Toan Ly

Lecture 9 Exercise

1)

a)

```julia
using RDatasets;

data = dataset("car", "Davis");
first(data, 5)
```

✓ 0.1s

5×5 DataFrame

Row	Sex	Weight	Height	RepWt	RepHt
	Cat…	Int32	Int32	Int32?	Int32?
1	M	77	182	77	180
2	F	58	161	51	159
3	F	53	161	54	158
4	M	68	177	70	175
5	F	59	157	59	155

b)

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$\hat{\sigma} = \frac{1}{n}\sum_{i=1}^{n} (x_i - \mu)^2$$

```julia
data = data[:, [:Weight, :Height]];
data = Matrix{Float64}(data);
```

✓ 0.0s

```julia
mu_weight = mean(data[:, 1]);
mu_height = mean(data[:, 2]);
var_weight = mean((data[:, 1] .- mu_weight).^2);
var_height = mean((data[:, 2] .- mu_height).^2);

println("Weight: mean = $mu_weight, variance = $var_weight");
println("Height: mean = $mu_height, variance = $var_height");
```

✓ 0.0s

```
Weight: mean = 65.8, variance = 226.71999999999997
Height: mean = 170.02, variance = 143.46959999999996
```

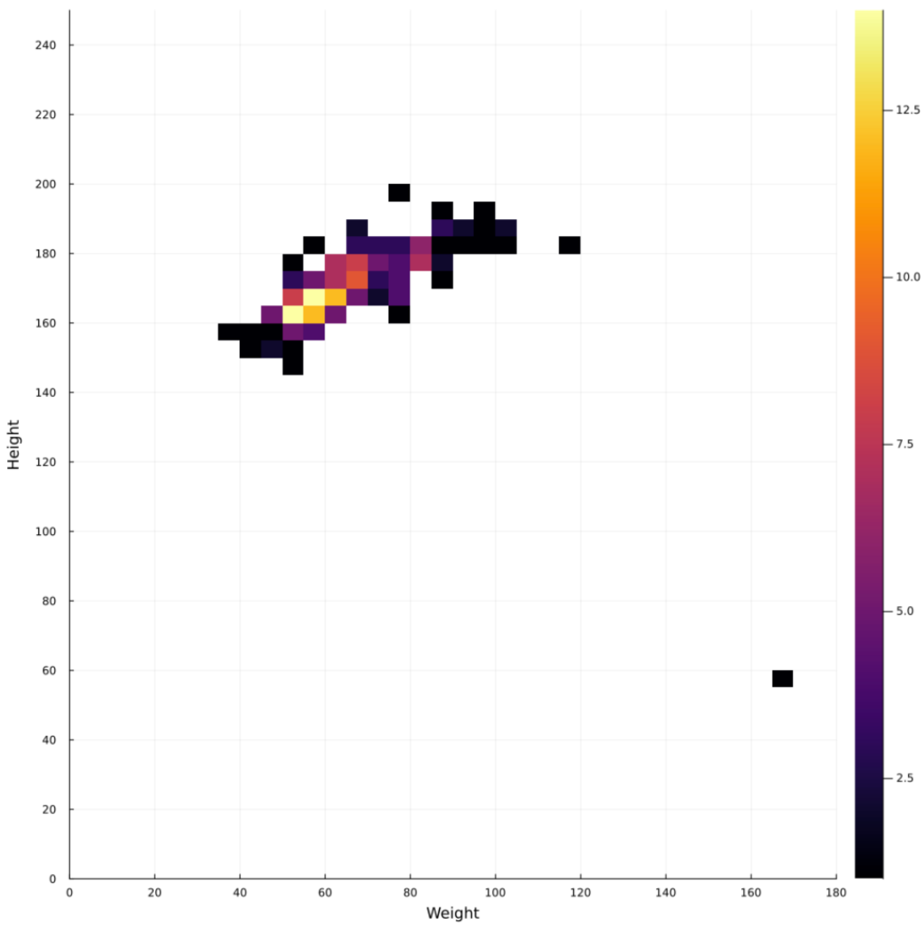
c)

```julia
using Plots;
```

✓ 4.1s

```julia
gr(size = (2000, 2000))
histogram2d(
    data1.Weight, data1.Height,
    nbins=40,
    xaxis=("Weight", (0, 180), 0:20:180),
    yaxis=("Height", (0, 250), 0:20:250),
)
```

