ADS 506 - Final Project EDA

Emanuel Lucban, Sean Torres, Christopher Richardson

```
library(astsa)
library(caret)
library(corrplot)
library(DataExplorer)
library(DMwR)
library(dplyr)
library(ggplot2)
library(forecast)
library(neuralnet)
library(tidyr)
library(tidyverse)

set.seed(1)
```

Loading Data

DF summary/information and dimensions

```
df <- read_csv('Data/water_quality_2011_2019_datasd.csv')
summary(df)
```

```
##      sample            station             depth_m       date_sample
##  Min.   :1.011e+08   Length:338978      Min.   : 1.00   Min.   :2011-01-01
##  1st Qu.:1.301e+09   Class  :character  1st Qu.: 2.00   1st Qu.:2013-01-23
##  Median :1.505e+09   Mode   :character  Median : 9.00   Median :2015-05-08
##  Mean   :1.416e+09                      Mean   :13.18   Mean   :2015-05-10
##  3rd Qu.:1.708e+09                      3rd Qu.:18.00   3rd Qu.:2017-08-02
##  Max.   :1.912e+09                      Max.   :98.00   Max.   :2019-12-30
##                                         NA's   :29894
##      time             project            parameter          qualifier
##  Length:338978      Length:338978      Length:338978      Length:338978
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##      value              units
##  Min.   :     0.00   Length:338978
##  1st Qu.:     2.00   Class :character
##  Median :     8.17   Mode  :character
##  Mean   :    77.16
##  3rd Qu.:    25.07
##  Max.   :180000.00
##  NA's   :670
```

```
dim(df)
```

```
## [1] 338978     10
```

Station Count

```
length <- length(unique(df$station))
line <- paste('There is a total of', length, 'stations!')
cat(line,'\n\nHere are the stations from the dataset.')
```

```
## There is a total of 105 stations!
##
## Here are the stations from the dataset.
```

```
sort(unique(df$station))
```

```
##   [1] "A1"   "A6"   "A7"   "C4"   "C5"   "C6"   "C7"   "C8"   "D10"  "D11" 
##  [11] "D12"  "D4"   "D5"   "D7"   "D8"   "D8-A" "D8-B" "D9"   "F01"  "F02" 
##  [21] "F03"  "F04"  "F05"  "F06"  "F07"  "F08"  "F09"  "F10"  "F11"  "F12" 
##  [31] "F13"  "F14"  "F15"  "F16"  "F17"  "F18"  "F19"  "F20"  "F21"  "F22" 
##  [41] "F23"  "F24"  "F25"  "F26"  "F27"  "F28"  "F29"  "F30"  "F31"  "F32" 
##  [51] "F33"  "F34"  "F35"  "F36"  "I1"   "I10"  "I11"  "I12"  "I13"  "I14" 
##  [61] "I15"  "I16"  "I17"  "I18"  "I19"  "I2"   "I20"  "I21"  "I22"  "I23" 
##  [71] "I24"  "I25"  "I26"  "I27"  "I28"  "I29"  "I3"   "I30"  "I31"  "I32" 
##  [81] "I33"  "I34"  "I35"  "I36"  "I37"  "I38"  "I39"  "I4"   "I40"  "I5"  
##  [91] "I6"   "I7"   "I8"   "I9"   "S0"   "S10"  "S11"  "S12"  "S2"   "S3"  
## [101] "S4"   "S5"   "S6"   "S8"   "S9"
```

Parameter Count

```
length <- length(unique(df$parameter))
line <- paste('There is a total of', length, 'parameters!')
cat(line,'\n\nHere are the parameters from the dataset.')
```

```
## There is a total of 12 parameters!
##
## Here are the parameters from the dataset.
```

```
sort(unique(df$parameter))
```

```
##  [1] "CHLOROPHYLL" "DENSITY"     "DO"          "ENTERO"      "FECAL"      
##  [6] "OG"          "PH"          "SALINITY"    "SUSO"        "TEMP"       
## [11] "TOTAL"       "XMS"
```

Kelp Stations and Parameters

```
# We are only interested in kelp stations (total count = 7)
kelp_stations <- c("I19", "I24", "I25", "I26", "I32", "I39", "I40")

# Variables/Parameters utilized
parameters <- c("CHLOROPHYLL",
                "DO",
                "ENTERO",
                "FECAL",
                "PH",
                "SALINITY",
                "TEMP")

df <- df[df$parameter %in% parameters,]
df <- df[df$station %in% kelp_stations,]
```

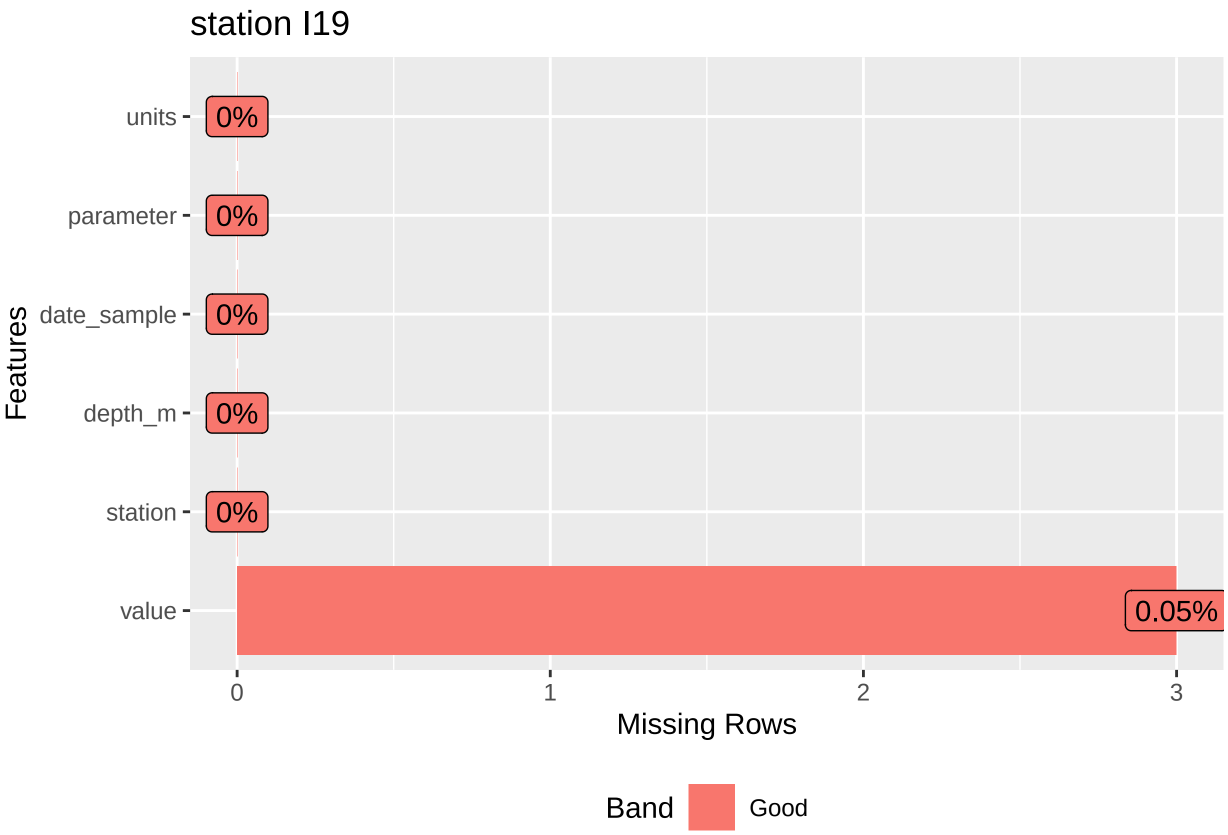
In total we only want the 7 kelp stations within the data set as well as the 7 types of measurements.

```
# Remove individual Date & Time features + project since all buoys are related to SBOO
columns2drop <- c('qualifier', 'project', 'sample', 'time')
df <- drop_columns(df, columns2drop)
head(df,3)
```

```
## # A tibble: 3 x 6
##   station depth_m date_sample parameter value units
##   <chr>     <dbl> <date>      <chr>     <dbl> <chr>
## 1 I25           2 2011-01-01  ENTERO       24 CFU/100 mL
## 2 I25           6 2011-01-01  ENTERO      110 CFU/100 mL
## 3 I25           9 2011-01-01  ENTERO      100 CFU/100 mL
```

Null Ratios by Station

```
for (s in kelp_stations){
  bouy <- df[df$station == s,]

  plot_missing(bouy, title=paste('station', s))
}
```



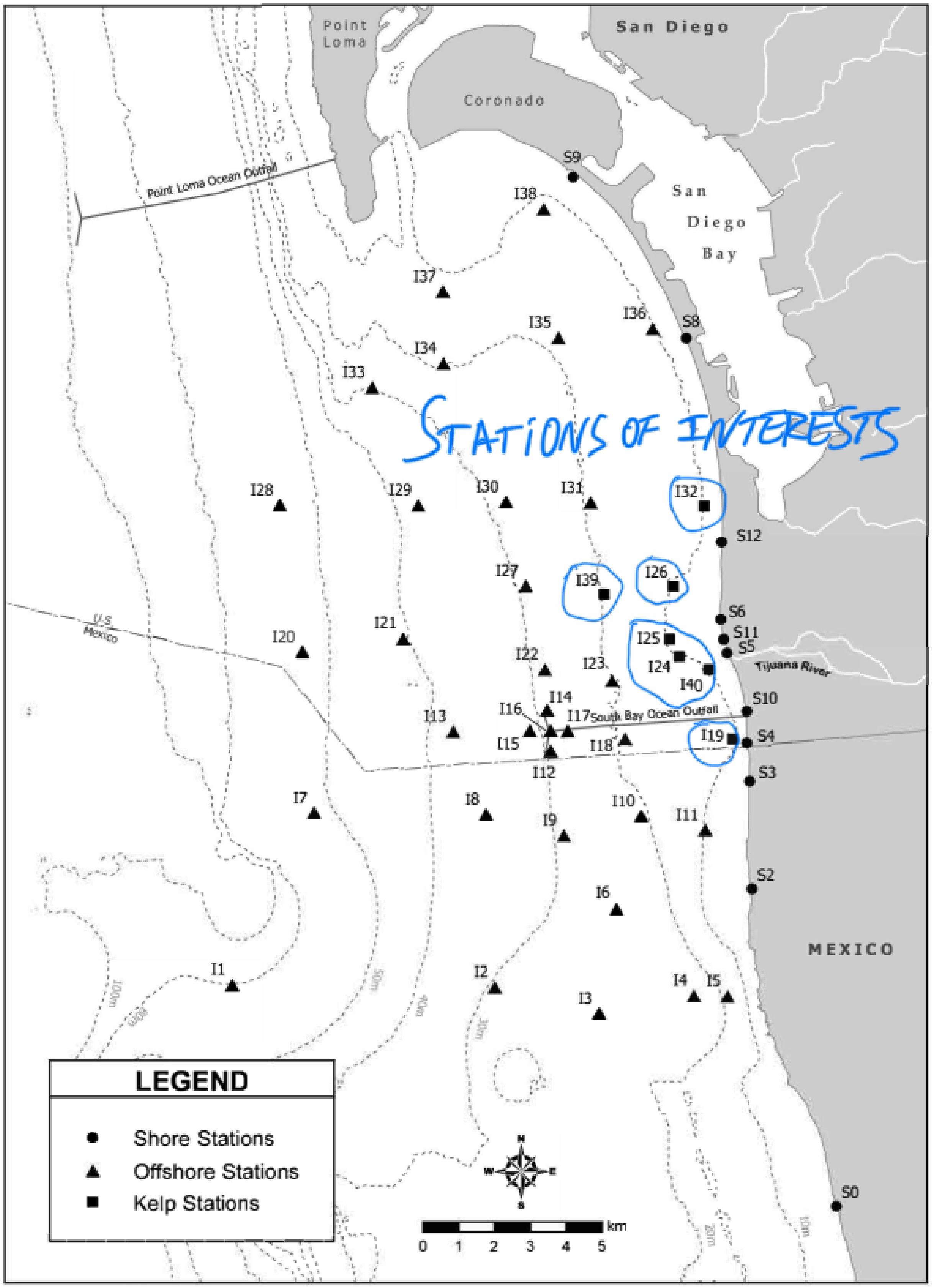
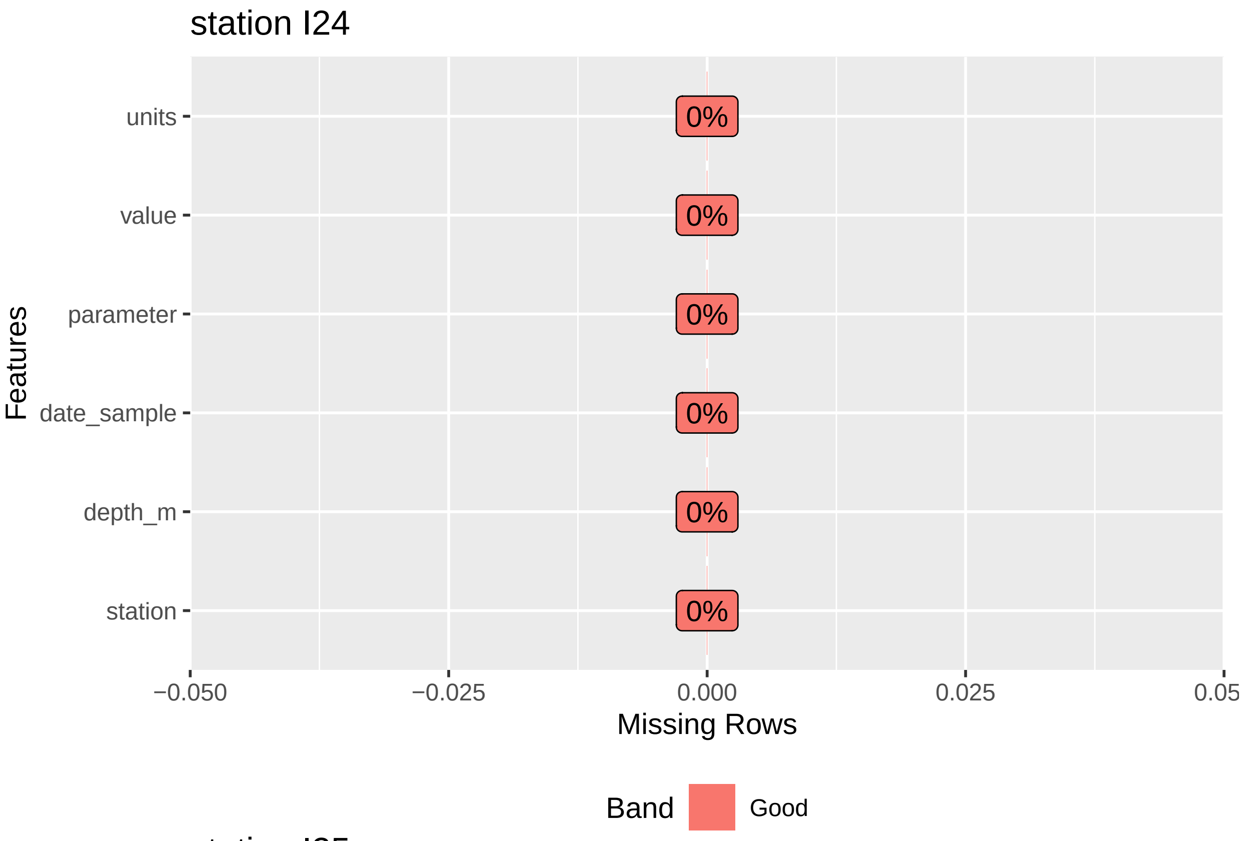
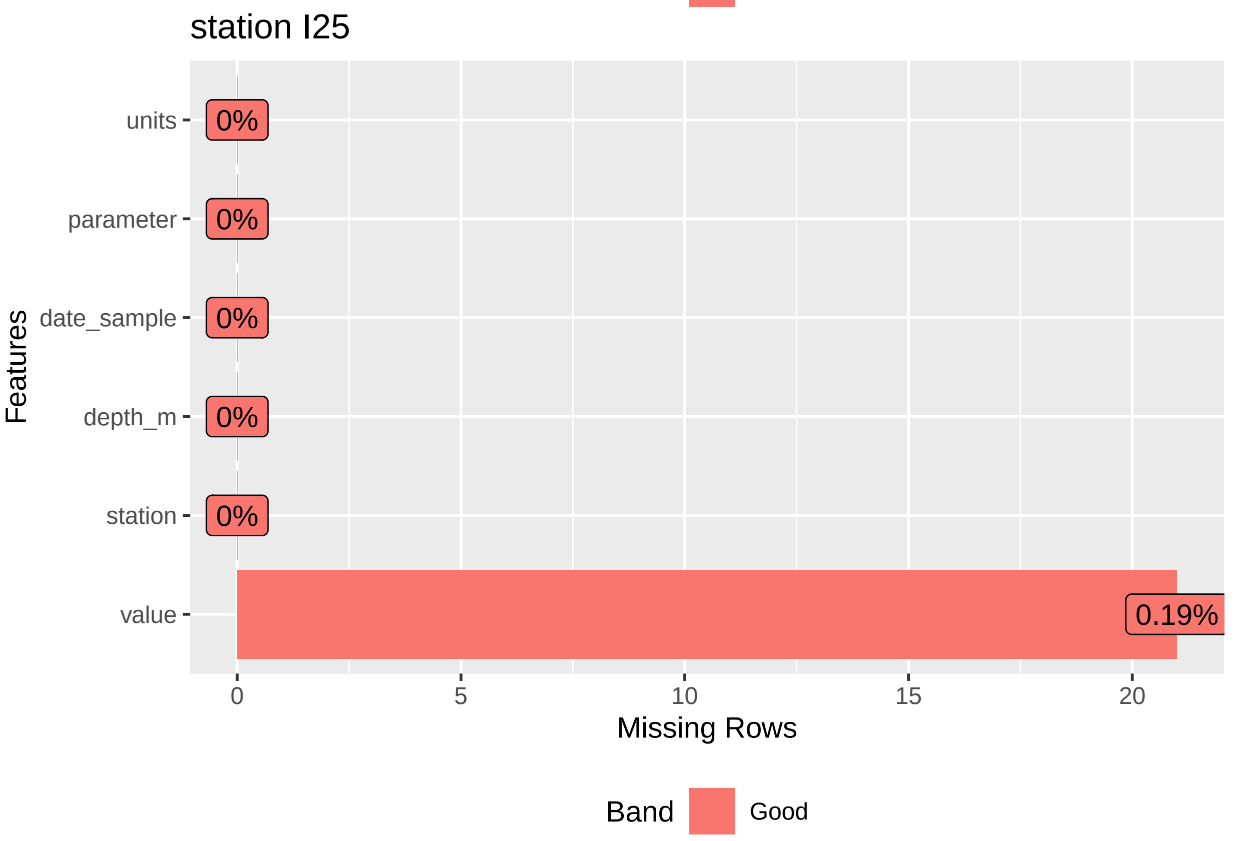
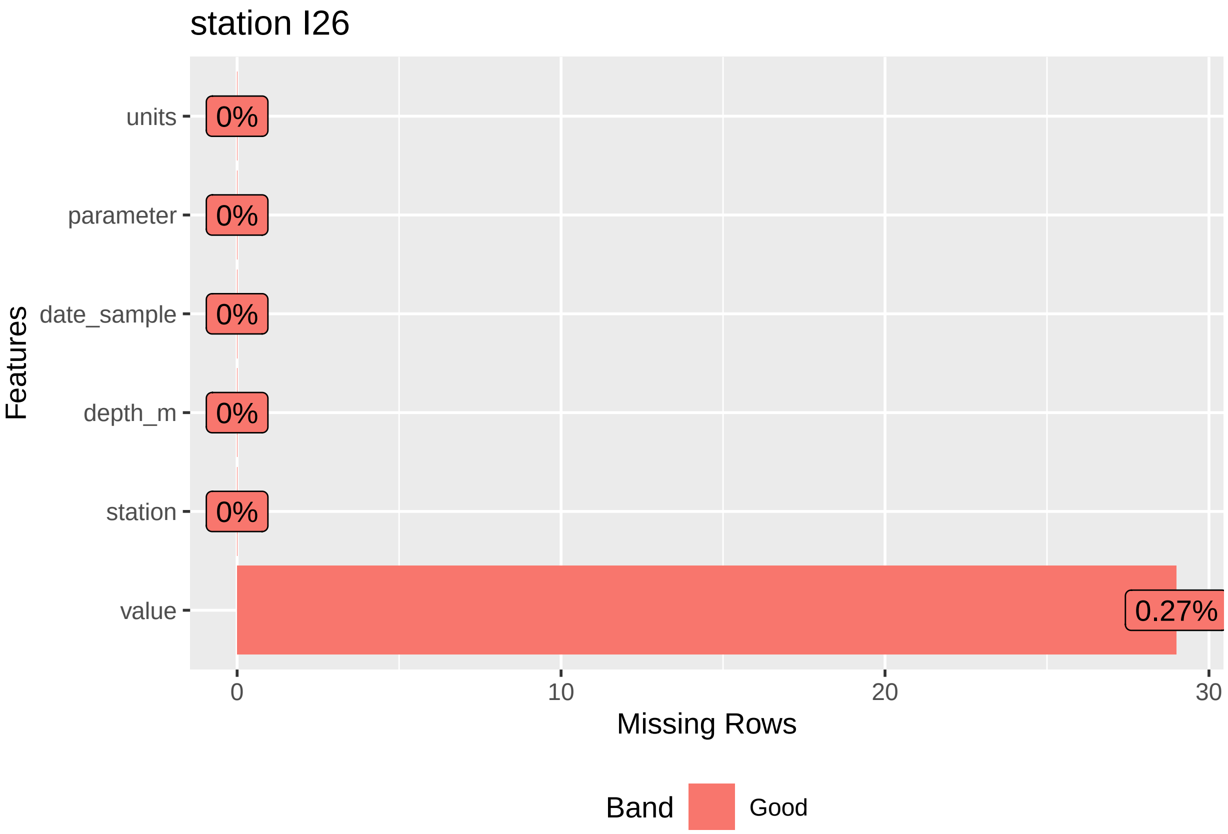
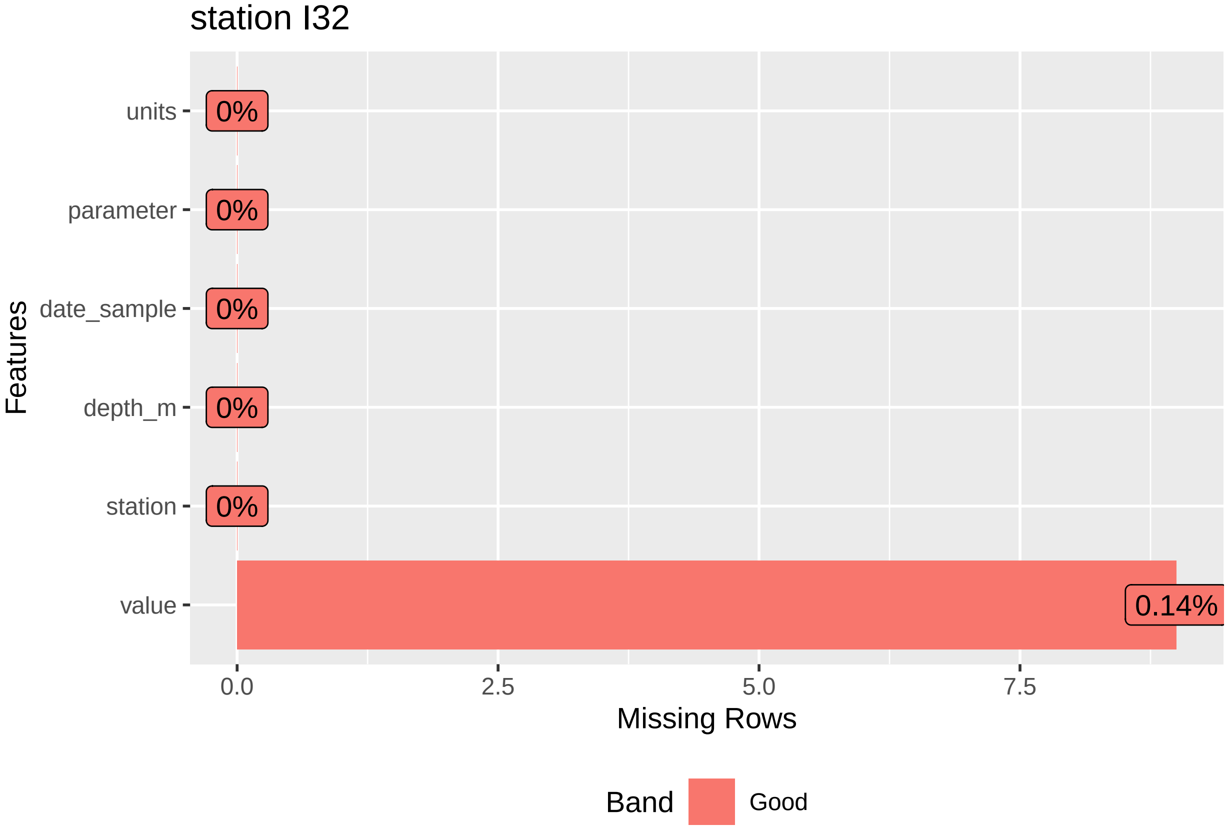


