MULTIVARIATE DATA AND ANALYSIS EXERCISES

Easy Questions:

1. Find the correlation matrix and covariance matrix of the **age**, **IQ** and **weight** variables in the hypo data after filling in the missing values with mean replacements (i.e. the mean of that column for the existing data) rounded to the nearest whole integer.

```
hypo <- data.frame(age=c(21,43,22,86,60,16,NA,43,22,80),
                   IQ=c(120,NA,135,150,92,130,150,NA,84,70),
                   weight=c(150,160,135,140,110,110,120,120,105,100))

# take a look
hypo

# function f() replaces NAs in a
# column with the mean for that column
f <- function(x){
  # convert the NA item
  # to mean value from the column
  # rounded to nearest digit
  x[is.na(x)] = round(colMeans(x, na.rm=TRUE))
  # note that each time an NA is
  # replaced, the overall colMean() changes
  # IQ first NA becomes 116, 2nd becomes 125
  # display the columns
  x
}

# call function assign output to f.hypo
f.hypo <- f(hypo)

# look at f.hypo
```

```
f.hypo

# covariance matrix is
cov(f(hypo)[, c("age", "IQ", "weight")])

# correlation matrix is
cor(f.hypo[, c("age", "IQ", "weight")])
```

2. Create and examine both the normal probability plots of each variable in the archaeology data and the chi-square plot of the data. Do the plots suggest anything unusual about the data?

```
# load the data, check it out
data("pottery", package = "HSAUR2")
head(pottery)
nrow(pottery)
ncol(pottery)
str(pottery) # all numeric except kiln
pottery$kiln
class(pottery$kiln) # kiln is a factor
levels(pottery$kiln) # with five levels
# variable names
names(pottery)

# covariance matrix is
round(cov(pottery[,names(pottery)[1:9]]),digits=3)

# correlation matrix is
round(cor(pottery[,names(pottery)[1:9]]),digits=3)

# Create and examine both the normal probability
# plots of each variable in the archaeology data.

# Probability plots for each separate variable
```

```
layout(matrix(1:9, nc = 3))
# Iterate over all variables with sapply() that
# loops over variable names except kiln.
# Notice are using an anonymous function as
# second argument to sapply()
sapply(names(pottery)[1:9], function(x) {
  qqnorm(pottery[[x]], main = x)
  qqline(pottery[[x]])
})

# several of the univariate plots show
# problematic 'patternistic' deviations: MgO,
# Fe203, CaO and K20 deviate considerably from
# linearity; Actually all nine plots show some
# degree of deviation.

# and the chi-square plot of the data. Do the
# plots suggest anything unusual about the data?

# just want first nine variables (columns)
x <- pottery[,1:9]
# derive means of each column
cm <- colMeans(x)
# calculate the covariance matrix
S <- cov(x)
d <- apply(x, 1, function(x) t(x - cm) %*% solve(S) %*% (x - cm))
plot(qc <- qchisq((1:nrow(x) - 1/2) / nrow(x), df = 9),
     sd <- sort(d),
     xlab = expression(paste(chi[9]^2, " Quantile")),
     ylab = "Ordered distances",
     xlim = range(qc) * c(1, 1.1))
abline(a = 0, b = 1)
```

```
# the multivariate plot looks better.
```

More Difficult Questions, 'Extra Credit':

3. Manually convert this covariance matrix into the corresponding correlation matrix:

   3.8778 2.8110 3.1480 3.5062
   2.8110 2.1210 2.2669 2.5690
   3.1480 2.2669 2.6550 2.8341
   3.5062 2.5690 2.8341 3.2352

