STAT 221 Project

John Rapp Farnes | 405461225

3/5 2020

TODO

- Prediction på known värden
- Tillbaka trend på pred
- Kommentera stora residualer? datum?
- Log: större skillnad större värden, plotta
- Periodogram residualer

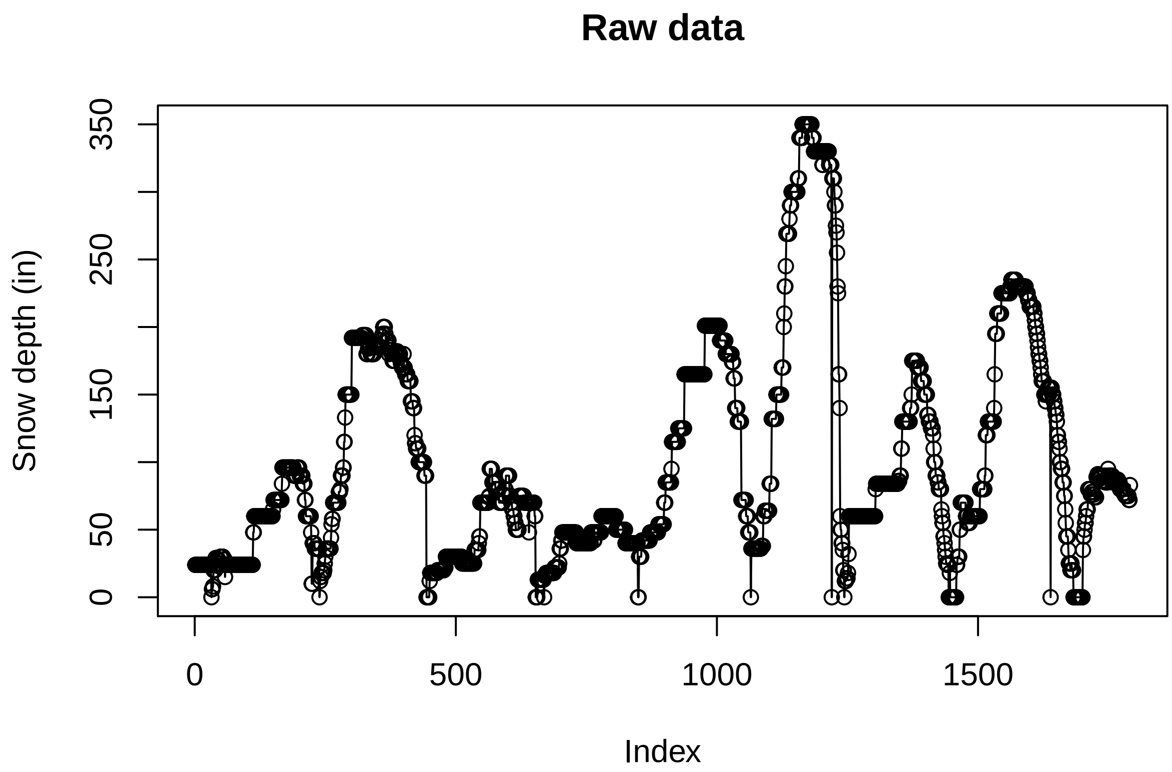
```
library(astsa)
library(imputeTS)
library(rjson)
library(forecast)
```

```
##
## Attaching package: 'forecast'

## The following object is masked from 'package:astsa':
##
##     gas
```

```
library(tseries)
```

```
##
## Attaching package: 'tseries'

## The following object is masked from 'package:imputeTS':
##
##     na.remove
```

```
data <- fromJSON(file = "./mammoth.json")
plot(data$depth, type="o", main="Raw data", ylab="Snow depth (in)")
```




```
data$dates <- as.Date(data$dates, "%Y-%m-%d")

all_dates <- seq(data$dates[1], data$dates[length(data$dates)], by = "day")

length(data$dates) / length(all_dates)
```

```
## [1] 0.5940299
```

```
all_depths <- rep(NA, length(all_dates))

for (date in intersect(data$dates, all_dates)) {
  all_depths[which(all_dates == date)] <- data$depth[which(data$dates == date)[1]]
}
```

```
table(as.numeric(format.Date(data$date, "%m")), data$depth == 0)
```

```
##
##      FALSE TRUE
##   1    274    0
##   2    247    0
##   3    241    0
##   4    226    0
##   5    177    7
##   6     57    3
```

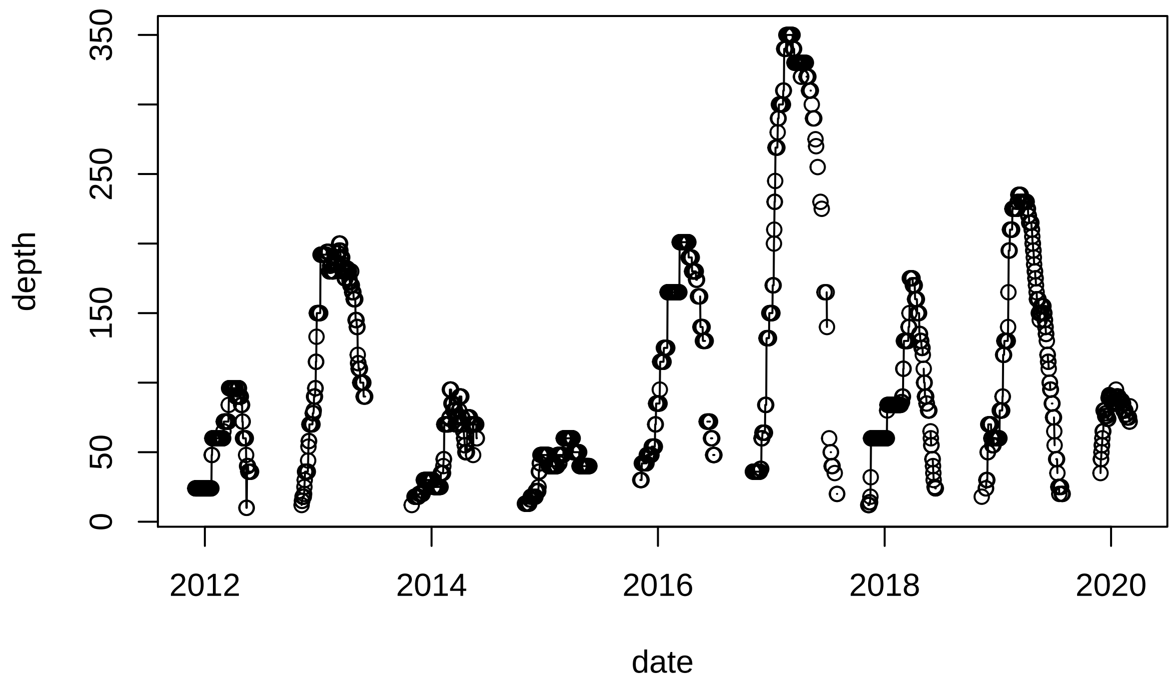
```
##   7     25    3
##   8      1    3
##   10     4    2
##   11   182   33
##   12   306    0
```

```r
all_depths[which(all_depths == 0)] <- NA
```

```r
data <- data.frame(date = all_dates, depth = all_depths)

nrow(data)
```

```
## [1] 3015
```

```r
plot(data, type="o")
```

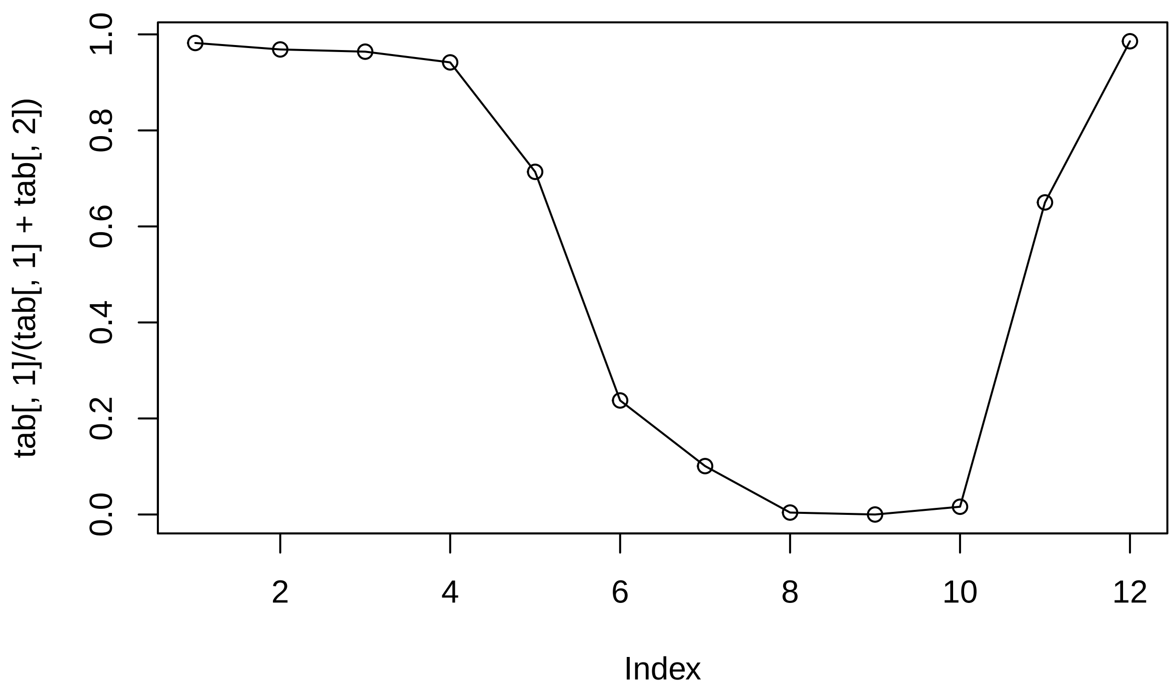



```r
(tab <- table(as.numeric(format.Date(data$date, "%m")), is.na(data$depth)))
```

```
##
##       FALSE TRUE
##   1     274    5
##   2     247    8
##   3     241    9
##   4     226   14
```

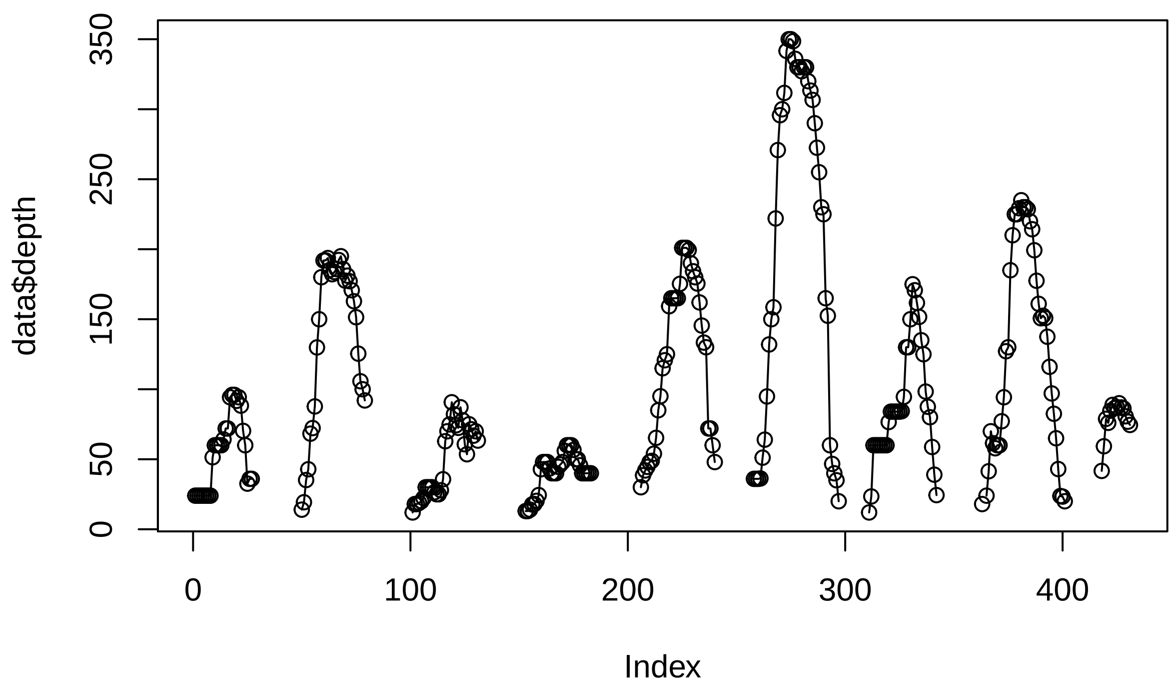
```
##    5     177    71
##    6      57   183
##    7      25   223
##    8       1   247
##    9       0   240
##    10      4   244
##    11    156    84
##    12    275     4
```

```
plot(tab[, 1] / (tab[, 1] + tab[, 2]), type="o")
```




```
mean_ignore_na <- function(x) {
  non_na <- x[which(!is.na(x))]
  return(ifelse(length(non_na) == 0, NA, mean(non_na)))
}

first_sat <- 3
last_sat <- length(data$date) - 2

week_length <- 7
nr_weeks <- (last_sat - first_sat) / week_length

week_start <- data$date[first_sat]
week_end <- data$date[last_sat]
```

```r
weekly_depths <- mean_ignore_na(data$depth[1:first_sat])

# for(i in seq(from = first_sat, to = last_sat, by = week_length)) {
for (i in 2:(nr_weeks + 1)) {
  sunday <- (first_sat + 1) + (i - 2)*week_length
  saturday <- sunday + 6

  weekly_depths[i] <- mean_ignore_na(data$depth[sunday:saturday])
}

weeks <- seq(data$date[first_sat], data$date[last_sat], by=7)

data <- data.frame(
  depth = weekly_depths,
  date = weeks
)

plot(data$depth, type="o")
```



