BIOST 2155: Assignment 3

YOUR NAME HERE (YOUR PITT EMAIL HERE)

Released: 2026-10-16. **Due by**: 2026-10-30, 11:59 PM

Instructions

- This file serves as a template for your assignment. **Remember to set eval = TRUE below when compiling the report.**

```
pacman::p_load(formatR)
knitr::opts_chunk$set(echo = TRUE, eval = FALSE,
                      fig.height = 4, fig.width = 6, fig.align = "center",
                      message=FALSE, tidy.opts = list(width.cutoff = 70), tidy = TRUE,
                      class.output  = "shadebox",
                      class.message = "shadebox",
                      class.warning = "shadebox",
                      class.error   = "shadebox")
```

- Use comments to annotate and explain your code whenever possible.
- This assignment covers **Classification II** (support vector machines) and **Clustering I** (k-means, cluster evaluation). It uses two data sets: `Wisconsin Breast Cancer` (`assign3_data.csv`) for Part A, and `Palmer Penguins` (`penguins.csv`) for Part B. Part C is a simulation study: you generate your own data.

Theory (12 points)

T1. The k-means objective and Lloyd's algorithm (7 + 5 = 12 points)

This problem involves the K-means clustering algorithm. Let $C_1, \ldots, C_K$ be a clustering of n data points. The objective function of K-means is given by

$$\min_{C_1,\ldots,C_K} \left\{ \sum_{k=1}^{K} n(C_k)^{-1} \sum_{j,j' \in C_k} \sum_{i=1}^{p} (x_{ji} - x_{j'i})^2 \right\},$$

where $n(C_k)$ is the number of data points in C_k.

(a) (7 points) Given that the center of C_k is $\boldsymbol{\mu}_k = n(C_k)^{-1} \sum_{j \in C_k} \boldsymbol{x}_j$, prove that the objective function above is equivalent to

$$\min_{(C_1,\ldots,C_K)} \sum_{k=1}^{K} \left\{ 2 \cdot \sum_{j \in C_k} \sum_{i=1}^{p} (x_{ji} - \mu_{ki})^2 \right\}.$$

(b) (5 points) On the basis of the identity proved in (a), argue that Lloyd's clustering algorithm decreases the above objective function at each iteration.

Show your working — you may upload handwritten answers for this problem.

Application, Part A: Classification II — Support Vector Machines (35 points)

Dataset: `Wisconsin Breast Cancer`, saved as `assign3_data.csv`.

The dataset contains `n = 569` observations on breast tumors, with 10 features computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. These describe characteristics of the cell nuclei present in the image. The target label `diagnosis` (`M`: malignant / `B`: benign) indicates whether the tumor is malignant or benign. Your goal is to build a support vector classifier to distinguish malignant from benign tumors. **Please set a random seed (`set.seed(42)`) before any step involving data splitting or randomization.**

```
rm(list = ls())
pacman::p_load(tidyverse, caret, e1071, pROC)
set.seed(42)
data <- read_csv("assign3_data.csv") |>
    dplyr::select(-1) |>
    drop_na()
data$diagnosis <- factor(data$diagnosis, levels = c("B", "M"))
```

A1. Data Preparation (3 + 3 = 6 points)

1. Split the data into a training set (75%) and a test set (25%), stratified by `diagnosis`.

```
# your code here
```

2. Support vector classifiers are sensitive to feature scale. Center and scale the 10 predictors in both the training and test sets (fit the scaling on the training set only, then apply it to the test set).

```
# your code here
```

A2. Linear Support Vector Classifier (6 + 4 = 10 points)

1. Using `e1071::tune()` (or `caret::train` with `method = "svmLinear"`), perform 10-fold cross-validation to select the optimal cost parameter `C` over a grid of your choice (e.g. `c(0.01, 0.1, 1, 5, 10, 100)`). Report the chosen `C` and the number of support vectors in the final model.

```
# your code here
```

2. Generate predictions on the test set and report the confusion matrix and test accuracy.

```
# your code here
```

A3. Non-linear (Radial) Kernel SVM (6 + 6 = 12 points)

1. Fit a support vector machine with a radial basis function kernel. Use 10-fold CV to jointly tune `cost` and `gamma` over grids of your choice. Report the chosen parameters.

```
# your code here
```

2. Generate predictions on the test set, report the confusion matrix and test accuracy, and compare to the linear SVC from A2. Did the non-linear kernel improve performance? Given what you know about the geometry of this problem, is that what you'd expect?

```
# your code here
```

A4. Sensitivity and the Cost of Errors (4 + 3 = 7 points)

1. From a clinical perspective, a false negative (a malignant tumor classified as benign) is far more costly than a false positive. Report the sensitivity (recall for the malignant class) of both the linear and radial SVMs from A2/A3.

```
# your code here
```

2. *Interpretation: Which model would you recommend for clinical deployment, and why — accuracy alone, or should sensitivity change your recommendation? Suggest one practical way to make an SVM more sensitive to the malignant class (e.g. adjusting the decision threshold or class weights).*

Application, Part B: Clustering I — k-means and Cluster Evaluation (38 points)

Dataset: `Palmer Penguins`, saved as `penguins.csv`.

This dataset contains morphological measurements for `n = 344` penguins across three species (Adelie, Chinstrap, Gentoo) observed on islands in the Palmer Archipelago, Antarctica. We will ignore the species labels during clustering and use them only afterward, to evaluate how well unsupervised clustering recovers the true species groupings.

