Primer of Biostatistics
The 1st Lecture

Hiroyoshi Iwata
aiwata@mail.ecc.u-tokyo.ac.jp

Recently, in the fields of agriculture and life sciences, various types of data are being collected and accumulated in large quantities. It is necessary to analyze the data using methods appropriate to the purpose of the research and the nature of the data, in order to ensure that we do not miss the “undiscovered knowledge” harbored in the data.

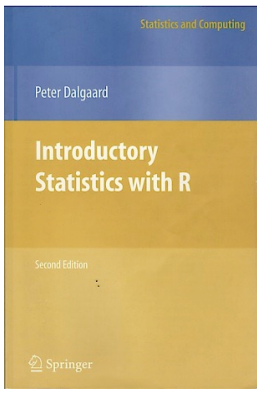
There are various methods for statistical analysis. In order to understand the features of the methods and the principles of analysis and also to interpret the analysis results properly, you should learn them accordingly. Also, in order to make your learning more effective, it is essential to have the experience of analyzing actual data. It is often the case that you can clearly understand what you have learned in lectures and textbooks when you analyze your data by yourself.

This course is intended to provide you with a "first step" to analyze their own data and improve their statistical analysis skills. Specifically, we will focus on practical data analysis methods using R, focusing on some of the statistical methods that may be required in your future research. The goal of this lecture is to acquire the skills to use it for data analysis of general-purpose statistical analysis methods, e.g., regression analysis, analysis of variance, and principal component analysis. Furthermore, the goal is to build a foothold to perform more advanced data analysis. Although it is a short course with four lectures, I will provide you this course so that you can be interested in the joy and skills of statistical analysis.

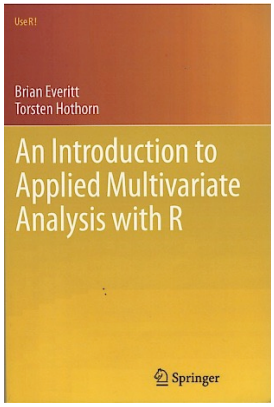
<R>

R is free software for statistical analysis. (To put it a little correctly, R is the name of a computer language. R installed on a PC as software is an "environment" for using R language). R has many functions, and its usages range from statistical analysis to pre-processing of data, overview of data, and creation of graphs for papers. In addition, various analysis can be easily performed by installing an extension program distributed as a “package”. Newly developed statistical methods will be available quickly in R. Thus, the skills for using R have become useful to researchers in agronomics and life sciences.

In addition, for learning R, a large number of reference books are available. Introductory books I recommend are:



Peter Dalgaard, Introductory Statistics with R (Statistics and Computing) Second Edition, Springer, 2008, ISBN: 978-0387790534



Brian Everitt, Torsten Hothorn, An Introduction to Applied Multivariate Analysis with R (Use R!), Springer, 2011, ISBN: 978-1441996497

<Simple calculation using R>

In analysis using R, the analysis is basically progressed interactively while sequentially inputting commands (However, when you actually perform analysis, it is useful to prepare a series of commands, as an R script, prior to the analysis. Remember that it is more convenient to execute the R script, because it allows us to do partial corrections and to review the history of analysis).

Let's start with getting used to R while doing simple calculations with command input.

The easiest way to use R is to enter a simple arithmetic expression and get the answer. For example,

```
> 3 + 5 * 3
[1] 18
```

If you want to perform the next calculation based on the obtained result, assign the resulted value to some variable as follows.

```
> x <- 1 + 2
> x
[1] 3
```

The assigned value can be used for another calculation through the variable name.

```
> x + 5 * x
[1] 18
```

Various calculations can be performed using functions.

```
> abs(x)                          #  absolute values
[1] 3
> sin(x)                          #   sine
[1] 0.14112
> atan(x)                         #  arctangent
[1] 1.249046
> log(x)                          #   natural logarithm
[1] 1.098612
> log10(x)                        #    base 10 logarithm
[1] 0.4771213
```

Let's perform a bit more complicated calculation. The probability density function of the normal distribution of mean μ and the variance σ^2 (Figure 1) is

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2} \right)$$

Calculate this with R.

```
> mu <- 3
> s2 <- 2
> x <- 5
> 1 / sqrt(2 * pi * s2) * exp(- (x - mu)^2 / (2 * s2))
[1] 0.1037769
```

