**Data, distributions, and correlation**

**The basics**

Experimental unit – The object in which information is collected upon. This information is organized into observed variable values.

Univariate data – Information measured on one variable

Multivariate data – Information measured on multiple variables

We are going to the look at the relationships that exist in multivariate data.

Example: Cereal data (cereal.csv)

One side of one aisle in most grocery stores usually contains all cereals. For example, this is what the cereal aisle looks like at one grocery store:



At a different grocery store, I collected information on the nutritional content of dry. This data was collected as a stratified random sample where shelf of the cereal was the stratum. The sample is called “stratified” because I randomly selected 10 cereals within a shelf.

There were only four shelves, where shelf #1 represents the bottom shelf and shelf #4 represents the top shelf. Below is the data:

| **ID** | | **Shelf** | **Cereal** | **Serving Size (g)** | **Sugar (g)** | **Fat (g)** | **Sodium (mg)** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | | 1 | Kellogg’s Razzle Dazzle Rice Crispies | 28 | 10 | 0 | 170 |
| 2 | | 1 | Post Toasties Corn Flakes | 28 | 2 | 0 | 270 |
| 3 | | 1 | Kellogg’s Corn Flakes | 28 | 2 | 0 | 300 |
| 4 | | 1 | Food Club Toasted Oats | 32 | 2 | 2 | 280 |
| 5 | | 1 | Frosted Cheerios | 30 | 13 | 1 | 210 |
| 6 | | 1 | Food Club Frosted Flakes | 31 | 11 | 0 | 180 |
| 7 | | 1 | Capn Crunch | 27 | 12 | 1.5 | 200 |
| 8 | | 1 | Capn Crunch's Peanut Butter Crunch | 27 | 9 | 2.5 | 200 |
| 9 | | 1 | Post Honeycomb | 29 | 11 | 0.5 | 220 |
| 10 | | 1 | Food Club Crispy Rice | 33 | 2 | 0 | 330 |
| 11 | | 2 | Rice Crispies Treats | 30 | 9 | 1.5 | 190 |
| 12 | | 2 | Kellogg's Smacks | 27 | 15 | 0.5 | 50 |
| 13 | | 2 | Kellogg's Froot Loops | 32 | 15 | 1 | 150 |
| 14 | | 2 | Capn Crunch's Peanut Butter Crunch | 27 | 9 | 2.5 | 200 |
| 15 | | 2 | Cinnamon Grahams | 30 | 11 | 1 | 230 |
| 16 | | 2 | Marshmallow Blasted Froot Loops | 30 | 16 | 0.5 | 105 |
| 17 | | 2 | Koala Coco Krunch | 30 | 13 | 1 | 170 |
| 18 | | 2 | Food Club Toasted Oats | 33 | 10 | 1.5 | 150 |
| 19 | | 2 | Cocoa Pebbles | 29 | 13 | 1 | 160 |
| 20 | | 2 | Oreo O's | 27 | 11 | 2.5 | 150 |
| 21 | 3 | Food Club Raisin Bran | 54 | 17 | 1 | 280 |
| 22 | 3 | Post Honey Bunches of Oats | 30 | 6 | 1.5 | 190 |
| 23 | 3 | Rice Chex | 31 | 2 | 0 | 290 |
| 24 | 3 | Kellogg's Corn Pops | 31 | 14 | 0 | 120 |
| 25 | 3 | Post Morning Traditions - Raisin, Date, Pecan | 54 | 14 | 5 | 160 |
| 26 | 3 | Post Shredded Wheat Spoon Size | 49 | 0 | 0.5 | 0 |
| 27 | 3 | Basic 4 | 55 | 14 | 3 | 320 |
| 28 | 3 | French Toast Crunch | 30 | 12 | 1 | 180 |
| 29 | 3 | Post Raisin Bran | 59 | 20 | 1 | 300 |
| 30 | 3 | Food Club Frosted Shredded Wheat | 50 | 1 | 1 | 0 |
| 31 | 4 | Total Raisin Bran | 55 | 19 | 1 | 240 |
| 32 | 4 | Food Club Wheat Crunch | 60 | 6 | 0 | 300 |
| 33 | 4 | Oatmeal Crisp Raisin | 55 | 19 | 2 | 220 |
| 34 | 4 | Food Club Bran Flakes | 31 | 5 | 0.5 | 220 |
| 35 | 4 | Cookie Crisp | 30 | 12 | 1 | 180 |
| 36 | 4 | Kellogg's All Bran Original | 31 | 6 | 1 | 65 |
| 37 | 4 | Food Club Low Fat Granola | 55 | 14 | 3 | 100 |
| 38 | 4 | Oatmeal Crisp Apple Cinnamon | 55 | 19 | 2 | 260 |
| 39 | 4 | Post Fruit and Fibre - Dates, Raisons, Walnuts | 55 | 17 | 3 | 280 |
| 40 | 4 | Total Corn Flakes | 30 | 3 | 0 | 200 |

Notes:

1. Each row represents a different experimental unit; these rows represent different cereal types
2. Each column represents a different measured variable

For most of the methods in this course, the experimental units are assumed to be independent. Are the experimental units in the cereal example independent?

To be independent, the observed values for one cereal cannot have an effect on another. For example, observed values for Marshmallow Blasted Froot Loops and Captain Crunch's Peanut Butter Crunch need to be independent of each other.

