WiCS_Hacks_2025

```r
file <- "r.csv"
df <- read_csv(file, col_names = c("Time", "Quarter", "Year", "LocationGroup", "Frequency"))
```

```
## Rows: 446 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (4): Quarter, Year, LocationGroup, Frequency
## dbl (1): Time
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

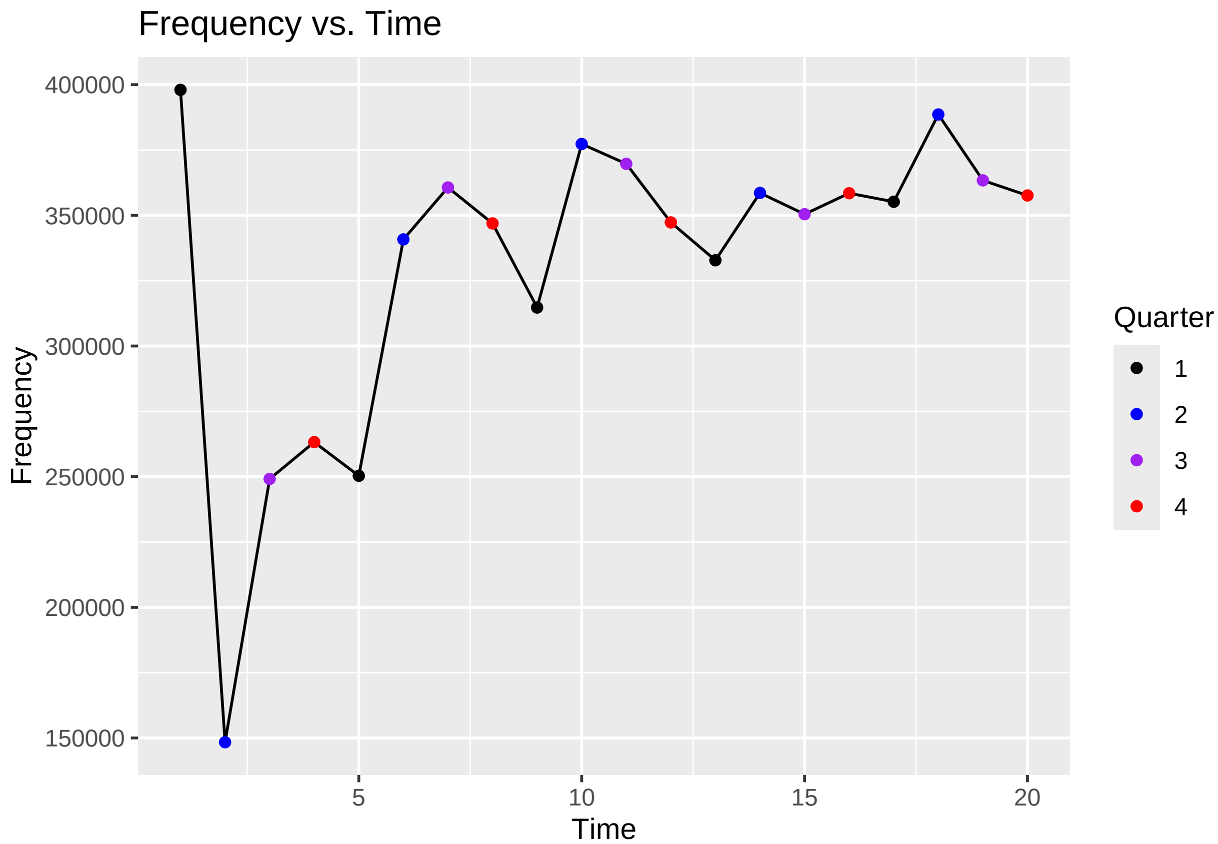
```r
df <- df[-1, ]
f <- function(group="West Campus") {
  temp <- df %>%
    filter(LocationGroup == group) %>%
    select(-LocationGroup) %>%
    arrange(Year, Quarter) %>%
    mutate(Time = row_number()) %>%
    #select(-Year) %>%
    mutate(Year = as.integer(Year)) %>%
    mutate(Quarter = as.integer(Quarter)) %>%
    mutate(Frequency = as.integer(Frequency)) %>%
    slice_head(n = -1)
  return(temp)
}
wampus <- f('Core')
first_year <- wampus$Year[1]
first_quarter <- wampus$Quarter[1]
yearquarter_input <- paste(c(first_year, " Q", first_quarter), collapse = "")
rows <- as.integer(nrow(wampus))
```

```r
#
# create tsibble
#
wampus_ts <- wampus %>%
  add_column(qtr=yearquarter(yearquarter_input) + 0:(rows-1), .before=TRUE) %>%
  as_tsibble(index=qtr)
#
# Compute log(Sales)
#
wampus_ts <-wampus_ts %>%
  mutate(LogFreq = log(Frequency)) %>%
  mutate(TimeSq = Time^2)
#
```

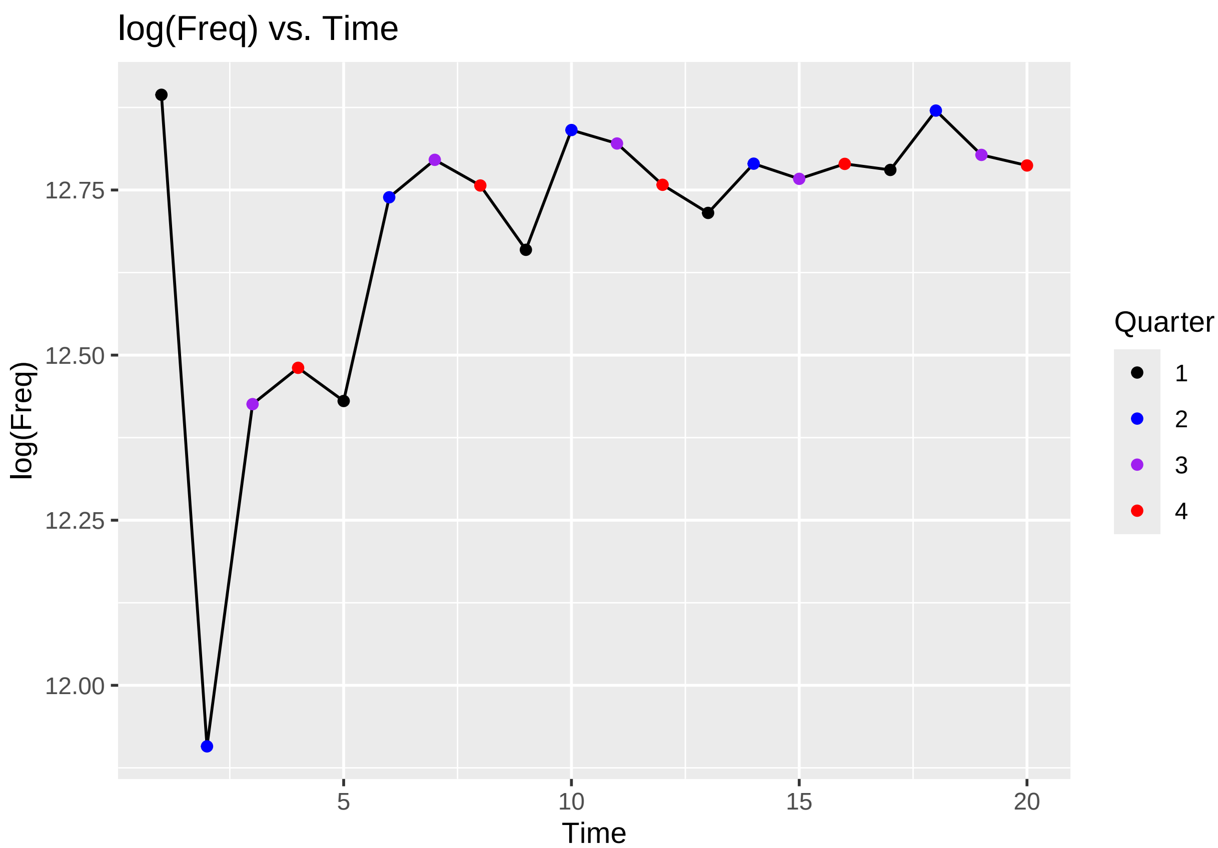
```r
# print tsibble
#
head(wampus_ts, n=10)
```

