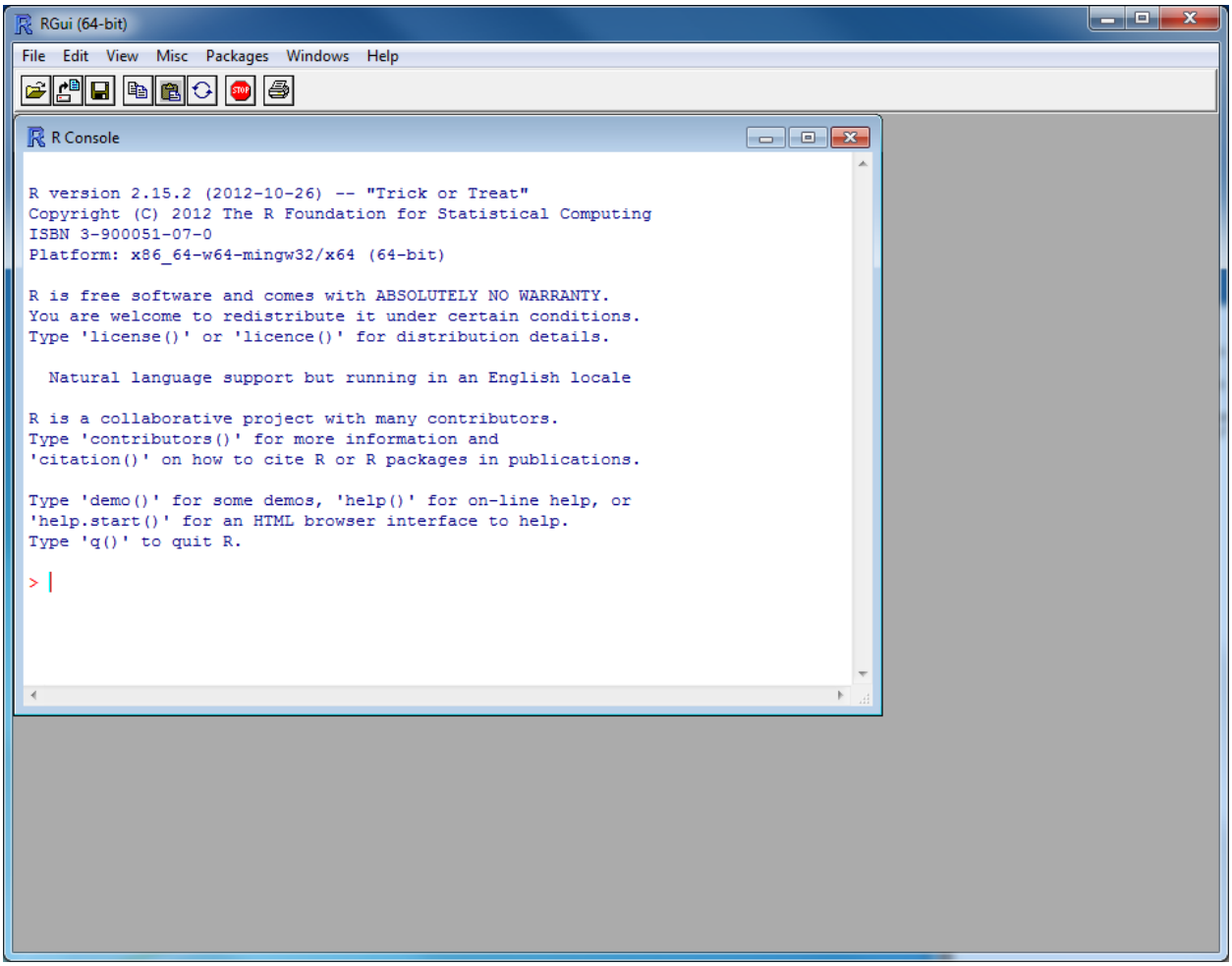
Summary of R software commands used to generate bootstrap and permutation test output and figures in Chapter 16

Since R is command line driven and the primary software of Chapter 16, this document details all commands used to generate Chapter 16's output and figures. To help familiarize yourself with the software, we suggest that you try to run these commands and compare your output to those in the chapter before trying to tackle the chapter exercises.

Below is a screenshot of R for Windows. To utilize functions from an R package, you need to



load the package. We'll assume you already have installed the **boot** package, so loading it involves the following set of clicks starting with the "Packages" item on the R menu.

Packages > Load Package > boot

Once you have the package loaded, you are now able to utilize the functions within this package.

