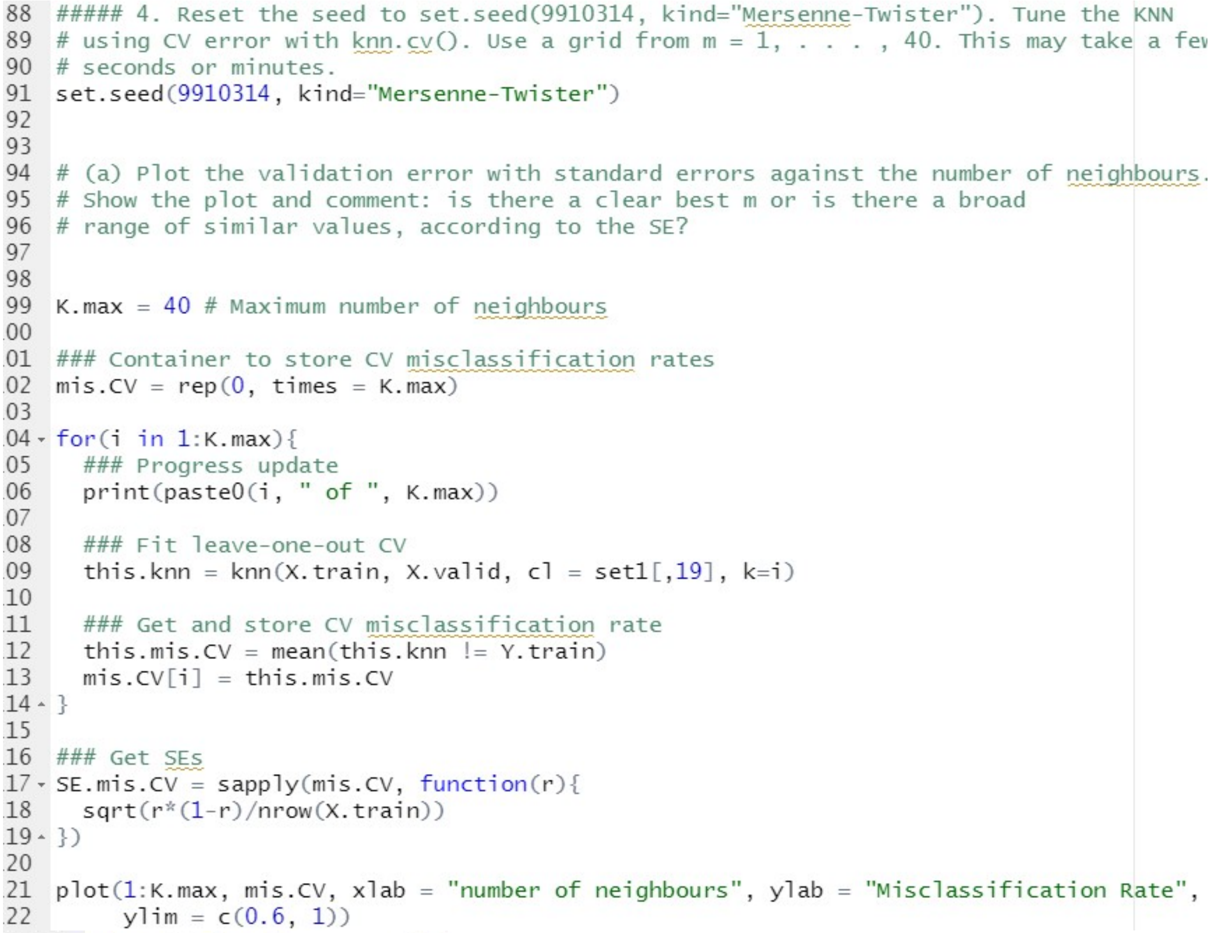
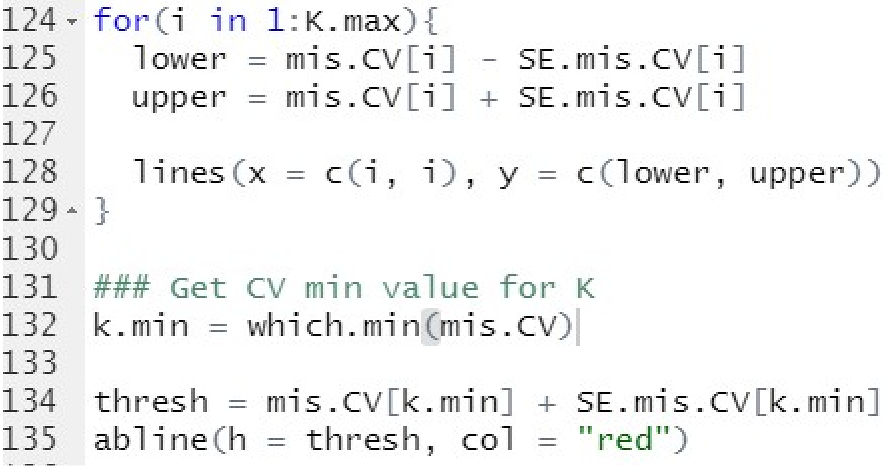
4. Reset the seed to set.seed(9910314, kind="Mersenne-Twister"). Tune the KNN using CV error with knn.cv(). Use a grid from *m* = 1, . . . , 40. This may take a few seconds or minutes.