   Note that there is an R function that already does this: **cov2cor(),** which you can use to check your work. Once you successfully 'hand-crank' the computations to convert a covariance matrix into a corresponding correlation matrix, then write a user-defined R function, **cor.matrix()** to do it for you. Make sure that your **cor.matrix()** function prints out the: (1) original covariance matrix; (2) the **cov2cor()** computed correlation matrix, and then finally, just beneath (3) your own **cor.matrix()** computed correlation matrix. Are they the same?

```
# Here is the covariance matrix:
covM <- matrix(
  c(3.8778,  2.8110,  3.1480, 3.5062,
    2.8110,  2.1210,  2.2669, 2.5690,
    3.1480,  2.2669,  2.6550, 2.8341,
    3.5062,  2.5690,  2.8341, 3.2352), ncol = 4)

# what does matrix look like?:
covM

# There is an R function that will convert
# a covariance matrix into a correlation
# matrix cov2cor()
corM <- cov2cor(covM);corM

# Or you can do it manually.
```

```
# The correlation coefficient is simply
# the covariance of row i and column j
# divided by the product of the standard
# deviations of row i and column j.


# The standard deviations of the original
# variables will be the square roots of the
# variances. These are now the diagonals
# of the covariance matrix.

# First, use the diag() function to extract
# the variances from the diagonal elements of
# the covariance matrix and then take the square
# roots of each:
sqrt(diag(covM))

# These are the standard deviations of the
# original 4 variables:
sds <- sqrt(diag(covM));sds

# Then divide each covariance matrix element
# by the product of the corresponding standard
# deviations for those two variables (row, col)

# so we need to divide element by element
# covM[i,j] by the product of sds[i] x sds[j]

# we initialize a new correlation matrix
# cor.M and fill it with zeros
cor.M <- matrix(numeric(16),nrow=nrow(covM))
cor.M
```

```
# first column of cor.M will be:
covM[1,1]/(sds[1]*sds[1])
covM[2,1]/(sds[2]*sds[1])
covM[3,1]/(sds[3]*sds[1])
covM[4,1]/(sds[4]*sds[1])

# second column of cor.M will be:
covM[1,2]/(sds[1]*sds[2])
covM[2,2]/(sds[2]*sds[2])
covM[3,2]/(sds[3]*sds[2])
covM[4,2]/(sds[4]*sds[2])

# third column of cor.M will be:
covM[1,3]/(sds[1]*sds[3])
covM[2,3]/(sds[2]*sds[3])
covM[3,3]/(sds[3]*sds[3])
covM[4,3]/(sds[4]*sds[3])

# fourth column of cor.M  will be:
covM[1,4]/(sds[1]*sds[4])
covM[2,4]/(sds[2]*sds[4])
covM[3,4]/(sds[3]*sds[4])
covM[4,4]/(sds[4]*sds[4])

# now we fill up cor.M:
# first column of cor.M:
cor.M[1,1] <- covM[1,1]/(sds[1]*sds[1])
cor.M[2,1] <- covM[2,1]/(sds[2]*sds[1])
cor.M[3,1] <- covM[3,1]/(sds[3]*sds[1])
cor.M[4,1] <- covM[4,1]/(sds[4]*sds[1])

# second column of cor.M:
cor.M[1,2] <- covM[1,2]/(sds[1]*sds[2])
```

```
cor.M[2,2] <- covM[2,2]/(sds[2]*sds[2])
cor.M[3,2] <- covM[3,2]/(sds[3]*sds[2])
cor.M[4,2] <- covM[4,2]/(sds[4]*sds[2])

# third column of cor.M:
cor.M[1,3] <- covM[1,3]/(sds[1]*sds[3])
cor.M[2,3] <- covM[2,3]/(sds[2]*sds[3])
cor.M[3,3] <- covM[3,3]/(sds[3]*sds[3])
cor.M[4,3] <- covM[4,3]/(sds[4]*sds[3])

# fourth column of cor.M:
cor.M[1,4] <- covM[1,4]/(sds[1]*sds[4])
cor.M[2,4] <- covM[2,4]/(sds[2]*sds[4])
cor.M[3,4] <- covM[3,4]/(sds[3]*sds[4])
cor.M[4,4] <- covM[4,4]/(sds[4]*sds[4])

corM
cor.M

# Let's simplify this:
sds <- sqrt(diag(covM));sds
corm <- matrix(numeric(16),nrow(covM))
for (c in 1:ncol(corm)){
  for (r in 1:nrow(corm)){
    corm[r,c] <- covM[r,c]/(sds[r]*sds[c])
  }
  print(corm)
}

# Let's write our own function to do this
# that just uses only the original covariance
# matrix as the only mandatory argument.
```

```
# Also, we assume no missing data.

cor.matrix <- function(covM){
  sds <- sqrt(diag(covM));sds
  corm <- matrix(numeric(16),nrow(covM))
  for (c in 1:ncol(corm)){
    for (r in 1:nrow(corm)){
      corm[r,c] <- covM[r,c]/(sds[r]*sds[c])
    }
  }
  cat("\n")
  cat("This is the original covariance matrix:","\n")
  print(covM)
  cat("\n")
  cat("This is computed cov2cor() correlation matrix:","\n")
  print(cov2cor(covM))
  cat("\n")
  cat("Here is our computed correlation matrix:","\n")
  return(corm)
}

covM
cor.matrix(covM)

# Look at output of cor.matrix in console.
```