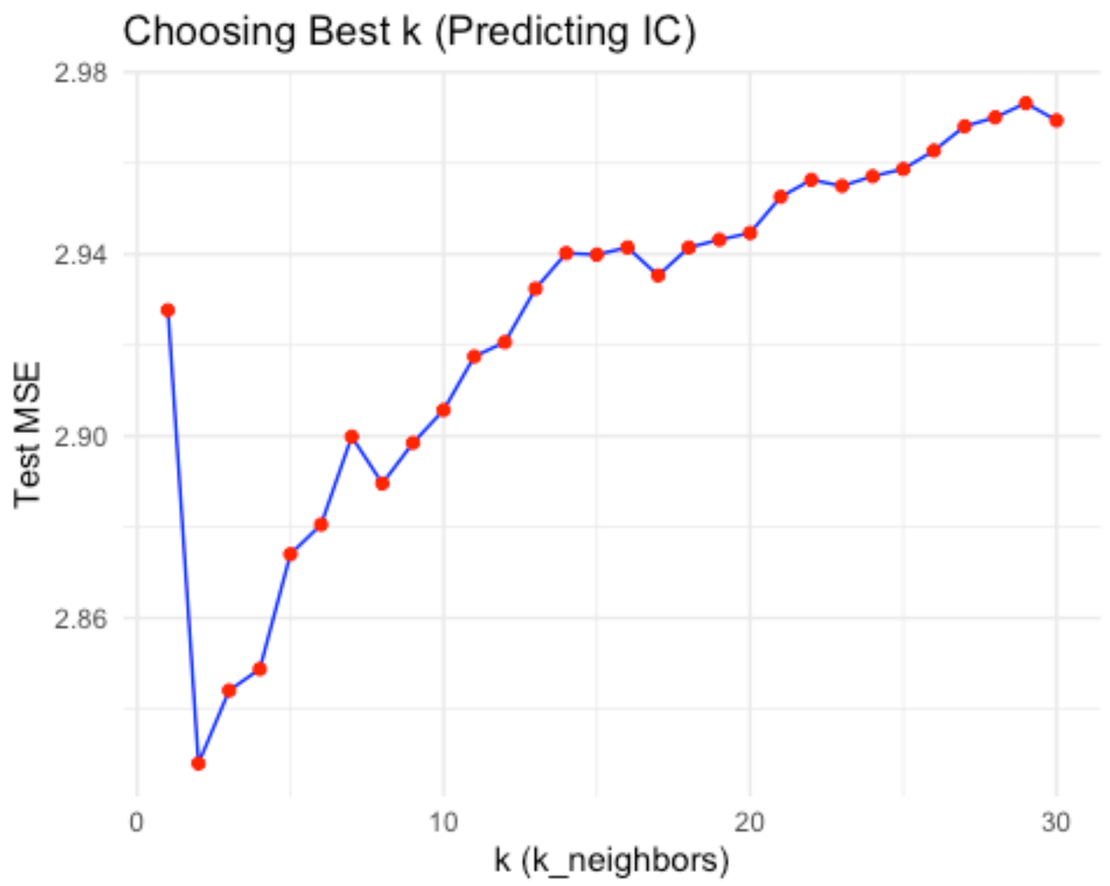
PM project KNN

2025-04-28

```
df <- data_raw %>%
  mutate(
    observation_date = ymd(observation_date),
    year = year(observation_date),
    month = month(observation_date),
    quarter = quarter(observation_date),
    time_since_start = as.numeric(difftime(observation_date,
min(observation_date), units = "weeks")) / 52
  ) %>%
  arrange(observation_date)

# --- Select important columns and refine them ---
important_columns <- c(
  "UMCSENT_interp", "hourly_earning", "BBKMGDP",
  "FEDFUNDS", "num_losers", "unemployment_level",
  "CPI", "monthly_average", "IC", "CC"
)

# clean data: fill NA
df <- df %>%
  mutate(across(all_of(important_columns), ~ ifelse(is.na(.), mean(., na.rm =
TRUE), .)))

# Rename monthly_average to NASDAQ_Index
df <- df %>%
  rename(NASDAQ_Index = monthly_average)

# --- Choose features ---
X_features <- c(
  "UMCSENT_interp", "hourly_earning", "BBKMGDP", "num_losers",
  "unemployment_level", "FEDFUNDS", "NASDAQ_Index", "CPI",
  "year", "month", "quarter", "time_since_start"
)

X_data <- df %>% select(all_of(X_features))  # Predictors
y_data <- df$IC                               # Target

# Combine scaling and splitting into a cleaner workflow

# scale y
y_mean <- mean(y_data)
y_sd <- sd(y_data)
y_scaled <- (y_data - y_mean) / y_sd

# scale X
```

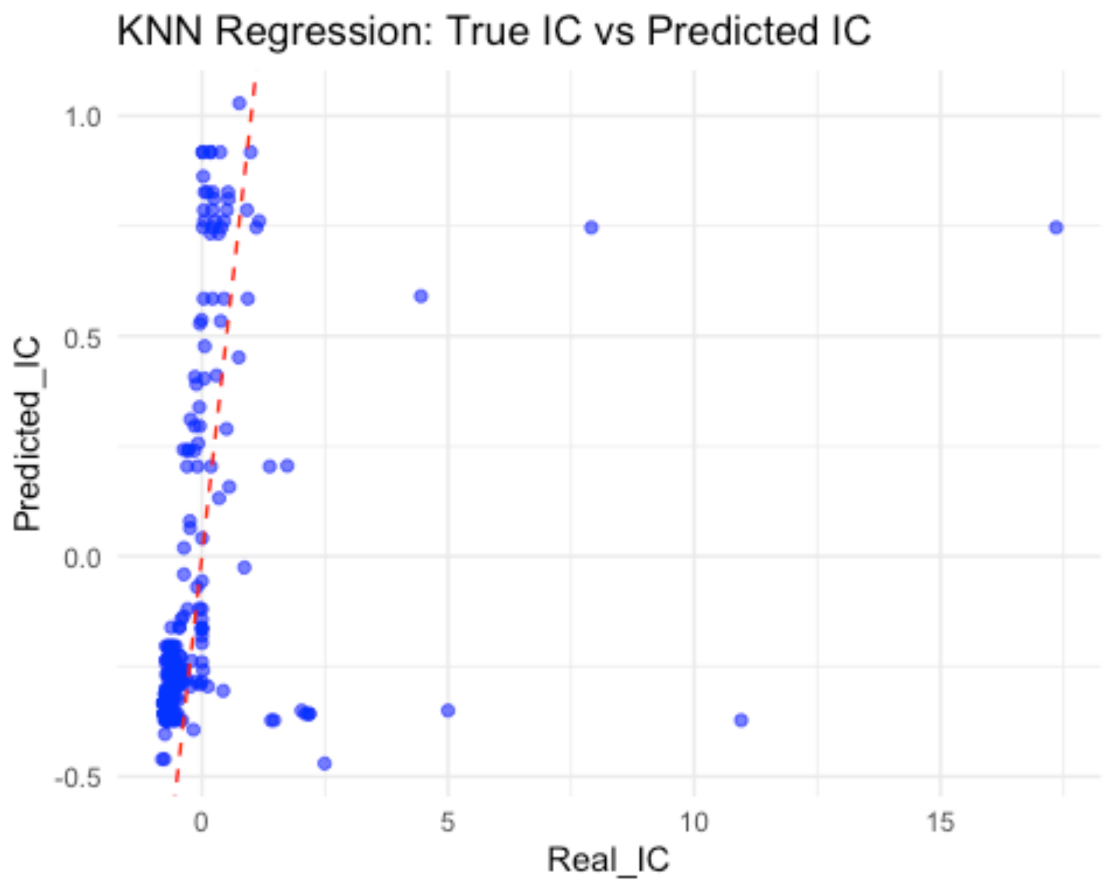
```r
X_means <- colMeans(X_data)
X_sds <- apply(X_data, 2, sd)
X_scaled <- (X_data - X_means) / X_sds

n_obs <- nrow(X_data)
split_point <- floor(0.7 * n_obs)

# Create training and test sets from scaled data
X_train <- X_scaled[1:split_point, ]
X_test <- X_scaled[(split_point+1):n_obs, ]
y_train <- y_scaled[1:split_point]
y_test <- y_scaled[(split_point+1):n_obs]

# --- choose k values, find MSE for each k ---
k_values <- 1:30
mse_list <- numeric(length(k_values))

for (i in seq_along(k_values)) {
  knn_fit <- knn.reg(train = X_train, test = X_test, y = y_train, k =
k_values[i])
  mse_list[i] <- mean((knn_fit$pred - y_test)^2)
}

# --- find the best k ---
best_k <- k_values[which.min(mse_list)]
cat("Best k", best_k, "\n")

## Best k 2

# --- plot k & MSE graph ---
k_plot_data <- data.frame(k = k_values, MSE = mse_list)

ggplot(k_plot_data, aes(x = k, y = MSE)) +
  geom_line(color = "blue") +
  geom_point(color = "red") +
  labs(
    title = "Choosing Best k (Predicting IC)",
    x = "k (k_neighbors)",
    y = "Test MSE"
  ) +
  theme_minimal()
```



