Kalinga University

Faculty of Computer Science & Information Technology

Course- BCAAIML
Subject- R Programming
Subject Code – BCAAIML401 **Sem- IV**

Unit 5

Interfacing R to other languages – Parallel R – Basic Statistics – Linear Model – Generalized Linear models – Non-linear Models – Time Series and Auto-Correlation – Clustering.

Interfacing R to Other Languages

R can interact with other languages like **C/C++**, **Python**, **Java**, **SQL**, and even **Fortran**. This is helpful to speed up computation, access specialized libraries, or leverage the strengths of other languages. Here are some common methods:

- **Rcpp**: This package is commonly used to integrate R with C++ for performance-critical applications.
- **reticulate**: Allows seamless integration between R and Python.
- **rJava**: Enables R to communicate with Java.
- **RMySQL**, **RODBC**: Used to interface R with SQL databases.
- **Rserve**: A middleware that allows R to interact with client applications in other languages.

Parallel R

Parallel computing in R is important when working with large datasets or computationally expensive algorithms. There are multiple ways to parallelize computations:

- **Parallel package**: Provides a high-level interface to parallel computing (e.g., mclapply, parLapply).
- **foreach package**: Commonly used for parallel loops, and it can be used with multiple backends such as **doParallel** or **doSNOW**.
- **future package**: A high-level framework for asynchronous parallel computing in R.
- **sparklyr**: Connects R with Apache Spark, enabling distributed computing on large datasets.

Basic Statistics

R is well-equipped for fundamental statistical analysis, such as:

- **Descriptive statistics**: mean(), sd(), summary(), hist(), etc.
- **Inferential statistics**: Hypothesis testing (t.test(), chisq.test(), aov()), confidence intervals, p-values.
- **Probability distributions**: dnorm(), pnorm(), qnorm(), rnorm() for normal distributions and similar functions for others (e.g., Poisson, Binomial).

Linear Models

Linear regression models are foundational in statistics. In R, you can fit linear models using the lm() function:

```
model <- lm(y ~ x1 + x2 + ..., data = dataset)
summary(model)
```

This will give you estimates for the coefficients, p-values, R-squared, etc.

Generalized Linear Models (GLMs)

GLMs extend linear models to handle non-normal error distributions. Common GLMs include logistic regression (for binary outcomes), Poisson regression, etc. You can fit a GLM using the glm() function:

```
glm_model <- glm(y ~ x1 + x2, family = binomial(link = "logit"), data = dataset)
summary(glm_model)
```

- **family**: Specifies the distribution (e.g., binomial for logistic regression, poisson for count data).
- **link**: Specifies the link function (e.g., logit for logistic regression).

Non-linear Models

In cases where the relationship between variables is not linear, you can fit non-linear models using the nls() (non-linear least squares) function:

```
nls_model <- nls(y ~ a * exp(b * x), data = dataset, start = list(a = 1, b = 0.1))
summary(nls_model)
```

This allows you to fit more complex, non-linear relationships, such as exponential or logarithmic models.

Time Series and Auto-Correlation

Time series analysis deals with sequential data. R has several tools for time series analysis:

- **ts()**: Create a time series object.

- **acf()**: Autocorrelation function to detect correlation between lagged observations.
- **arima()**: Fit ARIMA models for forecasting time series data.

Example:

```
ts_data <- ts(data, frequency = 12, start = c(2000, 1))
acf(ts_data)
model <- arima(ts_data, order = c(1, 0, 0))  # AR(1) model
```

Clustering

Clustering is a technique to group similar data points together. Common clustering methods include **K-means**, **Hierarchical clustering**, and **DBSCAN**.

- **K-means**: Partitions data into k clusters.

  ```
  kmeans_model <- kmeans(data, centers = 3)
  ```

- **Hierarchical Clustering**: Uses a dendrogram to visualize hierarchical relationships.

  ```
  dist_matrix <- dist(data)
  hc_model <- hclust(dist_matrix)
  plot(hc_model)
  ```

- **DBSCAN** (Density-Based Spatial Clustering of Applications with Noise): Useful for datasets with noise and outliers.

  ```
  library(dbscan)
  dbscan_model <- dbscan(data, eps = 0.5, minPts
  ```