```
## # A tsibble: 10 x 7 [1Q]
##        qtr  Time Quarter  Year Frequency LogFreq TimeSq
##      <qtr> <int>   <int> <int>     <int>   <dbl>  <dbl>
##  1 2020 Q1     1       1  2020    397999    12.9      1
##  2 2020 Q2     2       2  2020    148382    11.9      4
##  3 2020 Q3     3       3  2020    249142    12.4      9
##  4 2020 Q4     4       4  2020    263221    12.5     16
##  5 2021 Q1     5       1  2021    250328    12.4     25
##  6 2021 Q2     6       2  2021    340789    12.7     36
##  7 2021 Q3     7       3  2021    360666    12.8     49
##  8 2021 Q4     8       4  2021    346912    12.8     64
##  9 2022 Q1     9       1  2022    314733    12.7     81
## 10 2022 Q2    10       2  2022    377309    12.8    100
```

```r
#
#   Plot Sales against Time
#
wampus_ts$Quarter <- as.factor(wampus_ts$Quarter)
wampus_ts %>% ggplot() +
  geom_line(aes(x=Time, y=Frequency)) +
  geom_point(aes(x=Time, y=Frequency, color=Quarter)) +
  scale_color_manual(values = c("black", "blue", "purple", "red")) +
  ggtitle("Frequency vs. Time") + xlab("Time") + ylab("Frequency")
```



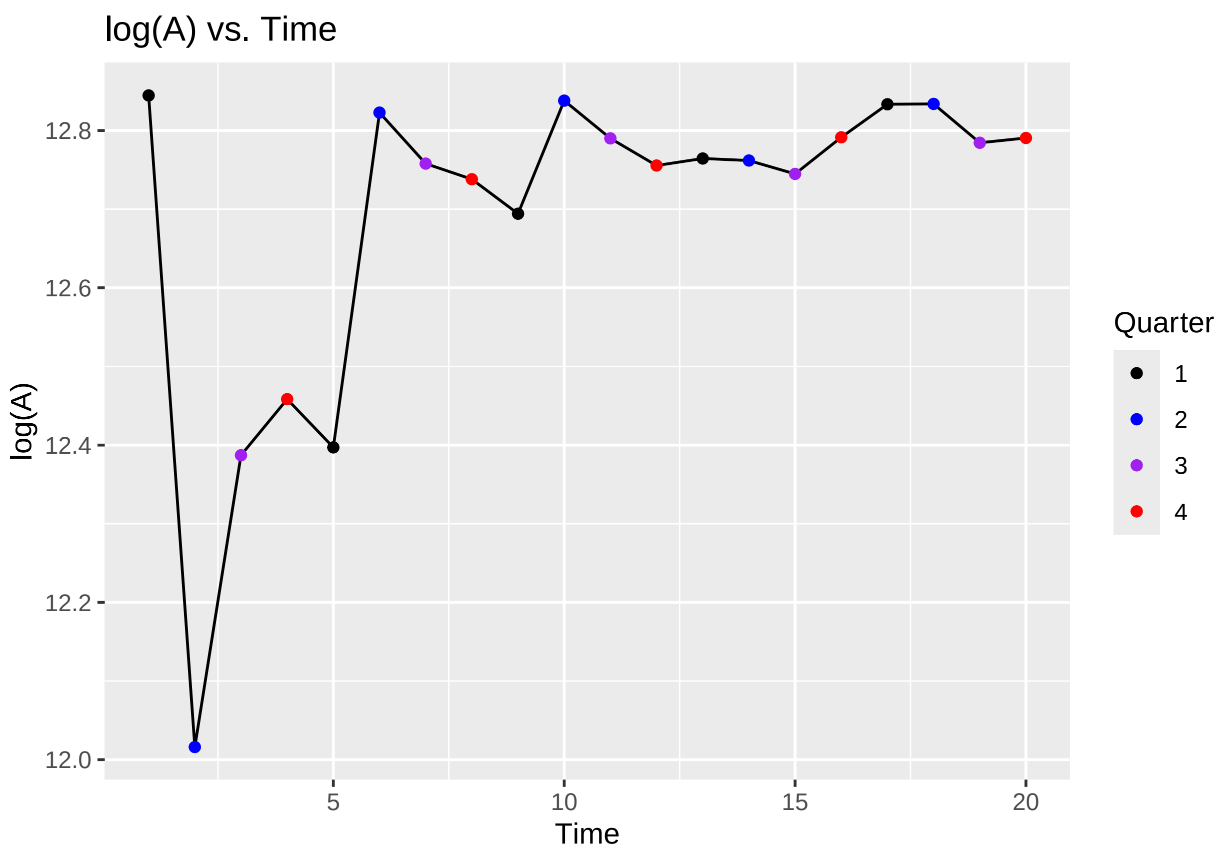

```r
#
#   Plot log(Frequency) against Time
#
wampus_ts %>% ggplot() +
  geom_line(aes(x=Time, y=LogFreq)) +
  geom_point(aes(x=Time, y=LogFreq, color=Quarter)) +
  scale_color_manual(values = c("black", "blue", "purple", "red")) +
  ggtitle("log(Freq) vs. Time") + xlab("Time") + ylab("log(Freq)")
```




```r
#
#   Decompose log(Sales) using STL decomposition
#
wampus_ts %>% model(STL(LogFreq ~ trend(window=7) + season(window=7))) %>%
  components() -> Log_Freq_components
wampus_ts$seasonal <- Log_Freq_components$season_year
#
#   Copy Log_Sales_time_series_components$season_adjust into Sales_table_ts
#
wampus_ts$logA <- Log_Freq_components$season_adjust
#
#   Seasonally adjust Sales values
#
wampus_ts$A <- exp(wampus_ts$logA)
head(wampus_ts)
```