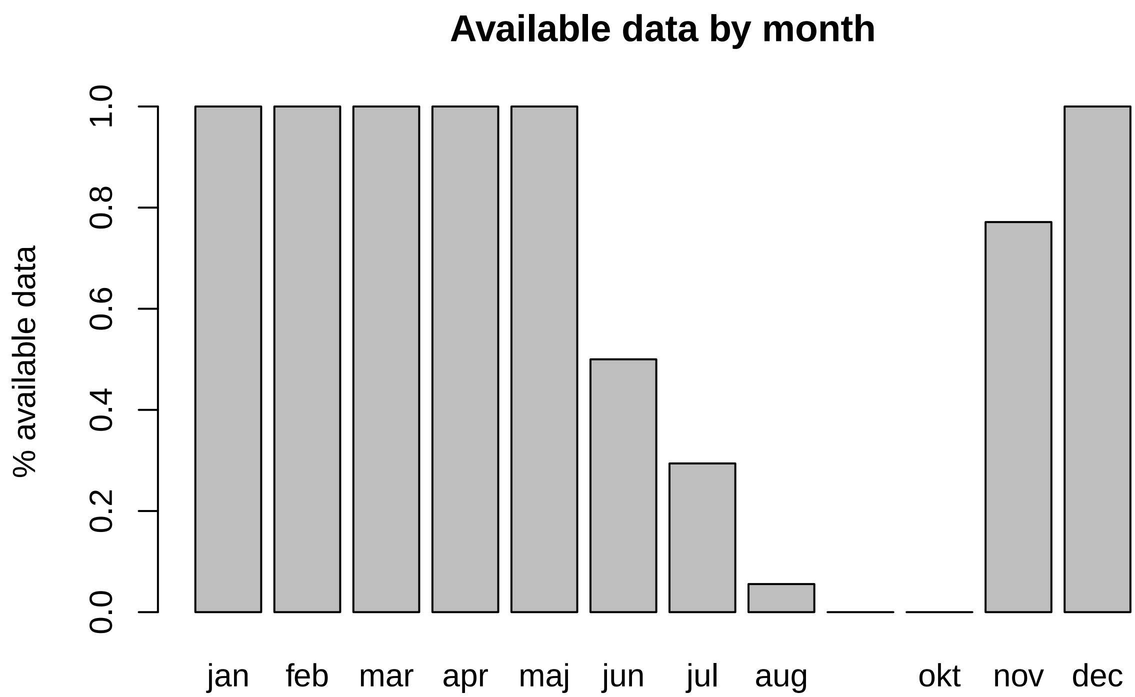
Missing values

```r
data$depth[which(is.na(data$depth))] <- 0
```

```
(tab <- table(as.numeric(format.Date(data$date, "%m")), data$depth == 0))
```

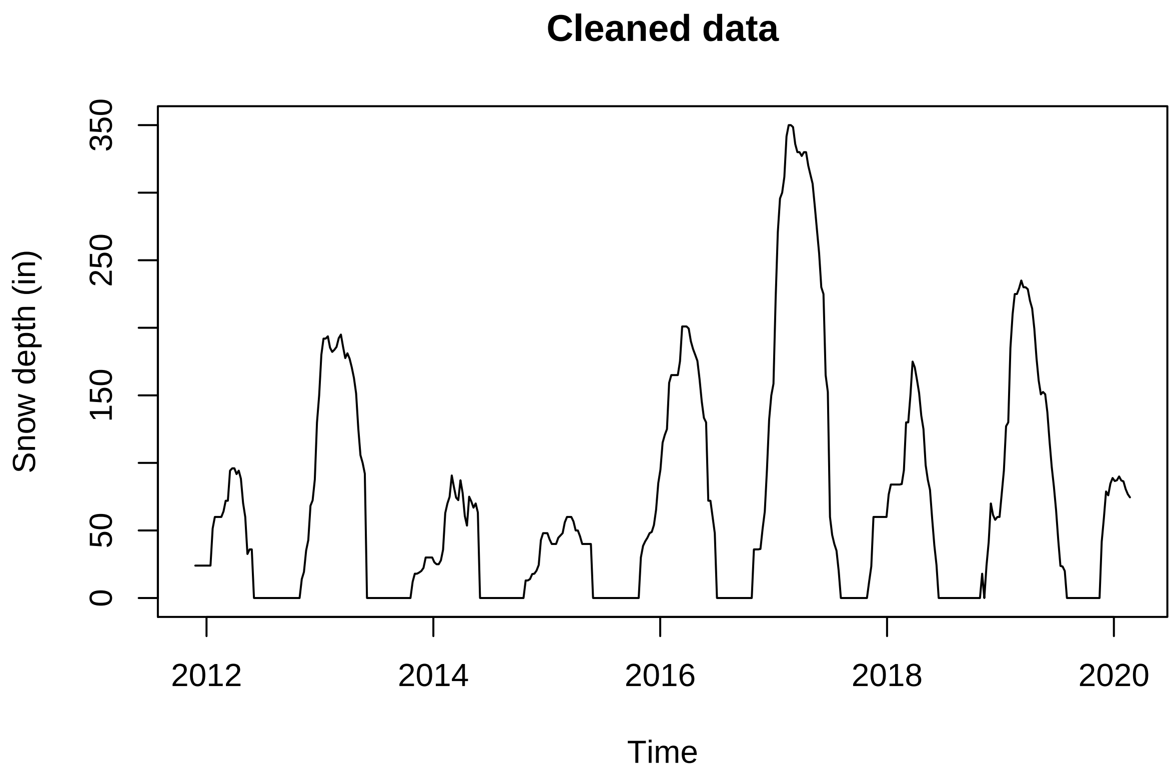
```
##
##      FALSE TRUE
##    1    38    0
##    2    37    0
##    3    37    0
##    4    34    0
##    5    34    0
##    6    18   18
##    7    10   24
##    8     2   34
##    9     0   35
##    10    0   34
##    11   27    8
##    12   41    0
```

```
month.labels <- format(ISOdatetime(2000,1:12,1,0,0,0),"%b")
barplot(tab[, 1] / (tab[, 1] + tab[, 2]), names.arg = month.labels, ylab="% available data", main="Avail
```



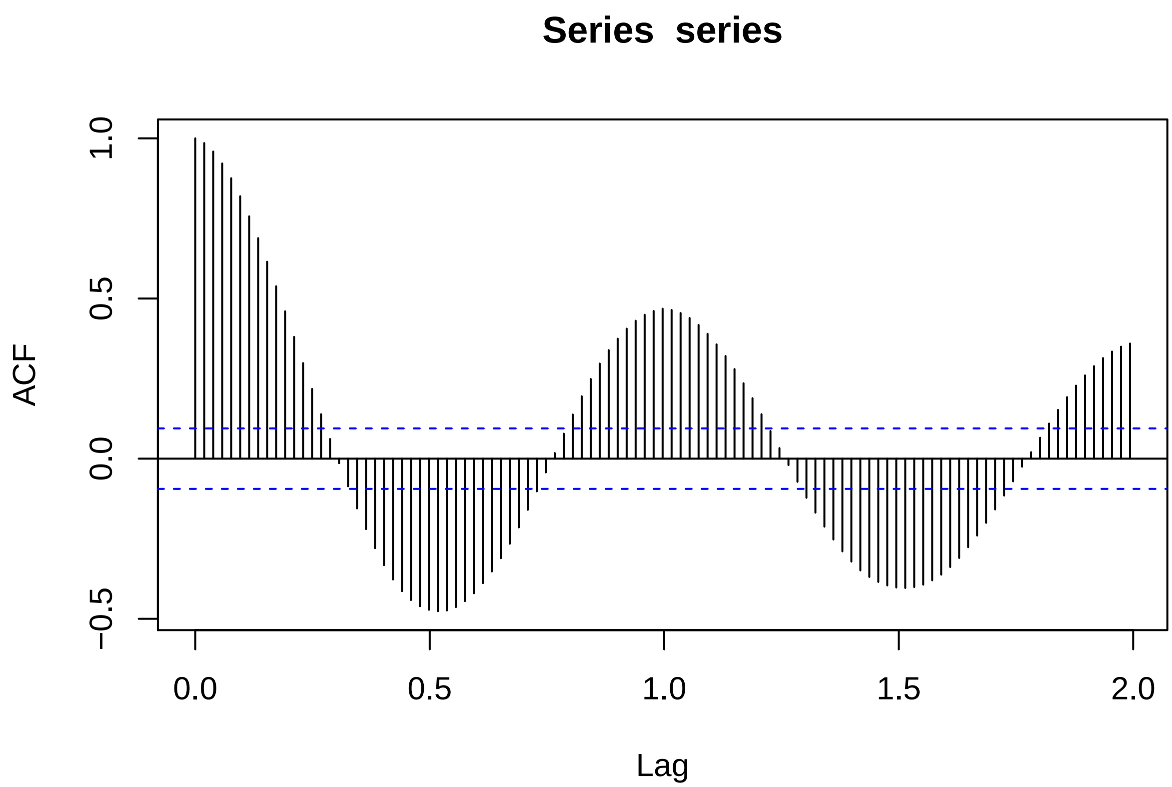

```
series <- ts(
  data$depth,
  start=c(as.numeric(format(data$date[1], "%Y")), as.numeric(format(data$date[1], "%U"))),
  # end=c(as.numeric(format(data$date[length(data$date)], "%Y")), as.numeric(format(data$date[length(da
```

```r
  frequency=365.25/7
)
plot(series, main="Cleaned data", ylab="Snow depth (in)")
```

