DSII

Jun Lu

5/14/2019

data cleaning

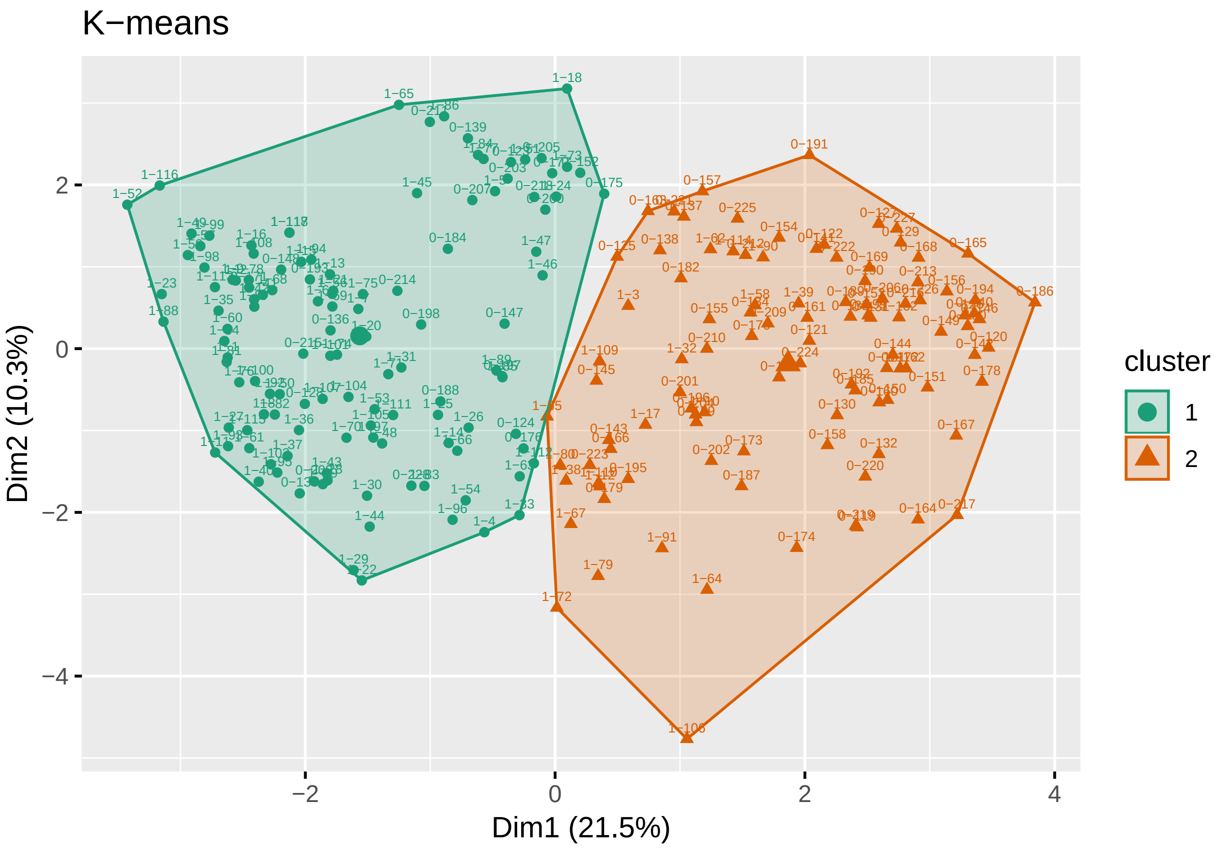
```r
heart_disease = read_csv("./data/heart.csv")
```

```
## Parsed with column specification:
## cols(
##   age = col_double(),
##   sex = col_double(),
##   cp = col_double(),
##   trestbps = col_double(),
##   chol = col_double(),
##   fbs = col_double(),
##   restecg = col_double(),
##   thalach = col_double(),
##   exang = col_double(),
##   oldpeak = col_double(),
##   slope = col_double(),
##   ca = col_double(),
##   thal = col_double(),
##   target = col_double()
## )
```

