Lab 04C: Week 13 (Solutions)

Contents

The **specific aims** of this lab are:

- estimate logistic regression models
- evaluate out-of-sample performance using cross validation
- perform classification using k-nearest neighbours and evaluate out-of-sample performance using cross validation
- perform dimension reduction using PCA/SVD
- perform cluster analysis using k-means and hierarchical approaches

The unit **learning outcomes** addressed are:

- LO1 Formulate domain/context specific questions and identify appropriate statistical analysis.
- LO3 Construct, interpret and compare numerical and graphical summaries of different data types including large and/or complex data sets.
- LO7 Perform statistical machine learning using a given classifier, and create a cross-validation scheme to calculate the prediction accuracy.

You can use time in this lab to ask your tutor for further guidance on your approach for the group project modelling and report writing. For example, you may want to ask for clarification on any of the feedback that the marker or your peers provided on your presentation.

1 Questions

1.1 Who has diabetes?

This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases

(Johannes 1988). The objective is to predict whether or not a patient has diabetes based on certain clinical measurements. The larger database of patients has been subset such that all observations here are females at least 21 years old of Pima Indian heritage.

The data sets consists of several medical predictor variables and one target variable, `y` which equals 1 if an individual is diabetic and 0 otherwise. Predictor variables includes the number of pregnancies the patient has had (`npreg`), their BMI, insulin level (`serum`), age, triceps skin fold thickness `skin`, diastolic blood pressure (`bp`), plasma glucose concentration (`glu`) and diabetes pedigree function (`ped`).

It is available from various places, including Kaggle and the **reglogit** R package and I've uploaded a copy to the GitHub server.

```
library(tidyverse)
pima = readr::read_csv("https://raw.githubusercontent.com/DATA2002/data/master/pima.csv")
glimpse(pima)
```

```
Rows: 768
Columns: 9
$ npreg <dbl> 6, 1, 8, 1, 0, 5, 3, 10, 2, 8, 4, 10, 10, 1, 5, 7, 0, …
$ glu   <dbl> 148, 85, 183, 89, 137, 116, 78, 115, 197, 125, 110, 16…
$ bp    <dbl> 72, 66, 64, 66, 40, 74, 50, 0, 70, 96, 92, 74, 80, 60,…
$ skin  <dbl> 35, 29, 0, 23, 35, 0, 32, 0, 45, 0, 0, 0, 0, 23, 19, 0…
$ serum <dbl> 0, 0, 0, 94, 168, 0, 88, 0, 543, 0, 0, 0, 0, 846, 175,…
$ bmi   <dbl> 34, 27, 23, 28, 43, 26, 31, 35, 30, 0, 38, 38, 27, 30,…
$ ped   <dbl> 0.63, 0.35, 0.67, 0.17, 2.29, 0.20, 0.25, 0.13, 0.16, …
$ age   <dbl> 50, 31, 32, 21, 33, 30, 26, 29, 53, 54, 30, 34, 57, 59…
$ y     <dbl> 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, …
```

1.1.1 k-nearest neighbours

1. Perform k-nearest neighbours on the data with $k = 5$. How does this perform in-sample? How does this perform out-of-sample? Don't use the **caret** package, write your own CV method.

In sample:

```
library(class)
library(caret)
library(cvTools)
set.seed(1)
norm_fn = function(x) {
    (x - min(x))/(max(x) - min(x))
}
pima_scaled = pima %>%
    dplyr::mutate(dplyr::across(c(bmi, bp, glu, serum, skin), .fns = ~dplyr::na_if(.,
        0))) %>%
    drop_na() %>%
    mutate(y = factor(y)) %>%
```

```
    mutate(across(where(is.numeric), .fns = scale))
X = pima_scaled %>%
    select(-y) %>%
    scale()
y = pima_scaled %>%
    select(y) %>%
    pull()
n = length(y)
knn5 = class::knn(train = X, test = X, cl = y, k = 5)
caret::confusionMatrix(knn5, y)$table
```

```
          Reference
Prediction   0   1
         0 239  38
         1  23  92
```

```
caret::confusionMatrix(knn5, y)$overall[1]
```

```
Accuracy
    0.84
```

Out of sample:

```
K = 5  # number of CV folds
cvSets = cvTools::cvFolds(n, K)  # permute all the data, into 5 folds
glimpse(cvSets)
```

```
List of 5
 $ n      : num 392
 $ K      : num 5
 $ R      : num 1
 $ subsets: int [1:392, 1] 330 263 329 79 213 37 105 217 366 165 ...


 $ which  : int [1:392] 1 2 3 4 5 1 2 3 4 5 ...
 - attr(*, "class")= chr "cvFolds"
```

```
cv_acc = NA  # initialise results vector
for (j in 1:K) {
    test_id = cvSets$subsets[cvSets$which == j]
    X_test = X[test_id, ]
    X_train = X[-test_id, ]
    y_test = y[test_id]
    y_train = y[-test_id]
    fit = class::knn(train = X_train, test = X_test, cl = y_train, k = 5)
    cv_acc[j] = caret::confusionMatrix(fit, y_test)$overall[1]
}
mean(cv_acc)
```