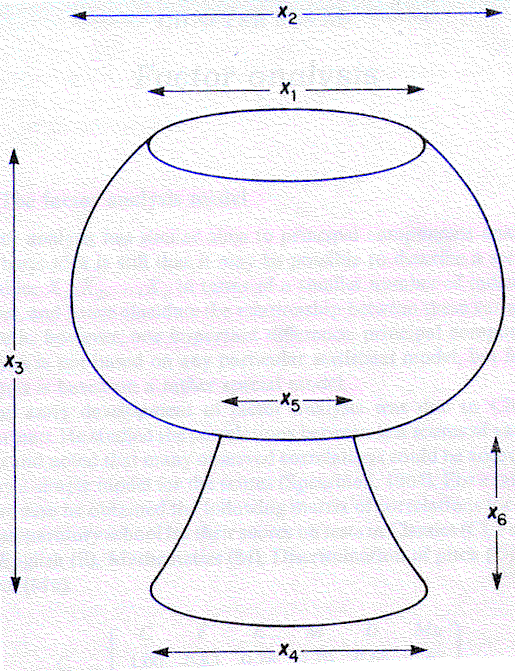
We will assume the experimental units are independent for this example.

What research questions may be of interest here?

Example: Goblet data

This data comes from an exercise in Manly and Alberto (2016).

Below is a diagram showing 6 different measurements taken on 25 pottery goblets excavated from sites in Thailand:



The corresponding data (measurements in centimeters):

| **Goblet** | **X1** | **X2** | **X3** | **X4** | **X5** | **X6** |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 13 | 21 | 23 | 14 | 7 | 8 |
| 2 | 14 | 14 | 24 | 19 | 5 | 9 |
| 3 | 19 | 23 | 24 | 20 | 6 | 12 |
| 4 | 17 | 18 | 16 | 16 | 11 | 8 |
| 5 | 19 | 20 | 16 | 16 | 10 | 7 |
| 6 | 12 | 20 | 24 | 17 | 6 | 9 |
| 7 | 12 | 19 | 22 | 16 | 6 | 10 |
| 8 | 12 | 22 | 25 | 15 | 7 | 7 |
| 9 | 11 | 15 | 17 | 11 | 6 | 5 |
| 10 | 11 | 13 | 14 | 11 | 7 | 4 |
| 11 | 12 | 20 | 25 | 18 | 5 | 12 |
| 12 | 13 | 21 | 23 | 15 | 9 | 8 |
| 13 | 12 | 15 | 19 | 12 | 5 | 6 |
| 14 | 13 | 22 | 26 | 17 | 7 | 10 |
| 15 | 14 | 22 | 26 | 15 | 7 | 9 |
| 16 | 14 | 19 | 20 | 17 | 5 | 10 |
| 17 | 15 | 16 | 15 | 15 | 9 | 7 |
| 18 | 19 | 21 | 20 | 16 | 9 | 10 |
| 19 | 12 | 20 | 26 | 16 | 7 | 10 |
| 20 | 17 | 20 | 27 | 18 | 6 | 14 |
| 21 | 13 | 20 | 27 | 17 | 6 | 9 |
| 22 | 9 | 9 | 10 | 7 | 4 | 3 |
| 23 | 8 | 8 | 7 | 5 | 2 | 2 |
| 24 | 9 | 9 | 8 | 4 | 2 | 2 |
| 25 | 12 | 19 | 27 | 18 | 5 | 12 |

Subject-matter researchers are interested in grouping goblets that have the same shape although they may have different sizes.

Example: Placekicking data

During one National Football League season, I collected information on all placekicks attempted. Variables within the data set are:

* Week: week of the season
* Distance: Distance of the placekick in yards
* Change: Binary variable denoting lead-change (1) vs. non-lead-change (0) placekicks; successful lead-change placekicks are those that change which team is winning the game.
* Elap30: Number of minutes remaining before the end of the half with overtime placekicks receiving a value of 0
* PAT: Binary variable denoting the type of placekick where a point after touchdown (PAT) is a 1 and a field goal is a 0
* Type: Binary variable denoting dome (0) vs. outdoor (1) placekicks
* Field: Binary variable denoting grass (1) vs. artificial turf (0) placekicks
* Wind: Binary variable for placekicks attempted in windy conditions (1) vs. non-windy conditions (0); I define windy as a wind stronger than 15 miles per hour at kickoff in an outdoor stadium

The response variable is referred to as “good” in the data set. Its value is a 1 for successful placekicks and a 0 for failed placekicks.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID | week | distance | change | elap30 | PAT | type | field | wind | good |
| 1 | 1 | 21 | 1 | 24.7167 | 0 | 1 | 1 | 0 | 1 |
| 2 | 1 | 21 | 0 | 15.85 | 0 | 1 | 1 | 0 | 1 |
| 3 | 1 | 20 | 0 | 0.45 | 1 | 1 | 1 | 0 | 1 |
|  |  |  |  |  |  |  |  |  |  |
| 1425 | 17 | 50 | 1 | 2.3833 | 0 | 0 | 0 | 0 | 0 |

Note that some variables, like “type”, were originally coded as characters. These have all been changed to numerical values.

There is a lot of information given! How can all of this information be summarized and is any of it meaningful? What types of inferences can be made from this data set?

Types of variables

Continuous variables – These have numerical values and can occur anywhere within some interval. There is no fixed number of values the variable can take on.

Discrete variables – These can be numerical or nonnumerical. There are a fixed number of values the variable can take on.

Example: Cereal data

Shelf and cereal are discrete variables. We can treat serving size, sugar, fat, and sodium as continuous variables even though they are probably reported on the cereal box as discrete variables.

Example: Placekicking data

Type is an example of a discrete variable. We can treat distance as a continuous variable.

Many multivariate methods were developed based on continuous variables using a multivariate normal distribution assumption (more on this later).

