Exercise 3.1, page 9

Since $X_1, \ldots, X_N \overset{iid}{\sim} \mathcal{N}(\mu, \sigma^2)$,

$$\hat{\mu} \sim \mathcal{N}(\mu, \sigma^2/N). \tag{22.18}$$

Hypothesis tests for the mean are based on

$$\frac{\text{observed} - \text{hypothesized}}{\text{standard error}} = \frac{\hat{\mu} - \mu}{\sigma/\sqrt{N}} \sim \mathcal{N}(0, 1). \tag{22.19}$$

Replacing σ^2 by the sample estimate $\hat{\sigma}^2$ leads to the t-distribution:

$$t = \frac{\hat{\mu} - \mu}{\hat{\sigma}/\sqrt{N}} \sim t_{N-1}. \tag{22.20}$$

By definition of $t_{\alpha,N-1}$,

$$1 - \alpha = P\left(-t_{\alpha/2,N-1} \leq t < t_{\alpha/2,N-1}\right) \tag{22.21}$$

$$= P\left(-t_{\alpha/2,N-1} \leq \frac{\hat{\mu} - \mu}{\hat{\sigma}/\sqrt{N}} < t_{\alpha/2,N-1}\right) \tag{22.22}$$

$$= P\left(-t_{\alpha/2,N-1} \leq \frac{\hat{\mu} - \mu}{\hat{\sigma}/\sqrt{N}} \text{ and } \frac{\hat{\mu} - \mu}{\hat{\sigma}/\sqrt{N}} < t_{\alpha/2,N-1}\right) \tag{22.23}$$

$$\tag{22.24}$$

We want to solve for μ by itself:

$$1 - \alpha = P\left(\mu \leq \hat{\mu} + t_{\alpha/2,N-1}\hat{\sigma}/\sqrt{N} \text{ and } \hat{\mu} - t_{\alpha/2,N-1}\hat{\sigma}/\sqrt{N} < \mu\right) \tag{22.25}$$

$$= P\left(\hat{\mu} - t_{\alpha/2,N-1}\hat{\sigma}/\sqrt{N} < \mu \leq \hat{\mu} + t_{\alpha/2,N-1}\hat{\sigma}/\sqrt{N}\right). \tag{22.26}$$

An R code for computing the coverage of this confidence interval is

```
ntrials  = 100000
nsamp    = 20
mu       = 10
stdv     = 3
alpha    = 0.05

x        = array(rnorm(nsamp*ntrials,mean=mu,sd=stdv),dim=c(ntrials,nsamp))
x.mean   = rowMeans(x)
x.stdv   = sqrt(rowSums((x-x.mean)^2)/(nsamp-1)/nsamp)
tcrit    = qt(alpha/2,nsamp-1,lower.tail=FALSE)

mean.ci  = cbind(x.mean-tcrit*x.stdv,x.mean+tcrit*x.stdv)
coverage = sum(mu >= mean.ci[,1] & mu <= mean.ci[,2])/ntrials

print(paste('coverage=',coverage))
[1] "coverage= 0.94961"
```

The coverage is very close to 95%.

