Clustering

Data Set – Iris

```
library(mlr)
```

```
## Loading required package: ParamHelpers
```

```
task = makeClusterTask(data = iris[,-5])
task
```

```
## Unsupervised task: iris[, -5]
## Type: cluster
## Observations: 150
## Features:
## numerics  factors  ordered
##        4        0        0
## Missings: FALSE
## Has weights: FALSE
## Has blocking: FALSE
```

k Means

```
learner = makeLearner("cluster.kmeans", par.vals = list(centers = 3))
model = train(learner, task)
model
```

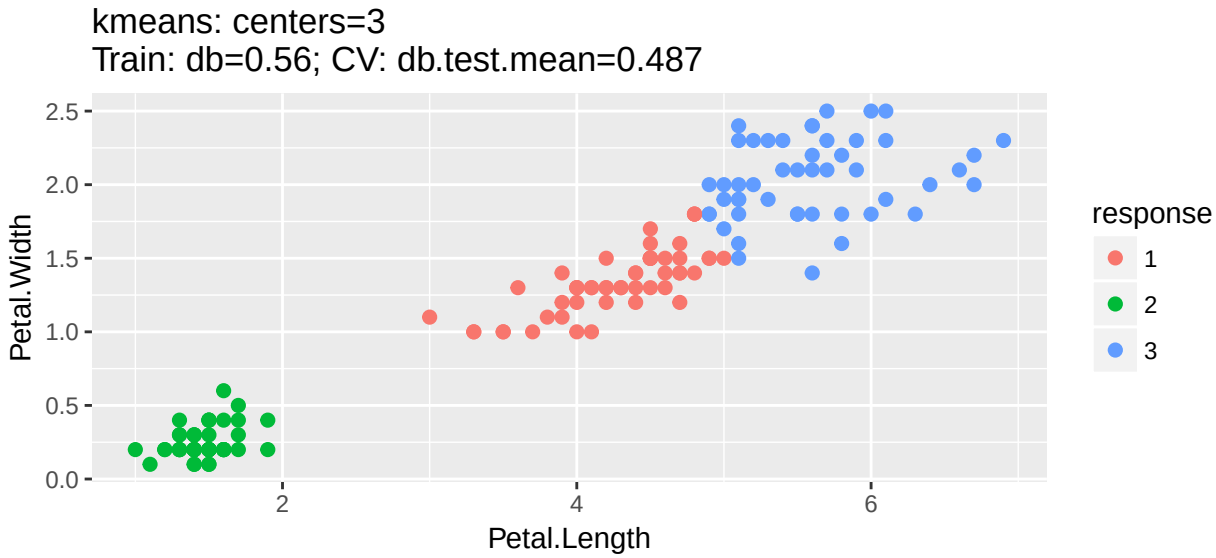
```
## Model for learner.id=cluster.kmeans; learner.class=cluster.kmeans
## Trained on: task.id = iris[, -5]; obs = 150; features = 4
## Hyperparameters: centers=3
```

```
preds = predict(model, task)
preds
```

```
## Prediction: 150 observations
## predict.type: response
## threshold:
## time: 0.00
##   id response
## 1  1        1
## 2  2        1
## 3  3        1
## 4  4        1
## 5  5        1
## 6  6        1
## ... (150 rows, 2 cols)
```

```
plotLearnerPrediction(learner, task, features = c("Petal.Length", "Petal.Width"))
```

```
##
## This is package 'modeest' written by P. PONCET.
## For a complete list of functions, use 'library(help = "modeest")' or 'help.start()'.
```



