

Forecasting Ocean Quality within the San Diego Coastal Region

Aaron Carr, Brianne Bell, Connie Chow

<https://github.com/USD-502-FinalProject/506-OceanWater-Team1>

University of San Diego

Master of Science, Applied Data Science

ADS-506 Fall 2022

December 5, 2022

Table of Contents

Table of Contents	2
List of Figures	4
List of Tables	4
Introduction	5
Problem Statement	5
Description of current state	5
The business objective	6
The success criteria	6
The issues	7
The threats	8
Literature Review	8
Source Synthesis	8
Methods	9
Data access and preparation	9
Exploratory Data Analysis (EDA)	9
Preprocessing	15
Modeling	17
Exponential smoothing: Holt-Winter's	17
Linear Regression: Time Series with Trend and Seasonality	18
Advanced regression: ARIMA	19
ARIMA Models at PLOO Project.	20
ARIMA Models at SBOO Project.	21
Results	23
PLOO Project	24
SBOO Project	25
Discussion	25
Conclusion	26
Future Directions	27
References	29
Appendix A	32
Appendix B	34
Source 1: Shajratul (2020)	34
Background Information	34
Overview of Main Points	34

FORECASTING OCEAN QUALITY	3
Strengths.	34
Weaknesses.	35
Key Findings	35
Source 2: Shamshirband et al. (2018)	36
Background Information	36
Overview of Main Points	36
Gaps in knowledge.	37
Strengths.	37
Weaknesses.	37
Key Findings	37
Source 3: Peng et al. (2019)	37
Background Information	38
Overview of Main Points	38
Key Findings	39
Source 4: City of San Diego. (n.d.-a)	39
Background Information	39
Overview of Main Points	40
Strengths.	40
Weaknesses.	40
Key Findings	40
Appendix C	42
Appendix D	47

List of Figures

Figure 1. Boxplots of ENTERO By Project	11
Figure 2. Boxplots of PH By Project By Depth	12
Figure 3. Time Series Decomposition Using stl() Function	13
Figure 4. ENTERO Decomposition (2000-2022) Shows Clear Seasonality Element for PLOO Location	14
Figure 5. ENTERO Decomposition (2000-2022) Shows Clear Seasonality and Trend for SBOO Location	15
Figure 6. Holt-Winter's Forecast for Multiple Parameters	18
Figure 7. Linear Regression Forecasts for Two Parameters (ENTERO and PH) at Two Sites (PLOO and SBOO)	19
Figure 8. ARIMA Forecast for Different Parameters at PLOO	21
Figure 9. ARIMA Forecast for Different Parameters at SBOO	22
Appendix C, Figure 10. Histograms of bacteriological features	42
Appendix C, Figure 11. Histograms of chemical features	43
Appendix C, Figure 12. Non time series correlation plot	44
Appendix C, Figure 13. Bar Graph of 'depth_m_bin' Feature	44
Appendix C, Figure 14. Boxplots of ENTERO By Project By Depth	45
Appendix C, Figure 15. Oceanographic mooring and monitoring station locations around the PLOO and SBOO	46

List of Tables

Table 1. Feature Total NA Count	10
Table 2. PLOO Model Evaluation via MAPE and MAE	23
Table 3. SBOO Model Evaluation via MAPE and MAE	24
Table 4. Feature NA Count for ENTERO Category	42

Introduction

In this project, a secondary study will be performed on time series data provided by the City of San Diego's Ocean Water Monitoring Program (City of San Diego, 2022b). This is data concerning the water quality collected in the surrounding region which extends 340 square miles from Northern San Diego to Baja Mexico up to depths of over 100m (City of San Diego, n.d.-b). The data contains physiological, biological, and chemical parameters of water tested throughout this region.

Studying and analyzing this data will provide an important representation of sea water composition, which naturally has a significant impact on the ocean environment, as well as public health. Due to increasing awareness of global warming and its impact on the natural environment, including human habitation and livelihood, there is great interest in studying and predicting sea water composition in hopes to improve the state of our oceans.

As a short introduction to the domain of sea water, the variables included with the data set are described in further detail in Appendix A.

Problem Statement

Description of current state

As part of the formal Ocean Monitoring Program (OMP), the City of San Diego performs regular testing of sea water characteristics to measure "physical, chemical, and biological parameters" such as salinity, temperature, certain bacterial levels, etc. (City of San Diego, n.d.-b; City of San Diego, 2022b). Thousands of data samples are collected every year at 157 different designated stations within two project regions, though the number of samples for each location are not consistent, with the minimum number of samples being 30 (at stations A15 and A16) and the maximum number being 55,217 (at A1) between November 1990 and December 2021. The current main focus of OMP's data collection is on time series analyses, reporting, and research. Descriptive summaries are used to determine where and how pollutants are entering marine ecosystems.

The business objective

From a business perspective, the aim is to: 1) Identify key ocean water quality parameter measurements, as well as their time series components (such as trend and seasonal changes), to better understand what could be causing peaks or dips in the measurements; and 2) Develop a system to accurately forecast levels of enterococci (entero) bacteria, which at levels above 104 per 100 milliliter (mL) cause serious public health risk (New Hampshire Department of Environmental Services, 2019). The refined time series analysis and forecasting would then be provided to public officials in both water safety and city environmental protection in order to inform decision making concerning waste water monitoring and regulation. For instance, if the information is used to determine that the issue is something that can be controlled through waste water treatment or implementing community beach cleaning days, then these appropriate mitigation procedures can be taken. Such data-driven information could also lead to increased public awareness campaigns, if for instance trends of peaking entero or fecal matter levels are measured around Summer holidays, the City could work with community-based businesses to provide swim diapers to families to help lessen the contamination. Lastly, and most crucially for long-term environmental impact purposes, improving the understanding of the ocean water quality changes over time can help increase public awareness of what is happening in the ocean, which is so integral to the sustainability of the community.

The success criteria

The success of the project depend on two criteria:

1. Building models that forecast accurately.
2. Translating modeling results into actionable insights.

Several time series models will be built to predict the sea water characteristics with as high of accuracy as possible for specific times of the year. Mean absolute percentage error (MAPE) will be used to evaluate the performance metrics, as it is scale-independent and thus can be used to compare between different models. Additionally, mean absolute error (MAE) will

be examined and compared across models to determine which performs the best for each time series.

Study authors plan to propose a few solutions based on the predictions of the seawater components, which are actionable and contribute to the understanding and addressing of global warming. This study will be considered successful if one or more models can be developed that achieve a MAPE less than 20% and an MAE less than the baseline model.

The issues

Several issues must be considered and addressed as part of implementing effective analysis and forecasting methods:

- Need data to be updated in a timely manner across all those stations determined as priority focus points.
- The data must be aggregated in such a way as to facilitate forecasting at the proper level.
- Based on limited public funding for cost-intensive solutions, the developed system needs to have minimal expenses for maintenance time and staff.
- Post-deployment model evaluation measures must be available to continually review forecasts for shifts in trends, seasonality, autocorrelation, and general drift.
- A robust model or set of models must be identified that minimize forecasting error without overfitting to training data.
- Given several available and potential features of interest, the team needs to determine whether simple extrapolation methods are sufficient or econometric models are needed to account for cross-correlation of features that individually make up a single time series.
- Need to determine whether there are inherent issues with the data, such as outliers and noise, and the best way to mitigate their effects.

The threats

The only threats that are present are the natural consequences of not taking care of the ocean, leaving the temperature to rise, the pH to fall, and the presence of algal blooms (toxic and otherwise) to increase in appearance. If it is discovered through coordinated efforts with other businesses and manufacturing centers that certain practices are causing ill effects to the ocean environment, those informational insights can be used by officials to develop consequences for the offenders and better solutions than what is presently enforced.

Literature Review

Source Synthesis

Sources: Shajratul (2020); Shamshirband et al. (2018); Peng et al. (2019); and City of San Diego (n.d.-a).

See Appendix B for details on each individual source, including background information, overview of main points (e.g., gaps in understanding, strengths, weaknesses), and key findings.

Though each of the studies reviewed did add important results to the overall literature, there were some general gaps that none of them completely filled. For instance, Shajratul (2020) and Shamshirband et al. (2018) focused on marine biology, which is extremely important, but both papers were less focused on short-term public health. On the other hand, Peng et al. (2019) did approach their study from more of a human habitat standpoint, but the study was based on chemical adulterants, as opposed to having a clear and defined focus on bacteria levels.

Across the literature, there was generally more focus on time series analysis, which is an important aspect of research, but does not always directly correspond with accurate prediction methods (i.e., fit does not equal high forecast performance); one example is City of San Diego (n.d.-a), which focused more on analyses of what had occurred as part of their mandatory regulation and environmental impact studies. Some forecasting was utilized, but for Shajratul

(2020), it was only to impute missing values for time series analysis, or as with the Peng et al. study, it lacked long-term forecasting— $t + 1$ -6 days was the maximum.

There was also a dearth of direct comparison of performance results from different model types within a single study. For instance, Shamshirband et al. (2018) focused only on artificial neural networks (ANN) and Peng et al. (2019) used dynamic forecast methods implemented through a package called EcoLake, without any more traditional forecasting methods to use as baseline comparisons against their more sophisticated forecasting methods.

As a result of the literature review, the current study authors will focus on creating time series analysis for features related to predicting bacterial levels in ocean water. Additionally, forecasting will be done for a full year. Lastly, a comparison between several modeling techniques will provide comparisons between algorithms with varying levels of complexity to see which predicts better with this specific data set.

Methods

Data access and preparation

The data was downloaded from City of San Diego (2022b) on November 17, 2022, as four separate data sets, each for a different time segmentation. The files were imported as data frames into R and all data manipulation and analyses were performed using RStudio. The four frames were merged and then sorted by the date_sampled feature, as required for time series analyses and forecasting methods. The overall distribution of the different parameters was explored and displayed as histograms for the different project locations (PLOO and SBOO for northern and southern collection regions, respectively). Additionally, a non-time series correlation plot was developed to explore any issues of collinearity. These plots can be seen in Appendix C, Figures 10-12.

Exploratory Data Analysis (EDA)

First, summary tables were created to review the count of missing (NA) values for each of the 10 data set features. As can be seen in Table 1, which displays the NA counts for the

entire dataset, the features with the largest number of missing values are depth_m ($n = 84,161$), time ($n = 160,840$), and qualifier ($n = 841,902$). For an example of a summary broken out by parameter feature category, see Table 4 in Appendix C.

Table 1*Feature Total NA Count*

	date_										
	sample	station	depth_m	sample	time	project	parameter	qualifier	value	units	
Not NA (n)	1,236,769	1,236,769	1,152,608	1,236,769	1,075,929	1,236,769	1,236,769	394,867	1,231,466	1,236,769	
NA (n)	0	0	84,161	0	160,840	0	0	841,902	5,303	0	
Not NA (%)	100.0%	100.0%	93.2%	100.0%	87.0%	100.0%	100.0%	31.9%	99.6%	100.0%	
NA (%)	0.0%	0.0%	6.8%	0.0%	13.0%	0.0%	0.0%	68.1%	0.4%	0.0%	

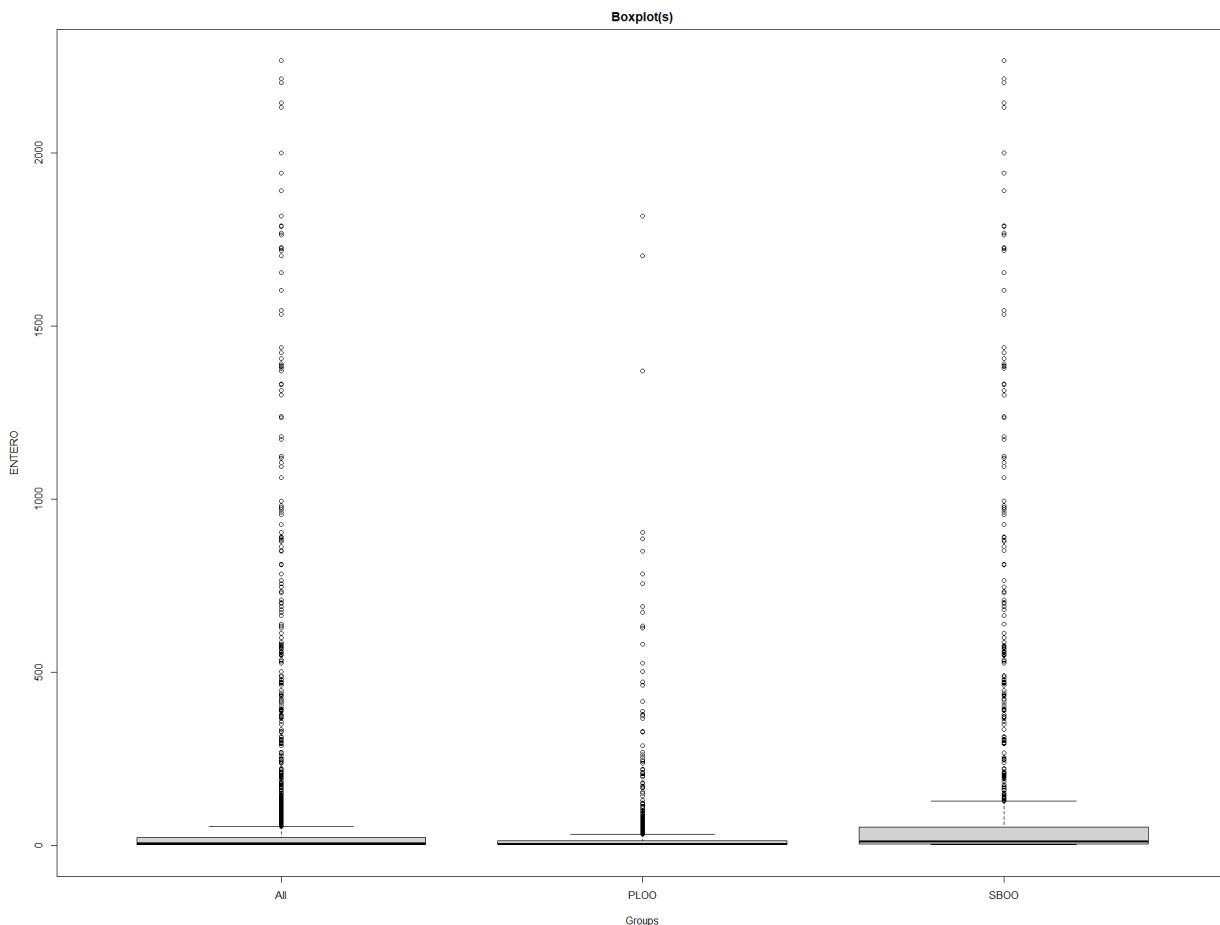
A quick review of the unique counts based on parameter, qualifier, and units was done to check whether qualifier was relevant to the current study; it was concluded that the categories (qualifier) were probably more relevant from a biological standpoint and omitting them would not cause issues for current project time series analysis and forecasting. This review also permitted confirmation that the unit was the same for every instance of each parameter. Lastly, the date_sample and time values were compared and it was determined that there were multiple times per day that sample collection occurred, but it was very inconsistent and uneven which may cause issues during the modeling phase. As a consequence of the above, time, qualifier, and units were all dropped from further analyses. Additionally, the sample column values were not unique to each instance, but they none-the-less represented identifiers that would most likely not add value to forecasting, and as such were removed.

Aggregation summaries based on count or mean were performed for combinations of the remaining six features (station, depth_m, date_sample, project, parameter, and value). Though depth_m was numerical, it only had 51 unique values, indicating that it should be treated more as discrete categorical. As a result, and to reduce dimensionality, it was segmented into eight different bins (see Appendix C, Figure 13), with the bin ranges based on segments of the original data that had approximate normal distributions.

Station, project, and parameter were strictly categorical variables, with 157, 2, and 12 unique categorical values respectively. Since the value feature corresponds to measurements of parameters, several boxplots were created to review their numerical characteristics based on multiple aggregation levels. Figure 1 indicates that the medians for the PLOO and SBOO projects are similar, but it is hard to tell exactly because the outliers have a very significant range. However, SBOO clearly has a larger interquartile range (IQR), indicating greater variability in values, and the maximum outlier values are much higher.

Figure 1

Boxplots of ENTERO By Project



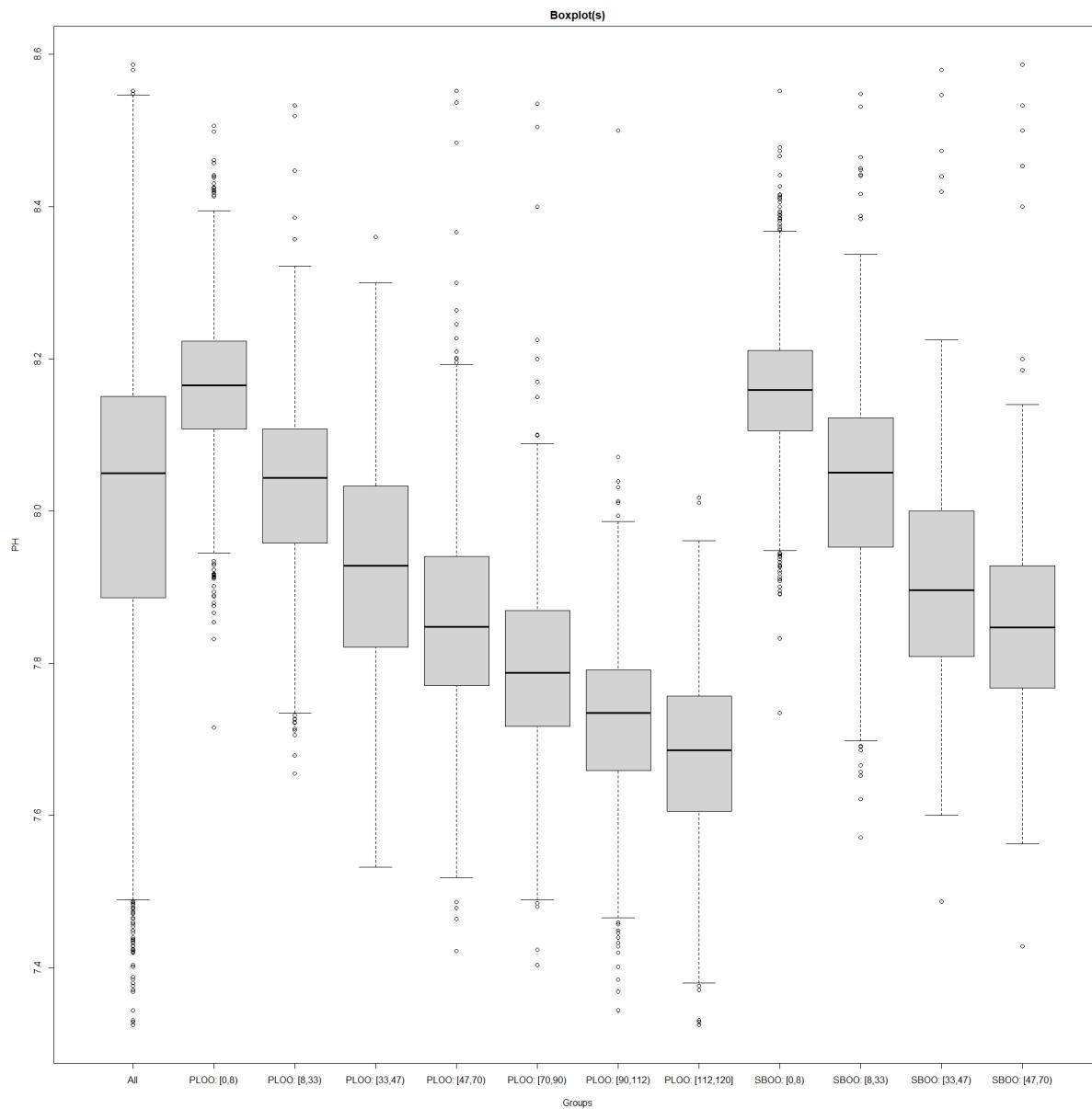
When depth_m is added in (as seen in Appendix C, Figure 14), it can be seen that most of the variability and outliers for SBOO is coming from the “Unknown” category, which was the

missingness bin that was created for NA values. The highest outliers for PLOO are also coming from the same depth_m bin.

To further explore the effect of depth, a boxplot was created for PH (Figure 2), which shows decreasing medians and IQR as depth increases, at both the PLOO and SBOO project regions

Figure 2

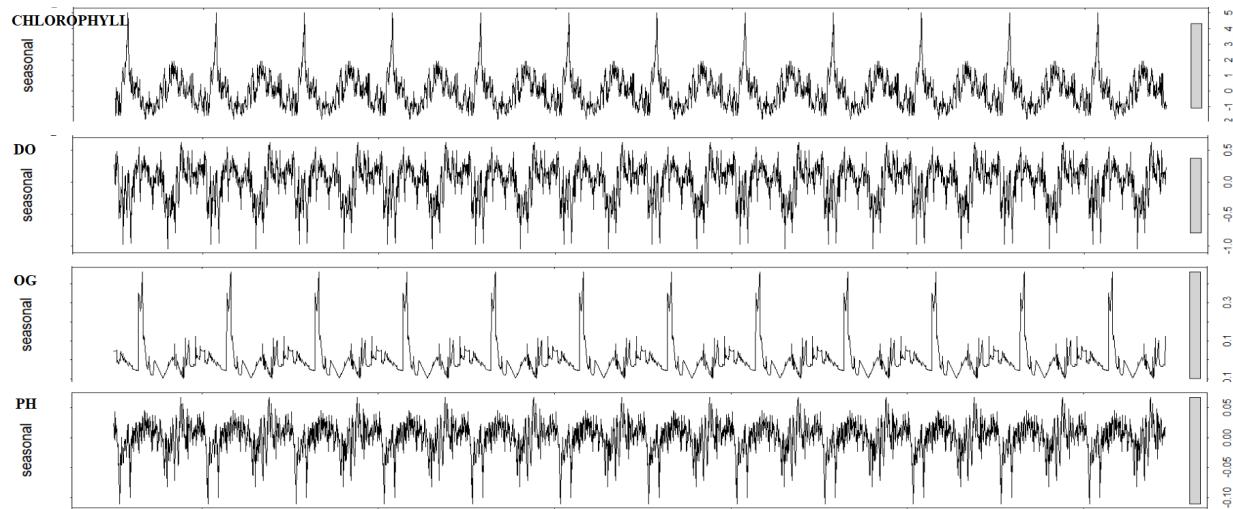
Boxplots of PH By Project By Depth



Shifting to time series specific EDA, in Figure 3 many years of data are required in order to decompose and uncover trends and seasonality patterns within the time series data. A call to R's `stl()` function revealed clear seasonal patterns for CHLOROPHYLL, DO, OG, SALINITY, SUSO, TEMP, and XMS. The FECAL and ENTERO features followed what looks to be cyclical patterns but more within the month-/year-long range.

Figure 3

Time Series Decomposition Using `stl()` Function Reveals Seasonality for Selected Parameters



As seen in Figures 4, ENTERO had a very cyclical trend pattern for the PLOO project, with definite peaks and valleys. Whereas, it was much less consistent for SBOO (see Figure 5), with minor bumps until around 2018 when it began a much more exaggerated increase and subsequent decrease.

Figure 4

ENTERO Decomposition (2000-2022) Shows Clear Seasonality Element for PLOO Location

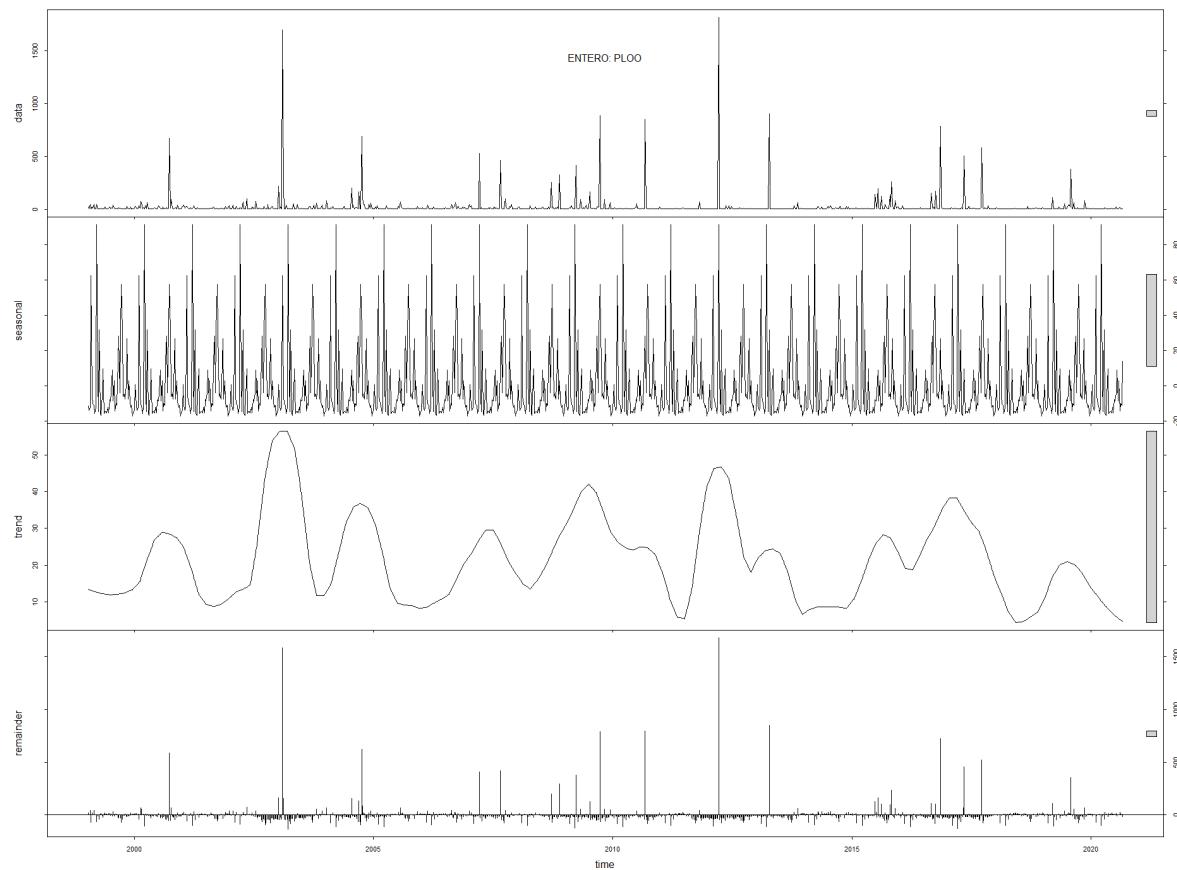
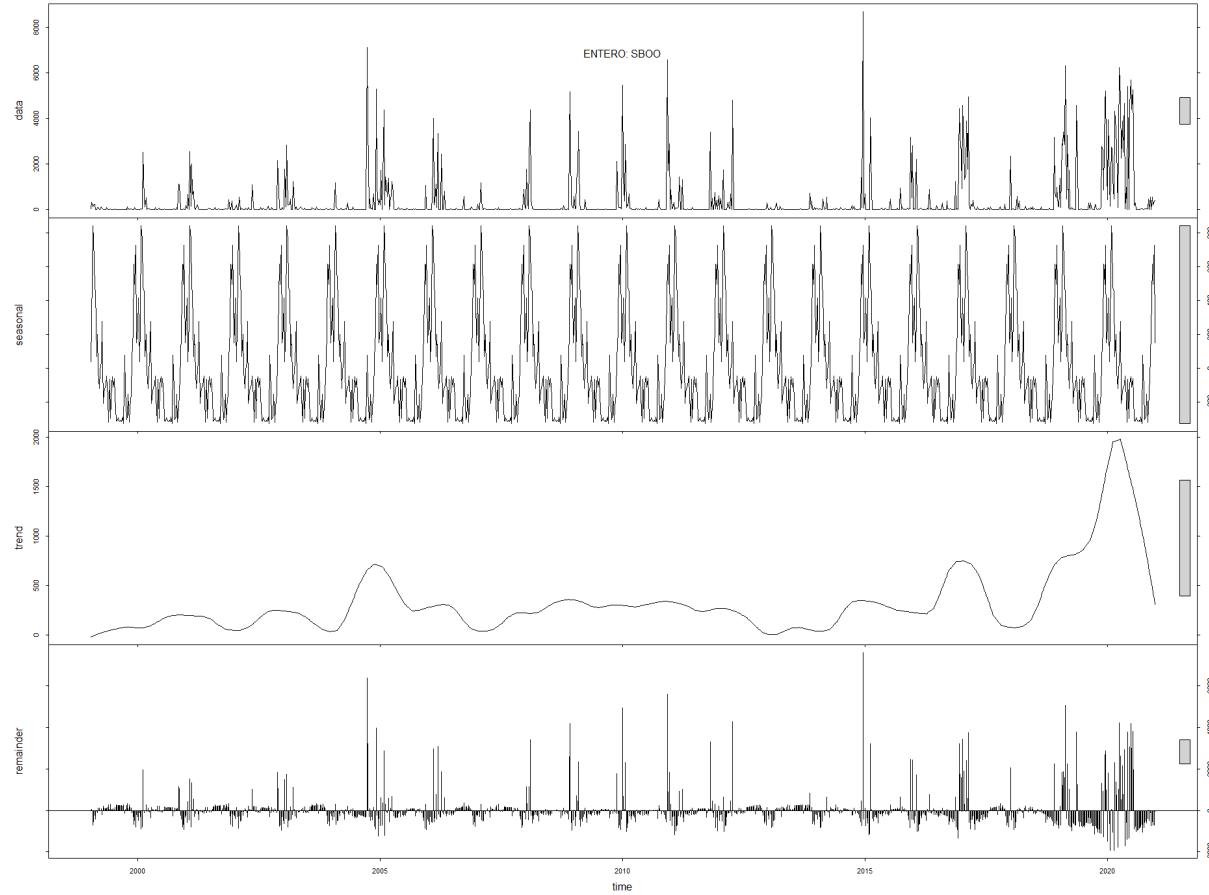


Figure 5

ENTERO Decomposition (2000-2022) Shows Clear Seasonality and Trend for SBOO Location

**Preprocessing**

Based on discoveries made during EDA, final aggregation levels were determined as the basis for data preprocessing. First, each of the 12 parameter categories were converted to columns. Second, due to time and resource limitations, station and depth_m were removed as factors, since each one of them had too many values to consider as candidates for reviewing individual time series based on their breakdown. Although it may prove valuable for a future study to examine per station time series characteristics—and clearly depth has an impact on measurements, as seen in Figure 2—the project variable (with values PLOO and SBOO, as

demarcated geographically in Appendix C, Figure 15) was deemed a sufficient proxy to represent differences in parameter values based on distinct spatial characteristics.

Since a component of modeling was to examine the potential cross-correlation of external variables to predict ENTERO levels, and there were unequally spaced periods within each parameter (but to varying degrees) based on sample collection date gaps, the frequency of periods was decreased from daily to weekly. The aggregation method was based on mean, with NAs removed from the calculation.

Outliers were not removed for the current study based on the assumption that they represented legitimate values, and not noise, as well the fact that due to the complexity of this data set, making comparison models would have exponentially increased the number of separate time series to investigate. This will be discussed further in the Future Directions section.

Of the 12 parameter values, TOTAL was removed due to lack of information on what it even measured; no description could be found on the City of San Diego website. Additionally, OG and SUSO did not have enough samples relative to ENTERO to prove useful as predictor variables, therefore they were also removed. That left nine parameters to potentially use during modeling: ENTERO, CHLOROPHYLL, DENSITY, DO, FECAL, PH, SALINITY, TEMP, and XMS.

Prior to modeling, all R data frames were converted to time series tibbles so that they could be further processed using the `timetk` package. First, the pad_by_time() function was used to add any weeks that were still missing after previous aggregation steps.

Second, the data was partitioned into training and test sets using filter_by_time(). Training windows for all sub datasets ran from 1/18/1999 to 1/4/2021 and the test window ran for a one year period from 1/11/2021 to 1/3/2022. Since some of the parameters have seasonal periods of a cycle of a year in duration, it is best to employ many years into the training set for a better fit.

Lastly, all missing values were imputed using `ts_impute_vec()`, which according to R documentation performs the following three steps: “Seasonal Trend Loess (STL) decomposition”; “linear interpolation...applied to the seasonally adjusted data”; and final reincorporation of the seasonal component (R Core Team, 2022).

Modeling

In this study, several modeling techniques with varying degrees of complexity were used in order to assess whether simpler models would suffice to achieve the business and technical objectives, or whether the structure of the data required more sophisticated techniques to achieve un-biased fit and highly accurate predictive performance.

Exponential smoothing: Holt-Winter’s

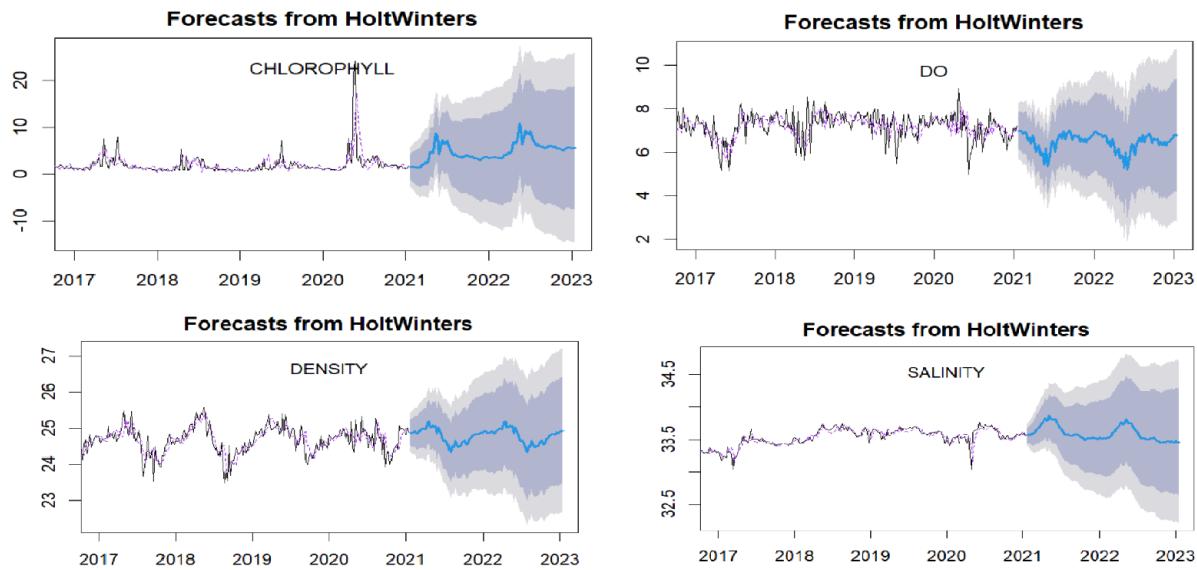
Holt-Winter’s Model takes into account seasonality and trend into the forecasting. It is referred to as triple exponential smoothing because it incorporates three aspects (Shmueli, & Lichtendahl Jr., 2018):

1. Simple exponential smoothing for forecasting data that has no trends or seasonality.
2. Holt’s smoothing method, which is a widely used smoothing method for forecasting data with trend.
3. Winter’s smoothing method, which allows inclusion of seasonality while forecasting along the trend.

For PLOO all training dataset, a generally good fit was able to be made with the automatic parameter detection `HoltWinters()` function call. See Figure 6 below.

Figure 6.

Holt-Winter's Forecast for Multiple Parameters



Linear Regression: Time Series with Trend and Seasonality

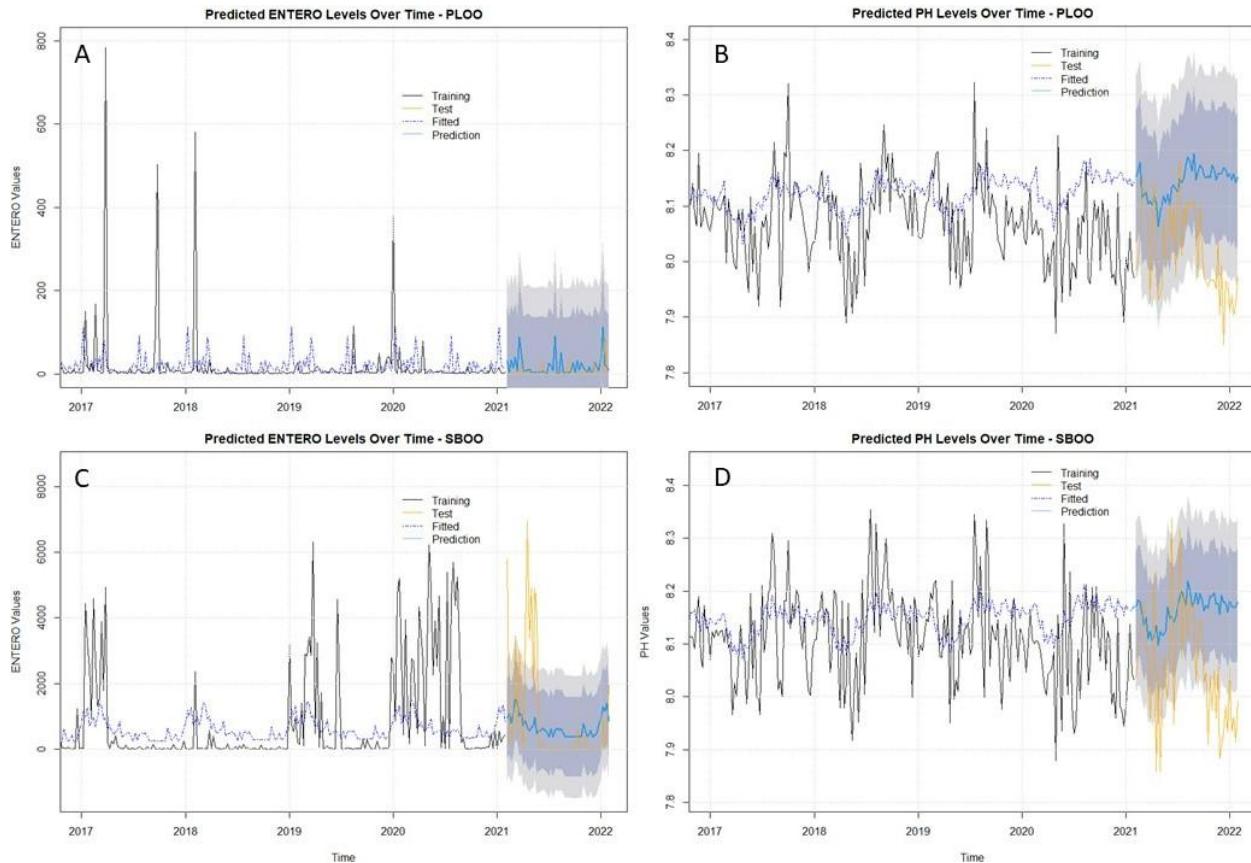
Two variations on linear regression models were used to forecast parameter values. The first, discussed in this section was time series linear regression with the inclusion of trend and seasonality; both components were included based on Figure 4 (for PLOO) and Figure 5 (for SBOO). However, based on the sizable differences between each STL breakdown, it is expected that there will be significant differences in efficacy of linear regression without including autocorrelation, as will be done in the ARIMA model. PLOO has a definite seasonal component, but in reviewing the data segment, there are large spikes as cycles larger than seasons that won't be accounted for in the linear regression models. The trend also may prove difficult to mitigate since it modulates quite significantly over the entire training period and does not cleanly match a clear sine or cosine pattern. SBOO is slightly more consistent in terms of seasonality, though there are still large differences between lows and highs. Also, trend is less consistent than with PLOO, which will most likely skew results. Figure 7 shows the linear regression forecasts for individual parameters ENTERO and PH time series, with the dashed

blue line as the fitted model, the orange line as the test data, and the light blue prediction line.

The details are discussed further in the Results section.

Figure 7

Linear Regression Forecasts for Two Parameters (ENTERO and PH) at Two Sites (PLOO and SBOO)



Advanced regression: ARIMA

Multiple ARIMA (AutoRegressive Integrated Moving Average) models were developed on the dataset after imputing missing values and splitting into the training window and test window. The models were made for PLOO and SBOO project regions with the initial priority to forecast ENTERO values for those general locations. Next, ARIMA models with predictors were implemented to forecast ENTERO based on PH, DO, FECAL, SALINITY, CHLOROPHYLL, TEMP, DENSITY, and XMS. Finally, additional models were developed that focused on

forecasting the important features from the regression model (PH, SALINITY, and DENSITY), as based on the relative level of coefficient values. In order to develop adequate models, the time series autocorrelation function (ACF) and partial ACF (PACF) plots were evaluated for each of the target variables to determine the model parameters.

ARIMA models have two potential orders, the first being the non-seasonal part or (p, d, q) where p is the number of autoregressive terms, d is the differencing that occurred, and q is the number of moving average terms (Shmueli & Lichtendahl Jr., 2018, p. 149). The second order set is for datasets where there is a seasonal component and is denoted as (P, D, Q) where P deals with patterns in the PACF plot, D is seasonal differencing, and Q deals with patterns in the ACF plot. The final models were tested with the test series and their MAPE and MAE values for training and testing were placed into a table for comparison with other modeling methods.

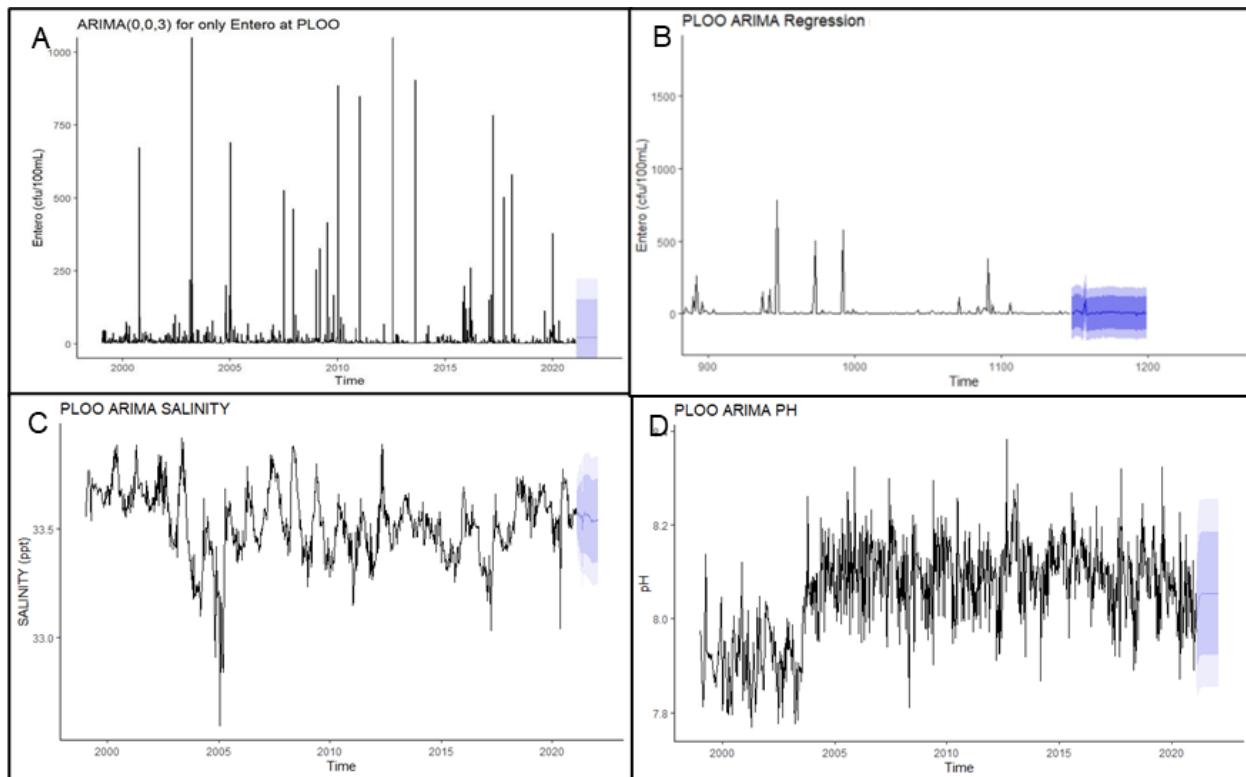
ARIMA Models at PLOO Project.

The ENTERO model at PLOO utilized only non-seasonal model components due to the lack of definitive seasonality in the time series plot of ENTERO. This model was tested at a p term of zero, differencing term of zero, and a q term of three and four, with three being the final model chosen by RMSE value comparison and simplicity, since the RMSE values were identical when rounding. The regression model at PLOO utilized the same model order but incorporated the predictors as a matrix to determine the ENTERO values at later time points. After running this model, the coefficient values of the different parameters were investigated. The absolute values of the parameter coefficients ranged from 0.418 (FECAL) to 73 (PH) with PH, DENSITY, and TEMP being the top three important coefficients. While the ENTERO regression plot did not incorporate seasonality in the model, the forecasts for DENSITY and SALINITY both required the seasonality component. The model for DENSITY at PLOO had a nonseasonal order of $(2, 0, 1)$ and a seasonal order $(1, 0, 0)$. The model for PH at PLOO utilized a non-seasonal order of $(1, 0, 0)$. The model for SALINITY utilized a nonseasonal order of $(2, 0, 1)$ and seasonal order $(1, 0,$

0). The resulting forecast plots of the PLOO data (i.e., both ENTERO models, PH, and SALINITY) can be seen in Figure 8 forecasts.

Figure 8

ARIMA forecast plots of select parameters at PLOO. A (ENTERO), C (SALINITY), and D (PH) are individual parameters while B is forecasting ENTERO based on regression of eight parameters as predictors.



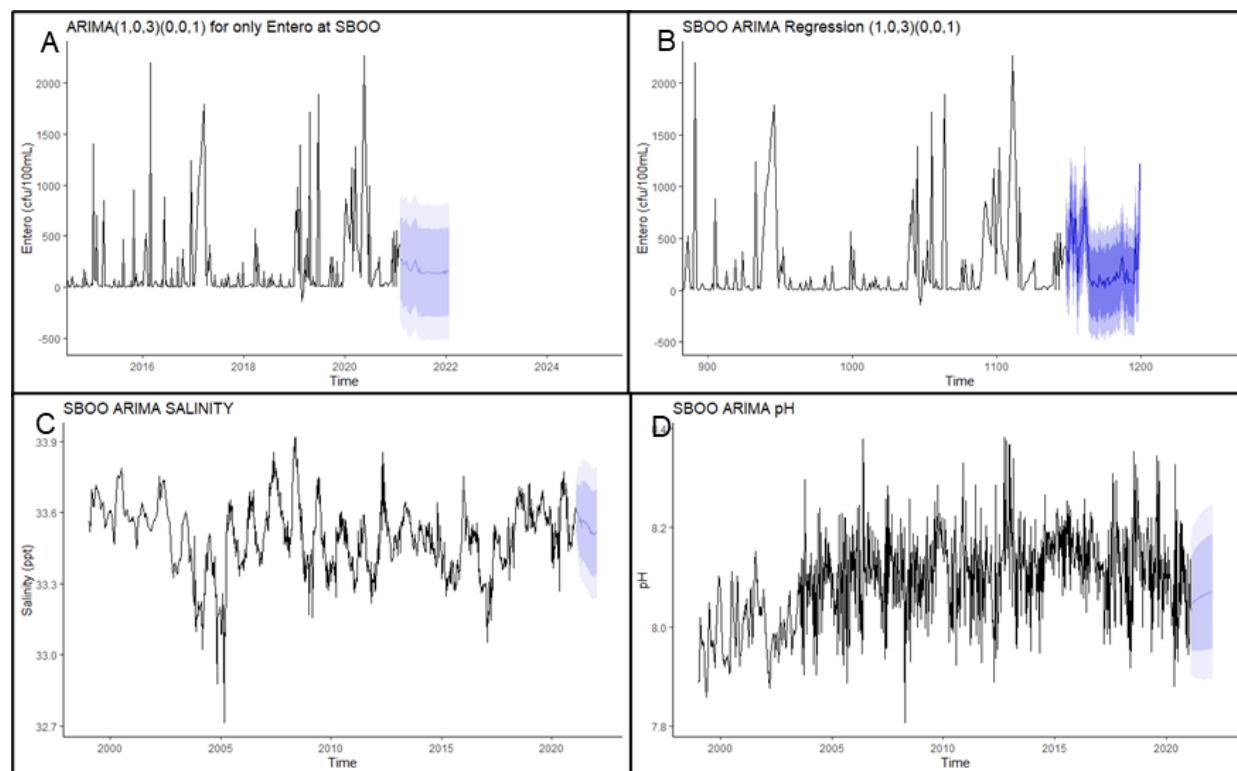
ARIMA Models at SBOO Project.

The SBOO project had more pronounced seasonality in the ENTERO time series so that was incorporated into the ENTERO alone model and the ENTERO regression model. Initially, two models were developed for the ENTERO alone ARIMA model with differing values for the q or moving average portion of the nonseasonal order. The first model had a non seasonal order of (1, 0, 2) and a seasonal order of (0, 0, 1) while the second model had a non seasonal order of (1, 0, 3) and a seasonal order of (0, 0, 1). The RMSE of these models was compared with the

lower value being the model that was selected as the final model, in this case the model with q valued at three. These parameters were then applied to the ENTERO ARIMA regression model with the eight parameters acting as predictors. After running the model, the coefficient values of the different parameters were examined. The values ranged from 0.389 (FECAL) to 224 (PH) with PH, SALINITY, and DENSITY being the most important predictors. Additional ARIMA models were developed for the SALINITY, PH, and DENSITY at SBOO, based on these being important parameters. The DENSITY and SALINITY both displayed seasonality while PH did not, similar to the behavior seen at PLOO. The DENSITY model utilized a nonseasonal order of (2, 0, 1) and a seasonal order of (1, 0, 0) while the SALINITY model had a nonseasonal order of (1, 0, 1) and seasonal order of (1, 0, 0). The PH model had only the non seasonal order at (2, 0, 1). The resulting forecasts from regression can be seen in Figure 9: A (ENTERO), C (SALINITY), and D (PH) are individual parameters, while B is forecasting ENTERO based on regression of eight parameters.

Figure 9

ARIMA Forecast Plots of Select Parameters at SBOO



Results

The MAPE provides the average percent difference between the forecasted value and the actual value and is a common metric for comparing to baseline model forecasts (Zach, 2020). The MAE provides the mean absolute difference between forecast values and actual values and is particularly useful when large errors are not harmful, whereas in root mean square error (RMSE) is better for when large errors are needing to be avoided (JJ, 2016). In terms of a “best” modeling technique, there was not one that did outstandingly well on everything. The results for each varied greatly both between each parameter, as well as across both projects.

Table 2 and Table 3 summarize the performance of each model on select parameters for the two project locations, PLOO and SBOO respectively.

Table 2
PLOO Model Evaluation via MAPE and MAE

PARAMETER, Model	MAPE		MAE	
	Training	Test	Training	Test
DENSITY				
Seasonal Naive	1.356	1.004	0.335	0.249
Holt-Winter's	0.839	2.184	0.207	0.552
Linear Regression	1.110	1.526	0.274	0.380
ARIMA	0.780	0.960	0.193	0.239
PH				
Seasonal Naive	0.988	0.777	0.080	0.062
Holt-Winter's	0.676	1.041	0.054	0.082
Linear Regression	0.881	1.629	0.071	0.130
ARIMA	0.660	0.936	0.053	0.075
SALINITY				
Seasonal Naive	0.368	0.113	0.123	0.038
Holt-Winter's	0.158	0.326	0.053	0.110
Linear Regression	0.335	0.285	0.112	0.096
ARIMA	0.130	0.202	0.044	0.068
ENTERO				
Seasonal Naive	511	150	32.6	8.88
Holt-Winter's	523	212	32.3	11.2
Linear Regression	401	510	27.6	16.7
ARIMA	437	577	27.9	18.1
ENTERO FROM EXTERNAL PREDICTORS				
Linear Regression	395	564	28.5	19.8
ARIMA	296	250	25.6	11.5

PLOO Project

The ARIMA model did very well on the test set as compared to the seasonal naive baseline, both in terms of MAPE and MAE. However, for PH, SALINITY, and ENTERO, none of the models did better than the baseline. Since the training MAPE and MAE values were lower than the baseline, this is a very strong indication that there was a great deal of overfitting occurring. These results vary pretty significantly from SBOO's, which are detailed below. For instance, in reviewing the ENTERO prediction model that included external predictors, the test MAE was 11.5, as compared to 151 for SBOO. Though the reasons for this would need to be explored further, it is clear that there is a strong justification for modeling the two projects separately based on vastly different patterns.

Table 3

SBOO Model Evaluation via MAPE and MAE

PARAMETER, Model	MAPE		MAE	
	Training	Test	Training	Test
DENSITY				
Seasonal Naive	1.377	1.305	0.339	0.323
Holt-Winter's	0.795	1.141	0.196	0.287
Linear Regression	1.032	1.445	0.254	0.360
ARIMA	0.742	1.120	0.183	0.277
PH				
Seasonal Naive	1.010	1.251	0.082	0.101
Holt-Winter's	0.712	0.987	0.058	0.079
Linear Regression	0.745	1.587	0.060	0.127
ARIMA	0.638	1.110	0.052	0.089
SALINITY				
Seasonal Naive	0.351	0.115	0.039	0.039
Holt-Winter's	0.120	0.150	0.040	0.050
Linear Regression	0.304	0.267	0.102	0.090
ARIMA	0.108	0.215	0.036	0.072
ENTERO				
Seasonal Naive	1913	828	490	939
Holt-Winter's	2346	3379	433	311
Linear Regression	3518	3298	482	1063
ARIMA	1105	885	161	294
ENTERO FROM EXTERNAL PREDICTORS				
Linear Regression	1165	970	220	327
ARIMA	648	446	122	151

SBOO Project

In comparison to the seasonal naive baseline model, the linear regression model that only included trend and seasonality did not fare very well; it consistently had a higher MAPE and MAE on the test data. On the other hand, both Holt-Winter's exponential smooth and ARIMA models generally outperformed the baseline. Only on SALINITY did they have higher MAE. It is interesting to note that ARIMA, the more sophisticated model, did not do better than Holt-Winter's, except on predicting DENSITY, where ARIMA had a slightly lower error. When looking at the results for the ENTERO-only models (excluding external predictors), all of the algorithms produced higher MAPE than the baseline, but oddly enough, the MAE for Holt-Winter's and ARIMA were lower on the test set than the baseline.

Discussion

With the slight improvement in ENTERO forecasting from utilizing all eight predictors, it can be argued that that method of forecasting ENTERO is the way to go, with some caveats. These caveats include the need to fine tune predictor selection in an attempt to decrease the MAPE and MAE scores. Other caveats could be to fine tune the ARIMA model parameters of the ENTERO only model which could include removing seasonality or trend more than was done in this project. All of these caveats would have to be taken under the assumption that samples of predictors are collected before or in lieu of ENTERO samples (e.g., they are available at the time of forecast). Under the assumption of predictor samples being taken at the same time as ENTERO samples, then an ENTERO alone model can be used while the other predictors can be used to monitor other ecological measures.

To forecast other parameters by themselves, it can be assumed that those would be sufficient if additional tests of ENTERO are performed when corresponding levels of pH, salinity, or density are forecast to be at a troubling level that correlates to higher levels of entero. These

other measures may be easier to monitor, if they do not require plates or growth media to determine colony forming units of microbial matter (the measure of ENTERO and FECAL).

Ultimately, for the current study, external predictors were only used to determine which parameters besides ENTERO would be worthwhile to examine for assessment as part of a more integrated modeling process in the future.

Conclusion

The initial combined data had information for nine chemical or biological parameters; depth, date, location, and time of sample for samples taken from 1991 to 2021; project; qualifiers (e.g, > or <); and units. Aggregating the data, imputing missing data, and handling outliers created many datasets with many modeling options. The group did model on many of these datasets but focused on reporting the models that used the dataset with imputed values and outliers. Overall, the models for forecasting non-ENTERO factors performed better than the ENTERO models. Through the regression models that forecast ENTERO through other parameters, we found that there is a strong relationship between PH, SALINITY, and DENSITY since these had high coefficient values at both locations. Interestingly, in the ARIMA model the FECAL value was not an important coefficient in either location which went against initial assumptions, since enterococci bacteria is commonly found in FECAL matter. With these insights, monitoring pH, water density, and/or salinity could help alert groups that entero tests need to be conducted in order to confirm that there is a looming public health matter that needs to be addressed. It will also provide insight to environmental groups and other monitoring groups that there is an issue with waste disposal nearby that is causing a peak or dip in the forecasted values that could negatively impact aquatic life. These insights will hopefully allow the community to be better aware of the consequences of their actions and allow for advanced knowledge of times when specific beaches need to be closed down or enhanced water treatment needs to be implemented to prevent unnecessary extremes of different parameters.

Future Directions

The team acknowledges that their models, while reasonably good, can be improved. Improvements to models in the future can include fine tuning parameters or selecting model windows that reflect long seasonal impacts, like an El Nino year. The regression models can go through parameter selection by focusing on important parameters and excluding the less important ones (e.g., FECAL, XMS, TEMP, CHLOROPHYLL) or focusing purely on chemical and/or chemical observational features (e.g., PH, TEMP, SALINITY, DENSITY, DO) since those measures are presumably cheaper to monitor than bacteriological ones that require microbial growth plates and colony counting.

Another aspect to explore further would be the interaction between different parameters and to confirm consistent relationships across all regions of the ocean and not just in the San Diego bay waters. If we can understand, more concretely and in-depth the relationship between the parameters discussed in this paper, we can look for leading or lagging indicators that would signal to us the upcoming entero levels and be able to warn the relevant San Diego bay waters organizations of any potential upcoming dangers for visitors to the beach. Further, a more global approach to maintaining ocean water quality standards can be developed to ensure a brighter ecological future for our ocean water and marine life and other ecosystems that depend on the health of our sea waters.

Lastly, as stated in the beginning of this paper, we hope to develop a system to accurately forecast levels of enterococci (entero) bacteria, which at levels above 104 per 100 milliliter (mL) cause serious public health risk (New Hampshire Department of Environmental Services, 2019). We believe that the combination of ARIMA, Holt-Winters and Linear Regression models discussed in this paper will be employed to predict entero levels on a monthly basis for the San Diego Bay seawater. In conjunction with the appropriate organization, this information can be used to warn beachgoers of any entero outbreaks which would pose serious health risks for swimmers and any in contact with the water. Further studies and work

would entail the study of interactions between the different parameters discussed in this paper and how they tie into the entero levels and other health hazards and marine life endangerment.

References

City of San Diego. (n.d.-a). *PI200707_ch2_ocean_conditions.pdf*.

https://www.sandiego.gov/sites/default/files/legacy/mwwd/pdf/pl200707_ch2_ocean_conditions.pdf

City of San Diego. (n.d.-b). *Public Utilities: Ocean Monitoring*.

<https://www.sandiego.gov/public-utilities/sustainability/ocean-monitoring>

City of San Diego (2022, June 30). *2020-2021 Biennial receiving waters monitoring and assessment report for the Point Loma and South Bay ocean outfalls*.

https://www.sandiego.gov/sites/default/files/compressed_2020-2021_biennial_receiving_waters_monitoring_report_0.pdf

City of San Diego. (2022, July 25). *Water Quality - Ocean Monitoring Program* [Data set]. Data SD. <https://data.sandiego.gov/datasets/monitoring-ocean-water-quality/>

Dao, H. N., Vu, H. T., Kay, S., & Sailley, S. (2021). Impact of Seawater Temperature on Coral Reefs in the Context of Climate Change. A Case Study of Cu Lao Cham – Hoi An Biosphere Reserve. *Frontiers in Marine Science*, 8.

<https://doi.org/10.3389/fmars.2021.704682>

JJ. (2016, March 23). MAE and RMSE — Which Metric is Better? Medium; Human in a Machine World.

<https://medium.com/human-in-a-machine-world/mae-and-rmse-which-metric-is-better-e60ac3bde13d>

NASA. (2019, August 31). *Sea Surface Temperature & Chlorophyll*. Nasa.gov; NASA Earth Observatory.

https://earthobservatory.nasa.gov/global-maps/MYD28M/MY1DMM_CHLORA

National Oceanic and Atmospheric Administration. (2020, April 1). *Ocean Acidification*. NOAA; U.S. Department of Commerce.

<https://www.noaa.gov/education/resource-collections/ocean-coasts/ocean-acidification>

New Hampshire Department of Environmental Services. (2019). Bacteria in Surface Waters. In *Environmental Fact Sheet*. New Hampshire Department of Environmental Services.

<https://www.des.nh.gov/sites/g/files/ehbemt341/files/documents/2020-01/bb-14.pdf>

New Zealand Government. (2010, June). *Ocean density*. Science Learning Hub;

<https://www.sciencelearn.org.nz/resources/687-ocean-density>

Peng, Z., Hu, W., Liu, G., Zhang, H., Gao, R., & Wei, W. (2019). Development and evaluation of a real-time forecasting framework for daily water quality forecasts for Lake Chaohu to Lead time of six days. *Science of The Total Environment*, 687, 218–231.

<https://doi.org/10.1016/j.scitotenv.2019.06.067>

R Core Team. (2022). R: A language and environment for statistical computing. *R Foundation for Statistical Computing*, Vienna, Austria. URL <https://www.R-project.org/>.

Shajratul, A. (2020). Analysis of missing data in marine dissolved oxygen time series using dynamic linear models. *Open Access Master's Theses*. Paper 1903.

<https://digitalcommons.uri.edu/theses/1903>

Shamshirband, S., Jafari Nodoushan, E., Adolf, J. E., Abdul Manaf, A., Mosavi, A., & Chau, K. (2018). Ensemble models with uncertainty analysis for multi-day ahead forecasting of chlorophyll a concentration in coastal waters. *Engineering Applications of Computational Fluid Mechanics*, 13(1), 91–101. <https://doi.org/10.1080/19942060.2018.1553742>

Shmueli, G., & Lichtendahl Jr., K. C. (2018). *Practical Time Series Forecasting with R: A Hands-On Guide* (2 ed.). Axelrod Schnall Publishers.

Snehal_bm. (2021, August 3). *Holt Winter's Method for Time Series Analysis*. Analytics Vidhya; Data Science Blogathon.

<https://www.analyticsvidhya.com/blog/2021/08/holt-winters-method-for-time-series-analysis/>

- US EPA. (2022, August 1). *Climate Change Indicators: Sea Surface Temperature*. Climate Change Indicators; United States Environmental Protection Agency.
<https://www.epa.gov/climate-indicators/climate-change-indicators-sea-surface-temperature>
- Water Education Foundation. (2014, September 15). *Aquapedia Background: Salinity*. Water Education Foundation. <https://www.watereducation.org/aquapedia-background/salinity>
- Water Research Center. (n.d.). *Surface Water: The Role of Fecal Coliform and Waterborne Pathogens*. <https://www.knowyourh2o.com/outdoor-4/fecal-coliform-bacteria-in-water>
- Wisconsin Department of Natural Resources. (n.d.). Factors that Affect Water Clarity Factors that Affect Water Clarity. In *WISCONSIN CITIZEN LAKE MONITORING TRAINING MANUAL* (pp. 15–20).
<https://www3.uwsp.edu/cnr-ap/UWEXLakes/Documents/programs/CLMN/publications/ChemistryManual/Chapter3.pdf>
- Zach. (2020, December 24). *What is Considered a Good Value for MAPE?* Statology.
<https://www.statology.org/what-is-a-good-mape/>

Appendix A

Description of Data Set Features

Chlorophyll: An important indicator of the presence of algae and other plant-like microorganisms that carry out photosynthesis (NASA, 2019). Phytoplankton, an essential food base in the marine food chain, feeds off of chlorophyll (NASA, 2019).

pH: According to the National Oceanic and Atmospheric Administration (2020), the average ocean pH right is 8.1, which is basic or alkaline. As the ocean absorbs more CO₂, it becomes more acidic. This creates a less supportive environment for marine life. Ocean acidity is already affecting oysters and corals which combine calcium and carbonate in the seawater to form their shells. Researchers have found that tiny “sea butterfly” organisms, who are an important base in the food web of larger animals like krill and whales, have already seen their shells dissolve due to increased acidity in the seawater in the Southern Ocean near Antarctica (National Oceanic and Atmospheric Administration, 2020).

Salinity: Saline water stunts plant growth by dehydration, preventing nitrogen uptake and poisoning with chloride ions (Water Education Foundation, 2014). In freshwater, salt damages the health of plants and aquatic life, putting species at risk.

Temperature: In the case of coral reefs, for example, prolonged increased temperatures lead to bleaching which overtime degrades the coral reefs (Dao et al., 2021). Changing temperatures make the water less suitable for supporting native species overtime.

Fecal: The source of the fecal matter may contain pathogens or disease producing bacteria and viruses. Untreated fecal matter depletes the level of oxygen in the water and such levels may kill off fish and other aquatic life native to those waters (Water Research Center, n.d.).

XMS: Transmissivity or water clarity? Water clarity is important for a number of reasons. It affects the depth to which aquatic plants can grow, dissolved oxygen content, and water

temperature. Fish and loons and other wildlife depend on good water clarity to find food (Wisconsin Department of Natural Resources, n.d.).

DO (Dissolved Oxygen): When the oxygen level in the water becomes too low, the water will not be able to support the wildlife. As water becomes warmer, it cannot hold as much oxygen which in turn affects the marine wildlife that it supports.

ENTERO: What level of Enterococci is acceptable? Enterococci levels at designated salt water, coastal beaches should not exceed 104 per 100 milliliter (mL) in any one sample, or exceed a three-sample geometric mean average over a 60-day period of 35/100 mL (New Hampshire Department of Environmental Services, 2019).

Density: Affects the temperature of the water and how heat is circulated throughout the ocean (Ocean Density, 2010).

SUSO: Suspended solids.

All of these are intertwined with each other. For example, salinity, temperature, and depth all affect the density of the seawater. As salinity increases, density also increases, etc.

Appendix B

Literature Review: Individual Source Analysis

Source 1: Shajratul (2020)

Research Paper Link: <https://digitalcommons.uri.edu/theses/1903>

Background Information

In this research paper, the sea water of Narragansett Bay by Rhode Island was analyzed with the hope of understanding the dynamics of the oxygen level in the water over time in order to predict the onset of hypoxia events which lead to the dying off of marine life due to inadequate oxygen. The dataset contains empirical data collected by the fixed-stations setup around the bay collecting data directly from sensors.

In terms of a literature review, our project will add to the existing body of knowledge of the study of sea water around the world. We will contribute our findings for the coastal waters of Southern California. We hope that this knowledge will contribute to a global understanding of how the sea water quality is changing into the future.

Overview of Main Points

Proper data analysis methods were employed for this study, including time series analysis combined with dynamic linear models to impute missing values. Characteristics of the sea water, like temperature and salinity were each modeled with their own time series to be studied independently.

Strengths.

- Problem statement is clear, they are trying to predict the onset of low oxygen (hypoxia) events in order to protect marine wildlife.
- Testing method was to determine the structure of model components and also looking through autocorrelation function (ACF), partial ACF (PACF), and CCF between the series.

- Experiment is well set up with the selected covariates to DO dissolved oxygen levels including collecting data on salinity, temperature, and river discharge.
- Use of time series forecasting methods and dynamic linear models used to impute missing values in the dataset.
- Possessed clear understanding of empirical data: When imputing missing data, it is a better method to work with DO% time series because this measurement is collected directly from water while the DO mg/l is calculated based on the measurement of temperature and salinity.

Weaknesses.

- Loss of focus on research paper, by the end discussion chapter the author states the focus of this paper is to fill in the missing parameters for water quality measurements and makes no conclusions about the effect of DO% and covariates salinity and temperature in the broader context.
- The broader implications of the results are not well-explained. For example, temperature time series results showed that temperature increased between 2011 and 2012 but no broader implications were discussed. As a result, there was no context as to what the author's findings mean.

Key Findings

The focus of the study was to analyze the water quality for Narragansett Bay, particularly the DO (dissolved oxygen) characteristic along with covariate terms like salinity, pH, etc.

- Salinity in the water was controlled mostly by the river discharge data from both sites. It is highly influenced by freshwater input from nearby rivers or precipitation.
- Variability in the oxygen level depended mostly on the water temperature, followed by the river discharge data.
- Time series analysis showed a periodicity of one year for temperature.

- A side-by-side comparison of the daily river discharge showed that the salinity decreased around the same time that the river discharge increased.
- The variability of salinity in water depends on freshwater inputs from rainfall or river discharge.

Source 2: Shamshirband et al. (2018)

Research Paper Link:

<https://www.tandfonline.com/doi/full/10.1080/19942060.2018.1553742>

Background Information

The group led by Shamshirband focused on utilizing different time series analysis methods, including ensemble models, to forecast the chlorophyll α and salinity in the Pacific Ocean off the coast of Hawaii. Chlorophyll α is a specific type of chlorophyll and is indicative of the same ecological measures for the presence of phytoplankton to supply food for the base level animals of the food chain. Water salinity indicates how salty the water is and for this study was used to determine the generality of the methods used to forecast chlorophyll α concentrations.

This group focused on using the Bates-Granger approach with least square method to create an artificial neural networks (ANN) model with discrete wavelet transformation. ANNs have three stages with the first being the input (measured values) that connects to the hidden layer then then connects to the output layer. This group used back propagation to determine weights of parameters for the best model. The group also utilized a discrete wavelet transform (DWT) to extract frequency and time information for time series decomposition purposes.

Overview of Main Points

The group utilized daily measures of chlorophyll α and salinity were obtained by averaging the measurements taken every 15 minutes from one research buoy at a depth of 0.75m from January 1, 2012 through December 31, 2016. The group removed measurements

or days that were missing or of low quality which resulted in fewer measurements for the main chlorophyll α study than the salinity study.

The group utilized ensemble modeling to improve forecast reliability through reducing bias and variability. They observed a time series for creating the forecast with up to three lags using the DWT in order to predict chlorophyll concentrations for three days in advance. Their models were evaluated using RMSE and R-squared measures to determine model performance.

Gaps in knowledge.

The lack of other groups utilizing ensemble methods to forecast water quality measures for specific parameters, at least at the time that this study was performed.

Strengths.

- Large timeframe to pull data from to determine trends.
- Utilized different methods to improve model performance and thus forecast performance.
- Their methods are general enough to apply to other water quality measures, as they confirmed by repeating their methods with the salinity data.

Weaknesses.

There was only forecasting for three days in advance. It would be interesting to see further forecasts even though the performance of forecasting decreases the further out you look.

Key Findings

Ensemble models were better than individual models and this improvement was enhanced by using an increased lead time. The improvement of ensemble over individual model performance is also seen in the narrowing of the width of the uncertainty band.

Source 3: Peng et al. (2019)

Research Paper Link:

<https://www.sciencedirect.com/science/article/abs/pii/S004896971932635X>

Background Information

In light of increased awareness of the impact of industrial and urban growth on drinking water, Peng et al. (2019) established a study focused on forecasting water characteristics of several factors that are seen as contributing to pollution in freshwater Lake Chaohu, which is “in the middle-lower Yangtze River basin of China.” However, instead of using traditional statistical model techniques, as often seen in the literature, Peng et al. focused on developing “dynamic forecast models”, specifically implemented as a package called EcoLake. These models—which use adulterants contained with the water itself, along with key physics-based factors (e.g., “hydrodynamics and nutrient cycling”)--are able to represent nonlinear interactions of diverse environmental variables with the goal of a more sophisticated and accurate forecasting system, while relying on less temporal-dependent data (i.e., over a shorter time period).

Overview of Main Points

The implementation of this study hinged upon several key characteristics that both bolstered its overall strength, but some of which contributed to the following issues:

- The study was based on relatively limited data 2016-2017, though study authors argued that the dynamic forecasting models are not as reliant on large chunks of historical data as statistical models.
- The lead period and resulting forecast period were relatively short (1-6 days), as the methods employed made larger forecasting periods increasingly less accurate.
- In terms of water quality, the study focused only on chemical adulterants, and not combination of chemical and biological. Therefore, the interaction between biological contaminants (e.g., enterococci) and environmental factors are not present and analyzable, which given their prevalence in many areas of increased pollution is a missed opportunity.

- Dynamic forecast modeling is a very complex system that relies on many factors that require complex sampling methods, as well as the incorporation of regression prediction for some features (which then adds further potential for uncertainty).

Key Findings

The methods used to provide short-term forecasts performed better than simple “persistence reference forecasts”. However, there was a large variation between forecasting performance for different water quality attributes (e.g., dissolved oxygen vs. total nitrogen). Lastly, the necessity to rely on additional layers of traditional forecasting methods for some predictor variables before insertion into the EcoLake system had high potential for introducing bias and skewing final forecasting results. As noted by the authors themselves, “due to erroneous inputs, approximation of ecosystem processes and unresolved model scales, uncertainty is inherent in water quality forecasts and must be taken into consideration before a forecast can be useful for management decisions”.

Source 4: City of San Diego. (n.d.-a)

Research Paper Link:

https://www.sandiego.gov/sites/default/files/legacy/mwwd/pdf/pl200707_ch2_ocean_conditions.pdf

Background Information

In this research paper, the sea water in the region extending from Northern San Diego to Baja Mexico, over 420 square miles are studied to analyze to understand the dynamics of the sea water quality with the input of treated wastewaters being released at depths of 94-98m 7.2km west of Point Loma. This paper covers the analysis on 4 key aspects:

1. How the wastewater discharge affects the ocean water quality
2. What are the effects of the wastewater discharge compared to other input sources
3. What are the effects of the dispersion of the wastewater discharge materials
4. Understand the effect of other inputs like El Nino.

In terms of a literature review, our project will add to the existing body of knowledge of the study of sea water around the world. We will contribute our findings for the coastal waters of Southern California. We hope that this knowledge will contribute to a global understanding of how the sea water quality is changing into the future.

Overview of Main Points

Thirty six off-shore stations utilize a SeaBird conductivity, temperature, and depth (CTD) instrument to collect ocean water quality measurements.

Strengths.

While the problem statement covers too many points and one clear focus is not apparent, each discussion section on salinity, temperature and dissolved oxygen does a good job tying the results back to the initial key four points of exploration.

Weaknesses.

Paper focused had too many aspects to cover in terms of how the wastewater discharge affected the ocean water quality. It should have focused on one of the points and made a comprehensive research paper out of each point. Study did not incorporate use of time series analysis or any data mining methods to forecast future pH, salinity, temperature or other ocean water quality measurements. It only made an observation into past values to judge if the wastewater discharge had affected the water at the surface taken at the PLOO station sensors.

Key Findings

- Water temperature is the main factor affecting water density and the best indicator of the “surfacing potential” of wastewaters is the differences in temperature between the bottom and top surface temperatures. The “thermal stratification” follows normal seasonal patterns.
- There were no apparent trends in PH values or dissolved oxygen concentrations related to the PLOO station.

- Dissolved oxygen and pH follow seasonal variations that make “temporal” changes difficult to evaluate.
- Oceanographic conditions generally followed seasonal patterns. Wastewater plume was not detectable via aerial imagery meaning that no wastewater had surfaced and that it had remained trapped in relatively deep waters.
- No long term changes to the water quality detected at the PLOO station were detected or attributed to the wastewater discharge. pH levels and dissolved oxygen content showed no apparent relation to the wastewater discharge.

Appendix C

Supplementary Tables and Figures

Table 4

Feature NA Count for ENTERO Category

	date_sample									
	sample	station	depth_m	sample	time	project	parameter	qualifier	value	units
Not NA (n)	144,341	144,341	116,186	144,341	144,012	144,341	144,341	136,251	143,075	144,341
NA (n)	0	0	28,155	0	329	0	0	8,090	1,266	0
Not NA (%)	100.0%	100.0%	80.5%	100.0%	99.8%	100.0%	100.0%	94.4%	99.1%	100.0%
NA (%)	0.0%	0.0%	19.5%	0.0%	0.2%	0.0%	0.0%	5.6%	0.9%	0.0%

Figure 10

Histograms per project for bacteriological factors (CHLOROPHYLL, ENTERO, FECAL, and SUSO) each with 7 bins

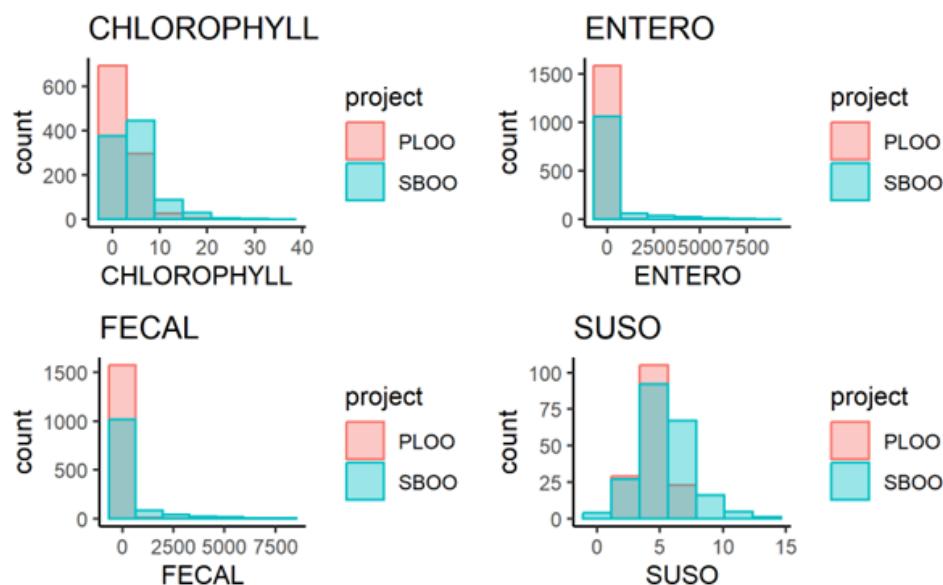


Figure 11

Histograms per project for chemical factors (DENSITY, DO, OG, XMS, PH, SALINITY, and TEMP) each with 7 bins

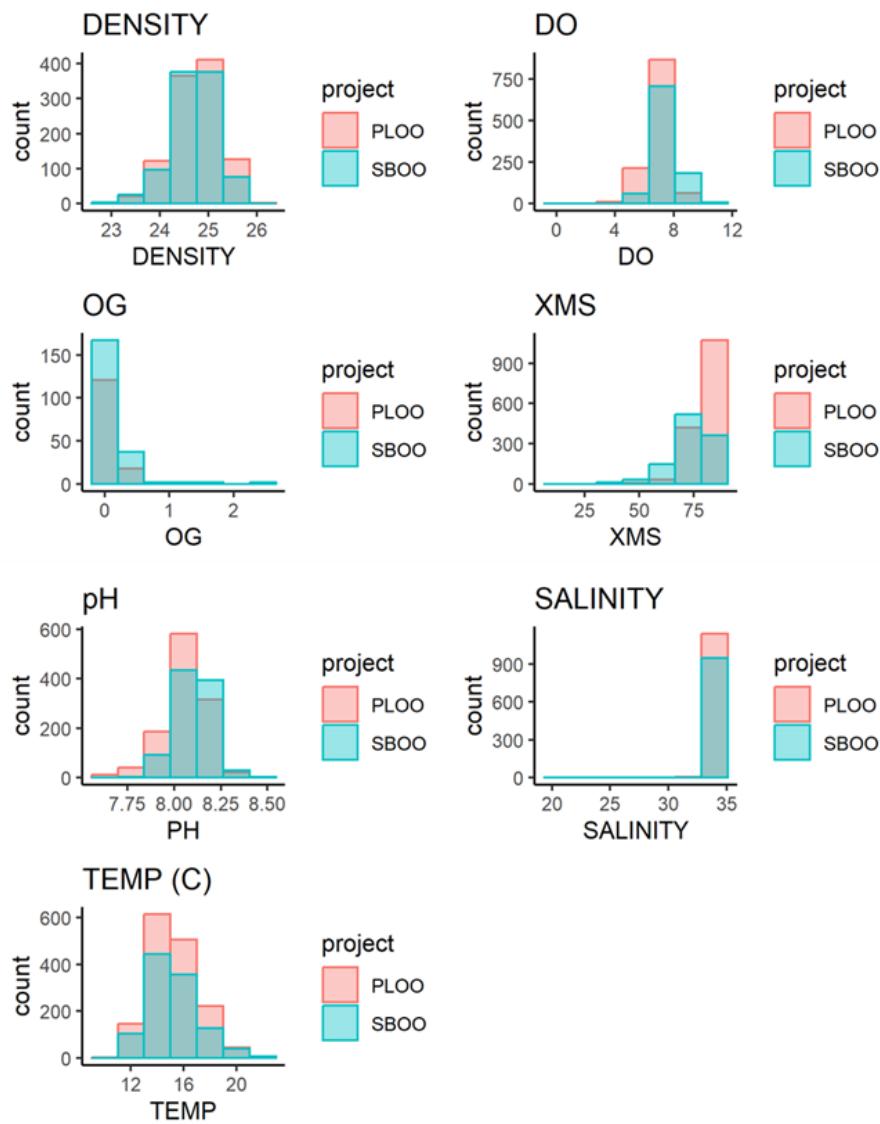


Figure 12

Correlation of features excluding OG and SUSO due to being unpopular measurements.

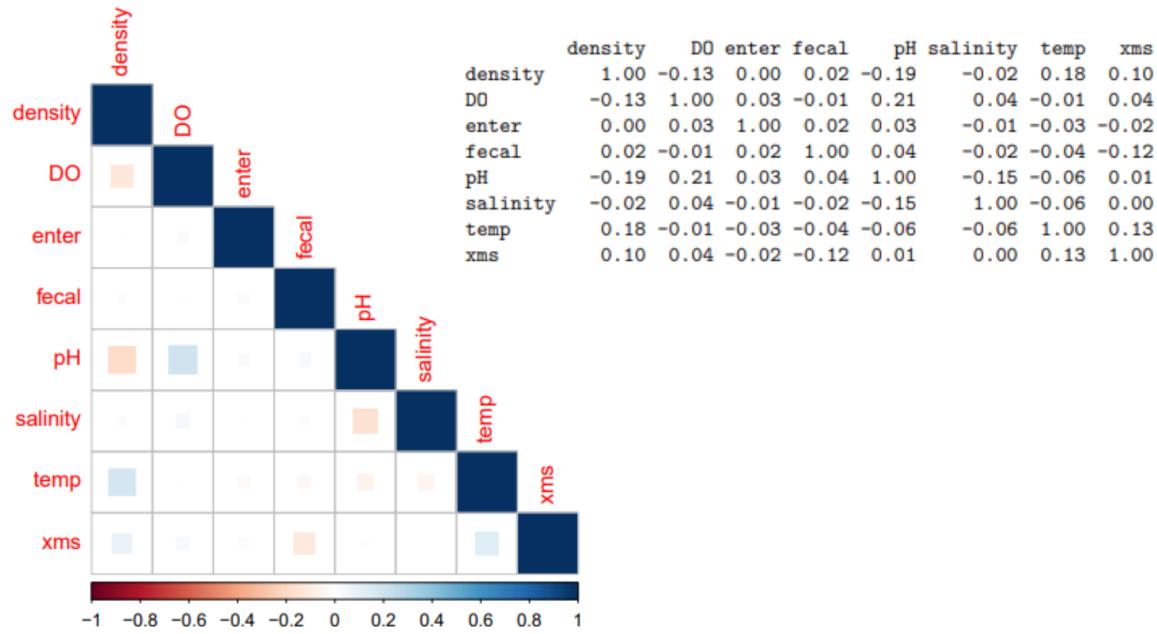
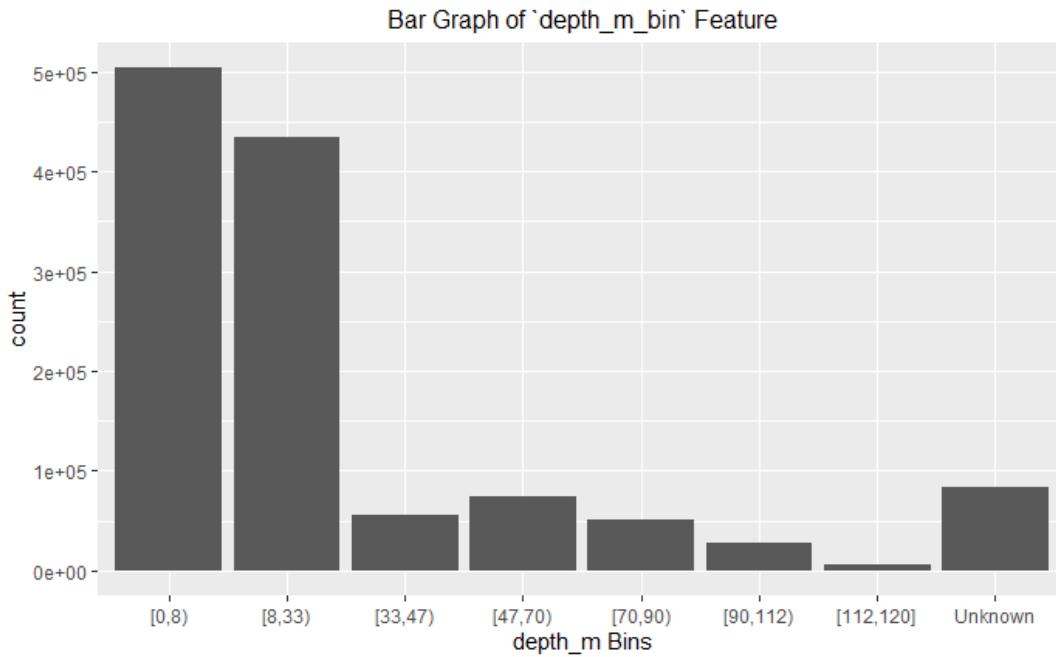
**Figure 13**

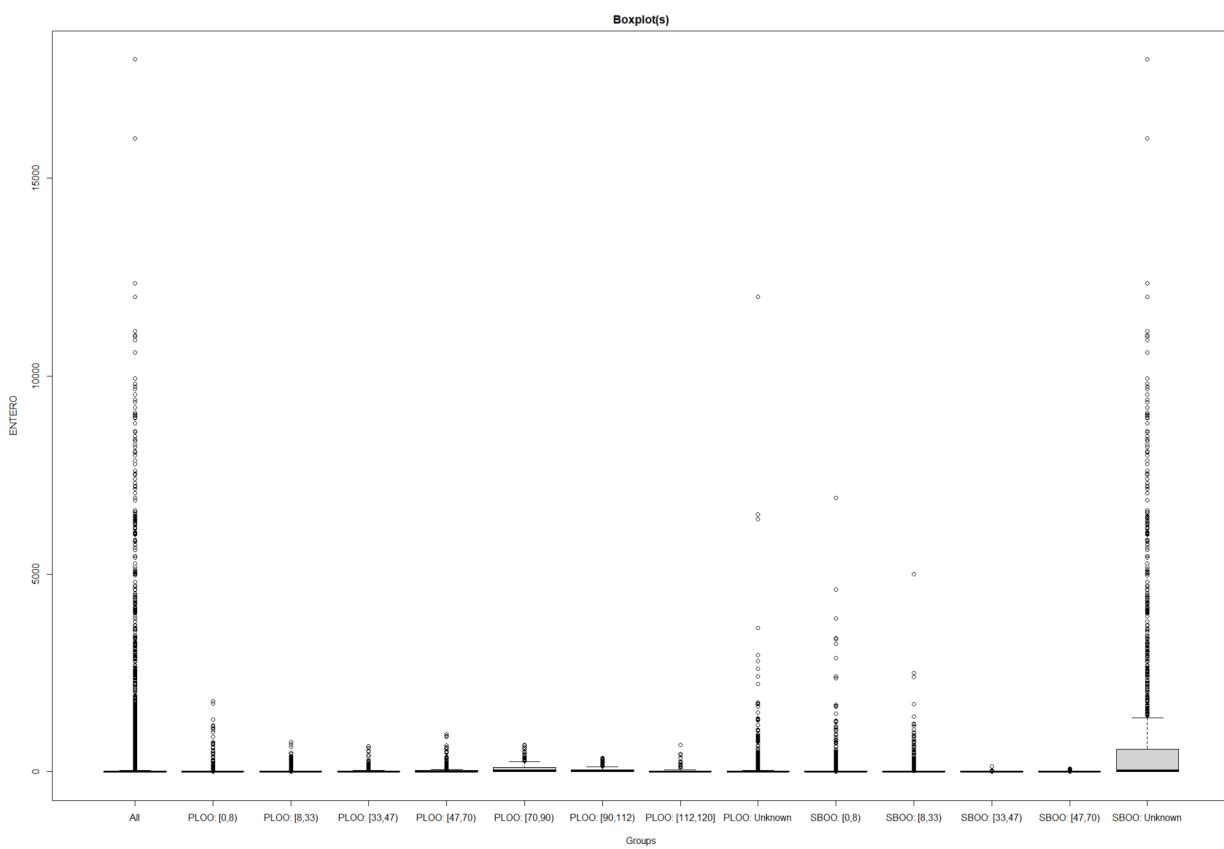
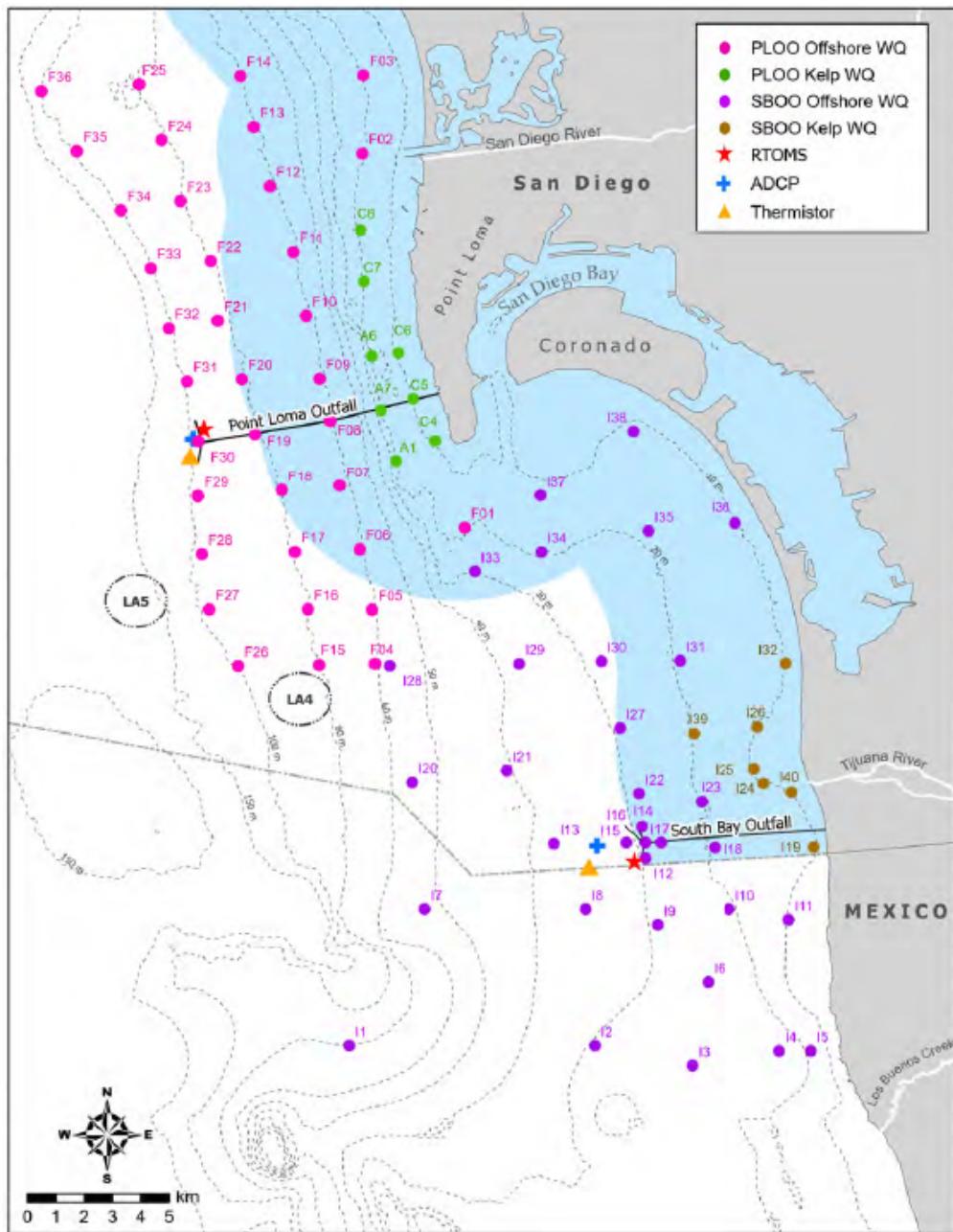
Figure 14*Boxplots of ENTERO By Project By Depth*

Figure 15

Oceanographic mooring and monitoring station locations around the PLOO and SBOO



From “2020-2021 Biennial Receiving Waters Monitoring and Assessment Report for the Point Loma and South Bay Ocean Outfalls,” by City of San Diego (2022, June 30).

https://www.sandiego.gov/sites/default/files/compressed_2020-2021_biennial_receiving_waters_monitoring_report_0.pdf

* Light blue shading represents State jurisdictional waters.

Appendix D - ADS506-01-FA22 - Final Project

Team 1

12/05/2022

RMarkdown global setup

```
knitr::opts_chunk$set(echo = TRUE)

knitr::opts_chunk$set(
  fig.align    = 'center', # how to align graphics in the final doc. 'left', 'right',
  ↪   'center'
  tidy         = TRUE,
  message      = FALSE,    # if FALSE knitr will not display any messages generated by
  ↪   code
  strip.white = TRUE,     # if FALSE knitr won't remove white spaces at beg/end of code
  ↪   chunk
  warning     = FALSE)    # if FALSE knitr will not display any warning messages in the
  ↪   final
```

Load packages

```
library(car)
library(caret)
library(class)
library(corrplot)
library(forecast)
library(gridExtra)
library(here)
library(imputeTS)
library(lubridate)
library(MLmetrics)
library(padr)
library(psych)
library(reshape2)
library(scales)
library(tidyverse)
library(timetk)
library(tseries)
library(zoo)

set.seed(1699)
```

Import and merge data sets

```
# Import 4 separate CSV files
curr_dir01 <- here("data/Ocean Water/water_quality_1990_1999_datasd.csv")
#curr_dir01
curr_dir02 <- here("data/Ocean Water/water_quality_2000_2010_datasd.csv")
#curr_dir02
curr_dir03 <- here("data/Ocean Water/water_quality_2011_2019_datasd.csv")
#curr_dir03
curr_dir04 <- here("data/Ocean Water/water_quality_2020_2021_datasd.csv")
#curr_dir04

owt_df01a <- read.csv(curr_dir01, header = TRUE, sep = ",")
owt_df01b <- read.csv(curr_dir02, header = TRUE, sep = ",")
owt_df01c <- read.csv(curr_dir03, header = TRUE, sep = ",")
owt_df01d <- read.csv(curr_dir04, header = TRUE, sep = ",")

# Merge 4 seperate dataframes into 1
owt_df01 <- rbind(owt_df01a, owt_df01b, owt_df01c, owt_df01d)

# DF Sort & Reindex Citation:
# https://gist.github.com/makexu93/05ef0f34306d3ab6c9ca8ddcd702cee1
owt_df01 <- owt_df01[order(owt_df01$date_sample, decreasing = FALSE), ]
row.names(owt_df01) <- NULL

print(head(owt_df01, 7))

##           sample station depth_m date_sample time project parameter qualifier
## 1 9011158743      C5      9 1990-11-15     PLOO CHLOROPHYLL
## 2 9011158743      C5      9 1990-11-15     PLOO      DENSITY
## 3 9011158743      C5      9 1990-11-15     PLOO       DO
## 4 9011158743      C5      9 1990-11-15     PLOO       PH
## 5 9011158743      C5      9 1990-11-15     PLOO SALINITY
## 6 9011158743      C5      9 1990-11-15     PLOO      TEMP
## 7 9011158743      C5      9 1990-11-15     PLOO      XMS

##           value units
## 1    0.870 ug/L
## 2   23.855 sigma-t
## 3    6.550 mg/L
## 4    8.080 pH
## 5  33.617 ppt
## 6   19.430 C
## 7   44.850 %

describe(owt_df01)

##        vars      n   mean      sd min   max range   se
## sample      1 1236769   NaN     NA Inf -Inf -Inf   NA
## station     2 1236769   NaN     NA Inf -Inf -Inf   NA
## depth_m     3 1152608 19.38  25.07   1   116   115 0.02
## date_sample 4 1236769   NaN     NA Inf -Inf -Inf   NA
## time        5 1236769   NaN     NA Inf -Inf -Inf   NA
## project     6 1236769   NaN     NA Inf -Inf -Inf   NA
## parameter   7 1236769   NaN     NA Inf -Inf -Inf   NA
```

```

## qualifier      8 1236769      NaN       NA Inf     -Inf     -Inf   NA
## value         9 1231466 124.24 1785.21 -37 1100000 1100037 1.61
## units        10 1236769      NaN       NA Inf     -Inf     -Inf   NA

```

Exploratory Data Analysis (EDA) and preprocessing, phase 1

Create custom function to generate boxplots and descriptive statistics for continuous variables

```

box_comp <- function(xcol = c(),
                      df = NA,
                      rtn_met = TRUE,
                      grph_title = "All",
                      box = TRUE) {
  # Define function to produce formatted boxplots & basic descriptive stats
  sig <- 3
  metrics_df01 <- data.frame(metric = c("",

                                         "Total N:",
                                         "Count",
                                         "NA Count",
                                         "Mean",
                                         "Median",
                                         "Standard Deviation",
                                         "Variance",
                                         "Range",
                                         "Min",
                                         "Max",
                                         "25th Percentile",
                                         "75th Percentile",
                                         "Subset w/o Outliers:",
                                         "Count",
                                         "%",
                                         "Outlier %",
                                         "NA Count",
                                         "Mean",
                                         "Median",
                                         "Standard Deviation",
                                         "Variance",
                                         "Range",
                                         "Min",
                                         "Max"
                                         )))

  for(var in xcol) {
    df_s1 <- df[, var]
    df_s1s1 <- data.frame(df_s1)
    df_s1_fit <- preProcess(df_s1s1,
                             method = c("center", "scale"))
    df_s1_trans <- predict(df_s1_fit, df_s1s1)

    # Calculate quartiles
    var_iqr_lim <- IQR(df_s1) * 1.5
    var_q1 <- quantile(df_s1, probs = c(.25))
    var_otlow <- var_q1 - var_iqr_lim
  }
}

```

```

var_q3 <- quantile(df_s1, probs = c(.75))
var_othigh <- var_q3 + var_iqr_lim

# Subset non-outlier data
var_non_otlr_df01 <- subset(df, (abs(df_s1_trans) <= 3))
df_s2 <- var_non_otlr_df01[, var]

# Begin calculating measures of centrality & dispersion
var_mean <- mean(df_s1, na.remove = TRUE)
var_non_otlr_df01_trunc_mean <- mean(df_s2, na.remove = TRUE)
var_med <- median(df_s1)
var_non_otlr_df01_trunc_med <- median(df_s2)
var_mode <- mode(df_s1)
var_non_otlr_df01_trunc_mode <- mode(df_s2)
var_stde <- sd(df_s1)
var_non_otlr_df01_trunc_stde <- sd(df_s2)
var_vari <- var(df_s1)
var_non_otlr_df01_trunc_vari <- var(df_s2)
var01_min <- min(df[, var])
var01_max <- max(df[, var])
var01_range <- var01_max - var01_min
var02_min <- min(var_non_otlr_df01[, var])
var02_max <- max(var_non_otlr_df01[, var])
var02_range <- var02_max - var02_min

# Configure y-axis min & max to sync graphs
plot_min <- min(var01_min, var02_min)
plot_max <- max(var01_max, var02_max)
nonoutlier_perc <- round((as.numeric(dim(var_non_otlr_df01)[1] /
→ as.numeric(dim(df)[1]))) * 100, 1)

# Fill in metrics table
measure_val01 <- c(paste0("Variable: ", var),
                     "",
                     as.character(dim(df)[1]),
                     sum(is.na(df_s1)),
                     round(var_mean, sig),
                     round(var_med, sig),
                     round(var_stde, sig),
                     round(var_vari, sig),
                     round(var01_range, sig),
                     round(var01_min, sig),
                     round(var01_max, sig),
                     round(var_q1, sig),
                     round(var_q3, sig),
                     "",
                     as.character(dim(var_non_otlr_df01)[1]),
                     paste0(nonoutlier_perc, "%"),
                     paste0(round(100 - nonoutlier_perc, 1), "%"),
                     sum(is.na(df_s2)),
                     round(var_non_otlr_df01_trunc_mean, sig),
                     round(var_non_otlr_df01_trunc_med, sig),
                     round(var_non_otlr_df01_trunc_stde, sig),

```

```

        round(var_non_otlr_df01_trunc_vari, sig),
        round(var02_range, sig),
        round(var02_min, sig),
        round(var02_max, sig)
    )

var_name <- paste0("Variable: ", var)
metrics_df01[, ncol(metrics_df01) + 1] <- measure_val01
}

# Format boxplot titles based on number of plots
if(box == TRUE) {
    if(length(xcol == 1)) {
        boxplot(df,
            ylab = "Parameter Values",
            main = paste0("Boxplot for ", xcol, " (", grph_title, ")"))
    }
    else {
        boxplot(df,
            ylab = "Parameter Values",
            main = paste0("Boxplot for Multiple Parameters", " (", grph_title, ")"))

    }
}
# Return & print metrics table(s)
if(rtn_met == TRUE) {
    print(metrics_df01)
    return(metrics_df01)
}
}

```

Factorize and format column types; print NA counts

```

# List of parameter values
param_lst01 <- c("CHLOROPHYLL",
                 "DENSITY",
                 "DO",
                 "ENTERO",
                 "FECAL",
                 "OG",
                 "PH",
                 "SALINITY",
                 "SUSO",
                 "TEMP",
                 "TOTAL",
                 "XMS")

# List of col names
col_lst01 <- c("sample",
               "station",
               "date_sample",
               "time",
               "project",

```

```

    "parameter",
    "qualifier",
    "units")

# Convert categorical features to factors
owt_df02 <- owt_df01
for(c in col_lst01) {
  owt_df02[, c] <- as.factor(owt_df02[, c])
}

# Generate NA Summary Tables Citation:
# https://www.geeksforgeeks.org/replace-character-value-with-na-in-r/
owt_df02[owt_df02 == ""] <- NA
print(head(owt_df02, 7))

##           sample station depth_m date_sample time project parameter qualifier
## 1 9011158743      C5      9 1990-11-15 <NA>     PLOO CHLOROPHYLL      <NA>
## 2 9011158743      C5      9 1990-11-15 <NA>     PLOO      DENSITY      <NA>
## 3 9011158743      C5      9 1990-11-15 <NA>     PLOO       DO      <NA>
## 4 9011158743      C5      9 1990-11-15 <NA>     PLOO       PH      <NA>
## 5 9011158743      C5      9 1990-11-15 <NA>     PLOO SALINITY      <NA>
## 6 9011158743      C5      9 1990-11-15 <NA>     PLOO      TEMP      <NA>
## 7 9011158743      C5      9 1990-11-15 <NA>     PLOO      XMS      <NA>

##           value units
## 1   0.870 ug/L
## 2 23.855 sigma-t
## 3   6.550 mg/L
## 4   8.080 pH
## 5 33.617 ppt
## 6 19.430 C
## 7 44.850 %

owt_df02_na <- sapply(owt_df02, function(x) sum(is.na(x)))
owt_df02_notna <- sapply(owt_df02, function(x) sum(!is.na(x)))
owt_df02_tbl01 <- rbind(owt_df02_notna, owt_df02_na)
owt_df02_tbl02 <- rbind(owt_df02_tbl01, round(prop.table(owt_df02_tbl01, margin = 2), 4))
print("All parameters")

## [1] "All parameters"

print(owt_df02_tbl02)

##           sample station depth_m date_sample      time project
## owt_df02_notna 1236769 1236769 1152608.000 1236769 1075929.00 1236769
## owt_df02_na      0        0     84161.000        0    160840.00      0
## owt_df02_notna      1        1      0.932        1      0.87      1
## owt_df02_na      0        0      0.068        0      0.13      0
##           parameter qualifier      value units
## owt_df02_notna 1236769 394867.0000 1231466.0000 1236769
## owt_df02_na          0 841902.0000 5303.0000      0
## owt_df02_notna      1      0.3193    0.9957      1
## owt_df02_na      0      0.6807    0.0043      0

owt_df02a <- owt_df02[which(is.na(owt_df02), arr.ind=TRUE), ]
#print(head(owt_df02a, 7))

```

```

for(p in param_lst01) {
  df = owt_df02[owt_df02$parameter == p, ]
  print(head(df[which(is.na(df$value)), arr.ind=TRUE], , 7))
  df_na <- sapply(df, function(x) sum(is.na(x)))
  df_notna <- sapply(df, function(x) sum(!is.na(x)))
  df_tbl01 <- rbind(df_notna, df_na)
  df_tbl02 <- rbind(df_tbl01, round(prop.table(df_tbl01, margin = 2), 4))
  rownames(df_tbl02) <- c("Not NA n", "NA n", "Not NA %", "NA %")
  print(p)
  print(df_tbl02) # This table w/ null counts and proportions for each feature not
  → displayed for space purposes
}

##           sample station depth_m date_sample time project parameter
## 272260 1229972635     B10      1.5 1997-12-29 <NA>    PL00 CHLOROPHYLL
## 272267 1229972638     B10     42.7 1997-12-29 <NA>    PL00 CHLOROPHYLL
## 272274 1229972633     B10     61.0 1997-12-29 <NA>    PL00 CHLOROPHYLL
## 272281 1229972634     B10     79.2 1997-12-29 <NA>    PL00 CHLOROPHYLL
## 272288 1229972637     B10     97.5 1997-12-29 <NA>    PL00 CHLOROPHYLL
## 272295 1229972636     B10    115.8 1997-12-29 <NA>    PL00 CHLOROPHYLL
## 272302 1229972643     B11      1.5 1997-12-29 <NA>    PL00 CHLOROPHYLL
##           qualifier value units
## 272260      <NA>   NA ug/L
## 272267      <NA>   NA ug/L
## 272274      <NA>   NA ug/L
## 272281      <NA>   NA ug/L
## 272288      <NA>   NA ug/L
## 272295      <NA>   NA ug/L
## 272302      <NA>   NA ug/L
## [1] "CHLOROPHYLL"
##           sample station depth_m date_sample      time project parameter
## Not NA n  88471    88471    88471      88471 74093.0000  88471    88471
## NA n      0        0        0          0 14378.0000    0        0
## Not NA %   1        1        1          1  0.8375      1        1
## NA %      0        0        0          0  0.1625      0        0
##           qualifier      value units
## Not NA n      0 87911.0000 88471
## NA n       88471  560.0000    0
## Not NA %      0    0.9937    1
## NA %       1    0.0063    0
##           sample station depth_m date_sample      time project parameter
## 429235 1018012132     E16      1.5 2001-10-18 8:32:00 PST    PL00 DENSITY
## 503878 513032821      C8       1.0 2003-05-13 10:30:00 PST    PL00 DENSITY
## 547108 601042705      C4       1.0 2004-06-01 11:37:00 PST    PL00 DENSITY
## 720423 702082492      A6       1.0 2008-07-02 9:42:00 PST    PL00 DENSITY
## 720433 702082490      A6      12.0 2008-07-02 9:42:00 PST    PL00 DENSITY
## 720443 702082491      A6      18.0 2008-07-02 9:42:00 PST    PL00 DENSITY
## 720453 702082495      A7       1.0 2008-07-02 9:05:00 PST    PL00 DENSITY
##           qualifier value units
## 429235      <NA>   NA sigma-t
## 503878      <NA>   NA sigma-t
## 547108      <NA>   NA sigma-t

```

```

## 720423      <NA>    NA sigma-t
## 720433      <NA>    NA sigma-t
## 720443      <NA>    NA sigma-t
## 720453      <NA>    NA sigma-t
## [1] "DENSITY"
##           sample station depth_m date_sample      time project parameter
## Not NA n   88317    88317    88317        88317 73999.0000 88317    88317
## NA n       0         0         0             0 14318.0000 0         0
## Not NA %    1         1         1             1 0.8379 1         1
## NA %       0         0         0             0 0.1621 0         0
##           qualifier      value units
## Not NA n     0 88256.0000 88317
## NA n       88317    61.0000 0
## Not NA %     0 0.9993 1
## NA %       1 0.0007 0
##           sample station depth_m date_sample      time project parameter
## 82185 CTD014940 A11    42.7 1993-05-06 10:45:00 PST PL00    DO
## 168703 1024942994 A1     1.5 1994-10-24 9:10:00 PST PL00    DO
## 168713 1024942995 A1     3.0 1994-10-24 9:10:00 PST PL00    DO
## 168723 1024942996 A1     6.1 1994-10-24 9:10:00 PST PL00    DO
## 168733 1024942997 A1    12.2 1994-10-24 9:10:00 PST PL00    DO
## 168734 1024942998 A1    12.2 1994-10-24 9:10:00 PST PL00    DO
## 168748 1024942999 A1    18.3 1994-10-24 9:10:00 PST PL00    DO
##           qualifier value units
## 82185      <NA>  NA mg/L
## 168703      <NA>  NA mg/L
## 168713      <NA>  NA mg/L
## 168723      <NA>  NA mg/L
## 168733      <NA>  NA mg/L
## 168734      <NA>  NA mg/L
## 168748      <NA>  NA mg/L
## [1] "DO"
##           sample station depth_m date_sample      time project parameter
## Not NA n 109542 109542 109542        109542 83392.0000 109542 109542
## NA n       0         0         0             0 26150.0000 0         0
## Not NA %    1         1         1             1 0.7613 1         1
## NA %       0         0         0             0 0.2387 0         0
##           qualifier      value units
## Not NA n     0 109437.000 109542
## NA n       109542    105.000 0
## Not NA %     0 0.999 1
## NA %       1 0.001 0
##           sample station depth_m date_sample      time project parameter
## 25057 M0001828 A2     1.0 1991-10-09 8:27:00 PST PL00    ENTERO
## 81650 SH000903 D5     NA 1993-05-04      <NA> PL00    ENTERO
## 103629 1208931380 A2     3.0 1993-12-08 8:20:00 PST PL00    ENTERO
## 103642 1208931381 A2     6.1 1993-12-08 8:20:00 PST PL00    ENTERO
## 103655 1208931382 A2    12.2 1993-12-08 8:20:00 PST PL00    ENTERO
## 103668 1208931383 A2    18.3 1993-12-08 8:20:00 PST PL00    ENTERO
## 103701 1208931385 A2    30.5 1993-12-08 8:20:00 PST PL00    ENTERO
##           qualifier value units
## 25057      e  NA CFU/100 mL
## 81650      <NA>  NA CFU/100 mL
## 103629      <NA>  NA CFU/100 mL

