Jackknife,Jackknife+ and their implementations

Introduction

Estimating the underlying distribution of the statistics of interest is a popular topic in the field of statistics. For instance, we are often interested in the sample mean. When the sample size is large enough, the sample means are approximately normally distributed based on the central limit theorem. However, when the sample size is small, the distribution of the sample mean becomes problematic, and the central limit theorem cannot be applied. Resampling methods such as jackknife are known to handle datasets with unknown distributions well.

Moreover, resampling methods can be applied to perform predictive inference. In many cases, we aim to make predictions about new data based on a fitted model. By using jackknife and jackknife+, we can estimate the distribution of prediction errors and construct confidence intervals for our predictions.

In this project, we discussed jackknife and its extension, jackknife+. We identified two uses for jackknife: assessing the uncertainty of parameter estimation, and estimating the prediction error and constructing prediction intervals. For the first purpose, we implemented jackknife on simulated data, fitted linear models, and constructed confidence intervals for the computed linear model coefficient. For the second purpose, we applied jackknife and jackknife+ to predictive inference on linear models with real datasets. We also presented visualizations.

Methods

Jackknife

Jackknife is a resampling technique that can be used to generate the distribution of the statistics of interest. With the distribution information of the statistics, we can generate the confidence interval of the parameter estimate. In addition, jackknife is also useful in computing the prediction intervals. In this section, we first illustrate the Jackknife methods and show

how it can be used to calculate the confidence interval for simple statistics such as mean and variance. Then, we apply jackknife on linear regression models to compute the confidence interval for regression coefficients. After that, we show how compute prediction interval using Jackknife.

Parameter estimation with jackknife (mean and variance)

The jackknife is performed by removing a single observation, calculating the statistics of interest without that observation, and repeat the process for each observation in the data.

Specifically, suppose θ is the parameter of interest. We denote the estimated parameter by $\hat{\theta} = f_n(x_1, \cdots, x_n)$. Note that $\hat{\theta}$ varies for each sample. By definition, the bias of $\hat{\theta}$ is calculated by

$$bias(\hat{\theta}) = E(\hat{\theta}) - \theta.$$

To approximate $E(\hat{\theta})$, we construct jackknife replicates as follows:

$$\hat{\theta}_{(i)} = f_{n-1}(x_1, \cdots, x_{i-1}, x_{i+1}, \cdots, x_n) \text{ for } i = 1, \cdots, n,$$

where $\hat{\theta}_{(i)}$ is the with i-th observation being removed. Then, the jackknife estimate is

$$\hat{\theta}_{jack} = \frac{1}{n}\sum_{i=1}^{n} \hat{\theta}_{(i)},$$

and the bias can be computed by $\widehat{bias}_{jack} = (n-1)(\hat{\theta}_{jack} - \theta)$. And the resulting biased corrected by the jackknife can be calculated by

$$\hat{\theta}^{*} = \hat{\theta} - \widehat{bias}_{jack}(\hat{\theta}).$$

In addition, the confidence interval can be computed by Equation 1.

$$\hat{\theta} \pm se_{\theta} t_{v,\frac{\alpha}{2}}. \tag{1}$$

Note that when computing the confidence interval using t-statistic, we are assuming $\hat{\theta}$ is normally distributed.

Adjustment of variance for resampling

Suppose interested in the distribution of mean and variance of a dataset. It is widely known that sample mean is an unbiased estimator of population mean Equation 2. The μ is represents the population men, x_i represents the i-th observation, and N represents the total number of observations.

$$\mu = \frac{\sum_{i=1}^{n} x_i}{N} \tag{2}$$

The sample variance

$$S^2 = \frac{(x_i - \bar{x})^2}{n-1} \tag{3}$$

is calculated by `var` in R. To make the sample variance an unbiased estimator for population variance, we need to slightly adjust the sample variance from Equation 3 to Equation 4.

$$S'^2 = \frac{(x_i - \bar{x})^2}{n} = \frac{(n-1)}{n} S^2 \tag{4}$$

Based on the adjustment above, we create a function `variance` to calculate the unbiased estimator of population variance.

```
variance <- function(x) {
    n <- length(x)
    if (n < 2) {
        return(NA)
    } else {
        return(var(x) * (n - 1)/n)
    }
}
```

The jackknife function for simple statistics

After creating the unbiased estimator, we create a jackknife function to estimate the statistics of interest. The bias and standard error are also presented. Inputs of the function are `data` and `statistics`, where `data` is a vector input of the variable, and `statistics` is the function that is used to calculate the statistics of interest, which can be mean, variance, standard error, etc. The output have three components, `results`, `bias`, and `se`. `results` include the jackknife estimation for each iteration. `bias` and `se` are the bias and standard error of the jackknife estimate.

```
jackknife_ss <- function(data, statistics) {
    # data: A vector input statistics: The function of the
    # statsitics of interest(mean,variance,etc)
    n <- length(data)
    results <- numeric(n)
    for (i in 1:n) {
        data_i <- data[-i]
        results[i] <- statistics(data_i)
    }

    # Calculate the bias and standard error of the estimate
    bias <- (n - 1) * (mean(results) - statistics(data))
    se <- sqrt(variance(results)/n)

    return(list(results = results, bias = bias, se = se))
}
```