```
## # A tsibble: 6 x 10 [1Q]
##       qtr  Time Quarter  Year Frequency LogFreq TimeSq seasonal  logA       A
##     <qtr> <int> <fct>   <int>     <int>   <dbl>  <dbl>    <dbl> <dbl>   <dbl>
## 1 2020 Q1     1 1        2020    397999    12.9      1   0.0496  12.8 378722.
## 2 2020 Q2     2 2        2020    148382    11.9      4  -0.109   12.0 165394.
## 3 2020 Q3     3 3        2020    249142    12.4      9   0.0387  12.4 239683.
## 4 2020 Q4     4 4        2020    263221    12.5     16   0.0224  12.5 257403.
## 5 2021 Q1     5 1        2021    250328    12.4     25   0.0334  12.4 242094.
## 6 2021 Q2     6 2        2021    340789    12.7     36  -0.0837  12.8 370556.
```

```r
#
#   Plot log(A) against Time
#
wampus_ts %>% ggplot() +
  geom_line(aes(x=Time, y=logA)) +
  geom_point(aes(x=Time, y=logA, color=Quarter)) +
  scale_color_manual(values = c("black", "blue", "purple", "red")) +
  ggtitle("log(A) vs. Time") + xlab("Time") + ylab("log(A)")
```




```r
#
#   Regress log(A) against Time and Time^2
#
reg_output <- wampus_ts %>% model(TSLM(logA ~ Time + TimeSq))
report(reg_output)
```

```
## Series: logA
## Model: TSLM
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.42932 -0.04379 -0.01552  0.03641  0.45160
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 12.336973   0.129423  95.323   <2e-16 ***
```

```
## Time          0.057744   0.028384   2.034   0.0578 .
## TimeSq       -0.001764   0.001313  -1.343   0.1968
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.174 on 17 degrees of freedom
## Multiple R-squared: 0.3977,  Adjusted R-squared: 0.3269
## F-statistic: 5.613 on 2 and 17 DF, p-value: 0.013437
```

```r
reg_output_tidy <- tidy(reg_output)
reg_output_tidy
```

```
## # A tibble: 3 x 6
##   .model                       term        estimate std.error statistic  p.value
##   <chr>                        <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 TSLM(logA ~ Time + TimeSq) (Intercept) 12.3       0.129       95.3  1.22e-24
## 2 TSLM(logA ~ Time + TimeSq) Time         0.0577    0.0284       2.03 5.78e- 2
## 3 TSLM(logA ~ Time + TimeSq) TimeSq      -0.00176   0.00131     -1.34 1.97e- 1
```

```r
reg_output_glance <- glance(reg_output)
reg_output_glance
```

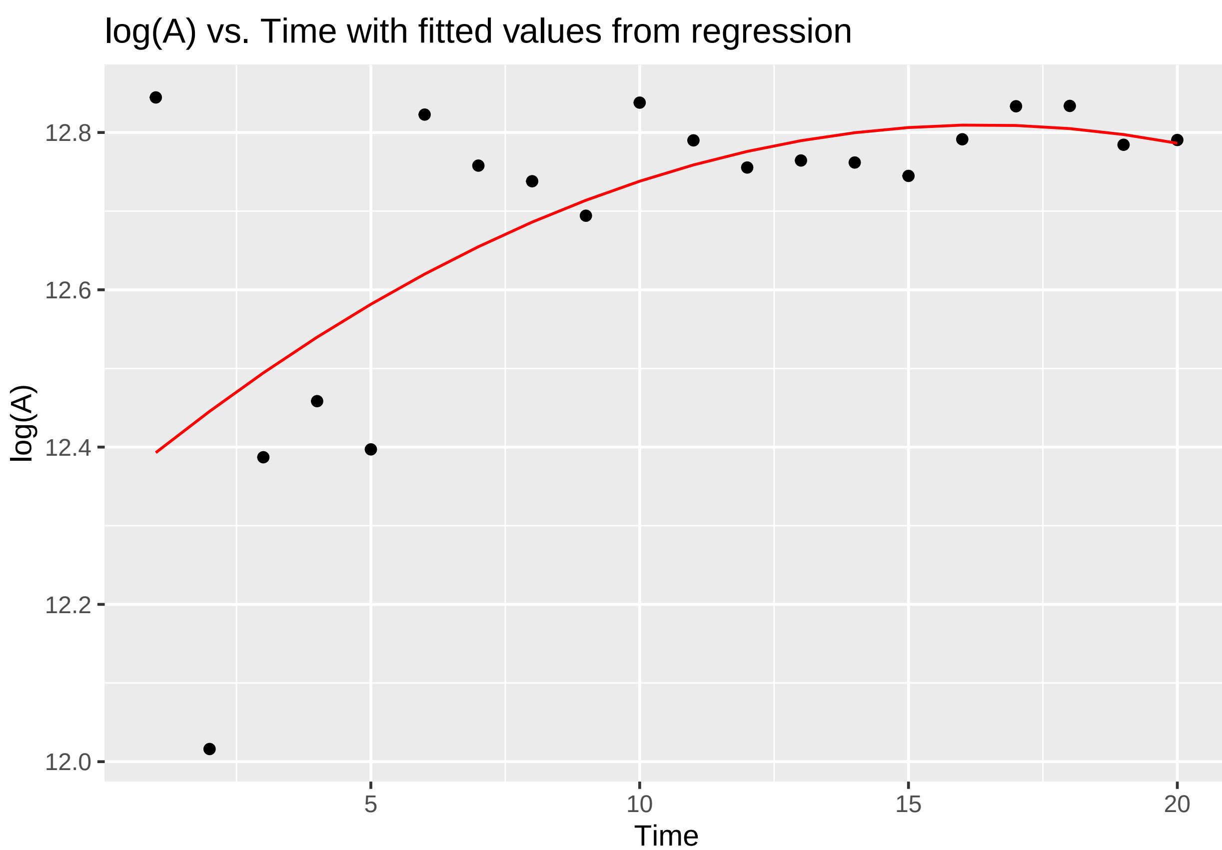
```
## # A tibble: 1 x 15
##   .model    r_squared adj_r_squared sigma2 statistic p_value    df log_lik   AIC
##   <chr>         <dbl>         <dbl>  <dbl>     <dbl>   <dbl> <int>   <dbl> <dbl>
## 1 TSLM(log~     0.398         0.327 0.0303      5.61  0.0134     3    8.23 -65.2
## # i 6 more variables: AICc <dbl>, BIC <dbl>, CV <dbl>, deviance <dbl>,
## #   df.residual <int>, rank <int>
```