```

```

## 103642      <NA>    NA CFU/100 mL
## 103655      <NA>    NA CFU/100 mL
## 103668      <NA>    NA CFU/100 mL
## 103701      <NA>    NA CFU/100 mL
## [1] "ENTERO"
##           sample station   depth_m date_sample       time project parameter
## Not NA n 144341 144341 116186.0000 144341 144012.0000 144341 144341
## NA n      0      0 28155.0000      0 329.0000      0      0
## Not NA %  1      1 0.8049      1 0.9977      1      1
## NA %      0      0 0.1951      0 0.0023      0      0
##           qualifier     value  units
## Not NA n 136251.000 143075.0000 144341
## NA n      8090.000 1266.0000      0
## Not NA %  0.944   0.9912      1
## NA %      0.056   0.0088      0
##           sample station depth_m date_sample       time project parameter
## 39760     M0003378 B3      61.0 1992-03-04 9:23:00 PST  PLOO  FECAL
## 73538     M0002093 A4      79.2 1993-02-03 9:15:00 PST  PLOO  FECAL
## 81651     SH000903 D5      NA 1993-05-04 <NA>  PLOO  FECAL
## 82671     SH001257 D7      NA 1993-05-11 9:43:00 PST  PLOO  FECAL
## 103630    1208931380 A2      3.0 1993-12-08 8:20:00 PST  PLOO  FECAL
## 103643    1208931381 A2      6.1 1993-12-08 8:20:00 PST  PLOO  FECAL
## 103656    1208931382 A2     12.2 1993-12-08 8:20:00 PST  PLOO  FECAL
##           qualifier value  units
## 39760      <NA> CFU/100 mL
## 73538      <NA> CFU/100 mL
## 81651      <NA> CFU/100 mL
## 82671      <NA> CFU/100 mL
## 103630     <NA> CFU/100 mL
## 103643     <NA> CFU/100 mL
## 103656     <NA> CFU/100 mL
## [1] "FECAL"
##           sample station   depth_m date_sample       time project parameter
## Not NA n 137649 137649 109633.0000 137649 137322.0000 137649 137649
## NA n      0      0 28016.0000      0 327.0000      0      0
## Not NA %  1      1 0.7965      1 0.9976      1      1
## NA %      0      0 0.2035      0 0.0024      0      0
##           qualifier     value  units
## Not NA n 127994.0000 136405.000 137649
## NA n      9655.0000 1244.000      0
## Not NA %  0.9299  0.991      1
## NA %      0.0701  0.009      0
##           sample station depth_m date_sample       time project parameter
## 112401    112941961 A1      12.2 1994-01-12 8:36:00 PST  PLOO  OG
## 125289    309941665 A4      79.2 1994-03-09 10:06:00 PST  PLOO  OG
## 136304    503941704 B5      1.5 1994-05-03 10:24:00 PST  PLOO  OG
## 136346    503941705 B5     24.4 1994-05-03 10:24:00 PST  PLOO  OG
## 136396    503941706 B5     61.0 1994-05-03 10:24:00 PST  PLOO  OG
## 136772    504941833 A2     24.4 1994-05-04 8:02:00 PST  PLOO  OG
## 137186    504941844 A6      1.5 1994-05-04 9:17:00 PST  PLOO  OG
##           qualifier value  units
## 112401     <NA>  NA mg/L
## 125289     <NA>  NA mg/L
## 136304     <NA>  NA mg/L

```

```

## 136346      <NA>    NA  mg/L
## 136396      <NA>    NA  mg/L
## 136772      <NA>    NA  mg/L
## 137186      <NA>    NA  mg/L
## [1] "OG"
##           sample station depth_m date_sample      time project parameter
## Not NA n   7944     7944    7944        7944 7940.0000 7944     7944
## NA n       0         0       0          0 4.0000 0         0
## Not NA %    1         1       1          1 0.9995 1         1
## NA %       0         0       0          0 0.0005 0         0
##           qualifier value units
## Not NA n 7673.0000 7922.0000 7944
## NA n     271.0000 22.0000 0
## Not NA %  0.9659  0.9972 1
## NA %     0.0341  0.0028 0
##           sample station depth_m date_sample      time project parameter
## 168706 1024942994     A1     1.5 1994-10-24 9:10:00 PST  PLOO     PH
## 168716 1024942995     A1     3.0 1994-10-24 9:10:00 PST  PLOO     PH
## 168726 1024942996     A1     6.1 1994-10-24 9:10:00 PST  PLOO     PH
## 168737 1024942997     A1    12.2 1994-10-24 9:10:00 PST  PLOO     PH
## 168738 1024942998     A1    12.2 1994-10-24 9:10:00 PST  PLOO     PH
## 168751 1024942999     A1    18.3 1994-10-24 9:10:00 PST  PLOO     PH
## 332755 923992590      A1     1.5 1999-09-23 7:40:00 PST  PLOO     PH
##           qualifier value units
## 168706      <NA>    NA  pH
## 168716      <NA>    NA  pH
## 168726      <NA>    NA  pH
## 168737      <NA>    NA  pH
## 168738      <NA>    NA  pH
## 168751      <NA>    NA  pH
## 332755      <NA>    NA  pH
## [1] "PH"
##           sample station depth_m date_sample      time project parameter
## Not NA n 107818 107818 107818        107818 816668.0000 107818 107818
## NA n       0         0       0          0 26150.0000 0         0
## Not NA %    1         1       1          1 0.7575 1         1
## NA %       0         0       0          0 0.2425 0         0
##           qualifier value units
## Not NA n      0 107564.0000 107818
## NA n       107818 254.0000 0
## Not NA %      0     0.9976 1
## NA %       1     0.0024 0
##           sample station depth_m date_sample      time project parameter
## 82188 CTD014940     A11    42.7 1993-05-06 10:45:00 PST  PLOO SALINITY
## 168707 1024942994     A1     1.5 1994-10-24 9:10:00 PST  PLOO SALINITY
## 168717 1024942995     A1     3.0 1994-10-24 9:10:00 PST  PLOO SALINITY
## 168727 1024942996     A1     6.1 1994-10-24 9:10:00 PST  PLOO SALINITY
## 168739 1024942997     A1    12.2 1994-10-24 9:10:00 PST  PLOO SALINITY
## 168740 1024942998     A1    12.2 1994-10-24 9:10:00 PST  PLOO SALINITY
## 168752 1024942999     A1    18.3 1994-10-24 9:10:00 PST  PLOO SALINITY
##           qualifier value units
## 82188      <NA>    NA  ppt
## 168707      <NA>    NA  ppt
## 168717      <NA>    NA  ppt

```

```

## 168727      <NA>    NA   ppt
## 168739      <NA>    NA   ppt
## 168740      <NA>    NA   ppt
## 168752      <NA>    NA   ppt
## [1] "SALINITY"
##           sample station depth_m date_sample      time project parameter
## Not NA n 109492  109492  109492      109492 83351.0000 109492  109492
## NA n      0       0       0          0 26141.0000 0       0
## Not NA %   1       1       1          1 0.7613   1       1
## NA %      0       0       0          0 0.2387   0       0
##           qualifier value units
## Not NA n      0 109318.0000 109492
## NA n      109492 174.0000 0
## Not NA %      0 0.9984   1
## NA %      1 0.0016   0
##           sample station depth_m date_sample      time project parameter
## 182188 119951890     B1    61.0 1995-01-19 7:25:00 PST  PL00  SUSO
## 197840 413951394     B9    97.5 1995-04-13 11:32:00 PST  PL00  SUSO
## 236137 821961617     B9    1.5 1996-08-21 8:23:00 PST  PL00  SUSO
## 236146 821961618     B9    42.7 1996-08-21 8:23:00 PST  PL00  SUSO
## 236171 821961619     B9    97.5 1996-08-21 8:23:00 PST  PL00  SUSO
## 236208 821961620     E10   1.5 1996-08-21 10:58:00 PST  PL00  SUSO
## 236217 821961621     E10   42.7 1996-08-21 10:58:00 PST  PL00  SUSO
##           qualifier value units
## 182188      <NA>    NA mg/L
## 197840      <NA>    NA mg/L
## 236137      <NA>    NA mg/L
## 236146      <NA>    NA mg/L
## 236171      <NA>    NA mg/L
## 236208      <NA>    NA mg/L
## 236217      <NA>    NA mg/L
## [1] "SUSO"
##           sample station depth_m date_sample      time project parameter
## Not NA n 27543  27543  27543      27543 27538.0000 27543  27543
## NA n      0       0       0          0 5.0000  0       0
## Not NA %   1       1       1          1 0.9998  1       1
## NA %      0       0       0          0 0.0002  0       0
##           qualifier value units
## Not NA n 1290.0000 27515.000 27543
## NA n      26253.0000 28.000 0
## Not NA %  0.0468  0.999   1
## NA %      0.9532  0.001   0
##           sample station depth_m date_sample      time project parameter
## 81483 CTD013369     K4    18.3 1993-05-03 7:07:00 PST  PL00  TEMP
## 82189 CTD014940     A11   42.7 1993-05-06 10:45:00 PST  PL00  TEMP
## 82454 CTD013374     K4    18.3 1993-05-06 8:52:00 PST  PL00  TEMP
## 82871 CTD013384     K4    18.3 1993-05-12 7:36:00 PST  PL00  TEMP
## 83293 CTD008245     C6    1.5 1993-05-19 12:10:00 PST  PL00  TEMP
## 83297 CTD008246     C6    3.0 1993-05-19 12:10:00 PST  PL00  TEMP
## 83301 CTD008247     C6    6.1 1993-05-19 12:10:00 PST  PL00  TEMP
##           qualifier value units
## 81483      <NA>    NA   C
## 82189      <NA>    NA   C
## 82454      <NA>    NA   C

```

```

## 82871      <NA>    NA     C
## 83293      <NA>    NA     C
## 83297      <NA>    NA     C
## 83301      <NA>    NA     C
## [1] "TEMP"
##           sample station depth_m date_sample          time project parameter
## Not NA n 139066  139066  139066       139066 112709.0000 139066  139066
## NA n      0        0        0            0 26357.0000   0        0
## Not NA %   1        1        1            1 0.8105      1        1
## NA %      0        0        0            0 0.1895      0        0
##           qualifier      value  units
## Not NA n      0 139046.0000 139066
## NA n      139066  20.0000   0
## Not NA %      0 0.9999     1
## NA %      1 0.0001     0
##           sample station depth_m date_sample          time project parameter
## 81652     SH000903    D5     NA 1993-05-04      <NA>  PLOO  TOTAL
## 82672     SH001257    D7     NA 1993-05-11 9:43:00 PST  PLOO  TOTAL
## 101335    1124931032   C6     1.5 1993-11-24 8:23:00 PST  PLOO  TOTAL
## 103637    1208931380   A2     3.0 1993-12-08 8:20:00 PST  PLOO  TOTAL
## 103650    1208931381   A2     6.1 1993-12-08 8:20:00 PST  PLOO  TOTAL
## 103663    1208931382   A2    12.2 1993-12-08 8:20:00 PST  PLOO  TOTAL
## 103676    1208931383   A2    18.3 1993-12-08 8:20:00 PST  PLOO  TOTAL
##           qualifier value  units
## 81652      <NA>  NA CFU/100 mL
## 82672      <NA>  NA CFU/100 mL
## 101335    <NA>  NA CFU/100 mL
## 103637    <NA>  NA CFU/100 mL
## 103650    <NA>  NA CFU/100 mL
## 103663    <NA>  NA CFU/100 mL
## 103676    <NA>  NA CFU/100 mL
## [1] "TOTAL"
##           sample station      depth_m date_sample          time project parameter
## Not NA n 137584  137584 109594.0000       137584 137257.0000 137584  137584
## NA n      0        0 27990.0000            0 327.0000   0        0
## Not NA %   1        1 0.7966             1 0.9976      1        1
## NA %      0        0 0.2034             0 0.0024      0        0
##           qualifier      value  units
## Not NA n 121659.0000 136141.0000 137584
## NA n      15925.0000 1443.0000   0
## Not NA %   0.8843    0.9895     1
## NA %      0.1157    0.0105     0
##           sample station depth_m date_sample          time project parameter
## 81485    CTD013369    K4    18.3 1993-05-03 7:07:00 PST  PLOO  XMS
## 82191    CTD014940    A11   42.7 1993-05-06 10:45:00 PST  PLOO  XMS
## 82456    CTD013374    K4    18.3 1993-05-06 8:52:00 PST  PLOO  XMS
## 82873    CTD013384    K4    18.3 1993-05-12 7:36:00 PST  PLOO  XMS
## 83497    CTD013399    K4    18.3 1993-05-20 7:20:00 PST  PLOO  XMS
## 83564    CTD004913    A7     1.5 1993-05-24 8:31:00 PST  PLOO  XMS
## 83569    CTD004914    A7     3.0 1993-05-24 8:31:00 PST  PLOO  XMS
##           qualifier value  units
## 81485      <NA>  NA   %
## 82191      <NA>  NA   %
## 82456      <NA>  NA   %

```

```

## 82873      <NA>    NA    %
## 83497      <NA>    NA    %
## 83564      <NA>    NA    %
## 83569      <NA>    NA    %
## [1] "XMS"
##           sample station depth_m date_sample          time project parameter
## Not NA n 139002 139002 139002      139002 112648.0000 139002 139002
## NA n      0      0      0            0 26354.0000      0      0
## Not NA %   1      1      1            1 0.8104      1      1
## NA %      0      0      0            0 0.1896      0      0
##           qualifier      value units
## Not NA n      0 138876.0000 139002
## NA n      139002 126.0000      0
## Not NA %      0      0.9991      1
## NA %      1      0.0009      0

# Display Value Summaries Citation:
# Hurst, D. (n.d.). ADS 506 [Course materials].
feature_cats01 <- owt_df02 %>%
  mutate(cat = paste(parameter, qualifier, units)) %>%
  count(cat)
print(feature_cats01)

##           cat     n
## 1 CHLOROPHYLL NA ug/L 88471
## 2 DENSITY NA sigma-t 88317
## 3 DO NA mg/L 109542
## 4 ENTERO < CFU/100 mL 95848
## 5 ENTERO > CFU/100 mL 452
## 6 ENTERO e CFU/100 mL 39766
## 7 ENTERO LA CFU/100 mL 23
## 8 ENTERO NA CFU/100 mL 8090
## 9 ENTERO ND CFU/100 mL 1
## 10 ENTERO NR CFU/100 mL 1
## 11 ENTERO NS CFU/100 mL 160
## 12 FECAL < CFU/100 mL 85819
## 13 FECAL > CFU/100 mL 746
## 14 FECAL e CFU/100 mL 41255
## 15 FECAL LA CFU/100 mL 8
## 16 FECAL NA CFU/100 mL 9655
## 17 FECAL ND CFU/100 mL 1
## 18 FECAL NR CFU/100 mL 4
## 19 FECAL NS CFU/100 mL 161
## 20 OG < mg/L 7673
## 21 OG NA mg/L 271
## 22 PH NA pH 107818
## 23 SALINITY NA ppt 109492
## 24 SUSO < mg/L 1290
## 25 SUSO NA mg/L 26253
## 26 TEMP NA C 139066
## 27 TOTAL < CFU/100 mL 66655
## 28 TOTAL > CFU/100 mL 2844
## 29 TOTAL >= CFU/100 mL 3
## 30 TOTAL e CFU/100 mL 51924

```

```

## 31 TOTAL LA CFU/100 mL      14
## 32 TOTAL NA CFU/100 mL  15925
## 33 TOTAL ND CFU/100 mL     54
## 34 TOTAL NR CFU/100 mL      4
## 35 TOTAL NS CFU/100 mL    161
## 36 XMS NA % 139002

feature_cats02 <- owt_df02 %>%
  mutate(cat = paste(date_sample, time)) %>%
  count(cat)
print(head(feature_cats02,7))

```

```

##                   cat n
## 1 1990-11-15 NA 14
## 2 1991-01-02 7:00:00 PST 25
## 3 1991-01-02 7:18:00 PST 25
## 4 1991-01-02 7:40:00 PST 15
## 5 1991-01-02 7:55:00 PST 15
## 6 1991-01-02 8:10:00 PST 25
## 7 1991-01-02 8:30:00 PST 15

```

Bin depth_m variable

```

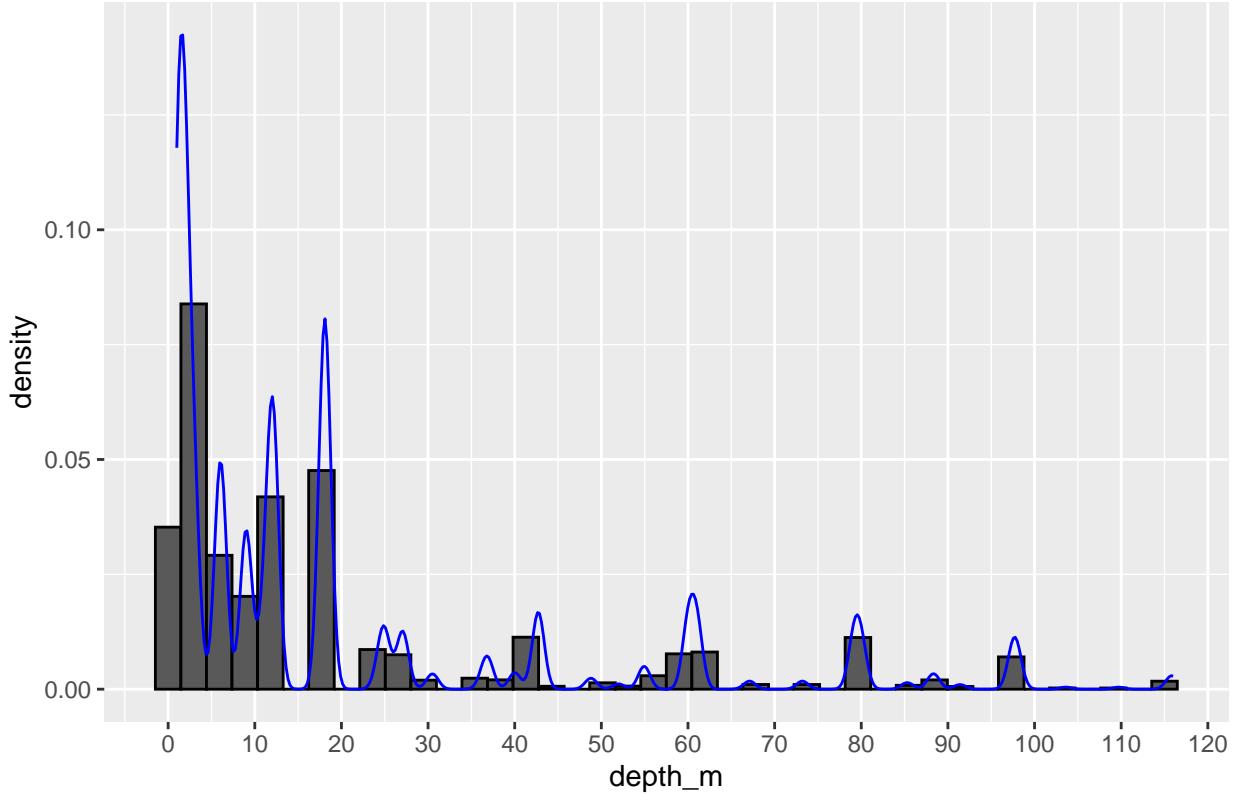
owt_df02$date_sample <- as.Date(owt_df02$date_sample, "%Y-%m-%d")

# Create bins for depth_m values
depth_lvls01 <- c("[0,8)",
                  "[8,33)",
                  "[33,47)",
                  "[47,70)",
                  "[70,90)",
                  "[90,112)",
                  "[112,120]",
                  "Unknown")

# Plot distribution of depth_m values
# Plot Formatting Citation:
#
→ https://community.rstudio.com/t/ggplot-x-axis-y-axis-ticks-labels-breaks-and-limits/119123/2
ggplot(owt_df02, aes(x = depth_m)) +
  geom_histogram(color = "black", bins = 40, aes(y = stat(density))) +
  geom_density(col = "blue") +
  labs(title = "Histogram of `depth_m` Feature") +
  scale_x_continuous(breaks=seq(0,120,10)) +
  theme(plot.title = element_text(hjust = 0.5, size = 12))

```

Histogram of `depth_m` Feature



```
# Create new column with bins
# Add Columns Using mutate() Citation:
#
# https://www.marsja.se/r-add-column-to-dataframe-based-on-other-columns-conditions-dplyr/
owt_df02 <- mutate(owt_df02, depth_m_bin = case_when(depth_m < 8 ~ "[0,8)",
                                                 depth_m < 33 ~ "[8,33)",
                                                 depth_m < 47 ~ "[33,47)",
                                                 depth_m < 70 ~ "[47,70)",
                                                 depth_m < 90 ~ "[70,90)",
                                                 depth_m < 112 ~ "[90,112)",
                                                 depth_m >= 112 ~ "[112,120]"))

# Replace NAs with "Unknown"
# Citation: https://statisticsglobe.com/r-replace-na-with-0/
owt_df02$depth_m_bin <- replace_na(owt_df02$depth_m_bin, "Unknown")

# Sort Bars Citation:
# https://www.statology.org/order-bars-ggplot2-bar-chart/
owt_df02$depth_m_bin_factr = factor(owt_df02$depth_m_bin, levels = depth_lvls01)
print(head(owt_df02, 7))
```

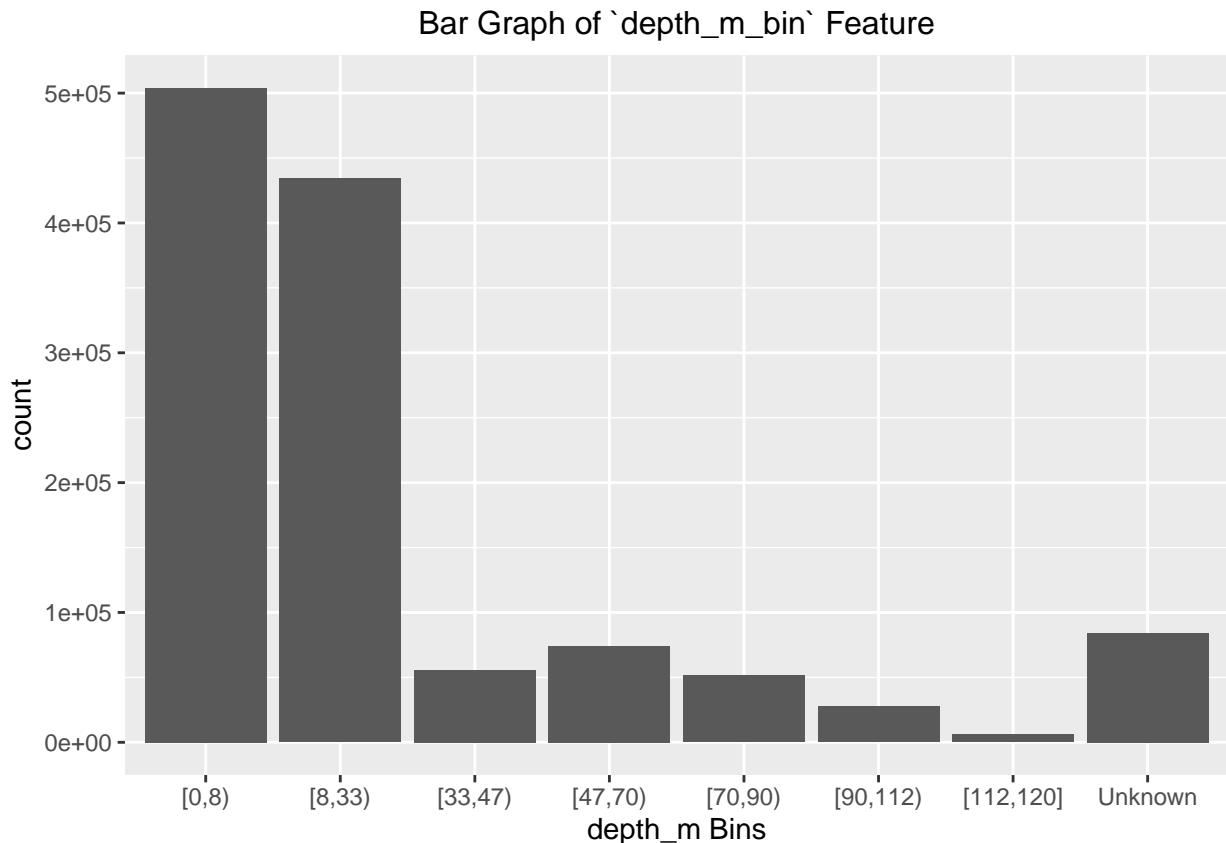
```
##      sample station depth_m date_sample time project parameter qualifier
## 1 9011158743      C5      9 1990-11-15 <NA>     PLOO CHLOROPHYLL      <NA>
## 2 9011158743      C5      9 1990-11-15 <NA>     PLOO    DENSITY      <NA>
## 3 9011158743      C5      9 1990-11-15 <NA>     PLOO        DO      <NA>
## 4 9011158743      C5      9 1990-11-15 <NA>     PLOO        PH      <NA>
```

```

## 5 9011158743      C5      9 1990-11-15 <NA>    PLOO    SALINITY    <NA>
## 6 9011158743      C5      9 1990-11-15 <NA>    PLOO    TEMP       <NA>
## 7 9011158743      C5      9 1990-11-15 <NA>    PLOO    XMS       <NA>
##   value  units depth_m_bin depth_m_bin_factr
## 1 0.870 ug/L   [8,33)      [8,33)
## 2 23.855 sigma-t [8,33)      [8,33)
## 3 6.550 mg/L   [8,33)      [8,33)
## 4 8.080 pH     [8,33)      [8,33)
## 5 33.617 ppt    [8,33)      [8,33)
## 6 19.430 C      [8,33)      [8,33)
## 7 44.850 %      [8,33)      [8,33)

# Display transformed bar chart
ggplot(owt_df02, aes(x = depth_m_bin_factr)) +
  geom_bar() +
  labs(title = "Bar Graph of `depth_m_bin` Feature") +
  xlab("depth_m Bins") +
  theme(plot.title = element_text(hjust = 0.5, size = 12))

```



Perform several aggregations on the data for performing EDA at multiple levels

```

# Display aggregations by different features
owt_df02_gb01 <- owt_df02 %>%
  group_by(station) %>%
  summarise(Count = n())

```

```

print(owt_df02_gb01[owt_df02_gb01$Count == min(owt_df02_gb01$Count), ])

## # A tibble: 2 x 2
##   station Count
##   <fct>    <int>
## 1 A15        30
## 2 A16        30

print(owt_df02_gb01[owt_df02_gb01$Count == max(owt_df02_gb01$Count), ])

## # A tibble: 1 x 2
##   station Count
##   <fct>    <int>
## 1 A1       55217

owt_df02_gb02 <- owt_df02 %>%
  group_by(project) %>%
  summarise(Count = n())

owt_df02_gb03 <- owt_df02 %>%
  group_by(date_sample) %>%
  summarise(Count = n())

# Main DF1
owt_df02_gb04 <- owt_df02 %>%
  group_by(date_sample, parameter) %>%
  summarise(Avg = mean(value, na.remove = TRUE))

owt_df02_gb05 <- owt_df02 %>%
  group_by(parameter) %>%
  summarise(Count = n())

owt_df02_gb06 <- owt_df02 %>%
  group_by(depth_m) %>%
  summarise(Count = n())

# Main DF3
owt_df02_gb07 <- owt_df02 %>%
  group_by(date_sample, project, depth_m_bin, parameter) %>%
  summarise(Avg = mean(value, na.remove = TRUE))

owt_df02_gb08 <- owt_df02 %>%
  group_by(depth_m_bin) %>%
  summarise(Count = n())

# Main DF2
owt_df02_gb09 <- owt_df02 %>%
  group_by(date_sample, project, parameter) %>%
  summarise(Avg = mean(value, na.remove = TRUE))

print(owt_df02_gb01)

## # A tibble: 157 x 2
##   station Count
##   <fct>    <int>
## 1 A15        30
## 2 A16        30
## 3 A17        30
## 4 A18        30
## 5 A19        30
## 6 A20        30
## 7 A21        30
## 8 A22        30
## 9 A23        30
## 10 A24        30
## 11 A25        30
## 12 A26        30
## 13 A27        30
## 14 A28        30
## 15 A29        30
## 16 A30        30
## 17 A31        30
## 18 A32        30
## 19 A33        30
## 20 A34        30
## 21 A35        30
## 22 A36        30
## 23 A37        30
## 24 A38        30
## 25 A39        30
## 26 A40        30
## 27 A41        30
## 28 A42        30
## 29 A43        30
## 30 A44        30
## 31 A45        30
## 32 A46        30
## 33 A47        30
## 34 A48        30
## 35 A49        30
## 36 A50        30
## 37 A51        30
## 38 A52        30
## 39 A53        30
## 40 A54        30
## 41 A55        30
## 42 A56        30
## 43 A57        30
## 44 A58        30
## 45 A59        30
## 46 A60        30
## 47 A61        30
## 48 A62        30
## 49 A63        30
## 50 A64        30
## 51 A65        30
## 52 A66        30
## 53 A67        30
## 54 A68        30
## 55 A69        30
## 56 A70        30
## 57 A71        30
## 58 A72        30
## 59 A73        30
## 60 A74        30
## 61 A75        30
## 62 A76        30
## 63 A77        30
## 64 A78        30
## 65 A79        30
## 66 A80        30
## 67 A81        30
## 68 A82        30
## 69 A83        30
## 70 A84        30
## 71 A85        30
## 72 A86        30
## 73 A87        30
## 74 A88        30
## 75 A89        30
## 76 A90        30
## 77 A91        30
## 78 A92        30
## 79 A93        30
## 80 A94        30
## 81 A95        30
## 82 A96        30
## 83 A97        30
## 84 A98        30
## 85 A99        30
## 86 A100       30
## 87 A101       30
## 88 A102       30
## 89 A103       30
## 90 A104       30
## 91 A105       30
## 92 A106       30
## 93 A107       30
## 94 A108       30
## 95 A109       30
## 96 A110       30
## 97 A111       30
## 98 A112       30
## 99 A113       30
## 100 A114       30
## 101 A115       30
## 102 A116       30
## 103 A117       30
## 104 A118       30
## 105 A119       30
## 106 A120       30
## 107 A121       30
## 108 A122       30
## 109 A123       30
## 110 A124       30
## 111 A125       30
## 112 A126       30
## 113 A127       30
## 114 A128       30
## 115 A129       30
## 116 A130       30
## 117 A131       30
## 118 A132       30
## 119 A133       30
## 120 A134       30
## 121 A135       30
## 122 A136       30
## 123 A137       30
## 124 A138       30
## 125 A139       30
## 126 A140       30
## 127 A141       30
## 128 A142       30
## 129 A143       30
## 130 A144       30
## 131 A145       30
## 132 A146       30
## 133 A147       30
## 134 A148       30
## 135 A149       30
## 136 A150       30
## 137 A151       30
## 138 A152       30
## 139 A153       30
## 140 A154       30
## 141 A155       30
## 142 A156       30
## 143 A157       30

```

```

##      <fct>   <int>
## 1 A1      55217
## 2 A10     5852
## 3 A11     6602
## 4 A12     5812
## 5 A13     6599
## 6 A14     5886
## 7 A15      30
## 8 A16      30
## 9 A17    2961
## 10 A2     7572
## # ... with 147 more rows
print(dim(owt_df02_gb01))

```

```
## [1] 157 2
```

```
print(owt_df02_gb02)
```

```

## # A tibble: 2 x 2
##   project  Count
##   <fct>    <int>
## 1 PL00     805915
## 2 SB00     430854
print(dim(owt_df02_gb02))

```

```
## [1] 2 2
```

```
print(head(owt_df02_gb03, 7))
```

```

## # A tibble: 7 x 2
##   date_sample  Count
##   <date>        <int>
## 1 1990-11-15     14
## 2 1991-01-02    195
## 3 1991-01-03    195
## 4 1991-01-07    190
## 5 1991-01-08    181
## 6 1991-01-09    577
## 7 1991-01-10    200
print(dim(owt_df02_gb03))

```

```
## [1] 5580 2
```

```
print(head(owt_df02_gb04, 7))
```

```

## # A tibble: 7 x 3
## # Groups:   date_sample [1]
##   date_sample parameter     Avg
##   <date>      <fct>     <dbl>
## 1 1990-11-15 CHLOROPHYLL  1.07
## 2 1990-11-15 DENSITY     23.9
## 3 1990-11-15 DO          6.98
## 4 1990-11-15 PH          8.13
## 5 1990-11-15 SALINITY    33.6

```

```

## 6 1990-11-15 TEMP      19.4
## 7 1990-11-15 XMS      60.2
print(dim(owt_df02_gb04))

## [1] 36811     3
print(owt_df02_gb05)

## # A tibble: 12 x 2
##   parameter  Count
##   <fct>     <int>
## 1 CHLOROPHYLL 88471
## 2 DENSITY     88317
## 3 DO          109542
## 4 ENTERO      144341
## 5 FECAL       137649
## 6 OG          7944
## 7 PH          107818
## 8 SALINITY    109492
## 9 SUSO         27543
## 10 TEMP        139066
## 11 TOTAL       137584
## 12 XMS         139002
print(dim(owt_df02_gb05))

## [1] 12  2
print(owt_df02_gb06)

## # A tibble: 51 x 2
##   depth_m  Count
##   <dbl>   <int>
## 1 1       119858
## 2 1.5     93937
## 3 2       120007
## 4 3       71058
## 5 6       55762
## 6 6.1     43265
## 7 9       65616
## 8 9.1    3052
## 9 11      23069
## 10 12     80308
## # ... with 41 more rows
print(dim(owt_df02_gb06))

## [1] 51  2
print(head(owt_df02_gb07, 7))

## # A tibble: 7 x 5
## # Groups:   date_sample, project, depth_m_bin [1]
##   date_sample project depth_m_bin parameter      Avg
##   <date>      <fct>    <chr>      <fct>     <dbl>
## 1 1990-11-15 PL00     [8,33]    CHLOROPHYLL  0.87

```

```

## 2 1990-11-15 PL00 [8,33) DENSITY 23.9
## 3 1990-11-15 PL00 [8,33) DO 6.55
## 4 1990-11-15 PL00 [8,33) PH 8.08
## 5 1990-11-15 PL00 [8,33) SALINITY 33.6
## 6 1990-11-15 PL00 [8,33) TEMP 19.4
## 7 1990-11-15 PL00 [8,33) XMS 44.8

print(dim(owt_df02_gb07))

## [1] 121286      5

print(owt_df02_gb08)

## # A tibble: 8 x 2
##   depth_m_bin  Count
##   <chr>        <int>
## 1 [0,8)        503887
## 2 [112,120]    5908
## 3 [33,47)      55629
## 4 [47,70)      74036
## 5 [70,90)      51358
## 6 [8,33)       434169
## 7 [90,112)     27621
## 8 Unknown      84161

print(dim(owt_df02_gb08))

## [1] 8 2

print(head(owt_df02_gb09, 7))

## # A tibble: 7 x 4
## # Groups:   date_sample, project [1]
##   date_sample project parameter     Avg
##   <date>      <fct>   <fct>      <dbl>
## 1 1990-11-15 PL00   CHLOROPHYLL  0.87
## 2 1990-11-15 PL00   DENSITY     23.9
## 3 1990-11-15 PL00   DO          6.55
## 4 1990-11-15 PL00   PH          8.08
## 5 1990-11-15 PL00   SALINITY    33.6
## 6 1990-11-15 PL00   TEMP         19.4
## 7 1990-11-15 PL00   XMS         44.8

print(dim(owt_df02_gb09))

## [1] 48395      4

```

Create custom function to perform df mutation/casting and boxplot display

```

ts_eda <- function(ts_df = NA,
                    form_lead = c(),
                    form_cast = c(),
                    param_lst = c(),
                    l1_col = c(),
                    l1_param = c(),
                    l2_col = c(),

```

```

l2_param = c(),
rtn_met = TRUE,
grph_title = "All",
box = TRUE,
week_agg = FALSE,
out_rm = FALSE) {

# Define function to dcast, print boxplot, and print desc stats for each parameter
dcast_form <- as.formula(paste(form_lead, collapse = "+"),
                           form_cast, sep = "~"))

#print(dcast_form)
#print(head(ts_df))

counter <- 1
# Loop to dcast parameters to cols
for(p in param_lst) {
  param_df <- ts_df[ts_df[, form_cast] == p, ]
  param_df <- param_df %>%
    drop_na()
  #print(param_df)

  #

  # If keyword set to TRUE, convert daily TS data to weekly
  if(week_agg == TRUE) {
    param_df02a <- param_df
    param_df02a$week <- isoweek(param_df02a$date_sample)

    # Date Handling Citation:
    # https://lubridate.tidyverse.org/reference/round_date.html
    param_df02a$week_end <- ceiling_date(param_df02a$date_sample,
                                             unit = "week",
                                             week_start = getOption("lubridate.week.start",
                                         ↳ 1))

    #print(head(param_df02a, 7))
    param_df$date_sample <- param_df02a$week_end
  }

  # Casting Citation:
  # https://www.datasciencemadesimple.com/melting-casting-r/
  param_df_cast <- dcast(param_df, dcast_form, mean)
  #print(head(param_df_cast, 7))

  #

  # If keyword set to TRUE, remove outliers
  if(out_rm == TRUE) {
    #print(head(ts_df_mrgd01c, 7))
    #print(dim(ts_df_mrgd01c))
    ts_df_mrgd01c_s1 <- subset(x = param_df_cast, select = p)
    ts_df_mrgd01c_s1_fit <- preProcess(ts_df_mrgd01c_s1,
                                         method = c("center", "scale"))
    ts_df_mrgd01c_s1_trans <- predict(ts_df_mrgd01c_s1_fit,
                                         ts_df_mrgd01c_s1)

    # Subset non-outlier data
  }
}

```

```

print(dim(param_df_cast))
param_df_cast <- subset(param_df_cast,
                        (abs(ts_df_mrgd01c_s1_trans) <= 3))
dfs1b <- param_df_cast[, p]
#print(head(param_df_cast, 7))
}

# For first item in vector, no merge is needed
if(counter == 1) {
  ts_df_mrgd <- param_df_cast
}
# For 2-n items in vector, merge dataframes iteratively
else {
  # Merge individual casted parameter df's into 1 df
  # DF Merge Citation:
  # https://www.geeksforgeeks.org/joining-of-dataframes-in-r-programming/
  ts_df_mrgd <- merge(x = ts_df_mrgd,
                        y = param_df_cast,
                        by = form_lead,
                        all = TRUE)
}

# Run custom function to ID outliers and generate boxplot
ts_df_mrgd01a <- na.omit(param_df_cast)
#print(head(ts_df_mrgd01a, 7))
#print(dim(ts_df_mrgd01a))
dfs1 <- subset(x = ts_df_mrgd01a, select = p)
dfs1a <- dfs1
#print(head(dfs1a, 7))

dfs1a$Groups <- as.factor(grph_title)
box_comp(xcol = p,
          df = dfs1,
          rtn_met = rtn_met,
          box = box)
#print(dfs_lst)

#
# -----
# Create subplots based on additional agg layers
#print(paste0("L2 length = ", length(l2_param)))
if(length(l2_param) == 0) {
  for(l1 in l1_param) {
    ts_df_mrgd_chl_dmb1 <- ts_df_mrgd01a[(ts_df_mrgd01a[, l1_col] == l1), ]

    # Run custom function to ID outliers and generate boxplot
    #print(head(ts_df_mrgd_chl_dmb1))
    # Error Handling Citation:
    # https://www.geeksforgeeks.org/handling-errors-in-r-programming/
    dfs2 <- subset(x = ts_df_mrgd_chl_dmb1, select = p)
    dfs2a <- dfs2
    try(dfs2a$Groups <- as.factor(l1))
    try(box_comp(xcol = p,
                 df = dfs2,

```

```

        rtn_met = rtn_met,
        grph_title = l1,
        box = box),
    silent = TRUE)
  dfs1a <- rbind(dfs1a, dfs2a)
}
}
else {
  for(l1 in l1_param) {
    for(l2 in l2_param) {
      ts_df_mrgd_chl_dmb1 <- ts_df_mrgd01a[(ts_df_mrgd01a[, l1_col] == l1) &
→ (ts_df_mrgd01a[, l2_col] == l2), ]

      # Run custom function to ID outliers and generate boxplot
      #print(head(ts_df_mrgd_chl_dmb1))
      dfs3 <- subset(x = ts_df_mrgd_chl_dmb1, select = p)
      dfs3a <- dfs3

      try(dfs3a$Groups <- as.factor(paste0(l1, ":", l2)))
      try(box_comp(xcol = p,
                    df = dfs3,
                    rtn_met = rtn_met,
                    grph_title = paste0(l1, ":", l2),
                    box = box),
          silent = TRUE)
      dfs1a <- rbind(dfs1a, dfs3a)
    }
  }
}

#-----
#print(dim(dfs1a))
#print(head(dfs1a, 7))
bx_form <- as.formula(paste(p, "Groups", sep = "~"))
#print(bx_form)
#print(head(dfs1a, 7))

boxplot(bx_form,
        dfs1a,
        main = "Boxplot(s)")

counter <- counter + 1
}
print(head(ts_df_mrgd, 7))
print(describe(ts_df_mrgd))

# Display summary of NA count
ts_df_na <- sapply(ts_df, function(x) sum(is.na(x)))
ts_df_notna <- sapply(ts_df, function(x) sum(!is.na(x)))
ts_df_tbl01 <- rbind(ts_df_notna, ts_df_na)
ts_df_tbl02 <- rbind(ts_df_tbl01, round(prop.table(ts_df_tbl01, margin = 2), 4))
rownames(ts_df_tbl02) <- c("Not NA n", "NA n", "Not NA %", "NA %")

```

```

print(ts_df_tbl02) # This table w/ null counts and proportions for each feature not
  ↵ displayed for space purposes
return(ts_df_mrgd)
}

```

Run custom function on data aggregated by date_sample and parameter

```

param_lst02 <- c("CHLOROPHYLL",
                 "DENSITY",
                 "DO",
                 "ENTERO",
                 "FECAL",
                 "OG",
                 "PH",
                 "SALINITY",
                 "SUSO",
                 "TEMP",
                 "XMS")

owt_df02_gb04_mrgd = ts_eda(ts_df = owt_df02_gb04,
                             form_lead = "date_sample",
                             form_cast = "parameter",
                             param_lst = param_lst02,
                             rtn_met = TRUE,
                             box = FALSE,
                             week_agg = FALSE
                             )

```

```

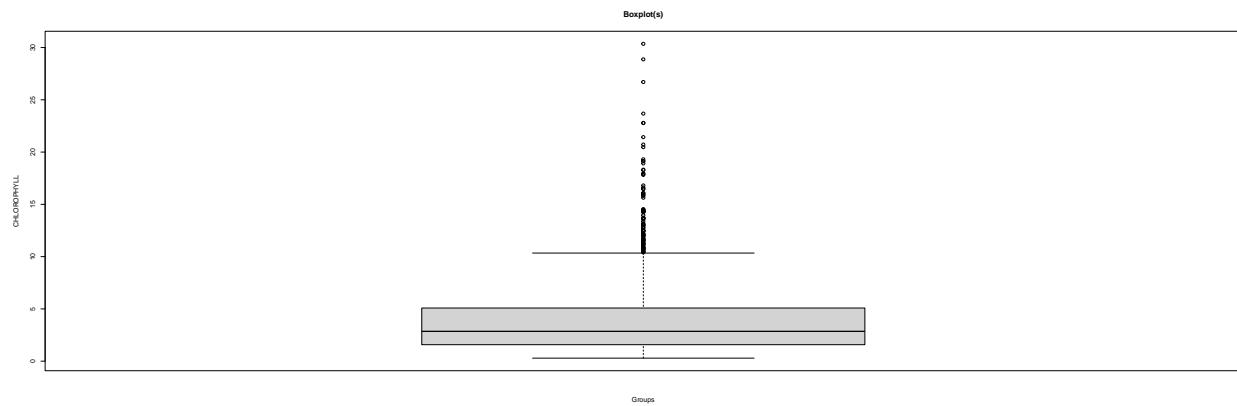
##               metric          V2
## 1             Variable: CHLOROPHYLL
## 2       Total N:
## 3           Count      2092
## 4        NA Count       0
## 5           Mean      3.91
## 6           Median     2.853
## 7 Standard Deviation    3.411
## 8           Variance    11.633
## 9           Range      30.064
## 10          Min       0.29
## 11          Max      30.353
## 12 25th Percentile     1.576
## 13 75th Percentile     5.083
## 14 Subset w/o Outliers:
## 15           Count      2050
## 16             %      98%
## 17        Outlier %      2%
## 18        NA Count       0
## 19           Mean      3.626
## 20           Median     2.791
## 21 Standard Deviation    2.74
## 22           Variance    7.507
## 23           Range      13.676
## 24           Min       0.29

```

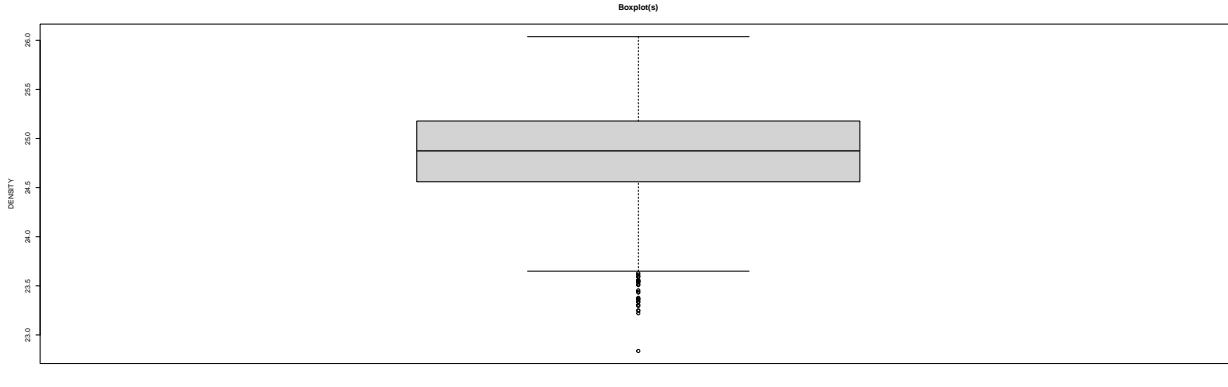
```
## 25
```

```
Max
```

```
13.965
```



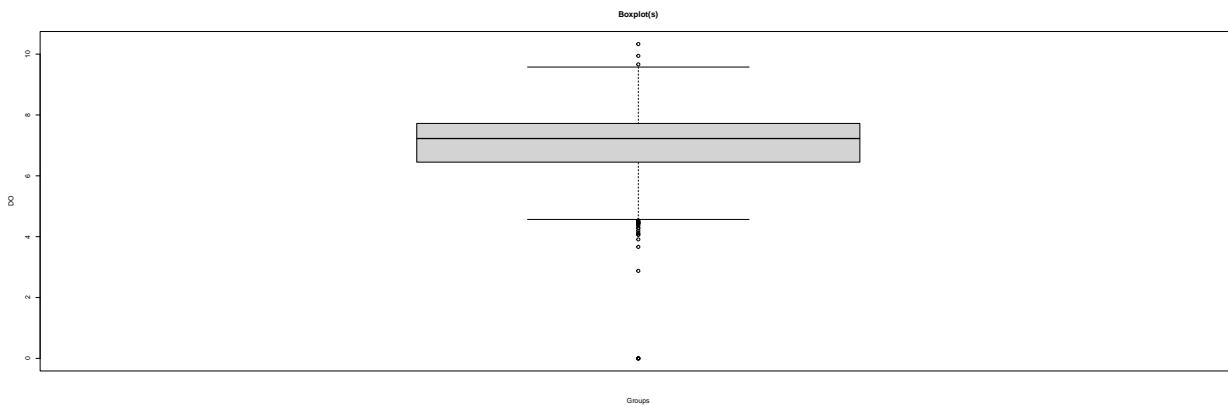
```
##          metric      V2
## 1        Variable: DENSITY
## 2        Total N:
## 3            Count     2246
## 4            NA Count    0
## 5            Mean      24.85
## 6            Median     24.874
## 7  Standard Deviation   0.466
## 8            Variance    0.217
## 9            Range       3.2
## 10           Min      22.837
## 11           Max      26.037
## 12      25th Percentile   24.56
## 13      75th Percentile   25.178
## 14 Subset w/o Outliers:
## 15            Count     2232
## 16            %      99.4%
## 17            Outlier %   0.6%
## 18            NA Count    0
## 19            Mean      24.86
## 20            Median     24.877
## 21  Standard Deviation   0.451
## 22            Variance    0.203
## 23            Range       2.583
## 24           Min      23.454
## 25           Max      26.037
```



```

##           metric      V2
## 1          Variable: D0
## 2          Total N:
## 3          Count     2504
## 4          NA Count    0
## 5          Mean      7.041
## 6          Median     7.226
## 7          Standard Deviation   0.99
## 8          Variance     0.98
## 9          Range      10.329
## 10         Min        0
## 11         Max      10.329
## 12         25th Percentile 6.451
## 13         75th Percentile 7.722
## 14 Subset w/o Outliers:
## 15         Count     2494
## 16         %       99.6%
## 17         Outlier % 0.4%
## 18         NA Count    0
## 19         Mean      7.057
## 20         Median     7.232
## 21         Standard Deviation   0.937
## 22         Variance     0.877
## 23         Range      5.822
## 24         Min        4.12
## 25         Max      9.942

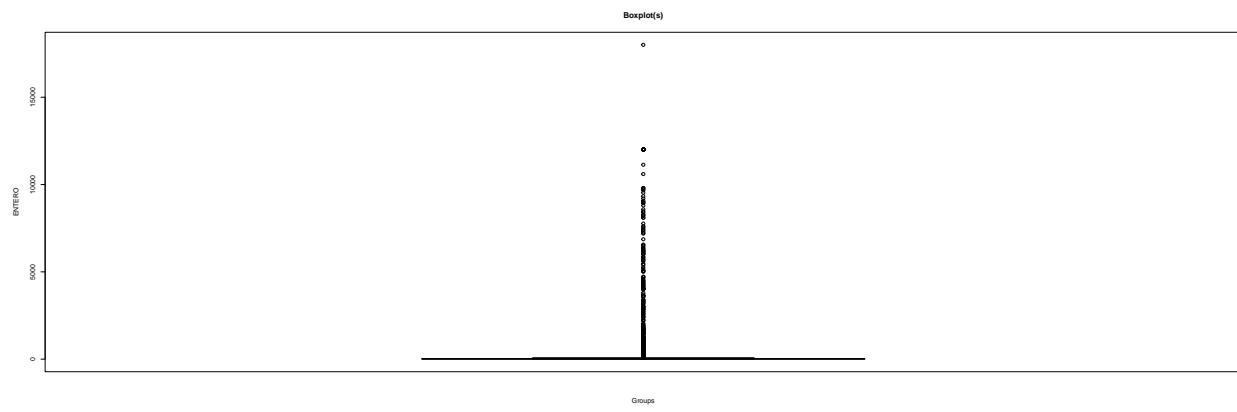
```



```

##               metric      V2
## 1          Variable: ENTERO
## 2          Total N:
## 3             Count      5197
## 4        NA Count       0
## 5            Mean     284.213
## 6            Median     7.333
## 7 Standard Deviation 1325.881
## 8            Variance 1757960.571
## 9            Range    18000
## 10           Min       0
## 11           Max    18000
## 12 25th Percentile     2.75
## 13 75th Percentile    35.455
## 14 Subset w/o Outliers:
## 15           Count      5090
## 16           %      97.9%
## 17        Outlier %     2.1%
## 18        NA Count       0
## 19            Mean    112.006
## 20            Median      7
## 21 Standard Deviation 417.536
## 22            Variance 174336.299
## 23            Range   4233.333
## 24           Min       0
## 25           Max    4233.333

```



```

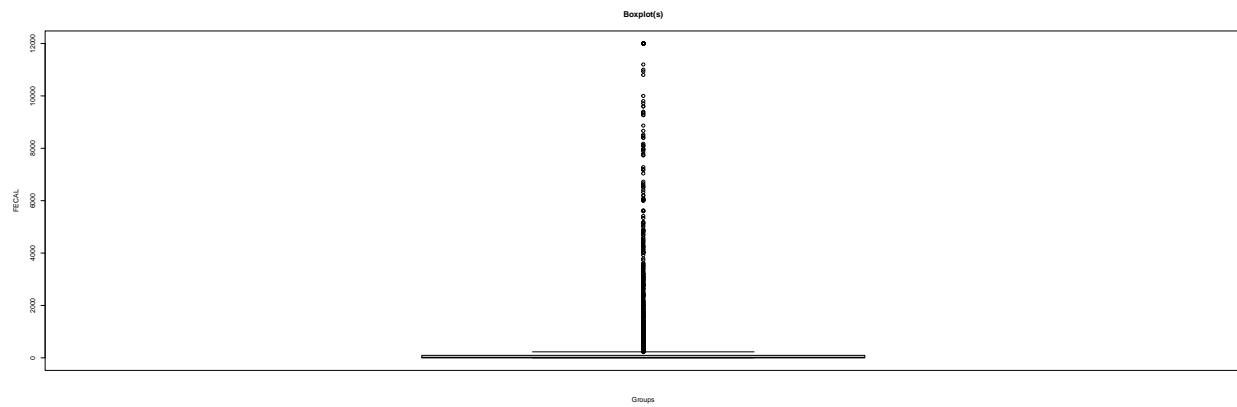
##               metric      V2
## 1          Variable: FECAL
## 2          Total N:
## 3             Count      5078
## 4        NA Count       0
## 5            Mean    409.934
## 6            Median    10.564
## 7 Standard Deviation 1513.552
## 8            Variance 2290838.653
## 9            Range    12000
## 10           Min       0
## 11           Max    12000
## 12 25th Percentile     3.404
## 13 75th Percentile    94.095

```

```

## 14 Subset w/o Outliers:
## 15             Count      4965
## 16             %       97.8%
## 17             Outlier %     2.2%
## 18             NA Count      0
## 19             Mean      208.248
## 20             Median      10
## 21 Standard Deviation   600.094
## 22             Variance  360113.198
## 23             Range      4900
## 24             Min        0
## 25             Max      4900

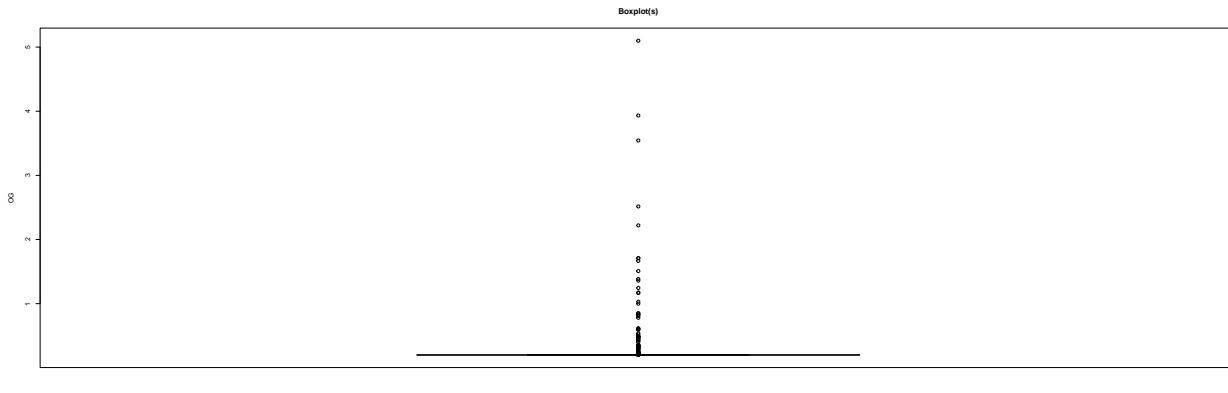
```



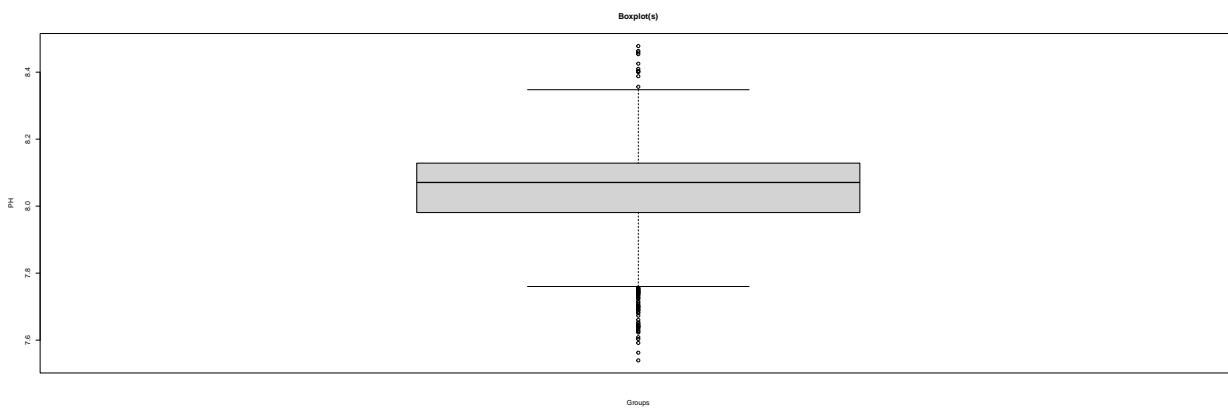
```

##               metric      V2
## 1             Variable: OG
## 2             Total N: 937
## 3             Count      937
## 4             NA Count      0
## 5             Mean      0.244
## 6             Median      0.2
## 7 Standard Deviation   0.286
## 8             Variance   0.082
## 9             Range      4.9
## 10            Min        0.2
## 11            Max       5.1
## 12            25th Percentile 0.2
## 13            75th Percentile 0.2
## 14 Subset w/o Outliers:
## 15             Count      923
## 16             %       98.5%
## 17             Outlier %     1.5%
## 18             NA Count      0
## 19             Mean      0.215
## 20             Median      0.2
## 21 Standard Deviation   0.078
## 22             Variance   0.006
## 23             Range      0.83
## 24             Min        0.2
## 25             Max       1.03

```



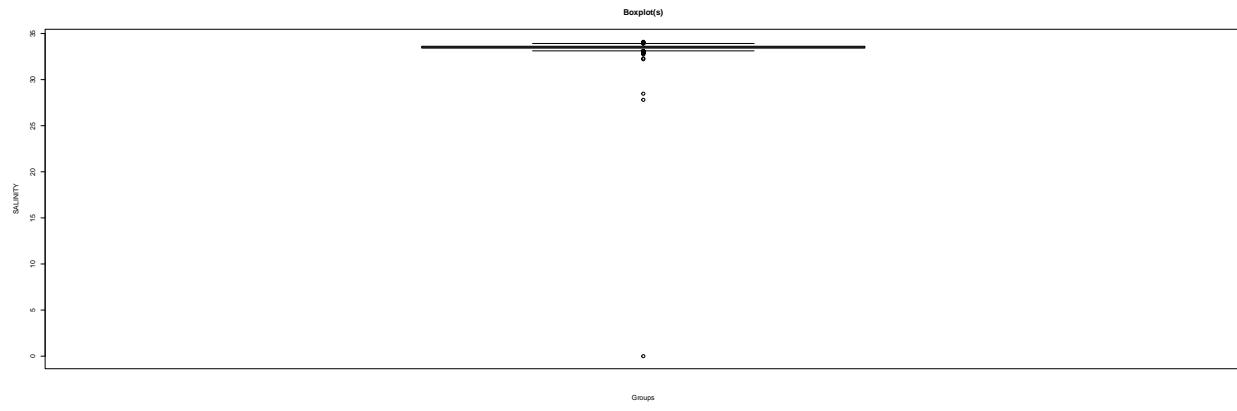
```
##               metric      V2
## 1             Variable: PH
## 2             Total N:
## 3             Count     2469
## 4             NA Count    0
## 5             Mean      8.05
## 6             Median     8.071
## 7             Standard Deviation 0.117
## 8             Variance    0.014
## 9             Range      0.938
## 10            Min       7.54
## 11            Max       8.478
## 12            25th Percentile 7.981
## 13            75th Percentile 8.128
## 14 Subset w/o Outliers:
## 15            Count     2435
## 16            %        98.6%
## 17            Outlier %   1.4%
## 18            NA Count    0
## 19            Mean      8.053
## 20            Median     8.071
## 21            Standard Deviation 0.108
## 22            Variance    0.012
## 23            Range      0.702
## 24            Min       7.7
## 25            Max       8.402
```



```

##               metric      V2
## 1          Variable: SALINITY
## 2          Total N:
## 3             Count     2505
## 4            NA Count     0
## 5              Mean    33.502
## 6              Median   33.527
## 7 Standard Deviation   0.706
## 8             Variance   0.499
## 9              Range    34.098
## 10             Min       0
## 11             Max    34.098
## 12        25th Percentile 33.418
## 13        75th Percentile 33.62
## 14 Subset w/o Outliers:
## 15             Count     2502
## 16             %    99.9%
## 17            Outlier %   0.1%
## 18            NA Count     0
## 19              Mean    33.519
## 20              Median   33.527
## 21 Standard Deviation   0.165
## 22             Variance   0.027
## 23              Range    1.888
## 24             Min     32.21
## 25             Max    34.098

```



```

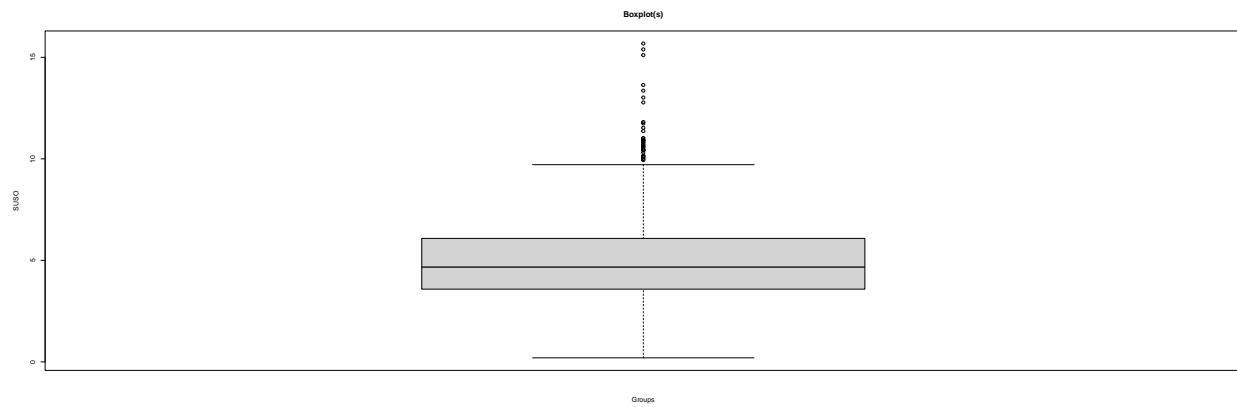
##               metric      V2
## 1          Variable: SUSO
## 2          Total N:
## 3             Count     976
## 4            NA Count     0
## 5              Mean    4.993
## 6              Median   4.67
## 7 Standard Deviation   2.164
## 8             Variance   4.681
## 9              Range    15.48
## 10             Min       0.2
## 11             Max    15.68
## 12        25th Percentile 3.583
## 13        75th Percentile 6.083

```

```

## 14 Subset w/o Outliers:
## 15             Count      966
## 16             %       99%
## 17        Outlier %      1%
## 18       NA Count       0
## 19             Mean     4.906
## 20             Median    4.639
## 21 Standard Deviation   1.991
## 22       Variance     3.966
## 23             Range    11.172
## 24             Min      0.2
## 25             Max     11.372

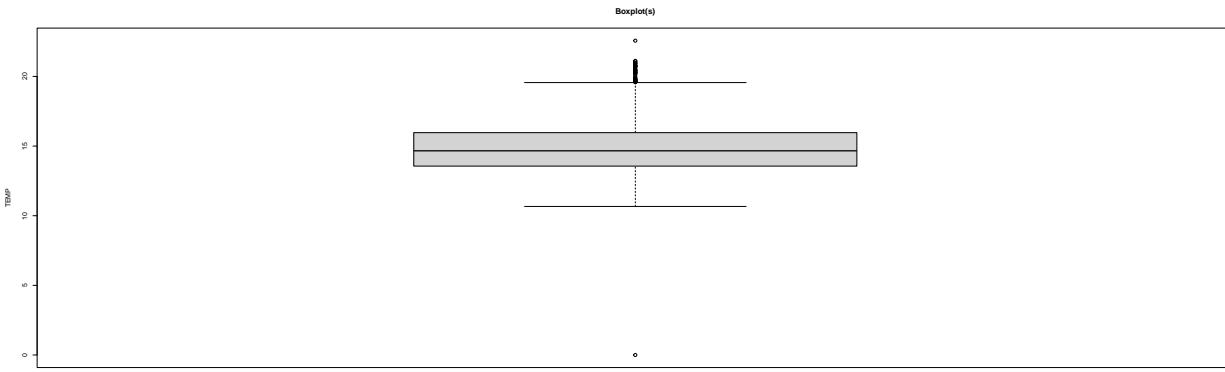
```



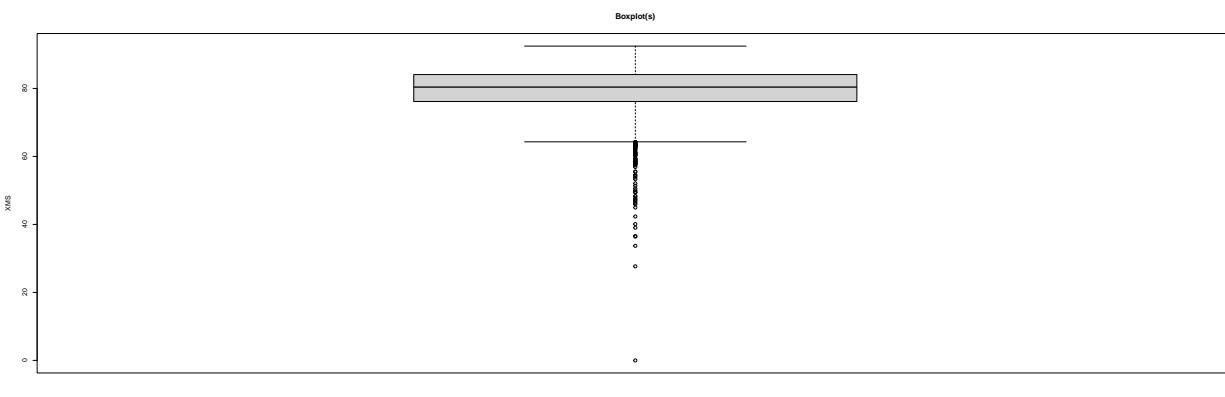
```

##               metric      V2
## 1           Variable: TEMP
## 2        Total N: 3373
## 3             Count      0
## 4       NA Count     14.843
## 5             Mean     14.661
## 6             Median   1.879
## 7 Standard Deviation   3.53
## 8       Variance     22.568
## 9             Range    13.561
## 10            Min      15.964
## 11            Max     20.46
## 12      25th Percentile
## 13      75th Percentile
## 14 Subset w/o Outliers:
## 15             Count      3357
## 16             %       99.5%
## 17        Outlier %      0.5%
## 18       NA Count       0
## 19             Mean     14.82
## 20             Median    14.65
## 21 Standard Deviation   1.821
## 22       Variance     3.315
## 23             Range    10.666
## 24             Min      20.46

```



```
##           metric      V2
## 1          Total N: Variable: XMS
## 2          Count      3344
## 3         NA Count       0
## 4          Mean     79.402
## 5          Median     80.4
## 6 Standard Deviation     6.979
## 7          Variance    48.713
## 8            Range    92.456
## 9            Min        0
## 10           Max    92.456
## 12 25th Percentile    76.156
## 13 75th Percentile    84.084
## 14 Subset w/o Outliers:
## 15          Count    3302
## 16            %     98.7%
## 17        Outlier %     1.3%
## 18         NA Count       0
## 19          Mean     79.791
## 20          Median    80.486
## 21 Standard Deviation    5.988
## 22          Variance   35.857
## 23            Range    33.88
## 24            Min    58.577
## 25            Max    92.456
```



```

##   date_sample CHLOROPHYLL DENSITY      DO      ENTERO      FECAL OG      PH SALINITY
## 1 1990-11-15          1.07 23.854 6.98000        NA        NA NA 8.130 33.59550
## 2 1991-01-02          NA       NA       NA 32.05128 125.64103 NA       NA NA
## 3 1991-01-03          NA       NA       NA 18.71795 51.28205 NA       NA NA
## 4 1991-01-07          NA       NA       NA 106.15385 80.76923 NA       NA NA
## 5 1991-01-08          NA       NA       NA 40.00000 40.55556 NA       NA NA
## 6 1991-01-09          NA       NA 5.47625 20.33898 40.50847 NA 8.195 33.51938
## 7 1991-01-10          NA       NA 5.50000 40.22727 78.18182 NA       NA 33.49300
##   SUSO      TEMP      XMS
## 1 NA 19.37000 60.22500
## 2 NA 14.50769 80.02564
## 3 NA 14.39231 83.89744
## 4 NA 14.54103 78.97436
## 5 NA 13.43000 80.20000
## 6 NA 14.26889 86.21111
## 7 NA 13.75000 80.45833
##             vars     n    mean      sd median trimmed   mad   min     max
## date_sample     1 5450    NaN      NA     NA  NaN  NA Inf -Inf
## CHLOROPHYLL   2 2092    3.91    3.41    2.85    3.33  2.28  0.29  30.35
## DENSITY        3 2246    24.85    0.47   24.87   24.87  0.46 22.84  26.04
## DO            4 2504    7.04    0.99    7.23    7.11  0.86  0.00 10.33
## ENTERO         5 5197 284.21 1325.88    7.33   25.74  7.91  0.00 18000.00
## FECAL          6 5078 409.93 1513.55   10.56   69.45 12.57  0.00 12000.00
## OG            7  937    0.24    0.29    0.20    0.20  0.20  0.00  0.20  5.10
## PH            8 2469    8.05    0.12    8.07    8.06  0.10  7.54  8.48
## SALINITY       9 2505    33.50    0.71   33.53   33.52  0.15  0.00 34.10
## SUSO          10  976    4.99    2.16    4.67    4.80  1.80  0.20 15.68
## TEMP          11 3373   14.84    1.88   14.66   14.75  1.78  0.00 22.57
## XMS           12 3344   79.40    6.98   80.40   80.08  5.76  0.00 92.46
##             range   skew kurtosis     se
## date_sample   -Inf    NA     NA  NA
## CHLOROPHYLL 30.06   2.25    7.84  0.07
## DENSITY      3.20  -0.42    0.25  0.01
## DO          10.33  -1.05    3.72  0.02
## ENTERO      18000.00   6.95   53.47 18.39
## FECAL       12000.00   5.81   36.94 21.24
## OG          4.90  10.89  144.21  0.01
## PH          0.94  -0.66   1.32  0.00
## SALINITY    34.10 -42.97 2021.03  0.01
## SUSO        15.48   1.09   2.23  0.07
## TEMP        22.57   0.35   1.15  0.03
## XMS         92.46  -1.86   9.43  0.12
##             date_sample parameter      Avg
## Not NA n      36811    36811 35669.000
## NA n          0        0 1142.000
## Not NA %      1        1  0.969
## NA %          0        0  0.031
owt_df02_gb04_wkly = ts_eda(ts_df = owt_df02_gb04,
                             form_lead = "date_sample",
                             form_cast = "parameter",
                             param_lst = param_lst02,
                             rtn_met = TRUE,
                             box = FALSE,

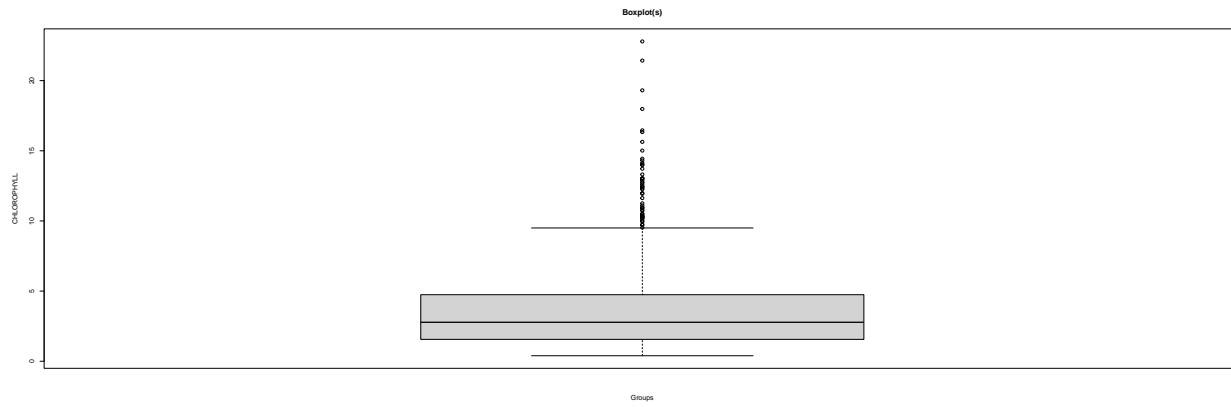
```

```

            week_agg = TRUE
        )

##          metric      V2
## 1      Variable: CHLOROPHYLL
## 2      Total N:
## 3          Count     1071
## 4          NA Count     0
## 5          Mean     3.664
## 6          Median    2.778
## 7      Standard Deviation    2.98
## 8          Variance   8.879
## 9          Range    22.399
## 10         Min     0.393
## 11         Max    22.792
## 12      25th Percentile   1.56
## 13      75th Percentile   4.744
## 14 Subset w/o Outliers:
## 15          Count     1049
## 16          %     97.9%
## 17      Outlier %     2.1%
## 18          NA Count     0
## 19          Mean     3.423
## 20          Median    2.726
## 21      Standard Deviation   2.465
## 22          Variance   6.074
## 23          Range    12.205
## 24         Min     0.393
## 25         Max    12.599

```



```

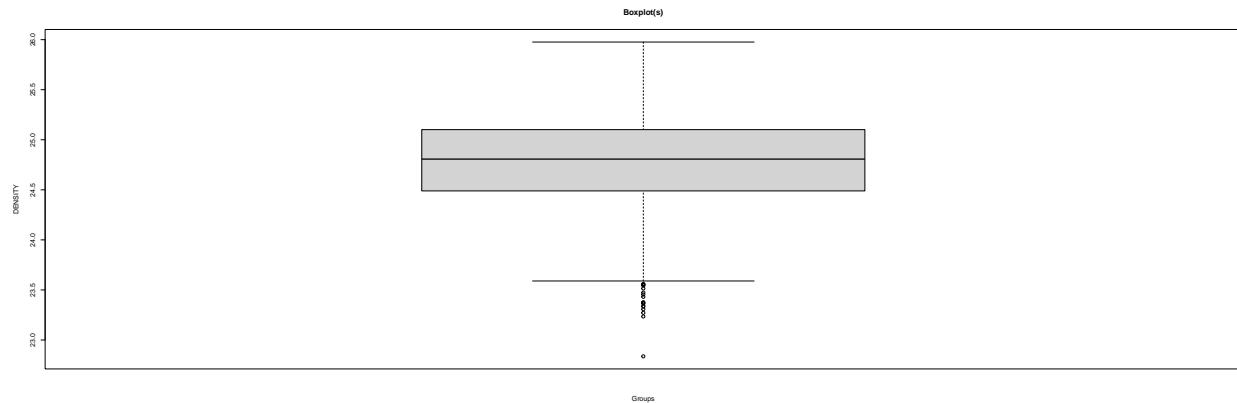
##          metric      V2
## 1      Variable: DENSITY
## 2      Total N:
## 3          Count     1098
## 4          NA Count     0
## 5          Mean     24.78
## 6          Median    24.807
## 7      Standard Deviation   0.467
## 8          Variance   0.218
## 9          Range    3.139

```

```

## 10             Min      22.837
## 11             Max      25.976
## 12    25th Percentile 24.489
## 13   75th Percentile  25.1
## 14 Subset w/o Outliers:
## 15             Count 1089
## 16             %     99.2%
## 17        Outlier %  0.8%
## 18       NA Count    0
## 19             Mean 24.793
## 20             Median 24.812
## 21 Standard Deviation 0.448
## 22            Variance 0.201
## 23             Range 2.545
## 24             Min   23.431
## 25             Max   25.976

```



```

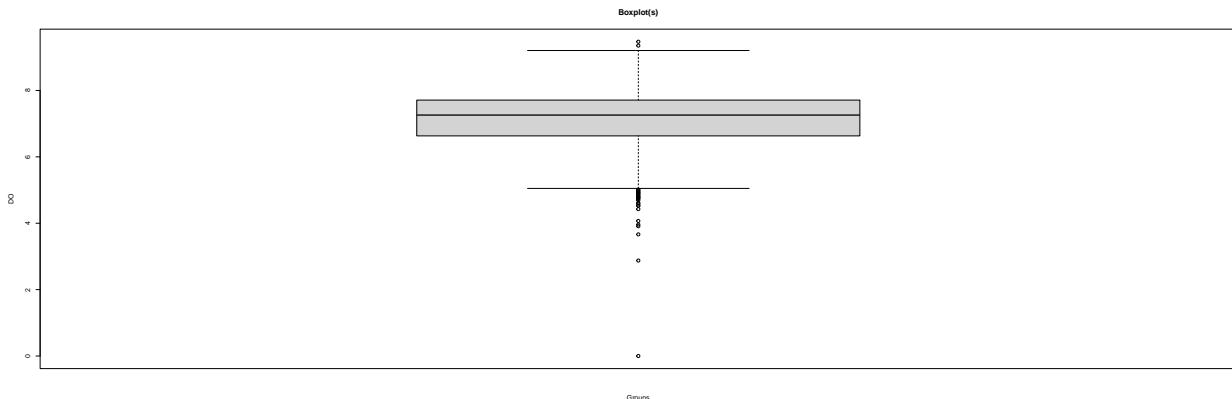
##           metric      V2
## 1 Variable: D0
## 2 Total N:
## 3   Count      1203
## 4  NA Count     0
## 5   Mean      7.102
## 6   Median     7.259
## 7 Standard Deviation 0.876
## 8   Variance    0.767
## 9   Range      9.468
## 10  Min        0
## 11  Max      9.468
## 12 25th Percentile 6.632
## 13 75th Percentile  7.71
## 14 Subset w/o Outliers:
## 15   Count      1196
## 16   %        99.4%
## 17  Outlier %  0.6%
## 18  NA Count     0
## 19   Mean      7.125
## 20   Median     7.265
## 21 Standard Deviation 0.821
## 22   Variance    0.674
## 23   Range      4.942

```

```

## 24          Min       4.527
## 25          Max      9.468

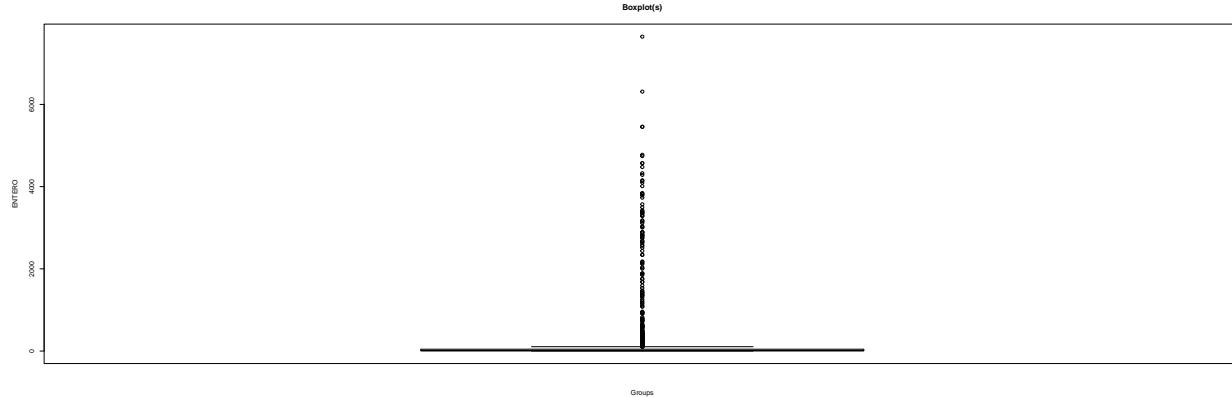
```



```

##           metric        V2
## 1 Variable: ENTERO
## 2 Total N:
## 3     Count      1614
## 4    NA Count       0
## 5     Mean     214.198
## 6     Median     12.258
## 7 Standard Deviation   718.919
## 8     Variance   516845.04
## 9     Range     7646.278
## 10    Min         2
## 11    Max     7648.278
## 12 25th Percentile   5.434
## 13 75th Percentile   45.567
## 14 Subset w/o Outliers:
## 15     Count      1557
## 16     %       96.5%
## 17 Outlier %      3.5%
## 18    NA Count       0
## 19     Mean     91.784
## 20     Median     11.698
## 21 Standard Deviation   275.13
## 22     Variance   75696.459
## 23     Range     2344.566
## 24    Min         2
## 25    Max     2346.566

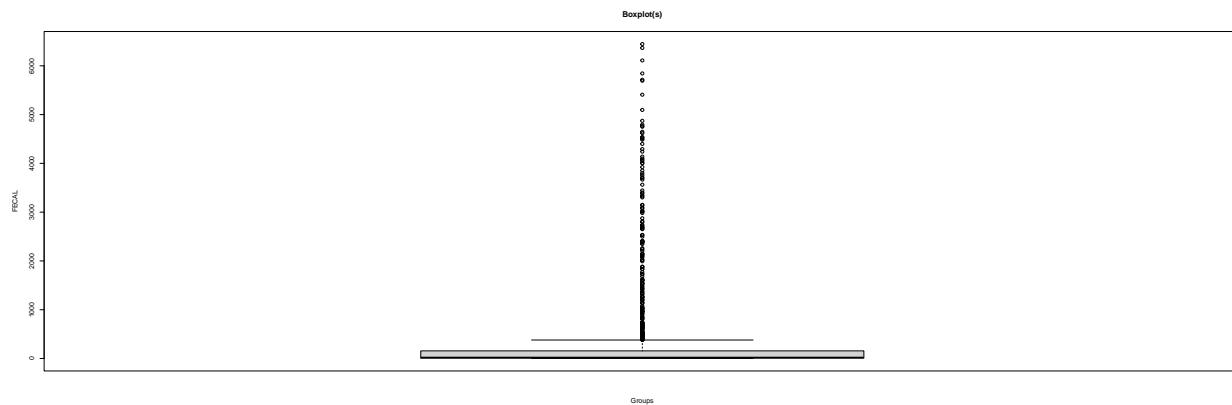
```



```

##           metric      V2
## 1          Total N: Variable: FECAL
## 2          Total N: 1614
## 3            Count      0
## 4        NA Count      0
## 5            Mean 304.597
## 6            Median 23.303
## 7 Standard Deviation 826.25
## 8            Variance 682689.747
## 9            Range 6443.07
## 10           Min      2
## 11           Max 6445.07
## 12      25th Percentile 6.236
## 13      75th Percentile 156.176
## 14 Subset w/o Outliers:
## 15            Count 1562
## 16            % 96.8%
## 17        Outlier % 3.2%
## 18        NA Count      0
## 19            Mean 177.977
## 20            Median 20.506
## 21 Standard Deviation 421.323
## 22            Variance 177512.97
## 23            Range 2737.584
## 24           Min      2
## 25           Max 2739.584

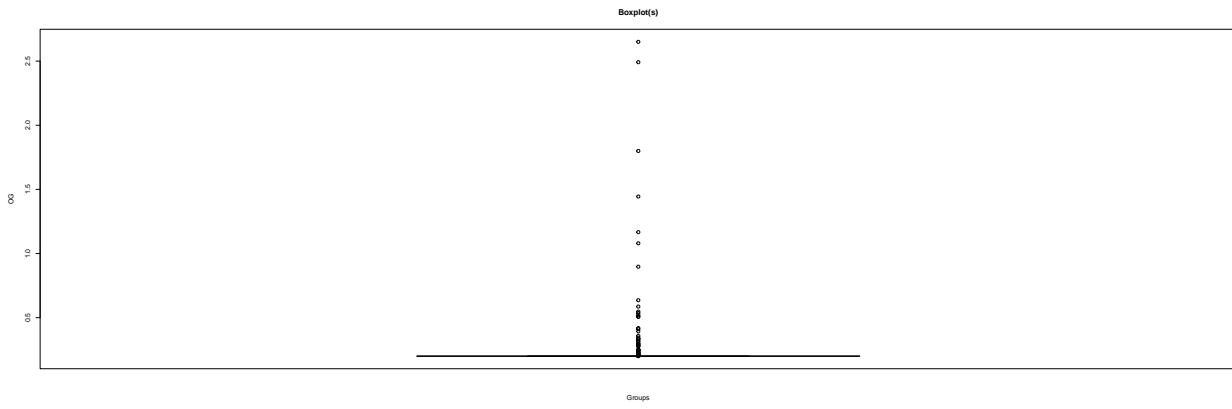
```



```

##               metric      V2
## 1             Variable: OG
## 2       Total N:
## 3           Count     351
## 4       NA Count      0
## 5           Mean    0.244
## 6           Median   0.2
## 7   Standard Deviation 0.227
## 8           Variance  0.052
## 9           Range    2.45
## 10          Min     0.2
## 11          Max    2.65
## 12   25th Percentile  0.2
## 13   75th Percentile  0.2
## 14 Subset w/o Outliers:
## 15          Count    345
## 16          %    98.3%
## 17   Outlier %    1.7%
## 18       NA Count      0
## 19           Mean    0.218
## 20           Median   0.2
## 21   Standard Deviation 0.068
## 22           Variance  0.005
## 23           Range    0.697
## 24          Min     0.2
## 25          Max    0.897

```



```

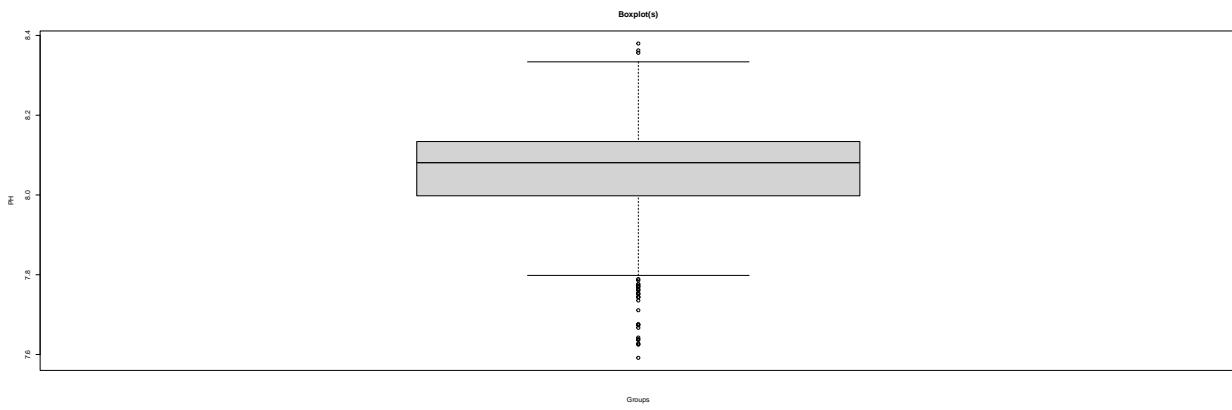
##               metric      V2
## 1             Variable: PH
## 2       Total N:
## 3           Count    1205
## 4       NA Count      0
## 5           Mean    8.062
## 6           Median   8.081
## 7   Standard Deviation 0.11
## 8           Variance  0.012
## 9           Range    0.788
## 10          Min    7.592
## 11          Max    8.38
## 12   25th Percentile  7.998
## 13   75th Percentile  8.134

```

```

## 14 Subset w/o Outliers:
## 15             Count      1193
## 16             %        99%
## 17       Outlier %       1%
## 18       NA Count       0
## 19             Mean     8.066
## 20             Median    8.082
## 21 Standard Deviation   0.102
## 22       Variance      0.01
## 23             Range    0.644
## 24             Min      7.735
## 25             Max      8.38

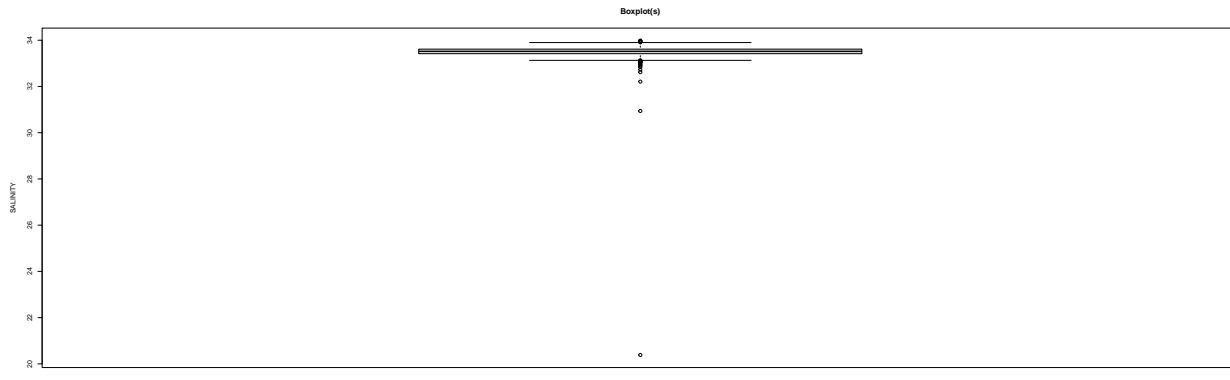
```



```

##               metric      V2
## 1          Variable: SALINITY
## 2       Total N:      1198
## 3             Count      1198
## 4       NA Count       0
## 5             Mean    33.501
## 6             Median   33.521
## 7 Standard Deviation   0.42
## 8       Variance     0.177
## 9             Range   13.596
## 10            Min     20.385
## 11            Max     33.981
## 12  25th Percentile  33.418
## 13  75th Percentile  33.615
## 14 Subset w/o Outliers:
## 15             Count      1195
## 16             %        99.7%
## 17       Outlier %       0.3%
## 18       NA Count       0
## 19             Mean    33.516
## 20             Median   33.522
## 21 Standard Deviation   0.161
## 22       Variance     0.026
## 23             Range   1.36
## 24             Min     32.621
## 25             Max     33.981

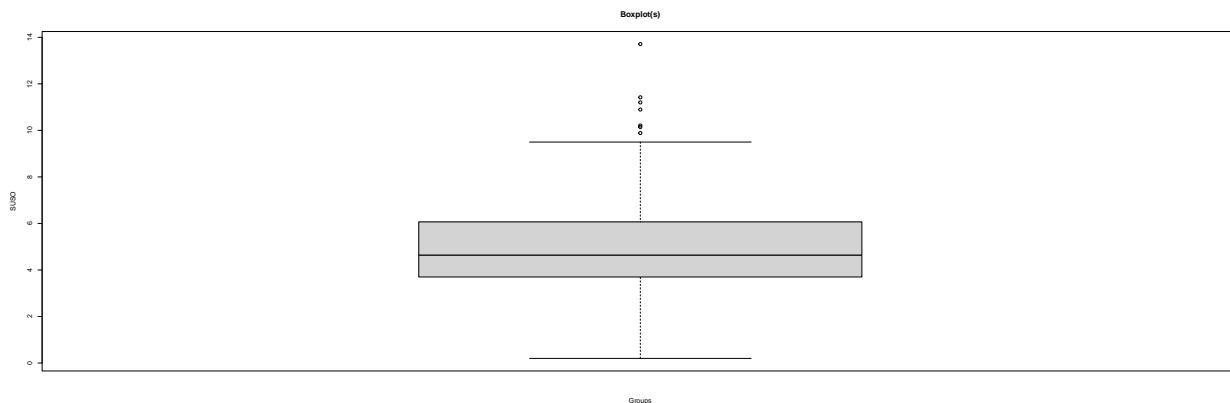
```



```

##           metric      V2
## 1          Variable: SUSO
## 2          Total N:
## 3          Count      367
## 4          NA Count     0
## 5          Mean       4.914
## 6          Median      4.637
## 7          Standard Deviation   1.85
## 8          Variance      3.424
## 9          Range        13.509
## 10         Min         0.2
## 11         Max        13.709
## 12         25th Percentile 3.699
## 13         75th Percentile 6.067
## 14 Subset w/o Outliers:
## 15         Count      363
## 16         %         98.9%
## 17         Outlier %    1.1%
## 18         NA Count     0
## 19         Mean       4.838
## 20         Median      4.611
## 21         Standard Deviation   1.708
## 22         Variance      2.918
## 23         Range        10.009
## 24         Min         0.2
## 25         Max        10.209

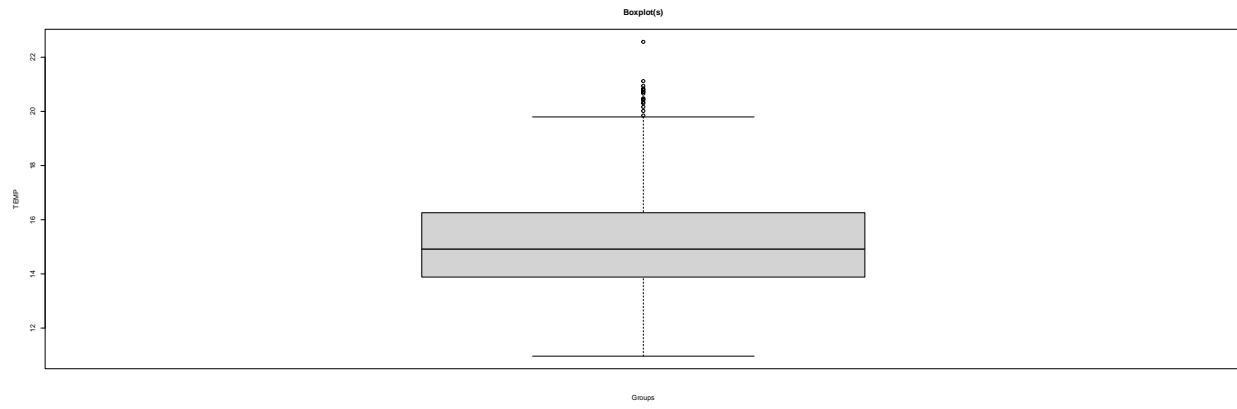
```



```

##               metric      V2
## 1          Variable: TEMP
## 2          Total N:
## 3             Count     1547
## 4        NA Count      0
## 5            Mean    15.145
## 6            Median   14.915
## 7 Standard Deviation    1.783
## 8            Variance   3.179
## 9            Range    11.605
## 10           Min    10.963
## 11           Max    22.568
## 12      25th Percentile 13.883
## 13      75th Percentile 16.258
## 14 Subset w/o Outliers:
## 15            Count     1538
## 16            %    99.4%
## 17        Outlier %    0.6%
## 18        NA Count      0
## 19            Mean    15.11
## 20            Median   14.897
## 21 Standard Deviation    1.73
## 22            Variance  2.992
## 23            Range    9.52
## 24           Min    10.963
## 25           Max    20.483

```



```

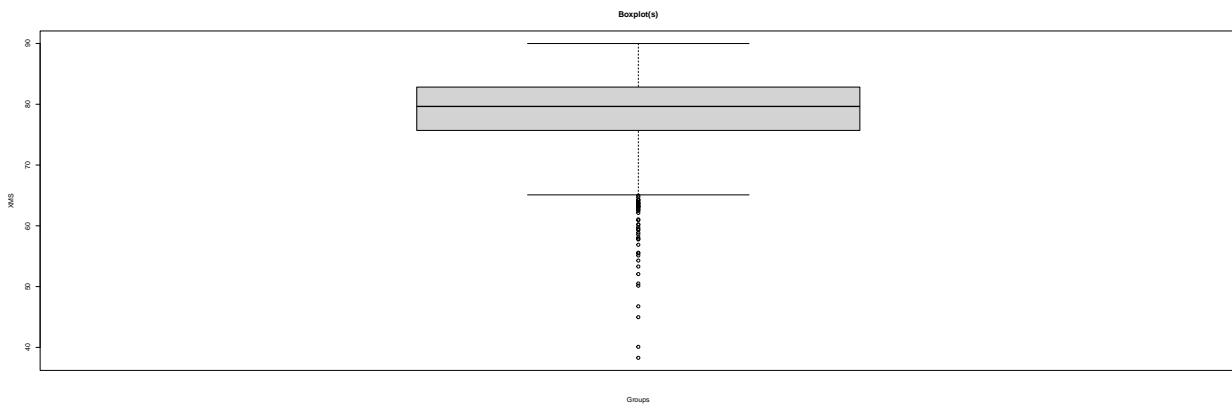
##               metric      V2
## 1          Variable: XMS
## 2          Total N:
## 3             Count     1540
## 4        NA Count      0
## 5            Mean    78.614
## 6            Median   79.642
## 7 Standard Deviation    6.152
## 8            Variance  37.845
## 9            Range    51.708
## 10           Min    38.282
## 11           Max    89.99
## 12      25th Percentile 75.699
## 13      75th Percentile 82.824

```

```

## 14 Subset w/o Outliers:
## 15          Count      1519
## 16          %       98.6%
## 17     Outlier %      1.4%
## 18     NA Count        0
## 19          Mean    78.961
## 20          Median   79.742
## 21 Standard Deviation 5.384
## 22          Variance 28.987
## 23          Range   29.778
## 24          Min    60.212
## 25          Max    89.99

```



```

##   date_sample CHLOROPHYLL DENSITY      DO      ENTERO      FECAL OG     PH
## 1 1990-11-19      1.07 23.854 6.980000      NA      NA NA 8.130
## 2 1991-01-07      NA     NA      NA 25.38462 88.46154 NA     NA
## 3 1991-01-14      NA     NA 5.445417 58.12973 137.21730 NA 8.195
## 4 1991-01-21      NA     NA      NA 23.20513 59.61538 NA     NA
## 5 1991-01-28      NA     NA      NA 31.81891 69.78365 NA     NA
## 6 1991-02-04      NA     NA      NA 154.23077 428.58974 NA     NA
## 7 1991-02-11      NA     NA      NA 16.83814 14.63141 NA     NA
##   SALINITY SUSO      TEMP      XMS
## 1 33.59550  NA 19.37000 60.22500
## 2      NA  NA 14.45000 81.96154
## 3 33.50113  NA 14.07478 82.04876
## 4      NA  NA 14.54487 70.19231
## 5      NA  NA 14.28846 77.61167
## 6      NA  NA 14.23462 81.83333
## 7      NA  NA 14.46154 70.83333
##   vars     n   mean      sd median trimmed   mad   min   max range
##   date_sample  1 1616    NaN     NA     NA  NaN  NA Inf -Inf -Inf
##   CHLOROPHYLL 2 1071    3.66    2.98    2.78   3.16  2.07  0.39  22.79 22.40
##   DENSITY     3 1098    24.78    0.47   24.81  24.80  0.45  22.84  25.98  3.14
##   DO          4 1203    7.10    0.88    7.26   7.18  0.74  0.00  9.47  9.47
##   ENTERO      5 1614   214.20  718.92   12.26  35.21 13.56  2.00 7648.28 7646.28
##   FECAL       6 1614   304.60  826.25   23.30  86.48 29.92  2.00 6445.07 6443.07
##   OG          7  351     0.24    0.23    0.20   0.20  0.00  0.20  2.65  2.45
##   PH          8 1205     8.06    0.11    8.08   8.07  0.10  7.59  8.38  0.79
##   SALINITY    9 1198    33.50    0.42   33.52  33.52  0.15  20.38  33.98 13.60
##   SUSO        10  367     4.91    1.85    4.64   4.81  1.61  0.20 13.71 13.51
##   TEMP        11 1547    15.14    1.78   14.92  15.04  1.75 10.96 22.57 11.61

```

```

## XMS          12 1540 78.61   6.15 79.64 79.25 5.14 38.28 89.99 51.71
##              skew kurtosis    se
## date_sample    NA        NA      NA
## CHLOROPHYLL  1.97      5.38  0.09
## DENSITY       -0.45     0.37  0.01
## DO            -1.19     4.14  0.03
## ENTERO        4.84     26.56 17.89
## FECAL         4.13     18.61 20.57
## OG            7.92     70.11  0.01
## PH            -0.84     1.63  0.00
## SALINITY      -25.63    791.34 0.01
## SUSO          0.79      1.80  0.10
## TEMP          0.59      0.20  0.05
## XMS           -1.47     4.30  0.16
##              date_sample parameter      Avg
## Not NA n     36811     36811 35669.000
## NA n          0          0     1142.000
## Not NA %      1          1     0.969
## NA %          0          0     0.031

owt_df02_gb04_wkly02 = ts_eda(ts_df = owt_df02_gb04,
                               form_lead = "date_sample",
                               form_cast = "parameter",
                               param_lst = param_lst02,
                               rtn_met = TRUE,
                               box = FALSE,
                               week_agg = TRUE,
                               out_rm = TRUE
)

```

```

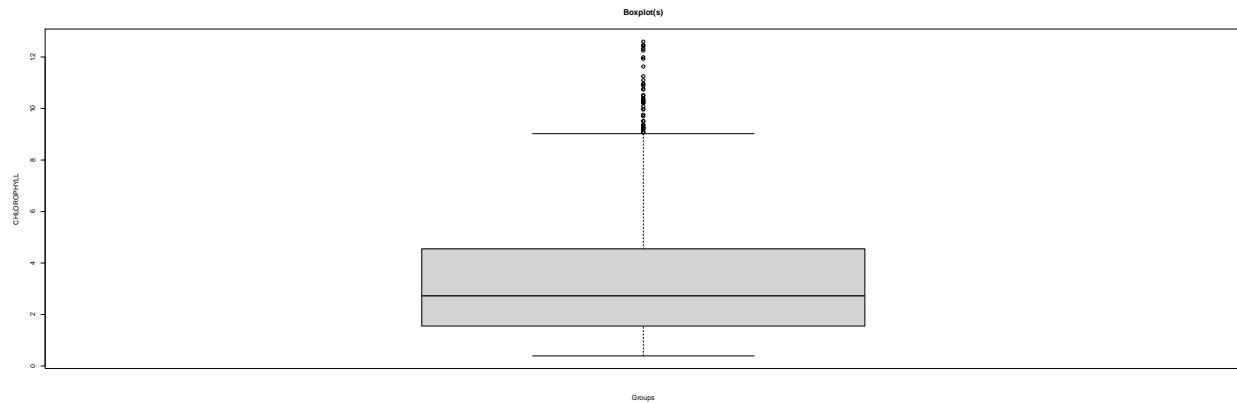
## [1] 1071    2
##              metric          V2
## 1                  Variable: CHLOROPHYLL
## 2                  Total N:
## 3                  Count          1049
## 4                  NA Count       0
## 5                  Mean          3.423
## 6                  Median         2.726
## 7      Standard Deviation      2.465
## 8                  Variance        6.074
## 9                  Range          12.205
## 10                 Min          0.393
## 11                 Max          12.599
## 12      25th Percentile       1.552
## 13      75th Percentile        4.55
## 14 Subset w/o Outliers:
## 15                  Count        1035
## 16                  %          98.7%
## 17      Outlier %          1.3%
## 18                  NA Count       0
## 19                  Mean          3.31
## 20                  Median         2.699
## 21      Standard Deviation      2.278
## 22                  Variance        5.188

```

```

## 23          Range      10.373
## 24          Min       0.393
## 25          Max      10.767

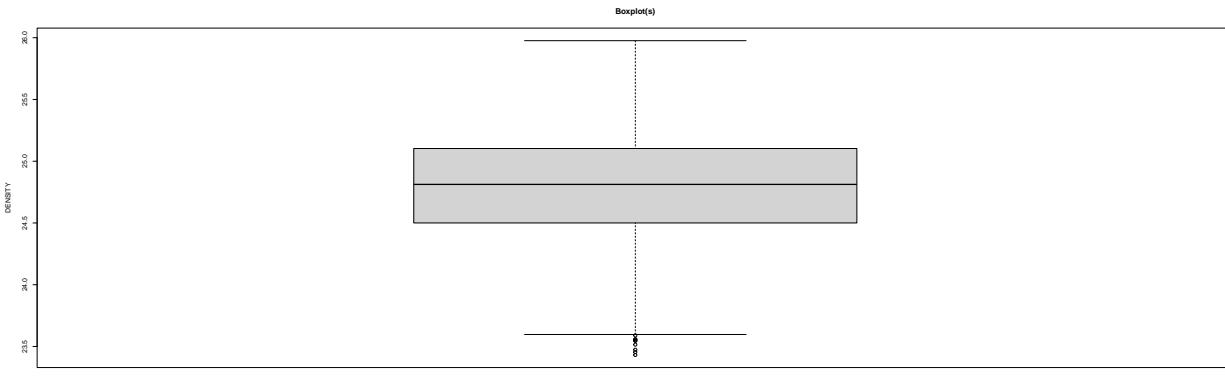
```



```

## [1] 1098    2
##               metric      V2
## 1                   Variable: DENSITY
## 2                   Total N:
## 3                   Count     1089
## 4                   NA Count   0
## 5                   Mean      24.793
## 6                   Median     24.812
## 7   Standard Deviation   0.448
## 8           Variance      0.201
## 9           Range        2.545
## 10          Min         23.431
## 11          Max         25.976
## 12   25th Percentile    24.5
## 13   75th Percentile    25.103
## 14 Subset w/o Outliers:
## 15          Count     1088
## 16          %        99.9%
## 17 Outlier %
## 18          NA Count   0
## 19          Mean      24.794
## 20          Median     24.813
## 21 Standard Deviation   0.446
## 22           Variance   0.199
## 23           Range      2.522
## 24          Min         23.454
## 25          Max         25.976

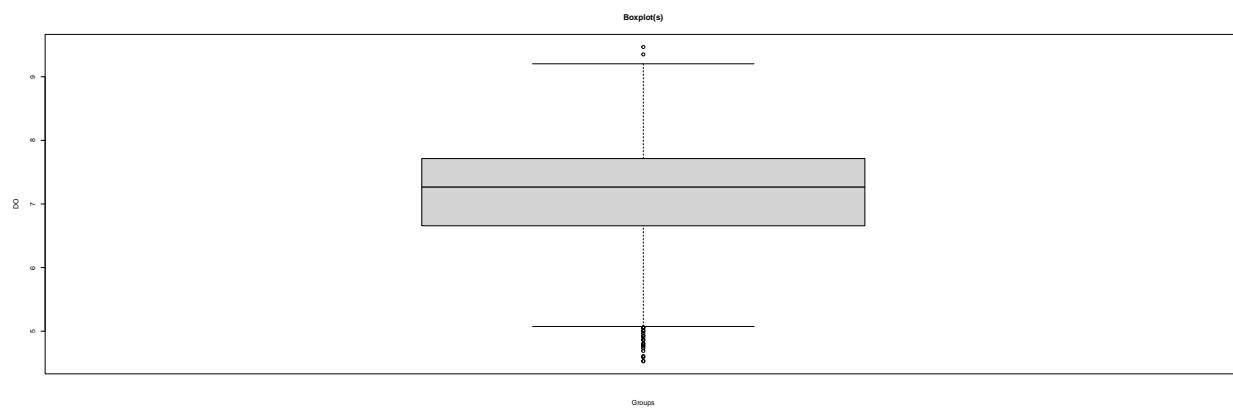
```



```

## [1] 1203      2
##           metric          V2
## 1             Variable: D0
## 2             Total N:
## 3             Count       1196
## 4             NA Count     0
## 5             Mean        7.125
## 6             Median       7.265
## 7             Standard Deviation 0.821
## 8             Variance      0.674
## 9             Range         4.942
## 10            Min          4.527
## 11            Max          9.468
## 12            25th Percentile 6.657
## 13            75th Percentile 7.714
## 14 Subset w/o Outliers:
## 15            Count       1192
## 16            %          99.7%
## 17            Outlier %    0.3%
## 18            NA Count     0
## 19            Mean        7.133
## 20            Median       7.267
## 21            Standard Deviation 0.809
## 22            Variance      0.654
## 23            Range         4.778
## 24            Min          4.69
## 25            Max          9.468

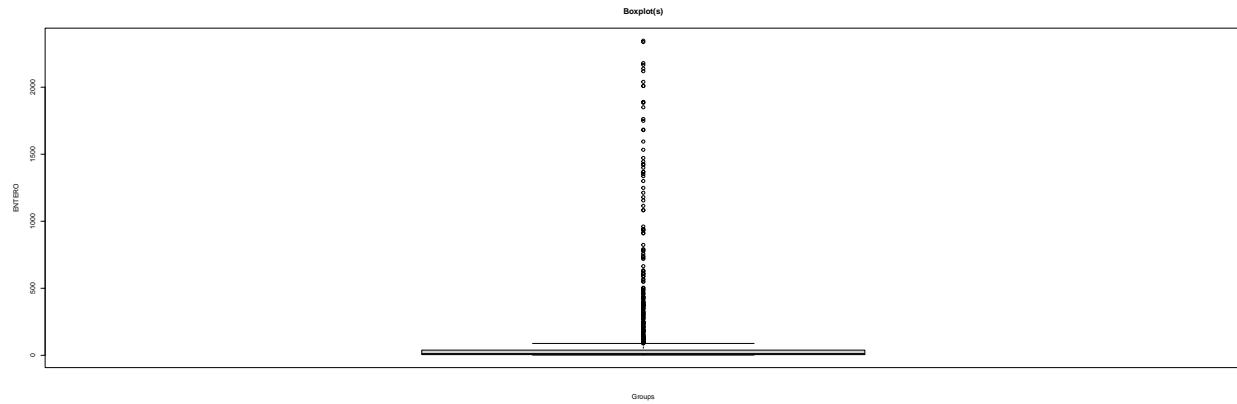
```



```

## [1] 1614    2
##               metric          V2
## 1                      Variable: ENTERO
## 2      Total N:
## 3          Count       1557
## 4          NA Count      0
## 5          Mean        91.784
## 6          Median       11.698
## 7  Standard Deviation     275.13
## 8          Variance     75696.459
## 9          Range       2344.566
## 10         Min          2
## 11         Max       2346.566
## 12  25th Percentile      5.128
## 13  75th Percentile      38.694
## 14 Subset w/o Outliers:
## 15         Count       1516
## 16         %        97.4%
## 17  Outlier %        2.6%
## 18          NA Count      0
## 19          Mean        52.359
## 20          Median       11.146
## 21  Standard Deviation   118.074
## 22          Variance     13941.503
## 23          Range       909.627
## 24         Min          2
## 25         Max       911.627

```



```

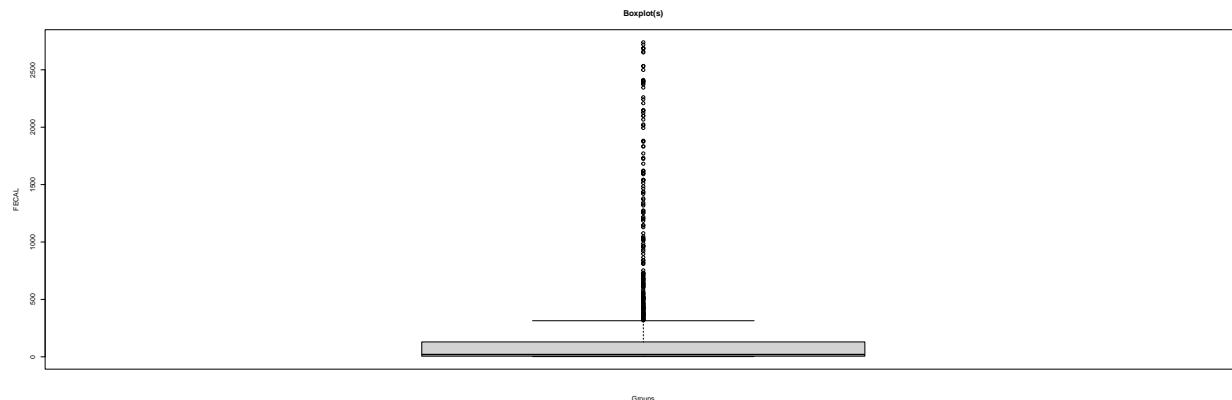
## [1] 1614    2
##               metric          V2
## 1                      Variable: FECAL
## 2      Total N:
## 3          Count       1562
## 4          NA Count      0
## 5          Mean        177.977
## 6          Median       20.506
## 7  Standard Deviation     421.323
## 8          Variance     177512.97
## 9          Range       2737.584
## 10         Min          2
## 11         Max       2739.584

