FOREST FIRES IN BRAZIL

1998 - 2017

Environmental Informatics

301035

Joshua Ham 18618046, Alexis Kim 19024383, James Monks 18500484

15. 10. 2019

Dr. Than Pe

Abstract

Forest fires are a serious problem for the preservation of the Tropical Forests. Understanding the frequency of forest fires in a time series can help prevent them. Brazil has the largest rainforest on the planet, the Amazon rainforest. This report analyses the frequency of forest fires by the state in which the fire occurred as well as the time or season.

Contents

1 Introduction

1.1 The Amazon Dataset

The dataset “amazon” has been obtained from (cite). It allows the assessment of the evolution of fires over the period of approximately 10 years (from 1998 to 2017) as well as the regions where they were concentrated. The legal Amazon comprises the states of Acre, Amapá, Pará, Amazonas, Rondonia, Roraima, and part of Mato Grosso, Tocantins, and Maranhão.

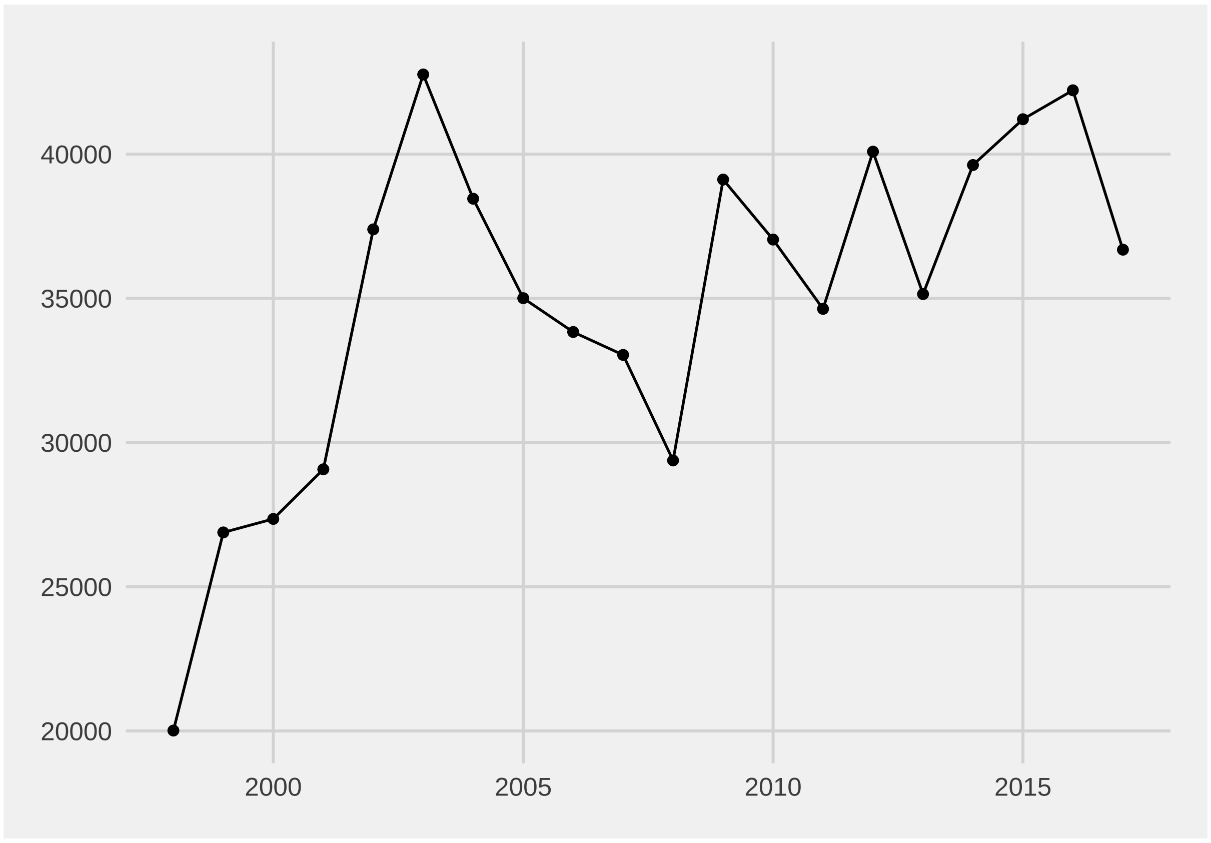
```r
fires <- read_rds(here::here("data-raw", "fires_date.rds"))
head(fires)
```

```
#  # A tibble: 6 x 9
#     year state month number date       month_clean   day date_month
#    <dbl> <chr> <chr>  <dbl> <date>     <chr>       <dbl> <date>
#  1  1998 Acre  Jane~      0 1998-01-01 January         1 1998-01-01
#  2  1999 Acre  Jane~      0 1999-01-01 January         1 1999-01-01
#  3  2000 Acre  Jane~      0 2000-01-01 January         1 2000-01-01
#  4  2001 Acre  Jane~      0 2001-01-01 January         1 2001-01-01
#  5  2002 Acre  Jane~      0 2002-01-01 January         1 2002-01-01
#  6  2003 Acre  Jane~     10 2003-01-01 January         1 2003-01-01
#  # ... with 1 more variable: date_index <mth>
```

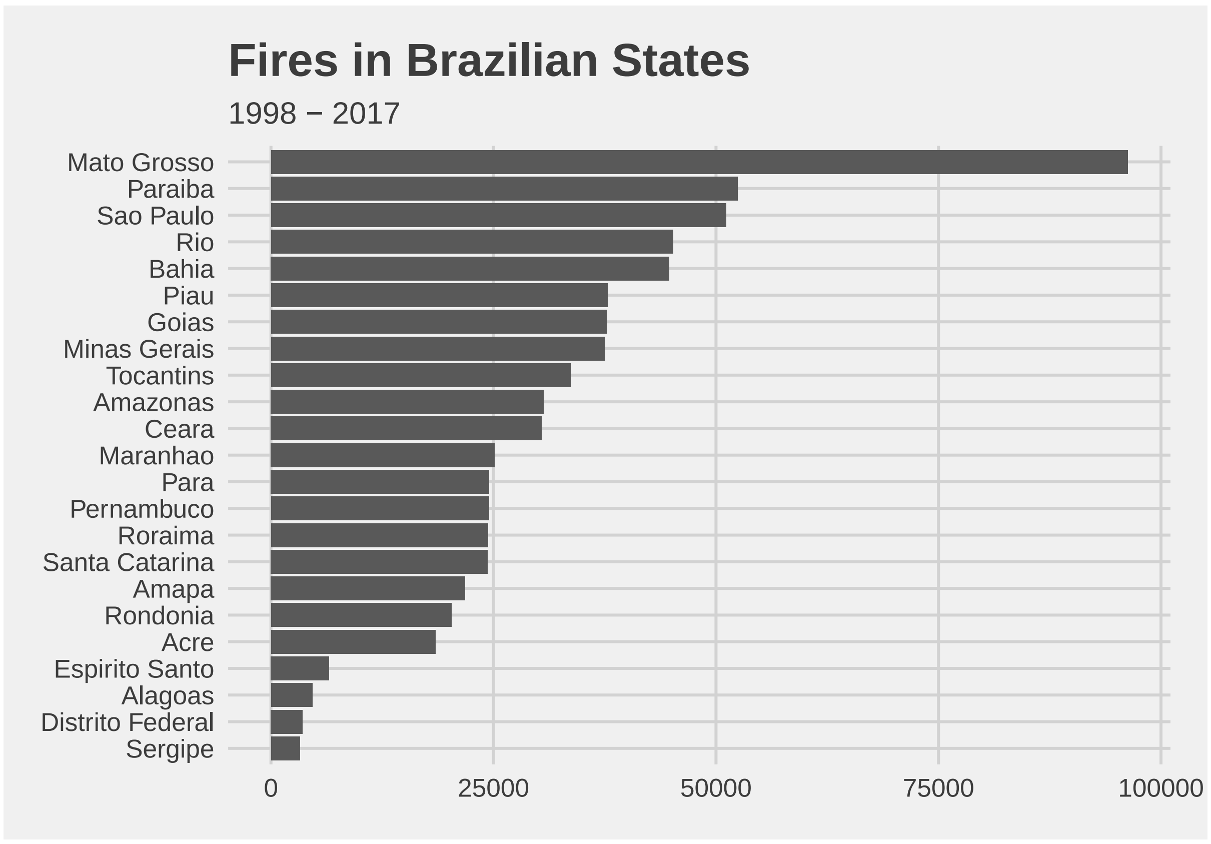
1.2 Descriptive Statistics

We begin by visualising the data.

```r
fires %>%
  count(date, wt = number) %>%
  ggplot(aes(x = date, y = n)) +
  geom_point() +
  geom_line() +
  ggthemes::theme_fivethirtyeight()
```



```
fires %>%
  as_tibble() %>%
  group_by(state) %>%
  summarise(fires = sum(number, nana.rm = TRUE)) %>%
  mutate(state=forcats::fct_reorder(state,fires)) %>%
  ggplot(aes(x = state, y = fires))+
  geom_bar(stat = "identity")+
  coord_flip()+
  ggthemes :: theme_fivethirtyeight() +
  ggtitle("Fires in Brazilian States", subtitle = "1998 - 2017")
```



From the plot of date against the number of forest fires reported it can be seen that forest fires were the most prevalent during 2003 with over 42500 counts of forest fires. 2016 follows close by with a count slightly below 42500 fires. Over the years, the prevalence of forest fires appear to increase closer to 40000 forest fires.

The plot of state against the number of forest fires reported demonstrates the outstanding prevalence of forest fires in Mato Grosso compared to the other states. Whether this is an outlier or error due to measurement or data compilation should be investigated during the data cleaning process.

2 Data Cleaning

In the original data set, the month variable was in Poruguese. Poruguese is not parsable by standard date conversion functions, and hence the months were translated. This included having to deal with non-standard characters that could not be manually translated, and needed to be extracted as references from the original data.

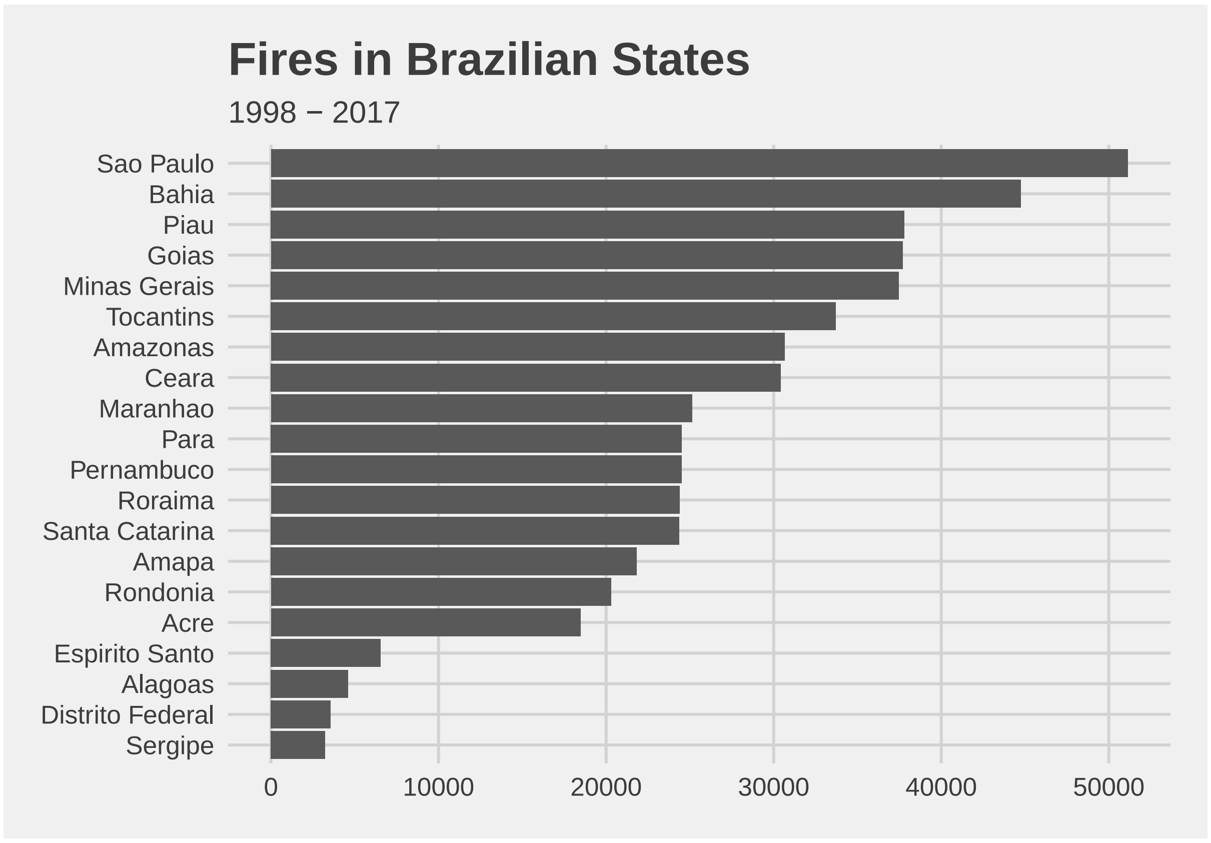
Exploring the data has revealed some states have duplicated data for the same instance of time. In the case of Alagoas these were exact duplicates and it was simple to take only one of each occurrence. In other cases, however, the duplicates were drastically different, specifically the states of Mato Grosso, Rio and Paraiba. As this data was collected from the Brazillian government website by (Kaggle Reference), the source was not contacted to clarify this (due to language and access constraints) and these states were excluded from analysis.