```r
reg_output_augment <- augment(reg_output)
reg_output_augment
```

```
## # A tsibble: 20 x 6 [1Q]
## # Key:       .model [1]
##    .model                          qtr  logA .fitted   .resid   .innov
##    <chr>                         <qtr> <dbl>   <dbl>    <dbl>    <dbl>
##  1 TSLM(logA ~ Time + TimeSq) 2020 Q1  12.8    12.4  0.452    0.452
##  2 TSLM(logA ~ Time + TimeSq) 2020 Q2  12.0    12.4 -0.429   -0.429
##  3 TSLM(logA ~ Time + TimeSq) 2020 Q3  12.4    12.5 -0.107   -0.107
##  4 TSLM(logA ~ Time + TimeSq) 2020 Q4  12.5    12.5 -0.0813  -0.0813
##  5 TSLM(logA ~ Time + TimeSq) 2021 Q1  12.4    12.6 -0.185   -0.185
##  6 TSLM(logA ~ Time + TimeSq) 2021 Q2  12.8    12.6  0.203    0.203
##  7 TSLM(logA ~ Time + TimeSq) 2021 Q3  12.8    12.7  0.103    0.103
##  8 TSLM(logA ~ Time + TimeSq) 2021 Q4  12.7    12.7  0.0520   0.0520
##  9 TSLM(logA ~ Time + TimeSq) 2022 Q1  12.7    12.7 -0.0196  -0.0196
## 10 TSLM(logA ~ Time + TimeSq) 2022 Q2  12.8    12.7  0.0999   0.0999
## 11 TSLM(logA ~ Time + TimeSq) 2022 Q3  12.8    12.8  0.0312   0.0312
## 12 TSLM(logA ~ Time + TimeSq) 2022 Q4  12.8    12.8 -0.0205  -0.0205
## 13 TSLM(logA ~ Time + TimeSq) 2023 Q1  12.8    12.8 -0.0252  -0.0252
## 14 TSLM(logA ~ Time + TimeSq) 2023 Q2  12.8    12.8 -0.0379  -0.0379
## 15 TSLM(logA ~ Time + TimeSq) 2023 Q3  12.7    12.8 -0.0615  -0.0615
## 16 TSLM(logA ~ Time + TimeSq) 2023 Q4  12.8    12.8 -0.0180  -0.0180
```

```
## 17 TSLM(logA ~ Time + TimeSq) 2024 Q1  12.8    12.8  0.0244   0.0244
## 18 TSLM(logA ~ Time + TimeSq) 2024 Q2  12.8    12.8  0.0289   0.0289
## 19 TSLM(logA ~ Time + TimeSq) 2024 Q3  12.8    12.8 -0.0130  -0.0130
## 20 TSLM(logA ~ Time + TimeSq) 2024 Q4  12.8    12.8  0.00410  0.00410
```

```
alpha <- reg_output_tidy$estimate[1]
beta1 <- reg_output_tidy$estimate[2]
beta2 <- reg_output_tidy$estimate[3]
fitted_values_logA <- alpha + beta1*wampus_ts$Time + beta2*wampus_ts$TimeSq
#
#   Plot log(A) against Time with fitted values from regression
#
wampus_ts %>% ggplot() +
  geom_point(aes(x=Time, y=logA)) +
  geom_line(aes(x=Time, y=fitted_values_logA), color="Red") +
  ggtitle("log(A) vs. Time with fitted values from regression") + xlab("Time") + ylab("log(A)")
```




```
#
#   Construct Time, TimeSq and Seasonal and seasonal_forecast vectors with extra four rows for forecast
#
Time <- c(wampus_ts$Time,(rows+1):(rows+4))
Time_sq <- Time^2
seasonal <- c(wampus_ts$seasonal, NA, NA, NA, NA)
seasonal_forecast <- c(NA, NA, NA, NA, wampus_ts$seasonal)
wampus_extended_ts <- tsibble(qtr = yearquarter(yearquarter_input) + 0:(rows+3),
```

```
                                    Time=Time, Time_sq=Time_sq, seasonal = seasonal,
                                    seasonal_forecast = seasonal_forecast,
                                    index = qtr)
head(wampus_extended_ts)
```

```
## # A tsibble: 6 x 5 [1Q]
##       qtr  Time Time_sq seasonal seasonal_forecast
##     <qtr> <int>   <dbl>    <dbl>             <dbl>
## 1 2020 Q1     1       1   0.0496           NA
## 2 2020 Q2     2       4  -0.109            NA
## 3 2020 Q3     3       9   0.0387           NA
## 4 2020 Q4     4      16   0.0224           NA
## 5 2021 Q1     5      25   0.0334            0.0496
## 6 2021 Q2     6      36  -0.0837           -0.109
```

```
wampus_extended_ts[(rows-3):(rows+4),]
```

```
## # A tsibble: 8 x 5 [1Q]
##       qtr  Time Time_sq seasonal seasonal_forecast
##     <qtr> <int>   <dbl>    <dbl>             <dbl>
## 1 2024 Q1    17     289 -0.0529           -0.0491
## 2 2024 Q2    18     324  0.0365            0.0281
## 3 2024 Q3    19     361  0.0188            0.0221
## 4 2024 Q4    20     400 -0.00330          -0.00176
## 5 2025 Q1    21     441 NA                -0.0529
## 6 2025 Q2    22     484 NA                 0.0365
## 7 2025 Q3    23     529 NA                 0.0188
## 8 2025 Q4    24     576 NA                -0.00330
```

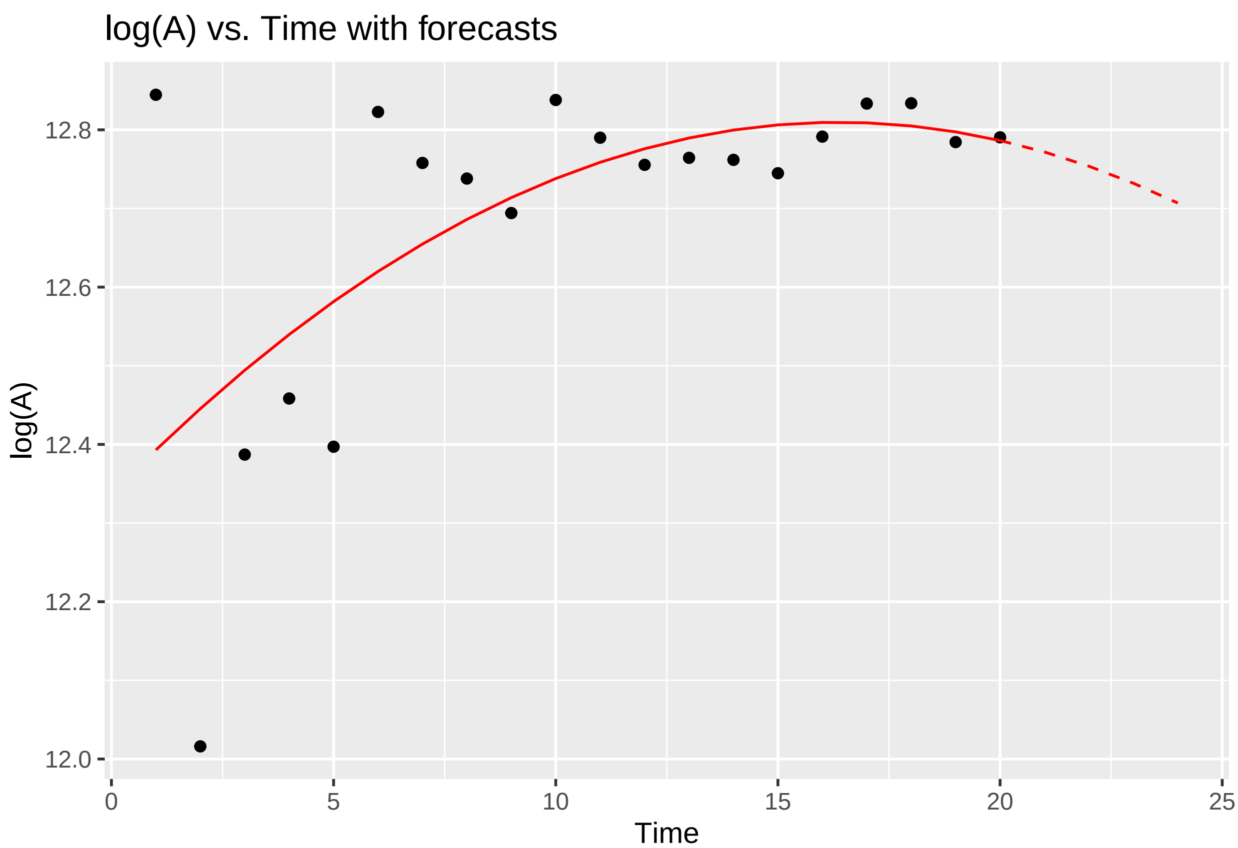
```
#
#   Compute in-sample and out-of-sample forecasts
#
wampus_extended_ts$forecast_logA <- alpha + beta1*wampus_extended_ts$Time +
  beta2*wampus_extended_ts$Time_sq
wampus_extended_ts$forecast_logFreq <- wampus_extended_ts$forecast_logA +
  wampus_extended_ts$seasonal_forecast
wampus_extended_ts$forecast_Frequency <- exp(wampus_extended_ts$forecast_logFreq)
head(wampus_extended_ts)
```