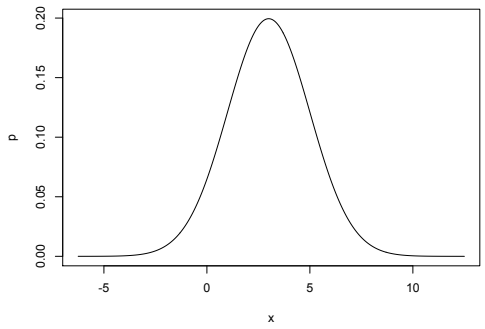


Figure 1. Normal distribution with mean 3 and

If you use the “dnorm” function to calculate the probability density of the normal distribution for confirming your calculation, you can get the same result.

```
> dnorm(x, mu, sqrt(s2))
[1] 0.1037769
```

<Calculation using vector or matrix>

A great advantage of R is that we can perform vector and matrix operations easily. Let's calculate some summary statistics using vector and matrix operations.

For example, we can easily create a vector of six elements as follows: this is the data which measured the grain length of six varieties/lines of rice in mm unit (the source of the data will be described later).

```
> length <- c(8.1, 7.7, 8.2, 9.7, 7.1, 7.3)   # mm scale
> length
[1] 8.1 7.7 8.2 9.7 7.1 7.3
```

Input grain widths of the same varieties/lines, and calculate the length-width ratio.

```
> width <- c(3.7, 3.0, 2.9, 2.4, 3.3, 2.5)
> ratio <- length / width
> ratio
[1] 2.189189 2.566667 2.827586 4.041667 2.151515 2.9200000
```

First, let's calculate the average of the length-width ratio of grains. The estimate of the population mean is

$$\sum_i^n x_i / n,$$

where x_i is the value of the ith sample and n is the number of samples.

```
> sum(ratio)                          # estimate sum
[1] 16.69662
> length(ratio)                       # length of the vector (number of samples)
[1] 6
> sum(ratio) / length(ratio)
[1] 2.782771
```

The mean can be calculated using the “mean” function.

```
> mean(ratio)
[1] 2.782771
```

Next, let's calculate the variance. An estimate of the population variance is

$$\sum_i^n (x_i - \bar{x})^2 \Big/ (n-1),$$

where $\bar{x}$ is the average calculated earlier.

```
> xbar <- mean(ratio)              # mean
> (ratio - xbar)^2                 # squares of deviation from the mean
[1] 0.352338947 0.046700930 0.002008434 1.584819189 0.398483500 0.018831895
> sum((ratio - xbar)^2)            # sum of squares of the deviation
[1] 2.403183
> sum((ratio - xbar)^2) / (length(ratio) - 1)
[1] 0.4806366
```

The variance can be calculated using the "var" function.

```
> var(ratio)
[1] 0.4806366
```

Next, let's calculate the covariance. An estimate of the covariance between bivariate x and y is

$$\sum_i^n (x_i - \bar{x})(y_i - \bar{y}) \Big/ (n-1),$$

where $\bar{x}$ and $\bar{y}$ represent the mean of each variable.

```
> xbar <- mean(length)             # mean of length
> ybar <- mean(width)              # mean of width
> sum((length - xbar) * (width - ybar)) / (length(length) - 1)  #  covariance
[1] -0.1773333
```

Note that we can also calculate the covariance using the "cov" function.

```
> cov(length, width)
[1] -0.1773333
```

Let's calculate Pearson product moment correlation coefficient (hereinafter referred to just as correlation coefficient). The correlation coefficient is

$$\frac{\sum_i^n (x_{1i} - \bar{x}_1)(x_{2i} - \bar{x}_2)}{\sqrt{\sum_i^n (x_{1i} - \bar{x}_1)^2}\sqrt{\sum_i^n (x_{2i} - \bar{x}_2)^2}}.$$

```
> s12 <- sum((length - xbar) * (width - ybar))
> s1 <- sum((length - xbar)^2)
> s2 <- sum((width - ybar)^2)
> s12 / (sqrt(s1) * sqrt(s2))
[1] -0.3901388
```

As you can see in the equation, the correlation coefficient is the covariance divided by the standard deviation of both variables. Let's actually calculate it and check the result.

```
> cov(length, width) / (sd(length) * sd(width))
[1] -0.3901388
```

The correlation coefficient is standardized by dividing it by the standard deviation of both variables. Unlike the covariance, we can understand the relationship between variables without being influenced by the scale of the measurement value. Thus, it is suitable for comparing the strength of relationships between variables measured at different scales (such as weight and length).

