```
my.pred = myKnn(train = traindata, test = testdata, cl, k = 5)
test.pred = knn(traindata, testdata, cl, k=5)
print(table(Ytest, test.pred))
```

```
##      test.pred
## Ytest    0    1
##     0 3915 1085
##     1 1929 3071
```

```
print(table(Ytest, my.pred))
```

```
##      my.pred
## Ytest    0    1
##     0 3915 1085
##     1 1928 3072
```

```
print(table(test.pred, my.pred))
```

```
##           my.pred
## test.pred    0    1
##         0 5843    1
##         1    0 4156
```

Explain the mis-matches

Part 2: cvKNN

Results do not seem correct.
BTW, “KNN” should be “1NN” or “KNN (K=1)”

```
for (n in 1:nTrials) {
  m0 = matrix(rnorm(csize*p), csize, p)*sigma +
  cbind( rep(0, csize), rep(1, csize))
  m1 = matrix(rnorm(csize*p), csize, p)*sigma +
  cbind( rep(1, csize), rep(0, csize))
```

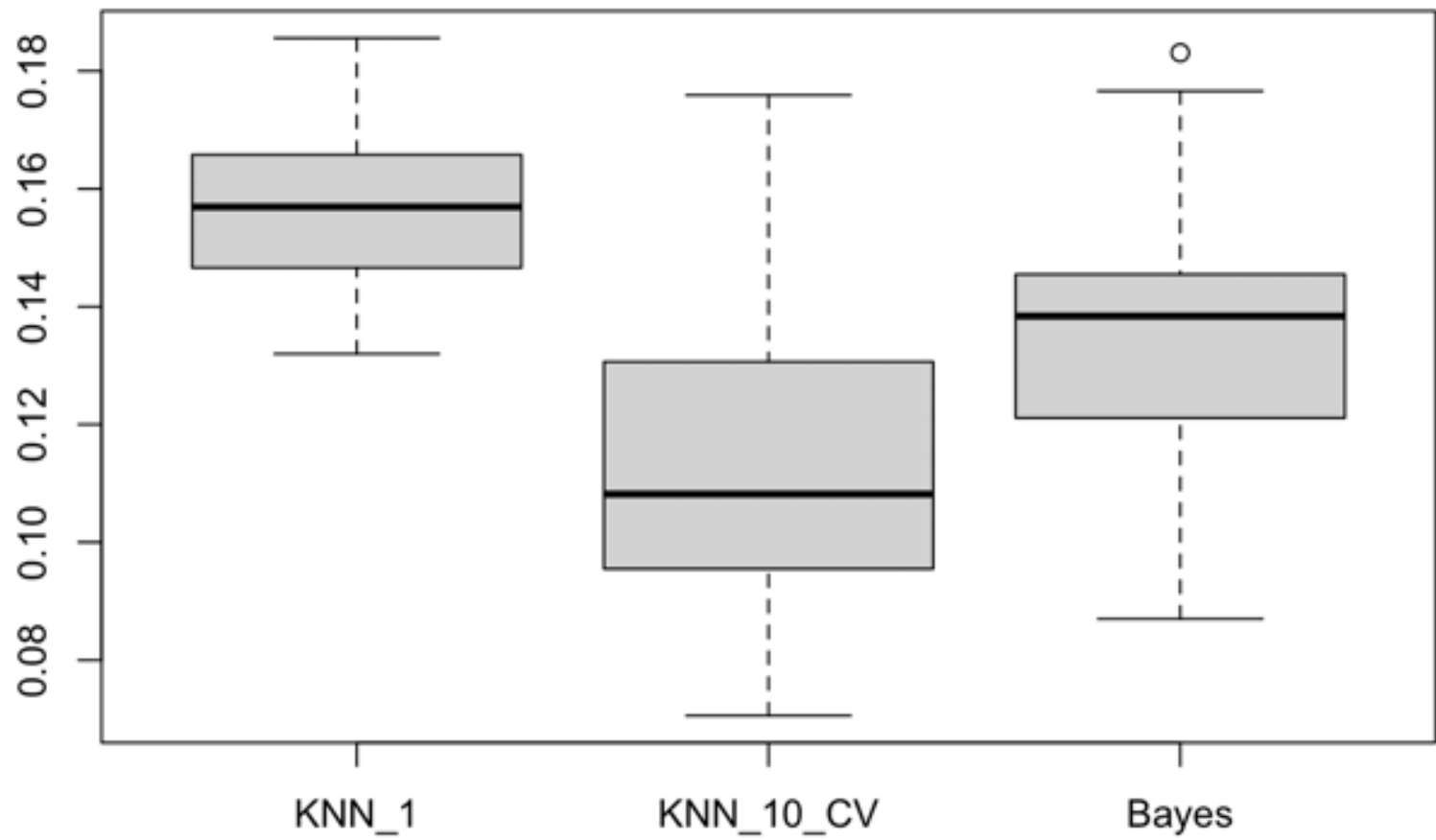
Generate m0 and m1 only ONCE.