Jackknife function for regression coefficients

To obtain the distribution of regression coefficients, we iteratively hold one sample out and fit the model. The input of `jackknife_reg` are `data` and `response`, where `data` is the data without response variable, and `response` should be the response variable vector. The output of the function is a dataframe consisting all regression coefficients from all fitted models. Based on this output, the confidence interval can be presented. In the result section, we show the dataframe with estimated results. In addition, visualization is also presented.

```
jackknife_reg <- function(data, response) {
    # data: matrix or dataframe without the response
    # variable response: the response variable
    n <- nrow(data)
    predictions <- rep(0, n)
    model_list <- list()

    # Compute leave-one-out predictions
    for (i in 1:n) {
        data_i <- data[-i, ]
        response_i <- response[-i, ] %>%
            unlist()
        df_i <- cbind(data_i, response_i)
        model_i <- lm(response_i ~ ., data = df_i)
        model_list[[i]] <- model_i
        predictions[i] <- predict(model_i, newdata = data[i,
            ])
    }

    coefficients_list <- lapply(model_list, coef)
    coefficients_df <- do.call(rbind, coefficients_list) %>%
        as.data.frame()
    # Add row names to the dataframe

    row.names(coefficients_df) <- paste0("Model", 1:n)
    colnames(coefficients_df) <- c("a1", "a2")
    return(coefficients_df)
}
```

Jackknife for predictive inference

The jackknife method can be further incorporated with regression techniques to generate confidence interval for the coefficients of interest for the regression model. Given the training data with n observations, we fit the model n times with the i-th observation being removed, and then compute the leave one out residuals.

Let $\hat{\mu}$ be the regression function fitted for training data $(X_1, Y_1), \cdots, (X_n, Y_n)$. We can write

$$\hat{\mu} = f_n((X_1, Y_1), \cdots, (X_n, Y_n)).$$

Then, to compute the leave-one-out residual, we let

$$\hat{\mu}_{-i} = f_{n-1}((X_1, Y_1), \cdots, (X_{i-1}, Y_{i-1}), (X_{i+1}, Y_{i+1}), \cdots, (X_n, Y_n)),$$

and then the jackknife prediction interval is

$$C_{n,\alpha}^{jackknife}(X_{n+1}) = \hat{\mu}(X_{n+1}) \pm se(R_i^{LOO}) t_{n-1,\frac{\alpha}{2}}$$

where $R_i^{LOO} = |Y_i - \hat{\mu}_{-i}(X_i)|$, which is the i-th leave-one-out residual.

Jackknife function for prediction intervals

```r
jackknife <- function(data, response, test_data, conf_level = 0.95) {
    # data: matrix or dataframe without the response
    # variable response: the response variable test_data:
    # matrix or dataframe of the test points to predict
    # conf_level: desired confidence level (default is
    # 0.95)

    n <- nrow(data)
    p <- ncol(data)
    predictions <- rep(0, n)
    model_list <- list()
    residuals <- rep(0, n)

    # Compute leave-one-out predictions and models
    for (i in 1:n) {
        data_i <- data[-i, ]
        response_i <- response[-i]
        model_i <- lm(response_i ~ lstat + crim + rm + dis +
            black + chas + nox + rad + tax + ptratio + I(lstat^2) +
            I(rm^2), data = data_i)
        model_list[[i]] <- model_i
        predictions[i] <- predict(model_i, newdata = data[i,
            , drop = FALSE])
        residuals[i] <- response[i] - predictions[i]
    }

    # Fit a model on the full training data
    full_model <- lm(response ~ lstat + crim + rm + dis + black +
        chas + nox + rad + tax + ptratio + I(lstat^2) + I(rm^2),
        data = data)

    # Predict test points using the full model
    test_point_count <- nrow(test_data)
    full_preds <- predict(full_model, newdata = test_data)

    # Compute the Jackknife confidence intervals
    t_stat <- qt(conf_level + (1 - conf_level)/2, n - p - 1)
    R_jack <- sqrt((n - 1) * mean(residuals^2))
    conf_interval <- t_stat * R_jack/sqrt(n)

    lower_bounds <- full_preds - conf_interval
    upper_bounds <- full_preds + conf_interval

    result <- data.frame(Test_Point = 1:test_point_count, Lower_Bound = lower_bounds,
        Upper_Bound = upper_bounds, Full_Prediction = full_preds)

    return(result)
}
```

Jackknife+ for prediction intervals

Jackknife+ is a modification of jackknife methods. Instead of centering the interval on the predicted value $\mu(X_n + 1)$ fitted on the full training data, jackknife+ use the leave-one-out predictions for the test point.

$$C_{n,\alpha}^{jackknife}(X_{n+1}) = \hat{\mu}_{-i}(X_{n+1}) \pm se(R_i^{LOO})t_{n,\alpha}$$