```

```

## 12      25th Percentile       6.048
## 13      75th Percentile     129.582
## 14 Subset w/o Outliers:
## 15          Count      1509
## 16          %        96.6%
## 17      Outlier %       3.4%
## 18      NA Count        0
## 19          Mean     111.629
## 20          Median    18.395
## 21 Standard Deviation   219.97
## 22          Variance  48386.762
## 23          Range    1436.388
## 24          Min      2
## 25          Max    1438.388

```

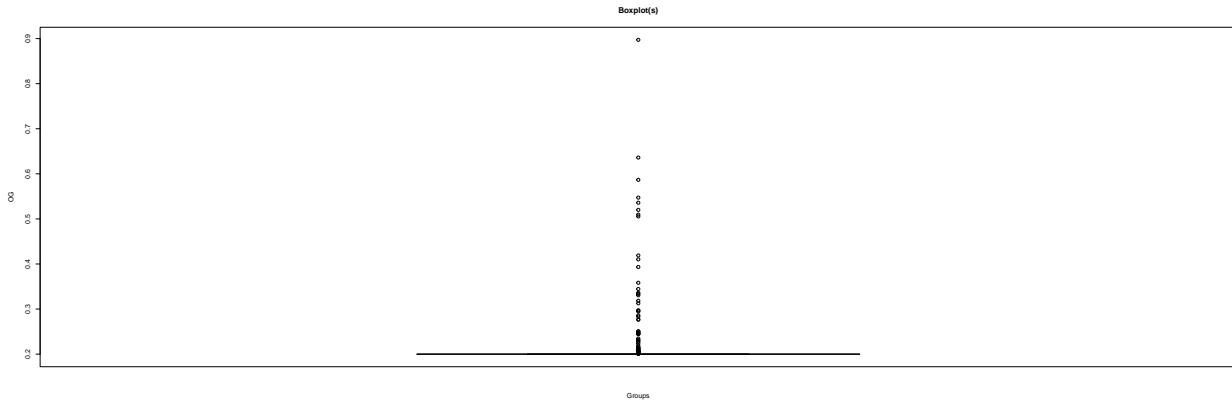


```

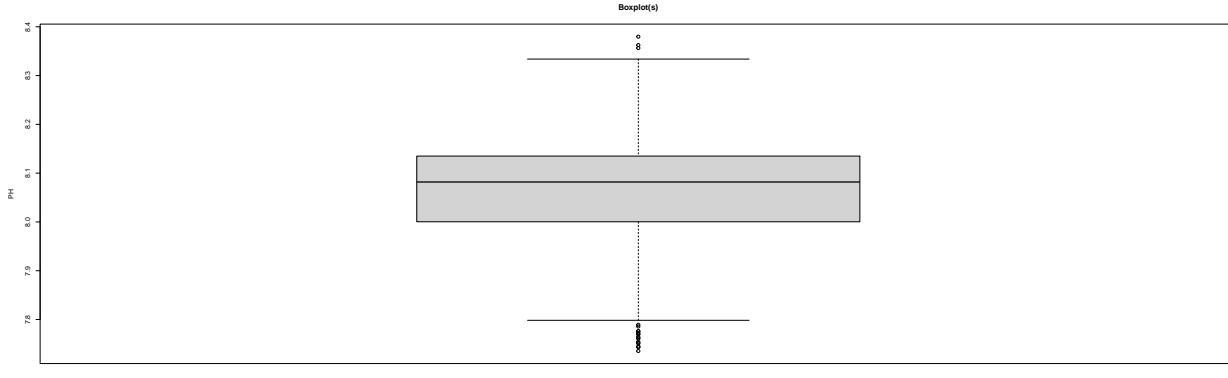
## [1] 351  2
##           metric      V2
## 1             Variable: OG
## 2      Total N:
## 3          Count      345
## 4          NA Count        0
## 5          Mean     0.218
## 6          Median    0.2
## 7 Standard Deviation  0.068
## 8          Variance  0.005
## 9          Range    0.697
## 10         Min      0.2
## 11         Max     0.897
## 12      25th Percentile    0.2
## 13      75th Percentile    0.2
## 14 Subset w/o Outliers:
## 15          Count      337
## 16          %        97.7%
## 17      Outlier %       2.3%
## 18      NA Count        0
## 19          Mean     0.209
## 20          Median    0.2
## 21 Standard Deviation  0.03
## 22          Variance  0.001
## 23          Range    0.219
## 24          Min      0.2

```

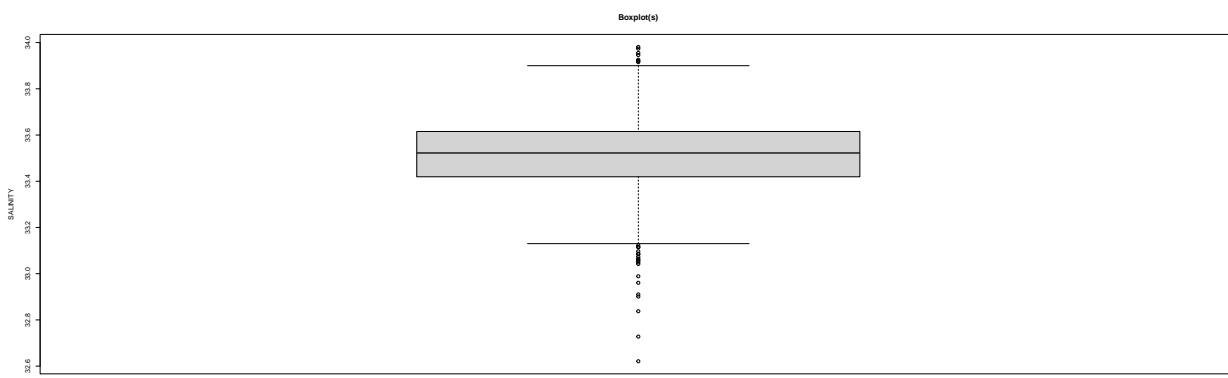
```
## 25          Max      0.419
```



```
## [1] 1205    2
##           metric      V2
## 1                      Variable: PH
## 2           Total N:
## 3           Count     1193
## 4       NA Count      0
## 5           Mean     8.066
## 6           Median    8.082
## 7 Standard Deviation   0.102
## 8           Variance   0.01
## 9           Range     0.644
## 10          Min      7.735
## 11          Max      8.38
## 12 25th Percentile     8
## 13 75th Percentile    8.135
## 14 Subset w/o Outliers:
## 15          Count     1185
## 16          %      99.3%
## 17 Outlier %      0.7%
## 18       NA Count      0
## 19           Mean     8.068
## 20           Median    8.082
## 21 Standard Deviation   0.099
## 22           Variance   0.01
## 23           Range     0.602
## 24          Min      7.761
## 25          Max      8.363
```



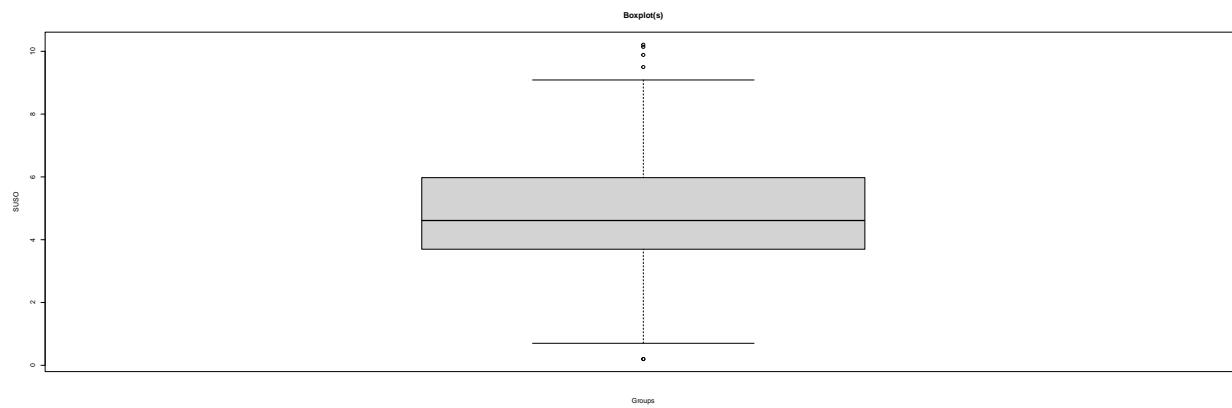
```
## [1] 1198    2
##           metric          V2
## 1             Variable: SALINITY
## 2             Total N:
## 3             Count        1195
## 4             NA Count      0
## 5             Mean       33.516
## 6             Median      33.522
## 7             Standard Deviation 0.161
## 8             Variance     0.026
## 9             Range        1.36
## 10            Min         32.621
## 11            Max         33.981
## 12            25th Percentile 33.419
## 13            75th Percentile 33.615
## 14 Subset w/o Outliers:
## 15            Count        1188
## 16            %         99.4%
## 17            Outlier %     0.6%
## 18            NA Count      0
## 19            Mean       33.52
## 20            Median      33.523
## 21            Standard Deviation 0.152
## 22            Variance     0.023
## 23            Range        0.939
## 24            Min         33.042
## 25            Max         33.981
```



```

## [1] 367    2
##               metric          V2
## 1                   Variable: SUSO
## 2           Total N:
## 3             Count      363
## 4            NA Count     0
## 5             Mean      4.838
## 6             Median     4.611
## 7 Standard Deviation     1.708
## 8             Variance    2.918
## 9             Range     10.009
## 10            Min       0.2
## 11            Max     10.209
## 12 25th Percentile     3.697
## 13 75th Percentile     5.977
## 14 Subset w/o Outliers:
## 15            Count      361
## 16            %      99.4%
## 17        Outlier %     0.6%
## 18            NA Count     0
## 19             Mean      4.808
## 20             Median     4.607
## 21 Standard Deviation     1.666
## 22             Variance    2.775
## 23             Range     9.685
## 24            Min       0.2
## 25            Max     9.885

```



```

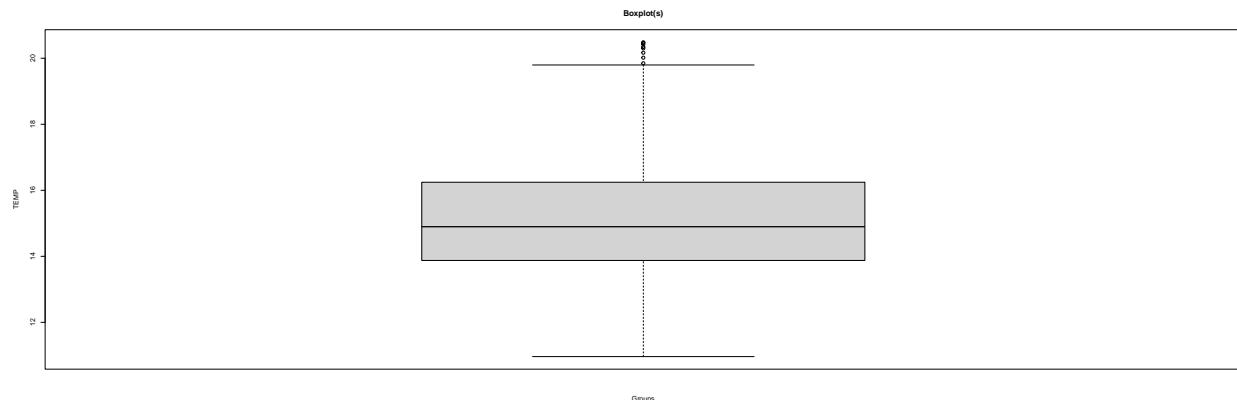
## [1] 1547    2
##               metric          V2
## 1                   Variable: TEMP
## 2           Total N:
## 3             Count      1538
## 4            NA Count     0
## 5             Mean      15.11
## 6             Median     14.897
## 7 Standard Deviation     1.73
## 8             Variance    2.992
## 9             Range      9.52
## 10            Min      10.963
## 11            Max      20.483

```

```

## 12      25th Percentile      13.876
## 13      75th Percentile      16.244
## 14 Subset w/o Outliers:
## 15          Count      1534
## 16          %      99.7%
## 17      Outlier %      0.3%
## 18      NA Count      0
## 19          Mean      15.097
## 20          Median      14.893
## 21 Standard Deviation      1.711
## 22          Variance      2.926
## 23          Range      9.335
## 24          Min      10.963
## 25          Max      20.298

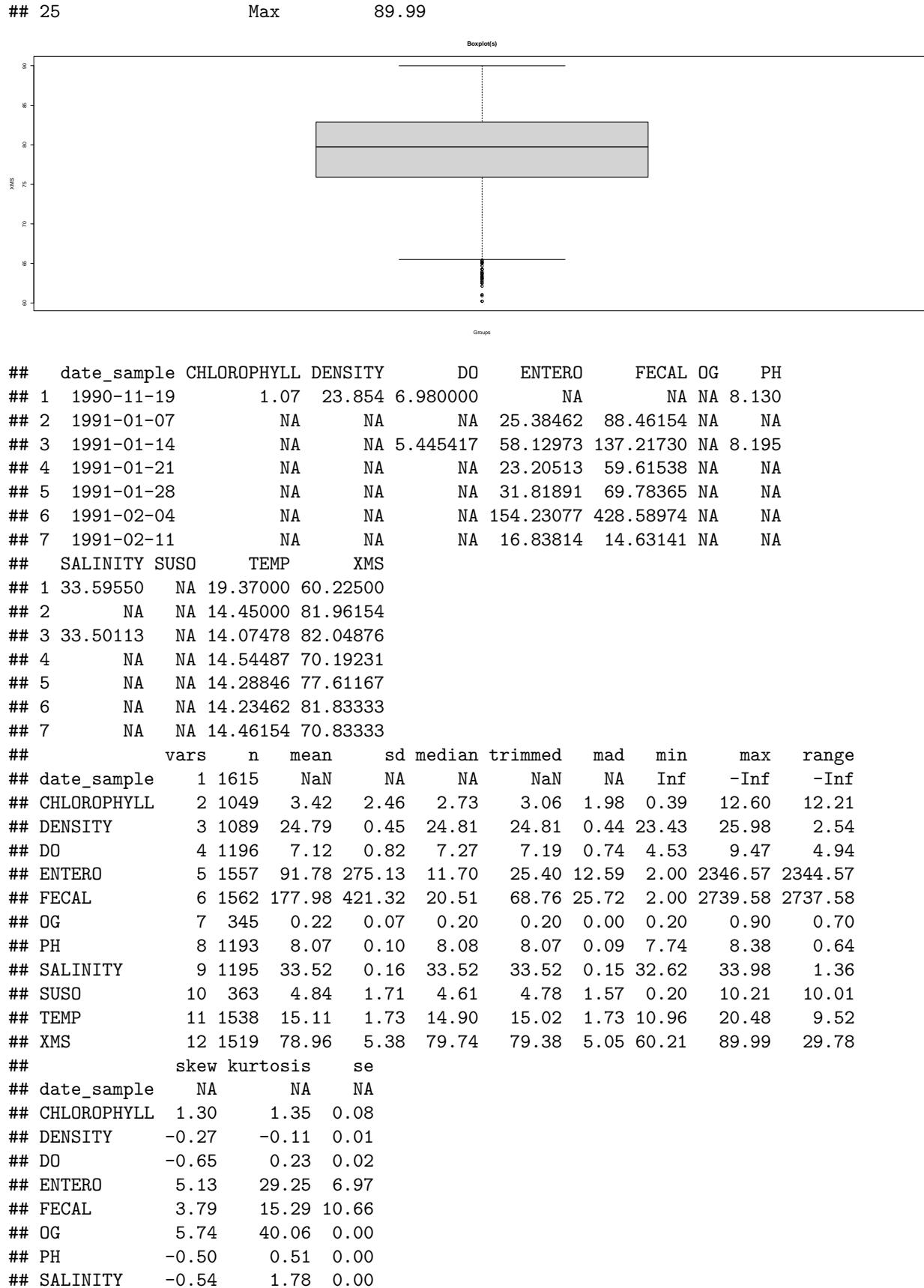
```



```

## [1] 1540    2
##               metric      V2
## 1                   Variable: XMS
## 2           Total N:
## 3           Count      1519
## 4           NA Count      0
## 5           Mean      78.961
## 6           Median      79.742
## 7 Standard Deviation      5.384
## 8           Variance      28.987
## 9           Range      29.778
## 10          Min      60.212
## 11          Max      89.99
## 12      25th Percentile      75.922
## 13      75th Percentile      82.87
## 14 Subset w/o Outliers:
## 15          Count      1511
## 16          %      99.5%
## 17      Outlier %      0.5%
## 18      NA Count      0
## 19          Mean      79.053
## 20          Median      79.78
## 21 Standard Deviation      5.246
## 22          Variance      27.52
## 23          Range      27.089
## 24          Min      62.901

```



```

## SUSO      0.37    0.29  0.09
## TEMP      0.48   -0.16  0.04
## XMS     -0.73    0.33  0.14
##          date_sample parameter      Avg
## Not NA n    36811     36811 35669.000
## NA n        0         0    1142.000
## Not NA %     1         1    0.969
## NA %        0         0    0.031

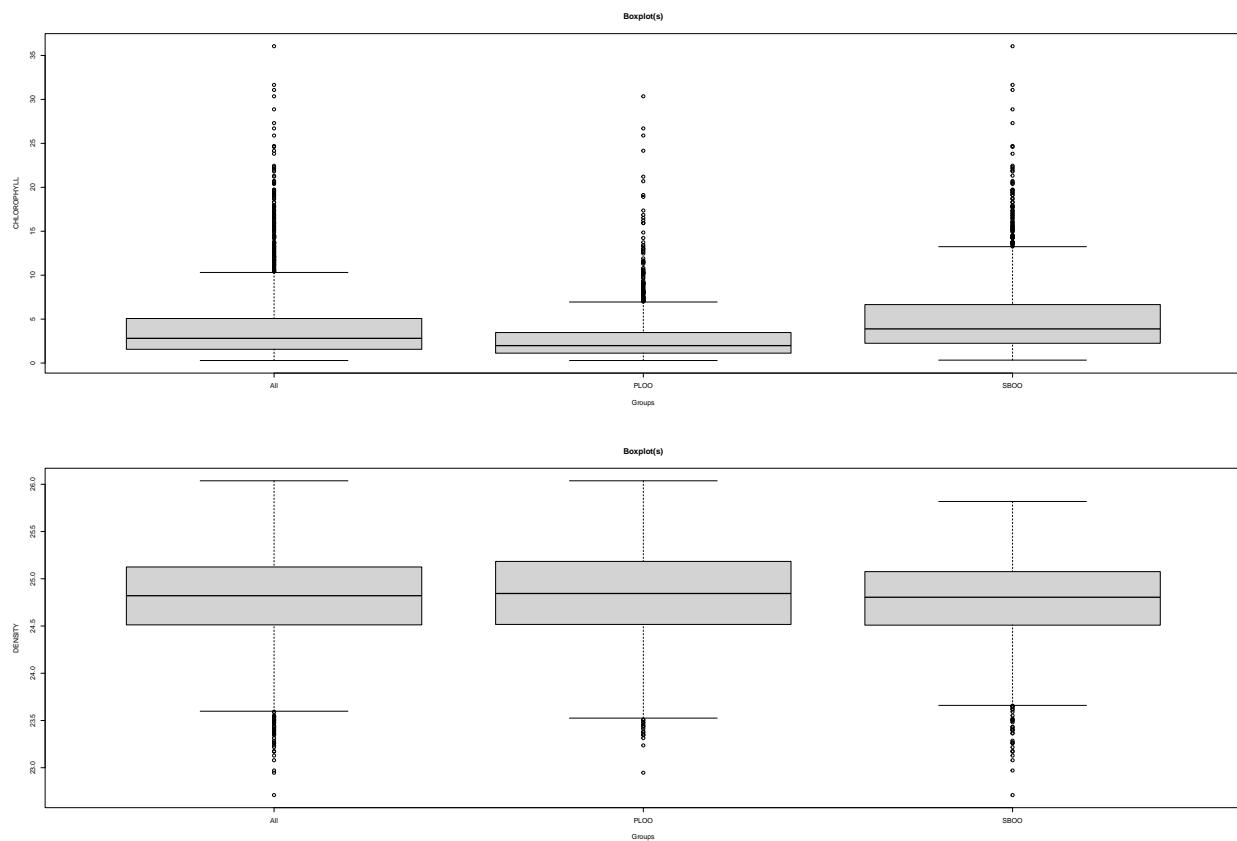
```

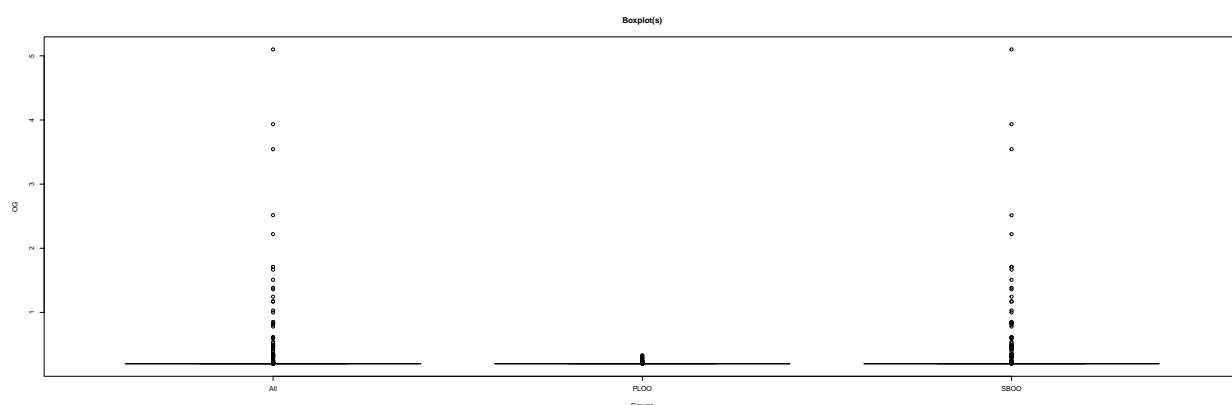
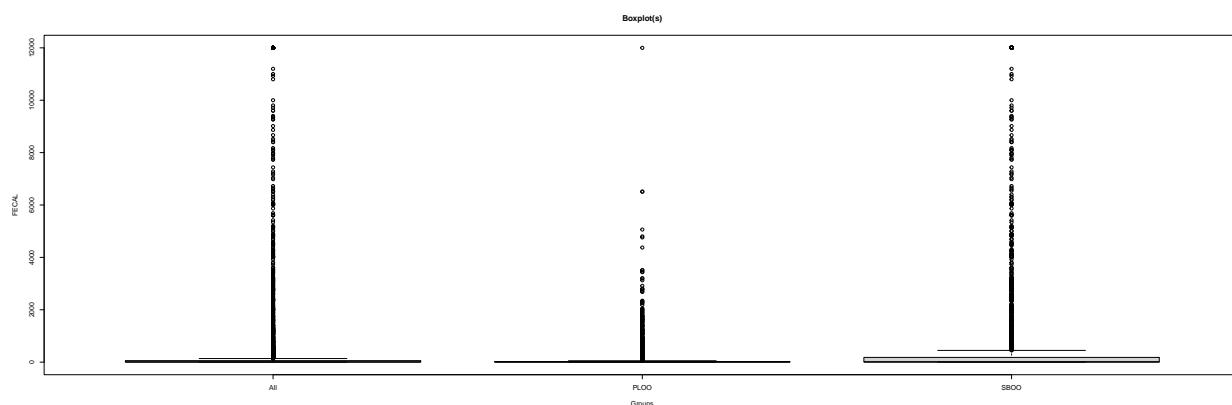
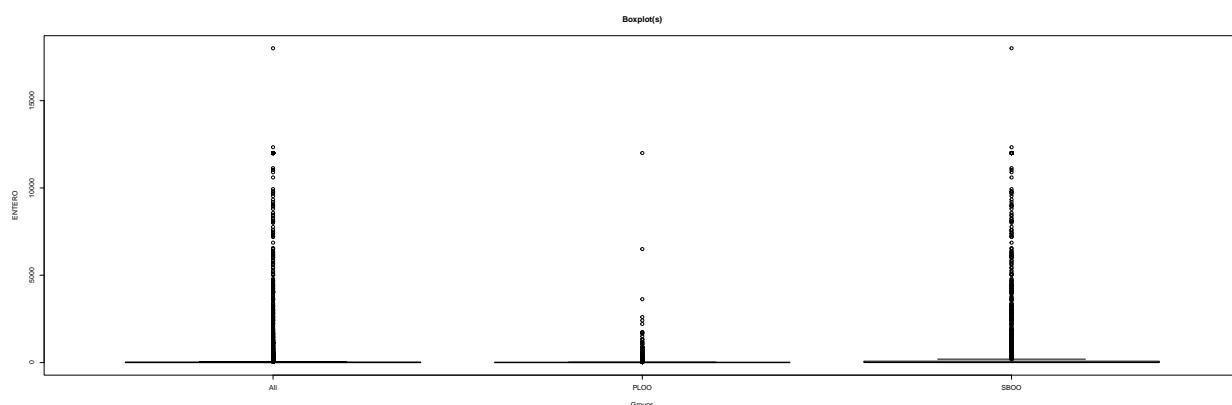
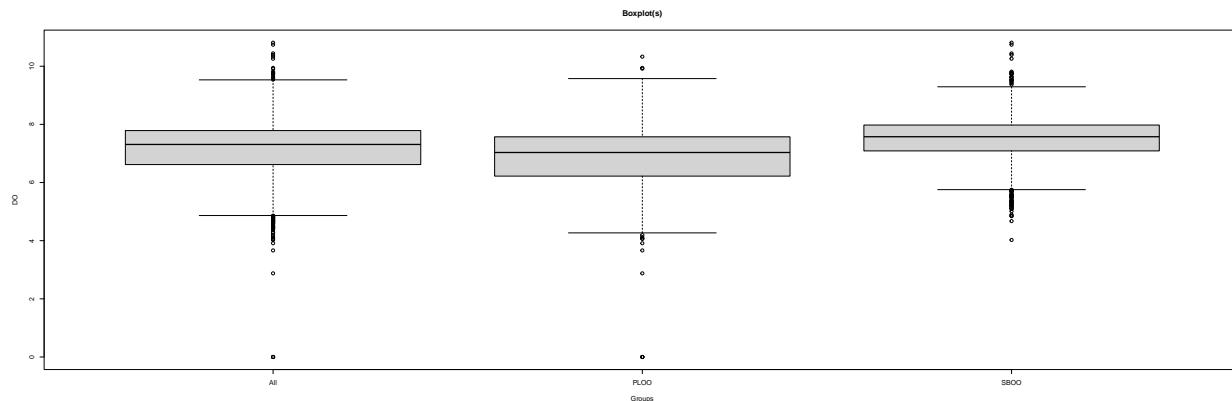
Run custom function on data aggregated by `date_sample`, `project` and `parameter`

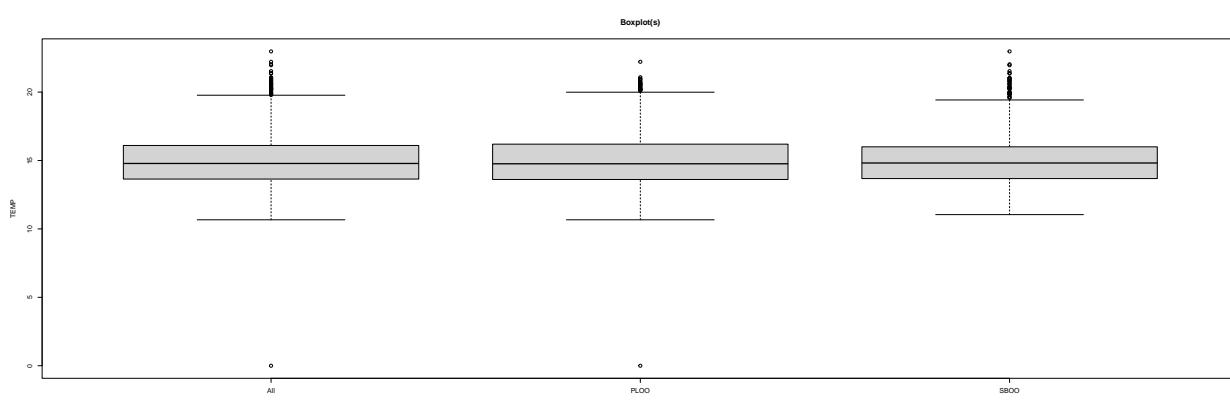
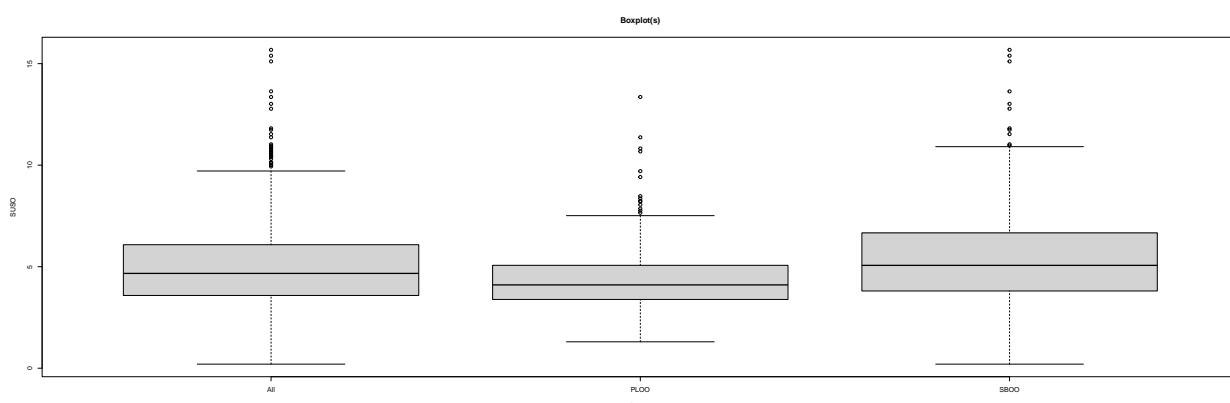
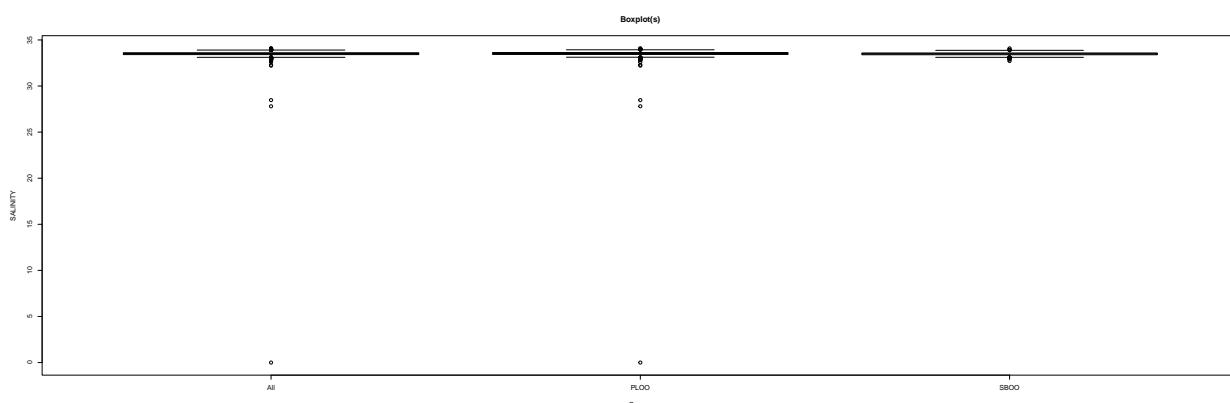
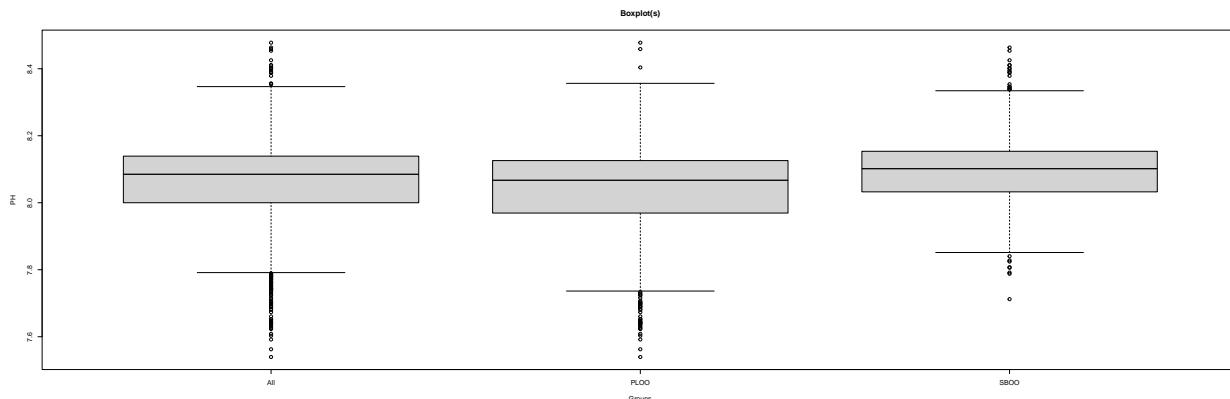
```

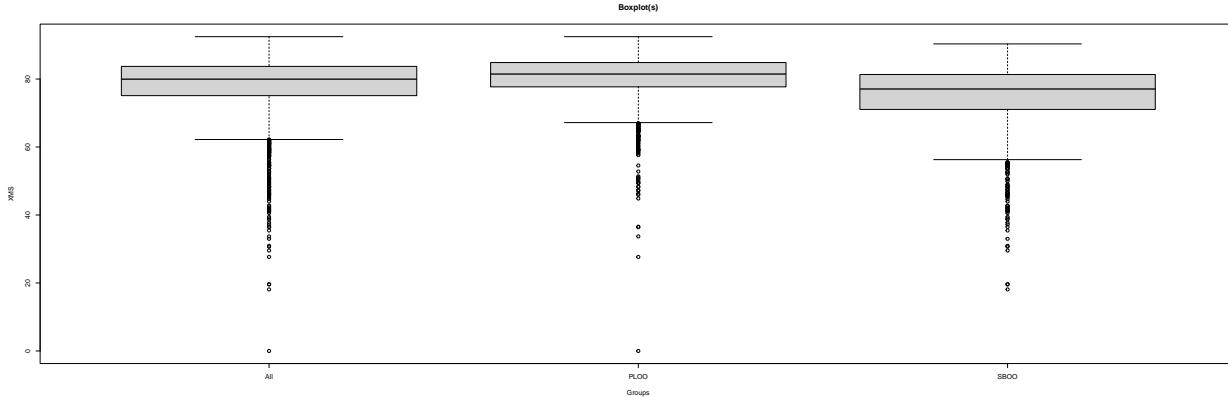
owt_df02_gb09_mrgd = ts_eda(ts_df = owt_df02_gb09,
                             form_lead = c("date_sample", "project"),
                             form_cast = "parameter",
                             param_lst = param_lst02,
                             l1_col = c("project"),
                             l1_param = c("PLOO", "SBOO"),
                             rtn_met = FALSE,
                             box = FALSE,
                             week_agg = FALSE
)

```









```

##   date_sample project CHLOROPHYLL DENSITY      DO      ENTERO      FECAL OG     PH
## 1 1990-11-15    PL00       0.87 23.855 6.55000      NA      NA NA 8.080
## 2 1990-11-15    SB00       1.27 23.853 7.41000      NA      NA NA 8.180
## 3 1991-01-02    PL00       NA     NA     NA 32.05128 125.64103 NA  NA
## 4 1991-01-03    PL00       NA     NA     NA 18.71795 51.28205 NA  NA
## 5 1991-01-07    PL00       NA     NA     NA 106.15385 80.76923 NA  NA
## 6 1991-01-08    PL00       NA     NA     NA 40.00000 40.55556 NA  NA
## 7 1991-01-09    PL00       NA     NA 5.47625 20.33898 40.50847 NA 8.195
##   SALINITY SUSO      TEMP      XMS
## 1 33.61700    NA 19.43000 44.85000
## 2 33.57400    NA 19.31000 75.60000
## 3    NA        NA 14.50769 80.02564
## 4    NA        NA 14.39231 83.89744
## 5    NA        NA 14.54103 78.97436
## 6    NA        NA 13.43000 80.20000
## 7 33.51938    NA 14.26889 86.21111
##   vars      n      mean       sd median trimmed  mad      min      max
##   date_sample 1 7037     NaN      NA     NA     NaN     NA Inf -Inf
##   project*    2 7037     1.43     0.50    1.00    1.42  0.00  1.00  2.00
##   CHLOROPHYLL 3 3023     4.01     3.81    2.81    3.31  2.27  0.29 36.05
##   DENSITY     4 3177    24.79     0.47   24.82   24.81  0.45 22.71 26.04
##   DO          5 3434     7.15     0.97    7.31    7.22  0.82  0.00 10.81
##   ENTERO      6 6776   253.64  1249.91   5.27   18.53  4.84  0.00 18000.00
##   FECAL       7 6594   355.65  1401.82   7.07   48.93  7.51  0.00 12000.00
##   OG          8  937     0.24     0.29    0.20    0.20  0.00  0.20  5.10
##   PH          9 3396     8.07     0.11    8.08    8.07  0.10  7.54  8.48
##   SALINITY    10 3434    33.50     0.61   33.52   33.51  0.15  0.00 34.10
##   SUSO        11  977     4.99     2.16    4.67    4.80  1.80  0.20 15.68
##   TEMP         12 4524    14.97     1.91   14.78   14.87  1.80  0.00 22.97
##   XMS         13 4494    78.31     8.29   79.96   79.37  6.18  0.00 92.46
##   range      skew kurtosis      se
##   date_sample -Inf     NA      NA     NA
##   project*     1.00    0.26   -1.93  0.01
##   CHLOROPHYLL 35.76    2.56    9.61  0.07
##   DENSITY      3.33   -0.46    0.38  0.01
##   DO          10.81   -0.89    3.49  0.02
##   ENTERO      18000.00   7.41   61.01 15.18
##   FECAL       12000.00   6.25   43.27 17.26
##   OG          4.90   10.89  144.21  0.01
##   PH          0.94   -0.67    1.45  0.00

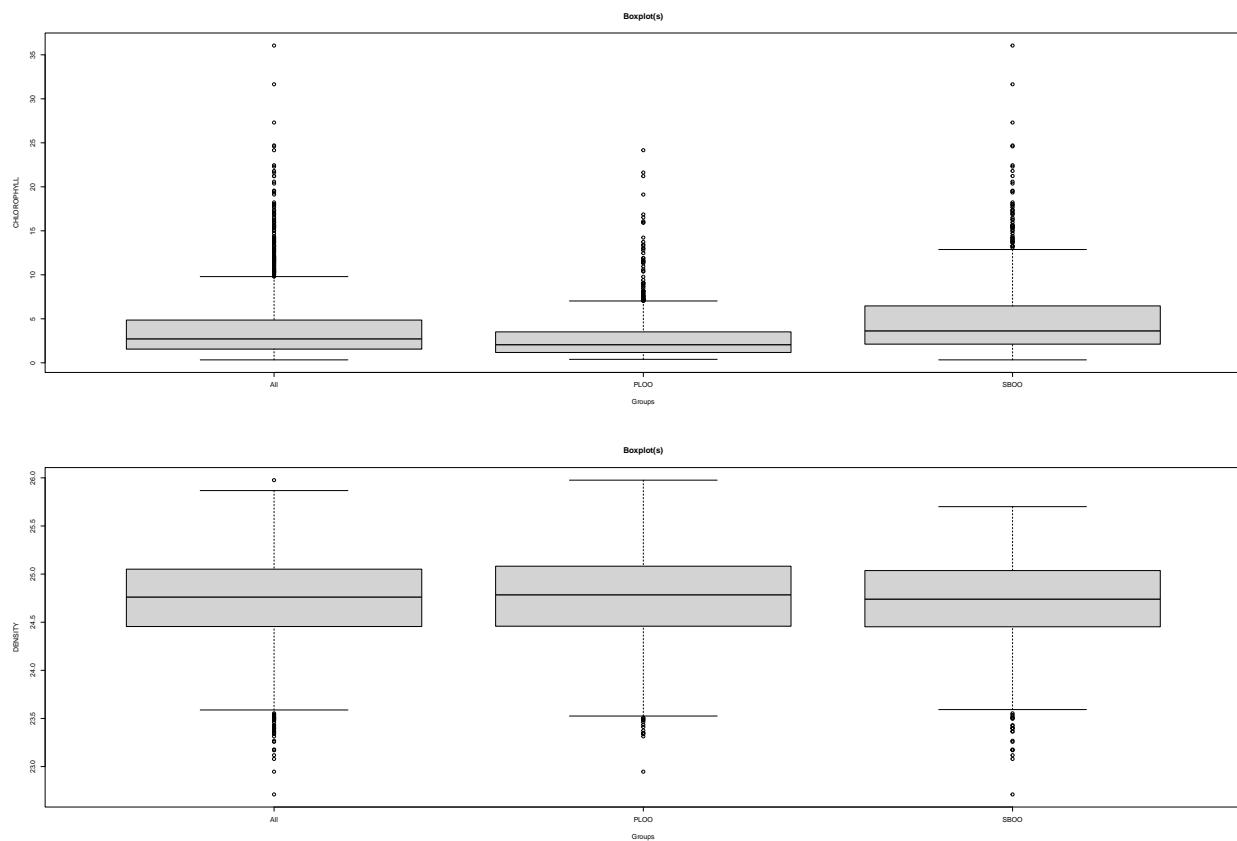
```

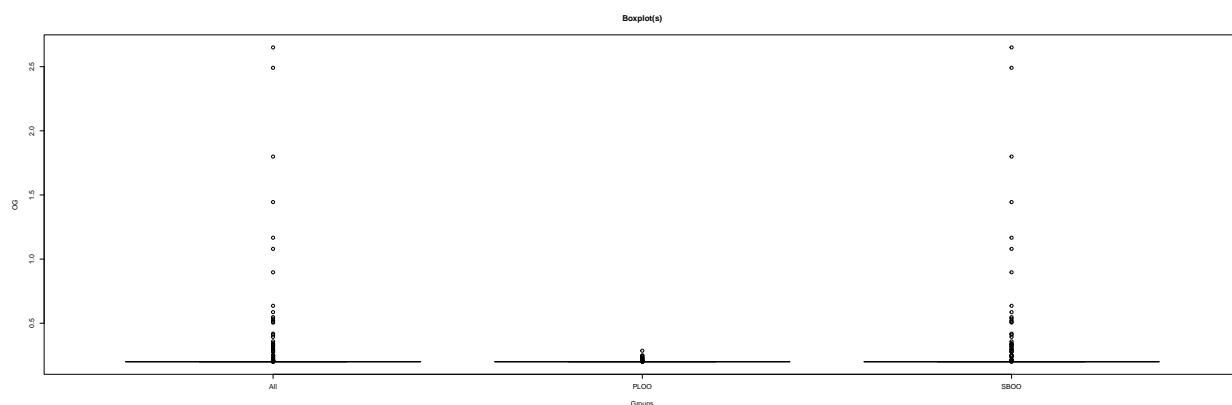
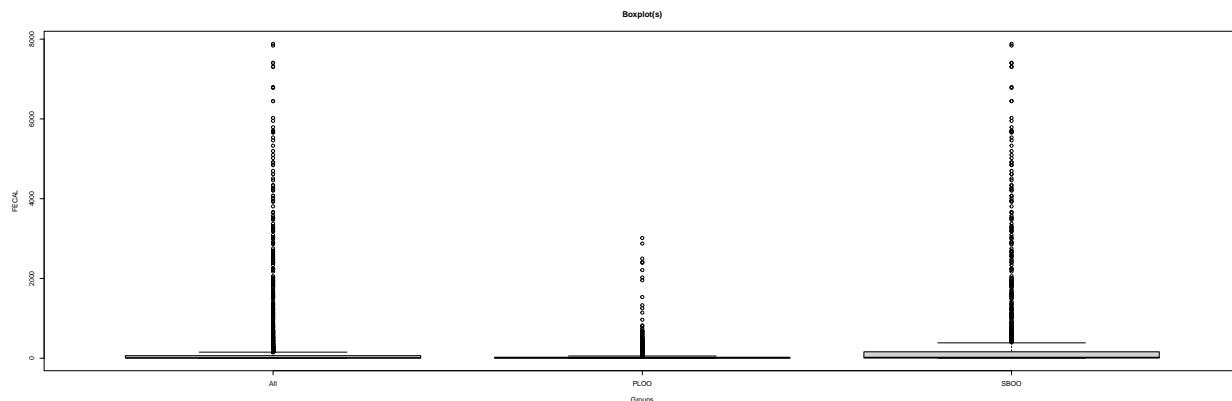
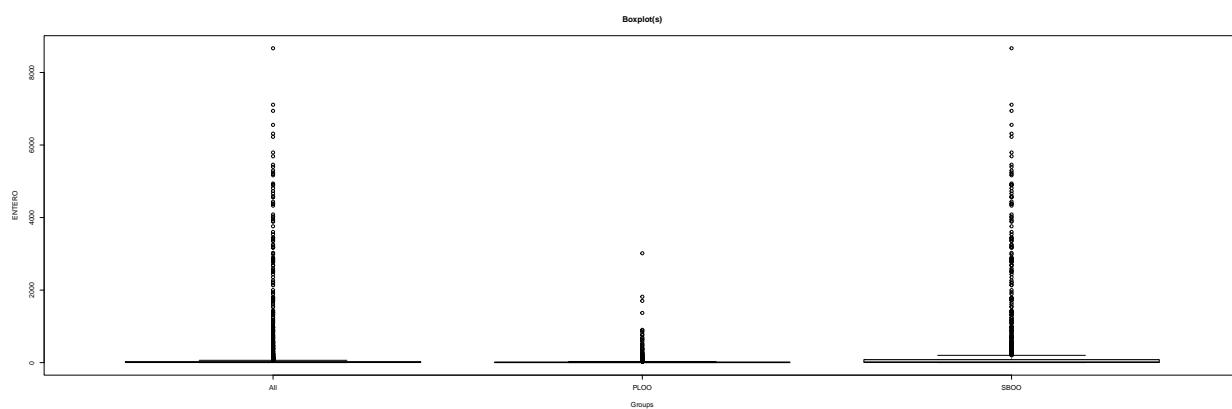
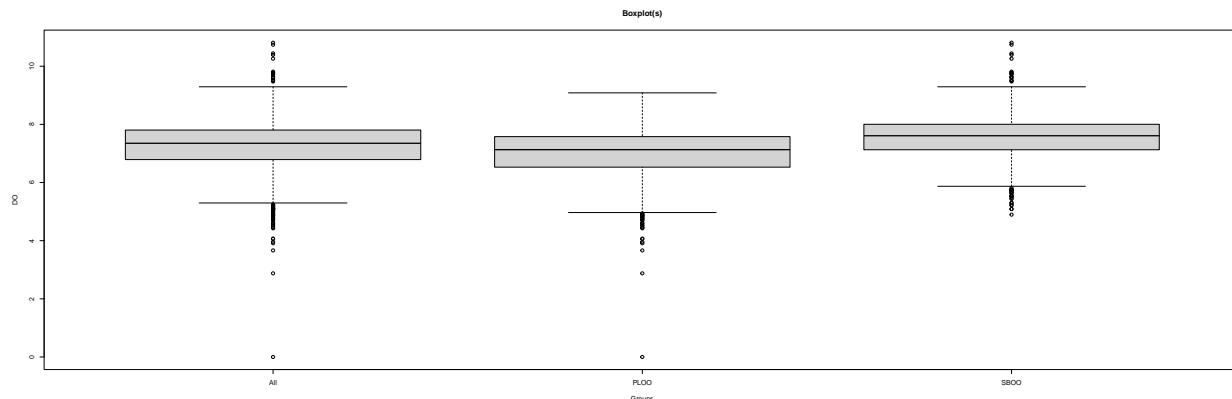
```

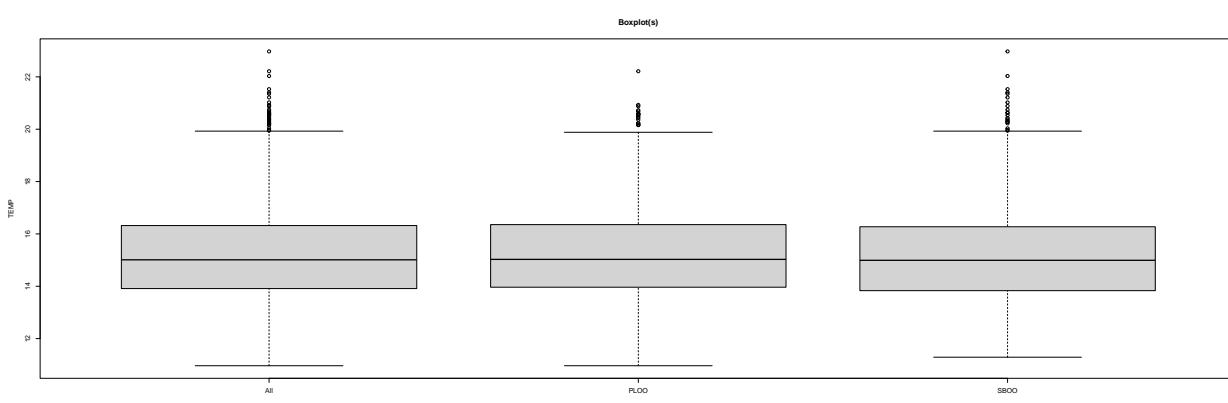
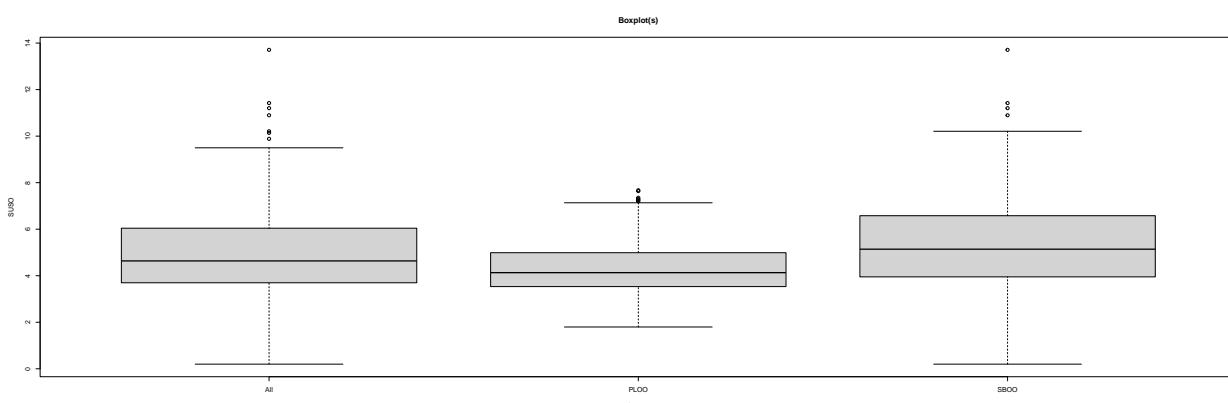
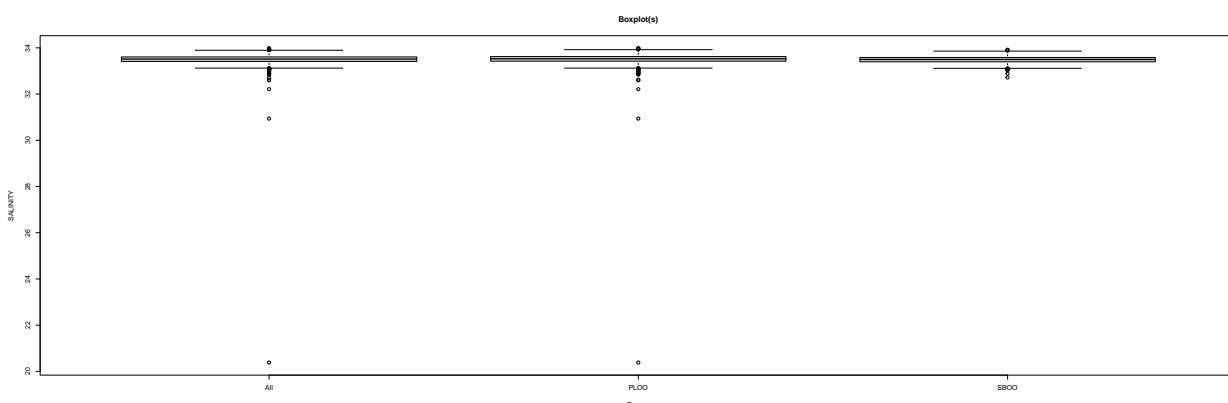
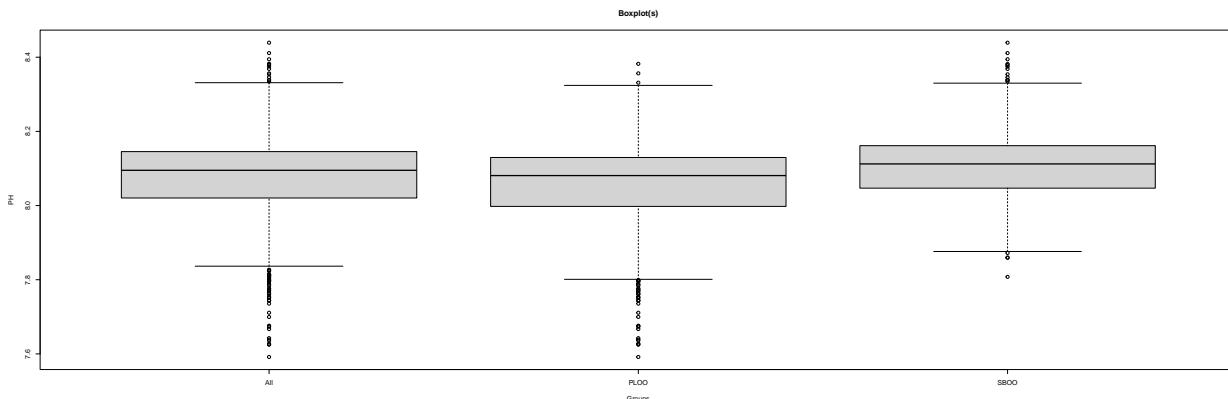
## SALINITY      34.10 -48.98  2674.00  0.01
## SUSO         15.48   1.09     2.23  0.07
## TEMP         22.97   0.43     0.95  0.03
## XMS          92.46  -2.00     7.54  0.12
##           date_sample project parameter      Avg
## Not NA n    48395    48395     48395 47222.0000
## NA n        0        0        0    1173.0000
## Not NA %    1        1        1    0.9758
## NA %        0        0        0    0.0242

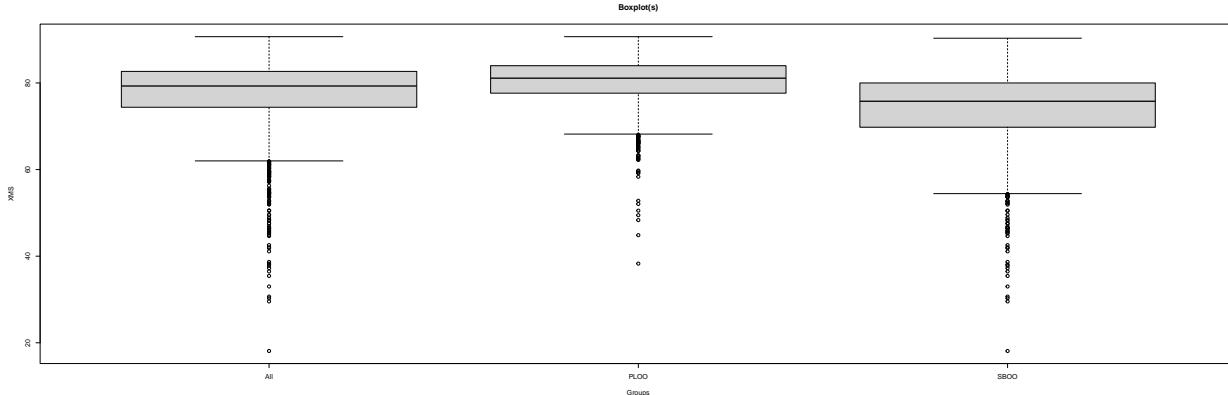
owt_df02_gb09_wkly = ts_eda(ts_df = owt_df02_gb09,
                             form_lead = c("date_sample", "project"),
                             form_cast = "parameter",
                             param_lst = param_lst02,
                             l1_col = c("project"),
                             l1_param = c("PLOO", "SBOO"),
                             rtn_met = FALSE,
                             box = FALSE,
                             week_agg = TRUE
)

```









```

##   date_sample project CHLOROPHYLL DENSITY      DO     ENTERO     FECAL OG     PH
## 1 1990-11-19    PLOO       0.87 23.855 6.550000      NA      NA NA 8.080
## 2 1990-11-19    SB00       1.27 23.853 7.410000      NA      NA NA 8.180
## 3 1991-01-07    PLOO       NA     NA     NA 25.38462 88.46154 NA     NA
## 4 1991-01-14    PLOO       NA     NA     NA 5.445417 58.12973 137.21730 NA 8.195
## 5 1991-01-21    PLOO       NA     NA     NA 23.20513 59.61538 NA     NA
## 6 1991-01-28    PLOO       NA     NA     NA 31.81891 69.78365 NA     NA
## 7 1991-02-04    PLOO       NA     NA     NA 154.23077 428.58974 NA     NA
##   SALINITY SUSO      TEMP     XMS
## 1 33.61700    NA 19.43000 44.85000
## 2 33.57400    NA 19.31000 75.60000
## 3     NA     NA 14.45000 81.96154
## 4 33.50113    NA 14.07478 82.04876
## 5     NA     NA 14.54487 70.19231
## 6     NA     NA 14.28846 77.61167
## 7     NA     NA 14.23462 81.83333
##   vars     n   mean      sd median trimmed   mad   min   max range
##   date_sample 1 2802    NaN     NA     NA    NaN    NA Inf -Inf -Inf
##   project*    2 2802    1.43    0.49    1.00   1.41   0.00  1.00  2.00  1.00
##   CHLOROPHYLL 3 1967    3.89    3.68    2.71   3.19   2.09  0.33 36.05 35.72
##   DENSITY     4 2000    24.73    0.47   24.76  24.75   0.44 22.71 25.98  3.27
##   DO          5 2101    7.24    0.87    7.35   7.29   0.75  0.00 10.81 10.81
##   ENTERO      6 2786   169.27 699.74    7.58  17.88   7.65  2.00 8668.00 8666.00
##   FECAL       7 2786   233.75 815.20    9.45  42.01 10.37  2.00 7881.92 7879.92
##   OG          8  351     0.24    0.23    0.20   0.20   0.00  0.20  2.65  2.45
##   PH          9 2105     8.08    0.10    8.10   8.09   0.09  7.59  8.44  0.85
##   SALINITY    10 2097    33.50    0.33   33.51  33.51   0.14 20.38 33.98 13.60
##   SUSO        11  369     4.91    1.85    4.64   4.80   1.61  0.20 13.71 13.51
##   TEMP         12 2616    15.22    1.83   15.01  15.11   1.77 10.96 22.97 12.01
##   XMS         13 2606    77.43    8.03   79.29  78.55   5.84 18.11 90.68 72.57
##   skew kurtosis se
##   date_sample    NA      NA NA
##   project*      0.30    -1.91 0.01
##   CHLOROPHYLL  2.63    10.24 0.08
##   DENSITY      -0.47     0.46 0.01
##   DO          -0.76     3.34 0.02
##   ENTERO       6.11    42.36 13.26
##   FECAL        5.43    33.41 15.44
##   OG          7.92    70.11 0.01
##   PH          -0.71     1.79 0.00

```

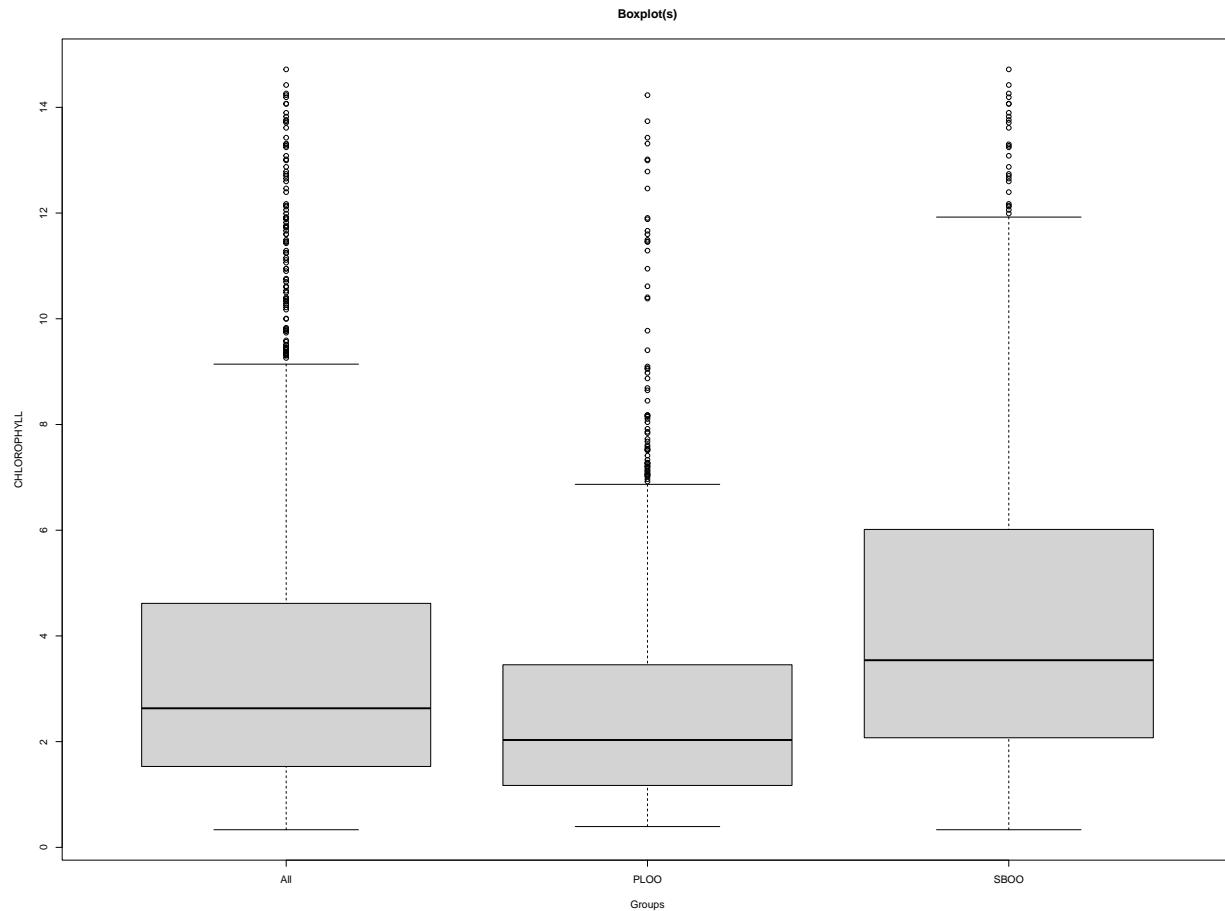
```

## SALINITY      -29.25   1137.46  0.01
## SUSO          0.79     1.80   0.10
## TEMP          0.61     0.27   0.04
## XMS           -1.91    5.85   0.16
##               date_sample project parameter      Avg
## Not NA n      48395    48395    48395 47222.0000
## NA n          0         0         0     1173.0000
## Not NA %      1         1         1     0.9758
## NA %          0         0         0     0.0242

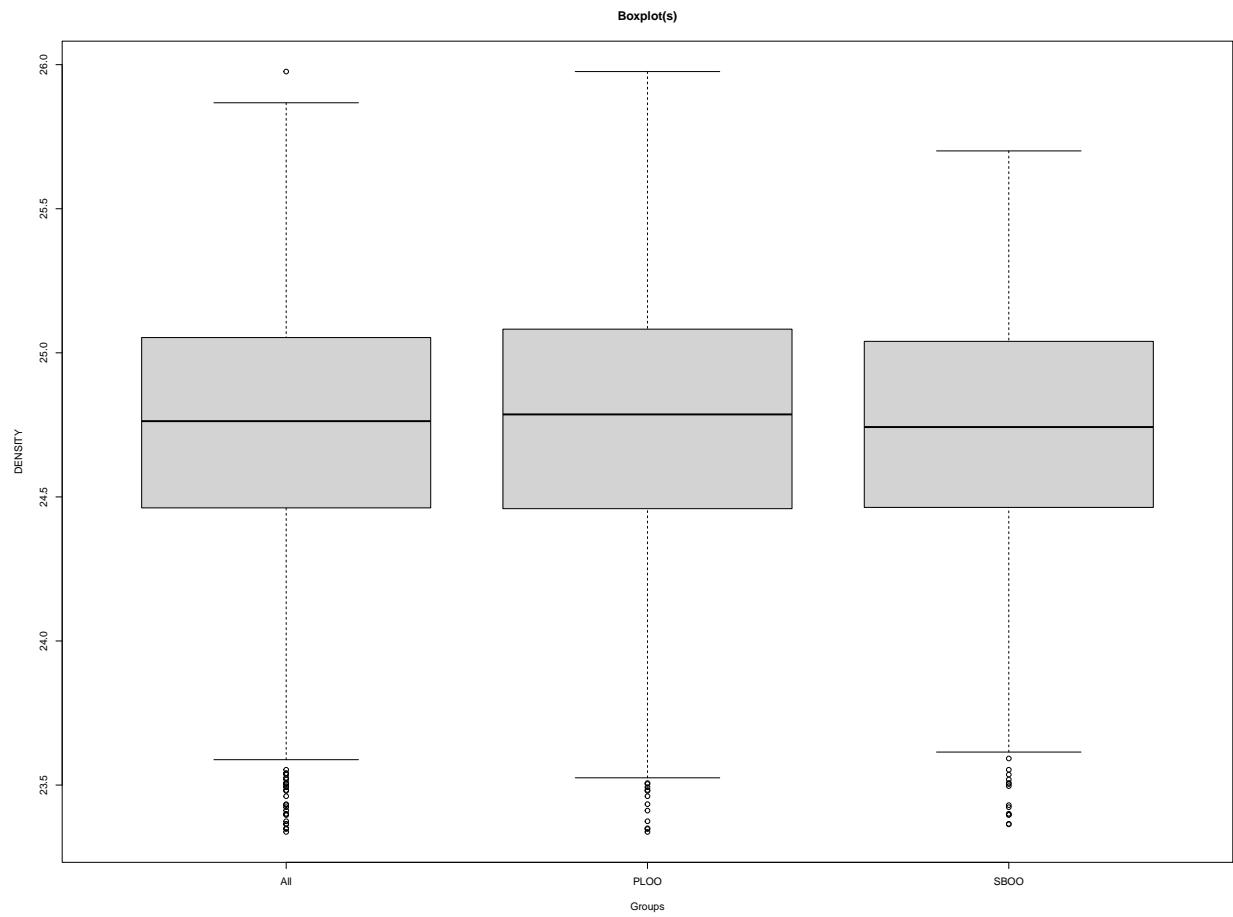
owt_df02_gb09_wkly02 = ts_eda(ts_df = owt_df02_gb09,
                               form_lead = c("date_sample", "project"),
                               form_cast = "parameter",
                               param_lst = param_lst02,
                               l1_col = c("project"),
                               l1_param = c("PLOO", "SBOO"),
                               rtn_met = FALSE,
                               box = FALSE,
                               week_agg = TRUE,
                               out_rm = TRUE
)

```

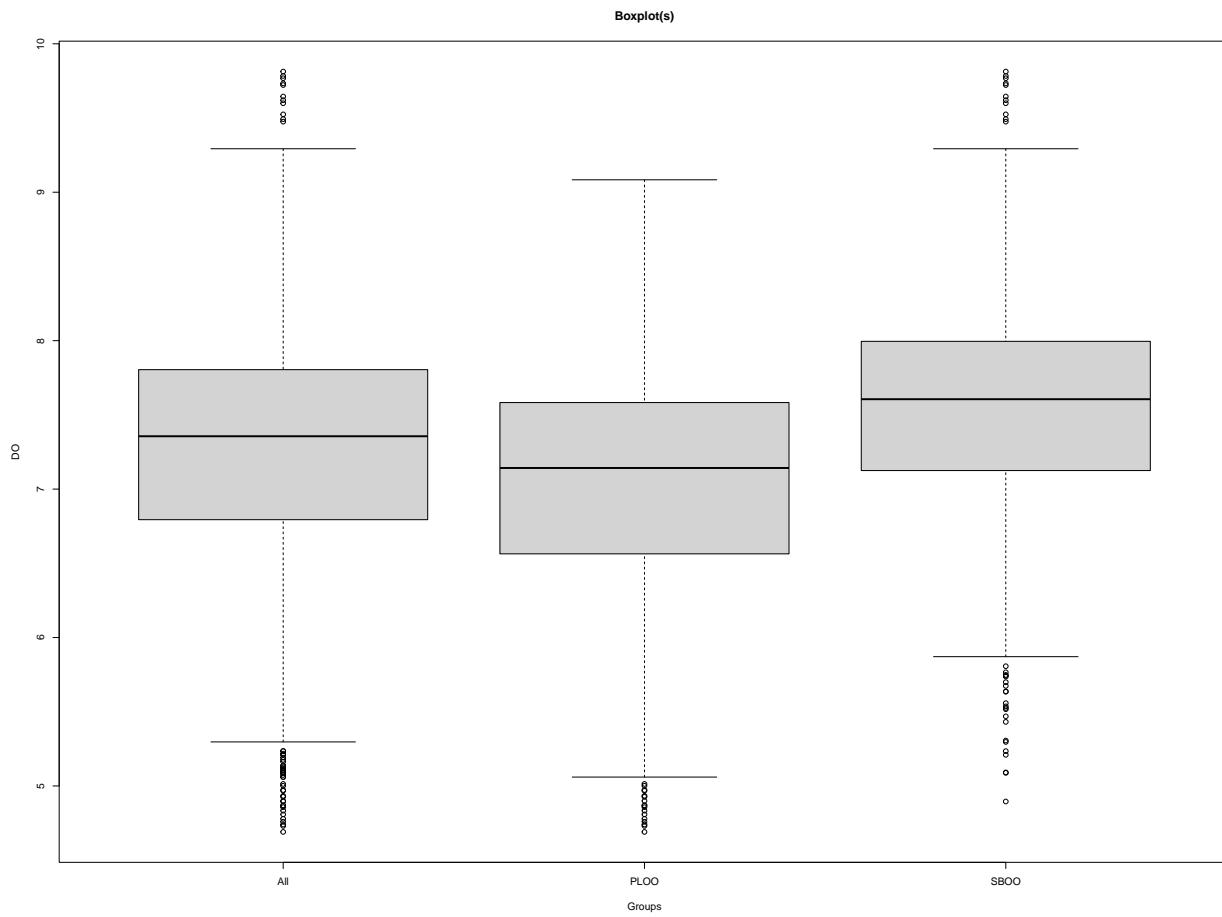
```
## [1] 1967      3
```



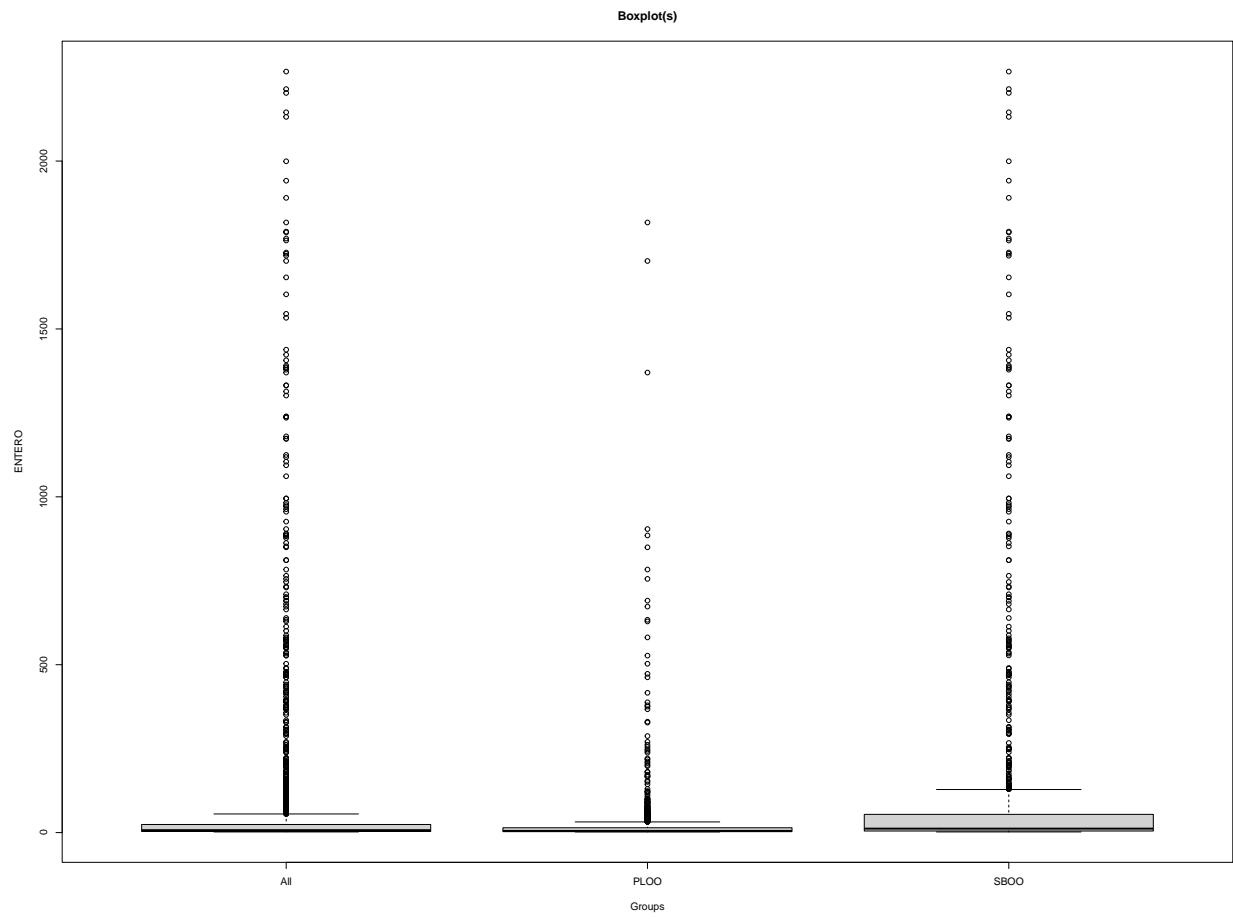
```
## [1] 2000      3
```



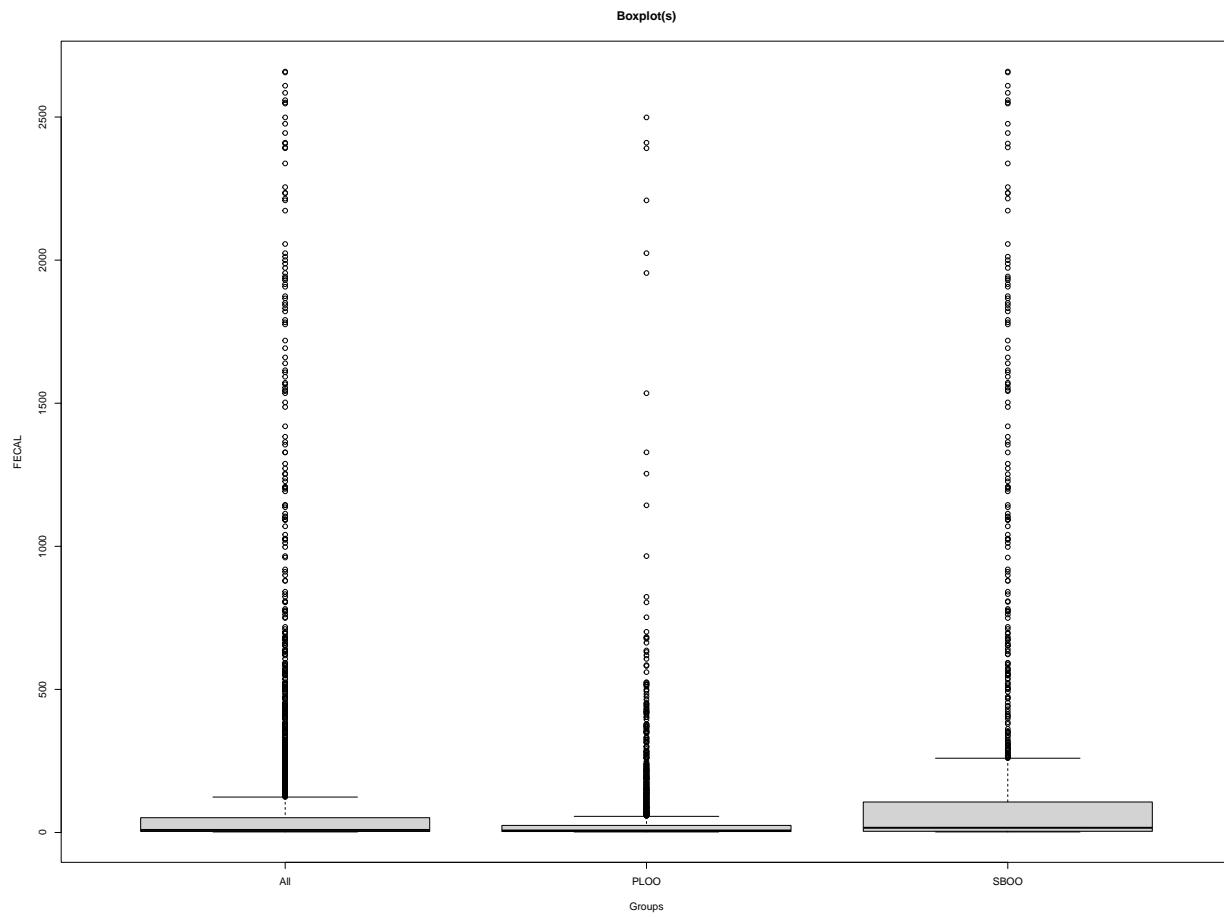
```
## [1] 2101      3
```



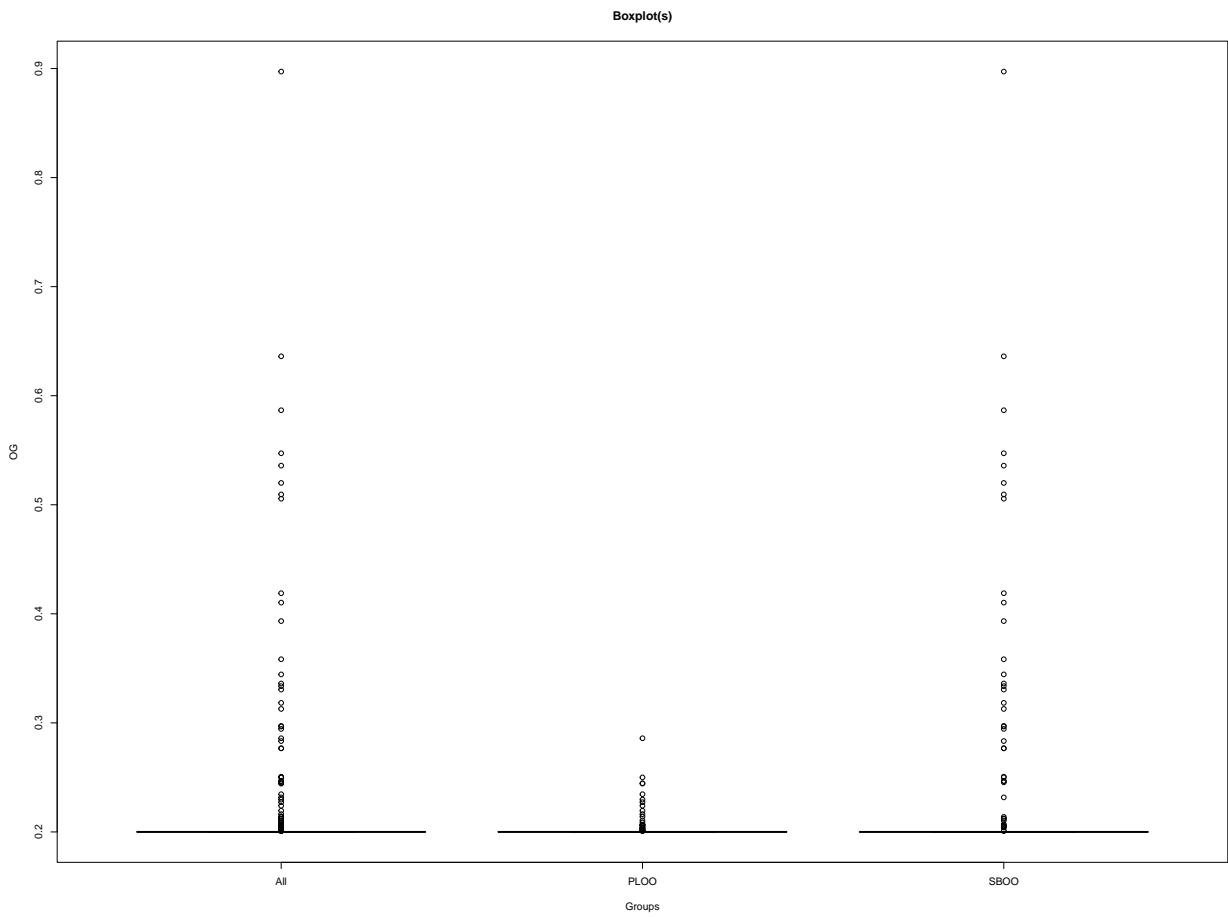
```
## [1] 2786      3
```



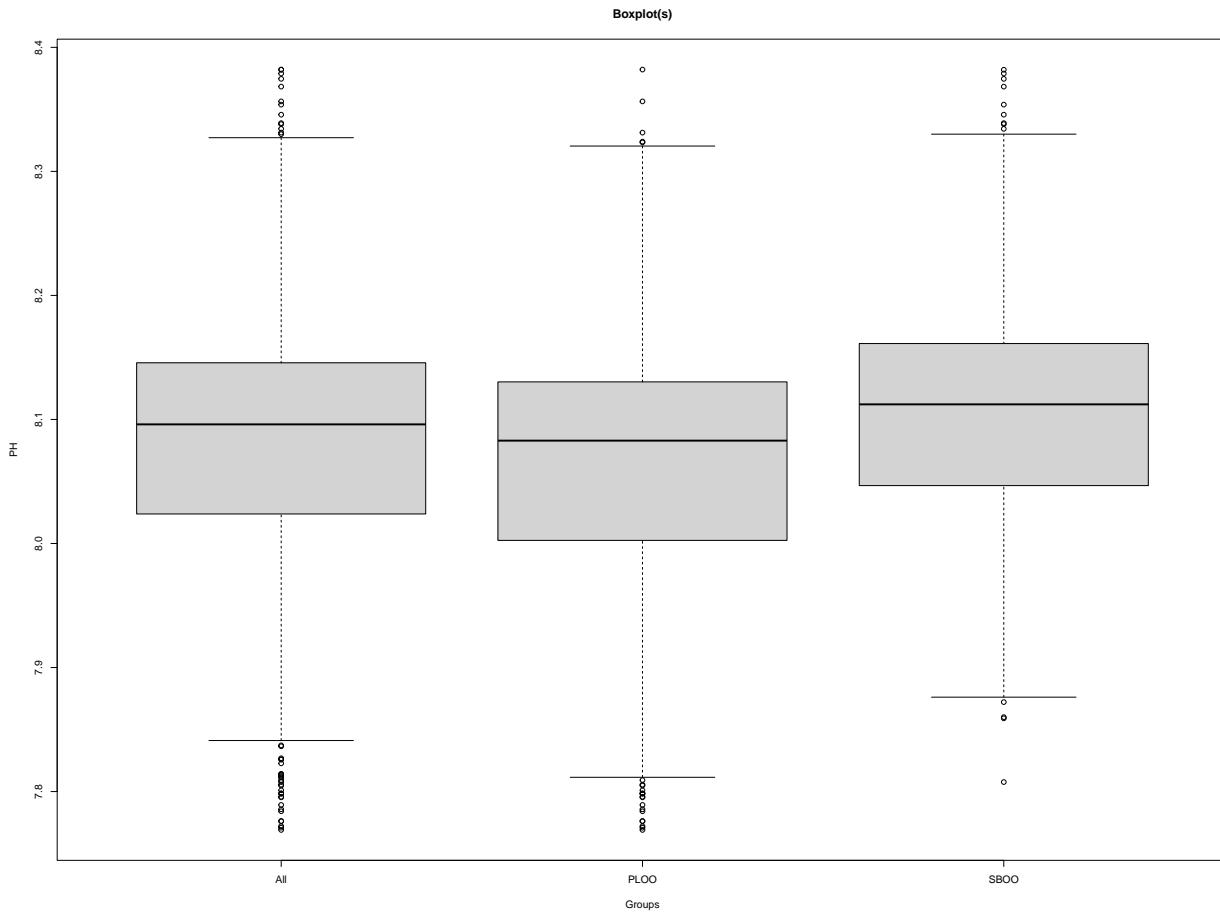
```
## [1] 2786      3
```



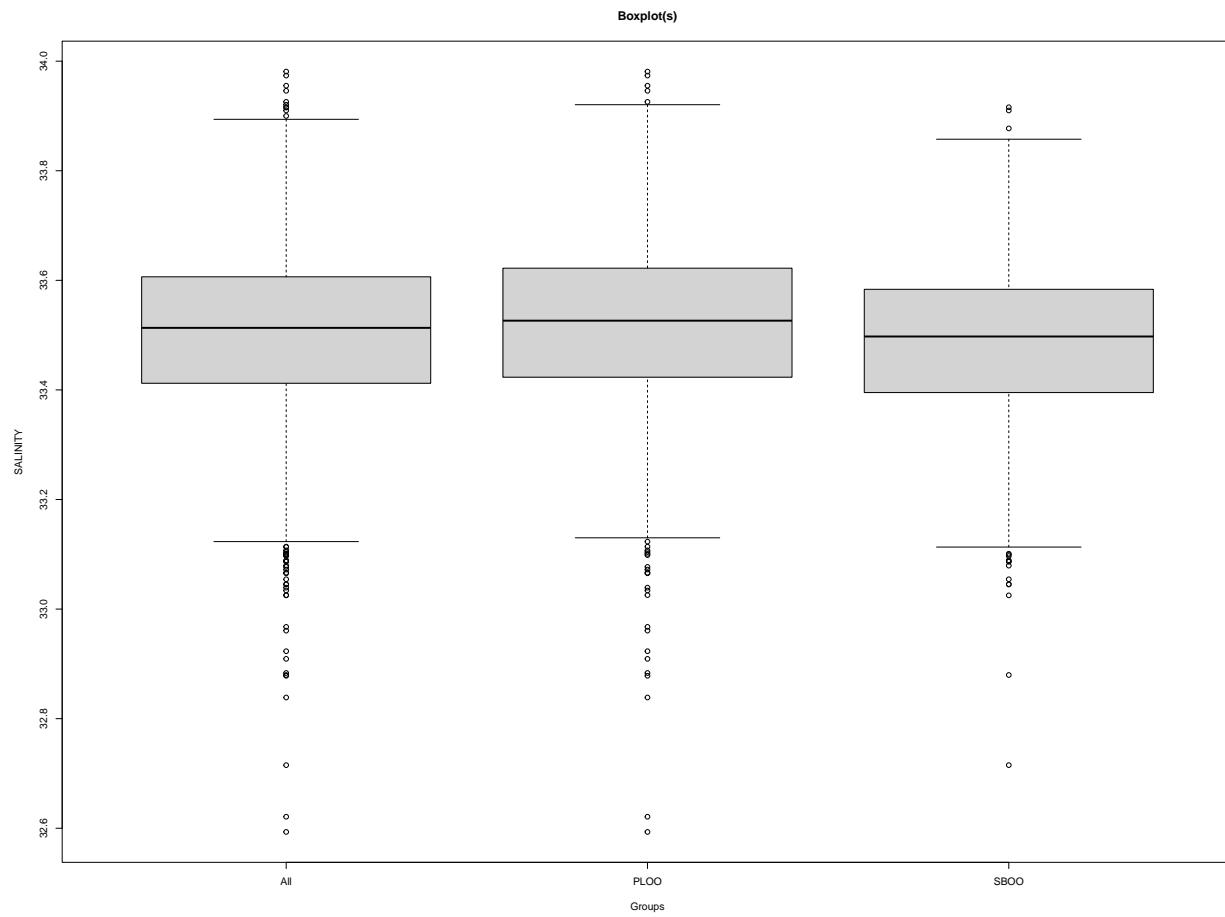
```
## [1] 351    3
```



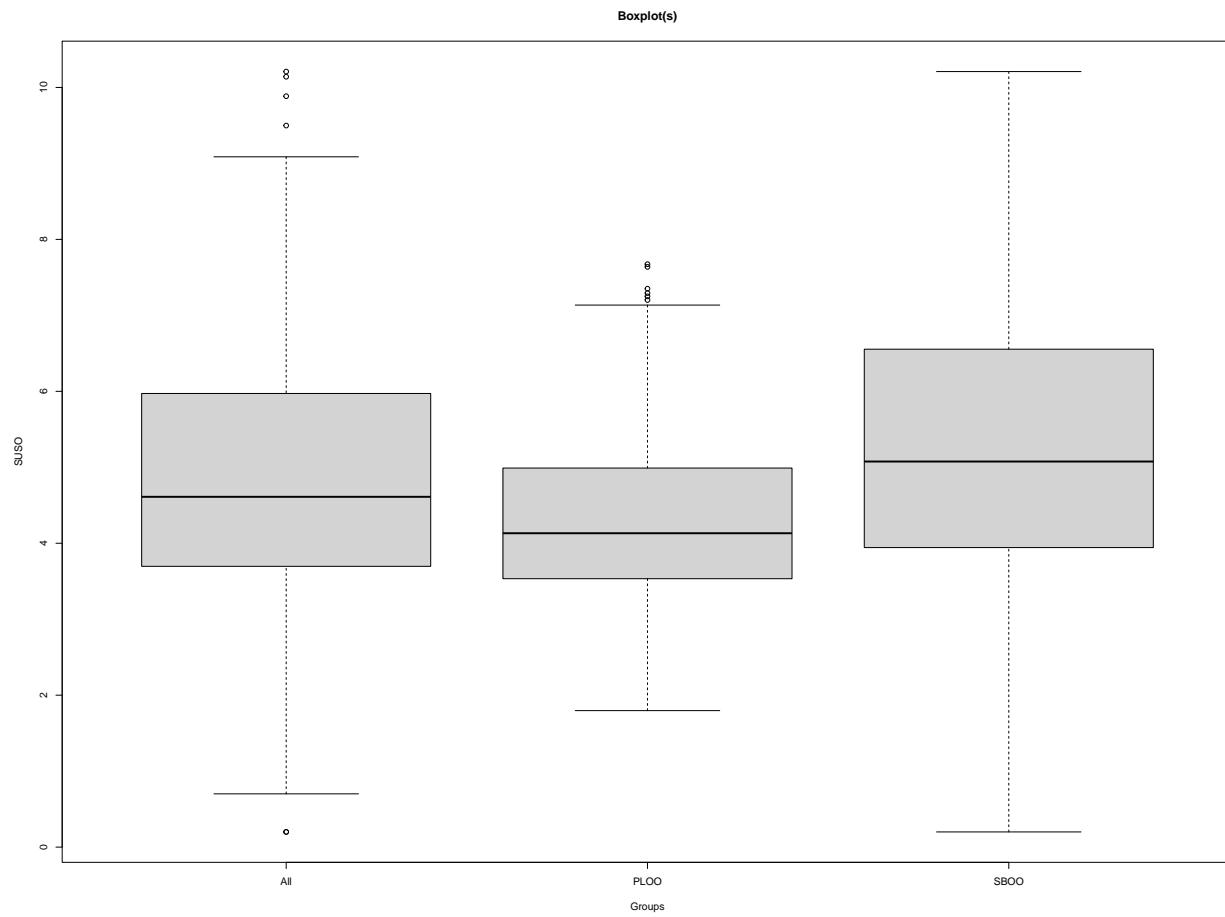
```
## [1] 2105      3
```



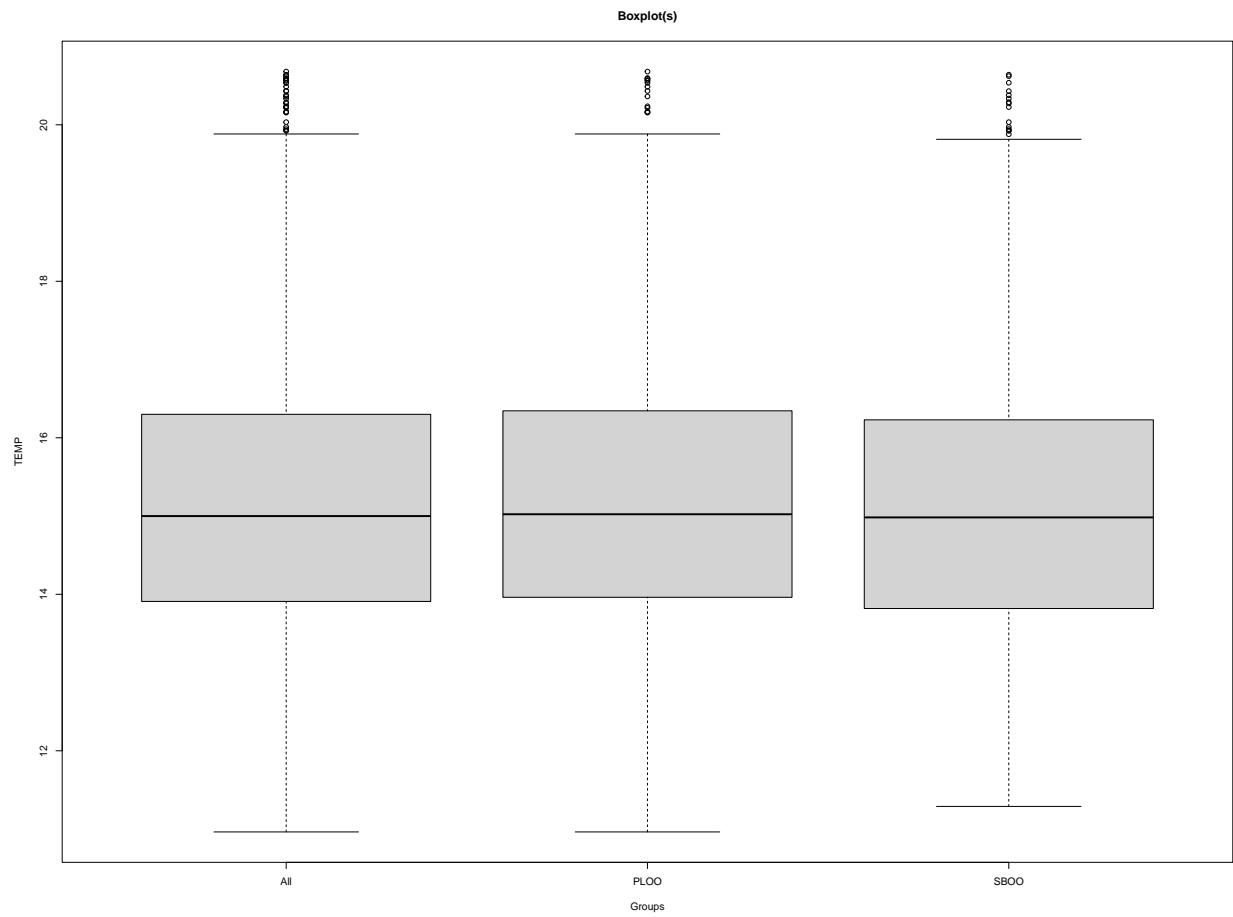
```
## [1] 2097      3
```



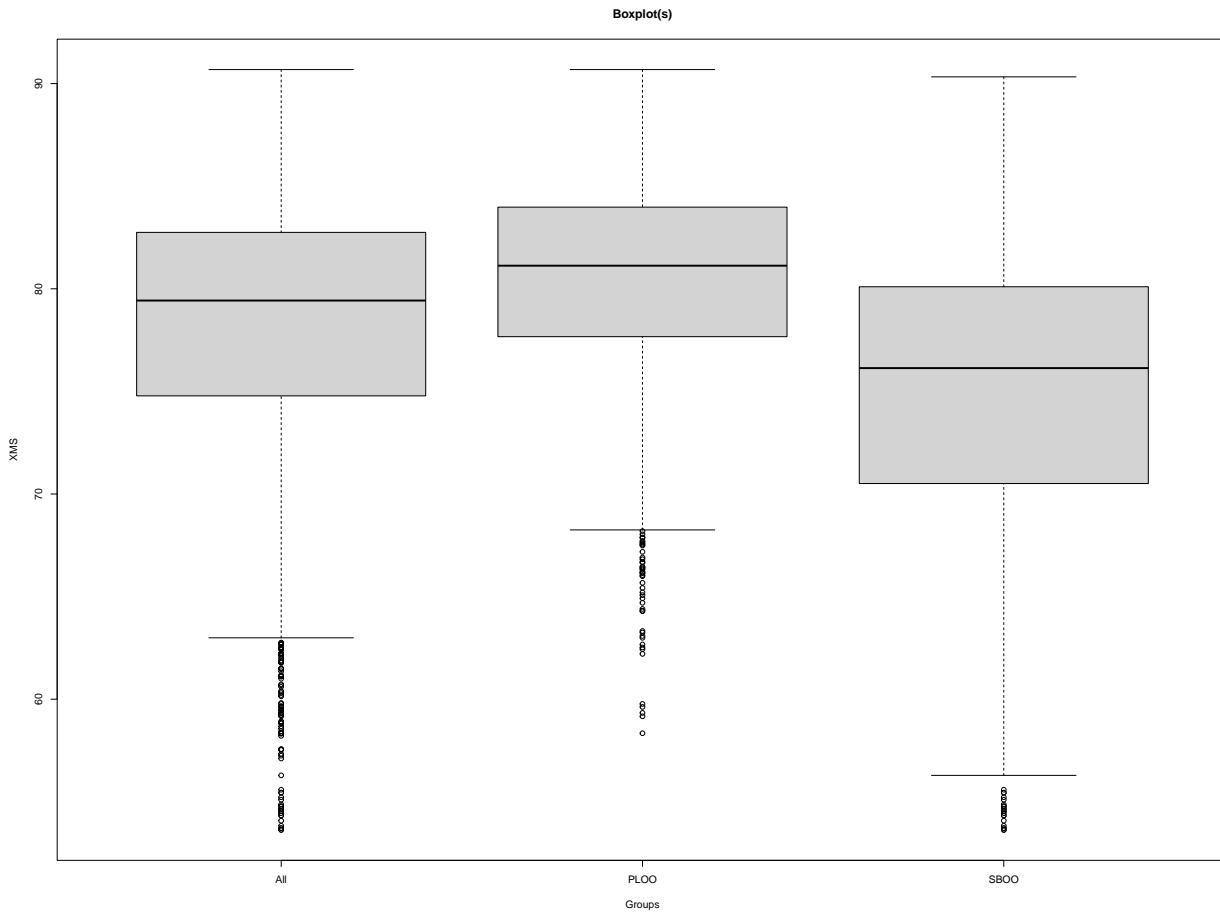
```
## [1] 369    3
```



```
## [1] 2616     3
```



```
## [1] 2606      3
```



```

##   date_sample project CHLOROPHYLL DENSITY      DO      ENTERO      FECAL OG     PH
## 1 1990-11-19    PLOO       0.87 23.855 6.550000      NA      NA NA 8.080
## 2 1990-11-19    SB00       1.27 23.853 7.410000      NA      NA NA 8.180
## 3 1991-01-07    PLOO       NA      NA      NA 25.38462 88.46154 NA     NA
## 4 1991-01-14    PLOO       NA      NA      NA 5.445417 58.12973 137.21730 NA 8.195
## 5 1991-01-21    PLOO       NA      NA      NA 23.20513 59.61538 NA     NA
## 6 1991-01-28    PLOO       NA      NA      NA 31.81891 69.78365 NA     NA
## 7 1991-02-04    PLOO       NA      NA      NA 154.23077 428.58974 NA     NA
##   SALINITY SUSO      TEMP      XMS
## 1 33.61700  NA 19.43000      NA
## 2 33.57400  NA 19.31000 75.60000
## 3      NA  NA 14.45000 81.96154
## 4 33.50113  NA 14.07478 82.04876
## 5      NA  NA 14.54487 70.19231
## 6      NA  NA 14.28846 77.61167
## 7      NA  NA 14.23462 81.83333
##           vars   n   mean      sd median trimmed   mad   min   max range
## date_sample    1 2799    NaN     NA     NA    NaN    NA Inf -Inf -Inf
## project*      2 2799    1.43    0.49    1.00    1.41  0.00  1.00  2.00  1.00
## CHLOROPHYLL  3 1919    3.51    2.75    2.63    3.06  1.98  0.33 14.72 14.38
## DENSITY       4 1990    24.74    0.46   24.76   24.76  0.44  23.34  25.98  2.64
## DO            5 2082     7.26    0.81    7.36    7.30  0.74  4.69  9.81  5.12
## ENTERO        6 2713    66.28   228.34    7.25   14.79  7.15  2.00 2266.55 2264.55
## FECAL         7 2711   115.89   335.46    8.89   33.22  9.50  2.00 2658.87 2656.87

```

```

## OG          8  345   0.22   0.07   0.20    0.20  0.00   0.20    0.90   0.70
## PH          9 2079   8.08   0.10   8.10    8.09  0.09   7.77    8.38   0.61
## SALINITY    10 2094  33.50   0.16  33.51   33.51  0.14  32.59   33.98   1.39
## SUSO         11  365   4.83   1.71   4.61    4.77  1.57   0.20   10.21  10.01
## TEMP         12 2603  15.19   1.78  15.00   15.10  1.76  10.96   20.68   9.72
## XMS          13 2560  78.03   6.67  79.43   78.77  5.69  53.63   90.68  37.05
##              skew kurtosis   se
## date_sample   NA       NA     NA
## project*      0.30    -1.91  0.01
## CHLOROPHYLL  1.53     2.20  0.06
## DENSITY       -0.34    0.08  0.01
## DO            -0.46    0.44  0.02
## ENTERO        5.78    37.71 4.38
## FECAL         4.74    24.82 6.44
## OG            5.74    40.06 0.00
## PH            -0.36    0.45  0.00
## SALINITY      -0.53    1.74  0.00
## SUSO          0.37     0.30  0.09
## TEMP          0.50    -0.07  0.03
## XMS          -1.07    1.10  0.13
##              date_sample project parameter      Avg
## Not NA n     48395   48395   48395 47222.0000
## NA n          0       0       0     1173.0000
## Not NA %      1       1       1     0.9758
## NA %          0       0       0     0.0242

owt_df02_gb09_wkly03 <- owt_df02_gb09_wkly
print(head(owt_df02_gb09_wkly03, 7))

```

```

##   date_sample project CHLOROPHYLL DENSITY      DO      ENTERO      FECAL OG      PH
## 1 1990-11-19    PLOO      0.87 23.855 6.550000      NA      NA NA 8.080
## 2 1990-11-19    SBOO      1.27 23.853 7.410000      NA      NA NA 8.180
## 3 1991-01-07    PLOO      NA     NA      NA 25.38462 88.46154 NA      NA
## 4 1991-01-14    PLOO      NA     NA 5.445417 58.12973 137.21730 NA 8.195
## 5 1991-01-21    PLOO      NA     NA      NA 23.20513 59.61538 NA      NA
## 6 1991-01-28    PLOO      NA     NA      NA 31.81891 69.78365 NA      NA
## 7 1991-02-04    PLOO      NA     NA      NA 154.23077 428.58974 NA      NA
##   SALINITY SUSO      TEMP      XMS
## 1 33.61700   NA 19.43000 44.85000
## 2 33.57400   NA 19.31000 75.60000
## 3      NA     NA 14.45000 81.96154
## 4 33.50113   NA 14.07478 82.04876
## 5      NA     NA 14.54487 70.19231
## 6      NA     NA 14.28846 77.61167
## 7      NA     NA 14.23462 81.83333

```

Displaying (Visually in Histograms) the Data per Project (Overarching Location)

```

# owt_df02_gb09_wkly03
# plotting histograms per project (no depth)

# chlorophyll
hist_chloro <- ggplot(owt_df02_gb09_wkly03, aes(x= CHLOROPHYLL)) +
  geom_histogram(aes(color= project, fill= project),

```

```

            position= "identity", bins=7, alpha=.4) +
labs(title= "CHLOROPHYLL") +
theme_classic()

# density
hist_density <- ggplot(owt_df02_gb09_wkly03, aes(x= DENSITY)) +
  geom_histogram(aes(color= project, fill= project),
                 position= "identity", bins=7, alpha=.4) +
  labs(title= "DENSITY")+
  theme_classic()

# DO
hist_do <- ggplot(owt_df02_gb09_wkly03, aes(x= DO)) +
  geom_histogram(aes(color= project, fill= project),
                 position= "identity", bins=7, alpha=.4) +
  labs(title= "DO")+
  theme_classic()

# ENTERO
hist_entero <- ggplot(owt_df02_gb09_wkly03, aes(x= ENTERO)) +
  geom_histogram(aes(color= project, fill= project),
                 position= "identity", bins=7, alpha=.4) +
  labs(title= "ENTERO")+
  theme_classic()

# FECAL
hist_fecal <- ggplot(owt_df02_gb09_wkly03, aes(x= FECAL)) +
  geom_histogram(aes(color= project, fill= project),
                 position= "identity", bins=7, alpha=.4) +
  labs(title= "FECAL")+
  theme_classic()

# OG
hist_og <- ggplot(owt_df02_gb09_wkly03, aes(x= OG)) +
  geom_histogram(aes(color= project, fill= project),
                 position= "identity", bins=7, alpha=.4) +
  labs(title= "OG")+
  theme_classic()

# PH
hist_ph <- ggplot(owt_df02_gb09_wkly03, aes(x= PH)) +
  geom_histogram(aes(color= project, fill= project),
                 position= "identity", bins=7, alpha=.4) +
  labs(title= "pH")+
  theme_classic()

# SALINITY
hist_salinity <- ggplot(owt_df02_gb09_wkly03, aes(x= SALINITY)) +
  geom_histogram(aes(color= project, fill= project),
                 position= "identity", bins=7, alpha=.4) +
  labs(title= "SALINITY")+
  theme_classic()

```

```

# SUSO
hist_suso <- ggplot(owt_df02_gb09_wkly03, aes(x= SUSO)) +
  geom_histogram(aes(color= project, fill= project),
                 position= "identity", bins=7, alpha=.4) +
  labs(title= "SUSO")+
  theme_classic()

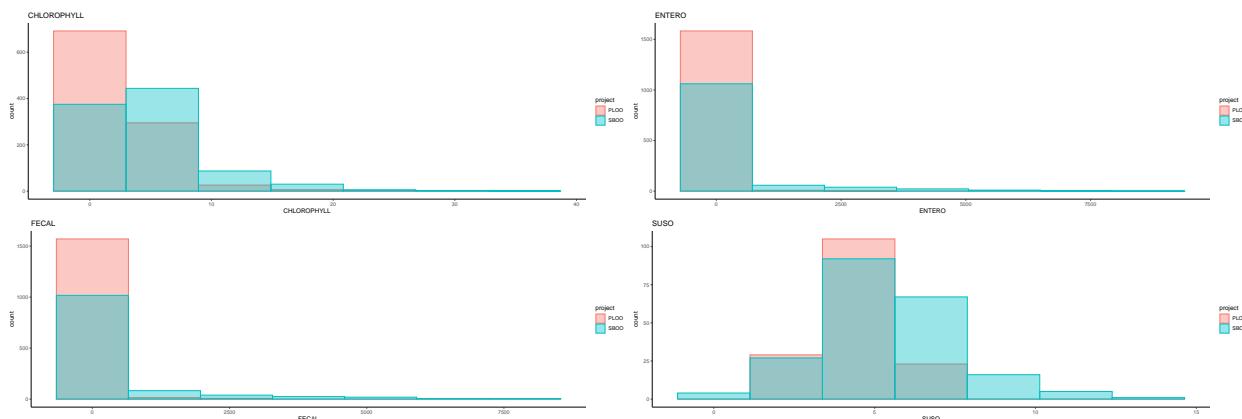
# TEMP
hist_temp <- ggplot(owt_df02_gb09_wkly03, aes(x= TEMP)) +
  geom_histogram(aes(color= project, fill= project),
                 position= "identity", bins=7, alpha=.4) +
  labs(title= "TEMP (C)")+
  theme_classic()

# XMS
hist_xms <- ggplot(owt_df02_gb09_wkly03, aes(x= XMS)) +
  geom_histogram(aes(color= project, fill= project),
                 position= "identity", bins=7, alpha=.4) +
  labs(title= "XMS")+
  theme_classic()

# plotting groups:

# bacteriological basis (chlorophyll, entero, fecal, suso (dissolved solids))
grid.arrange(hist_chloro, hist_entero, hist_fecal,hist_suso,
             ncol=2, nrow=2) +
  theme(legend.key.size = unit(0.5, "cm"),
        legend.title = element_text(size=10),
        legend.text = element_text(size=8),
        text = element_text(size=10))