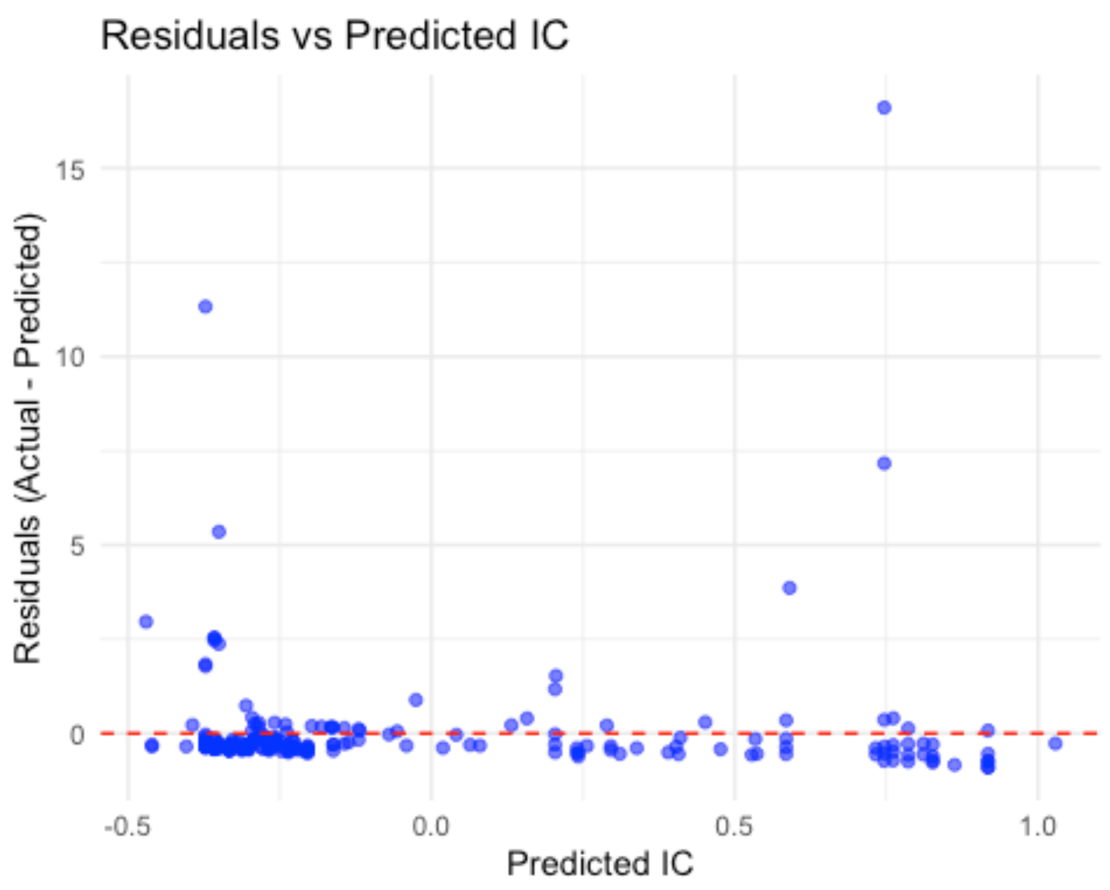
```r
# --- Use best k found to build knn model ---
final_knn_model <- knn.reg(train = X_train, test = X_test, y = y_train, k =
best_k)
y_pred_final <- final_knn_model$pred

# --- calculate the final model MSE ---
final_mse <- mean((y_pred_final - y_test)^2)
cat("Final best MSE", round(final_mse, 2), "\n")
```

```
## Final best MSE 2.83
```

