Final Project

Baili Chen

2024-12-13

## R Markdown

In this project, I engage in an extensive analysis and forecasting of 518 annual tourism series, each reflecting various aspects of tourism activity. The data, transformed from wide to long format for ease of manipulation, spans lengths from 7 to 43 years and includes diverse metrics such as visitor expenditures and overnight stays. I implement a multi-faceted approach to understanding and predicting these series, starting with a thorough visualization to identify patterns and anomalies. Subsequent steps include the development of naive forecasts and the application of more sophisticated statistical techniques, such as linear regression and exponentially weighted models, aimed at enhancing predictive accuracy for up to four years ahead. The project has a robust evaluation of model performance using metrics like MAE, MAPE, and RMSE, and explores advanced forecasting methods, including performance evaluation techniques, to optimize the forecasts. This analysis not only aims to achieve high accuracy but also deepens the understanding of large-scale time series forecasting within the tourism sector.

tourism\_data<- read.csv("C:/Users/Baili/Downloads/tourism\_data.csv",stringsAsFactors = TRUE)  
  
library(tidyr)  
library(dplyr)

##   
## 载入程序包：'dplyr'

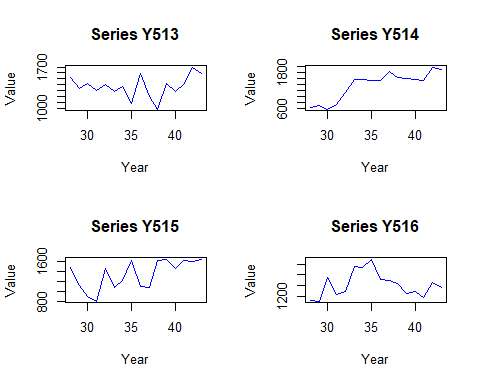
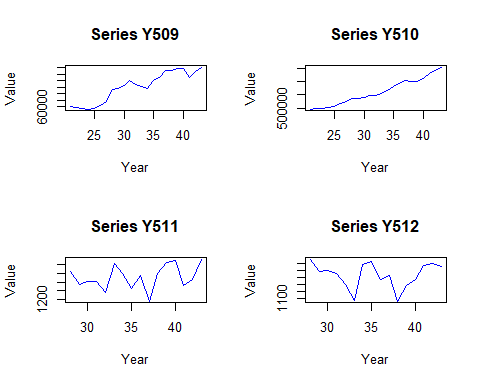
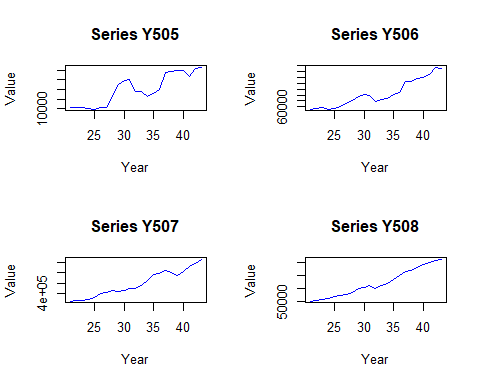
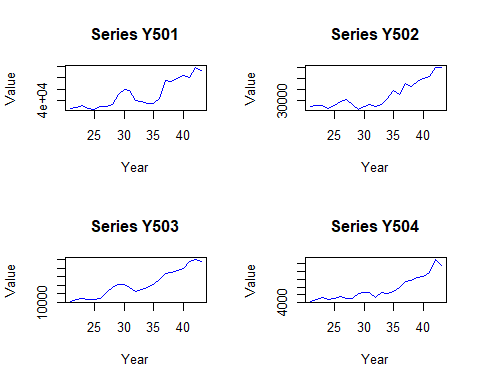
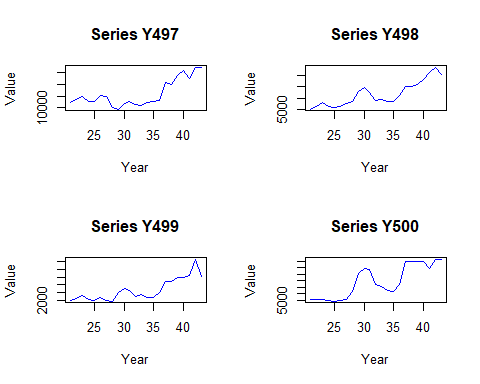
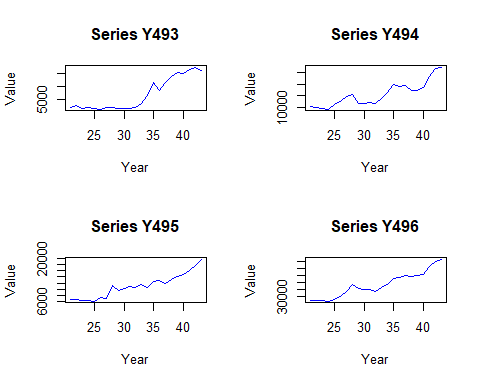
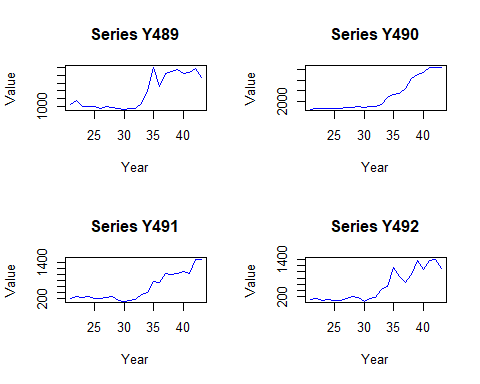
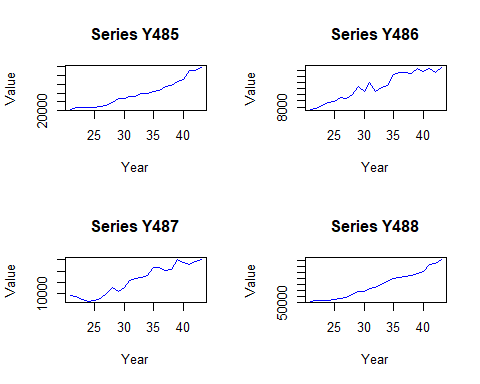
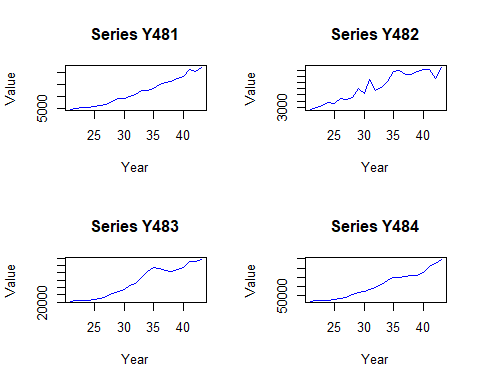
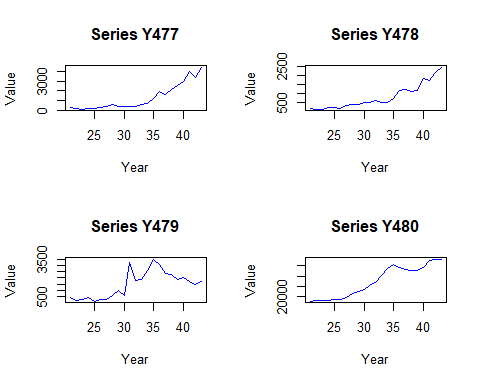
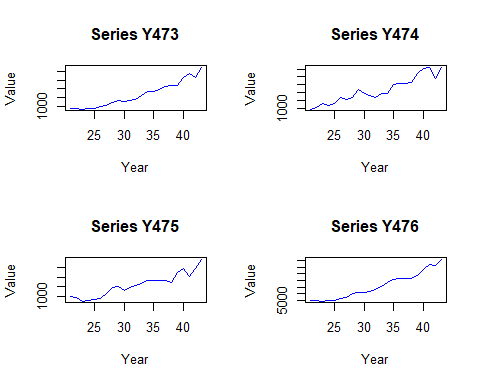
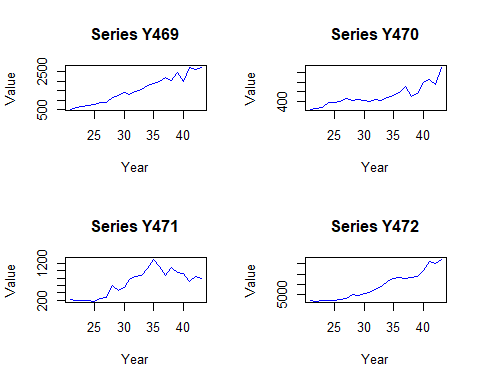
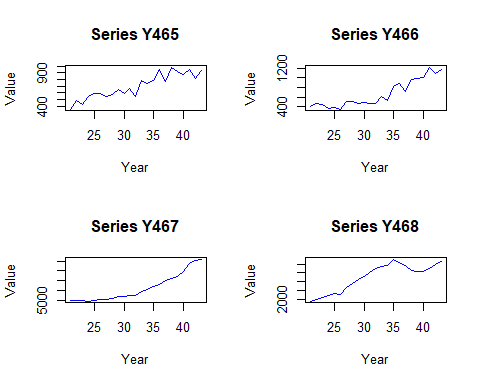
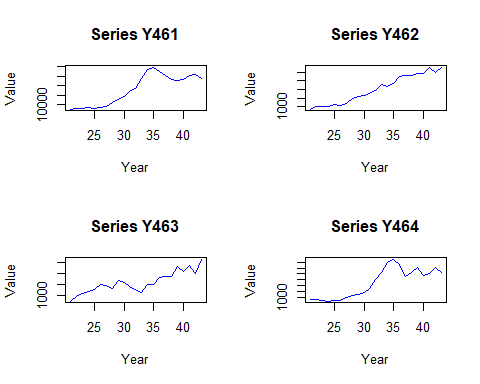
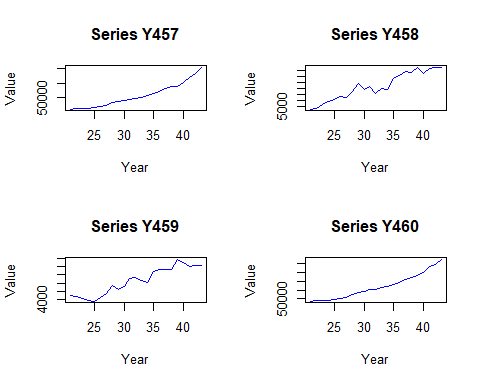
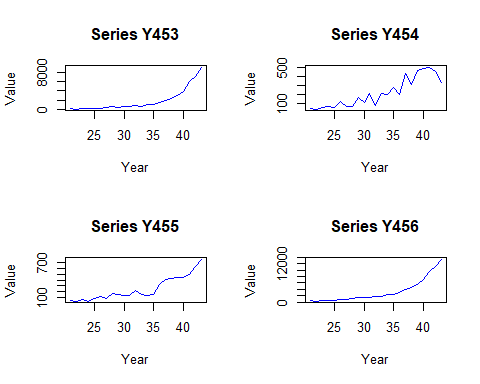
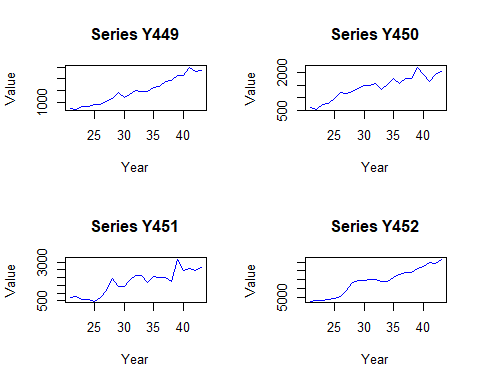
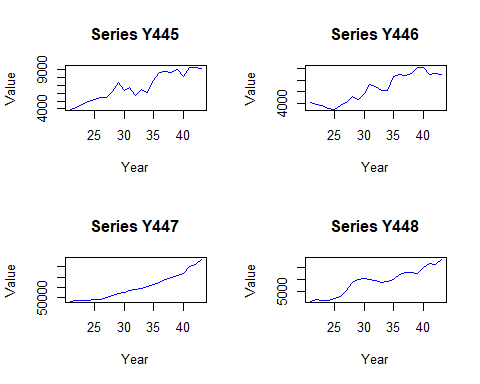
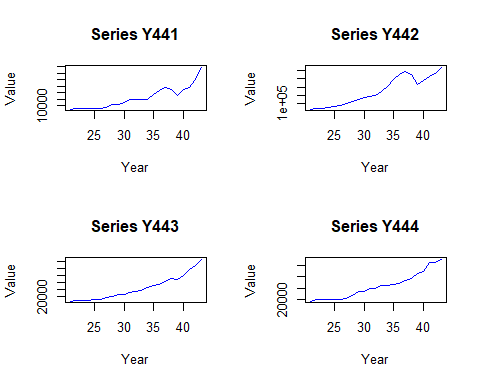
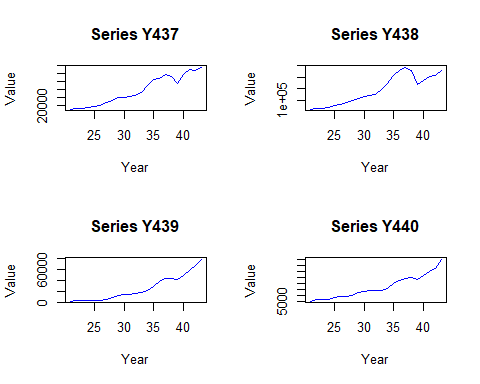
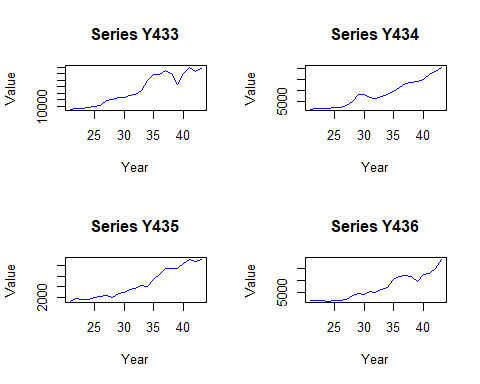
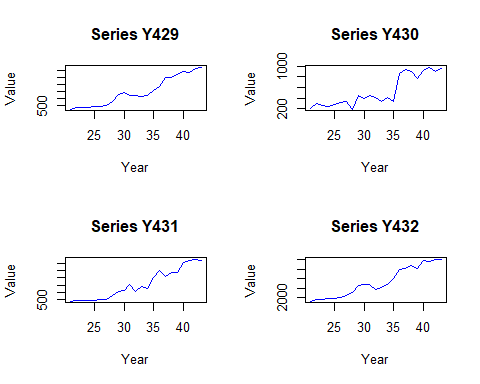