Note that we can also calculate the correlation coefficient using the "cor" function.

```
> cor(length, width)
[1] -0.3901388
```

Now let's calculate variance and covariance using matrix calculations. First, combine length and width to create a 6 by 2 matrix.

```
> x <- cbind(length, width)
> x
 (The result is omitted)
```

Then we use the function apply to find the average of each column.

```
> m <- apply(x, 2, mean)
> m
  length    width
8.016667 2.966667
```

Subtract the column average from each column.

```
> z <- sweep(x, 2, m)
> z
 (The result is omitted)
```

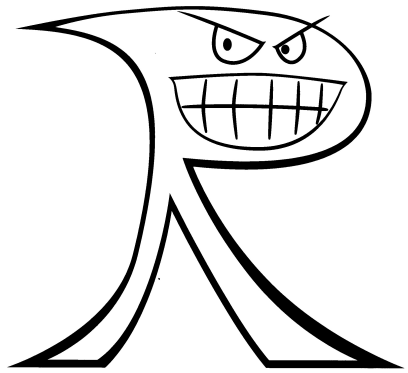
After that, we can calculate variance and covariance (variance-covariance matrix) by using the product of the matrix.

```
> t(z) %*% z / (nrow(z) - 1)
          length     width
length  0.8656667 -0.1773333
width  -0.1773333  0.2386667
```

The diagonal components are variances, and the nondiagonal components are covariances.

The variance-covariance matrix can be calculated with the “cov” function.

```
> cov(x)
          length     width
length  0.8656667 -0.1773333
width  -0.1773333  0.2386667
```



K.W.

<Import and analyze external data>

If you use R for your own research, I think that most of the time you read and analyze data organized with spreadsheet software. Here, I will explain the procedure for reading data prepared by other software into R and analyzing it. Here, we will use the data which was used in genome-wide association studies of rice genetic resources (Zhao et al. 2011; Nature Communications 2: 467). The data can be downloaded from Rice Diversity (http://www.ricediversity.org/data/).

The function "read.csv" is used to read the file saved in csv format.

```
> pheno <- read.csv("RiceDiversityPheno.csv")        # read csv file
```

To check the size of imported data or a part of the data as follows.

```
> dim(pheno)                          #  the size of the data
[1] 413  38
> head(pheno)                         #  First 6 rows of the data
 (The result is omitted)
```

The data have a separate file that describes the origin of each genetic resource. Here, we load the file and combine it with "pheno" data. First, read the file.

```
> line <- read.csv("RiceDiversityLine.csv")
> head(line)
 (The result is omitted)
```

Since NSFTV.ID in the "line" data and NSFTVID in the "pheno" data correspond to each other, the two data are combined based on the information of these columns.

```
> data <- merge(line, pheno, by.x = "NSFTV.ID", by.y = "NSFTVID")
> head(data)
 (The result is omitted)
```

<Analysis of the data>

We often want to analyze a large number of variables to look at their distributions and relationships among them. Measurement data, howover, often include missing entries with some experimental reasons. Here, we will analyze the data which we load from the csv file.

Let's calculate the length-width ratio of grains and its average in the same way as mentioned above.

```
> ratio <- data$Seed.length / data$Seed.width
                    # $ is used to specify a column in the dataframe
> mean(ratio)
[1] NA
```

We cannot calculate the average, and only get the value of “NA”. Why is that?

This is because the ratio contains missing values (represented as NA in R).

```
> ratio                                #  check the entries
 (The result is omitted)
```

In such cases, specify the option na.rm in the calculation.

```
> mean(ratio, na.rm = T)   # na.rm = T, then missing entries are ignores
[1] 2.752084
```

Find the mean for all variables in the data with the “sapply” function.

```
> sapply(data, mean, na.rm = T)
 (The result is omitted)
```

A warning message will be displayed for data that is not numeric data, and the calculated result will be NA.

Using the following command, we can calculate not only averages but also quartiles, minimum values and maximum values for numeric data, and can count up samples belonging to each class for factor data.

```
> summary(data)
 (The result is omitted)
```

Let's calculate the correlation coefficient for grain length and width.

```
> cor(data$Seed.length, data$Seed.width)
[1] NA
```

The result is NA. This is due to missing data as before.

