Missing Data Lab

In this lab, we will implement some of the techniques we have looked at in class to deal with missing data. You will be using a dataset on paua, or abalone. The data set includes the following variables:

Name	Data Type	Measurement Unit	Description
Type	nominal	–	M, F, and I (infant)
Length	continuous	mm	Longest shell measurement
Diameter	continuous	mm	perpendicular to length
Height	continuous	mm	with meat in shell
Whole.weight	continuous	grams	whole abalone
Shucked.weight	continuous	grams	weight of meat
Viscera.weight	continuous	grams	gut weight (after bleeding)
Shell.weight	continuous	grams	after being dried
Rings	integer	–	+1.5 gives the age in years

Start by downloading the dataset.

```
path <- "http://archive.ics.uci.edu/ml/machine-learning-databases/abalone/"
abalone <- read.csv(paste(path,"abalone.data",sep=""), header = FALSE)
names(abalone) <- c("Type", "Length", "Diameter", "Height", "Whole.weight",
    "Shucked.weight", "Viscera.weight", "Shell.weight", "Rings")
summary(abalone)
```

```
##      Type               Length          Diameter          Height         Whole.weight
##  Length:4177        Min.   :0.075   Min.   :0.0550   Min.   :0.0000   Min.   :0.0020
##  Class :character   1st Qu.:0.450   1st Qu.:0.3500   1st Qu.:0.1150   1st Qu.:0.4415
##  Mode  :character   Median :0.545   Median :0.4250   Median :0.1400   Median :0.7995
##                     Mean   :0.524   Mean   :0.4079   Mean   :0.1395   Mean   :0.8287
##                     3rd Qu.:0.615   3rd Qu.:0.4800   3rd Qu.:0.1650   3rd Qu.:1.1530
##                     Max.   :0.815   Max.   :0.6500   Max.   :1.1300   Max.   :2.8255
##  Shucked.weight   Viscera.weight    Shell.weight        Rings
##  Min.   :0.0010   Min.   :0.0005   Min.   :0.0015   Min.   : 1.000
##  1st Qu.:0.1860   1st Qu.:0.0935   1st Qu.:0.1300   1st Qu.: 8.000
##  Median :0.3360   Median :0.1710   Median :0.2340   Median : 9.000
##  Mean   :0.3594   Mean   :0.1806   Mean   :0.2388   Mean   : 9.934
##  3rd Qu.:0.5020   3rd Qu.:0.2530   3rd Qu.:0.3290   3rd Qu.:11.000
##  Max.   :1.4880   Max.   :0.7600   Max.   :1.0050   Max.   :29.000
```

We will investigate the age of the paua, so let's start by randomly dropping about 10% of the data. We will set a random seed, so that our output is repeatable for debugging purposes, and make a copy of the original data with NAs for 400 of the recorded values of the *Rings* variable.

```
nMissing <- 400
nData <- dim(abalone)[1]
thisData <- abalone
set.seed(1)
missingInd <- sample(1:nData,size=nMissing,replace=FALSE)
thisData$Rings[missingInd] <- NA
```

Estimation using complete data

Start by computing a 95% confidence interval for the mean age of the paua in the population using only complete data (be careful to account for the relationship between age and number of rings). My output is shown below - note that the output from the earlier summary function tells us the sample average is about 11.43 ($9.934 + 1.5$).

```
## [1] 11.31828 11.52393
## attr(,"conf.level")
## [1] 0.95
```

Estimation using mean imputation

Next, let's implement simple imputation to estimate the mean age, by making a copy of thisData, replacing the NAs with the mean, and recomputing the 95% confidence interval. My output is below.

```
## [1] 11.32813 11.51408
## attr(,"conf.level")
## [1] 0.95
```

Estimation using hotdeck imputation

Now try using random draws, conditioning on *Type*. We can do this by looping on the paua type, and replacing the missing values for each type by an appropriately sized draw with replacement from the non-missing values for that type. I have again set a seed to ensure you can verify your answer is correct.

```
set.seed(2)
thisData$Type <- as.factor(thisData$Type)
impData <- thisData$Rings
for (t in levels(thisData$Type)) {
  thisMissing <-  ***A***
  numDraws <- ***B***
  hotDeck <- ***C***
  impData[thisMissing]<-hotDeck
}
```