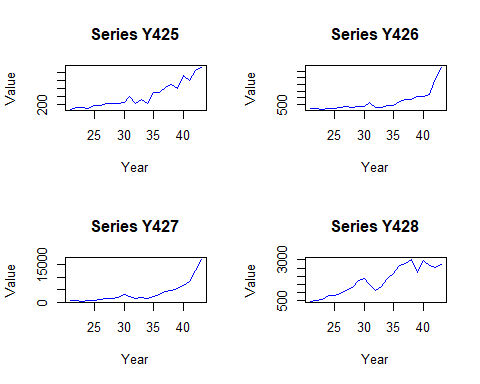
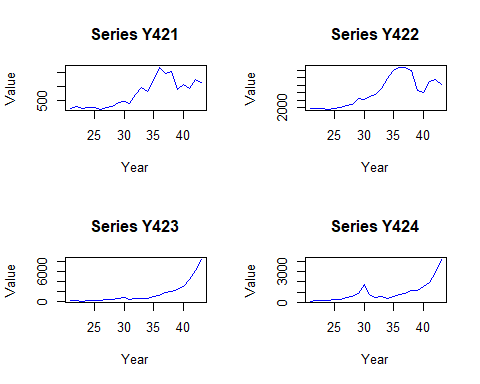
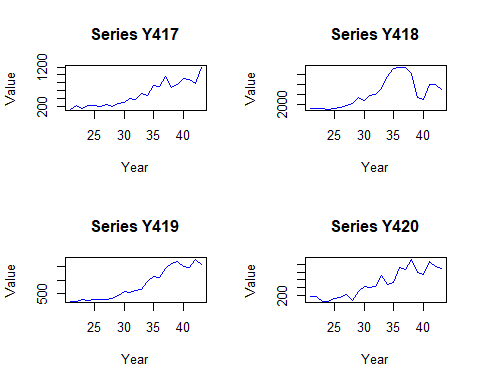
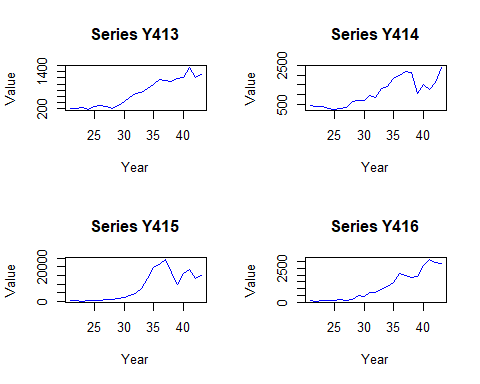
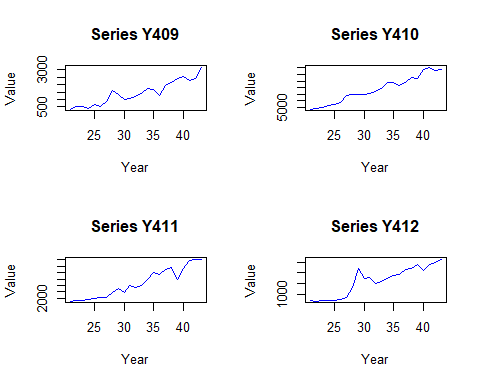
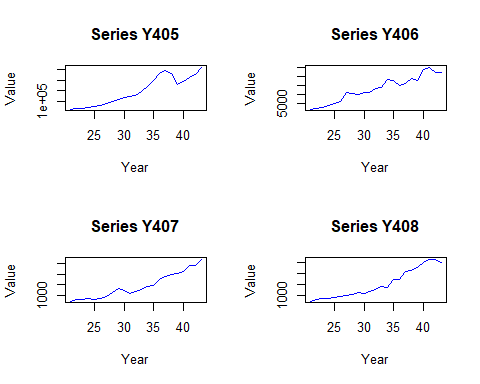
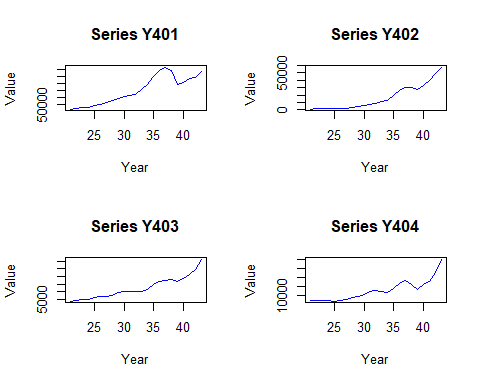
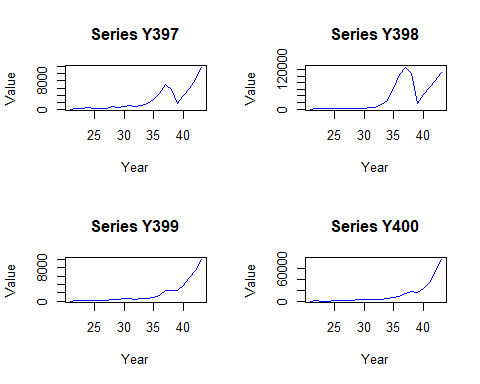
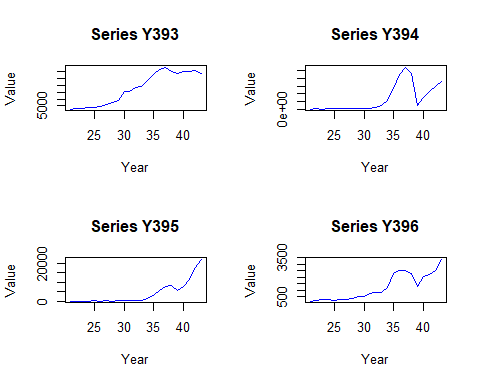
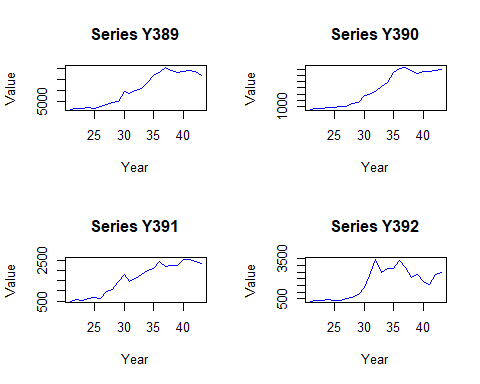
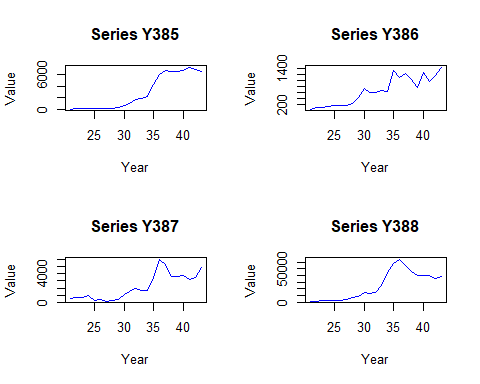
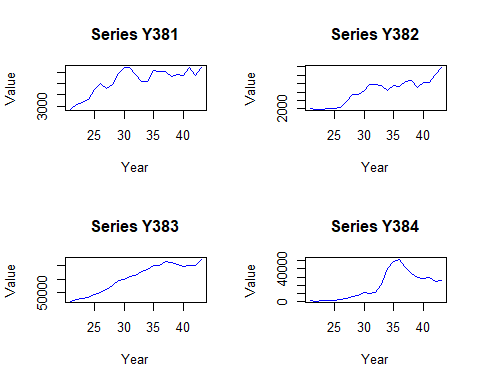
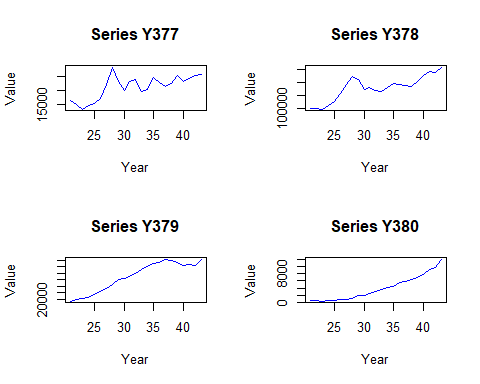
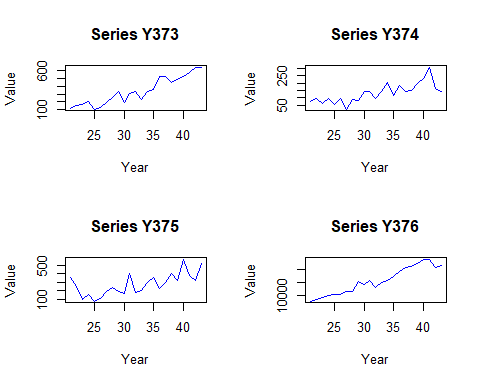
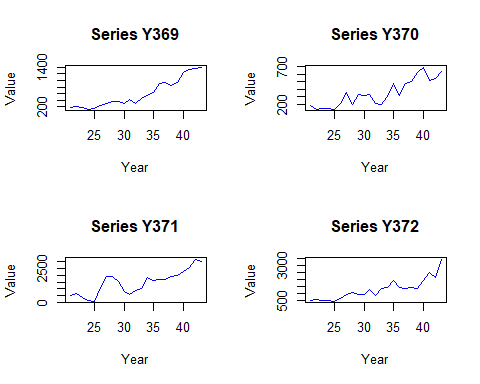
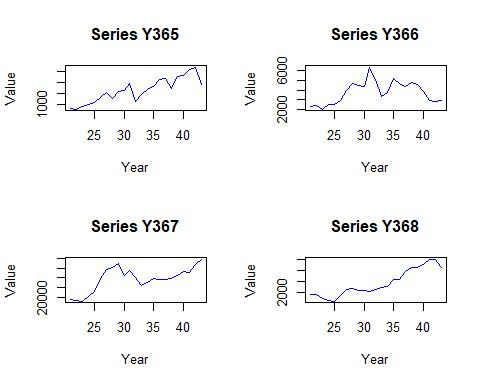
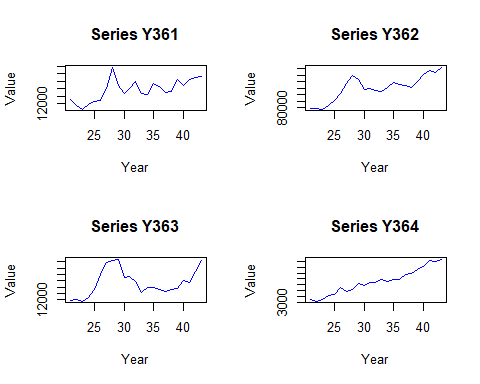
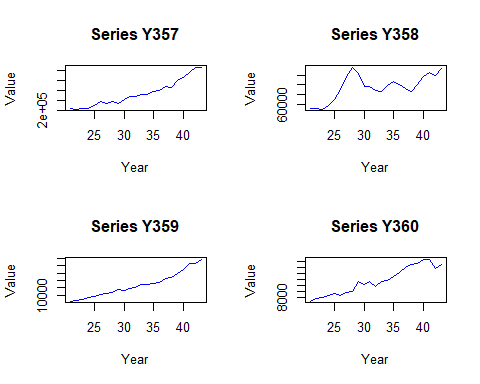
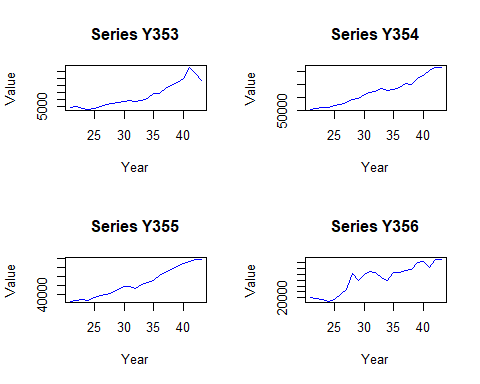
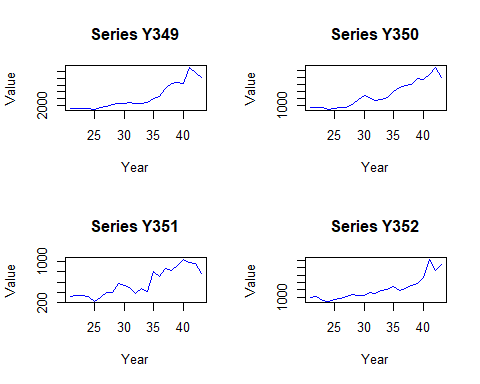
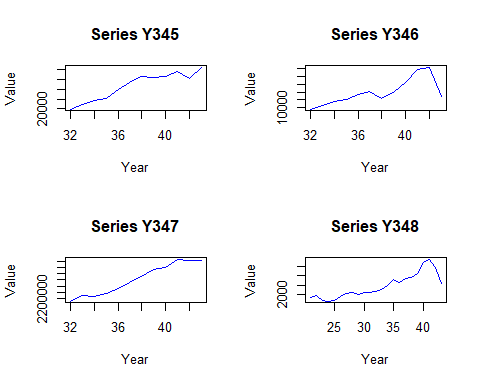
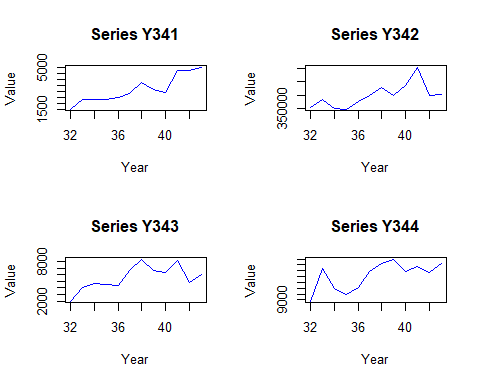
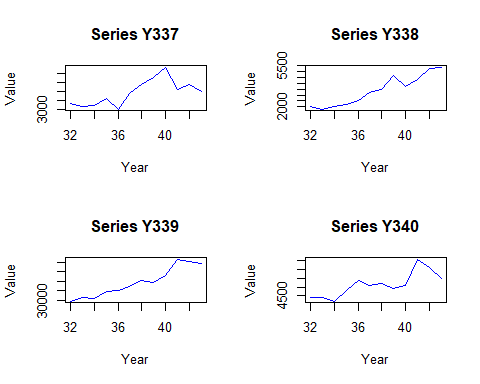
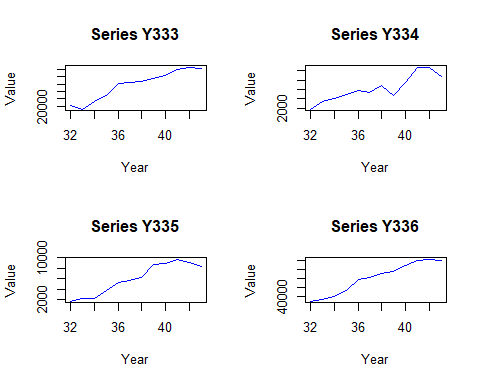
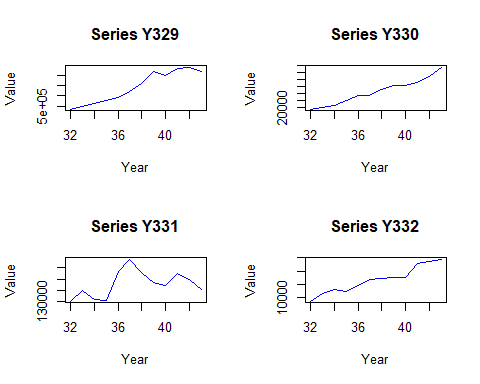
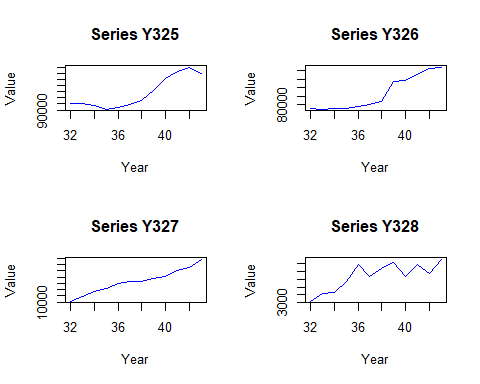
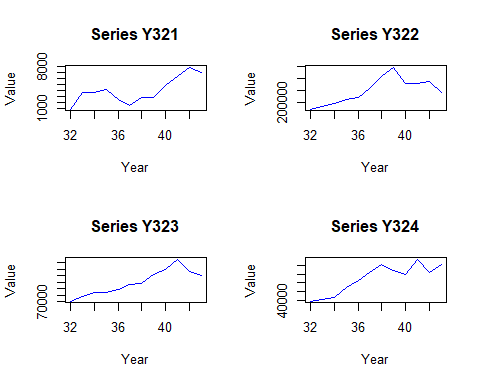