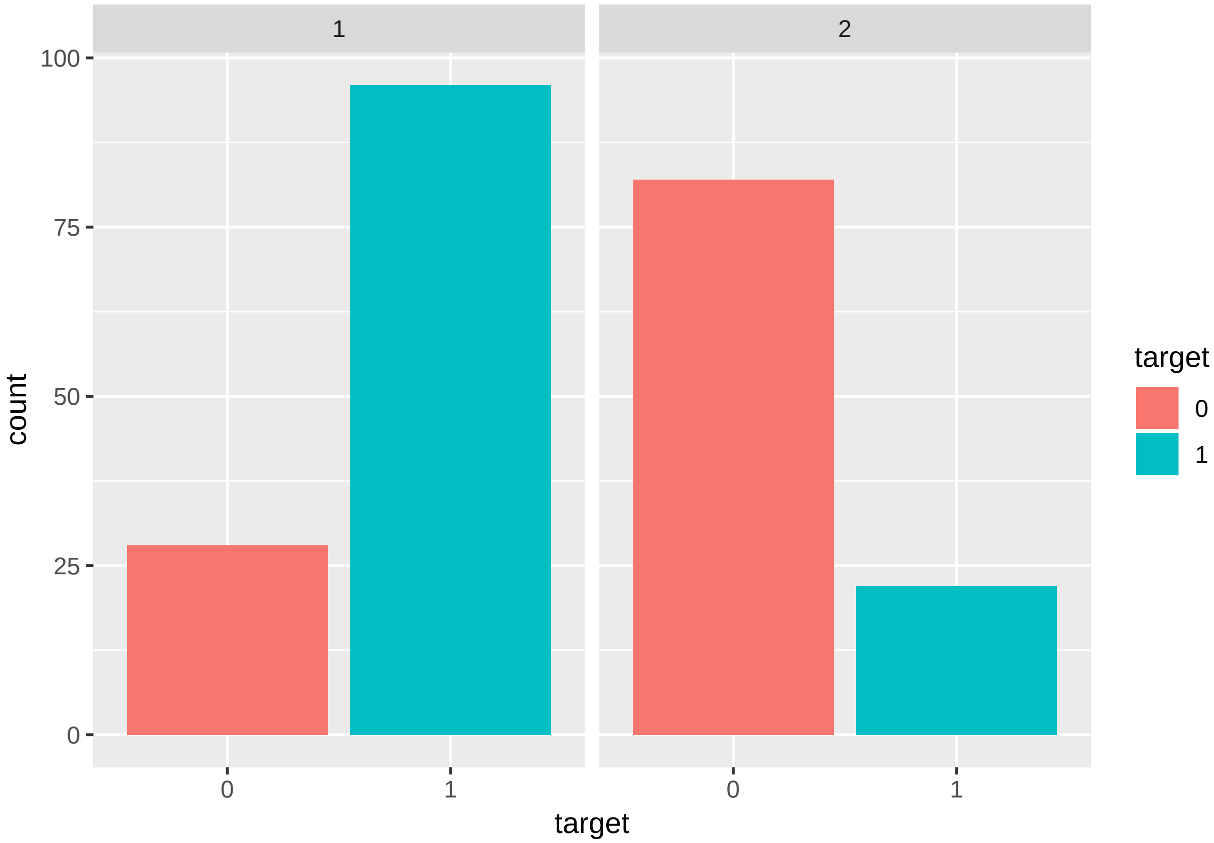
```r
set.seed(1)
trRows = createDataPartition(heart_disease$target, p = .75, list = FALSE)
train = heart_disease[trRows,]
test = heart_disease[-trRows,]

train = train %>%
    mutate(cp=as.factor(cp),
           restecg=as.factor(restecg),
           slope=as.factor(slope),
           thal=as.factor(thal))
train.x <- model.matrix(target~.,train)[,-1]
train.y <- train$target

test = test %>%
    mutate(cp=as.factor(cp),
           restecg=as.factor(restecg),
           slope=as.factor(slope),
           thal=as.factor(thal))

test.x <- model.matrix(target~.,test)[,-1]
test.y <- test$target

train = train %>% mutate(target=as.factor(target))
test = test %>% mutate(target=as.factor(target))
```

K-means

```
set.seed(1)

train.x_scale = scale(train.x)

rownames(train.x_scale) = paste(train$target, 1:228, sep = "-")


km = kmeans(train.x_scale, centers = 2, nstart = 20)
km_vis = fviz_cluster(list(data = train.x_scale,
                           cluster = km$cluster),
                      ellipse.type = "convex",
                      geom = c("point","text"),
                      labelsize = 5,palette = "Dark2") +
    labs(title = "K-means")

km_vis
```



```
train_kmeans = train
train_kmeans$kmean = km$cluster
train_kmeans %>% ggplot(aes(x = target, fill = target)) +
    geom_bar() +
    facet_grid(.~kmean)
```