```

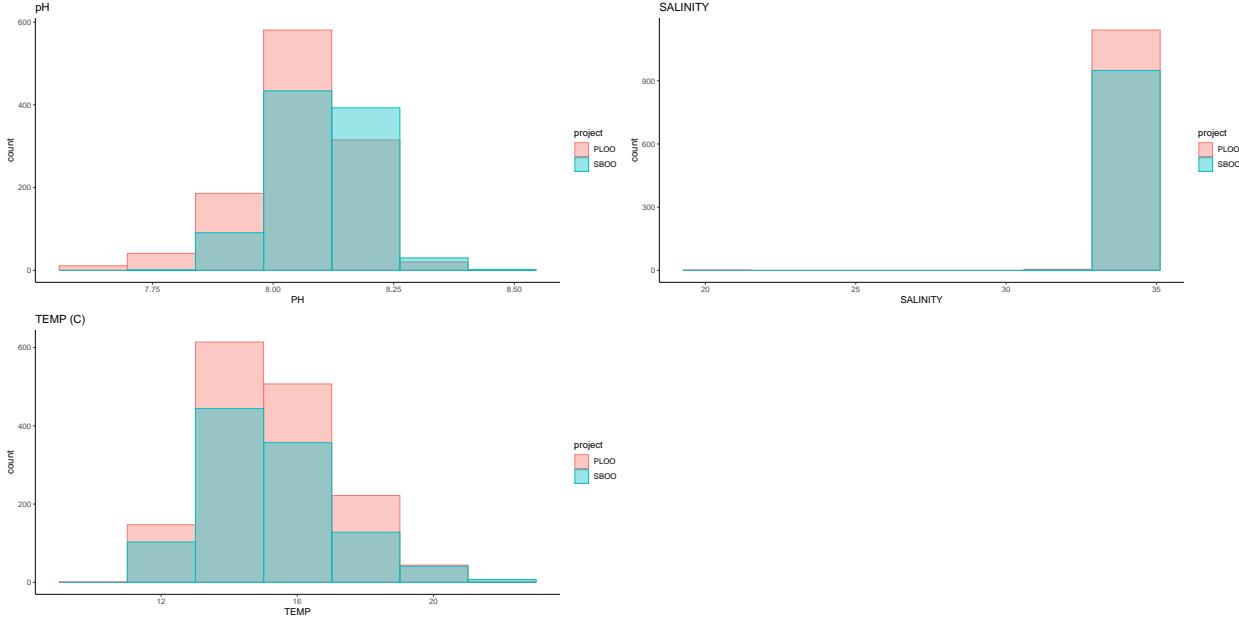


```

## NULL

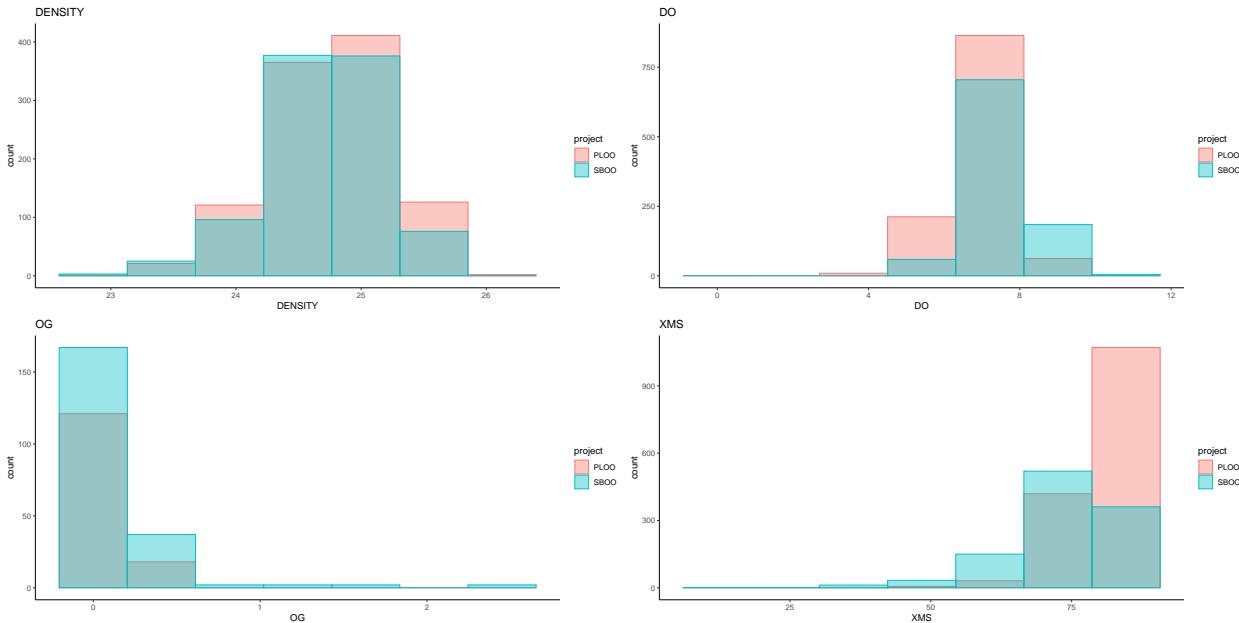
# plotting chemical types (pH, salinity, temp)
grid.arrange(hist_ph, hist_salinity, hist_temp,
             ncol=2, nrow=2) +
  theme(legend.key.size = unit(0.5, "cm"),
        legend.title = element_text(size=10),
        legend.text = element_text(size=8),
        text = element_text(size=10))

```



```
## NULL
```

```
# plotting optical-chemical type (density, DO, OG, xms)
grid.arrange(hist_density, hist_do, hist_og, hist_xms,
             ncol=2, nrow=2) +
  theme(legend.key.size = unit(0.5, "cm"),
        legend.title = element_text(size=10),
        legend.text = element_text(size=8),
        text = element_text(size=10))
```



```
## NULL
```

Setting up for Descriptive Statistics and Looking at Non-Time Series Correlations

```
# descriptive stats (no depth) for numeric columns for whole time frame
owt_df02_gb09_wkly03_nums <- subset(owt_df02_gb09_wkly03, select= -c(date_sample))
summary(owt_df02_gb09_wkly03_nums)

##   project      CHLOROPHYLL       DENSITY        DO
## PL00:1606   Min.    : 0.3333   Min.    :22.71   Min.    : 0.000
## SB00:1196   1st Qu.: 1.5610   1st Qu.:24.46   1st Qu.: 6.789
##                               Median : 2.7145   Median :24.76   Median : 7.350
##                               Mean   : 3.8885   Mean   :24.73   Mean   : 7.243
##                               3rd Qu.: 4.8582   3rd Qu.:25.05   3rd Qu.: 7.805
##                               Max.   :36.0533   Max.   :25.98   Max.   :10.810
##                               NA's    :835     NA's    :802     NA's    :701
##   ENTERO        FECAL          OG          PH
##   Min.    : 2.000   Min.    : 2.000   Min.    :0.2000   Min.    :7.592
##   1st Qu.: 3.279   1st Qu.: 3.709   1st Qu.:0.2000   1st Qu.:8.020
##   Median : 7.579   Median : 9.445   Median :0.2000   Median :8.095
##   Mean   :169.274   Mean   :233.754   Mean   :0.2441   Mean   :8.080
##   3rd Qu.: 27.682   3rd Qu.: 63.223   3rd Qu.:0.2000   3rd Qu.:8.145
##   Max.   :8668.000   Max.   :7881.917   Max.   :2.6500   Max.   :8.439
##   NA's    :16       NA's    :16       NA's    :2451     NA's    :697
##   SALINITY      SUSO          TEMP         XMS
##   Min.    :20.38   Min.    : 0.200   Min.    :10.96   Min.    :18.11
##   1st Qu.:33.41   1st Qu.: 3.697   1st Qu.:13.91   1st Qu.:74.39
##   Median :33.51   Median : 4.637   Median :15.01   Median :79.29
##   Mean   :33.50   Mean   : 4.909   Mean   :15.22   Mean   :77.43
##   3rd Qu.:33.61   3rd Qu.: 6.043   3rd Qu.:16.32   3rd Qu.:82.67
##   Max.   :33.98   Max.   :13.709   Max.   :22.97   Max.   :90.68
##   NA's    :705     NA's    :2433     NA's    :186     NA's    :196
```

If we remove the NA's of the owt_df02_gb09_wkly03 full time frame

```
chlornum <- data.frame("CHLOROPHYLL" = owt_df02_gb09_wkly03_nums$CHLOROPHYLL)
chlornum <- chlornum %>% drop_na()
summary(chlornum)
```

```
##   CHLOROPHYLL
##   Min.    : 0.3333
##   1st Qu.: 1.5610
##   Median : 2.7146
##   Mean   : 3.8885
##   3rd Qu.: 4.8582
##   Max.   :36.0533

densnum <- data.frame("DENSITY" = owt_df02_gb09_wkly03_nums$DENSITY)
densnum <- densnum %>% drop_na()
summary(densnum)
```

```
##   DENSITY
##   Min.    :22.71
##   1st Qu.:24.46
##   Median :24.76
##   Mean   :24.73
```

```

## 3rd Qu.:25.05
## Max. :25.98

donum <- data.frame("DO" = owt_df02_gb09_wkly03_nums$DO)
donum <- donum %>% drop_na()
summary(donum)

##          DO
##  Min.   : 0.000
##  1st Qu.: 6.789
##  Median : 7.350
##  Mean   : 7.243
##  3rd Qu.: 7.805
##  Max.   :10.810

enteronum <- data.frame("ENTERO" = owt_df02_gb09_wkly03_nums$ENTERO)
enteronum <- enteronum %>% drop_na()
summary(enteronum)

##          ENTERO
##  Min.   : 2.000
##  1st Qu.: 3.279
##  Median : 7.579
##  Mean   : 169.274
##  3rd Qu.: 27.682
##  Max.   :8668.000

fecalnum <- data.frame("FECAL" = owt_df02_gb09_wkly03_nums$FECAL)
fecalnum <- fecalnum %>% drop_na()
summary(fecalnum)

##          FECAL
##  Min.   : 2.000
##  1st Qu.: 3.709
##  Median : 9.445
##  Mean   : 233.754
##  3rd Qu.: 63.223
##  Max.   :7881.917

ognum <- data.frame("OG" = owt_df02_gb09_wkly03_nums$OG)
ognum <- ognum %>% drop_na()
summary(ognum)

##          OG
##  Min.   :0.2000
##  1st Qu.:0.2000
##  Median :0.2000
##  Mean   :0.2441
##  3rd Qu.:0.2000
##  Max.   :2.6500

phnum <- data.frame("PH" = owt_df02_gb09_wkly03_nums$PH)
phnum <- phnum %>% drop_na()
summary(phnum)

##          PH
```

```

##   Min.    :7.592
##   1st Qu.:8.020
##   Median :8.095
##   Mean   :8.080
##   3rd Qu.:8.145
##   Max.    :8.439

salinnum <- data.frame("SALINITY" = owt_df02_gb09_wkly03_nums$SALINITY)
salinnum <- salinnum %>% drop_na()
summary(salinnum)

##      SALINITY
##   Min.    :20.38
##   1st Qu.:33.41
##   Median :33.51
##   Mean   :33.50
##   3rd Qu.:33.61
##   Max.    :33.98

susunum <- data.frame("SUSO" = owt_df02_gb09_wkly03_nums$SUSO)
susunum <- susunum %>% drop_na()
summary(susunum)

##      SUSO
##   Min.    : 0.200
##   1st Qu.: 3.697
##   Median : 4.637
##   Mean   : 4.909
##   3rd Qu.: 6.043
##   Max.    :13.709

tempnum <- data.frame("TEMP" = owt_df02_gb09_wkly03_nums$TEMP)
tempnum <- tempnum %>% drop_na()
summary(tempnum)

##      TEMP
##   Min.    :10.96
##   1st Qu.:13.91
##   Median :15.01
##   Mean   :15.22
##   3rd Qu.:16.32
##   Max.    :22.97

xmsnum <- data.frame("XMS" = owt_df02_gb09_wkly03_nums$XMS)
xmsnum <- xmsnum %>% drop_na()
summary(xmsnum)

##      XMS
##   Min.    :18.11
##   1st Qu.:74.39
##   Median :79.29
##   Mean   :77.43
##   3rd Qu.:82.67
##   Max.    :90.68

```

Looking at each column if we remove NA's (for 1/1/2000 onward)

```
# descriptive stats (no depth) for numeric columns for arbitrary after 1/1/2000
owt_df02_gb09_wkly03_nums_2000s <- owt_df02_gb09_wkly03 %>% filter(date_sample >=
  ~ "2000-01-01")
owt_df02_gb09_wkly03_nums_2000s <- subset(owt_df02_gb09_wkly03_nums_2000s, select =
  -c(date_sample))
summary(owt_df02_gb09_wkly03_nums_2000s)

##   project      CHLOROPHYLL       DENSITY        DO
## PL00:1141    Min. : 0.3333    Min. :22.71    Min. : 2.876
## SB00:1146    1st Qu.: 1.5956   1st Qu.:24.45    1st Qu.: 6.852
##                      Median : 2.7296   Median :24.75    Median : 7.388
##                      Mean   : 3.9388   Mean   :24.72    Mean   : 7.305
##                      3rd Qu.: 4.9511   3rd Qu.:25.04    3rd Qu.: 7.819
##                      Max.   :36.0533   Max.   :25.98    Max.   :10.810
##                      NA's   :386      NA's   :368     NA's   :373
##   ENTERO        FECAL          OG          PH
##   Min.   : 2.000   Min.   : 2.000   Min.   :0.2000   Min.   :7.628
##   1st Qu.: 3.141   1st Qu.: 3.480   1st Qu.:0.2000   1st Qu.:8.031
##   Median : 6.909   Median : 8.333   Median :0.2000   Median :8.100
##   Mean   :197.919   Mean   :255.213   Mean   :0.2652   Mean   :8.088
##   3rd Qu.: 26.498   3rd Qu.: 47.318   3rd Qu.:0.2000   3rd Qu.:8.147
##   Max.   :8668.000   Max.   :7881.917   Max.   :2.6500   Max.   :8.439
##   NA's   :12       NA's   :13       NA's   :2057     NA's   :369
##   SALINITY       SUSO          TEMP         XMS
##   Min.   :32.21   Min.   : 0.200   Min.   :10.96   Min.   :18.11
##   1st Qu.:33.41   1st Qu.: 3.804   1st Qu.:13.92   1st Qu.:73.89
##   Median :33.51   Median : 4.954   Median :15.05   Median :79.06
##   Mean   :33.50   Mean   : 5.173   Mean   :15.25   Mean   :77.02
##   3rd Qu.:33.60   3rd Qu.: 6.423   3rd Qu.:16.33   3rd Qu.:82.45
##   Max.   :33.92   Max.   :13.709   Max.   :22.97   Max.   :90.68
##   NA's   :377     NA's   :2041     NA's   :167     NA's   :177
```

Dropping NA's

```
chlornum2000s <- data.frame("CHLOROPHYLL" = owt_df02_gb09_wkly03_nums_2000s$CHLOROPHYLL)
chlornum2000s <- chlornum2000s %>% drop_na()
summary(chlornum2000s)

##   CHLOROPHYLL
##   Min.   : 0.3333
##   1st Qu.: 1.5956
##   Median : 2.7296
##   Mean   : 3.9388
##   3rd Qu.: 4.9511
##   Max.   :36.0533

densnum2000s <- data.frame("DENSITY" = owt_df02_gb09_wkly03_nums_2000s$DENSITY)
densnum2000s <- densnum2000s %>% drop_na()
summary(densnum2000s)

##   DENSITY
##   Min.   :22.71
```

```

## 1st Qu.:24.45
## Median :24.75
## Mean   :24.72
## 3rd Qu.:25.04
## Max.   :25.98

donum2000s <- data.frame("D0" = owt_df02_gb09_wkly03_nums_2000s$D0)
donum2000s <- donum2000s %>% drop_na()
summary(donum2000s)

##          D0
## Min.   : 2.876
## 1st Qu.: 6.852
## Median : 7.389
## Mean   : 7.305
## 3rd Qu.: 7.819
## Max.   :10.810

enteronum2000s <- data.frame("ENTERO" = owt_df02_gb09_wkly03_nums_2000s$ENTERO)
enteronum2000s <- enteronum2000s %>% drop_na()
summary(enteronum2000s)

##          ENTERO
## Min.   : 2.000
## 1st Qu.: 3.141
## Median : 6.909
## Mean   :197.919
## 3rd Qu.: 26.498
## Max.   :8668.000

fecalnum2000s <- data.frame("FECAL" = owt_df02_gb09_wkly03_nums_2000s$FECAL)
fecalnum2000s <- fecalnum2000s %>% drop_na()
summary(fecalnum2000s)

##          FECAL
## Min.   : 2.000
## 1st Qu.: 3.480
## Median : 8.333
## Mean   :255.213
## 3rd Qu.: 47.318
## Max.   :7881.917

ognum2000s <- data.frame("OG" = owt_df02_gb09_wkly03_nums_2000s$OG)
ognum2000s <- ognum2000s %>% drop_na()
summary(ognum2000s)

##          OG
## Min.   :0.2000
## 1st Qu.:0.2000
## Median :0.2000
## Mean   :0.2652
## 3rd Qu.:0.2000
## Max.   :2.6500

phnum2000s <- data.frame("PH" = owt_df02_gb09_wkly03_nums_2000s$PH)
phnum2000s <- phnum2000s %>% drop_na()

```

```

summary(phnum2000s)

##      PH
##  Min.   :7.628
##  1st Qu.:8.031
##  Median :8.100
##  Mean   :8.088
##  3rd Qu.:8.147
##  Max.   :8.439

salinnum2000s <- data.frame("SALINITY" = owt_df02_gb09_wkly03_nums_2000s$SALINITY)
salinnum2000s <- salinnum2000s %>% drop_na()
summary(salinnum2000s)

##      SALINITY
##  Min.   :32.21
##  1st Qu.:33.41
##  Median :33.51
##  Mean   :33.50
##  3rd Qu.:33.60
##  Max.   :33.92

susunum2000s <- data.frame("SUSO" = owt_df02_gb09_wkly03_nums_2000s$SUSO)
susunum2000s <- susunum2000s %>% drop_na()
summary(susunum2000s)

##      SUSO
##  Min.   : 0.200
##  1st Qu.: 3.804
##  Median : 4.954
##  Mean   : 5.173
##  3rd Qu.: 6.423
##  Max.   :13.709

tempnum2000s <- data.frame("TEMP" = owt_df02_gb09_wkly03_nums_2000s$TEMP)
tempnum2000s <- tempnum2000s %>% drop_na()
summary(tempnum2000s)

##      TEMP
##  Min.   :10.96
##  1st Qu.:13.92
##  Median :15.05
##  Mean   :15.25
##  3rd Qu.:16.33
##  Max.   :22.97

xmsnum2000s <- data.frame("XMS" = owt_df02_gb09_wkly03_nums_2000s$XMS)
xmsnum2000s <- xmsnum2000s %>% drop_na()
summary(xmsnum2000s)

##      XMS
##  Min.   :18.11
##  1st Qu.:73.89
##  Median :79.06
##  Mean   :77.02

```

```
## 3rd Qu.:82.45
## Max. :90.68
```

looking at non time series correlations out of curiosuty

```
print(paste("chlorophyll", nrow(chlornum), ", density", nrow(densnum), ", DO",
           "nrow(donum),",
           "entero", nrow(enteronum), ", fecal", nrow(fecalnum), ", OG", nrow(ognum),
           ", pH", nrow(phnum), ", salinity", nrow(salinnum), ", Suso", nrow(susunum),
           ", Temp", nrow(tempnum), ", XMS", nrow(xmsnum)))
```

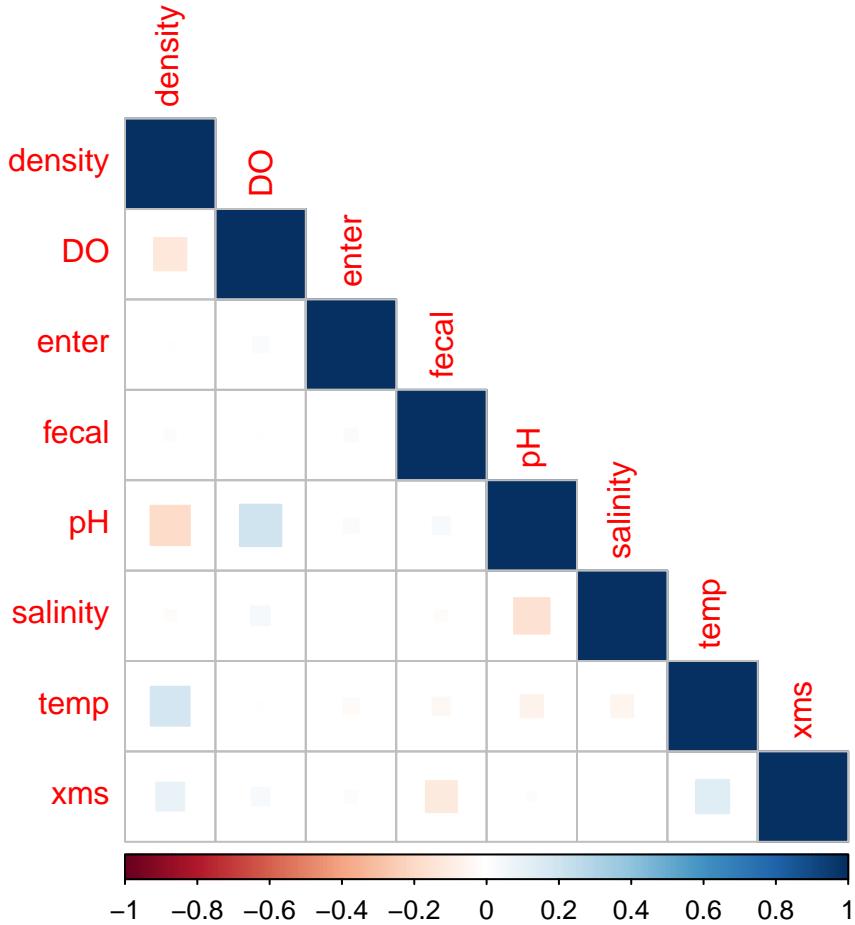
```
## [1] "chlorophyll 1967 , density 2000 , DO 2101 , entero 2786 , fecal 2786 , OG 351 , pH 2105 , salin"
```

```
# least populated in OG at 351 and SUSO at 369.
# focusing on others, the limit will be Chlorophyll at 1967
```

```
densnum_sub <- densnum[1:1967,]
donum_sub <- donum[1:1967,]
enteronum_sub <- enteronum[1:1967,]
fecalnum_sub <- fecalnum[1:1967,]
phnum_sub <- phnum[1:1967,]
salinnum_sub <- salinnum[1:1967,]
tempnum_sub <- tempnum[1:1967,]
xmsnum_sub <- xmsnum[1:1967,]

splitnum_sub <- data.frame("density" = densnum_sub, "DO" = donum_sub,
                           "enter" = enteronum_sub, "fecal" = fecalnum_sub,
                           "pH" = phnum_sub, "salinity" = salinnum_sub,
                           "temp" = tempnum_sub, "xms" = xmsnum_sub)

corrplot(
  cor(splitnum_sub),
  method = "square",
  type= "lower")
```



```
round(cor(splitnum_sub), 2)
```

```
##      density    DO enter  fecal     pH salinity   temp    xms
## density  1.00 -0.13  0.00  0.02 -0.19 -0.02  0.18  0.10
## DO       -0.13  1.00  0.03 -0.01  0.21  0.04 -0.01  0.04
## enter     0.00  0.03  1.00  0.02  0.03 -0.01 -0.03 -0.02
## fecal     0.02 -0.01  0.02  1.00  0.04 -0.02 -0.04 -0.12
## pH        -0.19  0.21  0.03  0.04  1.00 -0.15 -0.06  0.01
## salinity   -0.02  0.04 -0.01 -0.02 -0.15  1.00 -0.06  0.00
## temp      0.18 -0.01 -0.03 -0.04 -0.06 -0.06  1.00  0.13
## xms       0.10  0.04 -0.02 -0.12  0.01  0.00  0.13  1.00
```

Run custom function on data aggregated by `date_sample`, `project`, `depth_m_bin` and `parameter`

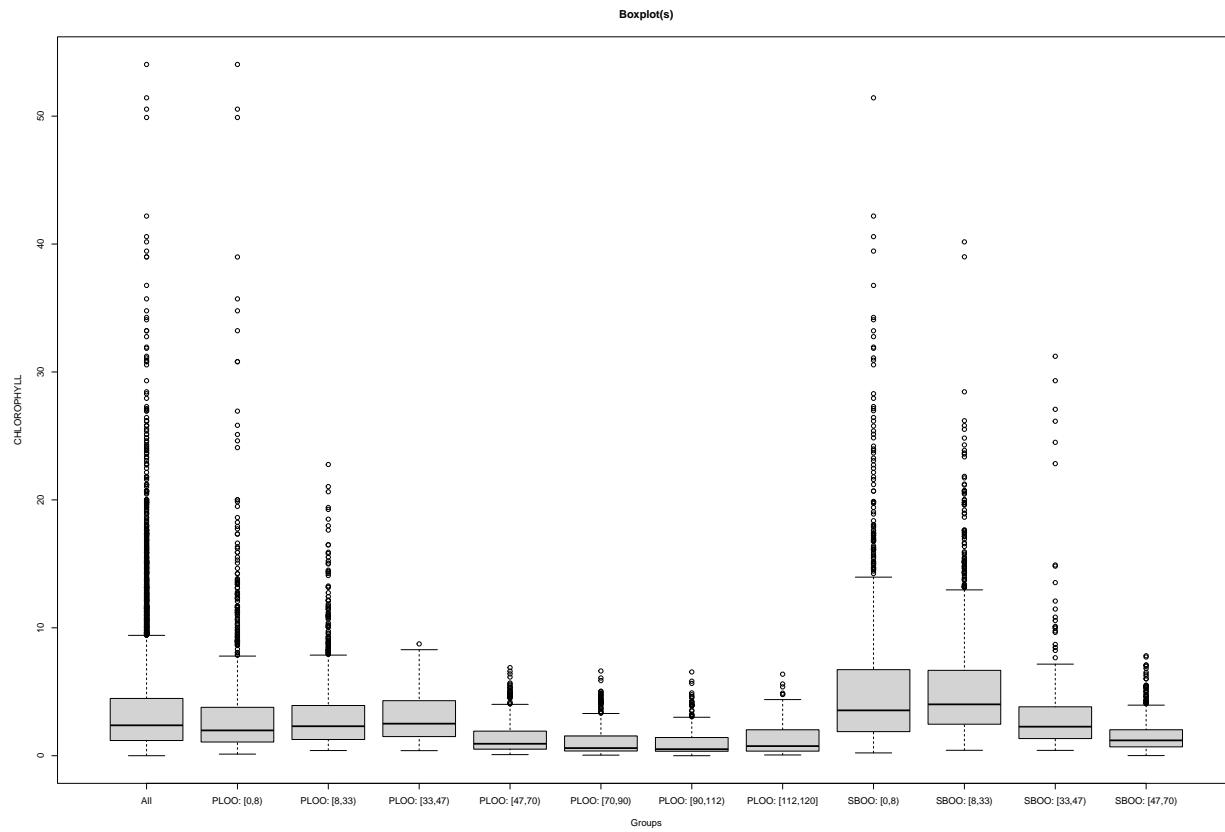
```
owt_df02_gb07_mrgd = ts_eda(ts_df = owt_df02_gb07,
                           form_lead = c("date_sample", "project", "depth_m_bin"),
                           form_cast = "parameter",
                           param_lst = param_lst02,
                           l1_col = c("project"),
                           l1_param = c("PL00", "SB00"),
                           l2_col = c("depth_m_bin"),
```

```

    l2_param = depth_lvls01,
    rtn_met = FALSE,
    box = FALSE
  )

## Error in `$.data.frame`(`*tmp`, "Groups", value = structure(1L, levels = "PLOO: Unknown", class =
##   replacement has 1 row, data has 0
## Error in `$.data.frame`(`*tmp`, "Groups", value = structure(1L, levels = "SBOO: [70,90)", class =
##   replacement has 1 row, data has 0
## Error in `$.data.frame`(`*tmp`, "Groups", value = structure(1L, levels = "SBOO: [90,112)", class =
##   replacement has 1 row, data has 0
## Error in `$.data.frame`(`*tmp`, "Groups", value = structure(1L, levels = "SBOO: [112,120]", class =
##   replacement has 1 row, data has 0
## Error in `$.data.frame`(`*tmp`, "Groups", value = structure(1L, levels = "SBOO: Unknown", class =
##   replacement has 1 row, data has 0

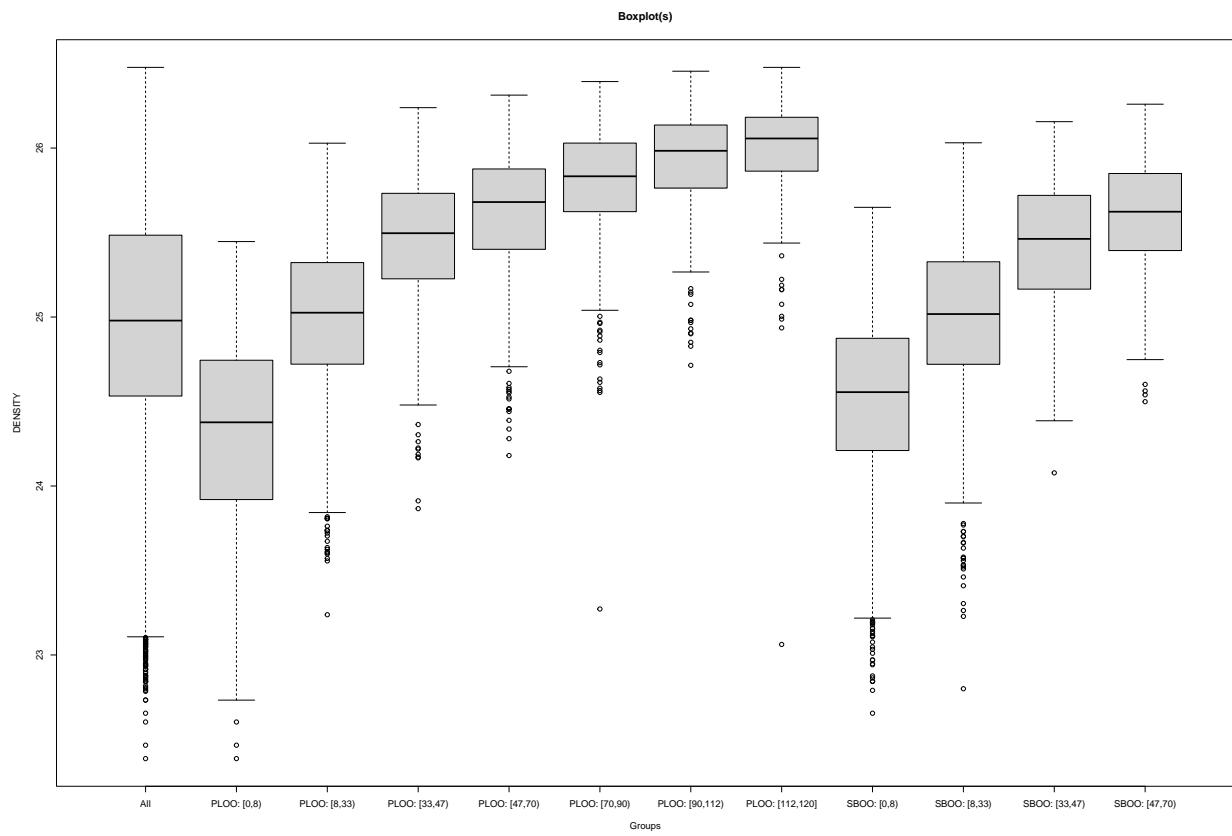
```



```

## Error in `$.data.frame`(`*tmp`, "Groups", value = structure(1L, levels = "PLOO: Unknown", class =
##   replacement has 1 row, data has 0
## Error in `$.data.frame`(`*tmp`, "Groups", value = structure(1L, levels = "SBOO: [70,90)", class =
##   replacement has 1 row, data has 0
## Error in `$.data.frame`(`*tmp`, "Groups", value = structure(1L, levels = "SBOO: [90,112)", class =
##   replacement has 1 row, data has 0
## Error in `$.data.frame`(`*tmp`, "Groups", value = structure(1L, levels = "SBOO: [112,120]", class =
##   replacement has 1 row, data has 0
## Error in `$.data.frame`(`*tmp`, "Groups", value = structure(1L, levels = "SBOO: Unknown", class =
##   replacement has 1 row, data has 0

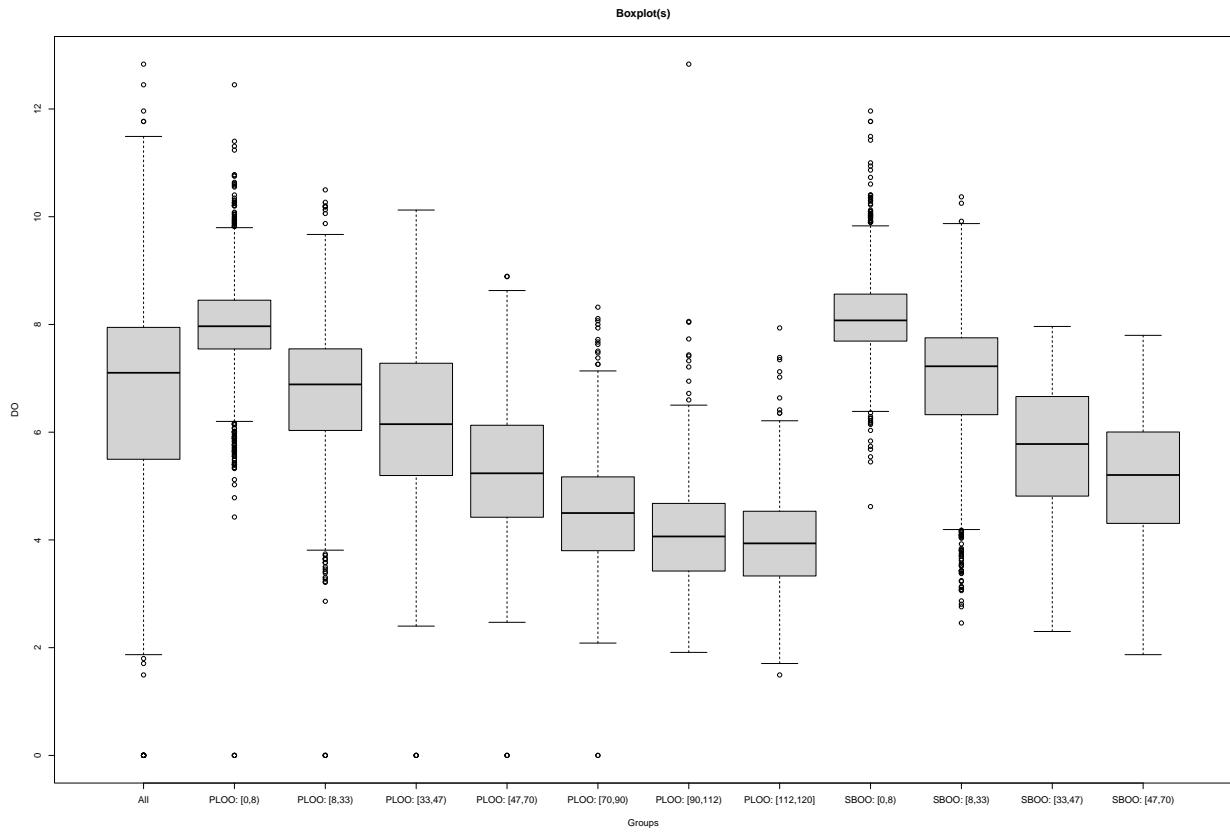
```



```

## Error in `$<- .data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "PLOO: Unknown", class =
##   replacement has 1 row, data has 0
## Error in `$<- .data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [70,90)", class =
##   replacement has 1 row, data has 0
## Error in `$<- .data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [90,112)", class =
##   replacement has 1 row, data has 0
## Error in `$<- .data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [112,120]", class =
##   replacement has 1 row, data has 0
## Error in `$<- .data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: Unknown", class =
##   replacement has 1 row, data has 0

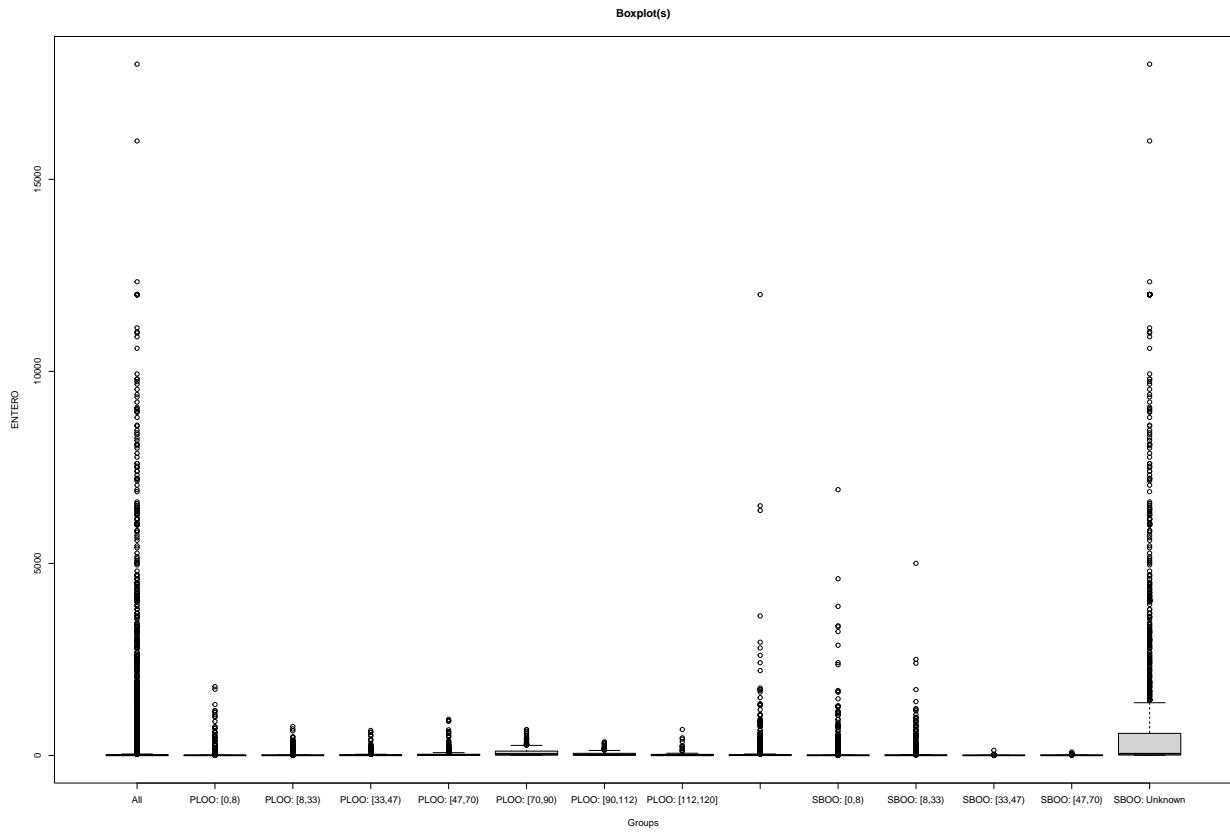
```



```

## Error in `$<- .data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [70,90)", class =
##   replacement has 1 row, data has 0
## Error in `$<- .data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [90,112)", class =
##   replacement has 1 row, data has 0
## Error in `$<- .data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [112,120]", class =
##   replacement has 1 row, data has 0

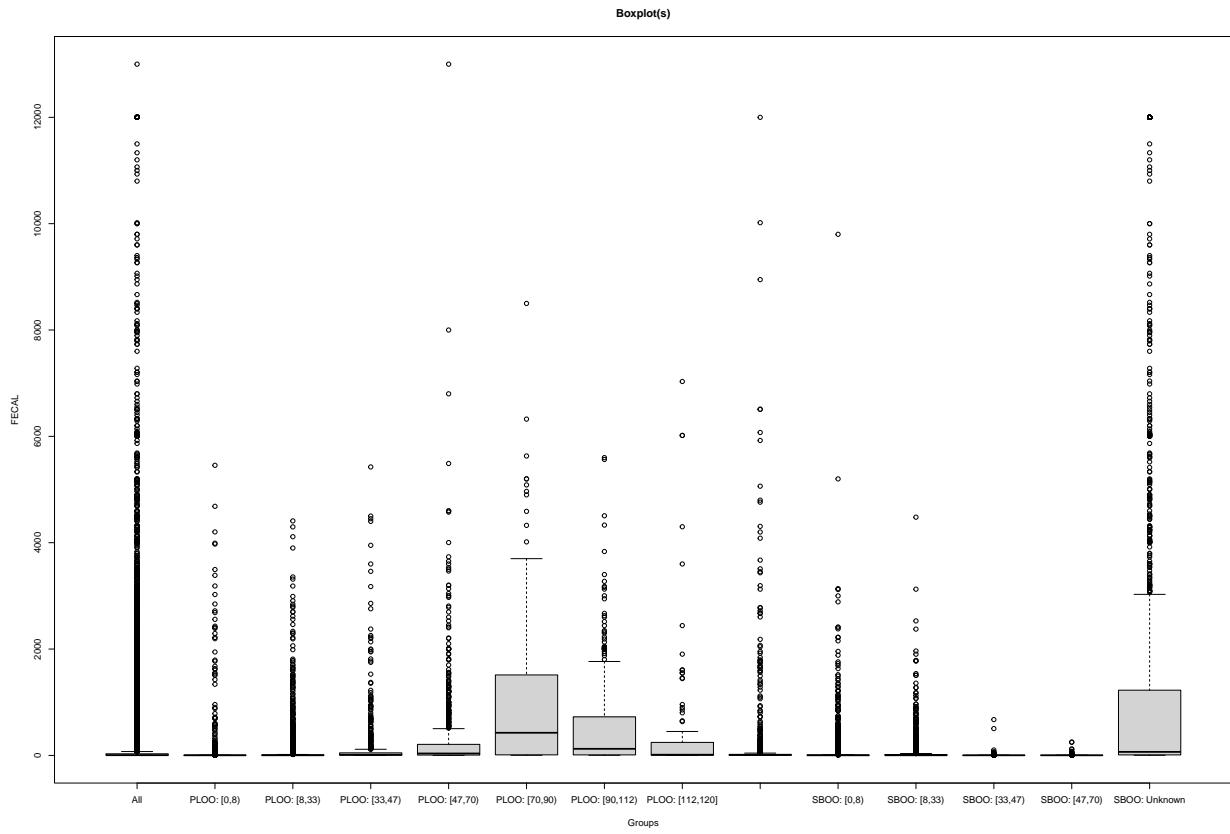
```



```

## Error in `$<- .data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [70,90]", class =
##   replacement has 1 row, data has 0
## Error in `$<- .data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [90,112]", class =
##   replacement has 1 row, data has 0
## Error in `$<- .data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [112,120]", class =
##   replacement has 1 row, data has 0

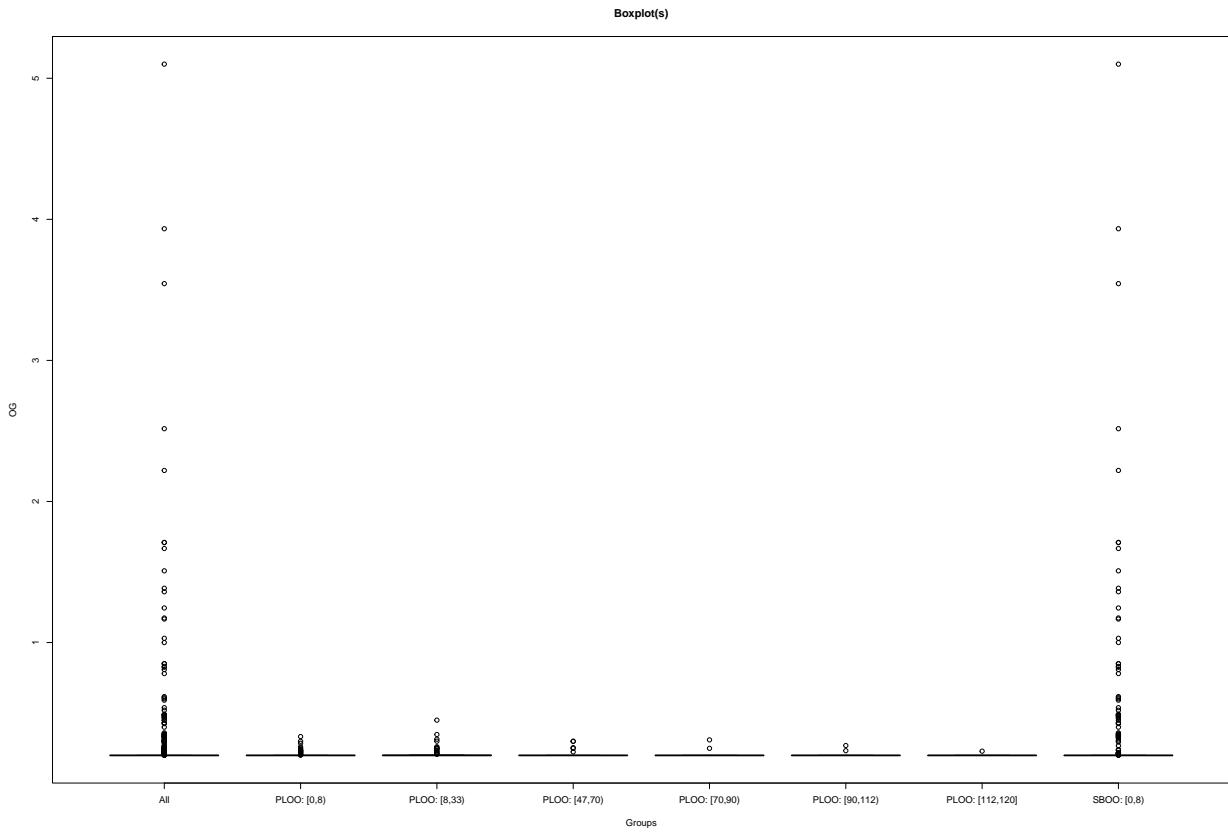
```



```

## Error in `$<- .data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "PLOO: [33,47]", class =
##   replacement has 1 row, data has 0
## Error in `$<- .data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "PLOO: Unknown", class =
##   replacement has 1 row, data has 0
## Error in `$<- .data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [8,33]", class =
##   replacement has 1 row, data has 0
## Error in `$<- .data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [33,47]", class =
##   replacement has 1 row, data has 0
## Error in `$<- .data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [47,70]", class =
##   replacement has 1 row, data has 0
## Error in `$<- .data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [70,90]", class =
##   replacement has 1 row, data has 0
## Error in `$<- .data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [90,112]", class =
##   replacement has 1 row, data has 0
## Error in `$<- .data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [112,120]", class =
##   replacement has 1 row, data has 0
## Error in `$<- .data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: Unknown", class =
##   replacement has 1 row, data has 0

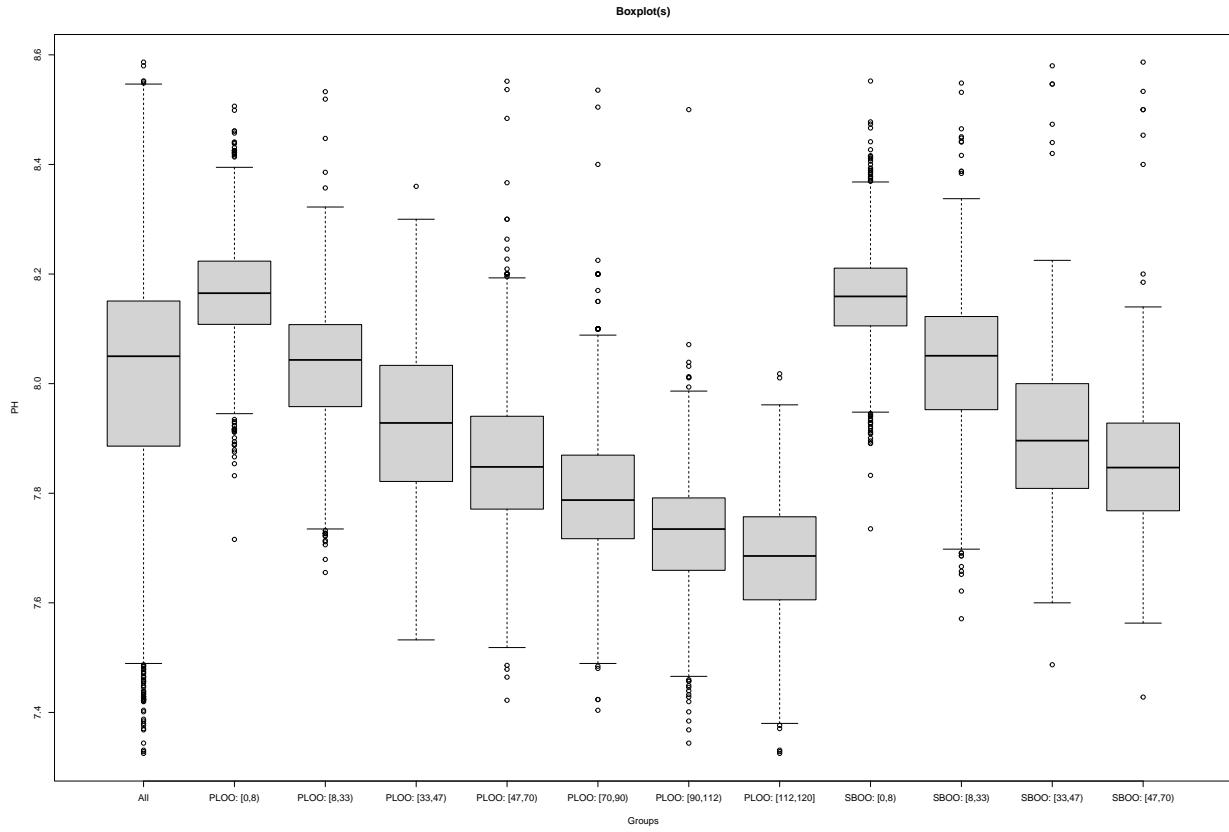
```



```

## Error in `$<- .data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "PLOO: Unknown", class =
##   replacement has 1 row, data has 0
## Error in `$<- .data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [70,90)", class =
##   replacement has 1 row, data has 0
## Error in `$<- .data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [90,112)", class =
##   replacement has 1 row, data has 0
## Error in `$<- .data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [112,120]", class =
##   replacement has 1 row, data has 0
## Error in `$<- .data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: Unknown", class =
##   replacement has 1 row, data has 0

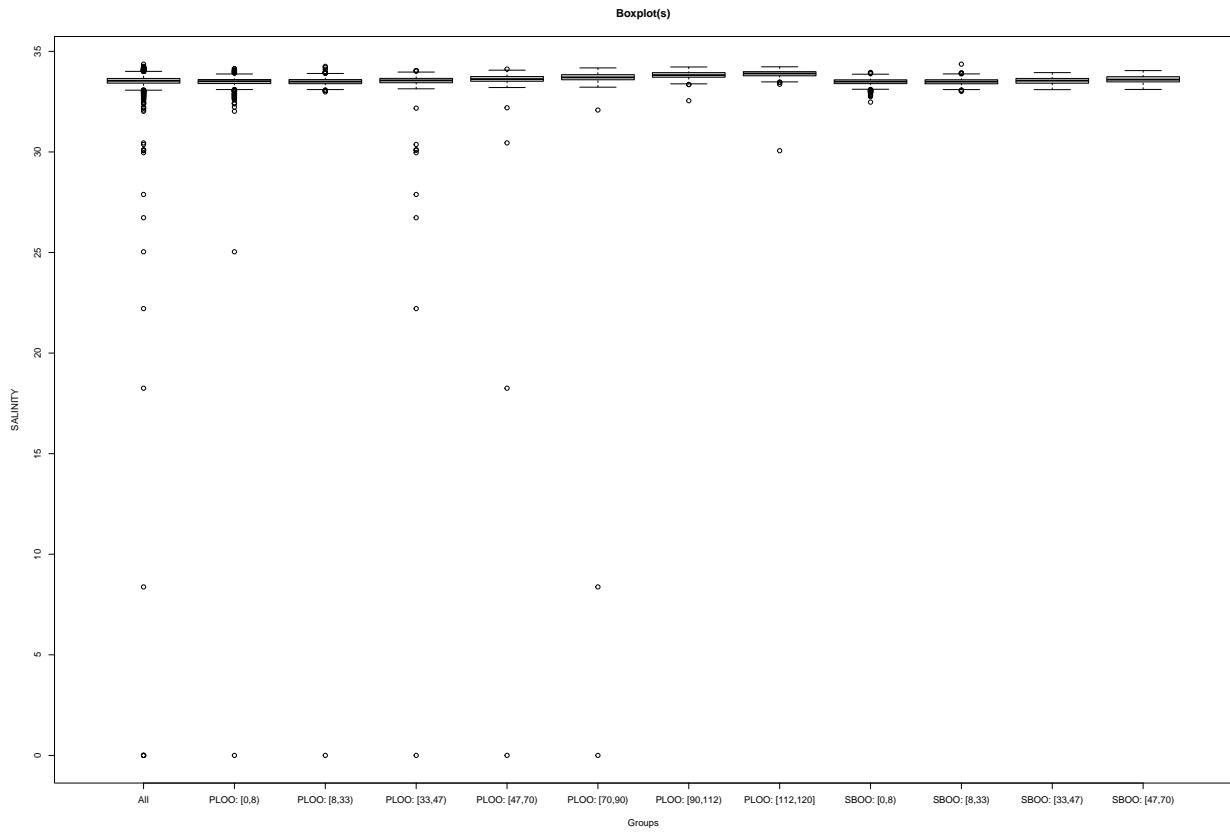
```



```

## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "PLOO: Unknown", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [70,90)", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [90,112)", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [112,120]", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: Unknown", class =
##   replacement has 1 row, data has 0

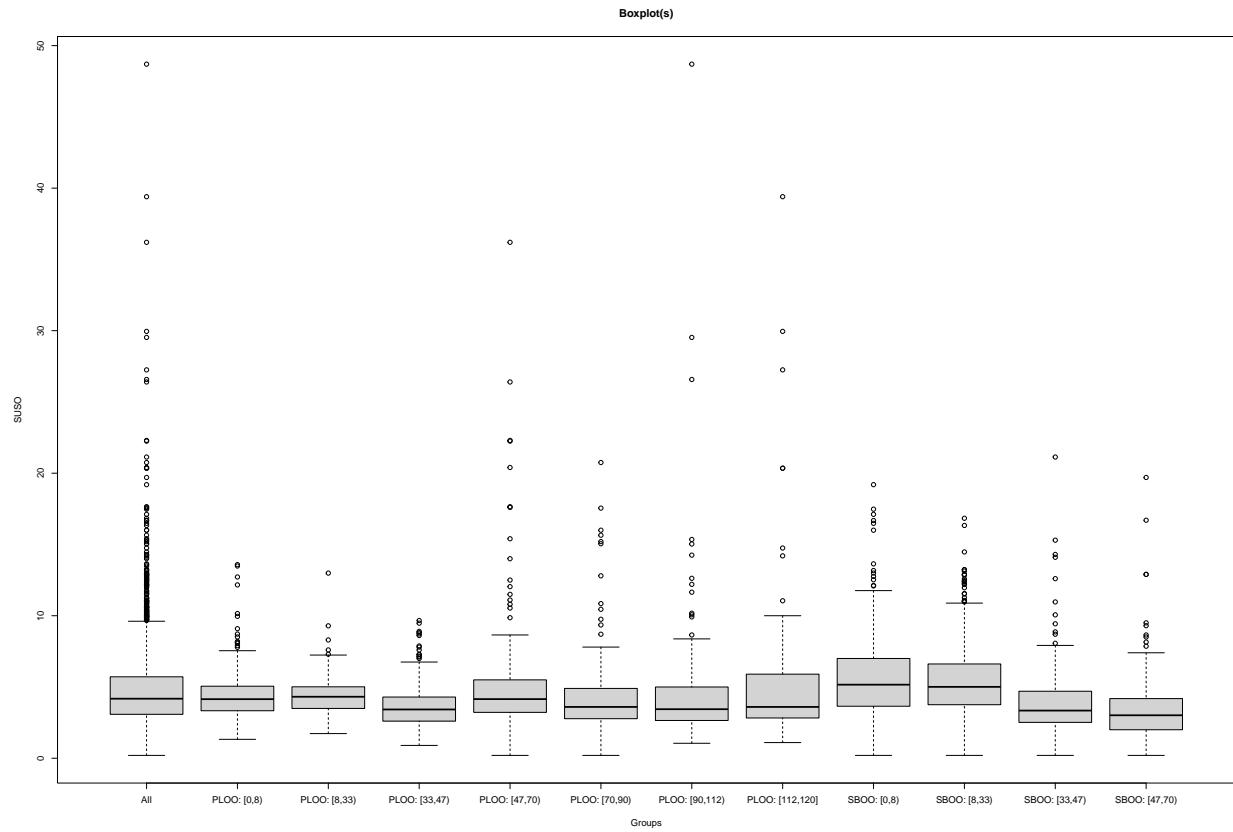
```



```

## Error in `$<- .data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "PLOO: Unknown", class =
##   replacement has 1 row, data has 0
## Error in `$<- .data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [70,90)", class =
##   replacement has 1 row, data has 0
## Error in `$<- .data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [90,112)", class =
##   replacement has 1 row, data has 0
## Error in `$<- .data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [112,120]", class =
##   replacement has 1 row, data has 0
## Error in `$<- .data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: Unknown", class =
##   replacement has 1 row, data has 0

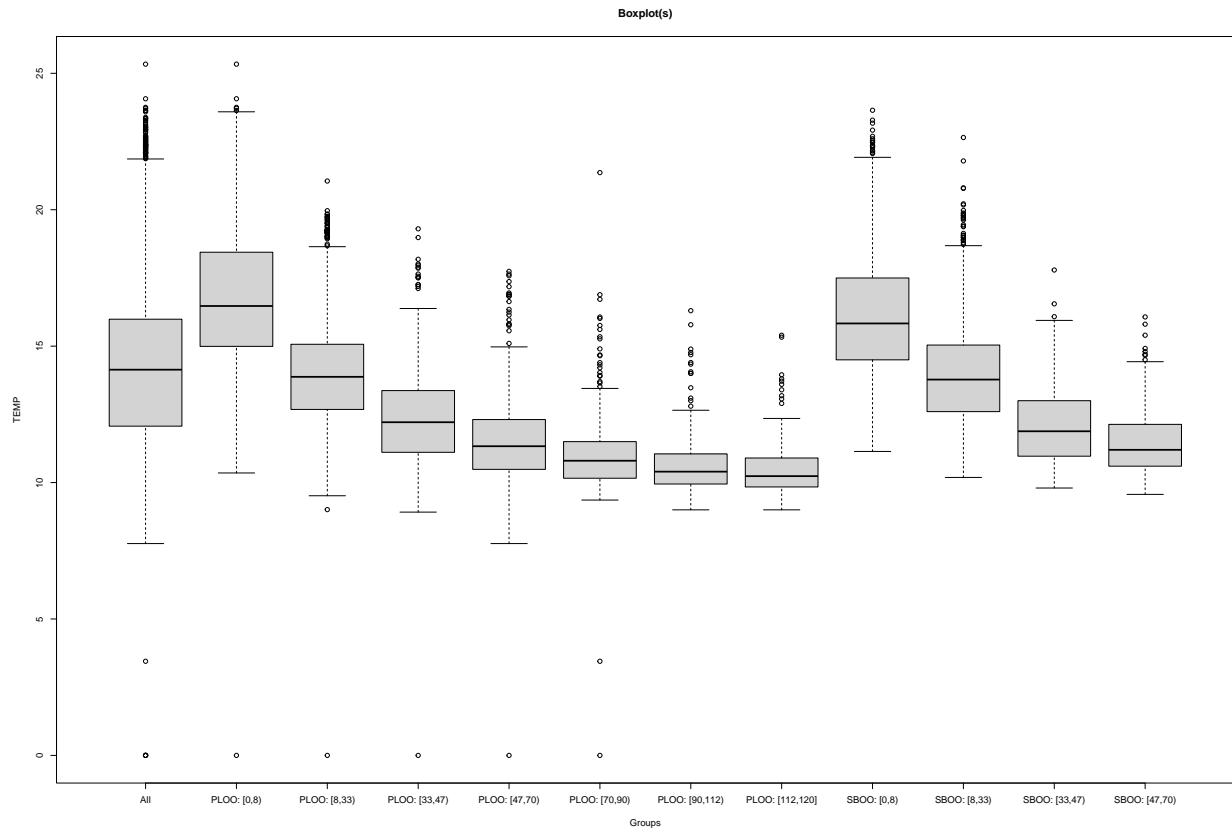
```



```

## Error in `$<- .data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "PLOO: Unknown", class =
##   replacement has 1 row, data has 0
## Error in `$<- .data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [70,90]", class =
##   replacement has 1 row, data has 0
## Error in `$<- .data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [90,112]", class =
##   replacement has 1 row, data has 0
## Error in `$<- .data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [112,120]", class =
##   replacement has 1 row, data has 0
## Error in `$<- .data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: Unknown", class =
##   replacement has 1 row, data has 0

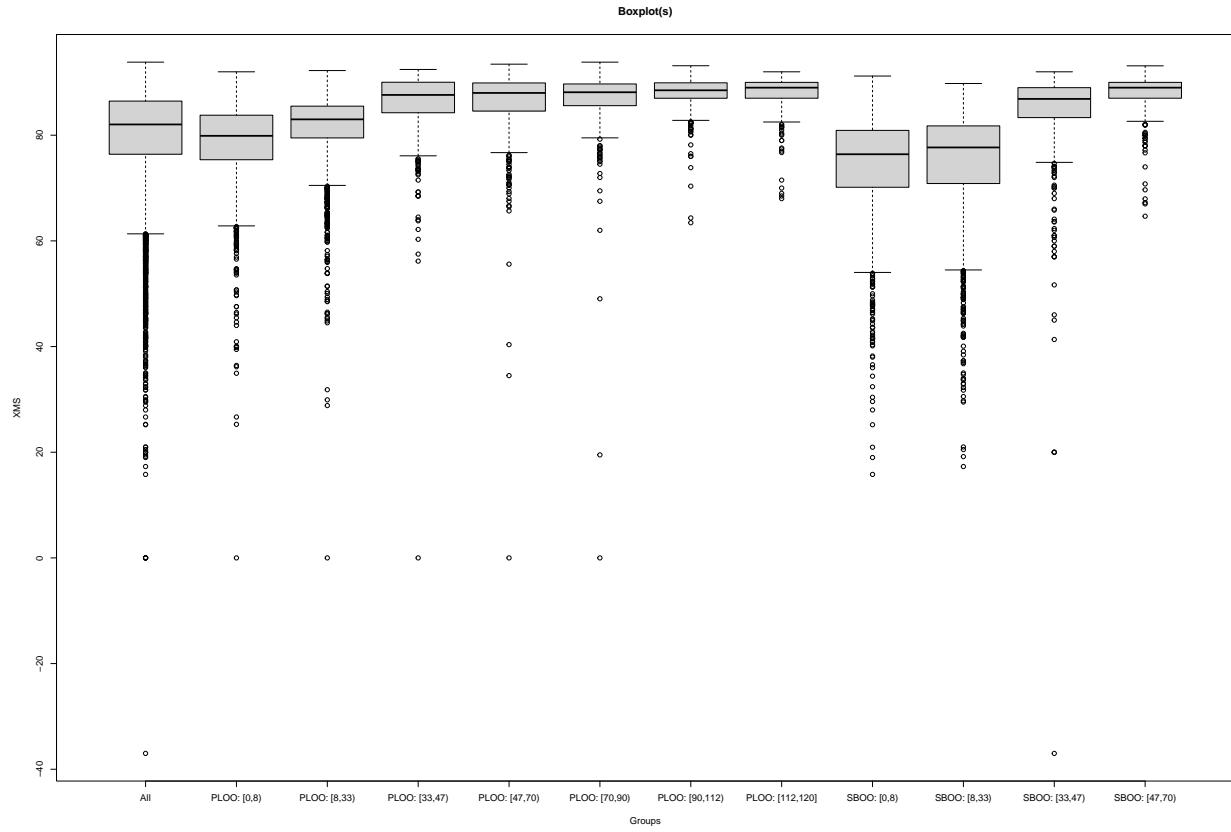
```



```

## Error in `$<- .data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "PLOO: Unknown", class =
##   replacement has 1 row, data has 0
## Error in `$<- .data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [70,90]", class =
##   replacement has 1 row, data has 0
## Error in `$<- .data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [90,112]", class =
##   replacement has 1 row, data has 0
## Error in `$<- .data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [112,120]", class =
##   replacement has 1 row, data has 0
## Error in `$<- .data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: Unknown", class =
##   replacement has 1 row, data has 0

```



```

##   date_sample project depth_m_bin CHLOROPHYLL DENSITY DO ENTERO FECAL
## 1 1990-11-15     PL00      [8,33)      0.87 23.855 6.55    NA    NA
## 2 1990-11-15     SB00      [0,8)       1.27 23.853 7.41    NA    NA
## 3 1991-01-02     PL00      [0,8)        NA     NA 34.07407 134.81481
## 4 1991-01-02     PL00      [8,33)        NA     NA 27.50000 105.00000
## 5 1991-01-03     PL00      [0,8)        NA     NA 17.40741 45.55556
## 6 1991-01-03     PL00      [8,33)        NA     NA 21.66667 64.16667
## 7 1991-01-07     PL00      [0,8)        NA     NA 77.40741 44.07407
##   OG PH SALINITY SUSO TEMP XMS
## 1 NA 8.08 33.617  NA 19.43000 44.85000
## 2 NA 8.18 33.574  NA 19.31000 75.60000
## 3 NA  NA  NA 14.56667 77.03704
## 4 NA  NA  NA 14.37500 86.75000
## 5 NA  NA  NA 14.48519 83.55556
## 6 NA  NA  NA 14.18333 84.66667
## 7 NA  NA  NA 14.61852 77.62963
##          vars   n   mean      sd median trimmed  mad   min   max
## date_sample 1 17003   NaN     NA     NA  NaN  NA Inf -Inf
## project*    2 17003 1.37    0.48   1.00  1.34 0.00  1.00  2.00
## depth_m_bin* 3 17003 4.49    2.61   5.00  4.48 2.97  1.00  8.00
## CHLOROPHYLL 4  8386  3.60    4.09   2.38  2.82 2.13  0.00 54.04
## DENSITY      5  8627 24.96    0.69  24.98 24.99 0.70 22.39 26.48
## DO           6  9842  6.70    1.69   7.10  6.80 1.58  0.00 12.83
## ENTERO       7 15533 150.11  925.42  2.80 10.66 1.19  0.00 18000.00
## FECAL        8 14874 269.82 1129.98  3.85 28.41 2.74  0.00 13000.00
## OG           9  1115  0.24    0.26   0.20  0.20 0.00  0.20  5.10
## PH          10  9695  8.01    0.18   8.05  8.02 0.18  7.33  8.59

```

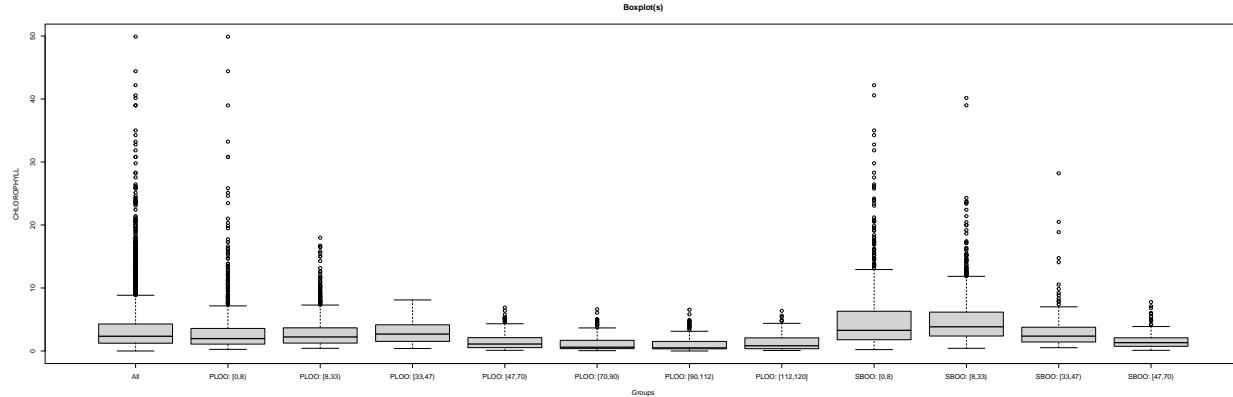
```

## SALINITY      11  9844  33.52     0.86  33.54   33.54 0.17    0.00   34.36
## SUSO         12  3023   4.77     2.99   4.18    4.40 1.89    0.20   48.70
## TEMP         13 12010  14.27     2.80  14.14   14.11 2.90    0.00   25.34
## XMS          14 11969  80.22     9.07  82.04   81.41 7.25   -37.00  93.83
##              range skew kurtosis se
## date_sample   -Inf    NA      NA  NA
## project*      1.00   0.54   -1.71 0.00
## depth_m_bin*  7.00  -0.11   -1.42 0.02
## CHLOROPHYLL  54.04   3.70   22.87 0.04
## DENSITY       4.09  -0.36   -0.15 0.01
## DO            12.83  -0.55   -0.16 0.02
## ENTERO        18000.00  9.88  111.40 7.43
## FECAL         13000.00  7.12   59.92 9.27
## OG            4.90   11.87  171.74 0.01
## PH            1.26  -0.50   -0.10 0.00
## SALINITY      34.36 -34.13  1283.78 0.01
## SUSO          48.50   3.91   33.72 0.05
## TEMP          25.34   0.42   0.06 0.03
## XMS          130.83 -2.14    9.79 0.08
##              date_sample project depth_m_bin parameter      Avg
## Not NA n      121286   121286    121286   121286 119635.0000
## NA n          0        0        0        0      1651.0000
## Not NA %      1        1        1        1      0.9864
## NA %          0        0        0        0      0.0136

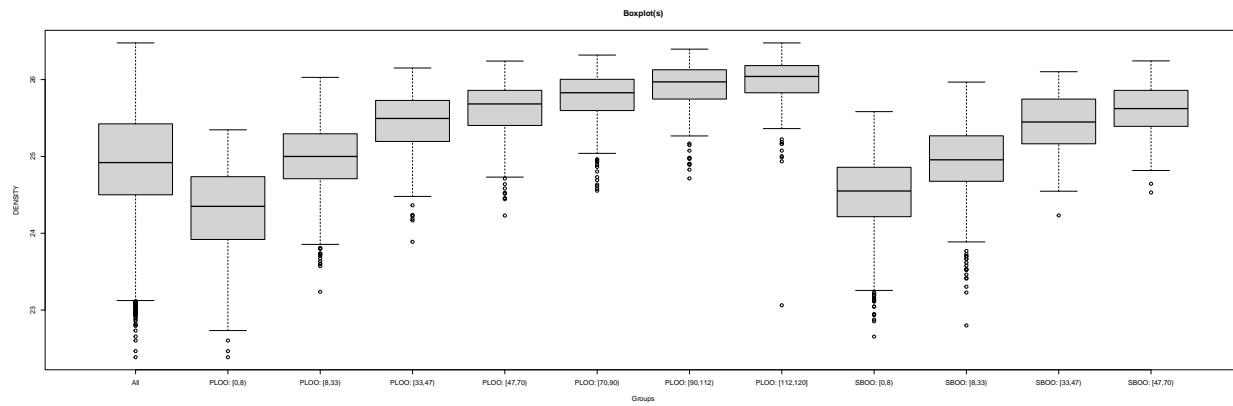
owt_df02_gb07_wkly = ts_eda(ts_df = owt_df02_gb07,
                             form_lead = c("date_sample", "project", "depth_m_bin"),
                             form_cast = "parameter",
                             param_lst = param_lst02,
                             l1_col = c("project"),
                             l1_param = c("PL00", "SB00"),
                             l2_col = c("depth_m_bin"),
                             l2_param = depth_lvls01,
                             rtn_met = FALSE,
                             box = FALSE,
                             week_agg = TRUE
                           )

## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "PL00: Unknown", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SB00: [70,90)", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SB00: [90,112)", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SB00: [112,120]", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SB00: Unknown", class =
##   replacement has 1 row, data has 0

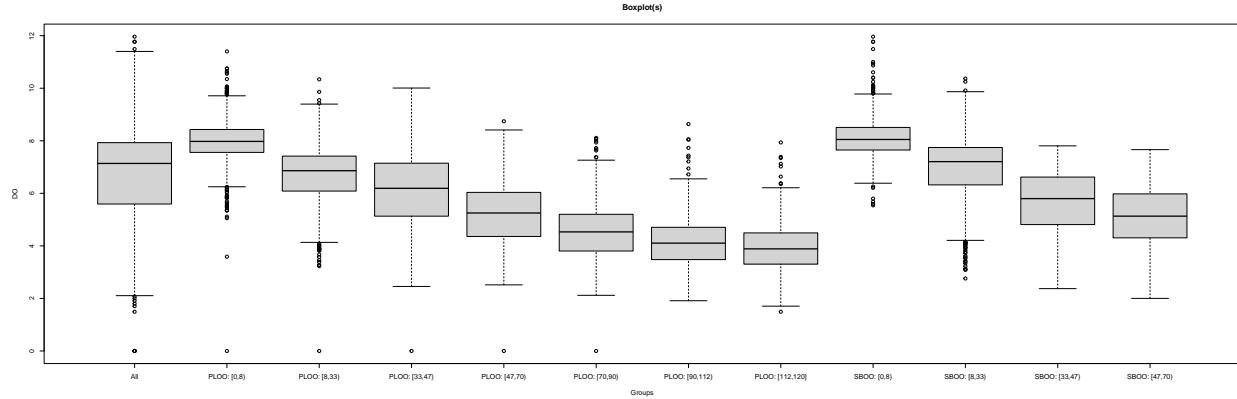
```



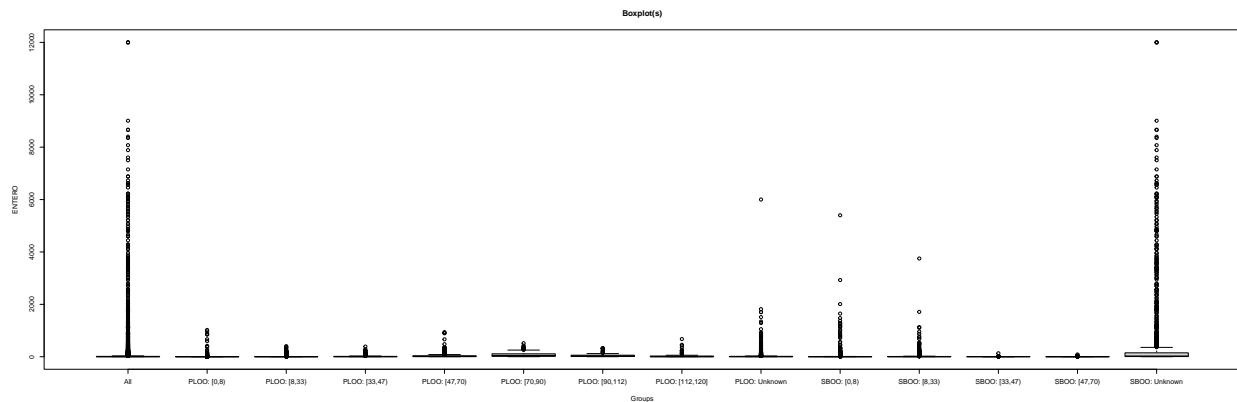
```
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "PLOO: Unknown", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SB00: [70,90]", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SB00: [90,112]", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SB00: [112,120]", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SB00: Unknown", class =
##   replacement has 1 row, data has 0
```



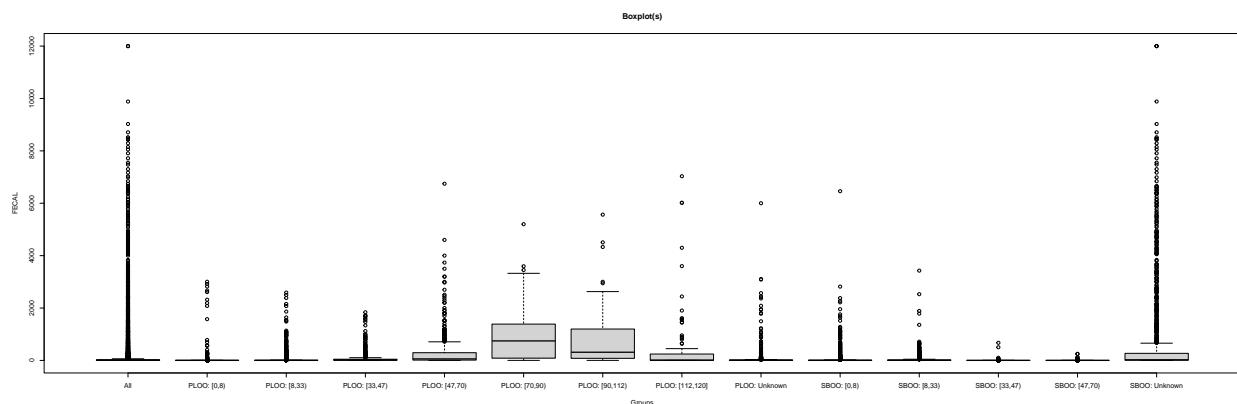
```
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "PLOO: Unknown", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SB00: [70,90]", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SB00: [90,112]", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SB00: [112,120]", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SB00: Unknown", class =
##   replacement has 1 row, data has 0
```



```
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [70,90)", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [90,112)", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [112,120]", class =
##   replacement has 1 row, data has 0
```



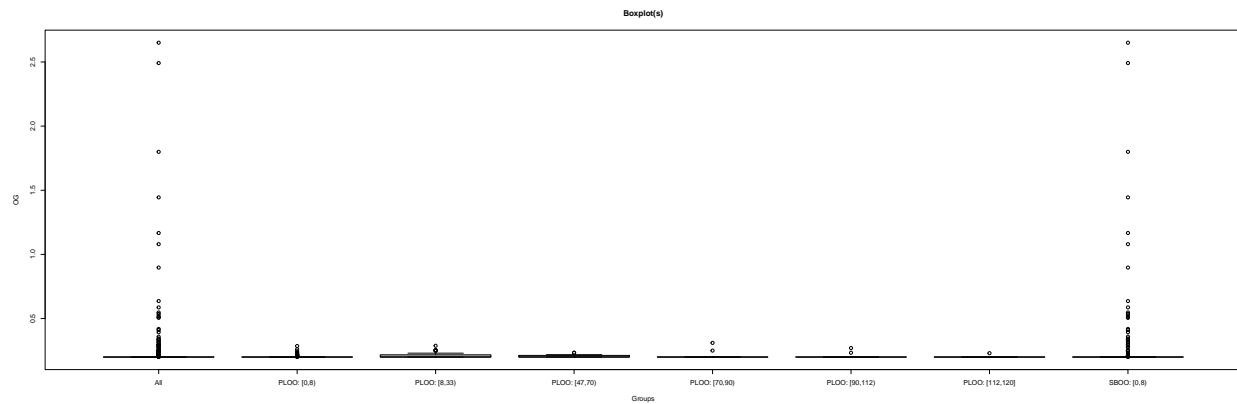
```
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [70,90)", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [90,112)", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [112,120]", class =
##   replacement has 1 row, data has 0
```



```

## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "PLOO: [33,47)", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "PLOO: Unknown", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [8,33)", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [33,47)", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [47,70)", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [70,90)", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [90,112)", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [112,120]", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: Unknown", class =
##   replacement has 1 row, data has 0

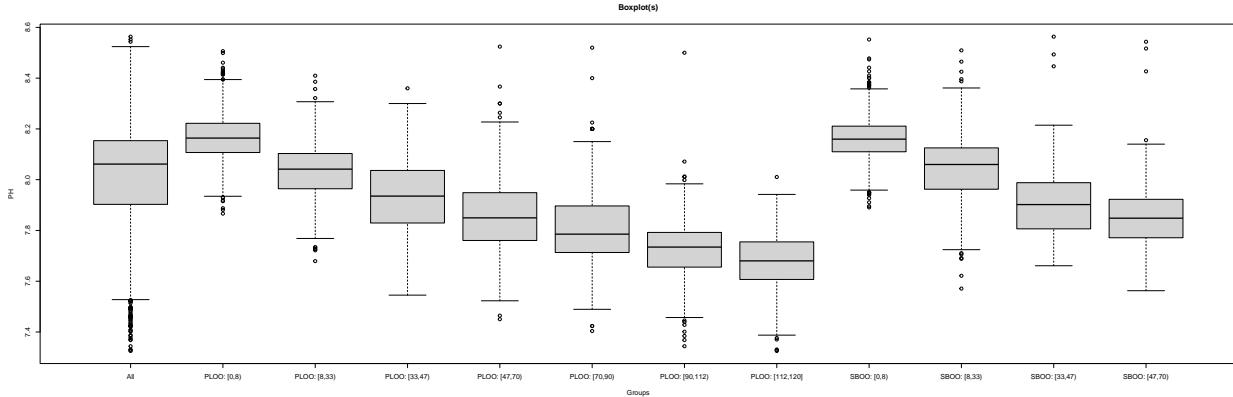
```



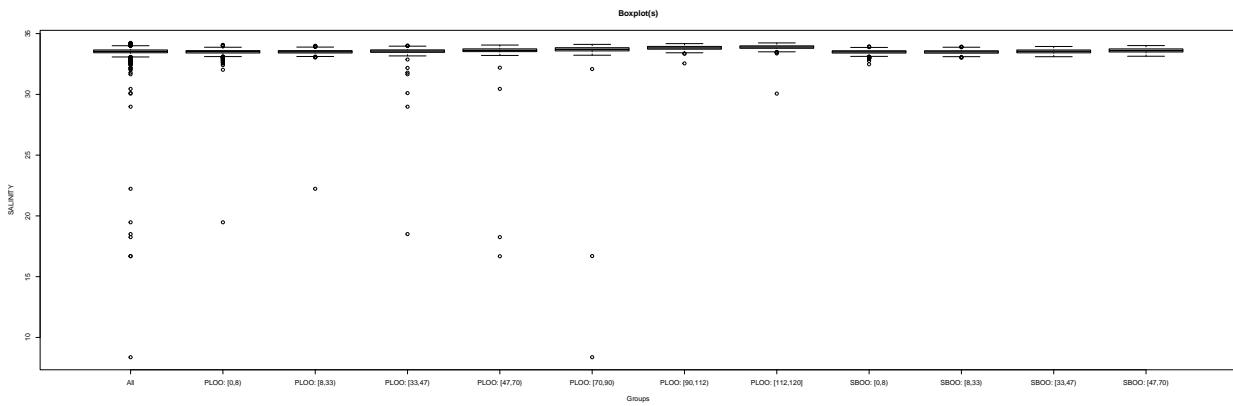
```

## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "PLOO: Unknown", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [70,90)", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [90,112)", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [112,120]", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: Unknown", class =
##   replacement has 1 row, data has 0

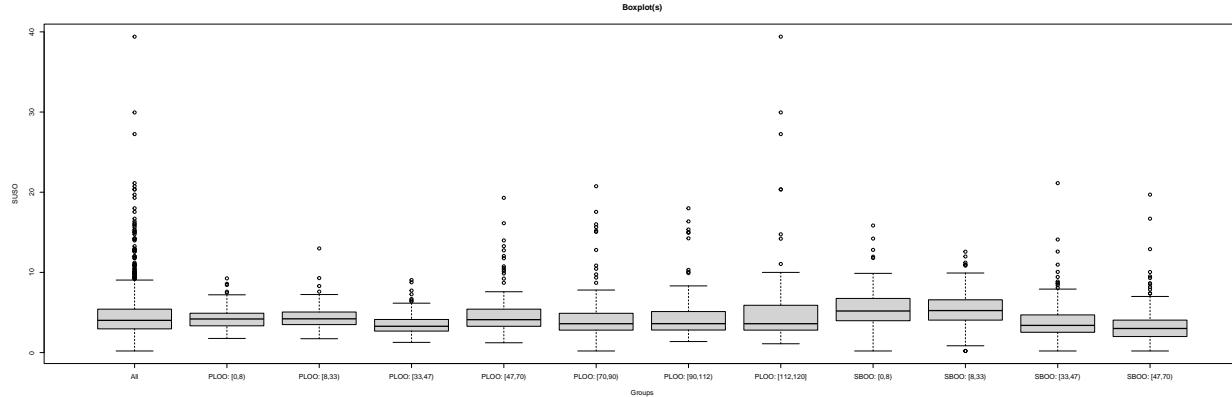
```



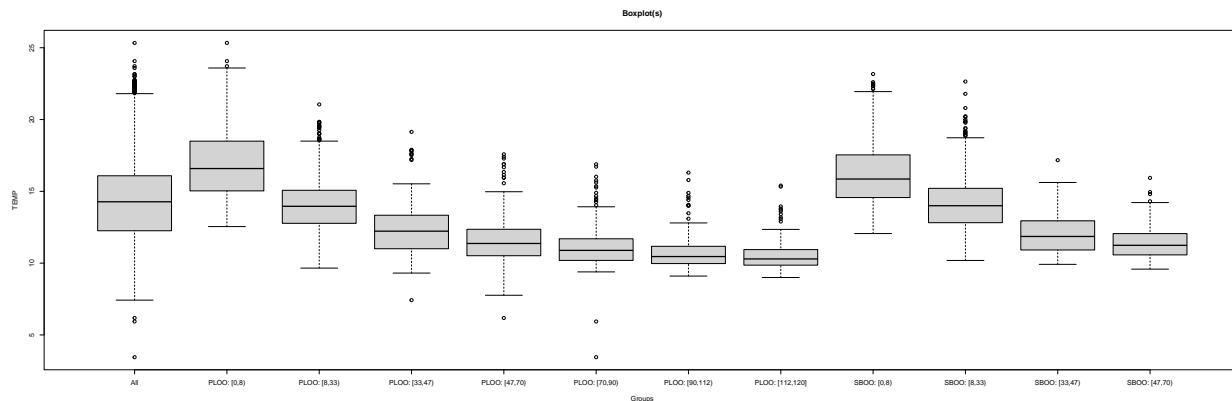
```
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "PLOO: Unknown", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [70,90]", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [90,112]", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [112,120]", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: Unknown", class =
##   replacement has 1 row, data has 0
```



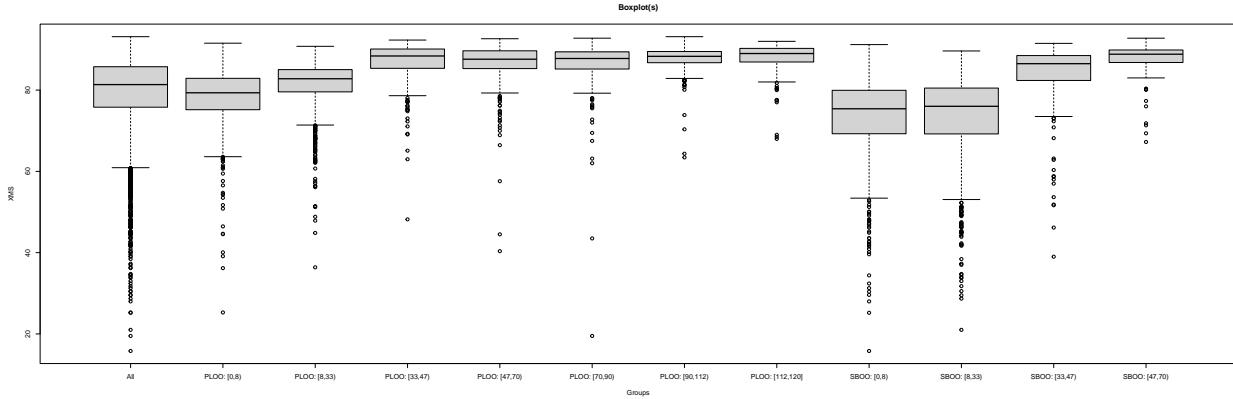
```
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "PLOO: Unknown", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [70,90]", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [90,112]", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: [112,120]", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SBOO: Unknown", class =
##   replacement has 1 row, data has 0
```



```
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "PLOO: Unknown", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SB00: [70,90]", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SB00: [90,112]", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SB00: [112,120]", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SB00: Unknown", class =
##   replacement has 1 row, data has 0
```



```
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "PLOO: Unknown", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SB00: [70,90]", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SB00: [90,112]", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SB00: [112,120]", class =
##   replacement has 1 row, data has 0
## Error in `$<-.data.frame`(`*tmp*`, "Groups", value = structure(1L, levels = "SB00: Unknown", class =
##   replacement has 1 row, data has 0
```



```

##   date_sample project depth_m_bin CHLOROPHYLL DENSITY      DO      ENTERO
## 1 1990-11-19    PL00      [8,33)       0.87 23.855 6.550000     NA
## 2 1990-11-19    SB00      [0,8)        1.27 23.853 7.410000     NA
## 3 1991-01-07    PL00      [0,8)        NA     NA     NA 25.74074
## 4 1991-01-07    PL00      [8,33)       NA     NA     NA 24.58333
## 5 1991-01-14    PL00      [0,8)        NA     NA 5.550000 36.09512
## 6 1991-01-14    PL00      [33,47)      NA     NA 5.347222 66.25000
## 7 1991-01-14    PL00      [47,70)      NA     NA 5.300000 13.33333
##          FECAL OG      PH SALINITY SUSO      TEMP      XMS
## 1         NA NA 8.080000 33.61700     NA 19.43000 44.85000
## 2         NA NA 8.180000 33.57400     NA 19.31000 75.60000
## 3 90.18519 NA         NA        NA 14.52593 80.29630
## 4 84.58333 NA         NA        NA 14.27917 85.70833
## 5 33.47643 NA 8.200000 33.49806     NA 14.19729 82.40221
## 6 352.50000 NA 8.200000 33.50861     NA 14.15278 82.25000
## 7 31.66667 NA 8.192857 33.52500     NA 13.70000 85.50000
##             vars     n   mean      sd median trimmed  mad   min     max
## date_sample      1 10371    NaN     NA    NaN  NA Inf -Inf
## project*        2 10371  1.36    0.48   1.00  1.33  0.00  1.00  2.00
## depth_m_bin*     3 10371  4.69    2.66   6.00  4.74  2.97  1.00  8.00
## CHLOROPHYLL     4  5198   3.50    3.91   2.34  2.76  1.97  0.00 49.89
## DENSITY          5  5282  24.93    0.69  24.92  24.95  0.68 22.39 26.48
## DO               6  5851   6.72    1.66   7.14  6.84  1.49  0.00 11.96
## ENTERO           7  9986  92.65  572.67   3.16  9.41  1.72  1.93 12000.00
## FECAL            8  9842 175.67  733.02   4.00 21.43  2.97  1.00 12000.00
## OG               9   458   0.24    0.20   0.20  0.20  0.00  0.20  2.65
## PH               10  5856   8.02    0.18   8.06  8.03  0.17  7.33  8.56
## SALINITY          11  5851  33.52    0.62  33.54  33.54  0.17  8.38 34.23
## SUSO              12  1709   4.54    2.80   4.02  4.19  1.74  0.20 39.40
## TEMP              13  6958  14.37    2.78  14.27  14.23  2.84  3.45 25.34
## XMS              14  6937  79.59    8.90  81.36  80.71  7.23 15.80 93.15
##             range   skew kurtosis   se
## date_sample    -Inf    NA     NA  NA
## project*       1.00   0.56  -1.68  0.00
## depth_m_bin*    7.00  -0.21  -1.42  0.03
## CHLOROPHYLL  49.89   3.74  22.93  0.05
## DENSITY         4.09  -0.29  -0.15  0.01
## DO              11.96  -0.59  -0.20  0.02
## ENTERO          11998.07 10.78 141.08  5.73
## FECAL           11999.00   7.39 68.78  7.39

```

```

## OG          2.45   9.07   92.51 0.01
## PH          1.24  -0.62    0.11 0.00
## SALINITY    25.86 -26.02   816.01 0.01
## SUSO         39.20   3.44   24.87 0.07
## TEMP         21.89   0.42   -0.15 0.03
## XMS          77.35  -1.69    4.81 0.11
##           date_sample project depth_m_bin parameter      Avg
## Not NA n     121286   121286    121286   121286 119635.0000
## NA n          0        0        0        0       1651.0000
## Not NA %      1        1        1        1       0.9864
## NA %          0        0        0        0       0.0136

# Define function that imputes the missing values to parameter dataframe using linear
# interpolation
impute_missing_values <- function(df_data) {

  params <- c("CHLOROPHYLL",
             "DENSITY",
             "DO",
             "ENTERO",
             "FECAL",
             "OG",
             "PH",
             "SALINITY",
             "SUSO",
             "TEMP",
             "XMS")

  for (param in params) {
    print(head(df_data, 7))
    print(param)
    print(date_sample)

    x <- zoo(df_data$param, df_data$date_sample)
    df_data$param <- na.interpolation(x, option = "linear")
    return(df_data)
  }
}

# remove NAs
# Linear Interpolation

# owt_df02_gb04_mrgd
x <- zoo(owt_df02_gb04_mrgd$CHLOROPHYLL, owt_df02_gb04_mrgd$date_sample)
owt_df02_gb04_mrgd$CHLOROPHYLL <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb04_mrgd$DENSITY, owt_df02_gb04_mrgd$date_sample)
owt_df02_gb04_mrgd$DENSITY <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb04_mrgd$DO, owt_df02_gb04_mrgd$date_sample)
owt_df02_gb04_mrgd$DO <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb04_mrgd$ENTERO, owt_df02_gb04_mrgd$date_sample)

```

```

owt_df02_gb04_mrgd$ENTERO <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb04_mrgd$FECAL, owt_df02_gb04_mrgd$date_sample)
owt_df02_gb04_mrgd$FECAL <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb04_mrgd$OG, owt_df02_gb04_mrgd$date_sample)
owt_df02_gb04_mrgd$OG <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb04_mrgd$PH, owt_df02_gb04_mrgd$date_sample)
owt_df02_gb04_mrgd$PH <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb04_mrgd$SALINITY, owt_df02_gb04_mrgd$date_sample)
owt_df02_gb04_mrgd$SALINITY <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb04_mrgd$SUSO, owt_df02_gb04_mrgd$date_sample)
owt_df02_gb04_mrgd$SUSO <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb04_mrgd$TEMP, owt_df02_gb04_mrgd$date_sample)
owt_df02_gb04_mrgd$TEMP <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb04_mrgd$XMS, owt_df02_gb04_mrgd$date_sample)
owt_df02_gb04_mrgd$XMS <- na.interpolation(x, option = "linear")

# owt_df02_gb04_wkly
x <- zoo(owt_df02_gb04_wkly$CHLOROPHYLL, owt_df02_gb04_wkly$date_sample)
owt_df02_gb04_wkly$CHLOROPHYLL <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb04_wkly$DENSITY, owt_df02_gb04_wkly$date_sample)
owt_df02_gb04_wkly$DENSITY <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb04_wkly$DO, owt_df02_gb04_wkly$date_sample)
owt_df02_gb04_wkly$DO <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb04_wkly$ENTERO, owt_df02_gb04_wkly$date_sample)
owt_df02_gb04_wkly$ENTERO <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb04_wkly$FECAL, owt_df02_gb04_wkly$date_sample)
owt_df02_gb04_wkly$FECAL <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb04_wkly$OG, owt_df02_gb04_wkly$date_sample)
owt_df02_gb04_wkly$OG <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb04_wkly$PH, owt_df02_gb04_wkly$date_sample)
owt_df02_gb04_wkly$PH <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb04_wkly$SALINITY, owt_df02_gb04_wkly$date_sample)
owt_df02_gb04_wkly$SALINITY <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb04_wkly$SUSO, owt_df02_gb04_wkly$date_sample)
owt_df02_gb04_wkly$SUSO <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb04_wkly$TEMP, owt_df02_gb04_wkly$date_sample)

```

```

owt_df02_gb04_wkly$TEMP <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb04_wkly$XMS, owt_df02_gb04_wkly$date_sample)
owt_df02_gb04_wkly$XMS <- na.interpolation(x, option = "linear")

# owt_df02_gb09_mrgd
x <- zoo(owt_df02_gb09_mrgd$CHLOROPHYLL, owt_df02_gb09_mrgd$date_sample)
owt_df02_gb09_mrgd$CHLOROPHYLL <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb09_mrgd$DENSITY, owt_df02_gb09_mrgd$date_sample)
owt_df02_gb09_mrgd$DENSITY <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb09_mrgd$D0, owt_df02_gb09_mrgd$date_sample)
owt_df02_gb09_mrgd$D0 <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb09_mrgd$ENTERO, owt_df02_gb09_mrgd$date_sample)
owt_df02_gb09_mrgd$ENTERO <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb09_mrgd$FECAL, owt_df02_gb09_mrgd$date_sample)
owt_df02_gb09_mrgd$FECAL <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb09_mrgd$OG, owt_df02_gb09_mrgd$date_sample)
owt_df02_gb09_mrgd$OG <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb09_mrgd$PH, owt_df02_gb09_mrgd$date_sample)
owt_df02_gb09_mrgd$PH <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb09_mrgd$SALINITY, owt_df02_gb09_mrgd$date_sample)
owt_df02_gb09_mrgd$SALINITY <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb09_mrgd$SUSO, owt_df02_gb09_mrgd$date_sample)
owt_df02_gb09_mrgd$SUSO <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb09_mrgd$TEMP, owt_df02_gb09_mrgd$date_sample)
owt_df02_gb09_mrgd$TEMP <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb09_mrgd$XMS, owt_df02_gb09_mrgd$date_sample)
owt_df02_gb09_mrgd$XMS <- na.interpolation(x, option = "linear")

# owt_df02_gb09_wkly <- impute_missing_values(owt_df02_gb09_wkly)
x <- zoo(owt_df02_gb09_wkly$CHLOROPHYLL, owt_df02_gb09_wkly$date_sample)
owt_df02_gb09_wkly$CHLOROPHYLL <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb09_wkly$DENSITY, owt_df02_gb09_wkly$date_sample)
owt_df02_gb09_wkly$DENSITY <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb09_wkly$D0, owt_df02_gb09_wkly$date_sample)
owt_df02_gb09_wkly$D0 <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb09_wkly$ENTERO, owt_df02_gb09_wkly$date_sample)
owt_df02_gb09_wkly$ENTERO <- na.interpolation(x, option = "linear")

```

```

x <- zoo(owt_df02_gb09_wkly$FECAL, owt_df02_gb09_wkly$date_sample)
owt_df02_gb09_wkly$FECAL <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb09_wkly$OG, owt_df02_gb09_wkly$date_sample)
owt_df02_gb09_wkly$OG <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb09_wkly$PH, owt_df02_gb09_wkly$date_sample)
owt_df02_gb09_wkly$PH <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb09_wkly$SALINITY, owt_df02_gb09_wkly$date_sample)
owt_df02_gb09_wkly$SALINITY <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb09_wkly$SUSO, owt_df02_gb09_wkly$date_sample)
owt_df02_gb09_wkly$SUSO <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb09_wkly$TEMP, owt_df02_gb09_wkly$date_sample)
owt_df02_gb09_wkly$TEMP <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb09_wkly$XMS, owt_df02_gb09_wkly$date_sample)
owt_df02_gb09_wkly$XMS <- na.interpolation(x, option = "linear")

# owt_df02_gb07_mrgd
x <- zoo(owt_df02_gb07_mrgd$CHLOROPHYLL, owt_df02_gb07_mrgd$date_sample)
owt_df02_gb07_mrgd$CHLOROPHYLL <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb07_mrgd$DENSITY, owt_df02_gb07_mrgd$date_sample)
owt_df02_gb07_mrgd$DENSITY <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb07_mrgd$DO, owt_df02_gb07_mrgd$date_sample)
owt_df02_gb07_mrgd$DO <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb07_mrgd$ENTERO, owt_df02_gb07_mrgd$date_sample)
owt_df02_gb07_mrgd$ENTERO <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb07_mrgd$FECAL, owt_df02_gb07_mrgd$date_sample)
owt_df02_gb07_mrgd$FECAL <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb07_mrgd$OG, owt_df02_gb07_mrgd$date_sample)
owt_df02_gb07_mrgd$OG <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb07_mrgd$PH, owt_df02_gb07_mrgd$date_sample)
owt_df02_gb07_mrgd$PH <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb07_mrgd$SALINITY, owt_df02_gb07_mrgd$date_sample)
owt_df02_gb07_mrgd$SALINITY <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb07_mrgd$SUSO, owt_df02_gb07_mrgd$date_sample)
owt_df02_gb07_mrgd$SUSO <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb07_mrgd$TEMP, owt_df02_gb07_mrgd$date_sample)
owt_df02_gb07_mrgd$TEMP <- na.interpolation(x, option = "linear")

```

```

x <- zoo(owt_df02_gb07_mrgd$XMS, owt_df02_gb07_mrgd$date_sample)
owt_df02_gb07_mrgd$XMS <- na.interpolation(x, option = "linear")

# owt_df02_gb07_mrgd
x <- zoo(owt_df02_gb07_wkly$CHLOROPHYLL, owt_df02_gb07_wkly$date_sample)
owt_df02_gb07_wkly$CHLOROPHYLL <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb07_wkly$DENSITY, owt_df02_gb07_wkly$date_sample)
owt_df02_gb07_wkly$DENSITY <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb07_wkly$DO, owt_df02_gb07_wkly$date_sample)
owt_df02_gb07_wkly$DO <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb07_wkly$ENTERO, owt_df02_gb07_wkly$date_sample)
owt_df02_gb07_wkly$ENTERO <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb07_wkly$FECAL, owt_df02_gb07_wkly$date_sample)
owt_df02_gb07_wkly$FECAL <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb07_wkly$OG, owt_df02_gb07_wkly$date_sample)
owt_df02_gb07_wkly$OG <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb07_wkly$PH, owt_df02_gb07_wkly$date_sample)
owt_df02_gb07_wkly$PH <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb07_wkly$SALINITY, owt_df02_gb07_wkly$date_sample)
owt_df02_gb07_wkly$SALINITY <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb07_wkly$SUSO, owt_df02_gb07_wkly$date_sample)
owt_df02_gb07_wkly$SUSO <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb07_wkly$TEMP, owt_df02_gb07_wkly$date_sample)
owt_df02_gb07_wkly$TEMP <- na.interpolation(x, option = "linear")

x <- zoo(owt_df02_gb07_wkly$XMS, owt_df02_gb07_wkly$date_sample)
owt_df02_gb07_wkly$XMS <- na.interpolation(x, option = "linear")

# https://cran.r-project.org/web/packages/padr/vignettes/padr.html
#print(head(owt_df02_gb04_mrgd, 7))
#print(tail(owt_df02_gb04_mrgd, 7))

owt_df02_gb04_mrgd %>%
  pad()

##      date_sample CHLOROPHYLL    DENSITY        DO      ENTERO      FECAL
## 1 1990-11-15     1.0700000  23.85400  6.9800000  32.051282 125.641026
## 2 1990-11-16     1.0690476  23.85410  6.6792500  32.051282 125.641026
## 3 1990-11-17     1.0680952  23.85420  6.3785000  18.717949  51.282051
## 4 1990-11-18     1.0671429  23.85431  6.0777500 106.153846  80.769231
## 5 1990-11-19     1.0661905  23.85441  5.7770000  40.000000  40.555556
## 6 1990-11-20     1.0652381  23.85451  5.4762500  20.338983  40.508475