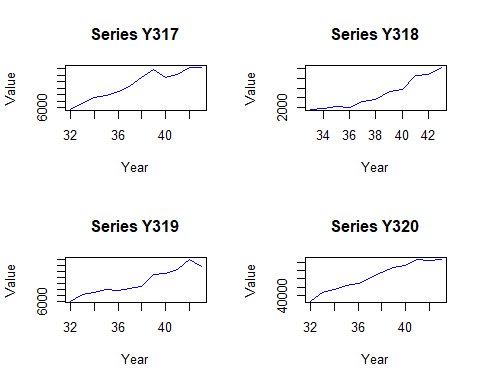
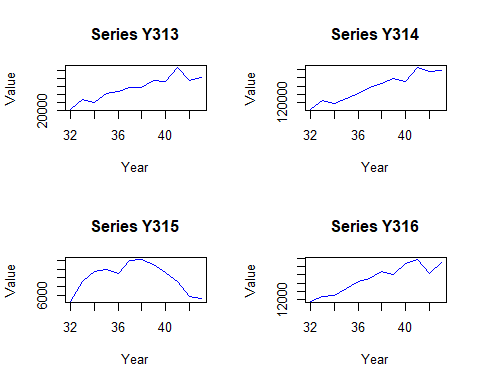
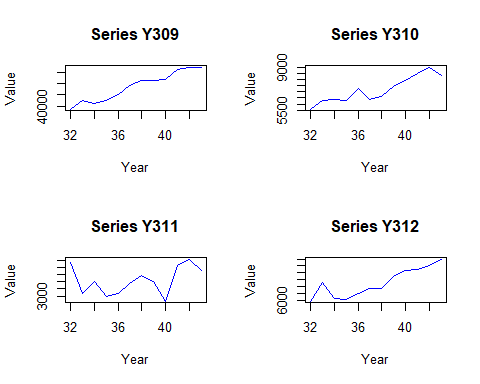
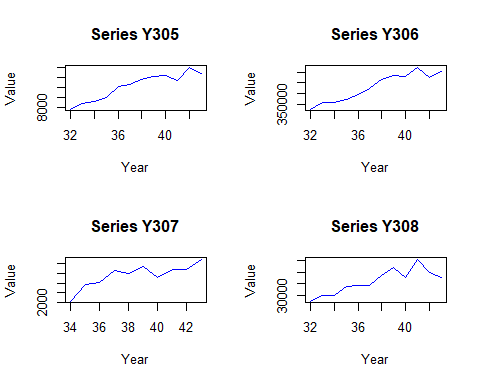
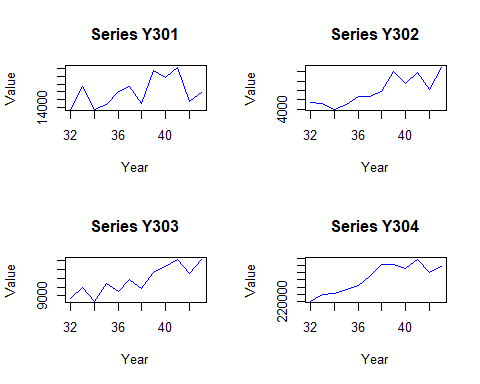
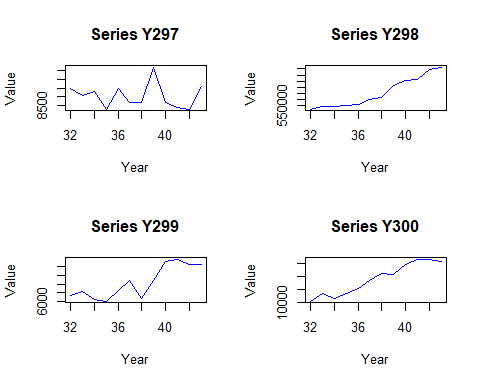
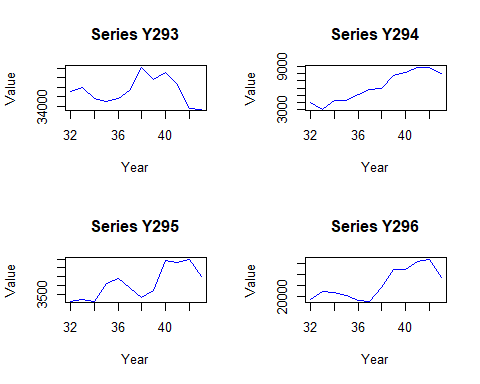
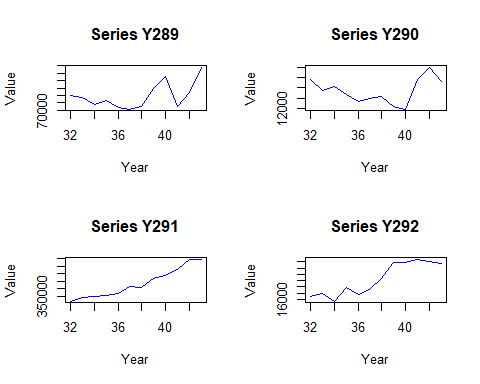
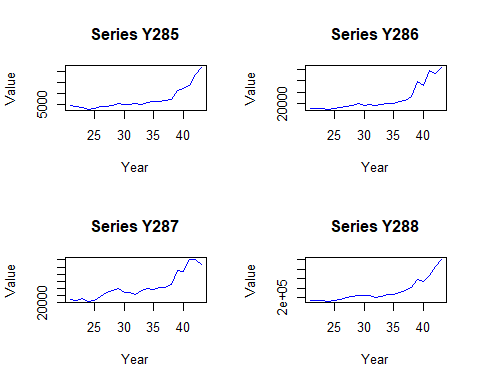
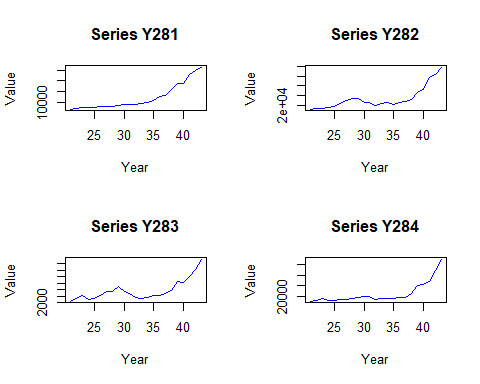
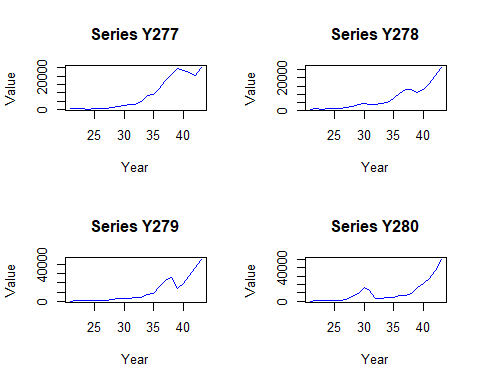
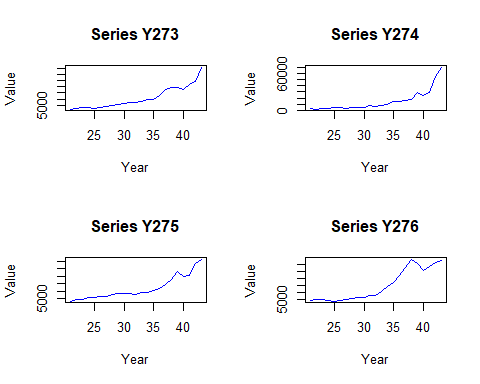
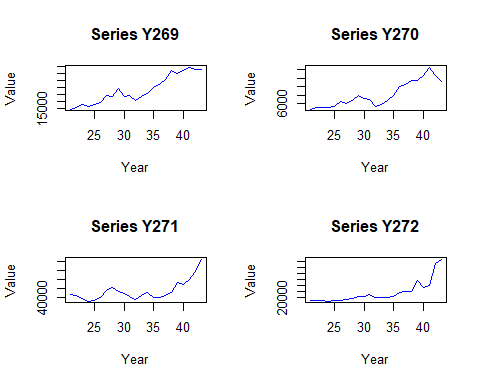
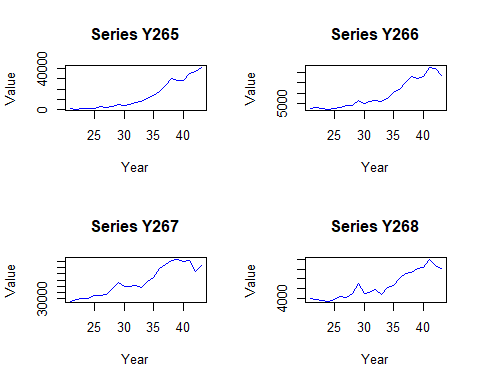
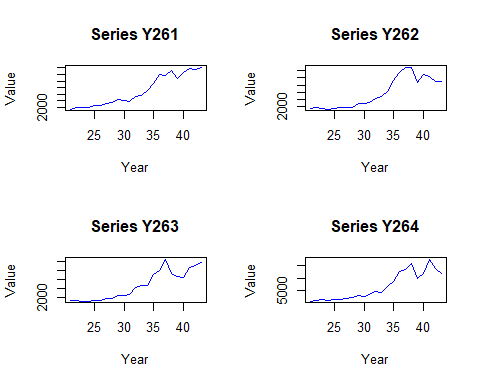
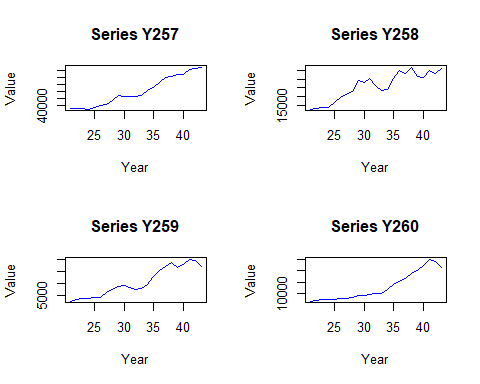
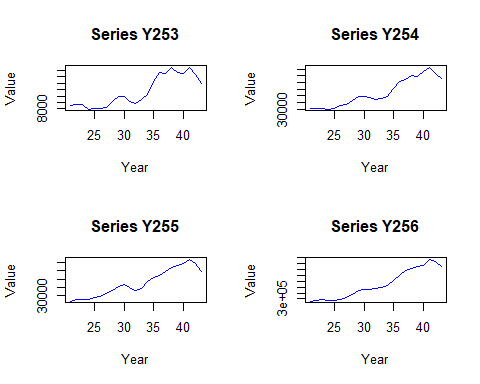
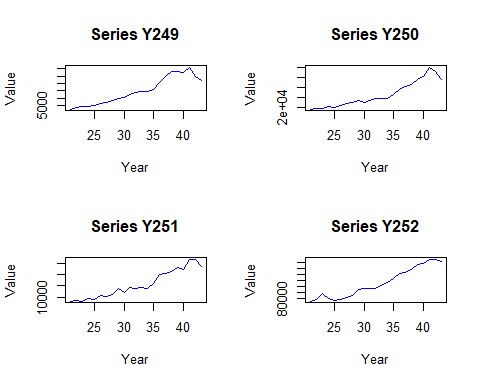
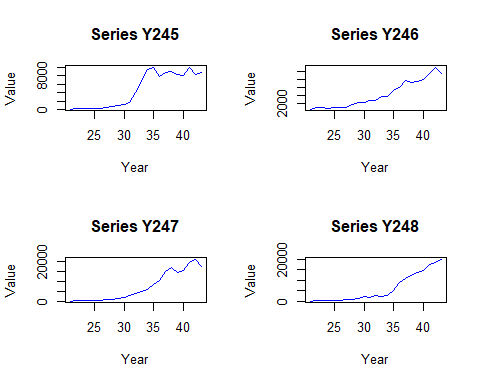
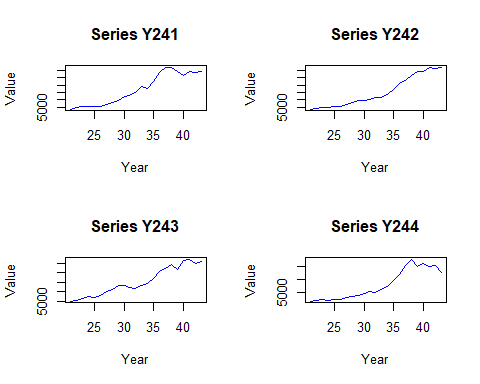
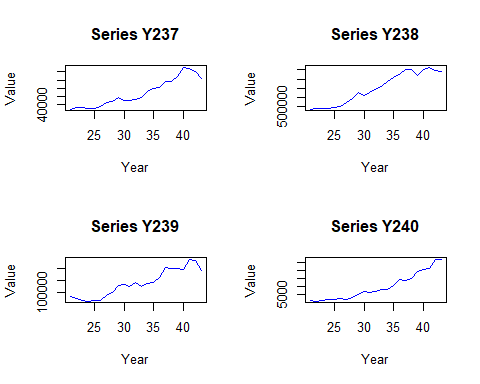
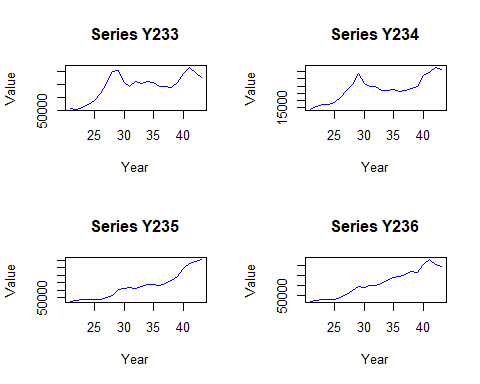
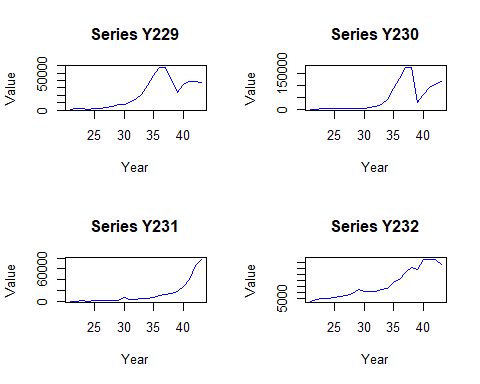
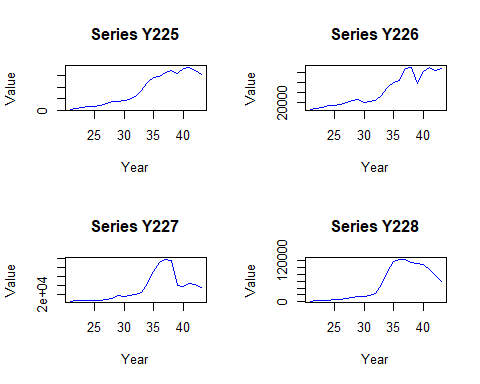
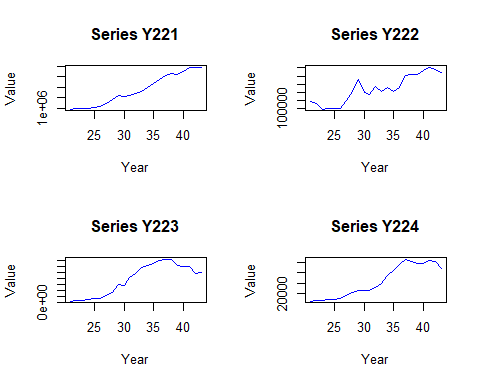
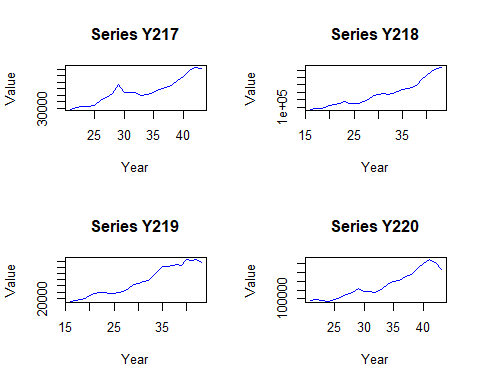
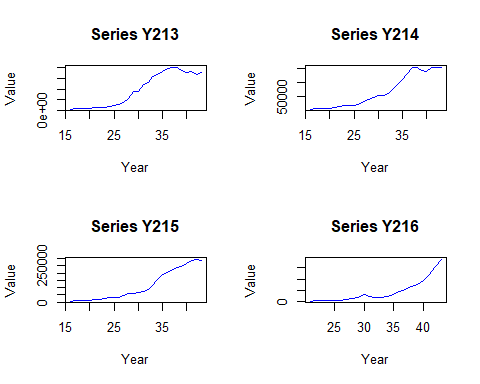