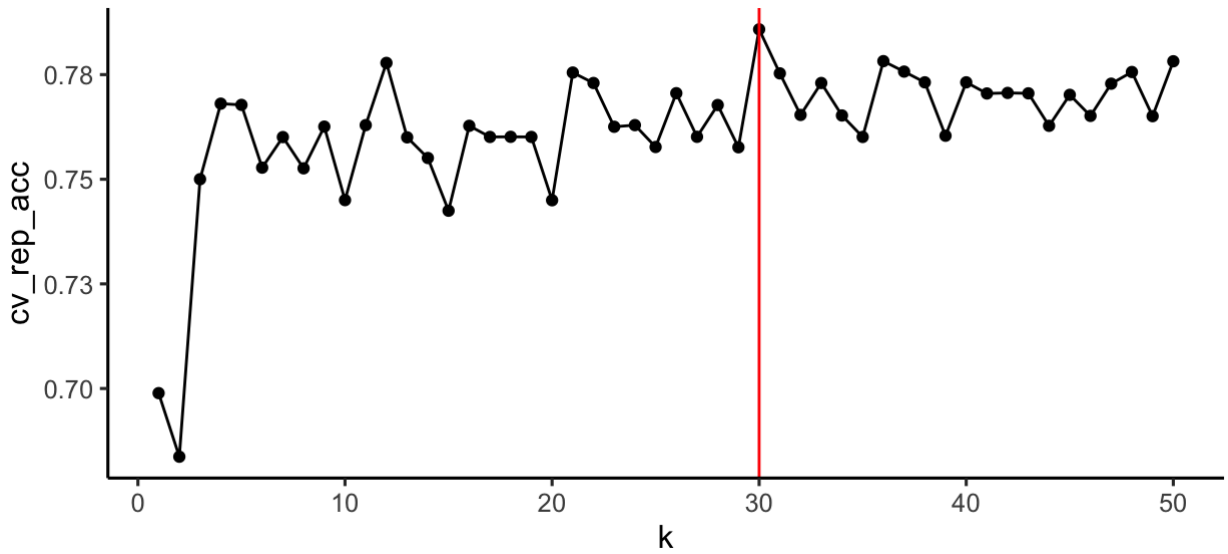
```
[1] 0.77
```

The out of sample accuracy is 0.77, which is quite a bit lower than the in-sample accuracy of 0.84.

2. Let's establish out of sample performance over a range of k values. Put an additional `for` loop around your `CV` loop above to get an estimate of out-of-sample performance for $k = 1, 2, \ldots, 50$. Visualise your results.

```
# range of k in knn to consider
res = data.frame(k = 1:50, cv_rep_acc = NA)

for (i in res$k) {
    cvSets = cvTools::cvFolds(n, K)  # permute all the data, into 5 folds
    cv_acc = NA  # initialise results vector
    for (j in 1:K) {
        test_id = cvSets$subsets[cvSets$which == j]
        X_test = X[test_id, ]
        X_train = X[-test_id, ]
        y_test = y[test_id]
        y_train = y[-test_id]
        fit = class::knn(train = X_train, test = X_test, cl = y_train,
            k = i)
        cv_acc[j] = caret::confusionMatrix(fit, y_test)$overall[1]
    }
    res$cv_rep_acc[i] = mean(cv_acc)
}
res %>%
    ggplot() + aes(x = k, y = cv_rep_acc) + geom_point() + geom_line() +
    geom_vline(xintercept = which.max(res$cv_rep_acc), colour = "red")
```



Looks like there's a maximum around 30.

