig에 matplotlib과 seaborn을 겹쳐 그리기 위한 for문을 하나에 넣어 돌리면 legend() 범주 표시 오류
for a, b in enumerate(fig_species) :

    axes[0].scatter(penguin_data["body_mass_g"].loc[penguin_data["species"]==b],
                    penguin_data["flipper_length_mm"].loc[penguin_data["species"]==b],
                    c=f"C{a}", label=b, alpha=0.4 )

    sns.regplot(x = "body_mass_g", y = "flipper_length_mm", data=penguin_data.loc[penguin_data["species"]==b], scatter=False, ax=axes[0])
```

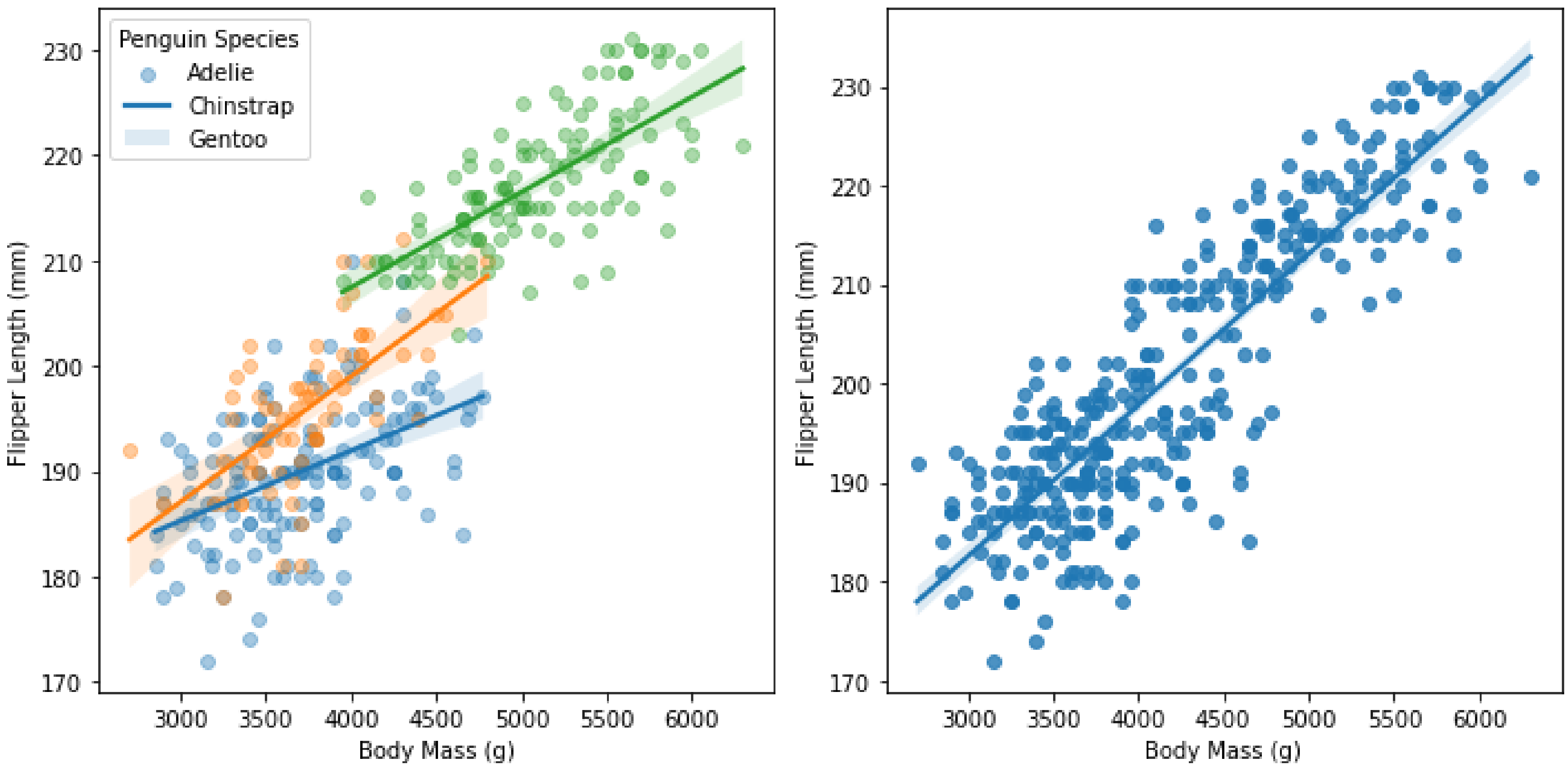
Statistics

▪ Matplotlib & Seaborn Collaboration

```
axes[0].legend(fig_species, title = "Penguin Species")
axes[0].set_xlabel("Body Mass (g)")
axes[0].set_ylabel("Flipper Length (mm)")

sns.regplot(x = "body_mass_g", y = "flipper_length_mm", ax=axes[1], data=penguin_data)
axes[1].set_xlabel("Body Mass (g)")
axes[1].set_ylabel("Flipper Length (mm)")

fig.tight_layout()
```

Statistics

- Matplotlib & Seaborn Collaboration



Statistics

- Matplotlib & Seaborn Collaboration

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

penguin_data = sns.load_dataset('penguins')

fig_species = penguin_data["species"].unique()

fig, axes = plt.subplots(ncols=2, figsize=(10,5))
```

