ISA 444: Business Forecasting

26: Time Series Regression

Fadel M. Megahed, PhD

Endres Associate Professor
Farmer School of Business
Miami University

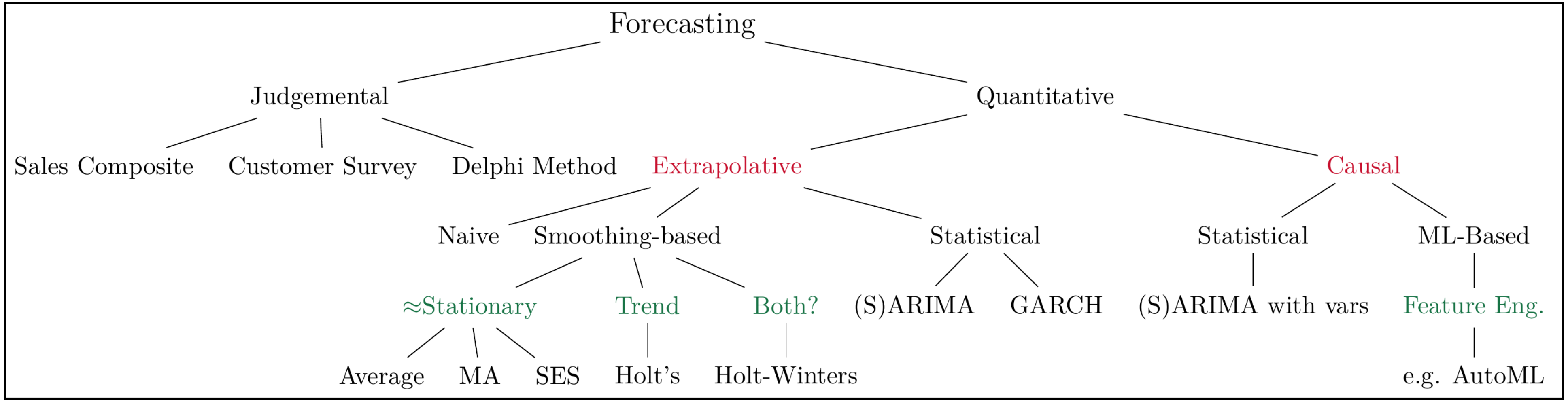
@FadelMegahed
fmegahed
fmegahed@miamioh.edu
Automated Scheduler for Office Hours

Spring 2023

Quick Refresher from Last Class

- ✅ Explain the simple and multiple linear regression models and interpret the parameters.
- ✅ Interpret the sample linear regression coefficients in the language of the problem.
- ✅ Use a simple linear regression model for trend adjustment (time-series data).

Overview of Univariate Forecasting Methods



A 10,000 foot view of forecasting techniques

Notes: My (incomplete) classification of **univariate** forecasting techniques, i.e., they exclude popular approaches used in multivariate time series forecasting.

Learning Objectives for Today's Class

- Use a simple linear regression model for trend adjustment (time-series data).
- Interpret regression diagnostic plots.
- Create prediction intervals for individual values of the response variable.
- Use regression to account for seasonality in a time series.

Using Simple Linear Regression Model for Trend Adjustment (with Time Series Data)

Continuing our Example from Last Class

```
jj = astsa::jj

# Step 1: Plot the ts data (saved to object to not 
p = forecast::autoplot(jj) +
  ggplot2::theme_bw() +
  ggplot2::geom_point(size = 1)

# stabilizing the variance
log_jj = log(jj)

# Step 1b: Our updated plot
p2 = forecast::autoplot(log_jj) +
  ggplot2::theme_bw() +
  ggplot2::geom_point(size = 1)

# Step 2: Extract time
year = time(log_jj)

# Step 3: Fit the regression model
reg_model = lm(log_jj ~ year)

summary(reg_model) # prints top right

anova(reg_model) # prints bottom right
```

```
##
## Call:
## lm(formula = log_jj ~ year)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.38309 -0.08569  0.00297  0.09984  0.38016
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.275e+02  5.623e+00  -58.25   <2e-16 ***
## year         1.668e-01  2.854e-03   58.45   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1
##
## Residual standard error: 0.1585 on 82 degrees of freedom
## Multiple R-squared:  0.9766,    Adjusted R-squared:  0.9
## F-statistic:  3416 on 1 and 82 DF,  p-value: < 2.2e-16
```

```
## Analysis of Variance Table
##
## Response: log_jj
##           Df Sum Sq Mean Sq F value    Pr(>F)
## year       1 85.872  85.872  3416.5 < 2.2e-16 ***
## Residuals 82  2.061   0.025
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1
```

Continuing our Example from Last Class

05:00

What would be our forecast values log EPS for 1981?

