Assignment 8: CME Monthly Seat Prices

Joshua Goldberg

June, 12 2019

Data

```
all_divisions <- readRDS("all_divisions_clean.rds") %>%
  mutate(division = tolower(division))

all_divisions_train <-
  all_divisions %>% filter_index("2001 Jan" ~ "2012 Dec")

all_divisions_test <- all_divisions %>%
  anti_join(all_divisions_train, c("division", "year_month"))

contracts_volume <-
  read_csv("Contracts_Volume.csv") %>% clean_names()

contracts_classification <-
  read_csv("Contracts_Classification.csv") %>%
  clean_names() %>%
  mutate(division = tolower(division))
```

Tasks

Prep data

1. Commodities are traded on the Floor (crazy people screaming at each other in the pits like you have seen in movies) and electronically. The Volume data set has Total volume and Electronic volume.

```
contracts_volume <- contracts_volume %>%
  mutate(floor_volume = total_volume - electronic_volume,
         date = mdy(date))
```

1. Sort out of the volume data, those commodities that are relevant for the particular badge (CME, IMM, IOM). Keep in mind that the CME can trade EVERYTHING, not just what the list says.
2. Aggregate the data for each Commodity Indicator for each month. Don't worry about futures / options, just add them all up.
3. Create a table that looks like this: Date Elec.Vol Tot.Vol Flo.Vol 01/01/2001 4,769,234 31,746,144 26,976,910

```
contract_volume_divisions <- contracts_volume %>%
  inner_join(contracts_classification,
             by = c("commodity_indicator" = "commodity_code"))
```

```
filter_volume <-
  function(.data,
           .division,
```

```r
           .date = list(floor = "2001-01-01", ceiling = "2012-12-01"),
           .date_format = "%Y-%m-%d",
           test_set = FALSE) {
    dates <- map(.date, ~ lubridate::as_date(.x, .date_format))
    if (.division %in% c("iom", "imm")) {
      .data <- .data %>% filter(division == .division)
    }

    if (.division == "cme") {
      .data <- .data
    }

    if (test_set) {
      .data <- .data %>% filter(date >= dates)
    } else {
      .data <- .data %>% filter(between(date, dates$floor, dates$ceiling))
    }
    .data
  }

aggregate_volume <- function(.data) {
  .data %>%
    group_by(date) %>%
    summarize(
      total_volume = sum(total_volume) %>% as.double(),
      electronic_volume = sum(electronic_volume) %>% as.double(),
      floor_volume = sum(floor_volume) %>% as.double()
    ) %>%
    mutate(year_month = yearmonth(date)) %>%
    select(year_month, everything(), -date)
}

transform_by_division <- function(.division, .data, ...) {
  .data %>%
    filter_volume(.division, ...) %>%
    aggregate_volume()
}

divisions <-
  c(
    cme = "cme",
    imm = "imm",
    iom = "iom"
  )

train_volumes <-
  map(divisions, transform_by_division, contract_volume_divisions)

test_volumes <-
  map(
    divisions,
    ~ transform_by_division(
      .division = .x,
```

```
      .data = contract_volume_divisions,
      .date = "2013-01-01",
      test_set = TRUE
    )
  )

train_volumes$cme
```

```
## # A tibble: 144 x 4
##    year_month total_volume electronic_volume floor_volume
##         <mth>        <dbl>             <dbl>        <dbl>
##  1   2001 Jan     55810609           5385520     50425089
##  2   2001 Feb     45437337           5342048     40095289
##  3   2001 Mar     57351206           7354218     49996988
##  4   2001 Apr     53638046           7156921     46481125
##  5   2001 May     56644028           7278038     49365990
##  6   2001 Jun     55914286           7294761     48619525
##  7   2001 Jul     49984671           6950118     43034553
##  8   2001 Aug     58443942           7661934     50782008
##  9   2001 Sep     69137252           7517115     61620137
## 10   2001 Oct     62032707          11150303     50882404
## # ... with 134 more rows
```

```
test_volumes$cme
```

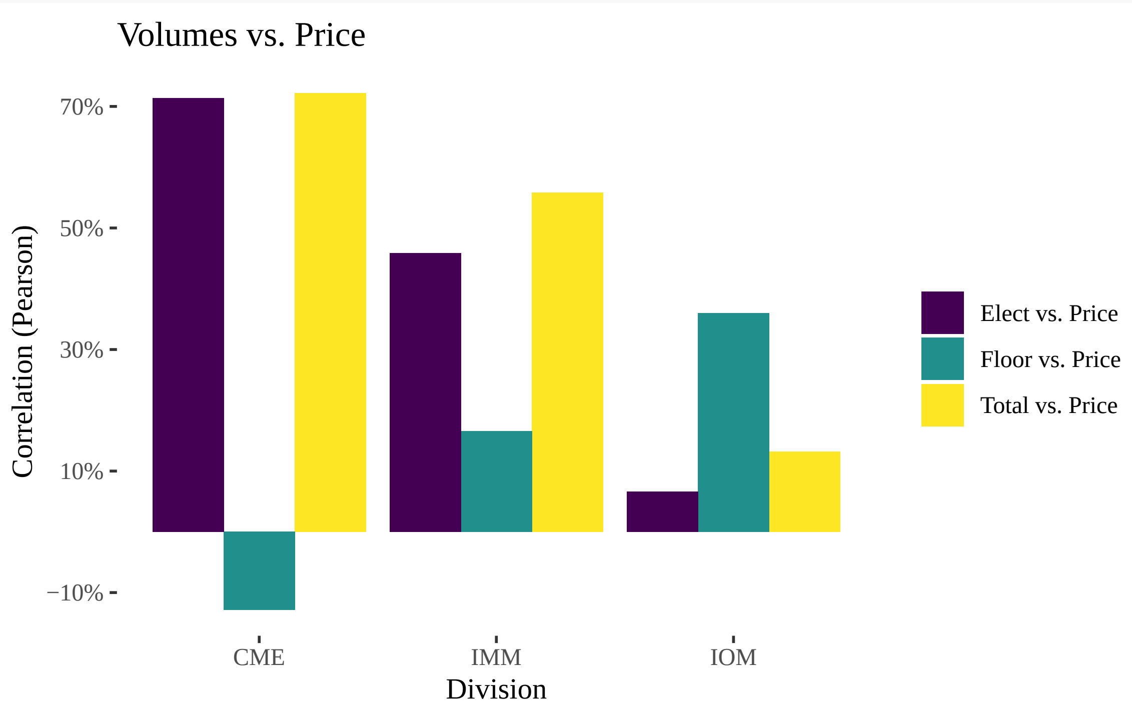
```
## # A tibble: 12 x 4
##    year_month total_volume electronic_volume floor_volume
##         <mth>        <dbl>             <dbl>        <dbl>
##  1   2013 Jan    177033977         163707422     13326555
##  2   2013 Feb    165071221         153931463     11139758
##  3   2013 Mar    190175803         175534139     14641664
##  4   2013 Apr    169265756         156599041     12666715
##  5   2013 May    224494774         206674615     17820159
##  6   2013 Jun    285292038         256753377     28538661
##  7   2013 Jul    179154982         161073572     18081410
##  8   2013 Aug    181498081         160171217     21326864
##  9   2013 Sep    216339389         190081253     26258136
## 10   2013 Oct    197540594         176274440     21266154
## 11   2013 Nov    164659709         149495942     15163767
## 12   2013 Dec    182878564         163411793     19466771
```

Exploratory data analysis

Your task is to use the trading volume information to forecast the CME monthly seat price for 2013. It is recommended to do exploratory data analysis to find initial data relationships such as correlations. For example, the total trade volume for all CME products might be a good predictor for CME seat class, but not for the others. You may have to choose and select which commodities have influence on the IMM and IOM seat prices.

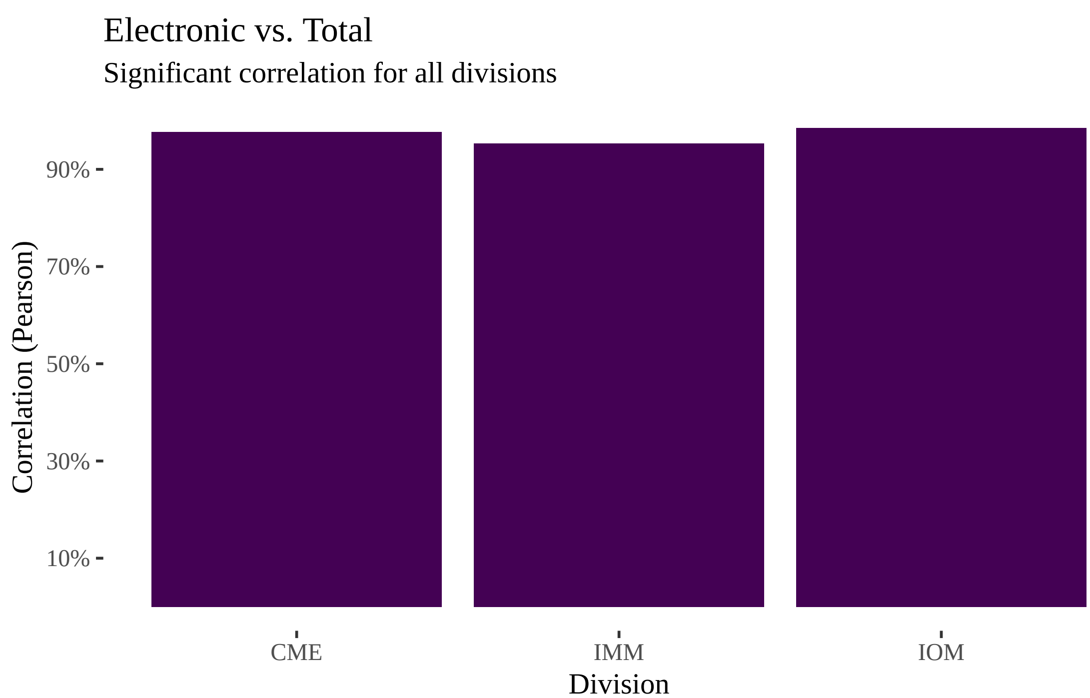
```
explore_price_volume <- train_volumes %>%
  bind_rows(.id = "division") %>%
  left_join(all_divisions_train, c("division", "year_month")) %>%
  group_by(division) %>%
```

```r
  mutate(
    floor_vs_price = cor(floor_volume, price),
    elect_vs_price = cor(electronic_volume, price),
    total_vs_price = cor(total_volume, price),
    elect_vs_total = cor(electronic_volume, total_volume),
  ) %>%
  gather(key = corr_group, value = corr, -c(1:7))

explore_price_volume %>%
  filter(corr_group != "elect_vs_total") %>%
  ggplot(aes(division, corr, fill = corr_group)) +
  geom_col(position = "dodge") +
  scale_fill_viridis_d(
    name = NULL,
    labels = c("Elect vs. Price",
               "Floor vs. Price",
               "Total vs. Price")
  ) +
  scale_x_discrete(labels = toupper) +
  scale_y_continuous(
    breaks = seq(
      plyr::round_any(min(explore_price_volume$corr), .10, ceiling),
      max(explore_price_volume$corr),
      .20
    ),
    labels = scales::percent_format(accuracy = 1)
  ) +
  labs(title = "Volumes vs. Price",
       x = "Division",
       y = "Correlation (Pearson)")
```