Some of the topics for this semester:

* Methods to summarize data
  + Graphics – We will go beyond simple 2D scatter plots. We will use graphical techniques, such as parallel coordinate plots, to graph multivariate data
  + Principle components analysis and factor analysis – These methods provide ways to reduce the dimension (number of variables) of the data set. New variables are formed which are “hopefully” interpretable.
  + Cluster analysis – Methods to group previously ungrouped items
* Methods to predict classifications (where classifications are previously known)
  + Discriminant analysis
  + Nearest neighbor analysis
  + Logistic and multinomial regression
  + Classification trees
* Classical methods
  + Hotelling’s T2 provides a multivariate extension to a t-test
  + Multivariate analysis of variance (MANOVA) provides a multivariate extension to analysis of variance (ANOVA).

**Data matrices and vectors**

Notation:

* p = number of numerical variables of interest
* N = number of experimental units on which the variables are being measured
* xrj = value of the jth variable on the rth experimental unit for r = 1, …,N and j = 1, …, p
* **X** = data matrix - xrj are arranged in matrix form so that xrj is the rth row and jth column element of **X**; rows represent experimental units and columns represent variables.
* **X** has dimension N×p
*  = vector of data for the rth experimental unit
*  = [xr1, xr2, …, xrp] = “transpose” of 
* 

Example: Cereal data

Consider only the shelf, sugar, fat, and sodium variables. We need to adjust the variables for the serving size by taking their observed values divided by serving size. For example, for Kellog’s Razzle Dazzle Rice Crispies the “sugar” value is

(sugar grams per serving)/(# of grams per serving size) = 10/28 = 0.3571

The data looks as follows:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Cereal** | **Shelf** | **Sugar** | **Fat** | **Sodium** |
| 1 | Kellog’s Razzle Dazzle Rice Crispies | 1 | 0.3571 | 0 | 6.0714 |
| 2 | Post Toasties Corn Flakes | 1 | 0.0714 | 0 | 9.6429 |
|  |  |  |  |  |  |
| 40 | Total Corn Flakes | 4 | 0.1 | 0 | 6.6667 |

The ID and Cereal variables are in the table to help identify the experimental units.

* p = 4
* N = 40
* x11 = 1, x12 = 0.3571, x13 = 0, x14 = 6.0714,   
  x21 = 1, x22 = 0.0714, …
* 
* 

**The multivariate normal distribution**

Let x be a random variable with an univariate normal distribution with mean E(x) = μ and variance Var(x) =   
E[(x-μ)2] = σ2. This is represented symbolically as   
x ~ N(μ, σ2). Note that x could be capitalized here if one wanted to represent it more formally. The probability density function of x is

 for - < x < .

Example: Univariate normal distribution plot (NormalPlot.R)

Suppose x~N(50, 32). Below is the code used to plot f(x|μ,σ):

> curve(expr = dnorm(x = x, mean = 50, sd = 3), xlim =

c(40, 60), col = "red", xlab = "x", ylab = "f(x)", main =

"Univariate normal distribution")

> dnorm(x = c(40, 50, 60), mean = 50, sd = 3)

[1] 0.000514093 0.132980760 0.000514093

> abline(h = 0)

A diagram of a normal distribution

Description automatically generated

The normal distribution for a random variable can be generalized to a multivariate normal distribution for a vector of random variables (random vector). Some of the theory for multivariate analysis methods relies on the assumption that a random vector



has a multivariate normal distribution.

To introduce the multivariate normal distribution function, we need examine a few other items first:

* Let E(xi) = μi
* The relationships between the xi’s can be measured by the covariances and/or the correlations.

Covariance of xi and xj:

Cov(xi, xj) = E[(xi-μi)(xj-μj)] = σij

Covariance of xi and xi (variance of xi):

Var(xi) = E[(xi-μi)2] = 

Correlation coefficient of xi and xj:



Remember that σij = σji and -1 ≤ ρij ≤ 1. Another commonly used notation for  is Corr(xi, xj). One just need to be careful with how this is used given how Corr() is used in the next bullet.

* The means, covariances, correlations can be put into a mean vector, covariance matrix, and a correlation matrix:

**μ** = E(**x**) = ,

, and .

* The covariance matrix can be expressed another way. To relate this to something you may have seen before, note that



Another way to express the covariance matrix then is



Notice that if the mean is **0**, we simply have .

* Remember that **Σ** and **P** are symmetric and knowing **Σ** is equivalent to knowing **P**.
* The notation, **x** ~ Np(**μ**,**Σ**), means that **x** has a p-dimension multivariate normal distribution with mean **μ** and covariance matrix **Σ**.

The multivariate normal density function is:



for - < xi < for i = 1,…,p and |**Σ**| > 0.

Example: Bivariate normal distribution plot (NormalPlot.R).

Suppose  has a multivariate normal distribution with

**μ** =  and .

We could write this as  or

.

Also, note that



For example, ρ12 =  = 0.45.

The multivariate normal distribution function is:





Because there are only two random variables, we can refer to the multivariate normal as a “bivariate” normal.