In addition, Jackknife+ prediction interval is constructed around median.

```
jackknife_plus <- function(data, response, test_data, conf_level = 0.95) {
    # data: matrix or dataframe without the response
    # variable response: the response variable test_data:
    # matrix or dataframe of the test points to predict
    # conf_level: desired confidence level (default is
    # 0.95)

    n <- nrow(data)
    p <- ncol(data)
    predictions <- rep(0, n)
    model_list <- list()
    residuals <- rep(0, n)

    # Compute leave-one-out predictions and models
    for (i in 1:n) {
        data_i <- data[-i, ]
        response_i <- response[-i]
        model_i <- lm(response_i ~ lstat + crim + rm + dis +
            black + chas + nox + rad + tax + ptratio + I(lstat^2) +
            I(rm^2), data = data_i)
        model_list[[i]] <- model_i
        predictions[i] <- predict(model_i, newdata = data[i,
            , drop = FALSE])
        residuals[i] <- response[i] - predictions[i]
    }

    # Predict test points using the leave-one-out
    # predictions
    test_point_count <- nrow(test_data)
    loo_preds <- matrix(0, nrow = test_point_count, ncol = n)
    for (i in 1:n) {
        loo_preds[, i] <- predict(model_list[[i]], newdata = test_data)
    }
    loo_medians <- apply(loo_preds, 1, median)  # compute median of leave-one-out predicti
    loo_std_errors <- apply(loo_preds, 1, sd) * sqrt((n - 1)/n)  # compute standard errors

    # Compute the Jackknife+ confidence intervals
    t_stat <- qt(conf_level + (1 - conf_level)/2, n - p - 1)
    R_jackplus <- sqrt((n - 1) * mean(residuals^2))
    conf_interval_plus <- t_stat * R_jackplus/sqrt(n) + t_stat *
        loo_std_errors

    lower_bounds_plus <- loo_medians - conf_interval_plus
    upper_bounds_plus <- loo_medians + conf_interval_plus

    result <- data.frame(Test_Point = 1:test_point_count, Lower_Bound_Plus = lower_bounds_
        Upper_Bound_Plus = upper_bounds_plus, Leave_One_Out_Median = loo_medians)
    return(result)
}
```

Results

Simple statistics

We use a simulated data called `sim1` available in `modelr` package. The simulated data has 30 observations and 2 variables, which are `x` and `y`, where y is the response variable. Figure 1 below shows a scatterplot of `sim1`.

```
ggplot(sim1, aes(x, y)) + geom_point()
```

