Rotman

INTRO TO R PROGRAMMING

R Tutorial (RSM358) – Session 3

September 24, 2024 Prepared by Jay Cao / TDMDAL

Website: https://tdmdal.github.io/r-intro-2024-fall/

Rotman School of Management
UNIVERSITY OF TORONTO

Any Questions about Lab 3.6 & A2 Coding?

- Loading the data (`Auto, Carseats`)?

- Carseats.csv (raw data) is in the zip file on the resource page
    - Alternatively, load `Carseats` data via the ISLR2 package: `library(ISLR2)`

- Good practise: Inspect the raw data file before calling `read.csv()`
    - This helps to determine the potential arguments of `read.csv()`
        - `E.g., na.strings = "?" or stringsAsFactors = T`

- `factor` column/variable (categorical variable)
    - `as.factor()`

Any Questions about Lab 3.6 & A2 Coding?

- Linear regression?

- Comment on the summary table result (section 3.4)
  - Is there a relationship, how strong, positive or negative
  - Confidence and prediction interval

- Interpretation of coefficients
  - Section 3.1 and 3.2 for quantitative/continuous predictors
  - Section 3.3.1 for qualitative/categorical predictors

- Outlier and high leverage observations
  - Use post-regression diagnostic plot; section 3.3.3

Linear Regression

- `my_lm <- lm(formula = …, data = …)`
- `summary(my_lm)`

- `plot()`
  - Two variable scatter plot: `plot(x, y)`
  - Regression line: `abline(my_lm)`
  - Post-regression diagnostic plot: `plot(my_lm)`

- `predict(object, new_data, interval, level=0.95)`
  - Confidence interval
    - `E.g., predict(my_lm, data.frame(x1 = (c(5, 10))), interval = "confidence")`
  - Prediction interval
    - `E.g., predict(my_lm, data.frame(x1 = (c(5, 10))), interval = "prediction")`

Create Data Frame using `data.frame()`

```r
# create a data frame
df1 <- data.frame(
  x = 1:3,
  y = letters[1:3],
  z = c(1.1, 2.2, 3.3)
)
```

x	**y**	**z**
1	"a"	1.1
2	"b"	2.2
3	"c"	3.3

lm() R Regression Formula - 1

my_df

y	x1	x2	x3
18	8	307	130
16	8	304	150
…	…	…	…

lm() Regression Formula	Regression Formula

`lm(formula = y ~ x1 + x2 + x3, data = my_df)`
`lm(formula = y ~ ., data = my_df)`
	$Y_i = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \epsilon_i$

`lm(formula = y ~ . - x3, data = my_df)`
`lm(formula = y ~ x1 + x2, data = my_df)`
	$Y_i = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon_i$

`lm(formula = y ~ 0 + x1 + x2 + x3, data = my_df)`	$Y_i = \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \epsilon_i$
…	…

Source: https://stat.ethz.ch/R-manual/R-devel/library/stats/html/formula.html my_df

y	x1	x2	x3
18	8	307	130
16	8	304	150
…	…	…	…

lm() R Regression Formula - 2

lm() Regression Formula	Regression Formula

`lm(formula = y ~ x1 * x2, data = my_df)`
`lm(formula = y ~ x1 + x2 + x1:x2, data = my_df)`
`lm(formula = y ~ x1 + x2 + I(x1 * x2), data = my_df)`
	$Y_i = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2 + \epsilon_i$

`lm(formula = y ~ x1 + I(x1^2), data = my_df)`	$Y_i = \beta_0 + \beta_1 X_1 + \beta_2 X_1^2 + \epsilon_i$
`lm(formula = y ~ x1 + log(x2), data = my_df)`	$Y_i = \beta_0 + \beta_1 X_1 + \beta_2 \ln(X_2) + \epsilon_i$
…	…

Source: https://stat.ethz.ch/R-manual/R-devel/library/stats/html/formula.html

Lab 3.6 Logistic Regression

- `my_model <- glm(formula = …, data = …, family=binomial)`
- `summary(my_model)`

- `predict(my_model, newdata = …, type = "response")`
  - Set the argument `type = "response"` to get predicted probabilities, i.e., $P(Y = 1|X)$
  - Otherwise, `predict(my_model)` gives log odds (logit)
  - If the `newdata` argument is not supplied, the prediction is applied on the training data set
  - Use `contrast()` to find out which y category is set to 1.

- Construct confusing matrix
  - Convert probability prediction to binary prediction (cutoff prob.)
  - `table()`

Lab 3.6 Training & Test Set

- Training and test set split
  - For time series data, need to respect the time when splitting the data
    - That is, train on early data, test on late data
  - Otherwise, randomly split data to train and test

```
# randomly split Auto dataset into training and test set
num_rows <- nrow(Auto)
train_fraction <- 0.7
train_idx = sample(1:num_rows, size = round(num_rows * train_fraction))
train_data <- Auto[train_idx, ]
test_data <- Auto[-train_idx, ]
```