Below is how the distribution can be plotted:

> library(rgl) #Needed for 3D plot

> library(mvtnorm) #Needed for dmvnorm()

> mu <- c(15, 20)

> sigma <- matrix(data = c(1, 0.5, 0.5, 1.25), nrow = 2,

ncol = 2, byrow = TRUE)

> P <- cov2cor(V = sigma)

> P

[,1] [,2]

[1,] 1.0000000 0.4472136

[2,] 0.4472136 1.0000000

> #Example f(x) calculation at mean

> dmvnorm(x = mu, mean = mu, sigma = sigma)

[1] 0.1591549

There is a not a curve() function in R that will produce a 3D rotatable plot if simply given dmvnorm(). Instead, I create a set of possible x1 and x2 values which are used to evaluate dmvnorm()to obtain f(**x**). I use these resulting values with a 3D rotatable plotting function and also a contour plotting function. Below is the code and output:

> # Calculate f(x) for a large number of possible values for x1

and x2

> x1 <- seq(from = 11, to = 19, by = 0.1)

> x2 <- seq(from = 17, to = 23, by = 0.1)

> all.x <- expand.grid(x1, x2) # Creates all possible

combinations

> head(all.x) # Notice that x1 is changing faster than x2

Var1 Var2

1 11.0 17

2 11.1 17

3 11.2 17

4 11.3 17

5 11.4 17

6 11.5 17

> eval.fx <- dmvnorm(x = all.x, mean = mu, sigma = sigma)

> # Arrange values in the following form:

> # x2

> # 17.0 17.1 17.2 ...

> # x1 11.0 f(x) f(x) f(x)

> # 11.1 f(x) f(x) f(x)

> # 11.2 f(x) f(x) f(x)

> # ...

> fx <- matrix(data = eval.fx, nrow = length(x1), ncol =

length(x2), byrow = FALSE)

> # Check if f(x) values are arranged correctly

> all.x[1:2,]

Var1 Var2

1 11.0 17

2 11.1 17

> dmvnorm(x = all.x[1:2,], mean = mu, sigma = sigma)

1 2

3.238300e-05 4.566735e-05

> fx[1:2, 1] # Part of column 1

[1] 3.238300e-05 4.566735e-05

> dmvnorm(x = c(11.0, 17.1), mean = mu, sigma = sigma)

[1] 3.561025e-05

> fx[1, 1:2] # Part of row 1

[1] 3.238300e-05 3.561025e-05

> # Matrix \*10^(-5)

> # x2

> # 17.0 17.1 ...

> # x1 11.0 3.24 3.56

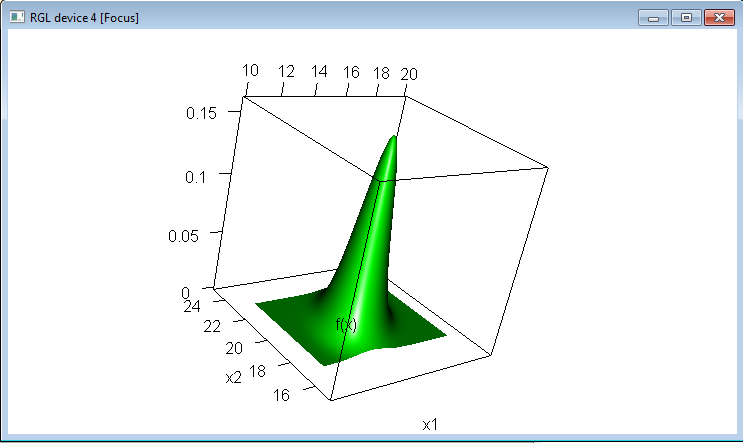
> # 11.1 4.57

> # ...

> # 3D plot

> persp3d(x = x1, y = x2, z = fx, col = "green", xlab =

"x1", ylab = "x2", zlab = "f(x)")



> # Contour plot – purposely made x and y-axes the same

length so that one can judge variability

> par(pty = "s")

> contour(x = x1, y = x2, z = fx, main = "Multivariate

normal", xlab = expression(x[1]), ylab =

expression(x[2]), xlim = c(10,20), ylim = c(15, 25))

> # panel.first = grid() does not work quite right with

contour() so used abline() instead

> abline(h = 17:23, lty = "dotted", col = "lightgray")

> abline(v = seq(from = 12, to = 18, by = 2), lty =

"dotted", col = "lightgray")



To rotate the first plot, hold your left mouse button down while you move the mouse. To zoom in on the first plot, hold your right mouse button down while you move the mouse.

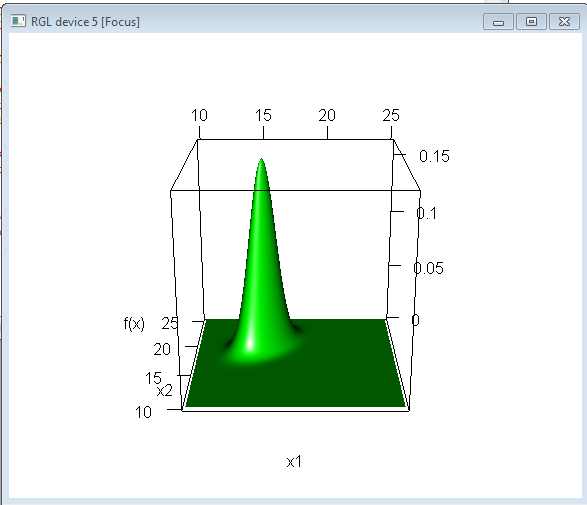
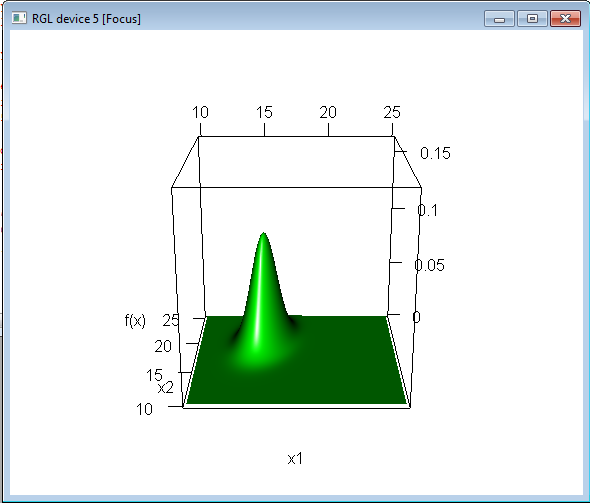
Examine the following:

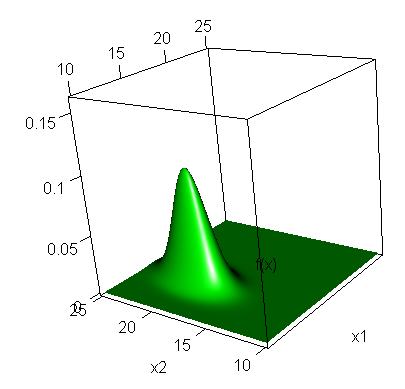
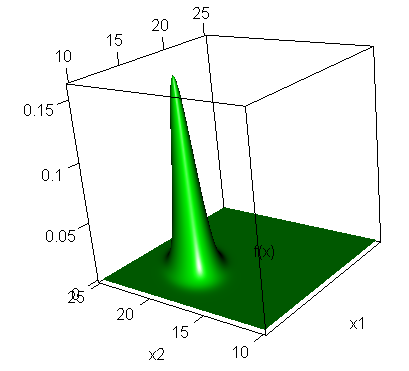
1. The surface is centered at **μ**.
2. The surface is wider in the x2 direction than in the x1. This is because there is more variability.
3. Notice the shape of the surface. The contour plot is an ellipse and the surface plot looks like a 3D normal curve.
4. Volume underneath the surface is 1.

Note that the 3D plot is not in the graphics window. To get the plot into Word, I used the Windows’ Snipping Tool.

Suppose **μ** =  and . The only change from the previous example is that σ22 has increased. Note that . Below are 3D surface and contour plots drawn on the same scale for the previous and new normal distributions.

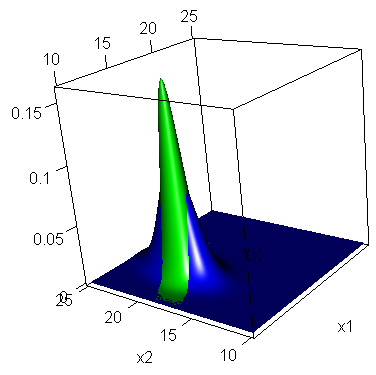
ρ12 = 0.45 ρ12 = 0.29



Overlaying the plots:





Questions about the bivariate normal:

1. What happens if the means change?
2. What happens if σ12 = 0?
3. What happens if σ11 = σ22 and σ12 = 0?
4. What happens if ρ12 = 1 or -1?

If you do not know the answers, use the R program to investigate!

Question: Suppose a random sample is taken from a population which can be characterized by the first multivariate normal distribution. A scatter plot is constructed of the observed x1 and x2 values. Where would most of the (x1, x2) points fall?

**Multivariate summary statistics**

Let **x**1, **x**2, …, **x**N be a random sample from a multivariate distribution (not necessarily normal) with a mean vector **μ**, covariance matrix **Σ**, and correlation matrix **P**. This can be represented symbolically as .

Note on notation: If it is understood that a random sample of size N is taken, then we could shorten this to .

Estimates of **μ**, **Σ**, and **P** based on the random sample are:





Notes:

1. 
2.   
   where  and 

Equivalently, you can express  as follows. Suppose  is the N×p matrix with mean adjusted values. In other words, the rth row and ith column element is . Then



One can see the two expressions for the estimated covariance matrix are equivalent by examining a very small example (e.g., let p = 2 and N = 3).

1. rij and **R** are used to denote the estimates of ρij and **P**, respectively.
2. 

and



Example: Cereal data (cereal.R, cereal.csv)

Let x1 denote sugar, x2 denote fat, and x3 denote sodium which are adjusted for the number of grams per serving. Suppose



meaning we have a random sample of size 40 from . Thus, we have **x**1, **x**2, …, **x**40. More simply, someone may say  with N = 40.

Using the observed values of **x**1, **x**2, …, **x**40 produces the estimates of **μ**, **Σ**, and **P**:





Below is my R code and output to find these items more easily:

> cereal <- read.csv(file = "c:\\data\\cereal.csv")

> head(cereal) # Shows first 6 observations

ID Shelf Cereal size\_g

1 1 1 Kellog's Razzle Dazzle Rice Crispies 28

2 2 1 Post Toasties Corn Flakes 28

3 3 1 Kellogg's Corn Flakes 28

4 4 1 Food Club Toasted Oats 32

5 5 1 Frosted Cheerios 30

6 6 1 Food Club Frosted Flakes 31

sugar\_g fat\_g sodium\_mg sugar fat sodium

1 10 0 170 0.35714286 0.00000000 6.071429

2 2 0 270 0.07142857 0.00000000 9.642857

3 2 0 300 0.07142857 0.00000000 10.714286

4 2 2 280 0.06250000 0.06250000 8.750000

5 13 1 210 0.43333333 0.03333333 7.000000

6 11 0 180 0.35483871 0.00000000 5.806452

> # Adjust data to take into account the different serving

sizes

> cereal$sugar <- cereal$sugar\_g/cereal$size\_g

> cereal$fat <- cereal$fat\_g/cereal$size\_g

> cereal$sodium <- cereal$sodium\_mg/cereal$size\_g

> head(cereal)