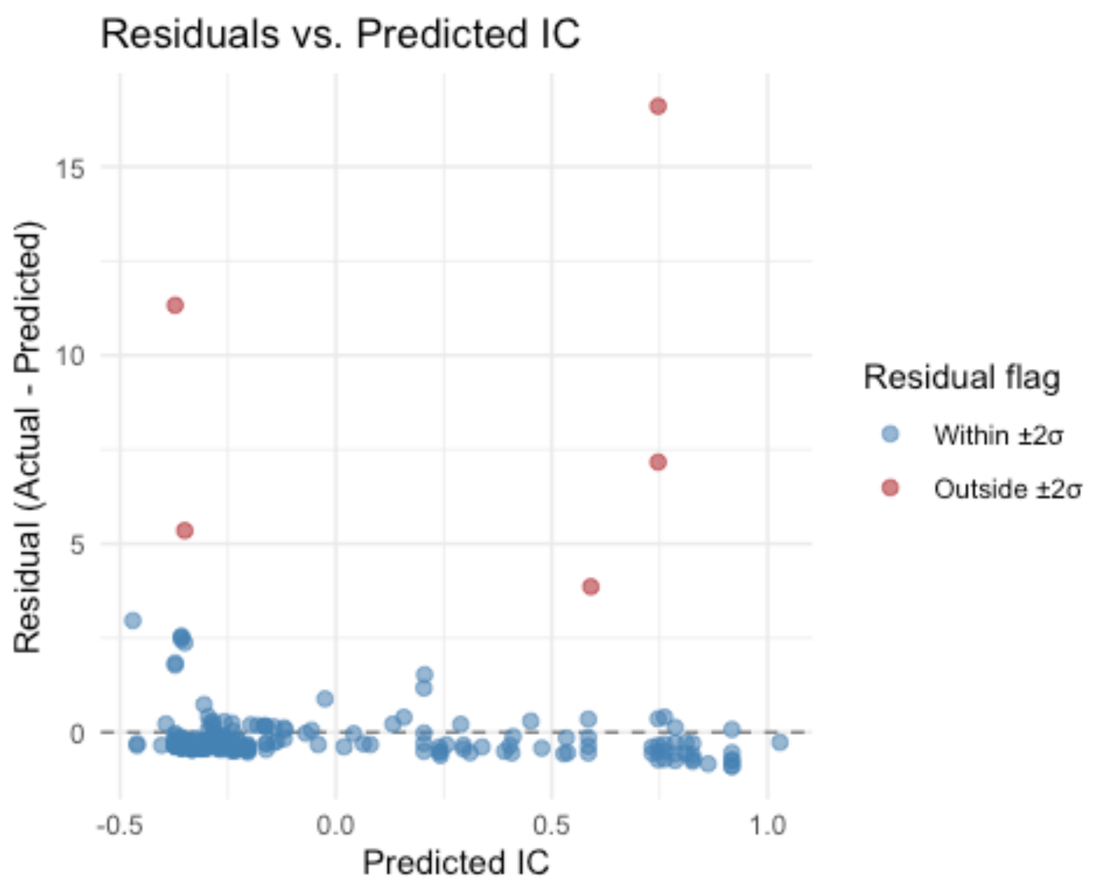
```r
# --- plot real vs. predicted y ---
result_plot_data <- data.frame(True = y_test, Predicted = y_pred_final)

ggplot(result_plot_data, aes(x = True, y = Predicted)) +
  geom_point(color = "blue", alpha = 0.6) +
  geom_abline(intercept = 0, slope = 1, color = "red", linetype = "dashed") +
  labs(
    title = "KNN Regression: True IC vs Predicted IC",
    x = "Real_IC",
    y = "Predicted_IC"
  ) +
  theme_minimal()
```



```r
# --- Calculate Residuals ---
residuals <- y_test - y_pred_final

# --- Prepare plot ---
residual_plot_data <- data.frame(
  Predicted = y_pred_final,
  Residuals = residuals
)

# --- Plot Residuals vs Predicted ---
ggplot(residual_plot_data, aes(x = Predicted, y = Residuals)) +
  geom_point(color = "blue", alpha = 0.6) +
  geom_hline(yintercept = 0, color = "red", linetype = "dashed") +
  labs(
    title = "Residuals vs Predicted IC",
    x = "Predicted IC",
    y = "Residuals (Actual - Predicted)"
  ) +
  theme_minimal()
```



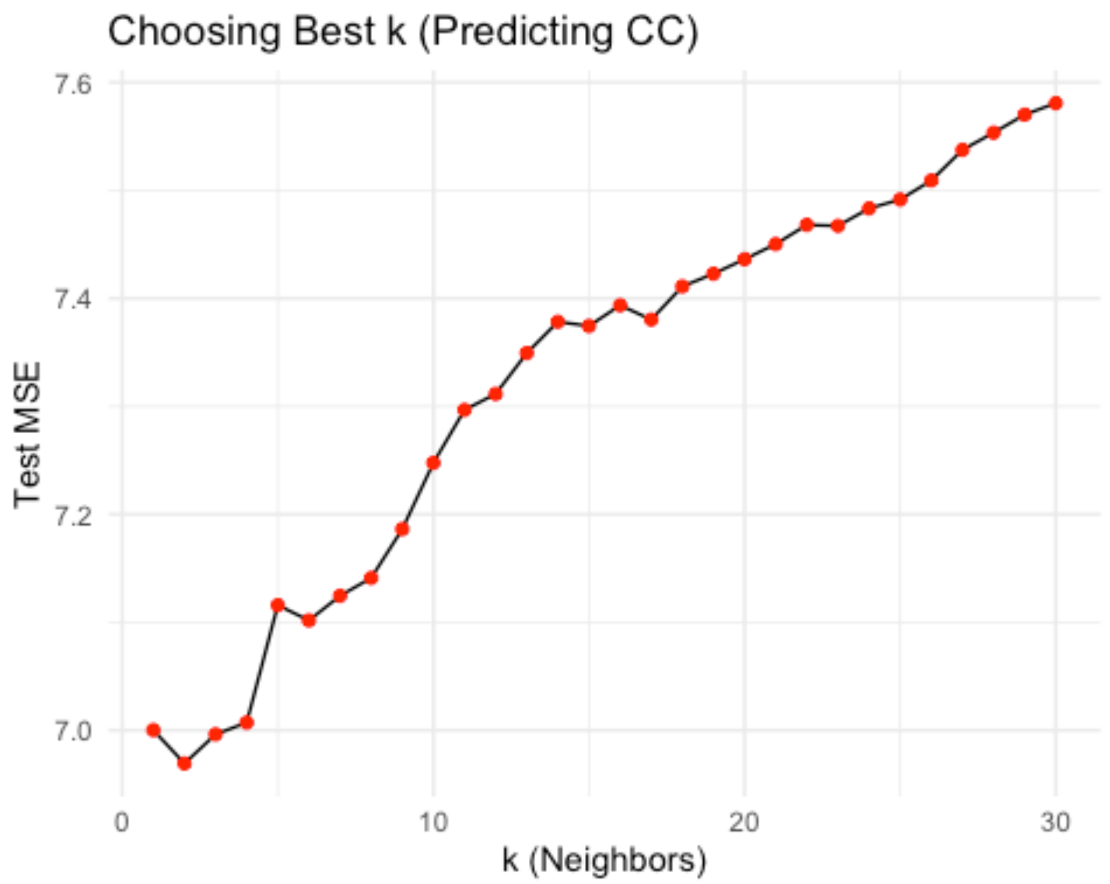
```r
# --- Compute threshold for outliers (±2σ here) ---
sigma    <- sd(residual_plot_data$Residuals)
threshold <- 2 * sigma

# --- Add a logical “is_outlier” column ---
residual_plot_data$is_outlier <- abs(residual_plot_data$Residuals) >
threshold

# --- Plot and colour by that flag ---
library(ggplot2)
ggplot(residual_plot_data, aes(x = Predicted, y = Residuals, color =
is_outlier)) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "grey50") +
  geom_point(alpha = 0.6, size = 2) +
  scale_color_manual(
    values = c(`FALSE` = "steelblue", `TRUE` = "firebrick"),
    labels = c("Within ±2σ", "Outside ±2σ"),
    name   = "Residual flag"
  ) +
  theme_minimal() +
  labs(
    title = "Residuals vs. Predicted IC",
    x     = "Predicted IC",
    y     = "Residual (Actual - Predicted)"
  )
```



```
# --- Model Evaluation for IC ---

# 1. Residuals (IC) found above

# 2. Residual Spread (standard deviation)
residual_sd_ic <- sd(residuals)
print(paste("Residual Standard Deviation (IC):", round(residual_sd_ic, 2)))

## [1] "Residual Standard Deviation (IC): 1.68"

# 3. Sum of Squared Residuals (SSR) and Total Sum of Squares (TSS)
SSR_ic <- sum(residuals^2)
TSS_ic <- sum((y_test - mean(y_test))^2)

# 4. Calculate pseudo R²
R2_ic <- 1 - (SSR_ic / TSS_ic)
print(paste("Pseudo R-squared (IC):", round(R2_ic, 4)))

## [1] "Pseudo R-squared (IC): 0.0716"
```

```r
## Change y from IC to CC
y_data_cc <- df$CC

# scale y=CC now
y_mean_cc <- mean(y_data_cc)
y_sd_cc <- sd(y_data_cc)
y_scaled_CC <- (y_data_cc - y_mean) / y_sd

n_obs <- nrow(X_data)
split_point <- floor(0.7 * n_obs)

# Create training and test sets from scaled data
y_train_CC_scaled <- y_scaled_CC[1:split_point]
y_test_CC_scaled <- y_scaled_CC[(split_point+1):n_obs]

# Finding best k
library(FNN)
library(ggplot2)

# Search best k
k_values <- 1:30
mse_values <- numeric(length(k_values))

for (i in seq_along(k_values)) {
  knn_model <- knn.reg(train = X_train, test = X_test, y = y_train_CC_scaled,
k = k_values[i])
  y_pred <- knn_model$pred
  mse_values[i] <- mean((y_pred - y_test_CC_scaled)^2)
}

# Plot k vs MSE
qplot(k_values, mse_values, geom = "line") +
  geom_point(color = "red") +
  labs(x = "k (Neighbors)", y = "Test MSE", title = "Choosing Best k
(Predicting CC)") +
  theme_minimal()
```

```
## Warning: `qplot()` was deprecated in ggplot2 3.4.0.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

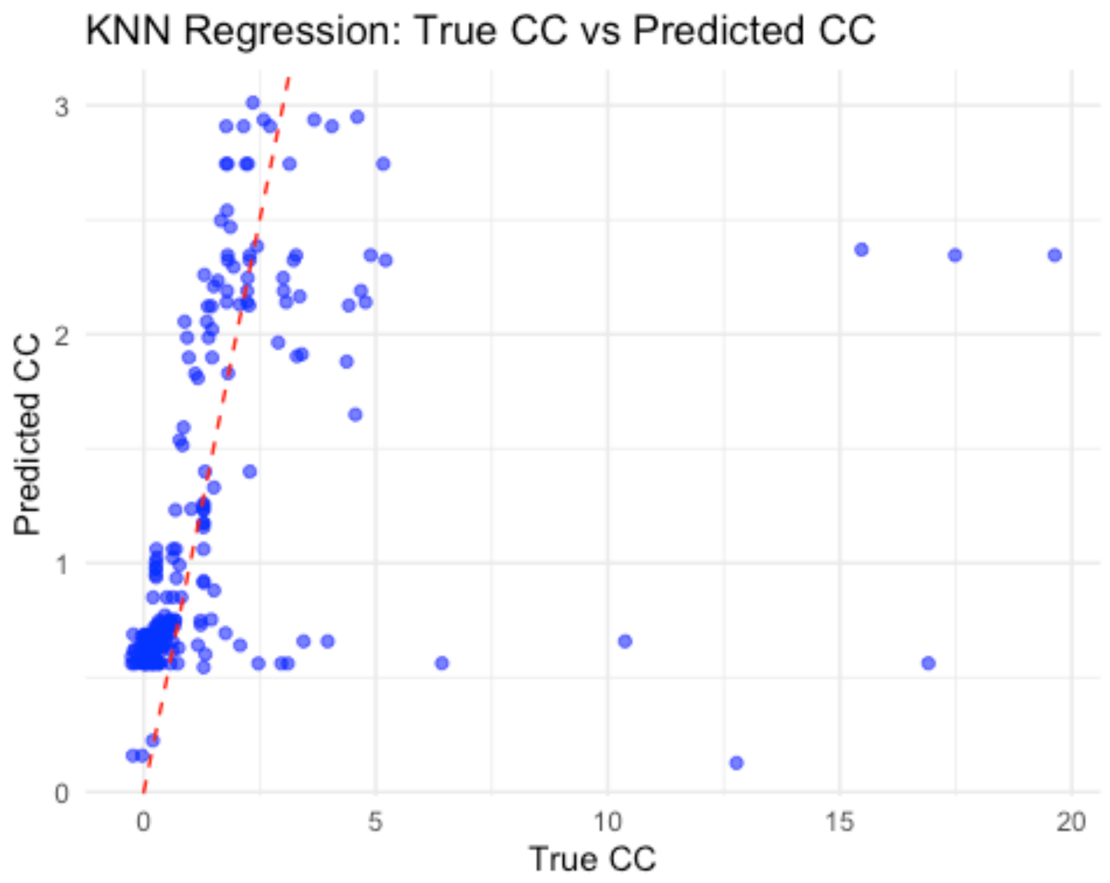


