Part1

Function that gets euclidean distance for a testrow

```
euc.dist <- function(testrow,traindata) {
  sum <- apply(((as.list(testrow)-traindata)^2),1,sum)
  return(sqrt(sum))
}
```

Function that gets corresponding y values given k

```
getY <- function(kDistDF, y) {
  n <- nrow(kDistDF)
  kYDist <- rep(NA, n)
  kYDist <- y[kDistDF[,1]]
  return(kYDist)
}
```

myknn function

```
myknn <- function(xtrain, xtest, ytrain, k) {
  nTest <- nrow(xtest)
  predicted <- vector('numeric')
#function that will be applied to xtest to get predicted values
  get.predic <- function(xtestrow,xtrain,ytrain,k) {
    eucDist<-as.vector(euc.dist(xtestrow,xtrain))
    #We identify each x row in training set with the row number
    numberedRows <- c(1:nrow(xtrain))
    eucDistDF <- data.frame(numberedRows,eucDist)
    sortDistDF <- eucDistDF[order(eucDistDF$eucDist),]
    kDistDF <- sortDistDF[1:k,]
    kYDist <- getY(kDistDF,y)
    p<-sum(kYDist)/k
    predicted <- c(predicted,p)
  }
  predicted <- apply(xtest,1,get.predic,xtrain=xtrain,ytrain=ytrain,k=k)
  return(as.vector(predicted))
}
```

Part2

Load the data first:

```
citibike_test <- read.csv("~/ProgramZ/stat365/HW1/citibike_test.csv")
citibike_train <- read.csv("~/ProgramZ/stat365/HW1/citibike_train.csv")
weather <- read.csv("~/ProgramZ/stat365/HW1/weather.csv")
```

STEP1-Merge data with weather and prepare test/val sets

```
new_training <- merge(citibike_train,weather,by="date")
y <- new_training$trips

#Transform data to get average temperature
new_training$TAVE <- (new_training$TMAX+new_training$TMIN)/2
original_training <- new_training
new_training <- new_training[,c(-1,-2,-4)]
new_test <- merge(citibike_test,weather,by="date")
new_test$TAVE <- (new_test$TMAX+new_test$TMIN)/2
testSet <- new_test[,c(-1,-3)]

#randomly split the training data into training and validation
set.seed(123)
ntrain <- nrow(new_training)
s <- sample(1:ntrain, (3*ntrain)/4, replace = FALSE)
trainSet <- new_training[s,]
valSet <- new_training[-s,]
```

STEP2-knn

```
#mse function
mse <- function(pred,actual) {
  e <- mean((pred-actual)^2)
  return(e)
}

require(FNN)
nK <- 750
kCount <- c(1:nK)
kDF <- data.frame(kCount,rep(NA,nK))
colnames(kDF) <- c("K","MSE")
for (i in 1:nK) {
  pred <- (knn.reg(trainSet,test=valSet,y=y,k=i))$pred
  e <- mse(pred,citibike_train[-s,]$trips)
  kDF$MSE[i] <- e
}
kDF <- kDF[order(kDF$MSE),]
rownames(kDF) <- NULL
head(kDF)
```

```
##     K       MSE
## 1 189 137092748
## 2 172 137136811
## 3 185 137147656
## 4 186 137184294
## 5 188 137191884
## 6 187 137206803
```

From step 2, we find that the k value which yields the lowest validation MSE is 189. We will use k=189:

```
#k we will use to get min mse:
k <- kDF[1,]$K
```

STEP3-linear regression

```
trainSet$trips <- citibike_train[s,]$trips
valSet$trips <- citibike_train[-s,]$trips

#try a number of different models:
formu <- c("trips~SNOW+TAVE","trips~SNWD+TAVE","trips~PRCP+TAVE","trips~SNOW+SNWD",
           "trips~SNOW+TAVE+AWND","trips~SNOW+TAVE+n_stations",
           "trips~PRCP+SNOW+TAVE","trips~SNOW+SNWD+TAVE",
           "trips~SNOW+SNWD+AWND","trips~PRCP+TAVE+AWND",
           "trips~PRCP+SNOW+SNWD+TAVE","trips~PRCP+SNOW+SNWD+TAVE+AWND")
lDF <- data.frame(formu,rep(NA,length(formu)))
colnames(lDF) <- c("formula","MSE")
for (i in 1:length(formu)) {
  m1 <- lm(as.formula(formu[i]), data=trainSet)
  p1.new <- predict(m1,newdata=valSet)
  p1.new.c <- predict(m1, newdata=valSet, interval="confidence", level=0.95)
  p1.new.p <- predict(m1, newdata=valSet, interval="prediction", level=0.95)
  e <- mean((p1.new-valSet$trips)^2)
  lDF$MSE[i] <- e
}
lDF <- lDF[order(lDF$MSE),]
rownames(lDF) <- NULL
head(lDF)
```

```
##                       formula       MSE
## 1            trips~SNOW+TAVE 127990025
## 2 trips~SNOW+TAVE+n_stations 128063719
## 3            trips~PRCP+TAVE 128465120
## 4       trips~SNOW+TAVE+AWND 128599756
## 5       trips~PRCP+SNOW+TAVE 128773427
## 6       trips~PRCP+TAVE+AWND 129076587
```

From step 3, we see that the linear models are producing a validation MSE that is lower than that of the knn method. We will use the linear model with the least validation MSE. It uses the predictors Snow and (the transformed variable) TAVE.

STEP4-Choose linear model (because of lower MSE) and predict for test set

```
#refit on entire training set
mfinal <- lm(trips~SNOW+TAVE,data=original_training)
pfinal <- predict(mfinal,newdata=testSet)
finalDF <- data.frame(new_test$date,pfinal)
colnames(finalDF) <- c("date","trips")
```