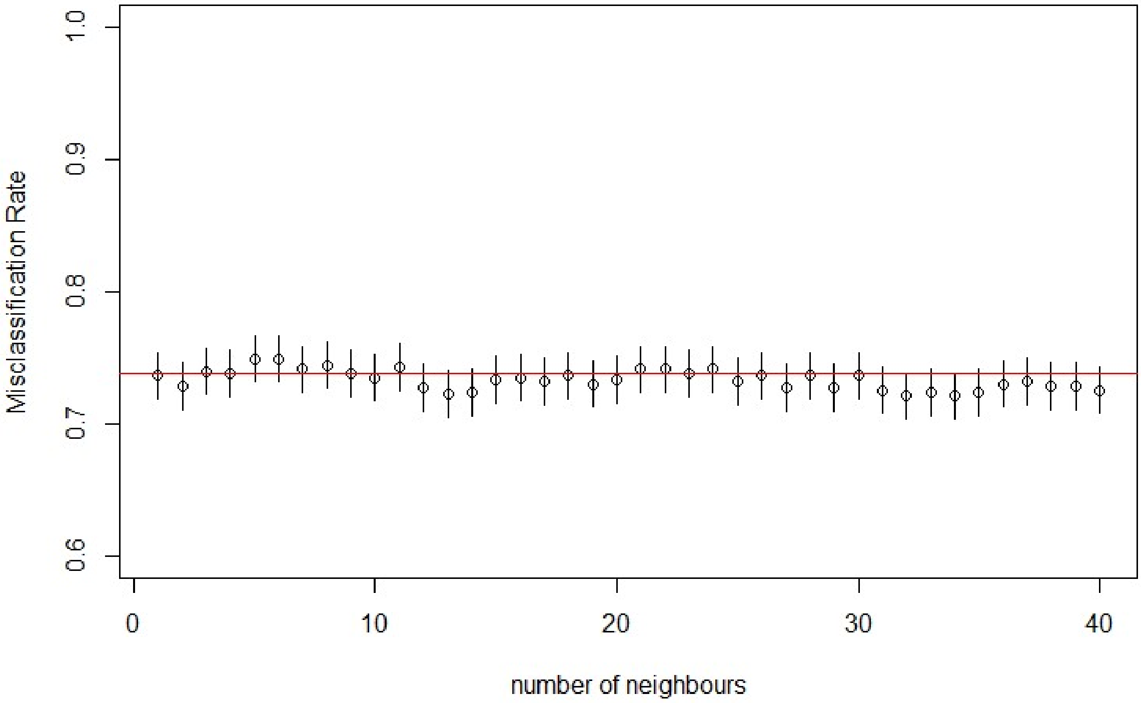
(a) Plot the validation error with standard errors against the number of neighbours. **Show the plot and comment: is there a clear best *m* or is there a broad range of similar values, according to the SE?**



```r
##### 4. Reset the seed to set.seed(9910314, kind="Mersenne-Twister"). Tune the KNN
# using CV error with knn.cv(). Use a grid from m = 1, . . . , 40. This may take a few
# seconds or minutes.
set.seed(9910314, kind="Mersenne-Twister")


# (a) Plot the validation error with standard errors against the number of neighbours.
# Show the plot and comment: is there a clear best m or is there a broad
# range of similar values, according to the SE?


K.max = 40 # Maximum number of neighbours

### Container to store CV misclassification rates
mis.CV = rep(0, times = K.max)

for(i in 1:K.max){
  ### Progress update
  print(paste0(i, " of ", K.max))

  ### Fit leave-one-out CV
  this.knn = knn(X.train, X.valid, cl = set1[,19], k=i)

  ### Get and store CV misclassification rate
  this.mis.CV = mean(this.knn != Y.train)
  mis.CV[i] = this.mis.CV
}

### Get SEs
SE.mis.CV = sapply(mis.CV, function(r){
  sqrt(r*(1-r)/nrow(X.train))
})

plot(1:K.max, mis.CV, xlab = "number of neighbours", ylab = "Misclassification Rate",
     ylim = c(0.6, 1))
```