```
# Find best k
best_k <- k_values[which.min(mse_values)]
print(paste("Best k for CC:", best_k))

## [1] "Best k for CC: 2"

# Calculate Residuals (CC)
# Use the best k to build final knn model
final_knn_model_cc <- knn.reg(train = X_train, test = X_test, y =
y_train_CC_scaled, k = best_k)

# get y_pred
y_pred_final_cc <- final_knn_model_cc$pred

# calculate residuals
residuals_cc <- y_test_CC_scaled - y_pred_final_cc

# put into df for plotting
residual_plot_data_cc <- data.frame(
  Predicted = y_pred_final_cc,
  Residuals = residuals_cc,
  True_CC = y_test_CC_scaled
)
```

```r
final_mse <- mean((y_pred_final_cc - y_test_CC_scaled)^2)
cat("Final best MSE", round(final_mse, 2), "\n")
```

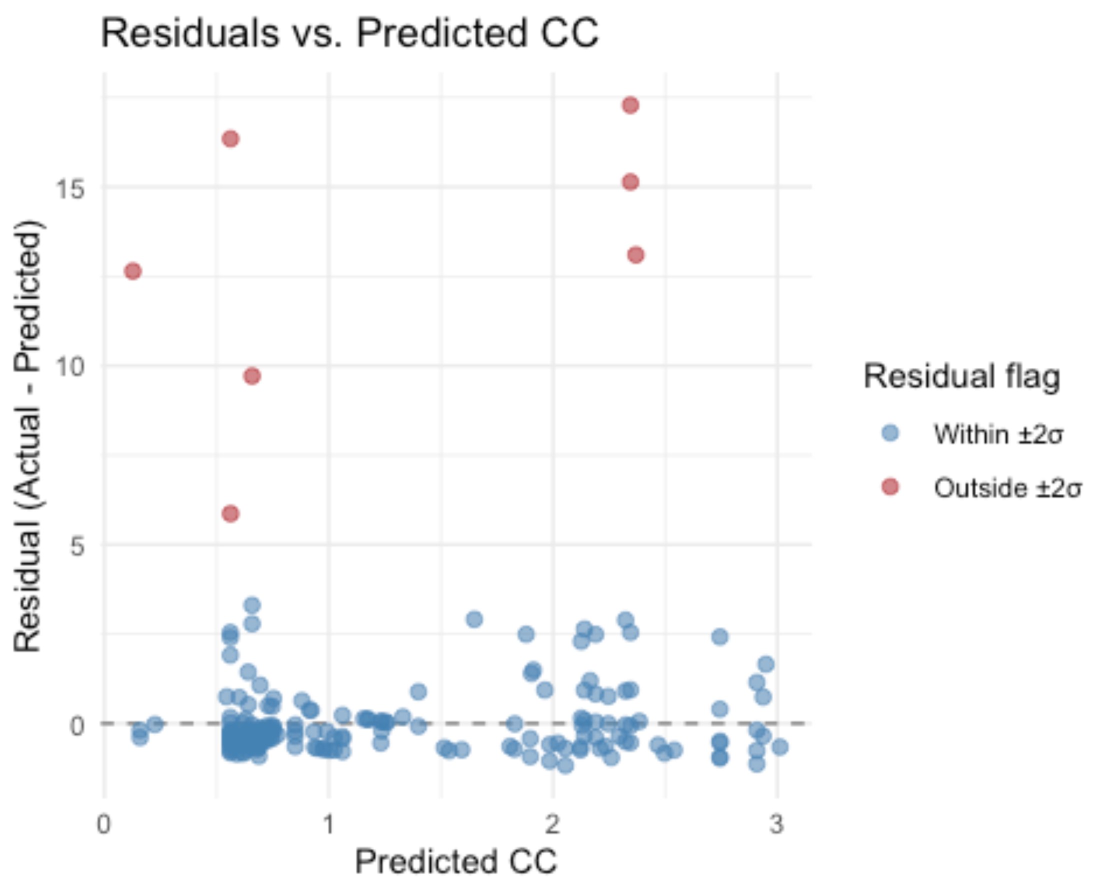
```
## Final best MSE 6.97
```