The standard data frame format lends itself well to visualisation and other data manipulation tasks, however is not aware inherently of any temporal component. The time series (ts) data format does have this "time aware" feature, however it does not have the adaptability of the data frame. It is for this reason that the tsibble object exists, allowing a time index to be specified on a standard data frame along with a key to specify separate time series.

For detailed information on data cleaning, see appendix 1.

The resultant clean data can be seen below:

```
fires <- fires_clean


fires %>%
  as_tibble() %>%
  group_by(state) %>%
  summarise(fires = sum(fires)) %>%
  mutate(state=forcats::fct_reorder(state,fires)) %>%
  ggplot(aes(x = state, y = fires))+
  geom_bar(stat = "identity")+
  coord_flip()+
  ggthemes :: theme_fivethirtyeight() +
  ggtitle("Fires in Brazilian States", subtitle = "1998 - 2017")
```



3 Data Analysis

3.1 Temporal Analysis

3.1.1 Modelling The Time Series

This analysis has been broken into 3 different components. The first is manual decomposition and manual modelling of data. This is to show the theory behind the forecasts made. The second component uses the “fable” forecasting framework for decomposition and modelling of the time series in order to show a traditional workflow utilising this framework. The third component utilises the automatic decomposition and “re-composition” functionality provided by fable, which allows for plots and forecasts to be made in the context of the original data, leading to a more understandable output.

3.1.1.1 Manual Decomposition and Modelling

We begin by deseasoning the data. A year comprises of four seasons and each season is approximately three months in length. Thus to summarise the data for each season, we first calculate the moving average fire counts of three consecutive months (MA). The centered moving average (CMA) is calculated by the average of two consecutive MAs:

```r
fires_a <- read_rds(here::here("data", "fires_clean_all.rds"))

MA <- NULL
CMA <- NULL

for(i in 1:238){
  MA[i+1] <- (fires_a$fires[i] + fires_a$fires[i+1]
              + fires_a$fires[i+2])/3
}

for(i in 1:237){
  CMA[i+2] <- (MA[i+1] + MA[i+2])/2
}
```

Now we divide the fire counts by the centred moving averages and call it "n_alt":

```r
n_alt <- NULL

for(i in 3:238){
  n_alt[i] <- fires_a$fires[i]/CMA[i]
}

n_alt[239] <- NA
n_alt[which(is.nan(n_alt))] <- 0
```

Now we build a new data set "fires_b" which contains the original cleaned data set, a month column, the moving average and centered moving average values as well as the new fire counts in n_alt:

```r
month_nr <- NULL
for(i in 1:239){
  month_nr[i] <- i%%12
}

fires_b <- as.data.frame(cbind(as_tibble(fires_a), month_nr, MA, CMA, n_alt))
head(fires_b)
```

```
#         date fires month_nr        MA       CMA    n_alt
#   1 1998 Jan     0        1        NA        NA       NA
#   2 1998 Feb     0        2    0.0000        NA       NA
#   3 1998 Mar     0        3    0.0000    0.0000 0.000000
#   4 1998 Apr     0        4    0.0000    0.0000 0.000000
#   5 1998 May     0        5  408.3333  204.1667 0.000000
#   6 1998 Jun  1225        6 1711.3333 1059.8333 1.155842
```

For each season, the relevant months are collected together:

```
winter <- subset(fires_b,
                 fires_b$month == 12 |
                   fires_b$month == 1 |
                   fires_b$month == 2)
```

```
#  Warning in `$.data.frame`(fires_b, month): Partial match of 'month' to
#  'month_nr' in data frame

#  Warning in `$.data.frame`(fires_b, month): Partial match of 'month' to
#  'month_nr' in data frame

#  Warning in `$.data.frame`(fires_b, month): Partial match of 'month' to
#  'month_nr' in data frame
```

```
spring <- subset(fires_b,
                 fires_b$month == 3 |
                   fires_b$month == 4 |
                   fires_b$month == 5)
```

```
#  Warning in `$.data.frame`(fires_b, month): Partial match of 'month' to
#  'month_nr' in data frame

#  Warning in `$.data.frame`(fires_b, month): Partial match of 'month' to
#  'month_nr' in data frame

#  Warning in `$.data.frame`(fires_b, month): Partial match of 'month' to
#  'month_nr' in data frame
```

```
summer <- subset(fires_b,
                 fires_b$month == 6 |
                   fires_b$month == 7 |
                   fires_b$month == 8)
```

```
#  Warning in `$.data.frame`(fires_b, month): Partial match of 'month' to
#  'month_nr' in data frame

#  Warning in `$.data.frame`(fires_b, month): Partial match of 'month' to
#  'month_nr' in data frame

#  Warning in `$.data.frame`(fires_b, month): Partial match of 'month' to
#  'month_nr' in data frame
```

```
autumn <- subset(fires_b,
                 fires_b$month == 9 |
                   fires_b$month == 10 |
                   fires_b$month == 11)
```

```
#  Warning in `$.data.frame`(fires_b, month): Partial match of 'month' to
#  'month_nr' in data frame

#  Warning in `$.data.frame`(fires_b, month): Partial match of 'month' to
#  'month_nr' in data frame

#  Warning in `$.data.frame`(fires_b, month): Partial match of 'month' to
#  'month_nr' in data frame
```

Now we calculate the average for each season as well as the sum of these average values:

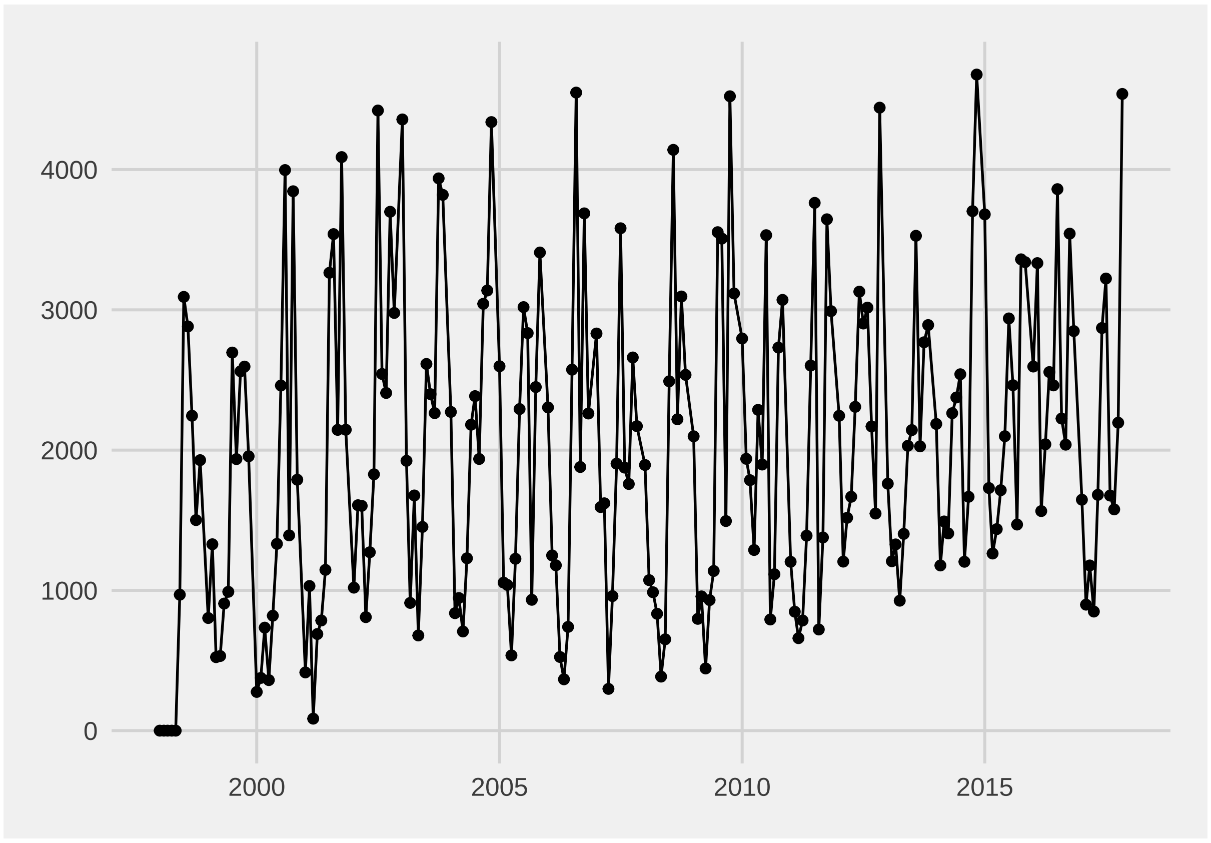
```
m_alt_winter <- mean(winter$n_alt, na.rm = TRUE)
m_alt_spring <- mean(spring$n_alt, na.rm = TRUE)
m_alt_summer <- mean(summer$n_alt, na.rm = TRUE)
m_alt_autumn <- mean(autumn$n_alt, na.rm = TRUE)

m_alt <- c(m_alt_winter, m_alt_spring, m_alt_summer, m_alt_autumn)
m_alt_sum <- sum(m_alt)
```

The proportion of each season's average is calculated as a decimal by dividing with the sum. Then these proportion are used to replace the m_alt values of each season with the fires count divided by the proportion:

```
for(i in 1:4){
  m_alt[i] <- m_alt[i]*4/m_alt_sum
}

for(i in 1:length(winter$fires)){
  winter[i,7] <- winter$fires[i]/m_alt[1]
}

for(i in 1:length(spring$fires)){
  spring[i,7] <- spring$fires[i]/m_alt[2]
}

for(i in 1:length(summer$fires)){
  summer[i,7] <- summer$fires[i]/m_alt[3]
}

for(i in 1:length(autumn$fires)){
  autumn[i,7] <- autumn$fires[i]/m_alt[4]
}
```

We now build the completely deseasoned dataset “fires_des”:

```
fires_des <- as.data.frame(rbind(winter,spring,summer,autumn))
fires_des$index <- as.numeric(row.names(fires_des))


fires_des %>%
  count(date, wt = V7) %>%
  ggplot(aes(x = date, y = n)) +
  geom_point() +
  geom_line() +
  ggthemes::theme_fivethirtyeight()
```