Exercise 3.2, page 10

An augmented `var.equal.test` that calculates a confidence interval is the following:

```
var.equal.test = function(data1,data2,alpha=0.05) {
### THIS FUNCTION TESTS EQUALITY OF VARIANCE OF TWO
### IID NORMALLY DISTRIBUTED RANDOM VARIABLES
###
# INPUT:
#   DATA1: [N1]-DIMENSIONAL VECTOR OF DATA
#   DATA2: [N2]-DIMENSIONAL VECTOR OF DATA
#   ALPHA: SIGNIFICANCE LEVEL OF THE TEST (DEFAULT = 5%)
# OUTPUT LIST:
#   F.MAX: RATIO OF VARIANCE, CONSTRUCTED TO BE GREATER THAN 1
#   F.CRIT: THE UPPER CRITICAL THRESHOLD OF SIGNIFICANCE
#   PVAL: P-VALUE OF THE F.MAX RATIO
#   VAR1: UNBIASED ESTIMATE OF THE VARIANCE OF DATA1
#   VAR2: UNBIASED ESTIMATE OF THE VARIANCE OF DATA2
#   RATIO: VAR1/VAR2
#   RATIO.LOWER: LOWER LIMIT OF ALPHA% CONFIDENCE INTERVAL FOR VARIANCE RATIO
#   RATIO.UPPER: UPPER LIMIT OF ALPHA% CONFIDENCE INTERVAL FOR VARIANCE RATIO

n1    = length(data1); n2    = length(data2)

mean1 = sum(data1)/n1; mean2 = sum(data2)/n2

var1  = sum((data1-mean1)^2)/(n1-1)
var2  = sum((data2-mean2)^2)/(n2-1)

if ( var1 > var2) {
  f.max  = var1/var2
  f.crit = qf(alpha/2,n1-1,n2-1,lower.tail=FALSE)
  pval   = 2*pf(f.max,n1-1,n2-1,lower.tail=FALSE)
} else {
  f.max  = var2/var1
  f.crit = qf(alpha/2,n2-1,n1-1,lower.tail=FALSE)
  pval   = 2*pf(f.max,n2-1,n1-1,lower.tail=FALSE)
}

f.upper = qf(alpha/2,n1-1,n2-1,lower.tail=FALSE)
f.lower = qf(alpha/2,n1-1,n2-1,lower.tail=TRUE)

ratio       = var1/var2
ratio.lower = ratio/f.upper
ratio.upper = ratio/f.lower

list(f.max=f.max,f.crit=f.crit,pval=pval,var1=var1,var2=var2,
  ratio=ratio,ratio.lower=ratio.lower,ratio.upper=ratio.upper)
}
```

Note that the critical F values `f.upper` and `f.lower` were recomputed after the `if` statement so that they correspond to the ratio `var1/var2` and not its reciprocal.

The output of this function for the PDO index is the following

```
> nbreak = which( year == 1977)
> pdo1 = pdo[1:nbreak]
> pdo2 = pdo[(nbreak+1):nyrs]
>
> ################################################
> ## TEST EQUALITY OF VARIANCE
> ################################################
> source(paste('var.equal.test.R',sep=""))
> print(var.equal.test(pdo1,pdo2))
$f.max
[1] 1.247894

$f.crit
[1] 2.074389

$pval
[1] 0.5522268

$var1
[1] 0.7420675

$var2
[1] 0.9260218

$ratio
[1] 0.80135

$ratio.lower
[1] 0.4043894

$ratio.upper
[1] 1.662312
```

Or if the data is entered in the opposite direction...

```
> var.equal.test(pdo2,pdo1)
$f.max
[1] 1.247894

$f.crit
[1] 2.074389

$pval
[1] 0.5522268

$var1
[1] 0.9260218

$var2
[1] 0.7420675

$ratio
[1] 1.247894

$ratio.lower
[1] 0.6015718

$ratio.upper
[1] 2.472864
```

Accordingly, the 95% confidence interval for the ratio of variances is (0.39, 1.63), or (0.61, 2.54), depending on how the two variances are ordered in the ratio. Note that this interval includes one, implying that the ratio is not significantly different from one at the 5% significance level, consistent with the p-value.