ID Shelf Cereal size\_g

1 1 1 Kellog's Razzle Dazzle Rice Crispies 28

2 2 1 Post Toasties Corn Flakes 28

3 3 1 Kellog's Corn Flakes 28

4 4 1 Food Club Toasted Oats 32

5 5 1 Frosted Cheerios 30

6 6 1 Food Club Frosted Flakes 31

sugar\_g fat\_g sodium\_mg sugar fat sodium

1 10 0 170 0.35714286 0.00000000 6.071429

2 2 0 270 0.07142857 0.00000000 9.642857

3 2 0 300 0.07142857 0.00000000 10.714286

4 2 2 280 0.06250000 0.06250000 8.750000

5 13 1 210 0.43333333 0.03333333 7.000000

6 11 0 180 0.35483871 0.00000000 5.806452

> # Estimated covariance and correlation matrices

> sigma.hat <- cov(cereal[, 8:10])

> R <- cor(cereal[, 8:10])

> sigma.hat

sugar fat sodium

sugar 0.0223681533 0.0009926903 -0.060242015

fat 0.0009926903 0.0007666194 -0.004509788

sodium -0.0602420149 -0.0045097883 6.064039752

> R

sugar fat sodium

sugar 1.0000000 0.2397225 -0.1635699

fat 0.2397225 1.0000000 -0.0661432

sodium -0.1635699 -0.0661432 1.0000000

> # Alternative way to find R

> cov2cor(sigma.hat)

sugar fat sodium

sugar 1.0000000 0.2397225 -0.1635699

fat 0.2397225 1.0000000 -0.0661432

sodium -0.1635699 -0.0661432 1.0000000

> # Estimated mean vector

> mu.hat <- colMeans(cereal[, 8:10])

> mu.hat

sugar fat sodium

0.28941556 0.03218277 5.61470382

> # Alternative way

> apply(X = cereal[, 8:10], MARGIN = 2, FUN = mean)

sugar fat sodium

0.28941556 0.03218277 5.61470382

> # Equivalent way to obtain covariance matrix

> X <- as.matrix(cereal[,8:10])

> X.tilde <- t(t(X) - mu.hat)

> N <- nrow(X)

> t(X.tilde)%\*%X.tilde/(N-1)

sugar fat sodium

sugar 0.0223681533 0.0009926903 -0.060242015

fat 0.0009926903 0.0007666194 -0.004509788

sodium -0.0602420149 -0.0045097883 6.064039752

Example: Bivariate normal distribution plot (NormalPlot.R)

Let  as before. The program below shows how to take a sample from a population characterized by this distribution

> mu <- c(15, 20)

> sigma <- matrix(data = c(1, 0.5, 0.5, 1.25), nrow = 2,

ncol = 2, byrow = TRUE)

> cov2cor(sigma)

[,1] [,2]

[1,] 1.0000000 0.4472136

[2,] 0.4472136 1.0000000

> N <- 20

> set.seed(7812) #Set a “seed number” so that I can

reproduce the exact same sample

> x <- rmvnorm(n = N, mean = mu, sigma = sigma)

> x

[,1] [,2]

[1,] 15.98996 20.40392

[2,] 15.49028 18.35249

[3,] 15.38903 21.06337

[4,] 16.03633 20.03042

[5,] 14.70052 18.29860

[6,] 14.40092 20.51145

[7,] 16.26059 20.42173

[8,] 14.91134 20.52605

[9,] 14.83376 21.30088

[10,] 14.79829 18.81859

[11,] 14.41775 19.39798

[12,] 13.96203 18.71438

[13,] 14.36722 20.45083

[14,] 13.93155 18.74072

[15,] 15.79059 19.56830

[16,] 17.66219 21.03899

[17,] 14.46738 19.26985

[18,] 15.98765 21.13704

[19,] 16.31064 21.81709

[20,] 16.04713 20.72254

> mu.hat <- colMeans(x)

> sigma.hat <- cov(x)

> R <- cor(x)

> mu.hat

[1] 15.28776 20.02926

> sigma.hat

[,1] [,2]

[1,] 0.9288198 0.5473159

[2,] 0.5473159 1.1268952

> R

[,1] [,2]

[1,] 1.0000000 0.5349714

[2,] 0.5349714 1.0000000

Compare the estimates to the true values. Are the estimates reasonable? What would happen if a larger sample size was taken?

Below is a scatter plot of the data overlaid on a contour plot. The code is the same as before except that I added the 0.001 contour and used the points() function to add the 20 observations to the plot.



Notice that the points are within the larger value contours. Why should this happen?

What if N = 200 instead? I used the same code as before except changed the value of N:

> mu.hat

[1] 15.09207 20.00076

> sigma.hat

[,1] [,2]

[1,] 1.2076645 0.4951661

[2,] 0.4951661 1.2458738

> R

[,1] [,2]

[1,] 1.0000000 0.4036833

[2,] 0.4036833 1.0000000



What if N = 2000 instead? I used the same code as before except changed the value of N:

> mu.hat

[1] 15.00649 19.99490

> sigma.hat

[,1] [,2]

[1,] 0.9726854 0.4203948

[2,] 0.4203948 1.2017892

> R

[,1] [,2]

[1,] 1.0000000 0.3888275

[2,] 0.3888275 1.0000000



I was a little surprised to see r12 get farther from the true value. I also tried N = 20000 and obtained r12= 0.44.

Questions:

* How would you simulate trivariate normal data?
* Why is it useful to be able to simulate data?

**Checking multivariate normality**

Some of the statistical methods that we will examine this semester have an underlying multivariate normal distribution assumption for the data.