Manual Calculations:

Recall that our regression equation from `summary(reg_model)` was:

$$\text{log\_jj} = -327.5 + 0.1668 \times \text{year}$$

- For Q1, my predicted log EPS = ...
- For Q2, my predicted log EPS = ...
- For Q3, my predicted log EPS = ...
- For Q4, my predicted log EPS = ...

R Functions:

```r
# Approach (a):
pred1981a = predict(
  object = reg_model,
  newdata =
    data.frame(year = c(1981, 1981.25, 1981.5, 1981
)

# Approach (b): I prefer this approach
pred1981b = forecast::forecast(
  object = reg_model,
  newdata =
    data.frame(year = c(1981, 1981.25, 1981.5, 1981
)
```

Why is approach (b) slightly better? (ignoring the need to load an extra package)

Approach (b) is better because:

- ...

`forecast::tslm()` vs. `lm()`

Let us examine and contrast the output from the `forecast::tslm()`, compared to what we obtained in Slide 6. There are **3** reasons for which I tend to prefer the `forecast::tslm()` (if you are willing to ignore the need to load an extra package):

Please fill after we go through the example:

- ...
- ...
- ...

Our Example with `forecast::tslm()`

```r
model_tslm =
  forecast::tslm(log_jj ~ trend)

summary(model_tslm)
```

From the output, what are the intercept and slope for year? Interpret their values.

Slope and intercept from `forecast::tslm()`:

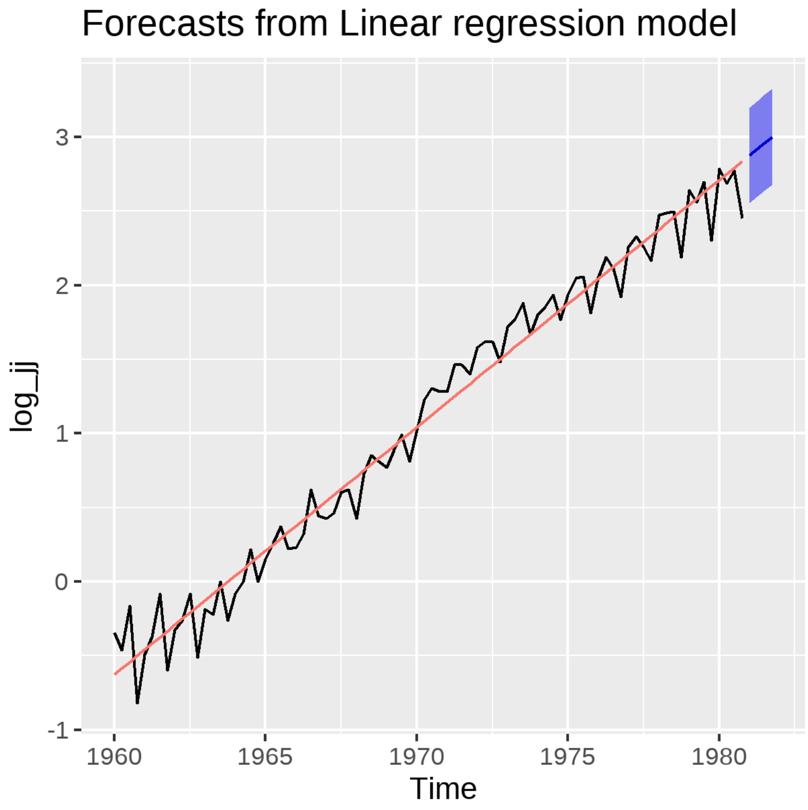
- Intercept: ...
- Interpretation: ...
- Slope: ...
- Interpretation: ...
- Difference from `lm()` output: ...

```
##
## Call:
## forecast::tslm(formula = log_jj ~ trend)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.38309 -0.08569  0.00297  0.09984  0.38016
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t
## (Intercept) -0.6677756  0.0349073  -19.13   <2e-
## trend        0.0416992  0.0007134   58.45   <2e-
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05
##
## Residual standard error: 0.1585 on 82 degrees of
## Multiple R-squared:  0.9766,	Adjusted R-squar
## F-statistic:  3416 on 1 and 82 DF,  p-value: < 2
```