```r
library(ggplot2)

ggplot(residual_plot_data_cc, aes(x = True_CC, y = Predicted)) +
  geom_point(color = "blue", alpha = 0.6) +
  geom_abline(intercept = 0, slope = 1,
              color = "red", linetype = "dashed") +
  labs(
    title = "KNN Regression: True CC vs Predicted CC",
    x = "True CC",
    y = "Predicted CC"
  ) +
  theme_minimal()
```



```r
# --- Compute threshold for outliers (±2σ here) ---
sigma     <- sd(residual_plot_data_cc$Residuals)
threshold <- 2 * sigma

# --- Add a logical "is_outlier" column ---
```

```r
residual_plot_data_cc$is_outlier <- abs(residual_plot_data_cc$Residuals) >
threshold

# --- Plot and colour by that flag ---
library(ggplot2)
ggplot(residual_plot_data_cc, aes(x = Predicted, y = Residuals, color =
is_outlier)) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "grey50") +
  geom_point(alpha = 0.6, size = 2) +
  scale_color_manual(
    values = c(`FALSE` = "steelblue", `TRUE` = "firebrick"),
    labels = c("Within ±2σ", "Outside ±2σ"),
    name   = "Residual flag"
  ) +
  theme_minimal() +
  labs(
    title = "Residuals vs. Predicted CC",
    x     = "Predicted CC",
    y     = "Residual (Actual - Predicted)"
  )
```



```r
# --- Model Evaluation for CC ---
# 1. Residuals (CC)
residuals_cc <- y_test_CC_scaled - y_pred_final_cc

# 2. Residual Spread (standard deviation)
residual_sd_cc <- sd(residuals_cc)
print(paste("Residual Standard Deviation (CC):", round(residual_sd_cc, 2)))

## [1] "Residual Standard Deviation (CC): 2.62"

# 3. Sum of Squared Residuals (SSR) and Total Sum of Squares (TSS)
SSR_cc <- sum(residuals_cc^2)
TSS_cc <- sum((y_test_CC_scaled - mean(y_test_CC_scaled))^2)

# 4. Calculate pseudo R²
R2_cc <- 1 - (SSR_cc / TSS_cc)
print(paste("Pseudo R-squared (CC):", round(R2_cc, 4)))

## [1] "Pseudo R-squared (CC): 0.1062"

## Random Forest (CC)
library(randomForest)

## randomForest 4.7-1.2

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin

## The following object is masked from 'package:dplyr':
##
##     combine

library(dplyr)
library(ggplot2)

# Step 1: Train Random Forest for CC
rf_model_cc <- randomForest(x = X_train, y = y_train_CC_scaled, ntree = 500,
mtry = floor(sqrt(ncol(X_train))), importance = TRUE)

# Step 2: Predict
y_pred_rf_cc <- predict(rf_model_cc, newdata = X_test)

# Step 3: Evaluate
mse_rf_cc <- mean((y_test_CC_scaled - y_pred_rf_cc)^2)
```

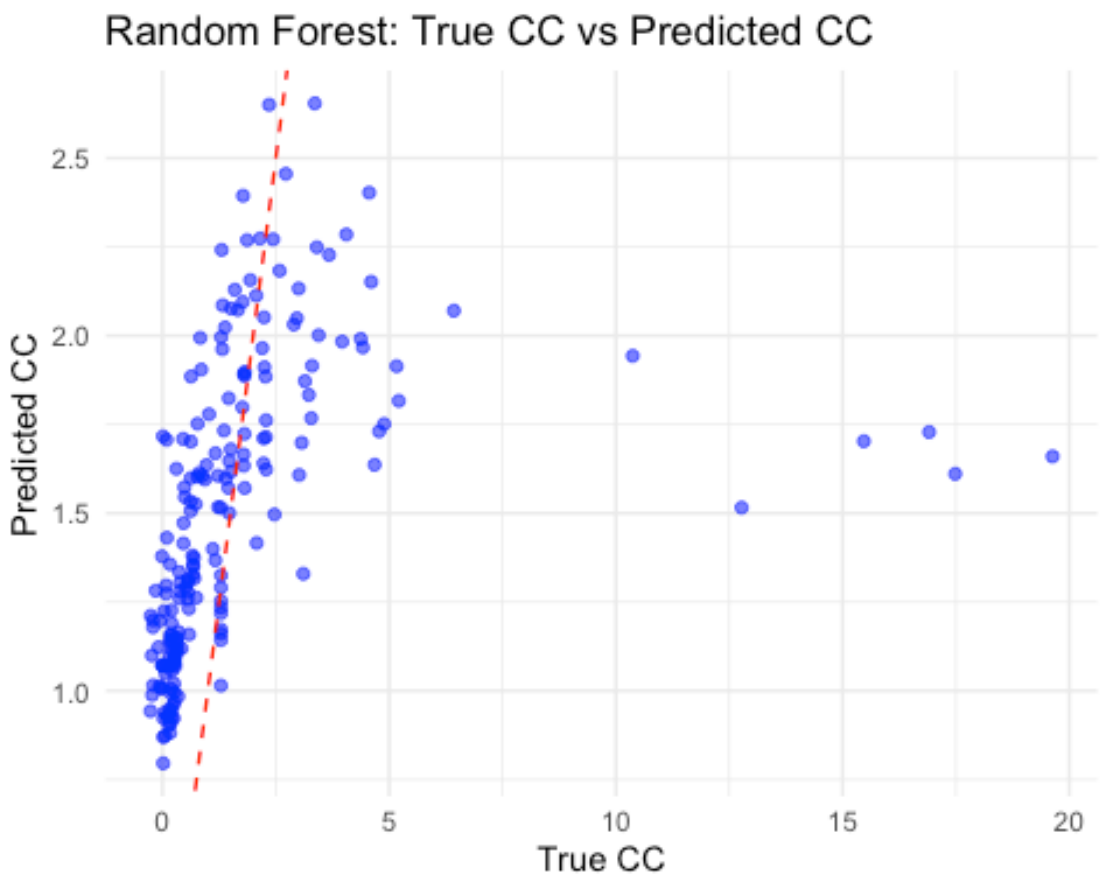
```r
residuals_rf_cc <- y_test_CC_scaled - y_pred_rf_cc
print(paste("Test MSE for RF CC:", round(mse_rf_cc, 2)))
```