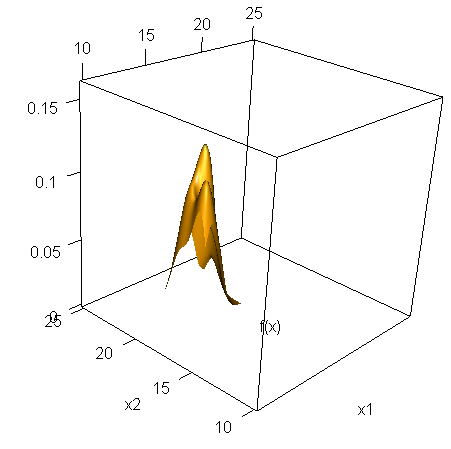
Questions:

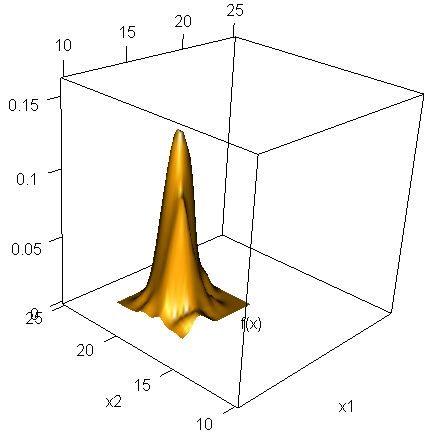
* Suppose a statistical method has a multivariate normal distribution assumption and it was not satisfied. What are the potential consequences?
* How could you approximately verify multivariate normality was true for
  + Data collected upon two variables
  + Data collected upon three variables
  + Data collected upon > three variables

An “ad-hoc” way for checking a multivariate normality assumption is to check for data falling into an approximate ellipse shape within each possible pairwise scatter plot (use side-by-side scatter plots – see the graphics section). This does NOT ensure multivariate normality, but it gives “some” credibility to the assumption.

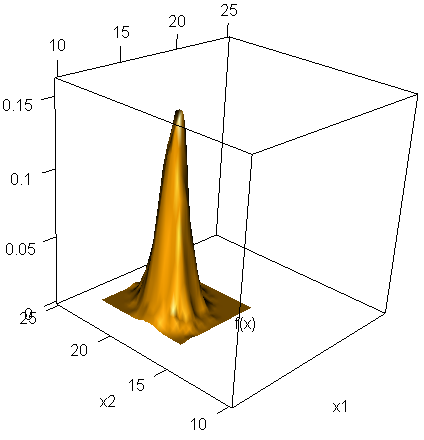
Example: Bivariate normal distribution plot (NormalPlot.R)

For the  data simulation example, below are 3D estimates of the density for **x**:

N = 20 (it’s best to rotate this plot):



N = 200 (it’s best to rotate this plot):

N= 2000:

What do you think about the bivariate normal assumption for each N?

Below is the code that I added to the previous data simulation example:

> fx.hat <- kde2d(x = x[,1], y = x[,2])

> # fx.hat$x

> # fx.hat$y

> # fx.hat$z # Matrix of estimated f(x) values for a grid of

x-axis and y-axis values

> persp3d(x = fx.hat$x, y = fx.hat$y, z = fx.hat$z, col =

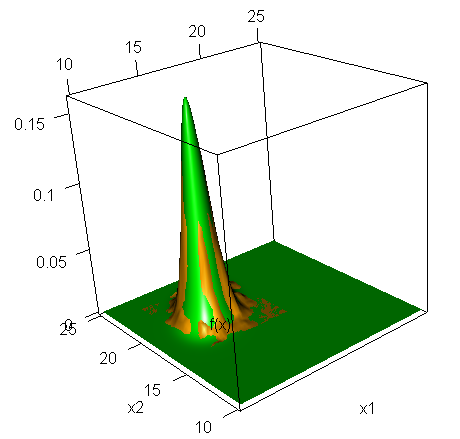
"orange", xlab = "x1", ylab = "x2", zlab = "f(x)", xlim

= c(10, 25), ylim = c(10,25), zlim = c(0, 0.16))

The kde2d() function produces a “kernel density estimate” for the probability density based on the sampled data. You are not responsible for knowing how kernel density estimation works.

The actual normal distribution can be added to the plots as well:

N = 2000:



Contour plots can be drawn as well! Please see the corresponding program.

**Standardized Data**

Sometimes it is easier to work with data which are on the same scale. Standardized data can be used to convert the data to a unitless scale. Let



for r = 1, …, N and j = 1, …, p. The zrj values together will have a sample mean of 0 and sample variance of 1 for each variable r.

We can put all of these values into a matrix:



Example: Cereal data (cereal.R, cereal.csv)

For sugar, fat, and sodium, I standardized their serving size adjusted values:

> Z <- scale(cereal[,8:10])

> head(Z)

sugar fat sodium

1 0.4528441 -1.16234084 0.18546994

2 -1.4575233 -1.16234084 1.63578030

3 -1.4575233 -1.16234084 2.07087341

4 -1.5172223 1.09496344 1.27320271

5 0.9622754 0.04155478 0.56255063

6 0.4374379 -1.16234084 0.07786627

> cov(Z)

sugar fat sodium

sugar 1.0000000 0.2397225 -0.1635699

fat 0.2397225 1.0000000 -0.0661432

sodium -0.1635699 -0.0661432 1.0000000

> cor(Z)

sugar fat sodium

sugar 1.0000000 0.2397225 -0.1635699

fat 0.2397225 1.0000000 -0.0661432

sodium -0.1635699 -0.0661432 1.0000000

> colMeans(Z)

sugar fat sodium

-1.380840e-16 -4.565575e-17 2.480655e-17

> #Check for sugar

> z.sugar <- (cereal$sugar - mean(cereal$sugar)) /

sd(cereal$sugar)

> mean(z.sugar)

[1] -1.380948e-16

> sd(z.sugar)

[1] 1

Questions:

* Why is the estimated covariance matrix equal to the estimated correlation matrix?
* Notice the estimated correlation matrix is the same as the estimated correlation matrix for the serving size adjusted values. Why?