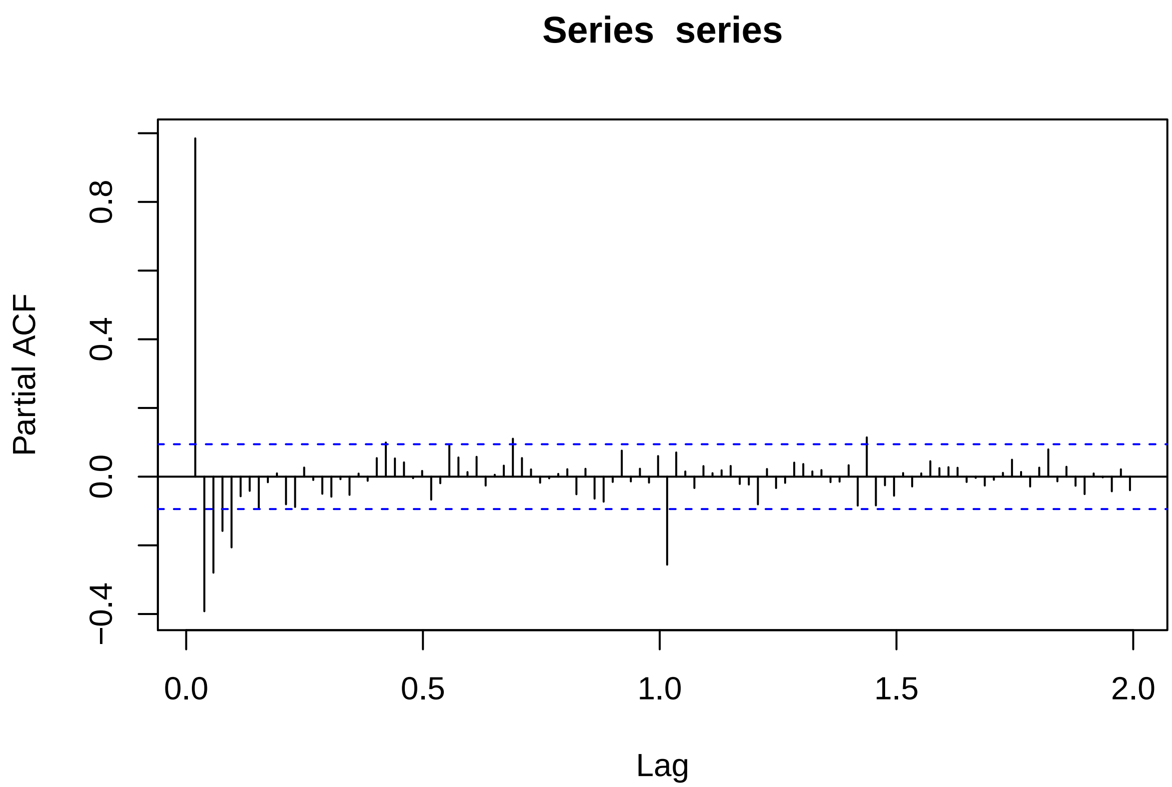


ACF

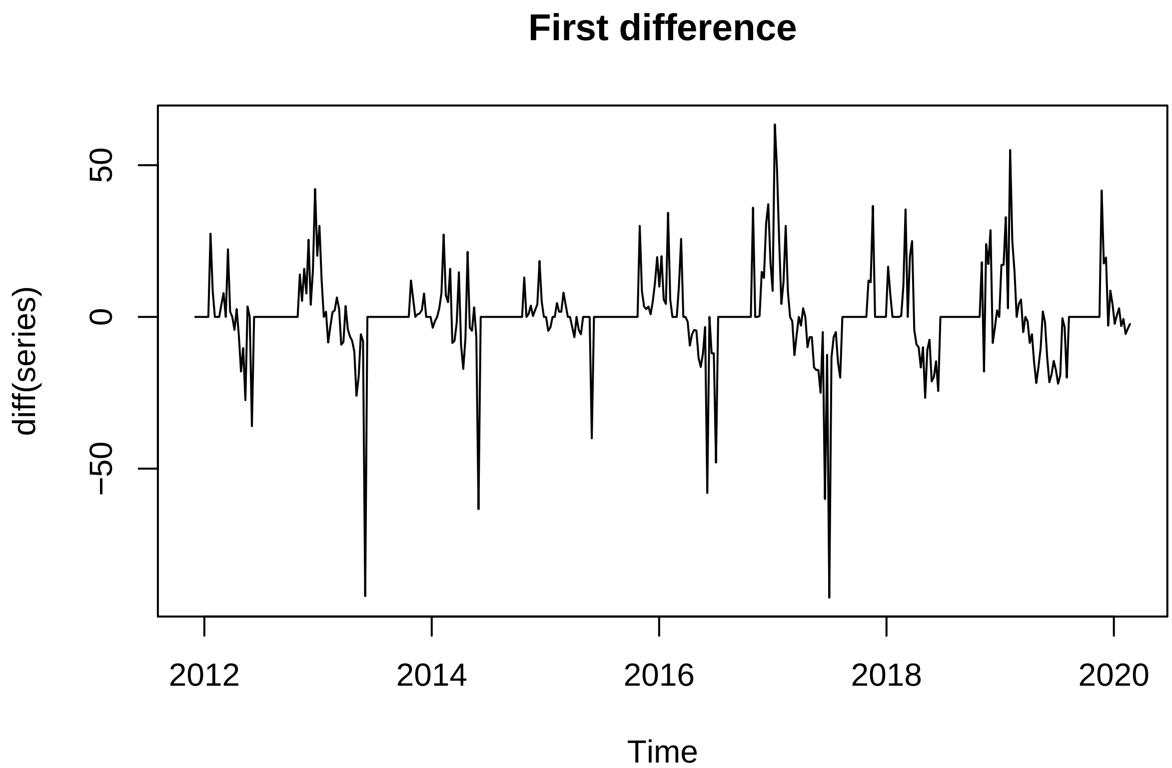
```r
# acf(na_interpolation(series, option = "linear"), lag.max = 52 * 2)
# pacf(na_interpolation(series, option = "linear"), lag.max = 52 * 2)

acf(series, lag.max = 52 * 2)
```



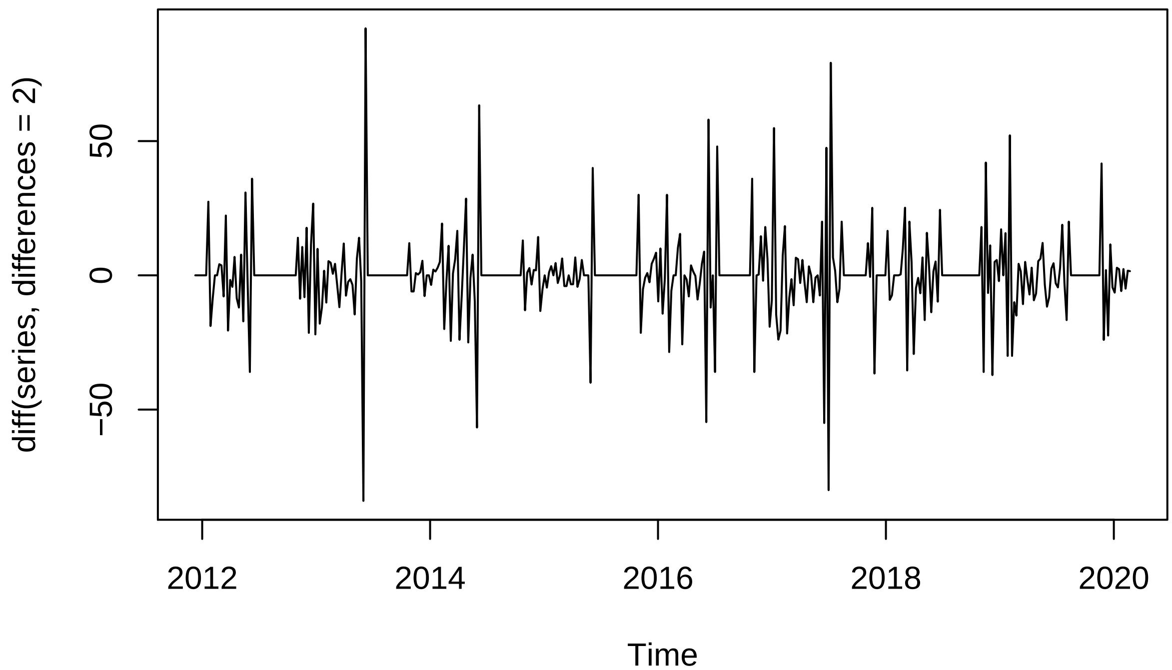

```
pacf(series, lag.max = 52 * 2)
```



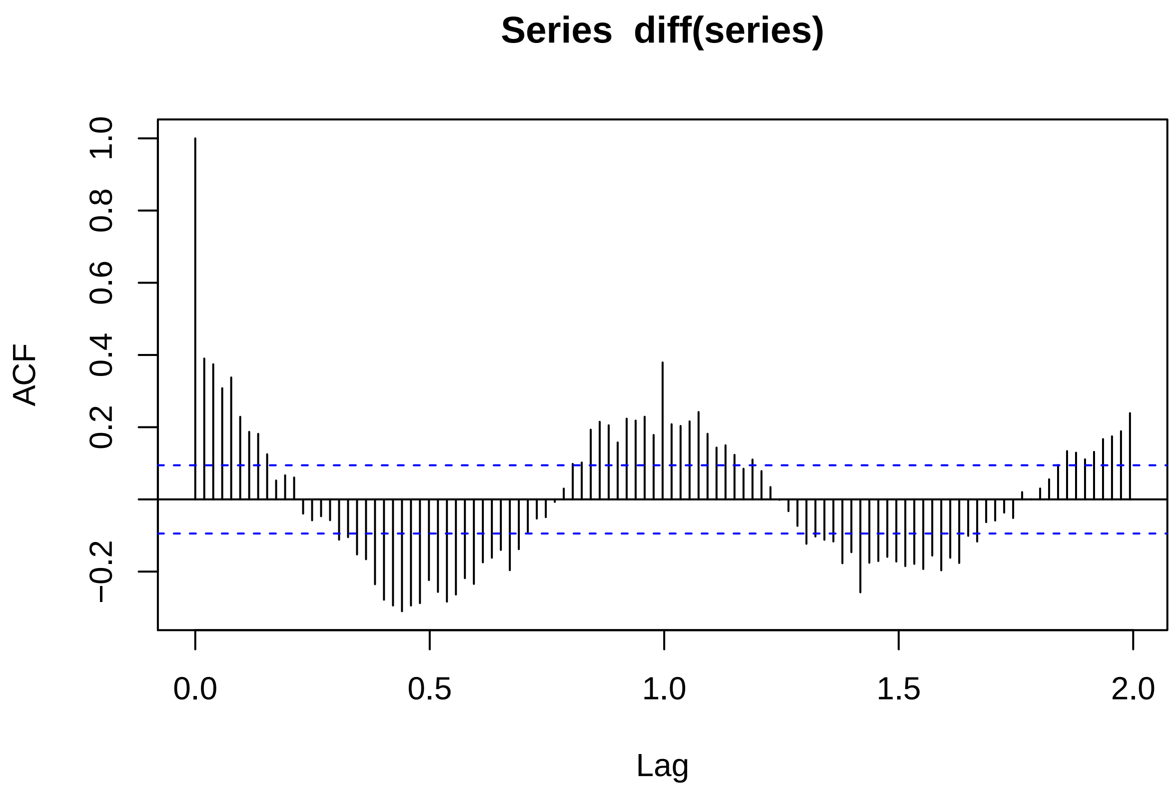

```r
plot(diff(series), main="First difference")
```



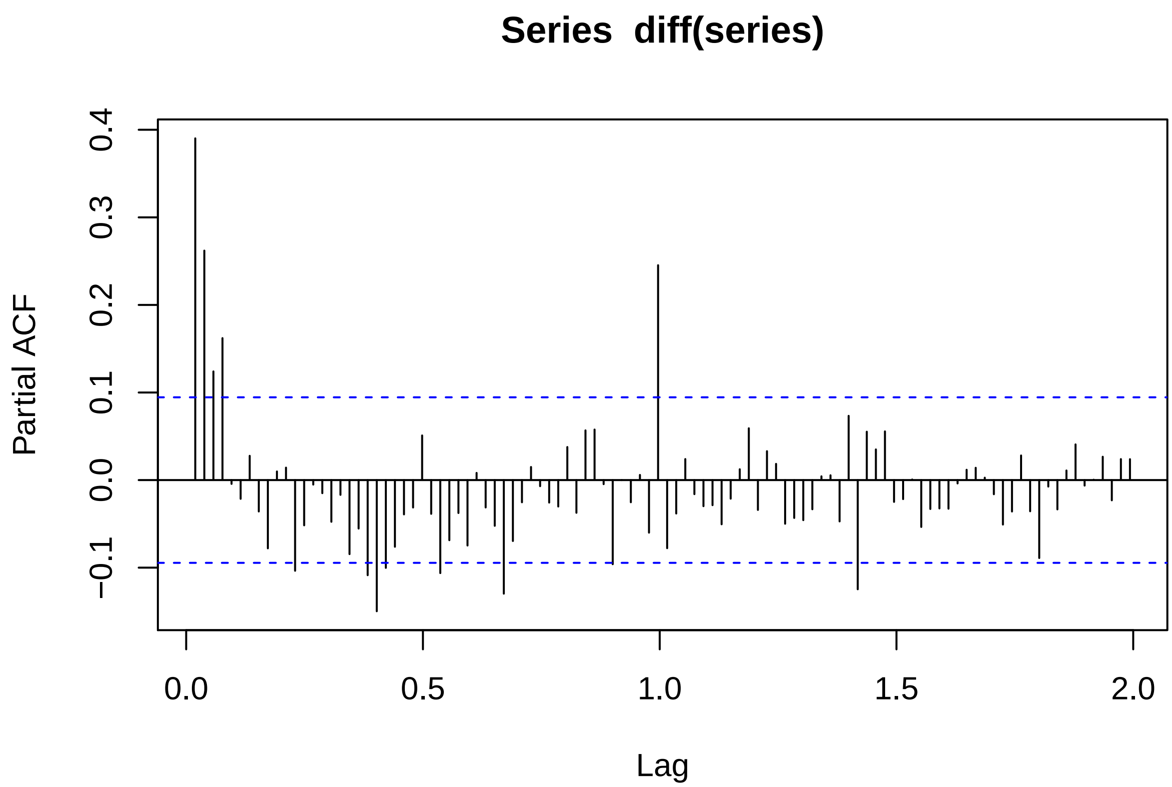

```
plot(diff(series, differences = 2))
```




```
acf(diff(series), lag.max = 52 * 2)
```




```r
pacf(diff(series), lag.max = 52 * 2)
```

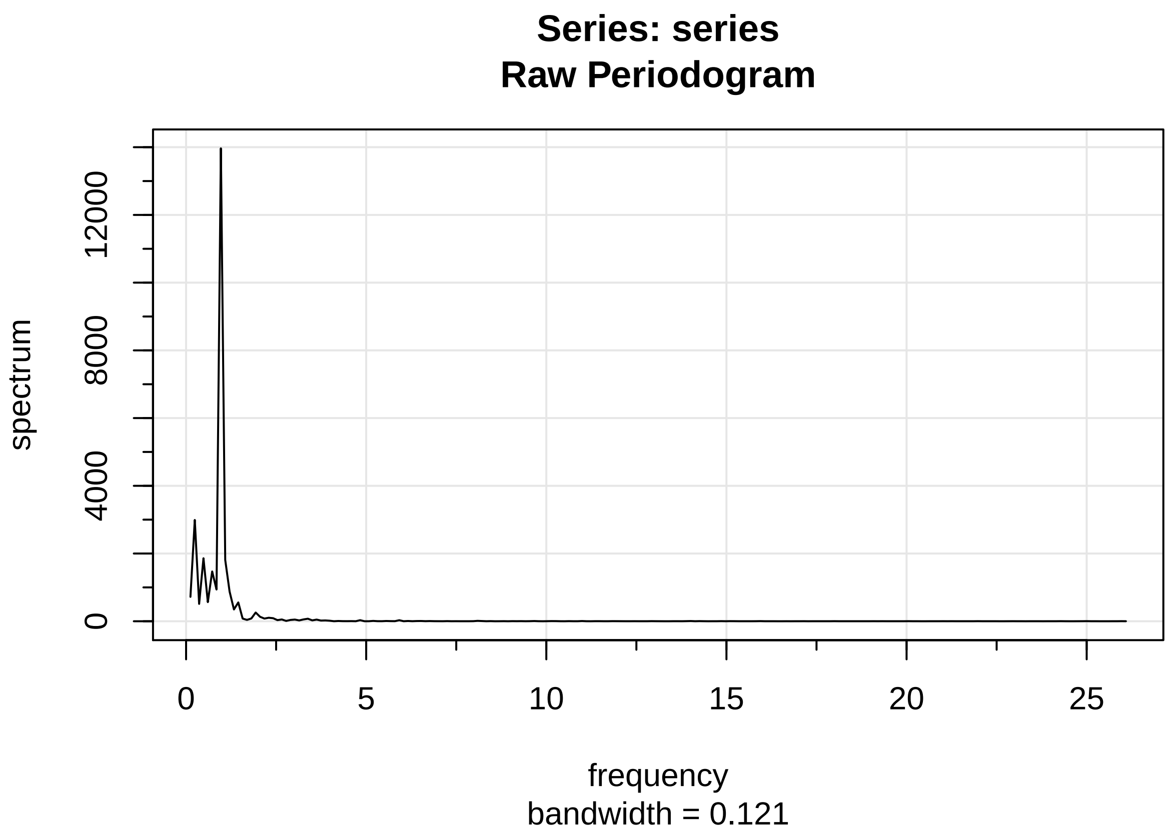



```r
adf.test(series)
```

```
## Warning in adf.test(series): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  series
## Dickey-Fuller = -5.5486, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