Our Example with `forecast::tslm()`

```r
model_tslm =
  forecast::tslm(log_jj ~ trend)

pred_1981_tslm =
  forecast::forecast(
    model_tslm, h = 4, level = 95
    )

# Printing the point estimates
pred_1981_tslm$mean

# TS Plot w/ forecast, fitted values, &
# 95% PI ( defined in forecast() )
forecast::autoplot(pred_1981_tslm) +
  forecast::autolayer(
    fitted(pred_1981_tslm)
    ) +
  ggplot2::theme(
    legend.position = 'none'
    )
```

```
##          Qtr1     Qtr2     Qtr3     Qtr4
## 1981 2.876655 2.918354 2.960053 3.001752
```



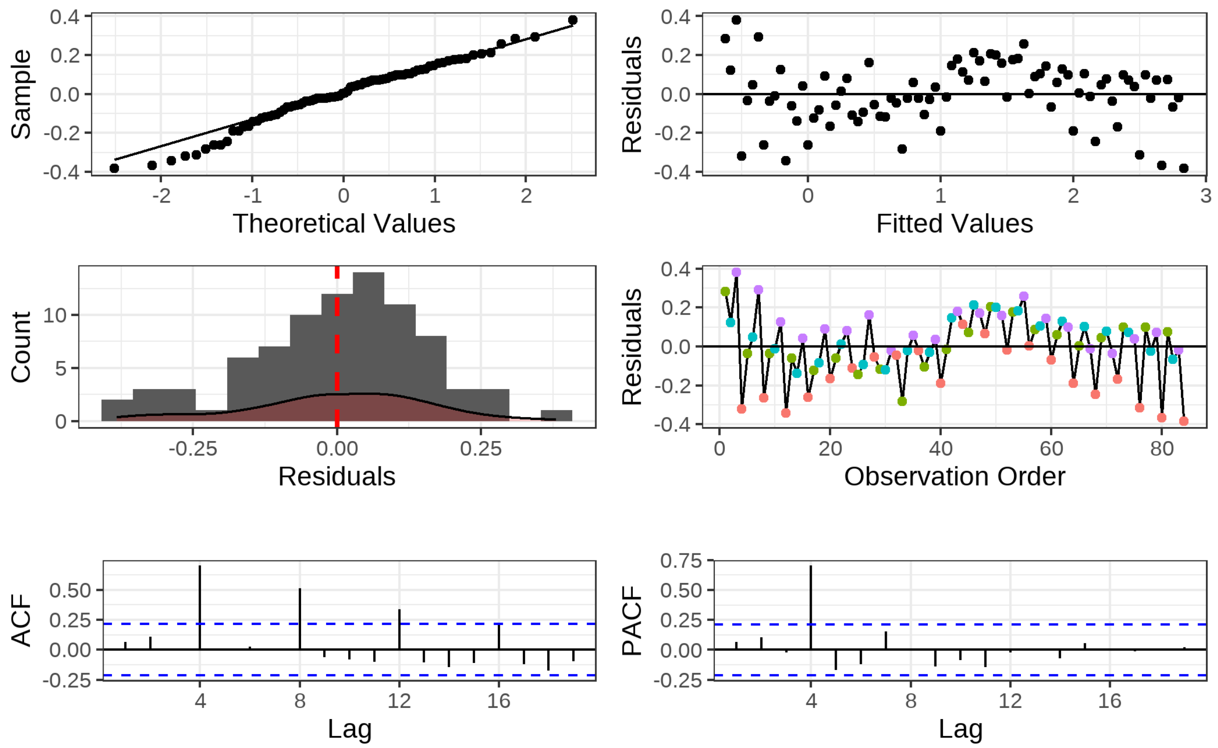
Regression Diagnostic Plots

Regression Diagnostic Plots

```r
resplot <- function(res, fit, ncol = 2, nrow = 3, freq = NULL){
  ggplot2::theme_set(ggplot2::theme_bw()) # setting the theme for all ggplots to be black and white

  df = data.frame(res = res, fit = fit) # creating a df of residuals and fitted values

  # Plot 1: QQ Plot for the residuals
  qq = ggplot2::ggplot(df, ggplot2::aes(sample = res)) +
    ggplot2::stat_qq() +  ggplot2::stat_qq_line() +  ggplot2::labs(x = "Theoretical Values", y = "Sample")

  # Plot 2: Scatter Plot of Residuals Vs. Fitted Values
  scatter = ggplot2::ggplot(df, ggplot2::aes(x = fit, y = res)) +
    ggplot2::geom_point() + ggplot2::geom_hline(yintercept = 0) +
    ggplot2::labs(x = "Fitted Values", y = "Residuals")

  # Plot 3: Histogram of Residuals with Density Plot
  histogram = ggplot2::ggplot(df, aes(x = res)) +
    ggplot2::geom_histogram(bins = 15) + ggplot2::geom_density(alpha = 0.2, fill = "red") +
    ggplot2::geom_vline(ggplot2::aes(xintercept= mean(res)), color="red", linetype="dashed", size=1) + ggplot2::labs(x = "Residuals", y = "Count")

  # Plot 4: Time Order Plot of Residuals
  if(is.numeric(freq)){
    colorFactor = as.factor( as.numeric(row.names(df)) %% freq ) # coloring based on the modulus operation
    timeOrder = ggplot2::ggplot(df, ggplot2::aes(x = as.numeric(row.names(df)), y = res) ) +
      ggplot2::geom_line() + ggplot2::geom_point(aes(color = colorFactor)) + ggplot2::geom_hline(yintercept = 0) +
      ggplot2::labs(x = "Observation Order", y = "Residuals") + ggplot2::theme(legend.position = "none")
  }else{
    timeOrder = ggplot2::ggplot(df, ggplot2::aes(x = as.numeric(row.names(df)), y = res)) +
      ggplot2::geom_line() + ggplot2::geom_point() + ggplot2::geom_hline(yintercept = 0) +
      ggplot2::labs(x = "Observation Order", y = "Residuals")