3. Use the **caret** package to perform *repeated CV* to identify the optimal value for k.

```
## 5-fold repeated CV with 10 repeats
```

```
fitControl = trainControl(method = "repeatedcv",
                          number = 5,
                          repeats = 10)
knnFit1 = train(y ~ .,
                data = pima_scaled,
                method = "knn",
                trControl = fitControl,
                tuneLength = 10)
knnFit1
```

```
k-Nearest Neighbors

392 samples
  8 predictor
  2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 10 times)
Summary of sample sizes: 314, 313, 314, 313, 314, 313, ...
Resampling results across tuning parameters:

  k   Accuracy  Kappa
   5  0.72      0.34
   7  0.73      0.37
   9  0.75      0.41
  11  0.75      0.41
  13  0.75      0.42
  15  0.76      0.42
  17  0.76      0.43
  19  0.76      0.42
  21  0.76      0.41
  23  0.76      0.41

Accuracy was used to select the optimal model using the
 largest value.
The final value used for the model was k = 17.
```

```
knnFit1$results
```

```
    k Accuracy Kappa AccuracySD KappaSD
1   5     0.72  0.34      0.045    0.10
2   7     0.73  0.37      0.055    0.12
3   9     0.75  0.41      0.053    0.12
4  11     0.75  0.41      0.051    0.12
5  13     0.75  0.42      0.051    0.12
6  15     0.76  0.42      0.049    0.12
7  17     0.76  0.43      0.051    0.12
8  19     0.76  0.42      0.054    0.13
9  21     0.76  0.41      0.054    0.13
10 23     0.76  0.41      0.046    0.11
```

```
knn_acc = max(knnFit1$results$Accuracy)
```

1.1.2 Comparison

4. Compare the out of sample accuracies for logistic regression, decision tree, random forest and k-nearest neighbours the different methods using 5 fold CV with 10 repeats.

```
set.seed(2021)
fitControl = trainControl(method = "repeatedcv",
                          number = 5,
                          repeats = 10)
# decision tree
rpartFit1 = train(y ~ ., data = pima_scaled,
                  method = "rpart",
                  trControl = fitControl)
rpart_acc = max(rpartFit1$results$Accuracy)
# random forests
rfFit1 = train(y ~ ., data = pima_scaled,
               method = "rf",
               trControl = fitControl)
rf_acc = max(rfFit1$results$Accuracy)
# logistic regression, full model (no selection has been done)
glmFit1 = train(y ~ ., data = pima_scaled,
                method = "glm", family="binomial",
                trControl = fitControl)
# logistic regression, stepwise model (see Lab 4b)
glm_stepFit1 = train(y ~ npreg + glu + bmi + ped,
                     data = pima_scaled,
                     method = "glm", family="binomial",
                     trControl = fitControl)
glm_acc = glmFit1$results$Accuracy
glm_step_acc = glm_stepFit1$results$Accuracy
```

Comparing the results

```
knn_acc
```

```
[1] 0.76
```

```
rpart_acc
```

```
[1] 0.76
```

```
rf_acc
```

```
[1] 0.78
```

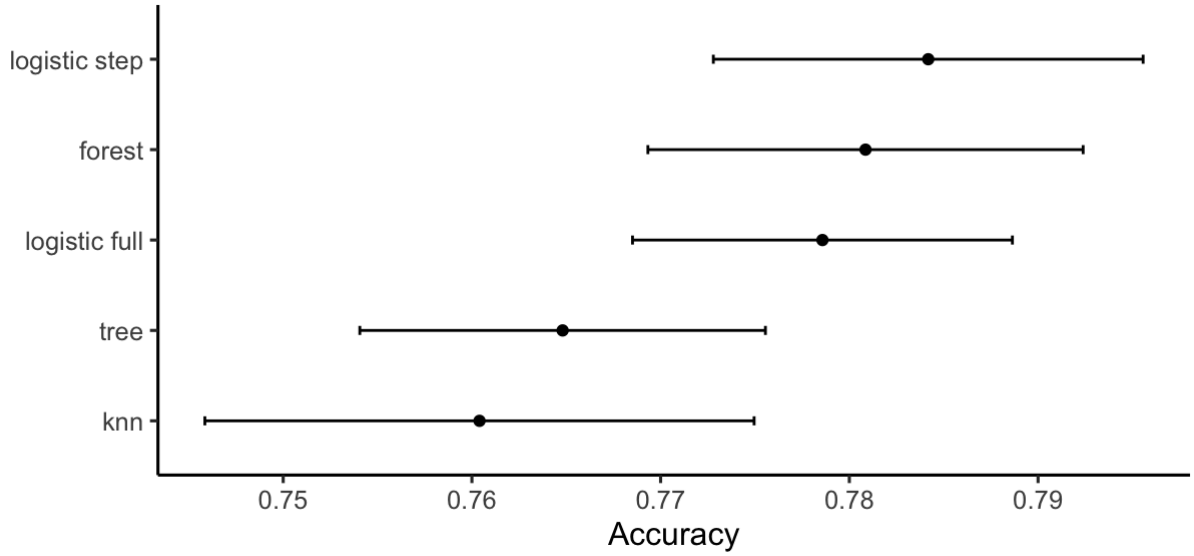
```
glm_acc
```

```
[1] 0.78
```

```
glm_step_acc
```

```
[1] 0.78
```

```
res = resamples(list(knn = knnFit1, tree = rpartFit1, forest = rfFit1,
    `logistic full` = glmFit1, `logistic step` = glm_stepFit1))
ggplot(res) + labs(y = "Accuracy")
```



A single decision tree performs worst, followed by knn. Logistic regression performed best (even better when some initial model selection was performed), closely followed by random forest.