```r
pg <- mvspec(series, log="no")
```




```
k = kernel("daniell", c(1,1))
pg <- mvspec(series, kernel=k, log="no")

l <- 1/sum(k$coef^2)

alpha <- 0.05
U = qchisq(alpha/2, 2*l)
L = qchisq(1-alpha/2, 2*l)

max_index <- which(pg$spec == max(pg$spec))
freq <- pg$freq[max_index]

1/freq
```

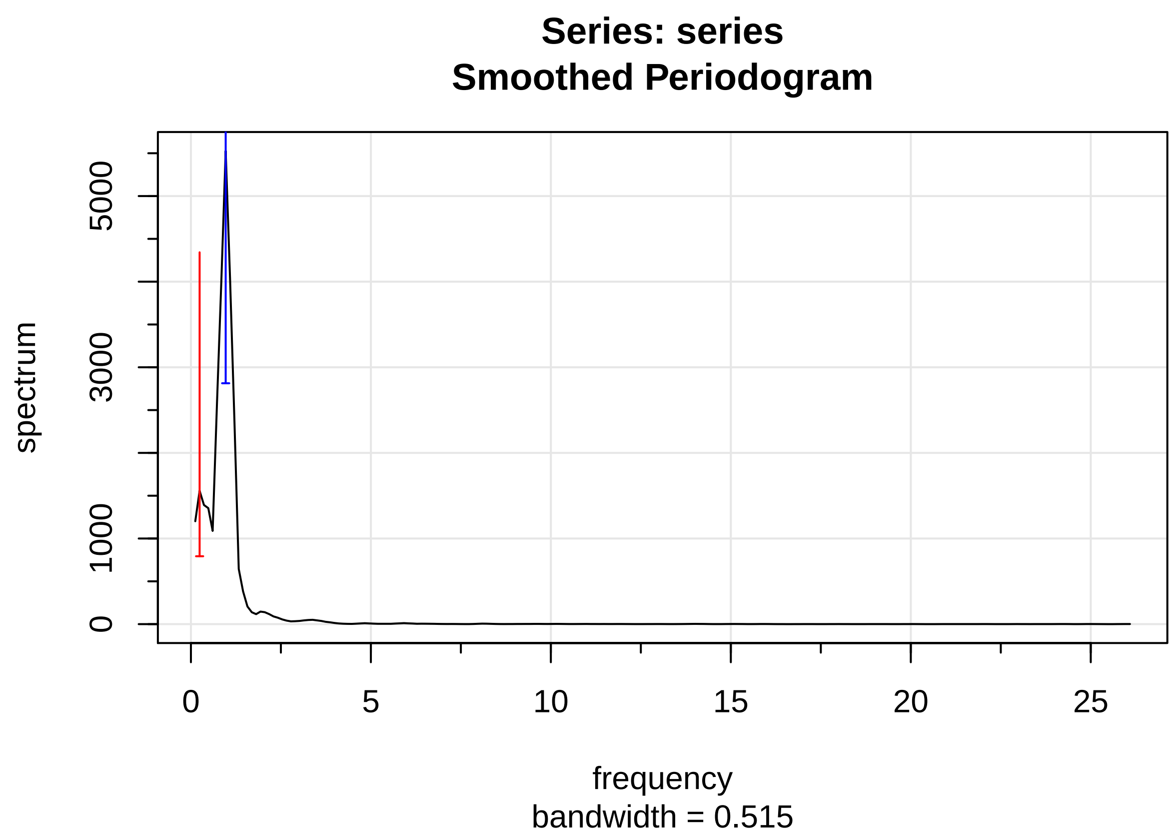
```
## [1] 1.034908
```

```
conf <- l*c(2*pg$spec[max_index]/L,2*pg$spec[max_index]/U)
segments(x0=freq,y0=conf[1],x1=freq,y1=conf[2],col="blue")
segments(x0=freq-0.1,y0=conf[1],x1=freq+0.1,y1=conf[1],col="blue")

# abline(v=freq, col="blue")


max_index2 <- which(pg$spec == max(pg$spec[which(pg$freq < 0.5)]))
freq <- pg$freq[max_index2]
```

```r
1/freq
```

```
## [1] 4.13963
```

```r
conf <- l*c(2*pg$spec[max_index2]/L,2*pg$spec[max_index2]/U)
segments(x0=freq,y0=conf[1],x1=freq,y1=conf[2],col="red")
segments(x0=freq-0.1,y0=conf[1],x1=freq+0.1,y1=conf[1],col="red")
```

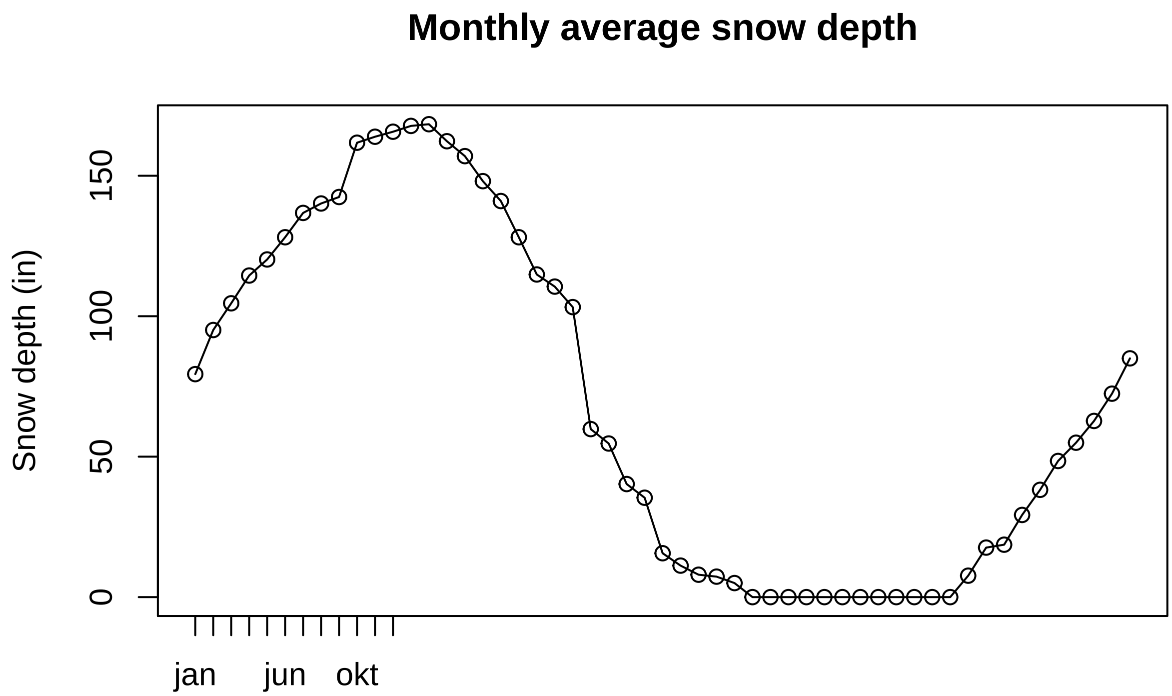



```r
# abline(v=freq, col="red")
```

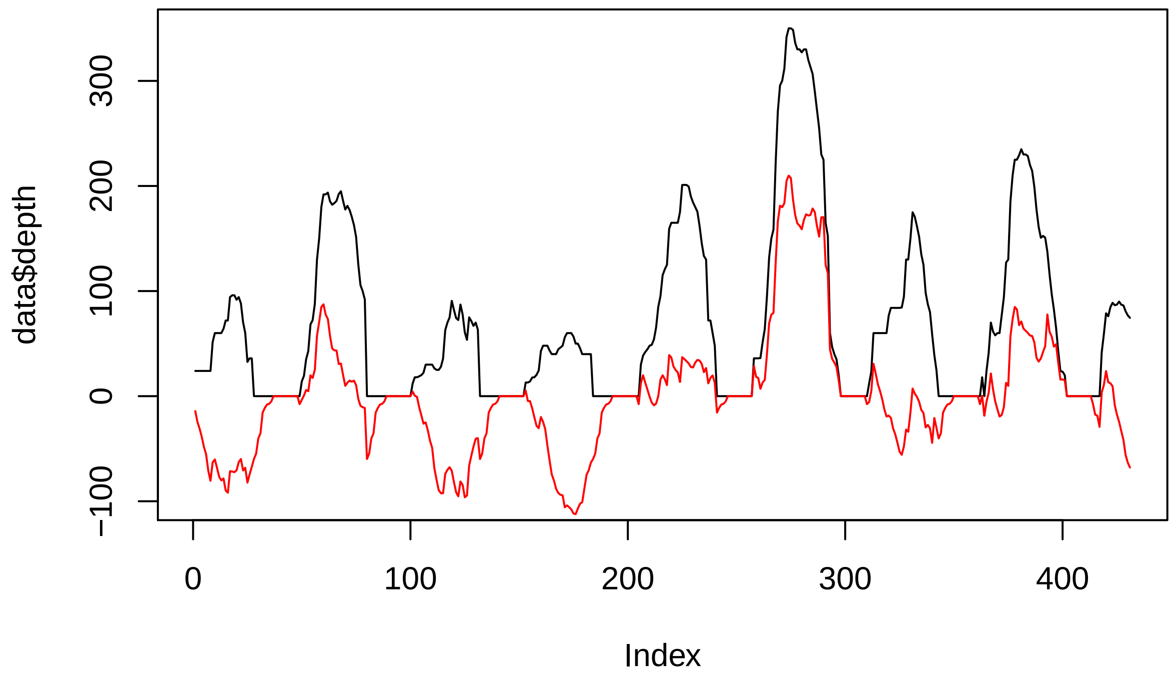
Detrend

```r
# lin.mod <- lm(series ~ time(series))
# summary(lin.mod)
#
# plot(series)
# lines(series*0 + predict(lin.mod, time(series)), type="l", col="red")
#
# detrended <- series - predict(lin.mod, time(series))
#
# plot(detrended)
```

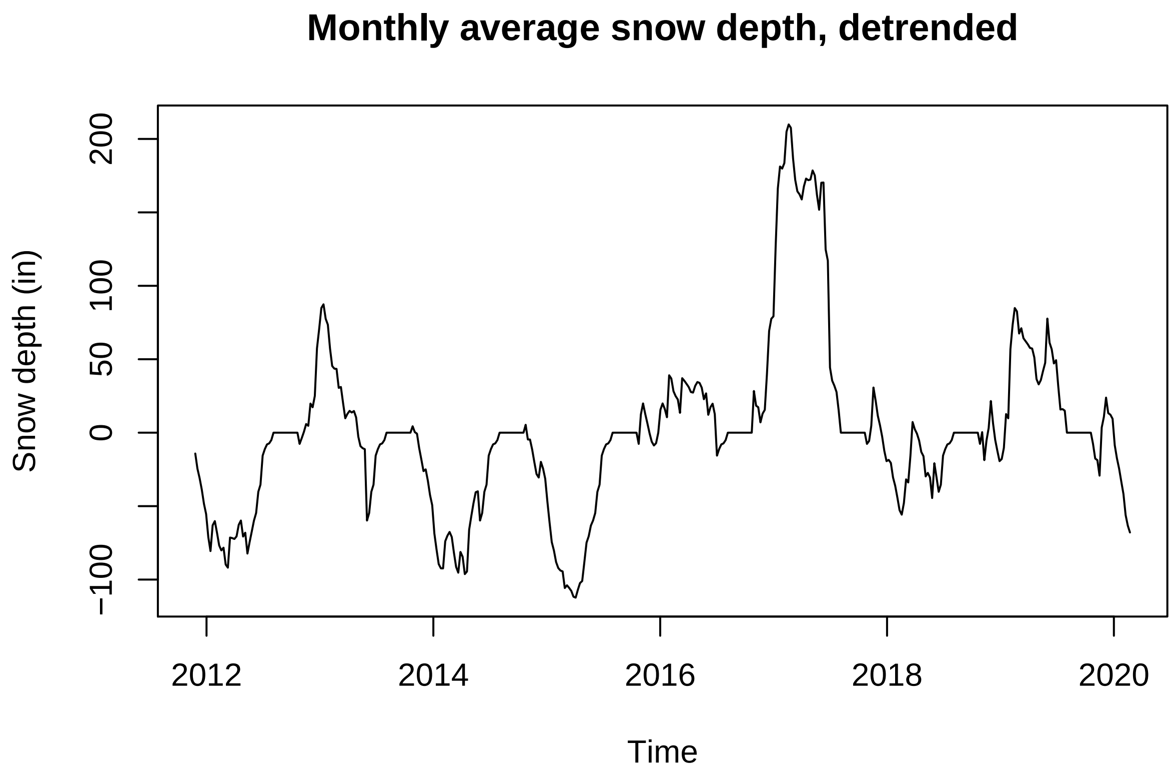
```r
periods <- as.numeric(format(data$date, "%V"))
# periods <- as.numeric(format(data$date, "%m"))

agg <- aggregate(
  data$depth,
  by=list(period = periods),
  FUN=mean
)

plot(agg$x, main="Monthly average snow depth", xaxt ="n", type="o", ylab="Snow depth (in)", xlab="")
axis(1, at=1:12, labels=month.labels)
```