```
## [1] "Test MSE for RF CC: 7.06"
```

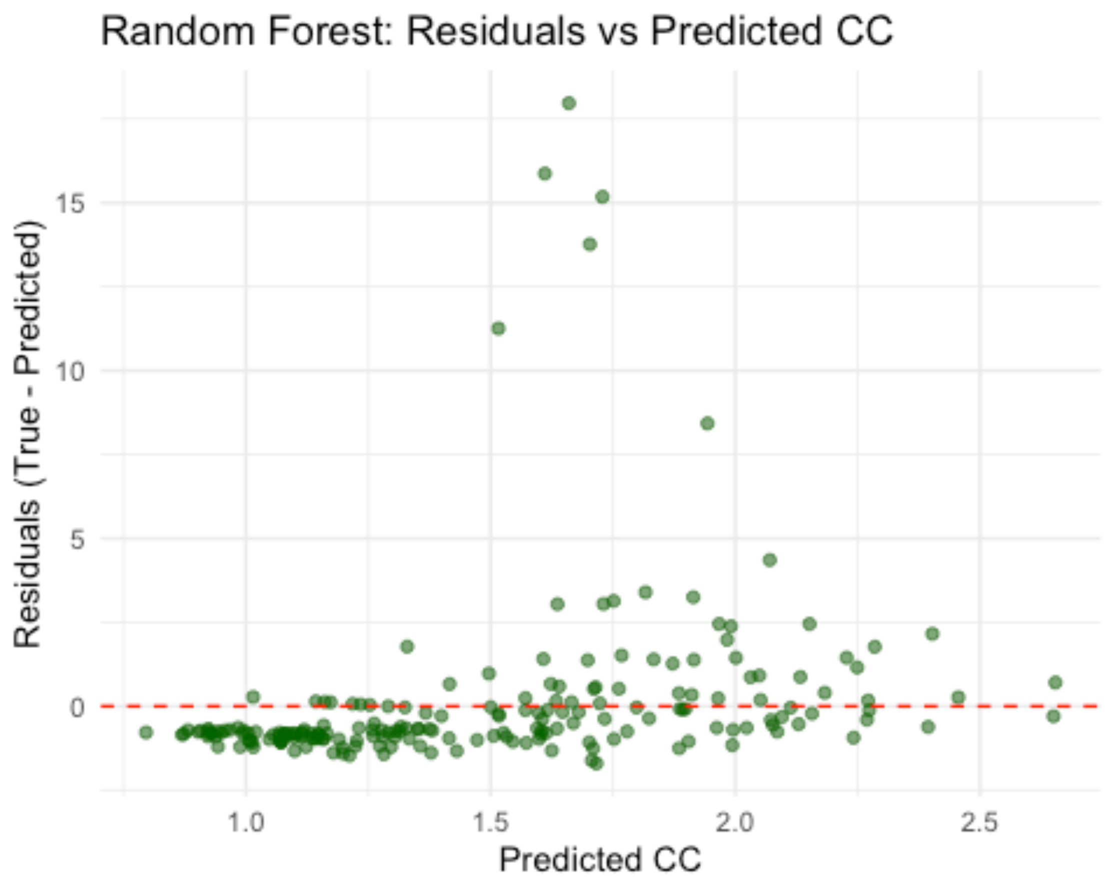
```r
pseudo_r2_rf_cc <- 1 - var(residuals_rf_cc) / var(y_test_CC_scaled)
print(paste("Pseudo R² for RF CC:", round(pseudo_r2_rf_cc, 4)))
```

```
## [1] "Pseudo R² for RF CC: 0.0971"
```

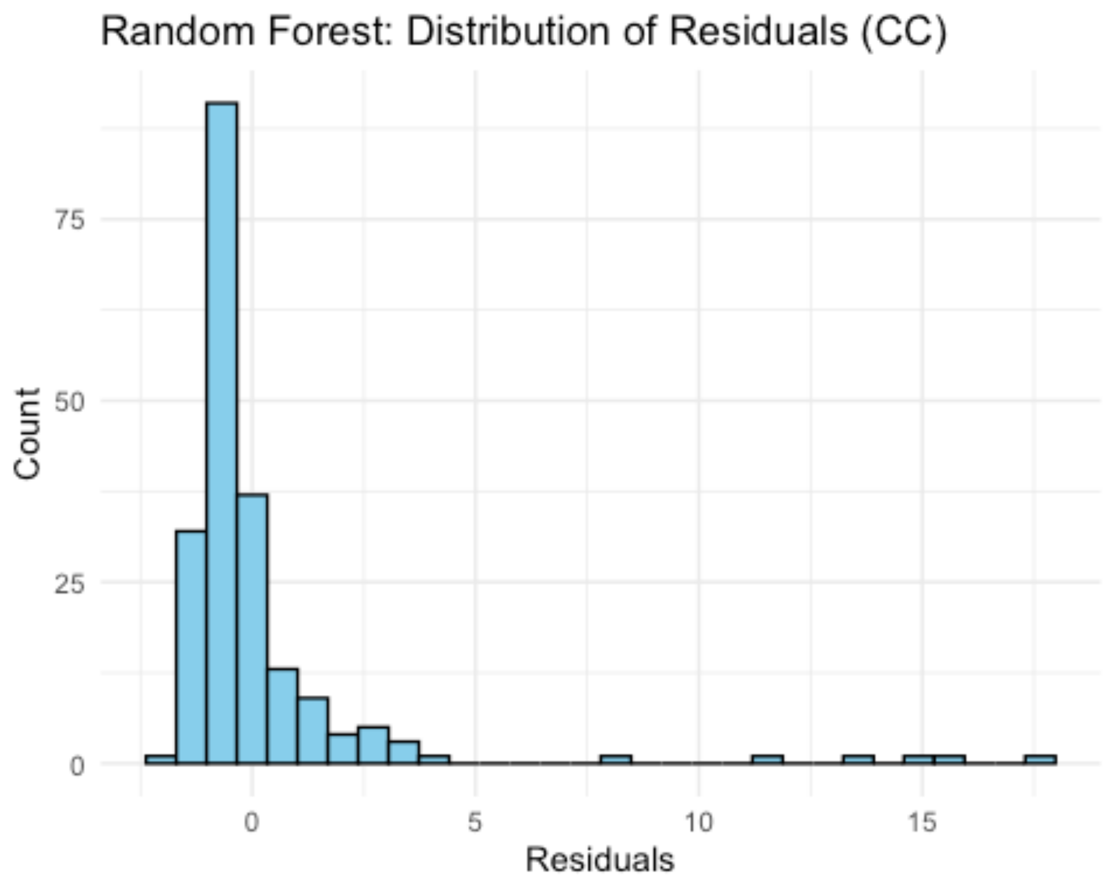
```r
# Step 4: Graphs
library(ggplot2)

# --- Prepare data for plotting ---
rf_residuals_cc <- y_test_CC_scaled - y_pred_rf_cc

plot_data_cc <- data.frame(
  True_CC = y_test_CC_scaled,
  Predicted_CC = y_pred_rf_cc,
  Residuals_CC = rf_residuals_cc
)

# --- 1. True vs Predicted CC Plot ---
ggplot(plot_data_cc, aes(x = True_CC, y = Predicted_CC)) +
  geom_point(color = "blue", alpha = 0.6) +
  geom_abline(intercept = 0, slope = 1, color = "red", linetype = "dashed") +
  labs(
    title = "Random Forest: True CC vs Predicted CC",
    x = "True CC",
    y = "Predicted CC"
  ) +
  theme_minimal()
```