Figure 16.1

First we need to load in the data set into R. R has a built in function "read.csv" that reads .csv files into R. We recommend that all Excel files to be read into R be saved as a .csv file. Once that is done, the following commands read the file into the data matrix "bc" and then attach it so that the columns can be referred to by their column names rather than referring to everything using rows and columns of "bc."

- bc <- read.csv("U:/IPS8/Datasets/chapter16/TIME50.csv")
- attach(bc)

The variable of interest here is **Time**. To get the mean and median, we can simply type the following commands. The output is shown below each command. Some of these basic summary statistic commands are helpful in checking that the data set has been read in correctly.

- mean(Time)

  [1] 23.26

- median(Time)

  [1] 12

The next set of commands generates the histogram and Normal quantile plot of Figure 16.1. The default histogram in R is one that displays the counts. To get the percentages, we alter the histogram heights to be the percentages and then use the **plot** command to create the histogram with appropriate axis labels. The Normal quantile plot utilizes the **qqnorm** function.

- h <- hist(Time)
- h$density = h$counts/sum(h$counts)
- plot(h,freq=F,main="",xlab="time (in days)",ylab="Percent",las=1)
- qqnorm(Time,main="",xlab="Normal score",ylab="times (in days)")

Figures 16.3 and 16.5

These are the first figures that use a function from the **boot** package. The **boot** function requires the sample statistic of interest be defined as an R function, so the first command defines the mean as an R function called **theta**. The second command requests that 3000 bootstrap sample means be generated from the variable **Time**. These 3000 sample means are saved in the data object named **time.boot**. By simply typing time.boot, you get a summary of the bootstrapping. That is what is shown in Figure 16.5.

```
theta <- function(data,indices)  {
 d <- data[indices]
 mean(d)
 }
time.boot <- boot(Time,theta,R=3000)
time.boot
```

Figure 16.3 is generated by the following set of commands. By requesting "prob=T" in the **hist** function, a density histogram (area of the histograms bars sums to 1) is generated. The Normal curve based on the 3000 generated samples is then drawn (need the mean and standard deviation of the bootstrap sample), followed by vertical lines representing the sample mean and bootstrap sample mean. For the Normal quantile plot, we also draw the line the observations should follow.

```
hist(time.boot$t,prob=T,axes=F,main="",xlab="mean times of resamples (in days)",ylab="")
axis(1)
jcc <- dnorm(seq(12,40,length=500),mean(time.boot$t),sqrt(var(time.boot$t)))
lines(seq(12,40,length=500),jcc)
lines(c(mean(Time),mean(Time)),c(0,dnorm(mean(Time),mean(time.boot$t),sqrt(var(time.boot$t)))),col=2)
afc <- mean(time.boot$t)
lines(c(afc,afc),c(0,dnorm(afc,mean(time.boot$t),sqrt(var(time.boot$t)))),lty=2)
qqnorm(time.boot$t,main="",xlab="Normal score",ylab="mean times of resamples (in days)")
qqline(time.boot$t, col = 2,lwd=1,lty=1)
```

A crude version of this plot can also be generated using the **plot** function. That command is simply

- plot(time.boot)

Figures 16.6 and 16.7

For this figure, we create our variable with the data rather than reading in a data set. This is only a viable option when the data set is small. The rest of the commands are similar to those used with the starting time data set.

- stout <- c(94,46,78,66,140,24)
- theta <- function(data,indices) {
  d <- data[indices]
  mean(d)
  }
- stout.boot <- boot(stout,theta,R=3000)
- stout.boot
- plot(stout.boot)

Figure 16.8

This figure is similar to Figure 16.1 so the commands are similar. If you compare these commands to those for Figure 16.1, only the variable names and data set names have changed.

- gpa <- read.csv("U:/IPS8/Datasets/chapter16/GPA.csv")
- attach(gpa)
- h = hist(GPA,nclass=20)
- h$density = h$counts/sum(h$counts)
- plot(h,freq=F,main="",xlab="GPA",ylab="Percent",las=1)
- qqnorm(GPA,main="",xlab="Normal score",ylab="GPA")

Figure 16.9 and accompanying output

For this figure and output, we are using the trimmed mean statistic. This is available within the **mean** function. This means we need to tweak the function **theta** that will be used within the boot function. The rest of the commands are similar to those used for Figure 16.3.