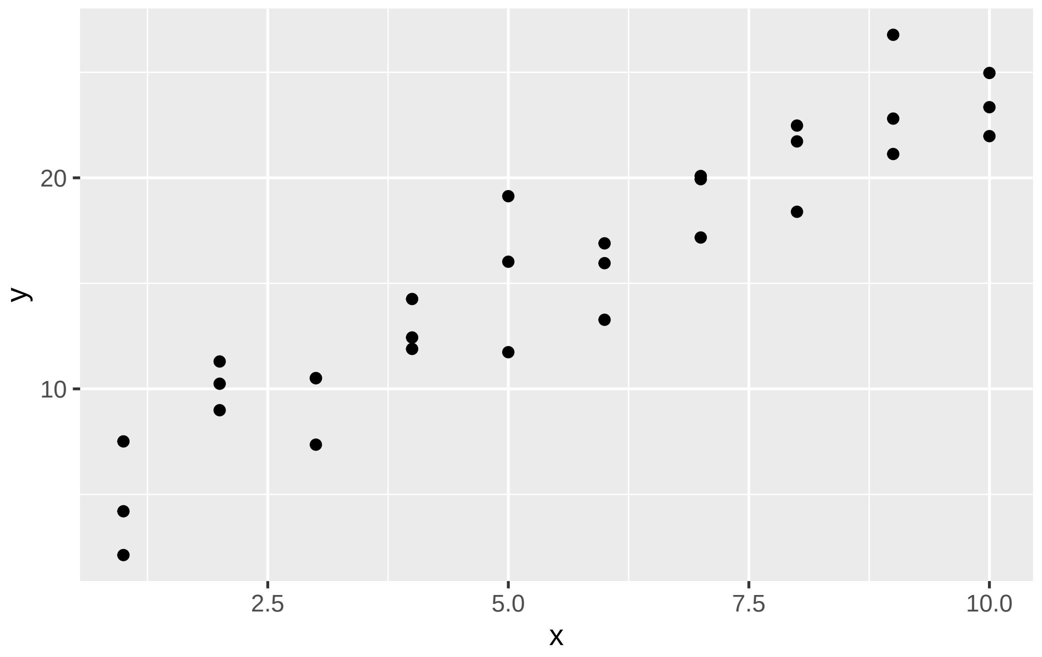

Figure 1: Scatterplot of sim1

Applying the `jackknife` function to simulated data `sim1`, the variance and the mean of the simulated data `sim1` is calculated as follows.

```
jackknife_ss(sim1$y, mean)
```

```
$results
 [1] 15.89406 15.77990 15.96559 15.72892 15.68567 15.64934 15.78522 15.67663
 [9] 15.67641 15.61010 15.62879 15.54723 15.37923 15.63412 15.48630 15.58116
[17] 15.48868 15.45630 15.34626 15.44675 15.35142 15.28971 15.40471 15.26386
[25] 15.11554 15.25250 15.31032 15.17791 15.23383 15.28112

$bias
[1] 0

$se
[1] 0.03944304
```

```
jackknife_ss(sim1$y, variance)
```

```
$results
 [1] 36.04672 38.32580 34.22020 39.09088 39.61777 39.97368 38.23698 39.71375
 [9] 39.71598 40.26903 40.13985 40.54975 40.13620 40.09917 40.59549 40.42772
[17] 40.59788 40.53618 39.85632 40.50596 39.90445 39.22434 40.30791 38.87154
[25] 36.07216 38.70376 39.47686 37.41024 38.41136 39.11147

$bias
[1] -1.353505

$se
[1] 0.275646
```

The results are summarized as shown in Table 1.

Table 1: Jackknife results on sim1

Statistics	$\hat{\theta}_{jack}$	se	95% CI
Mean	15.500	0.039	(15.420, 15.580)
Variance	39.205	0.276	(38.641, 39.769)

Regression Coefficients

To compute the result for regression coefficients estimation, we use the `sim1` dataset.

```
a <- jackknife_reg(sim1[1], sim1[2])
ci_a1 <- c(mean(a$a1) + qt(0.025, df = 29) * sd(a$a1)/sqrt(30),
    mean(a$a1) - qt(0.025, df = 29) * sd(a$a1)/sqrt(30))
ci_a2 <- c(mean(a$a2) + qt(0.025, df = 29) * sd(a$a2)/sqrt(30),
    mean(a$a2) - qt(0.025, df = 29) * sd(a$a2)/sqrt(30))
```

Table 2: CI for regression coefficient on Sim1

	Lower Limit	Upper Limit	Estimated value
a_1	4.152399	4.290375	4.221387
a_2	2.040739	2.062293	2.051516

The model is:

$$y_i = a_1 + a_2 x_i + \epsilon$$

The fitted model with the mean of the coefficients is:

$$\hat{y}_i = 4.221387 + 2.051516x_i$$

Figure 2 shows all 30 possible fitted models for `sim1`.

```
ggplot(sim1, aes(x, y)) + geom_point(size = 2, colour = "grey30") +
    geom_abline(aes(intercept = a1, slope = a2, color = row.names(a)),
        data = a %>%
            as.data.frame()) + scale_color_manual(name = "Model",
    values = rainbow(nrow(a)))
```

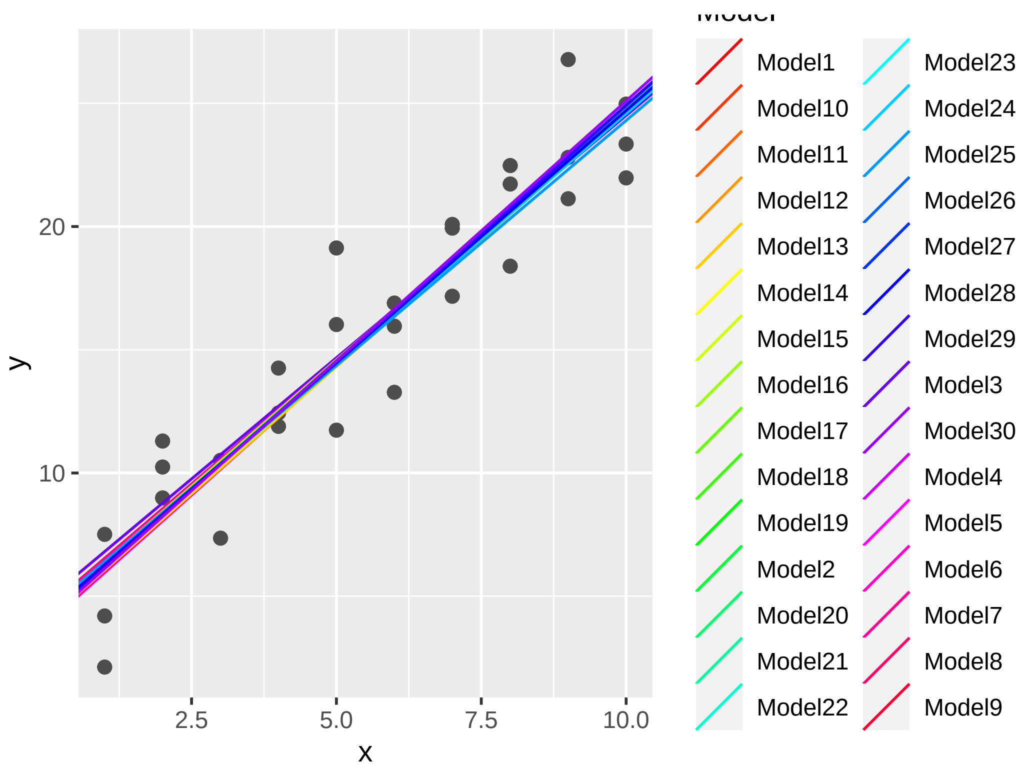

Figure 2: All possible fitted models

Figure 3 shows the distribution of the intercept a1. The blue line is the mean of the intercept estimation.

```
ggplot(data.frame(a[1]), aes(x = a1)) + geom_histogram(fill = "blue",
    alpha = 0.3) + xlab("Intercept value") + ylab("Count") +
    geom_vline(xintercept = mean(a$a1), color = "blue") + theme_bw()
```