5. Which model do you prefer?

The logistic regression and the random forest performed similarly well. I (Garth) have a preference towards interpretable and transparent models, and so given the similar performance, I would choose the logistic regression.

6. Are our results generalisable to other populations?

Not really. Our data only considered females at least 21 years old of Pima Indian heritage. If we were implementing it as a diagnostic tool, it's only been validated against the similar individuals. In order to extend it more broadly we would need to assess its predictive power on different populations (i.e. people of other heritages).

1.2 Violent crime rates by US states

This is the same data (`USArrests`) we considered for PCA in the lecture. It is built into R, so can be accessed just by typing `USArrests`.

1.2.1 Hierarchical clustering

The `hclust()` function can be used for hierarchical clustering after we have calculated a set of *distances* (a measure of dissimilarity) between each of the observation The distance calculation can be done using the `dist()` function, with the default distance metric being the Euclidean distance.

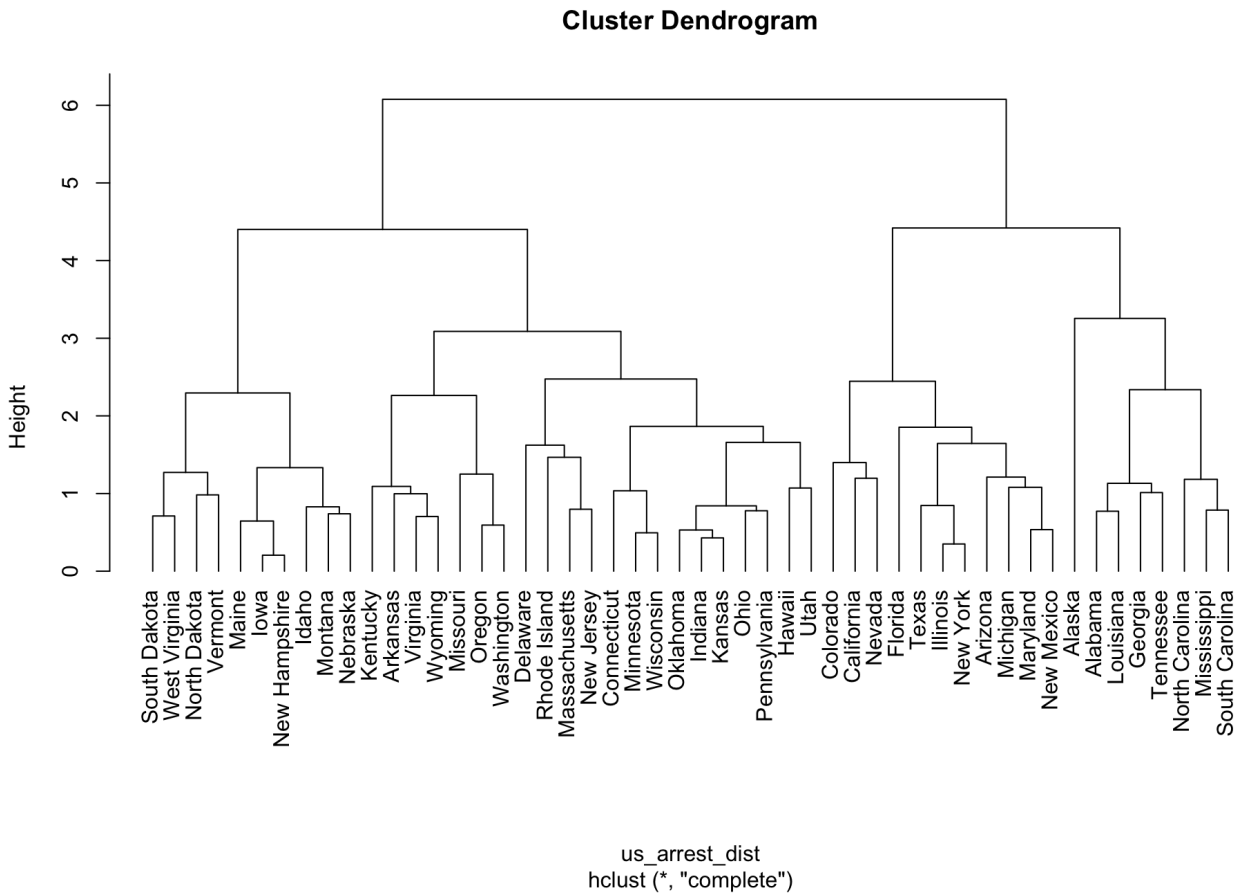
```
dplyr::glimpse(USArrests)
```

1. Calculate the pairwise distances using the `dist()` function applied to the scaled `USArrests` data frame.

```
us_arrest_dist = dist(scale(USArrests))
```

2. Use the `hclust()` function applied to the pairwise distances to perform a hierarchical cluster analysis. Visualise it using the `plot()` function.