- mean(GPA, trim=0.25)

```
theta <- function(data,indices) {
 d <- data[indices]
 mean(d,trim=0.25)
 }
gpa.boot <- boot(GPA,theta,R=3000)
hist(gpa.boot$t,prob=T,axes=F,main="",xlab="means of resamples",ylab="",ylim=c(0,max(dnorm(mean(GPA,trim=0.25),mean(gpa.boot$t),sqrt(var(gpa.boot$t))))))
axis(1)
jcc <- dnorm(seq(2.5,3.5,length=500),mean(gpa.boot$t),sqrt(var(gpa.boot$t)))
lines(seq(2.5,3.5,length=500),jcc)
lines(c(mean(GPA,trim=0.25),mean(GPA,trim=0.25)),c(0,dnorm(mean(GPA,trim=0.25),mean(gpa.boot$t),sqrt(var(gpa.boot$t)))),col=2)
afc <- mean(gpa.boot$t)
lines(c(afc,afc),c(0,dnorm(afc,mean(gpa.boot$t),sqrt(var(gpa.boot$t)))),lty=2)
qqnorm(gpa.boot$t,main="",xlab="Normal score",ylab="means of resamples")
qqline(gpa.boot$t, col = 2,lwd=1,lty=1)
```

Example 16.6

To generate the values that appear in the table, we use the **length**, **mean**, and **var** functions. Since **sex** is coded 1=Male and 2=Female, we can restrict attention to GPA scores of just one sex by adding the sex specification within brackets.

```
mean(GPA[sex==1])
sqrt(var(GPA[sex==1]))
length(GPA[sex==1])
mean(GPA[sex==2])
sqrt(var(GPA[sex==2]))
length(GPA[sex==2])
```

Figure 16.10

A function from another package is needed to generate the density curves. The first command loads this package. This command works only if the package has already been installed. The function is called **sm.density.compare.** This function allows you to place a legend anywhere on the plot using a click of the mouse. The other commands set up the legend for this example. Make sure you click on the figure to place the legend. Until you do this, you will not get another command prompt.

The last two commands generate the Normal quantile plot. The first plots the data for the males (sex=1) and the last two add the data for the females (sex=2) to this plot.

```
library(sm)

# create value labels
cyl.f <- factor(sex, levels= c(1,2), labels = c("1 Male", "2 Female"))


# plot densities
sm.density.compare(GPA, sex, xlab="GPA",xlim=c(0,4))


# add legend via mouse click
colfill<-c(2:(2+length(levels(cyl.f))))
legend(locator(1), levels(cyl.f), fill=colfill)


qqnorm(GPA[sex==1],main="",xlab="Normal score",ylab="GPA",col=2)

bcc <- qqnorm(GPA[sex==2],main="",xlab="Normal score",ylab="GPA",plot=F)

points(bcc$x,bcc$y,col=3)
```

Figure 16.11 and accompanying output

The commands for this figure and output are similar to those used for the trimmed mean (Figure 16.9). The key is defining the function to be used with the **boot** function. Rather than call this function **theta**, we call it **meanDiff** here to describe the function. Because this example compares the means of two groups, the function involves two columns from the gpa matrix. Column 2 contains the GPAs and Column 9 contains the sex variable.

- meanDiff <- function(x, w){
  y <- tapply(x[w,2], x[w,9], mean)
  y[1]-y[2]
  }

- gpa1.boot <- boot(gpa,meanDiff,R=3000,strata=sex)

- gpa1.boot

- hist(gpa1.boot$t,prob=T,axes=F,main="",xlab="Difference in means of resamples",ylab="",ylim=c(0,max(dnorm(mean(gpa1.boot$t),mean(gpa1.boot$t),sqrt(var(gpa1.boot$t))))))

- axis(1)

- jcc <- dnorm(seq(-0.6,0.4,length=500),mean(gpa1.boot$t),sqrt(var(gpa1.boot$t)))

- lines(seq(-0.6,0.4,length=500),jcc)

- lines(c(-0.1490259,-0.1490259),c(0,dnorm(-0.1490259,mean(gpa1.boot$t),sqrt(var(gpa1.boot$t)))),col=2)

- afc <- mean(gpa1.boot$t)

- lines(c(afc,afc),c(0,dnorm(afc,mean(gpa1.boot$t),sqrt(var(gpa1.boot$t)))),lty=2)

- qqnorm(gpa1.boot$t,main="",xlab="Normal score",ylab="difference in means of resamples")

- qqline(gpa1.boot$t, col = 2,lwd=1,lty=1)

Example 16.8

The R command to get the 2.5 and 97.5 percentiles uses the **quantile** function. You specify the variable containing the bootstrap sample statistics and then a vector containing the percentiles.