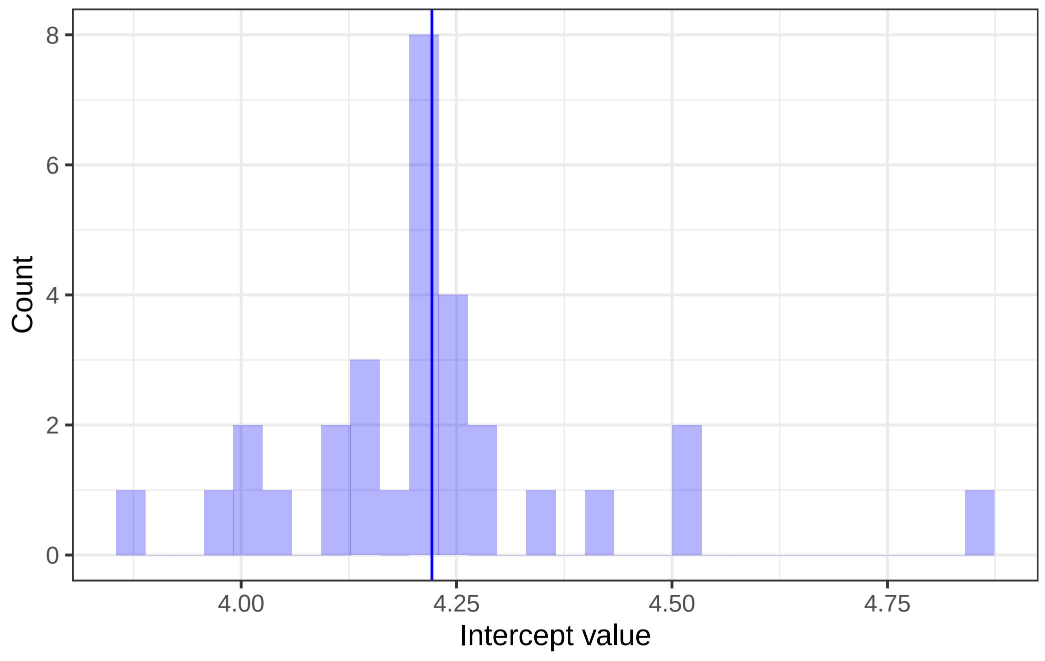

Figure 3: Distribution-of-a1

Figure 4 shows the distribution of the coefficient a2. The red line is the mean of the intercept estimation.

```r
ggplot(data.frame(a[2]), aes(x = a2)) + geom_histogram(fill = "red",
    alpha = 0.3) + xlab("a2 values") + geom_vline(xintercept = mean(a$a2),
    color = "red") + theme_bw()
```

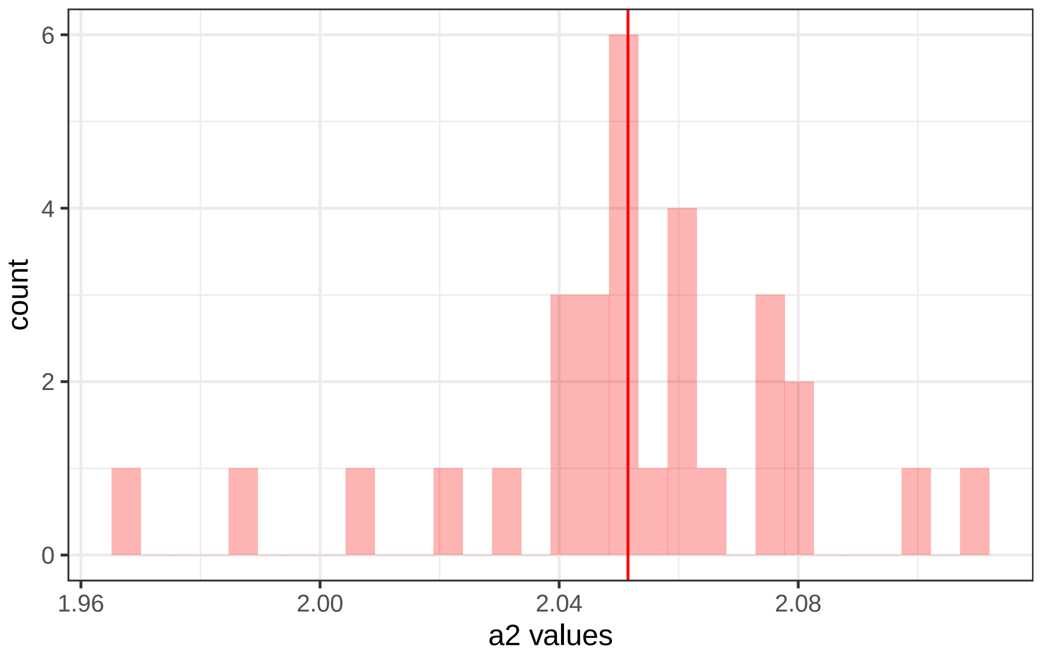

Figure 4: Distribution-of-a2

Prediction intervals

The example dataset we use to generate prediction interval is `Boston`, which is a famous dataset available in the **`MASS`** package in R, and it contains information on housing values in suburbs of Boston. The dataset contains 506 observations on 14 variables. Since the goal of this part is to illustrate and compare the predictive interval of the jackknife and jackknife+, omit the model fitting process and directly implement the linear model.