```

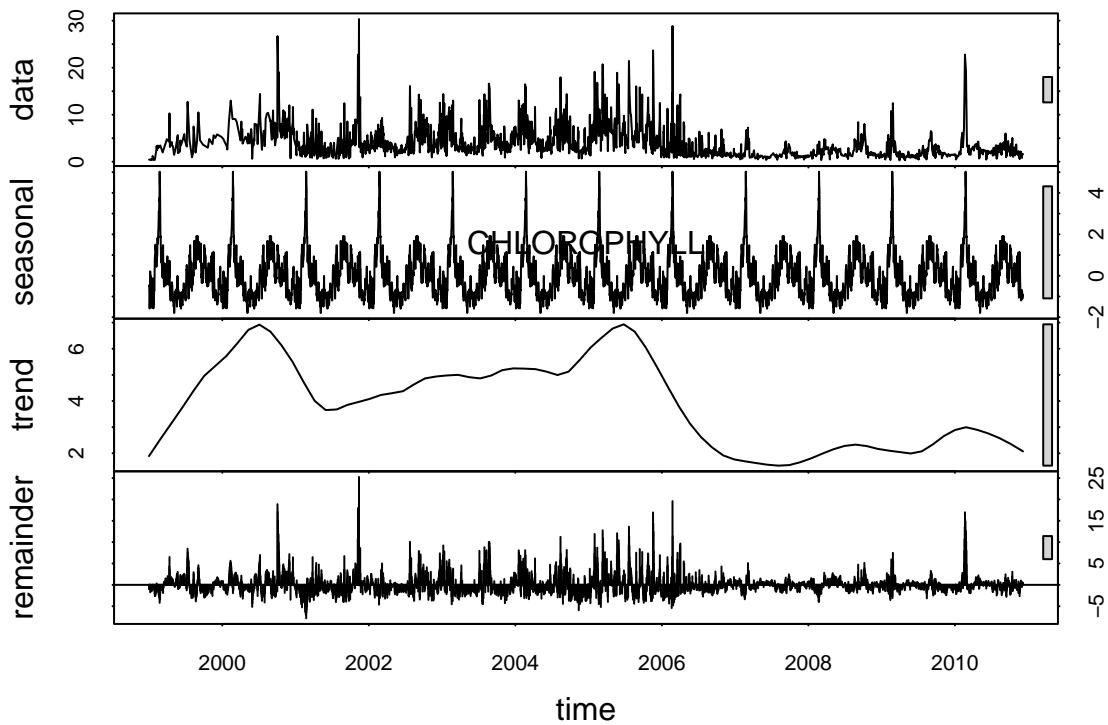
```

## 8305      NA      NA      NA      NA      NA      NA
## 8306      NA      NA      NA      NA      NA      NA
## 8307      NA      NA      NA      NA      NA      NA
## 8308      NA      NA      NA      NA      NA      NA
## 8309      NA      NA      NA      NA      NA      NA
## 8310      NA      NA      NA      NA      NA      NA
## 8311      NA      NA      NA      NA      NA      NA
## 8312      NA      NA      NA      NA      NA      NA
## 8313      NA      NA      NA      NA      NA      NA
## 8314      NA      NA      NA      NA      NA      NA
## 8315      NA      NA      NA      NA      NA      NA
## 8316      NA      NA      NA      NA      NA      NA
## 8317      NA      NA      NA      NA      NA      NA
## 8318      NA      NA      NA      NA      NA      NA
## 8319      NA      NA      NA      NA      NA      NA
## 8320      NA      NA      NA      NA      NA      NA
## 8321      NA      NA      NA      NA      NA      NA
## 8322      NA      NA      NA      NA      NA      NA
## 8323      NA      NA      NA      NA      NA      NA
## 8324      NA      NA      NA      NA      NA      NA
## 8325      NA      NA      NA      NA      NA      NA
## 8326      NA      NA      NA      NA      NA      NA
## 8327      NA      NA      NA      NA      NA      NA
## 8328      NA      NA      NA      NA      NA      NA
## 8329      NA      NA      NA      NA      NA      NA
## 8330      NA      NA      NA      NA      NA      NA
## 8331      NA      NA      NA      NA      NA      NA
## 8332      NA      NA      NA      NA      NA      NA
## 8333      NA      NA      NA      NA      NA      NA
## [ reached 'max' / getOption("max.print") -- omitted 3035 rows ]
gb04_start <- "1990-11-15"
print(lubridate::yday(ymd(gb04_start)))

## [1] 319

##### 5 YEAR - 1996 - 2001 #####
ts_chlorophyll <- ts(owt_df02_gb04_mrgd$CHLOROPHYLL, start = c(1996, 1), freq = 365)
ts_1996_2001_chl <- window(ts_chlorophyll, start=c(1999, 1))
stl_chl <- stl(ts_1996_2001_chl, s.window="periodic")
#labels.chart = seq(as.Date("1990-11-15"), as.Date("2021-12-29"), by = "weeks")
plot(stl_chl)
mtext("CHLOROPHYLL", side=3, line = -5)

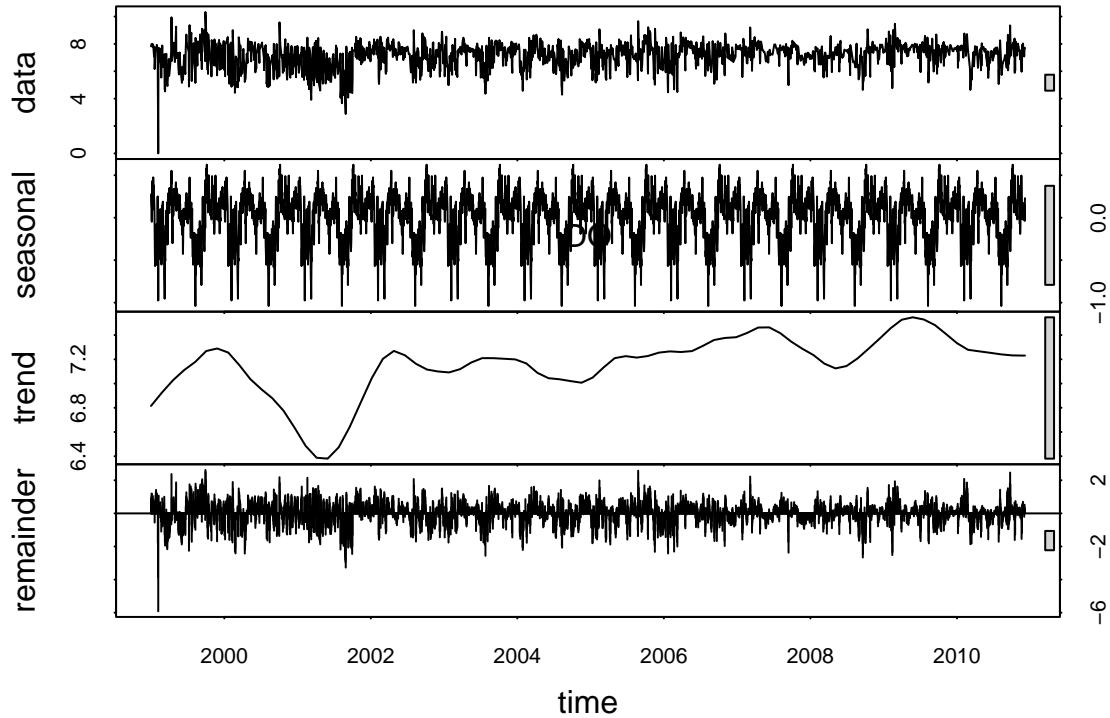
```



```

ts_do <- ts(owt_df02_gb04_mrgd$D0, start = c(1996, 1), freq = 365)
ts_1996_2001_do <- window(ts_do, start=c(1999, 1))
stl_do <- stl(ts_1996_2001_do, s.window="periodic")
#labels.chart = seq(as.Date("1990-11-15"), as.Date("2021-12-29"), by = "weeks")
plot(stl_do)
mtext("D0", side=3, line = -5)

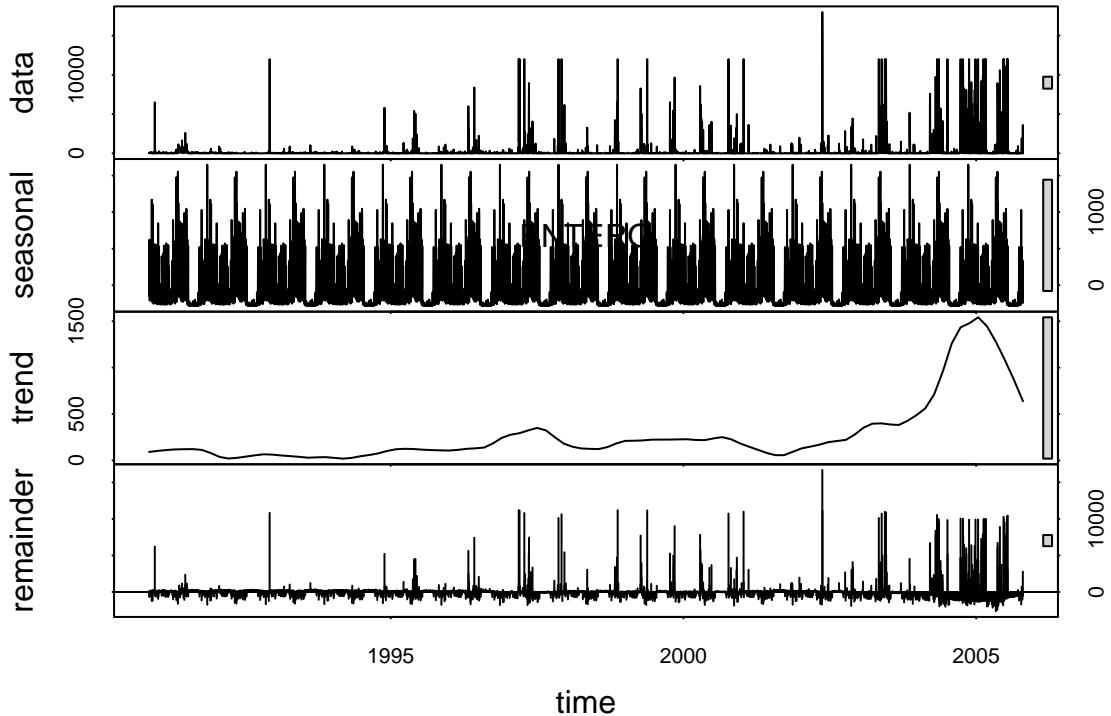
```



```

ts_ent <- ts(owt_df02_gb04_mrgd$ENTERO, start = c(1990, 319), freq = 365)
#ts_1996_2001_ent <- window(ts_ent, start=c(1999, 1))
stl_ent <- stl(ts_ent, s.window="periodic")
#labels.chart = seq(as.Date("1990-11-15"), as.Date("2021-12-29"), by = "weeks")
plot(stl_ent)
mtext("ENTERO", side=3, line = -5)

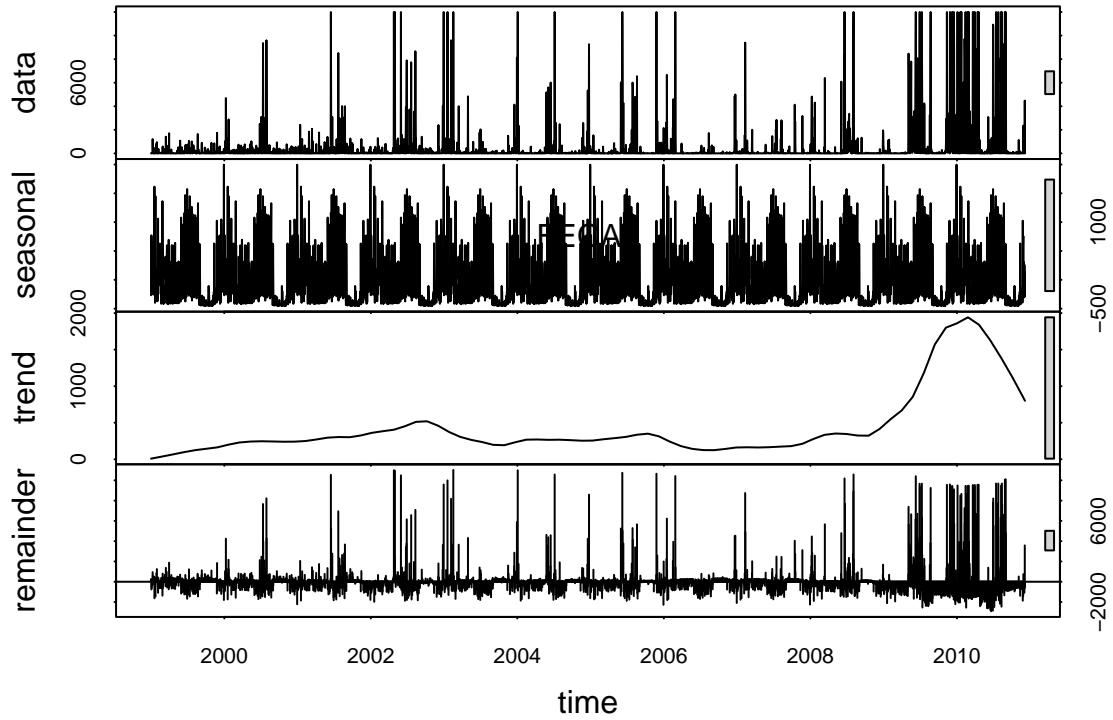
```



```

ts_fecal <- ts(owt_df02_gb04_mrgd$FECAL, start = c(1996, 1), freq = 365)
ts_1996_2001_fecal <- window(ts_fecal, start=c(1999, 1))
stl_fecal <- stl(ts_1996_2001_fecal, s.window="periodic")
#labels.chart = seq(as.Date("1990-11-15"), as.Date("2021-12-29"), by = "weeks")
plot(stl_fecal)
mtext("FECAL", side=3, line = -5)

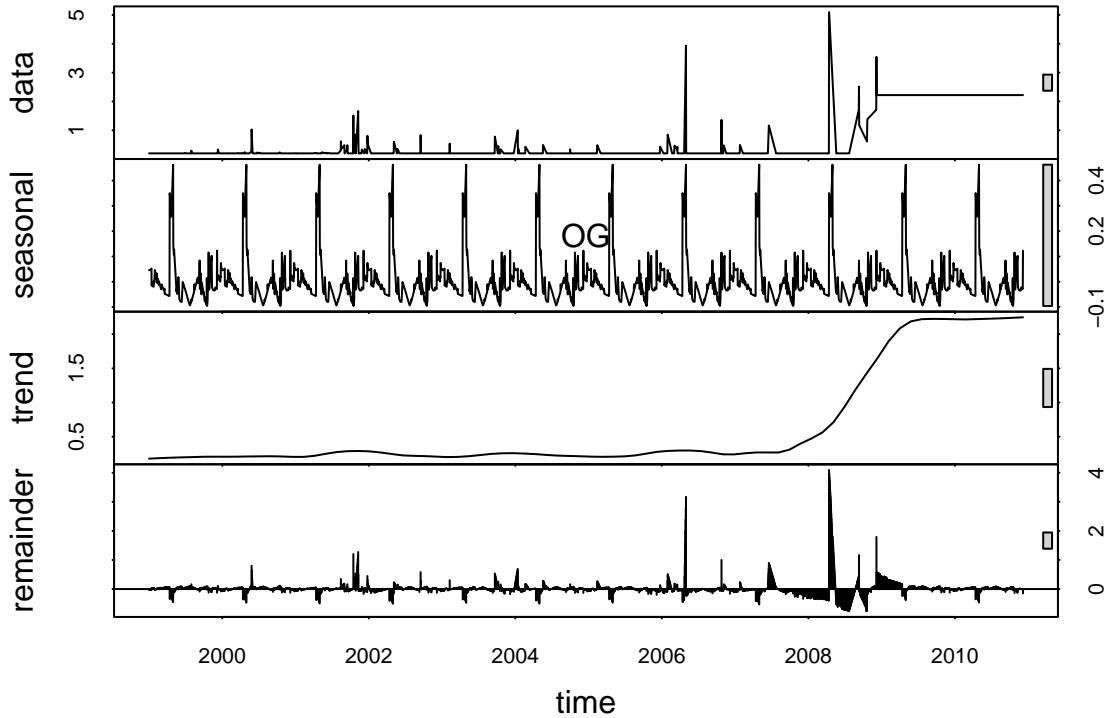
```



```

ts_og <- ts(owt_df02_gb04_mrgd$OG, start = c(1996, 1), freq = 365)
ts_1996_2001_og <- window(ts_og, start=c(1999, 1))
stl_og <- stl(ts_1996_2001_og, s.window="periodic")
#labels.chart = seq(as.Date("1990-11-15"), as.Date("2021-12-29"), by = "weeks")
plot(stl_og)
mtext("OG", side=3, line = -5)

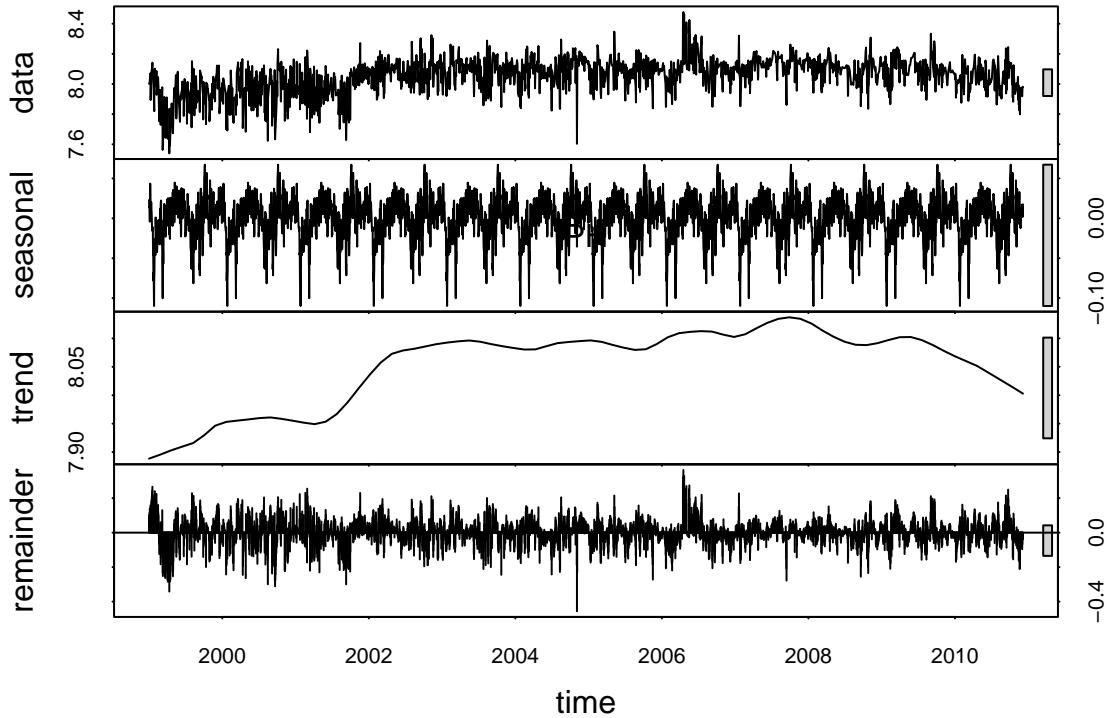
```



```

ts_ph <- ts(owt_df02_gb04_mrgd$PH, start = c(1996, 1), freq = 365)
ts_1996_2001_ph <- window(ts_ph, start=c(1999, 1))
stl_ph <- stl(ts_1996_2001_ph, s.window="periodic")
#labels.chart = seq(as.Date("1990-11-15"), as.Date("2021-12-29"), by = "weeks")
plot(stl_ph)
mtext("PH", side=3, line = -5)

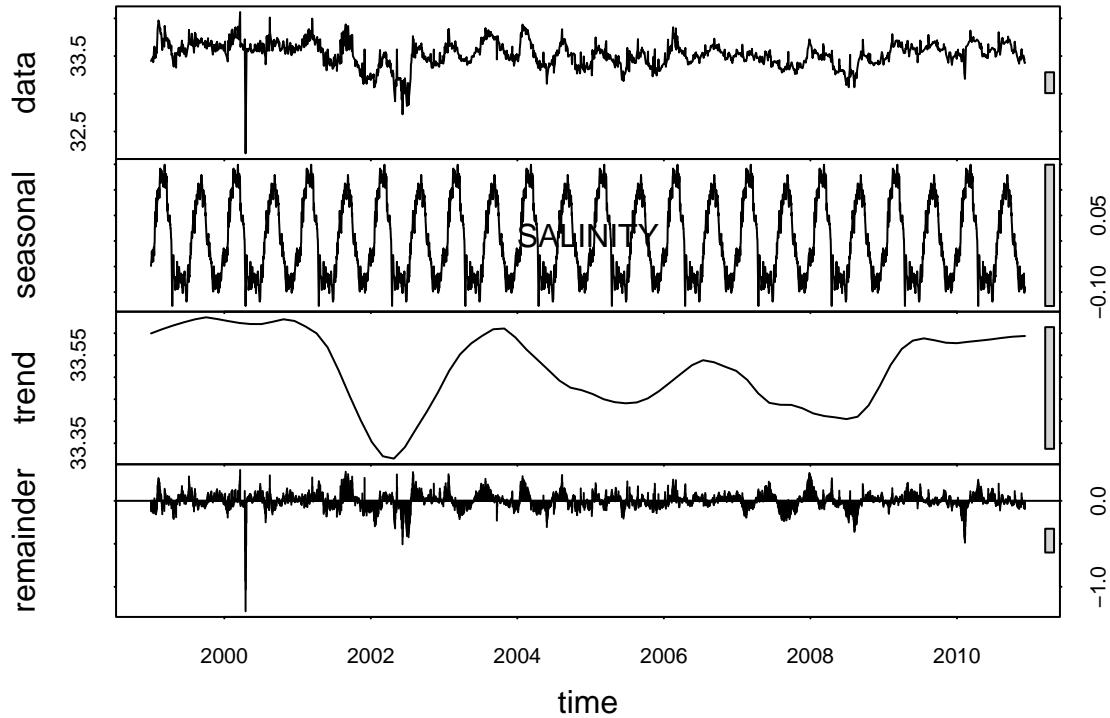
```



```

ts_salinity <- ts(owt_df02_gb04_mrgd$SALINITY, start = c(1996, 1), freq = 365)
ts_1996_2001_salinity <- window(ts_salinity, start=c(1999, 1))
stl_salinity <- stl(ts_1996_2001_salinity, s.window="periodic")
#labels.chart = seq(as.Date("1990-11-15"), as.Date("2021-12-29"), by = "weeks")
plot(stl_salinity)
mtext("SALINITY", side=3, line = -5)

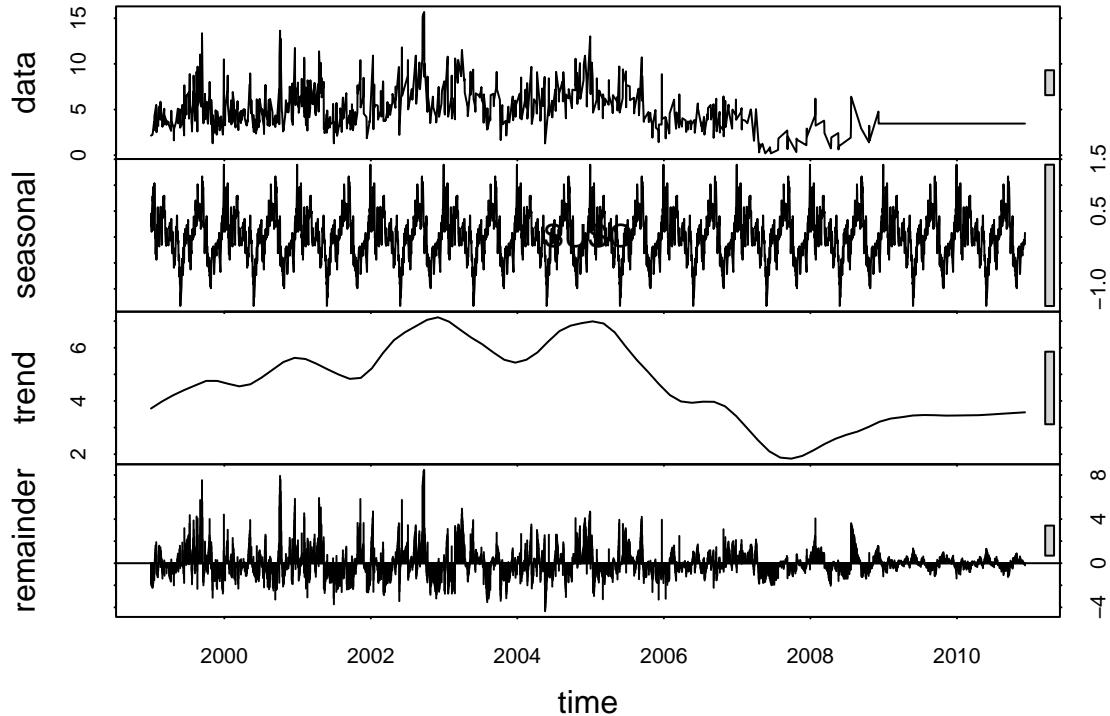
```



```

ts_suso <- ts(owt_df02_gb04_mrgd$SUSO, start = c(1996, 1), freq = 365)
ts_1996_2001_suso <- window(ts_suso, start=c(1999, 1))
stl_suso <- stl(ts_1996_2001_suso, s.window="periodic")
#labels.chart = seq(as.Date("1990-11-15"), as.Date("2021-12-29"), by = "weeks")
plot(stl_suso)
mtext("SUSO", side=3, line = -5)

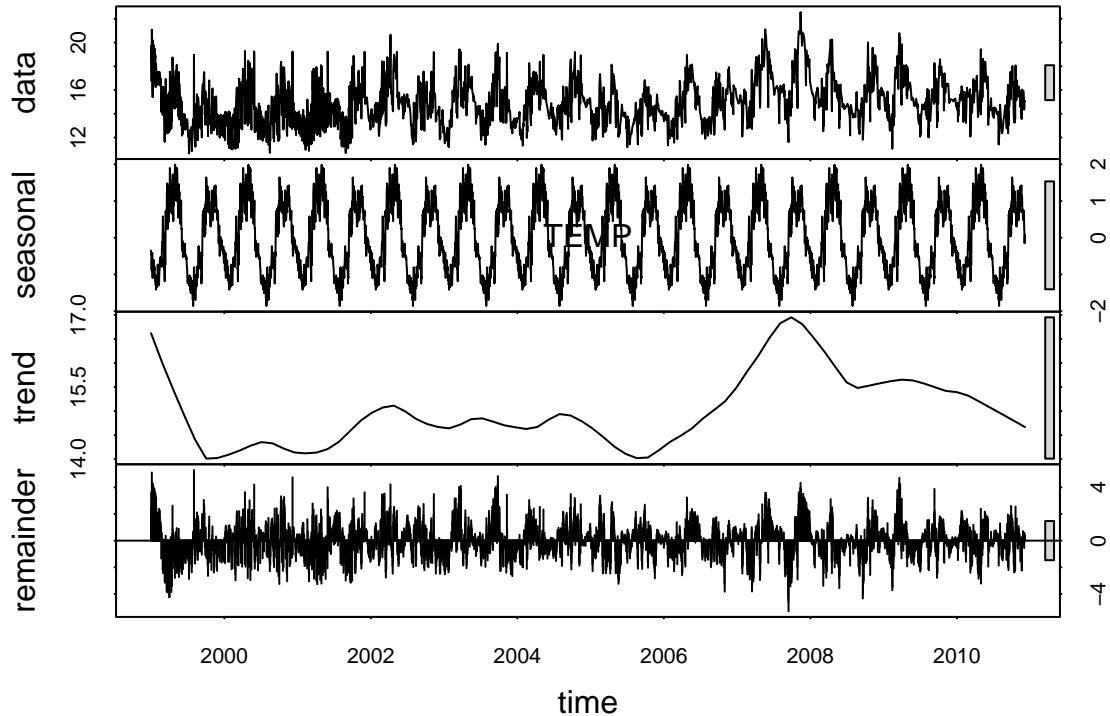
```



```

ts_temp <- ts(owt_df02_gb04_mrgd$TEMP, start = c(1996, 1), freq = 365)
ts_1996_2001_temp <- window(ts_temp, start=c(1999, 1))
stl_temp <- stl(ts_1996_2001_temp, s.window="periodic")
#labels.chart = seq(as.Date("1990-11-15"), as.Date("2021-12-29"), by = "weeks")
plot(stl_temp)
mtext("TEMP", side=3, line = -5)

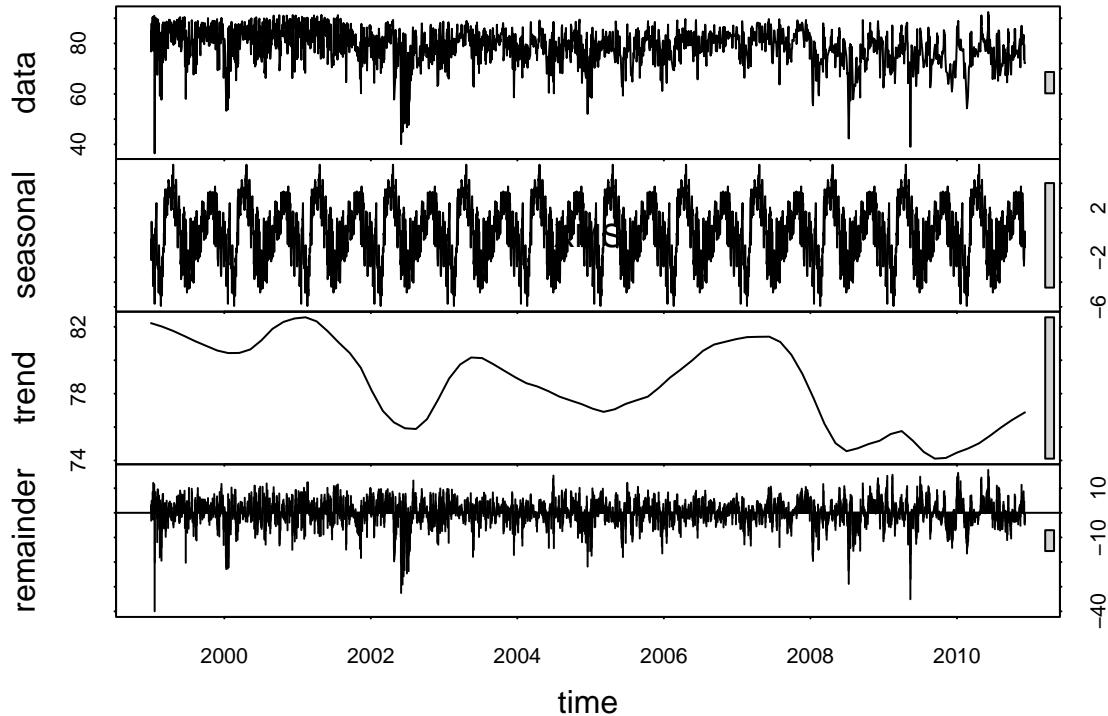
```



```

ts_xms <- ts(owt_df02_gb04_mrgd$XMS, start = c(1996, 1), freq = 365)
ts_1996_2001_xms <- window(ts_xms, start=c(1999, 1))
stl_xms <- stl(ts_1996_2001_xms, s.window="periodic")
#labels.chart = seq(as.Date("1990-11-15"), as.Date("2021-12-29"), by = "weeks")
plot(stl_xms)
mtext("XMS", side=3, line = -5)

```



```
# All parameters exhibited seasonal cycles within a year range, so
# it would make sense to have several years worth of datapoints for the training period.
# This set below can be used for all parameters.
train <- head(owt_df02_gb04_mrgd, round(NROW(owt_df02_gb04_mrgd) * 0.6))
h <- length(owt_df02_gb04_mrgd) - NROW(train)
test <- tail(owt_df02_gb04_mrgd, h)
```

Define custom function to produce scatterplots comparing multiple predictors to one target

```
param_lst03 <- c("CHLOROPHYLL",
                 "DENSITY",
                 "DO",
                 "FECAL",
                 "OG",
                 "PH",
                 "SALINITY",
                 "SUSO",
                 "TEMP",
                 "XMS")

scatr <- function(df = NA,
                  y = NA,
                  x_lst = c()) {
  for (p in x_lst) {
```

```

plot <- ggplot(df, aes(df[, p], df[, y])) +
  geom_point() +
  stat_smooth(method = "lm",
              formula = y ~ x,
              geom = "smooth") +
  labs(x = p,
       y = y,
       title = "Figure",
       subtitle = paste0("Scatterplot of ", p, " and ", y, " Series"))
  print(plot)
}
}

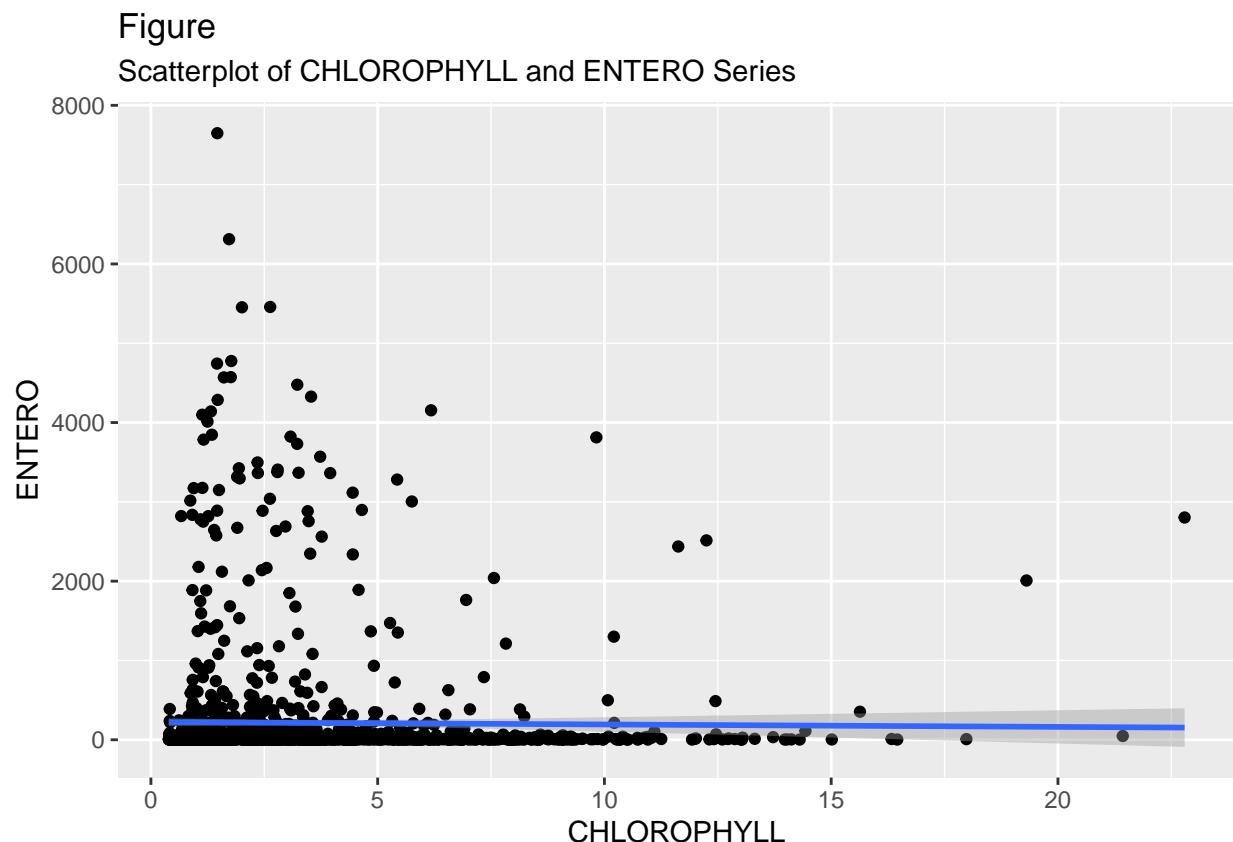
```

Run custom function on data aggregated by date_sample and parameter

```

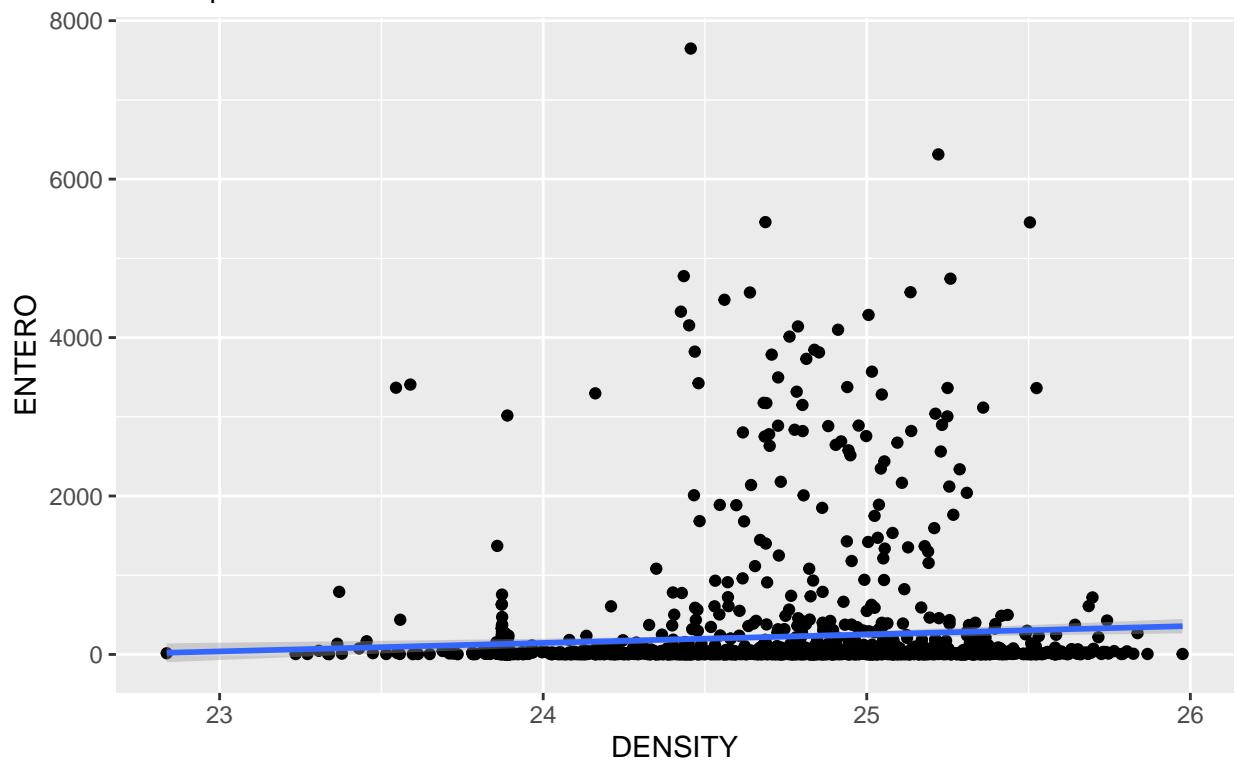
# All data
scatr(df = owt_df02_gb04_wkly,
      y = "ENTERO",
      x_lst = param_lst03)

```



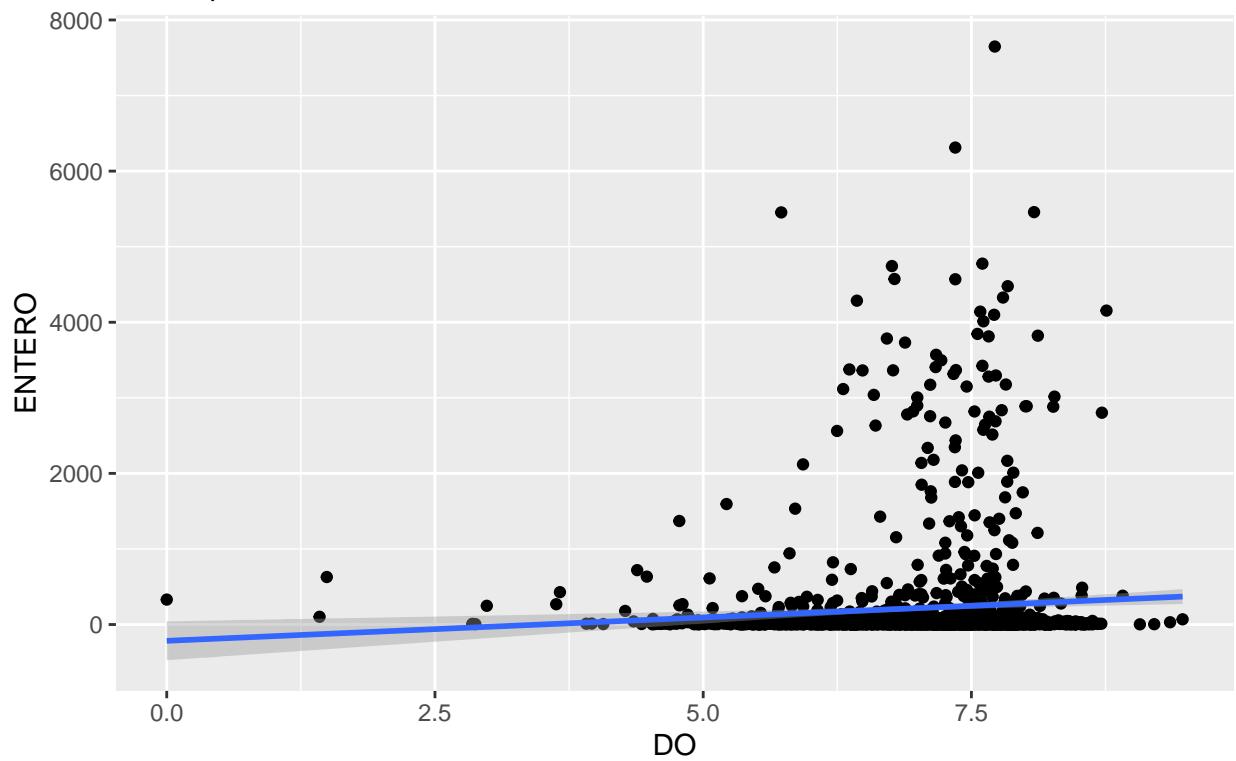
Figure

Scatterplot of DENSITY and ENTERO Series



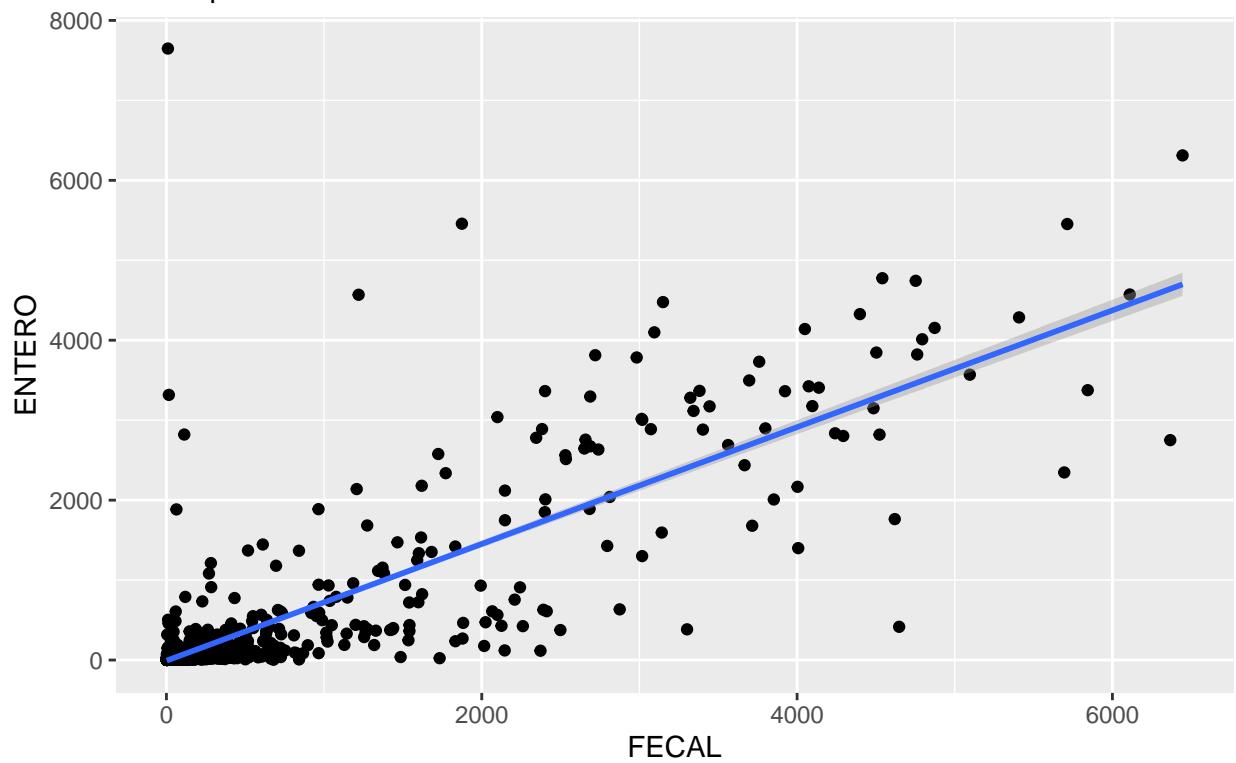
Figure

Scatterplot of DO and ENTERO Series



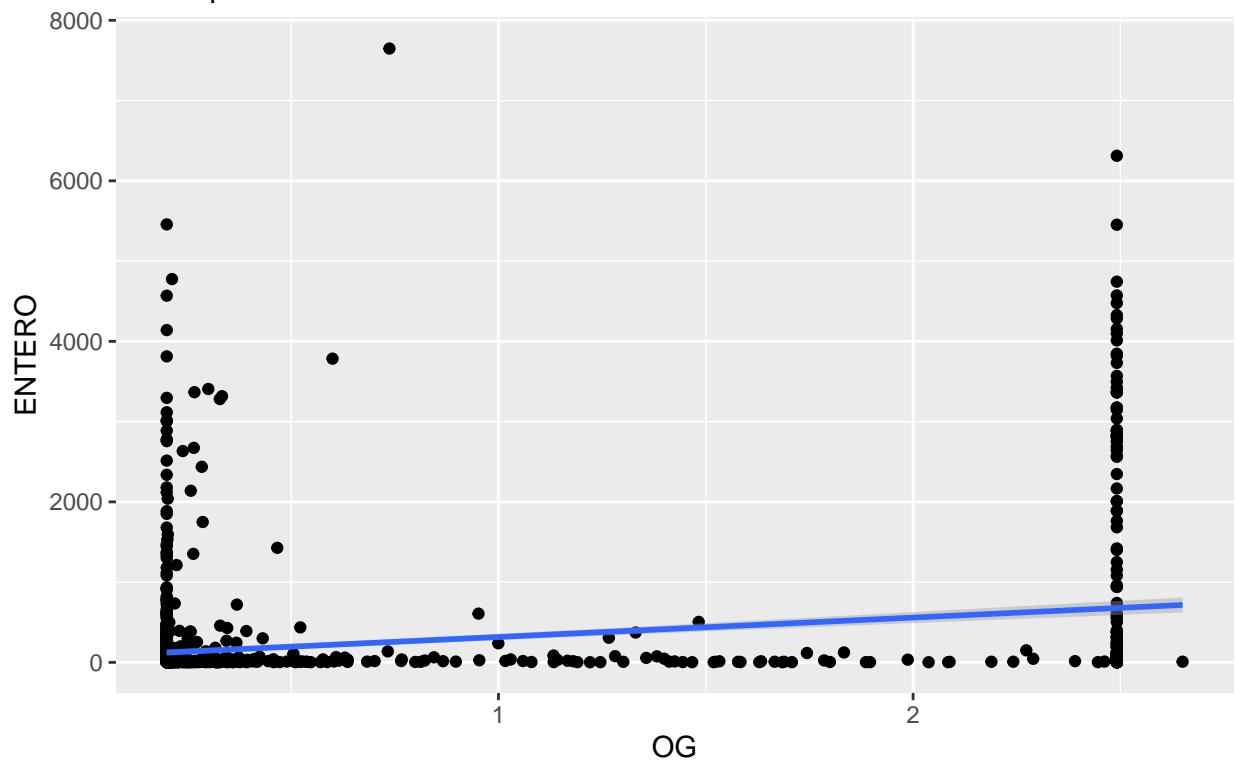
Figure

Scatterplot of FECAL and ENTERO Series



Figure

Scatterplot of OG and ENTERO Series



Figure

Scatterplot of PH and ENTERO Series

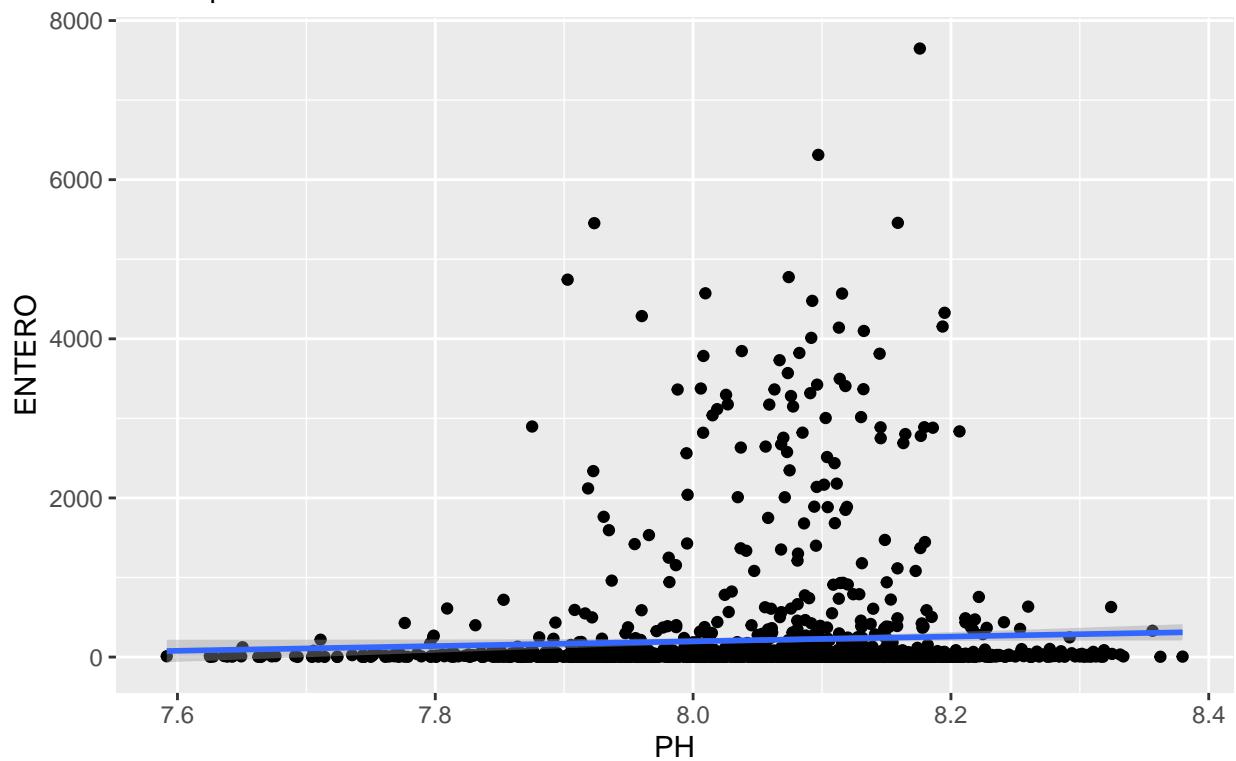


Figure
Scatterplot of SALINITY and ENTERO Series

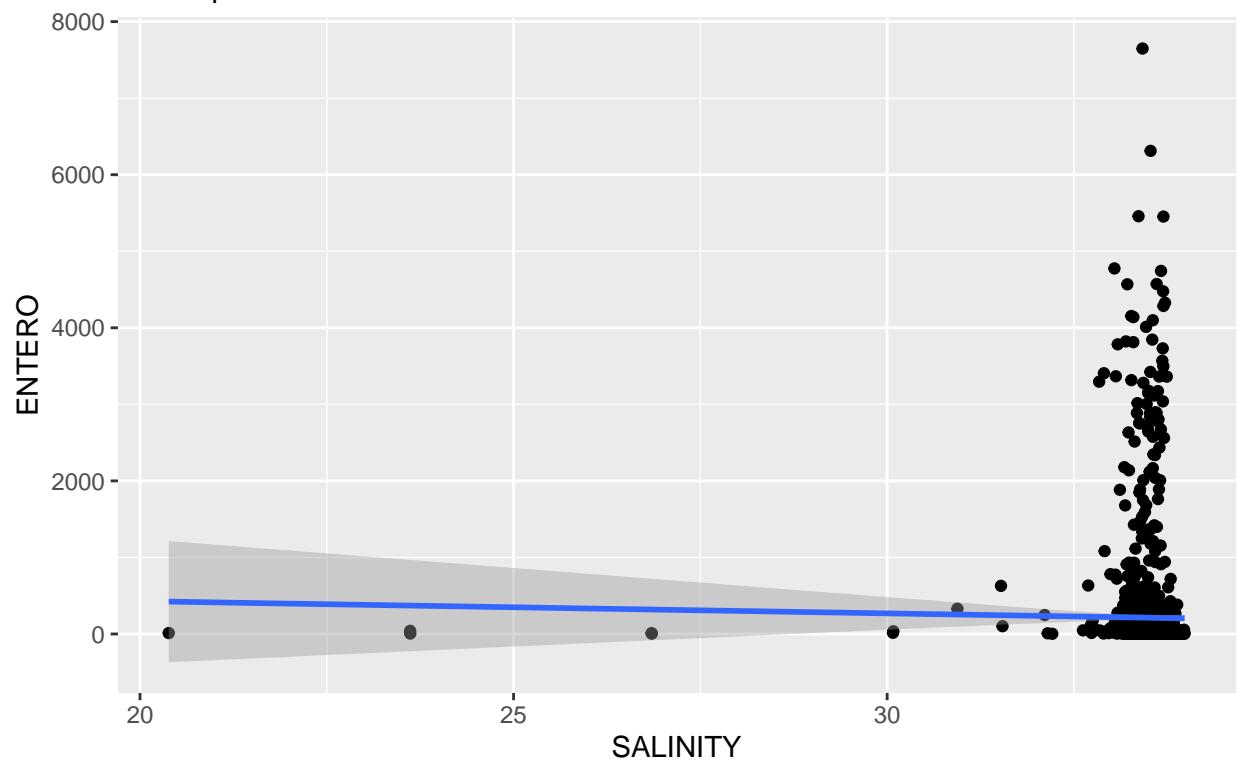


Figure
Scatterplot of SUSO and ENTERO Series

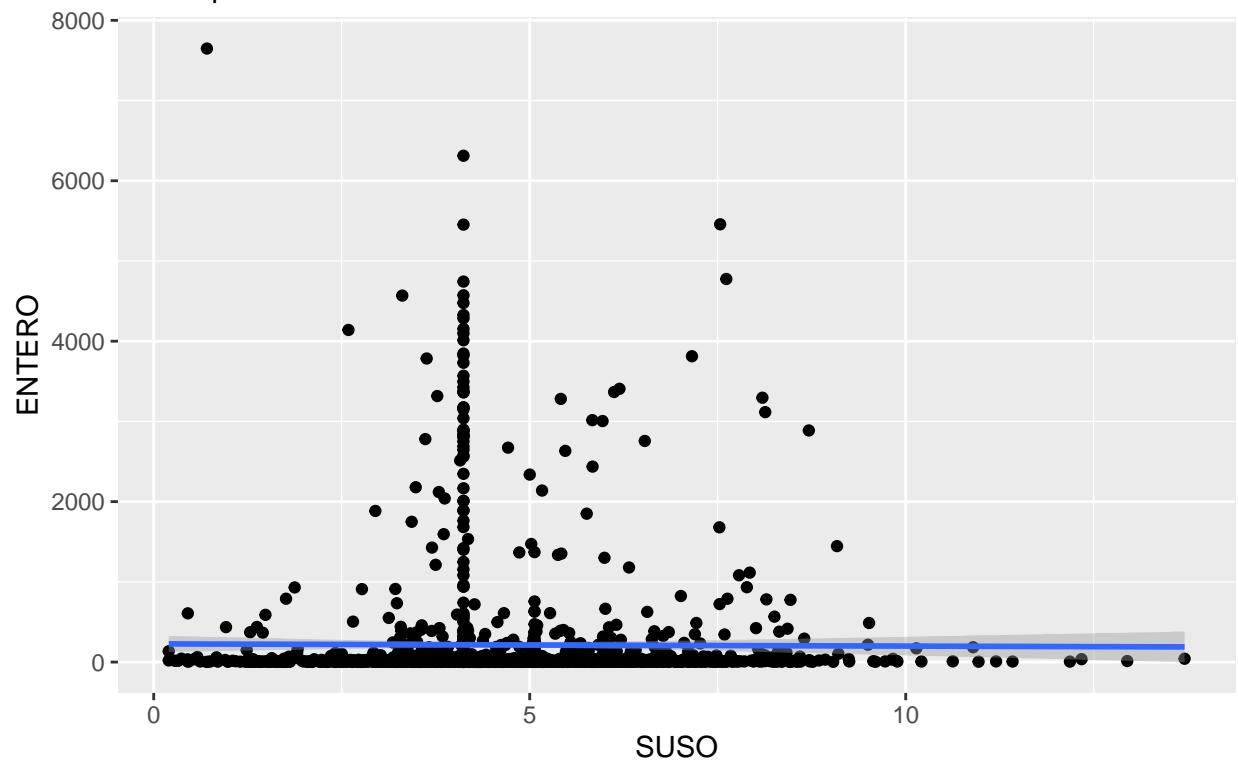
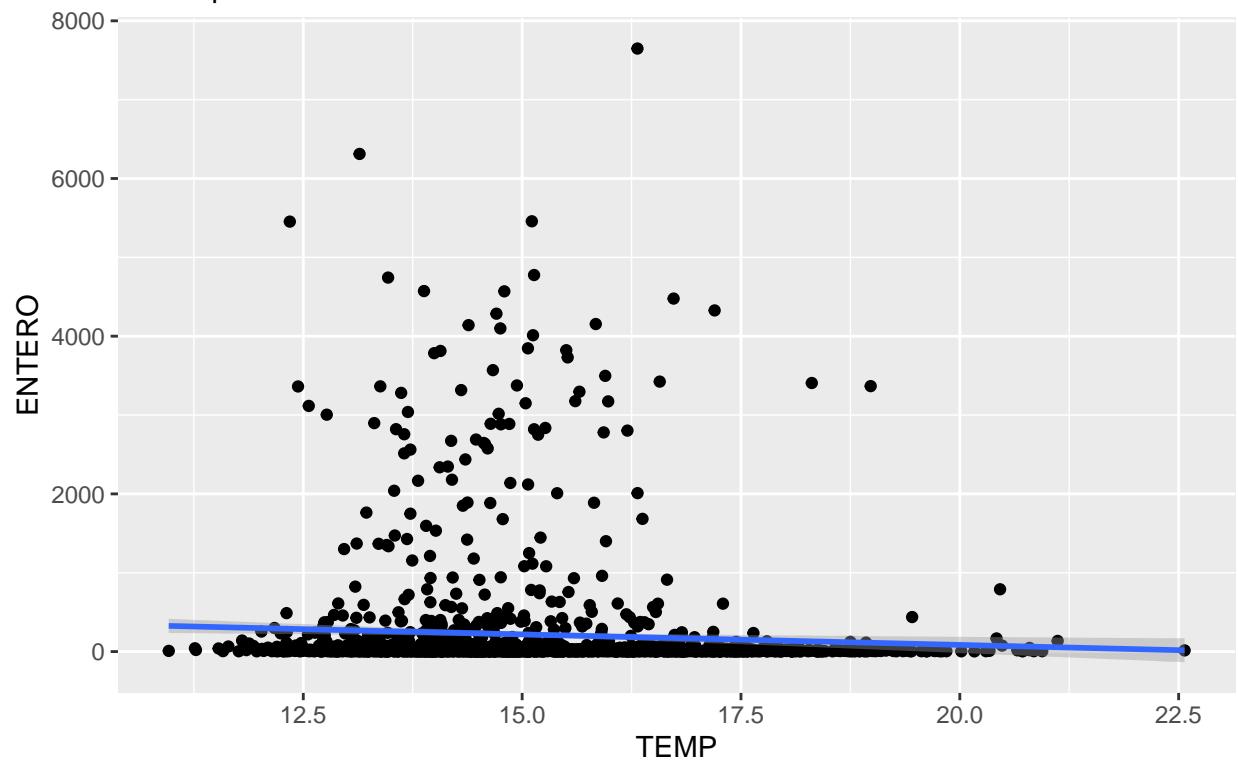
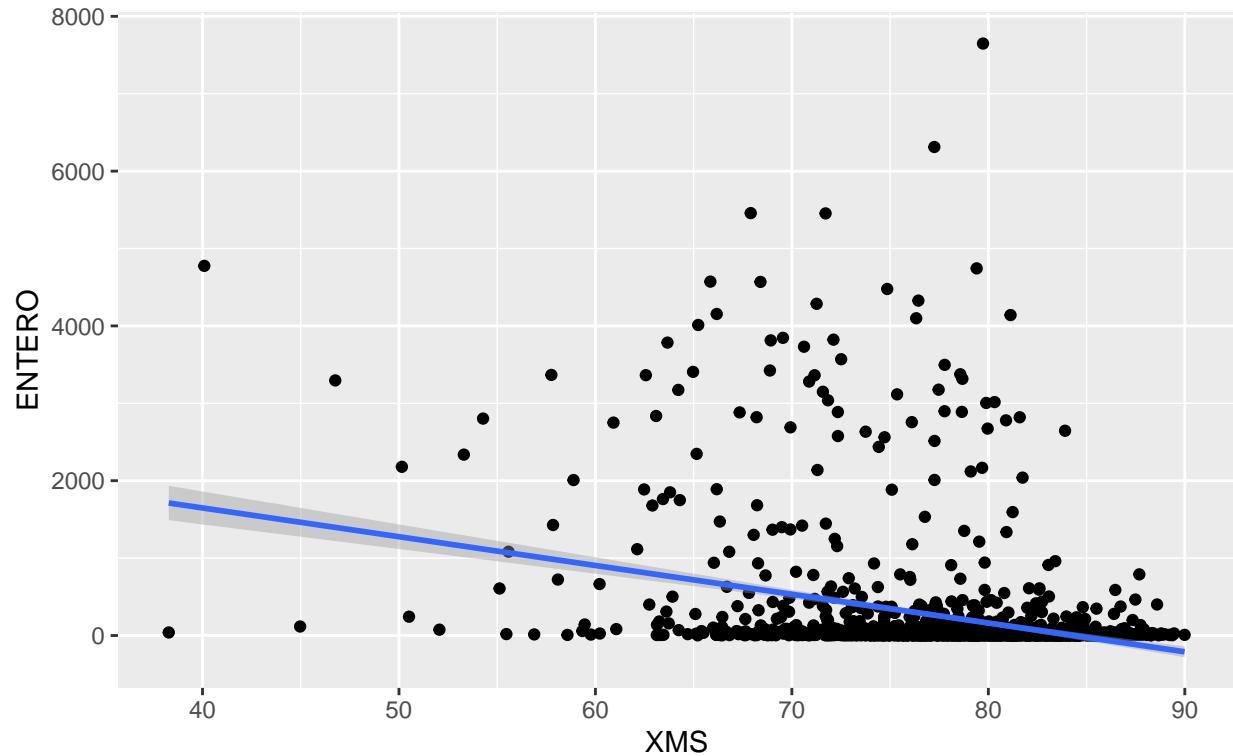


Figure
Scatterplot of TEMP and ENTERO Series



Figure

Scatterplot of XMS and ENTERO Series



```
# Excluding outliers
print(head(owt_df02_gb04_wkly02, 7))
```

```
##   date_sample CHLOROPHYLL DENSITY      DO     ENTERO    FECAL OG     PH
## 1 1990-11-19       1.07 23.854 6.980000      NA      NA NA 8.130
## 2 1991-01-07        NA     NA      NA 25.38462 88.46154 NA   NA
## 3 1991-01-14        NA     NA 5.445417 58.12973 137.21730 NA 8.195
## 4 1991-01-21        NA     NA      NA 23.20513 59.61538 NA   NA
## 5 1991-01-28        NA     NA      NA 31.81891 69.78365 NA   NA
## 6 1991-02-04        NA     NA      NA 154.23077 428.58974 NA   NA
## 7 1991-02-11        NA     NA      NA      NA 16.83814 14.63141 NA   NA
##   SALINITY SUSO      TEMP      XMS
## 1 33.59550   NA 19.37000 60.22500
## 2      NA   NA 14.45000 81.96154
## 3 33.50113   NA 14.07478 82.04876
## 4      NA   NA 14.54487 70.19231
## 5      NA   NA 14.28846 77.61167
## 6      NA   NA 14.23462 81.83333
## 7      NA   NA 14.46154 70.83333

scatr(df = owt_df02_gb04_wkly02,
      y = "ENTERO",
      x_lst = param_lst03)
```

Figure

Scatterplot of CHLOROPHYLL and ENTERO Series

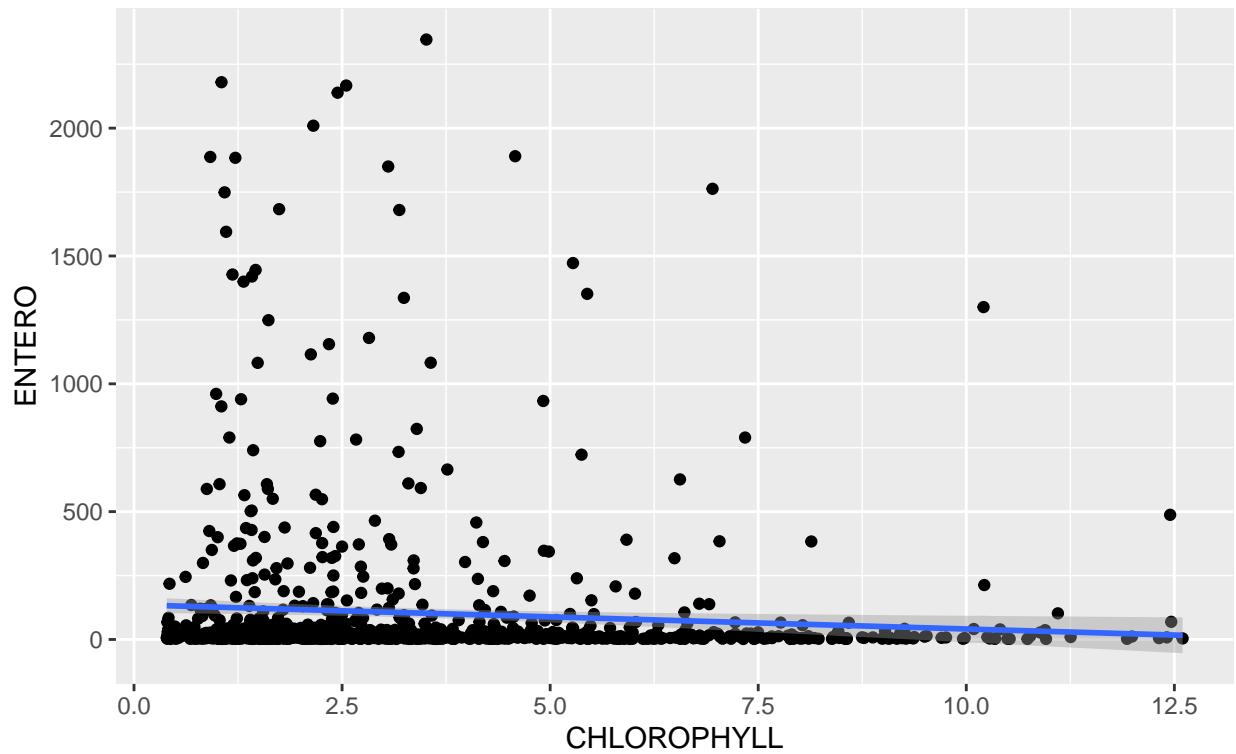
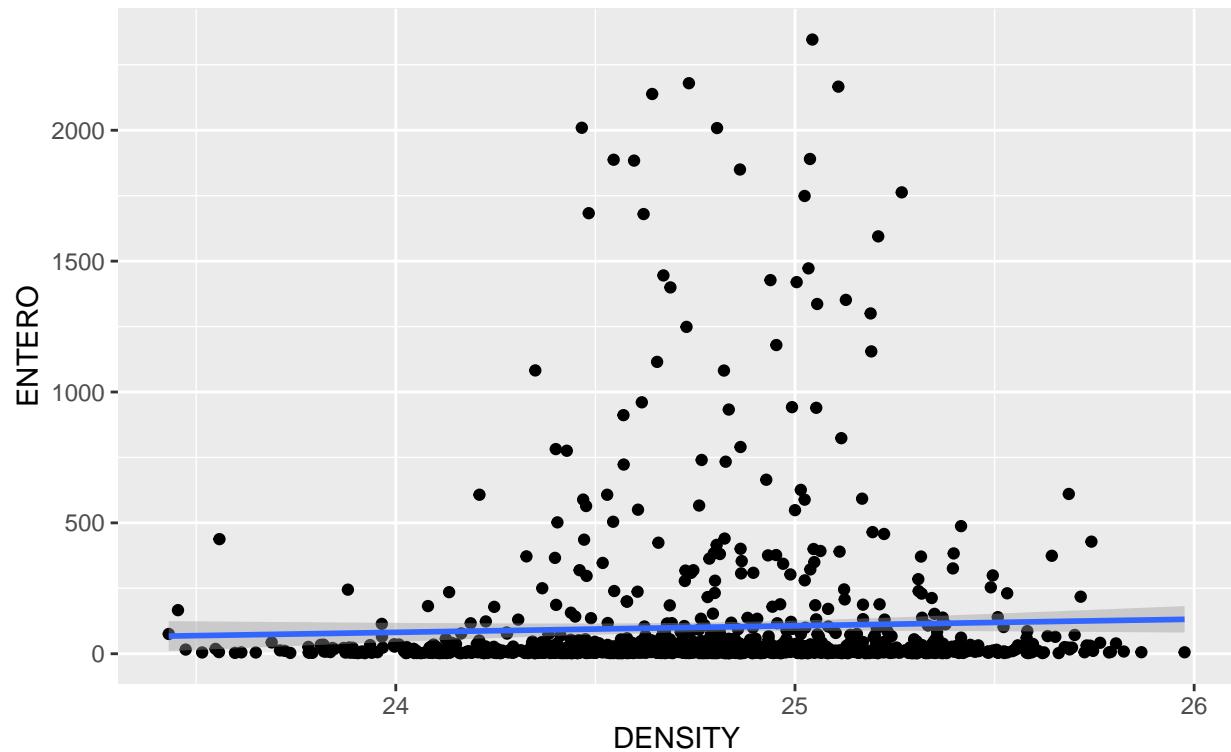


Figure
Scatterplot of DENSITY and ENTERO Series



Figure

Scatterplot of DO and ENTERO Series

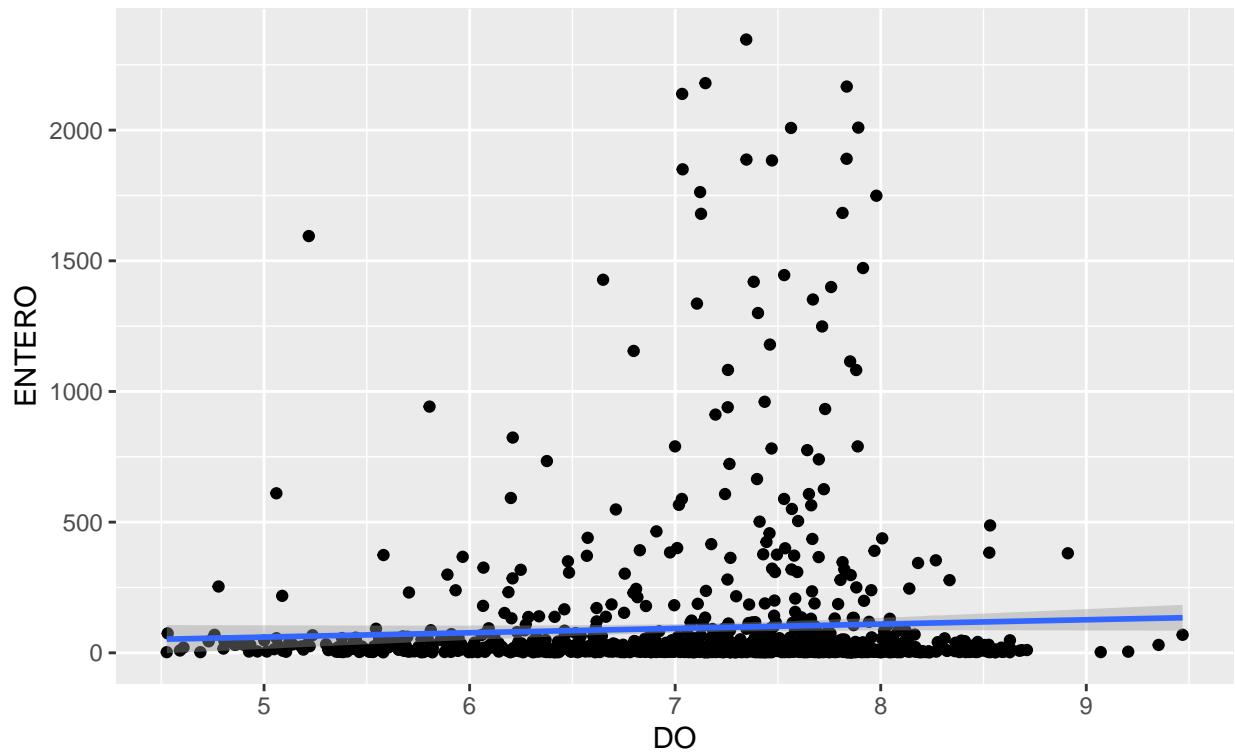


Figure
Scatterplot of FECAL and ENTERO Series

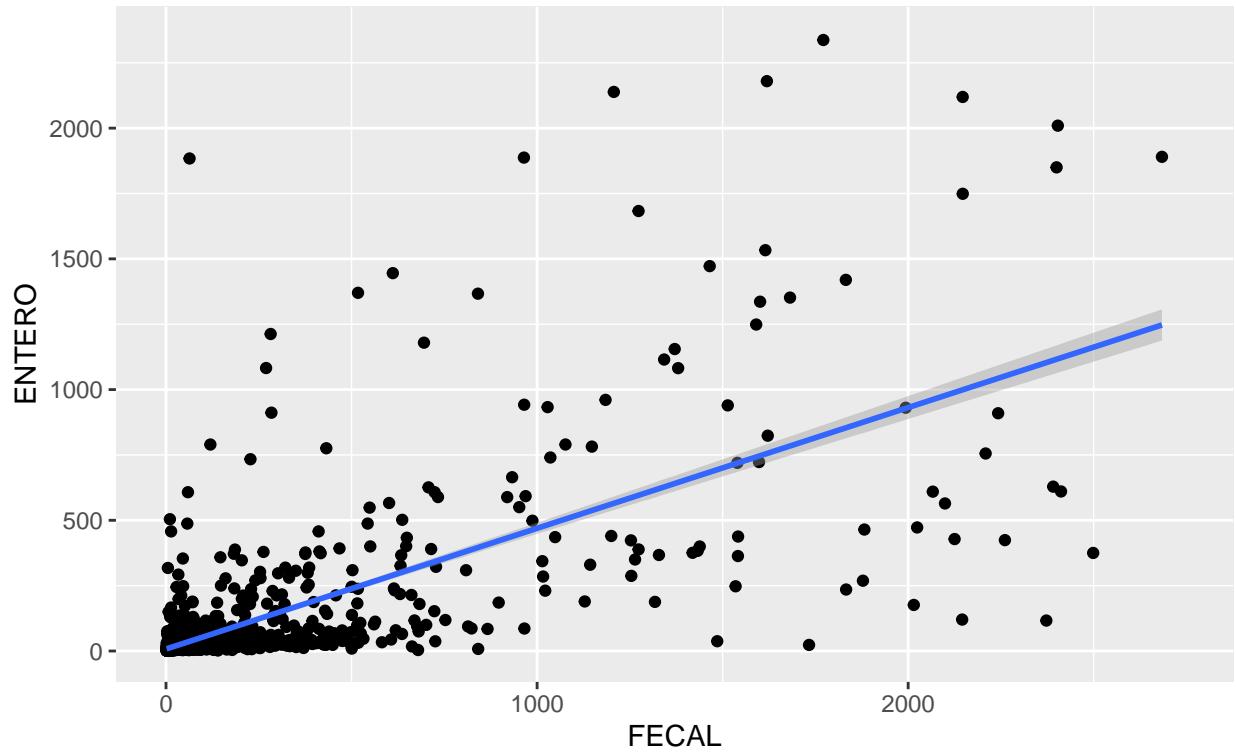
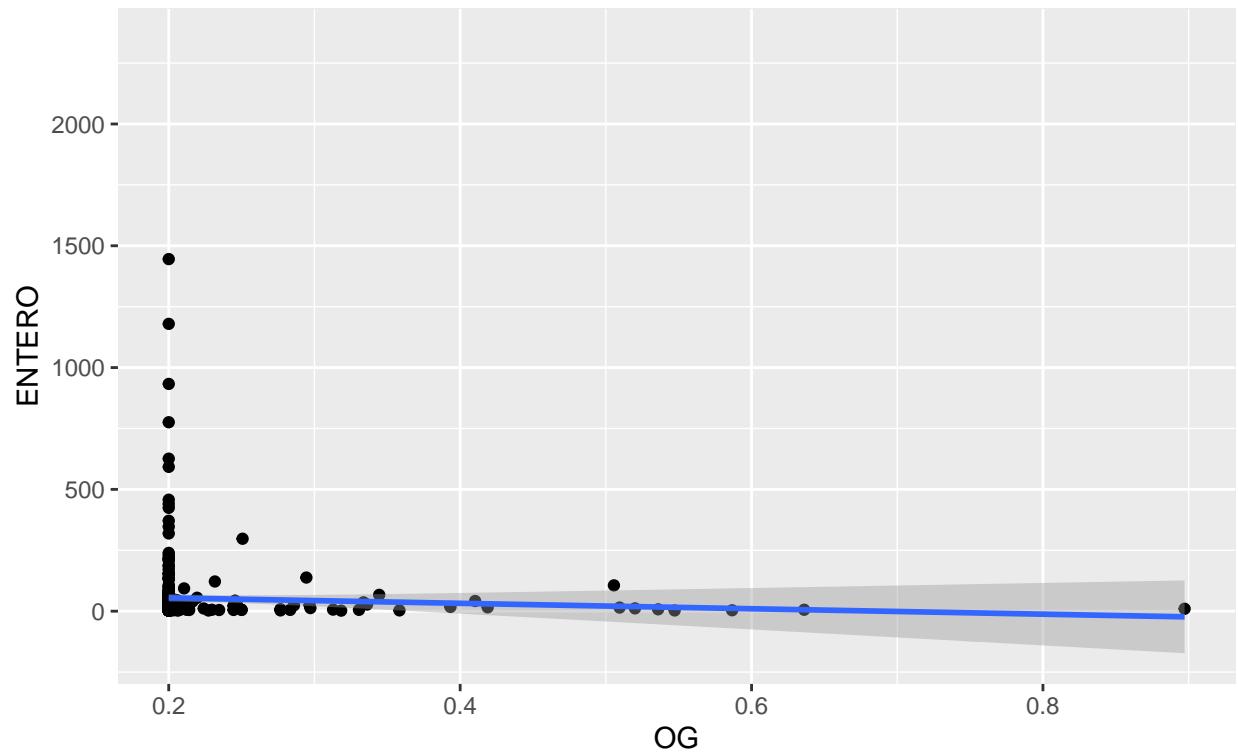


Figure
Scatterplot of OG and ENTERO Series



Figure

Scatterplot of PH and ENTERO Series

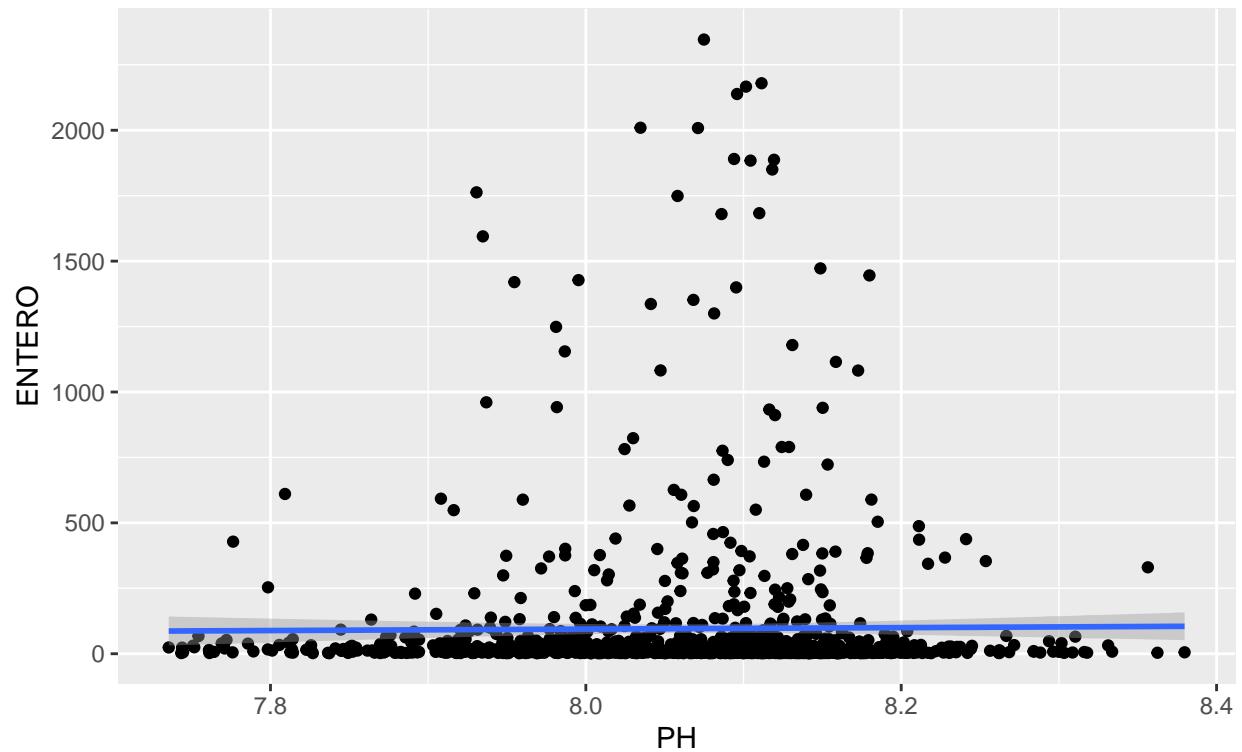


Figure
Scatterplot of SALINITY and ENTERO Series

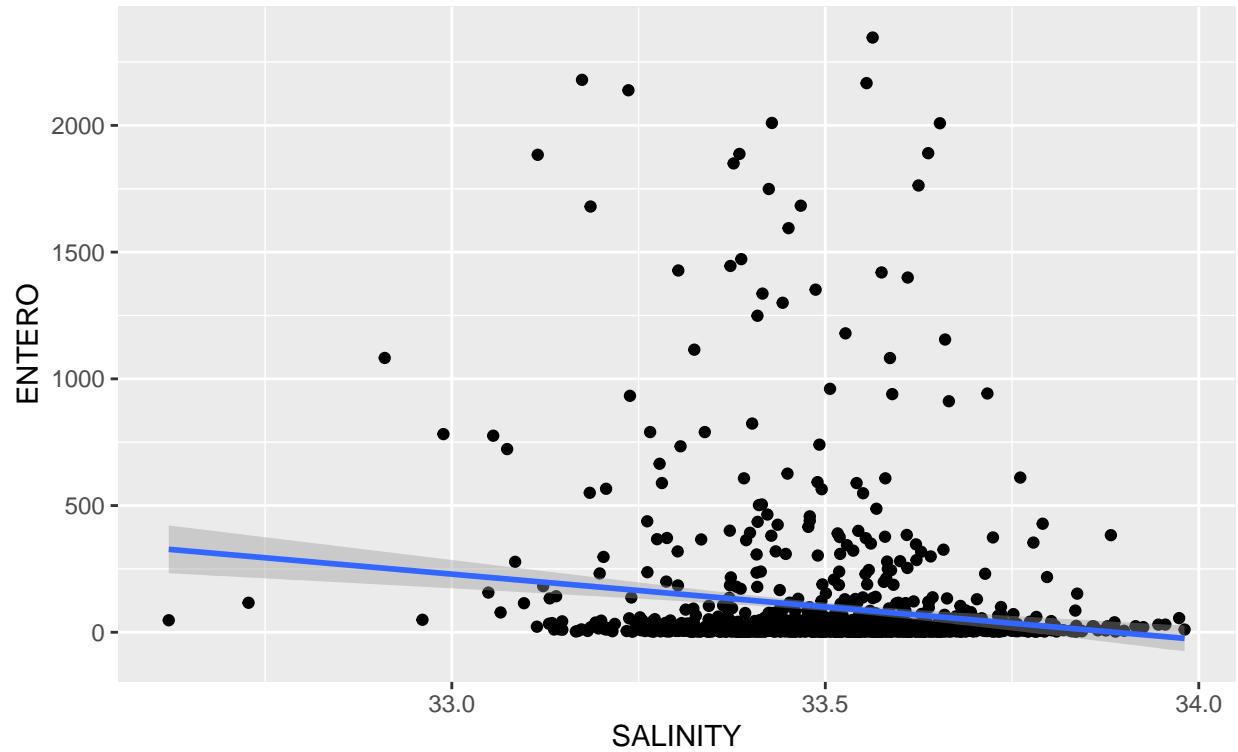


Figure
Scatterplot of SUSO and ENTERO Series

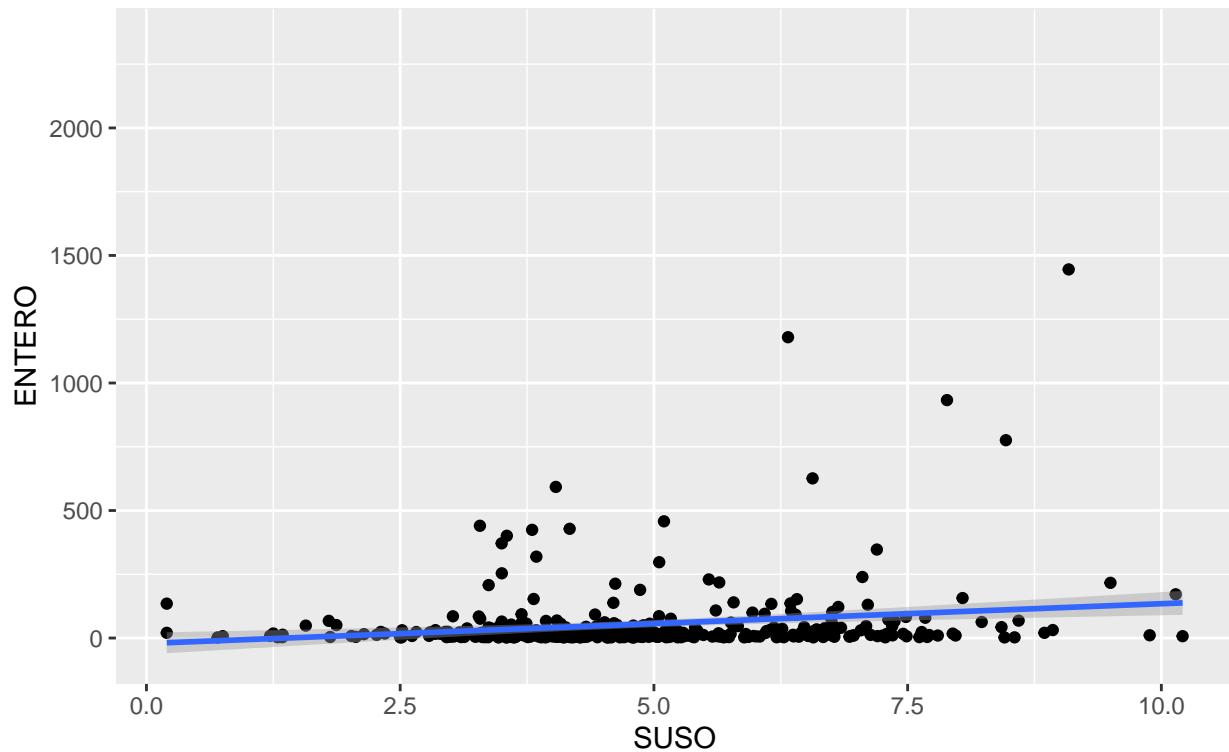
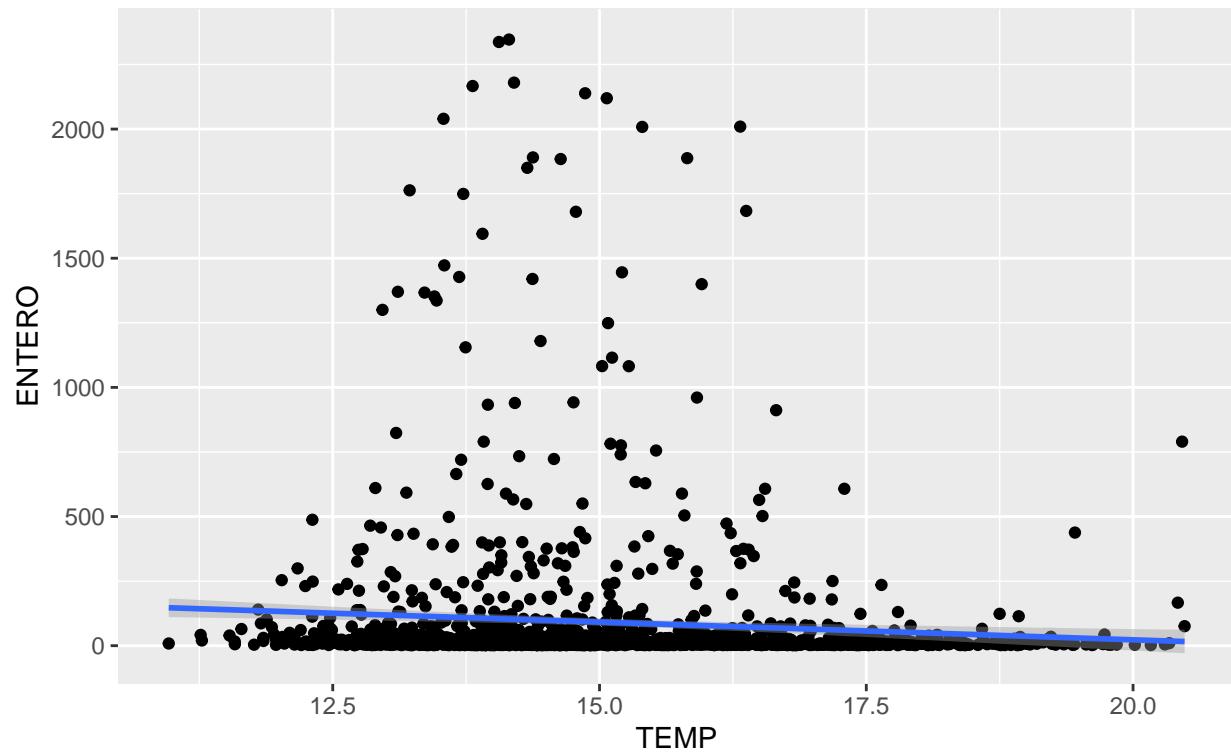
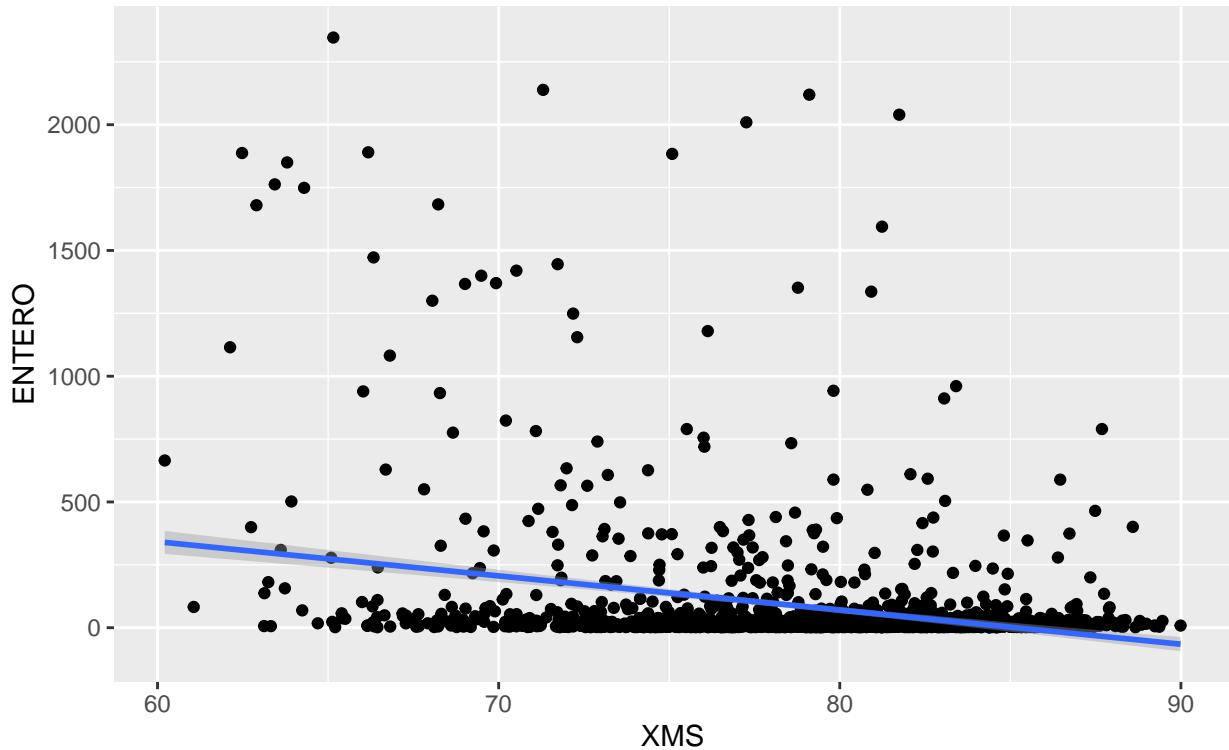


Figure
Scatterplot of TEMP and ENTERO Series



Figure

Scatterplot of XMS and ENTERO Series



Define custom function to display time series plots at multiple agg levels

```
# i = parameter
# j = project
# k = depth
time_plt <- function(df = NA,
                      date = NA,
                      l1 = c(),
                      l2_name = NA,
                      l2 = c(),
                      l3_name = NA,
                      l3 = c()) {
  if(length(l3) == 0 & length(l2) == 0) {
    for(i in l1) {
      df_01a <- df
      par(mar = c(5,5,5,2))
      try(print(acf(df_01a[, i],
                    pl = TRUE,
                    na.action = na.pass,
                    main = paste0("ACF Plot for ", i))))
      plot <- ggplot(df_01a, aes(x = df_01a[, date], y = df_01a[, i], group = 1)) +
        geom_line() +
        labs(x = "Time",
             y = i,
             title = "Figure",
             subtitle = "Time Series Plot") +
        theme_minimal()
      print(plot)
    }
  }
}
```

```

        subtitle = paste0("Time Series Plot for ", i))
    print(plot)
}
}
if(length(13) == 0) {
  for(i in 11) {
    for(j in 12) {
      df_01a <- df[df[, 12_name] == j, ]
      par(mar = c(5,5,5,2))
      try(print(acf(df_01a[, i],
                    pl = TRUE,
                    na.action = na.pass,
                    main = paste0("ACF Plot for ", i, " (", j, ")")))
      plot <- ggplot(df_01a, aes(x = df_01a[, date], y = df_01a[, i], group = 1)) +
        geom_line() +
        labs(x = "Time",
             y = i,
             title = "Figure",
             subtitle = paste0("Time Series Plot for ", i, " (", j, ")"))
      print(plot)
    }
  }
} else {
  for(i in 11) {
    for(j in 12) {
      for(k in 13) {
        df_01a <- df[df[, 12_name] == j & df[, 13_name] == k, ]
        par(mar = c(5,5,5,2))
        try(print(acf(df_01a[, i],
                      pl = TRUE,
                      na.action = na.pass,
                      main = paste0("ACF Plot for ", i, " (", j, ":", k, ")")))
        plot <- ggplot(df_01a, aes(x = df_01a[, date], y = df_01a[, i], group = 1)) +
          geom_line() +
          labs(x = "Time",
               y = i,
               title = "Figure",
               subtitle = paste0("Time Series Plot for ", i, " (", j, ":", k, ")"))
        print(plot)
      }
    }
  }
}
}

```

Setup reference value lists for time series plotting

```

param_lst04 <- c("CHLOROPHYLL",
                 "DENSITY",
                 "DO",
                 "ENTERO",

```

```

    "FECAL",
    "PH",
    "SALINITY",
    "TEMP",
    "XMS")

param_lst05 <- c("ENTERO")

uniq_proj <- unique(owt_df02_gb07_mrgd$project)
uniq_depth <- unique(owt_df02_gb07_mrgd$depth_m_bin)
uniq_proj

## [1] PL00 SB00
## Levels: PL00 SB00

uniq_depth

## [1] "[8,33)"    "[0,8)"      "[33,47)"    "[47,70)"    "[70,90)"    "Unknown"
## [7] "[112,120]" "[90,112]"

```

Preprocessing, phase 2

Convert df's to timetk tibbles for data aggregated by date_sample, project, and parameter

```

# Citation: https://business-science.github.io/timetk/index.html
train_start <- "1999-01-18"
train_end <- "2021-01-04"
test_start <- "2021-01-11"
test_end <- ""

#####
# Convert PL00 data for 01/18/1999 and 01/03/2022 to `timetk` TS tibble
owt_df02_gb09_ploo_wkly03_df01 <- owt_df02_gb09_wkly03[owt_df02_gb09_wkly03$project ==
  & "PL00" & owt_df02_gb09_wkly03$date_sample >= train_start, ]
owt_df02_gb09_ploo_wkly03_df02 <- owt_df02_gb09_ploo_wkly03_df01[, -c(2, 8, 11)]

owt_df02_gb09_ploo_wkly03_tb01 <- tibble(owt_df02_gb09_ploo_wkly03_df02)
owt_df02_gb09_ploo_wkly03_tb02 <- pad_by_time(.data = owt_df02_gb09_ploo_wkly03_tb01,
  .date_var = date_sample,
  .by = "week")

#-----
# Convert PL00 data (excluding outliers) for 01/18/1999 and 01/03/2022 to `timetk` TS
# tibble
owt_df02_gb09_ploo_wkly02_df01 <- owt_df02_gb09_wkly02[owt_df02_gb09_wkly02$project ==
  & "PL00" & owt_df02_gb09_wkly02$date_sample >= train_start, ]
owt_df02_gb09_ploo_wkly02_df02 <- owt_df02_gb09_ploo_wkly02_df01[, -c(2, 8, 11)]

owt_df02_gb09_ploo_wkly02_tb01 <- tibble(owt_df02_gb09_ploo_wkly02_df02)
owt_df02_gb09_ploo_wkly02_tb02 <- pad_by_time(.data = owt_df02_gb09_ploo_wkly02_tb01,
  .date_var = date_sample,
  .by = "week")

```

```

#####
# Convert SBOO data for 01/18/1999 and 01/03/2022 to `timetk` TS tibble
owt_df02_gb09_sboo_wkly03_df01 <- owt_df02_gb09_wkly03[owt_df02_gb09_wkly03$project ==
  "SBOO" & owt_df02_gb09_wkly03$date_sample >= train_start, ]
owt_df02_gb09_sboo_wkly03_df02 <- owt_df02_gb09_sboo_wkly03_df01[, -c(2, 8, 11)]

owt_df02_gb09_sboo_wkly03_tb01 <- tibble(owt_df02_gb09_sboo_wkly03_df02)
owt_df02_gb09_sboo_wkly03_tb02 <- pad_by_time(.data = owt_df02_gb09_sboo_wkly03_tb01,
  .date_var = date_sample,
  .by = "week")

#-----
# Convert SBOO data (excluding outliers) for 01/18/1999 and 01/03/2022 to `timetk` TS
# tibble
owt_df02_gb09_sboo_wkly02_df01 <- owt_df02_gb09_wkly02[owt_df02_gb09_wkly02$project ==
  "SBOO" & owt_df02_gb09_wkly02$date_sample >= train_start, ]
owt_df02_gb09_sboo_wkly02_df02 <- owt_df02_gb09_sboo_wkly02_df01[, -c(2, 8, 11)]

owt_df02_gb09_sboo_wkly02_tb01 <- tibble(owt_df02_gb09_sboo_wkly02_df02)
owt_df02_gb09_sboo_wkly02_tb02 <- pad_by_time(.data = owt_df02_gb09_sboo_wkly02_tb01,
  .date_var = date_sample,
  .by = "week")

```

Partition tibbles for data aggregated by date_sample, project, and parameter

```

#####
# PLOO All - Training partition (01/18/1999-01/04/2021)
owt_df02_gb09_ploo_wkly03_train_tb02 <- owt_df02_gb09_ploo_wkly03_tb02 %>%
  filter_by_time(date_sample,
    .end_date = train_end)

# PLOO All - Test partition (01/11/2021-01/03/2022)
owt_df02_gb09_ploo_wkly03_test_tb02 <- owt_df02_gb09_ploo_wkly03_tb02 %>%
  filter_by_time(date_sample,
    .start_date = test_start)

#-----
# PLOO Excluding Outliers - Training partition (01/18/1999-01/04/2021)
owt_df02_gb09_ploo_wkly02_train_tb02 <- owt_df02_gb09_ploo_wkly02_tb02 %>%
  filter_by_time(date_sample,
    .end_date = train_end)

# PLOO Excluding Outliers - Test partition (01/11/2021-01/03/2022)
owt_df02_gb09_ploo_wkly02_test_tb02 <- owt_df02_gb09_ploo_wkly02_tb02 %>%
  filter_by_time(date_sample,
    .start_date = test_start)

#####
# SBOO All - Training partition (01/18/1999-01/04/2021)
owt_df02_gb09_sboo_wkly03_train_tb02 <- owt_df02_gb09_sboo_wkly03_tb02 %>%
  filter_by_time(date_sample,
    .end_date = train_end)

# SBOO All - Test partition (01/11/2021-01/03/2022)
owt_df02_gb09_sboo_wkly03_test_tb02 <- owt_df02_gb09_sboo_wkly03_tb02 %>%
  filter_by_time(date_sample,
    .start_date = test_start)

```

```

    .end_date = train_end)

# SBOO All - Test partition (01/11/2021-01/03/2022)
owt_df02_gb09_sboo_wkly03_test_tb02 <- owt_df02_gb09_sboo_wkly03_tb02 %>%
  filter_by_time(date_sample,
                 .start_date = test_start)

#-----
# SBOO Excluding Outliers - Training partition (01/18/1999-01/04/2021)
owt_df02_gb09_sboo_wkly02_train_tb02 <- owt_df02_gb09_sboo_wkly02_tb02 %>%
  filter_by_time(date_sample,
                 .end_date = train_end)

# SBOO Excluding Outliers - Test partition (01/11/2021-01/03/2022)
owt_df02_gb09_sboo_wkly02_test_tb02 <- owt_df02_gb09_sboo_wkly02_tb02 %>%
  filter_by_time(date_sample,
                 .start_date = test_start)

```

Display tibbles available for modeling

```

# Data sets for modeling
#####
# PLOO All - Training partition (01/18/1999-01/04/2021)
#print(head(owt_df02_gb09_ploo_wkly03_train_tb02, 10))
#print(dim(owt_df02_gb09_ploo_wkly03_train_tb02))

# PLOO All - Test partition (01/11/2021-01/03/2022)
#print(head(owt_df02_gb09_ploo_wkly03_test_tb02, 10))
#print(dim(owt_df02_gb09_ploo_wkly03_test_tb02))

#-----
# PLOO Excluding Outliers - Training partition (01/18/1999-01/04/2021)
#print(head(owt_df02_gb09_ploo_wkly02_train_tb02, 10))
#print(dim(owt_df02_gb09_ploo_wkly02_train_tb02))

# PLOO Excluding Outliers - Test partition (01/11/2021-01/03/2022)
#print(head(owt_df02_gb09_ploo_wkly02_test_tb02, 10))
#print(dim(owt_df02_gb09_ploo_wkly02_test_tb02))

#####
# SBOO All - Training partition (01/18/1999-01/04/2021)
#print(head(owt_df02_gb09_sboo_wkly03_train_tb02, 10))
#print(dim(owt_df02_gb09_sboo_wkly03_train_tb02))

# SBOO All - Test partition (01/11/2021-01/03/2022)
#print(head(owt_df02_gb09_sboo_wkly03_test_tb02, 10))
#print(dim(owt_df02_gb09_sboo_wkly03_test_tb02))

#-----
# SBOO Excluding Outliers - Training partition (01/18/1999-01/04/2021)
#print(head(owt_df02_gb09_sboo_wkly02_train_tb02, 10))
#print(dim(owt_df02_gb09_sboo_wkly02_train_tb02))

```

```
# SBOO Excluding Outliers - Test partition (01/11/2021-01/03/2022)
#print(head(owt_df02_gb09_sboo_wkly02_test_tb02, 10))
#print(dim(owt_df02_gb09_sboo_wkly02_test_tb02))
```

Imputation using timetk ts_impute_vec() function

```
# Impute missing values using TS imputation w/ seasonality
qk_imp <- function(df = NA) {
  df$CHLOROPHYLL <- ts_impute_vec(df$CHLOROPHYLL,
                                      period = 3,
                                      lambda = NULL)

  df$DENSITY <- ts_impute_vec(df$DENSITY,
                                period = 3,
                                lambda = NULL)

  df$DO <- ts_impute_vec(df$DO,
                         period = 3,
                         lambda = NULL)

  df$ENTERO <- ts_impute_vec(df$ENTERO,
                               period = 3,
                               lambda = NULL)

  df$FECAL <- ts_impute_vec(df$FECAL,
                            period = 3,
                            lambda = NULL)

  df$PH <- ts_impute_vec(df$PH,
                         period = 3,
                         lambda = NULL)

  df$SALINITY <- ts_impute_vec(df$SALINITY,
                                period = 3,
                                lambda = NULL)

  df$TEMP <- ts_impute_vec(df$TEMP,
                           period = 3,
                           lambda = NULL)

  df$XMS <- ts_impute_vec(df$XMS,
                           period = 3,
                           lambda = NULL)
}

return(df)
}

# PLOO All - Training partition (01/18/1999-01/04/2021)
owt_df02_gb09_ploo_wkly03_train_tb03 <- qk_imp(df = owt_df02_gb09_ploo_wkly03_train_tb02)
#print(head(owt_df02_gb09_ploo_wkly03_train_tb03, 10))

# PLOO All - Test partition (01/11/2021-01/03/2022)
```

```

owt_df02_gb09_ploo_wkly03_test_tb03 <- qk_imp(df = owt_df02_gb09_ploo_wkly03_test_tb02)
#print(head(owt_df02_gb09_ploo_wkly03_test_tb03, 10))

#-----
# PL00 Excluding Outliers - Training partition (01/18/1999-01/04/2021)
owt_df02_gb09_ploo_wkly02_train_tb03 <- qk_imp(df = owt_df02_gb09_ploo_wkly02_train_tb02)
#print(head(owt_df02_gb09_ploo_wkly02_train_tb03, 10))

# PL00 Excluding Outliers - Test partition (01/11/2021-01/03/2022)
owt_df02_gb09_ploo_wkly02_test_tb03 <- qk_imp(df = owt_df02_gb09_ploo_wkly02_test_tb02)
#print(head(owt_df02_gb09_ploo_wkly02_test_tb03, 10))

#####
# SBOO All - Training partition (01/18/1999-01/04/2021)
owt_df02_gb09_sboo_wkly03_train_tb03 <- qk_imp(df = owt_df02_gb09_sboo_wkly03_train_tb02)
#print(head(owt_df02_gb09_sboo_wkly03_train_tb03, 10))

# SBOO All - Test partition (01/11/2021-01/03/2022)
owt_df02_gb09_sboo_wkly03_test_tb03 <- qk_imp(df = owt_df02_gb09_sboo_wkly03_test_tb02)
#print(head(owt_df02_gb09_sboo_wkly03_test_tb03, 10))

#-----
# SBOO Excluding Outliers - Training partition (01/18/1999-01/04/2021)
owt_df02_gb09_sboo_wkly02_train_tb03 <- qk_imp(df = owt_df02_gb09_sboo_wkly02_train_tb02)
#print(head(owt_df02_gb09_sboo_wkly02_train_tb03, 10))

# SBOO Excluding Outliers - Test partition (01/11/2021-01/03/2022)
owt_df02_gb09_sboo_wkly02_test_tb03 <- qk_imp(df = owt_df02_gb09_sboo_wkly02_test_tb02)
#print(head(owt_df02_gb09_sboo_wkly02_test_tb03, 10))

```

Modeling and Evaluation

Holt-Winters Modeling for all Parameters and all tables

```

# convert to time series object ts
params <- c("CHLOROPHYLL", "DENSITY", "DO", "ENTERO", "FECAL", "PH", "SALINITY", "TEMP",
           "XMS")

for (param in params) {
  z <- get(param, owt_df02_gb09_ploo_wkly03_train_tb03)
  ts_owt_df02_gb09_ploo_wkly03_train_tb03 <- ts(z, start = c(1999, 1), freq = 52)
  ts_owt_df02_gb09_ploo_wkly03_test_tb03 <- ts(z, start = c(2021, 1), freq = 52)

  HW1 <- HoltWinters(ts_owt_df02_gb09_ploo_wkly03_train_tb03)

  # forecast library more robust features vs. predict()
  HW1_for <- forecast(HW1, h=52, level=c(80,95))

  #visualize our predictions:
  plot(HW1_for, xlim=c(2017, 2023))
  lines(HW1_for$fitted, lty=2, col="purple")
  mtext(param, side=3, line = -2)
}

```

```

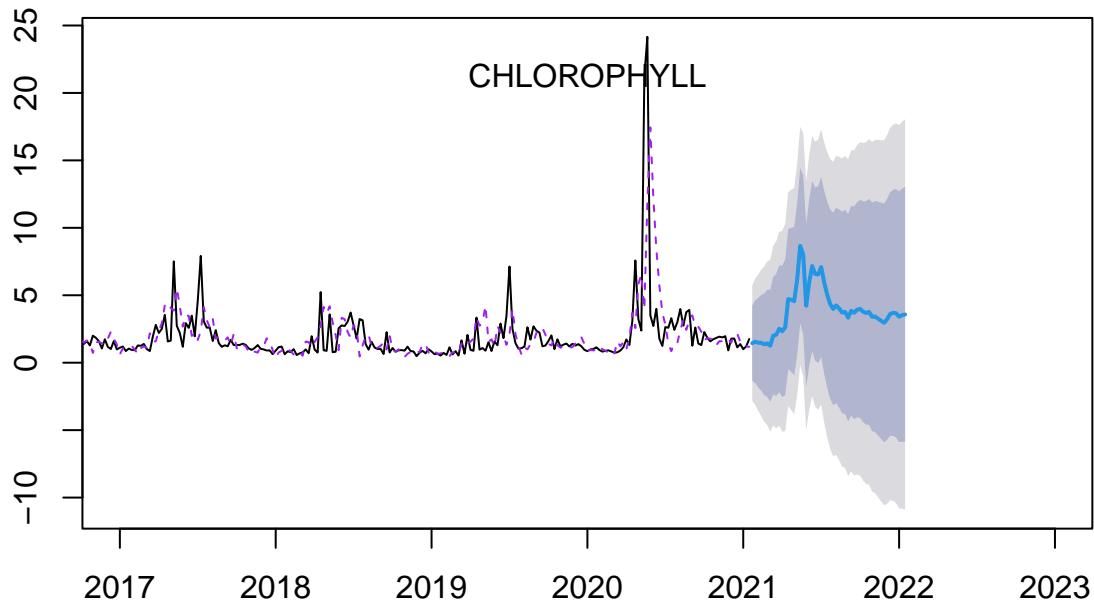
print("Training Set Accuracy: ")
print(param)
#print(accuracy(HW1_for))
print(accuracy(HW1_for, ts_owt_df02_gb09_ploo_wkly03_test_tb03))

#z2 <- get(param, owt_df02_gb09_ploo_wkly03_test_tb03)
#ts_owt_df02_gb09_ploo_wkly03_test_tb03 <- ts(z2, start = c(2021, 1), freq = 52)

#print("MAPE")
#print(MAPE(y_pred = HW1_for$fitted, y_true = ts_owt_df02_gb09_ploo_wkly03_test_tb03))
#print("MAE")
#print(MAE(y_pred = HW1_for$fitted, y_true = ts_owt_df02_gb09_ploo_wkly03_test_tb03))
}

```

Forecasts from HoltWinters

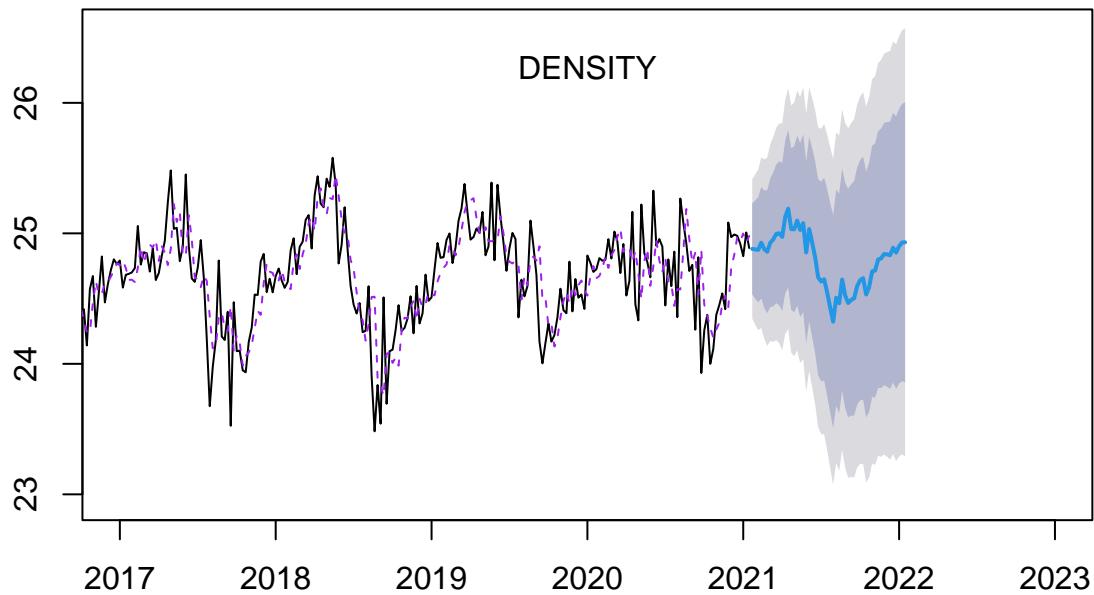


```

## [1] "Training Set Accuracy: "
## [1] "CHLOROPHYLL"
##          ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -0.09843704 2.180400 1.308285 -21.294434 51.71619 0.6397732
## Test set      0.16258431 2.054043 1.653307 -6.472758 47.75021 0.8084948
##          ACF1 Theil's U
## Training set 0.2038780      NA
## Test set     0.8406931  2.145953

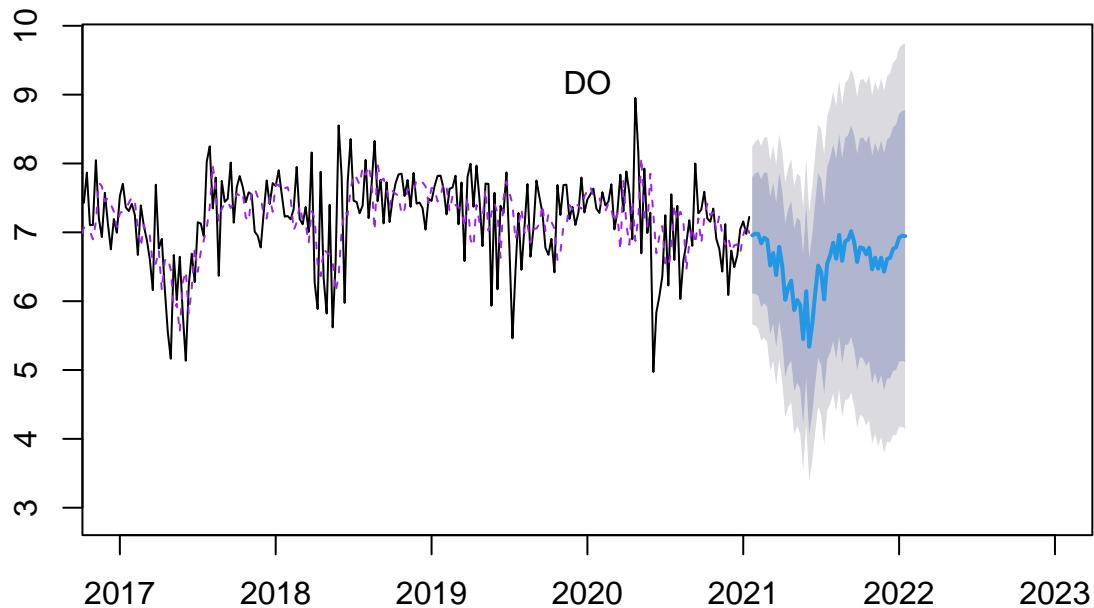
```

Forecasts from HoltWinters



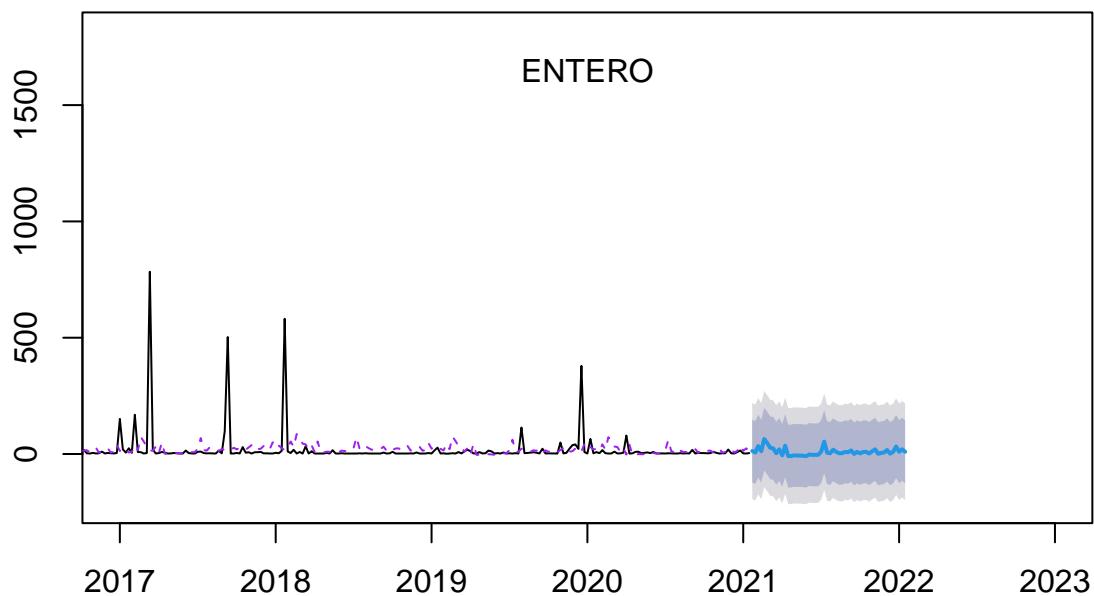
```
## [1] "Training Set Accuracy: "
## [1] "DENSITY"
##               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -0.00112412 0.2720603 0.2073202 -0.01345529 0.8388695 0.6194478
## Test set      0.47983679 0.5891640 0.5518780  1.88624392 2.1840248 1.6489449
##               ACF1 Theil's U
## Training set 0.1771812       NA
## Test set     0.7860227  2.680946
```

