IDA: Missing Values :-

→ library(mice). # multivariate imputation by chained equation.

→ . md.pairs()

→ md.pattern().

---

* . library(Vim) # package for visualization and imputation of missing values.

* . aggr()

* marginplot().

* scattmatrixMiss().

HW-3 :- * (or)

* prcomp()

corrplot(package) :- package is good.

PCA

- library(scatterplot3d)
- library(ggl).
- library(ggbiplot).

- biplot()
- ggbiplot().
- Scatterplot3d()
- plot3d()

---

lda and tsne.

LDA in MASS package.

- library(MASS)
- lda().
- lda.model %>% predict(transformed).
- lda needs data scaled and centered.

- ldahist().

- library("Rtsne"). (TSNE)
- Rtsne( ).

- Grubbs's test for outliers.
  - library(outliers)
  - grubbs.test( ). - univariate test.
  - identify( ).
  - outlier( ).

Assignment 4:-

Adaptive predictor modelling Textbook:-

~~* skewness () in e1071 package.~~

Chapter 3:-

Transformations

*skewness() in e1071 package.

* Calculates skewness of a predictor.

* Another function, preProcess applies this transformation to a set of predictors.

- spatialSign().
- impute.knn() in impute package
- Preprocess() in Caret. package.

(840x_transformation-1.R :-)

* Symbox function in "car" package.
  - produces ~~out~~ parallel box plots for different values of lambda.
  - helpful in Transformation.

* boxcox in "EnvStats"

In HW4, My solution

* I used,
  * BoxCoxTrans() from Caret package to find the required transformation.

HW5: (Imputations)

Mice :-

Multivariate Imputation by chained equations is used widely for handling missing data.

* How mice works :-

(1) Initialization :-

Missing values are initially replaced with simple methods. (eg. mean imputation) to create a starting point.

2. Chained Equations :-

For each variable with missing data, the missing values are imputed using a regression model that is a fitted based other variables in the data.

3. Iterative process :-

This process is repeated in an iterative manner, cycling through each variable until convergence is achieved.

Convergence occurs when distribution of imputed values stabilizes meaning that further iterations do not significantly change the ~~data~~ results.

4. Multiple iterations:

This iterative process is repeated m times to create multiple different imputed datasets. Each dataset represent a plausible set of values that could have been observed.

Pooling Analysis:

- After creating multiple datasets, analysis are performed on each dataset seperately. (like fitting a regression model on each imputed dataset).
- The results from each dataset are then pooled using Robin's rules to produce a single set of results.
- Pooling involves combining estimates (eg. means, regression coefficients) and calculating std errors. that account for both within-imputation variability.

(variability within each imputed dataset) and between-imputation variability. (variability between different imputed datasets).

- This provides final parameter estimates and std. errors that incorporate the uncertainity due to missing data.

How to choose m?

Common Recommendations:-

Missing data proportion.	m.
<10%.	5
10-30%.	10-20.
>30%.	30-50.
>50%.	>50
>70-80%.	drop.
>90	obviously drop.

- Rule of thumb from Robin (1987) :-

$$m \approx \frac{100}{F}$$

F is fraction of missing data.

$$F = \frac{\text{Total Number of missing values}}{\text{Total Number of observations.}}$$

Some Important functions:-

- mice()
- complete().
- with()
- pool()
- md.pattern()
- stripplot().
- densityplot().
- xyplot().
- init().
- complete.cases().
- set.seed().
- md.pairs().

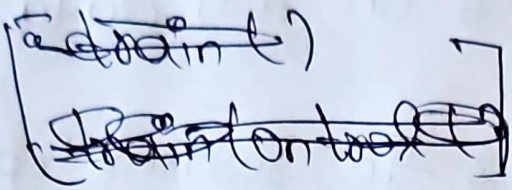
VIM (Visualization and Imputation methods)

- aggr() - Aggregation plot.
- matrixplot()
- marginplot()
- scattMiss()
- histMiss()

Imputation methods:-

- kNN().
- hotdeck().
- irmi() - Iterated Robust model-based Imputation.
- regressionImp() - Regression Imputation.
- robustRegressionImputation().

Unit 6:- Caret package.



(1) train():-

parameters:- method, trControl, tuneGrid, tuneLength.

(2) trainControl():-

parameters:- method, number, repeats.