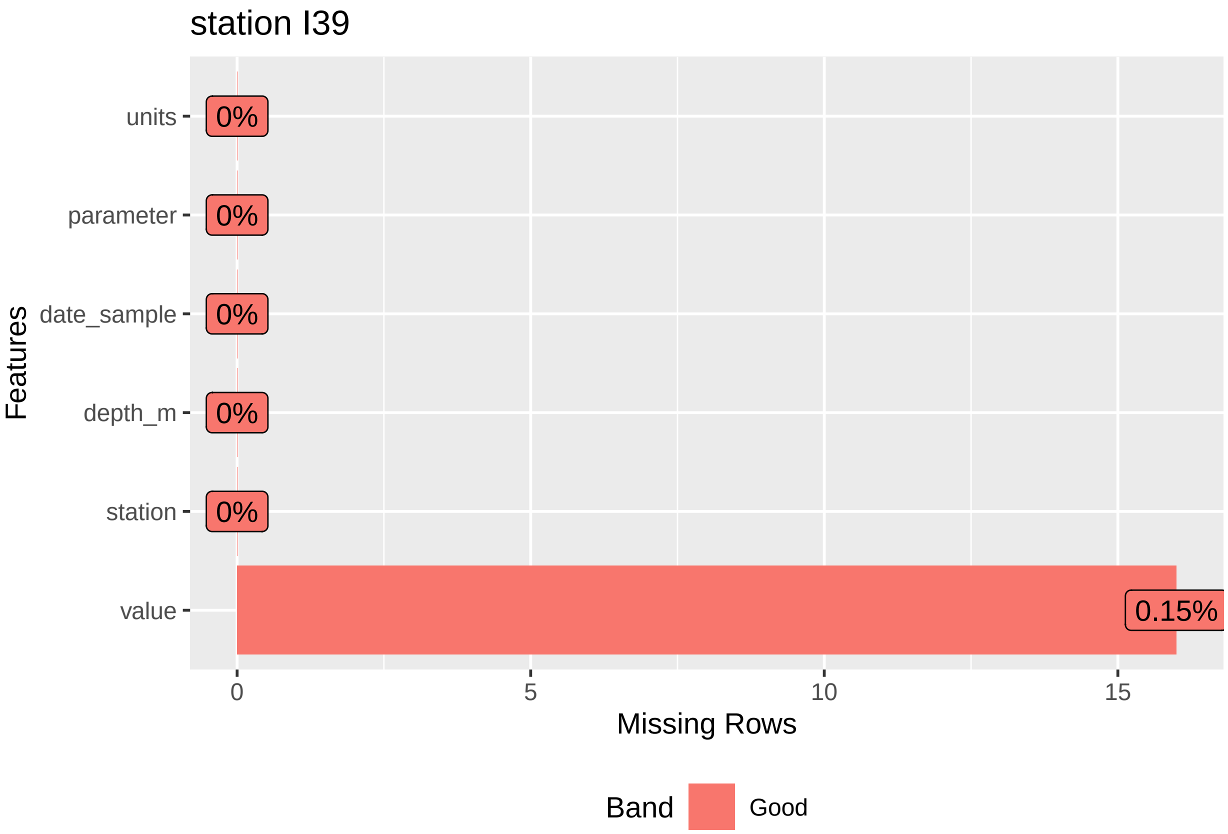
Figure 1: Map of interests

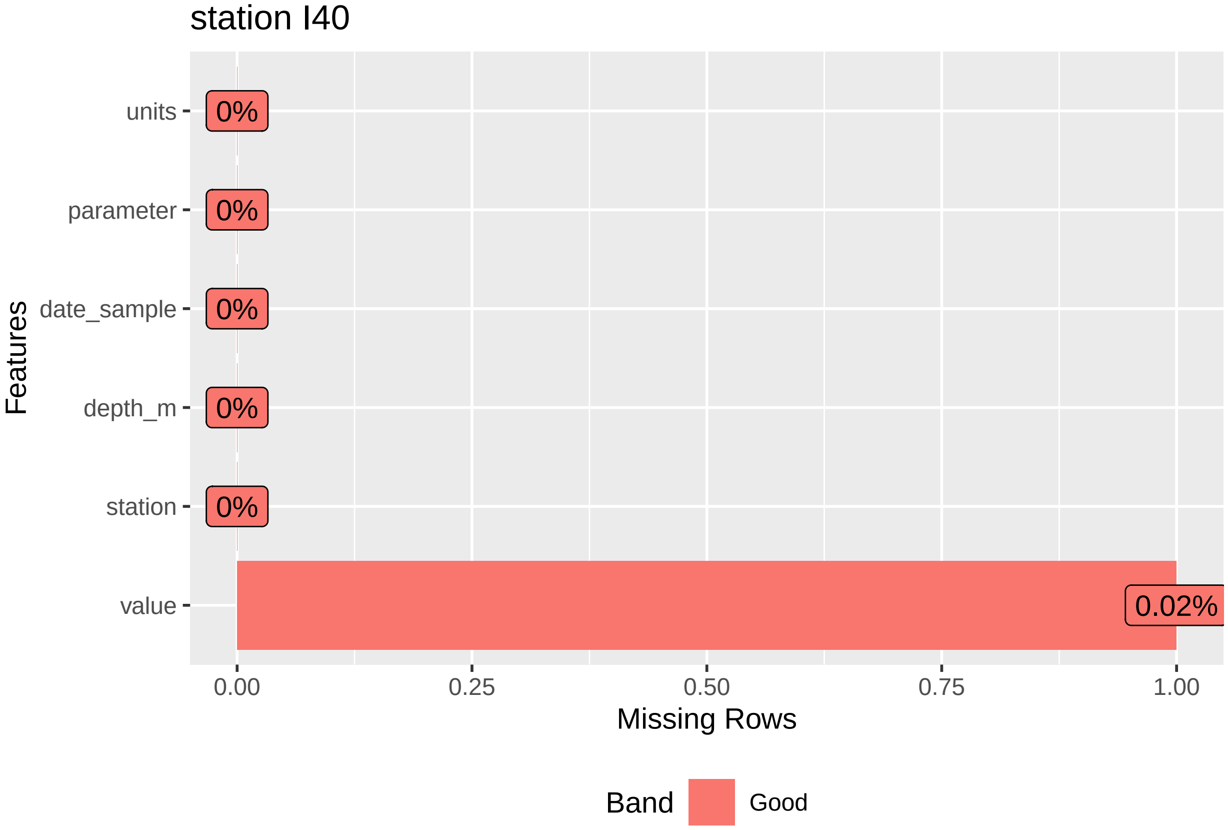

station I24
Features
units
value
parameter
date_sample
depth_m
station
0%
0%
0%
0%
0%
0%
−0.050
−0.025
0.000
0.025
0.05
Missing Rows
Band
Good


station I25
Features
units
parameter
date_sample
depth_m
station
value
0%
0%
0%
0%
0%
0.19%
0
5
10
15
20
Missing Rows
Band
Good

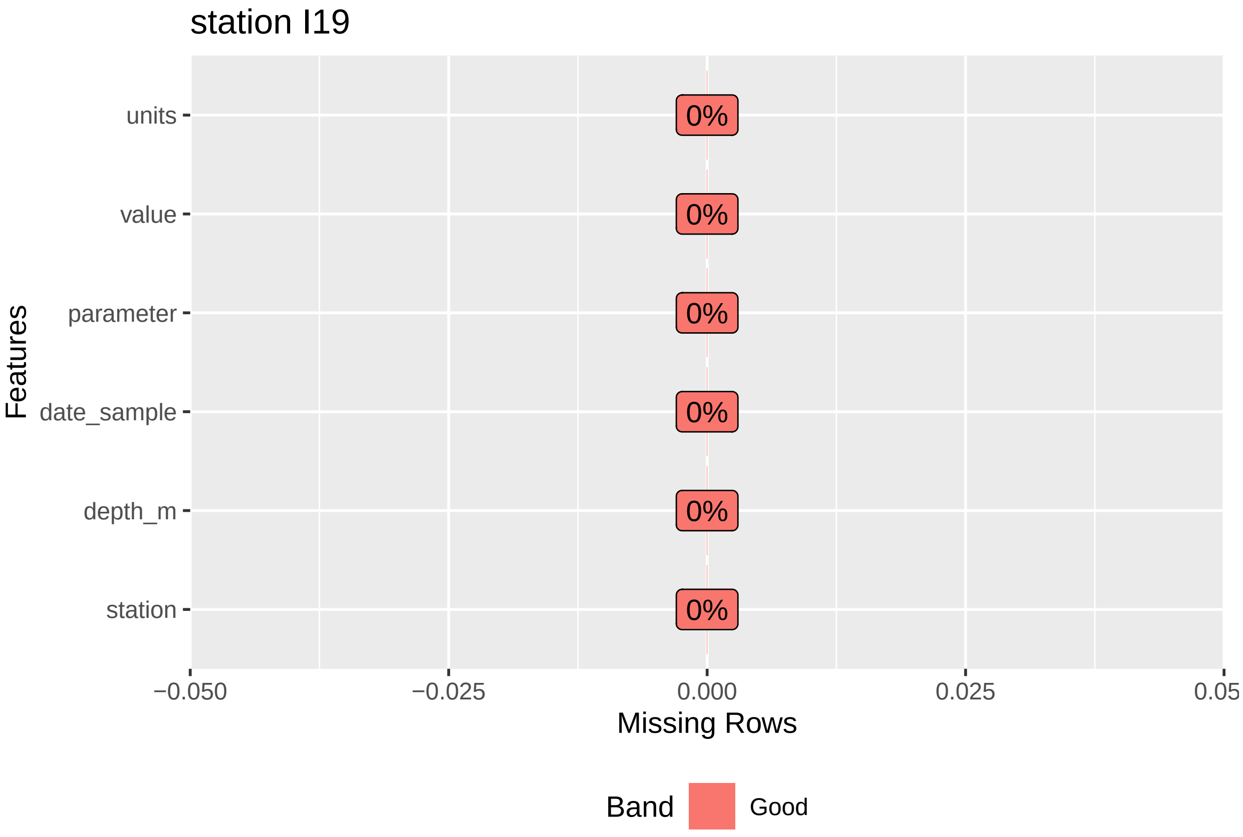

station I26
Features
units
parameter
date_sample
depth_m
station
value
0%
0%
0%
0%
0%
0.27%
0
10
20
30
Missing Rows
Band
Good