Specify the option for dealing with the missing value and try to calculate again.

```
> cor(data$Seed.length, data$Seed.width, use = "pair")
[1] -0.2837094
```

This time, the correlation is calculated successfully.

<Data visualization>

It is very important to look at the data from different angles before actually performing statistical analysis. For example, statistics such as mean and variance mentioned above are statistics for summarization, and even with variables with similar mean and variance, the distribution of observed values may differ greatly. Therefore, looking at the data carefully is important to understand the characteristics of the data. Data visualization is also necessary when we prepare the results of the analysis for the publication of a paper. Here we will learn various data visualization techniques.

First, let's make it possible to directly call variables in the data before explaining the visualization methods.

```
> attach(data)            #  for specifying a variable in the data directly
```

With the “attach” command, for example, we can now specify Plant.height without data$~. Otherise we have to specify the variable data$Plant.height,

Let's draw a histogram first.

```
> hist(Plant.height)
```

Let's draw a stem-and-leaf plot.

```
> stem(Plant.height)
（結果は省略）
```

This is not a graph. The result is shown with text.

Let's draw a box plot.

```
> boxplot(Plant.height)
```

Next, we will draw a histogram of blast resistance (Blast.resistance).

```
> hist(Blast.resistance)
```

It looks like the distribution is visualized well, but there is a “pitfall”.

The resistance to blast disease is expressed by the level of resistance with a score of 9 (0-9). Therefore, let's summarize how many accessions are included in each of 9 levels.

```
> t <- table(Blast.resistance) #  summarize the sample number of each score
> t
Blast.resistance
 0  1  2  3  4  5  6  7  8  9
 3 77 23 34 36 24 39 36 52 61
```

You can see that the histogram we drew did not represent the whole class well.

You can draw a bar plot from the data summarized using the “table” function as described above.

```
> plot(t)         # plot of the result of table, then get the bar plot.
> plot(t, xlab = "Blast resistance scores", ylab = "Frequency")
                                    #  add title to the bar plot
```

A bar chart can also be drawn using the “barplot” function. However, it looks a bit different from the bar chart drew above.

```
> barplot(t)
```

Drawing a pie chart allows you to illustrate the percentage of each score.

```
> pie(t)
> pie(t, main = "Blast resistance")         # add title to the graph
```

From here, let's look at the relationship between two variables.

```
> plot(Plant.height, Panicle.length)   #  1st variable is x, and 2nd y
```

Fit a straight line to the data by regression analysis.

```
> abline(lm(Panicle.length ~ Plant.height))
         # lm is a function for regression analysis
         # abline is a function for draw a line on a graph
```

Overlap the rug (textile) plot. This is useful for visualizing distribution density.

```
> rug(Plant.height, side = 1)       #  side = 1 is for x axis
> rug(Panicle.length, side = 2)     #  side = 2 is for y axis
```

Let's draw a slightly more complicated picture. Draw a scatter plot and a box plot simultaneously.

```
> def.par <- par(no.readonly = T)   # save the current graphic option
> layout(matrix(c(2, 0, 1, 3), nrow = 2, byrow = T),
          widths = c(2, 1), heights = c(1, 2), respect = T)
         # prepare 2 by 2 graphic region
> plot(Plant.height, Panicle.length)        #  scatterplot of x and y
> boxplot(Plant.height, horizontal = T)     #  boxplot of x
> boxplot(Panicle.length)                   #  boxplot of y
> par(def.par)                      # reset the graphic option
```

Outliers are indicated by ○ for both Plant.height and Panicle.length.

Let's illustrate the relationship between the two variables using kernel smoothing.

```
> require("KernSmooth")
> d <- bkde2D(x, bandwidth = 4)
> plot(x)
> contour(d$x1, d$x2, d$fhat, add = T)
```

The contours represent the density of the points smoothed by the kernel.

Let's display this smoothed density in three dimensions.

```
> persp(d$x1, d$x2, d$fhat, xlab = "Plant.height",
         ylab = "Panicle.length", zlab = "density",
         theta = -30, phi = 30)
```

The data of Zhao et al. (2011) read from the file contain not only trait data but also data of the genetic background of genetic resources. Let's visualize both data together and investigate what kind of relationship between genetic background and trait.