```r
detrended <- data$depth - agg$x[match(periods, agg$period)]

plot(data$depth, type="l", ylim=c(-100, 350))
lines(detrended, type="l", col="red")
```

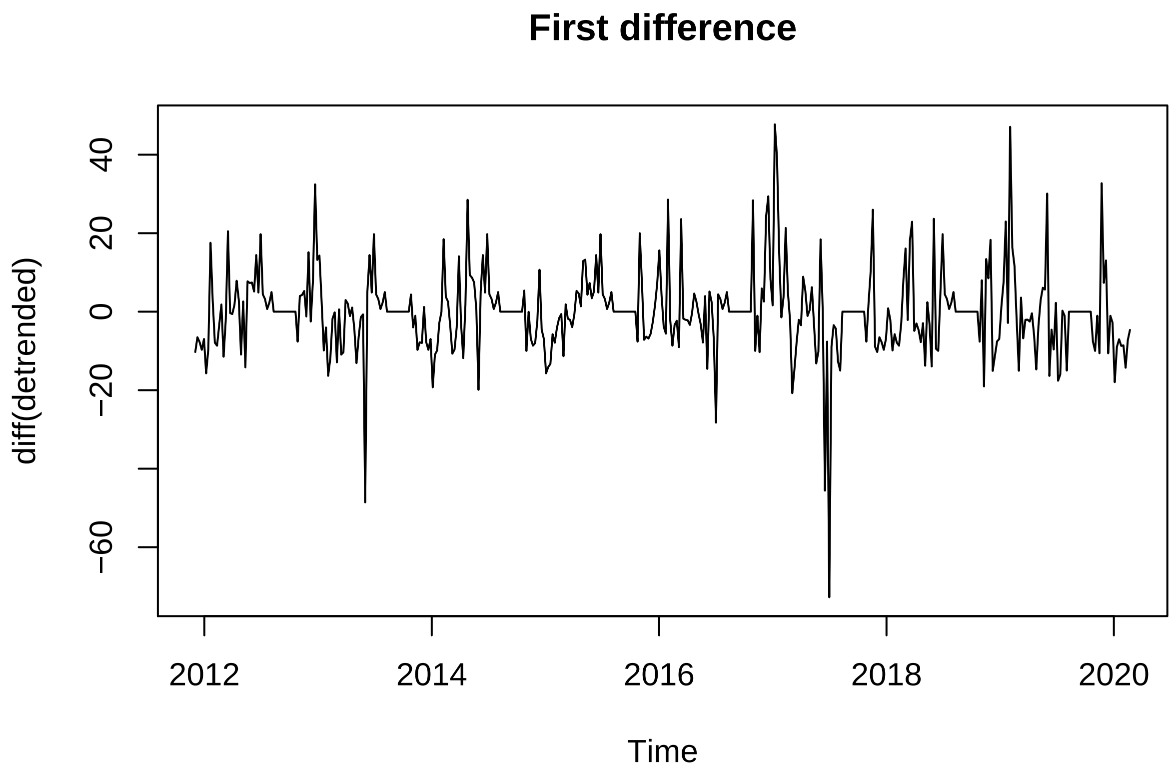



```
detrended <- ts(
  detrended,
  start=start(series),
  end=end(series),
  frequency=frequency(series)
)
plot(detrended, ylab="Snow depth (in)", main="Monthly average snow depth, detrended")
```

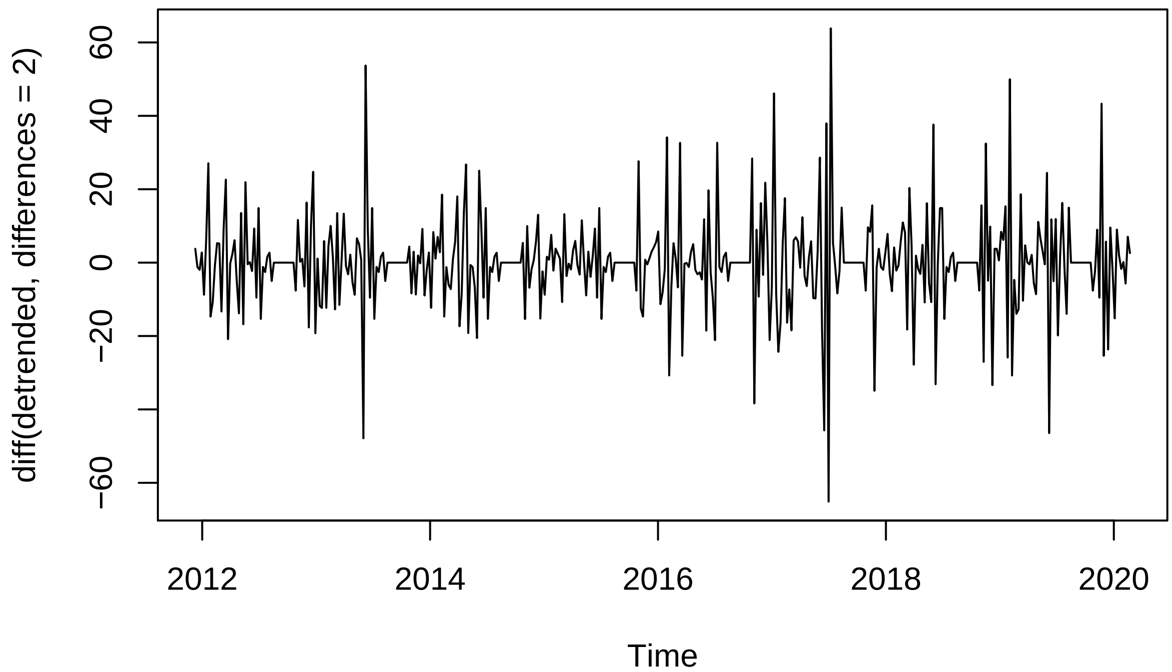



```r
# sum(is.na(data$depth) / length(data$depth))
#
# (tab <- table(as.numeric(format.Date(data$date, "%m")), is.na(data$depth)))
#
# plot(tab[, 1] / (tab[, 1] + tab[, 2]), type="o")
#
#
# plot(detrended)
# detrended <- na_interpolation(detrended, option = "linear")
# plot(detrended)
```

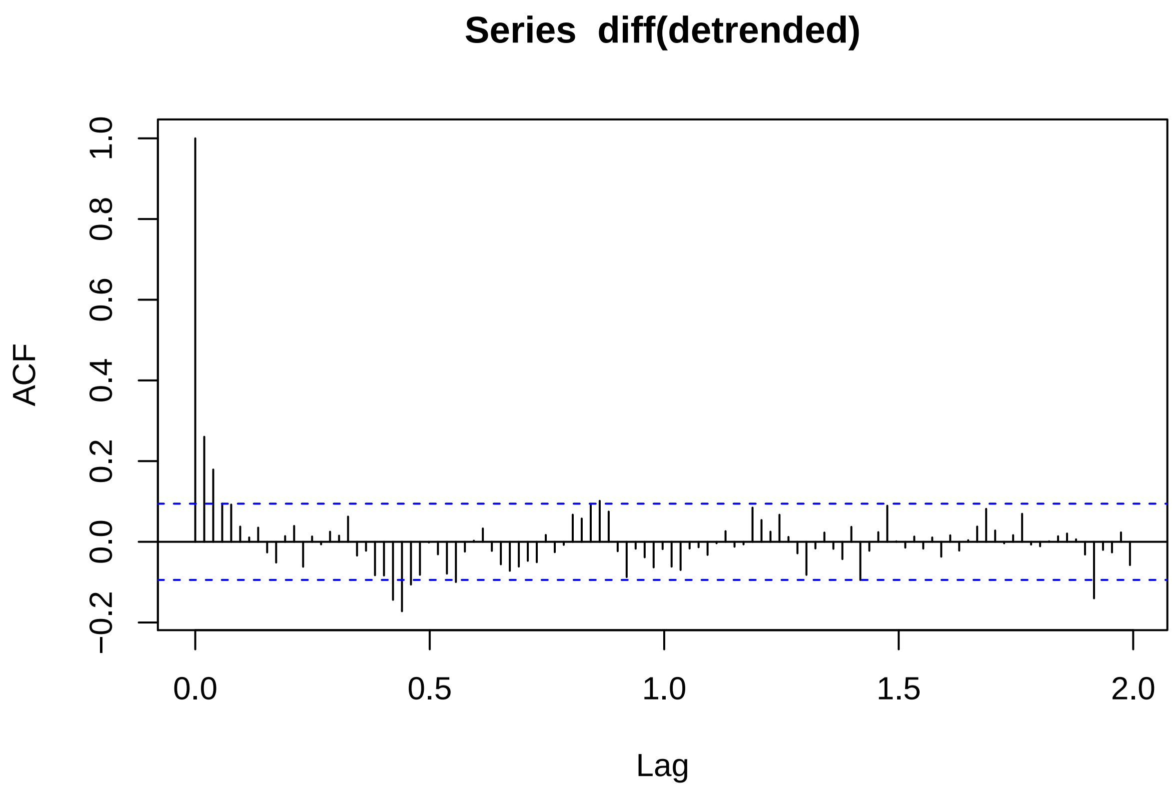
```r
plot(diff(detrended), main="First difference")
```



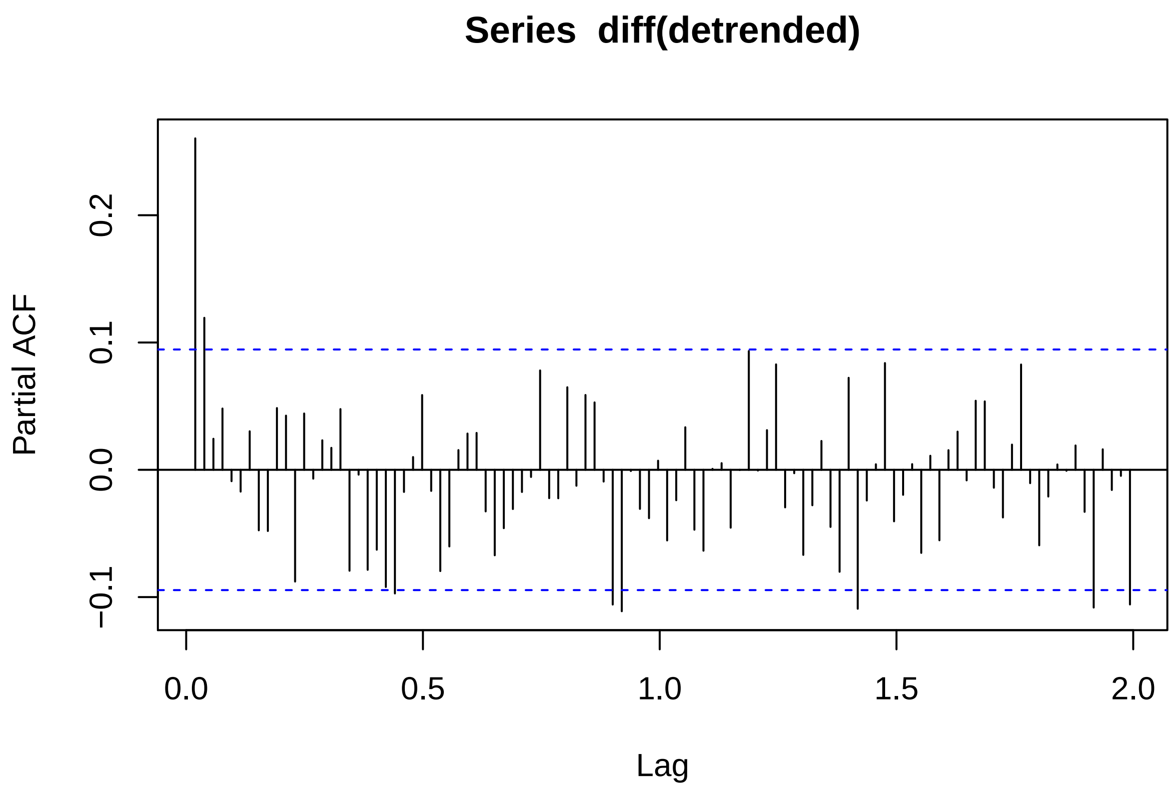

```r
plot(diff(detrended, differences = 2))
```




```r
acf(diff(detrended), lag.max = 52 * 2)
```




```
pacf(diff(detrended), lag.max = 52 * 2)
```




```
adf.test(detrended)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  detrended
## Dickey-Fuller = -3.2378, Lag order = 7, p-value = 0.08159
## alternative hypothesis: stationary
```

Periodogram