CME shows the strongest relationship across the aggregate volumes, with electronic and total representing the highest at 0.7140589 and 0.7215798, respectively. We will use `total_volume` as a predictor for CME/IMM since electronic and total are strongly correlated, as see in the plot below. IOM's highest correlation with `price` is `floor_volume`.

```
explore_price_volume %>%
  filter(corr_group == "elect_vs_total") %>%
  ggplot(aes(division, corr, fill = corr_group)) +
  geom_col(position = "dodge") +
  scale_fill_viridis_d() +
  scale_x_discrete(labels = toupper) +
  scale_y_continuous(
    breaks = seq(
      plyr::round_any(min(explore_price_volume$corr), .10, ceiling),
      max(explore_price_volume$corr),
      .20
    ),
    labels = scales::percent_format(accuracy = 1)
  ) +
  labs(
    title = "Electronic vs. Total",
    subtitle = "Significant correlation for all divisions",
    x = "Division",
    y = "Correlation (Pearson)"
  ) +
  theme(legend.position = "none")
```



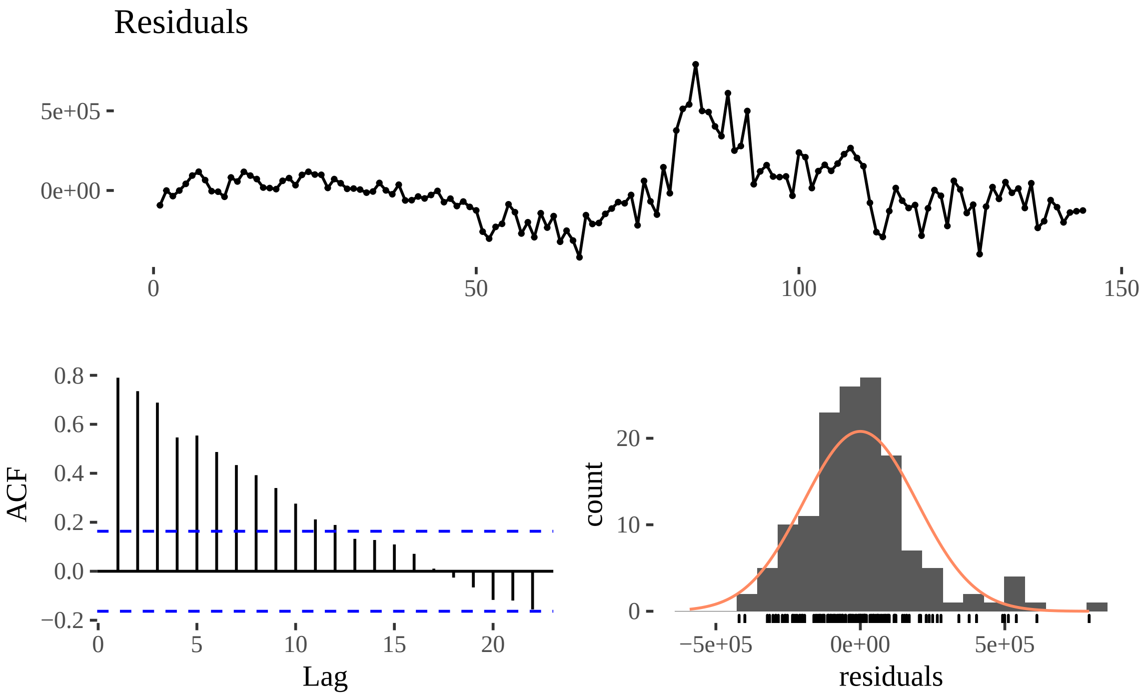
Modeling

Linear regression

Linear regression (seat price is independent, volume(s) dependent).

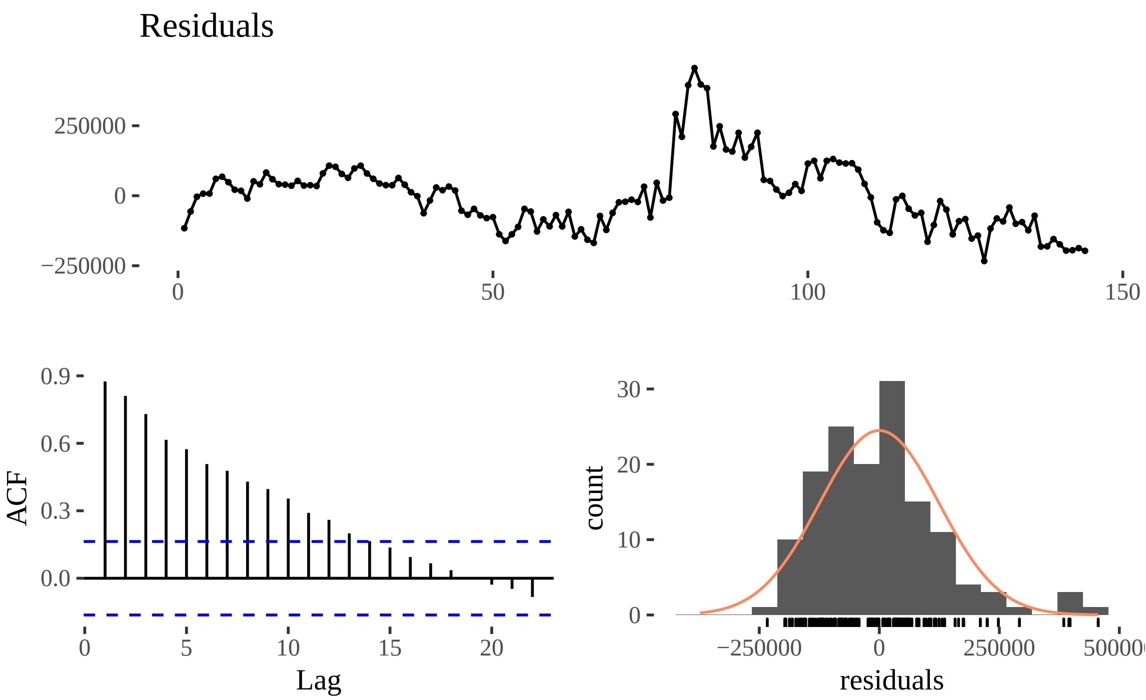
```
model_price_volume <-
  explore_price_volume %>%
  select(-contains("vs"),-contains("corr")) %>%
  distinct() %>%
  split(.$division) %>%
  map(
    ~ .x %>%
      ungroup %>%
      mutate(year_month = yearmonth(year_month)) %>%
      as_tsibble(key = division, index = year_month)
  )

lm_formulas <- c(
  cme = price ~ total_volume,
  imm = price ~ total_volume,
  iom = price ~ floor_volume
)

lm_models <-
  map2(lm_formulas, model_price_volume,
       function(.formula, .data) {
         lm(.formula, data = .data)
       })
```

```
checkresiduals(lm_models$cme)
```