Predicting Membership Probabilities

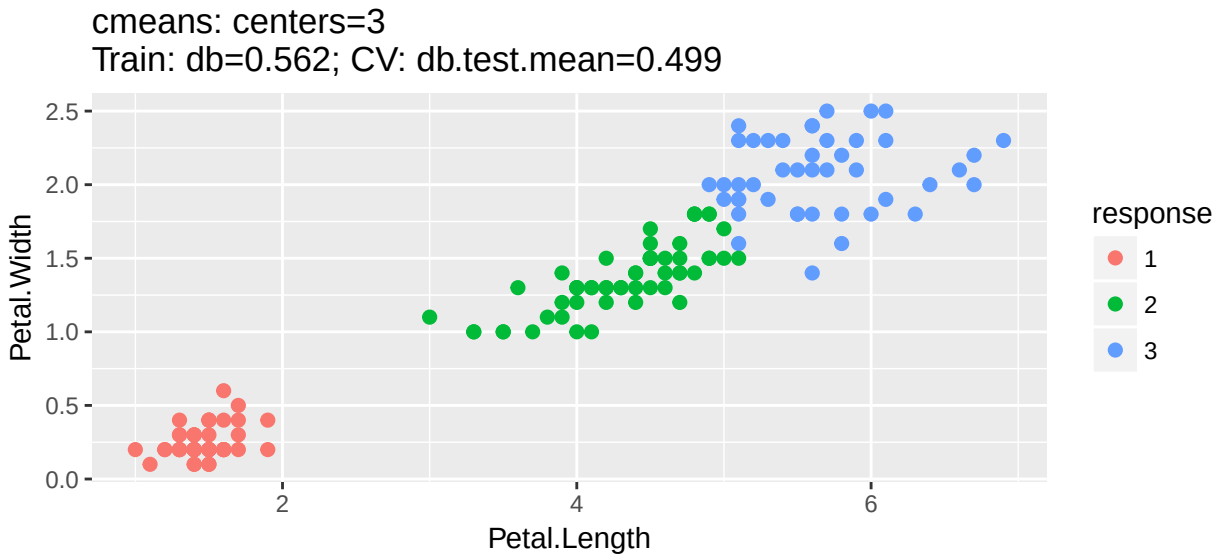
```
learner = makeLearner("cluster.cmeans", predict.type = "prob", par.vals = list(centers = 3))
model = train(learner, task)
model
```

```
## Model for learner.id=cluster.cmeans; learner.class=cluster.cmeans
## Trained on: task.id = iris[, -5]; obs = 150; features = 4
## Hyperparameters: centers=3
```

```
preds = predict(model, task)
preds
```

```
## Prediction: 150 observations
## predict.type: prob
## threshold:
## time: 0.00
##   id response    prob.1      prob.2      prob.3
## 1  1        1 0.9966236 0.002304387 0.001072020
## 2  2        1 0.9758508 0.016650837 0.007498389
## 3  3        1 0.9798248 0.013760367 0.006414864
## 4  4        1 0.9674251 0.022466794 0.010108104
## 5  5        1 0.9944703 0.003761757 0.001767929
## 6  6        1 0.9345703 0.044809007 0.020620667
## ... (150 rows, 5 cols)
```

```
plotLearnerPrediction(learner, task, features = c("Petal.Length", "Petal.Width"))
```