```
# TODO konfidens

# pg <- mvspec(detrended, log="no")

k = kernel("daniell", c(1,1))
pg <- mvspec(detrended, kernel=k, log="no")

l <- 1/sum(k$coef^2)

alpha <- 0.05
U = qchisq(alpha/2, 2*l)
L = qchisq(1-alpha/2, 2*l)
```

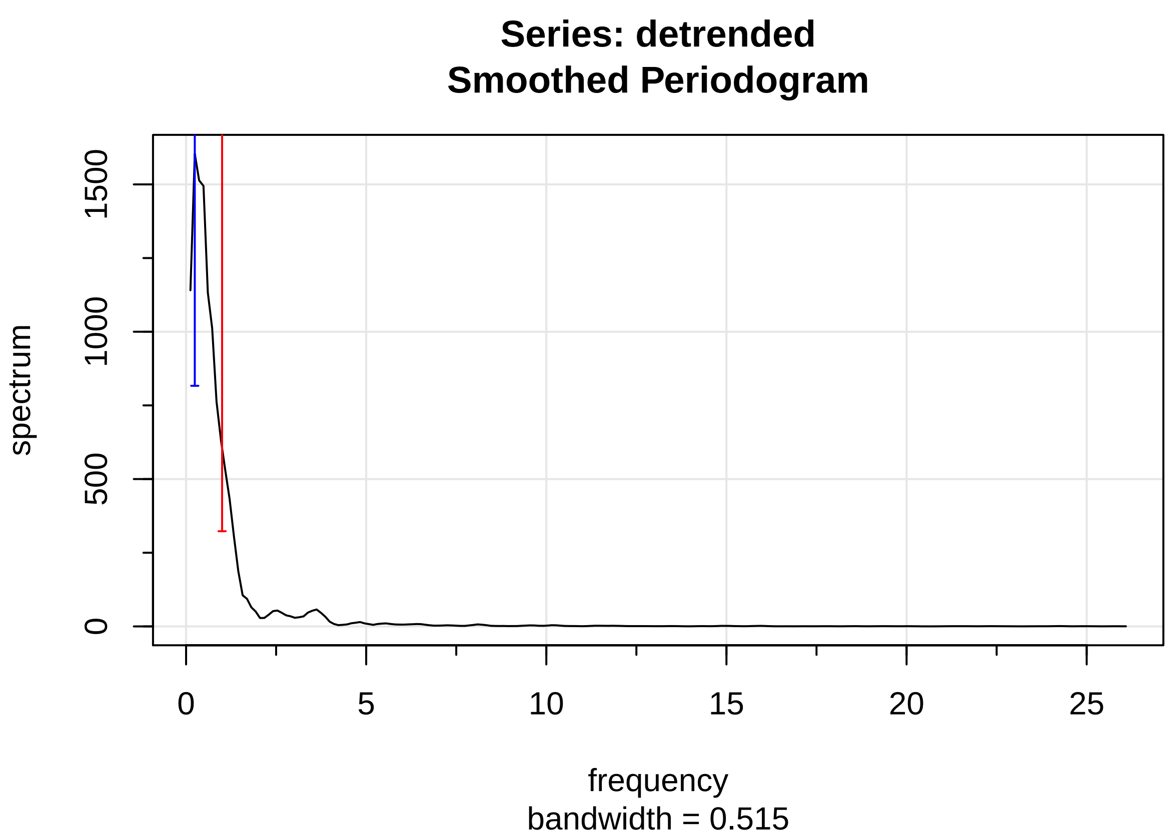
```r
max_index <- which(pg$spec == max(pg$spec))
freq <- pg$freq[max_index]

1/freq
```

```
## [1] 4.13963
```

```r
conf <- l*c(2*pg$spec[max_index]/L,2*pg$spec[max_index]/U)
segments(x0=freq,y0=conf[1],x1=freq,y1=conf[2],col="blue")
segments(x0=freq-0.1,y0=conf[1],x1=freq+0.1,y1=conf[1],col="blue")

# abline(v=freq, col="blue")


max_index2 <- which(pg$spec == max(pg$spec[which(abs(pg$freq - 1) == min(abs(pg$freq - 1)))]))
freq <- 1

1/freq
```

```
## [1] 1
```

```r
conf <- l*c(2*pg$spec[max_index2]/L,2*pg$spec[max_index2]/U)
segments(x0=freq,y0=conf[1],x1=freq,y1=conf[2],col="red")
segments(x0=freq-0.1,y0=conf[1],x1=freq+0.1,y1=conf[1],col="red")
```



```r
# abline(v=freq, col="red")
```

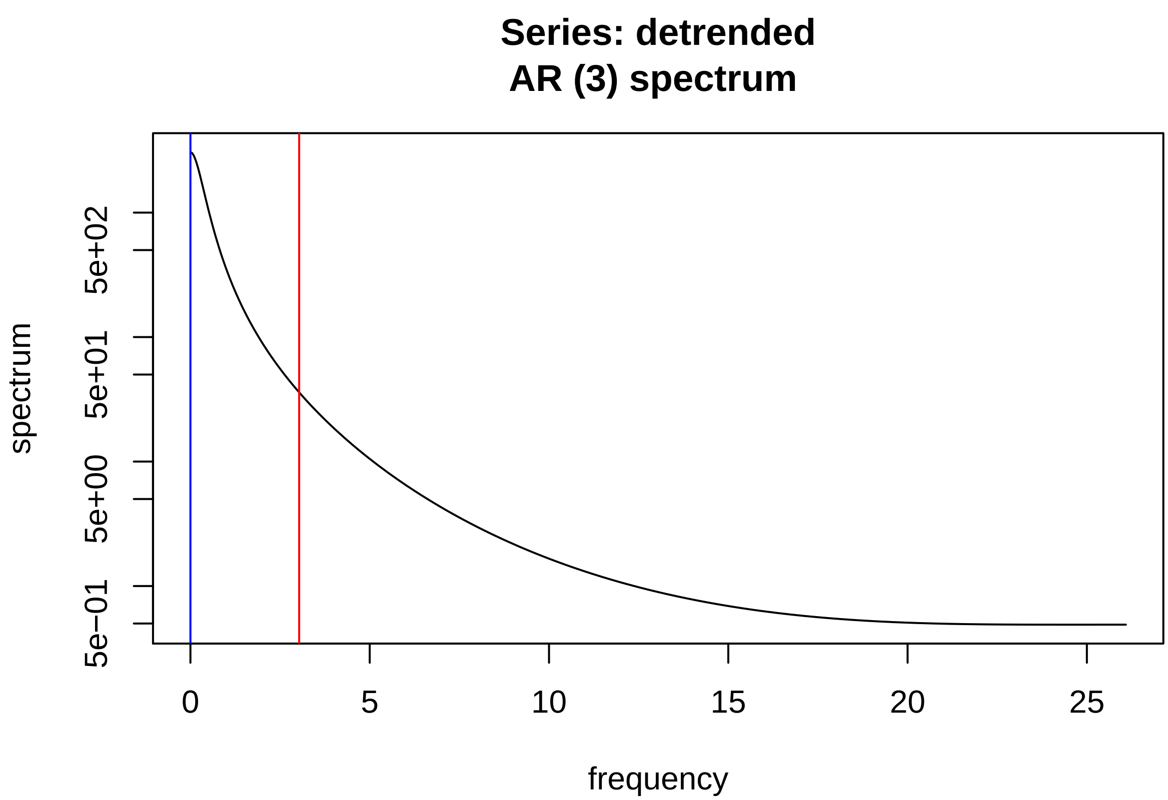
```r
pg <- spec.ar(detrended)

max_index <- which(pg$spec == max(pg$spec))
freq <- pg$freq[max_index]

1/freq
```

```
## [1] Inf
```

```r
abline(v=freq,col="blue")


max_index2 <- which(pg$spec == max(pg$spec[which(pg$freq > 3)]))
freq <- pg$freq[max_index2]

1/freq
```

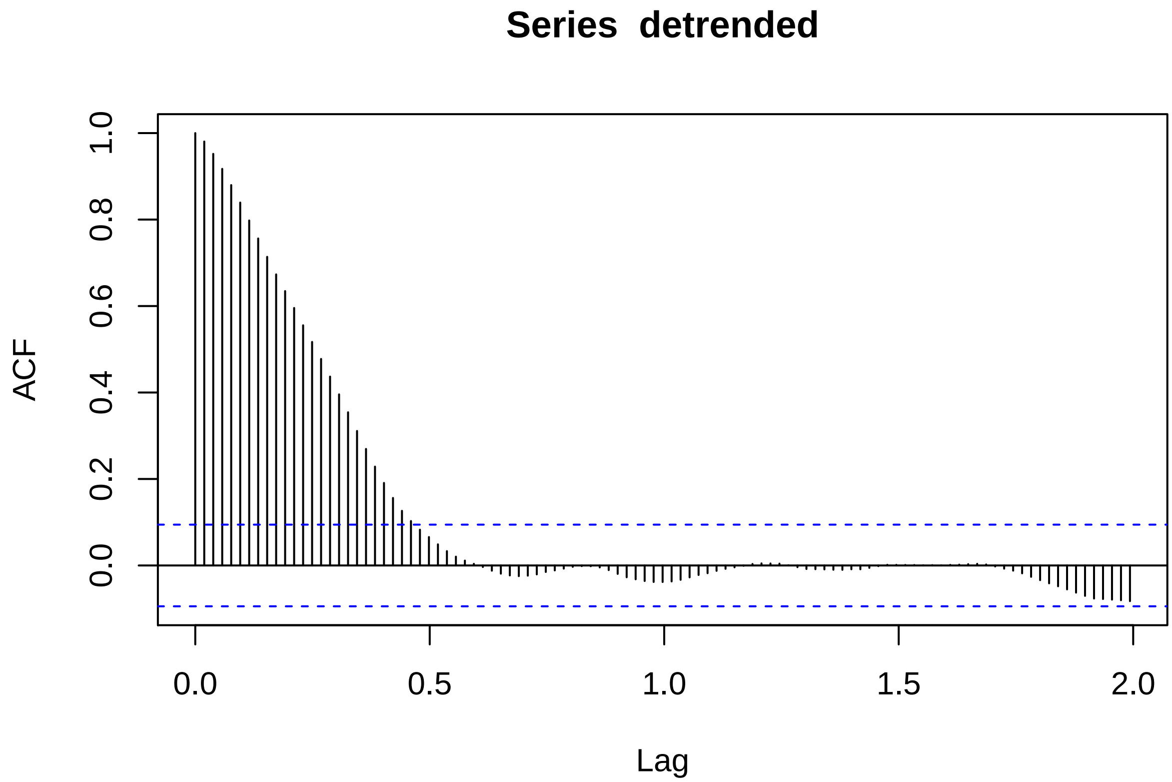
```
## [1] 0.3297694
```

```r
abline(v=freq,col="red")
```

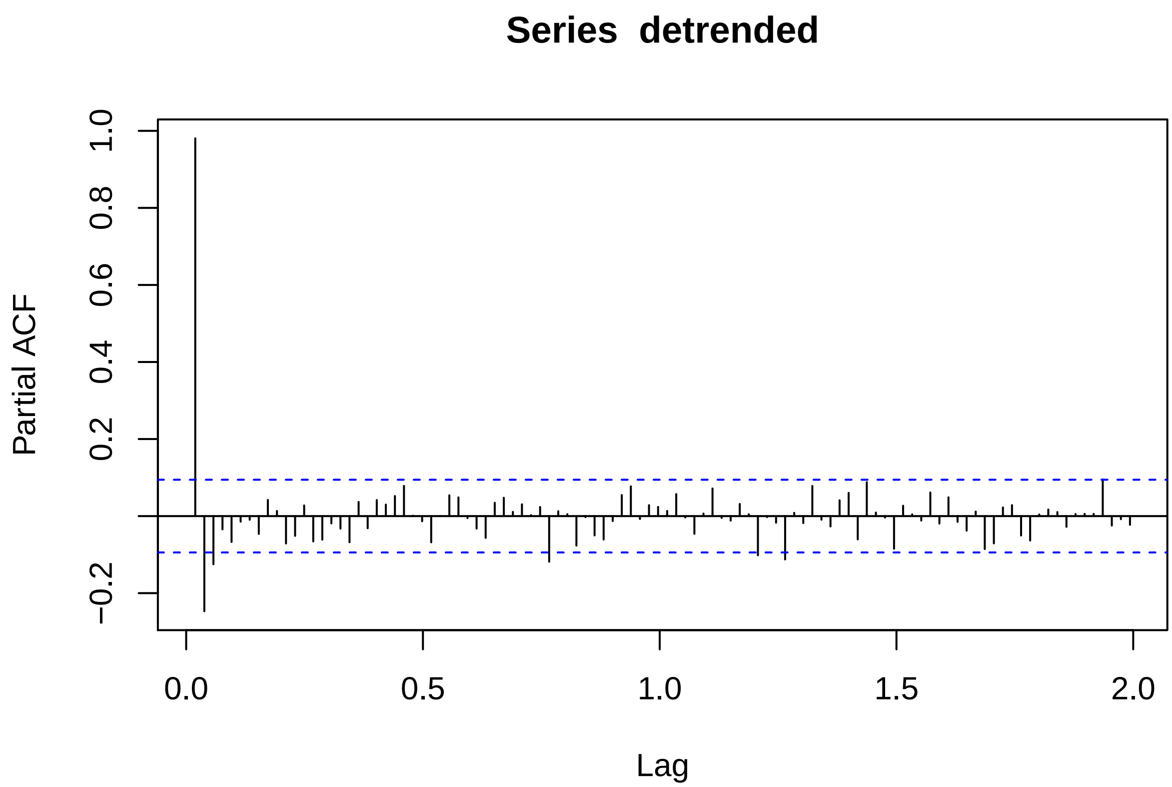


ACF