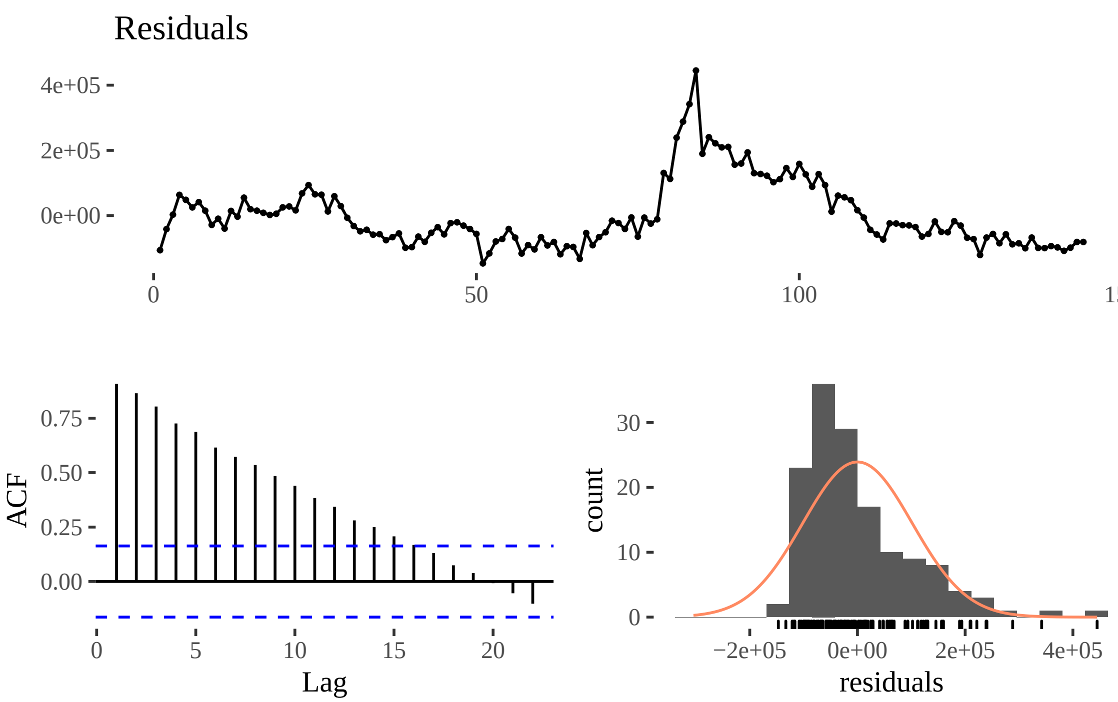
```
##
##  Breusch-Godfrey test for serial correlation of order up to 10
##
## data:  Residuals
## LM test = 103.1, df = 10, p-value < 2.2e-16
```

```
checkresiduals(lm_models$imm)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 10
##
## data:  Residuals
## LM test = 118.07, df = 10, p-value < 2.2e-16
```

```
checkresiduals(lm_models$iom)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 10
##
## data:  Residuals
## LM test = 123.17, df = 10, p-value < 2.2e-16
```

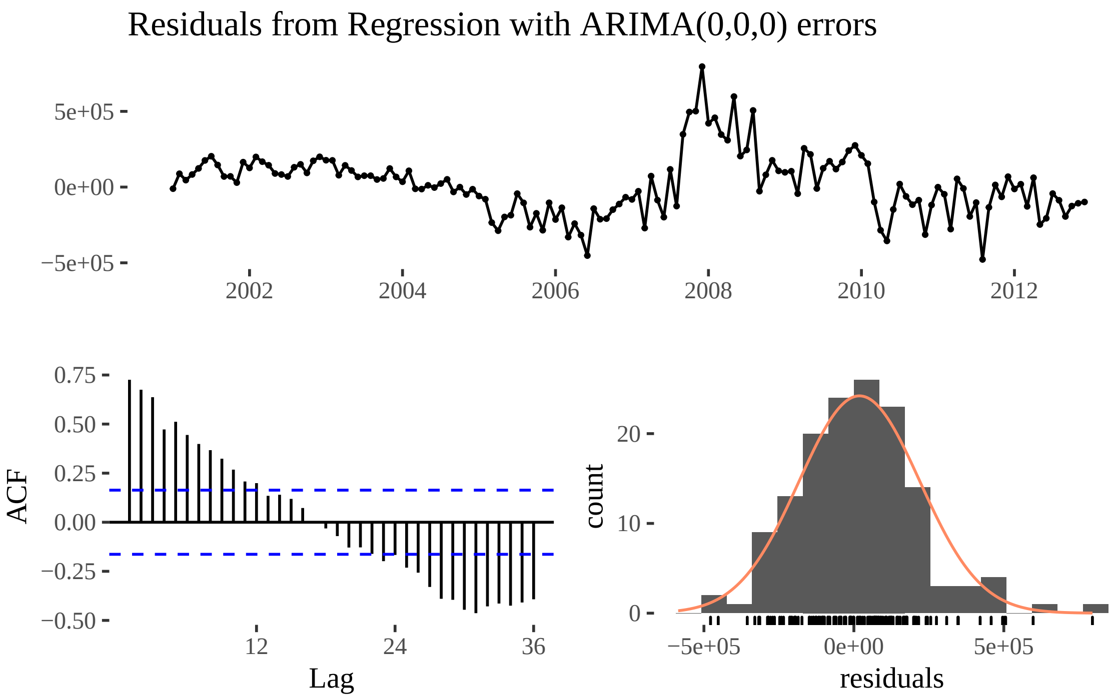
We see that a simple linear regression is fraught with residual issues, including auto correlation for double-digit lags and non-randomness.

Linear regression with ARMA errors (use arima with xreg)

```
lm_arma_params <- rlang::list2(
  cme = list(price = quo(price), total_volume = quo(total_volume)),
  imm = list(price = quo(price), total_volume = quo(total_volume)),
  iom = list(price = quo(price), floor_volume = quo(floor_volume))
)

lm_arma_models <-
  map2(lm_arma_params, model_price_volume,
       function(.params, .data) {

         .data <- .data %>% ungroup()
         price <- .data %>% select(!!.params[[1]]) %>%
           as.ts(frequency = 12)
         xreg <- .data %>% select(!!.params[[2]]) %>%
           as.ts(frequency = 12)
         auto.arima(price, xreg = xreg)
       })
```

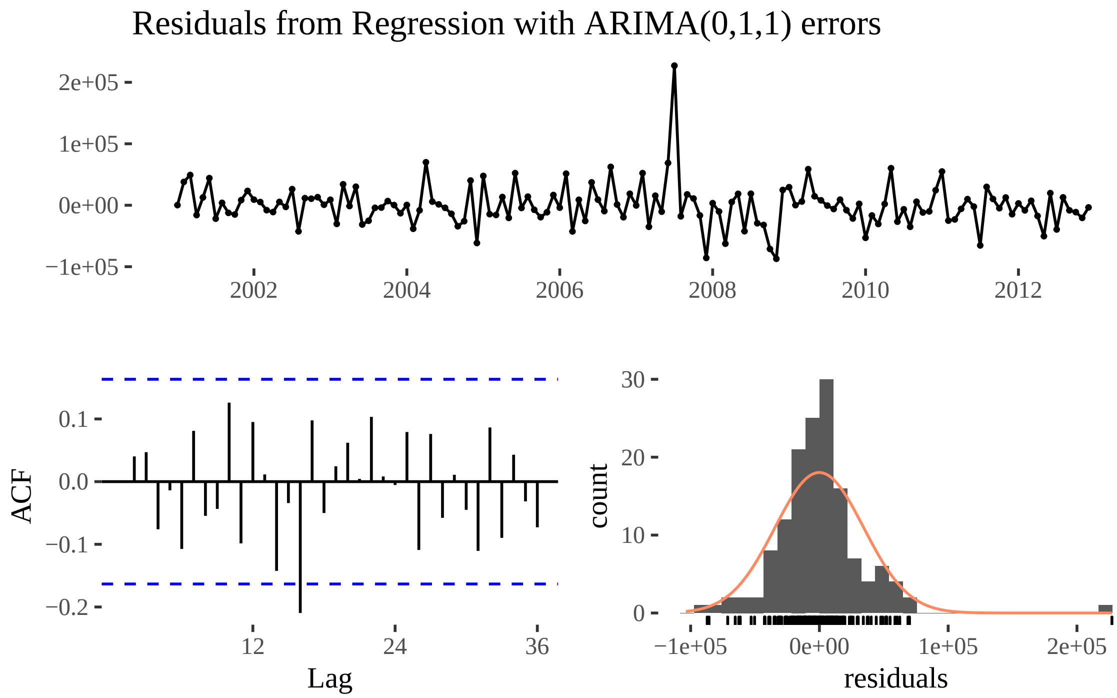
IMM looks better. However, we still have auto-correlation issues with CME/IOM.

```r
checkresiduals(lm_arma_models$cme)
```



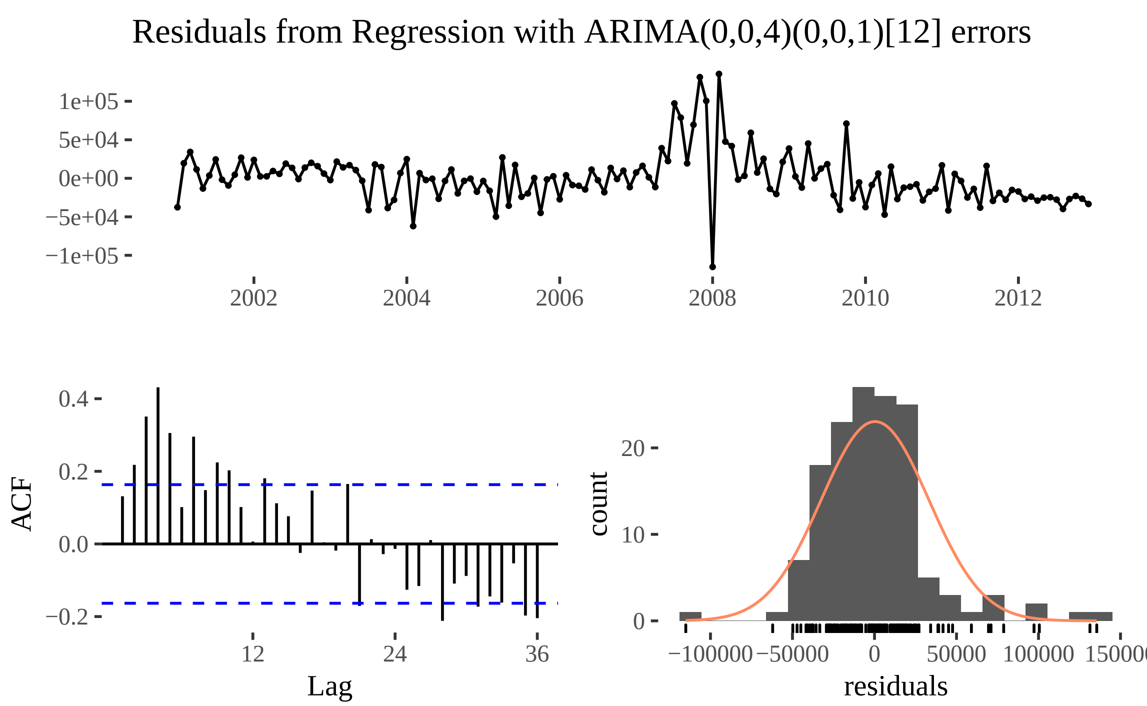

```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(0,0,0) errors
## Q* = 426.53, df = 23, p-value < 2.2e-16
##
## Model df: 1.   Total lags used: 24
```

```r
checkresiduals(lm_arma_models$imm)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(0,1,1) errors
## Q* = 25.77, df = 22, p-value = 0.2617
##
## Model df: 2.   Total lags used: 24
```

```r
checkresiduals(lm_arma_models$iom)
```