```
## # A tsibble: 6 x 8 [1Q]
##       qtr  Time Time_sq seasonal seasonal_forecast forecast_logA
##     <qtr> <int>   <dbl>    <dbl>             <dbl>         <dbl>
## 1 2020 Q1     1       1   0.0496           NA               12.4
## 2 2020 Q2     2       4  -0.109            NA               12.4
## 3 2020 Q3     3       9   0.0387           NA               12.5
## 4 2020 Q4     4      16   0.0224           NA               12.5
## 5 2021 Q1     5      25   0.0334            0.0496          12.6
## 6 2021 Q2     6      36  -0.0837           -0.109           12.6
## # i 2 more variables: forecast_logFreq <dbl>, forecast_Frequency <dbl>
```

```r
wampus_extended_ts[(rows-3):(rows+4),]
```

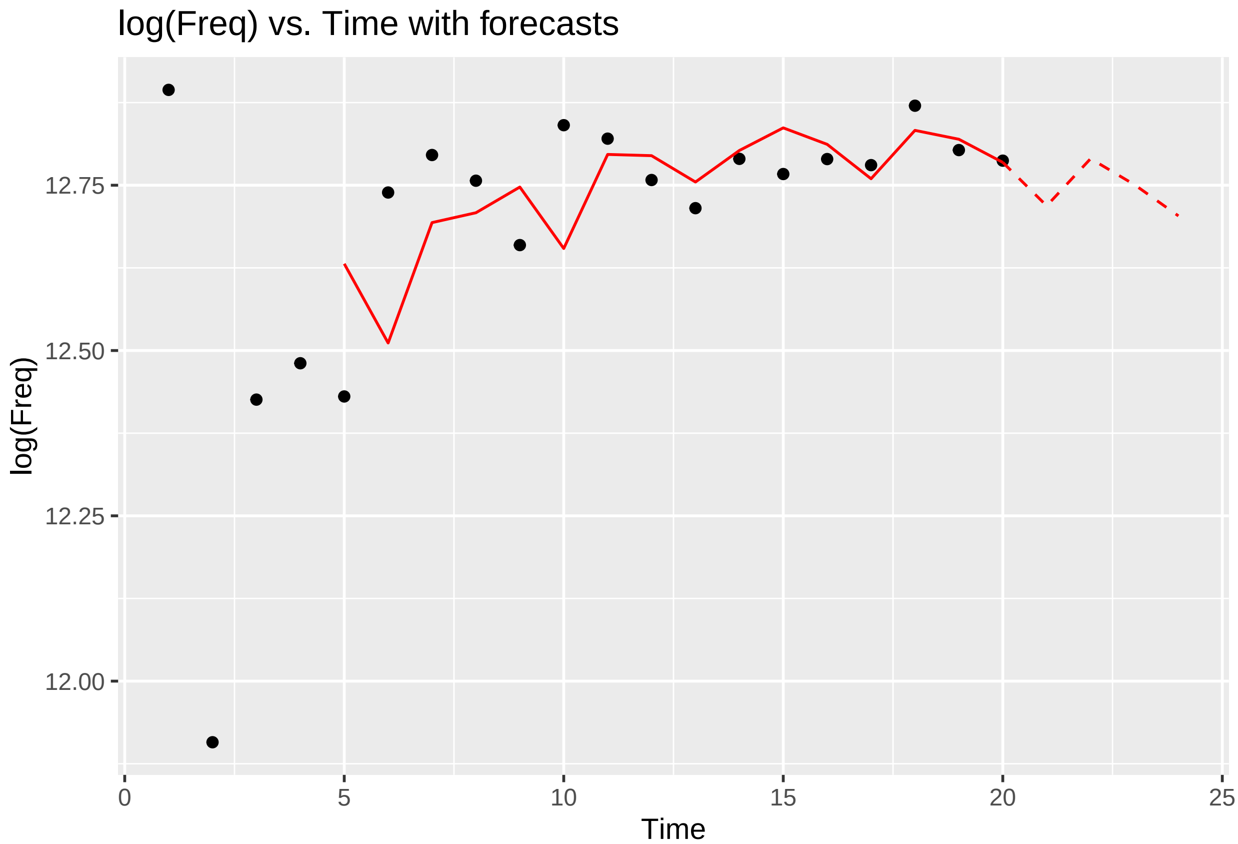
```
## # A tsibble: 8 x 8 [1Q]
##       qtr  Time Time_sq seasonal seasonal_forecast forecast_logA
##     <qtr> <int>   <dbl>    <dbl>             <dbl>         <dbl>
## 1 2024 Q1    17     289 -0.0529            -0.0491          12.8
## 2 2024 Q2    18     324  0.0365             0.0281          12.8
## 3 2024 Q3    19     361  0.0188             0.0221          12.8
## 4 2024 Q4    20     400 -0.00330           -0.00176         12.8
## 5 2025 Q1    21     441 NA                 -0.0529          12.8
## 6 2025 Q2    22     484 NA                  0.0365          12.8
## 7 2025 Q3    23     529 NA                  0.0188          12.7
## 8 2025 Q4    24     576 NA                 -0.00330         12.7
## # i 2 more variables: forecast_logFreq <dbl>, forecast_Frequency <dbl>
```

