Data Imputation, Visualization, and Descriptive Analysis in R

Dani

1 Introduction

This guide covers techniques for handling missing data, performing imputation, and visualizing various types of data. Understanding how to manage and analyze incomplete datasets is crucial in applied statistics and data science, especially when working with real-world data that often has missing or messy entries.

2 Handling Missing Data

2.1 Inspecting Missing Data

- Use the 'datasets' package to access built-in datasets like `airquality`.
- Calculate the proportion of missing data in each variable:

```r
library(datasets)
data("airquality")

# Proportion of missing observations
for (i in 1:6){
  print(sum(is.na(airquality[,i]))/length(airquality[,i])*100)
}
```

- **Interpretation**: Identifying variables with high proportions of missing data is the first step in deciding whether to impute, remove, or analyze them separately.

2.2 Omitting NA Values

- Omit missing values using `na.omit()`:

```r
air_omit <- na.omit(airquality)
summary(air_omit)

# Visualize pairwise relationships
plot(air_omit[,1:4])
```

- **Key Insight**: Use omission only when the proportion of missing data is minimal, as dropping rows may result in information loss and biased analyses.

2.3 Visualizing Missing Data Patterns

- Use the `mice` and `VIM` packages to visualize missing data patterns:

```r
install.packages("mice")
library(mice)

# Visualize missing data patterns
md.pattern(airquality)

install.packages("VIM")
library(VIM)
aggr(airquality)  # Aggregated plot of missing values
```

- **Best Practice**: Visualizations such as aggregated bar plots or missing data patterns help determine which variables have systematic missingness.

3 Data Imputation

3.1 Mean Imputation

- Replace missing values with the variable mean:

```
install.packages("Hmisc")
library(Hmisc)

airquality_imp <- airquality
airquality_imp$Ozone <- with(airquality_imp, impute(Ozone, mean))

# Verify imputation
summary(airquality_imp$Ozone)
```

- **Limitations**: Mean imputation reduces variance and may lead to biased estimates, especially for non-normally distributed variables.

3.2 Multivariate Imputation with `mice`

- Impute using multiple iterations to capture uncertainty:

```
library(mice)

# Mean imputation with one iteration
imp_mice <- mice(airquality, method = "mean", m = 1, maxit = 1)
complete_data <- complete(imp_mice)

# Compare distributions
plot(density(airquality$Ozone, na.rm = TRUE), ylim = c(0, 0.025))
lines(density(complete_data$Ozone), col = 2)
```

- **Practical Tip**: Use `m = 5` or more for robust imputations. Visualize imputed vs. original distributions to ensure consistency.

3.3 Regression-Based Imputation

- Impute using linear regression:

```
reg_imp <- mice(airquality, method = "norm.predict", m = 1)
air_reg_imp <- complete(reg_imp)

# Visualize imputed values
hist(airquality$Ozone)
hist(reg_imp$imp$Ozone[,1], add = TRUE, col = 2)
```

- **Note**: Regression imputation preserves relationships but may artificially inflate R-squared values.

4 Poverty Study Data: Missing Data Codifications

- Handle non-standard missing value codifications (e.g., -99):

```
library(readr)
pov_data <- read_table("PovertyStudy.dat")
pov_data$GNP[pov_data$GNP == -99] <- NA

# Replace with median
pov_data$GNP[is.na(pov_data$GNP)] <- median(pov_data$GNP, na.rm = TRUE)
```

- **Best Practice**: Convert placeholders to NA and then impute to avoid misleading results.

5 Data Transformation: Box-Cox Transformation

- Use Box-Cox transformation to stabilize variance:

```
library(MASS)

# Box-Cox transformation on GNP
boxcox(pov_data$GNP ~ 1, lambda = seq(-1, 1, by = 0.1))

# Log transformation
lGNP <- log(pov_data$GNP)
```

- **Tip**: Choose the transformation that minimizes skewness and maximizes normality.

6 Data Visualization Techniques

6.1 Visualization of Numerical Variables

- Use **Chernoff Faces** to visualize multidimensional data:

```
library(aplpack)
data(longley)
faces(longley[1:9,], face.type = 0)
```

- **Application**: Useful for detecting patterns in small datasets with multiple variables.

6.2 Correlation Plots

- Visualize correlation matrices with `corrplot`:

```
install.packages("corrplot")
library(corrplot)

cr <- cor(pov_data[, 1:6])
corrplot(cr)
```

- **Tip**: Use the `method = "number"` argument to show correlation values directly.

7 Visualizing Categorical Data

7.1 Bar Plots and Mosaic Plots

- Create bar plots for categorical variables:

```
barplot(prop.table(tab), col = c("red", "blue"),
        legend.text = c("vshaped", "straight"),
        main = "Distribution of Satisfaction Level",
        ylim = c(0, 0.7))
```

- Use mosaic plots for visualizing interactions:

```
library(RColorBrewer)

# Define pastel colors
pastel_colors <- brewer.pal(3, "Pastel1")
mosaicplot(tab, main = "Transmission Type vs. Engine Shape", color = pastel_
    colors)
```

8 Summary Tables and Factor Data

- Cross-tabulations and probability tables:

```
# Conditional probabilities by row
prop.table(tab, 1)

# Joint and marginal probabilities
prop.table(tab)
```