Exercise 3.3, page 10

A modified version of `cor.equal.test` that also computes a confidence interval is the following:

```
cor.equal.test = function(data1,data2,alpha=0.05) {
## THIS FUNCTION TESTS VANISHING CORRELATION BETWEEN
## TWO BI-VARIATE, NORMALLY DISTRIBUTED RANDOM VARIABLES
##
# INPUT:
#   DATA1: [N]-DIMENSIONAL VECTOR OF DATA
#   DATA2: [N]-DIMENSIONAL VECTOR OF DATA
#   ALPHA: SIGNIFICANCE LEVEL OF THE TEST (DEFAULT = 5%)
# OUTPUT LIST:
#   RHO: SAMPLE CORRELATION BETWEEN DATA1 AND DATA2
#   RHO.CRIT: 100*ALPHA% CRITICAL VALUE FOR THE CORRELATION
#   PVAL: P-VALUE OF THE STATISTIC RHO
#   RHO.LIMITS: [2] (1-ALPHA/2)*100% CONFIDENCE LIMITS OF THE CORRELATION

if ( length(data1) != length(data2)) stop('data sets are not the same length')

ntot    = length(data1)

mean1 = sum(data1)/ntot
mean2 = sum(data2)/ntot

ssr1  = sum((data1-mean1)^2)
ssr2  = sum((data2-mean2)^2)
sscx  = sum((data1-mean1)*(data2-mean2))

rho.hat = sscx / sqrt(ssr1*ssr2)

t.crit  = qt(alpha/2,ntot-2,lower.tail=FALSE)
rho.crit = t.crit / sqrt( ntot - 2 + t.crit^2)

tval = rho.hat * sqrt(ntot-2) / sqrt(1-rho.hat^2)
pval = 2*pt(abs(tval),ntot-2,lower.tail=FALSE)

z       = 0.5 * log((1+rho.hat)/(1-rho.hat))
zc      = qnorm(alpha/2,mean=0,sd=1/sqrt(ntot-3),lower.tail=FALSE)
z.limits = c(z - zc, z+zc)

rho.limits = (exp(2*z.limits)-1)/(exp(2*z.limits)+1)

list(rho=rho.hat,rho.crit=rho.crit,pval=pval,tval=tval,rho.limits=rho.limits)
}
```

Applying this function to the PDO data gives the following

```
> ## test vanishing correlation for first half
> l0 = 1:(length(pdo1)-1)
> pdo1.l0 = pdo1[l0]; pdo1.l1 = pdo1[l0+1]
> print(cor.equal.test(pdo1.l0,pdo1.l1))
$rho
[1] 0.05495659

$rho.crit
[1] 0.3808629

$pval
[1] 0.7854252

$tval
[1] 0.2751988

$rho.limits
[1] -0.3319908  0.4260723
```

```
> ## test vanishing correlation for 2nd half
> l0 = 1:(length(pdo2)-1)
> pdo2.l0 = pdo2[l0]; pdo2.l1 = pdo2[l0+1]
> print(cor.equal.test(pdo2.l0,pdo2.l1))
$rho
[1] 0.3836199

$rho.crit
[1] 0.3160319

$pval
[1] 0.01591744

$tval
[1] 2.526791

$rho.limits
[1] 0.07748101 0.62365106
```

```
> ## test vanishing correlation for whole period
> l0 = 1:(length(pdo)-1)
> pdo.l0 = pdo[l0]; pdo.l1 = pdo[l0+1]
> print(cor.equal.test(pdo.l0,pdo.l1))
$rho
[1] 0.4915623

$rho.crit
[1] 0.2404471

$pval
[1] 2.398144e-05

$tval
[1] 4.550883

$rho.limits
[1] 0.2850066 0.6544904
```

The first confidence interval includes 0, implying that the correlation is not significantly difference from zero. The second confidence interval does not include 0, but only marginally. The third confidence interval does not include 0, hence the correlation is significantly different from zero. All three conclusions are consistent with the corresponding hypothesis tests.