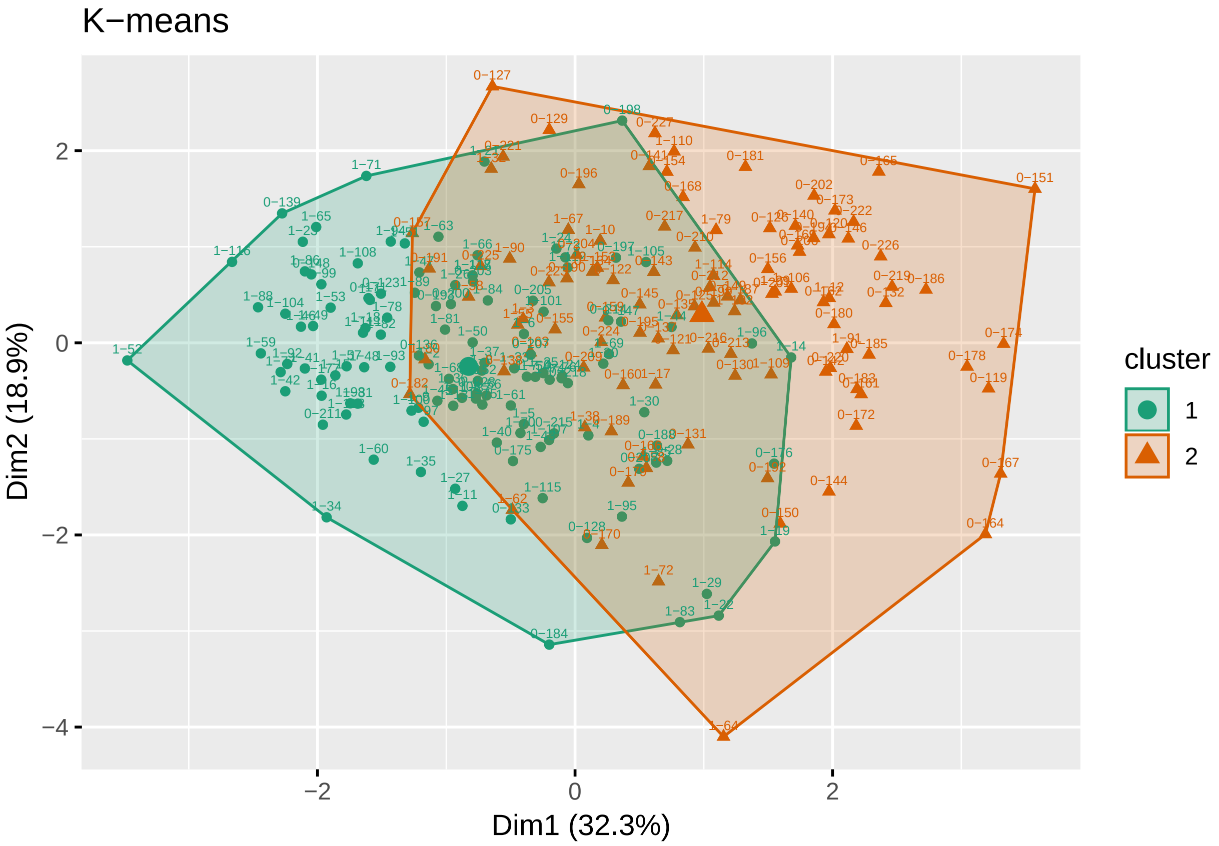
```
km$centers  # %>% knitr::kable()
```

```
##          age        sex        cp1        cp2         cp3   trestbps
## 1 -0.3026905 -0.1876911  0.3374140  0.1932071 -0.05360776 -0.1386814
## 2  0.3609002  0.2237856 -0.4023013 -0.2303623  0.06391694  0.1653509
##          chol         fbs   restecg1   restecg2     thalach      exang
## 1 -0.04357903 -0.05568027  0.1743137 -0.1152166  0.5362300 -0.4510651
## 2  0.05195961  0.06638802 -0.2078356  0.1373736 -0.6393512  0.5378083
##       oldpeak     slope1     slope2         ca      thal2      thal3
## 1 -0.5847214 -0.6229942  0.6881705 -0.2300021  0.4769883 -0.4190347
## 2  0.6971678  0.7428007 -0.8205110  0.2742333 -0.5687168  0.4996183
```