```r
#
#   Plot log(A) against Time with forecasts - Cannot use pipeline notation because Sales_table_ts and
#     Sales_table_extended_ts are different length
#
ggplot() +
  geom_point(aes(x=wampus_ts$Time, y=wampus_ts$logA), color="Black") +
  geom_line(aes(x=wampus_extended_ts$Time[1:rows], y=wampus_extended_ts$forecast_logA[1:rows]),
            linetype=1, color="Red") +
  geom_line(aes(x=wampus_extended_ts$Time[rows:(rows+4)], y=wampus_extended_ts$forecast_logA[rows:(rows
            linetype=2, color="Red") +
  ggtitle("log(A) vs. Time with forecasts") + xlab("Time") + ylab("log(A)")
```

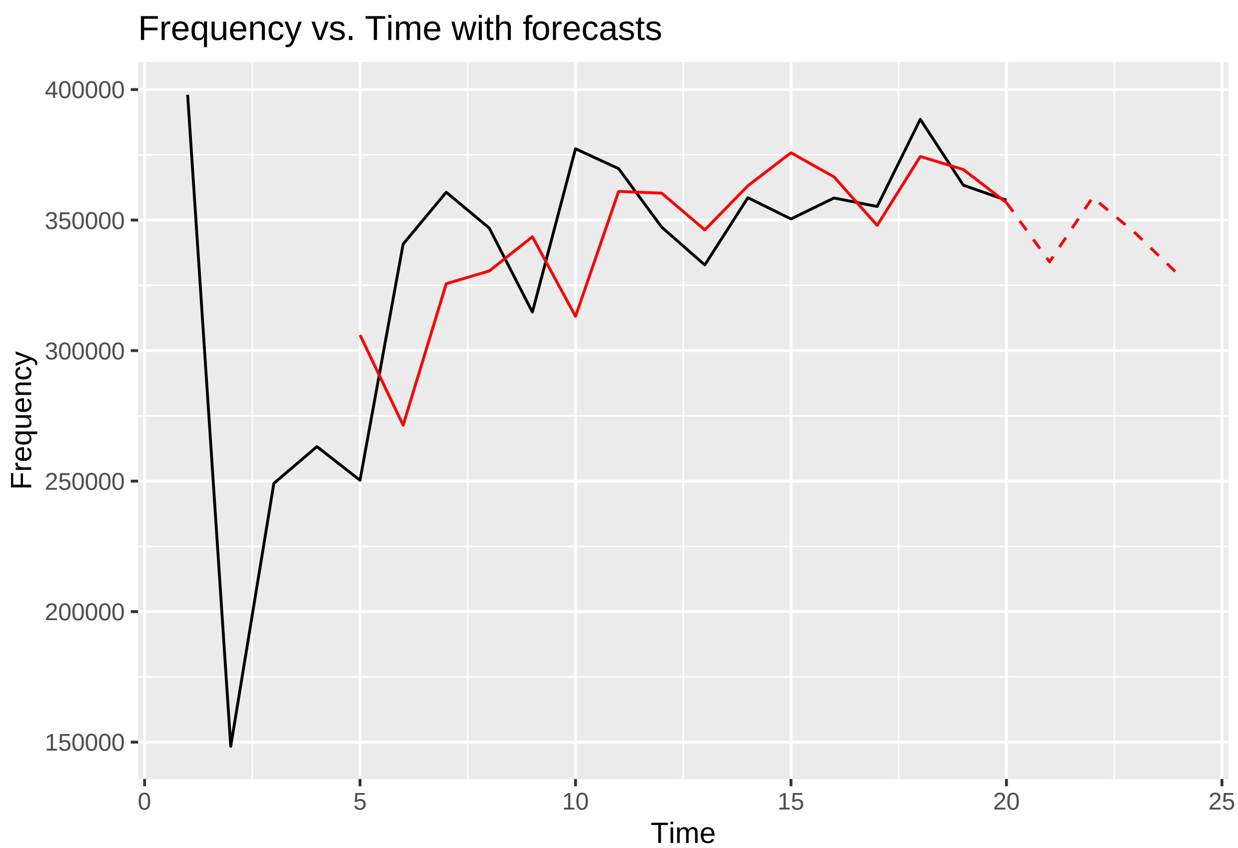



```
#
#   Plot log(Sales) against Time with forecasts
#

ggplot() +
  geom_point(aes(x=wampus_ts$Time, y=wampus_ts$LogFreq), color="Black") +
  geom_line(aes(x=wampus_extended_ts$Time[5:rows], y=wampus_extended_ts$forecast_logFreq[5:rows]),
            linetype=1, color="Red") +
  geom_line(aes(x=wampus_extended_ts$Time[rows:(rows+4)], y=wampus_extended_ts$forecast_logFreq[rows:(r
            linetype=2, color="Red") +
  ggtitle("log(Freq) vs. Time with forecasts") + xlab("Time") + ylab("log(Freq)")
```

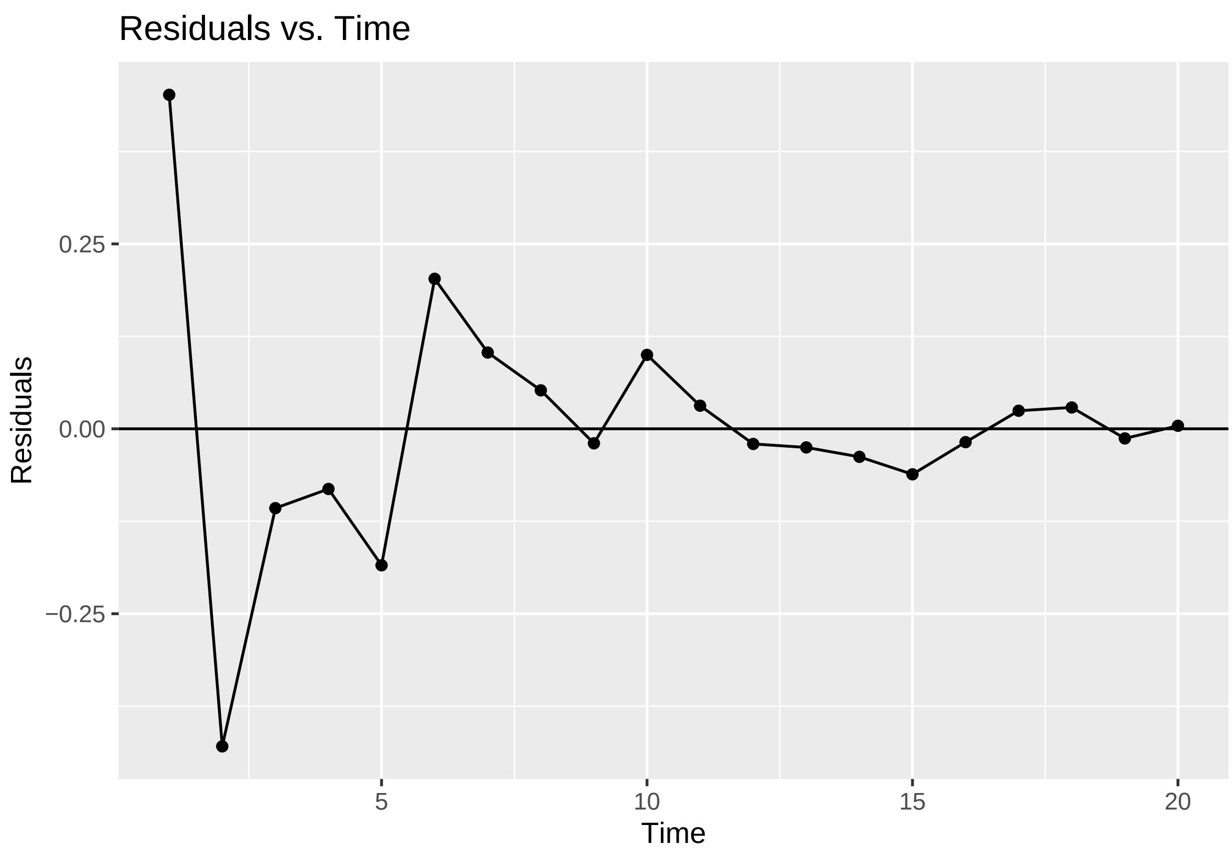