Note, code fragment `***A***` produces True if the corresponding row is of type t and has missing data, fragment `***B***` returns the number of missing data for type t, and fragment `***C***` performs the appropriate draw from the non-missing data (using the function sample()). The output is shown below:

```
## [1] 11.31159 11.50551
## attr(,"conf.level")
## [1] 0.95
```

Estimation using bootstrap replication

Having implemented a couple of standard imputation techniques, let's try bootstrap replication. We will generate 200 bootstrap replicates, impute missing data via random draws conditioning on Type, and thus include the effect of missing data in our 95% CI for mean age of the population. In the code that follows, missing code fragment `***A***` samples the appropriate number of row indices with replacement, for using to build the bootstrap replicate dataframe. Fragments `***B***`, `***C***` and `***D***` then repeat the hotdeck imputation steps as performed in the previous section.

```
set.seed(3)
numBoots <- 200
bootReps <- numeric(numBoots)
for (i in 1:numBoots) {
  thisBootInd <- ***A***
  thisBoot <- thisData[thisBootInd,]
  for (t in levels(thisData$Type)) {
    thisMissing <- ***B***
    numDraws <- ***C***
    hotDeck <- ***D***
    thisBoot$Rings[thisMissing]<-hotDeck
  }
  bootReps[i] <- mean(thisBoot$Rings+1.5)
}
```

To compute our confidence interval, we will use the estimate with hotdeck imputation as our "actual estimate", and build our confidence interval from that value using the quantiles of the bootstrap replicates, as shown in class.

```
actualEstimate <- mean(impData+1.5)
Iemp2 <- c(2*actualEstimate,2*actualEstimate)-quantile(bootReps,c(0.975,0.025))
Iemp2

##    97.5%     2.5%
## 11.28947 11.49047
```

Alterntaively we can build a confidence interval directly from our bootstrap replicates:

```
theta_boot <- (1/numBoots)*sum(bootReps)
theta_boot

## [1] 11.43038

V_boot <- (1/(numBoots-1))*sum((bootReps-theta_boot)^2)
V_boot

## [1] 0.00277278

Inorm <- c(theta_boot-qnorm(0.975)*sqrt(V_boot),theta_boot+
             qnorm(0.975)*sqrt(V_boot))
Inorm

## [1] 11.32717 11.53358
```

Task: Estimation using multiple imputation

Our final approach for building a confidence interval for the mean of the variable with missing data is to implement multiple imputation. We will condition on Type, and impute n=500 values for each missing value, to build a 95% CI for mean age of the population that reflects the uncertainty due to missing data.

```
set.seed(4)
nImps <- 500
MIEst <- numeric(nImps)
MIvar <- numeric(nImps)
for (i in 1:nImps) {
  impData <- thisData$Rings
  for (t in levels(thisData$Type)) {
    thisMissing <- ***A***
    numDraws <- ***B***
    hotDeck <- ***C***
    impData[thisMissing]<-hotDeck
  }
  MIEst[i] <- ***D***
  MIvar[i] <- ***E***
}
```

Note that code fragments ***A***, ***B*** and ***C*** again repeat the hotdeck imputation steps as performed in the previous sections. Fragment ***D*** computes the mean for iteration i. Fragment ***E*** computes the variance of our estimator (the mean). To compute this note that:

$$\begin{aligned} Var[\overline{X}] &= Var\left[\tfrac{1}{n}\textstyle\sum_1^n X_i\right] \\ &= \tfrac{1}{n^2}Var\left[\textstyle\sum_1^n X_i\right] \\ &= \tfrac{1}{n^2}\textstyle\sum_1^n Var\left[X_i\right] \\ &= \tfrac{1}{n^2}\textstyle\sum_1^n \sigma \\ &= \tfrac{1}{n}\sigma^2 \end{aligned}$$

Of course we don't know σ^2 as this is a population parameter, but an unbiased estimate for it is $\frac{1}{n-1}\sum_1^n(X_i - \overline{X})^2$. We complete the analysis by computing our estimate (***F***), the within imputation variance (***G***), the between imputation variance (***H***), the estimate of fraction of information lost due to missing data (***I***), the degrees of freedom (***J***) and the CI width (***K***).

```
ptEst <- ***F***
ptEst
```

```
## [1] 11.4245
```

```
WEst <- ***G***
WEst
```

```
## [1] 0.002486657
```

```
BEst <- ***H***
BEst
```

```
## [1] 0.0001774159
```

```
TEst <- WEst+((nImps+1)/nImps)*BEst

gamma <- ***I***
gamma
```

```
## [1] 0.06672003
```

```
df <- ***J***
wdth <- ***K***
c(ptEst-wdth,ptEst+wdth)
```

```
## [1] 11.33944 11.50955
```