✓ 0.8s



Observation: The shape is elliptical with slight up-right direction, meaning the height tends to increase when the weight increases, or taller people tends to be heavier. However, this positive trend is not perfect, most of the correlation has black color (<3.0), meaning the weight and height are not strongly dependent on each other. Additionally, there are a few outliers.

d)

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$\hat{\Sigma} = \frac{1}{n}\sum_{i=1}^{n} (x_i - \hat{\mu})(x_i - \hat{\mu})^T$$

```julia
mu = mean(data, dims=1);
cov = (data .- mu)' * (data .- mu) / size(data, 1);

println("Mean = $mu\n");
print("Covariance:");
cov
```

✓ 0.0s

```
Mean = [65.8 170.02]

Covariance:

2×2 Matrix{Float64}:
 226.72    34.204
  34.204  143.47
```

e)

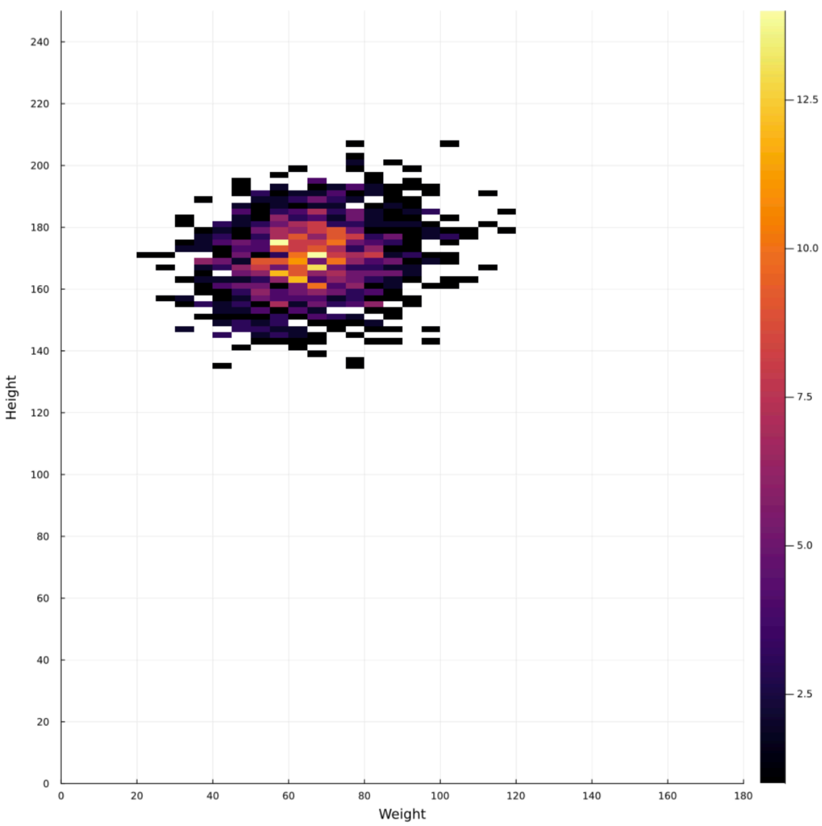
```julia
d = MvNormal(mu[:], cov);
sample = rand(d, 1000)
```

✓ 0.0s

```
2×1000 Matrix{Float64}:
  40.3563   67.3039   61.9032   65.5276  …   61.4591   63.9006   78.3129
 162.153   195.755   171.29    172.942      156.989   162.212   193.433
```

```julia
myplot = histogram2d(
    sample[1, :], sample[2, :],
    nbins=40,
    xaxis=("Weight", (0, 180), 0:20:180),
    yaxis=("Height", (0, 250), 0:20:250),
)
```

✓ 0.1s



Above histogram has smoother shape compared to plot in (c) with the mean around (65.8, 170.02). Overall, the above histogram is able to generate the mean and variance pretty well, it's a smoother version of the real data. However, it tends to assume the data distribution has smooth shape, which smooths away outliers

f) Since covariance between weight and height is 34.204 ≠ 0, they are somewhat dependent.