$$\text{medv} = \beta_0 + \beta_1\text{lstat} + \beta_2\text{crim} + \beta_3\text{rm} + \beta_4\text{dis} + \beta_5\text{black} + \beta_6\text{chas} + \beta_7\text{nox} + \beta_8\text{rad} + \beta_9\text{tax} + \beta_{10}\text{ptratio} + \beta_{11}\text{L}$$

After training the model on the training data, the code uses the jackknife+ resampling method to estimate the prediction error and construct confidence intervals for the predictions. Finally, the median prediction and upper and lower bounds of the confidence interval are saved in a data frame.

```r
set.seed(34)
data("Boston")
n <- nrow(Boston)
train_indices <- sample(1:n, size = 0.8 * n)
train_data <- Boston[train_indices, ]
```

```r
test_data <- Boston[-train_indices, ]
response <- train_data$medv
test_response <- test_data$medv
train_data_no_response <- train_data[, -which(names(train_data) ==
    "medv")]

test_data_no_response <- test_data[, -which(names(test_data) ==
    "medv")]

Result <- jackknife(train_data_no_response, response, test_data_no_response)
result <- cbind(Result, test_response)
result_plus <- jackknife_plus(train_data_no_response, response,
    test_data_no_response)
head(result_plus)
```

```
  Test_Point Lower_Bound_Plus Upper_Bound_Plus Leave_One_Out_Median
1          1        24.935880         42.15675             33.54632
2          2         8.664823         25.98338             17.32410
3          3        10.396245         27.68281             19.03953
4          4        12.040952         29.25747             20.64921
5          5         7.582634         24.78698             16.18480
6          6        10.161597         27.35293             18.75726
```

```r
result_plus <- cbind(result_plus, test_response)
```

```r
head(result)
```

```
   Test_Point Lower_Bound Upper_Bound Full_Prediction test_response
3           1   24.992587    42.10004        33.54631          34.7
10          2    8.771072    25.87852        17.32480          18.9
13          3   10.485811    27.59326        19.03954          21.7
16          4   12.095738    29.20319        20.64946          19.9
18          5    7.631158    24.73861        16.18488          17.5
20          6   10.203803    27.31125        18.75753          18.2
```

We plot the predicted intervals obtained from jackknife as shown in Figure 5 below. The red points are true values.

```
ggplot(result, aes(x = Test_Point, y = Full_Prediction)) + geom_point(alpha = 0.8) +
    geom_errorbar(aes(ymin = Lower_Bound, ymax = Upper_Bound),
        width = 0.2, alpha = 0.4) + labs(x = "True Values (y_test)",
    y = "Predicted Values (preds)") + theme_bw() + geom_point(aes(x = Test_Point,
    y = test_response), color = "red")
```