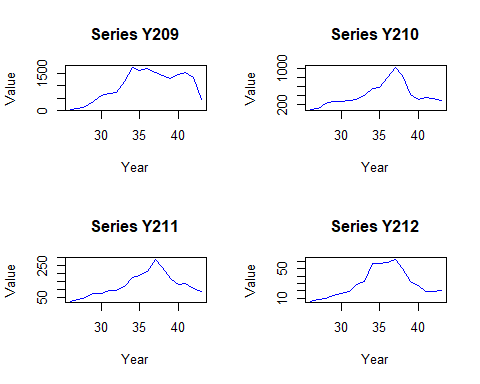
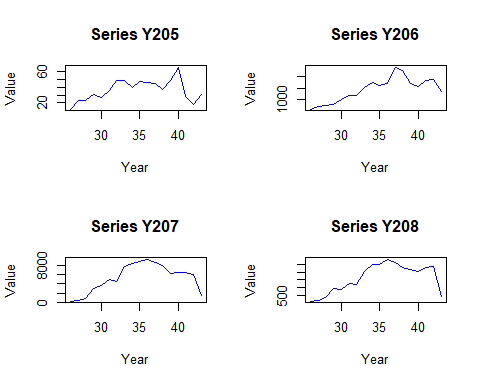
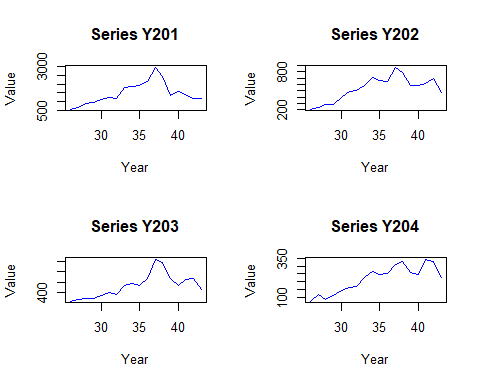
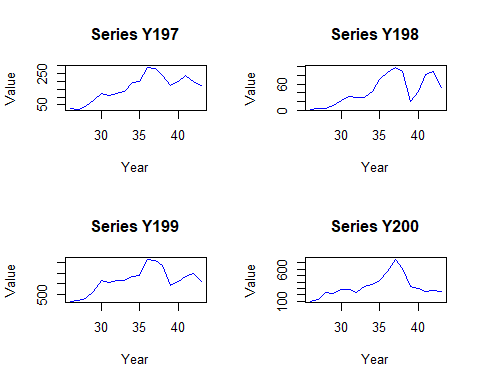
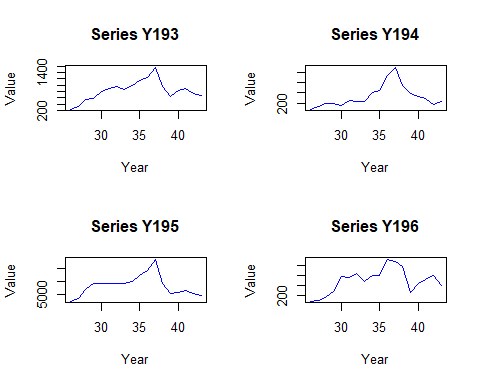
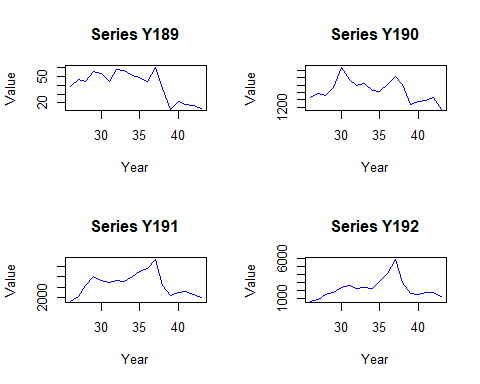
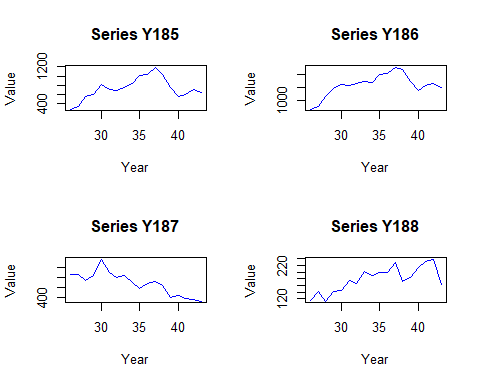
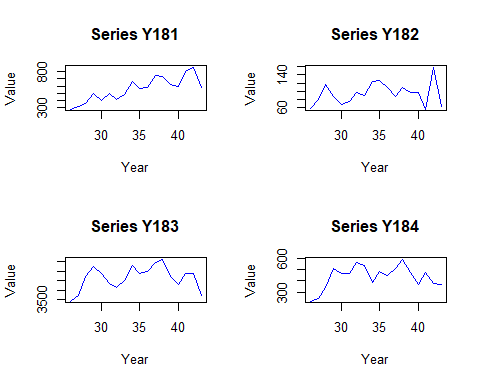
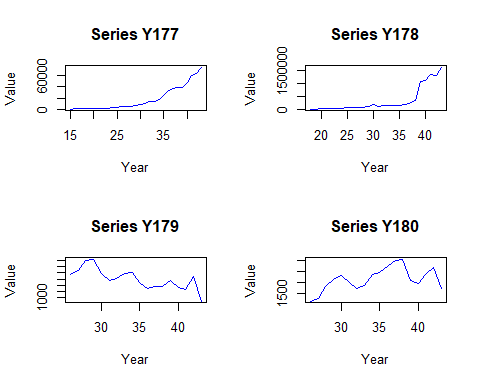
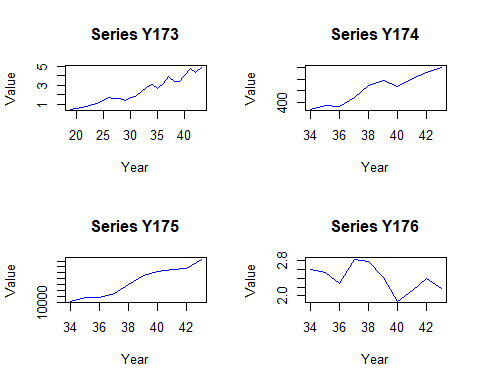
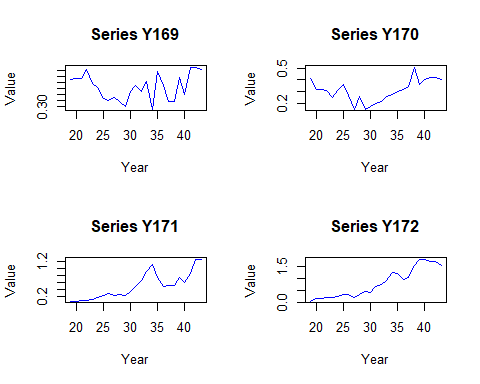
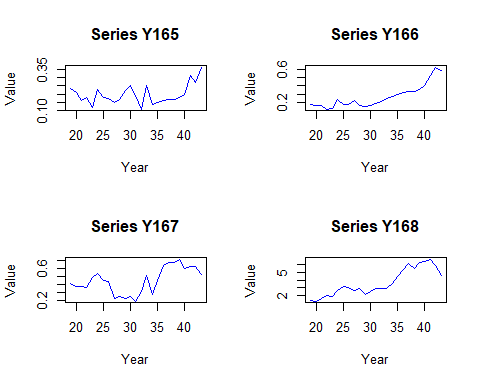
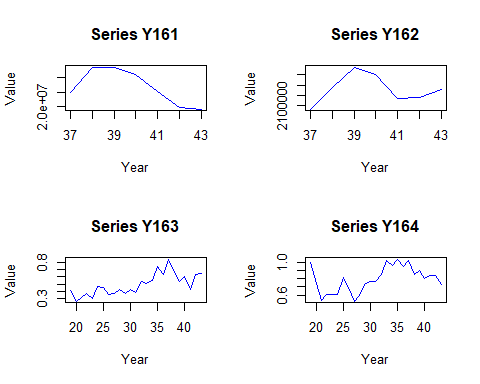
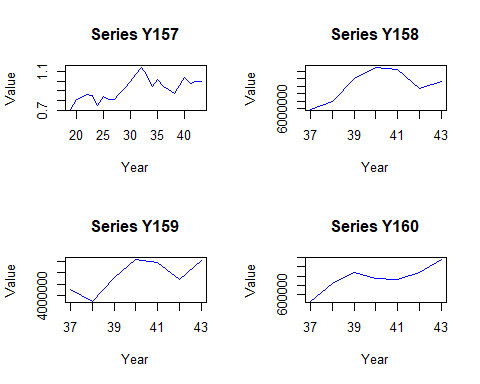
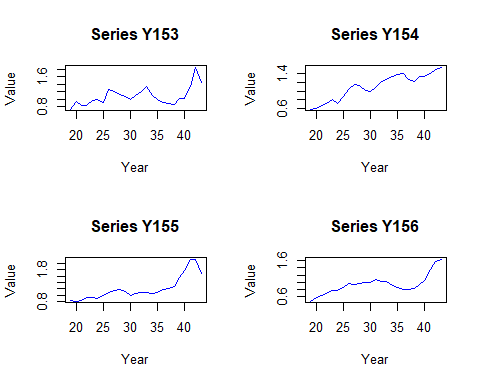
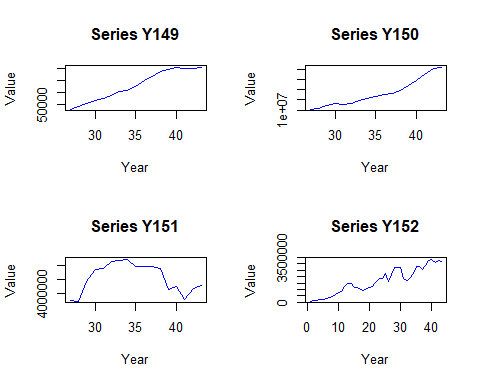
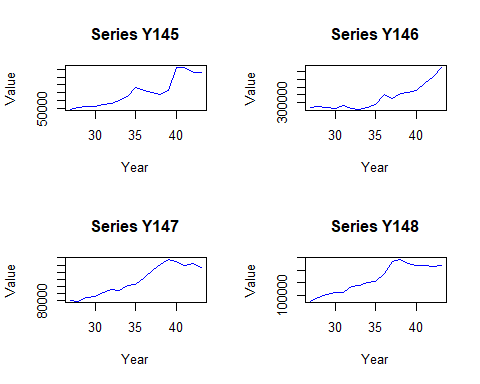
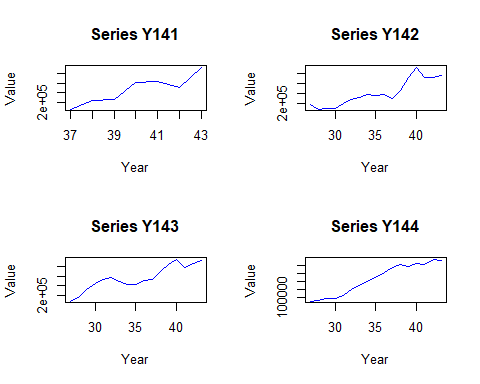
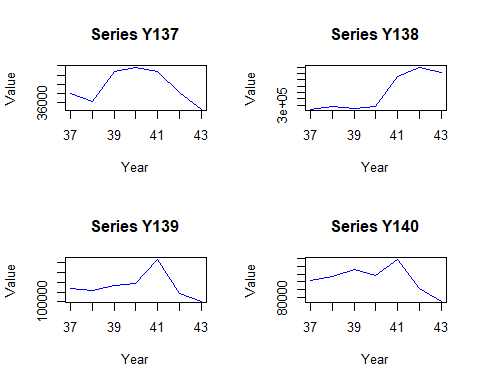
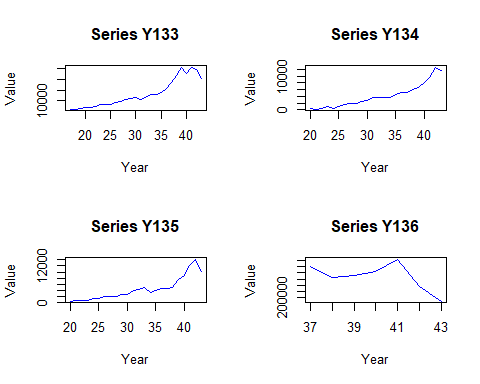
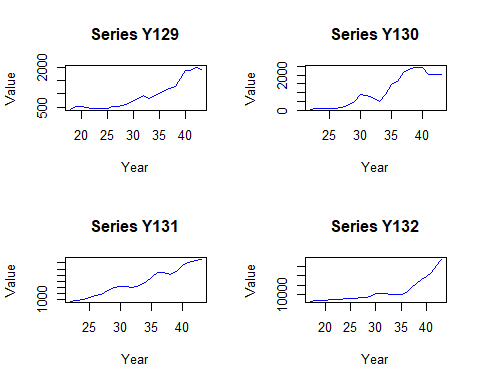
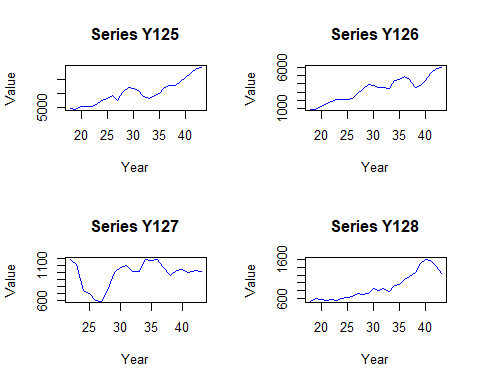
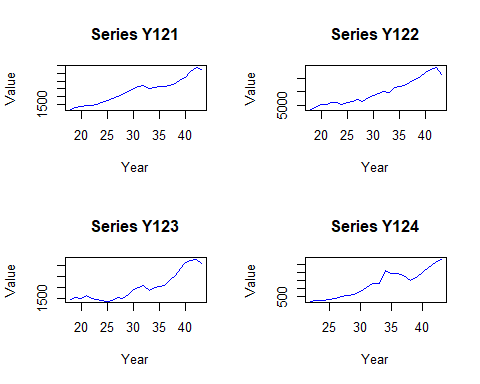
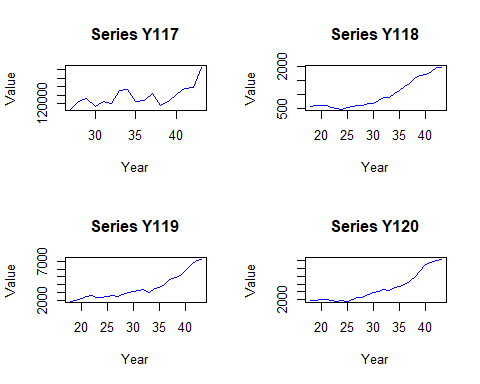
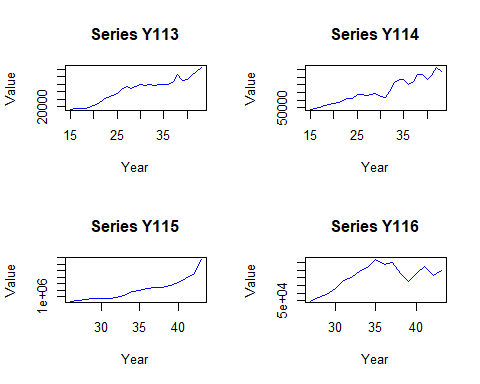
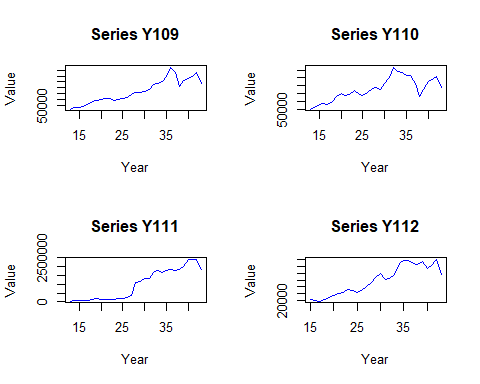
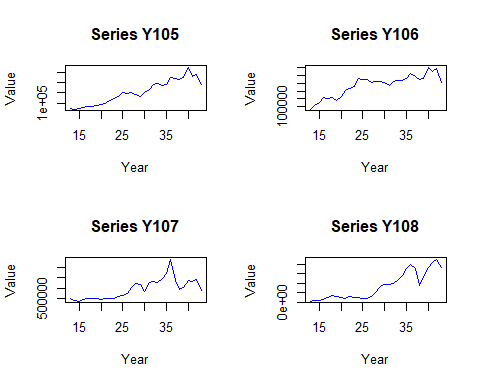