Statistics

▪ Matplotlib & Seaborn Collaboration

```
# 하나의 fig에 matplotlib과 seaborn을 겹쳐 그리기 위한 for문을 따로 돌리면 legend() 범주 표시 오류 해결
for a, b in enumerate(fig_species) :

    axes[0].scatter(penguin_data["body_mass_g"].loc[penguin_data["species"]==b],
                    penguin_data["flipper_length_mm"].loc[penguin_data["species"]==b],
                    c=f"C{a}", label=b, alpha=0.4 )

for a, b in enumerate(fig_species) :

    sns.regplot(x = "body_mass_g", y = "flipper_length_mm", data=penguin_data.loc[penguin_data["species"]==b], scatter=False, ax=axes[0])
```

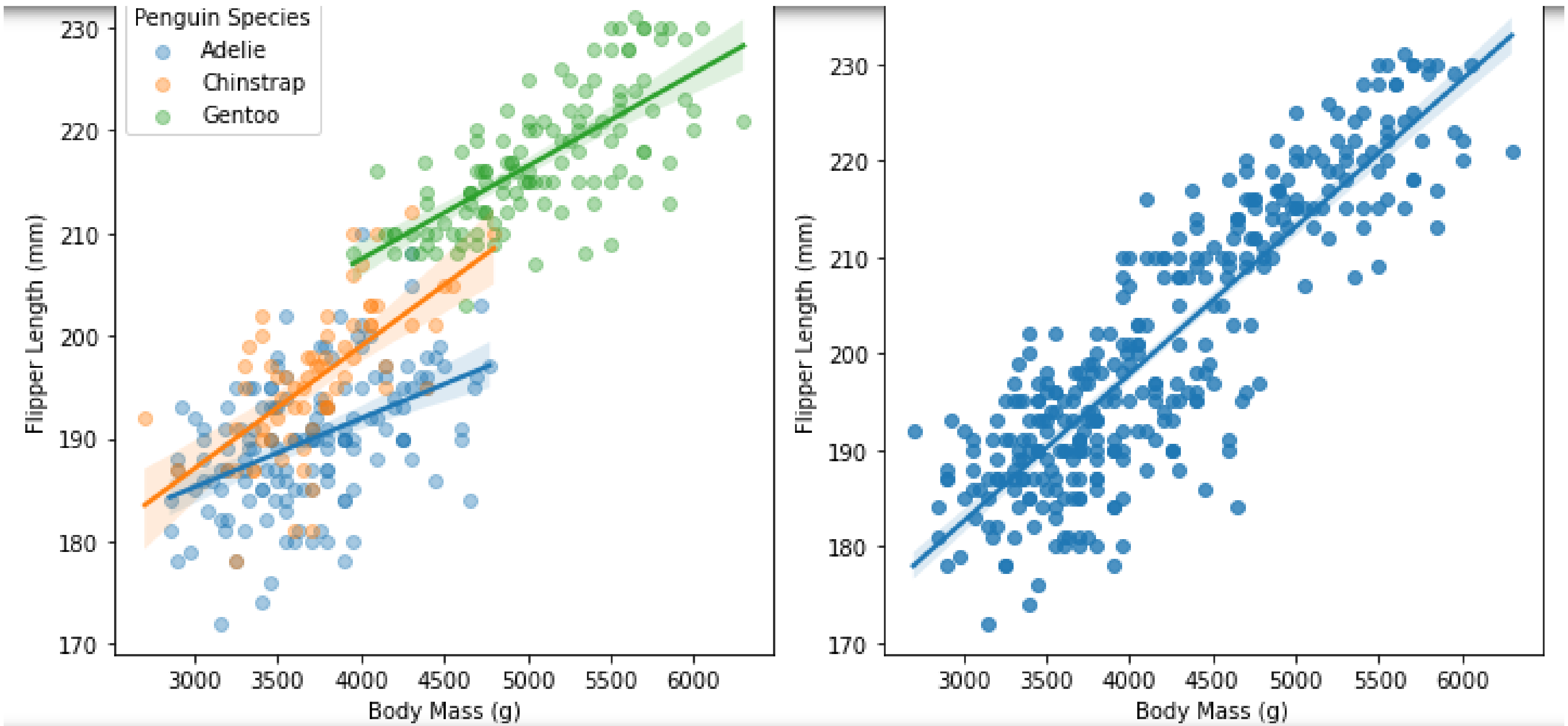
Statistics

- Matplotlib & Seaborn Collaboration

```
axes[0].legend(fig_species, title = "Penguin Species")
axes[0].set_xlabel("Body Mass (g)")
axes[0].set_ylabel("Flipper Length (mm)")

sns.regplot(x = "body_mass_g", y = "flipper_length_mm", ax=axes[1], data=penguin_data)
axes[1].set_xlabel("Body Mass (g)")
axes[1].set_ylabel("Flipper Length (mm)")

fig.tight_layout()
```

Statistics

- Matplotlib & Seaborn Collaboration



Conclusion & Realization

- 선형대수와 확률과 통계 뿐만 아니라 모든 계산은 코드 구현 및 계산 가능하다는 사실 다시 한 번 더 깨닫는 계기
- 학습한 내용 정리의 중요성
  - 발표 내용을 보다 더 흥미롭고 이해가 쉽게 설명할 좋은 아이디어 고찰
  - 꽤 오랜 시간 전에 블로그에 정리해둔 학습 내용에서 아이디어 착안
  - 정리해둔 학습 내용에 없던 연장 학습 실행

Thank you for your Attention!

경상국립대학교
Gyeongsang National University

Improving lives through learning

IDEALAB