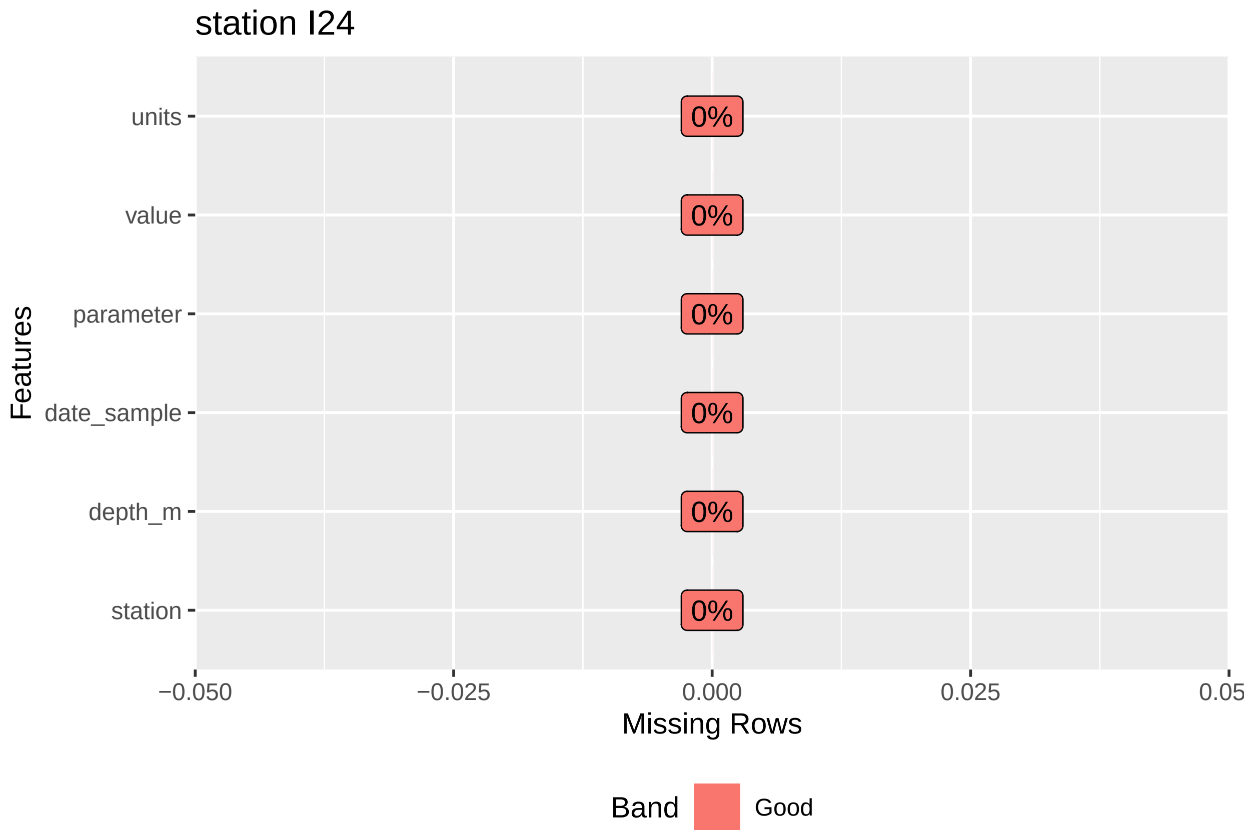

station I32
Features
units
parameter
date_sample
depth_m
station
value
0%
0%
0%
0%
0%
0.14%
0.0
2.5
5.0
7.5
Missing Rows
Band
Good

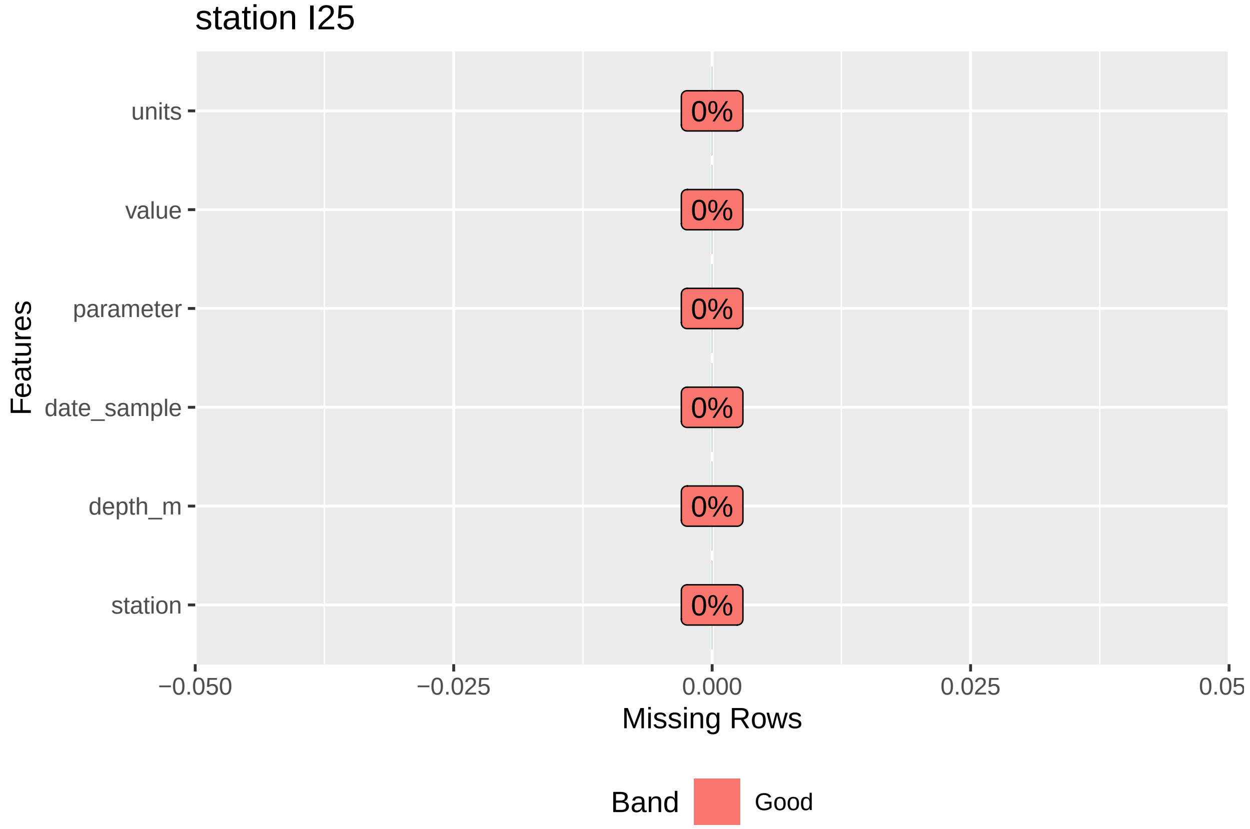

station I39
Features
units
parameter
date_sample
depth_m
station
value
0%
0%
0%
0%
0%
0.15%
0
5
10
15
Missing Rows
Band
Good

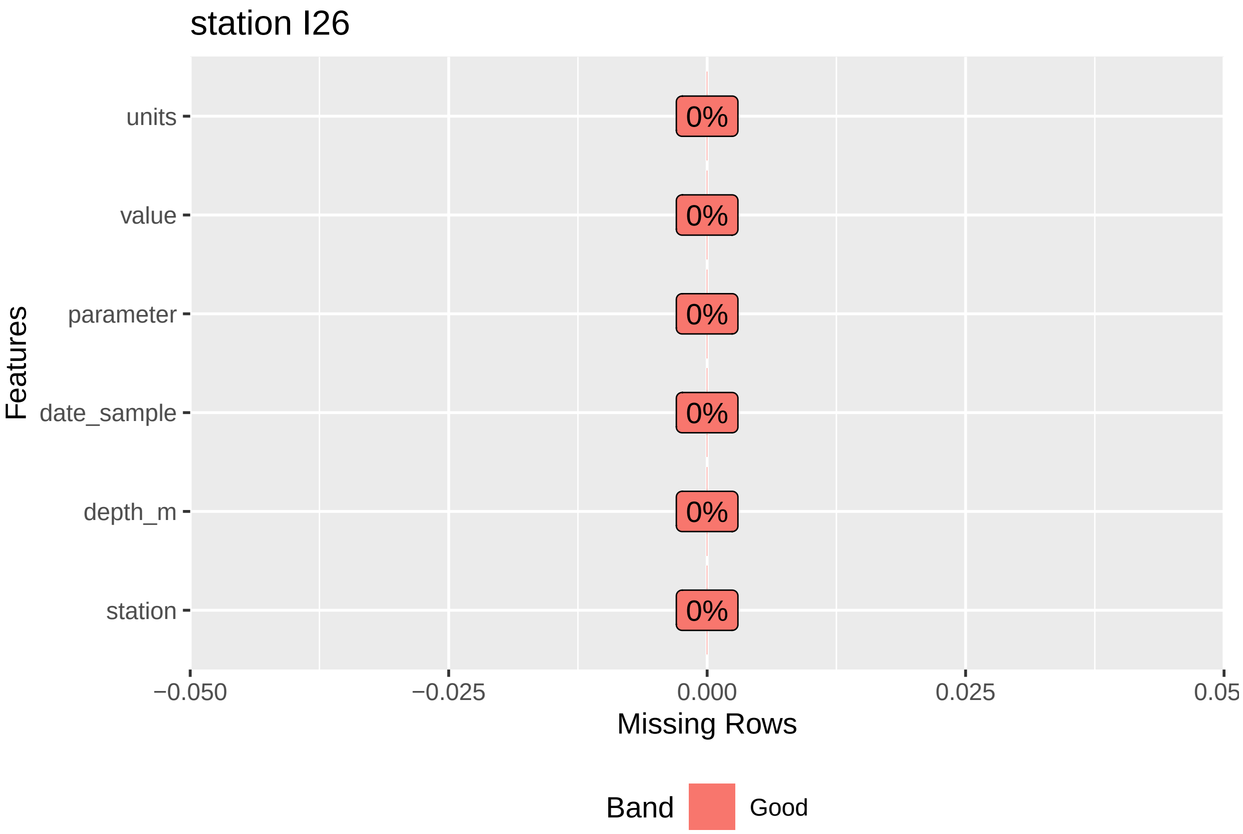

station I40
Features
units
parameter
date_sample
depth_m
station
value
0%
0%
0%
0%
0%
0.02%
0.00
0.25
0.50
0.75
1.00
Missing Rows
Band
Good

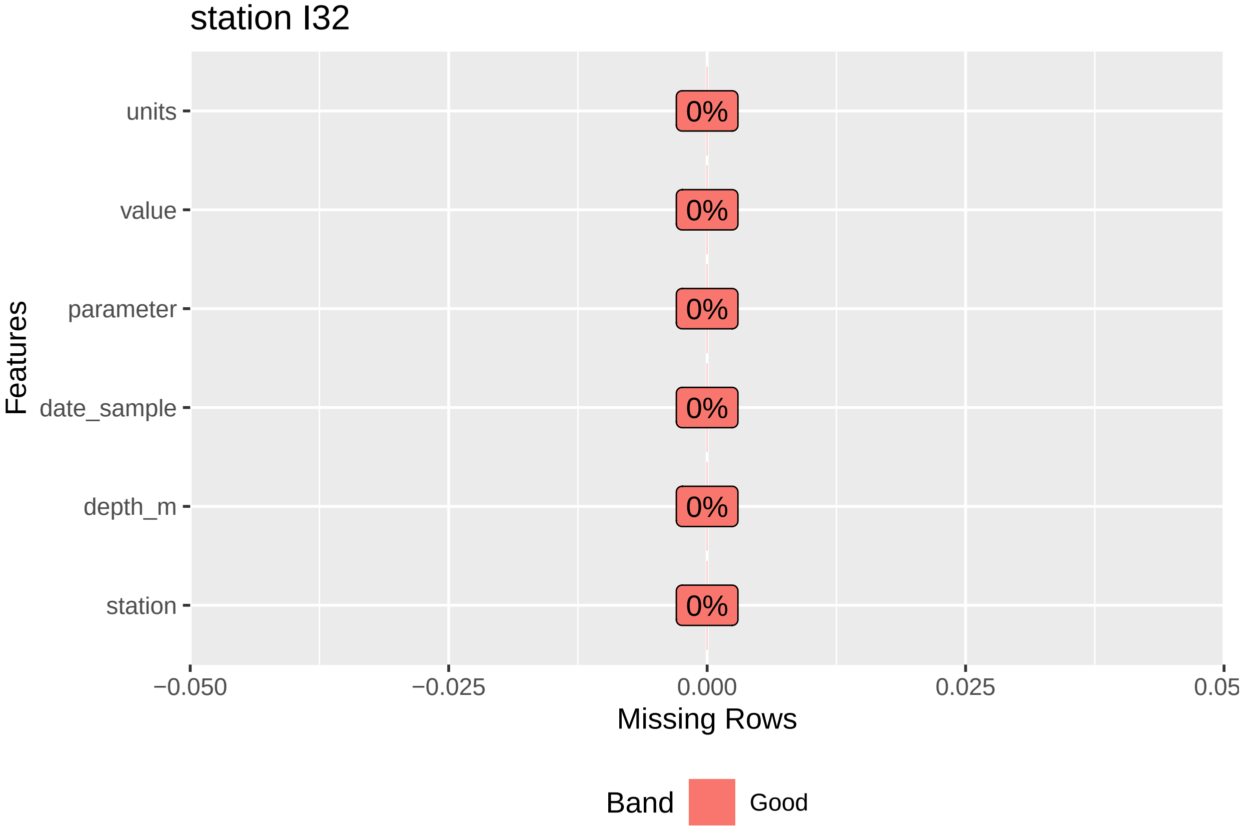
```r
# Removal of null values
df <- df[!is.na(df$value),]

# validation of dropping of NA rows
for (s in kelp_stations){
  bouy <- df[df$station == s,]

  plot_missing(bouy, title=paste('station', s))
}
```