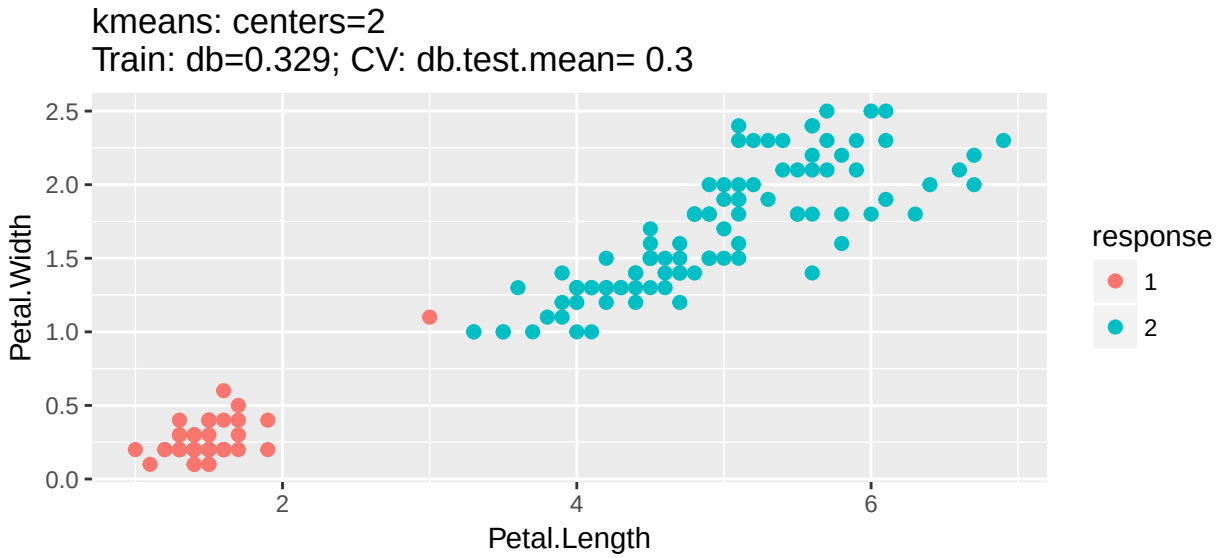


Different Number of Clusters

```
learner = makeLearner("cluster.kmeans", par.vals = list(centers = 2))
model = train(learner, task)
preds = predict(model, task)
preds
```

```
## Prediction: 150 observations
## predict.type: response
## threshold:
## time: 0.00
##   id response
## 1  1        2
## 2  2        2
## 3  3        2
## 4  4        2
## 5  5        2
## 6  6        2
## ... (150 rows, 2 cols)
```

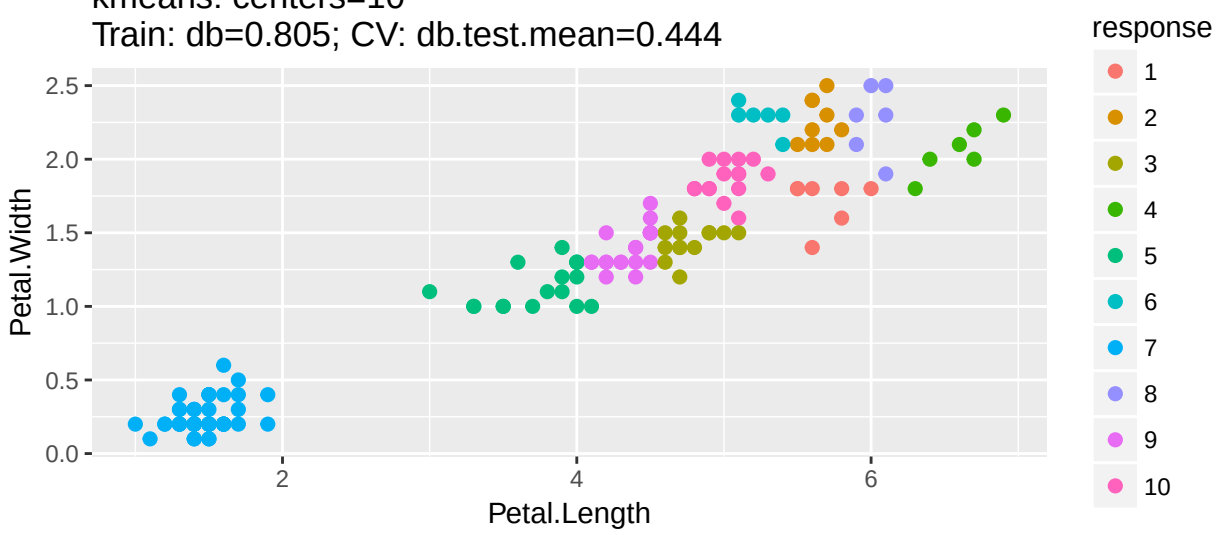
```
plotLearnerPrediction(learner, task, features = c("Petal.Length", "Petal.Width"))
```




```
learner = makeLearner("cluster.kmeans", par.vals = list(centers = 10))
model = train(learner, task)
preds = predict(model, task)
preds
```

```
## Prediction: 150 observations
## predict.type: response
## threshold:
## time: 0.01
##   id response
## 1  1        6
## 2  2       10
## 3  3       10
## 4  4       10
## 5  5        1
## 6  6        7
## ... (150 rows, 2 cols)
```

```
plotLearnerPrediction(learner, task, features = c("Petal.Length", "Petal.Width"))
```