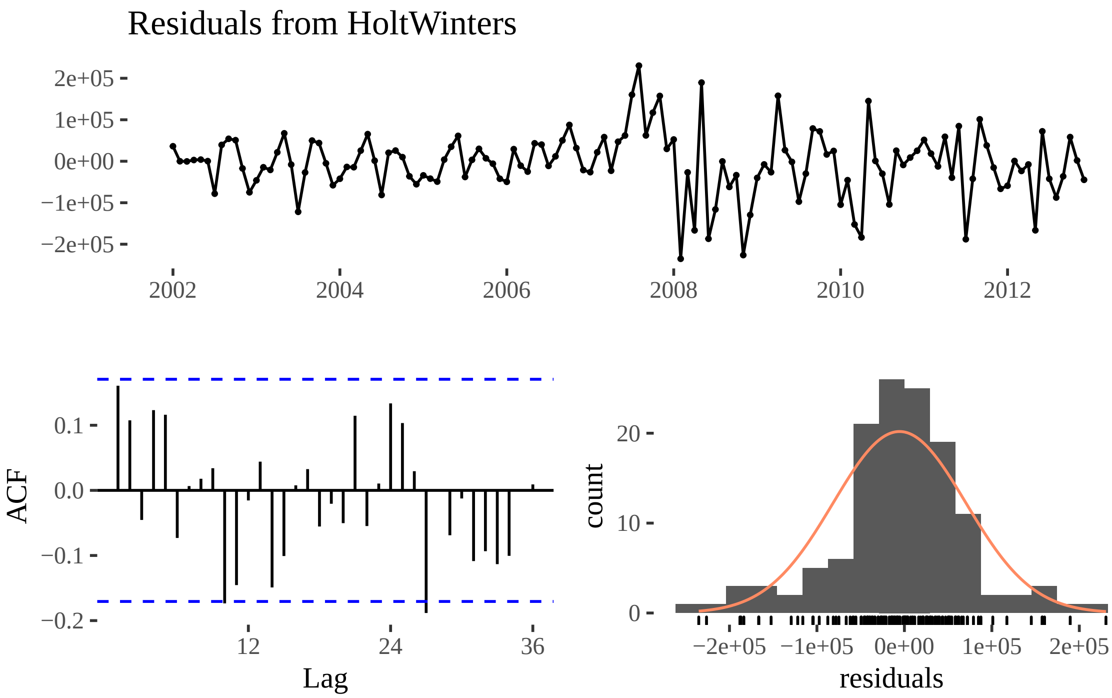
```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(0,0,4)(0,0,1)[12] errors
## Q* = 126.04, df = 17, p-value < 2.2e-16
##
## Model df: 7.   Total lags used: 24
```

Holts Winters

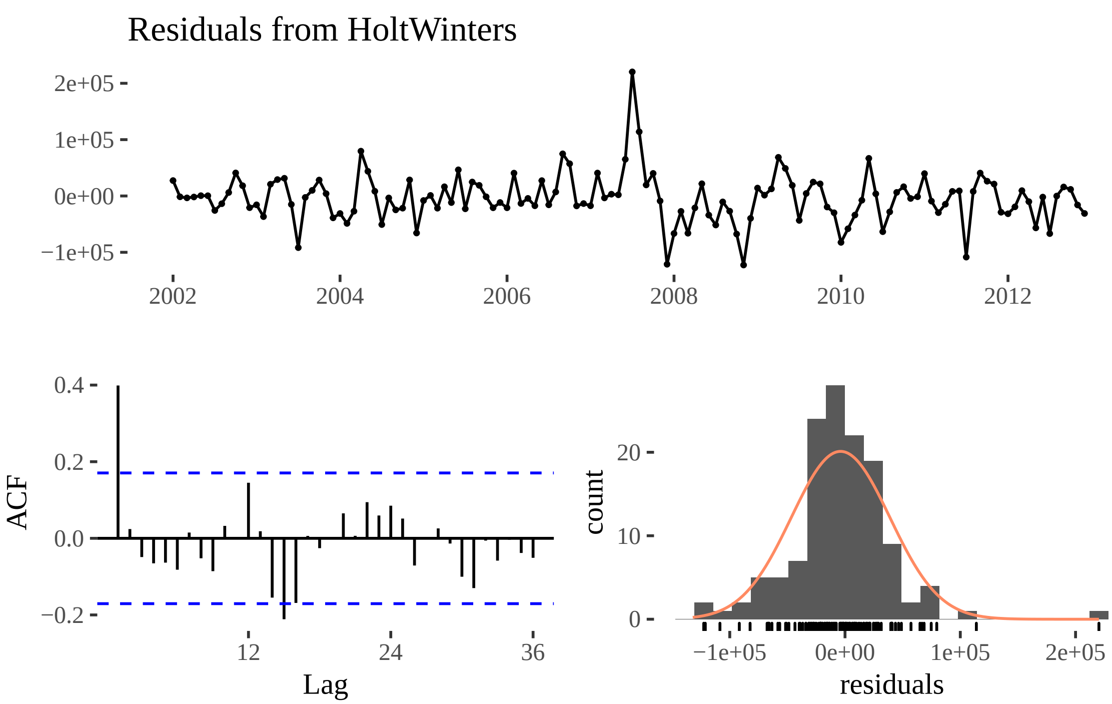
```r
fit_hw <-
  function(.ts, seasonal = TRUE, mult = FALSE) {
    if (seasonal == TRUE) {
      if (mult) {
        model <-
          HoltWinters(.ts, seasonal = "mult")
      } else {
        model <- HoltWinters(.ts)
      }

    } else {
      model <- HoltWinters(.ts, gamma = FALSE)
    }
    model
  }

hw_models <-
  map(model_price_volume, function(.data) {
    fit_hw(.data %>%
             select(price, year_month) %>%
             as.ts(frequency = 12))
  })
```

Holt Winters does a pretty good job with all three divisions. Auto-correlation is less of a problem and residuals look closer to white noise. However, CME variance in residuals spikes after 2007.

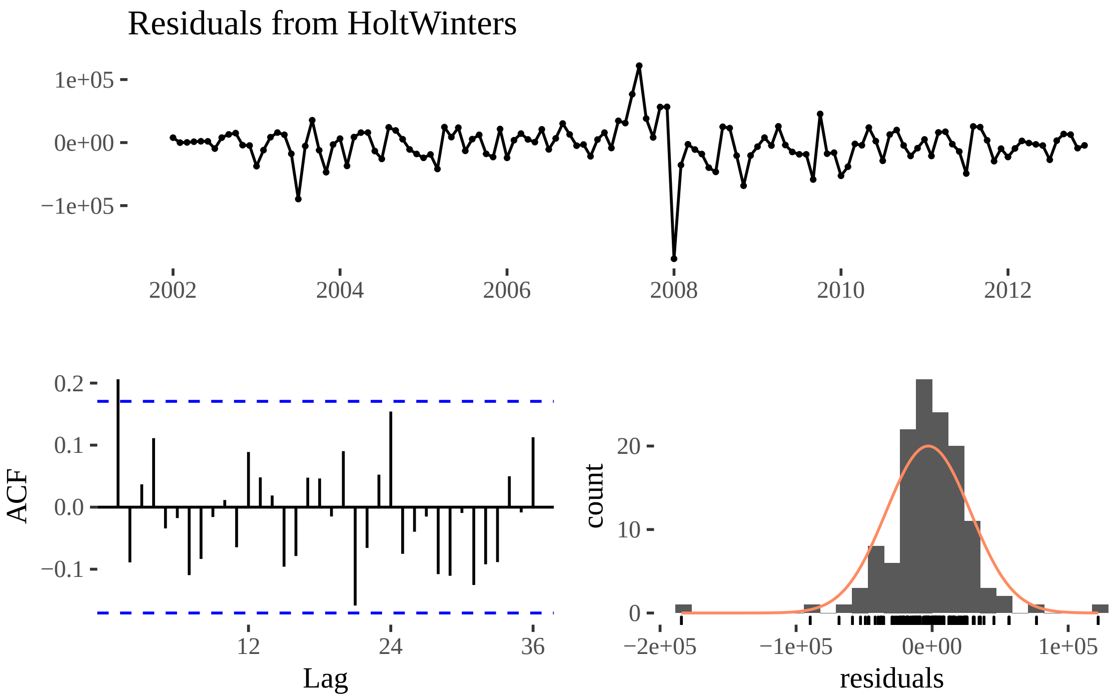
```r
checkresiduals(hw_models$cme)
```




```
checkresiduals(hw_models$imm)
```




```
checkresiduals(hw_models$iom)
```