  }

  # Plot 5: ACF Plot of Residuals
  acfPlot = forecast::ggAcf(res) + ggplot2::labs(title = "")

  # Plot 6: ACF Plot of Residuals
  pacfPlot = forecast::ggPacf(res) + ggplot2::labs(title = "")

  # Putting all 6 figures together
  ggpubr::ggarrange(qq, scatter, histogram, timeOrder, acfPlot, pacfPlot, ncol = ncol, nrow = nrow)
}
```

Applying the `resPlot()`

```
residuals_tslm = model_tslm$residuals
fitted_tslm = model_tslm$fitted.values

resplot(res = residuals_tslm, fit = fitted_tslm, freq = 4)
```



Insights from the `resPlot()`

- Assumption behind using the forecast function is that we have a model that fits well to our data.
- The diagnostics on the regression model confirmed that we do not have an excellent model since our residuals are not iid. Specifically, the:
  - The residuals **vary in magnitude as a function of the fitted values**.
  - The residuals show a **seasonal pattern**, where Q2 residuals are typically large and Q4 residuals are typically small.
  - The ACF and PACF show a **seasonal pattern in the residuals** – likely need to fit an AR model.

Seasonal Regression

Using the base `lm()`

```r
# extracting time and quarter
t = time(log_jj) # renamed year var from last class
q = cycle(log_jj) |> factor()

model3 = lm(log_jj ~ t + q)

# checking out how the data is set up for lm
model.matrix(model3)
```

```
##    (Intercept)       t q2 q3 q4
## 1            1 1960.00  0  0  0
## 2            1 1960.25  1  0  0
## 3            1 1960.50  0  1  0
## 4            1 1960.75  0  0  1
## 5            1 1961.00  0  0  0
## 6            1 1961.25  1  0  0
## 7            1 1961.50  0  1  0
## 8            1 1961.75  0  0  1
## 9            1 1962.00  0  0  0
## 10           1 1962.25  1  0  0
## 11           1 1962.50  0  1  0
## 12           1 1962.75  0  0  1
## 13           1 1963.00  0  0  0
## 14           1 1963.25  1  0  0
## 15           1 1963.50  0  1  0
## 16           1 1963.75  0  0  1
## 17           1 1964.00  0  0  0
## 18           1 1964.25  1  0  0
## 19           1 1964.50  0  1  0
## 20           1 1964.75  0  0  1
## 21           1 1965.00  0  0  0
## 22           1 1965.25  1  0  0
## 23           1 1965.50  0  1  0
## 24           1 1965.75  0  0  1
## 25           1 1966.00  0  0  0
## 26           1 1966.25  1  0  0
## 27           1 1966.50  0  1  0
## 28           1 1966.75  0  0  1
```