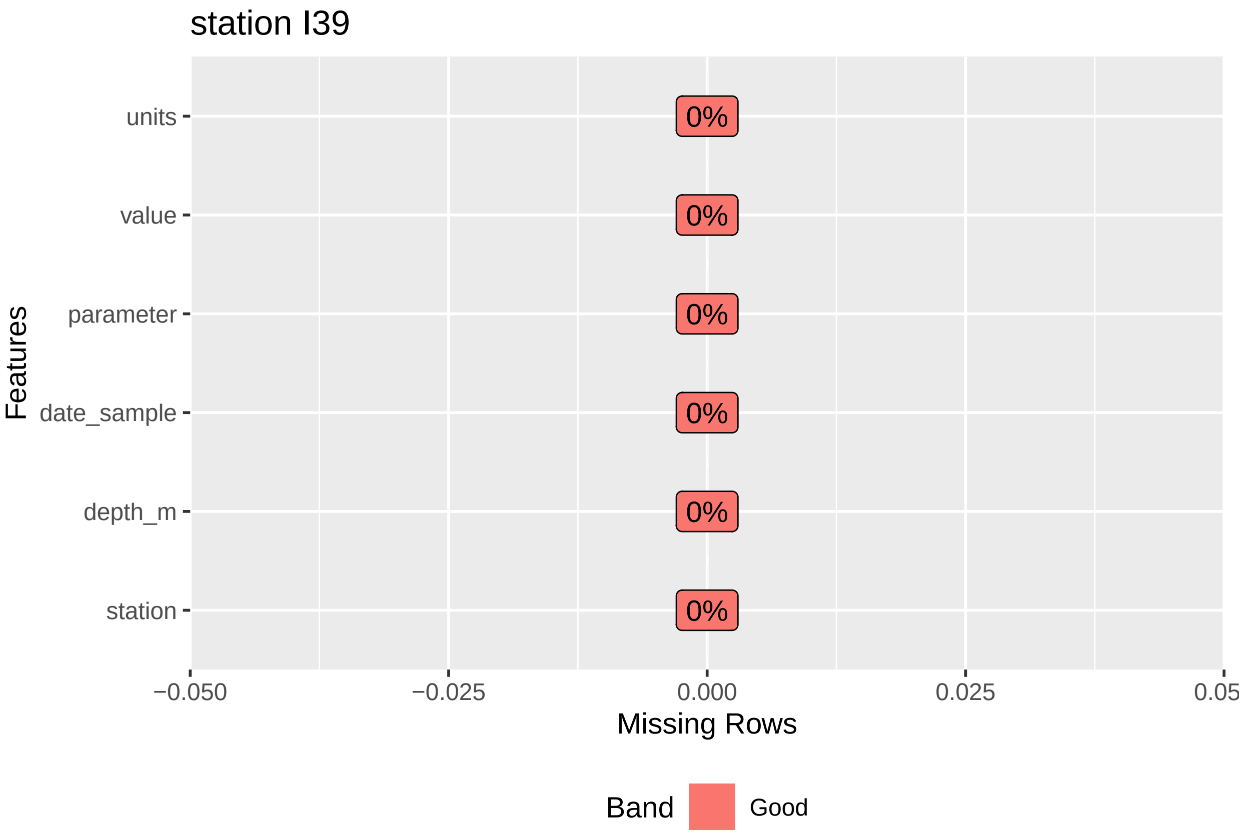


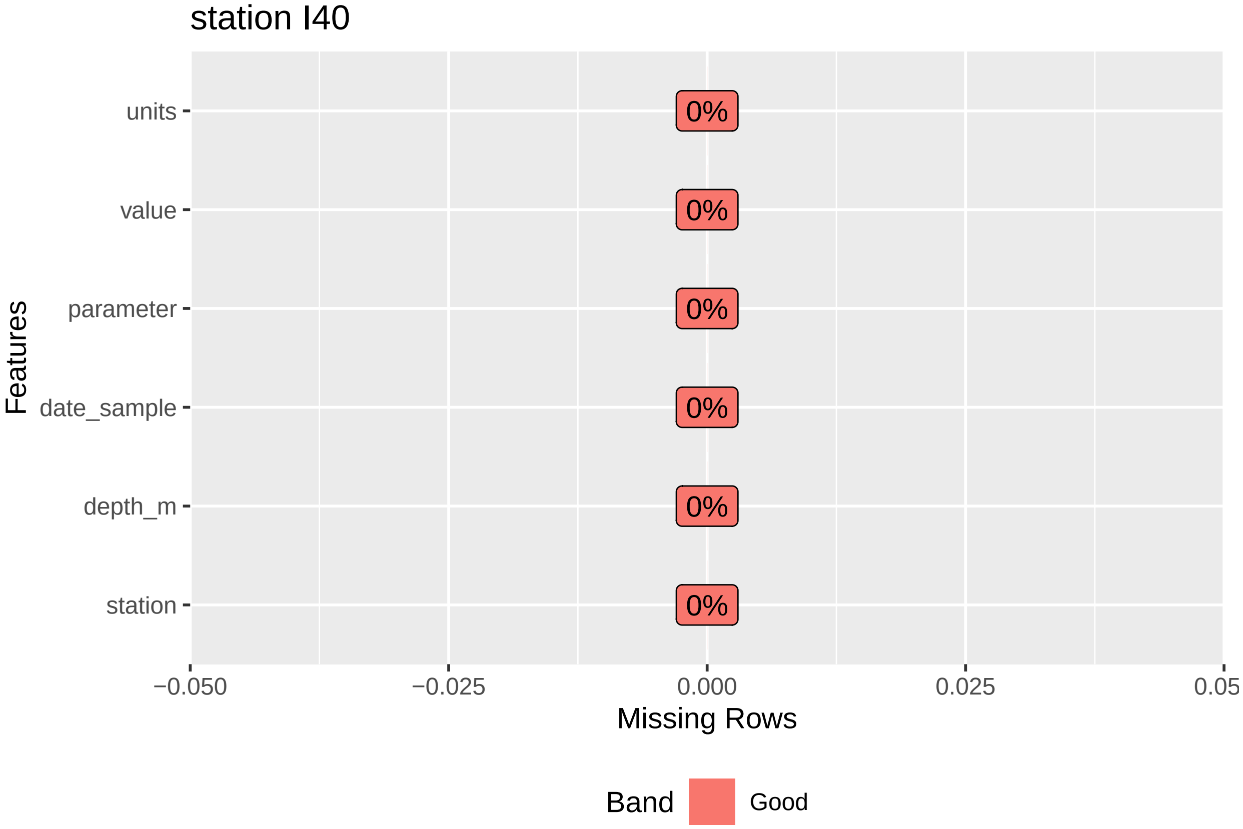

station I24
Features
units
value
parameter
date_sample
depth_m
station
0%
0%
0%
0%
0%
0%
−0.050
−0.025
0.000
0.025
0.05
Missing Rows
Band
Good


station I25
Features
units
value
parameter
date_sample
depth_m
station
0%
0%
0%
0%
0%
0%
−0.050
−0.025
0.000
0.025
0.05
Missing Rows
Band
Good


station I26
Features
units
value
parameter
date_sample
depth_m
station
0%
0%
0%
0%
0%
0%
−0.050
−0.025
0.000
0.025
0.05
Missing Rows
Band
Good


station I32
Features
units
value
parameter
date_sample
depth_m
station
0%
0%
0%
0%
0%
0%
−0.050
−0.025
0.000
0.025
0.05
Missing Rows
Band
Good


station I39
Features
units
value
parameter
date_sample
depth_m
station
0%
0%
0%
0%
0%
0%
−0.050
−0.025
0.000
0.025
0.05
Missing Rows
Band
Good


station I40
Features
units
value
parameter
date_sample
depth_m
station
0%
0%
0%
0%
0%
0%
−0.050
−0.025
0.000
0.025
0.05
Missing Rows
Band
Good

Date Sampling Deltas/Intervals for Disolved Oxygen

Here we display the inconsistent sampling intervals which has led us into aggregating all stations as one, allowing for less likelihood of not having 4 samples per month.

```
# create I19 & I25 independent DFs for delta manipulation
I19 <- df %>%
  arrange(date_sample) %>%
  filter(station == "I19") %>%
  filter(parameter == "DO") %>%
  filter(date_sample >= "2011-01-01" & date_sample <= "2011-12-31") %>%
  select(-units)

I25 <- df %>%
  arrange(date_sample) %>%
  filter(station == "I25") %>%
  filter(parameter == "DO") %>%
  filter(date_sample >= "2011-01-01" & date_sample <= "2011-12-31") %>%
  select(-units)

I25_dates <- unique(I25$date_sample)
cat('\n','-----I25 Date Deltas -----','\n')
```

```
##
##  -----I25 Date Deltas -----
```

```
I25_sampling_deltas <- (I25_dates[2:length(I25_dates)] - I25_dates)
I25_sampling_deltas[1:(length(I25_sampling_deltas)-1)]
```

```
## Time differences in days
##  [1]  8  6  5  8  1  4 12  6  5  2 10  4  6  5  5  5  7  5  8 11  3  9  6  6  3
## [26]  5  6  6  6 10  7  6  6  6  6  4  9  4  7  7  8  4  5  5  3  4  7  6  6 11
## [51]  4  7  4  9  4  5  8  3  3
```

```
cat('\n\n')
```

```
I19_dates <- unique(I19$date_sample)
cat('\n','-----I19 Date Deltas -----','\n')
```

```
##
##  -----I19 Date Deltas -----
```

```
I19_sampling_deltas <- (I19_dates[2:length(I19_dates)] - I19_dates)
I19_sampling_deltas[1:(length(I19_sampling_deltas)-1)]
```

```
## Time differences in days
##  [1] 28 29 35 34 29 28 48 22 21 34 29
```

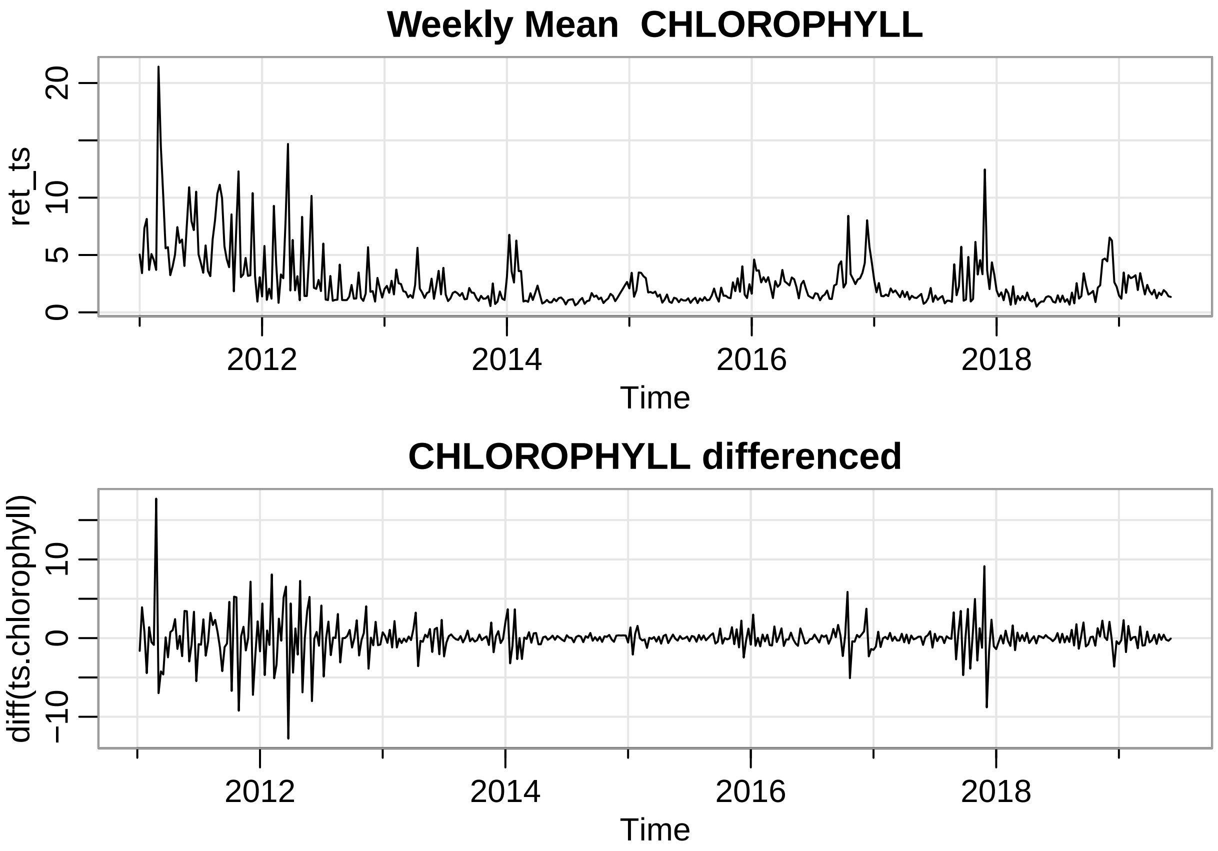
As we can see, DO for station I25 was sampled almost irregularly. With sampling intervals ranging from the sampling the next day to almost 2 weeks out versus the sampling frequency for I19 of once per month at irregular intervals of every 4-7 weeks.

Create time series data for each parameter, resampled to weekly observations, aggregated across all stations. 1 series per parameter

```
# Function to create time series from dataframe
create_ts <- function(parameter){
  p_ts <- wq_df[wq_df$parameter == parameter, c("datetime", "value")]
  p_ts_clean <- p_ts[!is.na(p_ts$datetime), ]
  # Order by datetime then convert to date
  p_ts_clean <- p_ts_clean[order(p_ts_clean$datetime), ]
  p_ts_clean$datetime <- as.Date(p_ts_clean$datetime)

  # Resample to weekly values by aggregating by week and taking the mean
  p_ts_rsmp <- p_ts_clean %>%
    mutate(week = cut.Date(datetime, breaks = "1 week", labels = FALSE)) %>%
    group_by(week) %>%
    summarize(mean = mean(value, na.rm = TRUE))

  # set nan values to mean after resample
  p_ts_rsmp$mean[is.na(p_ts_rsmp$mean)] = mean(p_ts_clean$value, na.rm = TRUE)

  ret_ts <- ts(p_ts_rsmp$mean, start=c(2011, 1), frequency=52)
  tsplot(ret_ts, main = paste("Weekly Mean ", parameter))

  return (ret_ts)
}
```

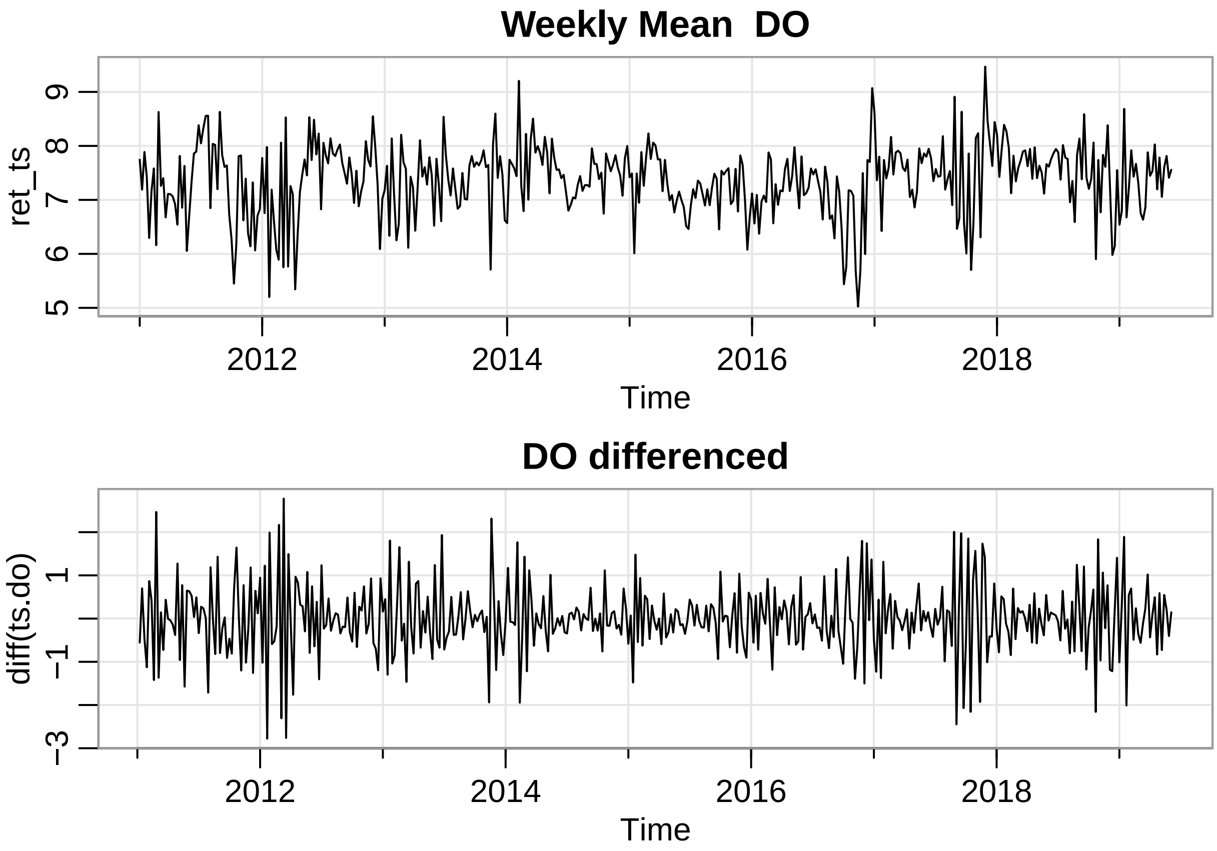
CHLOROPHYLL

```
par(mfrow=c(2, 1))
ts.chlorophyll <- create_ts('CHLOROPHYLL')
tsplot(diff(ts.chlorophyll), main = "CHLOROPHYLL differenced")
```