Forecasts from HoltWinters



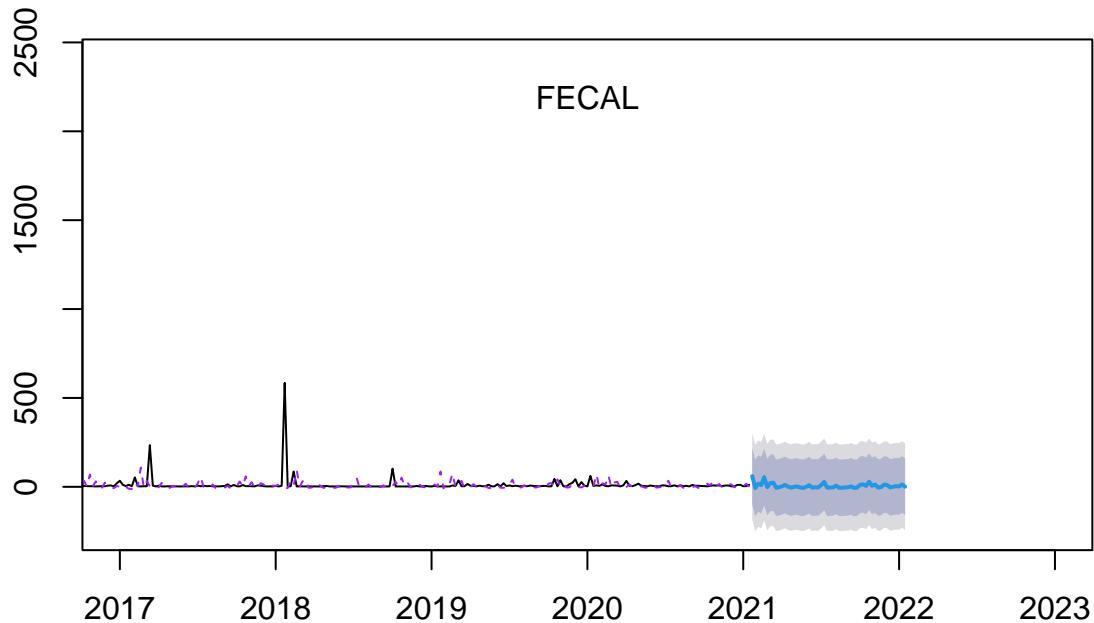
```
## [1] "Training Set Accuracy: "
## [1] "DO"
##          ME      RMSE      MAE      MPE      MAPE      MASE
## Training set 0.0258745 0.6604220 0.4966317 -0.3115491 7.357875 0.6600327
## Test set     0.4134250 0.8891192 0.6746798  5.2426163 9.421445 0.8966620
##          ACF1 Theil's U
## Training set 0.1605063       NA
## Test set     0.8829317  3.938266
```

Forecasts from HoltWinters



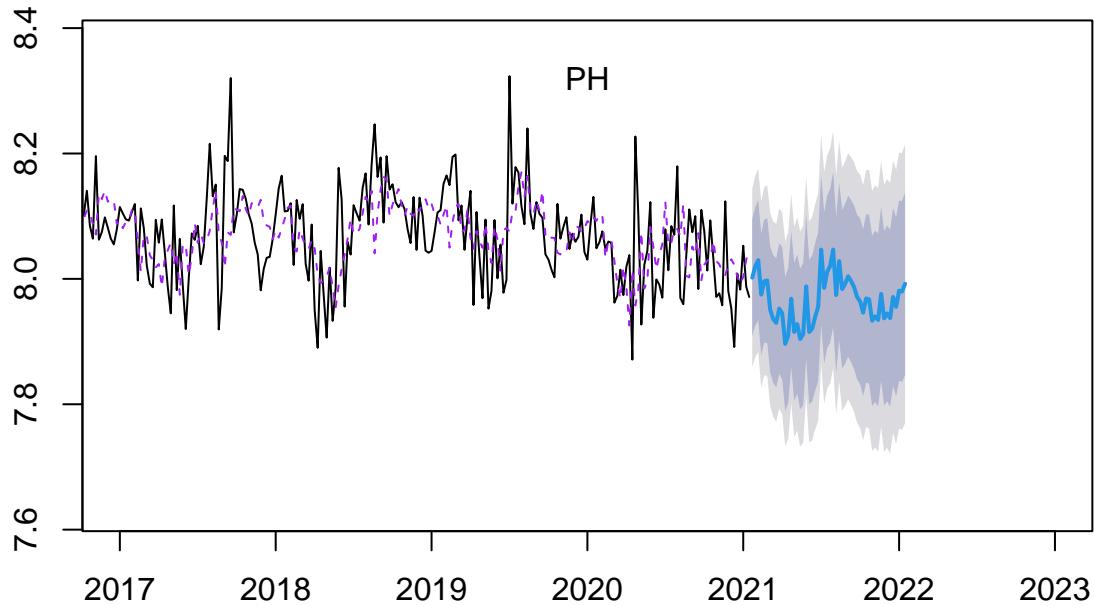
```
## [1] "Training Set Accuracy: "
## [1] "ENTERO"
##               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -3.2046505 105.31660 32.34395 -474.3885 522.7562 0.9910613
## Test set      0.7104022 16.36471 11.20878 -53.5389 211.5065 0.3434518
##               ACF1 Theil's U
## Training set  0.02565651       NA
## Test set      -0.07099528 1.667067
```

Forecasts from HoltWinters



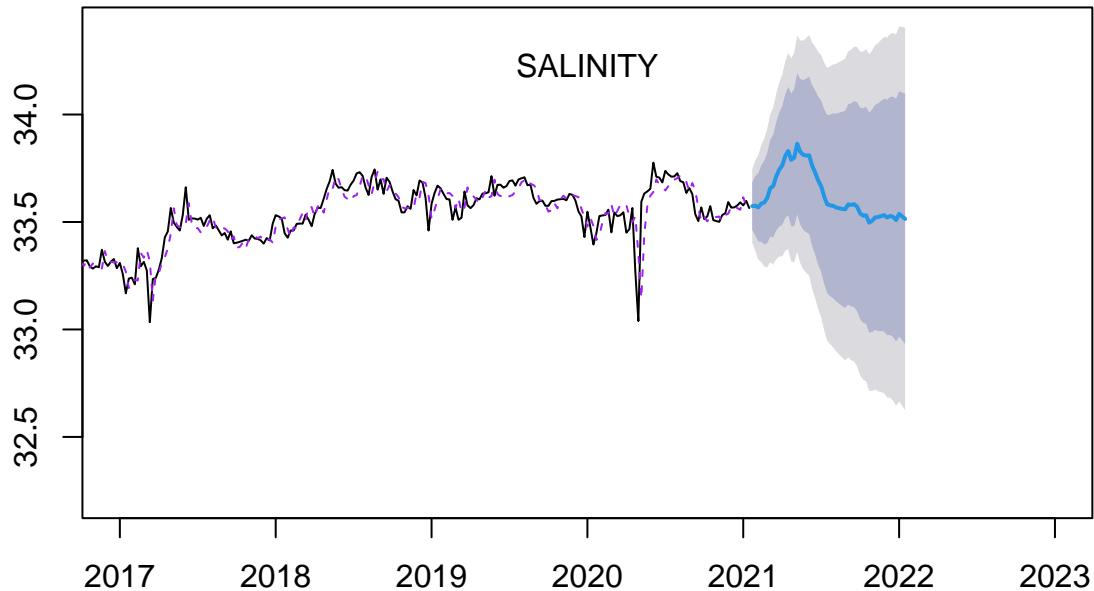
```
## [1] "Training Set Accuracy: "
## [1] "FECAL"
##          ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -3.474252 124.1303 44.49475 -376.5720525 568.9474 0.9692438
## Test set     65.223857 121.6356 71.26946    0.8179895 207.6003 1.5524860
##          ACF1 Theil's U
## Training set  0.0009713377      NA
## Test set     -0.2023185473  0.975268
```

Forecasts from HoltWinters



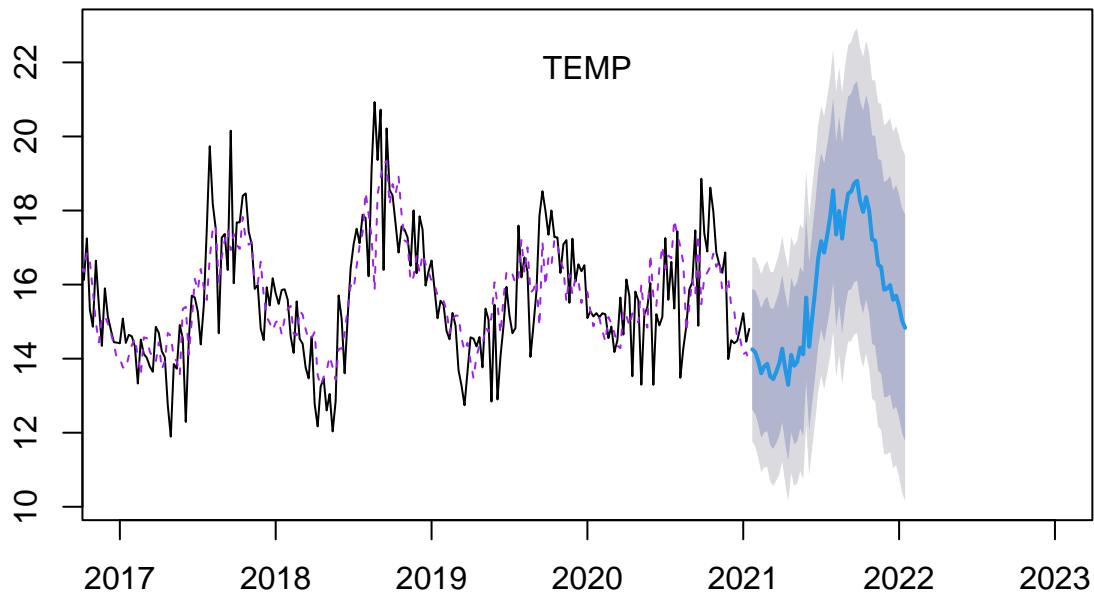
```
## [1] "Training Set Accuracy: "
## [1] "PH"
##               ME        RMSE       MAE       MPE       MAPE       MASE
## Training set 0.00285767 0.07267275 0.05446864 0.02974103 0.6761781 0.684549
## Test set     -0.05082892 0.09895746 0.08224903 -0.64988867 1.0407910 1.033686
##               ACF1 Theil's U
## Training set 0.2544343      NA
## Test set     0.8222910 2.455689
```

Forecasts from HoltWinters



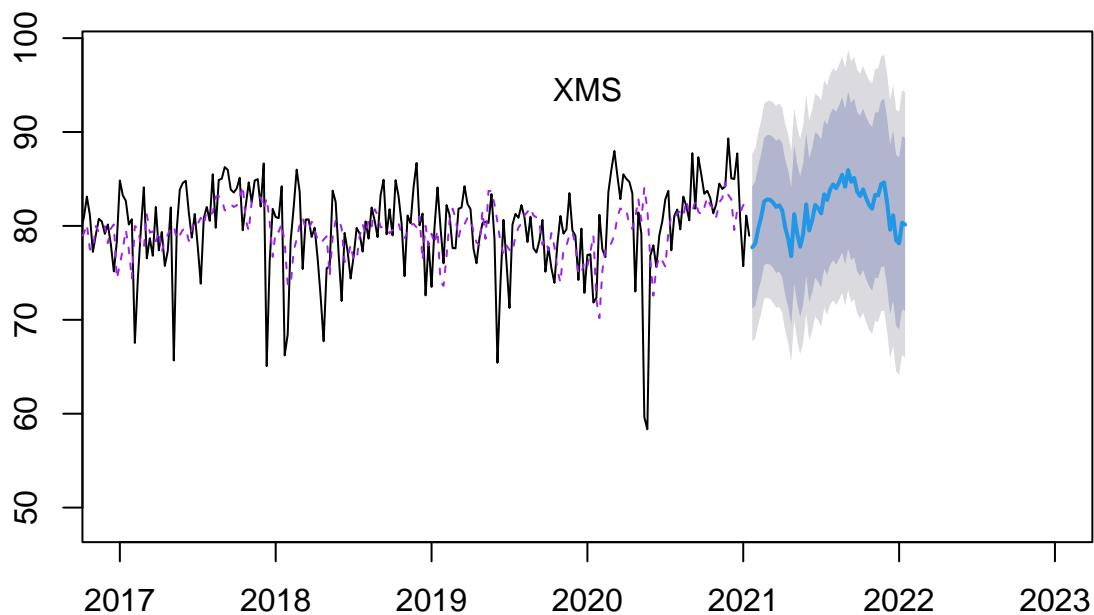
```
## [1] "Training Set Accuracy: "
## [1] "SALINITY"
##               ME        RMSE       MAE       MPE       MAPE       MASE
## Training set 0.001385062 0.08670362 0.05302033 0.003693586 0.1584602 0.4309180
## Test set      0.050533571 0.11827019 0.10982092 0.149994522 0.3260853 0.8925598
##               ACF1 Theil's U
## Training set 0.0629771      NA
## Test set     0.9261584  3.553131
```

Forecasts from HoltWinters



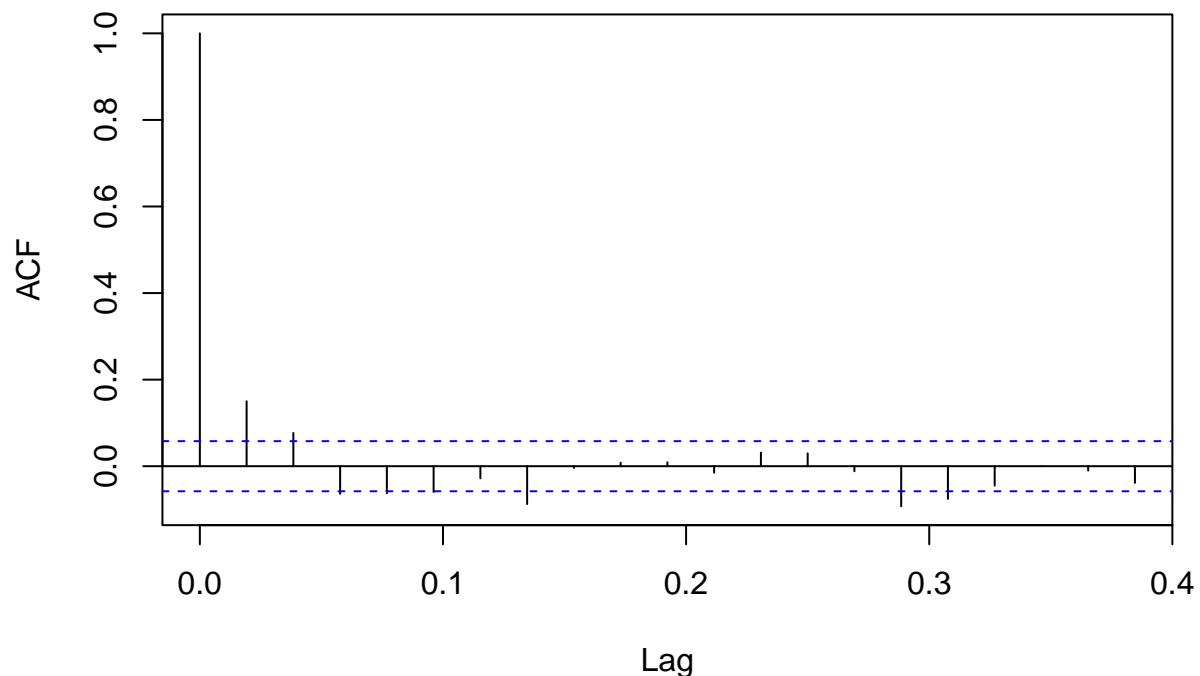
```
## [1] "Training Set Accuracy: "
## [1] "TEMP"
##          ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -0.03670735 1.269100 0.9629089 -0.8010885 6.336469 0.7015829
## Test set     -1.89679587 2.539536 2.1773233 -14.0113180 15.796642 1.5864146
##          ACF1 Theil's U
## Training set 0.1725499       NA
## Test set     0.7074838 1.964391
```

Forecasts from HoltWinters



```
## [1] "Training Set Accuracy: "
## [1] "XMS"
##          ME      RMSE      MAE      MPE      MAPE      MASE
## Training set 0.1441697 5.093418 3.646157 -0.1597366 4.690327 0.7195253
## Test set     0.2534947 4.553785 3.430603  0.0610891 4.282363 0.6769882
##          ACF1 Theil's U
## Training set 0.1500687       NA
## Test set     0.3510161 0.9311002
acf(HW1_for$residuals, lag.max=20, na.action=na.pass)
```

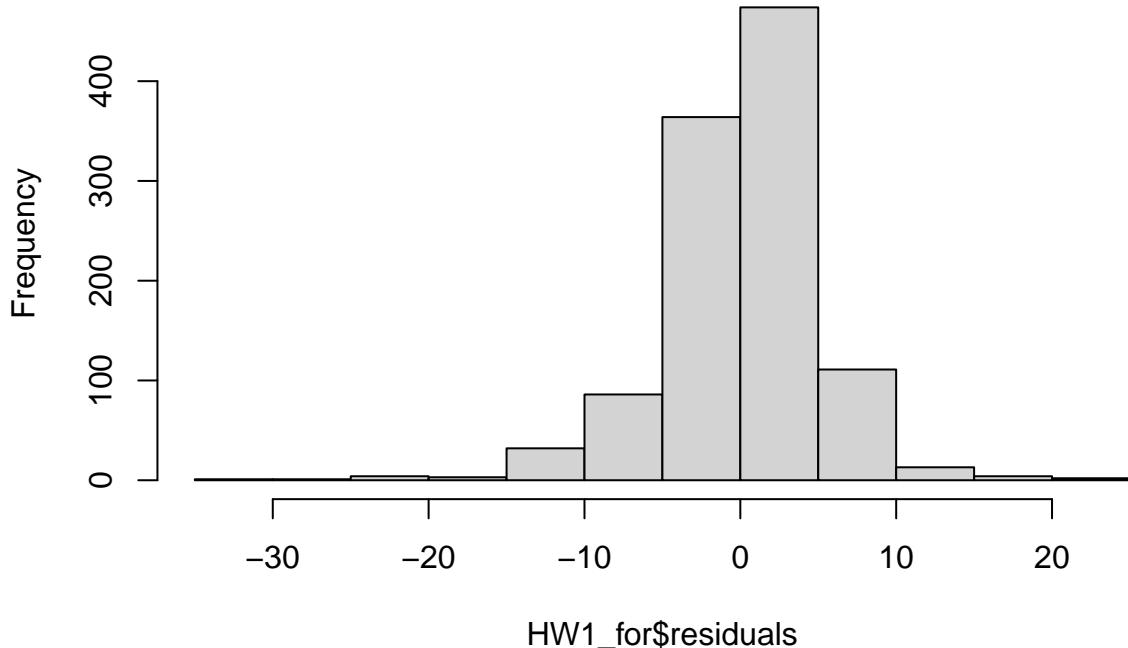
Series HW1_for\$residuals



```
Box.test(HW1_for$residuals, lag=20, type="Ljung-Box")
```

```
##  
## Box-Ljung test  
##  
## data: HW1_for$residuals  
## X-squared = 75.799, df = 20, p-value = 2.004e-08  
hist(HW1_for$residuals)
```

Histogram of HW1_for\$residuals



```
# convert to time series object ts
params <- c("CHLOROPHYLL", "DENSITY", "DO", "ENTERO", "FECAL", "PH", "SALINITY", "TEMP",
          "XMS")

for (param in params) {
  #print(param)

  z11 <- get(param, owt_df02_gb09_ploo_wkly02_train_tb03)
  ts_owt_df02_gb09_ploo_wkly02_train_tb03 <- ts(z11, start = c(1999, 1), freq = 52)
  ts_owt_df02_gb09_ploo_wkly02_test_tb03 <- ts(z11, start = c(2021, 1), freq = 52)

  HW1 <- HoltWinters(ts_owt_df02_gb09_ploo_wkly02_train_tb03)

  # forecast library more robust features vs. predict()

  HW1_for11 <- forecast(HW1, h=104, level=c(80,95))
  #visualize our predictions:
  plot(HW1_for11, xlim=c(2017, 2023))
  lines(HW1_for11$fitted, lty=2, col="purple")
  mtext(param, side=3, line = -2)

  print("Training Set Accuracy: ")
  print(param)
  #print(accuracy(HW1_for11))
  print(accuracy(HW1_for11, ts_owt_df02_gb09_ploo_wkly02_test_tb03))
```

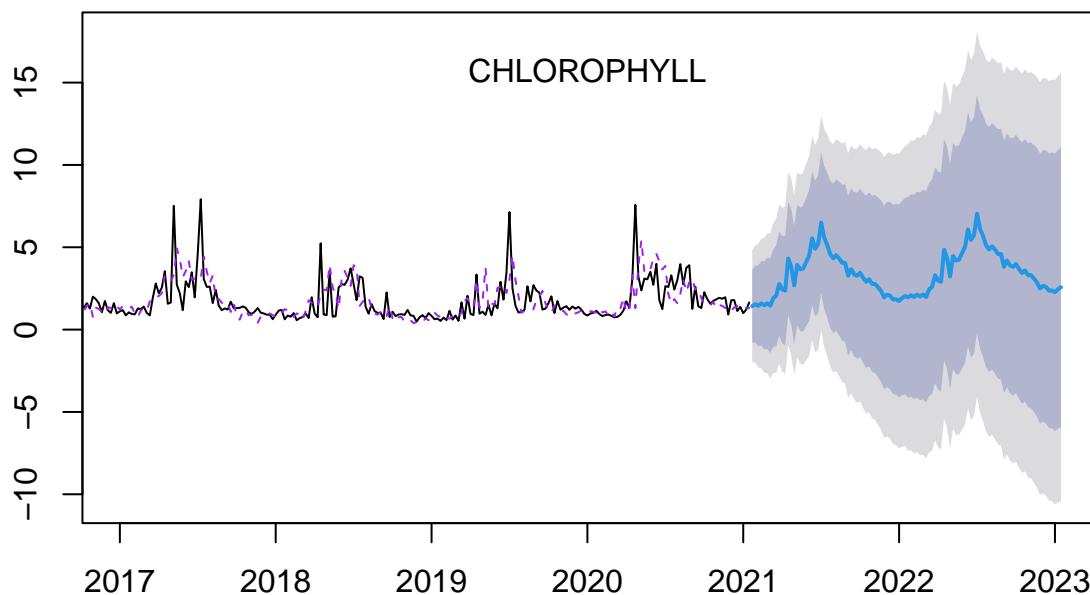
```

#z211 <- get(param, owt_df02_gb09_ploo_wkly02_test_tb03)
#ts_owt_df02_gb09_ploo_wkly02_test_tb03 <- ts(z211, start = c(2021, 1), freq = 52)

#print("Test set: MAPE")
#print(MAPE(y_pred = HW1_for11$fitted, y_true =
#  ts_owt_df02_gb09_ploo_wkly02_test_tb03))
#print("Test Set: MAE")
#print(MAPE(y_pred = HW1_for11$fitted, y_true =
#  ts_owt_df02_gb09_ploo_wkly02_test_tb03))
}

```

Forecasts from HoltWinters

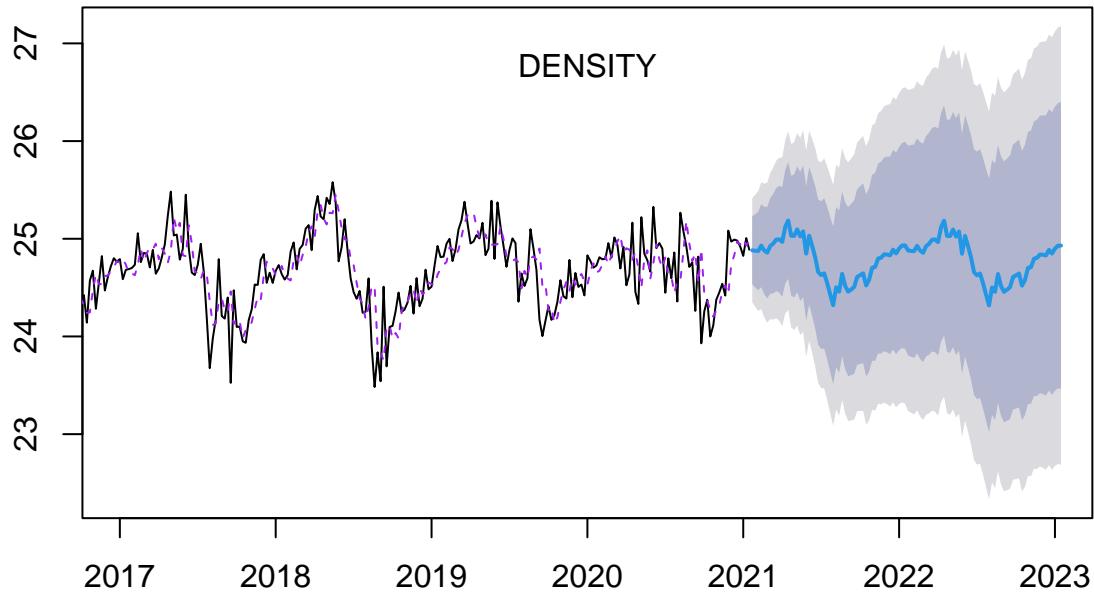


```

## [1] "Training Set Accuracy: "
## [1] "CHLOROPHYLL"
##          ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -0.07230778 1.725906 1.134430 -16.83940 46.75320 0.6076721
## Test set      1.46859743 2.740949 2.012627 15.57971 43.70445 1.0780900
##          ACF1 Theil's U
## Training set 0.2434585      NA
## Test set     0.9008239  0.820493

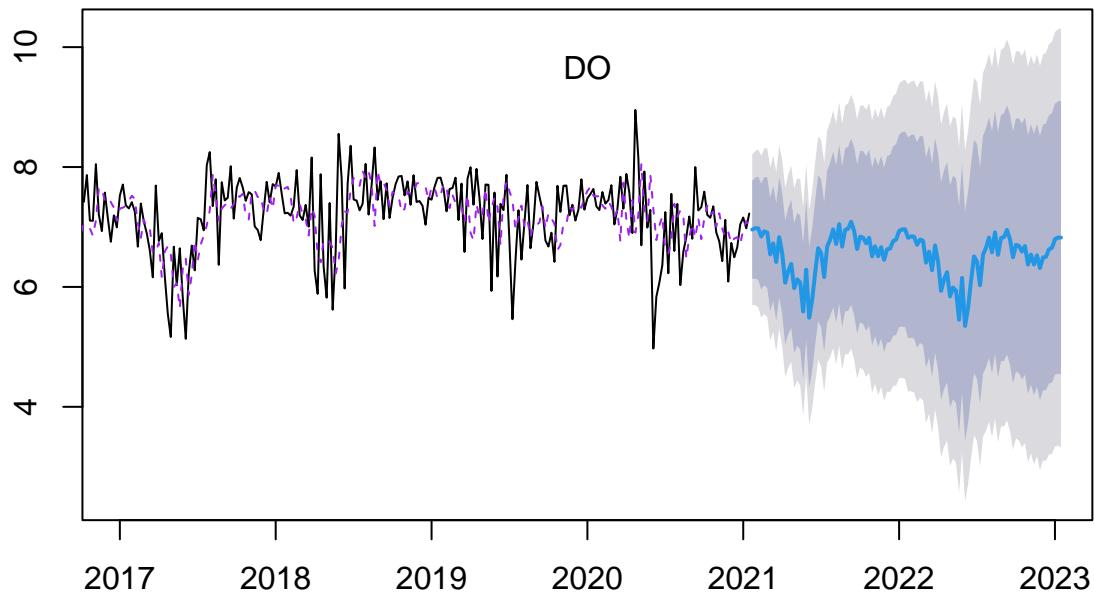
```

Forecasts from HoltWinters



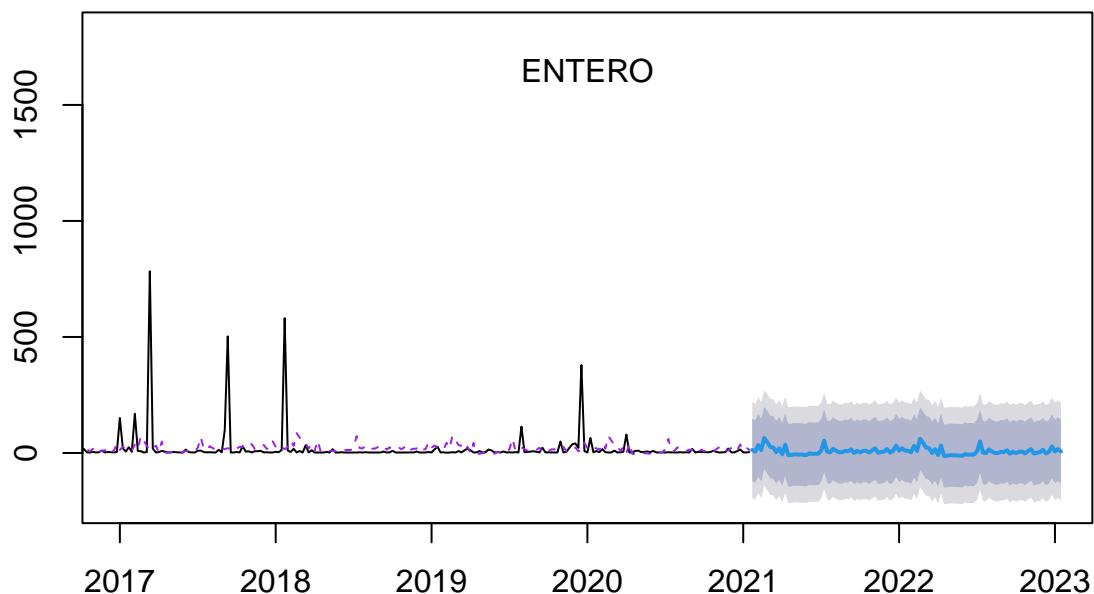
```
## [1] "Training Set Accuracy: "
## [1] "DENSITY"
##          ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -0.001112587 0.2715231 0.2071139 -0.01337993 0.8378777 0.6208778
## Test set      0.465221312 0.5774284 0.5399175  1.82521152 2.134650 1.6185433
##          ACF1 Theil's U
## Training set 0.1791689       NA
## Test set     0.7086409  2.218613
```

Forecasts from HoltWinters



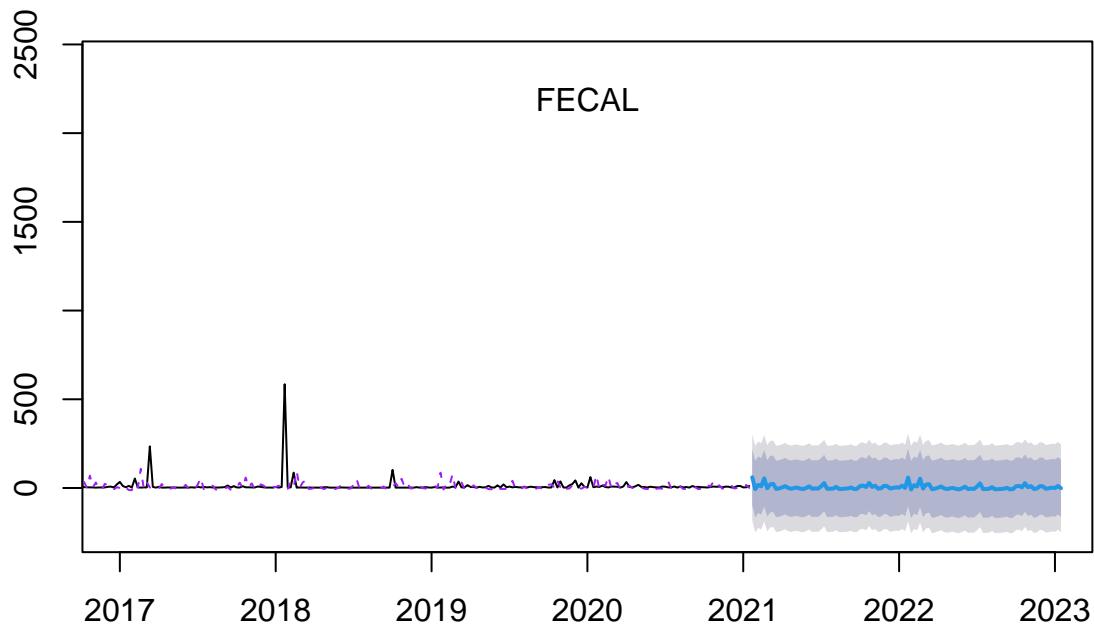
```
## [1] "Training Set Accuracy: "
## [1] "DO"
##          ME      RMSE      MAE      MPE      MAPE      MASE
## Training set 0.02545076 0.6398213 0.4815780 -0.2703567 7.050892 0.6767534
## Test set     0.15675584 0.7675907 0.5989928  1.3562919 8.964217 0.8417544
##          ACF1 Theil's U
## Training set 0.1841093       NA
## Test set     0.6931196 1.591299
```

Forecasts from HoltWinters



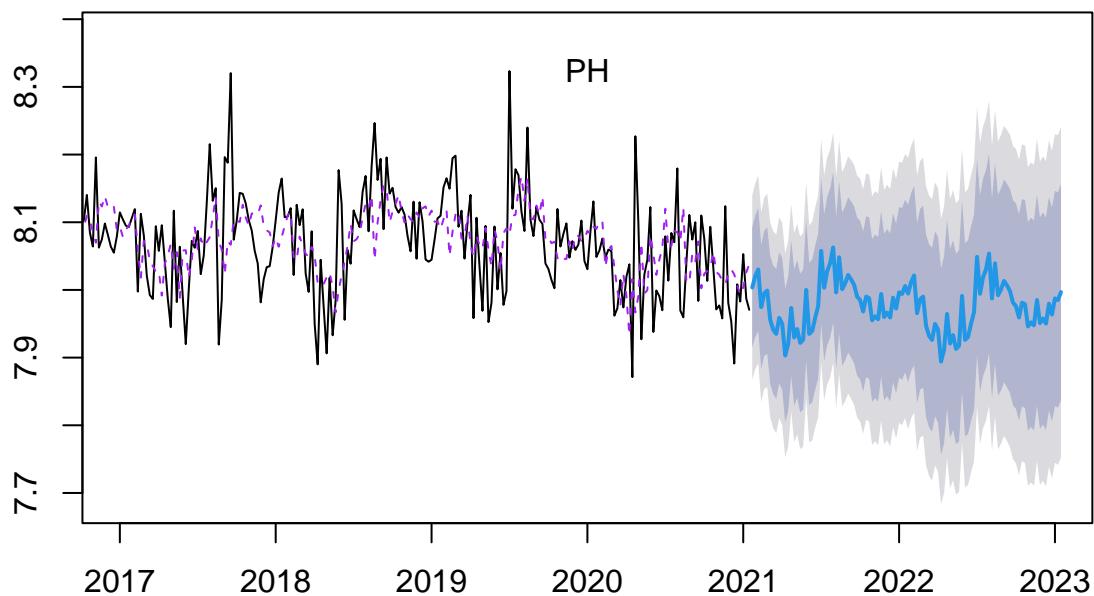
```
## [1] "Training Set Accuracy: "
## [1] "ENTERO"
##           ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -3.204651 105.31660 32.34395 -474.38849 522.7562 0.9910613
## Test set     11.352043 67.78757 18.96754   13.89866 193.6654 0.5811905
##           ACF1 Theil's U
## Training set 0.025656510       NA
## Test set     0.005257075  1.023268
```

Forecasts from HoltWinters



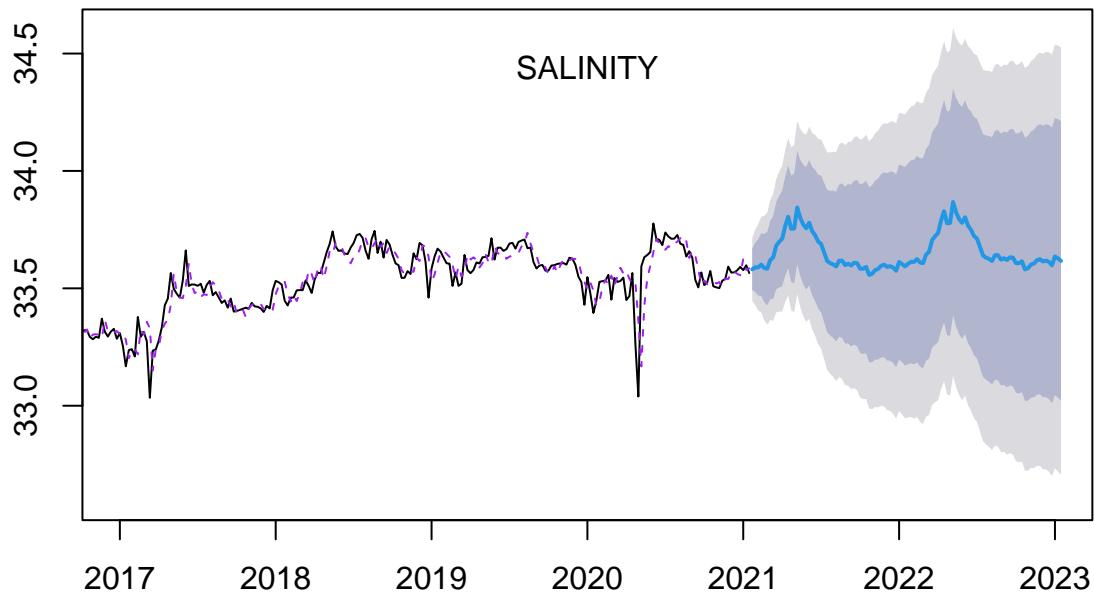
```
## [1] "Training Set Accuracy: "
## [1] "FECAL"
##               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -3.474252 124.1303 44.49475 -376.5721 568.9474 0.9692438
## Test set      67.346465 123.6525 71.99182   49.9243 179.3118 1.5682215
##                         ACF1 Theil's U
## Training set  0.0009713377       NA
## Test set      -0.1457515171 0.9931063
```

Forecasts from HoltWinters



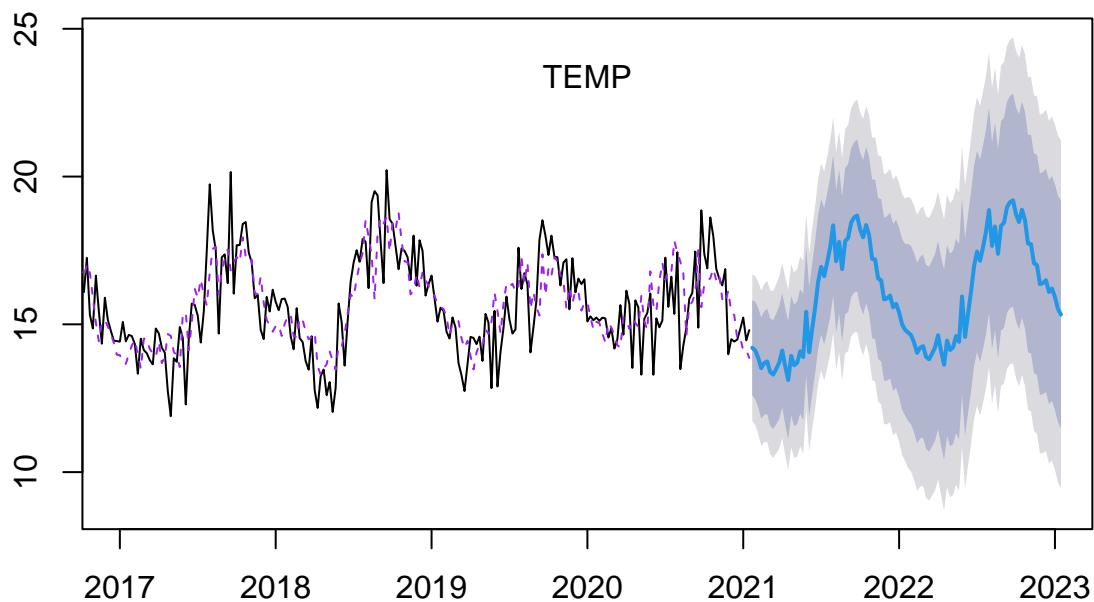
```
## [1] "Training Set Accuracy: "
## [1] "PH"
##               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set 0.001620217 0.06933147 0.05215344 0.01451201 0.6469997 0.6740749
## Test set     -0.064314804 0.10090137 0.08533079 -0.81999341 1.0807314 1.1028868
##               ACF1 Theil's U
## Training set 0.2460483       NA
## Test set     0.6859840 2.022263
```

Forecasts from HoltWinters



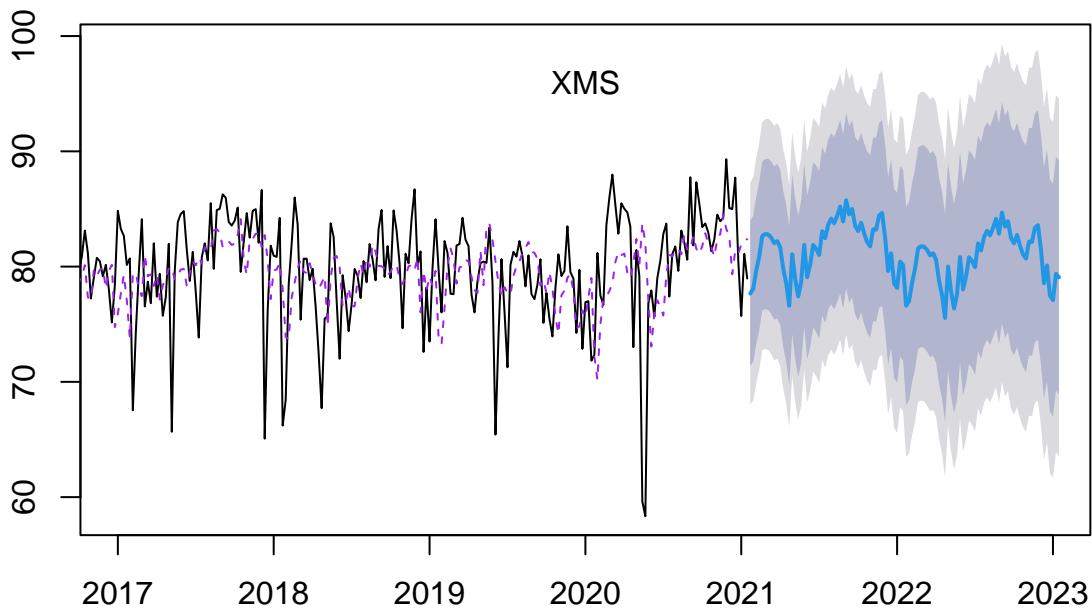
```
## [1] "Training Set Accuracy: "
## [1] "SALINITY"
##               ME      RMSE      MAE      MPE      MAPE
## Training set -0.0007432887 0.06837147 0.04810312 -0.002499238 0.143628
## Test set      0.0252913576 0.07093237 0.05722505  0.074943441 0.169826
##               MASE      ACF1 Theil's U
## Training set 0.4088266 0.05644603       NA
## Test set     0.4863535 0.73566685 1.674141
```

Forecasts from HoltWinters



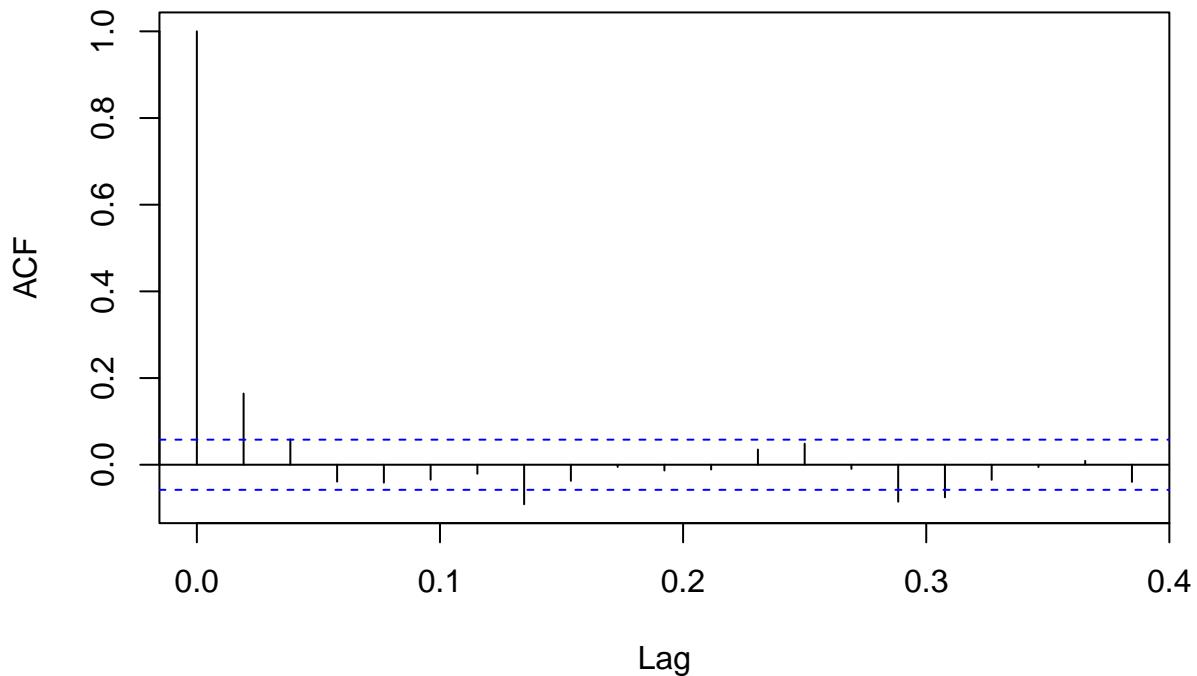
```
## [1] "Training Set Accuracy: "
## [1] "TEMP"
##               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -0.03940712 1.257878 0.958221 -0.8131094 6.317697 0.7049208
## Test set      -1.75534741 2.349357 2.012914 -12.8461813 14.461637 1.4808119
##               ACF1 Theil's U
## Training set 0.1825240        NA
## Test set     0.4857403 1.501594
```

Forecasts from HoltWinters



```
## [1] "Training Set Accuracy: "
## [1] "XMS"
##          ME      RMSE      MAE      MPE      MAPE      MASE
## Training set 0.1664348 4.909606 3.580805 -0.09478933 4.566404 0.7202584
## Test set     0.3826506 4.679530 3.531065  0.16848625 4.469536 0.7102536
##          ACF1 Theil's U
## Training set 0.1642423       NA
## Test set     0.1978604 0.7965938
acf(HW1_for11$residuals, lag.max=20, na.action=na.pass)
```

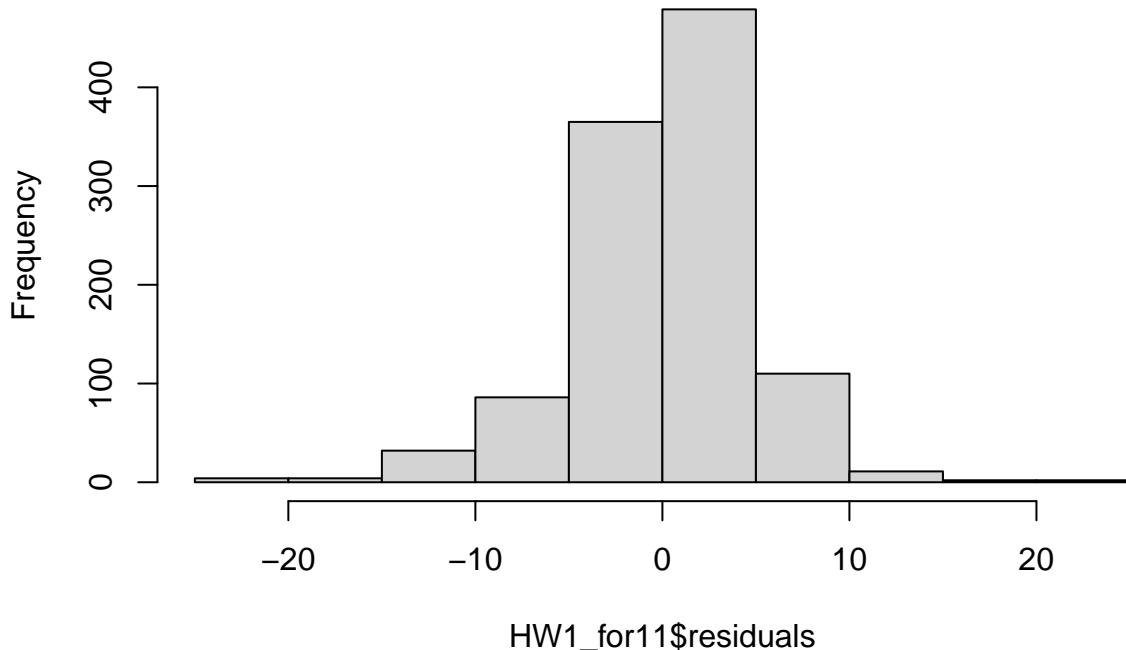
Series HW1_for11\$residuals



```
Box.test(HW1_for11$residuals, lag=20, type="Ljung-Box")
```

```
##  
## Box-Ljung test  
##  
## data: HW1_for11$residuals  
## X-squared = 71.463, df = 20, p-value = 1.049e-07  
hist(HW1_for11$residuals)
```

Histogram of HW1_for11\$residuals



```
# convert to time series object ts
params <- c("CHLOROPHYLL", "DENSITY", "DO", "ENTERO", "FECAL", "PH", "SALINITY", "TEMP",
          "XMS")

for (param in params) {
  z <- get(param, owt_df02_gb09_sboo_wkly03_train_tb03)
  ts_owt_df02_gb09_sboo_wkly03_train_tb03 <- ts(z, start = c(1999, 1), freq = 52)
  ts_owt_df02_gb09_sboo_wkly03_test_tb03 <- ts(z, start = c(2021, 1), freq = 52)

  HW1 <- HoltWinters(ts_owt_df02_gb09_sboo_wkly03_train_tb03)

  # forecast library more robust features vs. predict()

  HW1_for <- forecast(HW1, h=104, level=c(80,95))
  #visualize our predictions:
  plot(HW1_for, xlim=c(2017, 2023))
  lines(HW1_for$fitted, lty=2, col="purple")
  mtext(param, side=3, line = -2)

  print("Accuracy: ")
  print(param)
  #print(accuracy(HW1_for))
  print(accuracy(HW1_for, ts_owt_df02_gb09_sboo_wkly03_test_tb03))

  # test set
  #z2 <- get(param, owt_df02_gb09_sboo_wkly03_test_tb03)
```

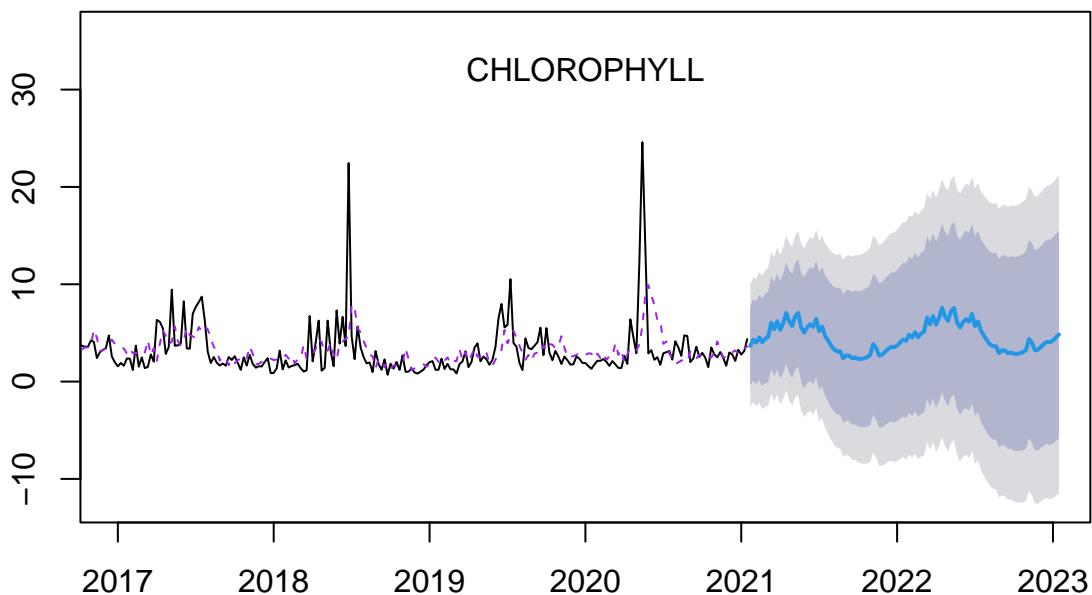
```

#ts_owt_df02_gb09_sboo_wkly03_test_tb03 <- ts(z2, start = c(2021, 1), freq = 52)

#print("Test set: MAPE")
#print(MAPE(y_pred = HW1_for$fitted, y_true = ts_owt_df02_gb09_sboo_wkly03_test_tb03))
#print("Test Set: MAE")
#print(MAE(y_pred = HW1_for$fitted, y_true = ts_owt_df02_gb09_sboo_wkly03_test_tb03))
}

```

Forecasts from HoltWinters

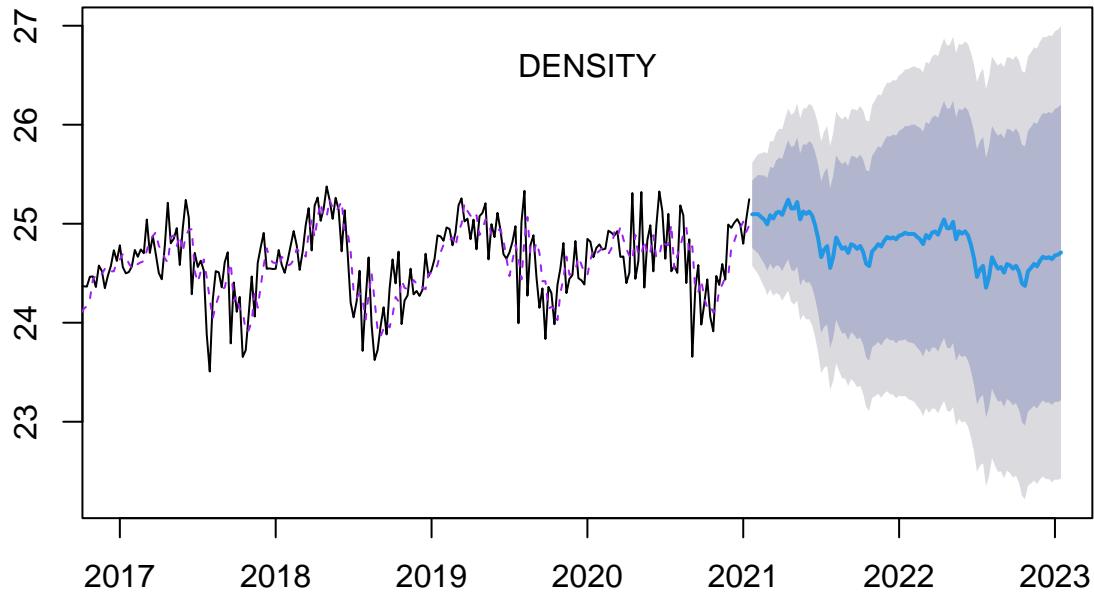


```

## [1] "Accuracy: "
## [1] "CHLOROPHYLL"
##               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -0.1022793 3.190700 1.971592 -26.97607 48.25139 0.6671699
## Test set      2.2463908 3.116269 2.841425  20.96796 49.37168 0.9615141
##               ACF1 Theil's U
## Training set 0.1393202       NA
## Test set     0.8615906  4.386688

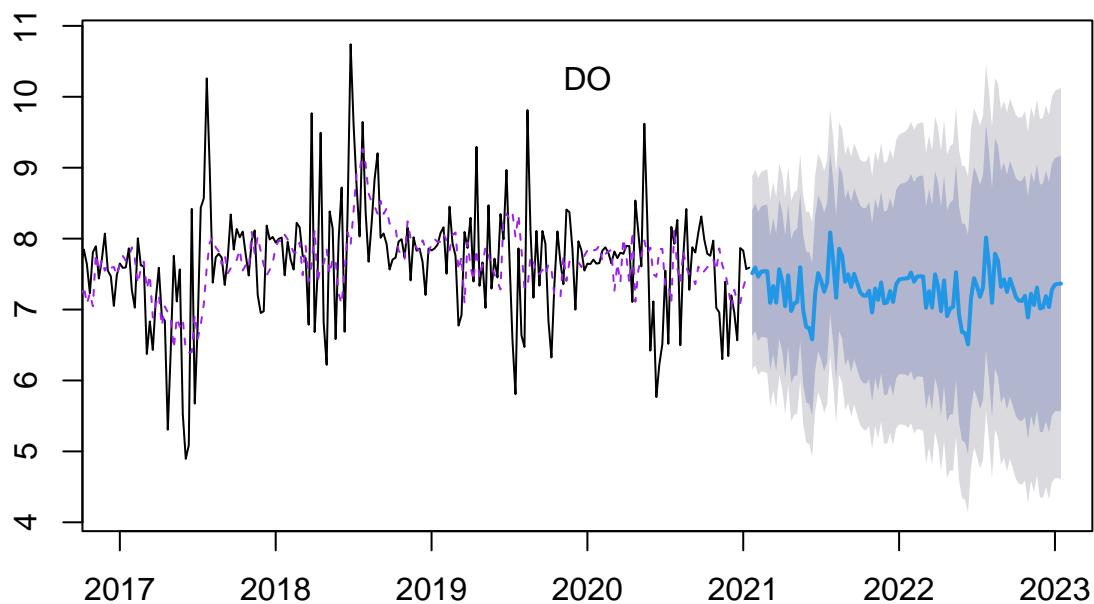
```

Forecasts from HoltWinters



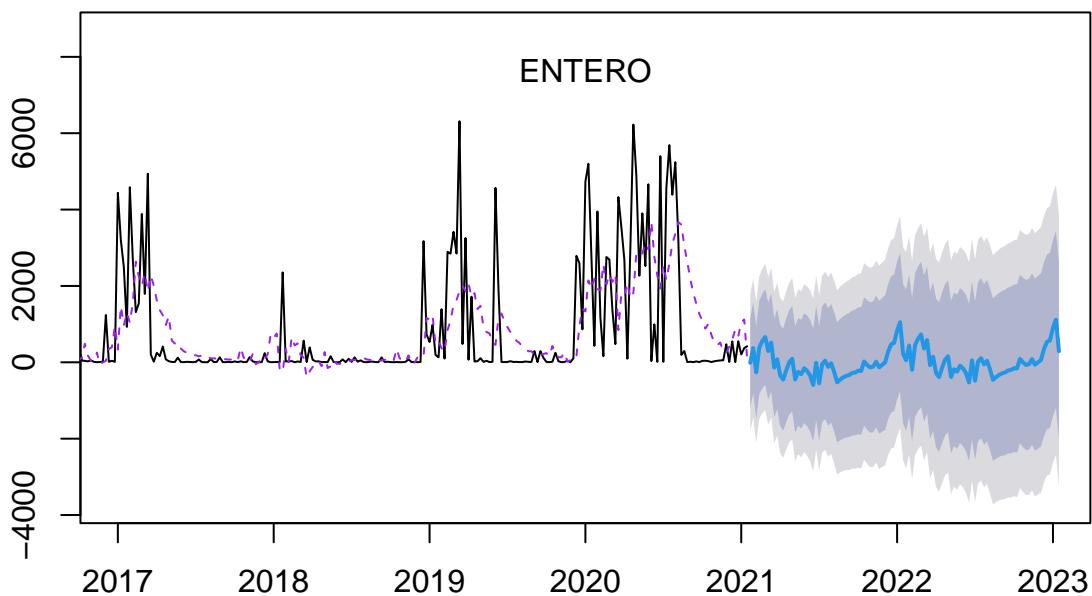
```
## [1] "Accuracy: "
## [1] "DENSITY"
##          ME      RMSE      MAE      MPE      MAPE      MASE
## Training set 0.00868597 0.2661097 0.1956398 0.02645349 0.7947002 0.5767382
## Test set     0.27439396 0.3133471 0.2866552 1.09067922 1.1405087 0.8450478
##          ACF1 Theil's U
## Training set 0.09458735       NA
## Test set     0.79082440 4.18588
```

Forecasts from HoltWinters



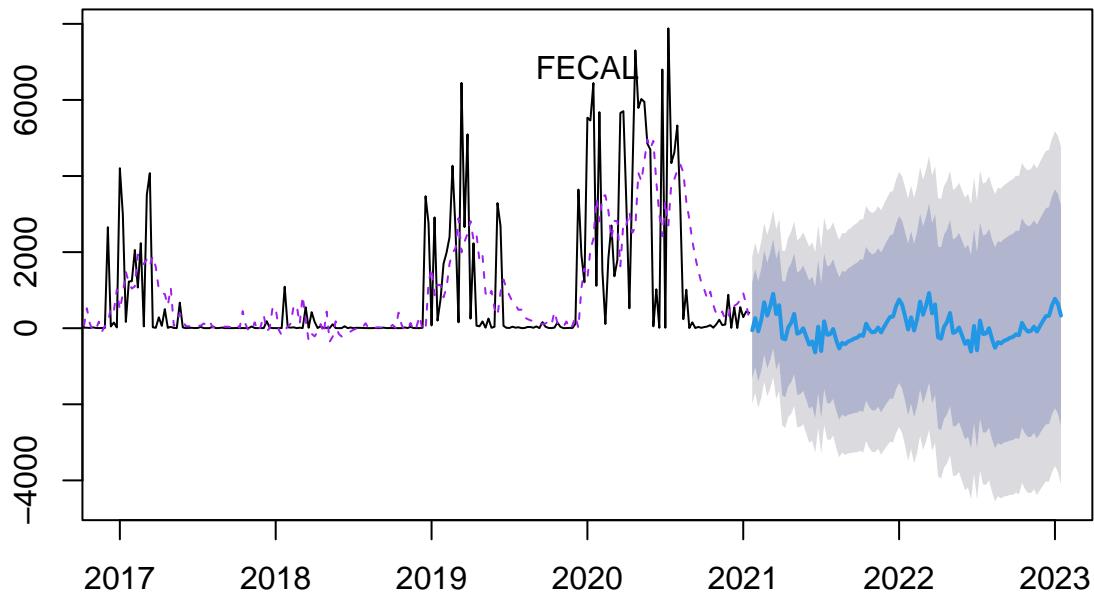
```
## [1] "Accuracy: "
## [1] "DO"
##          ME      RMSE      MAE      MPE      MAPE      MASE
## Training set 0.02593868 0.6952922 0.4998909 -0.3376457 6.742103 0.6979608
## Test set     0.47952239 0.7183909 0.6073764  5.7991563 7.665913 0.8480349
##          ACF1 Theil's U
## Training set 0.1367533       NA
## Test set     0.7973523  3.82567
```

Forecasts from HoltWinters



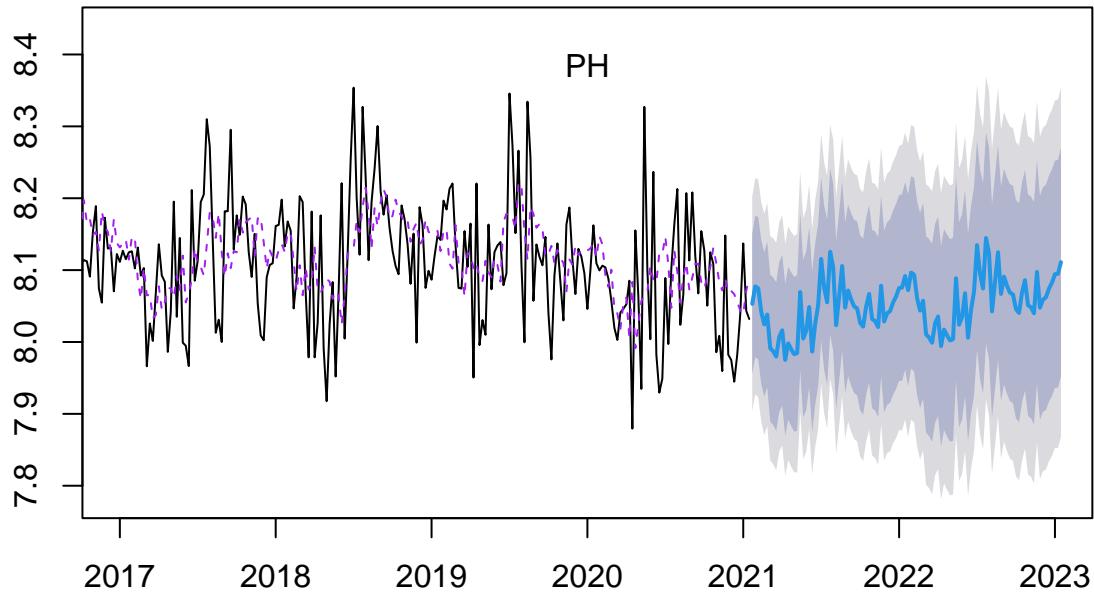
```
## [1] "Accuracy: "
## [1] "ENTERO"
##               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -8.237838 913.6779 433.4422 -1818.639 2346.492 0.8845875
## Test set     88.810647 427.8780 310.5345  1109.308 3379.248 0.6337522
##               ACF1 Theil's U
## Training set 0.07100116       NA
## Test set     0.43072703 3.482345
```

Forecasts from HoltWinters



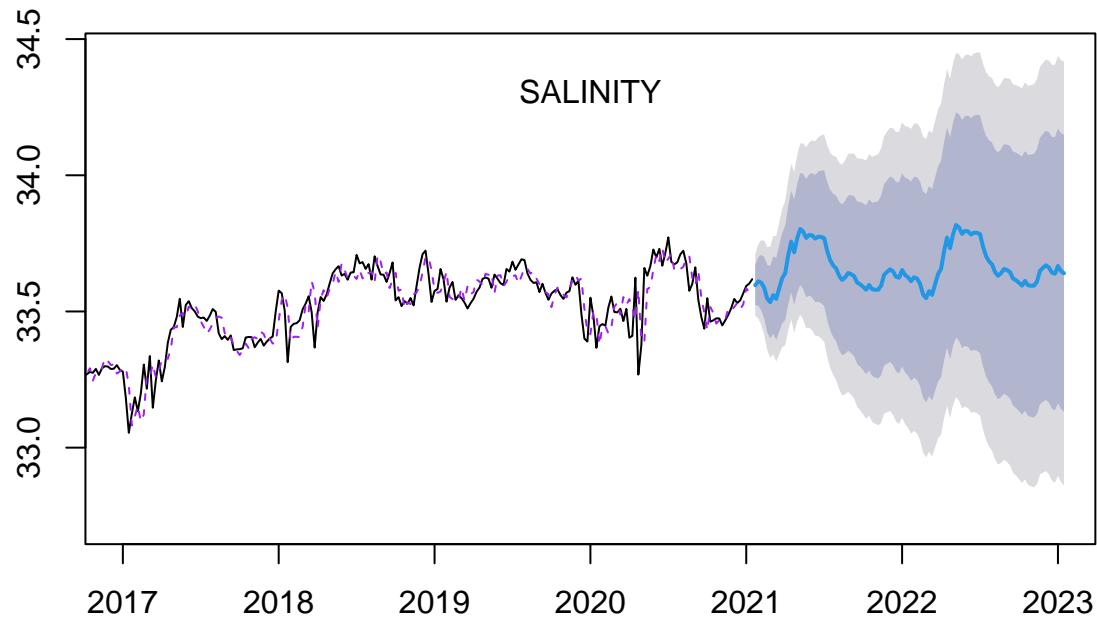
```
## [1] "Accuracy: "
## [1] "FECAL"
##          ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -2.423009 986.0725 516.5492 -1721.0533 2728.081 0.8481016
## Test set     124.304142 436.6030 327.8944    760.6152 2377.755 0.5383567
##          ACF1 Theil's U
## Training set 0.08313998       NA
## Test set     0.27502492 1.737434
```

Forecasts from HoltWinters



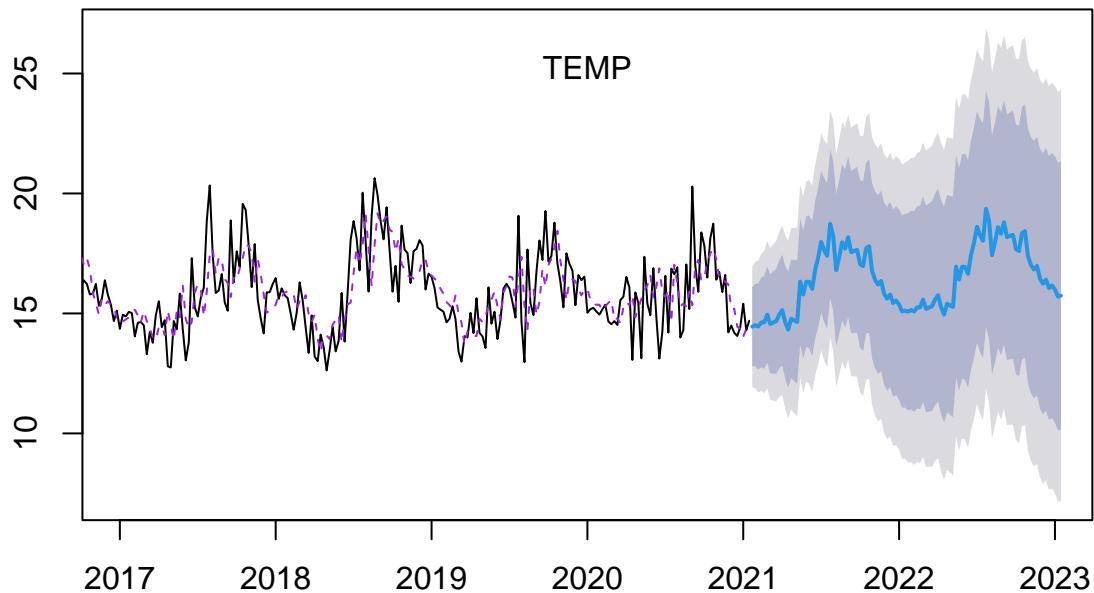
```
## [1] "Accuracy: "
## [1] "PH"
##               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -0.002480009 0.07616416 0.05771275 -0.03748209 0.7123814 0.7056536
## Test set      -0.070034174 0.08993278 0.07855987 -0.88159567 0.9870175 0.9605513
##               ACF1 Theil's U
## Training set 0.1898242       NA
## Test set     0.7330344 3.832939
```

Forecasts from HoltWinters



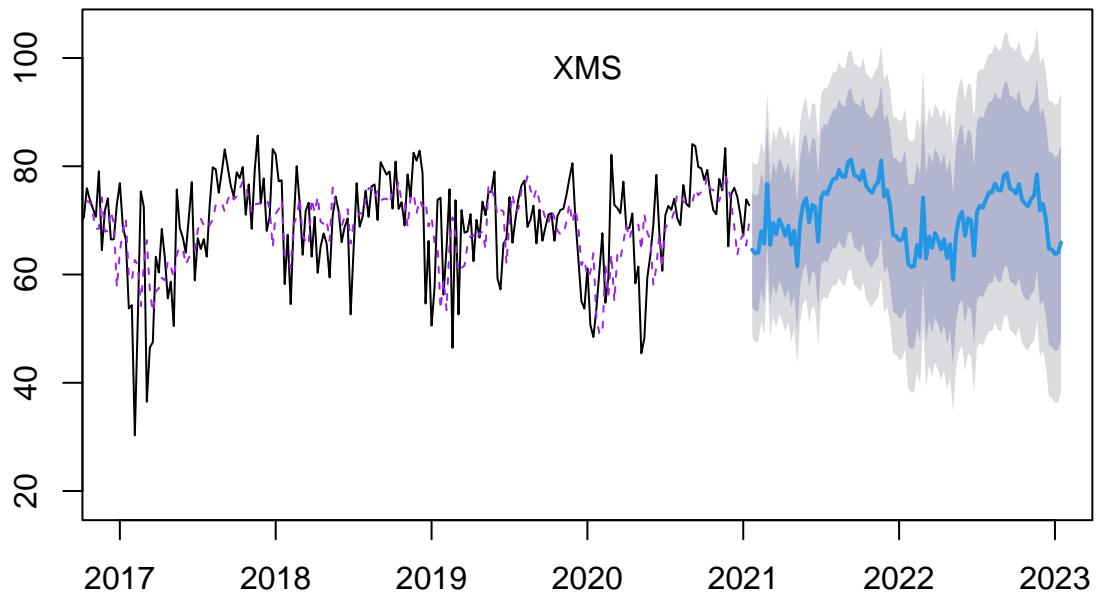
```
## [1] "Accuracy: "
## [1] "SALINITY"
##               ME      RMSE      MAE      MPE      MAPE
## Training set -0.0003848443 0.05763247 0.04015352 -0.00132097 0.1199759
## Test set      -0.0358582292 0.06060877 0.05026666 -0.10671437 0.1495097
##             MASE      ACF1 Theil's U
## Training set 0.3420641 0.07059847       NA
## Test set     0.4282170 0.84038594 3.028187
```

Forecasts from HoltWinters



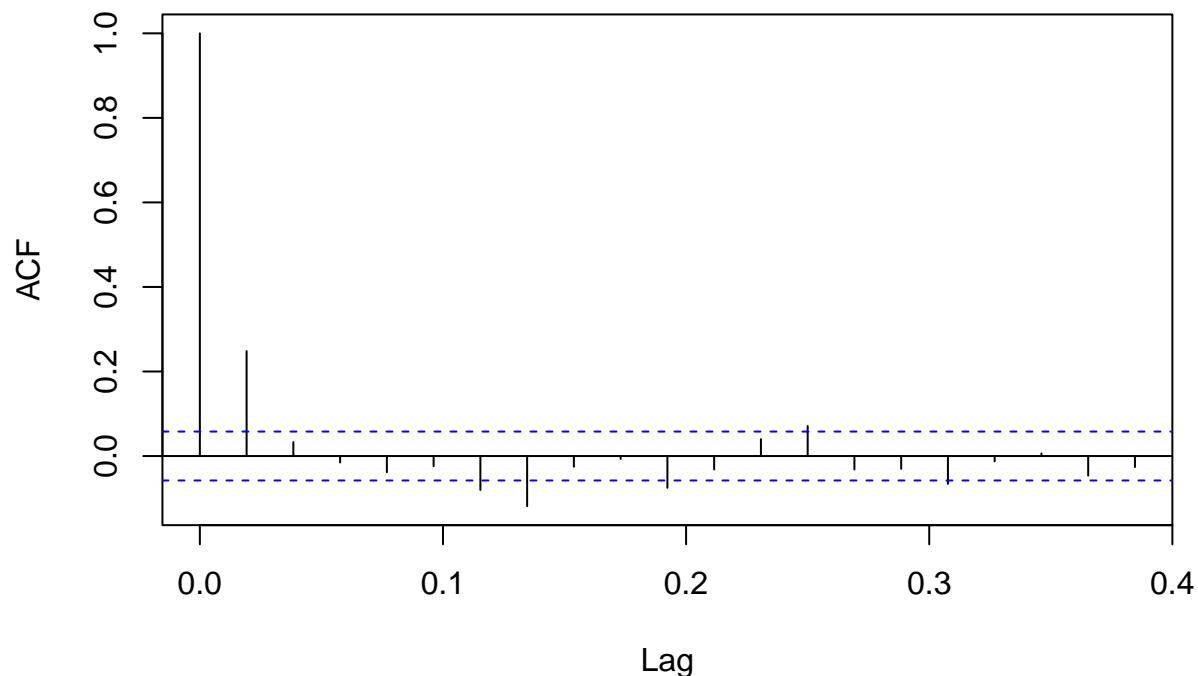
```
## [1] "Accuracy: "
## [1] "TEMP"
##               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -0.04812992 1.284493 0.9654215 -0.8723829 6.330799 0.6500255
## Test set      -2.01096071 2.299520 2.0576695 -14.4547515 14.730957 1.3854442
##               ACF1 Theil's U
## Training set 0.1110604        NA
## Test set     0.3652595  2.025057
```

Forecasts from HoltWinters



```
## [1] "Accuracy: "
## [1] "XMS"
##          ME      RMSE      MAE      MPE      MAPE      MASE
## Training set 0.3391581 8.286845 5.843124 -0.9457464 8.95783 0.7447144
## Test set     4.3649706 9.367104 7.762394  4.3315522 11.00968 0.9893282
##          ACF1 Theil's U
## Training set 0.2482904        NA
## Test set     0.4794833 0.9970226
acf(HW1_for$residuals, lag.max=20, na.action=na.pass)
```

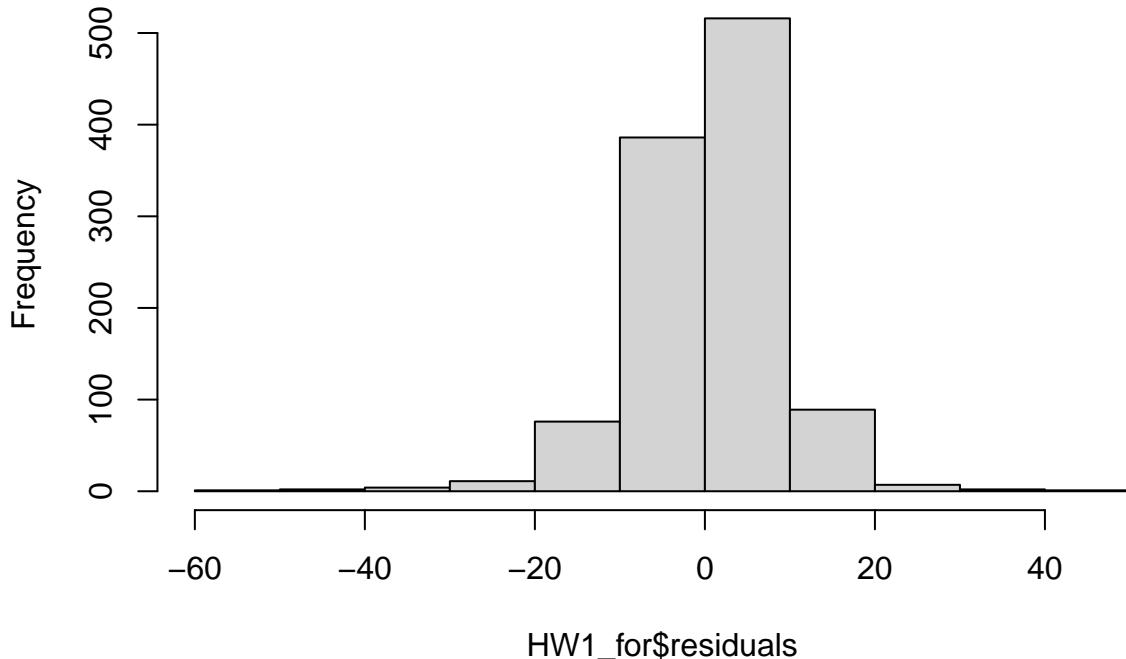
Series HW1_for\$residuals



```
Box.test(HW1_for$residuals, lag=20, type="Ljung-Box")
```

```
##  
## Box-Ljung test  
##  
## data: HW1_for$residuals  
## X-squared = 119.98, df = 20, p-value = 3.331e-16  
hist(HW1_for$residuals)
```

Histogram of HW1_for\$residuals



```
# convert to time series object ts
params <- c("CHLOROPHYLL", "DENSITY", "DO", "ENTERO", "FECAL", "PH", "SALINITY", "TEMP",
          "XMS")

for (param in params) {
  #print(param)

  z <- get(param, owt_df02_gb09_sboo_wkly02_train_tb03)
  ts_owt_df02_gb09_sboo_wkly02_train_tb03 <- ts(z, start = c(1999, 1), freq = 52)
  ts_owt_df02_gb09_sboo_wkly02_test_tb03 <- ts(z, start = c(2021, 1), freq = 52)

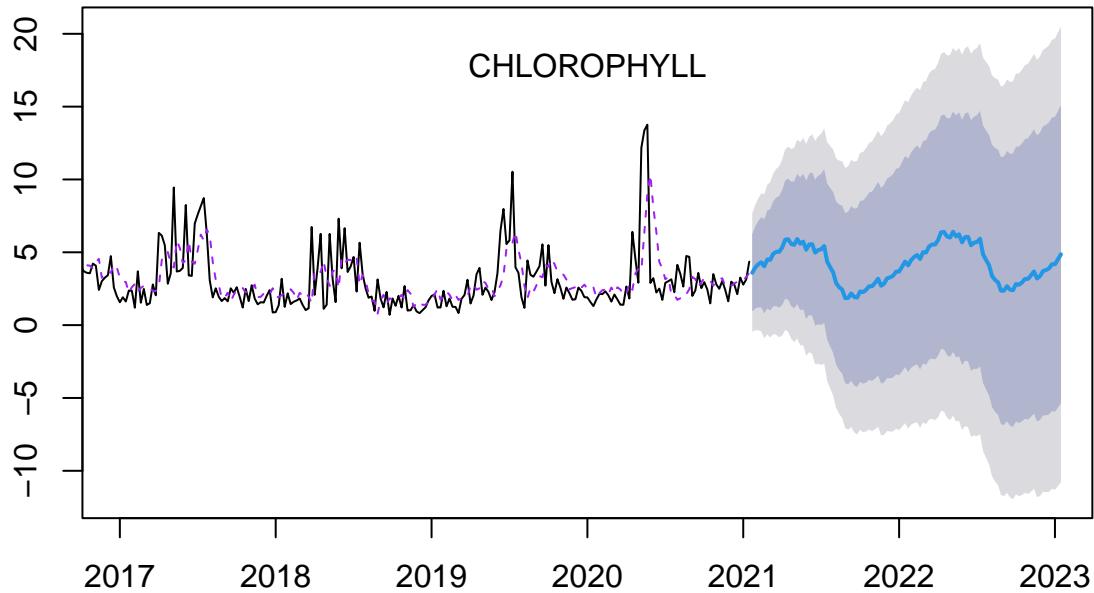
  HW1 <- HoltWinters(ts_owt_df02_gb09_sboo_wkly02_train_tb03)

  # forecast library more robust features vs. predict()

  HW1_for <- forecast(HW1, h=104, level=c(80,95))
  #visualize our predictions:
  plot(HW1_for, xlim=c(2017, 2023))
  lines(HW1_for$fitted, lty=2, col="purple")
  mtext(param, side=3, line = -2)

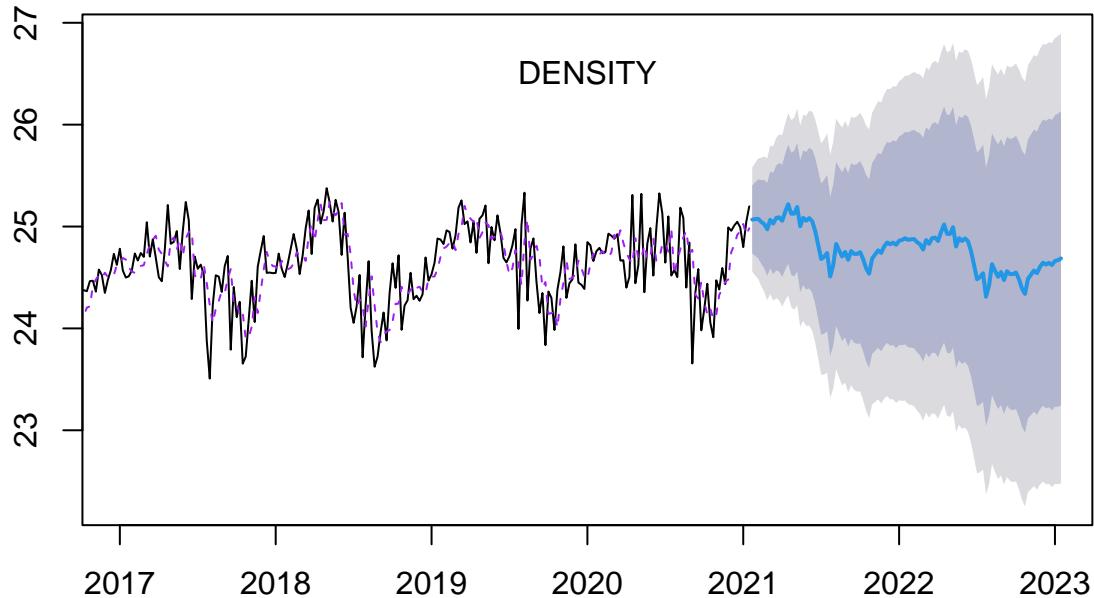
  print("Accuracy: ")
  print(param)
  #print(accuracy(HW1_for))
  print(accuracy(HW1_for, ts_owt_df02_gb09_sboo_wkly02_test_tb03))
}
```

Forecasts from HoltWinters



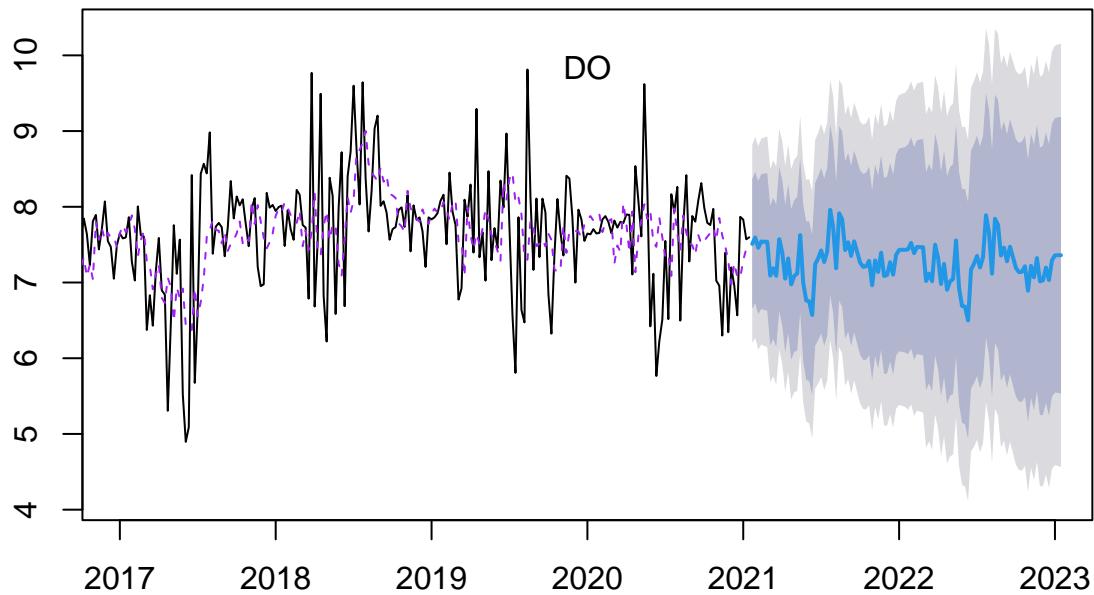
```
## [1] "Accuracy: "
## [1] "CHLOROPHYLL"
##               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -0.06694596 2.07424 1.446251 -19.13824 39.30235 0.6219099
## Test set      2.58216928 3.26962 3.062433 26.85520 52.02599 1.3168933
##               ACF1 Theil's U
## Training set 0.1182405       NA
## Test set     0.8906571 4.286236
```

Forecasts from HoltWinters



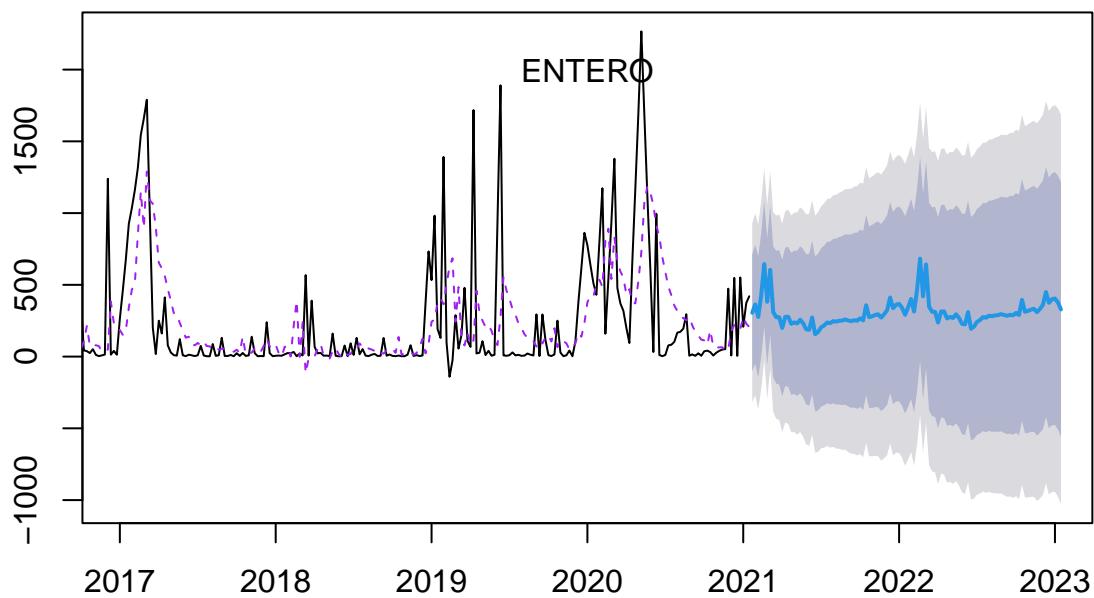
```
## [1] "Accuracy: "
## [1] "DENSITY"
##               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set 0.008688512 0.2608637 0.1927412 0.02679585 0.7821223 0.5777927
## Test set     0.300455960 0.3358497 0.3088433 1.19467905 1.2288210 0.9258395
##               ACF1 Theil's U
## Training set 0.09581955       NA
## Test set     0.77732726 4.435291
```

Forecasts from HoltWinters



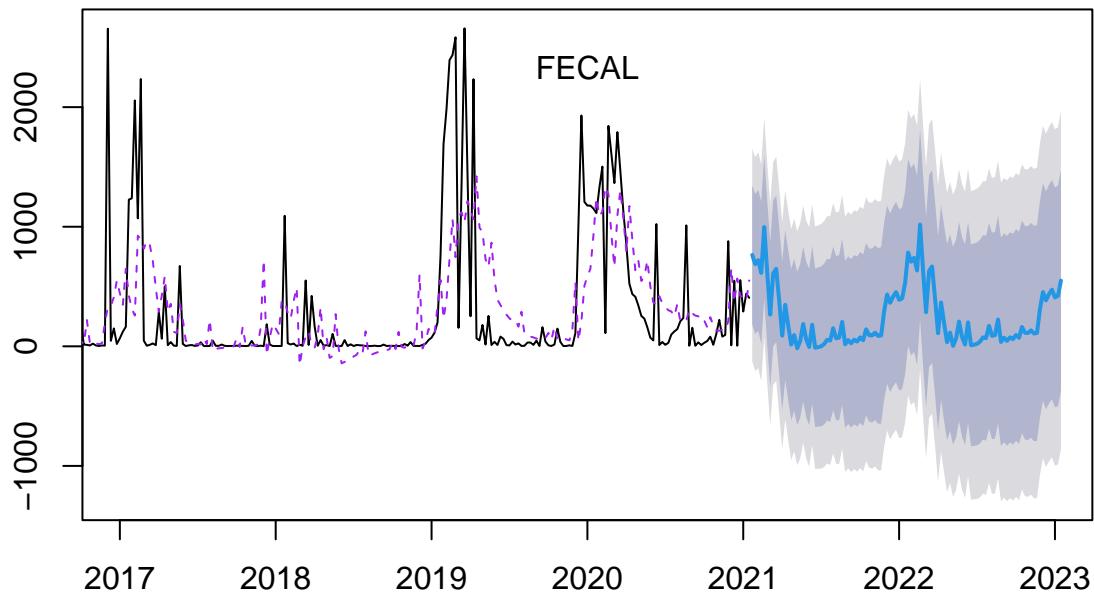
```
## [1] "Accuracy: "
## [1] "DO"
##               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set 0.02467105 0.6680607 0.4898582 -0.3108788 6.644379 0.6989592
## Test set     0.48146070 0.7209477 0.6108443  5.8234369 7.703702 0.8715894
##               ACF1 Theil's U
## Training set 0.1516258       NA
## Test set     0.8095373 3.736292
```

Forecasts from HoltWinters



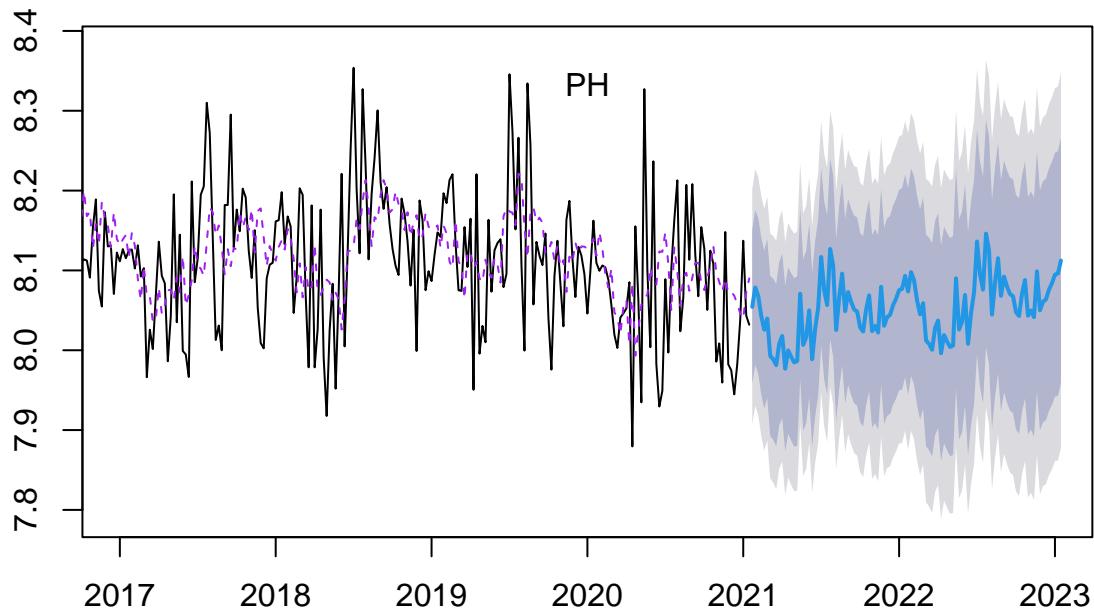
```
## [1] "Accuracy: "
## [1] "ENTERO"
##               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -2.640892 317.5309 168.5037 -947.1598 1027.944 0.8465839
## Test set     -243.884222 293.2887 274.0600 -3284.0870 3286.794 1.3769121
##               ACF1 Theil's U
## Training set 0.1238312       NA
## Test set     0.4393069 3.004776
```

Forecasts from HoltWinters



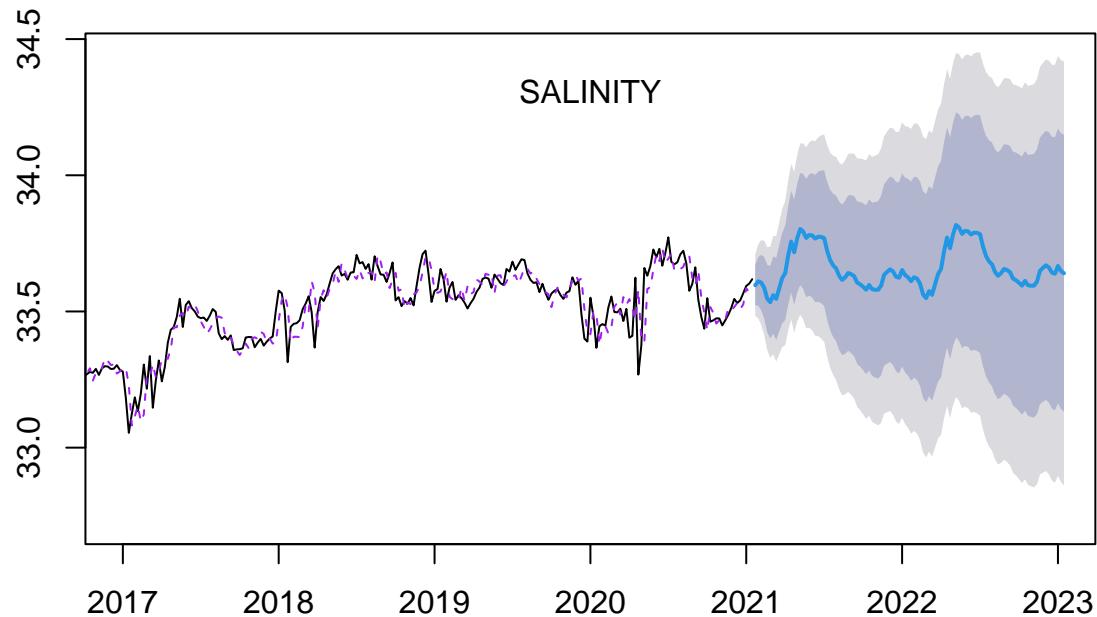
```
## [1] "Accuracy: "
## [1] "FECAL"
##               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -2.411267 453.1053 257.4368 -1237.933 1623.945 0.8751207
## Test set     -95.636725 376.7525 242.4843 -1263.927 1306.991 0.8242916
##               ACF1 Theil's U
## Training set 0.1190698       NA
## Test set     0.1129563 1.18184
```

Forecasts from HoltWinters



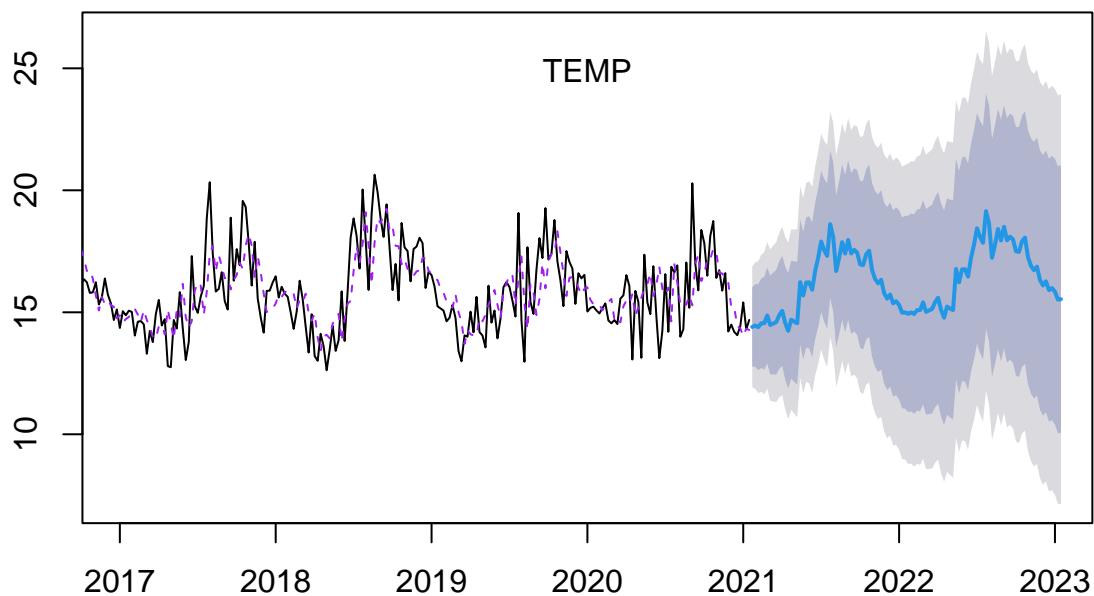
```
## [1] "Accuracy: "
## [1] "PH"
##               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -0.002574413 0.07509104 0.05699028 -0.03846183 0.7037097 0.7067346
## Test set      -0.070896564 0.09016864 0.07907046 -0.89233246 0.9934076 0.9805502
##               ACF1 Theil's U
## Training set 0.2008171       NA
## Test set     0.7361375 3.900941
```

Forecasts from HoltWinters



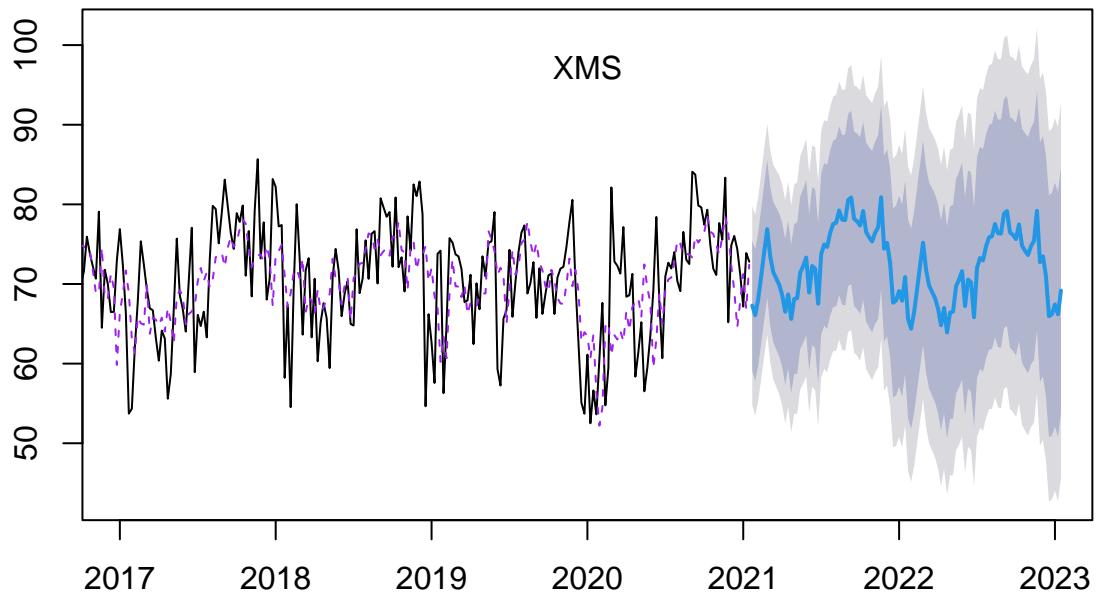
```
## [1] "Accuracy: "
## [1] "SALINITY"
##               ME      RMSE      MAE      MPE      MAPE
## Training set -0.0003848443 0.05763247 0.04015352 -0.00132097 0.1199759
## Test set      -0.0358582292 0.06060877 0.05026666 -0.10671437 0.1495097
##             MASE      ACF1 Theil's U
## Training set 0.3420641 0.07059847       NA
## Test set     0.4282170 0.84038594 3.028187
```

Forecasts from HoltWinters



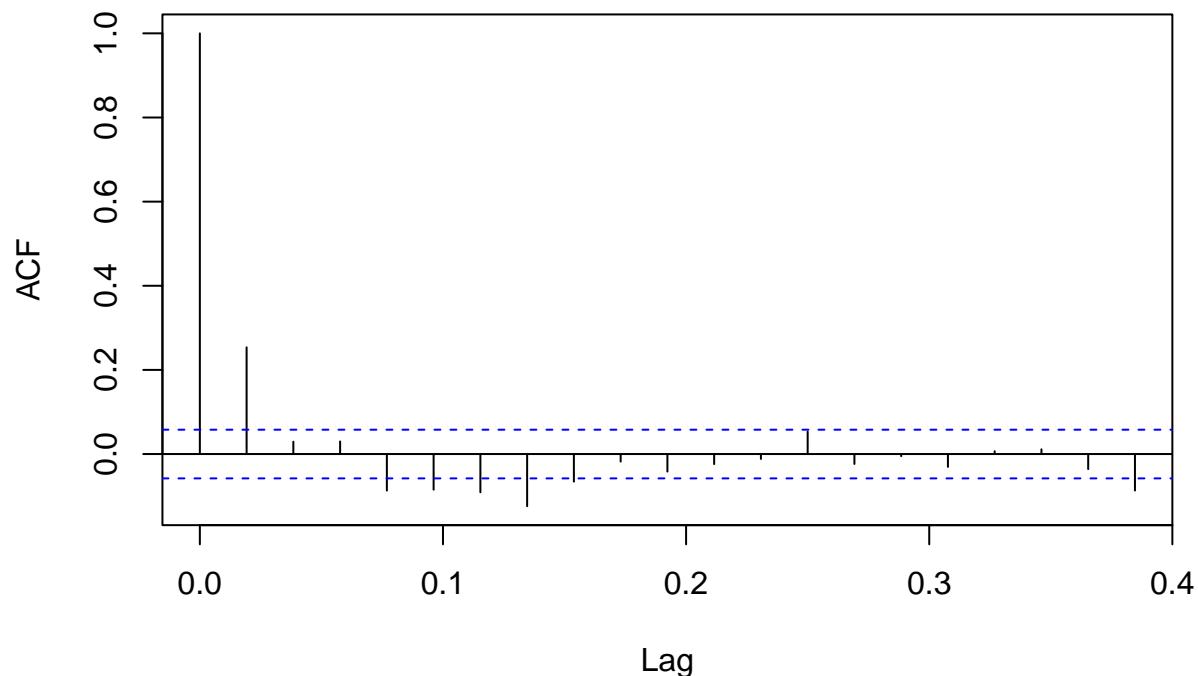
```
## [1] "Accuracy: "
## [1] "TEMP"
##               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -0.04561308 1.266222 0.9546388 -0.8460127 6.281475 0.6534357
## Test set      -1.86895425 2.177601 1.9362525 -13.4828783 13.876546 1.3253353
##               ACF1 Theil's U
## Training set 0.1068047       NA
## Test set     0.3822746  1.922185
```

Forecasts from HoltWinters



```
## [1] "Accuracy: "
## [1] "XMS"
##          ME      RMSE      MAE      MPE      MAPE      MASE
## Training set 0.1912608 6.364792 4.912165 -0.3472524 6.839092 0.7313659
## Test set     3.6391402 7.886891 6.643305  4.1186290 8.814771 0.9891132
##          ACF1 Theil's U
## Training set 0.2537163       NA
## Test set     0.5227412 1.286813
acf(HW1_for$residuals, lag.max=20, na.action=na.pass)
```

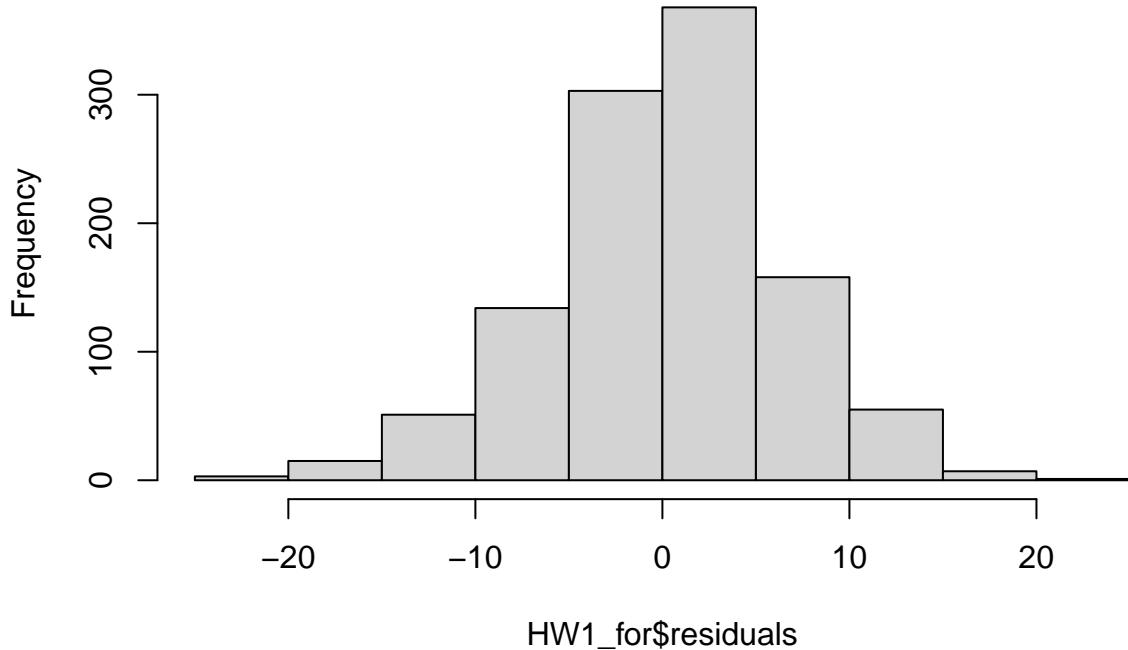
Series HW1_for\$residuals



```
Box.test(HW1_for$residuals, lag=20, type="Ljung-Box")
```

```
##  
## Box-Ljung test  
##  
## data: HW1_for$residuals  
## X-squared = 137.71, df = 20, p-value < 2.2e-16  
hist(HW1_for$residuals)
```

Histogram of HW1_for\$residuals



```
train_window <- window(ts_owt_df02_gb09_sboo_wkly02_train_tb03, start=c(2013, 1),
  end=c(2021, 1))

fit.mmn <- ets(train_window)
summary(fit.mmn)

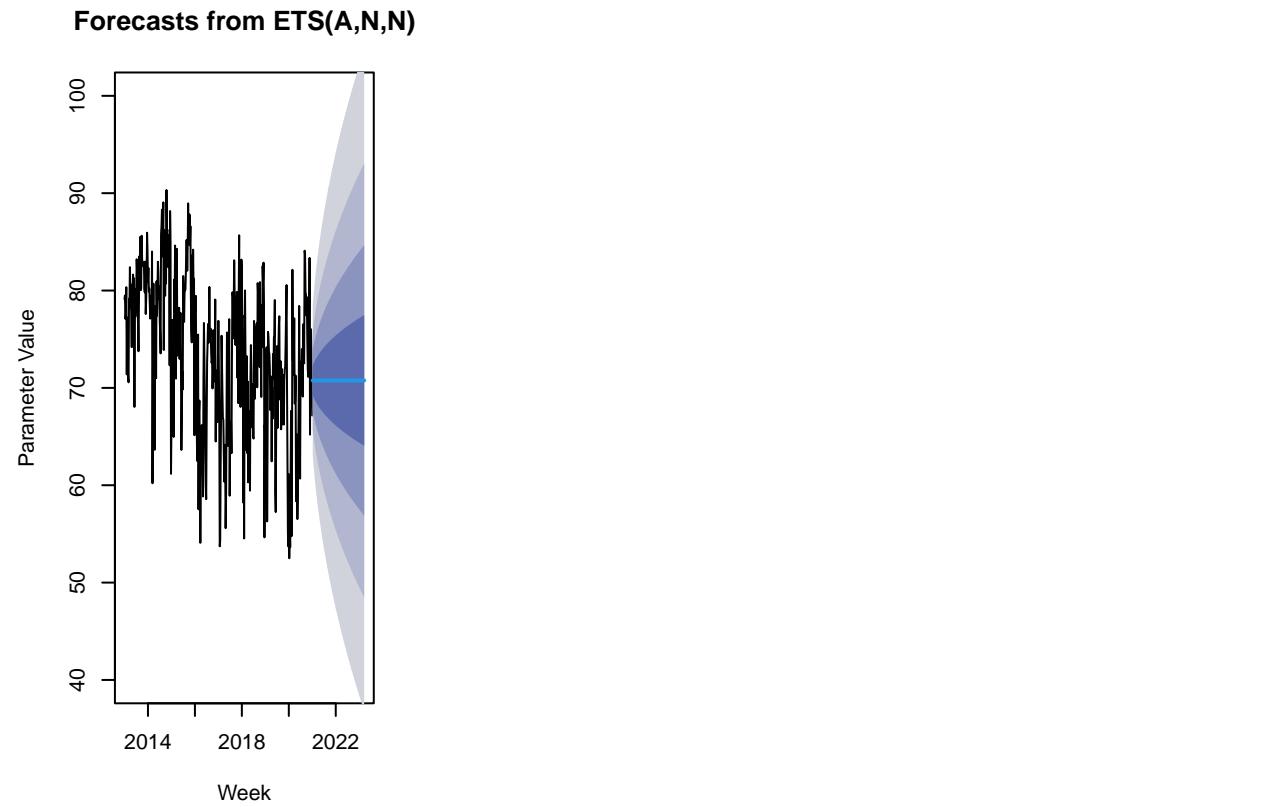
## ETS(A,N,N)
##
## Call:
##   ets(y = train_window)
##
##   Smoothing parameters:
##     alpha = 0.4044
##
##   Initial states:
##     l = 78.4301
##
##   sigma:  5.9761
##
##       AIC      AICc      BIC
## 4010.792 4010.850 4022.891
##
## Training set error measures:
##           ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -0.04546615 5.961759 4.449106 -0.6486891 6.329285 0.6455445
##           ACF1
```

```

## Training set 0.1009433
people.ets.AAN.pred <- forecast(fit.mmn, h = 115, level = c(0.2, 0.4, 0.6, 0.8))

par(mfrow = c(1, 3)) # This command sets the plot window to show 1 row of 3 plots.
plot(people.ets.AAN.pred, xlab = "Week", ylab = "Parameter Value", ylim = c(40, 100))