(3) expand.grid():- parameter changes according to algorithms.

(4) preProcess():- Apply transformations like. scaling, centering, missing values, skewness, pca etc.

(5) Predict():- Uses trained model to generate predictions on new data.

(6) fit$resample:- Acess the resampling results.

(7) fit$finalModel:- Acess the underlying model object trained by caret.

Unit 7: Multiple linear Regression

(1) car library:

- vif() - Compute variance inflation factor for a linear model.

---

(2) MASS library:-

- lm.ridge - perform ridge.
- svd() - perform single value decomposition.
- qplot() - quick plot function for

creating simple plots.

(3) lars: library :-

- lars() - perform least angle regression or lasso.
- predict.lars()
- cv.lars().

---

(4) glmnet library :-

- cv.glmnet().

---

(5) pls library() :-

plsr() :- perform partial least squares regression.

RMSEP() :- compute rmse of prediction for a model.

⑥ Stepwise Regression

- step()

⑦ earth package:

earth(), plotmo().

⑧ rgl package:-

- plot3d().
- surface3d().
- planes3d()

⑨ Robust regression:- (MASS library).

- rlm() - fits a robust regression using M estimation.
- psi.bisquare():- function specifying the bisquare loss function for robust regression.

Unit 8: logistic regression :-

- glm() function used with family specified "binomial".
- glm() is in base R package.

Unit 9:- decision trees :-

rpart library :

- rpart() - Builds decision trees using CART algorithm.
- plotcp() - plots the cost complexity pruning path.
- prune() - prunes a tree based on complexity parameter (cp).

partykit library:-

- as.party() - converts a rpart object to "party" object for visualization. - plot() - plots a party tree.

---

caret library:-

- confusionMatrix() :- computes confusion matrix & performance metrics
- varImp() - computes variable importance of models.

---

rattle library:-

- fancyRpartPlot() - plots decision trees with enhanced visual representation.

---

adabag: library:

- boosting() - implements boosting algorithms for classification.

ipred library :-

- bagging() - Implements bagging for classification.
- errorest() - computes error estimates using cross validation.

---

randomForest library :-

- randomForest().
- varImpPlot().
- importance().

Unit 10 :- Support Vectors

- e1071 library :-
  - svm() - fits svm model.
  - sigest() - Estimates sigma parameter for radial basis kernels.

kernlab library :-

- By default caret uses kernlab for SVM.

Unit II :- Neural networks :-

Unit 12 :- Clustering :-

① stats (Base R stats package) :-

- kmeans() - performs k-means clustering
- princomp() - An alternative function for pca.

② Cluster library :-

- pam() - partitioning around medoids clustering
- silhouette() - compute silhouette width for clustering evaluation.

- daisy() - computes dissimilarity matrix for mixed data. (Numerical and Factor data).

③ useful library

- fitKMeans() - Fit k-means clustering with evaluations (eg:- Hartigan's rule).

- plotHartigan() - Visualize Hartigan's rule for cluster evaluation.

④ NbClust library

NbClust() - identifies the optimal number of clusters using multiple indices.

⑤ rgl library

plot3d() - creates interactive 3d plots.

⑥ Hmisc library:-

- varclus() - performs hierarchial clustering of variables.

⑦ dbscan library:-

- dbscan() - perform density-based spatial clustering.
- kNNdist() - Computes the k nearest neighbour distance.
- kNNdistplot() - Visualizes k-nearest neighbour distance to find the "knee" (optimal eps).
- optics() - performs OPTICS clustering.
- extractDBSCAN() - Extracts DBSCAN clustering from OPTICS results.

hullplot() - Visualizes clustering results with convex hulls.

Base R:

- dist() - computes distance matrix.
- hclust() - performs hierarchial clustering
- rect.hclust() - draws rectangles around hierarchial clustering.
- cutree() - cuts a dendrogram tree into specified clusters.

Unit 12:- Clustering

1. K-Means Clustering :-

- kMeans()
- FitkMeans() : Helps identify optimal clusters using Hartigans rule,
- wsplot(data, nc) - custom function to plot elbow method,
- plotHartigan()

2. Partitioning Around Mediods :-

- pam()
- silhoutte()

3. Hierarchial Clustering :-

- hclust()
- daisy()
- rect.hclust().