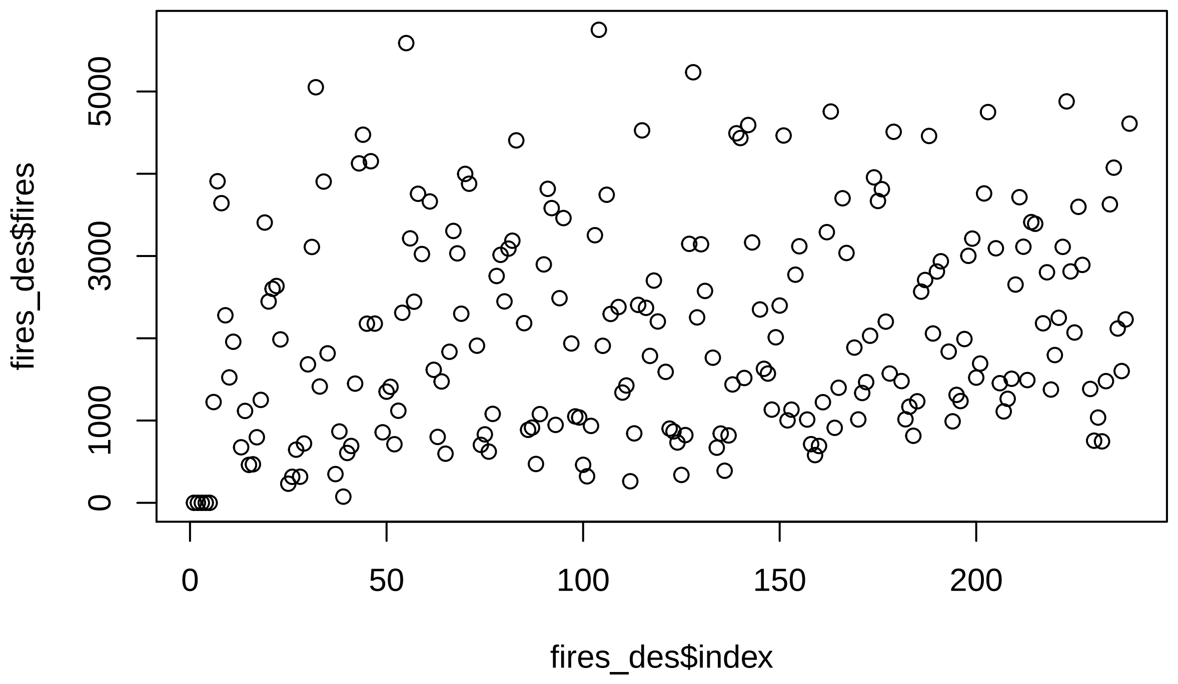
Now we model the deseasoned data to enable forecasting. We begin with a linear model, called fitA:

```
fitA <- lm(fires_des$fires ~ fires_des$index)
summary(fitA)
```

```
#
```

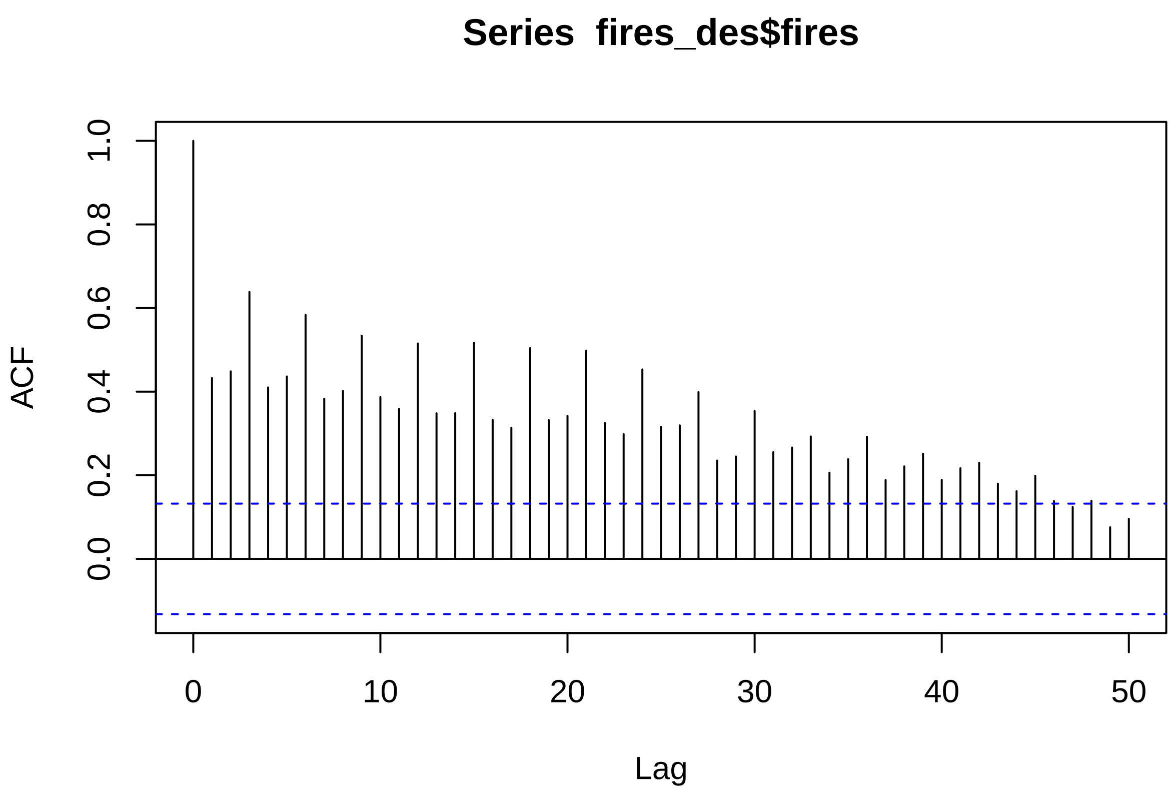
```
#  Call:
#  lm(formula = fires_des$fires ~ fires_des$index)
#
#  Residuals:
#      Min      1Q  Median      3Q     Max
#  -1815.8 -1088.5  -276.2   913.5  3702.6
#
#  Coefficients:
#                  Estimate Std. Error t value Pr(>|t|)
#  (Intercept)     1700.243    174.331   9.753  < 2e-16 ***
#  fires_des$index    3.371      1.258   2.679  0.00795 **
#  ---
#  Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
#  Residual standard error: 1293 on 218 degrees of freedom
#  Multiple R-squared:  0.03187,    Adjusted R-squared:  0.02743
#  F-statistic: 7.177 on 1 and 218 DF,  p-value: 0.007949
```

```r
plot(fires_des$index, fires_des$fires)
```

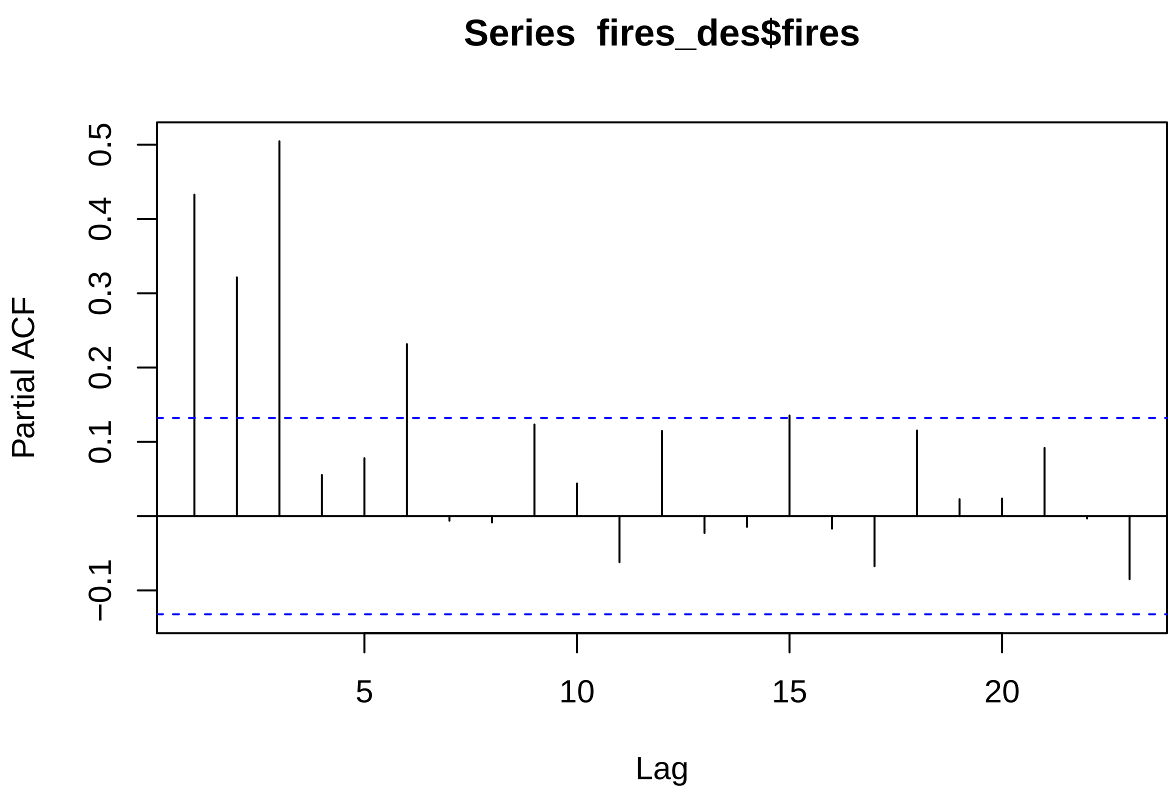


The summary statistics show significance in the p-values. However, let's go further and find a better model. Checking the ACF and PACF plots:

```r
library(lmtest)
acf(fires_des$fires, lag.max = 50)
```



```r
pacf(fires_des$fires)
```



The ACF plot shows spikes outside the confidence interval until lag 50. The PACF plot shows spikes at lags 2, 3 and 6. Try ARIMA models:

```r
arima101 <- arima(fires_des$fires, order = c(1,0,1))
coeftest(arima101)
```

```
#
#  z test of coefficients:
#
#               Estimate  Std. Error  z value  Pr(>|z|)
#  ar1        9.8985e-01  9.4727e-03 104.4953 < 2.2e-16 ***
#  ma1       -8.5720e-01  3.3574e-02 -25.5320 < 2.2e-16 ***
#  intercept  2.0051e+03  6.8924e+02   2.9091  0.003624 **
#  ---
#  Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
arima202 <- arima(fires_des$fires, order = c(2,0,2))
coeftest(arima202)
```

```
#
```