Exercise 3.4, page 11

A modified version of `mean.equal.test` that also computes confidence limits for the difference in means is the following:

```
mean.equal.test = function(data1,data2,alpha=0.05) {
## THIS FUNCTION TESTS EQUALITY OF MEANS OF TWO IID
## NORMALLY DISTRIBUTED RANDOM VARIABLES
##
# INPUT:
#   DATA1: [N1]-DIMENSIONAL VECTOR OF DATA
#   DATA2: [N2]-DIMENSIONAL VECTOR OF DATA
#   ALPHA: SIGNIFICANCE LEVEL OF THE TEST (DEFAULT = 5%)
# OUTPUT LIST:
#   DIFF.MEAN: DIFFERENCE IN MEANS (MEAN1 - MEAN2)
#   DIFF.MEAN.CRIT: 100*ALPHA% LEVEL CRITICAL VALUE OF THE DIFFERENCE IN MEANS
#   PVAL: P-VALUE OF THE T-STATISTIC
#   T: T-STATISTIC FOR DIFFERENCE IN MEANS
#   T.CRIT: 100*ALPHA% LEVEL CRITICAL VALUE OF THE T STATISTIC
#   MEAN1: ESTIMATE OF THE MEAN OF DATA1
#   MEAN2: ESTIMATE OF THE MEAN OF DATA2
#   SPOOL: POOLED ESTIMATE OF THE STANDARD DEVIATION
#   DIFF.MEAN.LIMITS: [2] (1-ALPHA)100% CONFIDENCE LIMITS
#     FOR THE DIFFERENCE IN MEANS

n1    = length(data1)
n2    = length(data2)
dof   = n1 + n2 - 2

mean1 = sum(data1)/n1
mean2 = sum(data2)/n2
diff.mean = mean1 - mean2

ssr1  = sum((data1-mean1)^2)
ssr2  = sum((data2-mean2)^2)

spool = sqrt((ssr1 + ssr2)/dof)

tval   = diff.mean/spool/sqrt(1/n1 + 1/n2)
pval   = 2* pt(abs(tval),dof,lower.tail=FALSE)
t.crit = qt(alpha/2,dof,lower.tail=FALSE)

diff.mean.crit = t.crit * spool * sqrt(1/n1 + 1/n2)

diff.mean.limits = diff.mean + c(-1,1)*diff.mean.crit

list(diff.mean=diff.mean,diff.mean.crit=diff.mean.crit,pval=pval,
   tval=tval,t.crit=t.crit,mean1=mean1,mean2=mean2,spool=spool,
   diff.mean.limits=diff.mean.limits)
}
```

Applying this to the PDO data gives

```
> print(mean.equal.test(pdo1,pdo2))
$diff.mean
[1] -1.156262

$diff.mean.crit
[1] 0.4537684

$pval
[1] 3.22733e-06

$tval
[1] -5.08751

$t.crit
[1] 1.996564

$mean1
[1] -0.7405952

$mean2
[1] 0.4156667

$spool
[1] 0.9223707

$diff.mean.limits
[1] -1.6100303 -0.7024935
```

The confidence limits do not include 0, so the difference in means is significantly different from zero. This is consistent with the hypothesis test.

Exercise 3.5, page 12

An R function for constructing a confidence interval for a correlation coefficient is the following:

```
cor.equal.boot = function(x,y,alpha=0.05,nboot=100000) {
## ESTIMATES (1-ALPHA)100% CONFIDENCE INTERVAL
## FOR THE CORRELATION COEFFICIENT USING BOOTSTRAP METHODS
## INPUT:
##  X[NSAMP]: FIRST DATA SET
##  Y[NSAMP]: SECOND DATA SET
##  ALPHA: EQUIVALENT SIGNIFICANCE LEVEL (DEFAULT = 5%)
##  NBOOT: NUMBER OF BOOTSTRAP SAMPLES (DEFAULT = 100000)
## OUTPUT:
##  CONFIDENCE LIMITS[2]
nsamp  = length(x)
if (length(y) != nsamp) stop('x and y must be same length')

npic   = sample(nsamp,nboot*nsamp,replace=TRUE)
x.boot = x[npic]
y.boot = y[npic]
dim(x.boot) = c(nboot,nsamp)
dim(y.boot) = c(nboot,nsamp)
x.boot = x.boot - rowMeans(x.boot)
y.boot = y.boot - rowMeans(y.boot)
cor.boot = rowSums(x.boot*y.boot)/sqrt(rowSums(x.boot^2)*rowSums(y.boot^2))
cor.boot.conf = quantile(cor.boot,probs=c(alpha/2,1-alpha/2))

cor.boot.conf
}
```

The result of running this function is:

```
nsamp   = 20
rho     = 0.5
nboot   = 100000
set.seed(310)
x         = rnorm(nsamp)
w         = rnorm(nsamp)
y         = rho * x + sqrt(1-rho^2)*w

rho.test1 = cor.equal.test(x,y)
rho.test2 = cor.test(x,y)

print(paste('Traditional Confidence interval:',
  paste(signif(rho.test1$rho.limits,4),collapse=', ')))
print(paste('Bootstrap confidence interval  :',
  paste(signif(cor.boot.conf,4),collapse=', ')))



[1] "Traditional Confidence interval: 0.3301, 0.8601"
[1] "Bootstrap confidence interval  : 0.328, 0.8761"
```

The bootstrap interval is nearly identical to the standard interval.