2)

a)

```
iris = dataset("datasets", "iris");
groups = groupby(iris, :Species);
new_iris = vcat(groups[2], groups[3]) # versicolor, virginica
```

✓ 2.1s

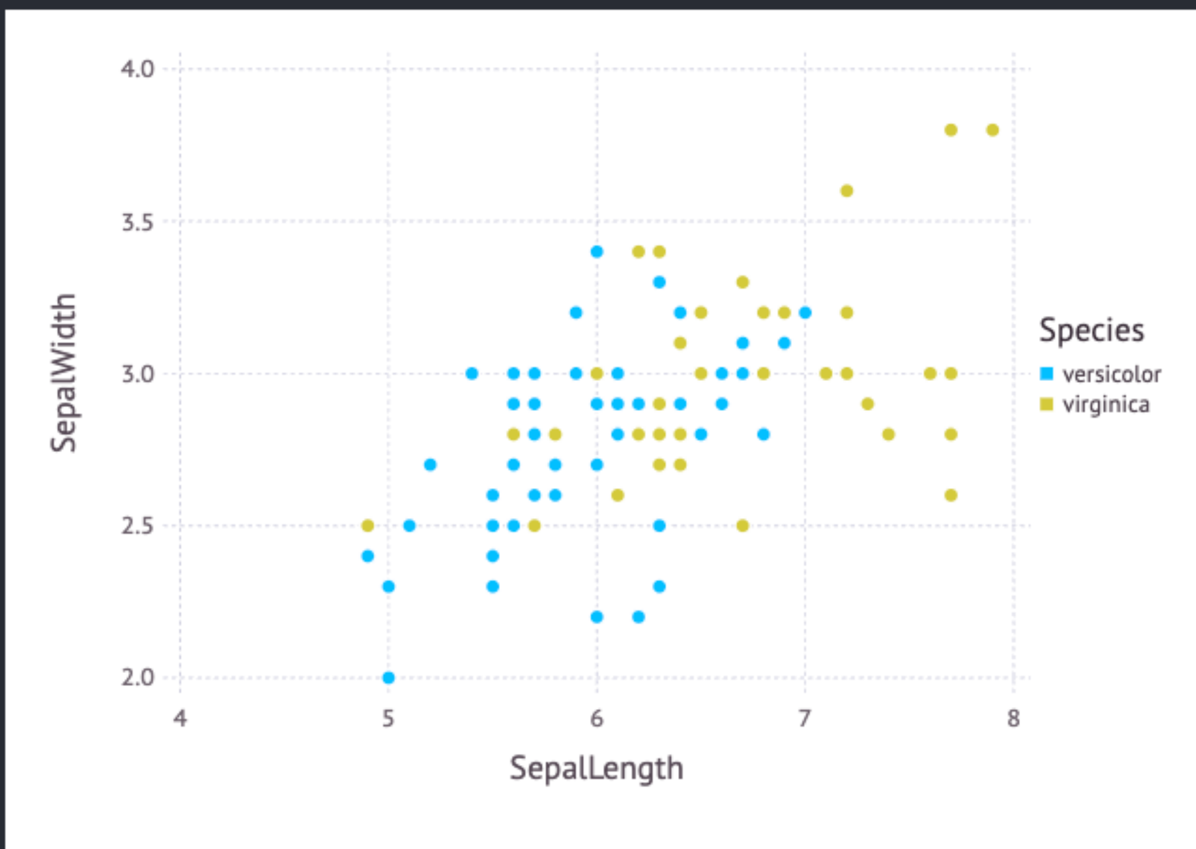
100×5 DataFrame

Row	SepalLength	SepalWidth	PetalLength	PetalWidth	Species
	Float64	Float64	Float64	Float64	Cat...
1	7.0	3.2	4.7	1.4	versicolor
2	6.4	3.2	4.5	1.5	versicolor
3	6.9	3.1	4.9	1.5	versicolor
4	5.5	2.3	4.0	1.3	versicolor
5	6.5	2.8	4.6	1.5	versicolor
6	5.7	2.8	4.5	1.3	versicolor
7	6.3	3.3	4.7	1.6	versicolor
8	4.9	2.4	3.3	1.0	versicolor
9	6.6	2.9	4.6	1.3	versicolor
10	5.2	2.7	3.9	1.4	versicolor
11	5.0	2.0	3.5	1.0	versicolor
12	5.9	3.0	4.2	1.5	versicolor
13	6.0	2.2	4.0	1.0	versicolor
⋮	⋮	⋮	⋮	⋮	⋮
89	6.0	3.0	4.8	1.8	virginica
90	6.9	3.1	5.4	2.1	virginica
91	6.7	3.1	5.6	2.4	virginica
92	6.9	3.1	5.1	2.3	virginica
93	5.8	2.7	5.1	1.9	virginica
94	6.8	3.2	5.9	2.3	virginica
95	6.7	3.3	5.7	2.5	virginica
96	6.7	3.0	5.2	2.3	virginica
97	6.3	2.5	5.0	1.9	virginica
98	6.5	3.0	5.2	2.0	virginica
99	6.2	3.4	5.4	2.3	virginica
100	5.9	3.0	5.1	1.8	virginica

b)

```
Gadfly.plot(
    new_iris,
    x=:SepalLength, y=:SepalWidth,
    color=:Species,
    Geom.point,
    Theme(background_color="white")
)
```

✓ 0.2s



c) Let y = 1 be virginica, y = 0 be versicolor.

Let $z = \beta_0 + \beta_1 \cdot PetalLength + \beta_2 \cdot PetalWidth$

$$p(y=1|x) = \sigma(z) = \frac{1}{1+e^{-z}} = \frac{1}{1+e^{-(\beta_0+\beta_1 \cdot PetalLength+\beta_2 \cdot PetalWidth)}}$$