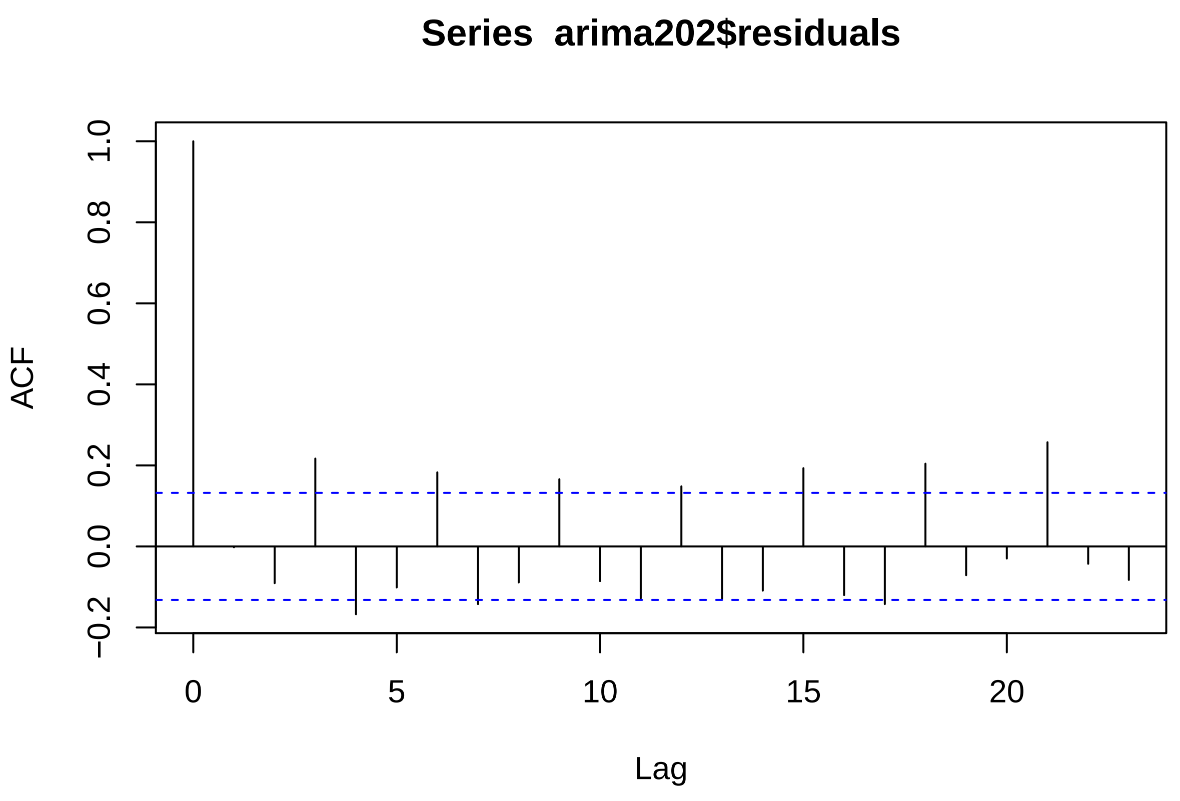
```
#  z test of coefficients:
#
#               Estimate Std. Error z value  Pr(>|z|)
#  ar1           1.09067    0.22323  4.8858 1.030e-06 ***
#  ar2          -0.10492    0.22099 -0.4748  0.634944
#  ma1          -1.11848    0.20859 -5.3621 8.228e-08 ***
#  ma2           0.29027    0.18014  1.6114  0.107099
#  intercept 1976.55544  635.98504  3.1079  0.001884 **
#  ---
#  Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The ARIMA(1,0,1) model seems significant but not the ARIMA(2,0,2). We implement differencing into our model:

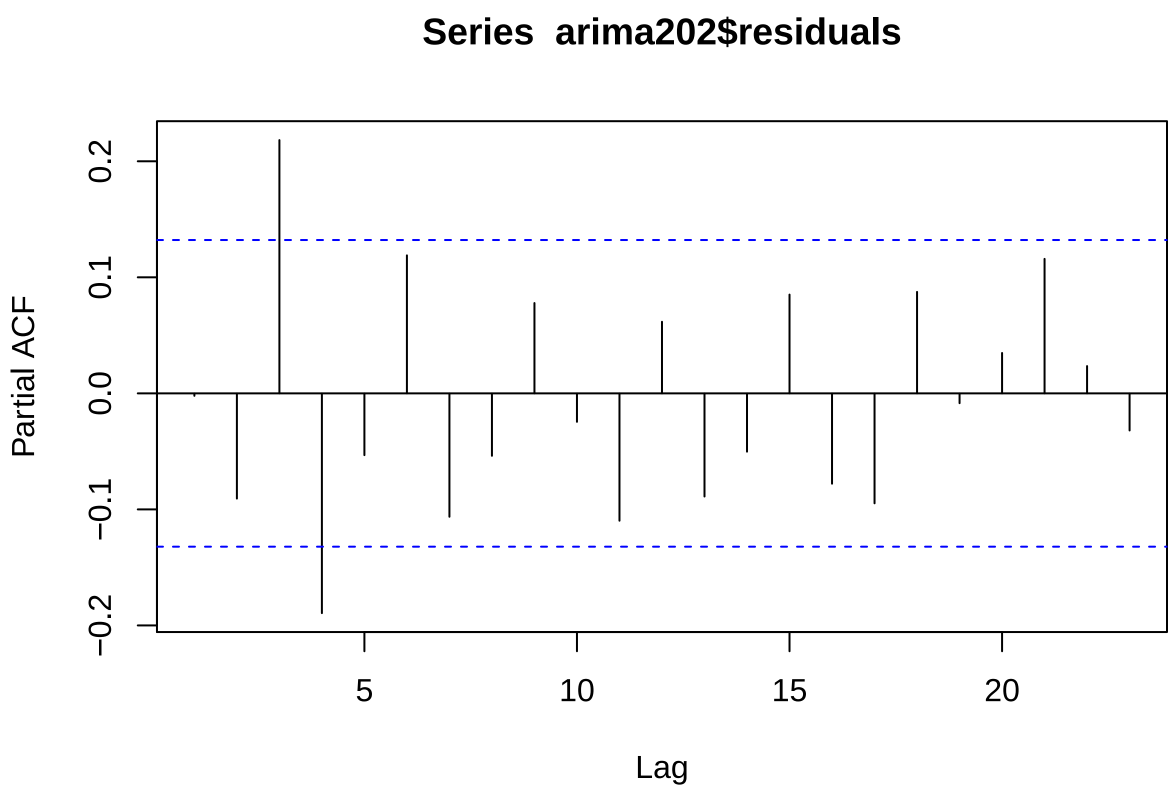
```r
arima111 <- arima(fires_des$fires, order = c(1,1,1))
coeftest(arima111)
```

```
#
#  z test of coefficients:
#
#       Estimate Std. Error  z value Pr(>|z|)
#  ar1 -0.176708   0.076624  -2.3062   0.0211 *
#  ma1 -0.817890   0.044377 -18.4307   <2e-16 ***
#  ---
#  Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
acf(arima202$residuals)
```



```
pacf(arima202$residuals)
```

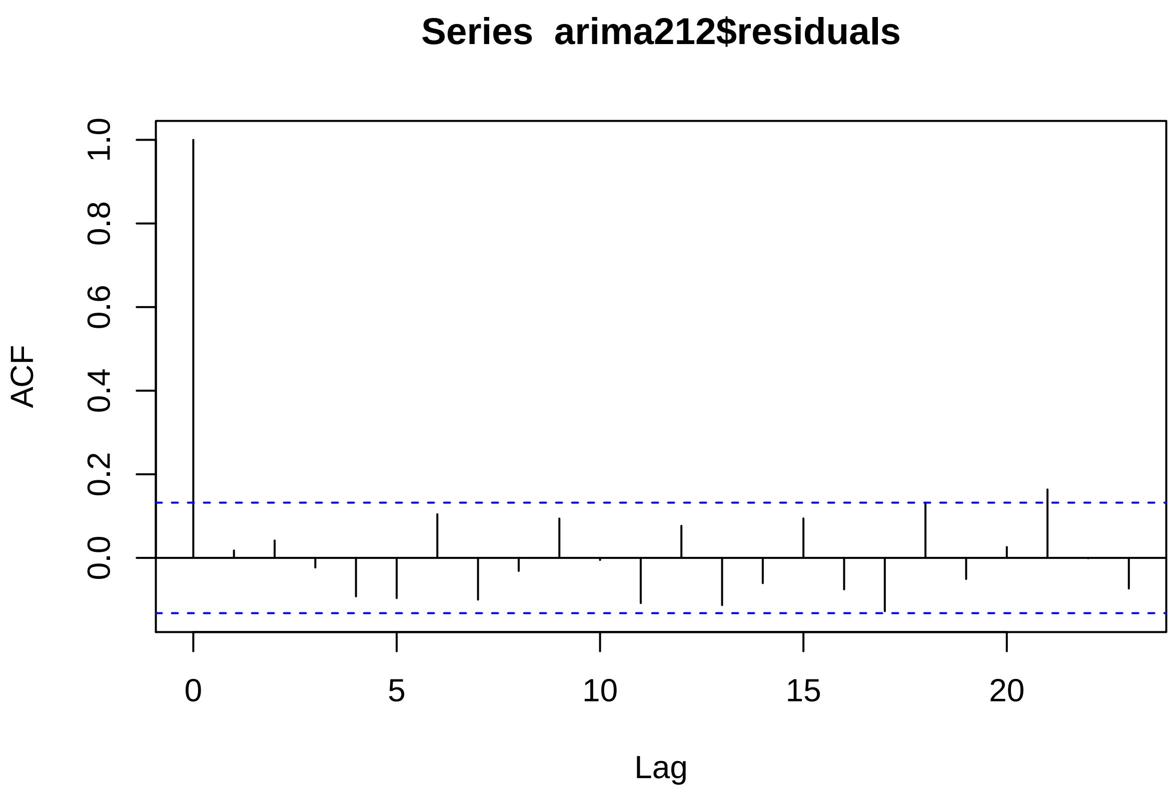


Although the coeftest function shows significance for our model, the ACF and PACF plots suggest we require more AR and MA:

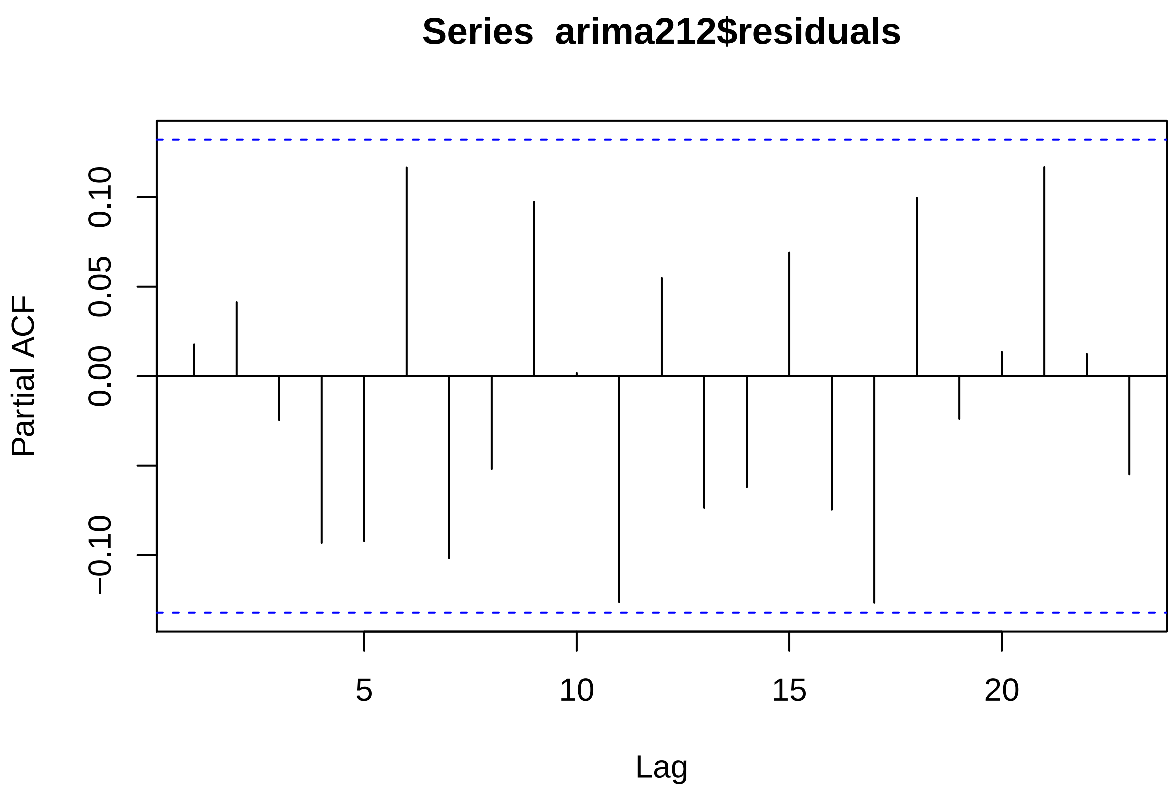
```
arima212 <- arima(fires_des$fires, order = c(2,1,2))
coeftest(arima212)
```