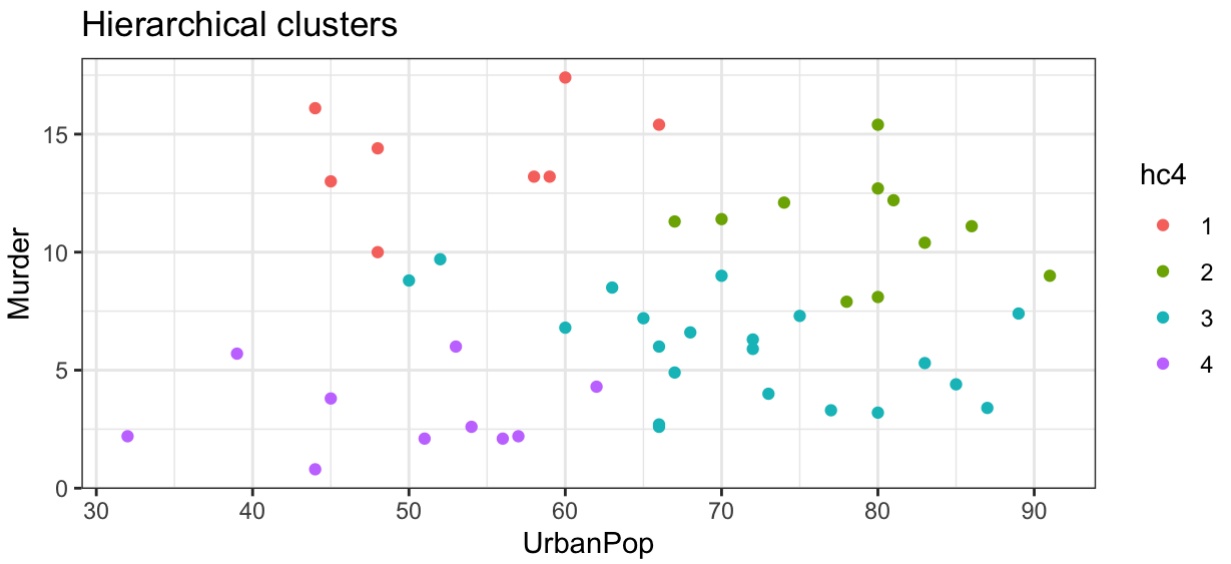
```
us_arrest_hc = hclust(us_arrest_dist)
plot(us_arrest_hc, hang = -1)
```



3. Use the `cutree()` function to cut the tree to make 4 groups. Add the identified clusters as a new

column in the `USArrests` data frame. Plot `Murder` against `UrbanPop` and colour the points by the identified cluster groups.

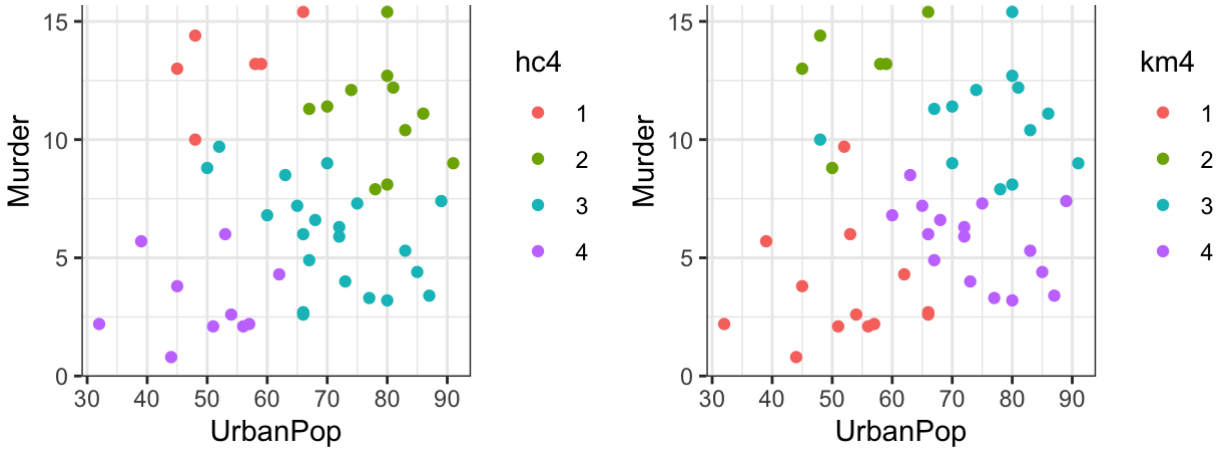
```
# group into k = 4
us_arrest_groups = cutree(us_arrest_hc, k = 4)
us_arrest = USArrests %>%
    tibble::rownames_to_column() %>%
    mutate(hc4 = factor(us_arrest_groups))
p1 = us_arrest %>%
    ggplot() + aes(x = UrbanPop, y = Murder, colour = hc4) + geom_point() +
    theme_bw() + labs(title = "Hierarchical clusters")
p1
```