Using the base `lm()`

```r
# extracting time and quarter
t = time(log_jj) # renamed year var from last class
q = cycle(log_jj) |> factor()

model3 = lm(log_jj ~ t + q)

# model summary
summary(model3)
```

```
##
## Call:
## lm(formula = log_jj ~ t + q)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.29318 -0.09062 -0.01180  0.08460  0.27644
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t
## (Intercept) -3.283e+02  4.451e+00 -73.761  < 2e-
## t            1.672e-01  2.259e-03  73.999  < 2e-
## q2           2.812e-02  3.870e-02   0.727   0.46
## q3           9.823e-02  3.871e-02   2.538   0.01
## q4          -1.705e-01  3.873e-02  -4.403 3.31e-
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05
##
## Residual standard error: 0.1254 on 79 degrees of
## Multiple R-squared:  0.9859,    Adjusted R-squar
## F-statistic:  1379 on 4 and 79 DF,  p-value: < 2
```

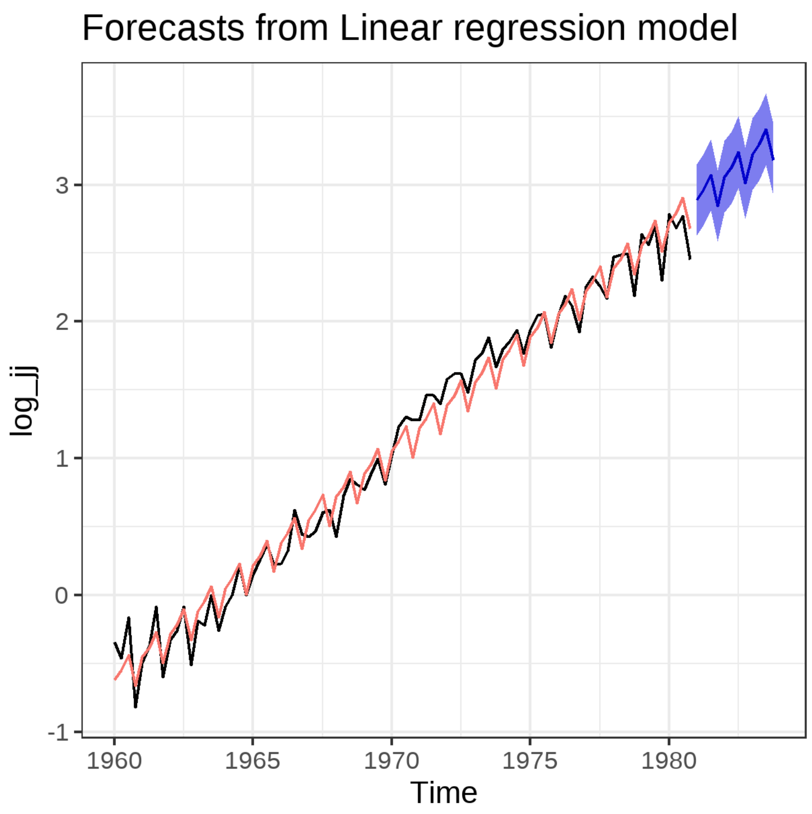
Using forecast::tslm()

```r
model4 =
  forecast::tslm(
    log_jj ~ trend + season
  )

summary(model4)
```

```
##
## Call:
## forecast::tslm(formula = log_jj ~ trend + season
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.29318 -0.09062 -0.01180  0.08460  0.27644
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t
## (Intercept) -0.6607215  0.0358430 -18.434  < 2e-
## trend        0.0417930  0.0005648  73.999  < 2e-
## season2      0.0281227  0.0386959   0.727   0.46
## season3      0.0982310  0.0387083   2.538   0.01
## season4     -0.1705267  0.0387289  -4.403 3.31e-
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05
##
## Residual standard error: 0.1254 on 79 degrees of
## Multiple R-squared:  0.9859,    Adjusted R-squar
## F-statistic:  1379 on 4 and 79 DF,  p-value: < 2
```

Using `forecast::tslm()`

```
# forecasting 12 quarters ahead
forecast_3yrs =
  forecast::forecast(
    model4, h = 12, level = 95
    )

# a quick plot of the forecast
forecast::autoplot(forecast_3yrs) +
  forecast::autolayer(fitted(forecast_3yrs)) +
  ggplot2::theme_bw() +
  ggplot2::theme(legend.position = 'none')
```