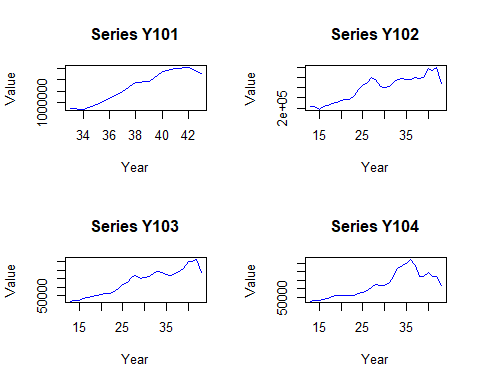
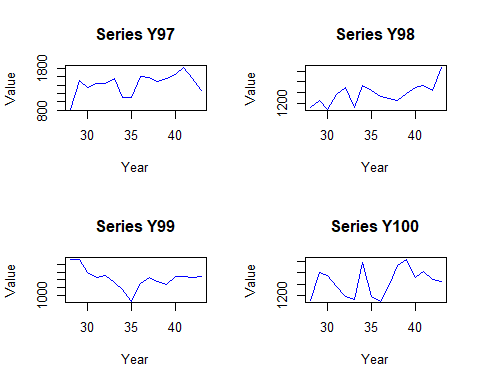
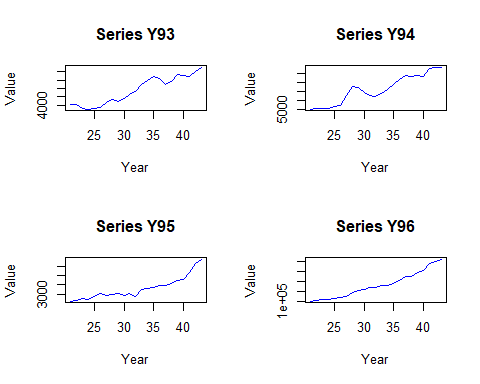
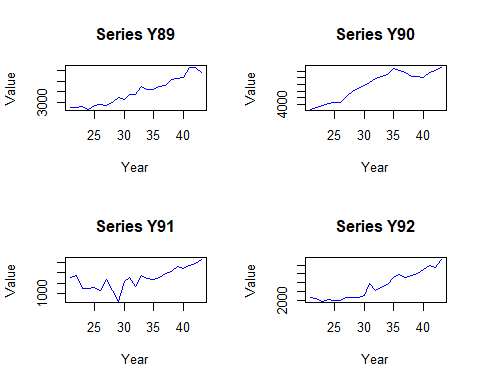
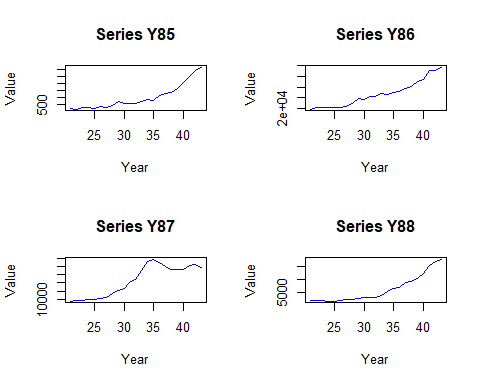
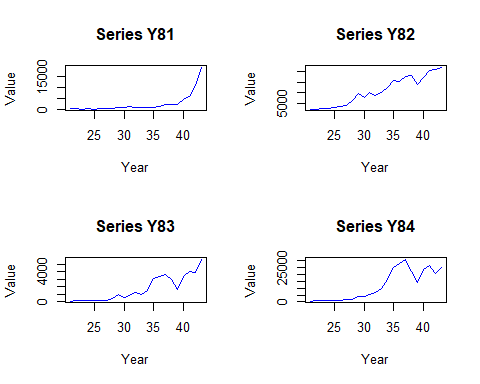
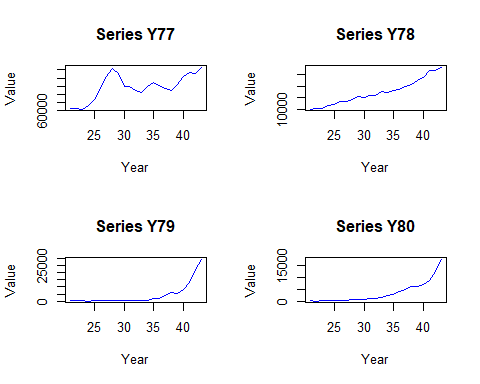
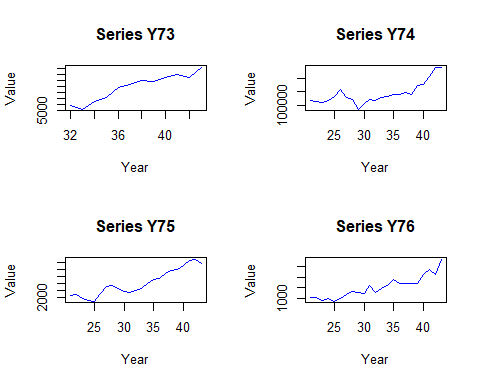
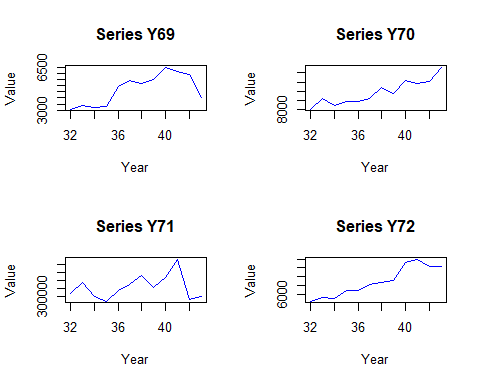
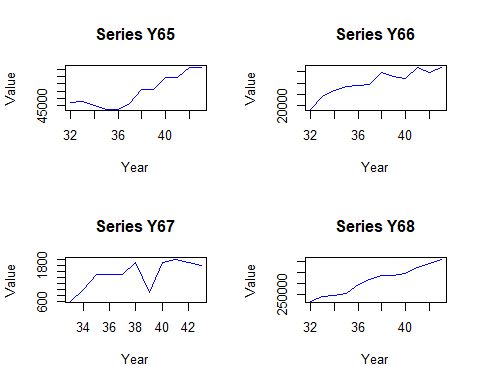
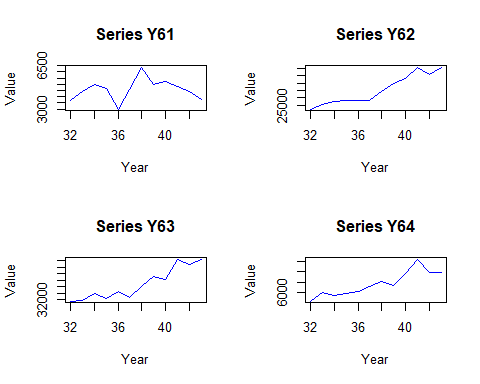
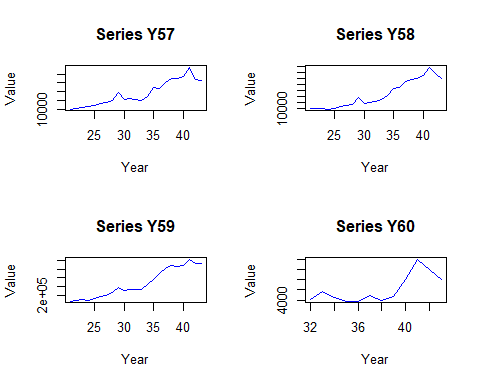
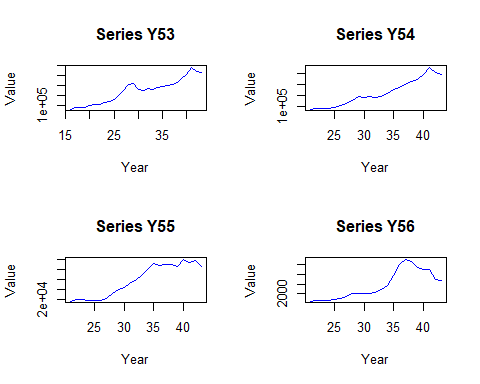
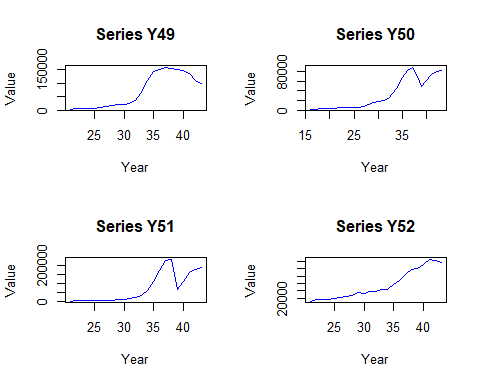
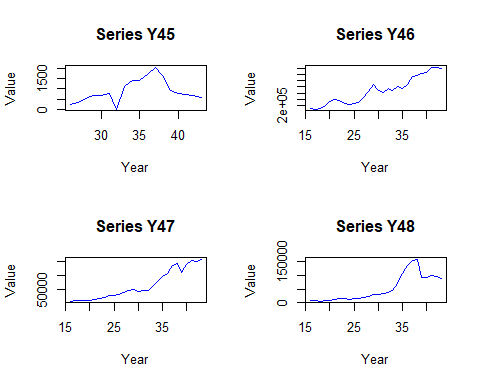
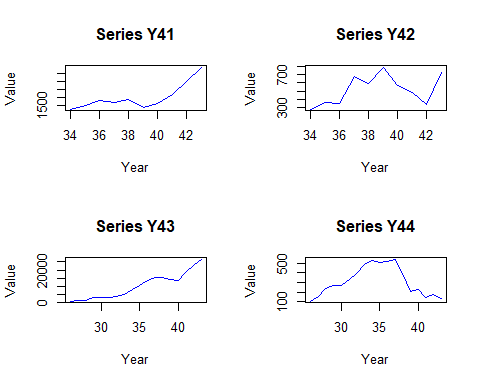
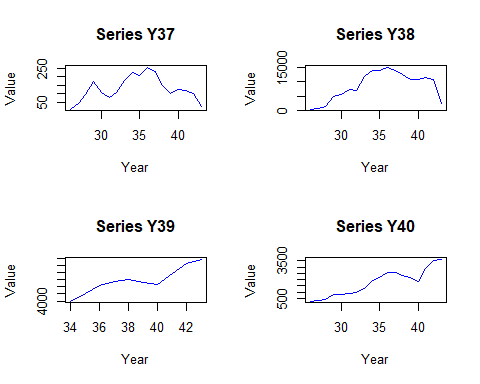
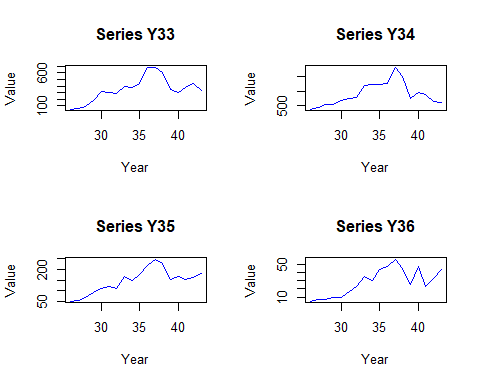
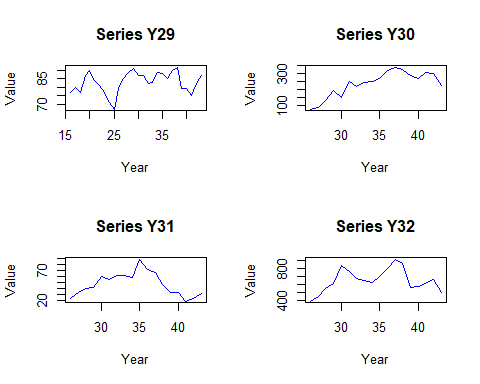
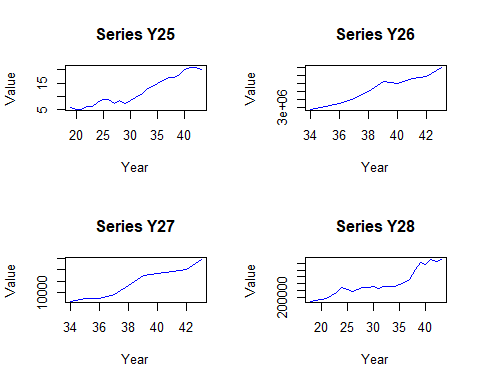
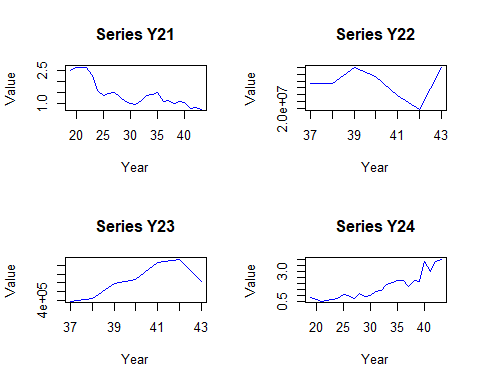
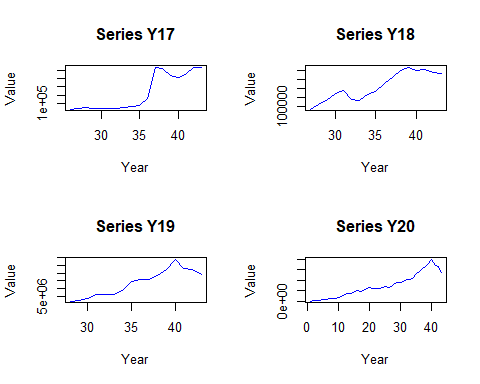
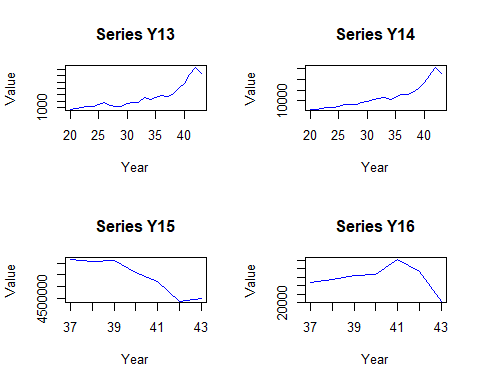
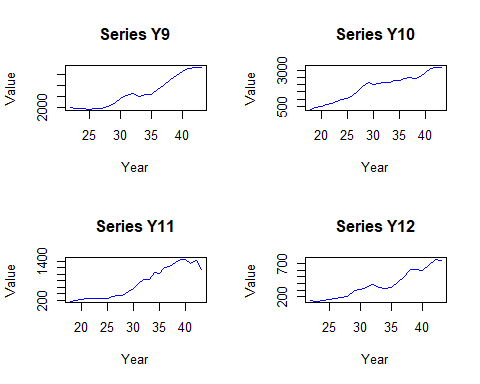
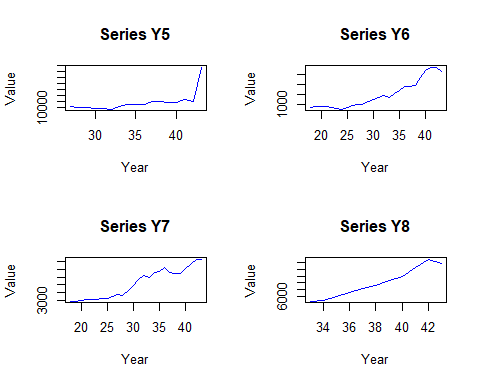
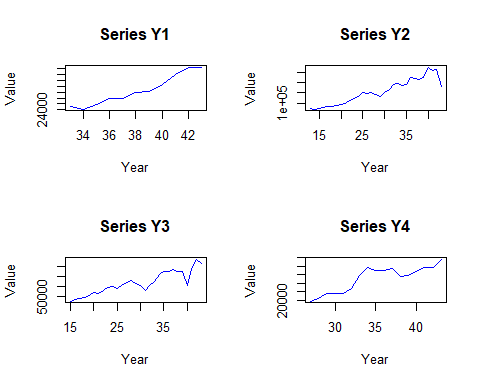
## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