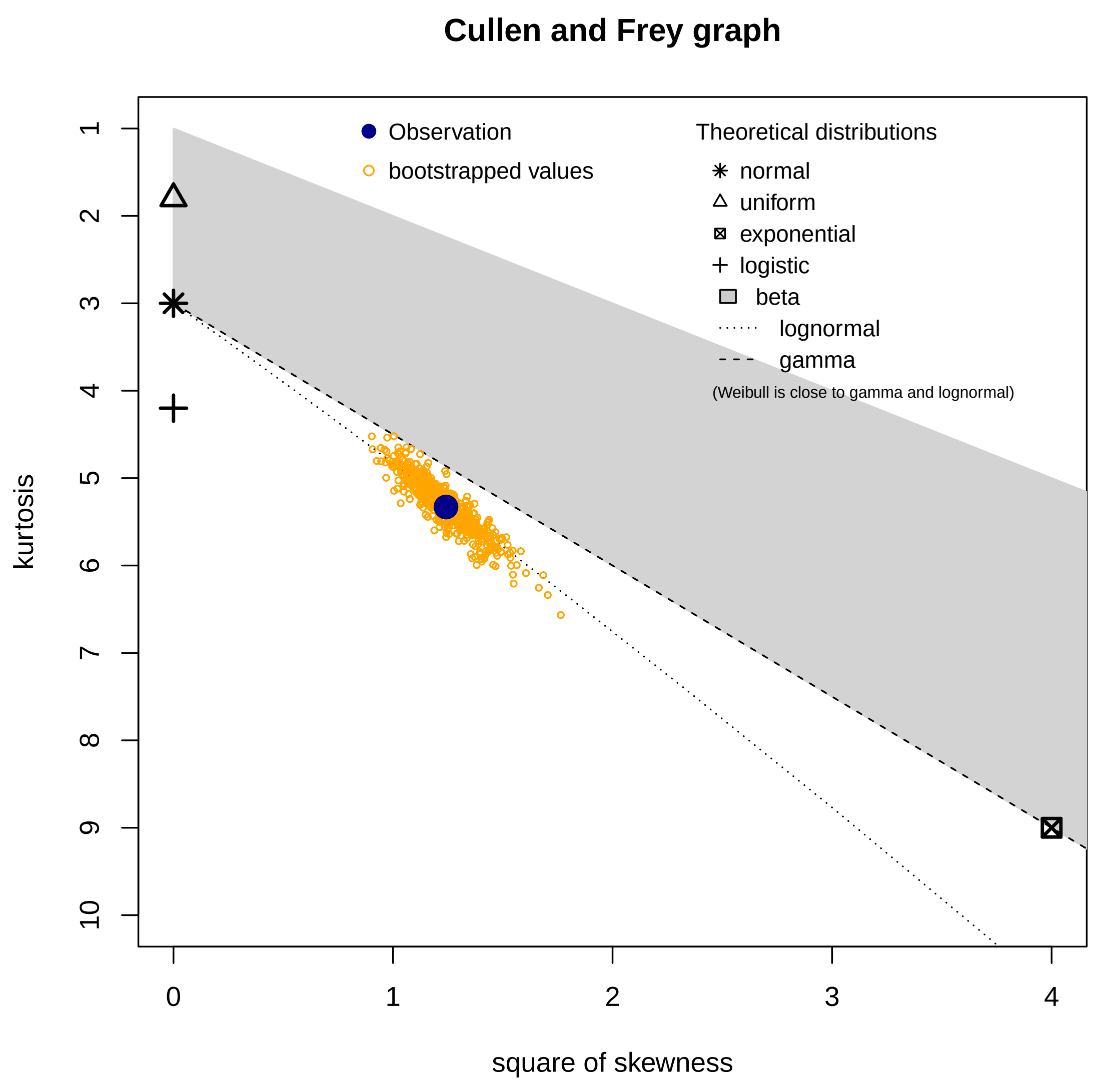
Distribution Fitting

We finish the lab with an example of data fitting via maximum likelihood. This section is not required for completion of the lab, it is an example of how to use the fitdistrplus package. Start by installing the `fitdistrplus` library.Note: it may be necessary to install `Rtools` first.

```
install.packages("fitdistrplus",
                 repo = "https://cran.stat.auckland.ac.nz/")
require(fitdistrplus)
```