```r
# --- 2. Residuals vs Predicted CC Plot ---
ggplot(plot_data_cc, aes(x = Predicted_CC, y = Residuals_CC)) +
  geom_point(color = "darkgreen", alpha = 0.6) +
  geom_hline(yintercept = 0, color = "red", linetype = "dashed") +
  labs(
    title = "Random Forest: Residuals vs Predicted CC",
    x = "Predicted CC",
    y = "Residuals (True - Predicted)"
  ) +
  theme_minimal()
```

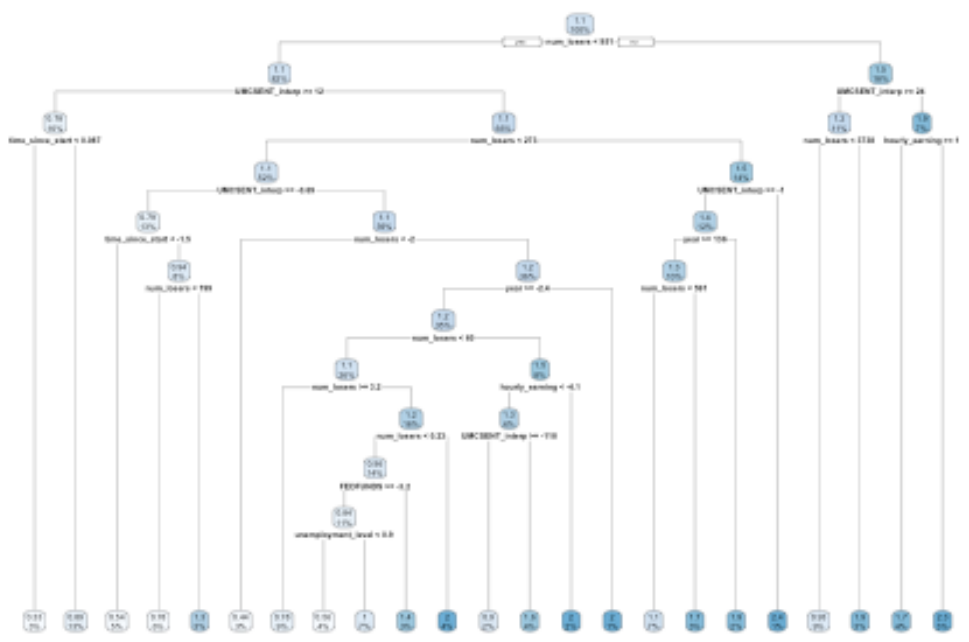


```r
# --- 3. Residuals Histogram ---
ggplot(plot_data_cc, aes(x = Residuals_CC)) +
  geom_histogram(bins = 30, fill = "skyblue", color = "black") +
  labs(
    title = "Random Forest: Distribution of Residuals (CC)",
    x = "Residuals",
    y = "Count"
  ) +
  theme_minimal()
```



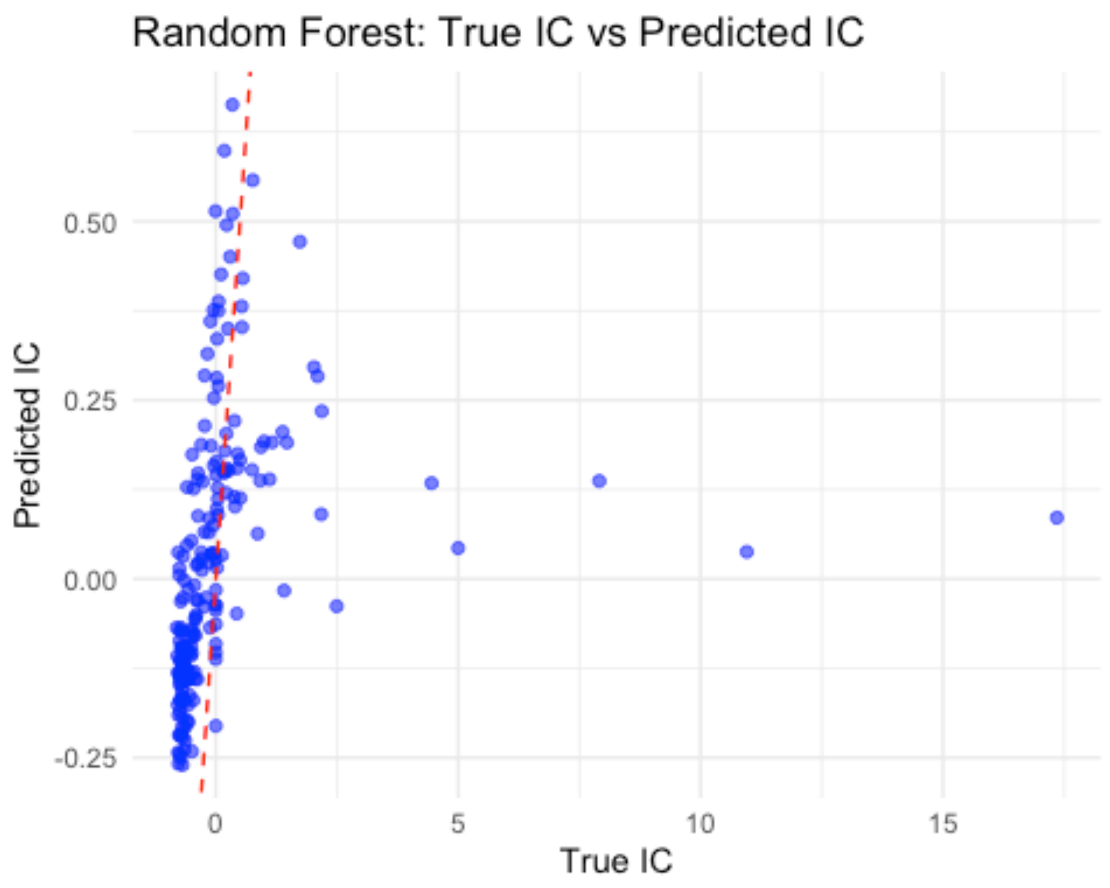
```
## a small representative tree diagram
library(rpart)
library(rpart.plot)

# Train a single decision tree (just for visualization)
single_tree_cc <- rpart(y_train_CC_scaled ~ ., data = as.data.frame(X_train),
method = "anova")