```



Linear Regression Modeling

Create custom function to automate linear regression modeling

```

lr_ts_mod <- function(tb_train = NA,
                      tb_test = NA,
                      date = NA,
                      param = NA,
                      ext_pred = c(),
                      train_start = c(),
                      test_start = c(),
                      freq = NA,
                      h = NA,
                      formula = NA) {
  # Function to perform TS linear regression w/ or w/o external variables
  #print(tb_train)
  #print(tb_test)
  tb_full01 <- rbind(tb_train, tb_test)
  #print(tb_full01)

```

```

ts_full01 <- ts(tb_full01[, 2:10],
                  start = train_start,
                  freq = freq)
#print(ts_full01)

train_ts01 <- window(ts_full01, end = c(2021, 5))
test_ts01 <- window(ts_full01, start = c(2021, 6))
#print(tail(train_ts01))
#print(colnames(train_ts01))
#print(colnames(test_ts01))

#print(train_ts01)
#print(test_ts01)
#print(test_ts01[, ext_pred])

# TSLM Citation:
# https://www.rdocumentation.org/packages/forecast/versions/8.19/topics/tslm
lin_mod_fit <- tslm(formula = as.formula(formula),
                     data = train_ts01)

lin_mod_prd <- forecast(lin_mod_fit,
                        h = h,
                        newdata = data.frame(test_ts01[, ext_pred]))


#print(lin_mod_prd)
#print(tb_test[, param])
print(accuracy(lin_mod_prd,
               as.matrix(tb_test[, param])))

# Multiple Object return() Citation:
# https://www.geeksforgeeks.org/return-value-from-r-function/
return(list(train_ts01[, param],
            test_ts01[, param],
            lin_mod_prd))
}

}

```

PLOO: Imputed, All - Including External Variables (Forecasting ENTERO)

```

# PLOO All: Imputed - Training partition (01/18/1999-01/04/2021)
#owt_df02_gb09_ploo_wkly03_train_tb03
# PLOO All: Imputed - Test partition (01/11/2021-01/03/2022)
#owt_df02_gb09_ploo_wkly03_test_tb03

```

```

owt_df02_gb09_ploo_wkly03_train_tb03 %>%
  plot_time_series(date_sample,
                   ENTERO)

```

```

owt_df02_gb09_ploo_wkly03_train_tb03 %>%
  plot_stl_diagnostics(date_sample,
                        ENTERO,
                        .frequency = "auto",
                        .trend = "auto")

```

```

#print(lubridate::isoweek(ymd(train_start)))
#print(lubridate::isoweek(ymd(test_start)))

#print(tail(owt_df02_gb09_ploo_wkly03_train_tb02))
#print(head(owt_df02_gb09_ploo_wkly03_test_tb02))

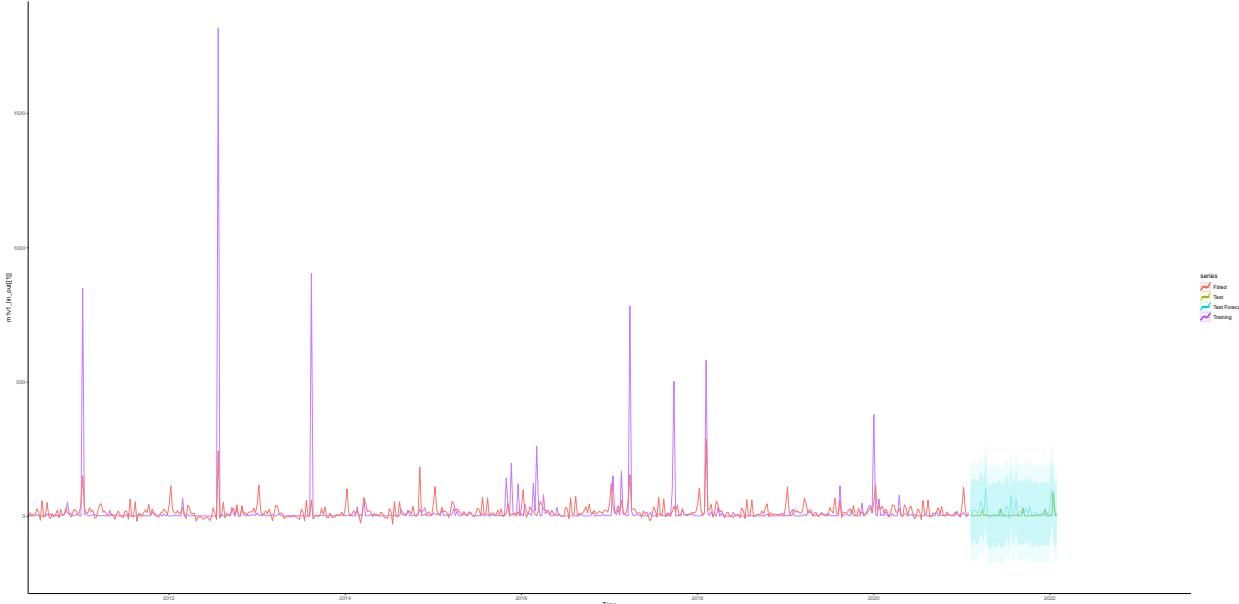
m1v1_lin_out <- lr_ts_mod(tb_train = owt_df02_gb09_ploo_wkly03_train_tb03,
                           tb_test = owt_df02_gb09_ploo_wkly03_test_tb03,
                           date = "date_sample",
                           param = "ENTERO",
                           ext_pred = c("CHLOROPHYLL",
                                       "DENSITY",
                                       "DO",
                                       "FECAL",
                                       "PH",
                                       "SALINITY",
                                       "TEMP",
                                       "XMS"),
                           train_start = c(1999, lubridate::isoweek(ymd(train_start))),
                           test_start = c(2021, 6),
                           freq = 52,
                           h = 52,
                           formula = "ENTERO ~ trend + season + CHLOROPHYLL + DENSITY + DO
                           ← + FECAL + PH + SALINITY + TEMP + XMS")

##               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -1.046722e-15 86.77428 28.51601 -201.2632 394.7822 0.8768111
## Test set      -1.359639e+01 30.01797 19.75175 -520.9396 564.2274 0.6073274
##               ACF1
## Training set 0.00946572
## Test set      NA

#print(m1v1_lin_out[[3]])

# Autoplot Citation:
# USD. (n.d.). ADS 506 Lab 3.2.R [Module 3 course materials].
autoplot(m1v1_lin_out[[1]], series = 'Training') +
  autolayer(m1v1_lin_out[[2]], series = 'Test') +
  autolayer(m1v1_lin_out[[3]], series = 'Test Forecast', alpha = .3) +
  autolayer(m1v1_lin_out[[3]]$fitted, series = 'Fitted') +
  theme_classic() +
  coord_cartesian(#ylim = c(-50, 400),
                 xlim = c(2011, 2023))

```



```
summary(m1v1_lin_out[[3]])
```

```
##
## Forecast method: Linear regression model
##
## Model Information:
##
## Call:
## tslm(formula = as.formula(formula), data = train_ts01)
##
## Coefficients:
## (Intercept)      trend    season2    season3    season4    season5
## 570.24826   0.02487   66.67114  -21.37696  -36.12629  -29.12206
## season6    season7    season8    season9    season10   season11
## -14.04468  -29.91413  -21.67273  -29.07968  -40.74223  -25.73621
## season12   season13   season14   season15   season16   season17
## 11.01291   5.96744  -20.16249  -20.45878  -22.42186  -34.73134
## season18   season19   season20   season21   season22   season23
## -30.49355  -25.85471  -32.51986  -33.84756  -25.75848  -21.82754
## season24   season25   season26   season27   season28   season29
## -35.33516  -38.07910  -35.25317  -6.16428  -15.81276  -46.54864
## season30   season31   season32   season33   season34   season35
## 28.14340  -32.83728  -31.49919  22.09408  -38.20827  -30.32434
## season36   season37   season38   season39   season40   season41
## -25.65753  -30.56741  -23.12911  -10.87565  -27.90870  -0.39450
## season42   season43   season44   season45   season46   season47
## -35.55193  -34.93334  -22.74350  -36.29355  -34.33324  -32.74456
## season48   season49   season50   season51   season52   CHLOROPHYLL
## -29.80809  -28.64711  -23.51158  -24.64886  -32.16327  0.34342
## DENSITY      DO      FECAL        PH      SALINITY     TEMP
## 38.85919   -0.56045    0.42542   54.67497  -60.46816   8.42882
## XMS
## -0.71608
```

```

##
## Error measures:
##               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -1.046722e-15 86.77428 28.51601 -201.2632 394.7822 0.8737683
##                      ACF1
## Training set 0.00946572
##
## Forecasts:
##             Point Forecast      Lo 80      Hi 80      Lo 95      Hi 95
## 2021.096    29.4680279   -87.97340 146.9095 -150.23415 209.1702
## 2021.115     7.4653502  -109.93365 124.8644 -172.17192 187.1026
## 2021.135    24.3485243   -92.95329 141.6503 -155.14003 203.8371
## 2021.154    23.3998700   -93.96530 140.7650 -156.18563 202.9854
## 2021.173     9.8409381  -107.34956 127.0314 -169.47728 189.1592
## 2021.192    21.6933632   -95.48725 138.8740 -157.60973 200.9965
## 2021.212    58.4282465   -58.86386 175.7204 -121.04545 237.9019
## 2021.231    35.9361707   -81.53464 153.4070 -143.81097 215.6833
## 2021.250    21.6577231   -95.54228 138.8577 -157.67505 200.9905
## 2021.269    104.0456494  -13.41867 221.5100  -75.69157 283.7829
## 2021.288     15.1157139  -102.14898 132.3804  -164.31604 194.5475
## 2021.308     -1.9715247  -119.29318 115.3501  -181.49044 177.5474
## 2021.327     0.7637612  -116.60771 118.1352  -178.83137 180.3589
## 2021.346     9.4352817  -107.81352 126.6841  -169.97216 188.8427
## 2021.365     9.8876088  -107.28083 127.0561  -169.39687 189.1721
## 2021.385     3.0450184  -114.14350 120.2335  -176.27018 182.3602
## 2021.404     7.5140680  -109.80647 124.8346  -172.00314 187.0313
## 2021.423     26.0014362  -91.35884 143.3617  -153.57658 205.5795
## 2021.442     -0.7386557  -117.98935 116.5120  -180.14900 178.6717
## 2021.462     3.7781511  -113.36451 120.9208  -175.46688 183.0232
## 2021.481     0.9642838  -116.41144 118.3400  -178.63736 180.5659
## 2021.500     36.3200348  -80.94624 153.5863  -143.11413 215.7542
## 2021.519     34.1802522  -83.25204 151.6125  -145.50796 213.8685
## 2021.538     -6.9186153  -124.21905 110.3818  -186.40507 172.5678
## 2021.558     77.3585967  -40.11327 194.8305  -102.39017 257.1074
## 2021.577     11.4755034  -105.77118 128.7222  -167.92869 190.8797
## 2021.596     13.2590347  -103.93369 130.4518  -166.06260 192.5807
## 2021.615     64.7489345  -52.39421 181.8921  -114.49684 243.9947
## 2021.635     5.0722820  -112.13887 122.2834  -174.27755 184.4221
## 2021.654     18.6293788  -98.63286 135.8916  -160.79861 198.0574
## 2021.673     17.6142618  -99.87302 135.1015  -162.15809 197.3866
## 2021.692     10.3404587  -107.08947 127.7704  -169.34413 190.0250
## 2021.712     20.3485507  -96.85490 137.5520  -158.98949 199.6866
## 2021.731     29.4222818  -87.87739 146.7220  -150.06300 208.9076
## 2021.750     10.2053873  -107.29138 127.7022  -169.58147 189.9922
## 2021.769     39.3936280  -78.09263 156.8799  -140.37715 219.1644
## 2021.788     4.5437551  -112.93016 122.0177  -175.20815 184.2957
## 2021.808     7.0126751  -110.53367 124.5590  -172.85004 186.8754
## 2021.827     17.1021792  -100.48607 134.6904  -162.82466 197.0290
## 2021.846     8.6768013  -108.90284 126.2564  -171.23687 188.5905
## 2021.865     6.4066977  -111.10639 123.9198  -173.40513 186.2185
## 2021.885     6.8703892  -110.40029 124.1411  -172.57053 186.3113
## 2021.904     12.3824905  -105.31637 130.0814  -167.71361 192.4786
## 2021.923     11.9819734  -105.37078 129.3347  -167.58453 191.5485
## 2021.942     5.5885194  -112.27808 123.4551  -174.76424 185.9413

```

```

## 2021.962      13.5630803 -104.07625 131.2024 -166.44193 193.5681
## 2021.981      6.1574358 -111.34330 123.6582 -173.63550 185.9504
## 2022.000      38.5978583 -79.24010 156.4358 -141.71108 218.9068
## 2022.019     101.5686784 -16.19915 219.3365 -78.63295 281.7703
## 2022.038      28.4714562 -89.04633 145.9892 -151.34757 208.2905
## 2022.058       0.8251637 -116.57631 118.2266 -178.81589 180.4662
## 2022.077     12.6666412 -104.72084 130.0541 -166.95300 192.2863

```

PLOO: Imputed, Excluding Outliers - Including External Variables (Forecasting ENTERO)

```

# PLOO Excluding Outliers: Imputed - Training partition (01/18/1999-01/04/2021)
#owt_df02_gb09_ploo_wkly02_train_tb03
# PLOO Excluding Outliers: Imputed - Test partition (01/11/2021-01/03/2022)
#owt_df02_gb09_ploo_wkly02_test_tb03

owt_df02_gb09_ploo_wkly02_train_tb03 %>%
  plot_time_series(date_sample,
                    ENTERO)

owt_df02_gb09_ploo_wkly02_train_tb03 %>%
  plot_stl_diagnostics(date_sample,
                        ENTERO,
                        .frequency = "auto",
                        .trend = "auto")

m2v1_lin_out <- lr_ts_mod(tb_train = owt_df02_gb09_ploo_wkly02_train_tb03,
                            tb_test = owt_df02_gb09_ploo_wkly02_test_tb03,
                            date = "date_sample",
                            param = "ENTERO",
                            ext_pred = c("CHLOROPHYLL",
                                        "DENSITY",
                                        "DO",
                                        "FECAL",
                                        "PH",
                                        "SALINITY",
                                        "TEMP",
                                        "XMS"),
                            train_start = c(1999, lubridate::isoweek(ymd(train_start))),
                            test_start = c(2021, 6),
                            freq = 52,
                            h = 52,
                            formula = "ENTERO ~ trend + season + CHLOROPHYLL + DENSITY + DO
                           + FECAL + PH + SALINITY + TEMP + XMS")

##               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -3.071546e-16 86.77712 28.67443 -201.1468 395.1431 0.8816822
## Test set      -1.247129e+01 29.36381 18.89189 -489.1608 535.6733 0.5808884
##               ACF1
## Training set 0.008521948
## Test set      NA

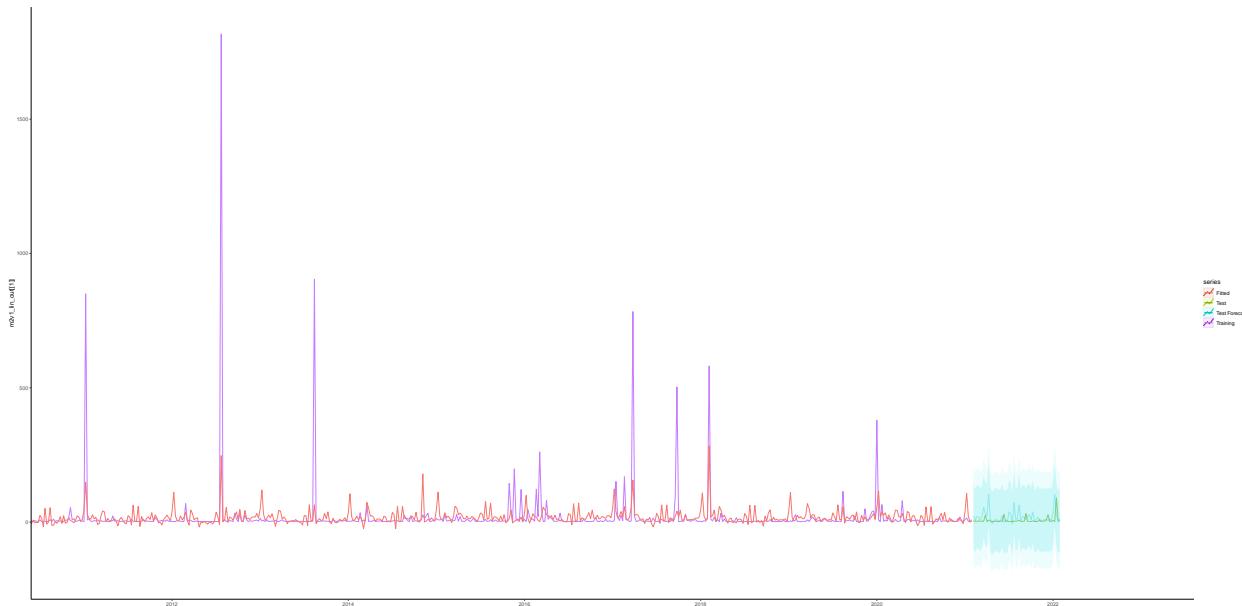
autoplot(m2v1_lin_out[[1]], series = 'Training') +
  autolayer(m2v1_lin_out[[2]], series = 'Test') +
  autolayer(m2v1_lin_out[[3]], series = 'Test Forecast', alpha = .3) +

```

```

autolayer(m2v1_lin_out[[3]]$fitted, series = 'Fitted') +
  theme_classic() +
  coord_cartesian(#ylim = c(-50, 400),
                 xlim = c(2011, 2023))

```



```
summary(m2v1_lin_out[[3]])
```

```

##
## Forecast method: Linear regression model
##
## Model Information:
##
## Call:
## tslm(formula = as.formula(formula), data = train_ts01)
##
## Coefficients:
## (Intercept)      trend    season2    season3    season4    season5
## 566.52070     0.01909   67.86909  -22.10525  -36.58812  -30.60997
## season6    season7    season8    season9    season10   season11
## -15.42820    -30.86519  -23.29226  -30.79680  -41.70029  -26.92888
## season12   season13   season14   season15   season16   season17
## 10.55011     6.55161  -18.54021  -19.75047  -20.62650  -31.81568
## season18   season19   season20   season21   season22   season23
## -27.18784   -22.71148  -30.11554  -30.27887  -23.13177  -18.84428
## season24   season25   season26   season27   season28   season29
## -33.20013   -33.77937  -30.24041  -3.40619   -12.38781  -42.98687
## season30   season31   season32   season33   season34   season35
## 31.31994   -31.11144  -30.20355  23.67099  -36.19962  -24.34365
## season36   season37   season38   season39   season40   season41
## -21.12460   -28.03628  -20.73011  -9.02166  -26.50083   1.68746
## season42   season43   season44   season45   season46   season47
## -34.03457   -33.75832  -20.19064  -34.22012  -32.25841  -31.69945
## season48   season49   season50   season51   season52 CHLOROPHYLL

```

```

## -28.60027 -28.30176 -23.36001 -24.36158 -32.13873 -1.18342
## DENSITY DO FECAL PH SALINITY TEMP
## 41.73314 2.54420 0.42466 56.87757 -63.89572 8.20753
## XMS
## -0.49305
##
##
## Error measures:
## ME RMSE MAE MPE MAPE MASE
## Training set -3.071546e-16 86.77712 28.67443 -201.1468 395.1431 0.8786224
## ACF1
## Training set 0.008521948
##
## Forecasts:
## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95
## 2021.096 24.2918214 -93.17017 141.7538 -155.44182 204.0255
## 2021.115 4.9027823 -112.51926 122.3248 -174.76974 184.5753
## 2021.135 20.3174627 -96.99865 137.6336 -159.19298 199.8279
## 2021.154 20.3769823 -96.99447 137.7484 -159.21812 199.9721
## 2021.173 8.2850284 -108.91208 125.4821 -171.04332 187.6134
## 2021.192 19.5146098 -97.67703 136.7062 -159.80536 198.8346
## 2021.212 55.1373452 -62.16457 172.4393 -124.35136 234.6260
## 2021.231 33.2545388 -84.24426 150.7533 -146.53543 213.0445
## 2021.250 23.4533801 -93.74827 140.6550 -155.88192 202.7887
## 2021.269 103.3003644 -14.16383 220.7646 -76.43666 283.0374
## 2021.288 12.0106577 -105.26060 129.2819 -167.43115 191.4525
## 2021.308 -5.3522353 -122.68154 111.9771 -184.88286 174.1784
## 2021.327 3.6094779 -113.75807 120.9770 -175.97967 183.1986
## 2021.346 9.6756462 -107.59628 126.9476 -169.76718 189.1185
## 2021.365 7.2843518 -109.88922 124.4579 -172.00798 186.5767
## 2021.385 1.9348547 -115.26357 119.1333 -177.39551 181.2652
## 2021.404 7.3168285 -110.00560 124.6393 -172.20326 186.8369
## 2021.423 22.9862070 -94.35294 140.3254 -156.55947 202.5319
## 2021.442 -0.5008495 -117.76090 116.7592 -179.92550 178.9238
## 2021.462 4.6632922 -112.47928 121.8059 -174.58161 183.9082
## 2021.481 2.6358012 -114.75302 120.0246 -176.98589 182.2575
## 2021.500 37.9537846 -79.33810 155.2457 -141.51957 217.4271
## 2021.519 30.1517554 -87.35057 147.6541 -149.64362 209.9471
## 2021.538 -7.4481895 -124.74606 109.8497 -186.93072 172.0343
## 2021.558 72.4441698 -45.04822 189.9366 -107.33600 252.2243
## 2021.577 9.6889927 -107.56416 126.9421 -169.72511 189.1031
## 2021.596 12.5907657 -104.60866 129.7902 -166.74113 191.9227
## 2021.615 64.0293987 -53.11569 181.1745 -115.21934 243.2781
## 2021.635 5.3048771 -111.90548 122.5152 -174.04374 184.6535
## 2021.654 21.4573252 -95.81112 138.7258 -157.98018 200.8948
## 2021.673 18.8427738 -98.65586 136.3414 -160.94695 198.6325
## 2021.692 9.9949824 -107.43631 127.4263 -169.69170 189.6817
## 2021.712 20.8338278 -96.38546 138.0531 -158.52845 200.1961
## 2021.731 28.1233994 -89.17837 145.4252 -151.36509 207.6119
## 2021.750 8.3785499 -109.15580 125.9129 -171.46582 188.2229
## 2021.769 38.1730304 -79.33998 155.6860 -141.63869 217.9848
## 2021.788 3.3254726 -114.17029 120.8212 -176.45986 183.1108
## 2021.808 7.7254556 -109.82480 125.2757 -172.14326 187.5942
## 2021.827 18.1941473 -99.41124 135.7995 -161.75892 198.1472

```

```

## 2021.846      8.4106520 -109.19158 126.0129 -171.53758 188.3589
## 2021.865      5.8309091 -111.71381 123.3756 -174.02932 185.6911
## 2021.885      7.0293792 -110.25205 124.3108 -172.42798 186.4867
## 2021.904      8.6491917 -109.07817 126.3766 -171.49052 188.7889
## 2021.923      9.8730096 -107.50148 127.2475 -169.72675 189.4728
## 2021.942      4.0254122 -113.87745 121.9283 -176.38283 184.4337
## 2021.962      12.9785344 -104.69052 130.6476 -167.07195 193.0290
## 2021.981      4.7061594 -112.85137 122.2637 -175.17368 184.5860
## 2022.000      35.9851031 -81.93069 153.9009 -144.44293 216.4131
## 2022.019      102.0493988 -15.72552 219.8243 -78.16308 282.2619
## 2022.038      26.2551581 -91.28878 143.7991 -153.60388 206.1142
## 2022.058      -0.7352182 -118.14393 116.6735 -180.38735 178.9169
## 2022.077      11.5191875 -105.89270 128.9311 -168.13780 191.1762

```

SBOO: Imputed, All - Including External Variables (Forecasting ENTERO)

```

# SBOO All: Imputed - Training partition (01/18/1999-01/04/2021)
#owt_df02_gb09_sboo_wkly03_train_tb03
# SBOO All: Imputed - Test partition (01/11/2021-01/03/2022)
#owt_df02_gb09_sboo_wkly03_test_tb03

owt_df02_gb09_sboo_wkly03_train_tb03 %>%
  plot_time_series(date_sample,
                    ENTERO)

owt_df02_gb09_sboo_wkly03_train_tb03 %>%
  plot_stl_diagnostics(date_sample,
                        ENTERO,
                        .frequency = "auto",
                        .trend = "auto")

m3v1_lin_out <- lr_ts_mod(tb_train = owt_df02_gb09_sboo_wkly03_train_tb03,
                           tb_test = owt_df02_gb09_sboo_wkly03_test_tb03,
                           date = "date_sample",
                           param = "ENTERO",
                           ext_pred = c("CHLOROPHYLL",
                                       "DENSITY",
                                       "DO",
                                       "FECAL",
                                       "PH",
                                       "SALINITY",
                                       "TEMP",
                                       "XMS"),
                           train_start = c(1999, lubridate::isoweek(ymd(train_start))),
                           test_start = c(2021, 6),
                           freq = 52,
                           h = 52,
                           formula = "ENTERO ~ trend + season + CHLOROPHYLL + DENSITY + DO
                           + FECAL + PH + SALINITY + TEMP + XMS")

##               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -2.998788e-15 506.8252 220.4953 -3.930906 1165.3447 0.5455759
## Test set     -6.012224e+01 558.4625 327.0337 -907.847612 970.4077 0.8091859
##                      ACF1

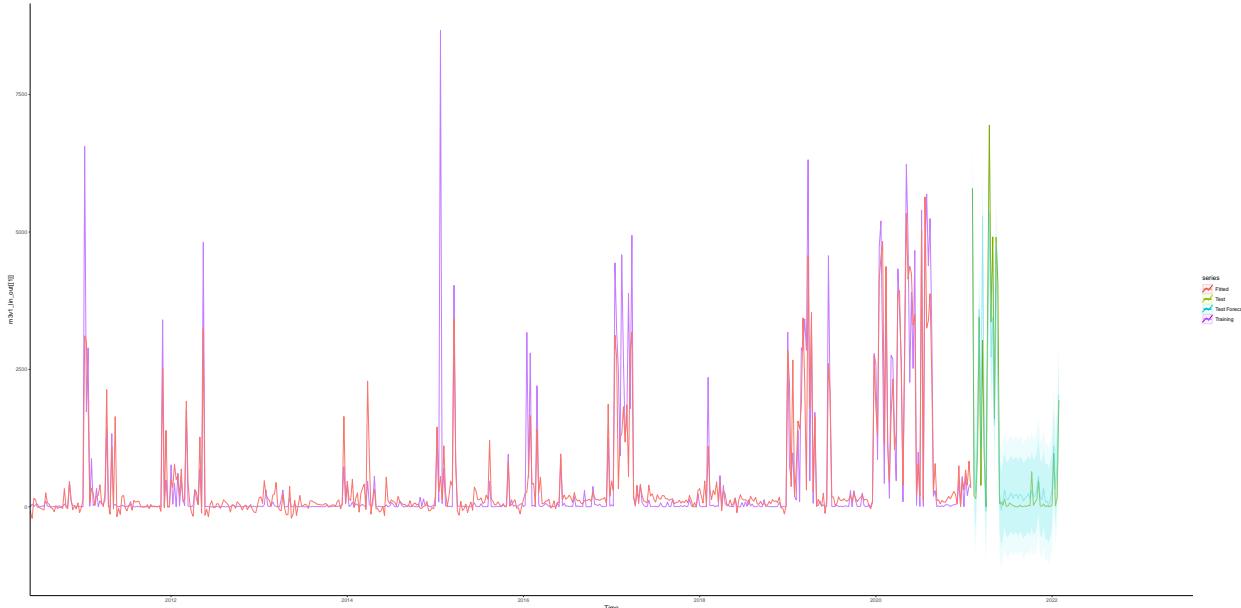
```

```

## Training set 0.009563594
## Test set NA

autoplot(m3v1_lin_out[[1]], series = 'Training') +
  autolayer(m3v1_lin_out[[2]], series = 'Test') +
  autolayer(m3v1_lin_out[[3]], series = 'Test Forecast', alpha = .3) +
  autolayer(m3v1_lin_out[[3]]$fitted, series = 'Fitted') +
  theme_classic() +
  coord_cartesian(#ylim = c(-50, 400),
                 xlim = c(2011, 2023))

```



```
summary(m3v1_lin_out[[3]])
```

```

##
## Forecast method: Linear regression model
##
## Model Information:
##
## Call:
## tslm(formula = as.formula(formula), data = train_ts01)
##
## Coefficients:
## (Intercept)      trend    season2    season3    season4    season5
##  9968.5071     0.1491   -32.6756   -128.9393   236.0108  -204.9225
##  season6    season7    season8    season9    season10   season11
##  -98.4207    -19.3807  -302.6474   -56.2837  -27.8692  179.9122
##  season12   season13   season14   season15   season16   season17
##  -244.5132   -253.5372  -389.2049  -365.1314  -198.7332 -325.1748
##  season18   season19   season20   season21   season22   season23
##  -310.9846   -221.5305  -341.7518  -286.6543  -181.6924 -381.3937
##  season24   season25   season26   season27   season28   season29
##  -131.5418   -59.3853  -165.0153  -144.7574  -123.4365 -145.0994
##  season30   season31   season32   season33   season34   season35
##  -171.8793   -65.1013  -102.5399  -109.7631  -112.5281 -156.7857

```

```

##    season36      season37      season38      season39      season40      season41
## -176.3867     -150.0919     -154.8852     -164.6777     -167.0776     -179.5270
##    season42      season43      season44      season45      season46      season47
## -190.8826     -172.5472     -169.8456     -176.9456     -201.5565     -173.0299
##    season48      season49      season50      season51      season52      CHLOROPHYLL
## -137.7666     -222.9988     -255.6710     -297.5341     -182.6593      -7.9164
##    DENSITY       DO        FECAL        PH        SALINITY      TEMP
## -65.6198      8.2910      0.7018     -228.9458     -168.0713     -14.6016
##    XMS
## -7.4523
##
##
## Error measures:
##               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -2.998788e-15 506.8252 220.4953 -3.930906 1165.345 0.4499963
##                      ACF1
## Training set 0.009563594
##
## Forecasts:
##      Point Forecast      Lo 80      Hi 80      Lo 95      Hi 95
## 2021.096      5801.68275 5103.12390 6500.2416 4732.78777 6870.5777
## 2021.115      613.66951   -71.35680 1298.6958 -434.51874 1661.8578
## 2021.135      256.39699   -428.24078 941.0348 -791.19674 1303.9907
## 2021.154      1434.03951   748.57957 2119.4994 385.18774 2482.8913
## 2021.173      3592.83105 2905.49836 4280.1637 2541.11370 4644.5484
## 2021.192      1034.78148  350.14153 1719.4214 -12.81559 2082.3785
## 2021.212      5293.06731 4598.69367 5987.4409 4230.57631 6355.5583
## 2021.231      1120.00407 433.92651 1806.0816 70.20725 2169.8009
## 2021.250      -76.65877  -761.19533 607.8778 -1124.09763 970.7801
## 2021.269      2982.52536 2294.68036 3670.3704 1930.02410 4035.0266
## 2021.288      5358.63325 4657.35806 6059.9084 4285.58189 6431.6846
## 2021.308      2725.02712 2036.80358 3413.2507 1671.94664 3778.1076
## 2021.327      3443.09651 2750.45638 4135.7366 2383.25802 4502.9350
## 2021.346      1489.33880  804.41493 2174.2627 441.30730 2537.3703
## 2021.365      4740.92494 4048.75430 5433.0956 3681.80484 5800.0450
## 2021.385      2283.04579 1596.79524 2969.2963 1232.98427 3333.1073
## 2021.404      233.97172 -452.64330 920.5867 -816.64748 1284.5909
## 2021.423      -54.75073 -739.72895 630.2275 -1102.86540 993.3639
## 2021.442      121.23137 -567.43501 809.8977 -932.52671 1174.9894
## 2021.462      313.65886 -372.43727 999.7550 -736.16638 1363.4841
## 2021.481      149.02770 -536.13529 834.1907 -899.36969 1197.4251
## 2021.500      130.79897 -553.95266 815.5506 -916.96899 1178.5669
## 2021.519      203.90110 -485.35152 893.1537 -850.75401 1258.5562
## 2021.538      247.58318 -438.40845 933.5748 -802.08214 1297.2485
## 2021.558      155.68109 -529.57563 840.9378 -892.85972 1204.2219
## 2021.577      214.30489 -470.54810 899.1579 -833.61817 1262.2280
## 2021.596      225.17207 -459.80489 910.1490 -822.94067 1273.2848
## 2021.615      155.03140 -529.94171 840.0045 -893.07544 1203.1382
## 2021.635      228.01771 -456.80787 912.8433 -819.86340 1275.8988
## 2021.654      208.03571 -476.57932 892.6507 -839.52323 1255.5947
## 2021.673      107.93657 -578.41147 794.2846 -942.27411 1158.1473
## 2021.692      156.67617 -528.54058 841.8929 -891.80349 1205.1558
## 2021.712      184.43735 -500.67401 869.5487 -863.88104 1232.7557
## 2021.731      246.17742 -438.61660 930.9714 -801.65540 1294.0103

```

```

## 2021.750    156.81105 -528.81650  842.4386  -892.29718 1205.9193
## 2021.769    309.22242 -376.15847  994.6033  -739.50840 1357.9532
## 2021.788    178.40081 -508.88625  865.6879  -873.24672 1230.0483
## 2021.808    211.59358 -474.81162  897.9988  -838.70457 1261.8917
## 2021.827    312.78000 -376.39401 1001.9540  -741.75483 1367.3148
## 2021.846    557.09652 -128.97868 1243.1717  -492.69668 1606.8897
## 2021.865    176.79566 -510.48585  864.0772  -874.84337 1228.4347
## 2021.885    129.77258 -555.07072  814.6159  -918.13564 1177.6808
## 2021.904    341.93195 -345.92050 1029.7844  -710.58071 1394.4446
## 2021.923     94.51956 -594.07779  783.1169  -959.13289 1148.1720
## 2021.942    104.48815 -583.26606  792.2424  -947.87419 1156.8505
## 2021.962     21.49827 -666.41098  709.4075  -1031.10129 1074.0978
## 2021.981    139.22373 -548.18872  826.6362  -912.61566 1191.0631
## 2022.000    309.36129 -378.15401  996.8766  -742.63548 1361.3581
## 2022.019   1100.82360  411.58417 1790.0630    46.18867 2155.4585
## 2022.038    334.23006 -352.43866 1020.8988  -716.47132 1384.9314
## 2022.058     775.02158   87.61685 1462.4263  -276.80600 1826.8492
## 2022.077   2041.32067 1354.89344 2727.7479   990.98880 3091.6525

```

SBOO: Imputed, Excluding Outliers - Including External Variables (Forecasting ENTERO)

```

# SBOO Excluding Outliers: Imputed - Training partition (01/18/1999-01/04/2021)
#owt_df02_gb09_sboo_wkly02_train_tb03
# SBOO Excluding Outliers: Imputed - Test partition (01/11/2021-01/03/2022)
#owt_df02_gb09_sboo_wkly02_test_tb03

owt_df02_gb09_sboo_wkly02_train_tb03 %>%
  plot_time_series(date_sample,
                    ENTERO)

owt_df02_gb09_sboo_wkly02_train_tb03 %>%
  plot_stl_diagnostics(date_sample,
                        ENTERO,
                        .frequency = "auto",
                        .trend = "auto")

m4v1_lin_out <- lr_ts_mod(tb_train = owt_df02_gb09_sboo_wkly02_train_tb03,
                            tb_test = owt_df02_gb09_sboo_wkly02_test_tb03,
                            date = "date_sample",
                            param = "ENTERO",
                            ext_pred = c("CHLOROPHYLL",
                                         "DENSITY",
                                         "DO",
                                         "FECAL",
                                         "PH",
                                         "SALINITY",
                                         "TEMP",
                                         "XMS"),
                            train_start = c(1999, lubridate::isoweek(ymd(train_start))),
                            test_start = c(2021, 6),
                            freq = 52,
                            h = 52,
                            formula = "ENTERO ~ trend + season + CHLOROPHYLL + DENSITY + DO
                                         + FECAL + PH + SALINITY + TEMP + XMS")

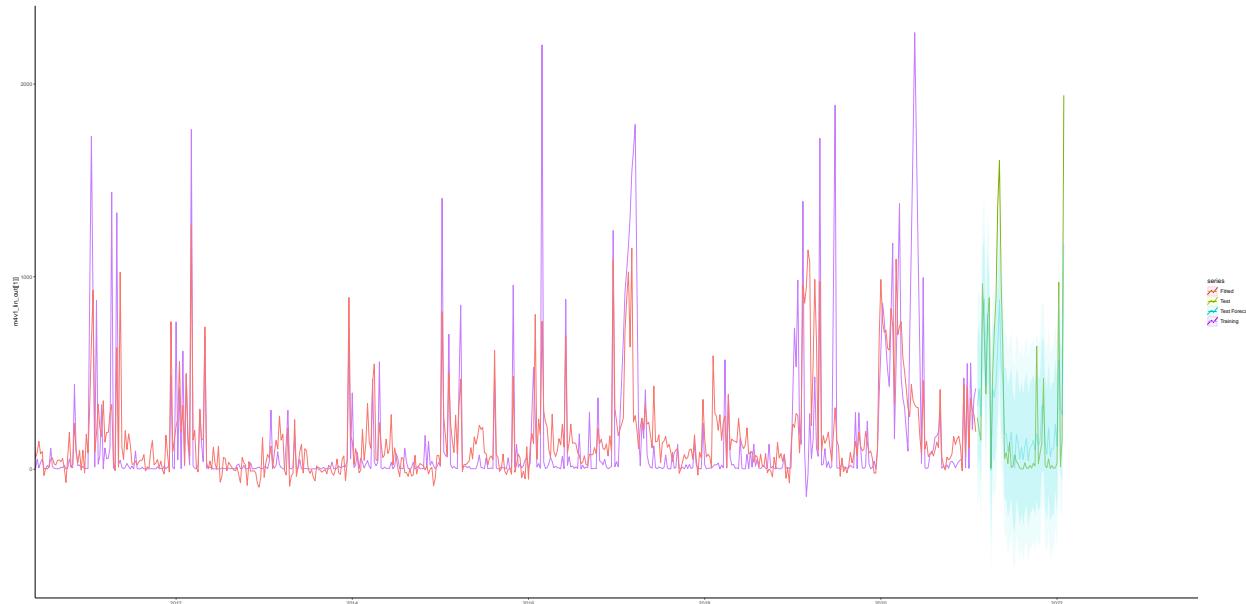
```

```

##               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set 1.257115e-14 252.9268 131.7942 -416.3427 828.0949 0.8154878
## Test set     2.250579e+01 249.7221 176.5312 -573.1820 603.3266 1.0923018
##             ACF1
## Training set 0.3185982
## Test set     NA

autoplot(m4v1_lin_out[[1]], series = 'Training') +
  autolayer(m4v1_lin_out[[2]], series = 'Test') +
  autolayer(m4v1_lin_out[[3]], series = 'Test Forecast', alpha = .3) +
  autolayer(m4v1_lin_out[[3]]$fitted, series = 'Fitted') +
  theme_classic() +
  coord_cartesian(#ylim = c(-50, 400),
                 xlim = c(2011, 2023))

```



```
summary(m4v1_lin_out[[3]])
```

```

##
## Forecast method: Linear regression model
##
## Model Information:
##
## Call:
## tslm(formula = as.formula(formula), data = train_ts01)
##
## Coefficients:
## (Intercept)      trend      season2      season3      season4      season5
##   1.132e+04    6.964e-02    2.816e+01    7.040e+01    2.186e+01   -4.172e+01
##   season6      season7      season8      season9      season10     season11
##   -6.058e+01   1.317e+02    2.653e+01    3.840e+01    2.094e+02    6.160e+01
##   season12     season13     season14     season15     season16     season17
##   9.932e+01   -3.406e+01   -1.130e+02   -3.840e+00   -9.550e+01   -5.590e+01
##   season18     season19     season20     season21     season22     season23

```

```

## -4.587e+01 -2.817e+00 -2.644e+01 4.028e+01 2.584e+01 4.764e+01
## season24 season25 season26 season27 season28 season29
## 7.138e-01 5.009e+00 6.930e+01 -3.577e+01 -5.507e+01 -2.167e+01
## season30 season31 season32 season33 season34 season35
## -1.513e+01 -5.060e+01 -3.650e+01 -3.183e+01 -2.637e+01 -3.140e+01
## season36 season37 season38 season39 season40 season41
## -4.765e+01 -2.689e+01 -3.760e+01 -4.059e+01 -4.512e+01 -5.418e+01
## season42 season43 season44 season45 season46 season47
## -4.104e+01 -4.196e+01 1.677e+01 -4.208e+01 -5.494e+01 -2.804e+01
## season48 season49 season50 season51 season52 CHLOROPHYLL
## -4.030e+01 -6.690e+01 -6.090e+01 -4.679e+01 1.070e+02 1.390e+00
## DENSITY DO FECAL PH SALINITY TEMP
## -1.081e+02 2.303e+01 3.628e-01 -3.638e+02 -1.492e+02 -1.722e+01
## XMS
## -7.965e+00
##
##
## Error measures:
## ME RMSE MAE MPE MAPE MASE
## Training set 1.257115e-14 252.9268 131.7942 -416.3427 828.0949 0.6621506
## ACF1
## Training set 0.3185982
##
## Forecasts:
## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95
## 2021.096 328.579212 -13.85744 671.0159 -195.39785 852.5563
## 2021.115 415.817329 73.91323 757.7214 -107.34485 938.9795
## 2021.135 276.622148 -64.96889 618.2132 -246.06101 799.3053
## 2021.154 782.226753 439.19071 1125.2628 257.33253 1307.1210
## 2021.173 884.389967 542.47802 1226.3019 361.21578 1407.5642
## 2021.192 433.971340 92.07520 775.8675 -89.17866 957.1213
## 2021.212 804.613955 462.00034 1147.2276 280.36612 1328.8618
## 2021.231 655.836992 313.02120 998.6528 131.27978 1180.3942
## 2021.250 -5.280339 -346.98084 336.4202 -528.13100 517.5703
## 2021.269 415.180072 72.89840 757.4617 -108.55986 938.9200
## 2021.288 374.550579 30.02199 719.0792 -152.62746 901.7286
## 2021.308 444.501128 102.17003 786.8322 -79.31442 968.3167
## 2021.327 643.969930 299.39761 988.5422 116.72499 1171.2149
## 2021.346 879.742355 535.65679 1223.8279 353.24222 1406.2425
## 2021.365 541.970641 199.85146 884.0898 18.47935 1065.4619
## 2021.385 472.606250 130.54437 814.6681 -50.79736 996.0099
## 2021.404 187.129094 -155.37969 529.6379 -336.95834 711.2165
## 2021.423 184.083771 -157.64121 525.8087 -338.80433 706.9719
## 2021.442 91.432632 -252.46972 435.3350 -434.78716 617.6524
## 2021.462 134.631652 -207.85191 477.1152 -389.41720 658.6805
## 2021.481 193.570829 -148.29371 535.4354 -329.53083 716.6725
## 2021.500 56.536691 -285.10934 398.1827 -466.23061 579.3040
## 2021.519 6.176224 -336.00992 348.3624 -517.41753 529.7700
## 2021.538 182.614613 -159.71561 524.9448 -341.19960 706.4288
## 2021.558 137.164301 -204.80048 479.1291 -386.09074 660.4193
## 2021.577 53.421233 -288.32964 395.1721 -469.50649 576.3490
## 2021.596 114.920179 -226.88117 456.7215 -408.08479 637.9251
## 2021.615 48.840847 -292.98721 390.6689 -474.20499 571.8867
## 2021.635 119.293456 -222.43518 461.0221 -403.60024 642.1872

```

```

## 2021.654    159.778854 -182.02297  501.5807 -363.22683  682.7845
## 2021.673    44.292194 -298.21089  386.7953 -479.78653  568.3709
## 2021.692    110.331331 -231.60614  452.2688 -412.88192  633.5446
## 2021.712    124.644389 -217.30221  466.5910 -398.58282  647.8716
## 2021.731    149.272854 -192.45454  491.0002 -373.61894  672.1647
## 2021.750    101.894205 -240.24754  444.0360 -421.63162  625.4200
## 2021.769    184.361732 -157.79508  526.5185 -339.18713  707.9106
## 2021.788    124.461260 -218.50474  467.4273 -400.32579  649.2483
## 2021.808    148.370544 -194.16584  490.9069 -375.75913  672.5002
## 2021.827    327.842739 -16.32666  672.0121 -198.78568  854.4712
## 2021.846    336.676912 -6.02420  679.3780 -187.70482  861.0586
## 2021.865    137.021702 -205.81632  479.8597 -387.56952  661.6129
## 2021.885    79.113959 -262.68863  420.9165 -443.89289  602.1208
## 2021.904    215.443377 -127.77561  558.6624 -309.73078  740.6175
## 2021.923    66.569991 -277.16863  410.3086 -459.39928  592.5393
## 2021.942    106.341905 -236.92140  449.6052 -418.90006  631.5839
## 2021.962    107.611363 -235.86000  451.0827 -417.94897  633.1717
## 2021.981    234.481737 -108.66648  577.6300 -290.58413  759.5476
## 2022.000    120.963269 -222.40221  464.3287 -404.43503  646.3616
## 2022.019    565.273282  220.50373  910.0428   37.72654 1092.8200
## 2022.038    318.880581 -23.60629  661.3675 -205.17333  842.9345
## 2022.058    282.790080 -60.20735  625.7875 -242.04505  807.6252
## 2022.077    1166.493195  820.68563 1512.3008   637.35815 1695.6282

```

PLOO: Imputed, All - Excluding External Variables (Forecasting ENTERO)

```

# PLOO All: Imputed - Training partition (01/18/1999-01/04/2021)
#owt_df02_gb09_ploo_wkly03_train_tb03
# PLOO All: Imputed - Test partition (01/11/2021-01/03/2022)
#owt_df02_gb09_ploo_wkly03_test_tb03

owt_df02_gb09_ploo_wkly03_train_tb03 %>%
  plot_time_series(date_sample,
                    ENTERO)

owt_df02_gb09_ploo_wkly03_train_tb03 %>%
  plot_stl_diagnostics(date_sample,
                        ENTERO,
                        .frequency = "auto",
                        .trend = "auto")

m1v2_lin_out <- lr_ts_mod(tb_train = owt_df02_gb09_ploo_wkly03_train_tb03,
                            tb_test = owt_df02_gb09_ploo_wkly03_test_tb03,
                            date = "date_sample",
                            param = "ENTERO",
                            ext_pred = c(),
                            train_start = c(1999, lubridate::isoweek(ymd(train_start))),
                            test_start = c(2021, 6),
                            freq = 52,
                            h = 52,
                            formula = "ENTERO ~ trend + season")

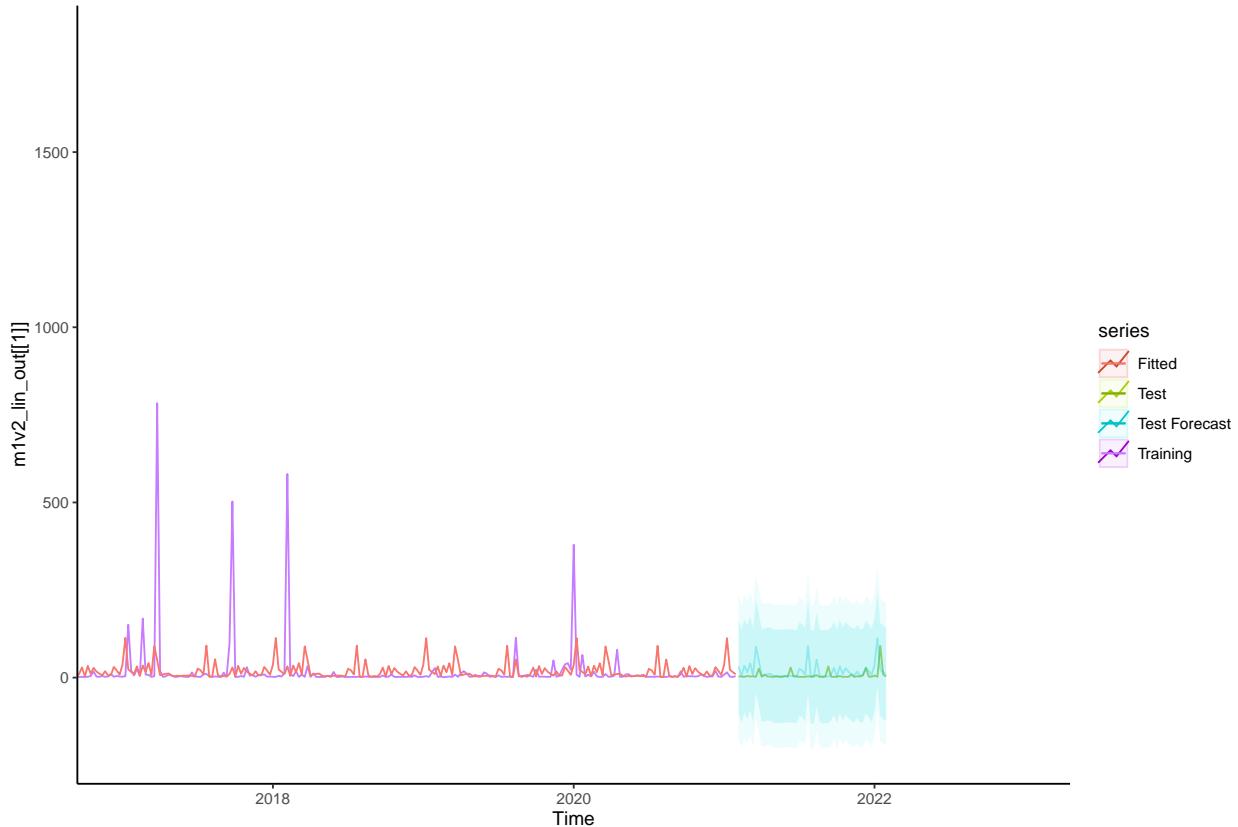
```

```

##               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set 3.739460e-16 98.91472 27.58437 -378.3600 400.8285 0.8481648
## Test set     -1.168561e+01 29.06294 16.71202 -490.5666 509.6556 0.5138615
##             ACF1
## Training set 0.02162603
## Test set      NA

autoplot(m1v2_lin_out[[1]], series = 'Training') +
  autolayer(m1v2_lin_out[[2]], series = 'Test') +
  autolayer(m1v2_lin_out[[3]], series = 'Test Forecast', alpha = .3) +
  autolayer(m1v2_lin_out[[3]]$fitted, series = 'Fitted') +
  coord_cartesian(#ylim = c(-50, 400),
                  xlim = c(2017, 2023)) +
  theme_classic()

```



```
summary(m1v2_lin_out[[3]])
```

```

##
## Forecast method: Linear regression model
##
## Model Information:
##
## Call:
## tslm(formula = as.formula(formula), data = train_ts01)
##
```

```

## Coefficients:
## (Intercept)      trend    season2    season3    season4    season5
## 40.822117     -0.004268   76.649781  -13.521074  -20.219680  -26.377586
## season6       season7    season8    season9    season10   season11
## -4.590507     -31.615742   -2.210463  -20.911339   4.748728  -28.893819
## season12      season13    season14    season15    season16   season17
## 52.622820     13.151594  -29.851194  -27.604926  -25.836072  -25.193009
## season18      season19    season20    season21    season22   season23
## -31.013795    -32.841052  -31.293345  -32.527051  -31.102873  -31.150050
## season24      season25    season26    season27    season28   season29
## -30.420647    -31.575550  -33.733539  -10.931432  -15.780782  -27.699048
## season30      season31    season32    season33    season34   season35
## 54.732247     -34.243870  -34.401081   15.705067  -29.491730  -34.066359
## season36      season37    season38    season39    season40   season41
## -31.631422    -33.608599  -26.303982  -7.834926  -33.407788  -3.131881
## season42      season43    season44    season45    season46   season47
## -25.860403    -9.119190  -19.416854  -25.557574  -30.488701  -18.444005
## season48      season49    season50    season51    season52
## -30.604981    -29.867781  -6.184421  -16.420304  -28.345843
##
##
## Error measures:
##               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set 3.73946e-16 98.91472 27.58437 -378.36 400.8285 0.8452214
## ACF1
## Training set 0.02162603
##
## Forecasts:
##           Point Forecast      Lo 80      Hi 80      Lo 95      Hi 95
## 2021.096    31.331803 -101.64498 164.3086 -172.14094 234.8045
## 2021.115    4.302301 -128.67448 137.2791 -199.17044 207.7750
## 2021.135    33.703312 -99.27347 166.6801 -169.76943 237.1761
## 2021.154    14.998167 -117.97861 147.9749 -188.47458 218.4709
## 2021.173    40.653966 -92.32282 173.6307 -162.81878 244.1267
## 2021.192    7.007151 -125.96963 139.9839 -196.46559 210.4799
## 2021.212    88.519522 -44.45726 221.4963 -114.95322 291.9923
## 2021.231    49.044027 -83.93275 182.0208 -154.42872 252.5168
## 2021.250    6.036971 -126.93981 139.0138 -197.43577 209.5097
## 2021.269    8.278971 -124.69781 141.2558 -195.19377 211.7517
## 2021.288    10.043557 -122.93322 143.0203 -193.42919 213.5163
## 2021.308    10.682352 -122.29443 143.6591 -192.79039 214.1551
## 2021.327    4.857298 -128.11948 137.8341 -198.61545 208.3300
## 2021.346    3.025773 -129.95101 136.0026 -200.44697 206.4985
## 2021.365    4.569212 -128.40757 137.5460 -198.90353 208.0420
## 2021.385    3.331238 -129.64554 136.3080 -200.14151 206.8040
## 2021.404    4.751147 -128.22563 137.7279 -198.72160 208.2239
## 2021.423    4.699702 -128.27708 137.6765 -198.77304 208.1724
## 2021.442    5.424837 -127.55194 138.4016 -198.04791 208.8976
## 2021.462    4.265666 -128.71112 137.2424 -199.20708 207.7384
## 2021.481    2.103409 -130.87337 135.0802 -201.36934 205.5762
## 2021.500    24.901248 -108.07553 157.8780 -178.57150 228.3740
## 2021.519    20.047630 -112.92915 153.0244 -183.42512 223.5204
## 2021.538    8.125095 -124.85169 141.1019 -195.34765 211.5978
## 2021.558    90.552122 -42.42466 223.5289 -112.92062 294.0249

```

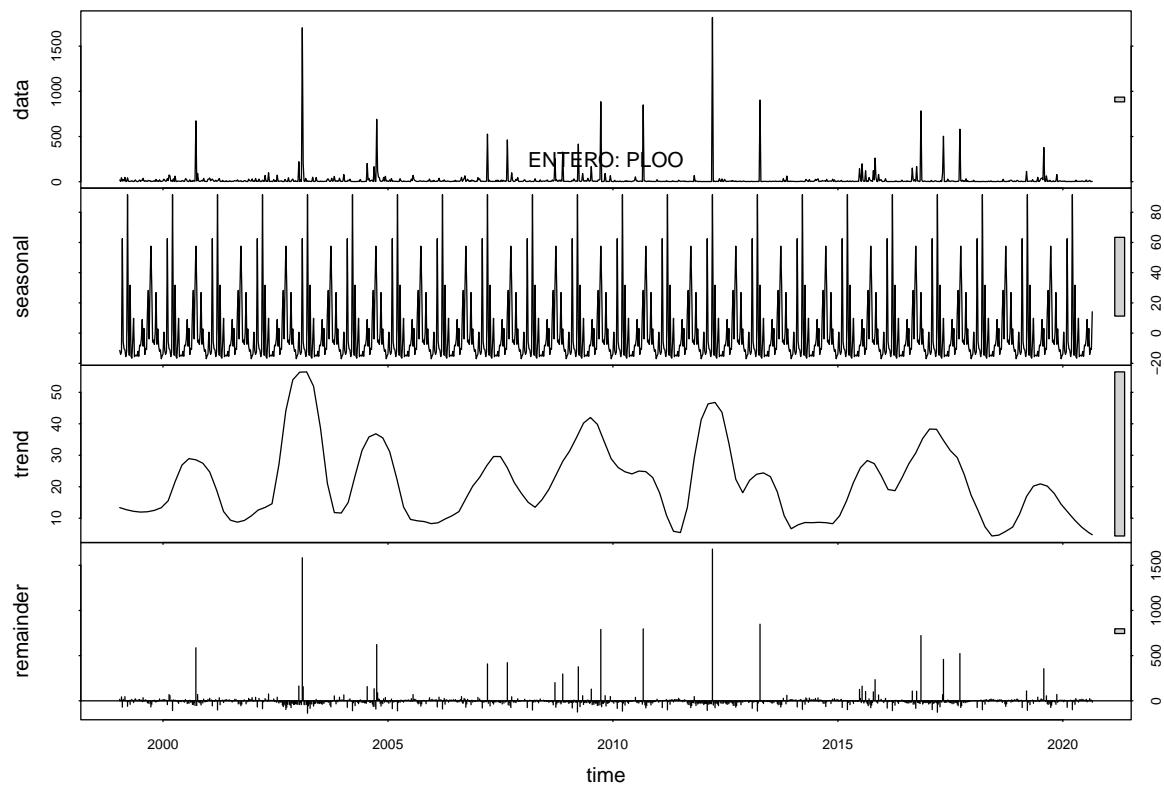
```

## 2021.577      1.571738 -131.40504 134.5485 -201.90101 205.0445
## 2021.596      1.410258 -131.56652 134.3870 -202.06249 204.8830
## 2021.615      51.512138 -81.46464 184.4889 -151.96061 254.9849
## 2021.635      6.311073 -126.66571 139.2879 -197.16167 209.7838
## 2021.654      1.732176 -131.24461 134.7090 -201.74057 205.2049
## 2021.673      4.162844 -128.81394 137.1396 -199.30990 207.6356
## 2021.692      2.181399 -130.79538 135.1582 -201.29135 205.6541
## 2021.712      9.481749 -123.49503 142.4585 -193.99100 212.9545
## 2021.731      27.946536 -105.03025 160.9233 -175.52621 231.4193
## 2021.750      2.369407 -130.60737 135.3462 -201.10334 205.8422
## 2021.769      32.641045 -100.33574 165.6178 -170.83170 236.1138
## 2021.788      9.908255 -123.06853 142.8850 -193.56449 213.3810
## 2021.808      26.645200 -106.33158 159.6220 -176.82754 230.1179
## 2021.827      16.343268 -116.63351 149.3200 -187.12948 219.8160
## 2021.846      10.198280 -122.77850 143.1751 -193.27447 213.6710
## 2021.865      5.262885 -127.71390 138.2397 -198.20986 208.7356
## 2021.885      17.303312 -115.67347 150.2801 -186.16943 220.7761
## 2021.904      5.138069 -127.83871 138.1149 -198.33468 208.6108
## 2021.923      5.871000 -127.10578 138.8478 -197.60174 209.3437
## 2021.942      29.550092 -103.42669 162.5269 -173.92265 233.0228
## 2021.962      19.309940 -113.66684 152.2867 -184.16280 222.7827
## 2021.981      7.380134 -125.59665 140.3569 -196.09261 210.8529
## 2022.000      35.721709 -97.25507 168.6985 -167.75104 239.1945
## 2022.019      112.367221 -20.60956 245.3440 -91.10552 315.8400
## 2022.038      22.192098 -110.67535 155.0595 -181.11335 225.4975
## 2022.058      15.489225 -117.37822 148.3567 -187.81622 218.7947
## 2022.077      9.327051 -123.54039 142.1945 -193.97839 212.6325

ts_full02 <- ts(na.remove(owt_df02_gb09_ploo_wkly03_train_tb02$ENTERO),
                 start = c(1999, lubridate::isoweek(ymd(train_start))),
                 freq = 52)

stl_ent02 <- stl(ts_full02, s.window="periodic")
plot(stl_ent02)
mtext("ENTERO: PLOO", side=3, line = -5)

```

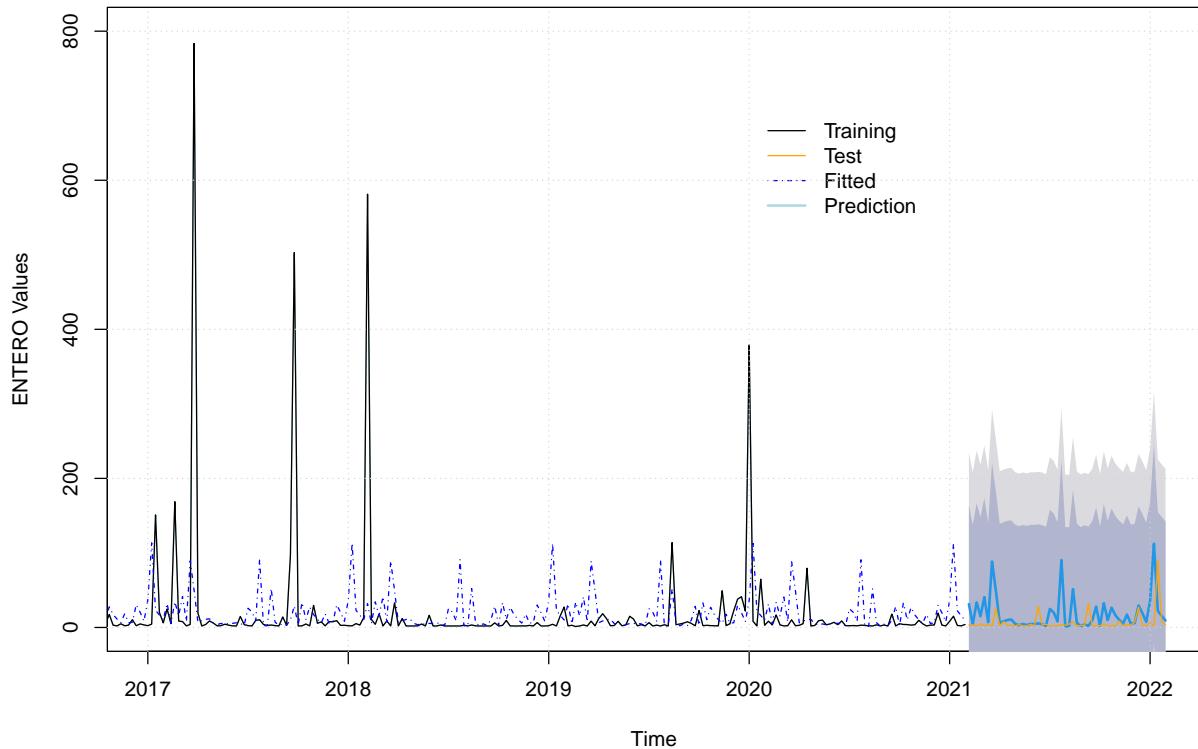


```

plot(m1v2_lin_out[[3]],
      xlim = c(2017, 2022.05),
      ylim = c(0, 800),
      xlab = "Time",
      ylab = "ENTERO Values",
      main = "Predicted ENTERO Levels Over Time - PLOO",
      col = "light blue")
lines(m1v2_lin_out[[1]], lty=1, col="black")
lines(m1v2_lin_out[[2]], lty=1, col="orange")
lines(m1v2_lin_out[[3]]$fitted, lty=4, col="blue")
legend(2020, 700,
      c("Training", "Test", "Fitted", "Prediction"),
      lty = c(1,1,4,1),
      lwd = c(1,1,1,2),
      col = c("black", "orange", "blue", "light blue"),
      bty = "n")
grid()

```

Predicted ENTERO Levels Over Time – PLOO



PLOO: Imputed, Excluding Outliers - Excluding External Variables (Forecasting ENTERO)

```
# PLOO Excluding Outliers: Imputed - Training partition (01/18/1999-01/04/2021)
#owt_df02_gb09_ploo_wkly02_train_tb03
# PLOO Excluding Outliers: Imputed - Test partition (01/11/2021-01/03/2022)
#owt_df02_gb09_ploo_wkly02_test_tb03

owt_df02_gb09_ploo_wkly02_train_tb03 %>%
  plot_time_series(date_sample,
    ENTERO)

owt_df02_gb09_ploo_wkly02_train_tb03 %>%
  plot_stl_diagnostics(date_sample,
    ENTERO,
    .frequency = "auto",
    .trend = "auto")

m2v2_lin_out <- lr_ts_mod(tb_train = owt_df02_gb09_ploo_wkly02_train_tb03,
  tb_test = owt_df02_gb09_ploo_wkly02_test_tb03,
  date = "date_sample",
  param = "ENTERO",
  ext_pred = c(),
  train_start = c(1999, lubridate::isoweek(ymd(train_start))),
  test_start = c(2021, 6),
  freq = 52,
```

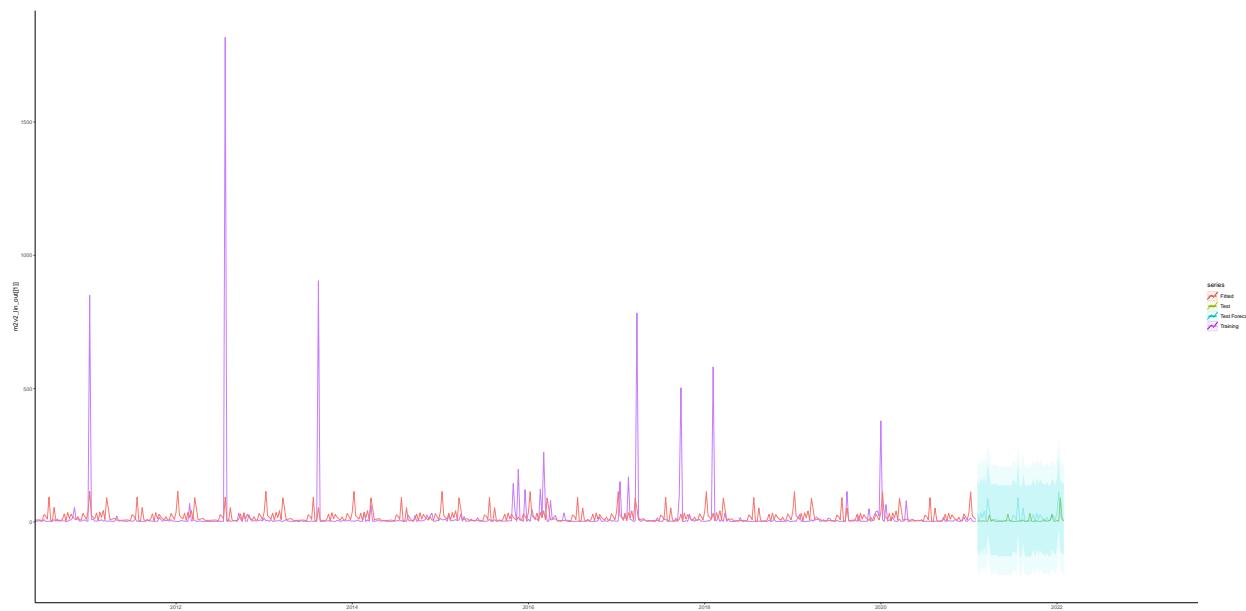
```

          h = 52,
          formula = "ENTERO ~ trend + season")

##               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set 3.739460e-16 98.91472 27.58437 -378.3600 400.8285 0.8481648
## Test set     -1.168561e+01 29.06294 16.71202 -490.5666 509.6556 0.5138615
## ACF1
## Training set 0.02162603
## Test set      NA

autoplot(m2v2_lin_out[[1]], series = 'Training') +
  autolayer(m2v2_lin_out[[2]], series = 'Test') +
  autolayer(m2v2_lin_out[[3]], series = 'Test Forecast', alpha = .3) +
  autolayer(m2v2_lin_out[[3]]$fitted, series = 'Fitted') +
  theme_classic() +
  coord_cartesian(#ylim = c(-50, 400),
                 xlim = c(2011, 2023))

```



```
summary(m2v2_lin_out[[3]])
```

```

##
## Forecast method: Linear regression model
##
## Model Information:
##
## Call:
## tslm(formula = as.formula(formula), data = train_ts01)
##
## Coefficients:
## (Intercept)      trend      season2      season3      season4      season5
## 40.822117    -0.004268    76.649781   -13.521074   -20.219680   -26.377586
## season6      season7      season8      season9      season10     season11
## -4.590507   -31.615742   -2.210463   -20.911339    4.748728   -28.893819
## season12     season13     season14     season15     season16     season17

```

```

##   52.622820  13.151594 -29.851194 -27.604926 -25.836072 -25.193009
## season18    season19    season20    season21    season22    season23
## -31.013795 -32.841052 -31.293345 -32.527051 -31.102873 -31.150050
## season24    season25    season26    season27    season28    season29
## -30.420647 -31.575550 -33.733539 -10.931432 -15.780782 -27.699048
## season30    season31    season32    season33    season34    season35
## 54.732247 -34.243870 -34.401081 15.705067 -29.491730 -34.066359
## season36    season37    season38    season39    season40    season41
## -31.631422 -33.608599 -26.303982 -7.834926 -33.407788 -3.131881
## season42    season43    season44    season45    season46    season47
## -25.860403 -9.119190 -19.416854 -25.557574 -30.488701 -18.444005
## season48    season49    season50    season51    season52
## -30.604981 -29.867781 -6.184421 -16.420304 -28.345843
##
##
## Error measures:
##               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set 3.73946e-16 98.91472 27.58437 -378.36 400.8285 0.8452214
##                         ACF1
## Training set 0.02162603
##
## Forecasts:
##             Point Forecast      Lo 80      Hi 80      Lo 95      Hi 95
## 2021.096     31.331803 -101.64498 164.3086 -172.14094 234.8045
## 2021.115      4.302301 -128.67448 137.2791 -199.17044 207.7750
## 2021.135     33.703312 -99.27347 166.6801 -169.76943 237.1761
## 2021.154     14.998167 -117.97861 147.9749 -188.47458 218.4709
## 2021.173     40.653966 -92.32282 173.6307 -162.81878 244.1267
## 2021.192      7.007151 -125.96963 139.9839 -196.46559 210.4799
## 2021.212     88.519522 -44.45726 221.4963 -114.95322 291.9923
## 2021.231     49.044027 -83.93275 182.0208 -154.42872 252.5168
## 2021.250     6.036971 -126.93981 139.0138 -197.43577 209.5097
## 2021.269      8.278971 -124.69781 141.2558 -195.19377 211.7517
## 2021.288     10.043557 -122.93322 143.0203 -193.42919 213.5163
## 2021.308     10.682352 -122.29443 143.6591 -192.79039 214.1551
## 2021.327      4.857298 -128.11948 137.8341 -198.61545 208.3300
## 2021.346     3.025773 -129.95101 136.0026 -200.44697 206.4985
## 2021.365     4.569212 -128.40757 137.5460 -198.90353 208.0420
## 2021.385     3.331238 -129.64554 136.3080 -200.14151 206.8040
## 2021.404     4.751147 -128.22563 137.7279 -198.72160 208.2239
## 2021.423     4.699702 -128.27708 137.6765 -198.77304 208.1724
## 2021.442     5.424837 -127.55194 138.4016 -198.04791 208.8976
## 2021.462     4.265666 -128.71112 137.2424 -199.20708 207.7384
## 2021.481     2.103409 -130.87337 135.0802 -201.36934 205.5762
## 2021.500     24.901248 -108.07553 157.8780 -178.57150 228.3740
## 2021.519     20.047630 -112.92915 153.0244 -183.42512 223.5204
## 2021.538      8.125095 -124.85169 141.1019 -195.34765 211.5978
## 2021.558     90.552122 -42.42466 223.5289 -112.92062 294.0249
## 2021.577      1.571738 -131.40504 134.5485 -201.90101 205.0445
## 2021.596      1.410258 -131.56652 134.3870 -202.06249 204.8830
## 2021.615     51.512138 -81.46464 184.4889 -151.96061 254.9849
## 2021.635      6.311073 -126.66571 139.2879 -197.16167 209.7838
## 2021.654     1.732176 -131.24461 134.7090 -201.74057 205.2049
## 2021.673     4.162844 -128.81394 137.1396 -199.30990 207.6356

```

```

## 2021.692      2.181399 -130.79538 135.1582 -201.29135 205.6541
## 2021.712      9.481749 -123.49503 142.4585 -193.99100 212.9545
## 2021.731     27.946536 -105.03025 160.9233 -175.52621 231.4193
## 2021.750      2.369407 -130.60737 135.3462 -201.10334 205.8422
## 2021.769     32.641045 -100.33574 165.6178 -170.83170 236.1138
## 2021.788      9.908255 -123.06853 142.8850 -193.56449 213.3810
## 2021.808     26.645200 -106.33158 159.6220 -176.82754 230.1179
## 2021.827     16.343268 -116.63351 149.3200 -187.12948 219.8160
## 2021.846     10.198280 -122.77850 143.1751 -193.27447 213.6710
## 2021.865      5.262885 -127.71390 138.2397 -198.20986 208.7356
## 2021.885     17.303312 -115.67347 150.2801 -186.16943 220.7761
## 2021.904      5.138069 -127.83871 138.1149 -198.33468 208.6108
## 2021.923      5.871000 -127.10578 138.8478 -197.60174 209.3437
## 2021.942     29.550092 -103.42669 162.5269 -173.92265 233.0228
## 2021.962     19.309940 -113.66684 152.2867 -184.16280 222.7827
## 2021.981      7.380134 -125.59665 140.3569 -196.09261 210.8529
## 2022.000     35.721709 -97.25507 168.6985 -167.75104 239.1945
## 2022.019    112.367221 -20.60956 245.3440 -91.10552 315.8400
## 2022.038     22.192098 -110.67535 155.0595 -181.11335 225.4975
## 2022.058     15.489225 -117.37822 148.3567 -187.81622 218.7947
## 2022.077     9.327051 -123.54039 142.1945 -193.97839 212.6325

```

SBOO: Imputed, All - Excluding External Variables (Forecasting ENTERO)

```

# SBOO All: Imputed - Training partition (01/18/1999-01/04/2021)
#owt_df02_gb09_sboo_wkly03_train_tb03
# SBOO All: Imputed - Test partition (01/11/2021-01/03/2022)
#owt_df02_gb09_sboo_wkly03_test_tb03

owt_df02_gb09_sboo_wkly03_train_tb03 %>%
  plot_time_series(date_sample,
                    ENTERO)

owt_df02_gb09_sboo_wkly03_train_tb03 %>%
  plot_stl_diagnostics(date_sample,
                        ENTERO,
                        .frequency = "auto",
                        .trend = "auto")

m3v2_lin_out <- lr_ts_mod(tb_train = owt_df02_gb09_sboo_wkly03_train_tb03,
                            tb_test = owt_df02_gb09_sboo_wkly03_test_tb03,
                            date = "date_sample",
                            param = "ENTERO",
                            ext_pred = c(),
                            train_start = c(1999, lubridate::isoweek(ymd(train_start))),
                            test_start = c(2021, 6),
                            freq = 52,
                            h = 52,
                            formula = "ENTERO ~ trend + season")

##               ME        RMSE       MAE       MPE       MAPE       MASE
## Training set 2.091498e-14 905.3827 482.0386 -2744.347 3518.457 1.192717
## Test set      2.722014e+02 1686.6494 1062.7625 -3264.586 3297.939 2.629614
## ACF1

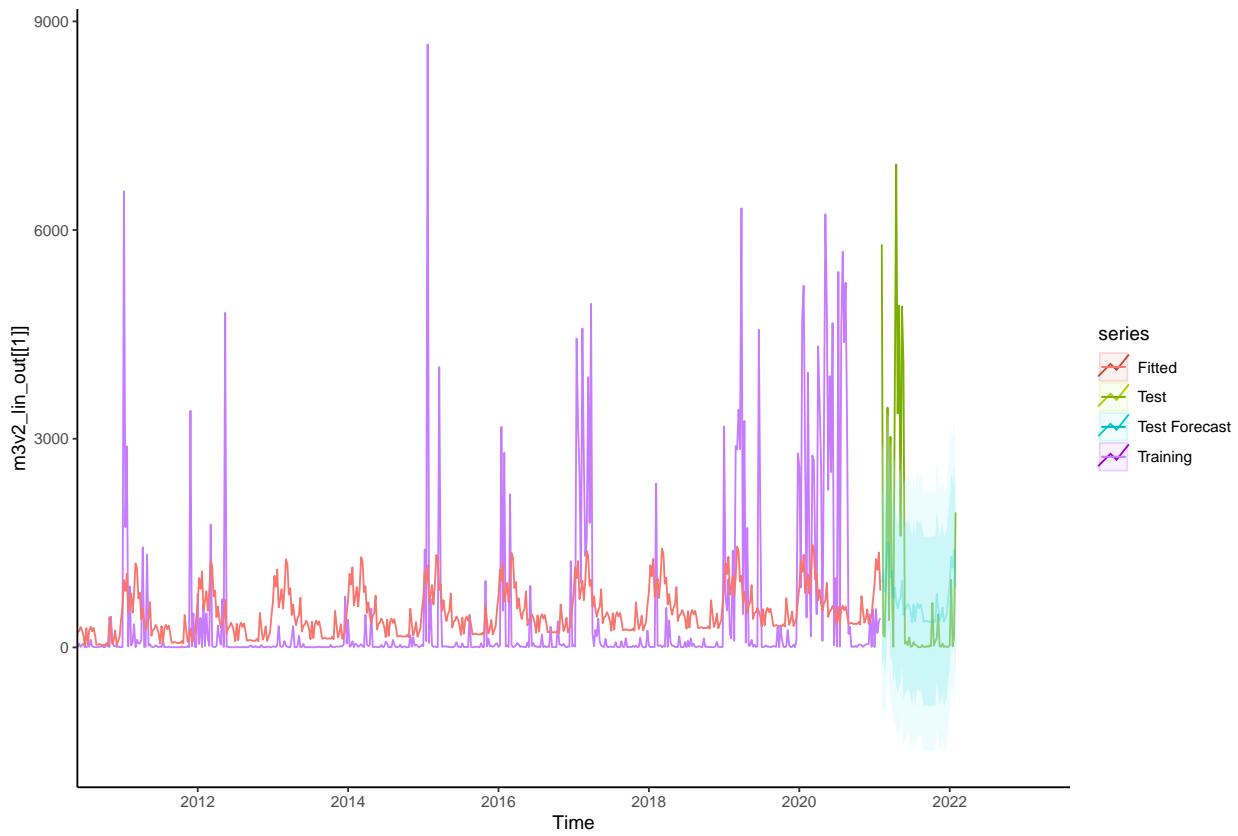
```

```

## Training set 0.3235555
## Test set NA

autoplot(m3v2_lin_out[[1]], series = 'Training') +
  autolayer(m3v2_lin_out[[2]], series = 'Test') +
  autolayer(m3v2_lin_out[[3]], series = 'Test Forecast', alpha = .3) +
  autolayer(m3v2_lin_out[[3]]$fitted, series = 'Fitted') +
  theme_classic() +
  coord_cartesian(#ylim = c(-50, 400),
                 xlim = c(2011, 2023))

```



```
summary(m3v2_lin_out[[3]])
```

```

##
## Forecast method: Linear regression model
##
## Model Information:
##
## Call:
## tslm(formula = as.formula(formula), data = train_ts01)
##
## Coefficients:
## (Intercept)      trend      season2      season3      season4      season5
##     225.589      0.584     377.402     220.371     468.883     -79.510
## season6      season7      season8      season9      season10     season11

```

```

##      37.908    182.148   -96.917    179.569    611.742    543.122
## season12    season13    season14    season15    season16    season17
##  107.326    192.044   -192.463   -41.844   -231.977   -341.002
## season18    season19    season20    season21    season22    season23
## -241.934   -197.504    52.943   -435.244   -353.935   -336.785
## season24    season25    season26    season27    season28    season29
## -278.963   -325.793   -402.667   -548.081   -297.207   -532.705
## season30    season31    season32    season33    season34    season35
## -331.355   -282.207   -345.915   -295.453   -401.295   -541.771
## season36    season37    season38    season39    season40    season41
## -542.016   -530.118   -542.174   -549.356   -543.853   -550.093
## season42    season43    season44    season45    season46    season47
## -522.702   -557.171   -149.617   -355.891   -534.109   -474.039
## season48    season49    season50    season51    season52
## -343.716   -554.582   -497.802   -435.141   -191.823
##
##
## Error measures:
##               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set 2.091498e-14 905.3827 482.0386 -2744.347 3518.457 0.9837651
##          ACF1
## Training set 0.3235555
##
## Forecasts:
##      Point Forecast     Lo 80     Hi 80     Lo 95     Hi 95
## 2021.096    933.9135 -283.24496 2151.072 -928.5062 2796.333
## 2021.115    1078.7367 -138.42179 2295.895 -783.6830 2941.156
## 2021.135     800.2559 -416.90261 2017.414 -1062.1638 2662.676
## 2021.154    1077.3259 -139.83257 2294.484 -785.0938 2939.746
## 2021.173    1510.0825 292.92405 2727.241 -352.3372 3372.502
## 2021.192    1442.0468 224.88834 2659.205 -420.3729 3304.467
## 2021.212    1006.8347 -210.32378 2223.993 -855.5850 2869.254
## 2021.231    1092.1371 -125.02136 2309.296 -770.2826 2954.557
## 2021.250     708.2144 -508.94410 1925.373 -1154.2053 2570.634
## 2021.269     859.4173 -357.74117 2076.576 -1003.0024 2721.837
## 2021.288     669.8676 -547.29088 1887.026 -1192.5521 2532.287
## 2021.308     561.4273 -655.73122 1778.586 -1300.9924 2423.847
## 2021.327     661.0793 -556.07921 1878.238 -1201.3404 2523.499
## 2021.346     706.0926 -511.06583 1923.251 -1156.3270 2568.512
## 2021.365     957.1243 -260.03420 2174.283 -905.2954 2819.544
## 2021.385     469.5212 -747.63727 1686.680 -1392.8985 2331.941
## 2021.404     551.4139 -665.74462 1768.572 -1311.0058 2413.834
## 2021.423     569.1475 -648.01098 1786.306 -1293.2722 2431.567
## 2021.442     627.5533 -589.60514 1844.712 -1234.8664 2489.973
## 2021.462     581.3083 -635.85022 1798.467 -1281.1114 2443.728
## 2021.481     505.0179 -712.14054 1722.176 -1357.4018 2367.438
## 2021.500     360.1879 -856.97056 1577.346 -1502.2318 2222.608
## 2021.519     611.6455 -605.51297 1828.804 -1250.7742 2474.065
## 2021.538     376.7316 -840.42688 1593.890 -1485.6881 2239.151
## 2021.558     578.6659 -638.49255 1795.824 -1283.7538 2441.086
## 2021.577     628.3976 -588.76084 1845.556 -1234.0221 2490.817
## 2021.596     565.2740 -651.88451 1782.432 -1297.1457 2427.694
## 2021.615     616.3201 -600.83841 1833.479 -1246.0996 2478.740
## 2021.635     511.0615 -706.09695 1728.220 -1351.3582 2373.481

```

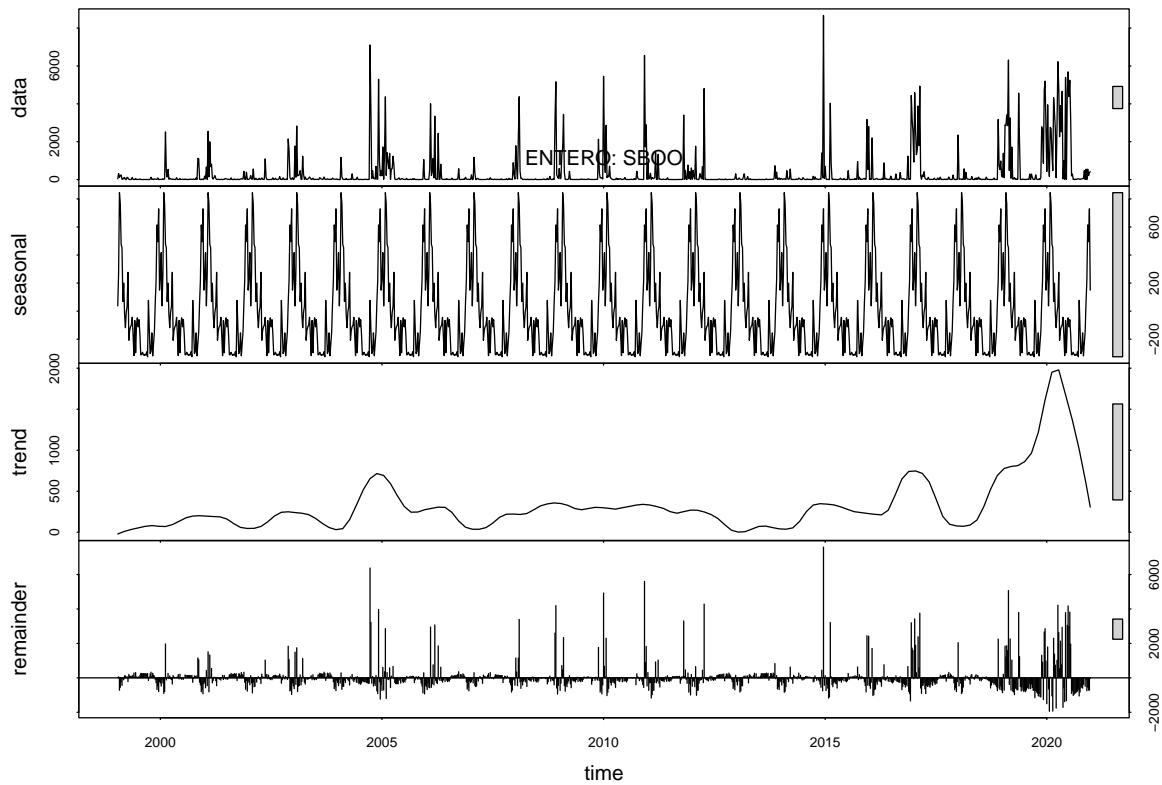
```

## 2021.654      371.1697 -845.98877 1588.328 -1491.2500 2233.589
## 2021.673      371.5091 -845.64940 1588.668 -1490.9106 2233.929
## 2021.692      383.9902 -833.16830 1601.149 -1478.4295 2246.410
## 2021.712      372.5186 -844.63991 1589.677 -1489.9011 2234.938
## 2021.731      365.9208 -851.23764 1583.079 -1496.4989 2228.341
## 2021.750      372.0077 -845.15073 1589.166 -1490.4119 2234.427
## 2021.769      366.3518 -850.80667 1583.510 -1496.0679 2228.771
## 2021.788      394.3269 -822.83160 1611.485 -1468.0928 2256.747
## 2021.808      360.4414 -856.71709 1577.600 -1501.9783 2222.861
## 2021.827      768.5797 -448.57877 1985.738 -1093.8400 2630.999
## 2021.846      562.8895 -654.26893 1780.048 -1299.5301 2425.309
## 2021.865      385.2560 -831.90252 1602.414 -1477.1637 2247.676
## 2021.885      445.9096 -771.24887 1663.068 -1416.5101 2308.329
## 2021.904      576.8165 -640.34200 1793.975 -1285.6032 2439.236
## 2021.923      366.5346 -850.62383 1583.693 -1495.8850 2228.954
## 2021.942      423.8983 -793.26020 1641.057 -1438.5214 2286.318
## 2021.962      487.1430 -730.01543 1704.302 -1375.2766 2349.563
## 2021.981      731.0454 -486.11309 1948.204 -1131.3743 2593.465
## 2022.000      923.4524 -293.70608 2140.611 -938.9673 2785.872
## 2022.019      1301.4385    84.28001 2518.597 -560.9812 3163.858
## 2022.038      1144.9914   -71.16632 2361.149 -715.8970 3005.880
## 2022.058      1394.0872   177.92955 2610.245 -466.8011 3254.976
## 2022.077      846.2787 -369.87902 2062.436 -1014.6097 2707.167

ts_full03 <- ts(na.remove(owt_df02_gb09_sboo_wkly03_train_tb02$ENTERO),
                 start = c(1999, lubridate::isoweek(ymd(train_start))),
                 freq = 52)

stl_ent03 <- stl(ts_full03, s.window="periodic")
plot(stl_ent03)
mtext("ENTERO: SBOO", side=3, line = -5)

```

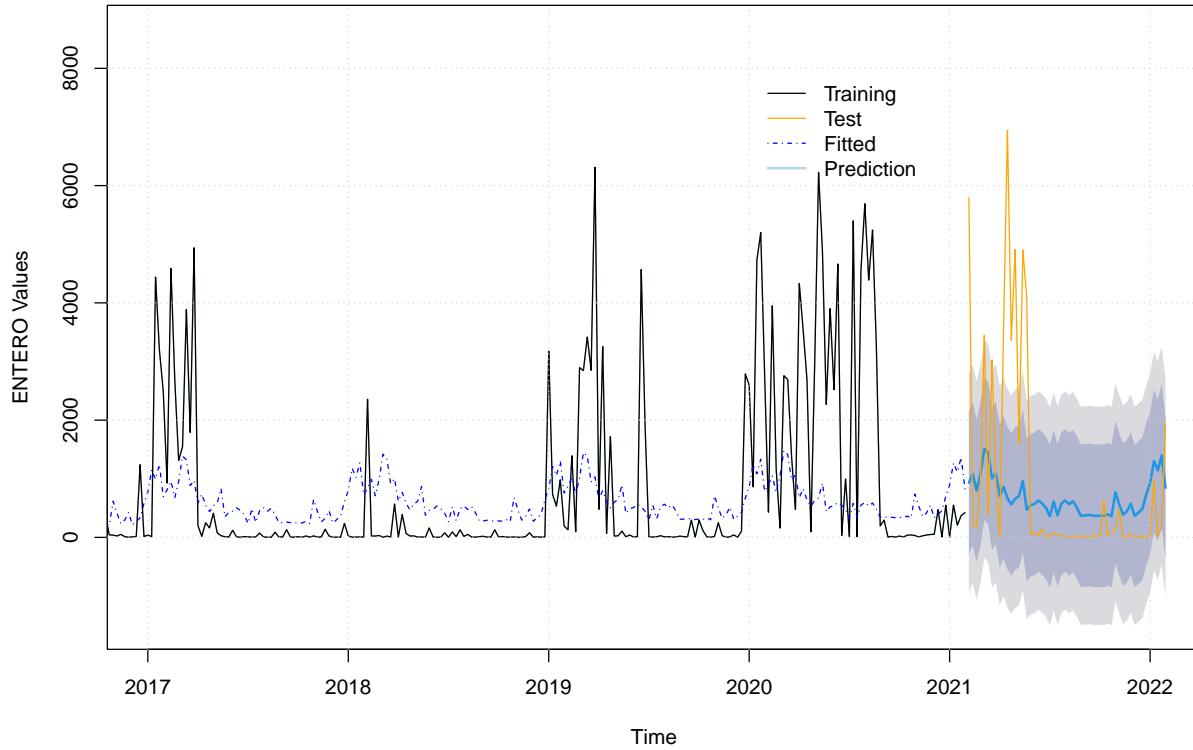


```

plot(m3v2_lin_out[[3]],
      xlim = c(2017, 2022.05),
      #ylim = c(0, 800),
      xlab = "Time",
      ylab = "ENTERO Values",
      main = "Predicted ENTERO Levels Over Time - SBOO",
      col = "light blue")
lines(m3v2_lin_out[[1]], lty=1, col="black")
lines(m3v2_lin_out[[2]], lty=1, col="orange")
lines(m3v2_lin_out[[3]]$fitted, lty=4, col="blue")
legend(2020, 8000,
      c("Training", "Test", "Fitted", "Prediction"),
      lty = c(1,1,4,1),
      lwd = c(1,1,1,2),
      col = c("black", "orange", "blue", "light blue"),
      bty = "n")
grid()

```

Predicted ENTERO Levels Over Time – SBOO



SBOO: Imputed, Excluding Outliers - Excluding External Variables (Forecasting ENTERO)

```
# SBOO Excluding Outliers: Imputed - Training partition (01/18/1999-01/04/2021)
#owt_df02_gb09_sboo_wkly02_train_tb03
# SBOO Excluding Outliers: Imputed - Test partition (01/11/2021-01/03/2022)
#owt_df02_gb09_sboo_wkly02_test_tb03

owt_df02_gb09_sboo_wkly02_train_tb03 %>%
  plot_time_series(date_sample,
    ENTERO)

owt_df02_gb09_sboo_wkly02_train_tb03 %>%
  plot_stl_diagnostics(date_sample,
    ENTERO,
    .frequency = "auto",
    .trend = "auto")

m4v2_lin_out <- lr_ts_mod(tb_train = owt_df02_gb09_sboo_wkly02_train_tb03,
  tb_test = owt_df02_gb09_sboo_wkly02_test_tb03,
  date = "date_sample",
  param = "ENTERO",
  ext_pred = c(),
  train_start = c(1999, lubridate::isoweek(ymd(train_start))),
  test_start = c(2021, 6),
  freq = 52,
```

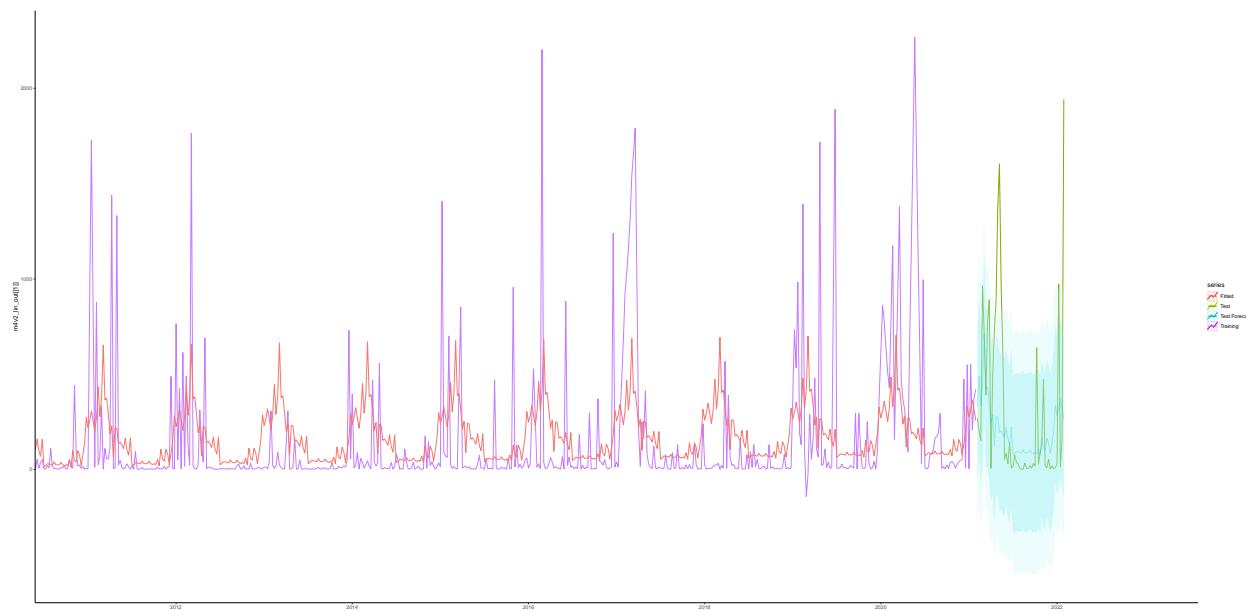
```

          h = 52,
          formula = "ENTERO ~ trend + season")

##               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set 3.250492e-14 308.4913 170.2934 -1106.7267 1236.1387 1.053705
## Test set     1.062070e+02 436.4987 271.1846  -756.8449  795.3771 1.677978
## ACF1
## Training set 0.3150918
## Test set      NA

autoplot(m4v2_lin_out[[1]], series = 'Training') +
  autolayer(m4v2_lin_out[[2]], series = 'Test') +
  autolayer(m4v2_lin_out[[3]], series = 'Test Forecast', alpha = .3) +
  autolayer(m4v2_lin_out[[3]]$fitted, series = 'Fitted') +
  theme_classic() +
  coord_cartesian(#ylim = c(-50, 400),
                 xlim = c(2011, 2023))

```



```
summary(m4v2_lin_out[[3]])
```

```

##
## Forecast method: Linear regression model
##
## Model Information:
##
## Call:
## tslm(formula = as.formula(formula), data = train_ts01)
##
## Coefficients:
## (Intercept)      trend      season2      season3      season4      season5
## 150.9575      0.1106     33.6051     87.0187     35.7086    -20.6701
## season6      season7      season8      season9      season10     season11
## 50.3431     212.9155     57.8940    154.1038    431.6208   143.4790
## season12     season13     season14     season15     season16     season17

```

```

##   154.6630    70.5984   -33.5729     8.0804   -155.0037   -0.8022
## season18    season19   season20   season21   season22   season23
##   -6.6287   -85.9898   -78.8099   -93.1244  -111.9608  -57.2997
## season24    season25   season26   season27   season28   season29
##  -119.6352  -146.6747   -58.2069  -203.1475  -197.0144  -186.8287
## season30    season31   season32   season33   season34   season35
## -188.6234  -190.2536  -193.4259  -178.4375  -192.4056  -193.0506
## season36    season37   season38   season39   season40   season41
## -192.8218  -180.4513  -192.0335  -198.7418  -192.7655  -198.5321
## season42    season43   season44   season45   season46   season47
## -170.6676  -204.6637  -118.7866  -171.1712  -180.1809  -119.6379
## season48    season49   season50   season51   season52
##  -142.9127  -199.2341  -141.9810   -78.8469    55.5372
##
##
## Error measures:
##               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set 3.250492e-14 308.4913 170.2934 -1106.727 1236.139 0.8555755
##          ACF1
## Training set 0.3150918
##
## Forecasts:
##       Point Forecast      Lo 80      Hi 80      Lo 95      Hi 95
## 2021.096    328.27252  -86.450263  742.9953 -306.31032  962.8554
## 2021.115    490.95556   76.232773  905.6783 -143.62728 1125.5384
## 2021.135    336.04459  -78.678190  750.7674 -298.53824  970.6274
## 2021.154    432.36500   17.642222  847.0878 -202.21783 1066.9478
## 2021.173    709.99258  295.269801 1124.7154   75.40975 1344.5754
## 2021.192    421.96145   7.238666  836.6842 -212.62139 1056.5443
## 2021.212    433.25602   18.533242  847.9788 -201.32681 1067.8389
## 2021.231    349.30200  -65.420787  764.0248 -285.28084  983.8848
## 2021.250    245.24129  -169.481492  659.9641 -389.34155  879.8241
## 2021.269    287.00523  -127.717553  701.7280 -347.57761  921.5881
## 2021.288    124.03171  -290.691077  538.7545 -510.55113  758.6145
## 2021.308    278.34388  -136.378897  693.0667 -356.23895  912.9267
## 2021.327    272.62793  -142.094852  687.3507 -361.95490  907.2108
## 2021.346    193.37749  -221.345297  608.1003 -441.20535  827.9603
## 2021.365    200.66792  -214.054862  615.3907 -433.91491  835.2508
## 2021.385    186.46409  -228.258690  601.1869 -448.11874  821.0469
## 2021.404    167.73822  -246.984564  582.4610 -466.84462  802.3211
## 2021.423    222.50992  -192.212867  637.2327 -412.07292  857.0928
## 2021.442    160.28511  -254.437674  575.0079 -474.29773  794.8679
## 2021.462    133.35616  -281.366620  548.0789 -501.22667  767.9390
## 2021.481    221.93456  -192.788224  636.6573 -412.64828  856.5174
## 2021.500    77.10454  -337.618244  491.8273 -557.47830  711.6874
## 2021.519    83.34826  -331.374518  498.0710 -551.23457  717.9311
## 2021.538    93.64455  -321.078231  508.3673 -540.93828  728.2274
## 2021.558    91.96050  -322.762284  506.6833 -542.62234  726.5433
## 2021.577    90.44087  -324.281916  505.1636 -544.14197  725.0237
## 2021.596    87.37918  -327.343607  502.1020 -547.20366  721.9620
## 2021.615    102.47814 -312.244640  517.2009 -532.10469  737.0610
## 2021.635    88.62067  -326.102116  503.3434 -545.96217  723.2035
## 2021.654    88.08633  -326.636455  502.8091 -546.49651  722.6692
## 2021.673    88.42570  -326.297080  503.1485 -546.15713  723.0085

```

```

## 2021.692    100.90680 -313.815983  515.6296 -533.67604  735.4896
## 2021.712    89.43519 -325.287588  504.1580 -545.14764  724.0180
## 2021.731    82.83747 -331.885316  497.5602 -551.74537  717.4203
## 2021.750    88.92437 -325.798412  503.6472 -545.65846  723.5072
## 2021.769    83.26843 -331.454348  497.9912 -551.31440  717.8513
## 2021.788    111.24350 -303.479284  525.9663 -523.33934  745.8263
## 2021.808    77.35802 -337.364767  492.0808 -557.22482  711.9409
## 2021.827    163.34576 -251.377025  578.0685 -471.23708  797.9286
## 2021.846    111.07170 -303.651085  525.7945 -523.51114  745.6545
## 2021.865    102.17259 -312.550196  516.8954 -532.41025  736.7554
## 2021.885    162.82623 -251.896548  577.5490 -471.75660  797.4091
## 2021.904    139.66199 -275.060795  554.3848 -494.92085  774.2448
## 2021.923    83.45127 -331.271508  498.1741 -551.13156  718.0341
## 2021.942    140.81490 -273.907880  555.5377 -493.76793  775.3977
## 2021.962    204.05967 -210.663109  618.7825 -430.52316  838.6425
## 2021.981    338.55431 -76.168468  753.2771 -296.02852  973.1371
## 2022.000    283.12774 -131.595043  697.8505 -351.45510  917.7106
## 2022.019    316.84349 -97.879294  731.5663 -317.73935  951.4263
## 2022.038    370.36768 -44.014108  784.7495 -263.69338  1004.4287
## 2022.058    319.16816 -95.213622  733.5499 -314.89290  953.2292
## 2022.077    262.90003 -151.481756  677.2818 -371.16103  896.9611

```

PLOO All: Imputed - Excluding External Variables (Forecasting DENSITY)

```

# PLOO All: Imputed - Training partition (01/18/1999-01/04/2021)
#owt_df02_gb09_ploo_wkly03_train_tb03
# PLOO All: Imputed - Test partition (01/11/2021-01/03/2022)
#owt_df02_gb09_ploo_wkly03_test_tb03

owt_df02_gb09_ploo_wkly03_train_tb03 %>%
  plot_time_series(date_sample,
                    DENSITY)

owt_df02_gb09_ploo_wkly03_train_tb03 %>%
  plot_stl_diagnostics(date_sample,
                        DENSITY,
                        .frequency = "auto",
                        .trend = "auto")

m1v3_lin_out <- lr_ts_mod(tb_train = owt_df02_gb09_ploo_wkly03_train_tb03,
                            tb_test = owt_df02_gb09_ploo_wkly03_test_tb03,
                            date = "date_sample",
                            param = "DENSITY",
                            ext_pred = c(),
                            train_start = c(1999, lubridate::isoweek(ymd(train_start))),
                            test_start = c(2021, 6),
                            freq = 52,
                            h = 52,
                            formula = "DENSITY ~ trend + season")

##               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -1.556207e-17 0.3503373 0.2742432 -0.02019261 1.110196 1.320782

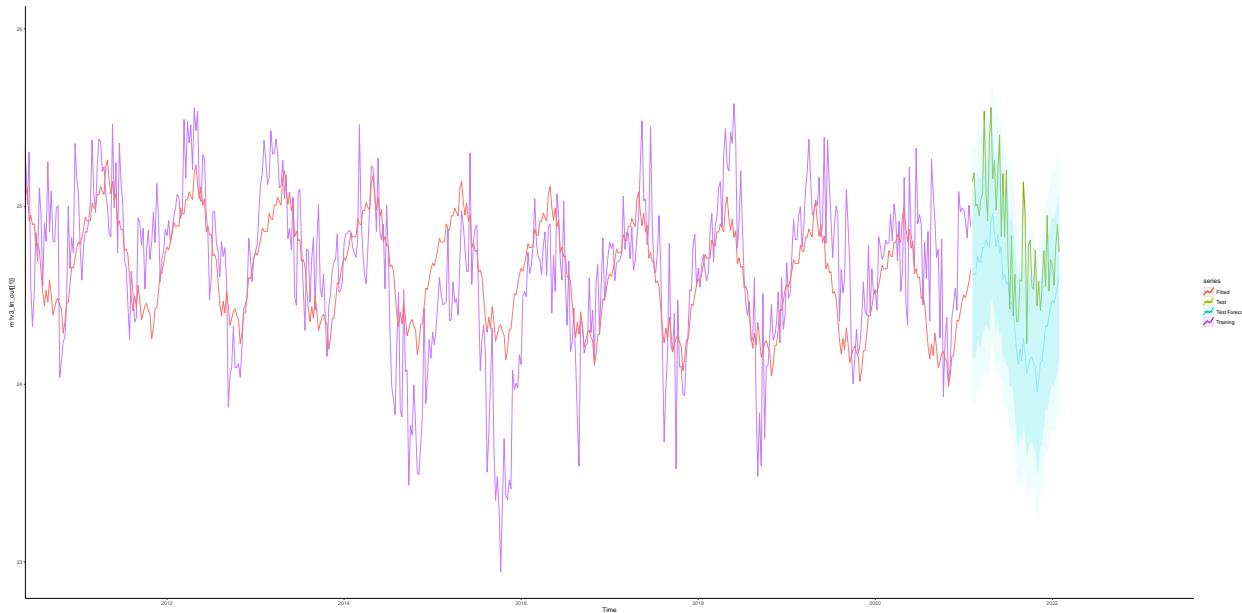
```

```

## Test set      3.777035e-01 0.4351803 0.3804691  1.51503645 1.526371 1.832377
##                      ACF1
## Training set 0.7064924
## Test set       NA

autoplot(m1v3_lin_out[[1]], series = 'Training') +
  autolayer(m1v3_lin_out[[2]], series = 'Test') +
  autolayer(m1v3_lin_out[[3]], series = 'Test Forecast', alpha = .3) +
  autolayer(m1v3_lin_out[[3]]$fitted, series = 'Fitted') +
  theme_classic() +
  coord_cartesian(#ylim = c(-50, 400),
                  xlim = c(2011, 2023))

```



```

summary(m1v3_lin_out[[3]])

##
## Forecast method: Linear regression model
##
## Model Information:
##
## Call:
## tslm(formula = as.formula(formula), data = train_ts01)
##
## Coefficients:
## (Intercept)      trend      season2      season3      season4      season5
## 25.1605258   -0.0005763   -0.0139985   0.0277806   0.0773615   0.1416388
##    season6      season7      season8      season9      season10     season11
## 0.1203487    0.1223233   0.1235907   0.2250269   0.1966168   0.1916811
##    season12     season13     season14     season15     season16     season17
## 0.2733601    0.2726401   0.3183751   0.2929450   0.2756329   0.4283849
##    season18     season19     season20     season21     season22     season23
## 0.4714110    0.3565331   0.2824009   0.3510641   0.2477110   0.2844833
##    season24     season25     season26     season27     season28     season29
## 0.1017596    0.1332048   0.0818576   0.0866414   -0.0332705  -0.0234003

```

```

##      season30      season31      season32      season33      season34      season35
## -0.0571402 -0.1796645 -0.2983021 -0.3700808 -0.2852211 -0.3496621
##      season36      season37      season38      season39      season40      season41
## -0.2241718 -0.3097002 -0.4174868 -0.3792932 -0.3384987 -0.3239866
##      season42      season43      season44      season45      season46      season47
## -0.3600467 -0.3826167 -0.5190471 -0.4593628 -0.3477045 -0.3443540
##      season48      season49      season50      season51      season52
## -0.2499449 -0.1436323 -0.1493562 -0.0926189 -0.0328901
##
##
## Error measures:
##               ME        RMSE       MAE       MPE       MAPE       MASE
## Training set -1.556207e-17 0.3503373 0.2742432 -0.02019261 1.110196 0.8194057
##          ACF1
## Training set 0.7064924
##
## Forecasts:
##      Point Forecast     Lo 80     Hi 80     Lo 95     Hi 95
## 2021.096    24.61928 24.14830 25.09026 23.89862 25.33995
## 2021.115    24.62068 24.14970 25.09166 23.90002 25.34134
## 2021.135    24.62137 24.15039 25.09235 23.90071 25.34203
## 2021.154    24.72223 24.25125 25.19321 24.00157 25.44289
## 2021.173    24.69325 24.22227 25.16422 23.97258 25.41391
## 2021.192    24.68773 24.21676 25.15871 23.96707 25.40840
## 2021.212    24.76884 24.29786 25.23982 24.04817 25.48950
## 2021.231    24.76754 24.29656 25.23852 24.04688 25.48820
## 2021.250    24.81270 24.34172 25.28368 24.09204 25.53336
## 2021.269    24.78669 24.31571 25.25767 24.06603 25.50735
## 2021.288    24.76880 24.29783 25.23978 24.04814 25.48947
## 2021.308    24.92098 24.45000 25.39196 24.20032 25.64164
## 2021.327    24.96343 24.49245 25.43441 24.24277 25.68409
## 2021.346    24.84798 24.37700 25.31895 24.12731 25.56864
## 2021.365    24.77327 24.30229 25.24425 24.05261 25.49393
## 2021.385    24.84135 24.37038 25.31233 24.12069 25.56202
## 2021.404    24.73742 24.26645 25.20840 24.01676 25.45809
## 2021.423    24.77362 24.30264 25.24460 24.05296 25.49428
## 2021.442    24.59032 24.11934 25.06130 23.86966 25.31098
## 2021.462    24.62119 24.15021 25.09217 23.90053 25.34185
## 2021.481    24.56927 24.09829 25.04024 23.84860 25.28993
## 2021.500    24.57347 24.10249 25.04445 23.85281 25.29414
## 2021.519    24.45299 23.98201 24.92396 23.73232 25.17365
## 2021.538    24.46228 23.99130 24.93326 23.74162 25.18294
## 2021.558    24.42796 23.95698 24.89894 23.70730 25.14863
## 2021.577    24.30486 23.83388 24.77584 23.58420 25.02552
## 2021.596    24.18565 23.71467 24.65663 23.46499 24.90631
## 2021.615    24.11329 23.64231 24.58427 23.39263 24.83396
## 2021.635    24.19758 23.72660 24.66856 23.47691 24.91824
## 2021.654    24.13256 23.66158 24.60354 23.41190 24.85322
## 2021.673    24.25747 23.78650 24.72845 23.53681 24.97814
## 2021.692    24.17137 23.70039 24.64235 23.45071 24.89203
## 2021.712    24.06301 23.59203 24.53398 23.34234 24.78367
## 2021.731    24.10062 23.62964 24.57160 23.37996 24.82129
## 2021.750    24.14084 23.66986 24.61182 23.42018 24.86150
## 2021.769    24.15478 23.68380 24.62576 23.43412 24.87544

```

```

## 2021.788      24.11814 23.64716 24.58912 23.39748 24.83880
## 2021.808      24.09499 23.62402 24.56597 23.37433 24.81566
## 2021.827      23.95799 23.48701 24.42897 23.23733 24.67865
## 2021.846      24.01710 23.54612 24.48807 23.29643 24.73776
## 2021.865      24.12818 23.65720 24.59916 23.40752 24.84884
## 2021.885      24.13095 23.65997 24.60193 23.41029 24.85161
## 2021.904      24.22479 23.75381 24.69576 23.50412 24.94545
## 2021.923      24.33052 23.85954 24.80150 23.60986 25.05118
## 2021.942      24.32422 23.85324 24.79520 23.60356 25.04488
## 2021.962      24.38038 23.90940 24.85136 23.65972 25.10104
## 2021.981      24.43953 23.96856 24.91051 23.71887 25.16020
## 2022.000      24.47185 24.00087 24.94283 23.75119 25.19251
## 2022.019      24.45727 23.98629 24.92825 23.73661 25.17794
## 2022.038      24.49848 24.02788 24.96907 23.77841 25.21855
## 2022.058      24.54748 24.07689 25.01807 23.82741 25.26755
## 2022.077      24.61118 24.14059 25.08177 23.89111 25.33125