Expectation Maximization

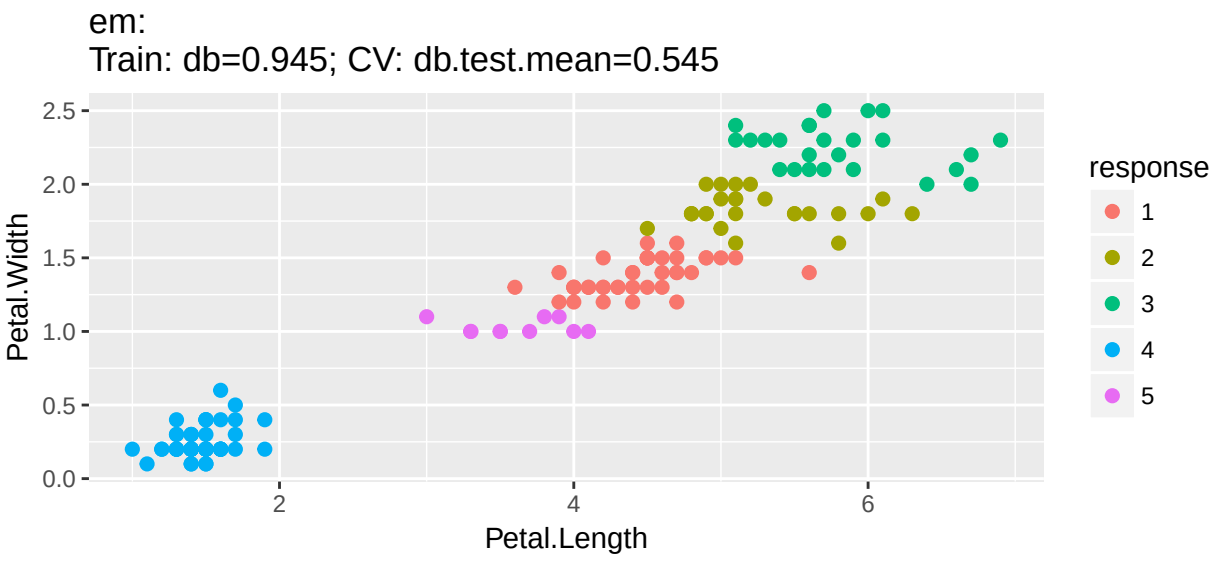
```
learner = makeLearner("cluster.EM")
model = train(learner, task)
model
```

```
## Model for learner.id=cluster.EM; learner.class=cluster.EM
## Trained on: task.id = iris[, -5]; obs = 150; features = 4
## Hyperparameters:
```

```
preds = predict(model, task)
preds
```

```
## Prediction: 150 observations
## predict.type: response
## threshold:
## time: 0.06
##   id response
## 1  1        1
## 2  2        1
## 3  3        1
## 4  4        1
## 5  5        1
## 6  6        4
## ... (150 rows, 2 cols)
```

```
plotLearnerPrediction(learner, task, features = c("Petal.Length", "Petal.Width"))
```



DBScan

```
learner = makeLearner("cluster.dbscan")
model = train(learner, task)
model
```