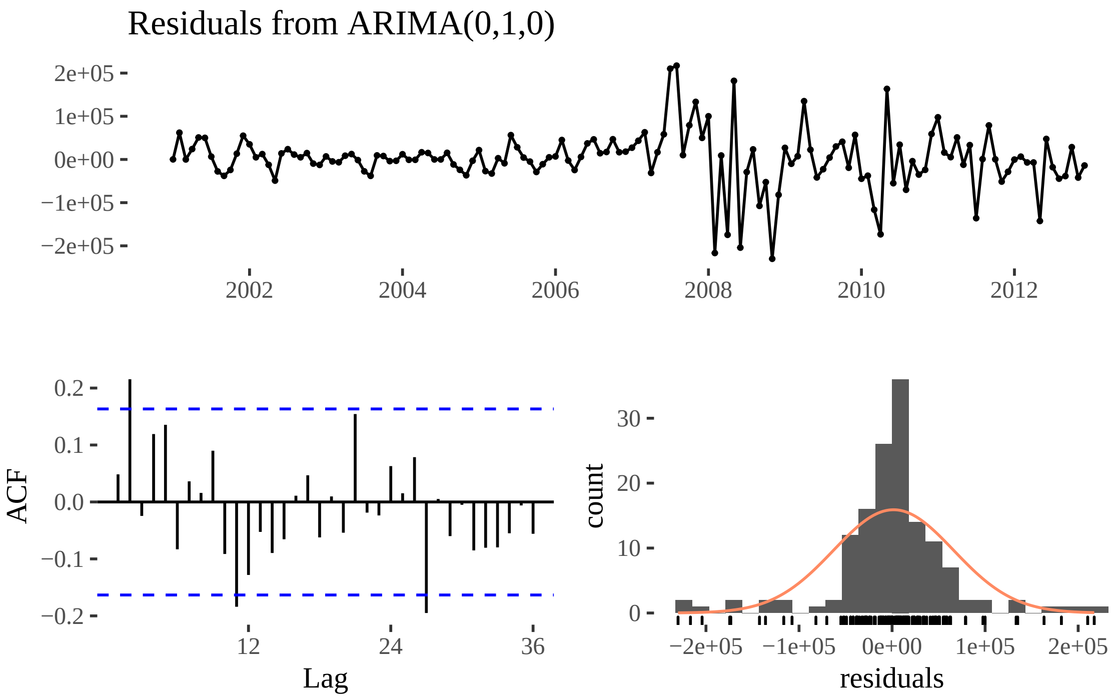


ARIMA

```
fit_arima <-
  function(.ts) {
    auto.arima(.ts, seasonal = FALSE)
  }

arima_models <-
  map(model_price_volume, function(.data) {
    fit_arima(.data %>%
                select(price, year_month) %>%
                as.ts(frequency = 12))
  })
```

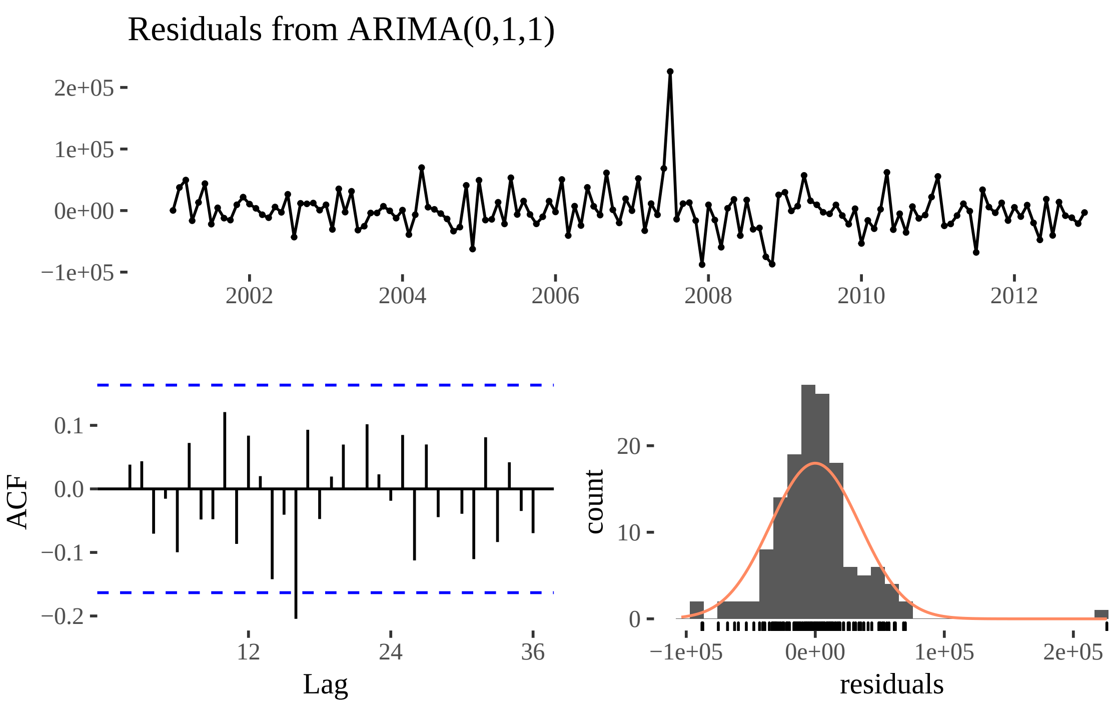
ARIMA is another good model for these data, but we have the same increasing variance problem for CME after 2007.

```
checkresiduals(arima_models$cme)
```



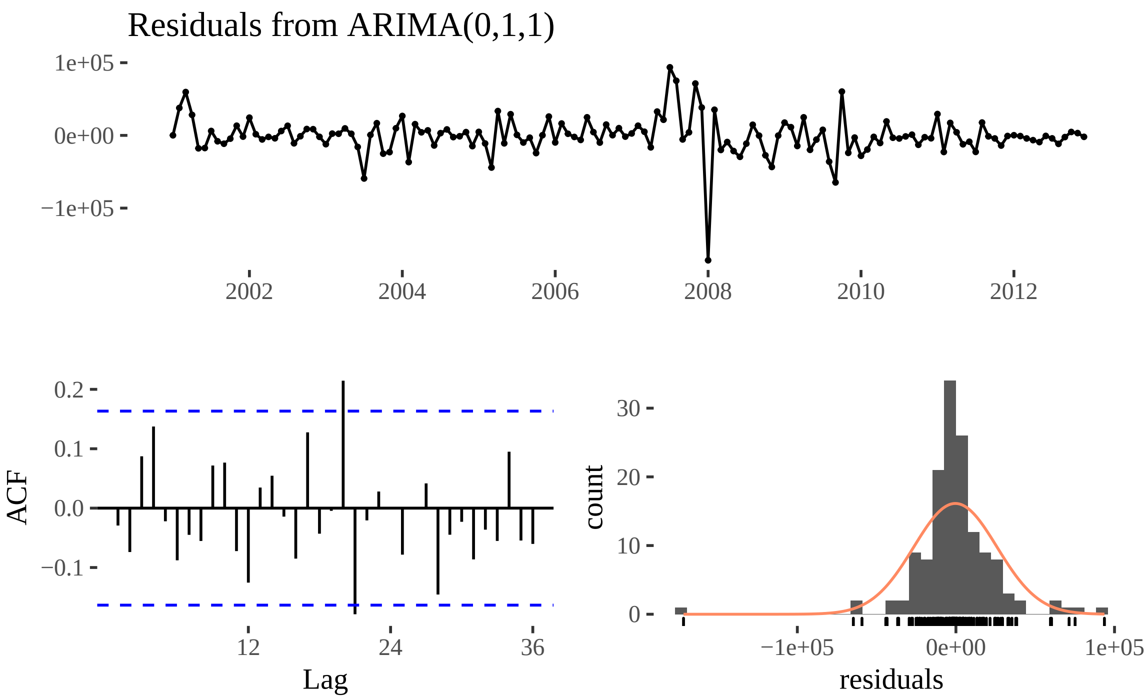

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,0)
## Q* = 32.945, df = 24, p-value = 0.1052
##
## Model df: 0.   Total lags used: 24
```

```
checkresiduals(arima_models$imm)
```



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,1)
## Q* = 24.002, df = 23, p-value = 0.4037
##
## Model df: 1.   Total lags used: 24
```

```
checkresiduals(arima_models$iom)
```



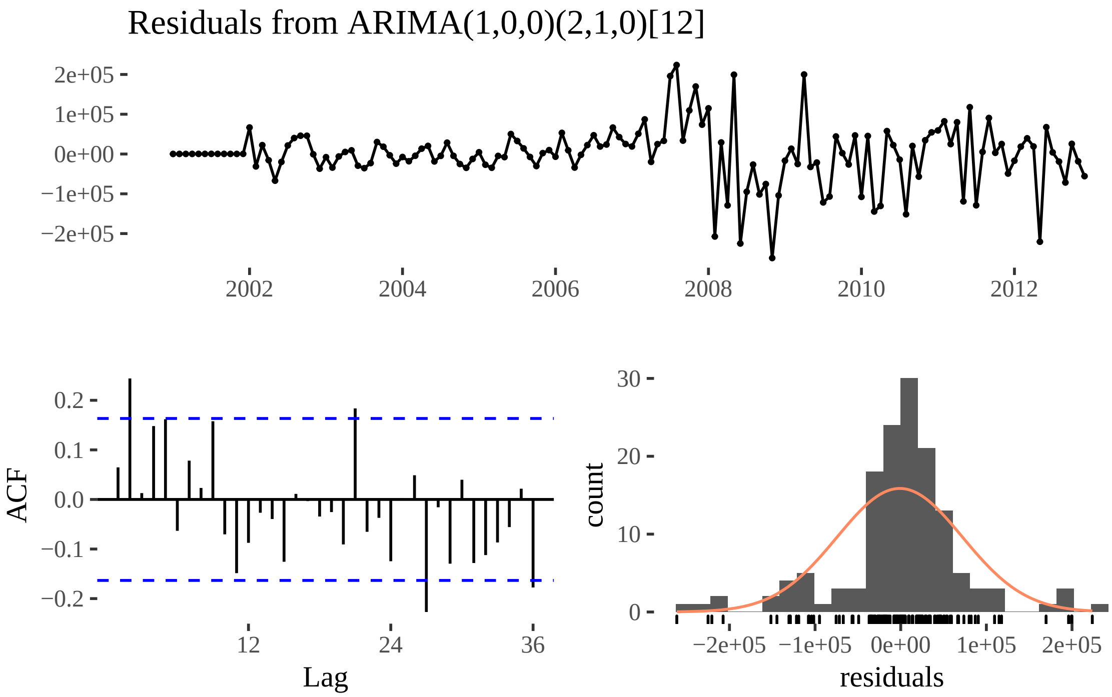

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,1)
## Q* = 30.364, df = 23, p-value = 0.1392
##
## Model df: 1.   Total lags used: 24
```

SARIMA (seasonality is monthly)