**Eigenvectors and multivariate normal distributions**

Let λ1 be the largest eigenvalue from **Σ**, let λ2 be the 2nd largest eigenvalue from **Σ**, …, and let λp be the smallest eigenvalue from **Σ**.

With respect to a bivariate normal distribution, the spread and direction of contours on a contour plot are related to the direction of the eigenvectors.

* The first eigenvector (corresponding to λ1) points in the direction of the major axis of the ellipse created by the contours. This is the direction of the largest variability.
* The second eigenvector (corresponding to λ2) points in the direction of the minor axis of the ellipse created by the contours. This is the direction of the smallest variability.

Example: Bivariate normal distribution plot (NormalPlot.R).

Let  as before. Plotted below are the f(**x**) = 0.01 and 0.001 contours and the eigenvectors with a length of 10.

> mu <- c(15, 20)

> sigma <- matrix(data = c(1, 0.5, 0.5, 1.25), nrow = 2,

ncol = 2, byrow = TRUE)

> x1 <- seq(from = 10, to = 30, by = 0.1)

> x2 <- seq(from = 10, to = 30, by = 0.1)

> all.x <- expand.grid(x1, x2) # Creates all possible

combinations

> eval.fx <- dmvnorm(x = all.x, mean = mu, sigma = sigma)

> fx <- matrix(data = eval.fx, nrow = length(x1), ncol =

length(x2), byrow = FALSE)

> par(pty = "s")

> contour(x = x1, y = x2, z = fx,

main = expression(paste("Multivariate normal contour

plot with eigenvectors for ", Sigma)),

xlab = expression(x[1]), ylab = expression(x[2]), xlim

= c(-10,30), ylim = c(-10, 30), levels = c(0.01,

0.001))

> abline(h = seq(from = -10, to = 30, by = 10), lty =

"dotted", col = "lightgray")

> abline(v = seq(from = -10, to = 30, by = 10), lty =

"dotted", col = "lightgray")

> #Put in lines at x1 = 0 and x2 = 0

> abline(h = 0, lwd = 2)

> abline(v = 0, lwd = 2)

> save.eig <- eigen(cov.mat)

> save.eig$values

[1] 1.6403882 0.6096118

> save.eig$vectors

[,1] [,2]

[1,] 0.6154122 -0.7882054

[2,] 0.7882054 0.6154122

> arrows(x0 = 0, y0 = 0, x1 = 10\*save.eig$vectors[1,1], y1

= 10\*save.eig$vectors[2,1], col = "red", lty = "solid")

> arrows(x0 = 0, y0 = 0, x1 = 10\*save.eig$vectors[1,2], y1

= 10\*save.eig$vectors[2,2], col = "red", lty = "solid")



The eigenvectors point in the direction of the ellipses’ major and minor axes. The first eigenvector points in the direction in the largest variability.

What if Var(X2) is smaller? Let



and note that |**Σ**| = 0.01 > 0, λ1 = 1.25, and λ2 = 0.01. The correlation is  = 0.98. Below is the contour plot. Notice that there is little variability for the minor axis.

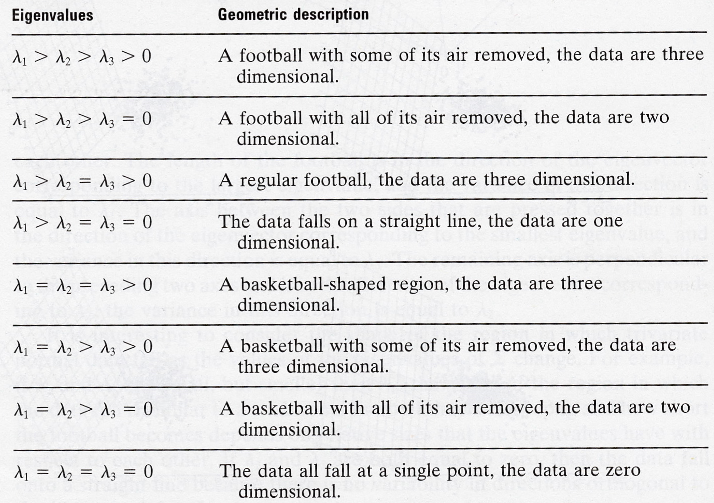


In fact, as λ2 goes to 0, the contours will become a straight line. Thus, we could essentially just use one dimension to view resulting data from this distribution!

What if we had a trivariate normal distribution?

Plot 3×1 vectors in 3-dimensions with each axis corresponding to one of the three variable values. A 3D contour plot could be drawn as well where a 3D ellipsoid would appear in it. The size of the 3D ellipsoid correspond to the size of the eigenvalues. A small eigenvalue corresponds to a small axis, and vice versa for a large eigenvalue.

One can think of the trivariate case in terms of a football and a basketball. Below is a table from Johnson (1998, p. 90):



Pay special attention to examples where an eigenvalue is 0.

What about the most general p-variate case?

Suppose there are 20 variables in a data set. A plot of these variables in 20 dimensions would be VERY difficult to do! If possible, we would prefer to view data values in 2 or 3 dimensions. Suppose that all of the estimated covariance matrix’s eigenvalues were approximately 0 except for 2 or 3. This implies that the data can be viewed in 2 or 3 dimensions without losing much information.

This is essentially what principal components analysis allows us to do: view our data in possibly a much smaller number of dimensions.