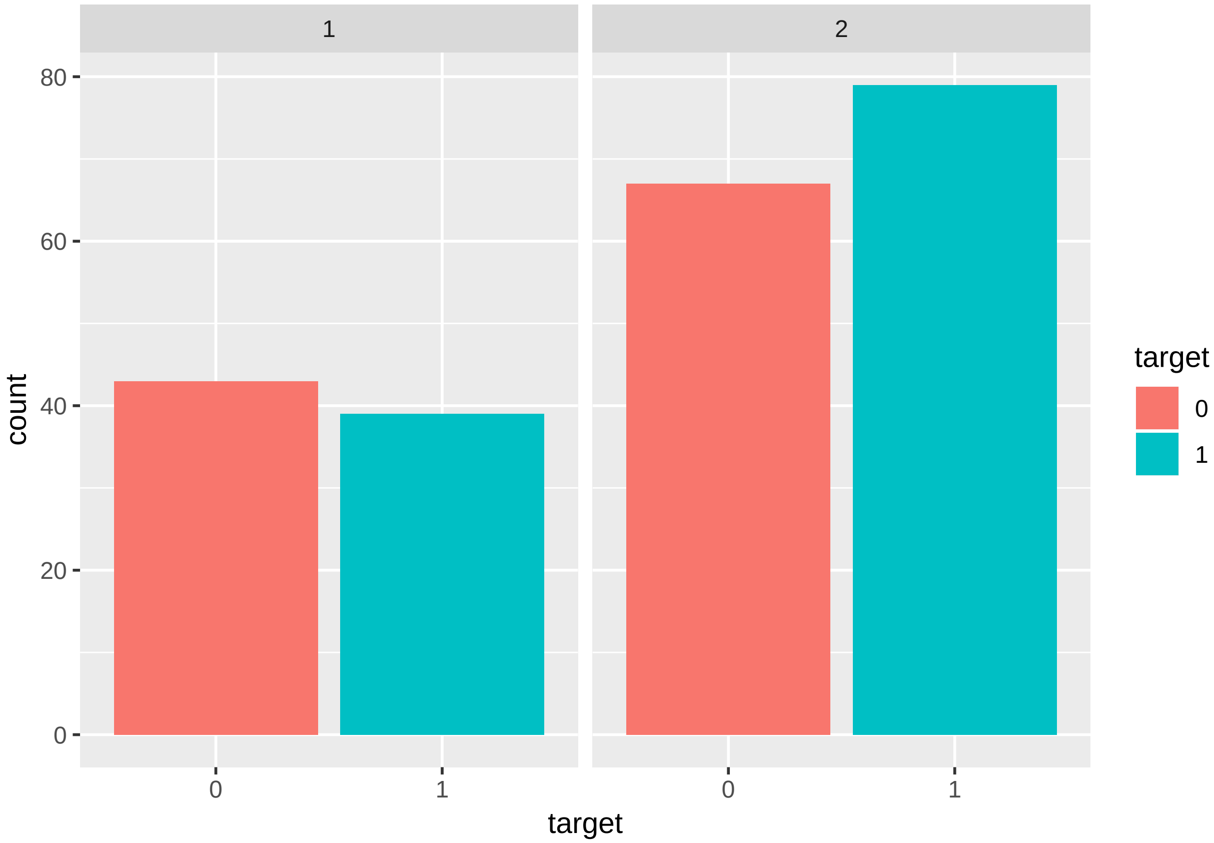
```
center = t(apply(km$centers, 1, function(r)r*attr(train.x_scale,'scaled:scale') + attr(train.x_scale, '
```

```
train_continu = train[c(1,4,5,8,10,12)]
set.seed(1)

train_continu_scale = scale(train_continu)

rownames(train_continu_scale) = paste(train$target, 1:228, sep = "-")

km_vis = fviz_cluster(list(data = train_continu_scale,
                            cluster = km$cluster),
                       ellipse.type = "convex",
                       geom = c("point","text"),
                       labelsize = 5,palette = "Dark2") +
    labs(title = "K-means")
```

```
km_vis
```



```
km_c = kmeans(train_continu, centers = 2, nstart = 20)
train_kmeans_c = train
train_kmeans_c$kmean = km_c$cluster
train_kmeans_c %>% ggplot(aes(x = target, fill = target)) +
    geom_bar() +
    facet_grid(.~kmean)
```