The variable Sub.population represents differences in the genetic background of genetic resources. This is estimated using Structure analysis (Pritchard et al. 2000, Genetics 155: 945). Now, let's visualize and see what kind of relationship between genetic background and plant height and ear length.

```
> pop.id <- as.numeric(Sub.population)
                  # convert factor levels of Sub.population into values
> plot(Plant.height, Panicle.length, col = pop.id) # specify color with value
> levels(Sub.population)   # We have 6 classes (levels) for the factor
[1] "ADMIX"    "AROMATIC" "AUS"      "IND"      "TEJ"      "TRJ"
> legend("bottomright", levels(Sub.population),
         col = 1:nlevels(Sub.population), pch = 1)
         # legend is added at the corner of bottom right
```

Let's show in boxplots how there are differences in values due to differences in genetic background.

```
> boxplot(Plant.height ~ Sub.population)
> boxplot(Plant.height ~ Sub.population, border = 1:nlevels(Sub.population))
                  # draw the graph with colors
```

Let's see the relationships among plant height, panicle length, and flag leaf length with a bubble plot. Here, the size of the bubble represents the flag leaf length.

```
> symbols(Plant.height, Panicle.length,
         circles = Flag.leaf.length, inches = 0.1, fg = pop.id)
         # specify the variable represented by the size of bubbles
```

Let's draw a scatterplot of the relationship between the three variables for all possible combinations.

```
> pairs(x, col = pop.id)
```

Let's make it a bit more complex scatterplot through adding regression lines.

```
> pairs(x, panel = function(x, y, ...) {
         points(x, y, ...)                  # scatter plot
         abline(lm(y ~ x), col = "gray")    # draw line of regression
 }, col = pop.id)                           # it's a bit complicating…
```

Let's visualize the relationship among the three variables by drawing a three-dimensional scatterplot.

```
> require(scatterplot3d)           # package scatterplot3d is necessary
> scatterplot3d(Plant.height, Panicle.length, Flag.leaf.length, color = pop.id)
```

The data also have the latitude and longitude of the place where each genetic resource originates. Let's map and confirm the origin of each genetic resource on the world map.

```
> require(maps)                    # map package is necessary
> require(mapdata)                 # mapdata package is necessary
> map('worldHires')                # plot world map
> points(Longitude, Latitude, col = pop.id)
                                   # we can place a plot with long and lat
> legend("bottomleft", levels(Sub.population),
                           col = 1:nlevels(Sub.population), pch = 1)
```

The above command only draws a much smaller number of points than the number of genetic resources. This is because genetic resources from the same area overlap each other. In order to prevent the overlapping, move overlapped points a little with the function "jitter".

```
> map('worldHires')
> points(jitter(Longitude, 200), Latitude, col = pop.id)
                                    # move plots with the jitter function
> legend("bottomleft", levels(Sub.population),
                            col = 1:nlevels(Sub.population), pch = 1)
```

<Output to file of diagram>

It is useful to be able to output a graph to a PDF file in order to use the graph in a paper or presentation. Here, the method to do that will be explained briefly.

Let's output the figure we drew earlier to a file called map.pdf.

```
> pdf("map.pdf")           # name of the output pdf file
> map('worldHires')        # you cannot see the graph in the graph window
> points(jitter(Longitude, 200), Latitude, col = pop.id)
> legend("bottomleft", levels(Sub.population),
                           col = 1:nlevels(Sub.population), pch = 1)
> dev.off()                # Important!: you should close the file finally!!
null device
        1
```

When the above command is executed, a file called map.pdf is output to the working directory of R.

Function “pdf” can specify the size of the output graph. When a larger size is required as is this figure, it is better to specify the size of the graph output explicitly.

```
> pdf("map_large.pdf", width = 20, height = 10)  # 20 inch x 10 inch
> map('worldHires')
> points(jitter(Longitude, 200), Latitude, col = pop.id)
> legend("bottomleft", levels(Sub.population),
                           col = 1:nlevels(Sub.population), pch = 1)
> dev.off()
null device
        1
```

In addition, if multiple figures are repeatedly output to the same pdf file, they will be saved in a pdf file with multiple pages. If you want to output similar and a large number of figures, it may be convenient to combine them into a single pdf file.