# Plot it
rpart.plot(single_tree_cc, main = "Single Decision Tree for CC (Not the full
Random Forest)")
```

Single Decision Tree for CC (Not the full Random Forest)

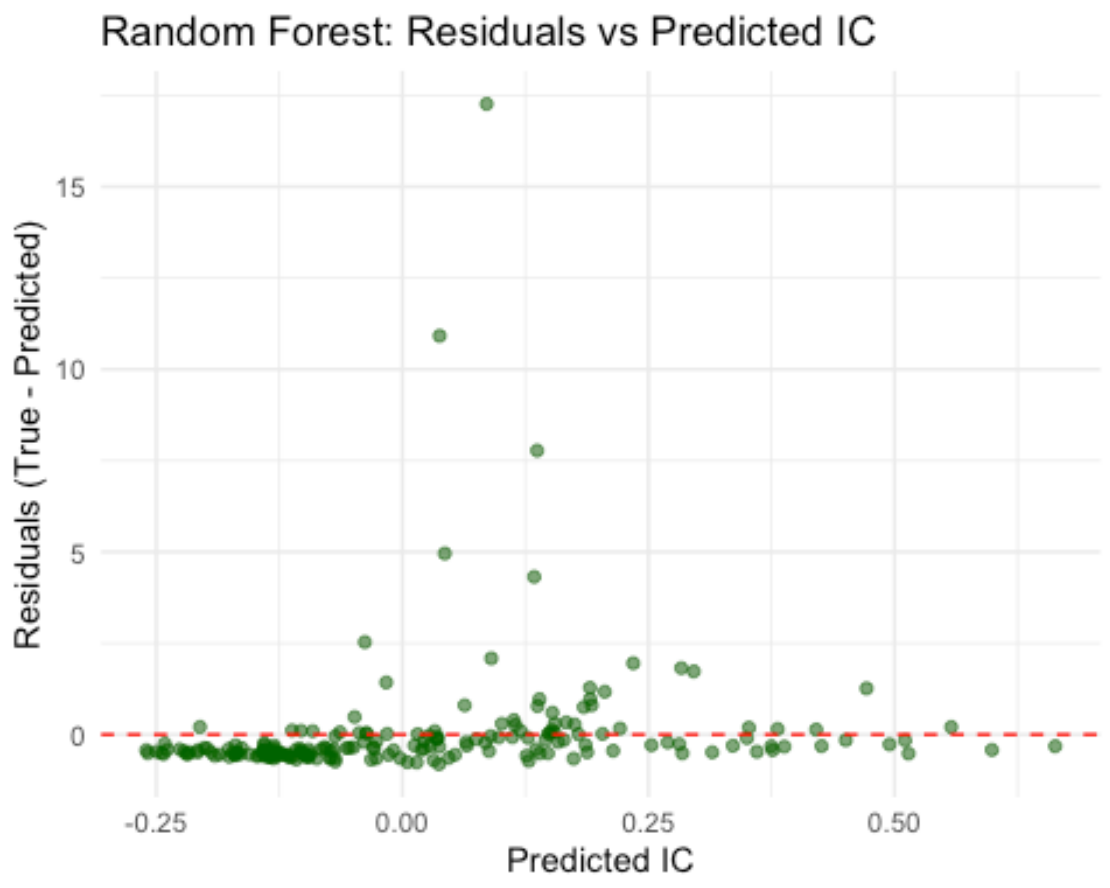


```
## Random Forest (IC)
# Train RF model for IC
rf_model_ic <- randomForest(x = X_train, y = y_train, ntree = 500, mtry =
floor(sqrt(ncol(X_train))), importance = TRUE)

# Predict IC
y_pred_rf_ic <- predict(rf_model_ic, newdata = X_test)

# Calculate residuals
residuals_rf_ic <- y_test - y_pred_rf_ic

# Calculate Test MSE
mse_rf_ic <- mean(residuals_rf_ic^2)
print(paste("Test MSE for RF IC:", round(mse_rf_ic, 2)))

## [1] "Test MSE for RF IC: 2.91"

# Calculate pseudo R²
pseudo_r2_rf_ic <- 1 - var(residuals_rf_ic) / var(y_test)
print(paste("Pseudo R² for RF IC:", round(pseudo_r2_rf_ic, 4)))

## [1] "Pseudo R² for RF IC: 0.0454"
```

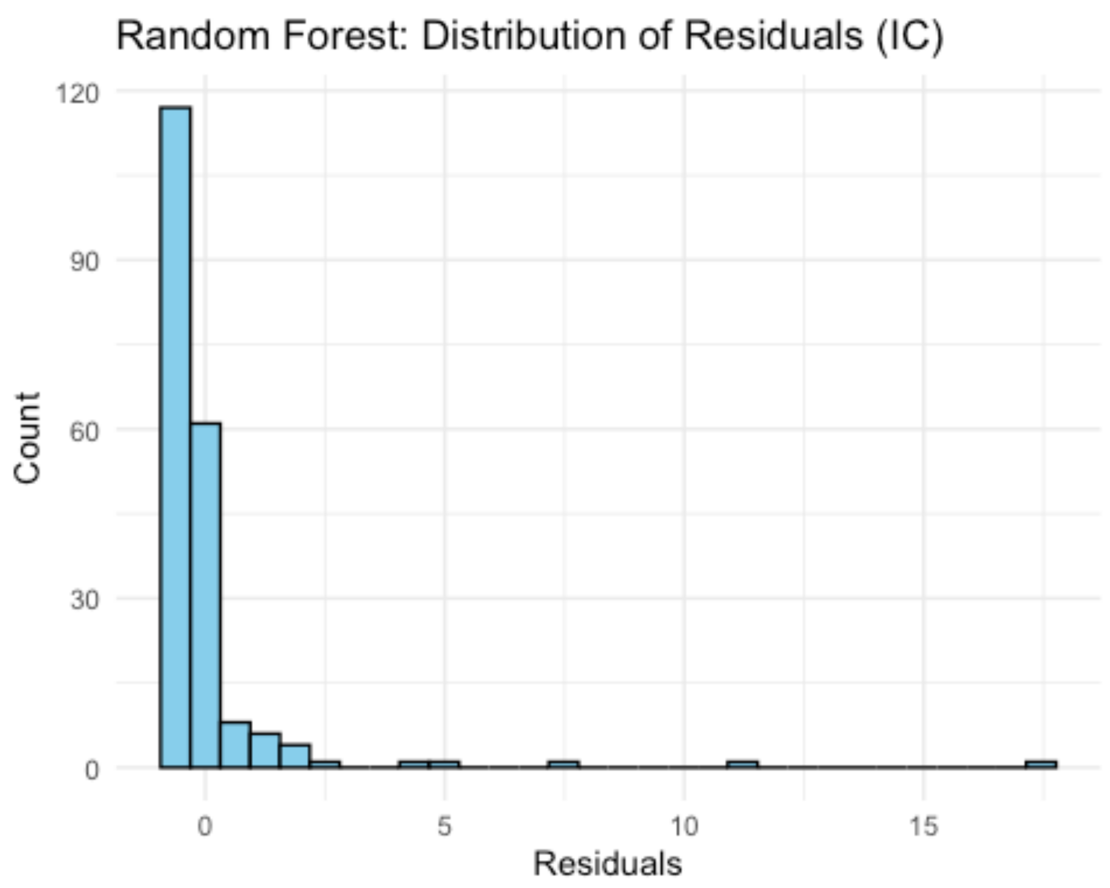
```r
library(ggplot2)

# Prepare data for plotting
plot_data_ic <- data.frame(
  True_IC = y_test,
  Predicted_IC = y_pred_rf_ic,
  Residuals_IC = residuals_rf_ic
)

# 1. True vs Predicted IC Plot
ggplot(plot_data_ic, aes(x = True_IC, y = Predicted_IC)) +
  geom_point(color = "blue", alpha = 0.6) +
  geom_abline(intercept = 0, slope = 1, color = "red", linetype = "dashed") +
  labs(
    title = "Random Forest: True IC vs Predicted IC",
    x = "True IC",
    y = "Predicted IC"
  ) +
  theme_minimal()
```



```r
# 2. Residuals vs Predicted IC Plot
ggplot(plot_data_ic, aes(x = Predicted_IC, y = Residuals_IC)) +
  geom_point(color = "darkgreen", alpha = 0.6) +
  geom_hline(yintercept = 0, color = "red", linetype = "dashed") +
  labs(
    title = "Random Forest: Residuals vs Predicted IC",
    x = "Predicted IC",
    y = "Residuals (True - Predicted)"
  ) +
  theme_minimal()
```



```r
# 3. Residuals Histogram
ggplot(plot_data_ic, aes(x = Residuals_IC)) +
  geom_histogram(bins = 30, fill = "skyblue", color = "black") +
  labs(
    title = "Random Forest: Distribution of Residuals (IC)",
    x = "Residuals",
    y = "Count"
  ) +
  theme_minimal()
```


Random Forest: Distribution of Residuals (IC)
Count
120
90
60
30
0
0
5
10
15
Residuals