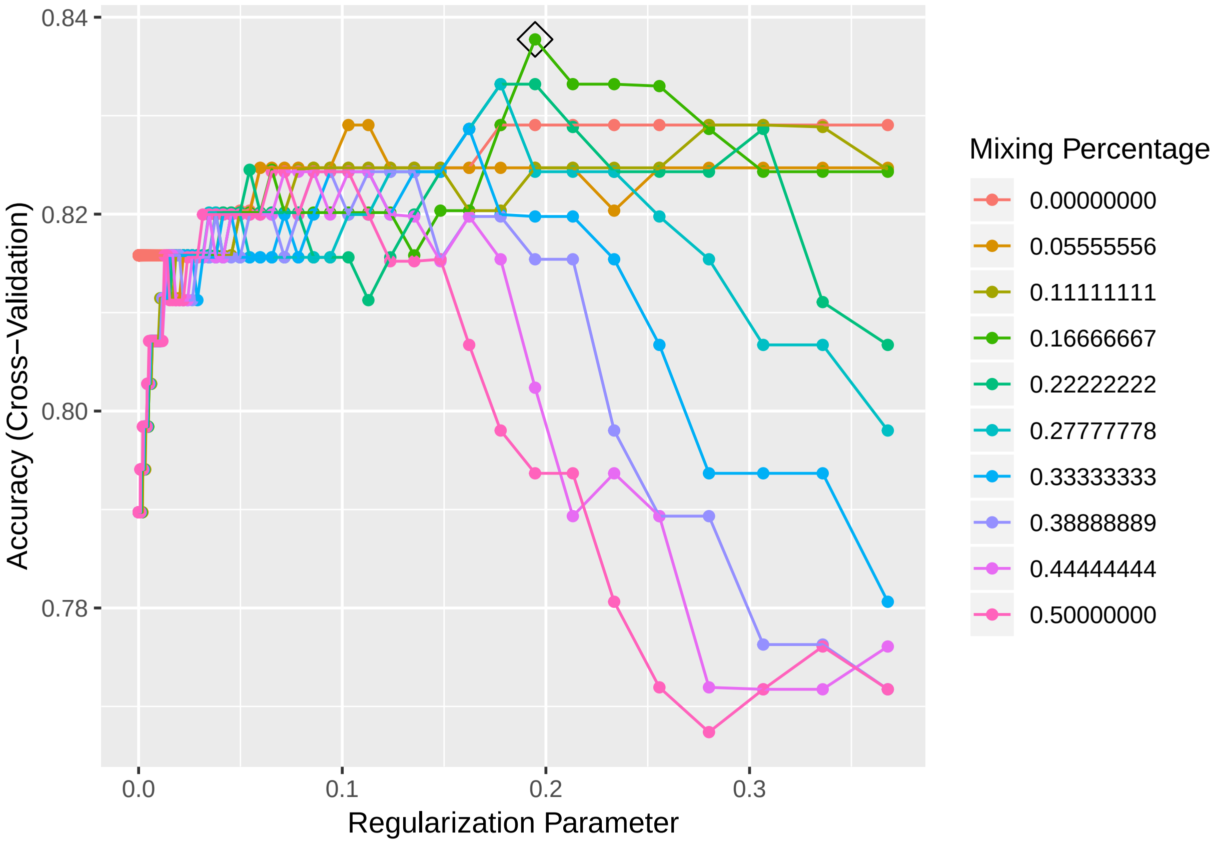
change to dummy ?can k-means apply to

Regularized logistic

```
train.y = factor(train.y, labels = c("absence","presence"))

ctrl = trainControl(method = "cv",
                    classProbs = TRUE)
set.seed(1)

glmnGrid <- expand.grid(.alpha = seq(0, 0.5, length = 10),
                        .lambda = exp(seq(-10,-1, length = 100)))

model.glm <- train(x = train.x,
                   y = train.y,
                   method = "glmnet",
                   tuneGrid = glmnGrid,
                   metric = "Accuracy",
                   trControl = ctrl)
```

```
ggplot(model.glm, highlight = T) +
    scale_shape_manual(values = rep(19,10),guide = FALSE)
```

```
## Scale for 'shape' is already present. Adding another scale for 'shape',
## which will replace the existing scale.
```



```
model.glm$bestTune
```

```
##         alpha    lambda
## 393 0.1666667 0.1946867
```