```
#
#  z test of coefficients:
#
#       Estimate Std. Error z value  Pr(>|z|)
#  ar1 -0.736739   0.123223 -5.9789 2.246e-09 ***
#  ar2 -0.388919   0.078585 -4.9490 7.459e-07 ***
#  ma1 -0.246865   0.125298 -1.9702   0.04881 *
#  ma2 -0.312528   0.116271 -2.6879   0.00719 **
#  ---
#  Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
acf(arima212$residuals)
```



```
pacf(arima212$residuals)
```

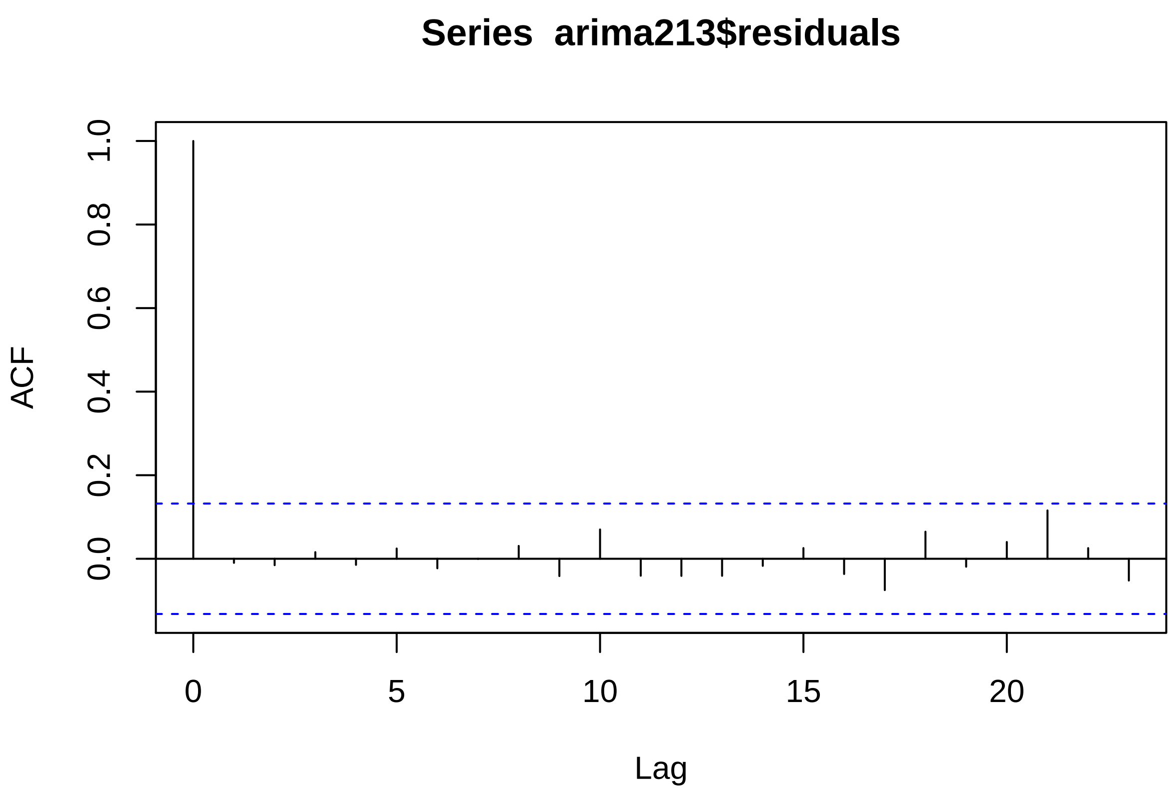


Now the coeftest shows significance and our PACF plot seems perfect! However the ACF plot suggests we need more MA:

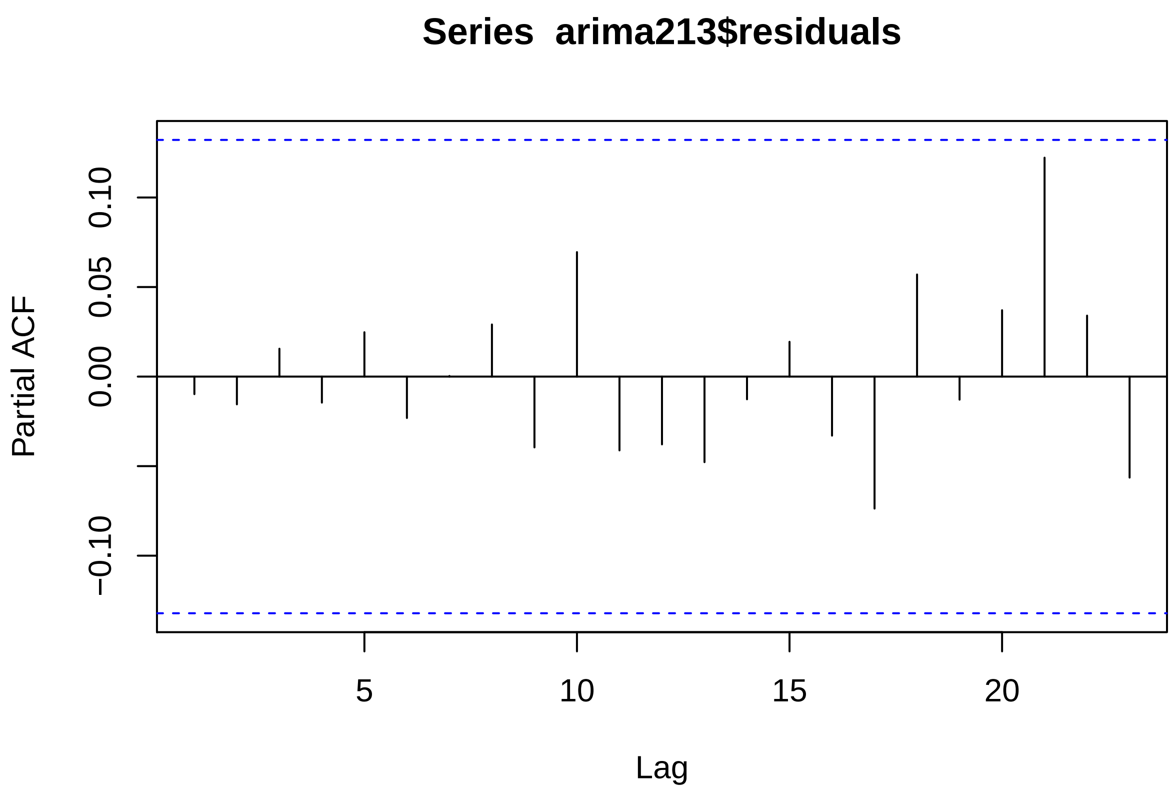
```
arima213 <- arima(fires_des$fires, order = c(2,1,3))
coeftest(arima213)
```

```
#
#  z test of coefficients:
#
#       Estimate Std. Error  z value Pr(>|z|)
#  ar1 -0.969974   0.030083 -32.2436  < 2e-16 ***
#  ar2 -0.966814   0.024951 -38.7481  < 2e-16 ***
#  ma1  0.045904   0.068188   0.6732  0.50082
#  ma2  0.158330   0.064658   2.4487  0.01434 *
#  ma3 -0.682569   0.058737 -11.6208  < 2e-16 ***
#  ---
#  Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
acf(arima213$residuals)
```



```
pacf(arima213$residuals)
```



```
fit_B <- arima213
```

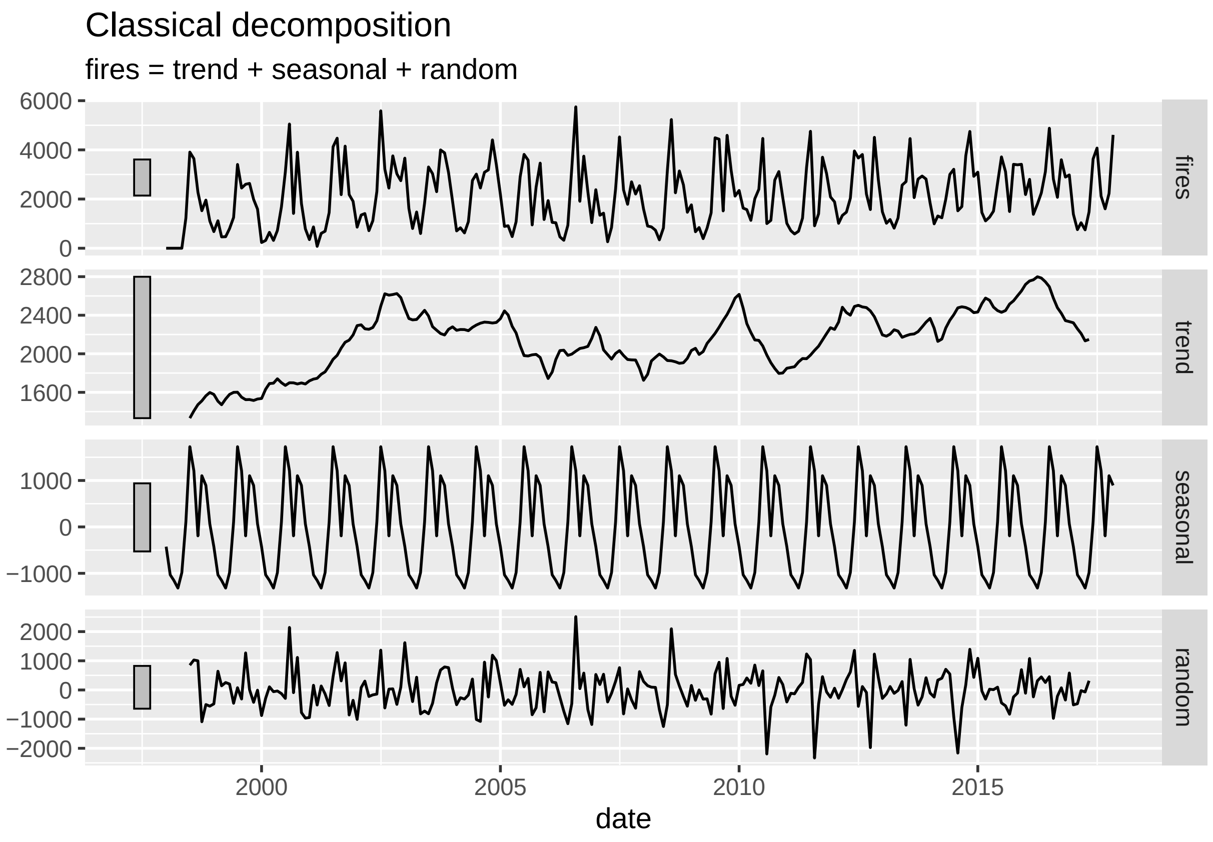
In our final model ARIMA(2,1,3), the ma1 term is insignificant however the other terms highly significant. Both the ACF and PACF plots are also within the confidence interval for all lag values. Call this model fitB.

3.1.1.2 Fable Decomposition Modelling

The aggregate amazon fire data will be decomposed in order to analyse the error terms of the time series model.