When calculating the distances, I am using the order function to sort the distances in ascending order and then I select the k minimum values. So for the distance ties, if k = 3, I would select the 3 minimum distances which could all be the same for all we know - I didn't really have a special case if they were equal because those are all technically the 3 minimum values.

For the voting ties, we had odd k-values but if there was an equal number or greater number of observations (with a 1), I marked those as 1 because 1 usually denotes a true value.

---

1. Do not understand your explanation for voting ties
2. Your code should work when K is even, although you are not asked to show results with even k values.




```
# KNN with k chosen by 10-fold cross-validation
knn_cv = cvKNN(current_data$Xtrain, Ytrain, 10)
# calculate test error
knn_cv_errors[i] = knn_cv$cvError
# save chosen k val
k_vals[i] = knn_cv$bestK

# Bayes Rule - data prep
Xmin = min(current_data$Xtrain[, 1])
Xmax = max(current_data$Xtrain[, 1])
X1Vector = seq(Xmin, Xmax, (Xmax - Xmin)/99)
Xmin = min(current_data$Xtrain[, 2])
Xmax = max(current_data$Xtrain[, 2])
X2Vector = seq(Xmin, Xmax, (Xmax - Xmin)/99)
grid = expand.grid(X1Vector, X2Vector)
colnames(grid) = c('X1', 'X2')
# call Bayes
BayesRuleGrid = grid
BayesResults = BayesPredict(data.matrix(BayesRuleGrid), sim_params, FALSE)
# save error
bayes_errors[i] = BayesResults$error
```

Bayes Error is the error when applying the Bayes rule on the test data

Distance and Voting ties

This implementation of KNN calculates the euclidean distance between the training data and the test point and stores these distances (along with the associated labels) in a vector. It then sorts the vector by distance from low to high and picks the top K points.

In this logic, Voting Ties are defaulted to “zero winning”.

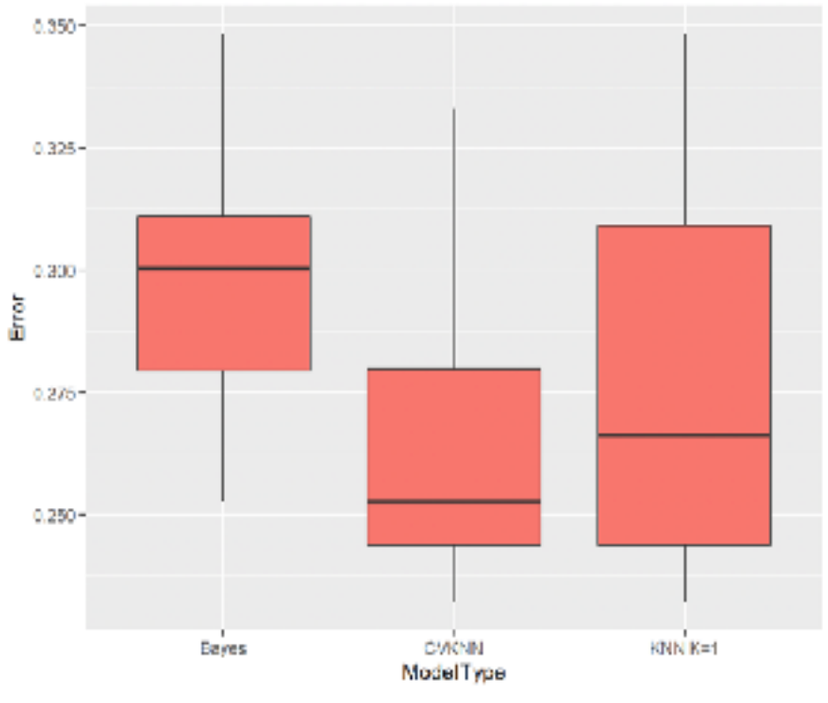
Distance Ties are equally counted as they will be placed next to each other when the distance vector is sorted.