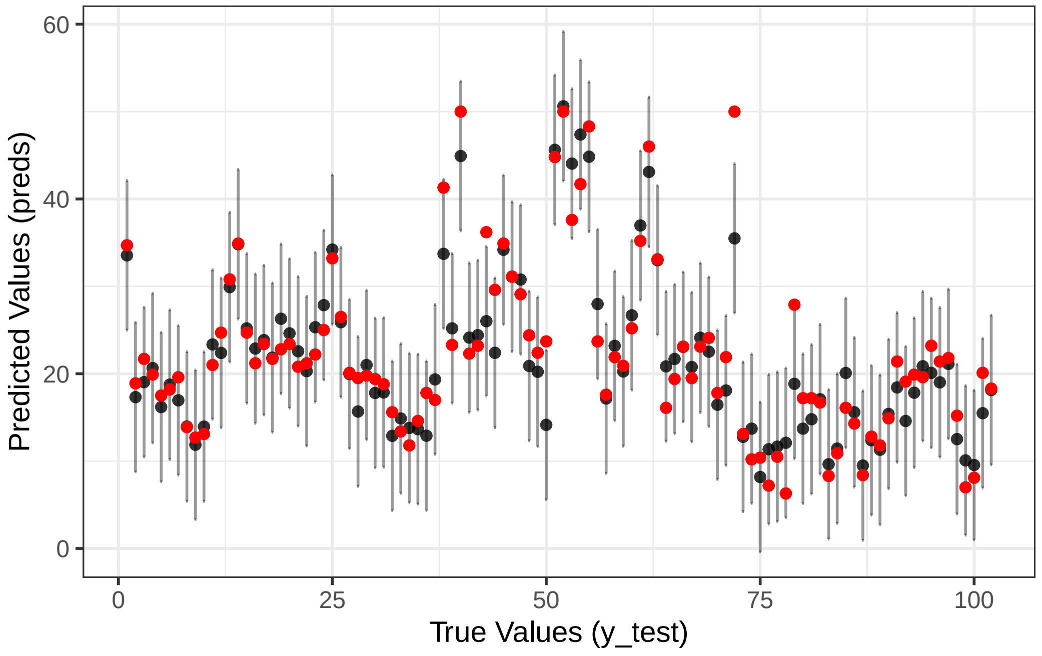

Figure 5: Prediction interval from jackknife

We plot the predicted intervals obtained from jackknife+ as shown in Figure 6 below. The red points are true values.

```
ggplot(result_plus, aes(x = Test_Point, y = Leave_One_Out_Median)) +
    geom_point(alpha = 0.8) + geom_errorbar(aes(ymin = Lower_Bound_Plus,
    ymax = Upper_Bound_Plus), width = 0.2, alpha = 0.4) + labs(x = "True Values (y_test)",
    y = "Predicted Values (preds)") + theme_bw() + geom_point(aes(x = Test_Point,
    y = test_response), color = "red")
```

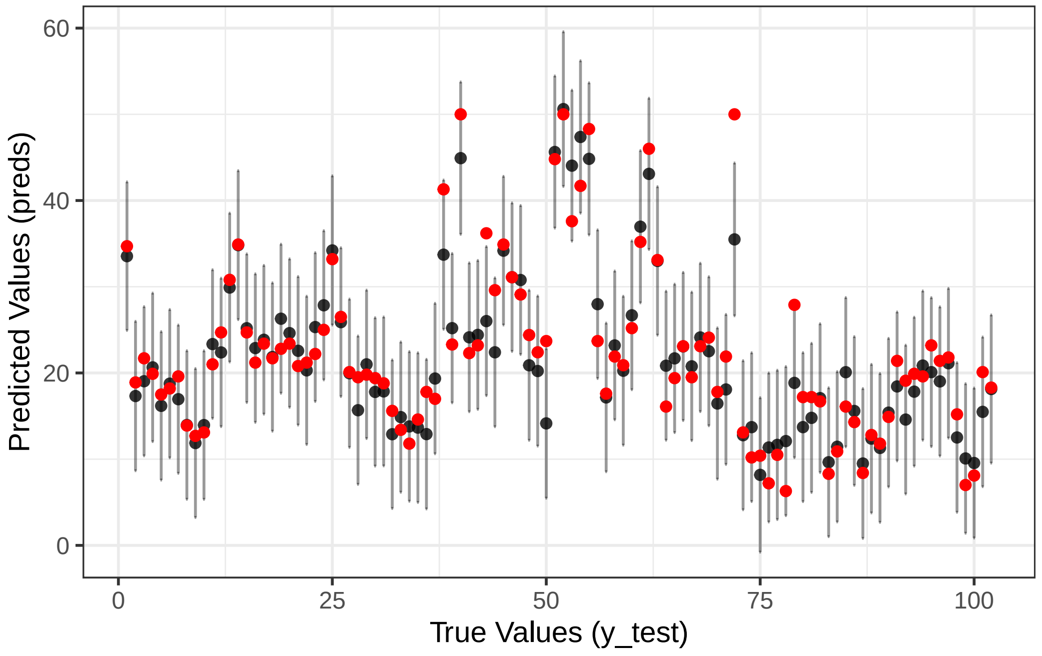

Figure 6: Prediction interval from jackknife+

We are interested in the performance of prediction with different sample size. Reducing the sample size in training set, we have

```
  Test_Point Lower_Bound_Plus Upper_Bound_Plus Leave_One_Out_Median
1          1         17.27792         36.66594             26.97193
2          2         12.68630         31.70367             22.19499
3          3         22.98935         42.77242             32.88088
4          4         22.79698         42.22446             32.51072
5          5         21.44805         41.24752             31.34778
6          6         16.82045         35.46727             26.14386
  test_response
1          24.0
2          21.6
3          34.7
4          33.4
5          36.2
6          28.7
```

We calculate and compare the MSEs from the two resampling methods with different sample size. The result is shown as follows Table 3.

