Stat 432 Homework 1 Solution

Assigned: Jan 24, 2019; Due: 11:59pm Feb 1, 2019

Question 1 (basic R)

Perform the following tasks on the `iris` dataset:

a. (1 point) We can do this in R as follows.

```
# load data
data(iris)
# check the labels
levels(iris$Species)
```

```
## [1] "setosa"     "versicolor" "virginica"
```

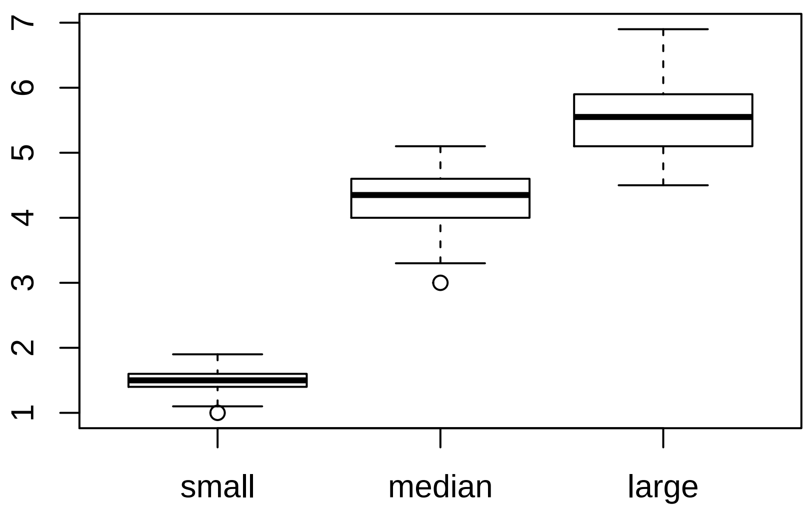
```
# change the labels
levels(iris$Species) <- c('small','median','large')
# check again
#head(iris)
```

b. (1 point) The variable name can be changed by `colnames` in R.

```
# change the variable name
colnames(iris)[colnames(iris)=='Species'] <- 'Size'
# check again
#head(iris)
```

c. (2 points) We can create the boxplot in R.

```
# create boxplot
boxplot(Petal.Length~Size,data=iris)
```



d. (2 points) Fit a linear model in R with `lm` function. Categorical variable `Size` will be treated as factor and associated dummy variables will be introduced.

```
# fit a linear model
linfit = lm(Petal.Length~., data=iris)
summary(linfit)
```

```
##
```

```
## Call:
## lm(formula = Petal.Length ~ ., data = iris)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.78396 -0.15708  0.00193  0.14730  0.65418
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.11099    0.26987  -4.117 6.45e-05 ***
## Sepal.Length  0.60801    0.05024  12.101  < 2e-16 ***
## Sepal.Width  -0.18052    0.08036  -2.246   0.0262 *
## Petal.Width   0.60222    0.12144   4.959 1.97e-06 ***
## Sizemedian    1.46337    0.17345   8.437 3.14e-14 ***
## Sizelarge     1.97422    0.24480   8.065 2.60e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2627 on 144 degrees of freedom
## Multiple R-squared:  0.9786, Adjusted R-squared:  0.9778
## F-statistic:  1317 on 5 and 144 DF,  p-value: < 2.2e-16
```

Seen from the summary report, `Sepal.Length` is the most significant variable. > Question 2 (a simple optimization)

a. (2 points) We can use the following R codes to define the target function.

```r
rosenbrock <- function(x) (1-x[1])^2 + 100*(x[2] - x[1]^2)^2
```

b. (2 points) After that, we can use `optim` fucntion to find the minimizer.

```r
res=optim(rep(0,2), rosenbrock)
res$par
```

```
## [1] 0.9999564 0.9999085
```

We see that the minimum is $(0.9999564, 0.9999085)$, with the minimal function varlue 3.7290519×10^{-9}.

c. (bonus, 3 points, 2 points for coding, 1 point for checking the result of minimum.) Here is an example of the implementation of the coordinate descent algorithm.

```r
x=rep(0,2);
f=rosenbrock(x);
eps=1e-20 # treshold to stop
while(1){
  x_=x; f_=f
  x[1]<- optim(x[1],function(x1)rosenbrock(c(x1,x[2])))$par
  x[2] <- optim(x[2],function(x2)rosenbrock(c(x[1],x2)))$par # replace x[1] with x_[1] for Jacobi imple
  f <- rosenbrock(x)
  if(abs(f-f_)<eps|norm(as.matrix(x-x_))<eps) break
}
x
```

```
## [1] 0.9999899 0.9999797
```

Therefore, the minimum we can get with the above coordinate descent algrithm is $(0.9999899, 0.9999797)$, with the minimal function varlue $1.0292359 \times 10^{-10}$.