LDA

```
set.seed(1)
model.lda = train(x = train.x,
                   y = train.y,
                  method = "lda",
                  metric = "Accuracy",
                  trControl = ctrl)
```

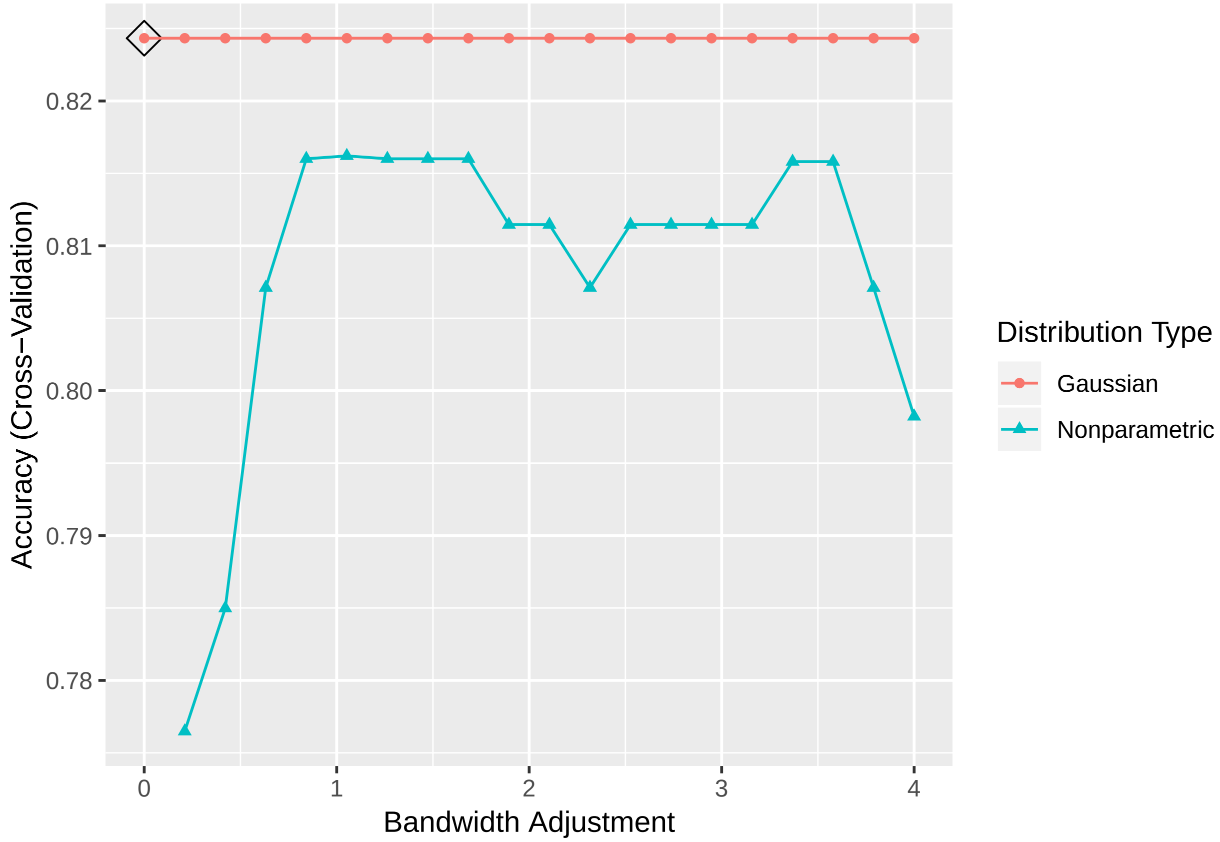
QDA

```
set.seed(1)
model.qda = train(x = train.x,
                   y = train.y,
                  method = "qda",
                  metric = "Accuracy",
                  trControl = ctrl)
```

