```
> #Solutions to S-Plus Worksheet 4.
> #Exercise 1
>  x <- seq(-3,3,0.01)
>  y <- exp(-abs(x))
>  plot(x,y,type="l",ylab="f(x)")
> title("Plot of the double exponential function in the range [-3, 3]")
># Exercise 2
> # First save the data PROB3.8.csv  to you working directory from Wattle and read in
> #Then read the file into a dataframe
> mussels <- read.csv("PROB3.8.csv",header=F, col.names=c("location","age","weight"))
>  attach(mussels)
> # Use logical expressions to subset the vectors
> age.l1 <- age[location==1]
> weight.l1 <- weight[location==1]
> age.l2 <- age[location==2]
>  weight.l2 <- weight[location==2]
> # create the output file
> plot(age.l2, weight.l2, type="b", pch="2", xlab="Age", ylab="Weight")
>  lines(age.l1, weight.l1, type="b", pch="1", lty=2)
>  title("Growth Characteristics of Mussels\nSouthwestern Virginia, USA")
> text(c(17,14),c(3.5,11), c("Location 1", "Location 2"))
> # This plots the two sets of points and "joins the dots", which
> # is a sensible approach in this instance, as the data is sorted by age
> # within each location:
>  reg.l1 <- lsfit(age.l1,weight.l1)
>  reg.l2 <- lsfit(age.l2,weight.l2)
> plot(age.l2, weight.l2, pch="2", xlab="Age", ylab="Weight")
```

```
> points(age.l1, weight.l1, pch="1")
>  abline(reg.l1$coef)
> abline(reg.l2$coef,lty=2)
>  title("Growth Characteristics of Mussels\nSouthwestern Virginia, USA")
> text(c(17,14),c(3.5,11), c("Location 1", "Location 2"))
> # Both locations show an increasing relationship between weight and age,
> # ie. mussels increase in weight as they get older. The mussels at
> # location 2 generally increase in weight faster than at location 1.
> #Exercise 3
>  x <- rnorm(50)
>  median(x)
[1] -0.1442079
>  mean(x)
[1] -0.1839359
> x[1] <-x[1] + 100
> median(x)
[1] -0.06532474
> mean(x)
[1] 1.816064
> # The median is only slightly affected by the addition of such a
> # distinct outlier, however the mean is definitely affected.
> # Note the above are only typical results, the results will change
> # slightly each time you repeat this simulation exercise!
>  means.from.normal <- rep(0, 100)
> medians.from.normal <- rep(0, 100)
> means.from.cauchy <- rep(0, 100)
>  medians.from.cauchy <- rep(0, 100)
```

```
> for(i in 1:100) {
+  x <- rnorm(50)
+  y <- rcauchy(50)
+  means.from.normal[i] <- mean(x)
+  medians.from.normal[i] <- median(x)
+  means.from.cauchy[i] <- mean(y)
+  medians.from.cauchy[i] <- median(y)}
> # Generating histograms, means and standard deviations. R makes it easy to combine multiple
#plots into one overall graph, using either the
> #par( ) or layout( ) function. With the par( ) function, you can include the option mfrow=c(nrows,
#ncols) to create a matrix of nrows x ncols plots that are filled in by row:
> par(mfrow=c(2,2))
>  hist(means.from.normal)
> hist(medians.from.normal)
>  hist(means.from.cauchy)
>  hist(medians.from.cauchy)
>  mean(means.from.normal)
[1] -0.003407404
> mean(medians.from.normal)
[1] -0.00541422
> mean(means.from.cauchy)
[1] 0.8033338
>  mean(medians.from.cauchy)
[1] -0.03417571
> median(means.from.normal)
[1] -0.01204789
> median(medians.from.normal)
[1] -0.01106449
```

```
>  median(means.from.cauchy)
[1] -0.1019886
> median(medians.from.cauchy)
[1] -0.03970603
>  sqrt(var(means.from.normal))
[1] 0.1376985
> sqrt(var(medians.from.normal))
[1] 0.1697628
> sqrt(var(means.from.cauchy))
[1] 11.15454
> sqrt(var(medians.from.cauchy))
[1] 0.2190563
> # All the above show that only the means of the samples from the cauchy
> # distribution are affected by the presence of outliers in the cauchy
> # distribution. The median is robust against the presence of these outliers.
>
> #Exercise 4
> # Using the "betachng()" function which was developed in Worksheet 3:
> betajck <- function(resp,pred) {
+  diff <- rep(0, length(resp))
+  n <- length(resp)
+  for(i in 1:n) {
+ diff[i] <- betachng(resp,pred,i)[2]}
+  (n-1)*mean(diff)}
> # To generalise this function, we would have to allow the user to pass in
> # their estimator, "theta()", as an argument to the function, so the
> # first line of the function would read:
```

```
> jcknfe <- function(data, theta) {
+ ##Also, we would have to replace the line with the "betachng()" function by:
+ diff[i] <- theta(data)-theta(data[-i])
+ ## So, the general jackknife function would look like:
>  jcknfe <- function(data,theta) {
+  diff <- rep(0, length(data))
+ n <- length(data)
+ for(i in 1:n) {
+ diff[i] <- theta(data)-theta(data[-i])}
+  (n-1)*mean(diff)
+  }
#example
x<-c(1,2,3,4)
> theta<-function(x){
+ y=mean(x)
+ y
+ }
> jcknfe(x,theta)


# Of course, this will only work if the input data is a vector. Try and
+ # think how you might modify the function even further to allow for either
+ # vector or matrix data input.
+
```