```
fires_all %>%
  classical_decomposition() %>%
  autoplot()
```

```
#  Model not specified, defaulting to automatic modelling of the `fires` variable. Override

#  Warning in as_dable.tbl_ts(dcmp, resp = !!sym(resp), method =
#  "Classical", : partial argument match of 'resp' to 'response'

#  Warning in as_dable.tbl_ts(out, method = attrs[["method"]], resp = !!
#  attrs[["response"]], : partial argument match of 'resp' to 'response'

#  Warning: Removed 6 rows containing missing values (geom_path).
```



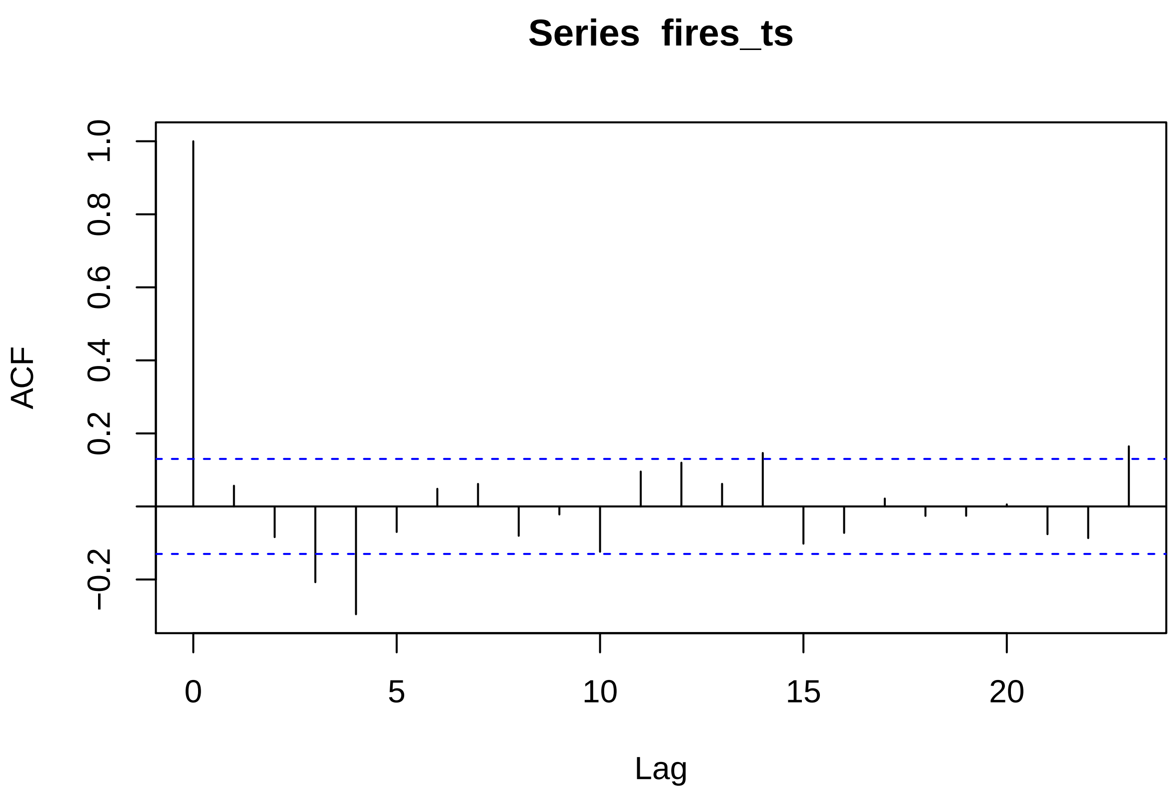
It can be seen that there is a strong seasonal component with two consistent peaks. The overall trend shows an overall increase, though it is not constantly increasing.

Now the random term can be examined in closer detail.

```
fires_decomp <- fires_all %>%
  classical_decomposition() %>%
  na.omit()
```

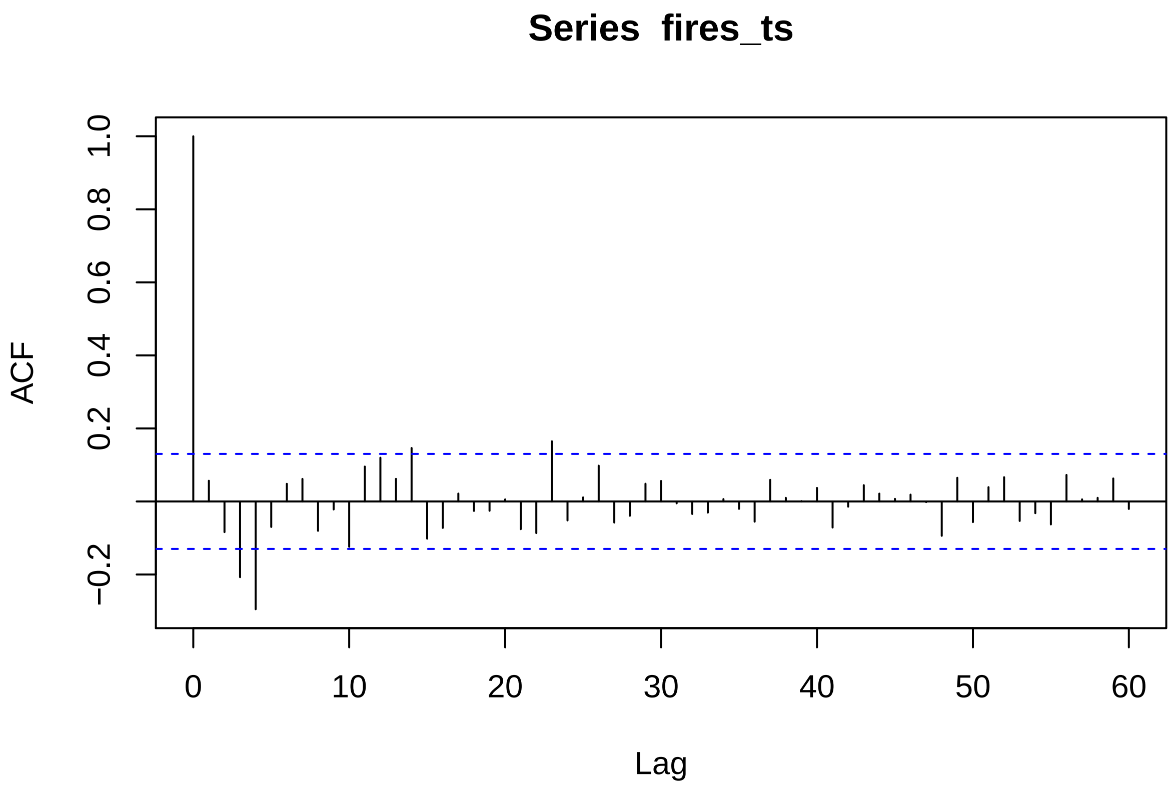
```
#  Model not specified, defaulting to automatic modelling of the `fires` variable. Override

#  Warning in as_dable.tbl_ts(dcmp, resp = !!sym(resp), method =
#  "Classical", : partial argument match of 'resp' to 'response'

#  Warning in as_dable.tbl_ts(out, method = attrs[["method"]], resp = !!
#  attrs[["response"]], : partial argument match of 'resp' to 'response'
```

```
fires_ts <- ts(fires_decomp$random)
acf(fires_ts)
```



The ACF of this data shows an alternating ACF, following what appears to follow a wave form shape. This shape has some peaks (both positive and negative), for a high number of lags (see 3, 4, 14, 23). It would be interesting to see this with an increased number of lags:

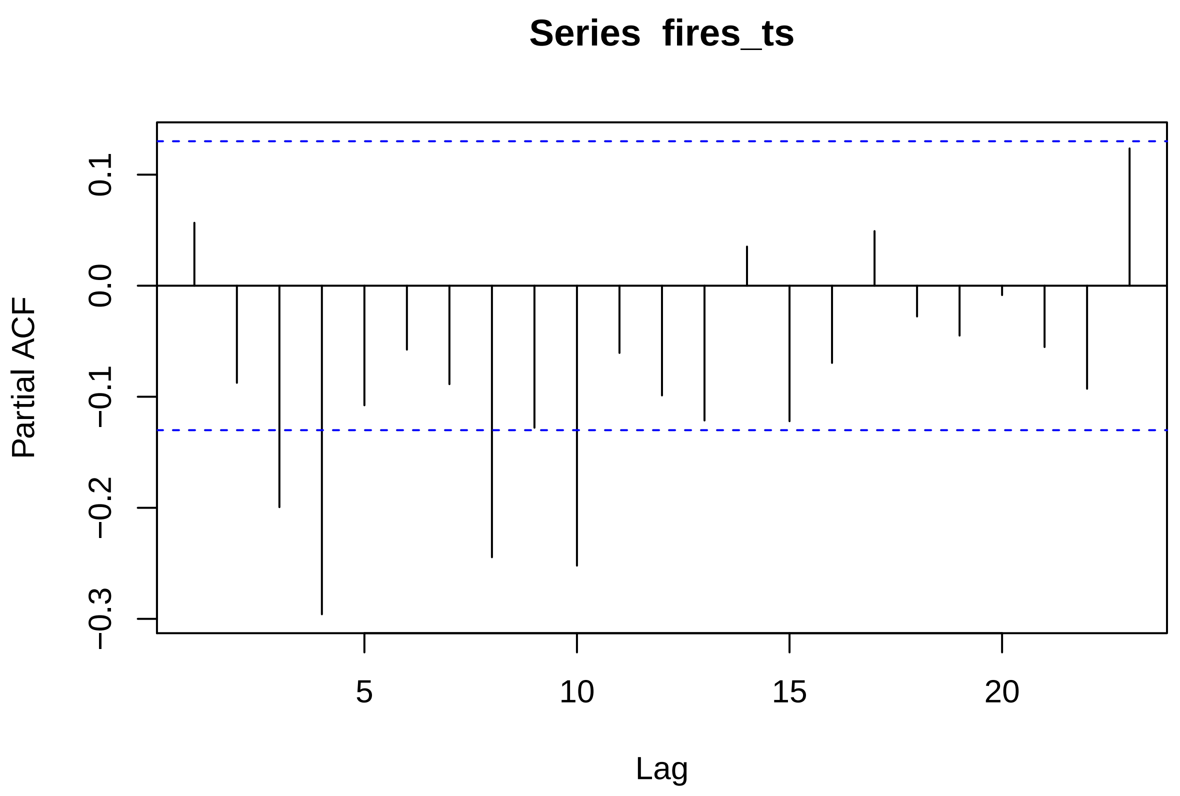
```
acf(fires_ts, lag.max = 60)
```



After displaying the ACF for up to 60 lags, it seems that this alternating wave pattern continues, but 23 is the last lag that has any significant influence.

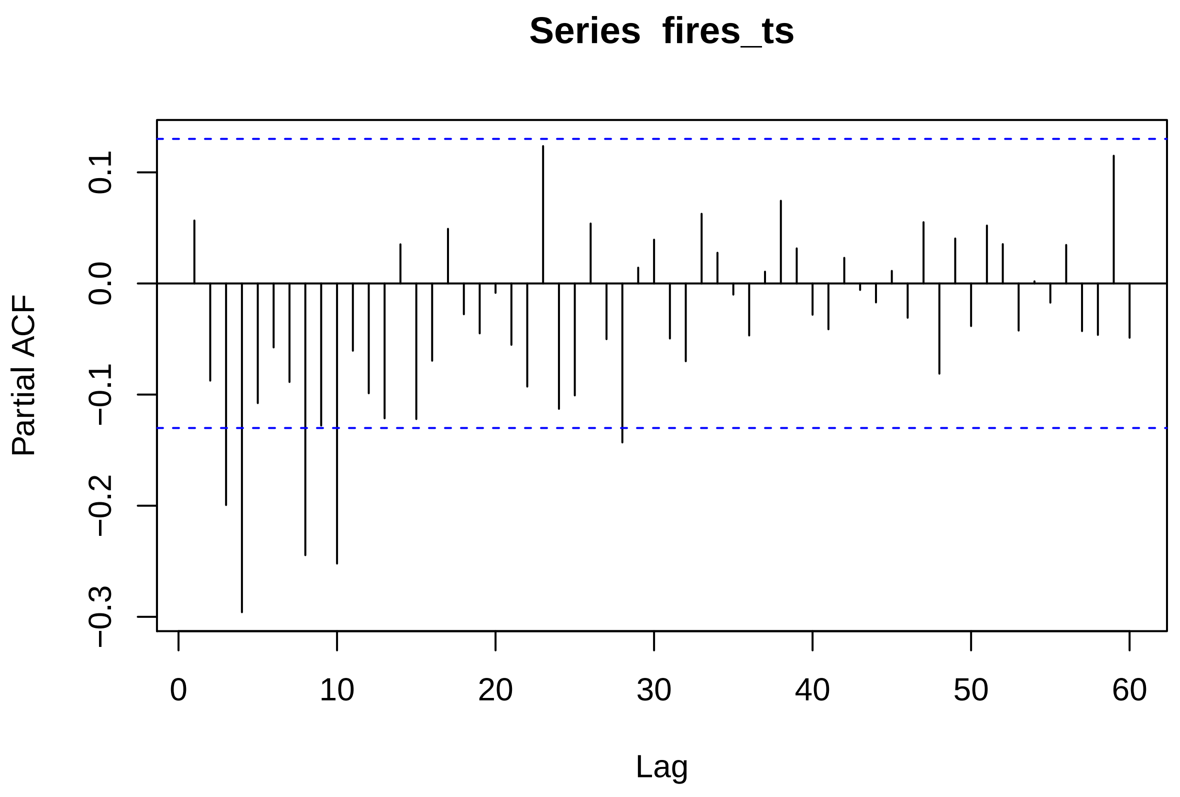
Now observing the PACF.