lda qda

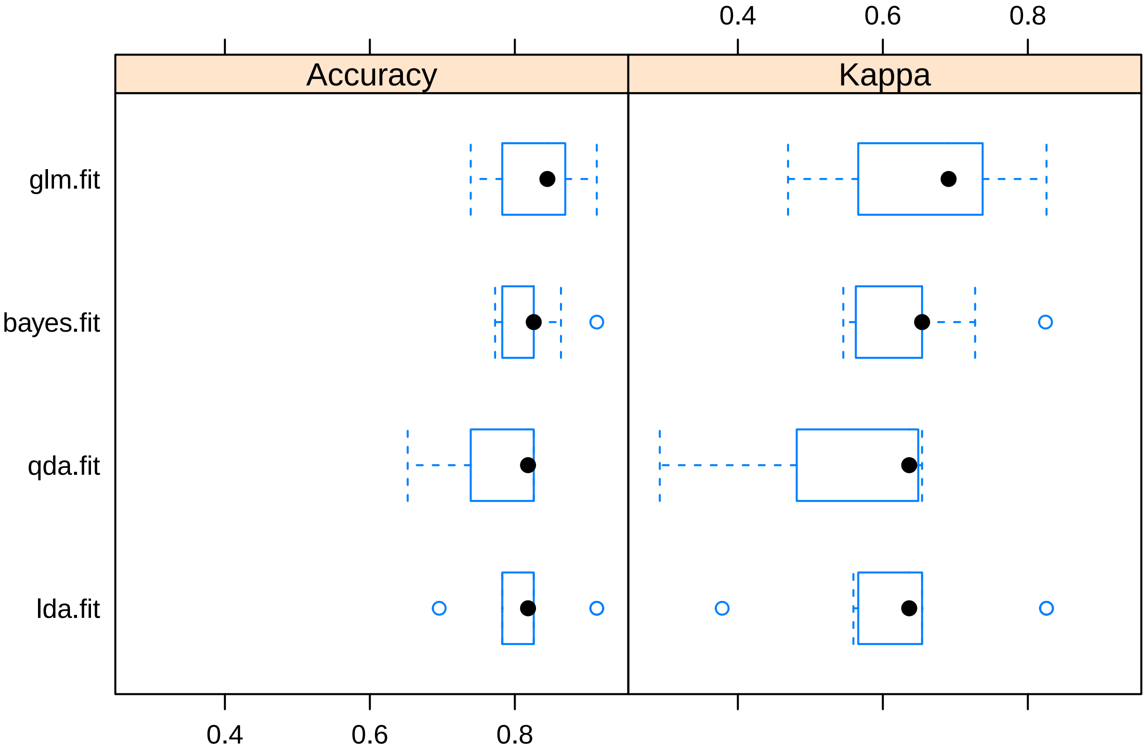
Naive bayes

```
set.seed(1)
nbGrid = expand.grid(usekernel = c(FALSE,TRUE),
                     fL = 1, adjust = seq(0, 4, length = 20))
model.bayes = train(x = train.x,
                    y = train.y,
                    method = "nb",
                    tuneGrid = nbGrid,
                    metric = "Accuracy",
                    trControl = ctrl)

ggplot(model.bayes, highlight = T)
```



resam

```
resamp <- resamples(list(glm.fit = model.glm,
                         lda.fit = model.lda,
                         qda.fit = model.qda,
                         bayes.fit = model.bayes
                         ))

bwplot(resamp)
```



```
summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: glm.fit, lda.fit, qda.fit, bayes.fit
## Number of resamples: 10
##
## Accuracy
##                Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## glm.fit   0.7391304 0.7915020 0.8448617 0.8377470 0.8695652 0.9130435    0
## lda.fit   0.6956522 0.7826087 0.8181818 0.8158103 0.8260870 0.9130435    0
## qda.fit   0.6521739 0.7391304 0.8181818 0.7808520 0.8260870 0.8260870    1
## bayes.fit 0.7727273 0.7826087 0.8260870 0.8243303 0.8260870 0.9130435    1
##
## Kappa
##                Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## glm.fit   0.4692308 0.5836192 0.6907040 0.6737217 0.7371400 0.8257576    0
## lda.fit   0.3783784 0.5660377 0.6363636 0.6297074 0.6528152 0.8257576    0
## qda.fit   0.2923077 0.4812030 0.6363636 0.5603749 0.6488550 0.6541353    1
## bayes.fit 0.5454545 0.5627376 0.6541353 0.6481334 0.6541353 0.8244275    1
```