Model Comparison: BIC & Forecast Accuracy

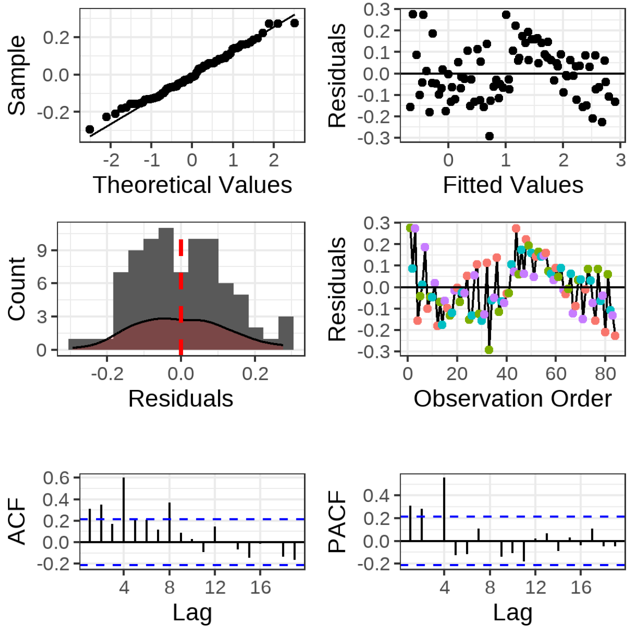
```r
# compare their BIC (lower better)
cbind(
  BIC(model_tslm), BIC(model4)
)

# comparing their forecast accuracy
acc_values = rbind(
  forecast::accuracy(model_tslm),
  forecast::accuracy(model4)
)

row.names(acc_values) = c('reg w/ trend', 'reg w/ tr

acc_values
```

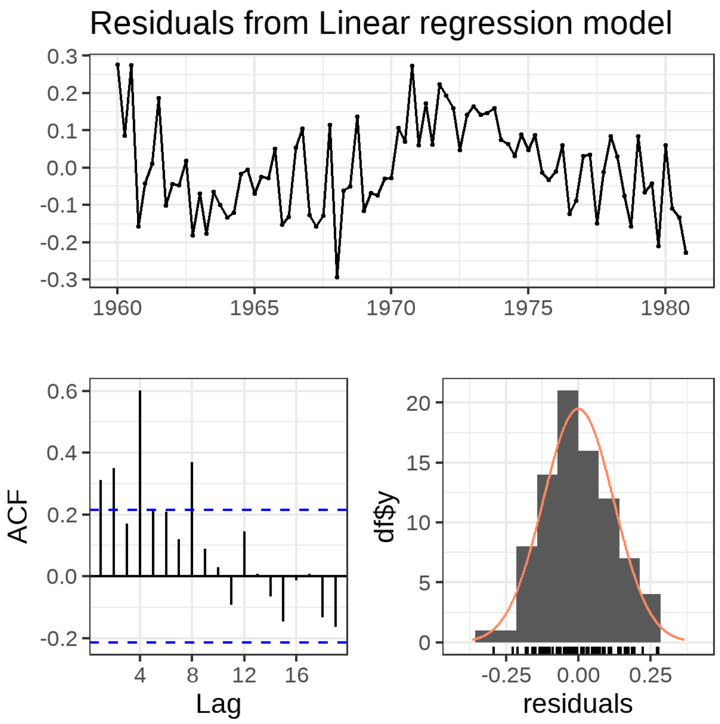
```
##             [,1]      [,2]
## [1,] -59.76488 -89.03043
```

```
##                                  ME      RMSE       MAE  MPE MAPE
## reg w/ trend          -5.046058e-18 0.1566402 0.1233944  NaN  Inf
## reg w/ trend + season  3.469447e-18 0.1215870 0.1007889 -Inf  Inf
##                             ACF1
## reg w/ trend          0.06453504
## reg w/ trend + season 0.31116820
```

Model 4's Assumptions

```r
# using resplot or check residuals
resplot(
  res = model4$residuals,
  fit = model4$fitted.values,
  freq = 4
  )

forecast::checkresiduals(model4)
```






```
##
##     Breusch-Godfrey test for serial correlation of order up to 8
##
## data:  Residuals from Linear regression model
## LM test = 39.408, df = 8, p-value = 4.128e-06
```

Comparison with `auto.arima()`

05:00

Is the `auto.arima()` model better for this dataset?

Recap

Summary of Main Points

By now, you should be able to do the following:

- Use a simple linear regression model for trend adjustment (time-series data).
- Interpret regression diagnostic plots.
- Create prediction intervals for individual values of the response variable.
- Use regression to account for seasonality in a time series.

Things to Do to Prepare for Next Class

- Go through the slides, examples and make sure you have a good understanding of what we have covered.
- Read Chapter 7 in our reference book Principles of Business Forecasting.