Simple Statistical Inference I

This part of the notes introduce some simple statistical procedures for the inferential problems associated with one and two samples. The first part review Exploratory Data Analysis (EDA) which is a collection of graphical techniques for data inspection. In the sequel we will see some commonly used tests like the t-test and the Wilcoxon test for drawing inference about the parameters of interest.

Exploratory Data Analysis

EDA uses plots to help you understand whether or not the classical assumptions assumptions underlying statistical models are true. Some questions of interest include:

- Are the data follow a normal distribution?
- Are there any outliers in the data?
- If the data were collected over time, is there any evidence of serial correlation?

The basic graphical displays to help you gain insight to the data are the following:

- histograms,
- boxplots,
- density plots,
- qq-plots (quantile-quantile plots).

The following is a simple S-PLUS function that generated all four suggested displays:

```
eda.shape <- function(x)
 {
 par(mfrow = c(2, 2))
 hist(x)          #histogram
 boxplot(x)       #boxplot
```

```
  iqd <- summary(x)[5] - summary(x)[2]  #interquartile range (summary() returns
                                        #the five number summary
  plot(density(x,width=2*iqd), xlab = "x", #density plot
  ylab = "", type = "l")
  qqnorm(x)                                #qq-plots
  qqline(x)                                #deviations from normal
  }
```

To check about evidence for serial correlations, it is useful to do a time series plots and then plot the autocorrelation function of the data. Here is another function that can handle this situation:

```
eda.ts <- function(x)
 {
 par(mfrow=c(2,1))
 ts.plot(x)   #time series plot
 acf(x)       #autocorrelation function
 invisible()
 }
```

The output of these functions is displayed in Figures 1 and 2, respectively.

```
x <- rnorm(100)
eda.shape(x)
eda.ts(x)
```

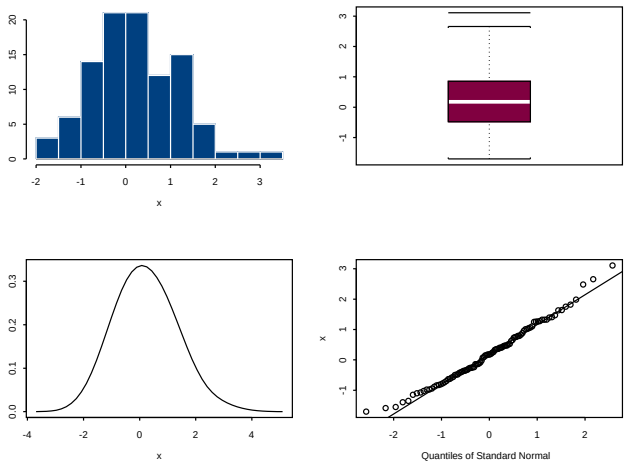

Figure 1: Output of eda.shape() function

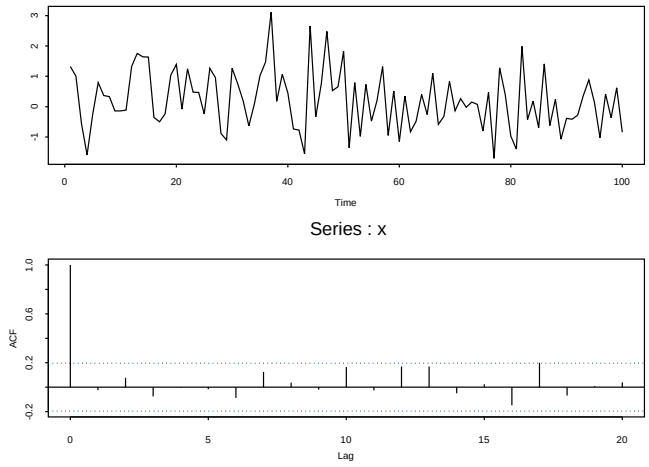

Figure 2: Output of eda.ts() function

One Sample Inference

Consider the data frame cats which consists of observations on the heart and body weights of samples of male and female cats used for an experiment. The data frame belongs to the library MASS (built by Venables & Ripley).

```
>library(mass)     #attach the library MASS
>cats             #list the data sets
>male   <-  cats[cats$Sex=="M",]$Hwt  #extract the heart weight for the males
>female <-  cats[cats$Sex=="F",]$Hwt  #extract the heart weight for females
>eda.shape(male)
>t.test(male, mu=10)

    One-sample t-Test

data:  male
t = 5.1241, df = 96, p-value = 0
alternative hypothesis: true mean is not equal to 10
95 percent confidence interval:
 10.81030 11.83506
sample estimates:
 mean of x
  11.32268

>t.test(male, mu=10, conf.level=0.99)
```

```
    One-sample t-Test

data:  male
t = 5.1241, df = 96, p-value = 0
alternative hypothesis: true mean is not equal to 10
99 percent confidence interval:
 10.64431 12.00105
sample estimates:
 mean of x
  11.32268
```