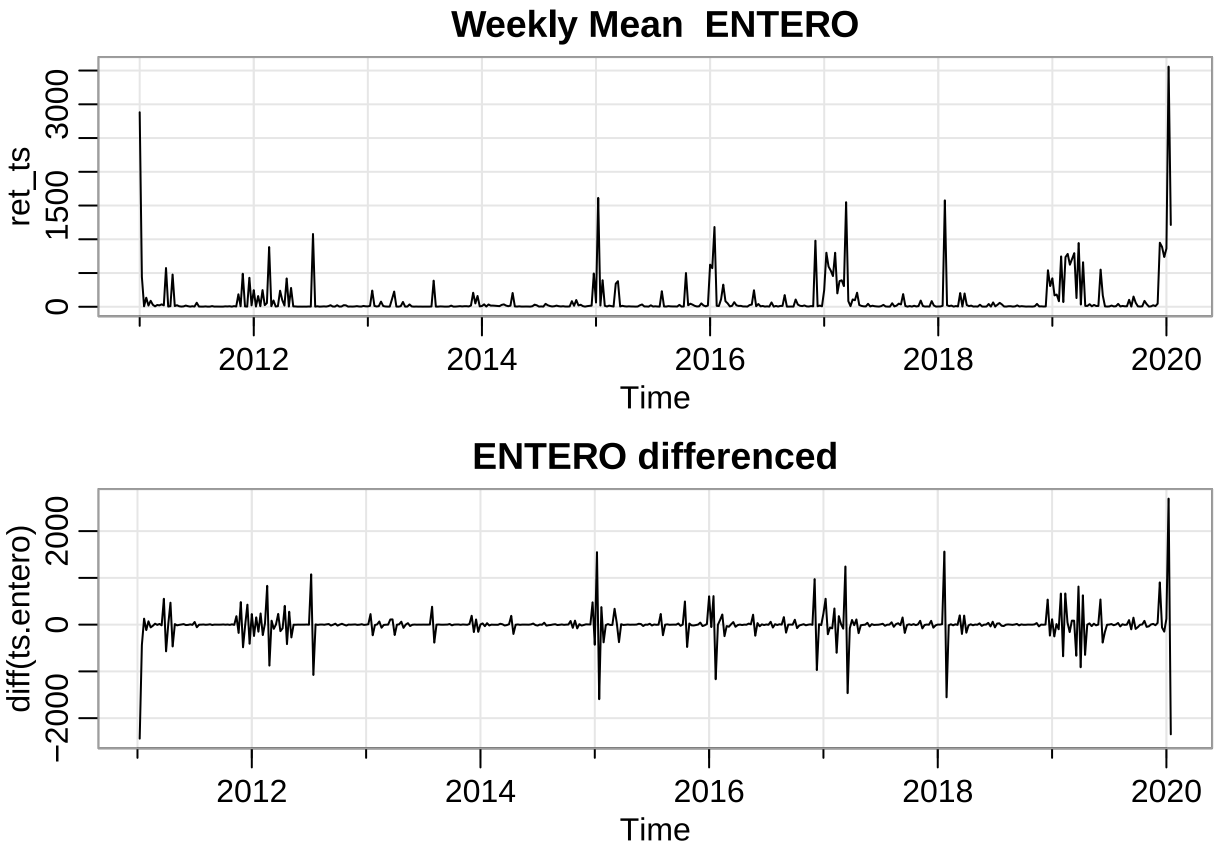
Dissolved Oxygen

```
par(mfrow=c(2, 1))
ts.do <- create_ts('DO')
tsplot(diff(ts.do), main = "DO differenced")
```



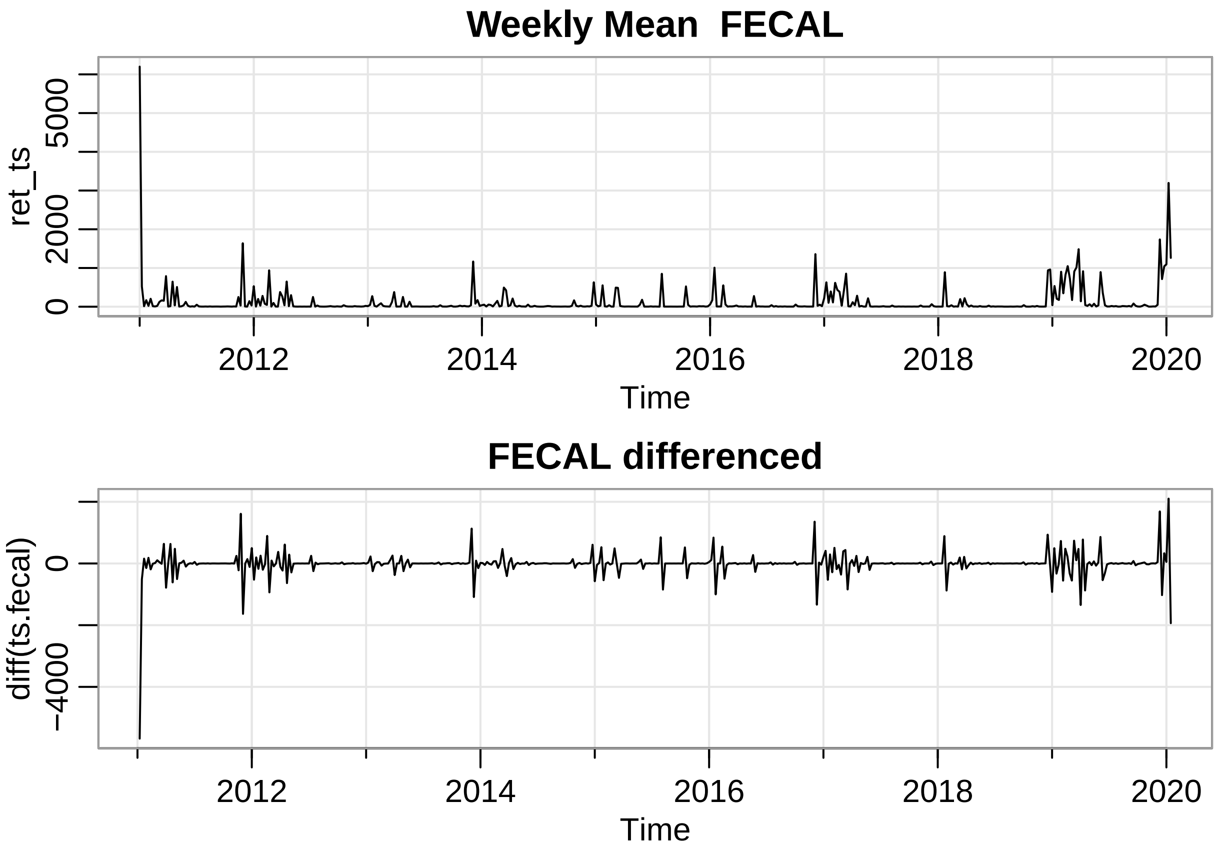
Entero

```
par(mfrow=c(2, 1))
ts.entero <- create_ts('ENTERO')
tsplot(diff(ts.entero), main = "ENTERO differenced")
```



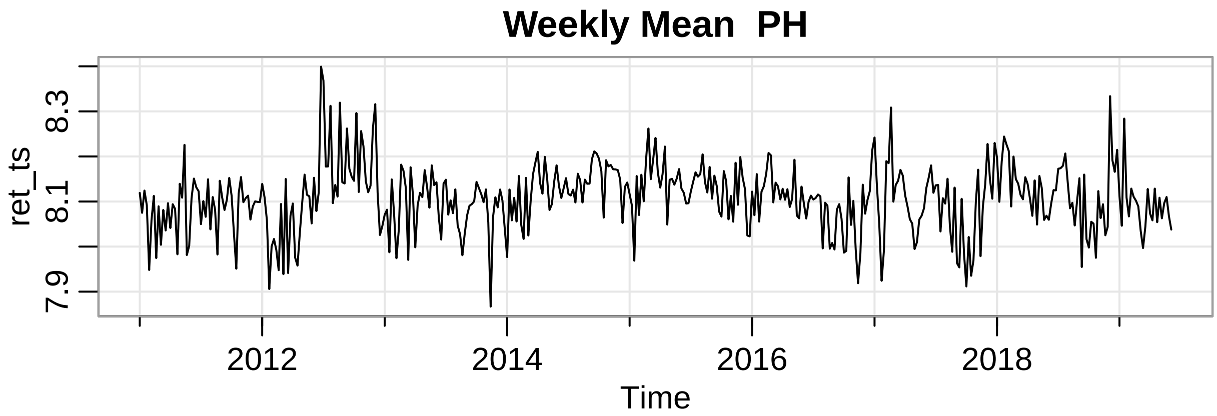
Fecal

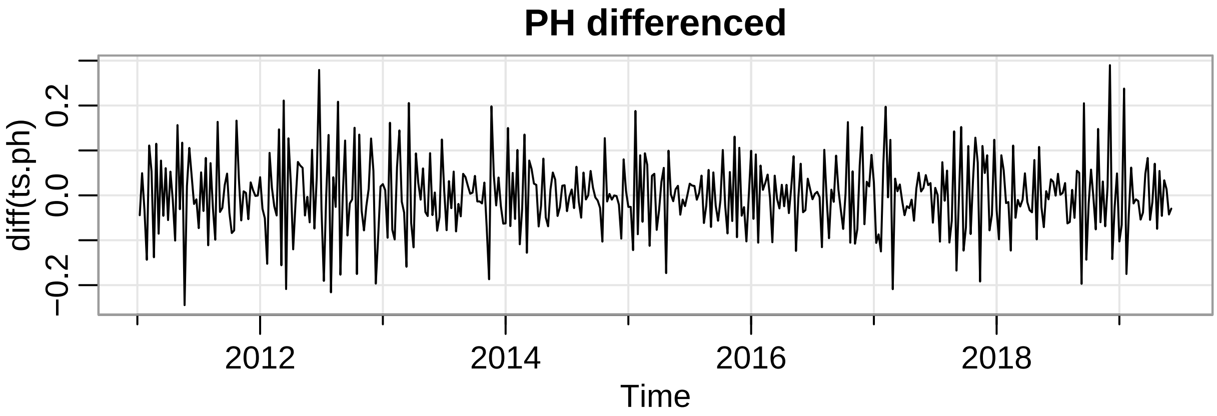
```
par(mfrow=c(2, 1))
ts.fecal <- create_ts('FECAL')
tsplot(diff(ts.fecal), main = "FECAL differenced")
```



PH

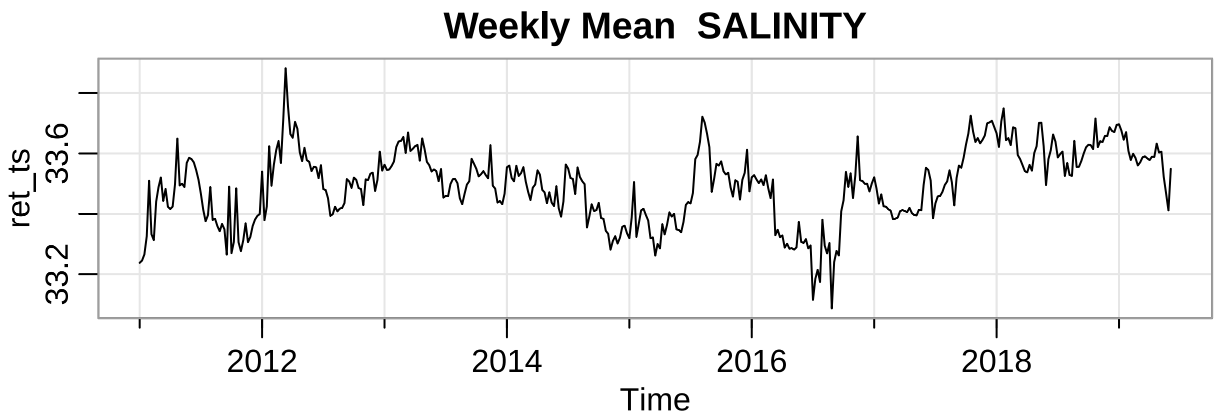
```
par(mfrow=c(2, 1))
ts.ph <- create_ts('PH')
tsplot(diff(ts.ph), main = "PH differenced")
```

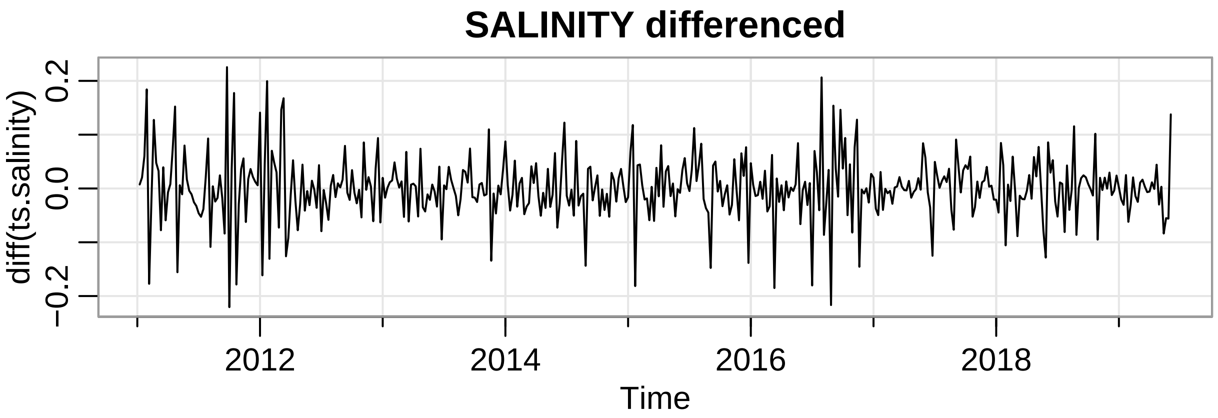




SALINITY

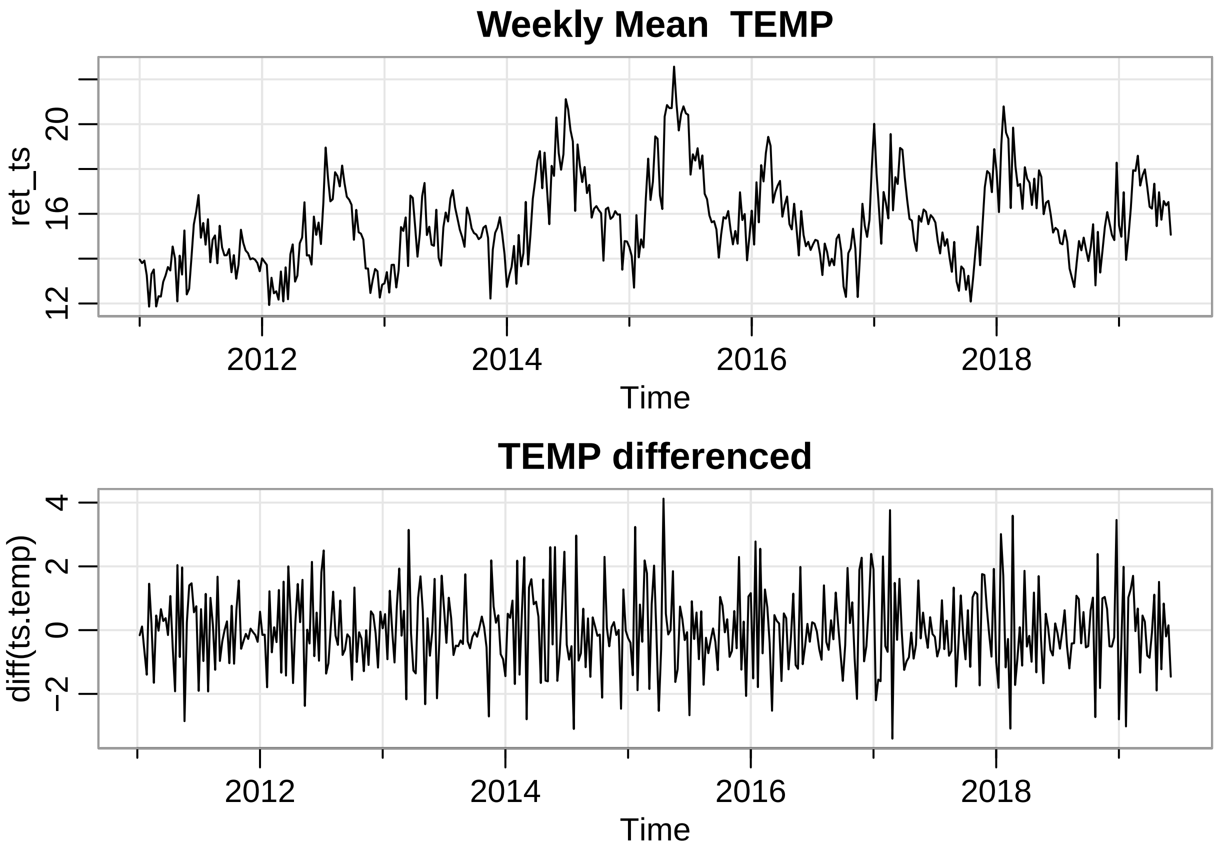
```
par(mfrow=c(2, 1))
ts.salinity <- create_ts('SALINITY')
tsplot(diff(ts.salinity), main = "SALINITY differenced")
```





TEMP

```
par(mfrow=c(2, 1))
ts.temp <- create_ts('TEMP')
tsplot(diff(ts.temp), main = "TEMP differenced")
```