```
fit_sarima <-
  function(.ts) {
    model <- auto.arima(.ts, D = 1)
  }

sarima_models <-
  map(model_price_volume, function(.data) {
    fit_sarima(.data %>%
                 select(price, year_month) %>%
                 as.ts(frequency = 12))
  })
```

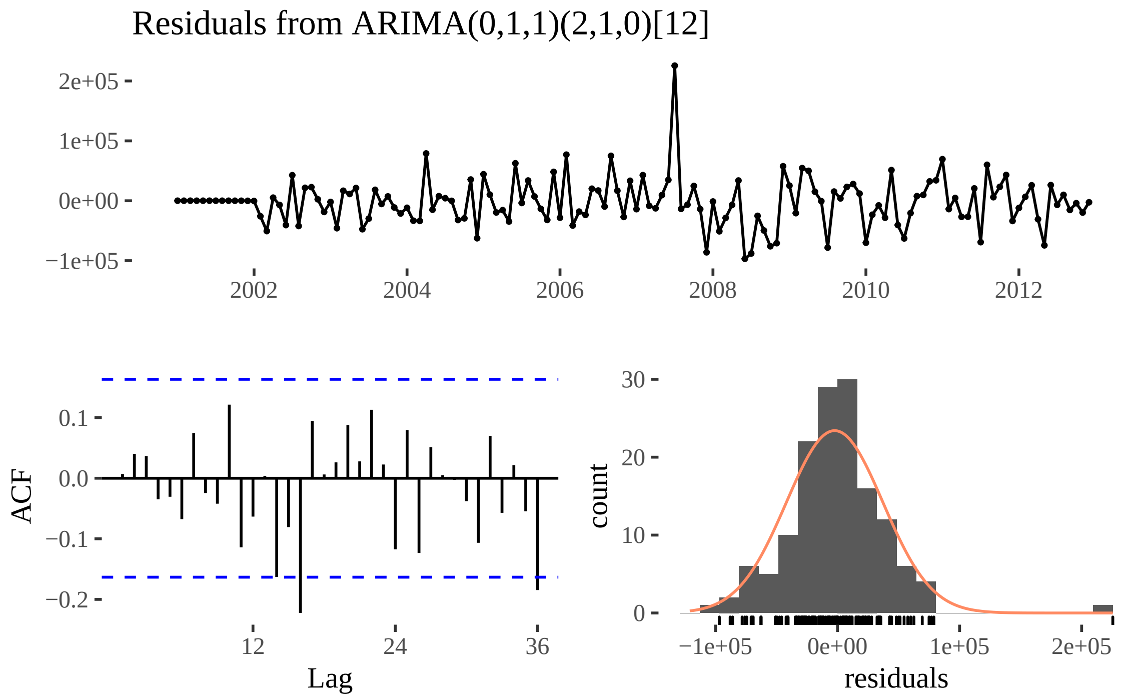
There is no benefit to increased complexity from SARIMA. We still have the same issues with residuals.

```r
checkresiduals(sarima_models$cme)
```



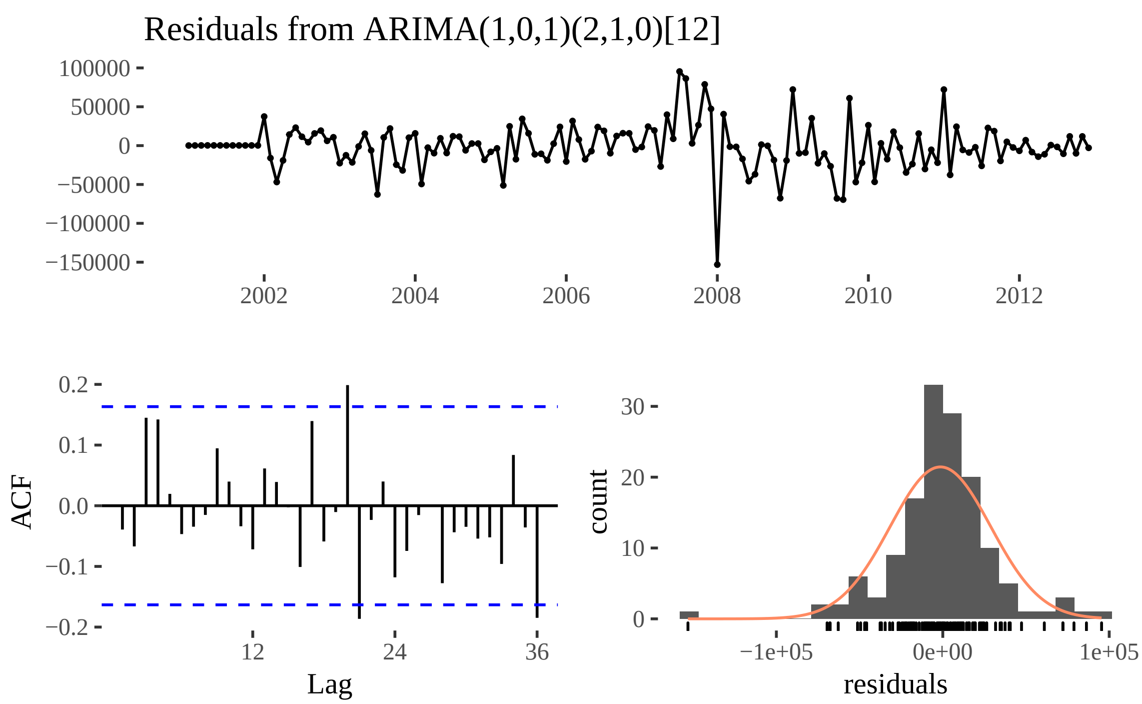

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,0)(2,1,0)[12]
## Q* = 41.817, df = 21, p-value = 0.004436
##
## Model df: 3.   Total lags used: 24
```

```r
checkresiduals(sarima_models$imm)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,1)(2,1,0)[12]
## Q* = 28.924, df = 21, p-value = 0.1158
##
## Model df: 3.   Total lags used: 24
```

```
checkresiduals(sarima_models$iom)
```



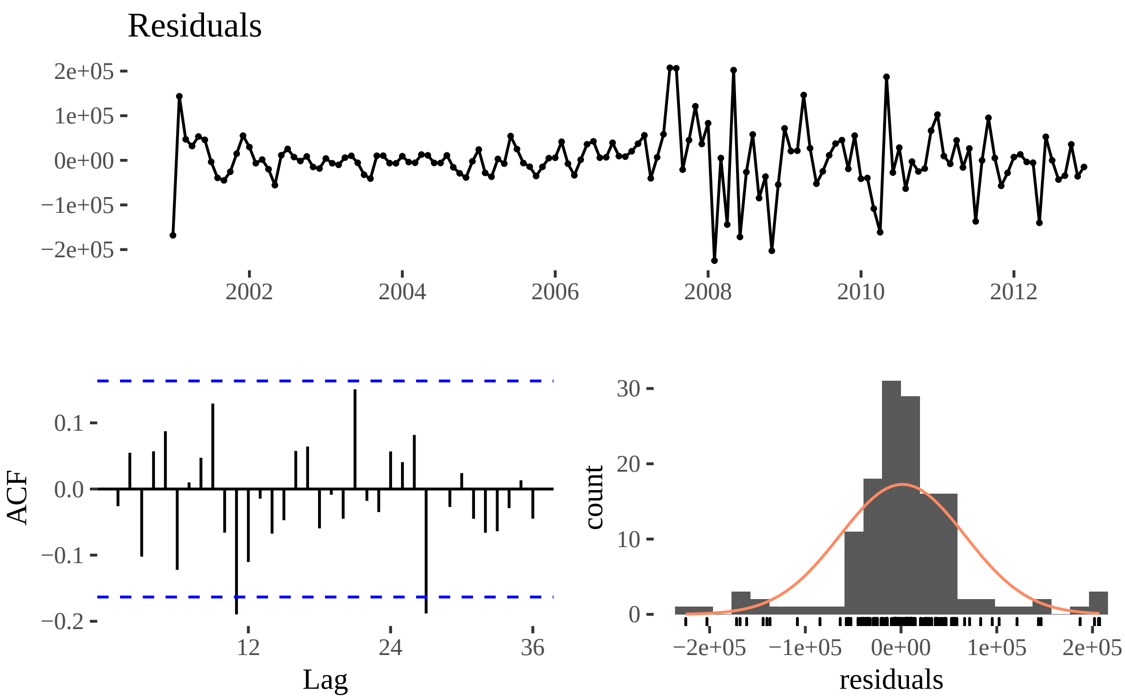
```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,1)(2,1,0)[12]
## Q* = 32.117, df = 20, p-value = 0.04207
##
## Model df: 4.   Total lags used: 24
```

Fractional ARIMA (ARFIMA)