```r
acf(detrended, lag.max = 52 * 2)
```




```r
pacf(detrended, lag.max = 52 * 2)
```



Fit ARIMA

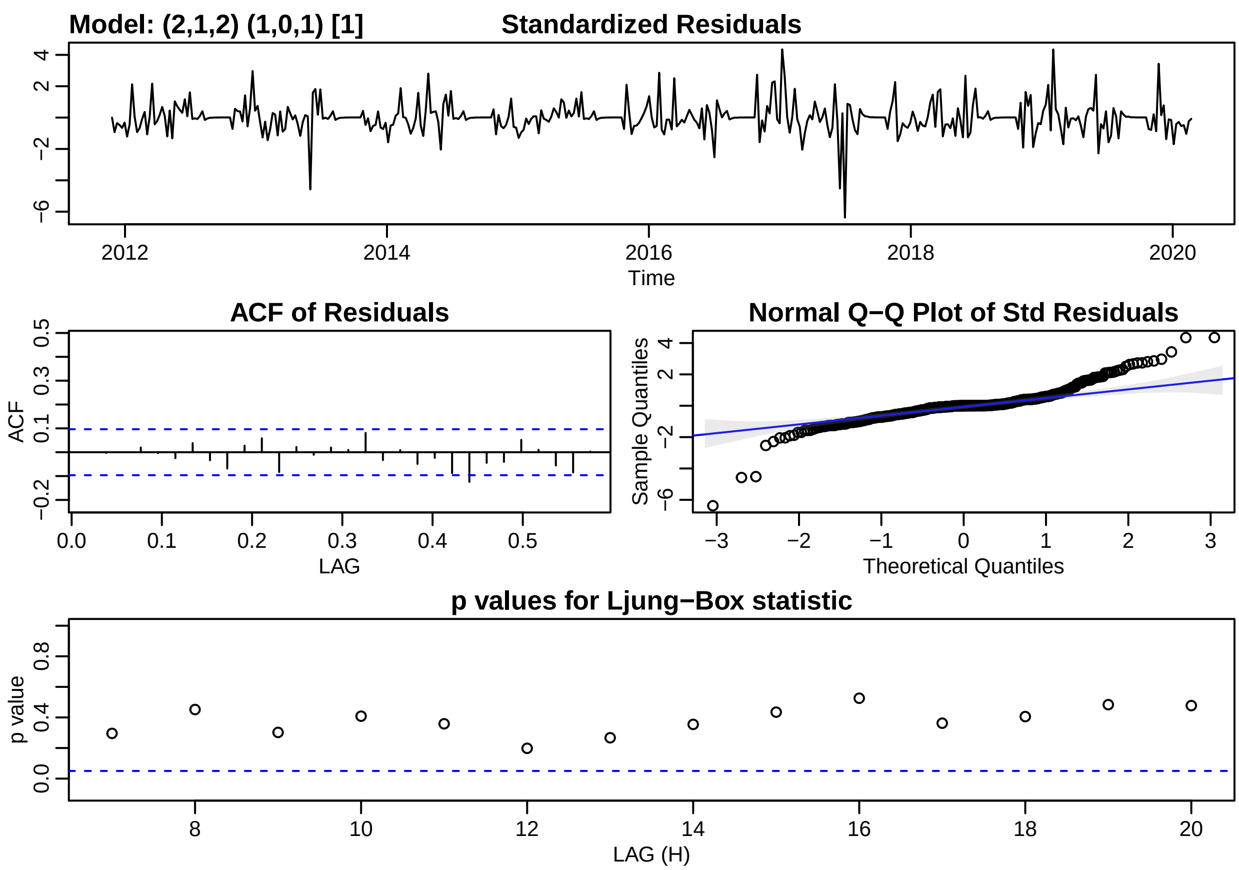
```
# auto.arima(detrended)
# ARIMA(2,0,1)(1,1,0)[52]

# Coefficients:
#          ar1      ar2      ma1     sar1
#       1.7781  -0.7911  -0.5474  -0.5192
# s.e.  0.0679   0.0666   0.0923   0.0433
#
# sigma^2 estimated as 148.9:  log likelihood=-1489.99
# AIC=2989.98   AICc=2990.14   BIC=3009.65

# sarima(detrended, 2,0,1, 1,1,0, 1)
# sarima(detrended, 2,0,1, 1,1,1, 1)


## OFF PEAKS
# Best model: ARIMA(2,1,2)(1,0,0)[52]
#
# Series: detrended
# ARIMA(2,1,2)(1,0,0)[52]
#
```

```r
# Coefficients:
#          ar1     ar2     ma1      ma2    sar1
#       0.0308  0.5816  -0.082  -0.4565  0.4746
# s.e.  0.1869  0.1543   0.202   0.1653  0.0445
#
# sigma^2 estimated as 265.3:  log likelihood=-1814.17
# AIC=3640.35   AICc=3640.54   BIC=3664.73

sarima(detrended, 2,1,2, 1,0,1, 1)
```

```
## initial  value 2.387386
## iter   2 value 2.349709
## iter   3 value 2.349113
## iter   4 value 2.347056
## iter   5 value 2.346919
## iter   6 value 2.345434
## iter   7 value 2.344444
## iter   8 value 2.344435
## iter   9 value 2.344247
## iter  10 value 2.344231
## iter  11 value 2.344206
## iter  12 value 2.344175
## iter  13 value 2.344119
## iter  14 value 2.344091
## iter  15 value 2.344071
## iter  16 value 2.344069
## iter  17 value 2.344068
## iter  18 value 2.344065
## iter  19 value 2.344062
## iter  20 value 2.344061
## iter  21 value 2.344061
## iter  22 value 2.344060
## iter  23 value 2.344059
## iter  24 value 2.344057
## iter  25 value 2.344056
## iter  26 value 2.344055
## iter  27 value 2.344054
## iter  28 value 2.344053
## iter  29 value 2.344052
## iter  30 value 2.344052
## iter  31 value 2.344051
## iter  32 value 2.344051
## iter  33 value 2.344050
## iter  34 value 2.344049
## iter  35 value 2.344049
## iter  36 value 2.344049
## iter  37 value 2.344048
## iter  38 value 2.344047
## iter  39 value 2.344047
## iter  40 value 2.344047
## iter  40 value 2.344047
## iter  40 value 2.344047
## final  value 2.344047
```

```
## converged
## initial  value 2.342541
## iter   2 value 2.342532
## iter   3 value 2.342523
## iter   4 value 2.342512
## iter   5 value 2.342496
## iter   6 value 2.342486
## iter   7 value 2.342485
## iter   8 value 2.342483
## iter   9 value 2.342478
## iter  10 value 2.342470
## iter  11 value 2.342462
## iter  12 value 2.342457
## iter  13 value 2.342455
## iter  14 value 2.342455
## iter  15 value 2.342453
## iter  16 value 2.342449
## iter  17 value 2.342445
## iter  18 value 2.342445
## iter  19 value 2.342444
## iter  20 value 2.342443
## iter  21 value 2.342437
## iter  22 value 2.342430
## iter  23 value 2.342425
## iter  24 value 2.342423
## iter  25 value 2.342422
## iter  26 value 2.342421
## iter  27 value 2.342417
## iter  28 value 2.342412
## iter  29 value 2.342407
## iter  30 value 2.342402
## iter  31 value 2.342397
## iter  32 value 2.342396
## iter  33 value 2.342392
## iter  34 value 2.342386
## iter  35 value 2.342382
## iter  36 value 2.342381
## iter  37 value 2.342381
## iter  38 value 2.342380
## iter  39 value 2.342378
## iter  40 value 2.342376
## iter  41 value 2.342375
## iter  42 value 2.342375
## iter  43 value 2.342375
## iter  44 value 2.342374
## iter  45 value 2.342373
## iter  46 value 2.342370
## iter  47 value 2.342368
## iter  48 value 2.342367
## iter  49 value 2.342367
## iter  50 value 2.342367
## iter  51 value 2.342366
## iter  52 value 2.342366
## iter  53 value 2.342366
```

```
## iter  54 value 2.342366
## iter  55 value 2.342365
## iter  56 value 2.342363
## iter  57 value 2.342360
## iter  58 value 2.342356
## iter  59 value 2.342354
## iter  60 value 2.342353
## iter  61 value 2.342353
## iter  62 value 2.342352
## iter  63 value 2.342352
## iter  64 value 2.342351
## iter  65 value 2.342350
## iter  66 value 2.342350
## iter  67 value 2.342350
## iter  67 value 2.342350
## iter  67 value 2.342350
## final  value 2.342350
## converged
```




```
## $fit
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
##     Q), period = S), xreg = constant, transform.pars = trans, fixed = fixed,
##     optim.control = list(trace = trc, REPORT = 1, reltol = tol))
```

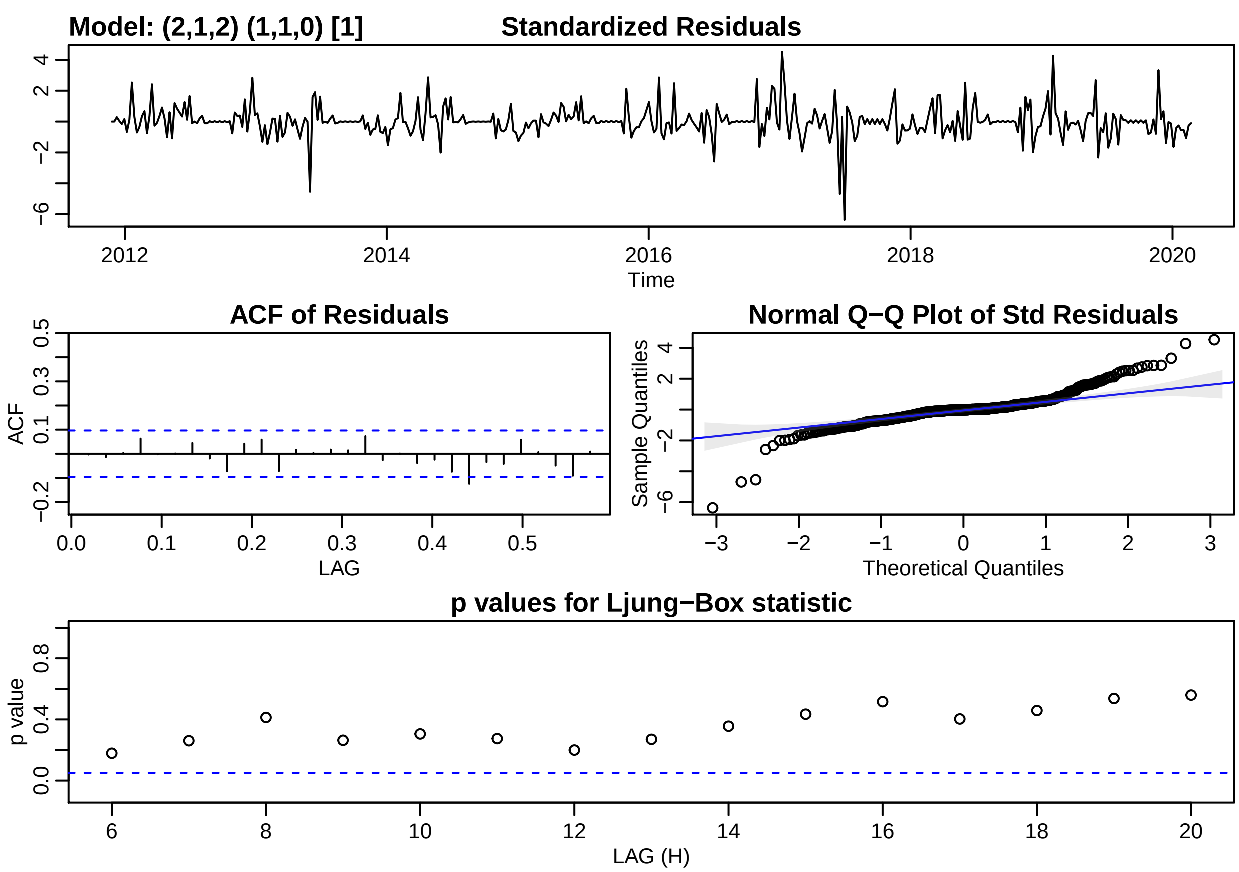
```
##
## Coefficients:
##          ar1     ar2      ma1      ma2     sar1    sma1  constant
##       0.1333  0.3550  -0.0929  -0.1771  -0.1508  0.3360   -0.1616
## s.e.  0.4056  0.2459   0.9888   0.4158   0.6352  0.9499    0.8292
##
## sigma^2 estimated as 108.3:  log likelihood = -1617.35,  aic = 3250.71
##
## $degrees_of_freedom
## [1] 423
##
## $ttable
##          Estimate     SE t.value p.value
## ar1        0.1333 0.4056  0.3287  0.7425
## ar2        0.3550 0.2459  1.4440  0.1495
## ma1       -0.0929 0.9888 -0.0940  0.9252
## ma2       -0.1771 0.4158 -0.4260  0.6703
## sar1      -0.1508 0.6352 -0.2375  0.8124
## sma1       0.3360 0.9499  0.3537  0.7237
## constant  -0.1616 0.8292 -0.1949  0.8455
##
## $AIC
## [1] 7.559786
##
## $AICc
## [1] 7.560403
##
## $BIC
## [1] 7.635392
```

```
sarima(detrended, 2,1,2, 1,1,0, 1)
```

```
## initial  value 2.585349
## iter   2 value 2.490130
## iter   3 value 2.441987
## iter   4 value 2.405801
## iter   5 value 2.395213
## iter   6 value 2.379620
## iter   7 value 2.373899
## iter   8 value 2.366838
## iter   9 value 2.365639
## iter  10 value 2.364993
## iter  11 value 2.364669
## iter  12 value 2.364644
## iter  13 value 2.364637
## iter  14 value 2.364602
## iter  15 value 2.364545
## iter  16 value 2.364452
## iter  17 value 2.364296
## iter  18 value 2.364266
## iter  19 value 2.364263
## iter  20 value 2.364262
## iter  21 value 2.364256
## iter  22 value 2.364242
```