1.2.2 k-means

Use the `kmeans()` function to performs k-means clustering on the scaled `USArrests` data frame and specify 4 clusters using the `centers` parameter. Plot `Murder` against `UrbanPop` and colour the points by the identified cluster groups.

```
us_arrests_kmeans = kmeans(scale(USArrests), centers = 4)
us_arrest = us_arrest %>%
    mutate(km4 = factor(us_arrests_kmeans$cluster))
p2 = us_arrest %>%
    ggplot() + aes(x = UrbanPop, y = Murder, colour = km4) + geom_point() +
    theme_bw() + labs(title = "k-means clusters")
gridExtra::grid.arrange(p1, p2, ncol = 2)
```



In this case, the k-means and the hierarchical clustering give very similar results when we ask for 4 clusters from both. Note that the group labelling is arbitrary in the colouring (above) or the group labelling below.

```
us_arrest %>%
    janitor::tabyl(hc4, km4)
```

```
hc4  1 2  3  4
  1  0 7  1  0
  2  0 0 11  0
  3  3 1  1 16
  4 10 0  0  0
```

1.3 Who has diabetes? [Revisited]

1.3.1 Clustering and dimension reduction

Let's ignore the labels for now and let the clustering algorithm try to discover structure in the data.

1. Perform k-means clustering on the Pima Indians data (without the `y` variable) for $k = 2$. Create a confusion matrix and compare the clusters discovered by the k-means algorithm with the observed outcomes (`pima$y`).

```
km = kmeans(X, centers = 2, nstart = 10)
tab = table(km$cluster, y)
sum(diag(tab))/sum(tab)
```

```
[1] 0.26
```

Without knowing anything about the labels, the k-means looks like it has achieved an accuracy of 0.26. HOWEVER! we should compare this with the baseline accuracy where we just assign all observations to the largest group:

```
tab = table(y)
tab
```

```
y
  0   1
262 130
```

```
max(tab)/sum(tab)
```

```
[1] 0.67
```

So if we just say all observations don't have diabetes our accuracy is 65%, so k-means hasn't done a particularly good job on differentiating by diabetes status, but that's OK, we weren't asking it to do that, we were just asking it to find two similar groups in the data.

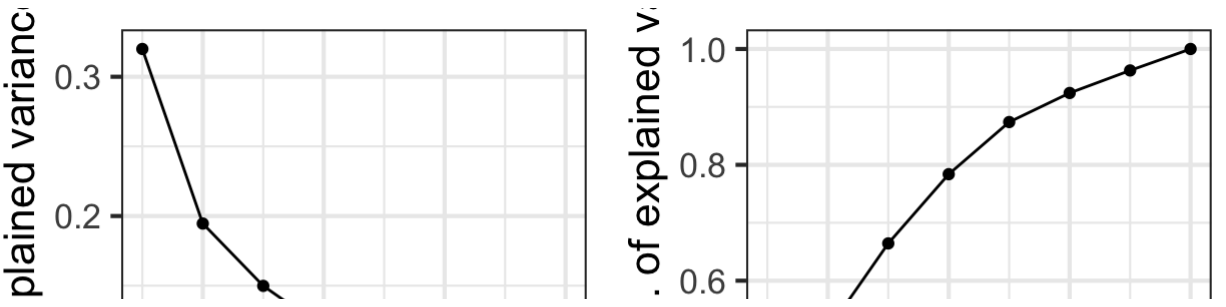
2. Use PCA to perform dimension reduction on the data (again without the `y` variable). How much of the variation in the data can be explained by the first two principal components?