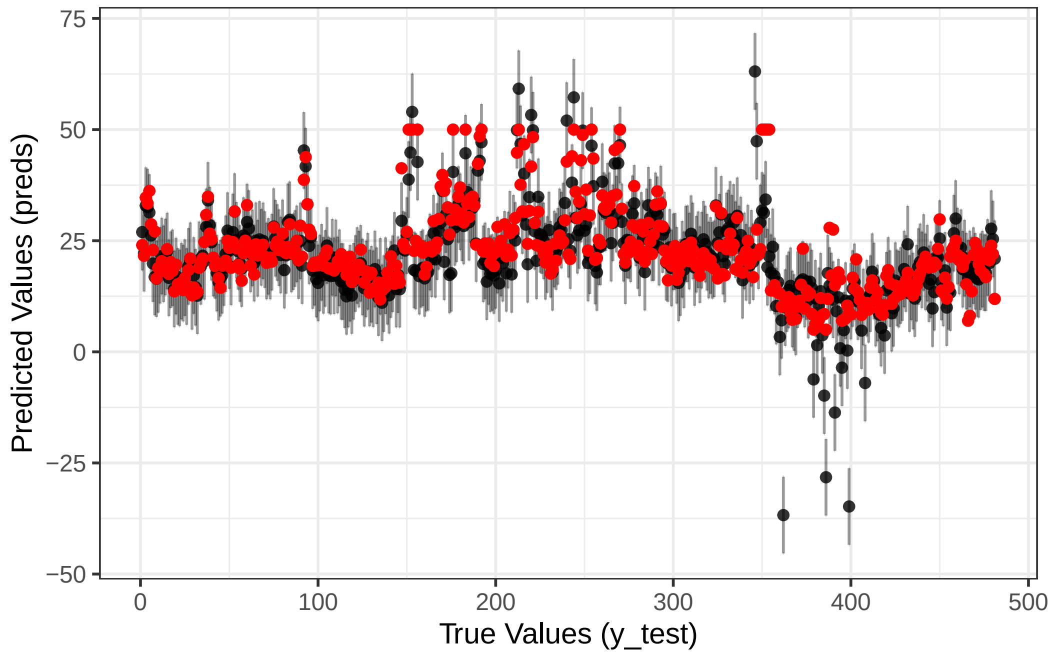

Figure 7: Prediction interval from jackknife with n=25

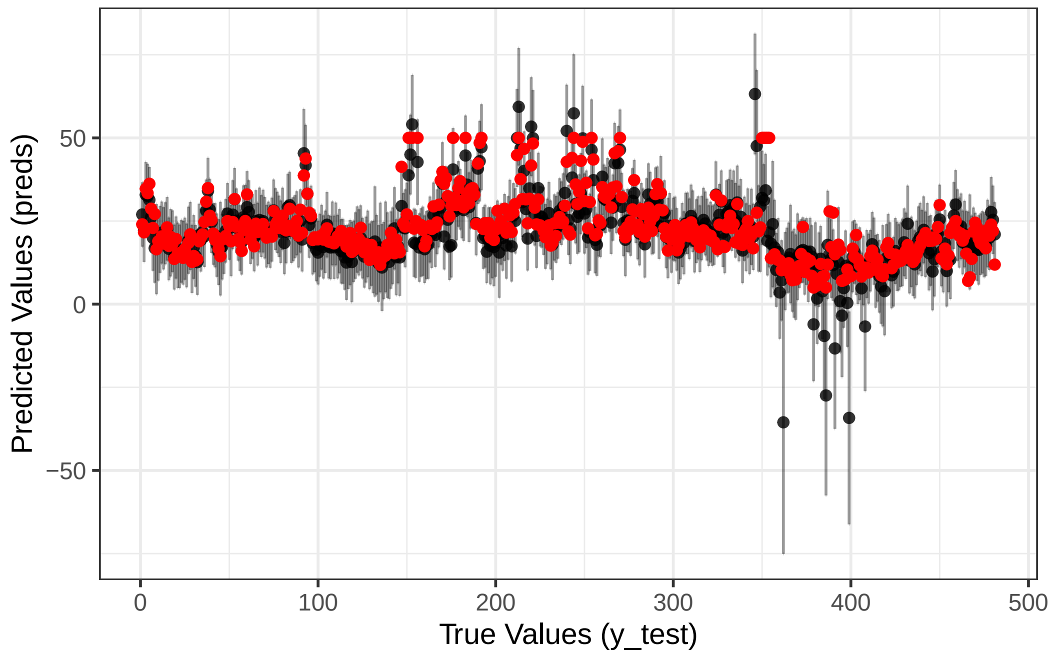

Figure 8: Prediction interval from jackknife with n=25

Table 3: Jackknife and jackknife+ results on Boston

	MSE of n=25	MSE of n=405
Jackknife	3.674	2.374
Jackknife+	3.664	2.374

Conclusion

In this project, we implemented the jackknife to generate the confidence interval for parameter estimation and predictions. In addition, for prediction interval, we compared jackknife and jackknife+ with different size of data. The result is that, for large training data, jackknife+ have a similar performance as jackknife; for small dataset, jackknife+ have a slightly better performance compared with jackknife.

When applying resampling methods to construct predictive intervals, the linear models can be extend to other regression models or even tree based methods to generate the predictive inference. Besides, jackknife+ can be further extend to CV+, where multiple samples are randomly hold out for estimation.

Efron, B. and Tibshirani, R. (1986) Bootstrap Methods for Standard Errors, Confidence Intervals, and Other Measures of Statistical Accuracy. Statistical Science, 1, 54-75. http://www.jstor.org/stable/2245500

Barber, Rina & Candès, Emmanuel & Ramdas, Aaditya & Tibshirani, Ryan. (2021). Predictive inference with the jackknife+. Annals of Statistics. 49. 486-507. 10.1214/20-AOS1965.