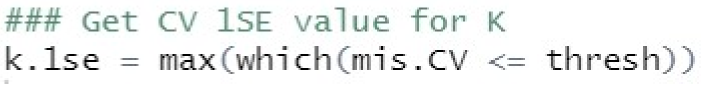
```r
for(i in 1:K.max){
  lower = mis.CV[i] - SE.mis.CV[i]
  upper = mis.CV[i] + SE.mis.CV[i]

  lines(x = c(i, i), y = c(lower, upper))
}

### Get CV min value for K
k.min = which.min(mis.CV)

thresh = mis.CV[k.min] + SE.mis.CV[k.min]
abline(h = thresh, col = "red")
```



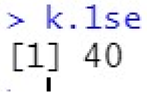
➔ The error is the lowest at k=32 but there is no clear difference.

(b) Report the value of *m* with lowest error, as well as the one selected by the 1SE rule.

M=32

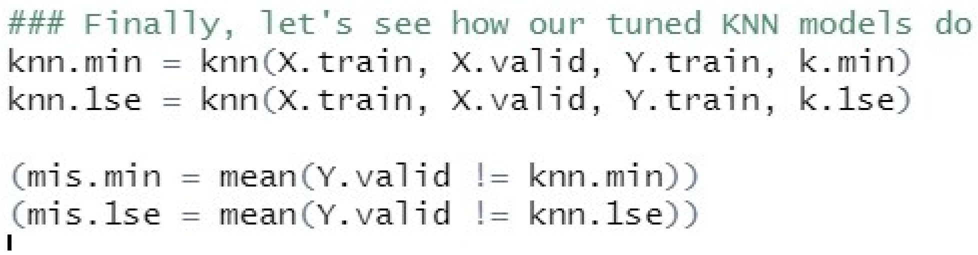


```
### Get CV 1SE value for K
k.1se = max(which(mis.CV <= thresh))
```

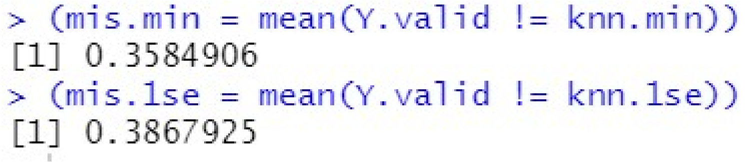


```
> k.1se
[1] 40
```

(c) Compute the test misclassification rate with both of these parameter values. **Report both error rates, using only one more digit than the first digit in their standard errors.** (For example, if the SE is 0.00375, the first digit in the SE 3 after the decimal, so report error to 4 after the decimal.)
Keep these error rates handy. You will use them for comparing all classification methods.
It would be better to use a more rigourous method like multiple reps of CV for computing error rates to make an “arena” for comparing methods. Classification is no different from regression in this way. However, this process is time consuming, and you have already practiced doing this on regression. I want to keep the classification assignments lighter, because they will accumulate rather quickly.



```
### Finally, let's see how our tuned KNN models do
knn.min = knn(X.train, X.valid, Y.train, k.min)
knn.1se = knn(X.train, X.valid, Y.train, k.1se)

(mis.min = mean(Y.valid != knn.min))
(mis.1se = mean(Y.valid != knn.1se))
```



```
> (mis.min = mean(Y.valid != knn.min))
[1] 0.3584906
> (mis.1se = mean(Y.valid != knn.1se))
[1] 0.3867925
```

- ➔ 2 after the decimal