```r
#
#   Plot Sales against Time with forecasts
#
ggplot() +
  geom_line(aes(x=wampus_ts$Time, y=wampus_ts$Frequency), linetype=1) +
  geom_line(aes(x=wampus_extended_ts$Time[5:rows], y=wampus_extended_ts$forecast_Frequency[5:rows]),
            linetype=1, color="Red") +
  geom_line(aes(x=wampus_extended_ts$Time[rows:(rows+4)], y=wampus_extended_ts$forecast_Frequency[rows:
            linetype=2, color="Red") +
  ggtitle("Frequency vs. Time with forecasts") + xlab("Time") + ylab("Frequency")
```




```
#
#   Plot Residuals [from the regression lm(logA ~ Time + TimeSq)] vs. Time
#

ggplot() +
  geom_line(aes(x=wampus_ts$Time, y=reg_output_augment$.resid)) +
  geom_point(aes(x=wampus_ts$Time, y=reg_output_augment$.resid)) +
  geom_hline(yintercept=0) +
  ggtitle("Residuals vs. Time") + xlab("Time") + ylab("Residuals")
```




```
#
#   Compute autocorrelation coefficients using regression
#
residuals <- reg_output_augment$.resid
residuals_lag1 <- lag(reg_output_augment$.resid, n = 1)
residuals_lag2 <- lag(reg_output_augment$.resid, n = 2)
residuals_lag3 <- lag(reg_output_augment$.resid, n = 3)
Residuals_ts <- tsibble(qtr = yearquarter(yearquarter_input) + 0:(rows-1),
                        residuals = residuals,
                        residuals_lag1 = residuals_lag1,
                        residuals_lag2 = residuals_lag2,
                        residuals_lag3 = residuals_lag3,
                        index = qtr)
Residuals_ts
```

```
## # A tsibble: 20 x 5 [1Q]
##        qtr residuals residuals_lag1 residuals_lag2 residuals_lag3
##      <qtr>     <dbl>          <dbl>          <dbl>          <dbl>
##  1 2020 Q1    0.452         NA             NA             NA
##  2 2020 Q2   -0.429          0.452         NA             NA
##  3 2020 Q3   -0.107         -0.429          0.452         NA
##  4 2020 Q4   -0.0813        -0.107         -0.429          0.452
##  5 2021 Q1   -0.185         -0.0813        -0.107         -0.429
##  6 2021 Q2    0.203         -0.185         -0.0813        -0.107
##  7 2021 Q3    0.103          0.203         -0.185         -0.0813
##  8 2021 Q4    0.0520         0.103          0.203         -0.185
```

```
##   9 2022 Q1  -0.0196          0.0520         0.103          0.203
## 10 2022 Q2   0.0999         -0.0196         0.0520         0.103
## 11 2022 Q3   0.0312          0.0999        -0.0196         0.0520
## 12 2022 Q4  -0.0205          0.0312         0.0999        -0.0196
## 13 2023 Q1  -0.0252         -0.0205         0.0312         0.0999
## 14 2023 Q2  -0.0379         -0.0252        -0.0205         0.0312
## 15 2023 Q3  -0.0615         -0.0379        -0.0252        -0.0205
## 16 2023 Q4  -0.0180         -0.0615        -0.0379        -0.0252
## 17 2024 Q1   0.0244         -0.0180        -0.0615        -0.0379
## 18 2024 Q2   0.0289          0.0244        -0.0180        -0.0615
## 19 2024 Q3  -0.0130          0.0289         0.0244        -0.0180
## 20 2024 Q4   0.00410        -0.0130         0.0289         0.0244
```

```r
reg_lag1 <- Residuals_ts %>% model(TSLM(residuals ~ residuals_lag1))
report(reg_lag1)
```

```
## Series: residuals
## Model: TSLM
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.29040 -0.03391  0.01746  0.06971  0.17962
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -0.02382    0.02872   -0.83    0.418
## residuals_lag1 -0.25485    0.17452   -1.46    0.162
##
## Residual standard error: 0.1252 on 17 degrees of freedom
## Multiple R-squared: 0.1115,  Adjusted R-squared: 0.05919
## F-statistic: 2.132 on 1 and 17 DF, p-value: 0.16244
```

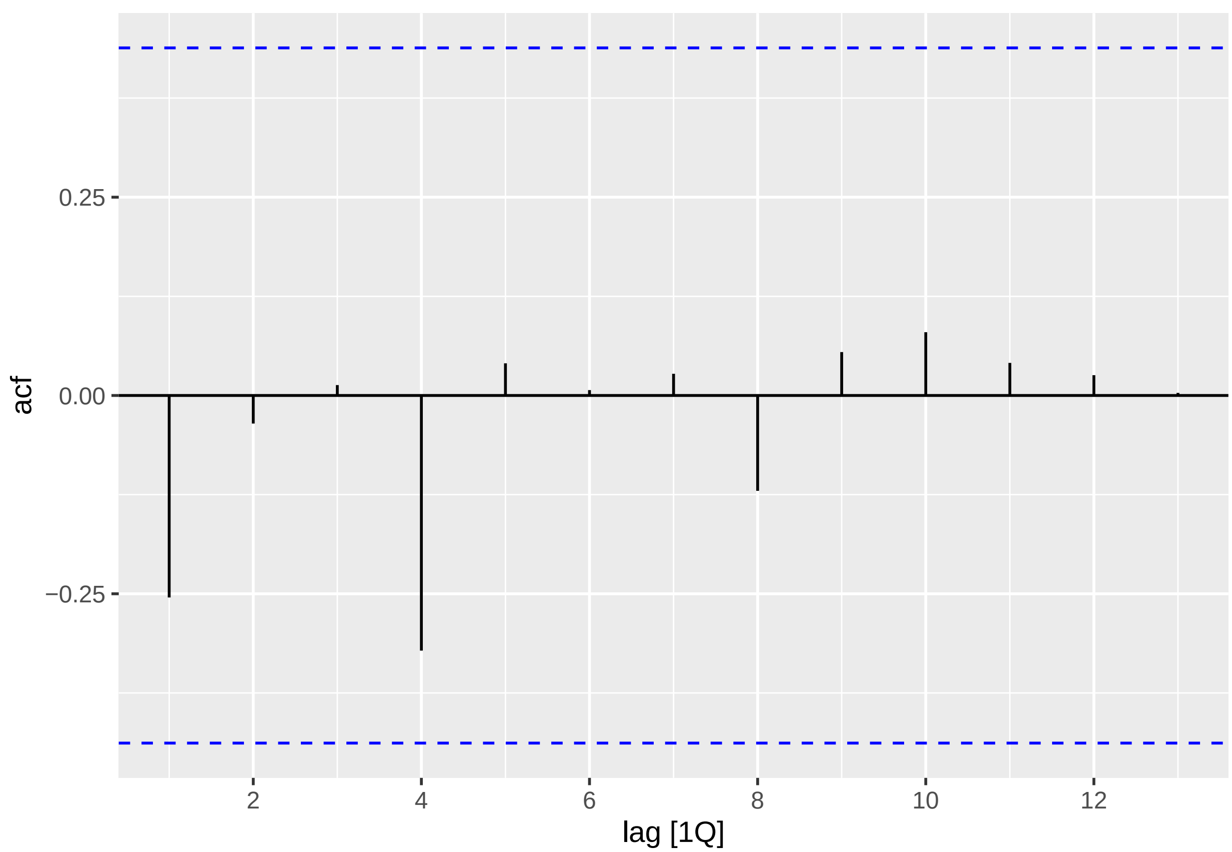
```r
reg_lag2 <- Residuals_ts %>% model(TSLM(residuals ~ residuals_lag2))
report(reg_lag2)
```