<Draw an interactive figure>

You can use the package “plotly” to draw interactive diagrams. Here, let's draw the figure drawn earlier using plotly function plot_ly.

First, draw a histogram.

```
> plot_ly(x = Plant.height, type = "histogram")
 警告メッセージ:
Ignoring 30 observations
                  # missing entries are ignored
```

You can easily draw a horizontal histogram.

```
> plot_ly(y = Plant.height, type = "histogram")
 警告メッセージ:
Ignoring 30 observations
```

Draw a boxplot.

```
> plot_ly(y = Plant.height, type = "box")
 警告メッセージ:
Ignoring 30 observations
```

Draw a boxplot for each Sub.population.

```
> plot_ly(y = Plant.height, color = Sub.population, type = "box")
 警告メッセージ:
Ignoring 30 observations
```

Draw a bargraph.

```
> t <- table(Blast.resistance) # do it again, just in case
> plot_ly(x = names(t), y = t, type = "bar")
```

Draw a pie chart.

```
> plot_ly(labels = names(t), values = t, type = "pie")
> # add title to the chart
> plot_ly(labels = names(t), values = t, type = "pie") %>%
+ layout(title = "Blast resistance")
```

Draw a bivariate scatter plot.

```
> plot_ly(x = Plant.height, y = Panicle.length, type = "scatter", mode =
"markers")
 警告メッセージ:
Ignoring 38 observations
```

Color plots with Sub.population data.

```
> plot_ly(x = Plant.height, y = Panicle.length, color = Sub.population, type
= "scatter", mode = "markers")
 警告メッセージ:
Ignoring 38 observations
```

Change the size of plots according to the flag leaf length

```
> plot_ly(x = Plant.height, y = Panicle.length, color = Sub.population, size
= Flag.leaf.length, type = "scatter", mode = "markers")
 警告メッセージ:
1: Ignoring 38 observations
2: `line.width` does not currently support multiple values.
3: `line.width` does not currently support multiple values.
4: `line.width` does not currently support multiple values.
5: `line.width` does not currently support multiple values.
6: `line.width` does not currently support multiple values.
7: `line.width` does not currently support multiple values.
```

Although error messages are displayed, the graph is successfully drawn.

When drawing the same graph using a data.frame object, put commands as follows.

```
>   df   <-   data.frame(Sub.population,   Plant.height,   Panicle.length,
Flag.leaf.length)
> df <- na.omit(df)
>  plot_ly(data = df, x = ~Plant.height, y = ~Panicle.length, color =
~Sub.population, size = ~Flag.leaf.length, type = "scatter", mode = "markers")
 (omit error messages)
```

Try to display the density of data in three dimensions.

```
> # draw a 3D smoothed density surface
> plot_ly(data = data.frame(d), x = d$x1, y = d$x2, z = d$fhat) %>%
+ add_surface()
```

Finally, we look at the relationship between the three variables.

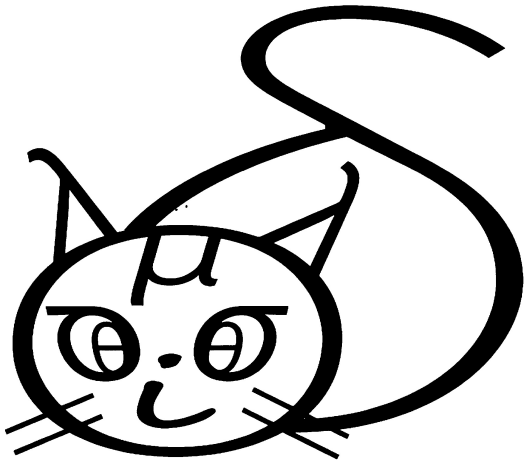
```
> # draw a 3D scatter plot
>  plot_ly(data = df, x = ~Plant.height, y = ~Panicle.length, z =
~Flag.leaf.length, color = ~Sub.population, type = "scatter3d", mode =
"markers")
```

<Report assignment>

Use the various data visualization methods learned in the lecture to illustrate the relationship between traits and the relationship between traits and genetic background. Describe the relationships that can be read from the graphs (figures) that you drew.

Submission method:

- Create a report as a pdf file and submit it as an email attachment.
- Send an e-mail to report@iu.a.u-tokyo.ac.jp.
- Make sure to write the affiliation, student number and name at the beginning of the report.
- The deadline for submission is May 10th.



K.W.