- `species`: penguin species (Adelie, Chinstrap, Gentoo) — **label, do not use as a predictor**
- `island`: island of observation
- `bill_length_mm`, `bill_depth_mm`: bill dimensions in mm
- `flipper_length_mm`: flipper length in mm
- `body_mass_g`: body mass in grams
- `sex`: sex of the penguin

```
pacman::p_load(tidyverse, cluster, factoextra)
set.seed(42)
penguins <- read_csv("penguins.csv") |>
    drop_na()
features <- penguins |>
    dplyr::select(bill_length_mm, bill_depth_mm, flipper_length_mm, body_mass_g)
features_scaled <- scale(features)
```

B1. Choosing k (6 + 6 = 12 points)

1. Using the four scaled morphological predictors, produce both a silhouette plot and a gap statistic plot (`factoextra::fviz_nbclust`) to help choose the number of clusters `k`. Report the value of `k` each method recommends.

```
# your code here
```

2. There are three known species in this data. Do the silhouette and gap statistic recommendations agree with this, or with each other? Briefly comment on why an unsupervised method might not recover the "true" number of groups.

```
# your code here
```

B2. K-Means Clustering (5 + 6 = 11 points)

1. Perform k-means clustering with `centers = 3` on the scaled features (use `nstart = 25`). Create a cross-tabulation (`table()`) comparing the true `species` labels to the cluster assignments.

```
# your code here
```

2. Make a scatterplot of `bill_length_mm` vs. `flipper_length_mm`, colored by cluster assignment and faceted (or shaped) by true species. Does k-means recover the species groups well? Which species, if any, gets confused with another?

```
# your code here
```

B3. Hierarchical Clustering & Comparison (7 + 8 = 15 points)

1. Perform hierarchical clustering on the scaled features using complete linkage on a Euclidean distance matrix, and cut the dendrogram into `k = 3` clusters. Create a cross-tabulation comparing true species to these cluster assignments.

```
# your code here
```

2. *Interpretation: Compare the k-means and hierarchical clustering solutions using their cross-tabulations against the true species labels — which method recovered the species structure more accurately? Given that k-means assumes roughly spherical, equal-variance clusters, does that assumption seem reasonable for this data? What does this exercise suggest about the general risk of trusting unsupervised cluster labels as if they were ground truth?*

Part C: Why not cross-validate K? A simulation study (15 points)

In Part A, and in earlier assignments, you chose tuning parameters (the SVM cost, γ) by cross-validation. For k-means, the lecture and lab used the elbow, the silhouette, and the gap statistic instead. In this part you find out why, with a simulation where you know the true answer.

A natural way to cross-validate k-means is this. Split the data into folds. For each fold, fit k-means on the other folds. Then compute the **held-out WSS**: the sum, over the held-out points, of the squared Euclidean

distance from each point to its **nearest training centroid**. Average over the folds, repeat for each K, and choose the K with the smallest average held-out WSS.

This part is **open-ended**. There is no single correct design. You are graded on a clear, reasoned design, correct code, and a careful explanation, not on matching a particular number. Keep the study small: it should run in under a minute.

The helper below computes the held-out WSS for one fold. You may use it or write your own.

```
# Held-out WSS: sum of squared distances from each test point to its
# nearest training centroid. `centers` is the K x p matrix from
# kmeans()$centers.
heldout_wss <- function(test, centers) {
    test <- as.matrix(test)
    d2 <- sapply(seq_len(nrow(centers)), function(k) colSums((t(test) -
        centers[k, ])^2))
    sum(apply(matrix(d2, nrow = nrow(test)), 1, min))
}
```

C1. Run the cross-validation (5 points)

Simulate a data set with a known number of clusters K_{true}. You choose n, the number of variables, K_{true}, and how well separated the clusters are. State your choices and say why you made them. Then run 5-fold cross-validation of the held-out WSS for $K = 1, \ldots, 10$ (use `nstart = 10` or more). Plot the average held-out WSS against K. Which K does the "smallest held-out WSS" rule choose?

```
# your code here
```

C2. Explain the result (4 points)

Why does held-out WSS behave this way as K grows? Compare with cross-validating the SVM cost in Part A: what makes the cross-validated error there go back up for very flexible models, and what is missing in the clustering problem? Does your plot still contain useful information about K_{true}?

C3. Fix it (4 points)

Propose and implement a procedure that uses held-out data or resampling and **does** recover K_{true} in your simulation. Some ideas, which you may use or ignore:

- **Stability:** cluster two overlapping subsamples, and measure how well the two clusterings agree on the shared points (e.g. with the adjusted Rand index, `mclust::adjustedRandIndex()`). Real clusters come back each time; splits made up by the algorithm do not.
- **Prediction strength** (Tibshirani and Walther, 2005): cluster a training half and a test half separately. Check how often pairs of test points that are in the same test cluster are also assigned to the same training centroid.
- **A reference comparison:** compare held-out WSS with its value on data that have no clusters, as the gap statistic does.

Show that your procedure finds K_{true}. Then change one thing in your simulation (for example, make the clusters overlap more, or make them unequal in size) and report what happens to your procedure.

```
# your code here
```

C4. Recommendation (2 points)

In 3–4 sentences, written for a collaborator with no statistics training: why can't we choose the number of clusters the same way we choose a tuning parameter for a classifier, and what would you do instead?