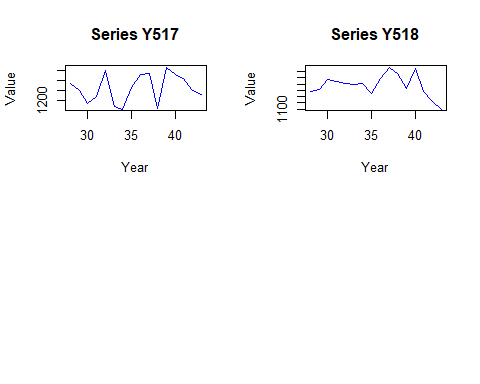
# Convert wide format to long format  
tourism\_data\_long <- tourism\_data |>  
 pivot\_longer(  
 cols = everything(),  
 names\_to = "Series",  
 values\_to = "Value"  
 ) |>  
 mutate(  
 Time = rep(1:nrow(tourism\_data), each = ncol(tourism\_data))  
 ) |>  
 filter(!is.na(Value))  
  
# Order series correctly  
tourism\_data\_long <- tourism\_data\_long |>  
 mutate(  
 # Extract numeric part from Series and convert to numeric for sorting  
 Series\_num = as.numeric(gsub("Y", "", Series))  
 ) |>  
 arrange(Series\_num, Time) |> # Sort by numeric series value and time  
 select(-Series\_num) # Remove the temporary numeric column  
  