$$p(y=0|x) = 1-\sigma(z) = \frac{e^{-z}}{1+e^{-z}} = \frac{e^{-(\beta_0+\beta_1 \cdot PetalLength+\beta_2 \cdot PetalWidth)}}{1+e^{-(\beta_0+\beta_1 \cdot PetalLength+\beta_2 \cdot PetalWidth)}}$$

d)

$$L(\beta; y|x) = p(y|x) = \prod_i f(y_i|x_i;\beta)$$

Let $p(y_i = 1|x_i;\beta)$ be p_i (success in a Bernoulli trial)

$p_i = \sigma(\beta_0 + \beta_1 \cdot PetalLength + \beta_2 \cdot PetalWidth)$

$$\ell = \sum_i \log\left(f(y_i|x_i;\beta\right) = \sum_{i;y_i=1} \log(p_i) + \sum_{i;y_i=0} \log\left(1-p_i\right)$$

e)

$$\frac{\partial \ell}{\partial \beta_j} = \sum_{i;y_i=1} \frac{\partial}{\partial \beta_j} \log(p_i) + \sum_{i;y_i=0} \frac{\partial}{\partial \beta_j} \log(1-p_i)$$

$= \sum_{i;y_i=1} \frac{1}{\mathrm{p_i}} \frac{\partial p_i}{\partial \beta_j} + \sum_{i;y_i=0} \frac{-1}{1-\mathrm{p_i}} \frac{\partial p_i}{\partial \beta_j}$ (1)

$\frac{\partial p_i}{\partial \beta_j} = p_i(1-p_i)x_j$ (2)

Plug (2) into (1),

$$\frac{\partial \ell}{\partial \beta_j} = \sum_{i;y_i=1} (1-p_i)x_{ij} + \sum_{i;y_i=0} (0-p_i)\, x_{ij} = \sum_i (y_i - p_i)x_{ij}$$

f) From (e), $\frac{\partial \ell}{\partial \beta_j} = \sum_i (y_i - p_i)x_{ij}$

$\frac{\partial^2 \ell}{\partial \beta_j\, \beta_k} = \frac{\partial}{\partial \beta_k} \frac{\partial \ell}{\partial \beta_j} = \frac{\partial}{\partial \beta_k} \left(\sum_i (y_i - p_i)x_{ij}\right)$

$$= \sum_i -\frac{\partial p_i}{\partial \beta_k} x_{ij} \; = -\sum_i p_i(1-p_i)x_{ik}x_{ij}$$