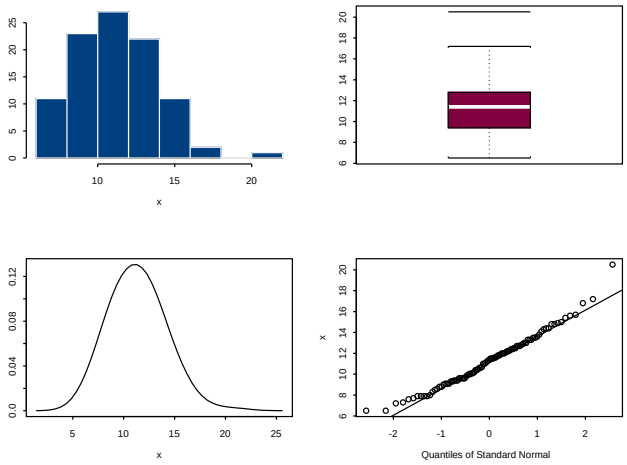

Figure 3: Output of eda.ts() function for the male weights

The output of `eda.shape()` suggests that it is reasonable to assume that the male heart weights follow the normal distribution where the mean is about 11 (see Figure 3). The function `t.test` can be used to test the null hypothesis that $\mu = \mu_0$ and the output shows that for $\mu_0 = 10$, the null hypothesis is rejected. Notice that the default output gives a 95% confidence interval but the `conf.level` can lead to any $(1 - \alpha)$% interval. To perform a nonparametric test use the `wilcox.test` function.

```
> wilcox.test(male, mu=10)
    Wilcoxon signed-rank test

data:  male
signed-rank normal statistic with correction Z = 4.4697, p-value = 0
alternative hypothesis: true mu is not equal to 10
```

Inference for Two Samples

Consider now the problem for testing the difference between two means based on a two sample problem. For instance, suppose that we are interested in the difference between the mean male and female heart weights.

```
> eda.shape(female)  #not far from normal.The plots are not displayed
> var.test(male, female) #Test for variance equality
    F test for variance equality

data:  male and female
F = 3.5064, num df = 96, denom df = 46, p-value = 0
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 2.075412 5.664645
sample estimates:
 variance of x variance of y
       6.46323      1.843256
> t.test(male, female)   #t-test with equal variances
    Standard Two-Sample t-Test

data:  male and female
t = 5.3539, df = 142, p-value = 0
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 1.337588 2.903517
sample estimates:
 mean of x mean of y
  11.32268  9.202128
> t.test(male, female, var.equal=F)   #t-test with unequal variances
    Welch Modified Two-Sample t-Test

data:  male and female
t = 6.5179, df = 140.608, p-value = 0
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 1.477352 2.763753
sample estimates:
 mean of x mean of y
  11.32268  9.202128
> wilcox.test(male,female)   #Wilcoxon test
    Wilcoxon rank-sum test

data:  male and female
rank-sum normal statistic with correction Z = 5.0309, p-value = 0
alternative hypothesis: true mu is not equal to 0
```

As we see, the variance equality test is rejected and therefore a formal application of the t-test does not apply to these data. However, the function `t.test` supports the argument `var.equal` which can be true or false. Therefore, an application of the t-test is possible and we reject the hypothesis of equal means. This conclusion is also supported by the nonparametric test of Wilcoxon.

Some Other Topics

Occasionally in applications we deal with paired data (X_i, Y_i). To perform a paired t-test, it is useful to define the observations $Z_i = X_i - Y_i$ and work as above.

To study the correlation between two variables it is helpful to explore their graphical relation and then use the function `cor.test()` to test whether the correlation coefficient is greater, less or different than 0. Here is a toy example

```
> x <- runif(20)
> y <- 2+3*x+rnorm(20)
> plot(x,y)     # x and y are positively related
> cor.test(x,y) # test whether the correlation coefficient is equal to 0
                # against the alternative it is not.
    Pearson's product-moment correlation

data:  x and y
t = 4.0049, df = 18, p-value = 0.0008
alternative hypothesis: true coef is not equal to 0
sample estimates:
       cor
 0.6864396
> cor.test(x,y, alt="l") # test whether the correlation coefficient is equal to 0
                         # against the alternative it is less.
    Pearson's product-moment correlation

data:  x and y
t = 4.0049, df = 18, p-value = 0.9996
alternative hypothesis: true coef is less than 0
sample estimates:
       cor
 0.6864396
```

We see that the first test is accepted while the second is rejected.