# Check the result  
head(tourism\_data\_long, 10)

## # A tibble: 10 × 3  
## Series Value Time  
## <chr> <dbl> <int>  
## 1 Y1 25092. 33  
## 2 Y1 24272. 34  
## 3 Y1 25829. 35  
## 4 Y1 27698. 36  
## 5 Y1 27956. 37  
## 6 Y1 29924. 38  
## 7 Y1 30217. 39  
## 8 Y1 32613. 40  
## 9 Y1 36053. 41  
## 10 Y1 38473. 42

par(mfrow = c(2, 2))   
  
# Filter out series and plot them individually  
unique\_series <- unique(tourism\_data\_long$Series)  
for (series in unique\_series) {  
 series\_data <- tourism\_data\_long[tourism\_data\_long$Series == series,]  
 plot(series\_data$Time, series\_data$Value, type = 'l', main = paste("Series", series),  
 xlab = "Year", ylab = "Value", col = "blue")  
}



#Trends are visible: Several of the series display clear trends. A trend exists when there is a long-term increase or decrease in the data. For example, Series Y1, Y6,and Y7 show a general upward trend.  
  
#Similarities: Some series share similar trends, particularly those continuously increasing or decreasing. This might suggest that these series are influenced by common factors such as overall economic growth, tourism trends, or policy changes affecting multiple areas similarly.  
  
#Differences: Some are relatively stable with slight fluctuations, while others show significant volatility.Besides, the starting and ending values vary widely, suggesting different scales or magnitudes of the underlying activities.  
  
  
  
  
# Partition into training and validation sets  
training\_data <- tourism\_data\_long |>  
 group\_by(Series) |>  
 filter(Time <= max(Time) - 4)   
  
validation\_data <- tourism\_data\_long |>  
 group\_by(Series) |>  
 filter(Time > max(Time) - 4)   
  
# Generate naive forecasts  
naive\_forecasts <- training\_data |>  
 group\_by(Series) |>  
 summarize(  
 Last\_Value = last(Value),  
 Last\_Time = last(Time)  
 ) |>  
 # Create forecasts for 1-4 years ahead  
 crossing(  
 Horizon = 1:4  
 ) |>  
 mutate(  
 Forecast\_Year = Last\_Time + Horizon,  
 Forecast\_Value = Last\_Value   
 ) |>  
 select(Series, Forecast\_Year, Forecast\_Value)  
  
# Calculate all error measures  
error\_measures <- validation\_data |>  
 left\_join(naive\_forecasts, by = c("Series", "Time" = "Forecast\_Year")) |>  
 group\_by(Series) |>  
 summarize(  
 MAE = mean(abs(Value - Forecast\_Value)),  
 AvgError = mean(Value - Forecast\_Value),  
 MAPE = mean(abs((Value - Forecast\_Value)/Value)) \* 100,  
 RMSE = sqrt(mean((Value - Forecast\_Value)^2))  
 )  
  
# 5. Calculate MAPE for both periods  
# Training period MAPE  
training\_mape <- training\_data |>  
 group\_by(Series) |>  
 mutate(Forecast\_Value = lag(Value)) |>  
 filter(!is.na(Forecast\_Value)) |>  
 summarize(Training\_MAPE = mean(abs((Value - Forecast\_Value)/Value)) \* 100)  
  
# Validation period MAPE  
validation\_mape <- validation\_data |>  
 left\_join(naive\_forecasts, by = c("Series", "Time" = "Forecast\_Year")) |>  
 group\_by(Series) |>  
 summarize(Validation\_MAPE = mean(abs((Value - Forecast\_Value)/Value)) \* 100)  
  
# Combine MAPE results  
mape\_results <- full\_join(training\_mape, validation\_mape, by = "Series") |>  
 arrange(Series)  
  
#MAPE is scale-independent, making it suitable for combining series of different magnitudes; MAE and RMSE might be less suitable due to different scales across series



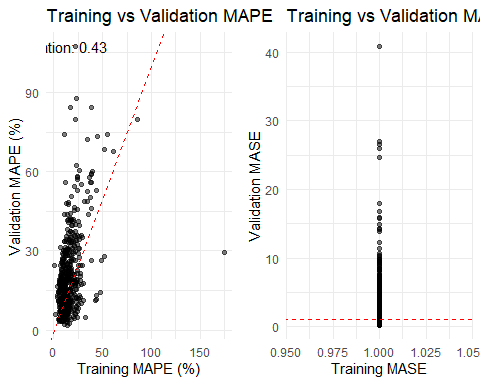
#The main advantages of Mean Absolute Scaled Error (MASE) are its scale-independence, which allows for direct comparison of forecast accuracy across different datasets with varying units, its robustness to outliers  
  
# Function to compute MASE  
compute\_mase <- function(actual, forecast, training\_data) {  
 # Calculate naive forecast errors from training data  
 training\_naive\_errors <- abs(diff(training\_data))  
 # Calculate mean training error (scaling factor)  
 scaling\_factor <- mean(training\_naive\_errors, na.rm = TRUE)  
 # Calculate scaled errors for the forecast  
 scaled\_errors <- abs(actual - forecast) / scaling\_factor  
 return(mean(scaled\_errors, na.rm = TRUE))  
}  
  
# Calculate MASE for training period  
training\_mase <- training\_data |>  
 group\_by(Series) |>  
 mutate(  
 Forecast\_Value = lag(Value), # Naive forecast  
 Training\_Values = list(Value)   
 ) |>  
 filter(!is.na(Forecast\_Value)) |>  
 summarize(  
 Training\_MASE = compute\_mase(Value, Forecast\_Value, first(Training\_Values))  
 )  
  
# Calculate MASE for validation period  
validation\_mase <- validation\_data |>  
 left\_join(naive\_forecasts, by = c("Series", "Time" = "Forecast\_Year")) |>  
 group\_by(Series) |>  
 mutate(  
 Training\_Values = list(training\_data$Value[training\_data$Series == first(Series)])  
 ) |>  
 summarize(  
 Validation\_MASE = compute\_mase(Value, Forecast\_Value, first(Training\_Values))  
 )  
  
# Combine results  
mase\_results <- full\_join(training\_mase, validation\_mase, by = "Series") |>  
 arrange(Series)

library(ggplot2)  
  
# Create scatter plot for MAPE  
mape\_plot <- ggplot(mape\_results, aes(x = Training\_MAPE, y = Validation\_MAPE)) +  
 geom\_point(alpha = 0.5) +  
 geom\_abline(intercept = 0, slope = 1, linetype = "dashed", color = "red") +  
 labs(  
 title = "Training vs Validation MAPE",  
 x = "Training MAPE (%)",  
 y = "Validation MAPE (%)"  
 ) +  
 theme\_minimal() +  
 annotate("text", x = min(mape\_results$Training\_MAPE, na.rm = TRUE),   
 y = max(mape\_results$Validation\_MAPE, na.rm = TRUE),  
 label = paste("Correlation:",   
 round(cor(mape\_results$Training\_MAPE,   
 mape\_results$Validation\_MAPE,   
 use = "complete.obs"), 2)))  
  
# Create scatter plot for MASE  
mase\_plot <- ggplot(mase\_results, aes(x = Training\_MASE, y = Validation\_MASE)) +  
 geom\_point(alpha = 0.5) +  
 geom\_abline(intercept = 0, slope = 1, linetype = "dashed", color = "red") +  
 labs(  
 title = "Training vs Validation MASE",  
 x = "Training MASE",  
 y = "Validation MASE"  
 ) +  
 theme\_minimal()  
  
# Display plots side by side  
library(gridExtra)

##   
## 载入程序包：'gridExtra'

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

grid.arrange(mape\_plot, mase\_plot, ncol = 2)



# Calculate summary statistics  
mape\_summary <- mape\_results |>  
 summarize(  
 Training\_Mean = mean(Training\_MAPE, na.rm = TRUE),  
 Training\_SD = sd(Training\_MAPE, na.rm = TRUE),  
 Validation\_Mean = mean(Validation\_MAPE, na.rm = TRUE),  
 Validation\_SD = sd(Validation\_MAPE, na.rm = TRUE)  
 )  
  
print("MAPE Summary Statistics:")

## [1] "MAPE Summary Statistics:"

print(mape\_summary)

## # A tibble: 1 × 4  
## Training\_Mean Training\_SD Validation\_Mean Validation\_SD  
## <dbl> <dbl> <dbl> <dbl>  
## 1 15.9 11.4 20.9 15.5

# Implementation native forecast with a constant trend   
naive\_trend\_forecast <- function(data, horizon = 1:4, growth\_rate = 1.06) {  
 last\_value <- tail(data$Value, 1)  
 forecasts <- last\_value \* growth\_rate^horizon  
 return(forecasts)  
}  
  
#Rational: Global tourism typically grows at a steady rate, and historical tourism data shows consistent growth patterns.  
  
# Time series linear regression model  
# Dependent variable: Tourism value  
# Predictor: Time index  
ts\_linear\_model <- function(data) {  
 # Convert to time series object  
 ts\_data <- ts(data$Value, frequency = 1)  
   
 # Create predictors  
 time\_index <- 1:length(ts\_data)  
   
 # Fit linear model  
 lm(ts\_data ~ time\_index)  
}

library(dplyr)  
  
# Get first 5 series  
first\_5\_series <- unique(training\_data$Series)[1:5]  
  
# Loop through each series  
for(series in first\_5\_series) {  
 # Get training and validation data for this series  
 train <- training\_data |> filter(Series == series)  
 valid <- validation\_data |> filter(Series == series)  
   
 # fit the model  
 model <- lm(Value ~ Time, data = train)  
 forecasts <- predict(model, newdata = data.frame(Time = valid$Time))  
 errors <- valid$Value - forecasts  
   
 # Print results  
 cat("\nSeries:", series, "\n")  
 cat("Forecasts:", round(forecasts, 2), "\n")  
 cat("Actual Values:", round(valid$Value, 2), "\n")  
 cat("Forecast Errors:", round(errors, 2), "\n")  
 cat("MAPE:", round(mean(abs(errors/valid$Value)) \* 100, 2), "%\n")  
}

##   
## Series: Y1   
## Forecasts: 31399.23 32428.05 33456.87 34485.68   
## Actual Values: 32613.5 36053.17 38472.75 38420.89   
## Forecast Errors: 1214.27 3625.12 5015.89 3935.21   
## MAPE: 9.26 %  
##   
## Series: Y2   
## Forecasts: 360295.9 373048.3 385800.6 398553   
## Actual Values: 450569 421513 426166 265729   
## Forecast Errors: 90273.06 48464.72 40365.38 -132824   
## MAPE: 22.75 %  
##   
## Series: Y3   
## Forecasts: 185908.8 192080.8 198252.9 204424.9   
## Actual Values: 104885 188069 236275 212260   
## Forecast Errors: -81023.77 -4011.83 38022.12 7835.06   
## MAPE: 24.79 %  
##   
## Series: Y4   
## Forecasts: 107664.6 114166.2 120667.8 127169.4   
## Actual Values: 88534 97115 96381 117330.2   
## Forecast Errors: -19130.58 -17051.2 -24286.82 -9839.25   
## MAPE: 18.19 %  
##   
## Series: Y5   
## Forecasts: 15324.08 15799.59 16275.11 16750.63   
## Actual Values: 14929 17057 15798 43985.64   
## Forecast Errors: -395.08 1257.41 -477.11 27235.01   
## MAPE: 18.74 %

#Using MASE to select polynomial order but R² for blending is inconsistent:MASE measures forecast accuracy,R² measures goodness of fit to training data  
  
#Higher-order polynomials risk overfitting:While they might give better R² values for training data  
  
#Holt's method (trend but no seasonality) because tourism data often shows clear trends and the data is annual, so seasonality isn't a concern.

# Automated ensemble weighting  
  
create\_automated\_ensemble <- function(training\_data, validation\_data) {  
 # Fit different models  
 models <- list(  
 # Naive forecast with trend  
 naive = function(x) forecast(naive(x), h = 4),  
 # Linear regression  
 linear = function(x) forecast(tslm(x ~ trend), h = 4),  
 # Exponential smoothing  
 ets = function(x) forecast(ets(x), h = 4)  
 )  
   
 # Calculate weights based on validation performance  
 weights <- sapply(models, function(model) {  
 accuracy(model(training\_data))[,"MASE"]  
 })  
   
 # Inverse MASE for weights (lower MASE = higher weight)  
 weights <- 1/weights  
 weights <- weights/sum(weights)  
   
 return(weights)  
}

Key differences in real-life scenarios: 1. Multiple Horizons: Competition: Fixed 4-year horizon Reality: Various planning horizons needed External Factors: Competition: Pure time series Reality: Economic indicators, events, policies Uncertainty Handling: Competition: Point forecasts Reality: Prediction intervals and scenarios 4. Stakeholder Needs: Competition: Single metric (MAPE) Reality: Multiple objectives (capacity planning, budgeting) 5. Data Frequency: Competition: Annual data Reality: Often monthly/quarterly with seasonality

Real-life steps would include: Stakeholder consultation External factor analysis Scenario planning Regular model updating Risk assessment Seasonal adjustments