```
## Series: residuals
## Model: TSLM
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.18709 -0.03376 -0.01284  0.03117  0.20116
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -0.001221   0.020877  -0.058    0.954
## residuals_lag2 -0.035361   0.123512  -0.286    0.778
##
## Residual standard error: 0.08857 on 16 degrees of freedom
## Multiple R-squared: 0.005097,    Adjusted R-squared: -0.05708
## F-statistic: 0.08196 on 1 and 16 DF, p-value: 0.77833
```

```r
reg_lag3 <- Residuals_ts %>% model(TSLM(residuals ~ residuals_lag3))
report(reg_lag3)
```

```
## Series: residuals
## Model: TSLM
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.18380 -0.03152 -0.01779  0.02551  0.19923
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.005014   0.021160   0.237    0.816
## residuals_lag3 0.013344   0.121760   0.110    0.914
##
## Residual standard error: 0.08724 on 15 degrees of freedom
## Multiple R-squared: 0.0008001,   Adjusted R-squared: -0.06581
## F-statistic: 0.01201 on 1 and 15 DF, p-value: 0.91418
```

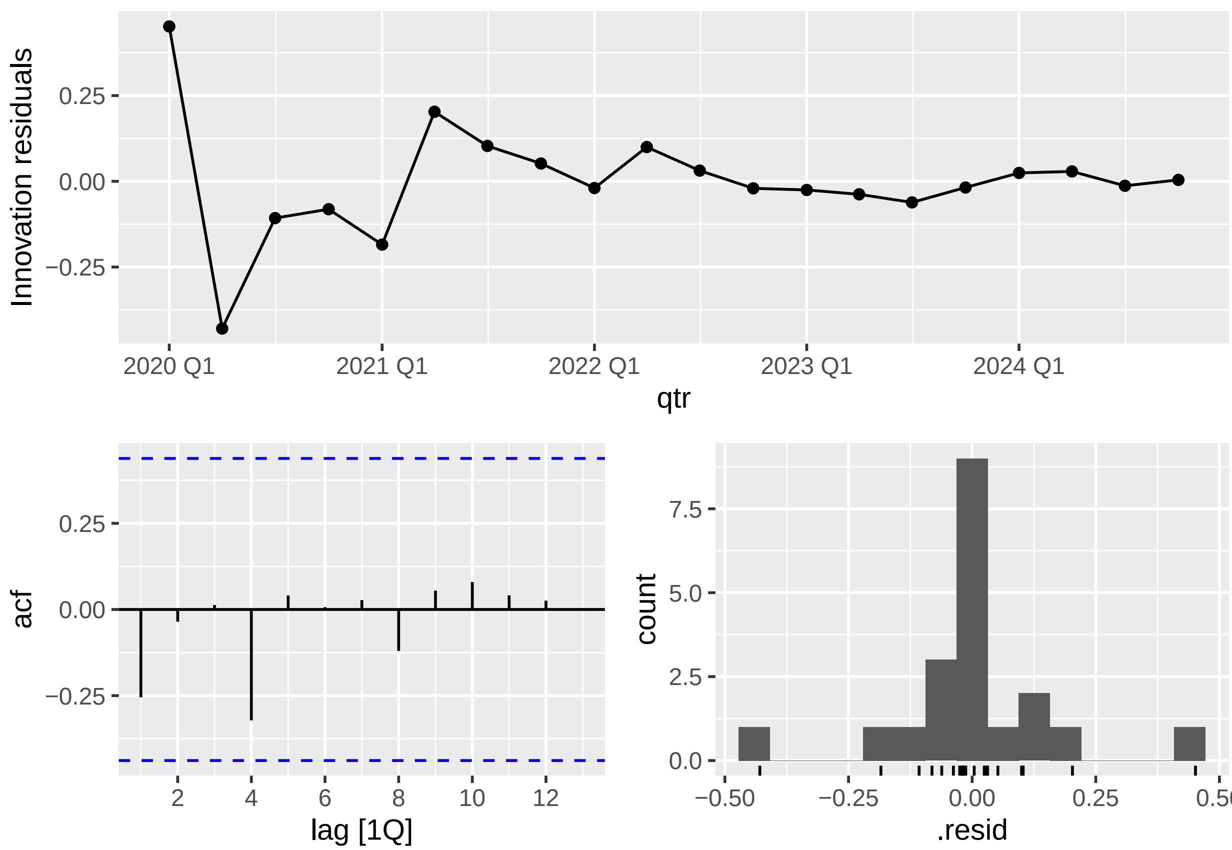
```r
#
#   Plot and print autocorrelation function of the residuals
#
result_ACF <- reg_output_augment %>% ACF(.resid)
result_ACF %>% autoplot()
```



```r
print(result_ACF, n=10)
```

```
## # A tsibble: 13 x 3 [1Q]
## # Key:       .model [1]
##    .model                                lag      acf
##    <chr>                            <cf_lag>    <dbl>
##  1 TSLM(logA ~ Time + TimeSq)             1Q -0.255
##  2 TSLM(logA ~ Time + TimeSq)             2Q -0.0354
##  3 TSLM(logA ~ Time + TimeSq)             3Q  0.0131
##  4 TSLM(logA ~ Time + TimeSq)             4Q -0.322
##  5 TSLM(logA ~ Time + TimeSq)             5Q  0.0406
##  6 TSLM(logA ~ Time + TimeSq)             6Q  0.00680
##  7 TSLM(logA ~ Time + TimeSq)             7Q  0.0274
##  8 TSLM(logA ~ Time + TimeSq)             8Q -0.120
##  9 TSLM(logA ~ Time + TimeSq)             9Q  0.0547
## 10 TSLM(logA ~ Time + TimeSq)            10Q  0.0798
## # i 3 more rows
```

```r
#
#   Check assumptions
#
reg_output %>% gg_tsresiduals()
```



```r
#
#   Compute Anderson-Darling test for the residuals to test the normality assumption
#
ad.test(reg_output_augment$.resid)
```

```
##
##  Anderson-Darling normality test
##
## data:  reg_output_augment$.resid
## A = 1.1019, p-value = 0.00528
```

```r
# 53->22, 56->25, 55->24, 0->0, 49->18, 1->1, 52->21, 5->5
```