Additionally,

$\frac{\partial^2 \ell}{\partial \beta_j\, \beta_j} = \frac{\partial^2 \ell}{\partial \beta_j^2} = -\sum_i p_i(1-p_i)x_{ij}^2$

g)

Step 1: Initialize $\beta = [\beta_0, \beta_1, \beta_2]$

Step 2: maxIter = 10000

Step 3: **for** i = 2 : maxIter

Step 4: $\beta \leftarrow \beta - \frac{\partial^2 \ell}{\partial \beta^2}^{-1} \frac{\partial \ell}{\partial \beta}$

Step 5: **if** $|\ell_i - \ell_{i-1}| < \epsilon, terminate$

Step 6: **end** for

h)

```
function compute_p(x, beta)
    return 1.0 ./ (1.0 .+ exp.(-(x * beta)))
end

function compute_l(x, y, beta)
    p = compute_p(x, beta)
    prob = y .* log.(p) .+ (1 .- y) .* log.(1 .- p)
    l = sum(prob[.!isnan.(prob)])
    return l;
end
```

```
function newtons(x, y)
    beta = zeros(3); # [beta0, beta1, beta2]
    l = compute_l(x, y, beta);
    for i in 1:50
        p = compute_p(x, beta);
        d1 = compute_first_derivative(x, y, p);
        d2 = compute_second_derivative(x, p);
        beta_new = beta - inv(d2) * d1;
        l_new = compute_l(x, y, beta_new);

        if abs(l_new - l) < 1e-6
            println("Converged after $i iterations")
            break;
        end

        l = l_new;
        beta = beta_new;
    end
    return beta;
end
```

```
function compute_first_derivative(x, y, p)
    d1 = zeros(3);
    d1[1] = sum((y .- p) .* x[:, 1]); # beta_0
    d1[2] = sum((y .- p) .* x[:, 2]); # beta_1
    d1[3] = sum((y .- p) .* x[:, 3]); # beta_2
    return d1;
end

function compute_second_derivative(x, p)
    d2 = zeros(3, 3);

    d2[1, 1] = -sum(p .* (1 .- p)); # beta_0
    d2[1, 2] = -sum(p .* (1 .- p) .* x[:, 2]); # beta_0, beta_1
    d2[1, 3] = -sum(p .* (1 .- p) .* x[:, 3]); # beta_0, beta_2

    d2[2, 1] = d2[1, 2]; # beta_1, beta_0
    d2[2, 2] = -sum(p .* (1 .- p) .* (x[:, 2].^2)); # beta_1
    d2[2, 3] = -sum(p .* (1 .- p) .* x[:, 2] .* x[:, 3]); # beta_1, beta_2

    d2[3, 1] = d2[1, 3]; # beta_2, beta_0
    d2[3, 2] = d2[2, 3]; # beta_2, beta_1
    d2[3, 3] = -sum(p .* (1 .- p) .* (x[:, 3].^2)); # beta_2
    return d2;
end
```

```
x1 = new_iris[:, :PetalLength];
x2 = new_iris[:, :PetalWidth];
X = hcat(ones(size(new_iris, 1)), x1, x2);
y = new_iris[:, :Species] .== "virginica";
beta = newtons(X, y);
println("Estimated beta: $beta");
```

✓ 0.2s

```
Converged after 9 iterations
Estimated beta: [-45.27232550690931, 5.754529731169695, 10.446696552949449]
```

```julia
println("beta_0 = $(beta[1])");
println("beta_1 = $(beta[2])");
println("beta_2 = $(beta[3])");
```

✓ 0.0s

```
beta_0 = -45.27232550690931
beta_1 = 5.754529731169695
beta_2 = 10.446696552949449
```