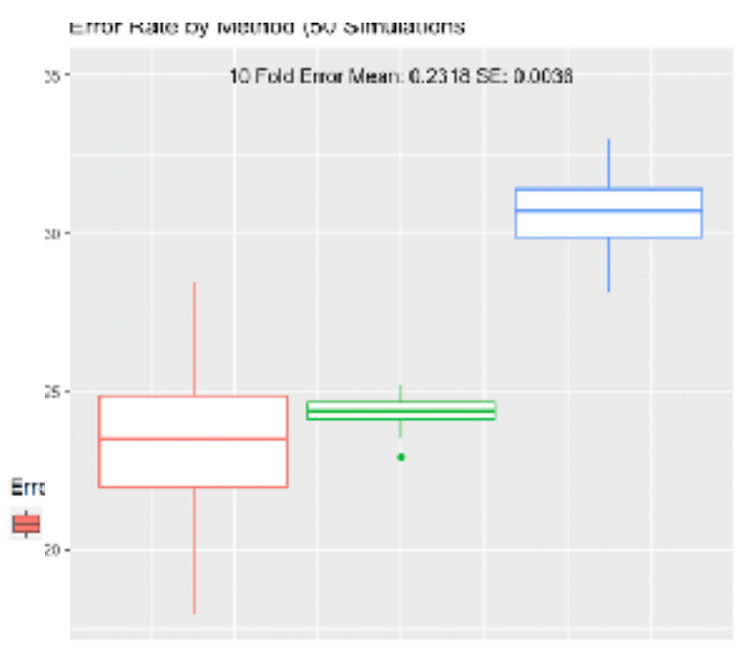
Do not understand your explanation for voting/distance ties

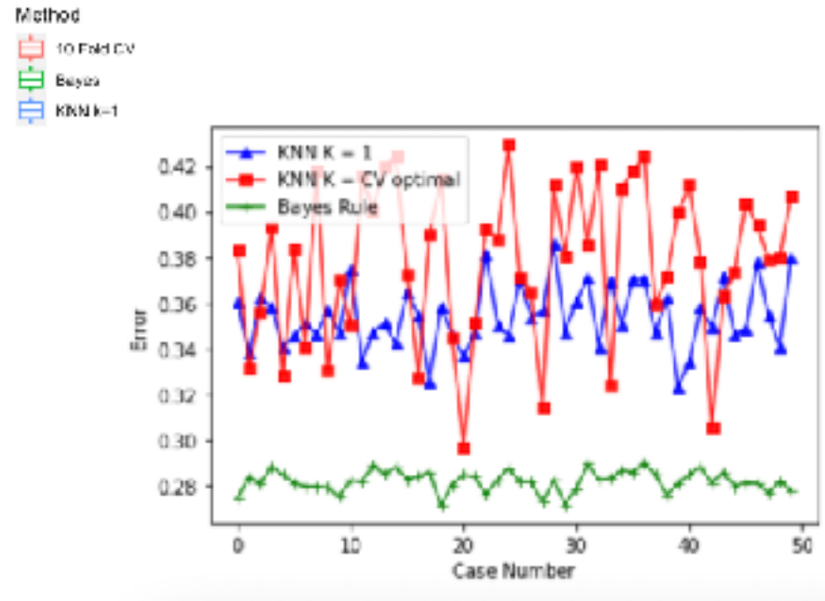
The default tolerance parameter `tol = 1e-4` mirrors the relative test for equality of distances that is hard-coded in `knn()`, such that the code first finds the k-th shortest distance, denoted by D. All points that are within $D \times (1 + 10^{-4})$ are included in the k-nearest-neighbors.