```
pacf(fires_ts)
```



The PACF funciton for this time series shows some significant lags at 3, 4, 8, 10 etc. This should be examined in the same way as the ACF (through viewing a larger number of lags):

```r
pacf(fires_ts, lag.max = 60)
```



The lags mentioned above seem to be the most influential, however, these spikes can be seen in the later lags as well. The overall pattern shows a diminishing alternating effect that trends towards an exaggerated version of the ACF function.

The results after looking at the ACF and PACF do not immediately point to a specific model, as there is a large number of lags that are influential and there seems to be something of a cyclic or seasonal effect in fucntion values.

Noting that there is not a model that is immediately obvious after observing both the PACF and ACF, the decomposed data will be modelled using a general framework. The `fable` framework, allows modelling on time series to be done while optimising the accuracy of results. This is done by allowing the values of the specified models to vary over a set range (for ARIMA it is generally 1:6). The best model is then selected.

```r
fires_decomp_models <- fires_decomp %>%
  rename(fires_original = fires, fires_var = random) %>%
  as_tsibble() %>%
  model(
    arima = ARIMA(fires_var),
    snaive = SNAIVE(fires_var)
  )
```

```
#  Using `date` as index variable.
```

```r
fires_decomp_models$arima
```

```
#  [1] ARIMA(2,0,0)(0,0,1)[12]
```

This creates a seasonal ARIMA model with parameters (2, 0, 0) (0, 0, 1).

A seasonal ARIMA model is formed by including additional seasonal terms in the ARIMA models:

$ARIMA\ (p, d, q)\ (P, D, Q)_m$

where (p, d, q) and $(P, D, Q)_m$ denotes the non-seasonal part and the seasonal part of the model, respectively. m denotes the number of observations per year.

The seasonal part of the model consists of terms that are similar to the non-seasonal components of the model, but involves backshifts of the seasonal period. The seasonal part of an AR or MA model will be seen in the seasonal lags of the PACF and ACF. For example, an $ARIMA\ (0,0,0)\ (0,0,1)_1 2$ model will show:

- a spike at lag 12 in the ACF but no other significant spikes;
- exponential decay in the seasonal lags of the PACF (that is, at lags 12, 24, 36, ...).

Similarly, an $ARIMA\ (0,0,0)\ (1,0,0)_1 2$ model will show:

- exponential decay in the seasonal lags of the ACF;
- a single significant spike at lag 12 in the PACF.

In considering the appropriate seasonal orders for a seasonal ARIMA model, attention is restricted to the seasonal lags. The modelling procedure is almost the samea s for non-seasonal data, except that it is needed to select seasonal AR and MA terms as well as the non-seasonal components of the model. [cite the website]

This also trains a seasonal naive model (random process with seasonal component). * This does not make sense?? if season has been removed why would this work?

3.1.1.3 Fable Automatic Modelling

- Fable can model the data and automatically add back in seasonality and trend, resulting in a wholistic forecasting model.
- Comment here that we are going to use the model created in previous section

```
fire_models <- fires_all %>%
  rename(fires_var = fires) %>%
  model(
    arima = ARIMA(fires_var),
    snaive = SNAIVE(fires_var) ,
    arima213 = ARIMA(fires_var ~ pdq(p = 2, d = 1, q = 3) + PDQ(P = 0, D = 0, Q = 0))
  )
```

3.1.2 Forecasting

3.1.2.1 Forecasting Manual

Using out two models (fitA and fitB), the fires count for December 2017 has been forecasted:

```
prediction_A <- (1700.243 + 3.371*240)*m_alt[1]
prediction_A
```

```
#  [1] 2109.442
```

```
prediction_B <- forecast(fit_B, h = 1)
prediction_B
```

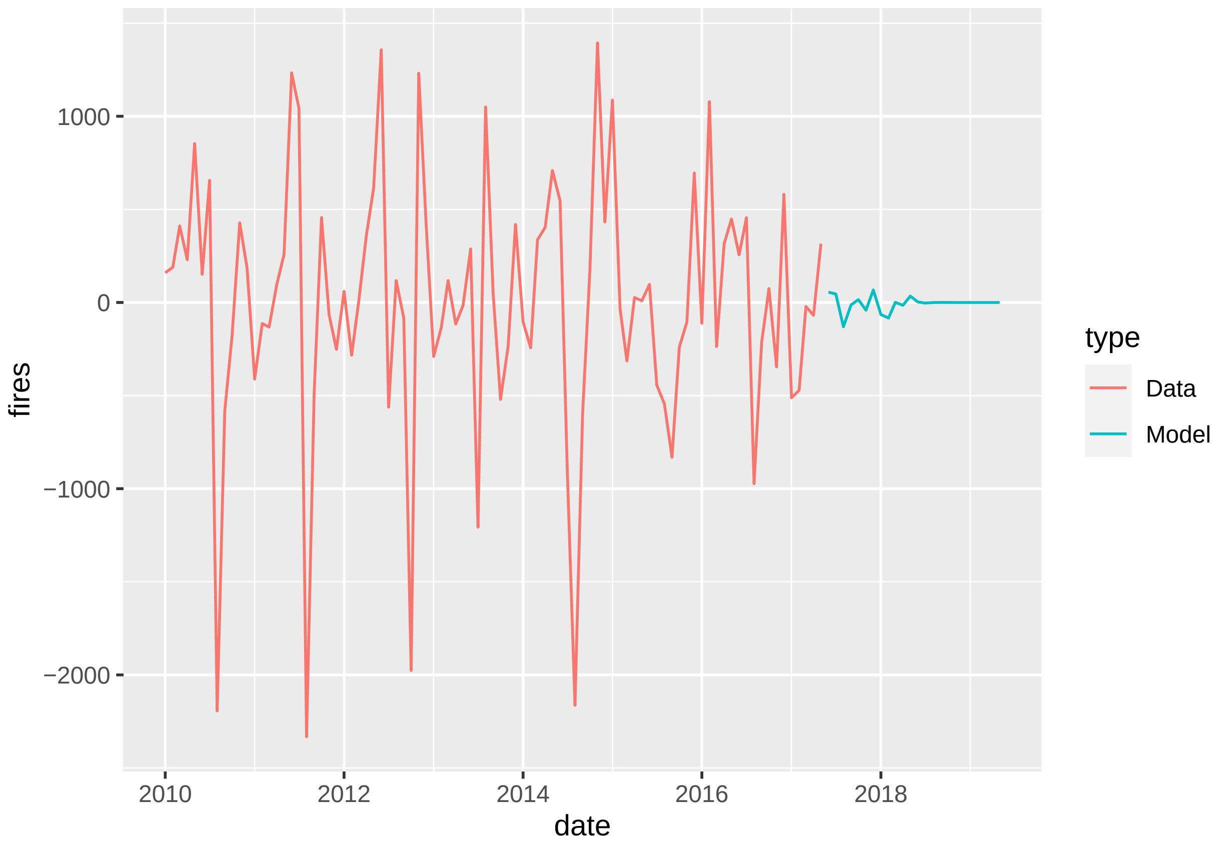
```
#      Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
#  221       1915.295 785.3757 3045.213 187.2328 3643.356
```

fitA forecasted the fires count to be roughly 2222.30 which is within the 95% prediction interval forecasted using fitB. The point forecast using fitB was roughly 1915.30.

```
fires_decomp_forecast <- fires_decomp_models %>%
  fabletools::forecast(h = "2 years")
```

```
#  Warning in as_fable.tbl_ts(fbl, resp = object$response, dist = !!
#  sym(".distribution")): partial argument match of 'resp' to 'response'

#  Warning in as_fable.tbl_ts(fbl, resp = object$response, dist = !!
#  sym(".distribution")): partial argument match of 'dist' to 'distribution'

#  Warning in as_fable.tbl_ts(fbl, resp = object$response, dist = !!
#  sym(".distribution")): partial argument match of 'resp' to 'response'

#  Warning in as_fable.tbl_ts(fbl, resp = object$response, dist = !!
#  sym(".distribution")): partial argument match of 'dist' to 'distribution'

#  Warning in as_fable.tbl_ts(out, resp = fbl_attr$response, dist = !!
#  fbl_attr$dist): partial argument match of 'resp' to 'response'

#  Warning in as_fable.tbl_ts(out, resp = fbl_attr$response, dist = !!
#  fbl_attr$dist): partial argument match of 'dist' to 'distribution'
```

```
forecast_clean <- fires_decomp_forecast %>%
  filter(.model == "arima") %>%
  as_tibble() %>%
  select(date, fires = fires_var) %>%
  mutate(type = "Model")


fires_decomp %>%
  select(- fires) %>%
  select(date, fires = random) %>%
  filter(date >= lubridate::ymd("20100101")) %>%
  mutate(type = "Data") %>%
  bind_rows(forecast_clean) %>%
  ggplot(aes(x = date,  y = fires, colour = type)) +
  geom_line()
```



It can be seen that this model trends towards 0 and has little variation comparatively to the original data.

3.1.2.2 Fable Automatic Forecasting

```r
forecast_2 <- fire_models %>%
  fabletools::forecast(h = "2 years")
```

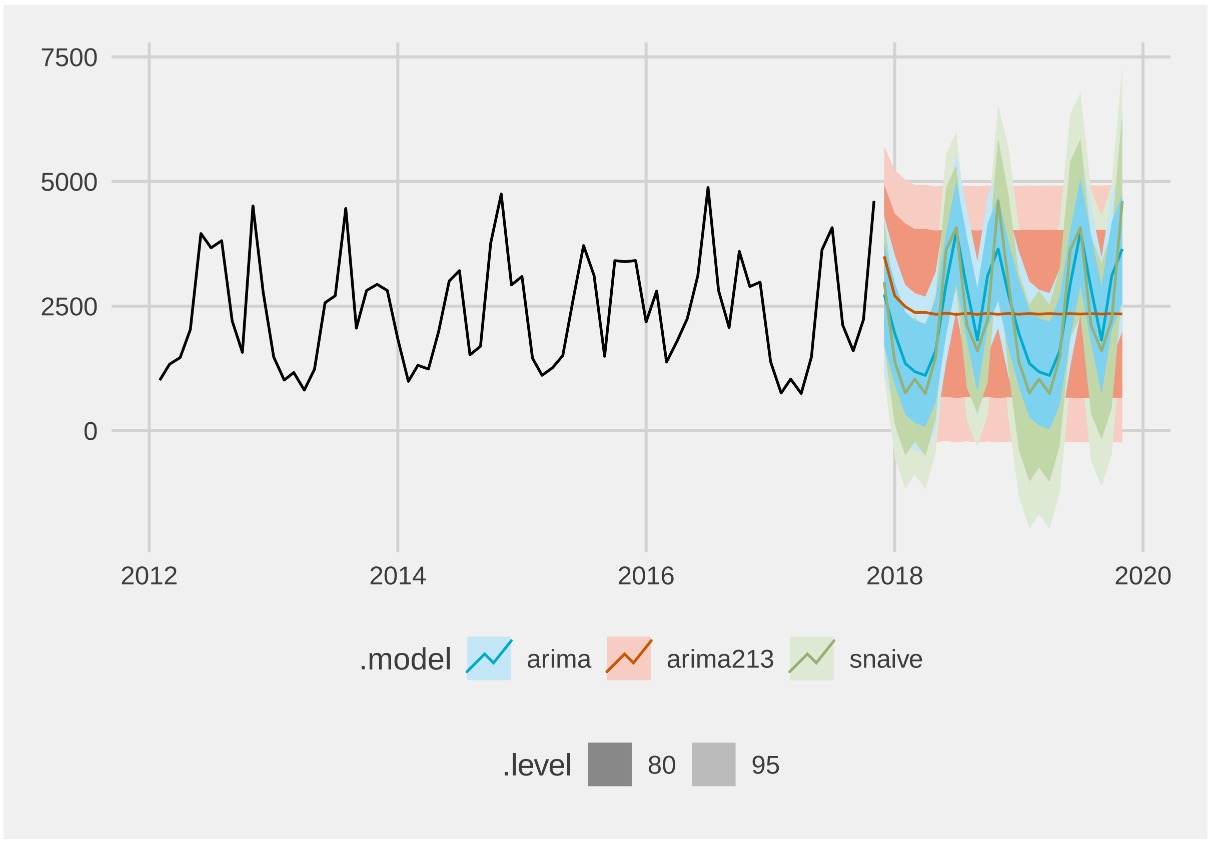
```
#  Warning in as_fable.tbl_ts(fbl, resp = object$response, dist = !!
#  sym(".distribution")): partial argument match of 'resp' to 'response'