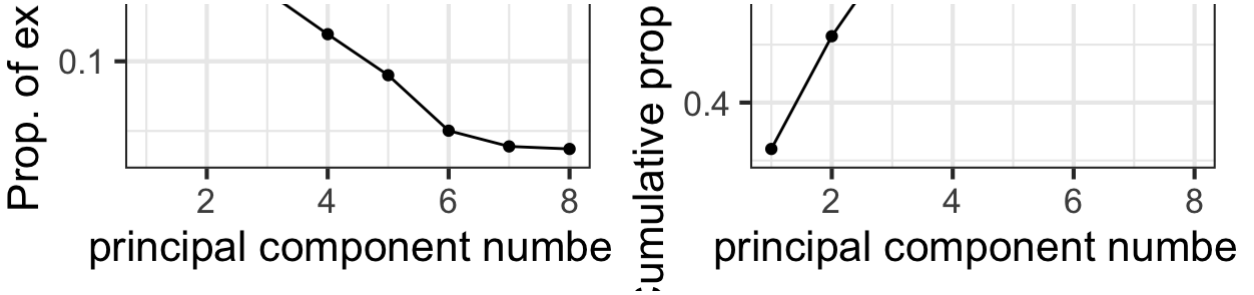
```
pca_pima = princomp(X)
options(digits = 2)
summary(pca_pima)
```

```
Importance of components:
                       Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6
Standard deviation       1.60   1.25   1.09   0.98   0.85   0.63
Proportion of Variance   0.32   0.19   0.15   0.12   0.09   0.05
Cumulative Proportion    0.32   0.51   0.66   0.78   0.87   0.92
                       Comp.7 Comp.8
Standard deviation      0.557  0.544
Proportion of Variance  0.039  0.037
Cumulative Proportion   0.963  1.000
```

The first two principal components account for 48% of the variance in the data.

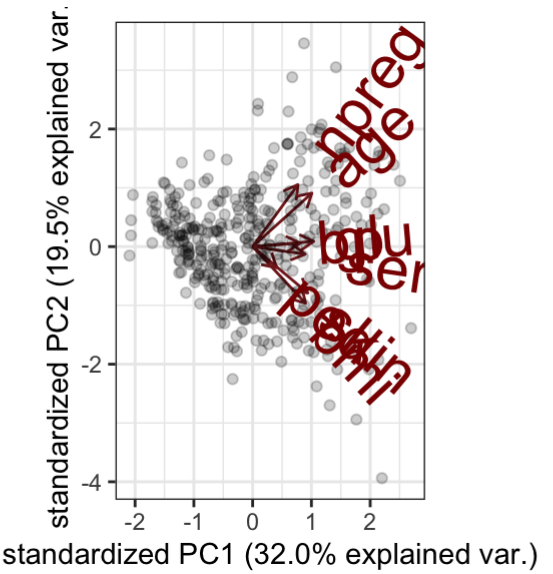
```
# install.packages('remotes') remotes::install_github('vqv/ggbiplot')
# library(ggbiplot)
p1 = ggbiplot::ggscreeplot(pca_pima) + theme_bw(base_size = 16) + labs(y = "Prop. of
        explained variance")
p2 = ggbiplot::ggscreeplot(pca_pima, type = "cev") + theme_bw(base_size = 16) +
    labs(y = " Cumulative prop. of explained variance")
gridExtra::grid.arrange(p1, p2, ncol = 2)
```



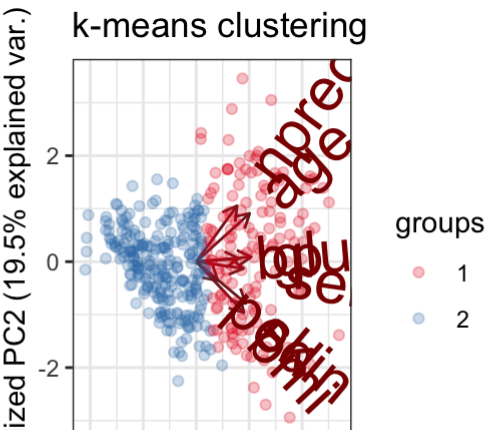


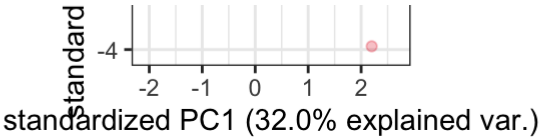
3. Visualise the first two principal components on a biplot. On separate plots, overlay
   a. the clusters identified by the k-means clustering, and
   b. the original data labels.

```
# biplot(pca_pima) # base graphics works just fine
ggbiplot::ggbiplot(pca_pima, labels.size = 5, varname.size = 8, alpha = 0.2) +
    theme_bw()
```