```
## iter  23 value 2.364191
## iter  24 value 2.364181
## iter  25 value 2.364168
## iter  26 value 2.364166
## iter  27 value 2.364156
## iter  28 value 2.364154
## iter  29 value 2.364136
## iter  30 value 2.364122
## iter  31 value 2.364118
## iter  32 value 2.364115
## iter  33 value 2.364109
## iter  34 value 2.364096
## iter  35 value 2.364091
## iter  36 value 2.364087
## iter  37 value 2.364084
## iter  38 value 2.364081
## iter  39 value 2.364078
## iter  40 value 2.364067
## iter  41 value 2.364054
## iter  42 value 2.364043
## iter  43 value 2.364037
## iter  44 value 2.364026
## iter  45 value 2.364000
## iter  46 value 2.363986
## iter  47 value 2.363982
## iter  48 value 2.363972
## iter  49 value 2.363965
## iter  50 value 2.363944
## iter  51 value 2.363933
## iter  52 value 2.363928
## iter  53 value 2.363923
## iter  54 value 2.363910
## iter  55 value 2.363870
## iter  56 value 2.360381
## iter  57 value 2.359817
## iter  58 value 2.357596
## iter  59 value 2.356405
## iter  60 value 2.356095
## iter  61 value 2.351487
## iter  62 value 2.351388
## iter  63 value 2.350766
## iter  64 value 2.350512
## iter  65 value 2.350108
## iter  66 value 2.350071
## iter  67 value 2.349884
## iter  68 value 2.349831
## iter  69 value 2.349739
## iter  70 value 2.349737
## iter  70 value 2.349737
## iter  70 value 2.349737
## final  value 2.349737
## converged
## initial  value 2.355281
## iter   2 value 2.349210
```

```
## iter   3 value 2.348998
## iter   4 value 2.348956
## iter   5 value 2.348917
## iter   6 value 2.348902
## iter   7 value 2.348889
## iter   8 value 2.348888
## iter   9 value 2.348888
## iter  10 value 2.348887
## iter  11 value 2.348884
## iter  12 value 2.348880
## iter  13 value 2.348877
## iter  14 value 2.348877
## iter  15 value 2.348877
## iter  16 value 2.348876
## iter  17 value 2.348875
## iter  18 value 2.348875
## iter  19 value 2.348873
## iter  20 value 2.348871
## iter  21 value 2.348869
## iter  22 value 2.348868
## iter  23 value 2.348868
## iter  23 value 2.348868
## final  value 2.348868
## converged
```




```
## $fit
```

```
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
##     Q), period = S), include.mean = !no.constant, transform.pars = trans, fixed = fixed,
##     optim.control = list(trace = trc, REPORT = 1, reltol = tol))
##
## Coefficients:
##           ar1     ar2      ma1      ma2     sar1
##       -0.4857  0.4803  -0.0091  -0.9909  -0.2702
## s.e.   0.0706  0.0617   0.0173   0.0172   0.0728
##
## sigma^2 estimated as 108.3:  log likelihood = -1616.39,  aic = 3244.78
##
## $degrees_of_freedom
## [1] 424
##
## $ttable
##      Estimate     SE  t.value p.value
## ar1   -0.4857 0.0706  -6.8814  0.0000
## ar2    0.4803 0.0617   7.7802  0.0000
## ma1   -0.0091 0.0173  -0.5278  0.5979
## ma2   -0.9909 0.0172 -57.5223  0.0000
## sar1  -0.2702 0.0728  -3.7099  0.0002
##
## $AIC
## [1] 7.563585
##
## $AICc
## [1] 7.563916
##
## $BIC
## [1] 7.620388
```

```
models <- list(
  # arima(detrended, order=c(2,0,1), seasonal=list(order=c(1,1,0), period=52))#,

  # arima(detrended, order=c(2,1,2), seasonal=list(order=c(1,0,1), period=52)),
  # # arima(detrended, order=c(2,1,1), seasonal=list(order=c(1,1,0), period=52))#,
  # # arima(detrended, order=c(3,1,2), seasonal=list(order=c(1,1,0), period=52))#,
  # # arima(detrended, order=c(2,0,1), seasonal=list(order=c(1,1,1), period=52)),
  # # arima(detrended, order=c(3,0,3), seasonal=list(order=c(3,1,3), period=52))
  # # arima(detrended, order=c(2,0,1), seasonal=list(order=c(0,1,1), period=52))#,
  # # arima(detrended, order=c(2,0,1), seasonal=list(order=c(0,1,2), period=52))#,
  # # arima(detrended, order=c(3,1,2), seasonal=list(order=c(2,1,1), period=52)),
  # # arima(detrended, order=c(5,1,2)),
  # arima(detrended, order=c(2,0,1), seasonal=list(order=c(1,0,0), period=52)),
  # arima(detrended, order=c(2,1,1), seasonal=list(order=c(1,0,0), period=52)),
  # arima(detrended, order=c(2,0,1), seasonal=list(order=c(2,0,0), period=52)),
  # arima(detrended, order=c(2,0,1), seasonal=list(order=c(1,1,0), period=52)),
  # arima(detrended, order=c(2,0,1), seasonal=list(order=c(1,1,1), period=52)),

  arima(detrended, order=c(2,1,2), seasonal=list(order=c(1,1,0), period=52))#,

  # arima(detrended, order=c(2,1,1), seasonal=list(order=c(1,0,1), period=52))#,
```

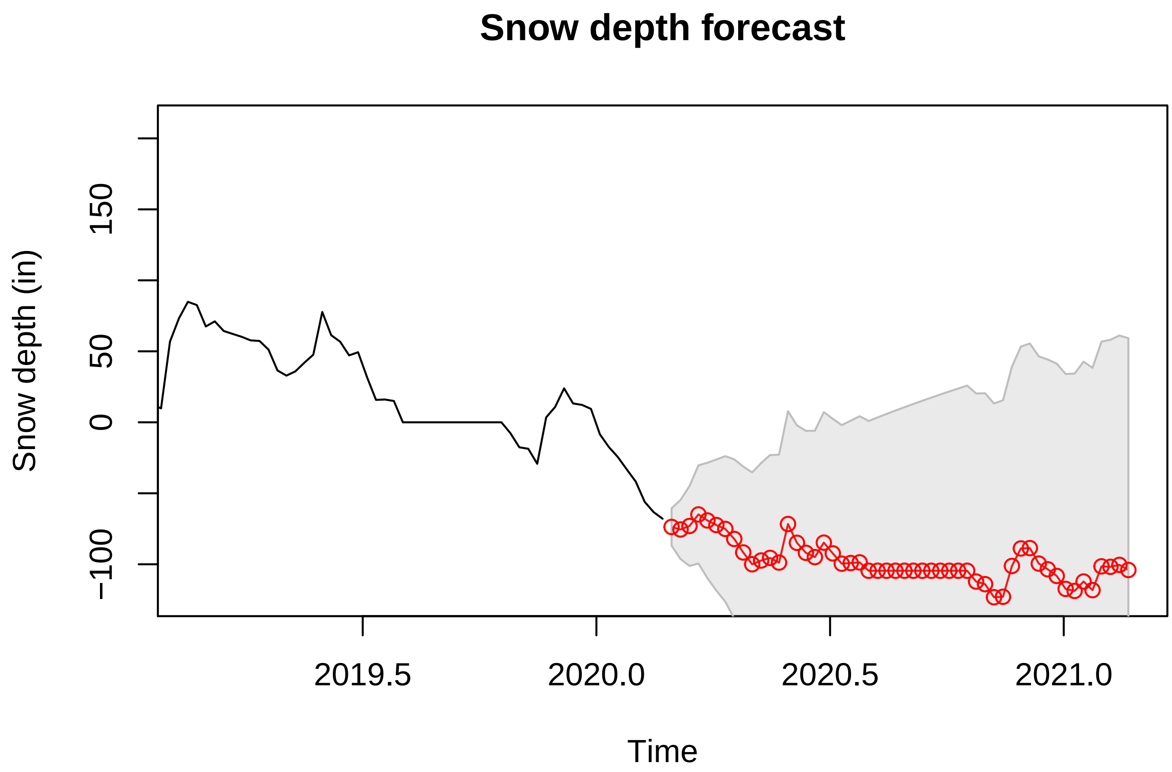
```r
  # arima(detrended, order=c(5,0,5), seasonal=list(order=c(5,0,5), period=52)),
  # arima(detrended, order=c(5,0,5), seasonal=list(order=c(5,0,5), period=52)),
)

models
```

```
## [[1]]
##
## Call:
## arima(x = detrended, order = c(2, 1, 2), seasonal = list(order = c(1, 1, 0),
##     period = 52))
##
## Coefficients:
##           ar1     ar2     ma1      ma2     sar1
##       -0.0665  0.4691  0.2940  -0.2423  -0.4549
## s.e.   0.2120  0.1621  0.2255   0.1588   0.0471
##
## sigma^2 estimated as 172.8:  log likelihood = -1516.21,  aic = 3044.41
```

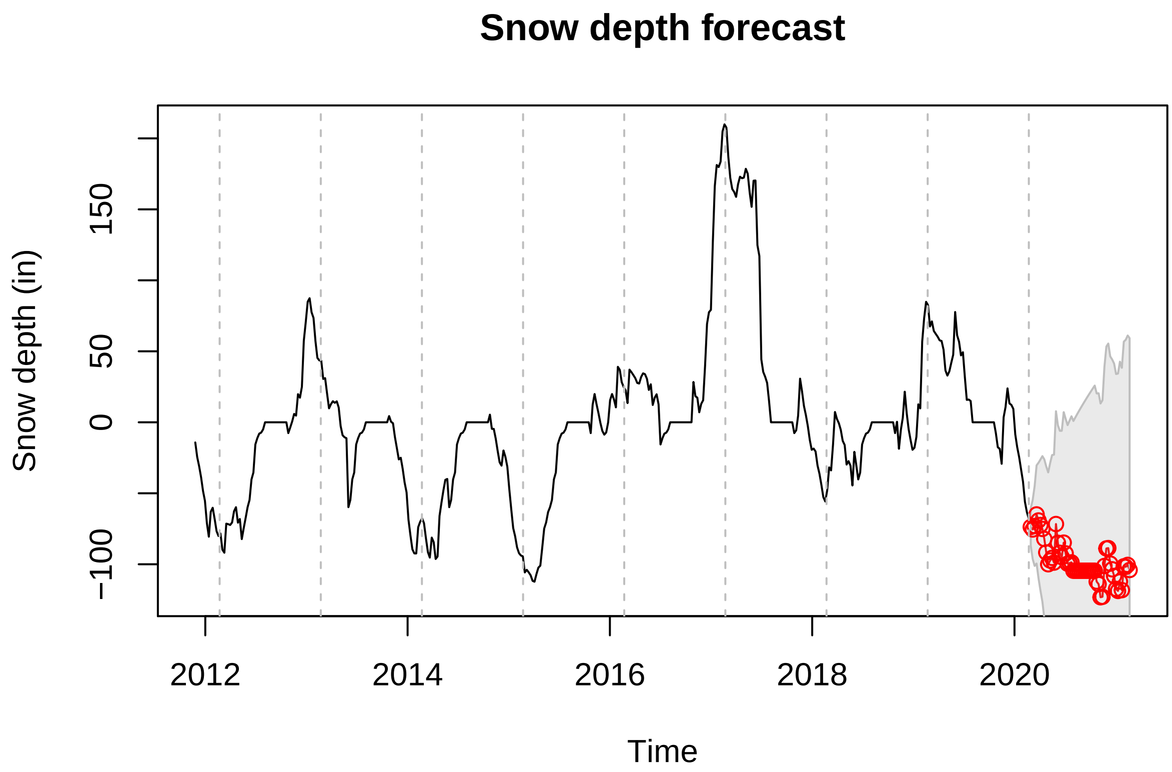
```r
lapply(models, function(x) {
  return(x$aic)
})
```

```
## [[1]]
## [1] 3044.414
```

```r
best_model <- models[[1]]

weeks_ahead <- 52
fore <- predict(best_model, n.ahead= weeks_ahead)

first_date <- time(series)[1]
last_date <- time(series)[length(time(series))]

ts.plot(detrended, fore$pred, col=1:2, xlim=c(last_date - 52 / 52, last_date + weeks_ahead / 52), ylab=
U = fore$pred+fore$se; L = fore$pred-fore$se
xx = c(time(U), rev(time(U))); yy = c(L, rev(U))
polygon(xx, yy, border = 8, col = gray(.6, alpha = .2))
lines(fore$pred, type="p", col=2)
```




```r
ts.plot(detrended, fore$pred, col=1:2, xlim=c(first_date,last_date + weeks_ahead / 52), ylab="Snow depth

U = fore$pred+fore$se; L = fore$pred-fore$se
xx = c(time(U), rev(time(U))); yy = c(L, rev(U))

polygon(xx, yy, border = 8, col = gray(.6, alpha = .2))
lines(fore$pred, type="p", col=2)

for (y in 0:8) {
  abline(v = last_date - y, lty="dashed", col="gray")
}
```


Snow depth forecast
Snow depth (in)
150
50
0
−100
2012
2014
2016
2018
2020
Time