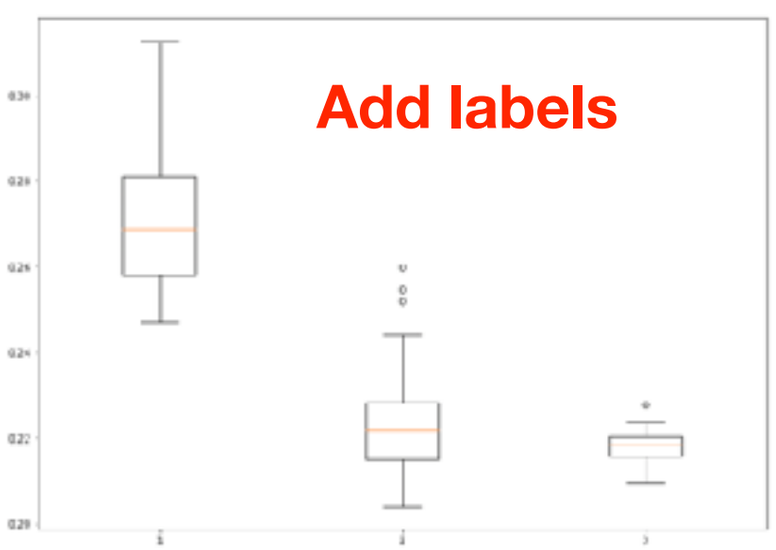
If no distance ties are desired, then set parameter `tol = 0`.

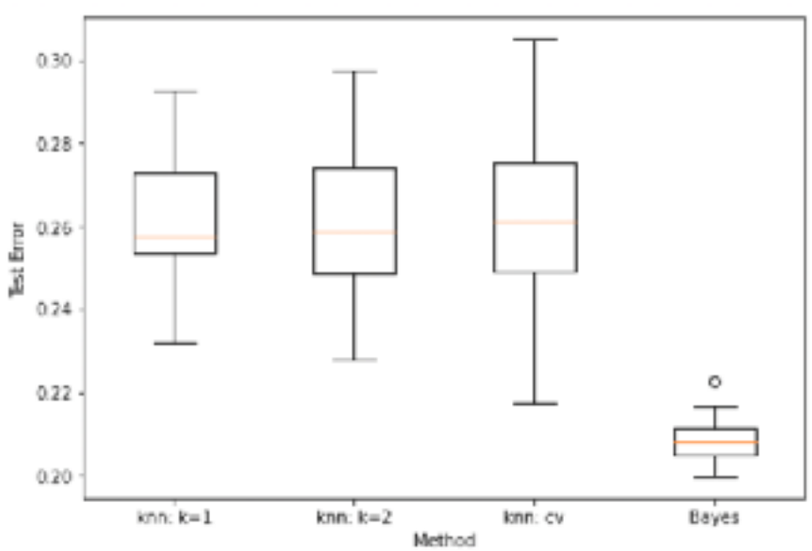
Do not understand your explanation for distance ties. You should have distance ties when tol = 0

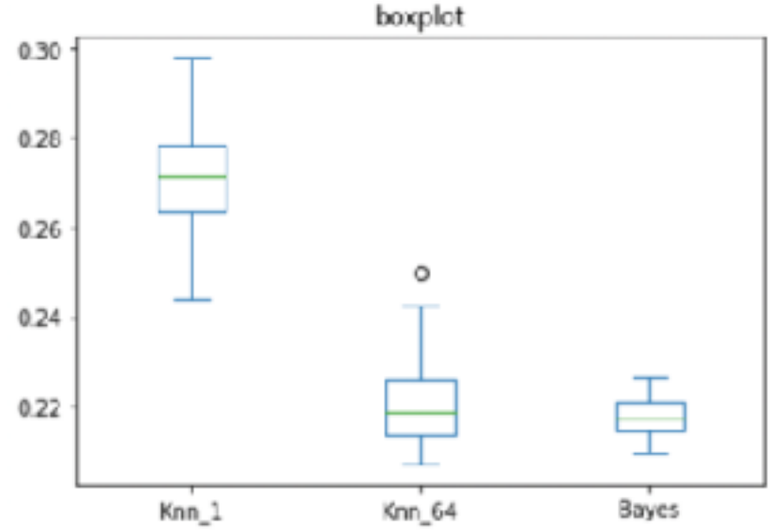












Results do not seem correct.

```
def bayesClassifier(x)
    d1 = sum(np.exp(-
    d0 = sum(np.exp(-
    ratio = d1/d0
    if ratio > 1/2:
        return 1
    else:
        return 0
```

Ratio > 1

```
r <-cvKNN(mydata$Xtrain, mydata$Ytrain, 10)
results[i, 'Ten_FoldK'] <- r$bestK
results[i, 'Ten_Fold'] <- r$cvError
```

You do not report cverror but test errors using bestK