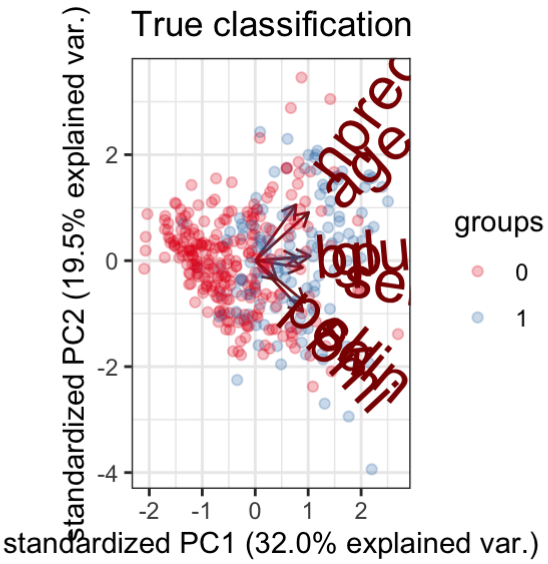



```
ggbiplot::ggbiplot(pca_pima, labels.size = 5, varname.size = 8, alpha = 0.25,
    groups = factor(km$cluster)) + theme_bw() + labs(title = "k-means clustering") +
    scale_color_brewer(palette = "Set1")
```



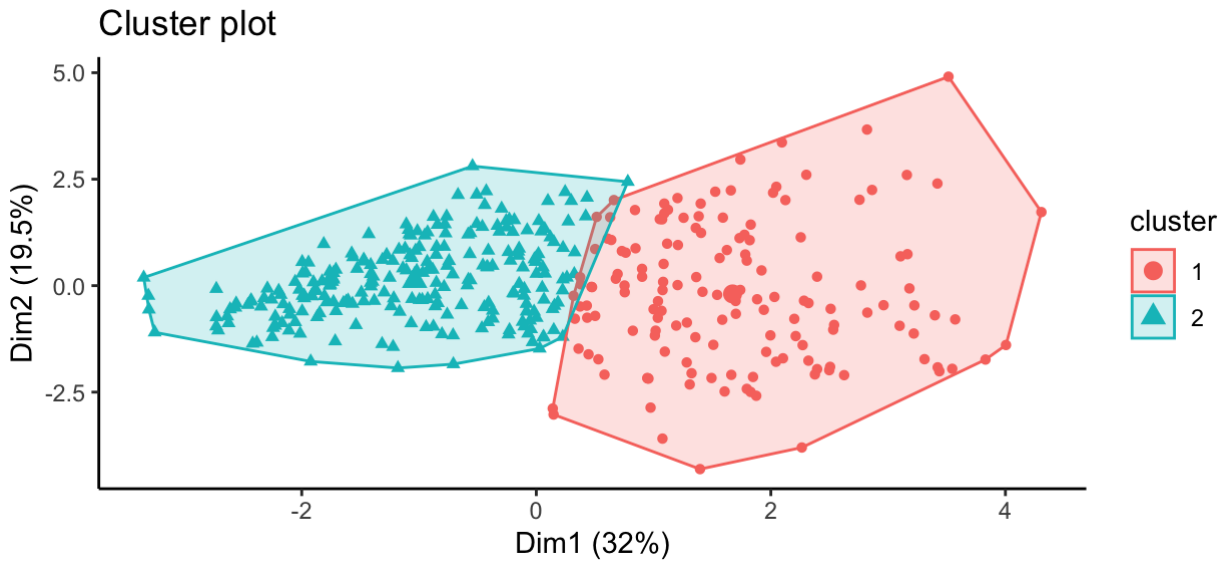



```
ggbiplot::ggbiplot(pca_pima, labels.size = 5, varname.size = 8, alpha = 0.25,
    groups = y) + theme_bw() + labs(title = "True classification") +
        scale_color_brewer(palette = "Set1")
```



Using the **factoextra** package which plots the two clusters in the space spanned by the first two principal components:

```
# install.packages('factoextra')
library(factoextra)
fviz_cluster(km, data = X, geom = "point") + theme_classic()
```



2 For practice after the computer lab

As additional practice questions, I recommend these two DataCamp chapters:

- </> Logistic Regression
- </> Decision trees
- </> k-nearest neighbors
- </> Cluster analysis in R
- </> Dimension reduction with PCA

References

Jed Wing, Max Kuhn. Contributions from, Steve Weston, Andre Williams, Chris Keefer, Allan Engelhardt, Tony Cooper, Zachary Mayer, et al. 2018. *Caret: Classification and Regression Training*. https://CRAN.R-project.org/package=caret.

Johannes, R S. 1988. "Using the ADAP Learning Algorithm to Forecast the Onset of Diabetes Mellitus." *Johns Hopkins APL Technical Digest* 10: 262–66.

Liaw, Andy, and Matthew Wiener. 2002. "Classification and Regression by randomForest." *R News* 2 (3): 18–22. https://CRAN.R-project.org/doc/Rnews/.

Therneau, Terry, and Beth Atkinson. 2018. *Rpart: Recursive Partitioning and Regression Trees*. https://CRAN.R-project.org/package=rpart.