- quantile(GPA.boot$t,c(.025,.975))

Example 16.9 and Figures 16.17 and 16.18

Similar to Figure 16.11 and accompanying output, which looked at the difference in the means of two groups, the function to be used in **boot** here is a bit more complicated. The function **ratvar** involves the second column, which contains the GPAs, and the ninth column, which contains the gender information. Rather than take a difference in means here, the statistic is a

ratio of variances. The other addition here is we utilize the function **boot.ci** to obtain the various confidence intervals based on our bootstrapping results. The remaining commands create the histogram in Figure 6.17.

- ratvar <- function(x, w) {
  y <- tapply(x[w,2], x[w,9], var)
  y[1]/y[2]
  }
- gpa2.boot <- boot(gpa,ratvar,R=5000,strata=sex)
- boot.ci(gpa2.boot)
- hist(gpa2.boot$t,prob=T,axes=F,main="",xlab="ratio of variances (male to female) of resamples",ylab="")
- axis(1)
- jcc <- dnorm(seq(0.5,3,length=500),mean(gpa2.boot$t),sqrt(var(gpa2.boot$t)))
- lines(seq(.5,3,length=500),jcc)
- lines(c(var(GPA[sex==1])/var(GPA[sex==2]),
  var(GPA[sex==1])/var(GPA[sex==2])),c(0,dnorm(var(GPA[sex==1])/var(GPA[sex==2]),mean(gpa2.boot$t),sqrt(var(gpa2.boot$t)))),col=2)
- afc <- mean(gpa2.boot$t)
- lines(c(afc,afc),c(0,dnorm(afc,mean(gpa2.boot$t),sqrt(var(gpa2.boot$t)))),lty=2)

Example 16.10 and Figure 16.20

For this example, we first need to read in the data set and attach it so we can refer to variables by their column names. After that, we define the function to be used for the bootstrapping. In this case, it is the correlation between the rating, which appears in column 3 of the data set, and the price, which appears in column 4. With this function defined, we then call the boot function to do the bootstrapping. The remainder of the commands generates the histogram and Normal quantile plot that we've seen before.

- bc <- read.csv("U:/IPS8/Datasets/CH16/LAUNDRY.csv")
- attach(bc)

```
theta <- function(data,indices){
d <- data[indices,]
 cov(d[,3],d[,4])/sqrt(var(d[,3])*var(d[,4]))
 }
corr.boot <- boot(bc,theta,R=5000)
hist(corr.boot$t,prob=T,axes=F,main="",xlab="Correlation coefficient of resamples",ylab="")
axis(1)
jcc <- dnorm(seq(0.25,.995,length=500),mean(corr.boot$t),sqrt(var(corr.boot$t)))
lines(seq(.25,.995,length=500),jcc)
lines(c(.6707535,.6707535),c(0,dnorm(.6707535,mean(corr.boot$t),sqrt(var(corr.boot$t)))),col=2)
afc <- mean(corr.boot$t)
lines(c(afc,afc),c(0,dnorm(afc,mean(corr.boot$t),sqrt(var(corr.boot$t)))),lty=2)
qqnorm(corr.boot$t,main="",xlab="Normal score",ylab="correlation coefficient")
qqline(corr.boot$t, col = 2,lwd=1,lty=1)
```

Software output of Example 16.12

For this example, we again enter the data into two variables directly rather than reading in a file. To do a permutation test, a different package is needed. The first command loads the **perm** package (assuming is has already been installed). The second and third commands enter the data. The function we need is called **permTS** and is controlled by a set of rules. Here we use the default rules except for the number of permutations, which we set to 5000.

```
library(perm)
trtgrp <- c(24,43,58,71,43,49,61,44,67,49,53,56,59,52,62,54,57,33,46,43,57)
ctrlgp <- c(42,43,55,26,62,37,33,41,19,54,20,85,46,10,17,60,53,42,37,42,55,28,48)
rules <- permControl(nmc=5000)
permTS(trtgrp,ctrlgp,alternative="greater",method="exact.mc",control=rules)
```

Example 16.13

For this example we compare the male and female GPAs using a permutation test. The commands will be very similar to those for Example 16.12. First the data set is read in. Then we call the function **permTS** using the "sex==" within brackets to separate the GPAs by gender.

- bc <- read.csv("U:/IPS8/Datasets/CH16/GPA.csv")
- attach(bc)
- rules <- permControl(nmc=5000)
- permTS(GPA[sex==1],GPA[sex==2],method="exact.mc",control=rules)