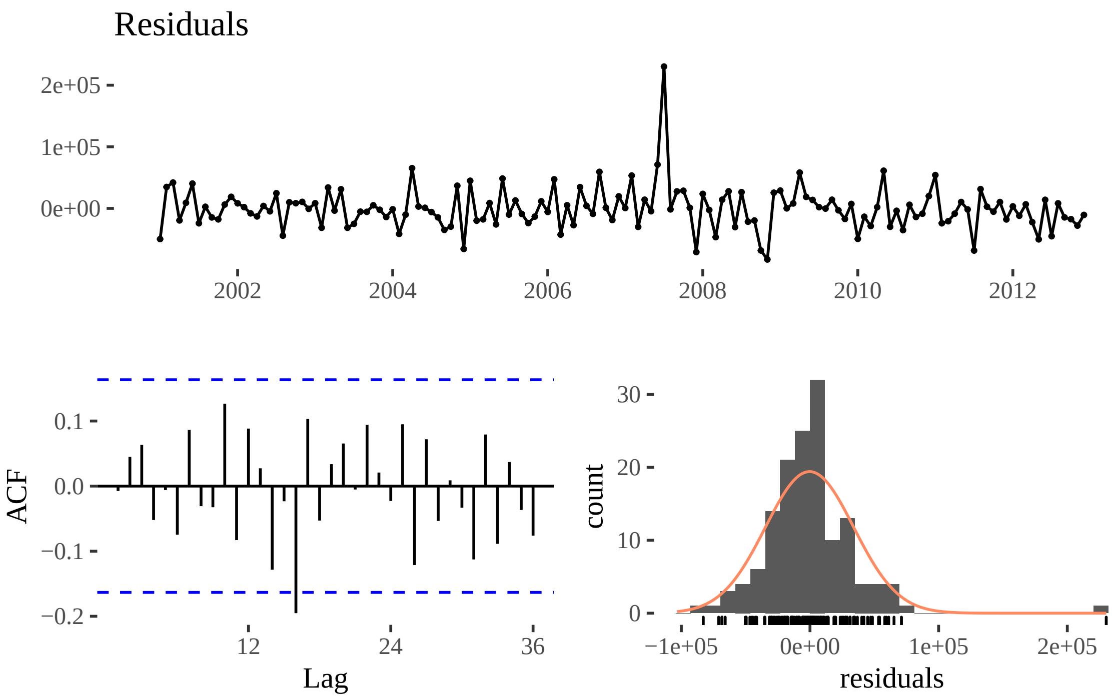
```r
fit_arfima <-
  function(.ts) {
    model <- forecast::arfima(.ts)
  }

arfima_models <-
  map(model_price_volume, function(.data) {
    fit_arfima(.data %>%
                 select(price, year_month) %>%
                 as.ts(frequency = 12))
  })
```

Fractional provides no benefits to our model fitting exercise.

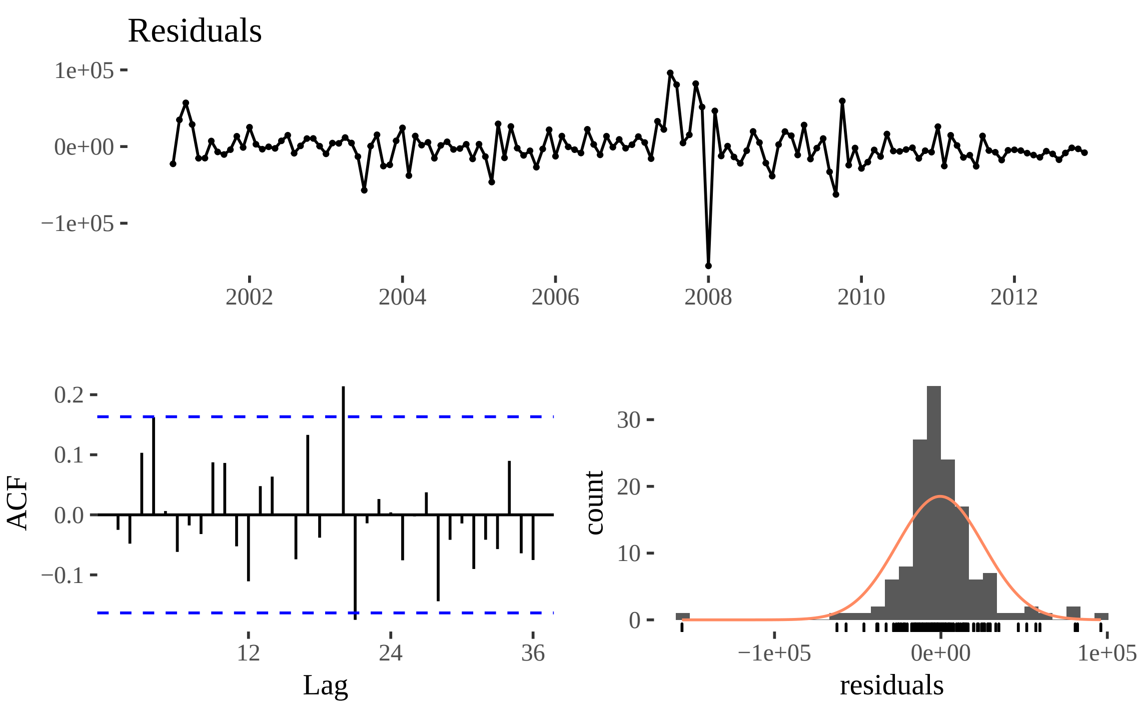
```r
checkresiduals(arfima_models$cme)
```



```r
checkresiduals(arfima_models$imm)
```




```r
checkresiduals(arfima_models$iom)
```



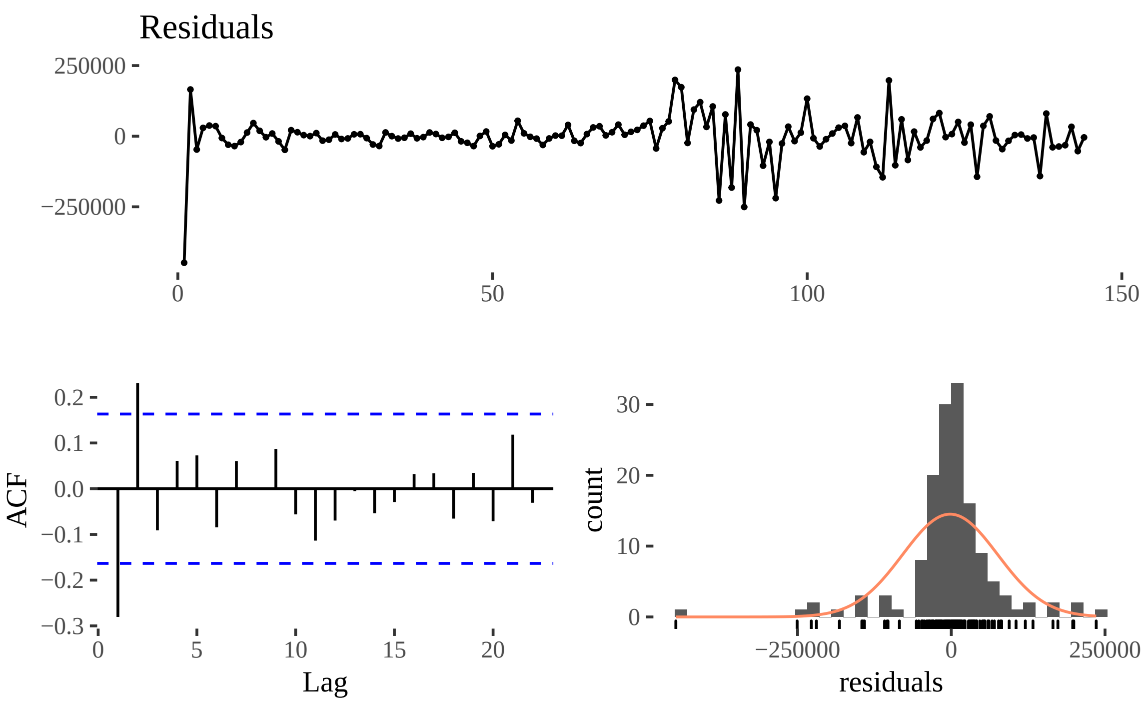
ARMA and GARCH combination - use the fGarch R library and garchFit()

```r
fit_garch <-
  function(.ts) {
```

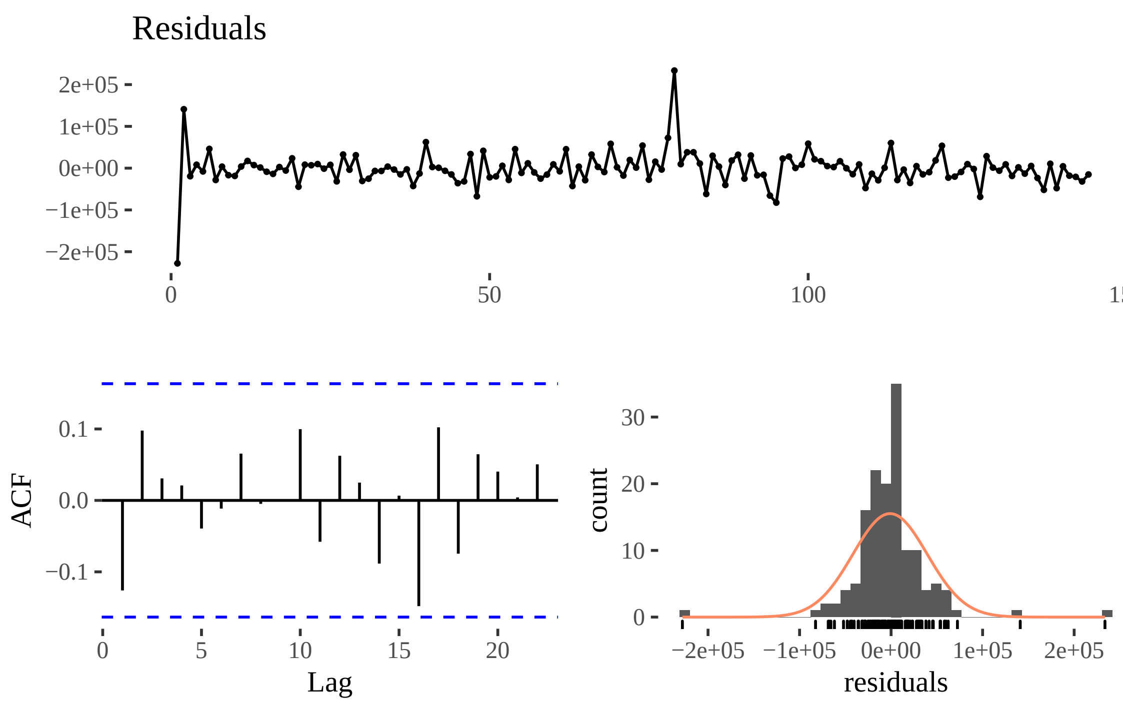
```
    arima_model <- auto.arima(.ts, seasonal = FALSE)
        .spec <- ugarchspec(
          variance.model = list(
          mean.model = list(
            armaOrder = arimaorder(arima_model),
            include.mean = T
          ),
          distribution.model = "std"
          )
        )
    ugarchfit(spec = .spec, data = .ts)
  }

garch_models <-
  map(model_price_volume, function(.data) {
    fit_garch(.data %>% select(price, year_month) %>% as.ts(frequency = 12))
  })
```

In spite of not fixing variance, GARCH does a good job reducing auto correlation.