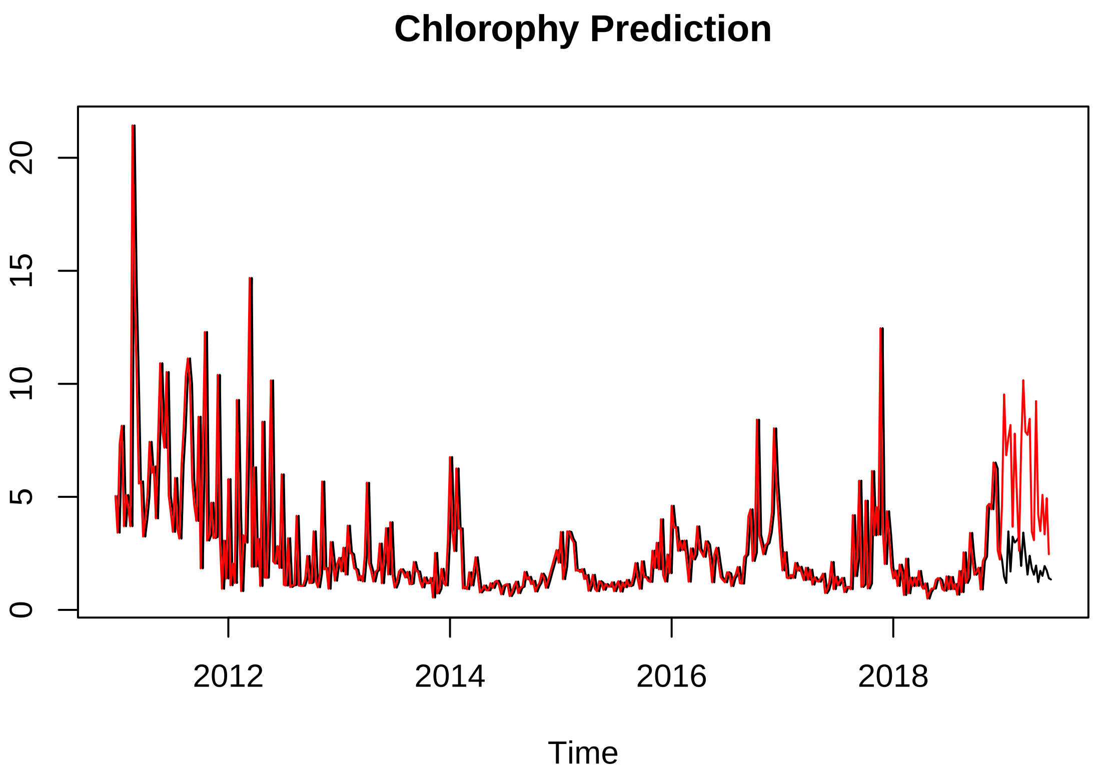


ARIMA + ANN Model

```
# GENERALIZED Function to train ARIMA + ANN model
ARIMANN <- function(ts, forecast){
  # Keep things consistent
  set.seed(42)
  ts.size <- length(diff(ts))

  arima.model <- arima(diff(ts), order=c(1,1,1))
  arima.res <- arima.model$residuals

  # NN Window is 52 + 1 for label
  nn_train_set <- data.frame(matrix(ncol = 53, nrow = 0))

  for (i in 1:(ts.size - 53)) {
    nn_train_set <- rbind(nn_train_set, arima.res[i:(i+52)])
  }

  # Change label col name
  colnames(nn_train_set)[53] <- "Y"

  n <- names(nn_train_set)
  # For some reason R's neuralnet library can't properly parse a formula, we have to explicitly create
  f <- as.formula(paste("Y ~", paste(n[!n %in% "Y"], collapse = " + ")))
  nn.model <- neuralnet(f, data = nn_train_set, linear.output = TRUE, learningrate = 0.01, hidden = 5)

  # Check if forecast is blank
  if (missing(forecast)){
    # Return training ts
    return (ts((diff(ts)[54:ts.size] - arima.res[54:ts.size]) + predict(nn.model, newdata = nn_train_se
  }

  # Get the last 52 residuals to start the rolling predictions
  nn.pred <- tail(arima.res, 52)
  # iterate to forecast horizon for NN prediction
  for (h in 1:forecast){
    pd.input <- data.frame(matrix(ncol = 52, nrow = 0))
    pd.input <- rbind(pd.input, tail(nn.pred, 52))
    colnames(pd.input) <- colnames(nn_train_set)[1:52]

    # append next prediction
    nn.pred <- append(nn.pred, predict(nn.model, newdata = pd.input))
  }

  pred.ts <- predict(arima.model, n.ahead=forecast)$pred + tail(nn.pred, forecast)

  return (pred.ts)
}
```

Modeling Chlorophyll

```
# Only include data to 2018, reserve 2019 for validation
chlor.train <- window(ts.chlorophyll, 2011, c(2018, 52))
chlor.results <- ARIMANN(chlor.train, 23)
chlor.combined <- ts(c(diff(chlor.train), chlor.results), start=start(chlor.train), frequency = frequen

# Invert the differencing
ts.plot(ts.chlorophyll, diffinv(chlor.combined, xi = ts.chlorophyll[1]), gpars = list(col = c("black",
abline(v=as.Date("2019-01-01"))
```



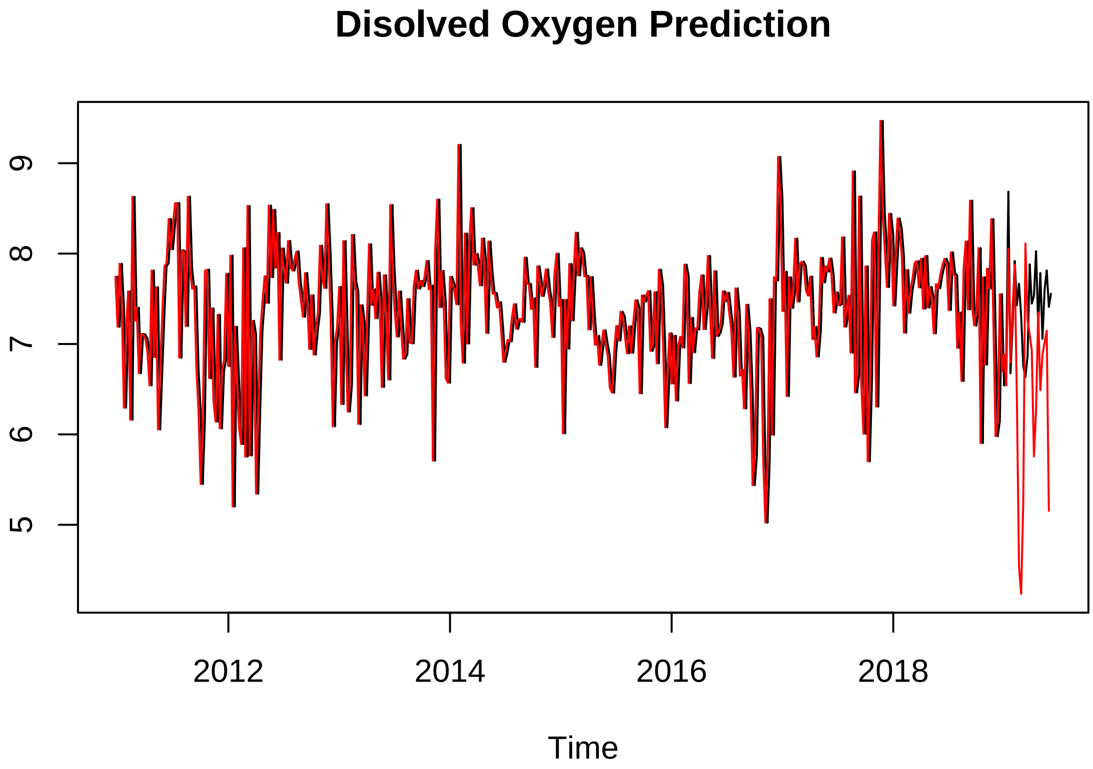
RMSE

```
RMSE(tail(diffinv(chlor.combined, xi = ts.chlorophyll[1]),23), diff(window(ts.chlorophyll, 2019)))
```

```
## [1] 6.351481
```

Modeling Dissolved Oxygen

```
# Only include data to 2018, reserve 2019 for validation
do.train <- window(ts.do, 2011, c(2018, 52))
do.results <- ARIMANN(do.train, 23)
do.combined <- ts(c(diff(do.train), do.results), start=start(do.train), frequency = frequency(do.train)
ts.plot(ts.do, diffinv(do.combined, xi = ts.do[1]), gpars = list(col = c("black", "red")), main = "Disol
abline(v=as.Date("2019-01-01"), col = "blue")
```



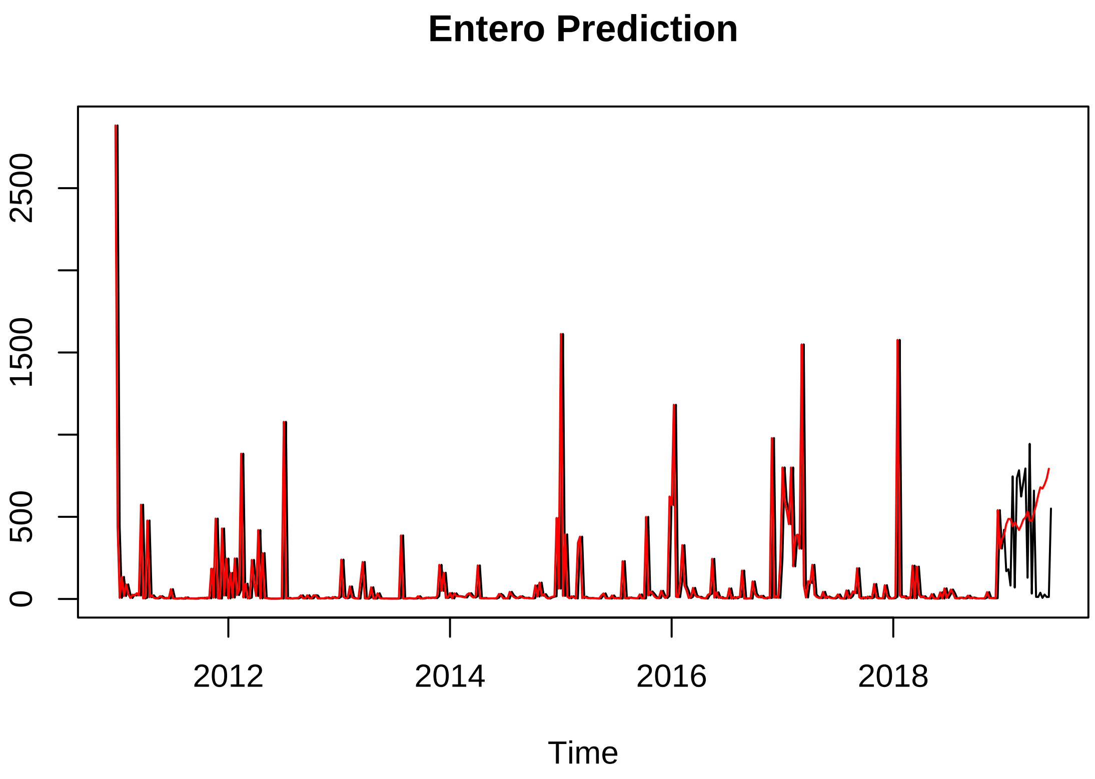
RMSE

```
RMSE(tail(diffinv(do.combined, xi = ts.do[1]),23), diff(window(ts.do, 2019)))
```

```
## [1] 6.650365
```

Modeling Entero

```
# Only include data to 2018, reserve 2019 for validation
entero.train <- window(ts.entero, 2011, c(2018, 52))
entero.results <- ARIMANN(entero.train, 23)
entero.combined <- ts(c(diff(entero.train), entero.results), start=start(entero.train), frequency = fre

ts.plot(window(ts.entero, 2011, c(2019, 23)), diffinv(entero.combined, xi = ts.entero[1]), gpars = list
abline(v=as.Date("2019-01-01"), col = "blue")
```



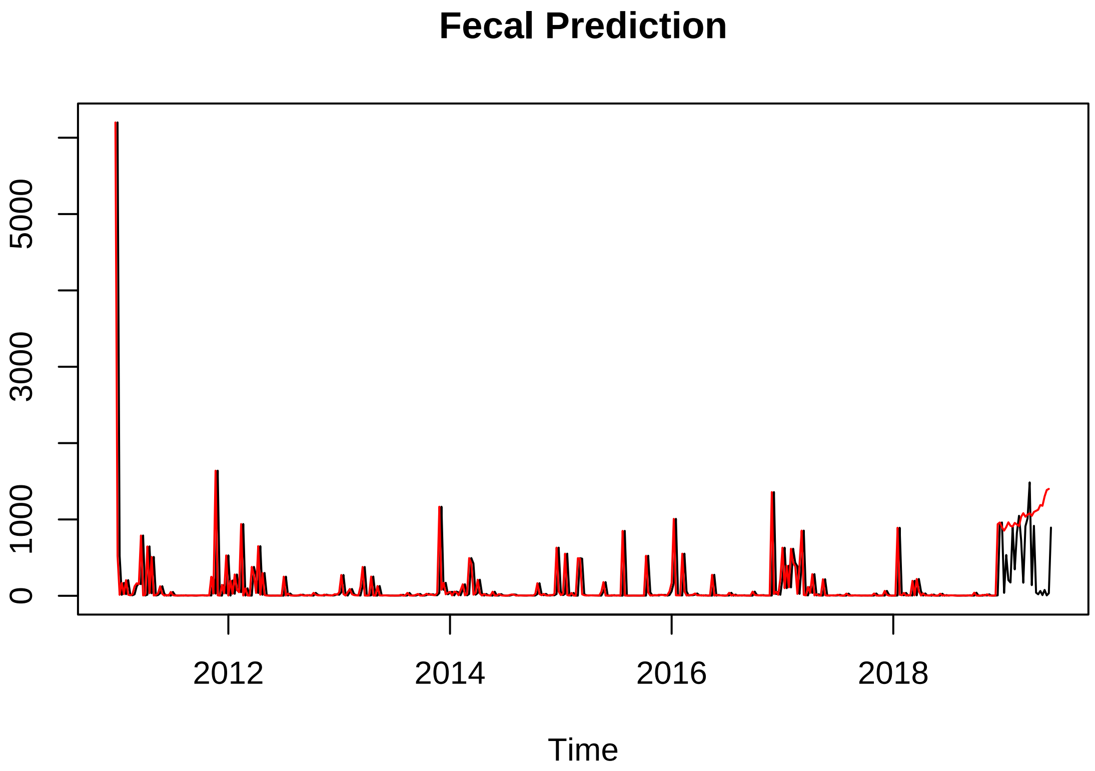
RMSE

```
RMSE(tail(diffinv(entero.combined, xi = ts.entero[1]),23), diff(window(ts.entero, 2019)))
```

```
## [1] 731.7727
```

Modeling Fecal

```
# Only include data to 2018, reserve 2019 for validation
fecal.train <- window(ts.fecal, 2011, c(2018, 52))
fecal.results <- ARIMANN(fecal.train, 23)
fecal.combined <- ts(c(diff(fecal.train), fecal.results), start=start(fecal.train), frequency = frequenc

ts.plot(window(ts.fecal, 2011, c(2019, 23)), diffinv(fecal.combined, xi = ts.fecal[1]), gpars = list(col
abline(v=as.Date("2019-01-01"), col = "blue")
```



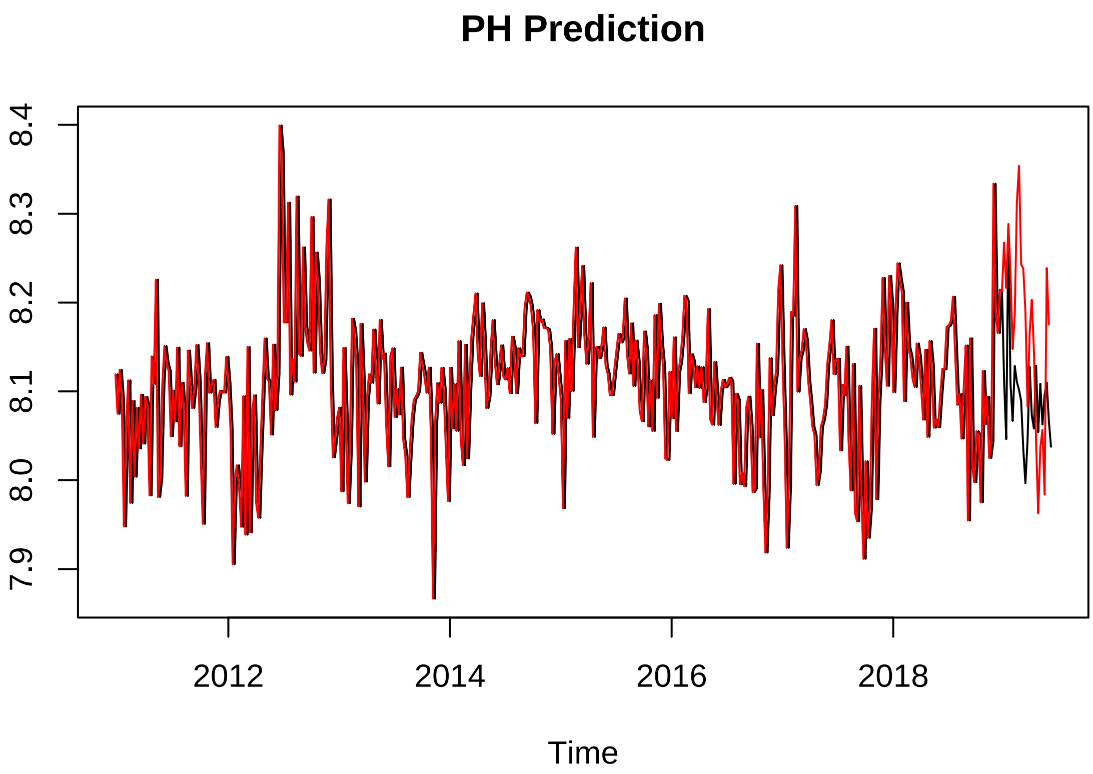
RMSE

```
RMSE(tail(diffinv(fecal.combined, xi = ts.fecal[1]), 23), diff(window(ts.fecal, 2019)))
```

```
## [1] 1216.726
```