#  Warning in as_fable.tbl_ts(fbl, resp = object$response, dist = !!
#  sym(".distribution")): partial argument match of 'dist' to 'distribution'

#  Warning in as_fable.tbl_ts(fbl, resp = object$response, dist = !!
#  sym(".distribution")): partial argument match of 'resp' to 'response'

#  Warning in as_fable.tbl_ts(fbl, resp = object$response, dist = !!
#  sym(".distribution")): partial argument match of 'dist' to 'distribution'

#  Warning in as_fable.tbl_ts(fbl, resp = object$response, dist = !!
#  sym(".distribution")): partial argument match of 'resp' to 'response'

#  Warning in as_fable.tbl_ts(fbl, resp = object$response, dist = !!
#  sym(".distribution")): partial argument match of 'dist' to 'distribution'

#  Warning in as_fable.tbl_ts(out, resp = fbl_attr$response, dist = !!
#  fbl_attr$dist): partial argument match of 'resp' to 'response'

#  Warning in as_fable.tbl_ts(out, resp = fbl_attr$response, dist = !!
#  fbl_attr$dist): partial argument match of 'dist' to 'distribution'
```

```r
forecast_5 <- fire_models %>%
  fabletools::forecast(h = "5 years")
```

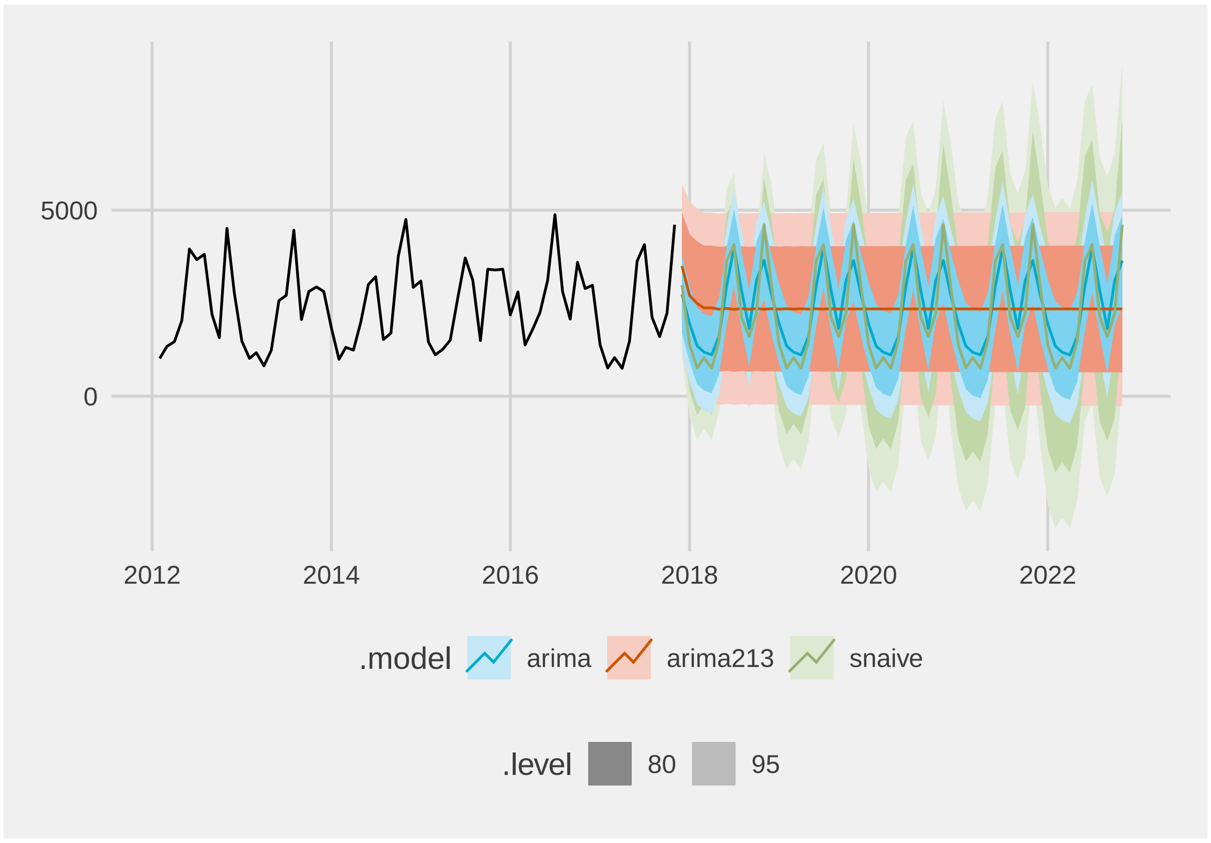
```
#  Warning in as_fable.tbl_ts(fbl, resp = object$response, dist = !!
#  sym(".distribution")): partial argument match of 'resp' to 'response'

#  Warning in as_fable.tbl_ts(fbl, resp = object$response, dist = !!
#  sym(".distribution")): partial argument match of 'dist' to 'distribution'
#  Warning in as_fable.tbl_ts(fbl, resp = object$response, dist = !!
#  sym(".distribution")): partial argument match of 'resp' to 'response'

#  Warning in as_fable.tbl_ts(fbl, resp = object$response, dist = !!
#  sym(".distribution")): partial argument match of 'dist' to 'distribution'

#  Warning in as_fable.tbl_ts(fbl, resp = object$response, dist = !!
#  sym(".distribution")): partial argument match of 'resp' to 'response'

#  Warning in as_fable.tbl_ts(fbl, resp = object$response, dist = !!
#  sym(".distribution")): partial argument match of 'dist' to 'distribution'

#  Warning in as_fable.tbl_ts(out, resp = fbl_attr$response, dist = !!
#  fbl_attr$dist): partial argument match of 'resp' to 'response'

#  Warning in as_fable.tbl_ts(out, resp = fbl_attr$response, dist = !!
#  fbl_attr$dist): partial argument match of 'dist' to 'distribution'
```

```
forecast_2 %>%
  autoplot(filter(rename(fires_all, fires_var = fires), date > lubridate::ymd("2012-01-01"))
  ggthemes::scale_color_fivethirtyeight() +
  ggthemes::theme_fivethirtyeight()
```

```
#  Warning in `$.data.frame`(data, "col"): Partial match of 'col' to 'colour'
#  in data frame

#  Warning in `$.data.frame`(data, "col"): Partial match of 'col' to 'colour'
#  in data frame


#  Warning in `$.data.frame`(data, "col"): Partial match of 'col' to 'colour'
#  in data frame


#  Warning in `$.data.frame`(data, "col"): Partial match of 'col' to 'colour'
#  in data frame


#  Warning in `$.data.frame`(data, "col"): Partial match of 'col' to 'colour'
#  in data frame
#  Warning in `$.data.frame`(data, "col"): Partial match of 'col' to 'colour'
#  in data frame
```



```r
forecast_5 %>%
  autoplot(filter(rename(fires_all, fires_var = fires), date > lubridate::ymd("2012-01-01"))
  ggthemes::scale_color_fivethirtyeight() +
  ggthemes::theme_fivethirtyeight()
```



It can be seen that the arima and snaive models stay relatively close together, though the prediction interval for arima seems to be constant with time whereas the snaive model is increasing with time. The prediciton interval of the arima213 model exhibits the same consistent and tight behaviour as the seasonal arima. However, the model itself trends towards the mean of the overall time series.

3.1.3 Forecast Validation and Testing

- Use data up until a certain point to use as training data.

Frist lets have a look at the time range for the fires data.

```
fires_all %>%
  pull(date) %>%
  range()
```

```
#  [1] "1998 Jan" "2017 Nov"
```

It goes from January of 1998 to November of 2017. In the previous examples, 2 years were forecast in order to look at the modelling process. This same amount of time will be forecast to assess the models.

Before assessing the models, the training data needs to be obtained by filtering anything after November 2015.

```
fires_train <- fires_all %>%
  filter(date <= lubridate::ymd("2015-11-01"))

fires_test <- fires_all %>%
  filter(date > lubridate::ymd("2015-11-01"))
```

The models will be trained on this data.

```
train_models <- fires_train %>%
  rename(fires_var = fires) %>%
  model(
    arima = ARIMA(fires_var),
    snaive = SNAIVE(fires_var)
  )
```

The remaining data will now be forecast for. The predicted data will be stored in a results object, which will be compared against the gruondtruth to obtain an accuracy metric.

```
forecast_data <- train_models %>%
  fabletools::forecast(h = "2 years")
```

```
#  Warning in as_fable.tbl_ts(fbl, resp = object$response, dist = !!
#  sym(".distribution")): partial argument match of 'resp' to 'response'

#  Warning in as_fable.tbl_ts(fbl, resp = object$response, dist = !!
#  sym(".distribution")): partial argument match of 'dist' to 'distribution'

#  Warning in as_fable.tbl_ts(fbl, resp = object$response, dist = !!
#  sym(".distribution")): partial argument match of 'resp' to 'response'

#  Warning in as_fable.tbl_ts(fbl, resp = object$response, dist = !!
#  sym(".distribution")): partial argument match of 'dist' to 'distribution'

#  Warning in as_fable.tbl_ts(out, resp = fbl_attr$response, dist = !!
#  fbl_attr$dist): partial argument match of 'resp' to 'response'

#  Warning in as_fable.tbl_ts(out, resp = fbl_attr$response, dist = !!
#  fbl_attr$dist): partial argument match of 'dist' to 'distribution'
```

```
arima_prediction <- forecast_data %>%
  filter(.model == "arima")

snaive_prediction <- forecast_data %>%
  filter(.model == "snaive")
```

```
arima_mse <- mean((fires_test$fires - arima_prediction$fires_var)^2)
snaive_mse <- mean((fires_test$fires - snaive_prediction$fires_var)^2)
glue::glue("The mean squared error of ARIMA is:  {arima_mse}
           The mean squared error of SNAIVE is: {snaive_mse}")
```

```
#  The mean squared error of ARIMA is:  502092.248833118
#  The mean squared error of SNAIVE is: 613217.855245167
```

3.2 Spatial Analysis

4 Disicussion

4.1 Temporal

- MSE comparison to actual statistics
- Reflect on arima vs seasonal arima
- State-wise time series analysis

4.2 Spatial

5 References

https://otexts.com/fpp2/seasonal-arima.html