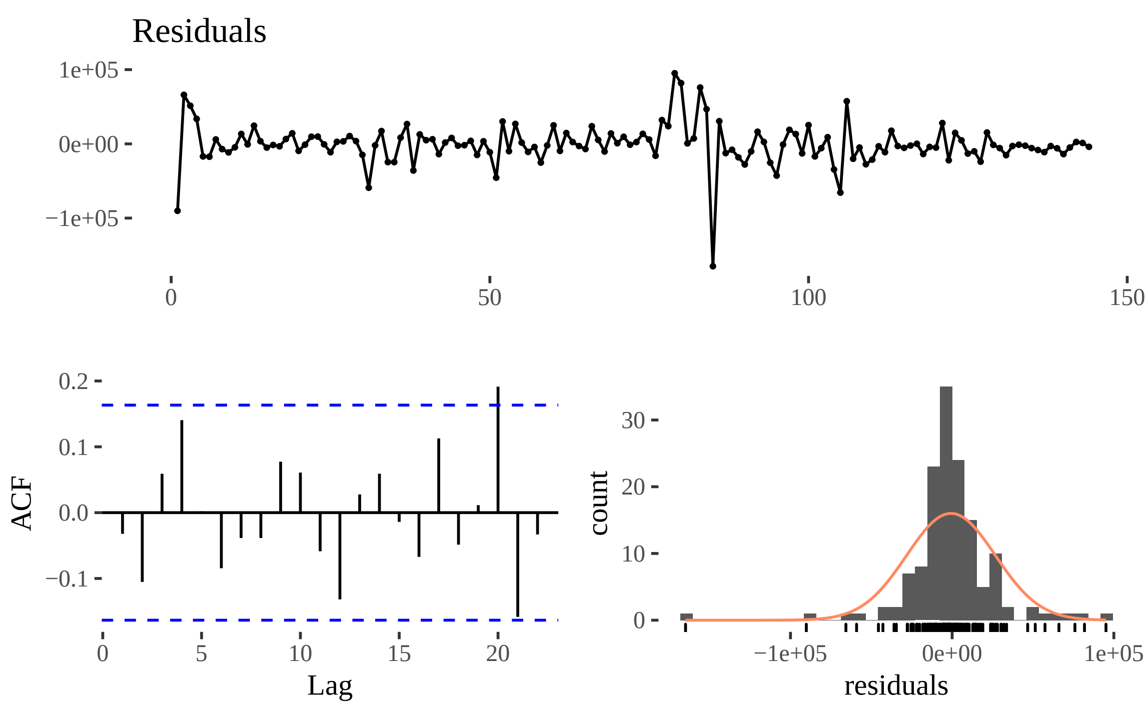
```
checkresiduals(garch_models$cme@fit)
```




```
checkresiduals(garch_models$imm@fit)
```




```r
checkresiduals(garch_models$iom@fit)
```



Model evaluation using sMAPE

Test data

```r
test_price_volume <- test_volumes %>%
  bind_rows(.id = "division") %>%
```

```r
  left_join(all_divisions_test, c("division", "year_month")) %>%
  distinct() %>%
  split(.$division) %>%
  map(
    ~ .x %>%
      ungroup %>%
      mutate(year_month = yearmonth(year_month)) %>%
      as_tsibble(key = division, index = year_month)
  )
```

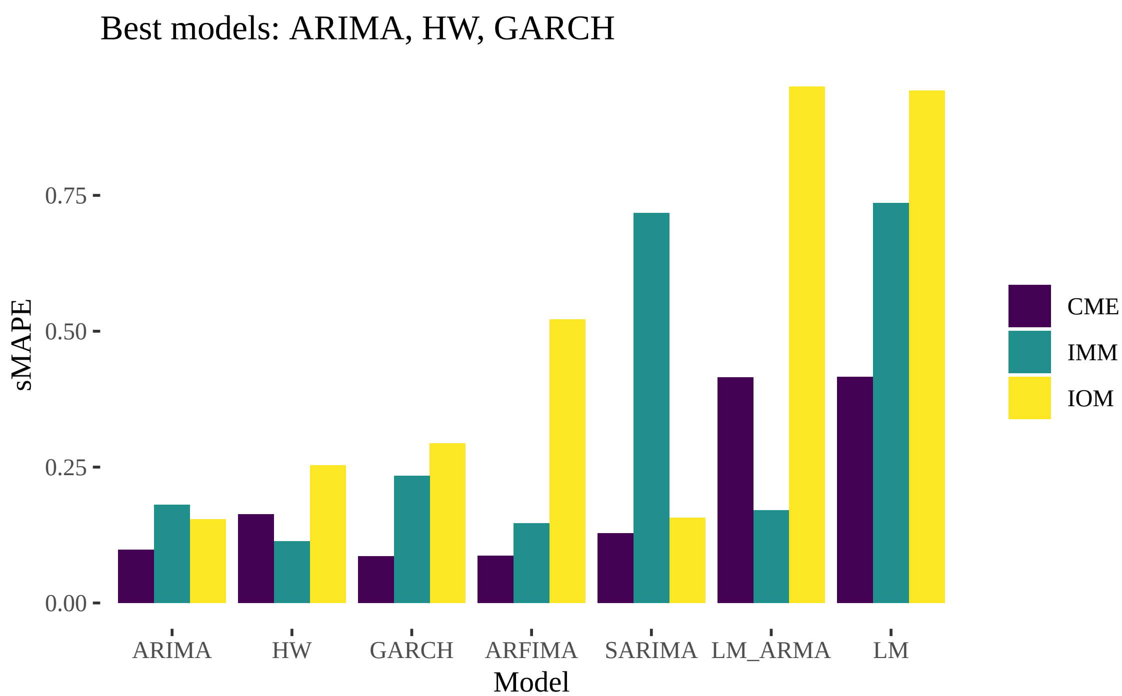
Evaluation

```r
smape <-
  function(prediction, actual) {
    pred_vs_actual <- abs(prediction - actual)
    n <- length(prediction)
    sum(pred_vs_actual / ((abs(actual) + abs(prediction)) / 2)) / n
  }

lm_smape <-
  map2(lm_models,
       test_price_volume,
       ~ predict(.x, newdata = .y) %>%
         smape(.y$price))

lm_arma_smape <-
  pmap(list(lm_arma_params, lm_arma_models, test_price_volume),
       function(.params, .model, .data) {
         .data <- .data %>% ungroup()
         xreg <-
           .data %>% select(!!.params[[2]]) %>% as.ts(frequency = 12)
         forecast(.model, xreg = xreg)$mean %>%
           smape(.data$price)
       })

hw_smape <-
  map2(hw_models,
       test_price_volume,
       ~ forecast(.x, 12)$mean %>%
         smape(.y$price))

arima_smape <-
  map2(arima_models,
       test_price_volume,
       ~ forecast(.x, 12)$mean %>%
         smape(.y$price))

sarima_smape <-
  map2(sarima_models,
       test_price_volume,
       ~ forecast(.x, 12)$mean %>%
         smape(.y$price))
```

```r
arfima_smape <-
  map2(arfima_models,
       test_price_volume,
       ~ forecast(.x, 12)$mean %>%
         smape(.y$price))

garch_smape <-
  map2(garch_models,
       test_price_volume,
       ~ as.vector(fitted(ugarchforecast(.x, n.ahead = 12))) %>%
         smape(.y$price))
```

We recommend any of the three models: ARIMA, HW, GARCH. Each performed relatively strongly with correctness of forecast. Additionally, these models had more favorable residual diagnostics than the high error models.

```r
smape_df <- bind_rows(
  lm_smape = lm_smape,
  lm_arma_smape = lm_arma_smape,
  hw_smape = hw_smape,
  arima_smape = arima_smape,
  sarima_smape = sarima_smape,
  arfima_smape = arfima_smape,
  garch_smape = garch_smape,
  .id = "model"
) %>%
  gather(key = division, value = smape, -1) %>%
  group_by(model) %>%
  mutate(total_smape = sum(smape)) %>%
  ungroup()

smape_df %>%
  ggplot(aes(fct_reorder(model, total_smape), smape, fill = division)) +
  geom_col(position = "dodge") +
  scale_fill_viridis_d(name = NULL, labels = toupper) +
  scale_x_discrete(
    labels = function(x)
      str_remove_all(x, "_smape") %>% toupper
  ) +
  labs(title = "Best models: ARIMA, HW, GARCH",
       x = "Model",
       y = "sMAPE")
```


Best models: ARIMA, HW, GARCH
sMAPE
0.75
0.50
0.25
0.00
ARIMA
HW
GARCH
ARFIMA
SARIMA
LM_ARMA
LM
Model
CME
IMM
IOM