Let's fit a distribution to the age of paua in our data. We begin with the Cullen and Frey graph (including bootstraps):

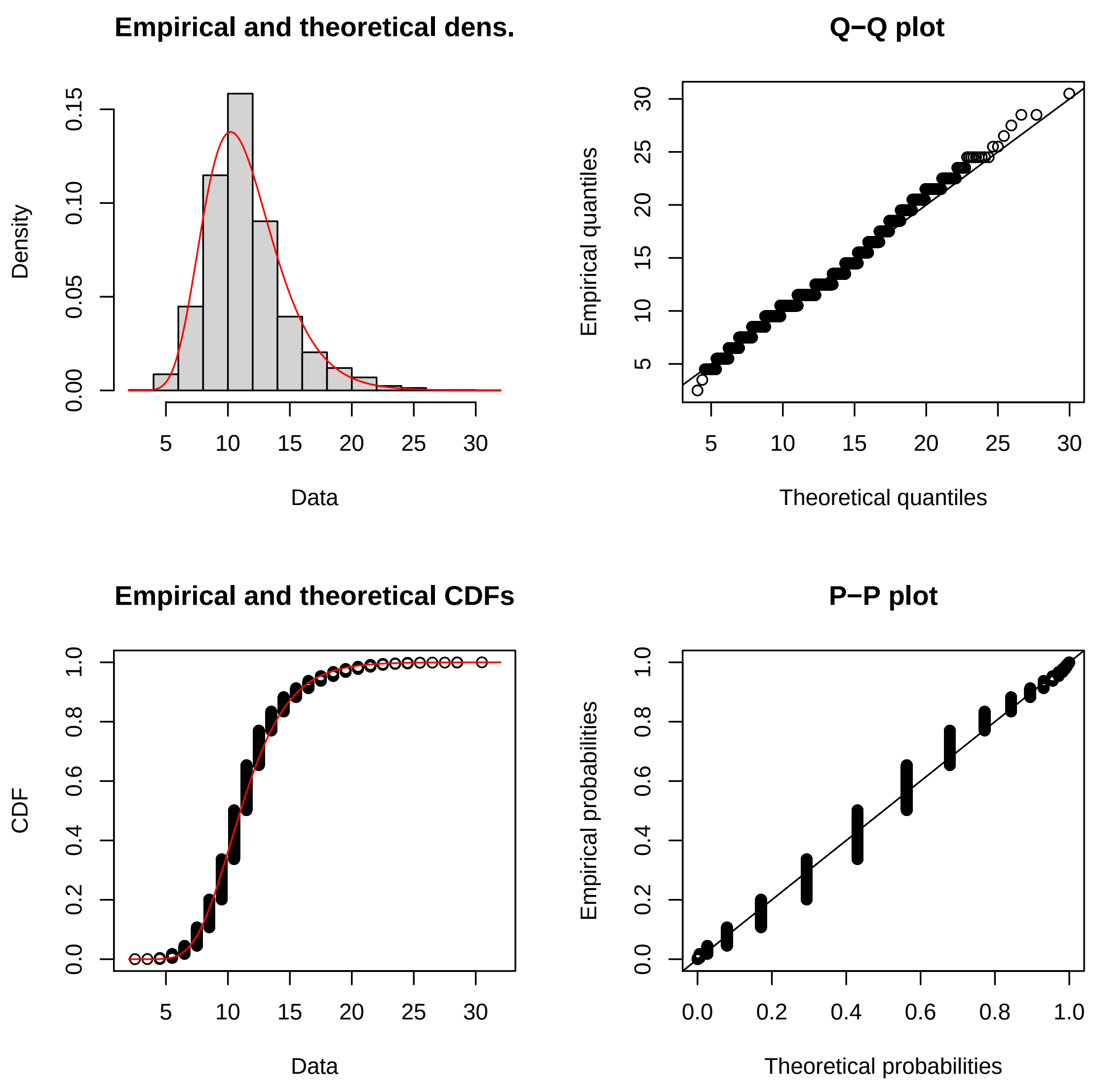
```
descdist(abalone$Rings+1.5,boot=500)
```



```
## summary statistics
## ------
## min:  2.5   max:  30.5
## median:  10.5
## mean:  11.43368
## estimated sd:  3.224169
## estimated skewness:  1.114102
## estimated kurtosis:  5.330687
```