Modeling PH

```
# Only include data to 2018, reserve 2019 for validation
ph.train <- window(ts.ph, 2011, c(2018, 52))
ph.results <- ARIMANN(ph.train, 23)
ph.combined <- ts(c(diff(ph.train), ph.results), start=start(ph.train), frequency = frequency(ph.train)

ts.plot(window(ts.ph, 2011, c(2019, 23)), diffinv(ph.combined, xi = ts.ph[1]), gpars = list(col = c("bl
abline(v=as.Date("2019-01-01"), col = "blue")
```



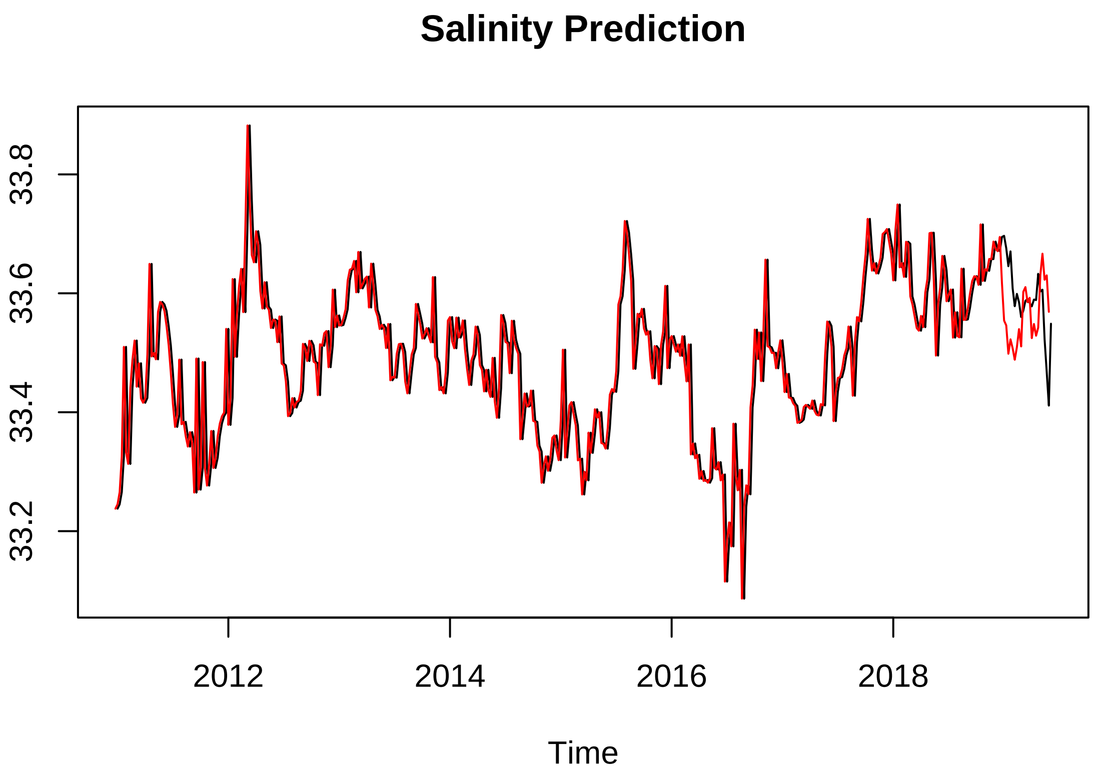
RMSE

```
RMSE(tail(diffinv(ph.combined, xi = ts.ph[1]), 23), diff(window(ts.ph, 2019)))
```

```
## [1] 8.17014
```

Modeling Salinity

```
# Only include data to 2018, reserve 2019 for validation
salinity.train <- window(ts.salinity, 2011, c(2018, 52))
salinity.results <- ARIMANN(salinity.train, 23)
salinity.combined <- ts(c(diff(salinity.train), salinity.results), start=start(salinity.train), frequenc

ts.plot(window(ts.salinity, 2011, c(2019, 23)), diffinv(salinity.combined, xi = ts.salinity[1]), gpars =
abline(v=as.Date("2019-01-01"), col = "blue")
```



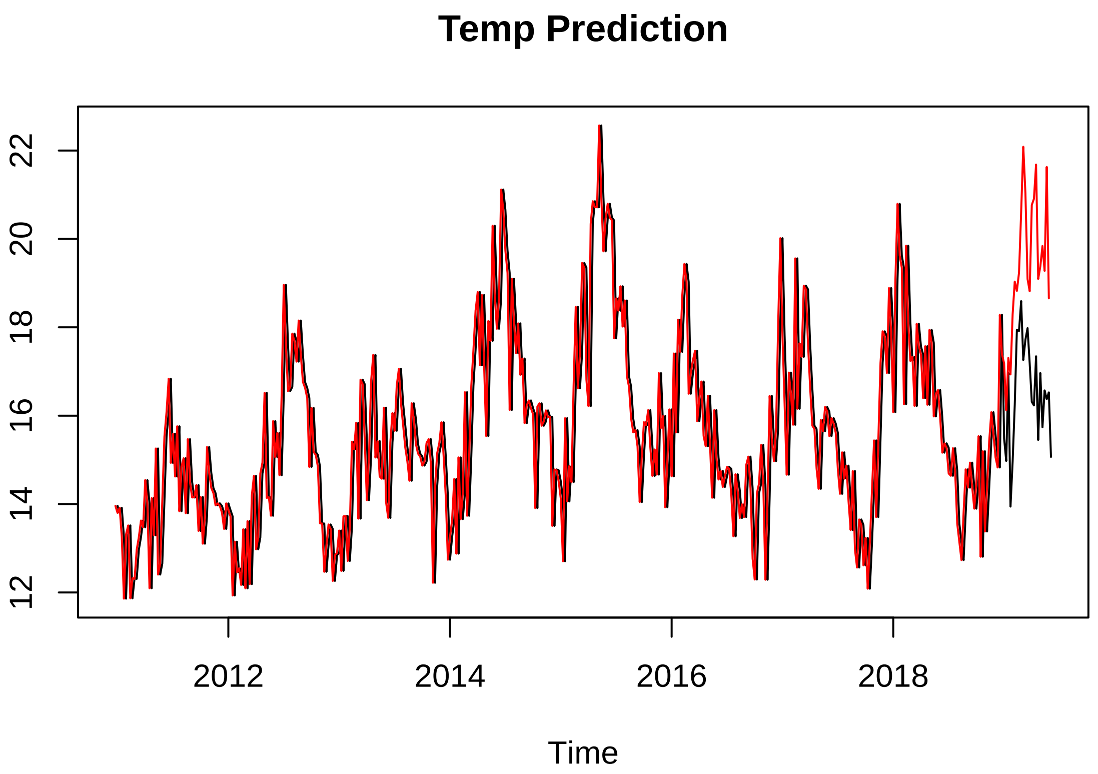
RMSE

```
RMSE(tail(diffinv(salinity.combined, xi = ts.salinity[1]), 23), diff(window(ts.salinity, 2019)))
```

```
## [1] 33.57443
```

Modeling Temp

```
# Only include data to 2018, reserve 2019 for validation
temp.train <- window(ts.temp, 2011, c(2018, 52))
temp.results <- ARIMANN(temp.train, 23)
temp.combined <- ts(c(diff(temp.train), temp.results), start=start(temp.train), frequency = frequency(t

ts.plot(window(ts.temp, 2011, c(2019, 23)), diffinv(temp.combined, xi = ts.temp[1]), gpars = list(col =
abline(v=as.Date("2019-01-01"), col = "blue")
```



RMSE

```
RMSE(tail(diffinv(temp.combined, xi = ts.temp[1]), 23), diff(window(ts.temp, 2019)))
```

```
## [1] 19.5054
```

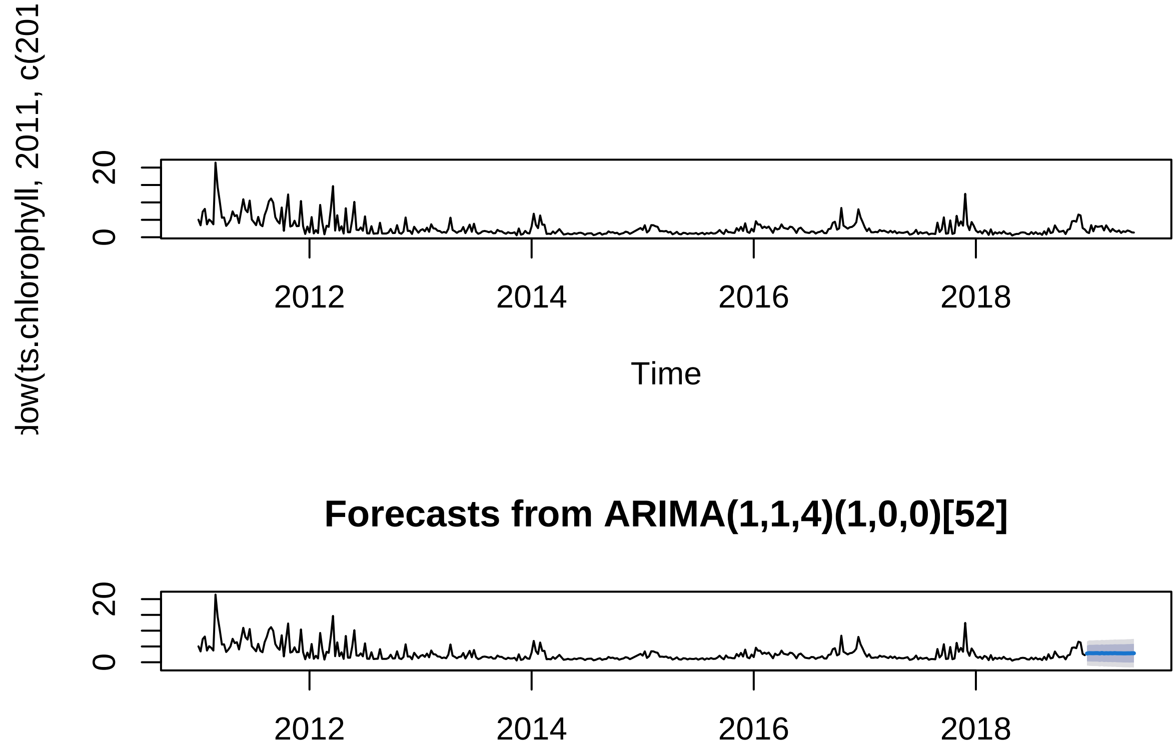
ARIMA Models

Chlorophyll

```
# Only include data to 2018, reserve 2019 for validation
chlor.arima <- auto.arima(window(ts.chlorophyll, 2011, c(2018, 52)))
summary(chlor.arima)
```

```
## Series: window(ts.chlorophyll, 2011, c(2018, 52))
## ARIMA(1,1,4)(1,0,0)[52]
##
## Coefficients:
##           ar1     ma1      ma2      ma3     ma4    sar1
##       -0.9455  0.2784  -0.8364  -0.2268  0.0144  0.0699
## s.e.   0.0485  0.0686   0.0580   0.0515  0.0495  0.0633
##
## sigma^2 estimated as 3.779:  log likelihood=-862.55
## AIC=1739.11   AICc=1739.38   BIC=1767.31
##
## Training set error measures:
##                         ME     RMSE       MAE       MPE      MAPE       MASE
## Training set -0.05421516 1.927541 1.181362 -31.3413 52.54665 0.7195104
##                      ACF1
## Training set -0.002045334
```

```
par(mfrow = c(2,1))
plot(window(ts.chlorophyll, 2011, c(2019, 23)))
plot(forecast(chlor.arima, h = 23))
```



RMSE

```
RMSE(forecast(chlor.arima, h = 23)$mean, window(ts.chlorophyll, 2019))
```

```
## [1] 1.042137
```

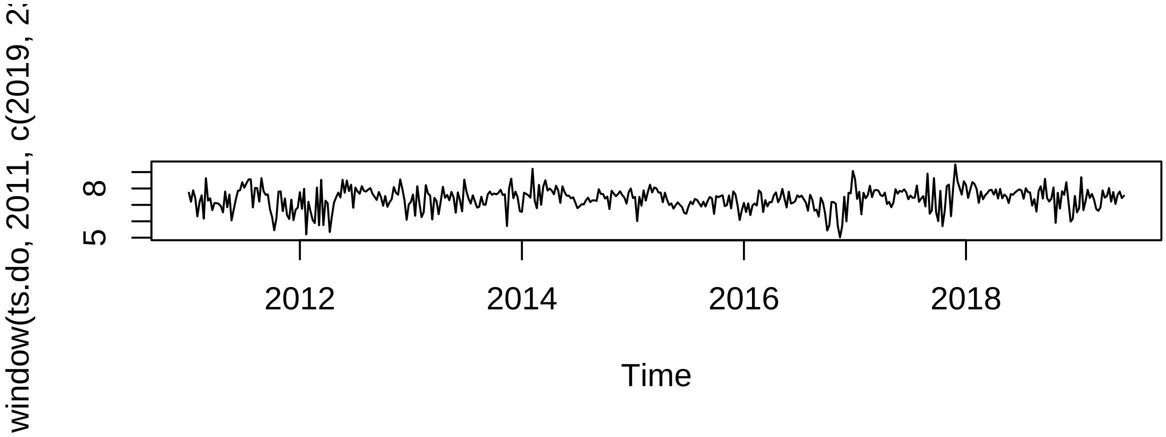
Disolved Oxygen

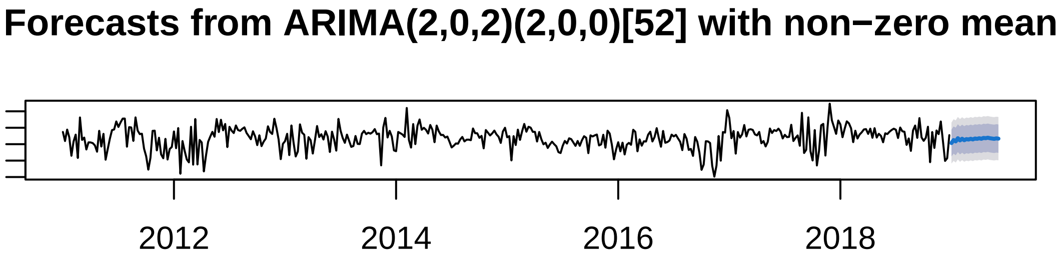
```
# Only include data to 2018, reserve 2019 for validation
do.arima <- auto.arima(window(ts.do, 2011, c(2018, 52)))
summary(do.arima)
```

```
## Series: window(ts.do, 2011, c(2018, 52))
## ARIMA(2,0,2)(2,0,0)[52] with non-zero mean
##
## Coefficients:
##           ar1     ar2     ma1      ma2    sar1     sar2    mean
##       -0.0749  0.7711  0.2401  -0.5834  0.0282  -0.0496  7.3649
## s.e.   0.1034  0.0927  0.1216   0.1052  0.0531   0.0613  0.0656
##
## sigma^2 estimated as 0.4059:  log likelihood=-399.47
## AIC=814.95   AICc=815.3   BIC=847.2
```

```
##
## Training set error measures:
##                         ME      RMSE       MAE        MPE     MAPE     MASE
## Training set -0.0007095884 0.6317496 0.4761292 -0.8049509 6.705114 0.689416
##                     ACF1
## Training set 0.001939922
```

```
par(mfrow = c(2,1))
plot(window(ts.do, 2011, c(2019, 23)))
plot(forecast(do.arima, h = 23))
```





##

RMSE

```
RMSE(forecast(do.arima, h = 23)$mean, window(ts.do, 2019))
```

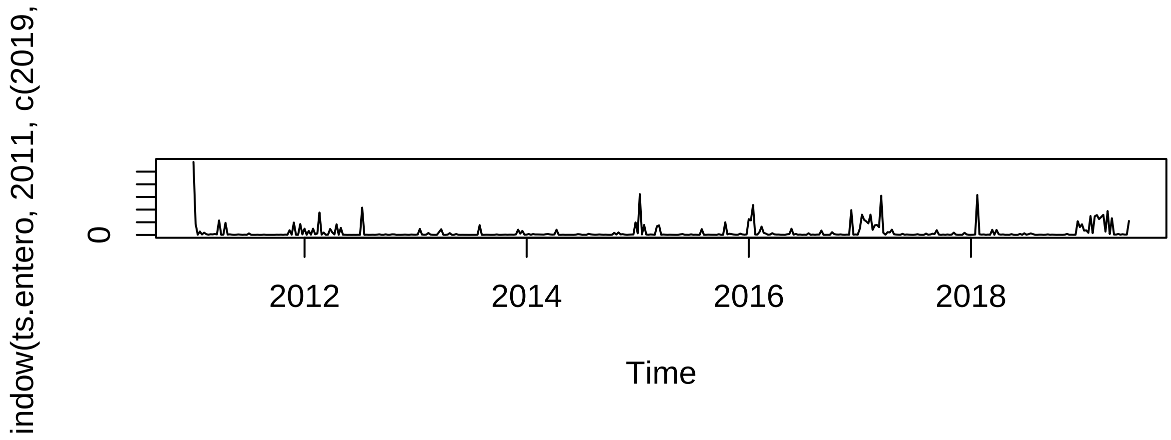
```
## [1] 0.5184086
```

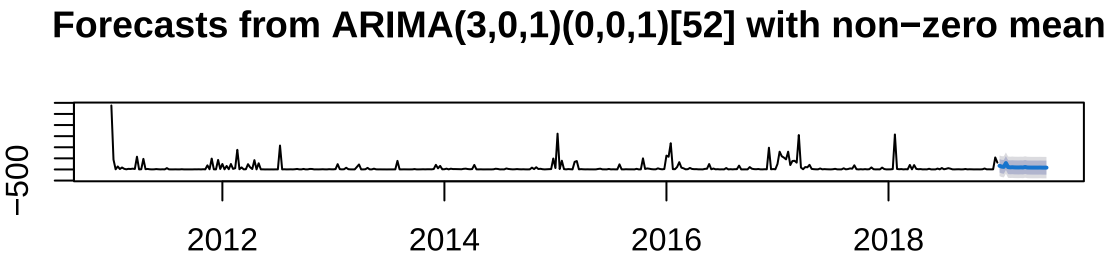
Entero

```
# Only include data to 2018, reserve 2019 for validation
entero.arima <- auto.arima(window(ts.entero, 2011, c(2018, 52)))
summary(entero.arima)
```

```
## Series: window(ts.entero, 2011, c(2018, 52))
## ARIMA(3,0,1)(0,0,1)[52] with non-zero mean
##
## Coefficients:
##          ar1     ar2      ar3      ma1    sma1     mean
##       0.8963  0.0443  -0.0662  -0.7547  0.1155  89.1923
## s.e.  0.1776  0.0819   0.0755   0.1656  0.0612  25.0402
##
## sigma^2 estimated as 56298:  log likelihood=-2862.87
## AIC=5739.73   AICc=5740.01   BIC=5767.95
##
## Training set error measures:
##                     ME     RMSE      MAE      MPE     MAPE      MASE
## Training set -5.198707 235.5545 107.4697 -1016.96 1035.669 0.9052769
##                     ACF1
## Training set -0.04759006
```

```
par(mfrow = c(2,1))
plot(window(ts.entero, 2011, c(2019, 23)))
plot(forecast(entero.arima, h = 23))
```