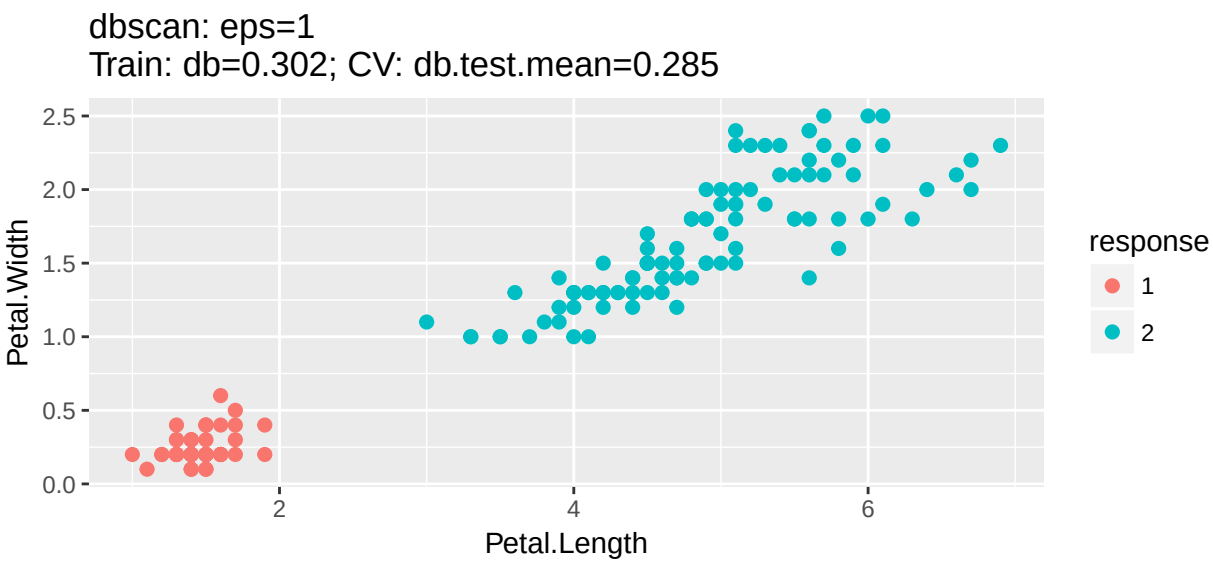
```
## Model for learner.id=cluster.dbscan; learner.class=cluster.dbscan
```

```
## Trained on: task.id = iris[, -5]; obs = 150; features = 4
## Hyperparameters: eps=1
```

```
preds = predict(model, task)
preds
```

```
## Prediction: 150 observations
## predict.type: response
## threshold:
## time: 0.00
##   id response
## 1  1        1
## 2  2        1
## 3  3        1
## 4  4        1
## 5  5        1
## 6  6        1
## ... (150 rows, 2 cols)
```

```
plotLearnerPrediction(learner, task, features = c("Petal.Length", "Petal.Width"))
```



More – survival analysis

```
learner = makeLearner("surv.coxph")
rdesc = makeResampleDesc(method = "Holdout", split = 2/3)
result = resample(learner, lung.task, rdesc, models = TRUE)
```

```
## [Resample] holdout iter 1: cindex.test.mean=0.604
## [Resample] Aggr. Result: cindex.test.mean=0.604
```

```
getRRPredictions(result)
```

```
## Resampled Prediction for:
## Resample description: holdout with 0.67 split rate.
## Predict: test
## Stratification: FALSE
```

```
## predict.type: response
## threshold:
## time (mean): 0.00
##    id truth.time truth.event    response iter  set
## 1  95         60        TRUE  0.19345247    1 test
## 2  64        806       FALSE -0.02323461    1 test
## 3 157        185       FALSE  0.84680915    1 test
## 4  32        583        TRUE  0.24478124    1 test
## 5 123        181        TRUE  0.38275695    1 test
## 6  86        524        TRUE  0.84459946    1 test
## ... (56 rows, 6 cols)
```

```
getLearnerModel(result$models[[1]])
```

```
## Call:
## survival::coxph(formula = f, data = data)
##
##                coef exp(coef)  se(coef)     z     p
## inst      -0.022057  0.978184  0.016110 -1.37 0.171
## age        0.016152  1.016284  0.016096  1.00 0.316
## sex       -0.555683  0.573681  0.251971 -2.21 0.027
## ph.ecog    0.705708  2.025280  0.308029  2.29 0.022
## ph.karno   0.021190  1.021416  0.014078  1.51 0.132
## pat.karno -0.019683  0.980509  0.010365 -1.90 0.058
## meal.cal   0.000405  1.000405  0.000317  1.28 0.202
## wt.loss   -0.015199  0.984916  0.008955 -1.70 0.090
##
## Likelihood ratio test=23.9  on 8 df, p=0.0024
## n= 111, number of events= 81
```