It looks like lognormal may be the right distribution, so let's try that.

```
fitln <- fitdist(abalone$Rings+1.5,"lnorm",method="mle")
plot(fitln)
```

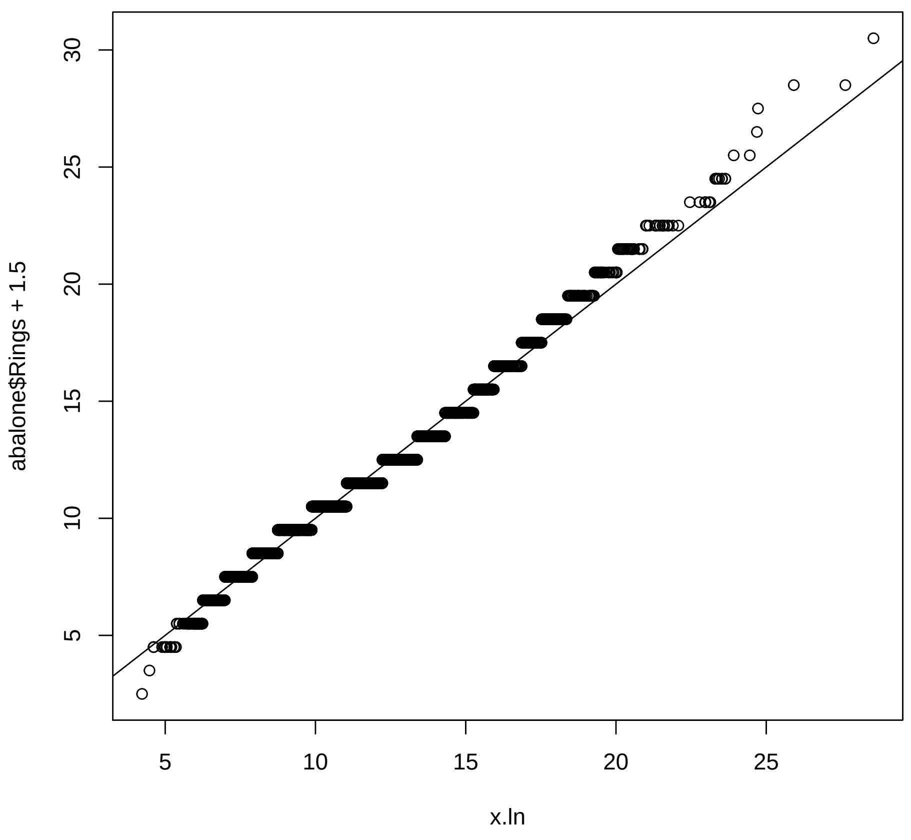


```
summary(fitln)

## Fitting of the distribution ' lnorm ' by maximum likelihood
## Parameters :
##           estimate  Std. Error
## meanlog 2.3992433 0.004217050
## sdlog   0.2725468 0.002981724
## Loglikelihood:  -10518.68   AIC:  21041.35   BIC:  21054.02
## Correlation matrix:
##         meanlog sdlog
## meanlog       1     0
## sdlog         0     1
```

The maximum likelihood estimates for the lognormal parameters (the mean and standard deviation on the log scale) are `meanlog` = 2.399 and `sdlog` = 0.273. These parameters give an acceptable fit. In practice it is easiest to evaluate this via a Q-Q plot (see plot above). We can produce one of these in the general case by simulating a synthetic data set and then plotting its quantiles against the quantiles of the actual data.

```
set.seed(7)
x.ln <- rlnorm(n=nData,meanlog=2.3992433,sdlog=0.2725468)
qqplot(x.ln,abalone$Rings+1.5)
abline(0,1)
```



The above plot is pretty good. Note that the actual data is only accurate to within a year, so the plot exhibits steps as the data jumps up a year.