##

RMSE

```
RMSE(forecast(do.arima, h = 23)$mean, window(ts.entero, 2019))
```

```
## [1] 467.3004
```

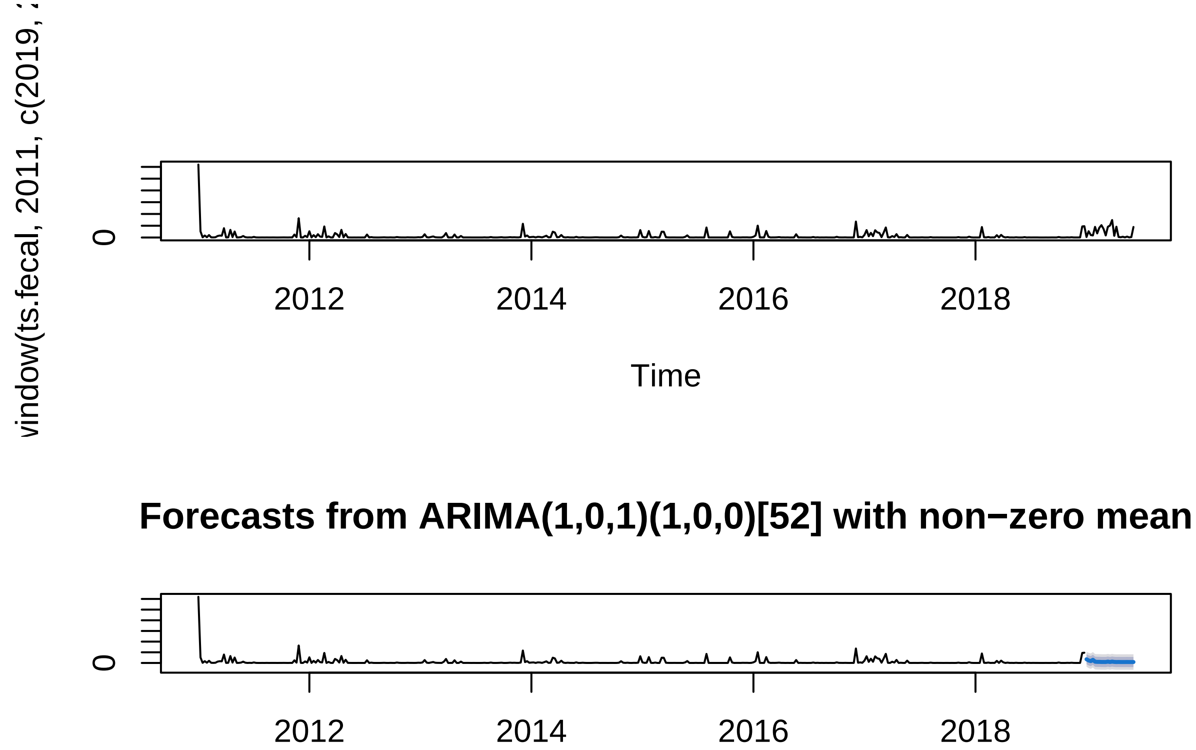
Fecal

```
# Only include data to 2018, reserve 2019 for validation
fecal.arima <- auto.arima(window(ts.fecal, 2011, c(2018, 52)))
summary(fecal.arima)
```

```
## Series: window(ts.fecal, 2011, c(2018, 52))
## ARIMA(1,0,1)(1,0,0)[52] with non-zero mean
##
## Coefficients:
##          ar1      ma1    sar1      mean
##       0.5660  -0.3393  0.1838  106.5577
## s.e.  1.0108   1.1306  0.0905   37.7856
##
```

```
## sigma^2 estimated as 127140:  log likelihood=-3033.84
## AIC=6077.67   AICc=6077.82   BIC=6097.82
##
## Training set error measures:
##                     ME     RMSE      MAE     MPE     MAPE    MASE        ACF1
## Training set -9.167291 354.8481 135.2333 -1358.5 1380.195 1.05456 -0.1425296
```

```
par(mfrow = c(2,1))
plot(window(ts.fecal, 2011, c(2019, 23)))
plot(forecast(fecal.arima, h = 23))
```



RMSE

```
RMSE(forecast(fecal.arima, h = 23)$mean, window(ts.fecal, 2019))
```

```
## [1] 567.3506
```

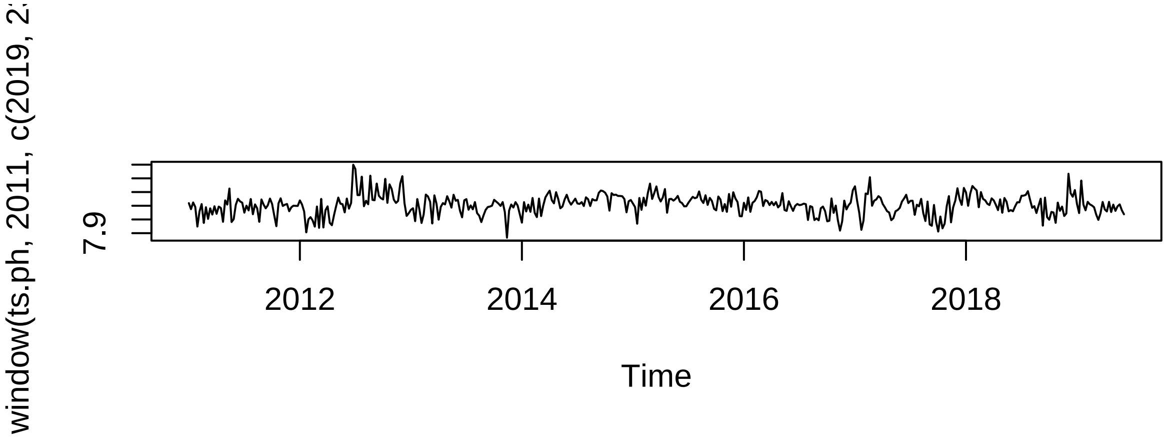
PH

```
# Only include data to 2018, reserve 2019 for validation
ph.arima <- auto.arima(window(ts.ph, 2011, c(2018, 52)))
summary(ph.arima)
```

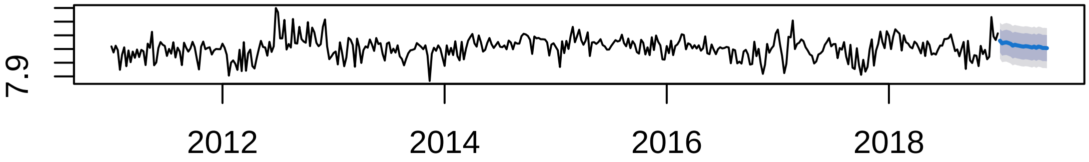
```
## Series: window(ts.ph, 2011, c(2018, 52))
## ARIMA(1,0,2)(0,0,1)[52] with non-zero mean
##
## Coefficients:
##          ar1      ma1      ma2    sma1    mean
##       0.8849  -0.5778  -0.0905  0.0907  8.1083
## s.e.  0.0462   0.0686   0.0560  0.0507  0.0100
```

```
##
## sigma^2 estimated as 0.004497:  log likelihood=535.97
## AIC=-1059.94   AICc=-1059.73   BIC=-1035.75
##
## Training set error measures:
##                        ME      RMSE        MAE          MPE      MAPE      MASE
## Training set 0.0001861881 0.0666553 0.05046257 -0.004469943 0.6227442 0.6627212
##                     ACF1
## Training set -0.00308448
```

```
par(mfrow = c(2,1))
plot(window(ts.ph, 2011, c(2019, 23)))
plot(forecast(ph.arima, h = 23))
```



Forecasts from ARIMA(1,0,2)(0,0,1)[52] with non−zero mean



RMSE

```
RMSE(forecast(ph.arima, h = 23)$mean, window(ts.ph, 2019))
```

```
## [1] 0.06198158
```

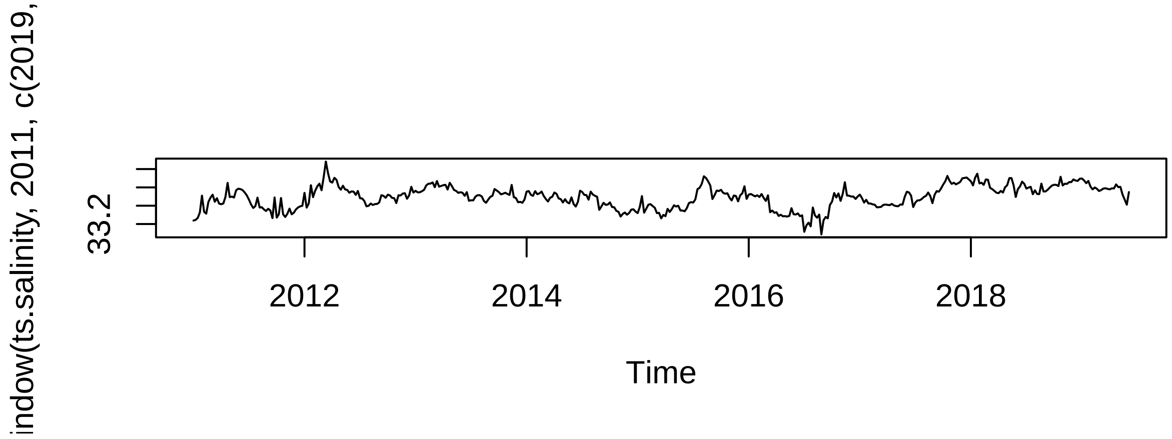
Salinity

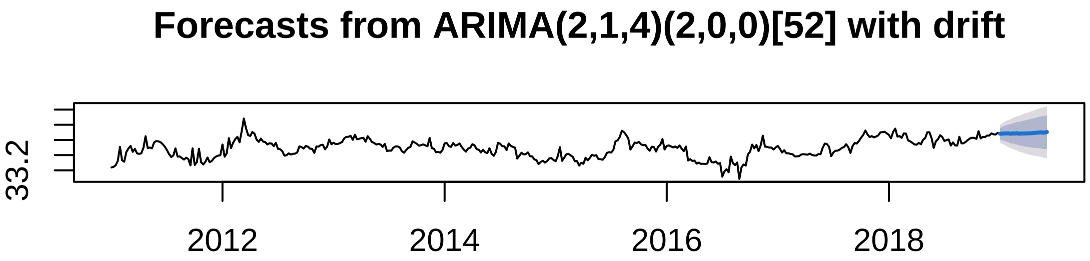
```
# Only include data to 2018, reserve 2019 for validation
salinity.arima <- auto.arima(window(ts.salinity, 2011, c(2018, 52)))
summary(salinity.arima)
```

```
## Series: window(ts.salinity, 2011, c(2018, 52))
## ARIMA(2,1,4)(2,0,0)[52] with drift
##
## Coefficients:
```

```
##           ar1      ar2     ma1     ma2      ma3     ma4    sar1    sar2   drift
##       -1.6533  -0.7358  1.3445  0.0797  -0.3442  0.0010  0.0424  0.0523  0.0011
## s.e.   0.2006   0.1591  0.2052  0.1383   0.1195  0.0612  0.0560  0.0592  0.0018
##
## sigma^2 estimated as 0.003259:  log likelihood=603.73
## AIC=-1187.46   AICc=-1186.92   BIC=-1147.18
##
## Training set error measures:
##                        ME       RMSE        MAE          MPE      MAPE
## Training set 6.855504e-05 0.05639824 0.04131292 1.535374e-05 0.1233572
##                   MASE          ACF1
## Training set 0.3319615 -0.0004661822
```

```
par(mfrow = c(2,1))
plot(window(ts.salinity, 2011, c(2019, 23)))
plot(forecast(salinity.arima, h = 23))
```





RMSE

```
RMSE(forecast(ph.arima, h = 23)$mean, window(ts.ph, 2019))
```

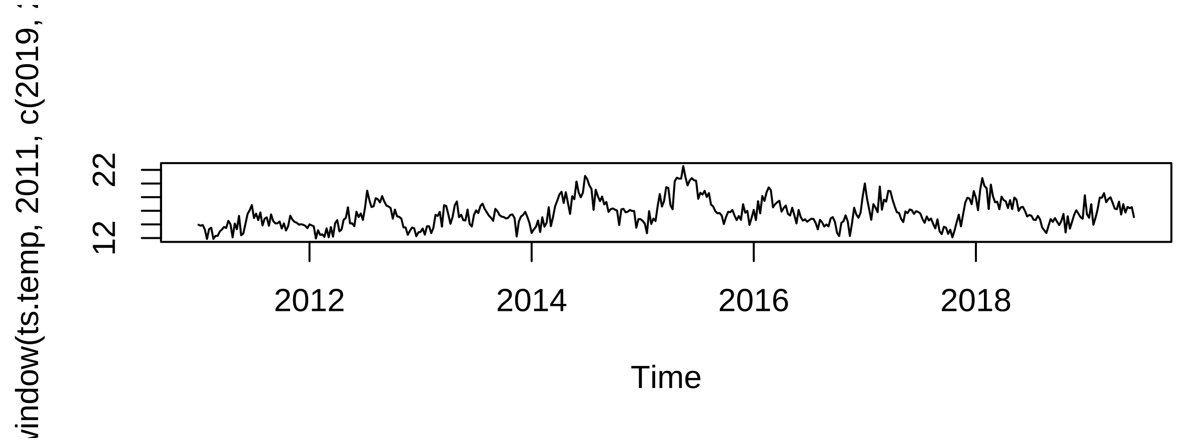
```
## [1] 0.06198158
```

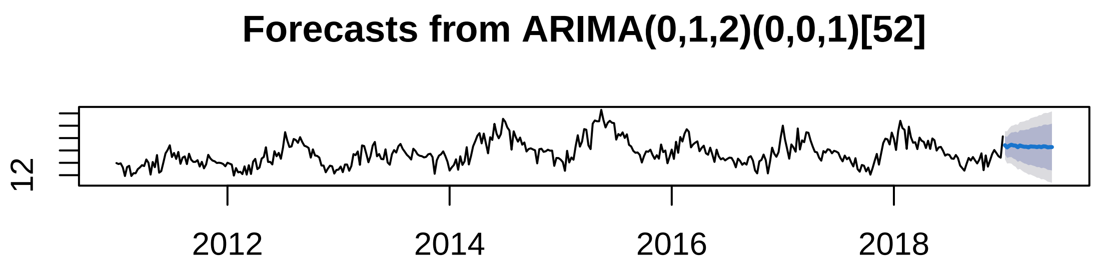
Temp

```
# Only include data to 2018, reserve 2019 for validation
temp.arima <- auto.arima(window(ts.temp, 2011, c(2018, 52)))
summary(temp.arima)
```

```
## Series: window(ts.temp, 2011, c(2018, 52))
## ARIMA(0,1,2)(0,0,1)[52]
##
## Coefficients:
##           ma1      ma2    sma1
##       -0.4368  -0.0679  0.0863
## s.e.   0.0493   0.0454  0.0490
##
## sigma^2 estimated as 1.34:  log likelihood=-648.38
## AIC=1304.77   AICc=1304.86   BIC=1320.88
##
## Training set error measures:
##                      ME     RMSE       MAE        MPE    MAPE      MASE
## Training set 0.01166253 1.151942 0.8978906 -0.3194333 5.76851 0.5398124
##                      ACF1
## Training set -0.001339407
```

```
par(mfrow = c(2,1))
plot(window(ts.temp, 2011, c(2019, 23)))
plot(forecast(temp.arima, h = 23))
```





RMSE

```
RMSE(forecast(temp.arima, h = 23)$mean, window(ts.temp, 2019))
```

```
## [1] 1.205087
```