```

PLOO Excluding Outliers: Imputed - Excluding External Variables (Forecasting DENSITY)

```

# PLOO Excluding Outliers: Imputed - Training partition (01/18/1999-01/04/2021)
#owt_df02_gb09_ploo_wkly02_train_tb03
# PLOO Excluding Outliers: Imputed - Test partition (01/11/2021-01/03/2022)
#owt_df02_gb09_ploo_wkly02_test_tb03

owt_df02_gb09_ploo_wkly02_train_tb03 %>%
  plot_time_series(date_sample,
                    DENSITY)

owt_df02_gb09_ploo_wkly02_train_tb03 %>%
  plot_stl_diagnostics(date_sample,
                        DENSITY,
                        .frequency = "auto",
                        .trend = "auto")

m2v3_lin_out <- lr_ts_mod(tb_train = owt_df02_gb09_ploo_wkly02_train_tb03,
                            tb_test = owt_df02_gb09_ploo_wkly02_test_tb03,
                            date = "date_sample",
                            param = "DENSITY",
                            ext_pred = c(),
                            train_start = c(1999, lubridate::isoweek(ymd(train_start))),
                            test_start = c(2021, 6),
                            freq = 52,
                            h = 52,
                            formula = "DENSITY ~ trend + season")

##               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set 2.796938e-17 0.3485679 0.2736529 -0.01996944 1.107586 1.321202
## Test set     3.761243e-01 0.4336376 0.3789111  1.50865787 1.520078 1.829391
##               ACF1
## Training set 0.7046418
## Test set     NA

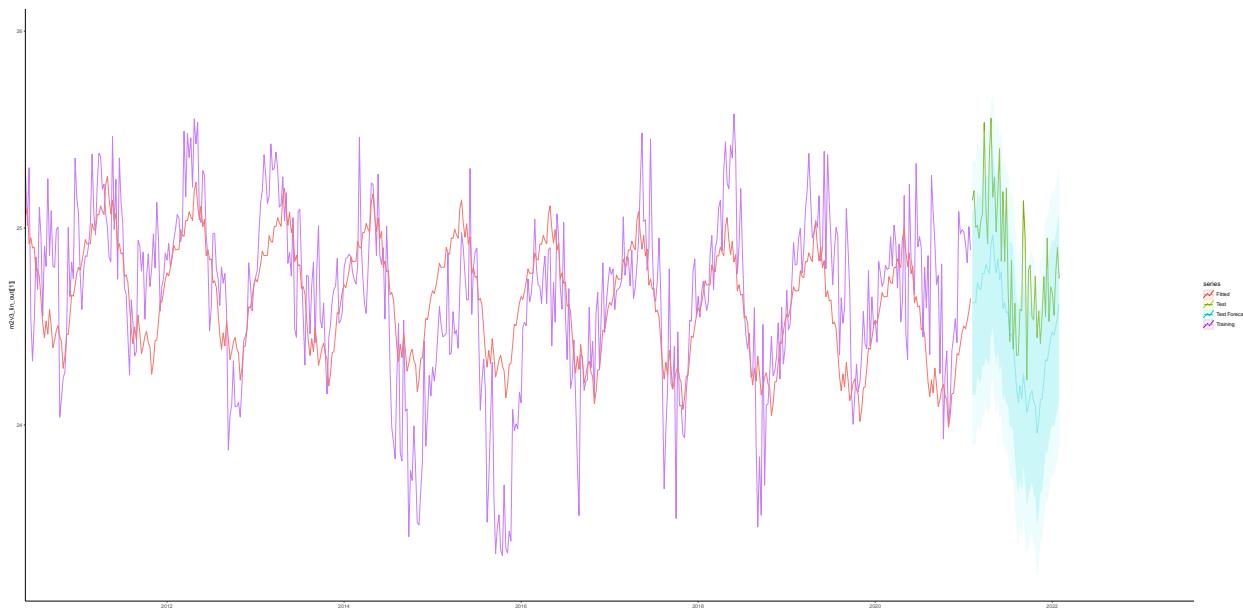
autoplot(m2v3_lin_out[[1]], series = 'Training') +
  autolayer(m2v3_lin_out[[2]], series = 'Test') +

```

```

autolayer(m2v3_lin_out[[3]], series = 'Test Forecast', alpha = .3) +
autolayer(m2v3_lin_out[[3]]$fitted, series = 'Fitted') +
theme_classic() +
coord_cartesian(#ylim = c(-50, 400),
                xlim = c(2011, 2023))

```



```
summary(m2v3_lin_out[[3]])
```

```

##
## Forecast method: Linear regression model
##
## Model Information:
##
## Call:
## tslm(formula = as.formula(formula), data = train_ts01)
##
## Coefficients:
## (Intercept)      trend   season2   season3   season4   season5
## 25.1592432 -0.0005744 -0.0139648  0.0278971  0.0781413  0.1422164
## season6    season7   season8   season9   season10  season11
## 0.1206828  0.1223464  0.1236649  0.2249494  0.1966133  0.1918159
## season12  season13   season14   season15  season16  season17
## 0.2732921  0.2727331  0.3187065  0.2929821  0.2758615  0.4275024
## season18  season19   season20   season21  season22  season23
## 0.4703928  0.3563714  0.2825144  0.3513254  0.2478709  0.2847145
## season24  season25   season26   season27  season28  season29
## 0.1017506  0.1332591  0.0818842  0.0866295 -0.0333260 -0.0233663
## season30  season31   season32   season33  season34  season35
## -0.0584356 -0.1794493 -0.2979102 -0.3700239 -0.2849689 -0.3496441
## season36  season37   season38   season39  season40  season41
## -0.2239530 -0.3095001 -0.4174060 -0.3790865 -0.3280387 -0.3046191
## season42  season43   season44   season45  season46  season47
## -0.3598039 -0.3822471 -0.5187880 -0.4592765 -0.3478977 -0.3441984

```

```

##      season48      season49      season50      season51      season52
## -0.2496997 -0.1436264 -0.1493883 -0.0927520 -0.0329772
##
##
## Error measures:
##               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set 2.796938e-17 0.3485679 0.2736529 -0.01996944 1.107586 0.8203458
##          ACF1
## Training set 0.7046418
##
## Forecasts:
##      Point Forecast    Lo 80     Hi 80    Lo 95     Hi 95
## 2021.096      24.62052 24.15192 25.08912 23.90350 25.33755
## 2021.115      24.62161 24.15301 25.09021 23.90459 25.33864
## 2021.135      24.62236 24.15376 25.09096 23.90533 25.33938
## 2021.154      24.72307 24.25447 25.19167 24.00604 25.44009
## 2021.173      24.69416 24.22556 25.16276 23.97713 25.41118
## 2021.192      24.68878 24.22018 25.15738 23.97176 25.40581
## 2021.212      24.76969 24.30109 25.23829 24.05266 25.48671
## 2021.231      24.76855 24.29995 25.23715 24.05153 25.48558
## 2021.250      24.81395 24.34535 25.28255 24.09693 25.53097
## 2021.269      24.78765 24.31905 25.25625 24.07063 25.50468
## 2021.288      24.76996 24.30136 25.23856 24.05294 25.48698
## 2021.308      24.92103 24.45243 25.38963 24.20400 25.63805
## 2021.327      24.96334 24.49474 25.43194 24.24632 25.68036
## 2021.346      24.84875 24.38015 25.31735 24.13172 25.56577
## 2021.365      24.77431 24.30571 25.24291 24.05729 25.49134
## 2021.385      24.84255 24.37395 25.31115 24.12553 25.55957
## 2021.404      24.73852 24.26992 25.20712 24.02150 25.45554
## 2021.423      24.77479 24.30619 25.24339 24.05777 25.49181
## 2021.442      24.59125 24.12265 25.05985 23.87423 25.30827
## 2021.462      24.62219 24.15359 25.09079 23.90516 25.33921
## 2021.481      24.57024 24.10164 25.03884 23.85321 25.28726
## 2021.500      24.57441 24.10581 25.04301 23.85739 25.29143
## 2021.519      24.45388 23.98528 24.92248 23.73686 25.17090
## 2021.538      24.46326 23.99466 24.93186 23.74624 25.18029
## 2021.558      24.42762 23.95902 24.89622 23.71060 25.14464
## 2021.577      24.30603 23.83743 24.77463 23.58901 25.02305
## 2021.596      24.18700 23.71840 24.65560 23.46997 24.90402
## 2021.615      24.11431 23.64571 24.58291 23.39729 24.83133
## 2021.635      24.19879 23.73019 24.66739 23.48177 24.91581
## 2021.654      24.13354 23.66494 24.60214 23.41652 24.85056
## 2021.673      24.25866 23.79006 24.72726 23.54163 24.97568
## 2021.692      24.17253 23.70393 24.64113 23.45551 24.88956
## 2021.712      24.06405 23.59545 24.53265 23.34703 24.78108
## 2021.731      24.10180 23.63320 24.57040 23.38478 24.81882
## 2021.750      24.15227 23.68367 24.62087 23.43525 24.86930
## 2021.769      24.17512 23.70652 24.64372 23.45810 24.89214
## 2021.788      24.11936 23.65076 24.58796 23.40234 24.83638
## 2021.808      24.09634 23.62774 24.56494 23.37932 24.81336
## 2021.827      23.95923 23.49063 24.42783 23.24220 24.67625
## 2021.846      24.01816 23.54956 24.48676 23.30114 24.73519
## 2021.865      24.12897 23.66037 24.59757 23.41195 24.84599
## 2021.885      24.13209 23.66349 24.60069 23.41507 24.84911

```

```

## 2021.904      24.22602 23.75742 24.69462 23.50899 24.94304
## 2021.923      24.33152 23.86292 24.80012 23.61449 25.04854
## 2021.942      24.32518 23.85658 24.79378 23.60816 25.04220
## 2021.962      24.38124 23.91264 24.84984 23.66422 25.09826
## 2021.981      24.44044 23.97184 24.90904 23.72342 25.15746
## 2022.000      24.47284 24.00424 24.94144 23.75582 25.18987
## 2022.019      24.45831 23.98971 24.92691 23.74128 25.17533
## 2022.038      24.49959 24.03138 24.96781 23.78316 25.21603
## 2022.058      24.54926 24.08105 25.01748 23.83283 25.26570
## 2022.077      24.61276 24.14455 25.08098 23.89633 25.32920

```

SBOO All: Imputed - Excluding External Variables (Forecasting DENSITY)

```

# SBOO All: Imputed - Training partition (01/18/1999-01/04/2021)
#owt_df02_gb09_sboo_wkly03_train_tb03
# SBOO All: Imputed - Test partition (01/11/2021-01/03/2022)
#owt_df02_gb09_sboo_wkly03_test_tb03

owt_df02_gb09_sboo_wkly03_train_tb03 %>%
  plot_time_series(date_sample,
                    DENSITY)

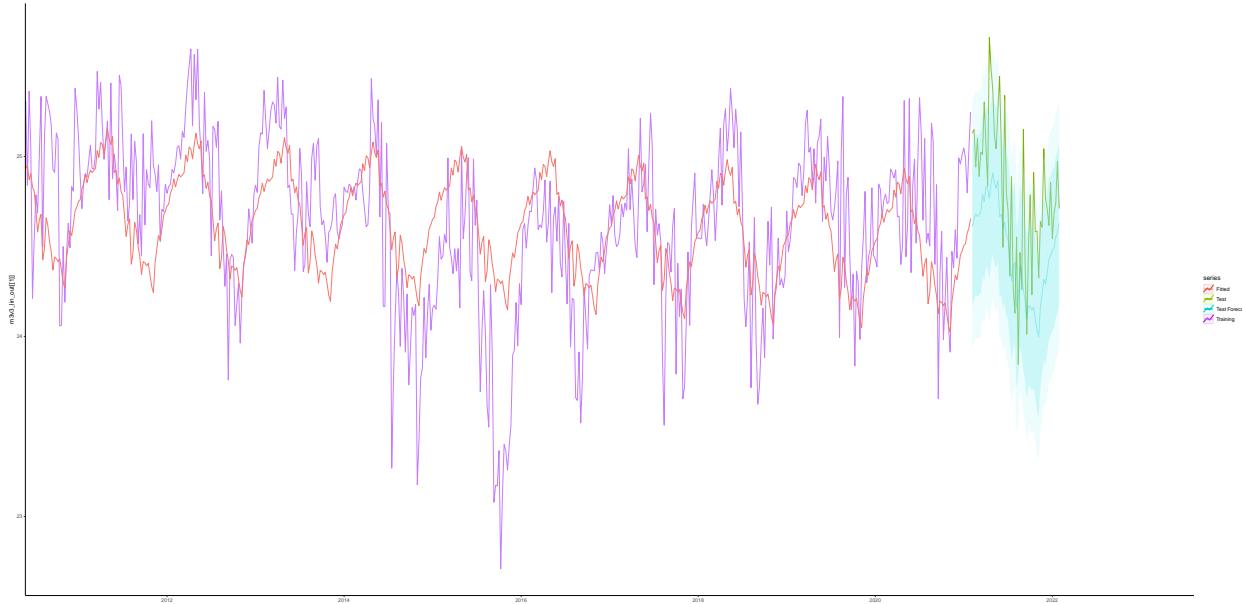
owt_df02_gb09_sboo_wkly03_train_tb03 %>%
  plot_stl_diagnostics(date_sample,
                        DENSITY,
                        .frequency = "auto",
                        .trend = "auto")

m3v3_lin_out <- lr_ts_mod(tb_train = owt_df02_gb09_sboo_wkly03_train_tb03,
                            tb_test = owt_df02_gb09_sboo_wkly03_test_tb03,
                            date = "date_sample",
                            param = "DENSITY",
                            ext_pred = c(),
                            train_start = c(1999, lubridate::isoweek(ymd(train_start))),
                            test_start = c(2021, 6),
                            freq = 52,
                            h = 52,
                            formula = "DENSITY ~ trend + season")

##               ME        RMSE       MAE       MPE       MAPE       MASE
## Training set -2.798007e-17 0.3326557 0.2541997 -0.01841579 1.032368 1.263408
## Test set      3.166582e-01 0.4235576 0.3602488  1.26390269 1.444778 1.790487
##               ACF1
## Training set 0.6520061
## Test set     NA

autoplot(m3v3_lin_out[[1]], series = 'Training') +
  autolayer(m3v3_lin_out[[2]], series = 'Test') +
  autolayer(m3v3_lin_out[[3]], series = 'Test Forecast', alpha = .3) +
  autolayer(m3v3_lin_out[[3]]$fitted, series = 'Fitted') +
  theme_classic() +
  coord_cartesian(#ylim = c(-50, 400),
                 xlim = c(2011, 2023))

```



```
summary(m3v3_lin_out[[3]])
```

```
##
## Forecast method: Linear regression model
##
## Model Information:
##
## Call:
## tslm(formula = as.formula(formula), data = train_ts01)
##
## Coefficients:
## (Intercept)      trend    season2    season3    season4    season5
## 25.0417603   -0.0004698   0.0110090   0.0712780   0.0980308   0.1525658
##   season6    season7    season8    season9    season10   season11
## 0.1078284   0.1471860   0.1784557   0.1650903   0.1773321   0.1894318
##   season12   season13   season14   season15   season16   season17
## 0.2886061   0.2512015   0.3338875   0.3233242   0.2689084   0.3502046
##   season18   season19   season20   season21   season22   season23
## 0.4120513   0.3532771   0.3264399   0.3700406   0.1716783   0.1838657
##   season24   season25   season26   season27   season28   season29
## 0.1083453   0.1423525   0.0683264   0.0448021  -0.0636511  -0.1804622
##   season30   season31   season32   season33   season34   season35
## -0.1129465  -0.0813626  -0.3329194  -0.2726196  -0.0988885  -0.1411537
##   season36   season37   season38   season39   season40   season41
## -0.2303418  -0.2876174  -0.3867274  -0.3155147  -0.3218860  -0.3383215
##   season42   season43   season44   season45   season46   season47
## -0.3233811  -0.3860008  -0.4507198  -0.4841374  -0.3174195  -0.2429598
##   season48   season49   season50   season51   season52
## -0.1657881  -0.1907846  -0.1278259  -0.0528468  -0.0307239
##
##
## Error measures:
##               ME        RMSE        MAE        MPE        MAPE        MASE
## Training set -2.798007e-17 0.3326557 0.2541997 -0.01841579 1.032368 0.7493704
```

```

##          ACF1
## Training set 0.6520061
##
## Forecasts:
##      Point Forecast    Lo 80     Hi 80     Lo 95     Hi 95
## 2021.096      24.61029 24.16308 25.05750 23.92600 25.29458
## 2021.115      24.64918 24.20197 25.09639 23.96489 25.33347
## 2021.135      24.67998 24.23277 25.12719 23.99569 25.36427
## 2021.154      24.66614 24.21894 25.11335 23.98185 25.35043
## 2021.173      24.67792 24.23071 25.12512 23.99363 25.36221
## 2021.192      24.68955 24.24234 25.13675 24.00526 25.37384
## 2021.212      24.78825 24.34104 25.23546 24.10396 25.47254
## 2021.231      24.75038 24.30317 25.19758 24.06609 25.43467
## 2021.250      24.83259 24.38538 25.27980 24.14830 25.51688
## 2021.269      24.82156 24.37435 25.26877 24.13727 25.50585
## 2021.288      24.76667 24.31947 25.21388 24.08238 25.45096
## 2021.308      24.84750 24.40029 25.29471 24.16321 25.53179
## 2021.327      24.90888 24.46167 25.35609 24.22459 25.59317
## 2021.346      24.84963 24.40242 25.29684 24.16534 25.53392
## 2021.365      24.82233 24.37512 25.26953 24.13804 25.50662
## 2021.385      24.86546 24.41825 25.31266 24.18117 25.54975
## 2021.404      24.66662 24.21942 25.11383 23.98233 25.35091
## 2021.423      24.67834 24.23113 25.12555 23.99405 25.36263
## 2021.442      24.60235 24.15514 25.04956 23.91806 25.28664
## 2021.462      24.63589 24.18868 25.08310 23.95160 25.32018
## 2021.481      24.56139 24.11419 25.00860 23.87710 25.24568
## 2021.500      24.53740 24.09019 24.98461 23.85311 25.22169
## 2021.519      24.42848 23.98127 24.87568 23.74419 25.11277
## 2021.538      24.31120 23.86399 24.75840 23.62691 24.99549
## 2021.558      24.37824 23.93103 24.82545 23.69395 25.06253
## 2021.577      24.40936 23.96215 24.85656 23.72507 25.09365
## 2021.596      24.15733 23.71012 24.60454 23.47304 24.84162
## 2021.615      24.21716 23.76995 24.66437 23.53287 24.90145
## 2021.635      24.39042 23.94321 24.83763 23.70613 25.07471
## 2021.654      24.34769 23.90048 24.79489 23.66340 25.03198
## 2021.673      24.25803 23.81082 24.70524 23.57374 24.94232
## 2021.692      24.20028 23.75307 24.64749 23.51599 24.88457
## 2021.712      24.10070 23.65349 24.54791 23.41641 24.78499
## 2021.731      24.17145 23.72424 24.61865 23.48716 24.85574
## 2021.750      24.16460 23.71740 24.61181 23.48031 24.84889
## 2021.769      24.14770 23.70049 24.59491 23.46341 24.83199
## 2021.788      24.16217 23.71496 24.60938 23.47788 24.84646
## 2021.808      24.09908 23.65187 24.54629 23.41479 24.78337
## 2021.827      24.03389 23.58668 24.48110 23.34960 24.71818
## 2021.846      24.00000 23.55280 24.44721 23.31571 24.68429
## 2021.865      24.16625 23.71904 24.61346 23.48196 24.85054
## 2021.885      24.24024 23.79303 24.68745 23.55595 24.92453
## 2021.904      24.31694 23.86974 24.76415 23.63265 25.00123
## 2021.923      24.29148 23.84427 24.73869 23.60719 24.97577
## 2021.942      24.35397 23.90676 24.80118 23.66968 25.03826
## 2021.962      24.42848 23.98127 24.87568 23.74419 25.11277
## 2021.981      24.45013 24.00292 24.89734 23.76584 25.13442
## 2022.000      24.48038 24.03318 24.92759 23.79609 25.16467
## 2022.019      24.49092 24.04371 24.93813 23.80663 25.17521

```

```

## 2022.038      24.55072 24.10388 24.99756 23.86699 25.23445
## 2022.058      24.57700 24.13016 25.02385 23.89328 25.26073
## 2022.077      24.63107 24.18423 25.07791 23.94734 25.31480

```

SBOO Excluding Outliers: Imputed - Excluding External Variables (Forecasting DENSITY)

```

# SBOO Excluding Outliers: Imputed - Training partition (01/18/1999-01/04/2021)
#owt_df02_gb09_sboo_wkly02_train_tb03
# SBOO Excluding Outliers: Imputed - Test partition (01/11/2021-01/03/2022)
#owt_df02_gb09_sboo_wkly02_test_tb03

owt_df02_gb09_sboo_wkly02_train_tb03 %>%
  plot_time_series(date_sample,
                    DENSITY)

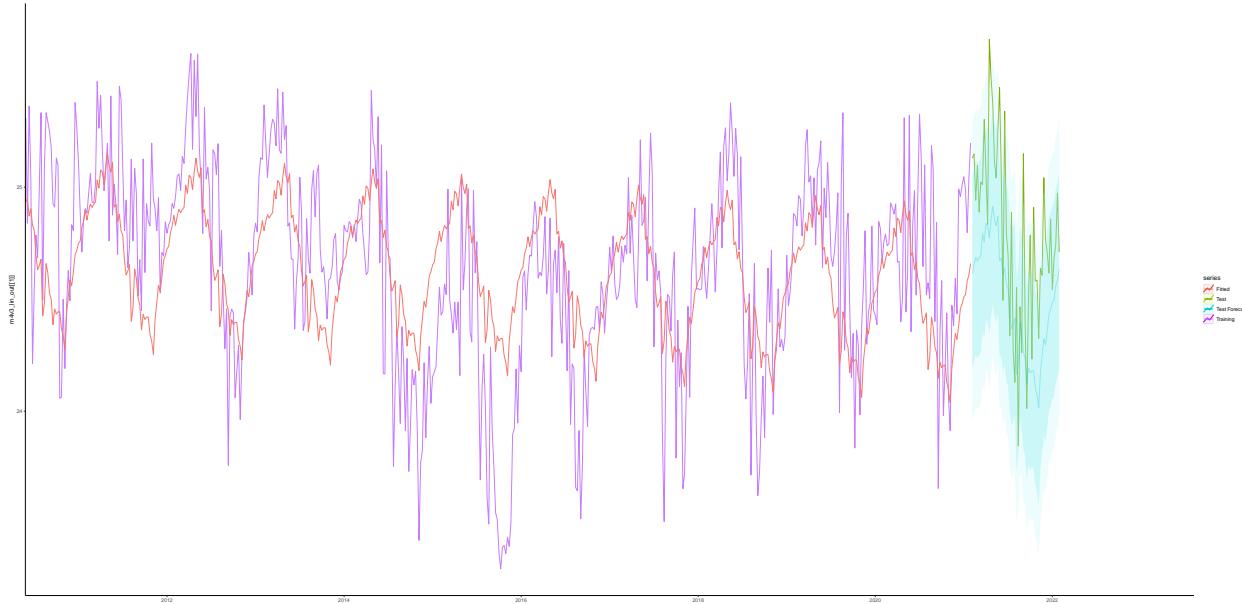
owt_df02_gb09_sboo_wkly02_train_tb03 %>%
  plot_stl_diagnostics(date_sample,
                        DENSITY,
                        .frequency = "auto",
                        .trend = "auto")

m4v3_lin_out <- lr_ts_mod(tb_train = owt_df02_gb09_sboo_wkly02_train_tb03,
                            tb_test = owt_df02_gb09_sboo_wkly02_test_tb03,
                            date = "date_sample",
                            param = "DENSITY",
                            ext_pred = c(),
                            train_start = c(1999, lubridate::isoweek(ymd(train_start))),
                            test_start = c(2021, 6),
                            freq = 52,
                            h = 52,
                            formula = "DENSITY ~ trend + season")

##               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set 1.537781e-17 0.3223322 0.2498774 -0.01720746 1.013680 1.259573
## Test set     3.070166e-01 0.4160535 0.3528652  1.22485103 1.415072 1.778710
##               ACF1
## Training set 0.6443114
## Test set     NA

autoplot(m4v3_lin_out[[1]], series = 'Training') +
  autolayer(m4v3_lin_out[[2]], series = 'Test') +
  autolayer(m4v3_lin_out[[3]], series = 'Test Forecast', alpha = .3) +
  autolayer(m4v3_lin_out[[3]]$fitted, series = 'Fitted') +
  theme_classic() +
  coord_cartesian(#ylim = c(-50, 400),
                 xlim = c(2011, 2023))

```



```
summary(m4v3_lin_out[[3]])
```

```
##
## Forecast method: Linear regression model
##
## Model Information:
##
## Call:
## tslm(formula = as.formula(formula), data = train_ts01)
##
## Coefficients:
## (Intercept)      trend    season2    season3    season4    season5
## 25.0356824 -0.0004601  0.0116001  0.0730078  0.0998429  0.1509991
##   season6    season7    season8    season9    season10   season11
##  0.1069505  0.1485823  0.1794448  0.1647678  0.1786420  0.1897574
##   season12   season13   season14   season15   season16   season17
##  0.2885086  0.2523327  0.3345139  0.3241873  0.2703485  0.3521452
##   season18   season19   season20   season21   season22   season23
##  0.4123527  0.3540656  0.3276523  0.3698347  0.1716294  0.1838007
##   season24   season25   season26   season27   season28   season29
##  0.1091518  0.1414890  0.0681124  0.0485362 -0.0632368 -0.1280942
##   season30   season31   season32   season33   season34   season35
## -0.1087901 -0.0809060 -0.3329658 -0.2747655 -0.0988908 -0.1408935
##   season36   season37   season38   season39   season40   season41
## -0.2296922 -0.2567784 -0.3691782 -0.2990309 -0.3212594 -0.3101461
##   season42   season43   season44   season45   season46   season47
## -0.3102373 -0.3851553 -0.4165798 -0.4753008 -0.3170392 -0.2427899
##   season48   season49   season50   season51   season52
## -0.1654119 -0.1909596 -0.1276126 -0.0521993 -0.0309902
##
## Error measures:
##               ME        RMSE        MAE        MPE        MAPE       MASE
## Training set 1.537781e-17 0.3223322 0.2498774 -0.01720746 1.01368 0.7490735
```

```

##          ACF1
## Training set 0.6443114
##
## Forecasts:
##           Point Forecast    Lo 80     Hi 80     Lo 95     Hi 95
## 2021.096      24.61448 24.18115 25.04781 23.95143 25.27754
## 2021.115      24.65566 24.22233 25.08899 23.99260 25.31871
## 2021.135      24.68606 24.25273 25.11939 24.02300 25.34911
## 2021.154      24.67092 24.23759 25.10425 24.00787 25.33397
## 2021.173      24.68433 24.25100 25.11766 24.02128 25.34739
## 2021.192      24.69499 24.26166 25.12832 24.03194 25.35804
## 2021.212      24.79328 24.35995 25.22661 24.13023 25.45634
## 2021.231      24.75665 24.32332 25.18998 24.09359 25.41970
## 2021.250      24.83837 24.40504 25.27170 24.17531 25.50142
## 2021.269      24.82758 24.39425 25.26091 24.16453 25.49063
## 2021.288      24.77328 24.33995 25.20661 24.11023 25.43634
## 2021.308      24.85462 24.42129 25.28795 24.19156 25.51767
## 2021.327      24.91436 24.48104 25.34769 24.25131 25.57742
## 2021.346      24.85562 24.42229 25.28895 24.19256 25.51867
## 2021.365      24.82874 24.39541 25.26207 24.16569 25.49180
## 2021.385      24.87047 24.43714 25.30380 24.20741 25.53352
## 2021.404      24.67180 24.23847 25.10513 24.00875 25.33486
## 2021.423      24.68351 24.25018 25.11684 24.02046 25.34657
## 2021.442      24.60840 24.17507 25.04173 23.94535 25.27146
## 2021.462      24.64028 24.20695 25.07361 23.97723 25.30334
## 2021.481      24.56644 24.13311 24.99977 23.90339 25.22950
## 2021.500      24.54641 24.11308 24.97974 23.88335 25.20946
## 2021.519      24.43417 24.00084 24.86750 23.77112 25.09723
## 2021.538      24.36886 23.93553 24.80219 23.70580 25.03191
## 2021.558      24.38770 23.95437 24.82103 23.72465 25.05076
## 2021.577      24.41513 23.98180 24.84846 23.75207 25.07818
## 2021.596      24.16261 23.72928 24.59594 23.49955 24.82566
## 2021.615      24.22035 23.78702 24.65368 23.55729 24.88340
## 2021.635      24.39576 23.96243 24.82909 23.73271 25.05881
## 2021.654      24.35330 23.91997 24.78663 23.69024 25.01635
## 2021.673      24.26404 23.83071 24.69737 23.60098 24.92709
## 2021.692      24.23649 23.80316 24.66982 23.57344 24.89955
## 2021.712      24.12363 23.69030 24.55696 23.46058 24.78669
## 2021.731      24.19332 23.75999 24.62665 23.53027 24.85637
## 2021.750      24.17063 23.73730 24.60396 23.50758 24.83369
## 2021.769      24.18128 23.74795 24.61461 23.51823 24.84434
## 2021.788      24.18073 23.74740 24.61406 23.51768 24.84379
## 2021.808      24.10536 23.67203 24.53869 23.44230 24.76841
## 2021.827      24.07347 23.64014 24.50680 23.41042 24.73653
## 2021.846      24.01429 23.58096 24.44762 23.35124 24.67734
## 2021.865      24.17209 23.73876 24.60542 23.50904 24.83515
## 2021.885      24.24588 23.81255 24.67921 23.58283 24.90893
## 2021.904      24.32280 23.88947 24.75613 23.65974 24.98585
## 2021.923      24.29679 23.86346 24.73012 23.63374 24.95984
## 2021.942      24.35968 23.92635 24.79301 23.69662 25.02273
## 2021.962      24.43463 24.00130 24.86796 23.77158 25.09769
## 2021.981      24.45538 24.02205 24.88871 23.79233 25.11843
## 2022.000      24.48591 24.05258 24.91924 23.82286 25.14896
## 2022.019      24.49705 24.06372 24.93038 23.83400 25.16010

```

```

## 2022.038      24.55800 24.12502 24.99097 23.89549 25.22051
## 2022.058      24.58437 24.15140 25.01735 23.92186 25.24688
## 2022.077      24.63507 24.20210 25.06804 23.97256 25.29758

```

PLOO All: Imputed - Excluding External Variables (Forecasting PH)

```

# PLOO All: Imputed - Training partition (01/18/1999-01/04/2021)
#owt_df02_gb09_ploo_wkly03_train_tb03
# PLOO All: Imputed - Test partition (01/11/2021-01/03/2022)
#owt_df02_gb09_ploo_wkly03_test_tb03

owt_df02_gb09_ploo_wkly03_train_tb03 %>%
  plot_time_series(date_sample,
                    PH)

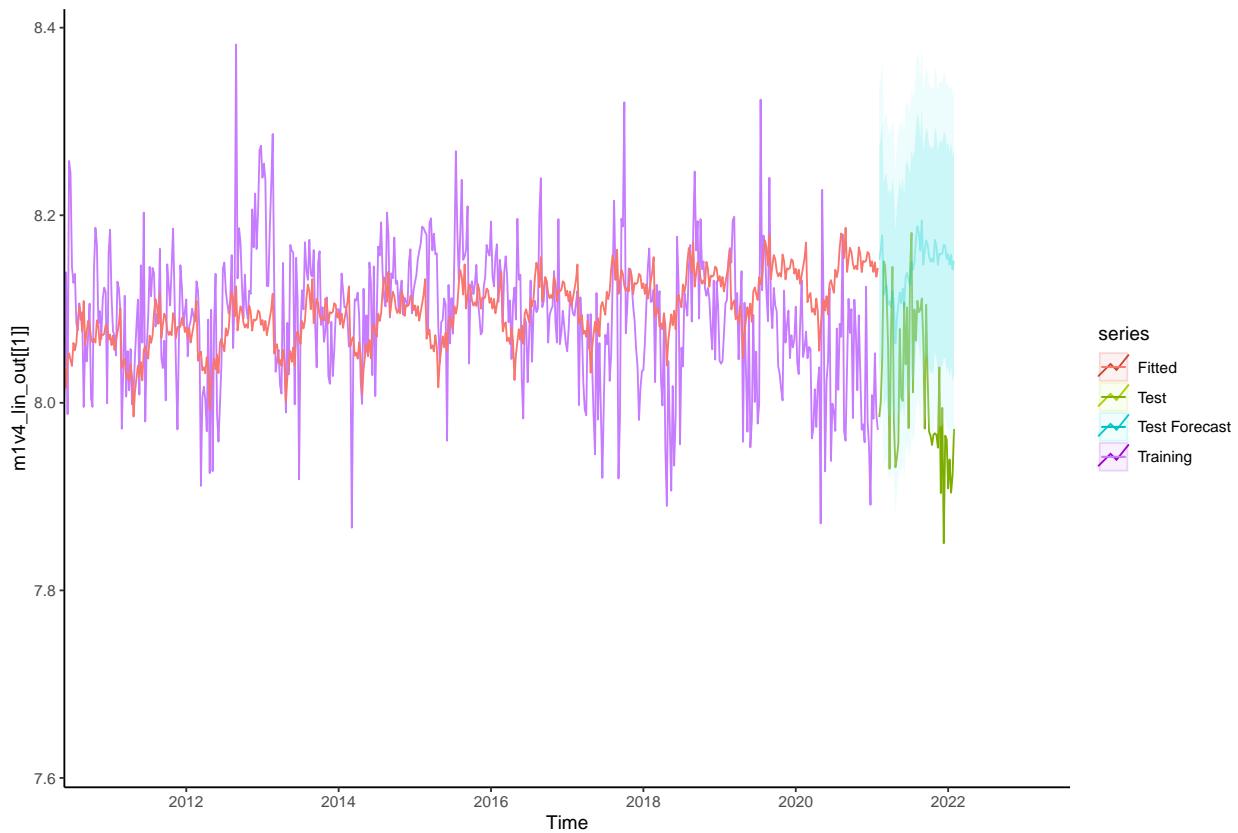
owt_df02_gb09_ploo_wkly03_train_tb03 %>%
  plot_stl_diagnostics(date_sample,
                        PH,
                        .frequency = "auto",
                        .trend = "auto")

m1v4_lin_out <- lr_ts_mod(tb_train = owt_df02_gb09_ploo_wkly03_train_tb03,
                            tb_test = owt_df02_gb09_ploo_wkly03_test_tb03,
                            date = "date_sample",
                            param = "PH",
                            ext_pred = c(),
                            train_start = c(1999, lubridate::isoweek(ymd(train_start))),
                            test_start = c(2021, 6),
                            freq = 52,
                            h = 52,
                            formula = "PH ~ trend + season")

##               ME        RMSE       MAE       MPE       MAPE       MASE
## Training set -9.269220e-18 0.08929282 0.07080095 -0.01237585 0.8807666 1.277369
## Test set      -1.245855e-01 0.15000693 0.12993314 -1.56348438 1.6290748 2.344214
##               ACF1
## Training set 0.6564464
## Test set     NA

autoplot(m1v4_lin_out[[1]], series = 'Training') +
  autolayer(m1v4_lin_out[[2]], series = 'Test') +
  autolayer(m1v4_lin_out[[3]], series = 'Test Forecast', alpha = .3) +
  autolayer(m1v4_lin_out[[3]]$fitted, series = 'Fitted') +
  theme_classic() +
  coord_cartesian(#ylim = c(-50, 400),
                 xlim = c(2011, 2023))

```



```
summary(m1v4_lin_out[[3]])
```

```
##
## Forecast method: Linear regression model
##
## Model Information:
##
## Call:
## tslm(formula = as.formula(formula), data = train_ts01)
##
## Coefficients:
## (Intercept)      trend    season2    season3    season4    season5
## 7.9792102  0.0001500 -0.0113925 -0.0012053 -0.0166947 -0.0075052
##   season6    season7    season8    season9    season10   season11
## 0.0011960  0.0106141  0.0269733 -0.0367986 -0.0302831 -0.0273073
##   season12   season13   season14   season15   season16   season17
## -0.0475963 -0.0444153 -0.0510328 -0.0338475 -0.0473597 -0.0897882
##   season18   season19   season20   season21   season22   season23
## -0.0631879 -0.0492182 -0.0359508 -0.0476054 -0.0322878 -0.0521070
##   season24   season25   season26   season27   season28   season29
## -0.0200069 -0.0156886 -0.0228587 -0.0294771 -0.0050681 -0.0130589
##   season30   season31   season32   season33   season34   season35
## -0.0052088  0.0185250  0.0330742  0.0258301  0.0065064  0.0386156
##   season36   season37   season38   season39   season40   season41
## -0.0086332  0.0050999  0.0168375  0.0122134 -0.0006408 -0.0062569
##   season42   season43   season44   season45   season46   season47
```

```

## -0.0029286 -0.0069489 0.0165172 0.0095988 -0.0102944 0.0014298
## season48 season49 season50 season51 season52
## 0.0001438 0.0014341 0.0105906 0.0085482 -0.0049809
##
##
## Error measures:
## ME RMSE MAE MPE MAPE MASE
## Training set -9.26922e-18 0.08929282 0.07080095 -0.01237585 0.8807666 0.8898097
## ACF1
## Training set 0.6564464
##
## Forecasts:
## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95
## 2021.096 8.152664 8.032622 8.272705 7.968984 8.336344
## 2021.115 8.162232 8.042190 8.282273 7.978552 8.345912
## 2021.135 8.178741 8.058699 8.298782 7.995061 8.362421
## 2021.154 8.115119 7.995078 8.235161 7.931439 8.298799
## 2021.173 8.121785 8.001743 8.241826 7.938105 8.305465
## 2021.192 8.124910 8.004869 8.244952 7.941230 8.308590
## 2021.212 8.104772 7.984730 8.224813 7.921092 8.288452
## 2021.231 8.108103 7.988061 8.228144 7.924423 8.291783
## 2021.250 8.101635 7.981594 8.221677 7.917955 8.285315
## 2021.269 8.118970 7.998929 8.239012 7.935290 8.302650
## 2021.288 8.105608 7.985567 8.225650 7.921928 8.289288
## 2021.308 8.063330 7.943288 8.183371 7.879650 8.247010
## 2021.327 8.090080 7.970039 8.210122 7.906400 8.273760
## 2021.346 8.104200 7.984158 8.224242 7.920520 8.287880
## 2021.365 8.117617 7.997576 8.237659 7.933937 8.301297
## 2021.385 8.106113 7.986071 8.226154 7.922433 8.289793
## 2021.404 8.121581 8.001539 8.241622 7.937901 8.305261
## 2021.423 8.101911 7.981870 8.221953 7.918231 8.285591
## 2021.442 8.134162 8.014120 8.254203 7.950482 8.317842
## 2021.462 8.138630 8.018588 8.258671 7.954950 8.322310
## 2021.481 8.131610 8.011568 8.251651 7.947930 8.315290
## 2021.500 8.125141 8.005100 8.245183 7.941461 8.308821
## 2021.519 8.149701 8.029659 8.269742 7.966021 8.333381
## 2021.538 8.141860 8.021818 8.261901 7.958180 8.325540
## 2021.558 8.149860 8.029818 8.269901 7.966180 8.333540
## 2021.577 8.173744 8.053702 8.293785 7.990064 8.357424
## 2021.596 8.188443 8.068402 8.308485 8.004763 8.372123
## 2021.615 8.181349 8.061307 8.301391 7.997669 8.365029
## 2021.635 8.162175 8.042134 8.282217 7.978495 8.345855
## 2021.654 8.194435 8.074393 8.314476 8.010755 8.378115
## 2021.673 8.147336 8.027294 8.267377 7.963656 8.331016
## 2021.692 8.161219 8.041177 8.281261 7.977539 8.344899
## 2021.712 8.173107 8.053065 8.293148 7.989427 8.356787
## 2021.731 8.168633 8.048591 8.288674 7.984953 8.352313
## 2021.750 8.155928 8.035887 8.275970 7.972248 8.339608
## 2021.769 8.150462 8.030421 8.270504 7.966782 8.334142
## 2021.788 8.153941 8.033899 8.273982 7.970261 8.337621
## 2021.808 8.150070 8.030029 8.270112 7.966390 8.333750
## 2021.827 8.173687 8.053645 8.293728 7.990007 8.357367
## 2021.846 8.166918 8.046877 8.286960 7.983238 8.350598
## 2021.865 8.147175 8.027134 8.267217 7.963495 8.330855

```

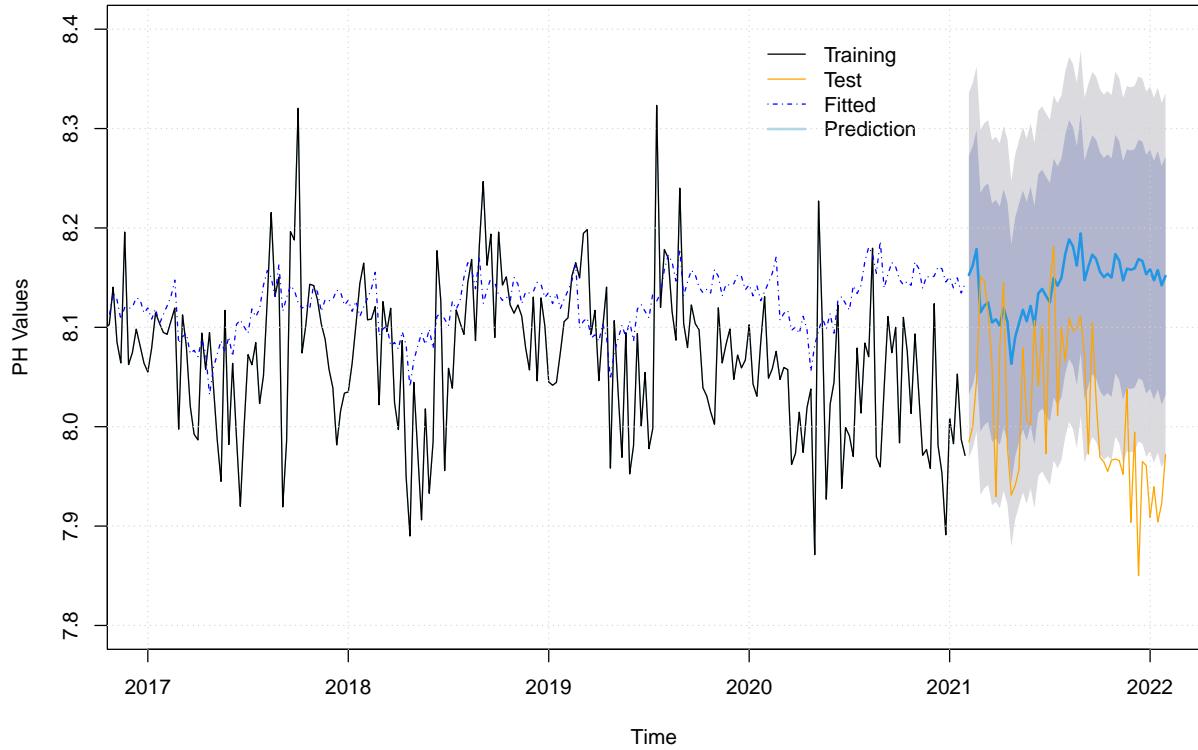
```

## 2021.885      8.159049 8.039008 8.279091 7.975369 8.342729
## 2021.904      8.157913 8.037872 8.277955 7.974233 8.341593
## 2021.923      8.159354 8.039312 8.279395 7.975674 8.343034
## 2021.942      8.168660 8.048619 8.288702 7.984980 8.352340
## 2021.962      8.166768 8.046727 8.286810 7.983088 8.350448
## 2021.981      8.153389 8.033347 8.273430 7.969709 8.337069
## 2022.000      8.158520 8.038478 8.278561 7.974840 8.342200
## 2022.019      8.147277 8.027236 8.267319 7.963597 8.330957
## 2022.038      8.157615 8.037672 8.277557 7.974086 8.341144
## 2022.058      8.142275 8.022333 8.262218 7.958746 8.325804
## 2022.077      8.151615 8.031672 8.271558 7.968086 8.335144

plot(m1v4_lin_out[[3]],
      xlim = c(2017, 2022.05),
      ylim = c(7.8, 8.4),
      xlab = "Time",
      ylab = "PH Values",
      main = "Predicted PH Levels Over Time - PLOO",
      col = "light blue")
lines(m1v4_lin_out[[1]], lty=1, col="black")
lines(m1v4_lin_out[[2]], lty=1, col="orange")
lines(m1v4_lin_out[[3]]$fitted, lty=4, col="blue")
legend(2020, 8.4,
       c("Training", "Test", "Fitted", "Prediction"),
       lty = c(1,1,4,1),
       lwd = c(1,1,1,2),
       col = c("black", "orange", "blue", "light blue"),
       bty = "n")
grid()

```

Predicted PH Levels Over Time – PLOO



PLOO Excluding Outliers: Imputed - Excluding External Variables (Forecasting PH)

```
# PLOO Excluding Outliers: Imputed - Training partition (01/18/1999-01/04/2021)
#owt_df02_gb09_ploo_wkly02_train_tb03
# PLOO Excluding Outliers: Imputed - Test partition (01/11/2021-01/03/2022)
#owt_df02_gb09_ploo_wkly02_test_tb03

owt_df02_gb09_ploo_wkly02_train_tb03 %>%
  plot_time_series(date_sample,
    PH)

owt_df02_gb09_ploo_wkly02_train_tb03 %>%
  plot_stl_diagnostics(date_sample,
    PH,
    .frequency = "auto",
    .trend = "auto")

m2v4_lin_out <- lr_ts_mod(tb_train = owt_df02_gb09_ploo_wkly02_train_tb03,
  tb_test = owt_df02_gb09_ploo_wkly02_test_tb03,
  date = "date_sample",
  param = "PH",
  ext_pred = c(),
  train_start = c(1999, lubridate::isoweek(ymd(train_start))),
  test_start = c(2021, 6),
  freq = 52,
```

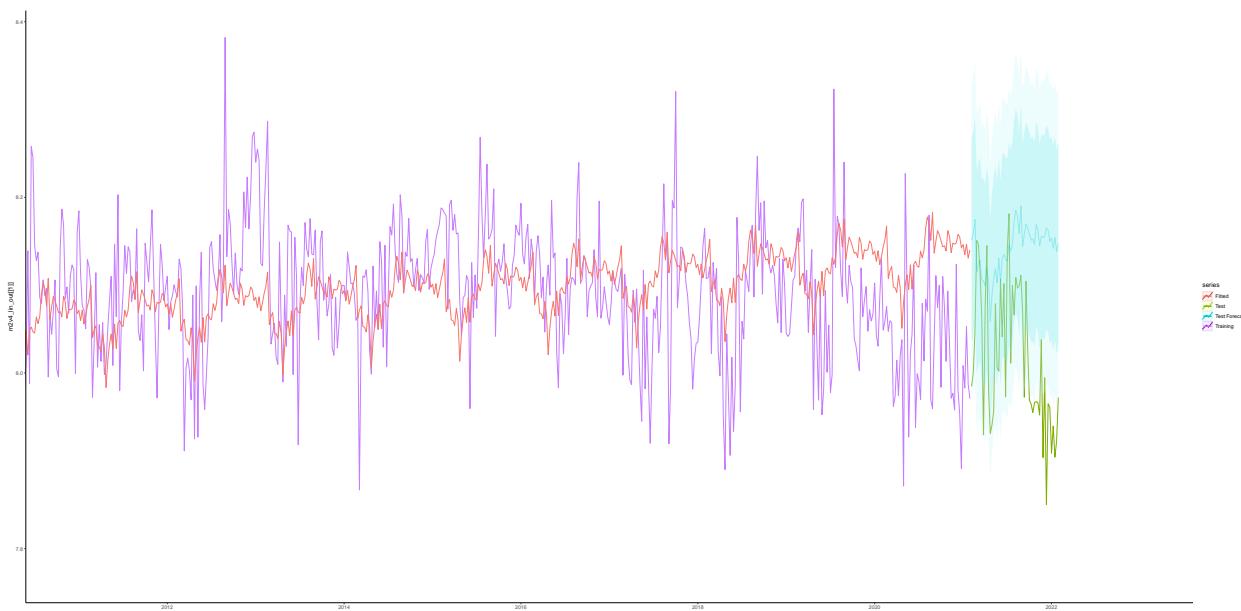
```

            h = 52,
            formula = "PH ~ trend + season")

##               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -1.104657e-17 0.08640738 0.06927462 -0.01155372 0.8610721 1.276506
## Test set     -1.225481e-01 0.14774555 0.12779428 -1.53798322 1.6023184 2.354833
## ACF1
## Training set 0.6478792
## Test set      NA

autoplot(m2v4_lin_out[[1]], series = 'Training') +
  autolayer(m2v4_lin_out[[2]], series = 'Test') +
  autolayer(m2v4_lin_out[[3]], series = 'Test Forecast', alpha = .3) +
  autolayer(m2v4_lin_out[[3]]$fitted, series = 'Fitted') +
  theme_classic() +
  coord_cartesian(#ylim = c(-50, 400),
                 xlim = c(2011, 2023))

```



```
summary(m2v4_lin_out[[3]])
```

```

##
## Forecast method: Linear regression model
##
## Model Information:
##
## Call:
## tslm(formula = as.formula(formula), data = train_ts01)
##
## Coefficients:
## (Intercept)      trend      season2      season3      season4      season5
## 7.9815006    0.0001441   -0.0116517    0.0002109   -0.0158001   -0.0065501
## season6      season7      season8      season9      season10     season11
## 0.0040766    0.0117627    0.0276796   -0.0322873   -0.0231654   -0.0189713
## season12     season13     season14     season15     season16     season17

```

```

## -0.0423245 -0.0432401 -0.0487886 -0.0294078 -0.0438132 -0.0900486
## season18 season19 season20 season21 season22 season23
## -0.0600143 -0.0412534 -0.0297337 -0.0466314 -0.0185566 -0.0464367
## season24 season25 season26 season27 season28 season29
## -0.0178877 -0.0149252 -0.0204894 -0.0219769 -0.0037518 -0.0115109
## season30 season31 season32 season33 season34 season35
## -0.0034972 0.0222177 0.0346162 0.0270503 0.0076727 0.0390644
## season36 season37 season38 season39 season40 season41
## -0.0071824 0.0062641 0.0181014 0.0131747 0.0064554 -0.0006943
## season42 season43 season44 season45 season46 season47
## 0.0003004 -0.0057324 0.0171976 0.0105407 -0.0081395 0.0026749
## season48 season49 season50 season51 season52
## 0.0012681 0.0024788 0.0121054 0.0091062 -0.0034408
##
##
## Error measures:
## ME RMSE MAE MPE MAPE
## Training set -1.104657e-17 0.08640738 0.06927462 -0.01155372 0.8610721
## MASE ACF1
## Training set 0.8953634 0.6478792
##
## Forecasts:
## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95
## 2021.096 8.150967 8.034804 8.267129 7.973222 8.328711
## 2021.115 8.158797 8.042635 8.274960 7.981053 8.336542
## 2021.135 8.174858 8.058696 8.291021 7.997114 8.352603
## 2021.154 8.115035 7.998873 8.231198 7.937291 8.292780
## 2021.173 8.124301 8.008139 8.240464 7.946557 8.302046
## 2021.192 8.128639 8.012477 8.244802 7.950895 8.306384
## 2021.212 8.105430 7.989268 8.221593 7.927686 8.283175
## 2021.231 8.104659 7.988496 8.220821 7.926914 8.282403
## 2021.250 8.099254 7.983092 8.215417 7.921510 8.276999
## 2021.269 8.118779 8.002617 8.234942 7.941035 8.296524
## 2021.288 8.104518 7.988355 8.220680 7.926773 8.282262
## 2021.308 8.058426 7.942264 8.174589 7.880682 8.236171
## 2021.327 8.088605 7.972442 8.204767 7.910860 8.266349
## 2021.346 8.107510 7.991347 8.223672 7.929765 8.285254
## 2021.365 8.119174 8.003011 8.235336 7.941429 8.296918
## 2021.385 8.102420 7.986257 8.218582 7.924675 8.280164
## 2021.404 8.130639 8.014476 8.246801 7.952894 8.308383
## 2021.423 8.102903 7.986740 8.219065 7.925158 8.280647
## 2021.442 8.131596 8.015433 8.247758 7.953851 8.309340
## 2021.462 8.134702 8.018540 8.250865 7.956958 8.312447
## 2021.481 8.129282 8.013120 8.245445 7.951538 8.307027
## 2021.500 8.127939 8.011776 8.244101 7.950194 8.305683
## 2021.519 8.146308 8.030146 8.262470 7.968563 8.324052
## 2021.538 8.138693 8.022530 8.254855 7.960948 8.316437
## 2021.558 8.146851 8.030688 8.263013 7.969106 8.324595
## 2021.577 8.172710 8.056547 8.288872 7.994965 8.350454
## 2021.596 8.185252 8.069090 8.301415 8.007508 8.362997
## 2021.615 8.177830 8.061668 8.293993 8.000086 8.355575
## 2021.635 8.158597 8.042434 8.274759 7.980852 8.336341
## 2021.654 8.190133 8.073970 8.306295 8.012388 8.367877
## 2021.673 8.144030 8.027868 8.260192 7.966285 8.321774

```

```

## 2021.692      8.157621 8.041458 8.273783 7.979876 8.335365
## 2021.712      8.169602 8.053439 8.285764 7.991857 8.347346
## 2021.731      8.164819 8.048657 8.280982 7.987075 8.342564
## 2021.750      8.158244 8.042082 8.274406 7.980500 8.335989
## 2021.769      8.151238 8.035076 8.267401 7.973494 8.328983
## 2021.788      8.152377 8.036215 8.268540 7.974633 8.330122
## 2021.808      8.146488 8.030326 8.262651 7.968744 8.324233
## 2021.827      8.169563 8.053400 8.285725 7.991818 8.347307
## 2021.846      8.163050 8.046887 8.279212 7.985305 8.340794
## 2021.865      8.144514 8.028351 8.260676 7.966769 8.322258
## 2021.885      8.155472 8.039310 8.271634 7.977727 8.333216
## 2021.904      8.154209 8.038047 8.270372 7.976465 8.331954
## 2021.923      8.155564 8.039402 8.271726 7.977820 8.333309
## 2021.942      8.165335 8.049172 8.281497 7.987590 8.343079
## 2021.962      8.162480 8.046317 8.278642 7.984735 8.340224
## 2021.981      8.150077 8.033914 8.266239 7.972332 8.327821
## 2022.000      8.153662 8.037499 8.269824 7.975917 8.331406
## 2022.019      8.142154 8.025991 8.258316 7.964409 8.319898
## 2022.038      8.154161 8.038094 8.270227 7.976562 8.331759
## 2022.058      8.138294 8.022227 8.254361 7.960695 8.315892
## 2022.077      8.147688 8.031621 8.263755 7.970089 8.325286

```

SBOO All: Imputed - Excluding External Variables (Forecasting PH)

```

# SBOO All: Imputed - Training partition (01/18/1999-01/04/2021)
#owt_df02_gb09_sboo_wkly03_train_tb03
# SBOO All: Imputed - Test partition (01/11/2021-01/03/2022)
#owt_df02_gb09_sboo_wkly03_test_tb03

owt_df02_gb09_sboo_wkly03_train_tb03 %>%
  plot_time_series(date_sample,
                    PH)

owt_df02_gb09_sboo_wkly03_train_tb03 %>%
  plot_stl_diagnostics(date_sample,
                        PH,
                        .frequency = "auto",
                        .trend = "auto")

m3v4_lin_out <- lr_ts_mod(tb_train = owt_df02_gb09_sboo_wkly03_train_tb03,
                            tb_test = owt_df02_gb09_sboo_wkly03_test_tb03,
                            date = "date_sample",
                            param = "PH",
                            ext_pred = c(),
                            train_start = c(1999, lubridate::isoweek(ymd(train_start))),
                            test_start = c(2021, 6),
                            freq = 52,
                            h = 52,
                            formula = "PH ~ trend + season")

##               ME        RMSE       MAE       MPE       MAPE
## Training set 7.593386e-17 0.07816866 0.06033009 -0.009313191 0.7453154
## Test set     -1.079938e-01 0.15322399 0.12694409 -1.358164499 1.5867912
##               MASE      ACF1

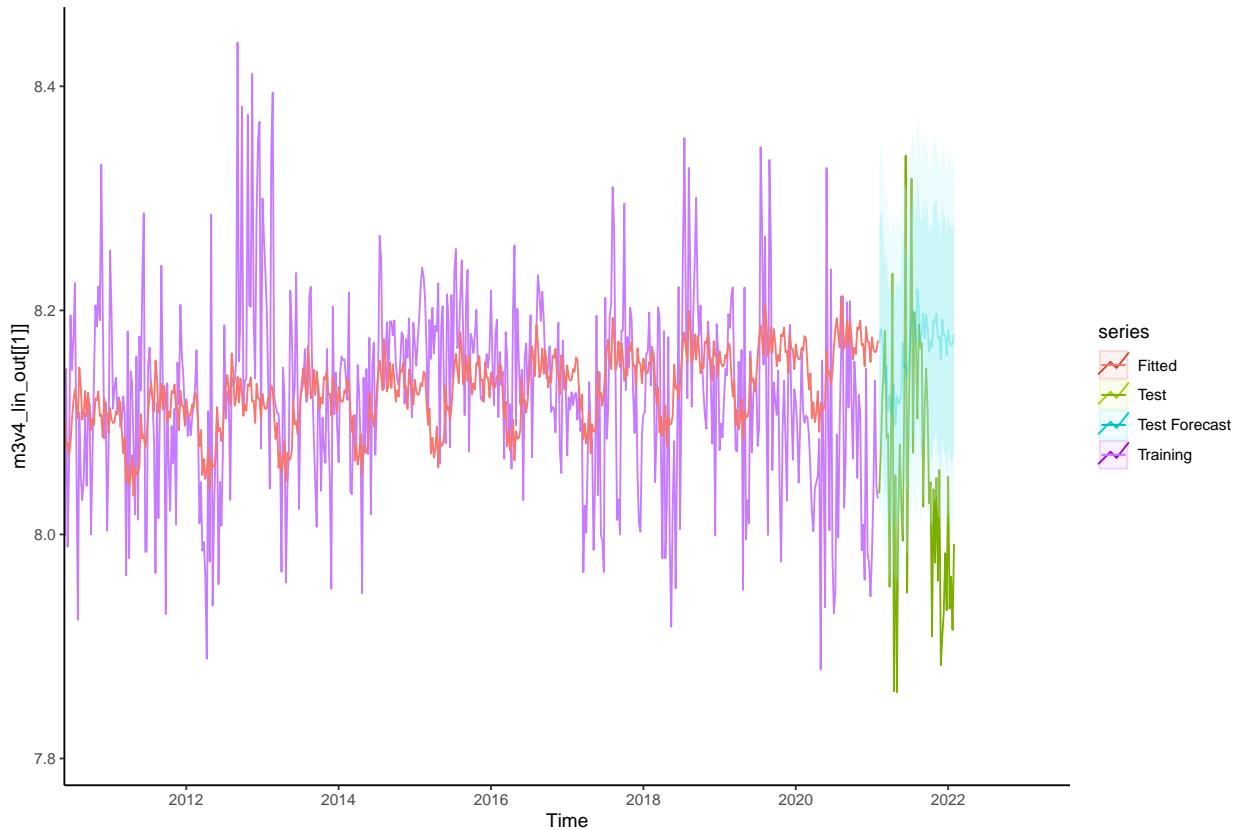
```

```

## Training set 0.9956236 0.4193761
## Test set      2.0949500      NA

autoplot(m3v4_lin_out[[1]], series = 'Training') +
  autolayer(m3v4_lin_out[[2]], series = 'Test') +
  autolayer(m3v4_lin_out[[3]], series = 'Test Forecast', alpha = .3) +
  autolayer(m3v4_lin_out[[3]]$fitted, series = 'Fitted') +
  theme_classic() +
  coord_cartesian(#ylim = c(-50, 400),
                 xlim = c(2011, 2023))

```



```
summary(m3v4_lin_out[[3]])
```

```

##
## Forecast method: Linear regression model
##
## Model Information:
##
## Call:
## tslm(formula = as.formula(formula), data = train_ts01)
##
## Coefficients:
## (Intercept)      trend      season2      season3      season4      season5
## 8.0368835    0.0001217   -0.0103065   -0.0136633   -0.0129479   -0.0035307
##   season6      season7      season8      season9      season10     season11

```

```

## -0.0063173 0.0066232 0.0039769 -0.0213799 -0.0413630 -0.0270645
## season12 season13 season14 season15 season16 season17
## -0.0672929 -0.0532560 -0.0710262 -0.0631263 -0.0438243 -0.0801844
## season18 season19 season20 season21 season22 season23
## -0.0543537 -0.0611338 -0.0606209 -0.0535703 -0.0078879 -0.0297383
## season24 season25 season26 season27 season28 season29
## -0.0269168 -0.0386381 -0.0276928 -0.0067797 0.0094032 0.0198705
## season30 season31 season32 season33 season34 season35
## 0.0058349 -0.0066782 0.0388753 0.0223285 -0.0080427 -0.0024534
## season36 season37 season38 season39 season40 season41
## 0.0173590 -0.0054138 0.0155995 0.0023678 -0.0180221 -0.0080228
## season42 season43 season44 season45 season46 season47
## -0.0136630 0.0099554 0.0081485 0.0158650 -0.0061413 -0.0052233
## season48 season49 season50 season51 season52
## -0.0253460 0.0096940 -0.0036287 -0.0043141 -0.0220040
##
##
## Error measures:
## ME RMSE MAE MPE MAPE
## Training set 7.593386e-17 0.07816866 0.06033009 -0.009313191 0.7453154
## MASE ACF1
## Training set 0.7376559 0.4193761
##
## Forecasts:
## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95
## 2021.096 8.170310 8.065223 8.275397 8.009513 8.331107
## 2021.115 8.183372 8.078286 8.288459 8.022575 8.344169
## 2021.135 8.180848 8.075761 8.285934 8.020051 8.341645
## 2021.154 8.155613 8.050526 8.260699 7.994815 8.316410
## 2021.173 8.135751 8.030664 8.240838 7.974954 8.296548
## 2021.192 8.150171 8.045085 8.255258 7.989374 8.310968
## 2021.212 8.110065 8.004978 8.215151 7.949268 8.270862
## 2021.231 8.124223 8.019137 8.229310 7.963426 8.285020
## 2021.250 8.106575 8.001488 8.211662 7.945778 8.267372
## 2021.269 8.114596 8.009510 8.219683 7.953799 8.275393
## 2021.288 8.134020 8.028934 8.239107 7.973223 8.294817
## 2021.308 8.097782 7.992695 8.202869 7.936985 8.258579
## 2021.327 8.123734 8.018648 8.228821 7.962937 8.284531
## 2021.346 8.117076 8.011989 8.222163 7.956279 8.277873
## 2021.365 8.117711 8.012624 8.222797 7.956914 8.278508
## 2021.385 8.124883 8.019796 8.229969 7.964086 8.285680
## 2021.404 8.170687 8.065600 8.275774 8.009890 8.331484
## 2021.423 8.148958 8.043872 8.254045 7.988161 8.309755
## 2021.442 8.151902 8.046815 8.256988 7.991104 8.312699
## 2021.462 8.140302 8.035215 8.245389 7.979505 8.301099
## 2021.481 8.151369 8.046282 8.256456 7.990572 8.312166
## 2021.500 8.172404 8.067317 8.277490 8.011607 8.333201
## 2021.519 8.188708 8.083622 8.293795 8.027911 8.349505
## 2021.538 8.199297 8.094211 8.304384 8.038500 8.360094
## 2021.558 8.185384 8.080297 8.290470 8.024587 8.346181
## 2021.577 8.172992 8.067906 8.278079 8.012195 8.333789
## 2021.596 8.218667 8.113581 8.323754 8.057870 8.379464
## 2021.615 8.202242 8.097156 8.307329 8.041445 8.363039
## 2021.635 8.171993 8.066906 8.277080 8.011196 8.332790

```

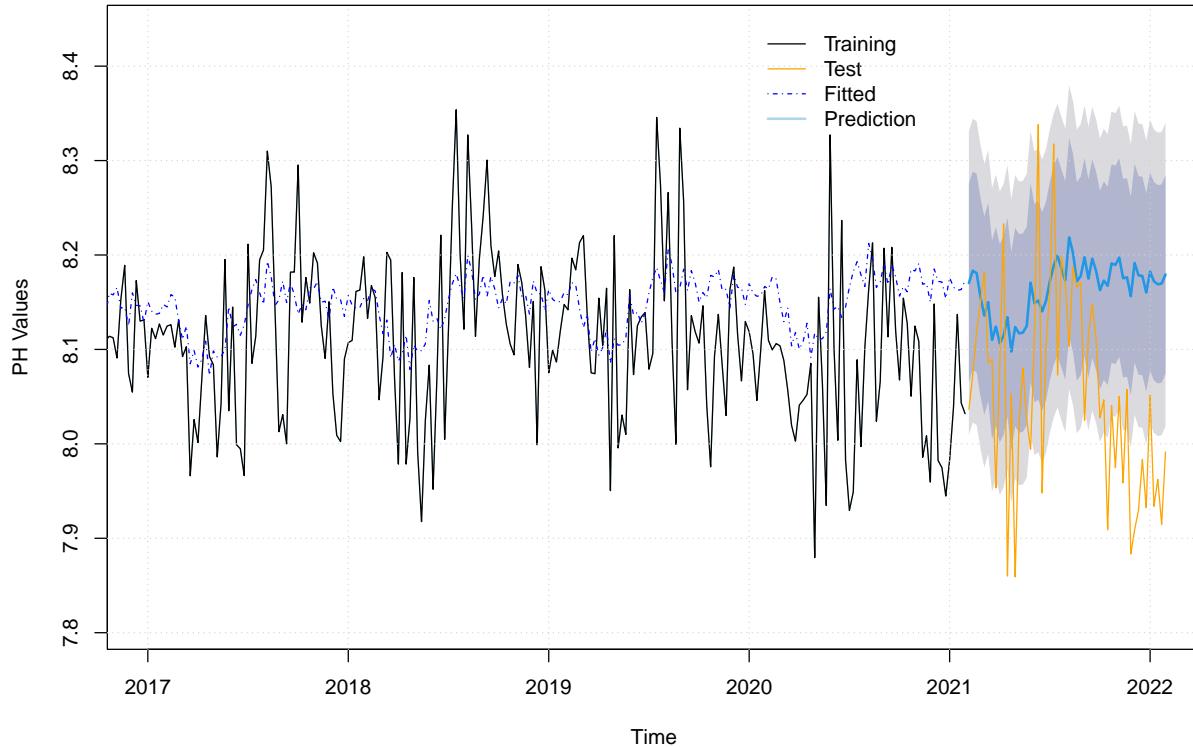
```

## 2021.654      8.177704 8.072617 8.282791 8.016907 8.338501
## 2021.673      8.197638 8.092551 8.302725 8.036841 8.358435
## 2021.692      8.174987 8.069900 8.280074 8.014190 8.335784
## 2021.712      8.196122 8.091035 8.301209 8.035325 8.356919
## 2021.731      8.183012 8.077925 8.288099 8.022215 8.343809
## 2021.750      8.162744 8.057657 8.267831 8.001947 8.323541
## 2021.769      8.172865 8.067778 8.277952 8.012068 8.333662
## 2021.788      8.167346 8.062260 8.272433 8.006549 8.328143
## 2021.808      8.191087 8.086000 8.296173 8.030290 8.351884
## 2021.827      8.189401 8.084315 8.294488 8.028604 8.350198
## 2021.846      8.197240 8.092153 8.302326 8.036443 8.358037
## 2021.865      8.175355 8.070268 8.280442 8.014558 8.336152
## 2021.885      8.176395 8.071308 8.281481 8.015598 8.337192
## 2021.904      8.156394 8.051307 8.261480 7.995597 8.317191
## 2021.923      8.191555 8.086469 8.296642 8.030758 8.352353
## 2021.942      8.178355 8.073268 8.283441 8.017558 8.339152
## 2021.962      8.177791 8.072704 8.282878 8.016994 8.338588
## 2021.981      8.160223 8.055136 8.265309 7.999426 8.321020
## 2022.000      8.182348 8.077262 8.287435 8.021551 8.343145
## 2022.019      8.172164 8.067077 8.277250 8.011367 8.332961
## 2022.038      8.168929 8.063928 8.273929 8.008264 8.329593
## 2022.058      8.169766 8.064765 8.274766 8.009101 8.330431
## 2022.077      8.179305 8.074304 8.284305 8.018640 8.339969

plot(m3v4_lin_out[[3]],
      xlim = c(2017, 2022.05),
      ylim = c(7.8, 8.4),
      xlab = "Time",
      ylab = "PH Values",
      main = "Predicted PH Levels Over Time - SB00",
      col = "light blue")
lines(m3v4_lin_out[[1]], lty=1, col="black")
lines(m3v4_lin_out[[2]], lty=1, col="orange")
lines(m3v4_lin_out[[3]]$fitted, lty=4, col="blue")
legend(2020, 8.45,
       c("Training", "Test", "Fitted", "Prediction"),
       lty = c(1,1,4,1),
       lwd = c(1,1,1,2),
       col = c("black", "orange", "blue", "light blue"),
       bty = "n")
grid()

```

Predicted PH Levels Over Time – SBOO



SBOO Excluding Outliers: Imputed - Excluding External Variables (Forecasting PH)

```
# SBOO Excluding Outliers: Imputed - Training partition (01/18/1999-01/04/2021)
#owt_df02_gb09_sboo_wkly02_train_tb03
# SBOO Excluding Outliers: Imputed - Test partition (01/11/2021-01/03/2022)
#owt_df02_gb09_sboo_wkly02_test_tb03

owt_df02_gb09_sboo_wkly02_train_tb03 %>%
  plot_time_series(date_sample,
    PH)

owt_df02_gb09_sboo_wkly02_train_tb03 %>%
  plot_stl_diagnostics(date_sample,
    PH,
    .frequency = "auto",
    .trend = "auto")

m4v4_lin_out <- lr_ts_mod(tb_train = owt_df02_gb09_sboo_wkly02_train_tb03,
  tb_test = owt_df02_gb09_sboo_wkly02_test_tb03,
  date = "date_sample",
  param = "PH",
  ext_pred = c(),
  train_start = c(1999, lubridate::isoweek(ymd(train_start))),
  test_start = c(2021, 6),
  freq = 52,
```

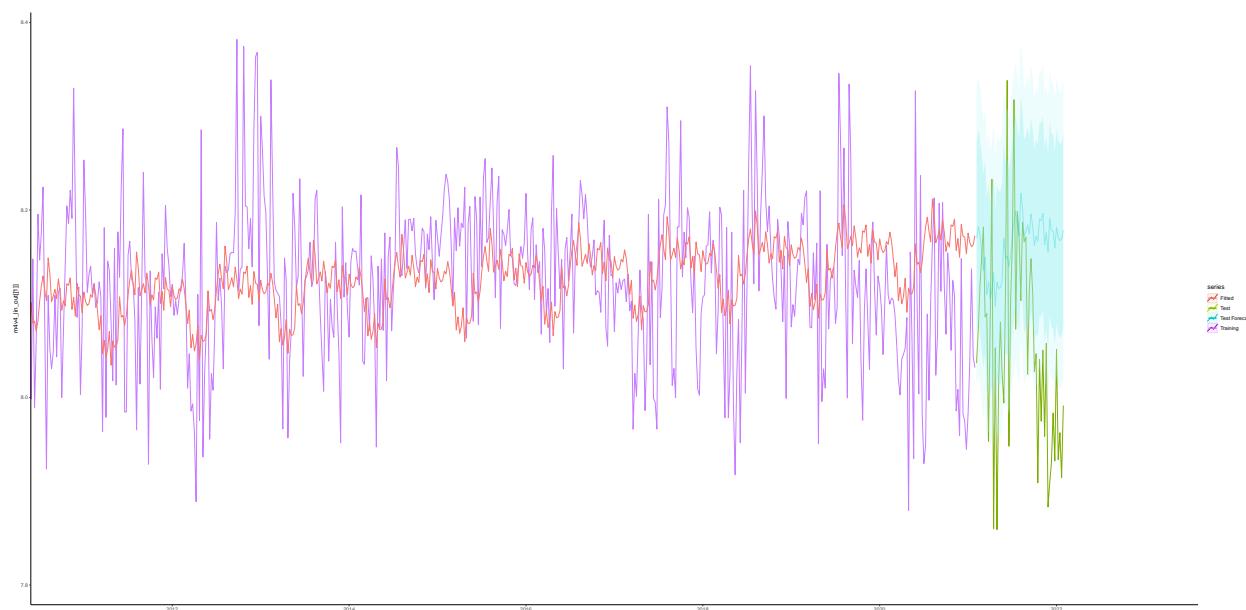
```

          h = 52,
          formula = "PH ~ trend + season")

##               ME        RMSE       MAE       MPE       MAPE
## Training set 5.344694e-17 0.07678678 0.05975517 -0.008998726 0.7384776
## Test set     -1.067714e-01 0.15220261 0.12592061 -1.342966486 1.5740041
##             MASE      ACF1
## Training set 1.002478 0.4212881
## Test set     2.112497      NA

autoplot(m4v4_lin_out[[1]], series = 'Training') +
  autolayer(m4v4_lin_out[[2]], series = 'Test') +
  autolayer(m4v4_lin_out[[3]], series = 'Test Forecast', alpha = .3) +
  autolayer(m4v4_lin_out[[3]]$fitted, series = 'Fitted') +
  theme_classic() +
  coord_cartesian(#ylim = c(-50, 400),
                 xlim = c(2011, 2023))

```



```
summary(m4v4_lin_out[[3]])
```

```

##
## Forecast method: Linear regression model
##
## Model Information:
##
## Call:
## tslm(formula = as.formula(formula), data = train_ts01)
##
## Coefficients:
## (Intercept)      trend      season2      season3      season4      season5
##   8.0374475    0.0001208   -0.0103021   -0.0147150   -0.0130777   -0.0034082
##   season6      season7      season8      season9      season10     season11
##  -0.0063599    0.0064637   -0.0035562   -0.0212836   -0.0413245   -0.0267645
##   season12     season13     season14     season15     season16     season17

```

```

## -0.0671798 -0.0532200 -0.0709409 -0.0631672 -0.0439279 -0.0801473
## season18 season19 season20 season21 season22 season23
## -0.0544994 -0.0612175 -0.0606101 -0.0535892 -0.0078435 -0.0296446
## season24 season25 season26 season27 season28 season29
## -0.0278384 -0.0385905 -0.0276332 -0.0067601 0.0094043 0.0198998
## season30 season31 season32 season33 season34 season35
## 0.0043485 -0.0067627 0.0387892 0.0223371 -0.0080748 -0.0023578
## season36 season37 season38 season39 season40 season41
## 0.0045422 -0.0053459 0.0156831 0.0024328 -0.0180544 -0.0080202
## season42 season43 season44 season45 season46 season47
## -0.0136544 0.0098567 0.0081359 0.0158884 -0.0177065 -0.0052211
## season48 season49 season50 season51 season52
## -0.0252353 0.0095988 -0.0030570 -0.0042321 -0.0220436
##
##
## Error measures:
##               ME      RMSE      MAE      MPE      MAPE
## Training set 5.344694e-17 0.07678678 0.05975517 -0.008998726 0.7384776
##                  MASE      ACF1
## Training set 0.7410219 0.4212881
##
## Forecasts:
##       Point Forecast   Lo 80    Hi 80    Lo 95    Hi 95
## 2021.096     8.169729 8.066500 8.272958 8.011775 8.327684
## 2021.115     8.182674 8.079445 8.285903 8.024719 8.340628
## 2021.135     8.172775 8.069546 8.276003 8.014820 8.330729
## 2021.154     8.155168 8.051939 8.258397 7.997214 8.313122
## 2021.173     8.135248 8.032019 8.238477 7.977293 8.293202
## 2021.192     8.149929 8.046700 8.253157 7.991974 8.307883
## 2021.212     8.109634 8.006405 8.212863 7.951680 8.267588
## 2021.231     8.123715 8.020486 8.226944 7.965760 8.281669
## 2021.250     8.106115 8.002886 8.209343 7.948160 8.264069
## 2021.269     8.114009 8.010780 8.217238 7.956054 8.271963
## 2021.288     8.133369 8.030140 8.236598 7.975415 8.291323
## 2021.308     8.097270 7.994041 8.200499 7.939316 8.255225
## 2021.327     8.123039 8.019810 8.226268 7.965085 8.280993
## 2021.346     8.116442 8.013213 8.219671 7.958487 8.274396
## 2021.365     8.117170 8.013941 8.220399 7.959215 8.275124
## 2021.385     8.124312 8.021083 8.227540 7.966357 8.282266
## 2021.404     8.170178 8.066949 8.273407 8.012224 8.328132
## 2021.423     8.148498 8.045269 8.251727 7.990543 8.306452
## 2021.442     8.150425 8.047196 8.253654 7.992470 8.308379
## 2021.462     8.139793 8.036564 8.243022 7.981839 8.297748
## 2021.481     8.150871 8.047642 8.254100 7.992917 8.308826
## 2021.500     8.171865 8.068636 8.275094 8.013911 8.329820
## 2021.519     8.188150 8.084922 8.291379 8.030196 8.346105
## 2021.538     8.198767 8.095538 8.301996 8.040812 8.356721
## 2021.558     8.183336 8.080107 8.286565 8.025382 8.341291
## 2021.577     8.172346 8.069117 8.275575 8.014391 8.330300
## 2021.596     8.218018 8.114790 8.321247 8.060064 8.375973
## 2021.615     8.201687 8.098458 8.304916 8.043733 8.359641
## 2021.635     8.171396 8.068167 8.274625 8.013442 8.329350
## 2021.654     8.177234 8.074005 8.280463 8.019279 8.335188
## 2021.673     8.184254 8.081026 8.287483 8.026300 8.342209

```

```

## 2021.692      8.174487 8.071258 8.277716 8.016533 8.332442
## 2021.712      8.195637 8.092408 8.298866 8.037682 8.353591
## 2021.731      8.182507 8.079279 8.285736 8.024553 8.340462
## 2021.750      8.162141 8.058912 8.265370 8.004186 8.320095
## 2021.769      8.172296 8.069067 8.275525 8.014342 8.330250
## 2021.788      8.166782 8.063554 8.270011 8.008828 8.324737
## 2021.808      8.190414 8.087185 8.293643 8.032460 8.348369
## 2021.827      8.188814 8.085585 8.292043 8.030860 8.346769
## 2021.846      8.196688 8.093459 8.299917 8.038733 8.354642
## 2021.865      8.163214 8.059985 8.266442 8.005259 8.321168
## 2021.885      8.175820 8.072591 8.279049 8.017865 8.333774
## 2021.904      8.155926 8.052697 8.259155 7.997972 8.313881
## 2021.923      8.190881 8.087652 8.294110 8.032927 8.348835
## 2021.942      8.178346 8.075117 8.281575 8.020392 8.336301
## 2021.962      8.177292 8.074063 8.280521 8.019337 8.335246
## 2021.981      8.159601 8.056372 8.262830 8.001647 8.317555
## 2022.000      8.181765 8.078536 8.284994 8.023811 8.339720
## 2022.019      8.171584 8.068355 8.274813 8.013630 8.329538
## 2022.038      8.167292 8.064148 8.270436 8.009467 8.325116
## 2022.058      8.169050 8.065906 8.272194 8.011225 8.326875
## 2022.077      8.178840 8.075696 8.281984 8.021016 8.336665

```

PLOO All: Imputed - Excluding External Variables (Forecasting SALINITY)

```

# PLOO All: Imputed - Training partition (01/18/1999-01/04/2021)
#owt_df02_gb09_ploo_wkly03_train_tb03
# PLOO All: Imputed - Test partition (01/11/2021-01/03/2022)
#owt_df02_gb09_ploo_wkly03_test_tb03

owt_df02_gb09_ploo_wkly03_train_tb03 %>%
  plot_time_series(date_sample,
                    SALINITY)

owt_df02_gb09_ploo_wkly03_train_tb03 %>%
  plot_stl_diagnostics(date_sample,
                        SALINITY,
                        .frequency = "auto",
                        .trend = "auto")

m1v5_lin_out <- lr_ts_mod(tb_train = owt_df02_gb09_ploo_wkly03_train_tb03,
                            tb_test = owt_df02_gb09_ploo_wkly03_test_tb03,
                            date = "date_sample",
                            param = "SALINITY",
                            ext_pred = c(),
                            train_start = c(1999, lubridate::isoweek(ymd(train_start))),
                            test_start = c(2021, 6),
                            freq = 52,
                            h = 52,
                            formula = "SALINITY ~ trend + season")

##               ME        RMSE       MAE       MPE       MAPE       MASE
## Training set -1.858927e-17 0.1494234 0.11208469 -0.002000979 0.3348794 2.384824

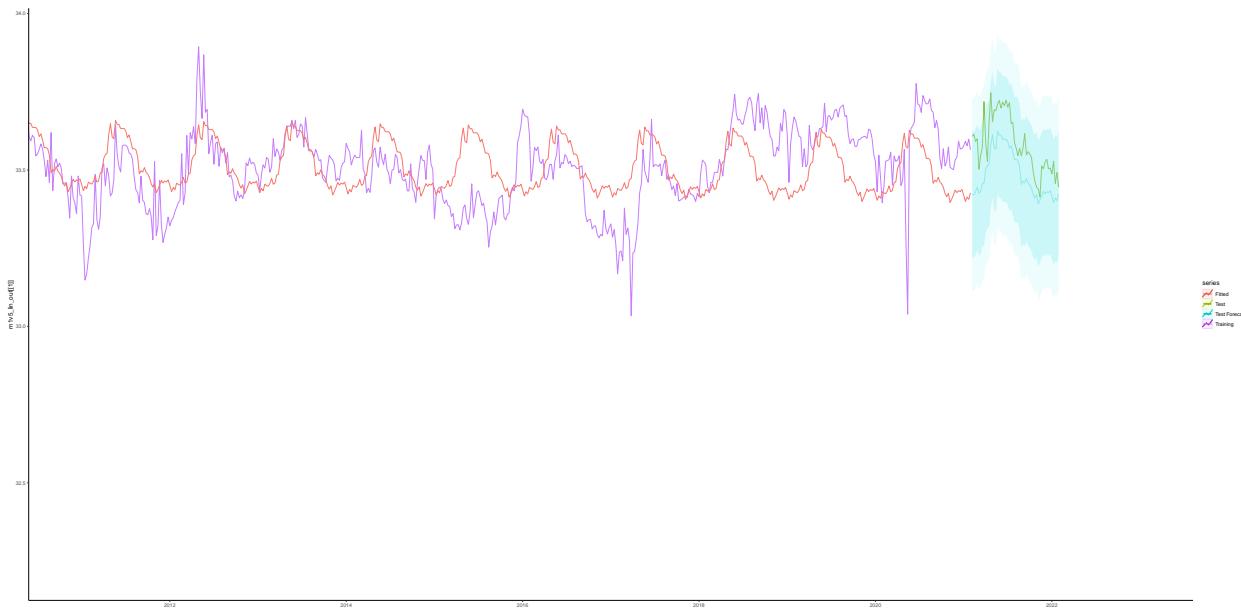
```

```

## Test set      9.563622e-02 0.1072698 0.09563622 0.284563198 0.2845632 2.034850
##          ACF1
## Training set 0.852831
## Test set      NA

autoplot(m1v5_lin_out[[1]], series = 'Training') +
  autolayer(m1v5_lin_out[[2]], series = 'Test') +
  autolayer(m1v5_lin_out[[3]], series = 'Test Forecast', alpha = .3) +
  autolayer(m1v5_lin_out[[3]]$fitted, series = 'Fitted') +
  theme_classic() +
  coord_cartesian(#ylim = c(-50, 400),
                  xlim = c(2011, 2023))

```



```
summary(m1v5_lin_out[[3]])
```

```

##
## Forecast method: Linear regression model
##
## Model Information:
##
## Call:
## tslm(formula = as.formula(formula), data = train_ts01)
##
## Coefficients:
## (Intercept)      trend      season2      season3      season4      season5
##   3.350e+01    -6.594e-05   -2.002e-02   -6.261e-03   -1.336e-02   7.450e-03
##   season6      season7      season8      season9      season10     season11
##   3.597e-03    1.405e-03   1.015e-02   2.636e-02   7.719e-03   1.171e-02
##   season12     season13     season14     season15     season16     season17
##   3.727e-02    3.966e-02   8.288e-02   9.583e-02   9.990e-02   1.622e-01
##   season18     season19     season20     season21     season22     season23
##   1.949e-01    1.530e-01   1.478e-01   2.052e-01   1.929e-01   1.929e-01
##   season24     season25     season26     season27     season28     season29
##   1.803e-01    1.799e-01   1.803e-01   1.726e-01   1.514e-01   1.596e-01

```

```

##    season30      season31      season32      season33      season34      season35
## 1.395e-01 1.156e-01 1.160e-01 1.130e-01 9.126e-02 3.638e-02
##    season36      season37      season38      season39      season40      season41
## 4.601e-02 4.128e-02 5.585e-02 4.119e-02 3.530e-02 2.417e-02
##    season42      season43      season44      season45      season46      season47
## 4.865e-03 -5.698e-03 4.897e-03 -2.422e-02 -1.326e-02 8.850e-04
##    season48      season49      season50      season51      season52
## 1.571e-02 7.648e-03 1.407e-02 1.089e-02 1.741e-02
##
##
## Error measures:
##               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -1.858927e-17 0.1494234 0.1120847 -0.002000979 0.3348794 0.9109584
##          ACF1
## Training set 0.852831
##
## Forecasts:
##             Point Forecast     Lo 80     Hi 80     Lo 95     Hi 95
## 2021.096      33.42302 33.22214 33.62390 33.11565 33.73039
## 2021.115      33.42077 33.21989 33.62164 33.11339 33.72814
## 2021.135      33.42944 33.22856 33.63032 33.12207 33.73681
## 2021.154      33.44559 33.24471 33.64647 33.13822 33.75296
## 2021.173      33.42688 33.22600 33.62776 33.11951 33.73425
## 2021.192      33.43081 33.22993 33.63169 33.12344 33.73818
## 2021.212      33.45630 33.25542 33.65718 33.14893 33.76367
## 2021.231      33.45863 33.25775 33.65951 33.15126 33.76600
## 2021.250      33.50178 33.30090 33.70266 33.19441 33.80915
## 2021.269      33.51466 33.31378 33.71554 33.20729 33.82203
## 2021.288      33.51867 33.31779 33.71955 33.21130 33.82604
## 2021.308      33.58090 33.38002 33.78178 33.27353 33.88827
## 2021.327      33.61351 33.41264 33.81439 33.30614 33.92089
## 2021.346      33.57152 33.37064 33.77240 33.26415 33.87889
## 2021.365      33.56633 33.36545 33.76721 33.25896 33.87370
## 2021.385      33.62365 33.42277 33.82453 33.31627 33.93102
## 2021.404      33.61126 33.41038 33.81214 33.30389 33.91863
## 2021.423      33.61125 33.41037 33.81213 33.30388 33.91862
## 2021.442      33.59852 33.39764 33.79940 33.29115 33.90589
## 2021.462      33.59807 33.39719 33.79894 33.29069 33.90544
## 2021.481      33.59845 33.39757 33.79933 33.29108 33.90582
## 2021.500      33.59068 33.38980 33.79156 33.28331 33.89805
## 2021.519      33.56942 33.36854 33.77030 33.26204 33.87679
## 2021.538      33.57752 33.37665 33.77840 33.27015 33.88490
## 2021.558      33.55736 33.35648 33.75823 33.24998 33.86473
## 2021.577      33.53334 33.33246 33.73422 33.22597 33.84071
## 2021.596      33.53376 33.33288 33.73464 33.22639 33.84113
## 2021.615      33.53062 33.32974 33.73150 33.22325 33.83799
## 2021.635      33.50884 33.30796 33.70972 33.20147 33.81621
## 2021.654      33.45389 33.25302 33.65477 33.14652 33.76127
## 2021.673      33.46346 33.26258 33.66434 33.15609 33.77083
## 2021.692      33.45866 33.25779 33.65954 33.15129 33.76604
## 2021.712      33.47316 33.27229 33.67404 33.16579 33.78054
## 2021.731      33.45844 33.25756 33.65932 33.15107 33.76581
## 2021.750      33.45248 33.25160 33.65336 33.14511 33.75985
## 2021.769      33.44129 33.24041 33.64217 33.13392 33.74866

```

```

## 2021.788      33.42192 33.22104 33.62280 33.11455 33.72929
## 2021.808      33.41129 33.21041 33.61217 33.10392 33.71866
## 2021.827      33.42182 33.22094 33.62270 33.11445 33.72919
## 2021.846      33.39264 33.19176 33.59352 33.08527 33.70001
## 2021.865      33.40353 33.20265 33.60441 33.09616 33.71090
## 2021.885      33.41761 33.21673 33.61849 33.11024 33.72498
## 2021.904      33.43236 33.23149 33.63324 33.12499 33.73974
## 2021.923      33.42424 33.22336 33.62512 33.11687 33.73161
## 2021.942      33.43060 33.22972 33.63148 33.12323 33.73797
## 2021.962      33.42735 33.22647 33.62823 33.11998 33.73472
## 2021.981      33.43381 33.23293 33.63468 33.12643 33.74118
## 2022.000      33.41633 33.21545 33.61721 33.10895 33.72370
## 2022.019      33.39624 33.19537 33.59712 33.08887 33.70362
## 2022.038      33.40993 33.20922 33.61065 33.10281 33.71705
## 2022.058      33.40277 33.20206 33.60348 33.09565 33.70989
## 2022.077      33.42351 33.22280 33.62423 33.11639 33.73063

```

PLOO Excluding Outliers: Imputed - Excluding External Variables (Forecasting SALINITY)

```

# PLOO Excluding Outliers: Imputed - Training partition (01/18/1999-01/04/2021)
#owt_df02_gb09_ploo_wkly02_train_tb03
# PLOO Excluding Outliers: Imputed - Test partition (01/11/2021-01/03/2022)
#owt_df02_gb09_ploo_wkly02_test_tb03

owt_df02_gb09_ploo_wkly02_train_tb03 %>%
  plot_time_series(date_sample,
    SALINITY)

owt_df02_gb09_ploo_wkly02_train_tb03 %>%
  plot_stl_diagnostics(date_sample,
    SALINITY,
    .frequency = "auto",
    .trend = "auto")

m2v5_lin_out <- lr_ts_mod(tb_train = owt_df02_gb09_ploo_wkly02_train_tb03,
                            tb_test = owt_df02_gb09_ploo_wkly02_test_tb03,
                            date = "date_sample",
                            param = "SALINITY",
                            ext_pred = c(),
                            train_start = c(1999, lubridate::isoweek(ymd(train_start))),
                            test_start = c(2021, 6),
                            freq = 52,
                            h = 52,
                            formula = "SALINITY ~ trend + season")

##               ME        RMSE       MAE       MPE       MAPE       MASE
## Training set -6.123497e-18 0.1411693 0.1097105 -0.001780718 0.3276193 2.457331
## Test set      1.001722e-01 0.1115920 0.1001722  0.298067126 0.2980671 2.243689
##               ACF1
## Training set 0.895257
## Test set     NA

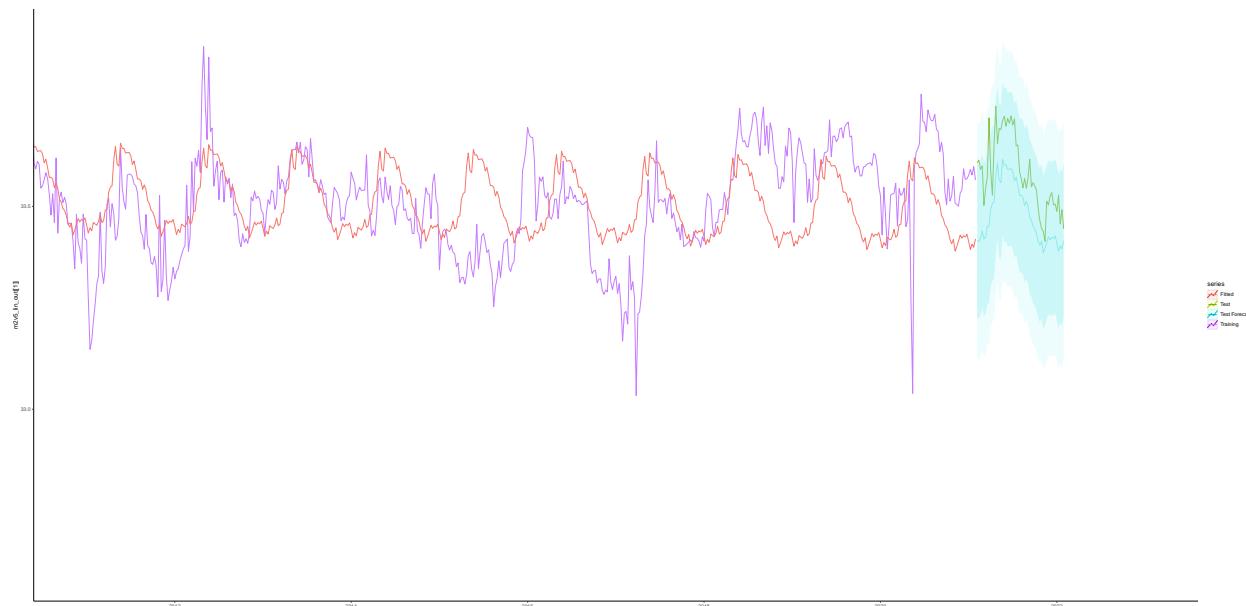
autoplot(m2v5_lin_out[[1]], series = 'Training') +
  autolayer(m2v5_lin_out[[2]], series = 'Test') +

```

```

autolayer(m2v5_lin_out[[3]], series = 'Test Forecast', alpha = .3) +
autolayer(m2v5_lin_out[[3]]$fitted, series = 'Fitted') +
theme_classic() +
coord_cartesian(#ylim = c(-50, 400),
                xlim = c(2011, 2023))

```



```
summary(m2v5_lin_out[[3]])
```

```

##
## Forecast method: Linear regression model
##
## Model Information:
##
## Call:
## tslm(formula = as.formula(formula), data = train_ts01)
##
## Coefficients:
## (Intercept)      trend    season2    season3    season4    season5
## 3.350e+01 -7.796e-05 -1.979e-02 -6.683e-03 -1.418e-02 6.508e-03
## season6    season7    season8    season9    season10   season11
## 2.798e-03 -2.054e-05 8.974e-03 2.623e-02 6.913e-03 1.128e-02
## season12   season13   season14   season15   season16   season17
## 3.781e-02 3.921e-02 8.166e-02 9.437e-02 9.800e-02 1.613e-01
## season18   season19   season20   season21   season22   season23
## 1.951e-01 1.515e-01 1.466e-01 2.036e-01 1.900e-01 1.907e-01
## season24   season25   season26   season27   season28   season29
## 1.791e-01 1.803e-01 1.811e-01 1.730e-01 1.513e-01 1.601e-01
## season30   season31   season32   season33   season34   season35
## 1.393e-01 1.156e-01 1.160e-01 1.112e-01 9.031e-02 1.015e-01
## season36   season37   season38   season39   season40   season41
## 8.887e-02 6.267e-02 5.520e-02 3.976e-02 3.454e-02 2.305e-02
## season42   season43   season44   season45   season46   season47
## 3.562e-03 -6.290e-03 4.220e-03 -2.456e-02 -1.323e-02 2.904e-04

```

```

##      season48      season49      season50      season51      season52
## 1.502e-02 7.585e-03 1.373e-02 1.081e-02 1.744e-02
##
##
## Error measures:
##               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -6.123497e-18 0.1411693 0.1097105 -0.001780718 0.3276193 0.9324258
##          ACF1
## Training set 0.895257
##
## Forecasts:
##      Point Forecast    Lo 80     Hi 80    Lo 95     Hi 95
## 2021.096   33.41613 33.22635 33.60591 33.12574 33.70652
## 2021.115   33.41323 33.22345 33.60301 33.12284 33.70362
## 2021.135   33.42215 33.23237 33.61193 33.13176 33.71254
## 2021.154   33.43933 33.24954 33.62911 33.14893 33.72972
## 2021.173   33.41993 33.23015 33.60971 33.12954 33.71032
## 2021.192   33.42422 33.23444 33.61400 33.13383 33.71461
## 2021.212   33.45068 33.26090 33.64046 33.16028 33.74107
## 2021.231   33.45199 33.26221 33.64178 33.16160 33.74239
## 2021.250   33.49437 33.30459 33.68415 33.20398 33.78476
## 2021.269   33.50700 33.31721 33.69678 33.21660 33.79739
## 2021.288   33.51055 33.32077 33.70033 33.22016 33.80094
## 2021.308   33.57382 33.38403 33.76360 33.28342 33.86421
## 2021.327   33.60745 33.41767 33.79723 33.31706 33.89784
## 2021.346   33.56377 33.37399 33.75355 33.27338 33.85416
## 2021.365   33.55882 33.36904 33.74860 33.26843 33.84921
## 2021.385   33.61577 33.42598 33.80555 33.32537 33.90616
## 2021.404   33.60206 33.41227 33.79184 33.31166 33.89245
## 2021.423   33.60268 33.41290 33.79246 33.31229 33.89307
## 2021.442   33.59103 33.40125 33.78081 33.30064 33.88142
## 2021.462   33.59210 33.40232 33.78188 33.30171 33.88249
## 2021.481   33.59292 33.40314 33.78270 33.30253 33.88331
## 2021.500   33.58473 33.39495 33.77451 33.29434 33.87512
## 2021.519   33.56291 33.37313 33.75270 33.27252 33.85331
## 2021.538   33.57159 33.38181 33.76138 33.28120 33.86199
## 2021.558   33.55072 33.36094 33.74050 33.26033 33.84111
## 2021.577   33.52700 33.33722 33.71678 33.23661 33.81739
## 2021.596   33.52729 33.33751 33.71707 33.23690 33.81768
## 2021.615   33.52239 33.33260 33.71217 33.23199 33.81278
## 2021.635   33.50146 33.31167 33.69124 33.21106 33.79185
## 2021.654   33.51262 33.32284 33.70240 33.22223 33.80301
## 2021.673   33.49987 33.31008 33.68965 33.20947 33.79026
## 2021.692   33.47358 33.28380 33.66336 33.18319 33.76397
## 2021.712   33.46603 33.27625 33.65582 33.17564 33.75643
## 2021.731   33.45051 33.26073 33.64029 33.16012 33.74091
## 2021.750   33.44522 33.25543 33.63500 33.15482 33.73561
## 2021.769   33.43365 33.24387 33.62343 33.14326 33.72404
## 2021.788   33.41409 33.22430 33.60387 33.12369 33.70448
## 2021.808   33.40416 33.21437 33.59394 33.11376 33.69455
## 2021.827   33.41459 33.22481 33.60437 33.12419 33.70498
## 2021.846   33.38573 33.19595 33.57551 33.09534 33.67612
## 2021.865   33.39698 33.20719 33.58676 33.10658 33.68737
## 2021.885   33.41042 33.22064 33.60021 33.12003 33.70082

```

```

## 2021.904      33.42508 33.23529 33.61486 33.13468 33.71547
## 2021.923      33.41756 33.22778 33.60734 33.12717 33.70796
## 2021.942      33.42363 33.23385 33.61342 33.13324 33.71403
## 2021.962      33.42063 33.23085 33.61041 33.13024 33.71102
## 2021.981      33.42718 33.23740 33.61696 33.13679 33.71757
## 2022.000      33.40967 33.21988 33.59945 33.11927 33.70006
## 2022.019      33.38980 33.20002 33.57958 33.09941 33.68019
## 2022.038      33.40283 33.21320 33.59245 33.11267 33.69298
## 2022.058      33.39525 33.20563 33.58488 33.10510 33.68541
## 2022.077      33.41586 33.22624 33.60549 33.12571 33.70602

```

SBOO All: Imputed - Excluding External Variables (Forecasting SALINITY)

```

# SBOO All: Imputed - Training partition (01/18/1999-01/04/2021)
#owt_df02_gb09_sboo_wkly03_train_tb03
# SBOO All: Imputed - Test partition (01/11/2021-01/03/2022)
#owt_df02_gb09_sboo_wkly03_test_tb03

owt_df02_gb09_sboo_wkly03_train_tb03 %>%
  plot_time_series(date_sample,
                    SALINITY)

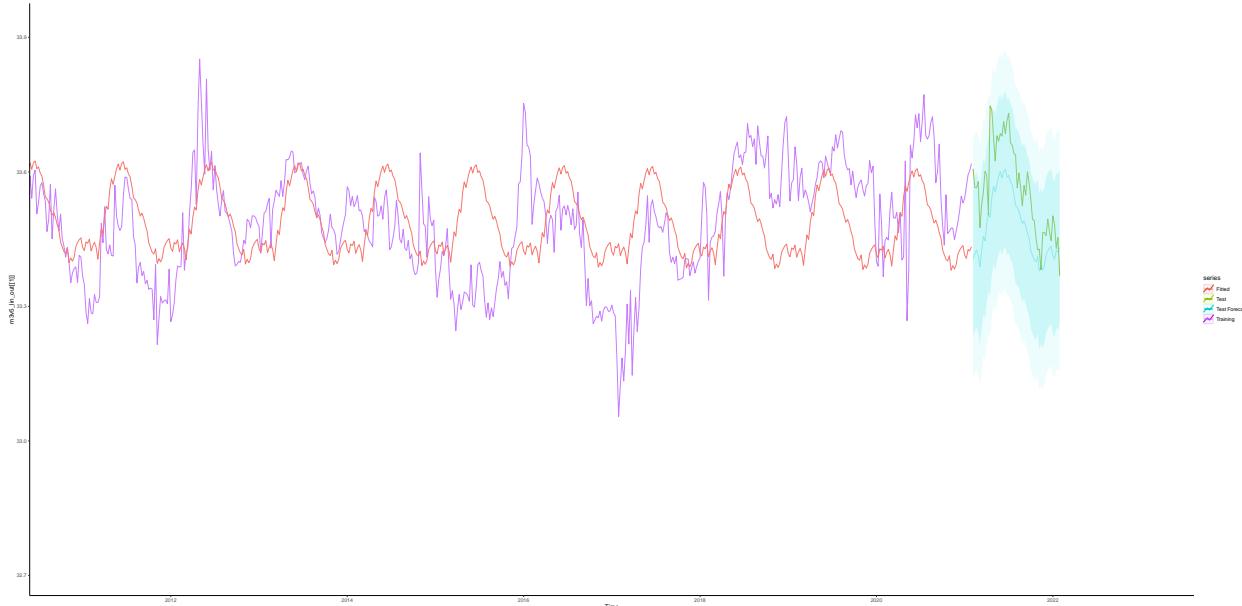
owt_df02_gb09_sboo_wkly03_train_tb03 %>%
  plot_stl_diagnostics(date_sample,
                        SALINITY,
                        .frequency = "auto",
                        .trend = "auto")

m3v5_lin_out <- lr_ts_mod(tb_train = owt_df02_gb09_sboo_wkly03_train_tb03,
                            tb_test = owt_df02_gb09_sboo_wkly03_test_tb03,
                            date = "date_sample",
                            param = "SALINITY",
                            ext_pred = c(),
                            train_start = c(1999, lubridate::isoweek(ymd(train_start))),
                            test_start = c(2021, 6),
                            freq = 52,
                            h = 52,
                            formula = "SALINITY ~ trend + season")

##               ME        RMSE       MAE        MPE       MAPE       MASE
## Training set -6.205923e-18 0.1289931 0.10160558 -0.001487309 0.3036048 2.764791
## Test set      8.702422e-02 0.1024643 0.08977283  0.258952595 0.2671888 2.442810
##               ACF1
## Training set 0.9099218
## Test set     NA

autoplot(m3v5_lin_out[[1]], series = 'Training') +
  autolayer(m3v5_lin_out[[2]], series = 'Test') +
  autolayer(m3v5_lin_out[[3]], series = 'Test Forecast', alpha = .3) +
  autolayer(m3v5_lin_out[[3]]$fitted, series = 'Fitted') +
  theme_classic() +
  coord_cartesian(#ylim = c(-50, 400),
                 xlim = c(2011, 2023))

```



```
summary(m3v5_lin_out[[3]])
```

```
##
## Forecast method: Linear regression model
##
## Model Information:
##
## Call:
## tslm(formula = as.formula(formula), data = train_ts01)
##
## Coefficients:
## (Intercept)      trend    season2    season3    season4    season5
## 3.345e+01 -3.326e-05 -7.255e-03 1.245e-02 9.157e-03 1.868e-02
##   season6    season7    season8    season9    season10   season11
## -7.978e-03 3.660e-03 1.141e-02 -4.467e-04 -2.599e-02 9.562e-03
##   season12   season13   season14   season15   season16   season17
## 4.175e-02 3.204e-02 7.053e-02 9.237e-02 8.538e-02 1.287e-01
##   season18   season19   season20   season21   season22   season23
## 1.536e-01 1.412e-01 1.592e-01 1.764e-01 1.867e-01 1.731e-01
##   season24   season25   season26   season27   season28   season29
## 1.875e-01 1.918e-01 1.747e-01 1.784e-01 1.647e-01 1.580e-01
##   season30   season31   season32   season33   season34   season35
## 1.375e-01 1.119e-01 1.076e-01 1.002e-01 8.406e-02 7.094e-02
##   season36   season37   season38   season39   season40   season41
## 7.706e-02 6.638e-02 5.009e-02 3.819e-02 1.112e-02 4.170e-04
##   season42   season43   season44   season45   season46   season47
## -9.674e-03 -1.335e-02 -1.778e-03 -3.414e-02 -2.373e-02 -3.116e-02
##   season48   season49   season50   season51   season52
## -2.158e-02 1.842e-03 1.049e-02 1.712e-02 2.109e-02
##
## Error measures:
##               ME        RMSE        MAE        MPE        MAPE        MASE
## Training set -6.205923e-18 0.1289931 0.1016056 -0.001487309 0.3036048 0.8655685
```

```

##          ACF1
## Training set 0.9099218
##
## Forecasts:
##           Point Forecast    Lo 80     Hi 80     Lo 95     Hi 95
## 2021.096      33.40632 33.23291 33.57974 33.14098 33.67167
## 2021.115      33.41793 33.24452 33.59134 33.15258 33.68327
## 2021.135      33.42565 33.25223 33.59906 33.16030 33.69099
## 2021.154      33.41375 33.24034 33.58717 33.14841 33.67910
## 2021.173      33.38818 33.21477 33.56159 33.12283 33.65353
## 2021.192      33.42370 33.25028 33.59711 33.15835 33.68904
## 2021.212      33.45586 33.28244 33.62927 33.19051 33.72120
## 2021.231      33.44611 33.27270 33.61952 33.18076 33.71145
## 2021.250      33.48456 33.31115 33.65798 33.21922 33.74991
## 2021.269      33.50638 33.33296 33.67979 33.24103 33.77172
## 2021.288      33.49935 33.32594 33.67276 33.23400 33.76469
## 2021.308      33.54267 33.36925 33.71608 33.27732 33.80801
## 2021.327      33.56751 33.39410 33.74092 33.30216 33.83286
## 2021.346      33.55503 33.38162 33.72844 33.28968 33.82037
## 2021.365      33.57301 33.39960 33.74643 33.30767 33.83836
## 2021.385      33.59018 33.41676 33.76359 33.32483 33.85552
## 2021.404      33.60043 33.42702 33.77384 33.33508 33.86577
## 2021.423      33.58688 33.41347 33.76029 33.32153 33.85223
## 2021.442      33.60118 33.42776 33.77459 33.33583 33.86652
## 2021.462      33.60542 33.43201 33.77884 33.34008 33.87077
## 2021.481      33.58835 33.41493 33.76176 33.32300 33.85369
## 2021.500      33.59204 33.41862 33.76545 33.32669 33.85738
## 2021.519      33.57832 33.40490 33.75173 33.31297 33.84366
## 2021.538      33.57157 33.39816 33.74498 33.30623 33.83692
## 2021.558      33.55098 33.37757 33.72439 33.28564 33.81633
## 2021.577      33.52537 33.35196 33.69879 33.26003 33.79072
## 2021.596      33.52099 33.34757 33.69440 33.25564 33.78633
## 2021.615      33.51361 33.34020 33.68702 33.24827 33.77896
## 2021.635      33.49743 33.32401 33.67084 33.23208 33.76277
## 2021.654      33.48427 33.31086 33.65769 33.21893 33.74962
## 2021.673      33.49036 33.31695 33.66378 33.22502 33.75571
## 2021.692      33.47965 33.30623 33.65306 33.21430 33.74499
## 2021.712      33.46333 33.28992 33.63674 33.19798 33.72868
## 2021.731      33.45140 33.27798 33.62481 33.18605 33.71674
## 2021.750      33.42429 33.25088 33.59771 33.15895 33.68964
## 2021.769      33.41355 33.24014 33.58697 33.14821 33.67890
## 2021.788      33.40343 33.23002 33.57684 33.13808 33.66878
## 2021.808      33.39972 33.22631 33.57313 33.13437 33.66506
## 2021.827      33.41126 33.23785 33.58467 33.14591 33.67660
## 2021.846      33.37886 33.20545 33.55227 33.11352 33.64421
## 2021.865      33.38924 33.21582 33.56265 33.12389 33.65458
## 2021.885      33.38178 33.20837 33.55519 33.11643 33.64712
## 2021.904      33.39132 33.21791 33.56474 33.12598 33.65667
## 2021.923      33.41471 33.24130 33.58813 33.14937 33.68006
## 2021.942      33.42333 33.24991 33.59674 33.15798 33.68867
## 2021.962      33.42993 33.25651 33.60334 33.16458 33.69527
## 2021.981      33.43387 33.26045 33.60728 33.16852 33.69921
## 2022.000      33.41274 33.23932 33.58615 33.14739 33.67808
## 2022.019      33.40545 33.23204 33.57886 33.14010 33.67079

```

```

## 2022.038      33.42512 33.25185 33.59839 33.15999 33.69025
## 2022.058      33.42180 33.24852 33.59507 33.15667 33.68692
## 2022.077      33.43128 33.25801 33.60455 33.16615 33.69641

```

SBOO Excluding Outliers: Imputed - Excluding External Variables (Forecasting SALINITY)

```

# SBOO Excluding Outliers: Imputed - Training partition (01/18/1999-01/04/2021)
#owt_df02_gb09_sboo_wkly02_train_tb03
# SBOO Excluding Outliers: Imputed - Test partition (01/11/2021-01/03/2022)
#owt_df02_gb09_sboo_wkly02_test_tb03

owt_df02_gb09_sboo_wkly02_train_tb03 %>%
  plot_time_series(date_sample,
                    SALINITY)

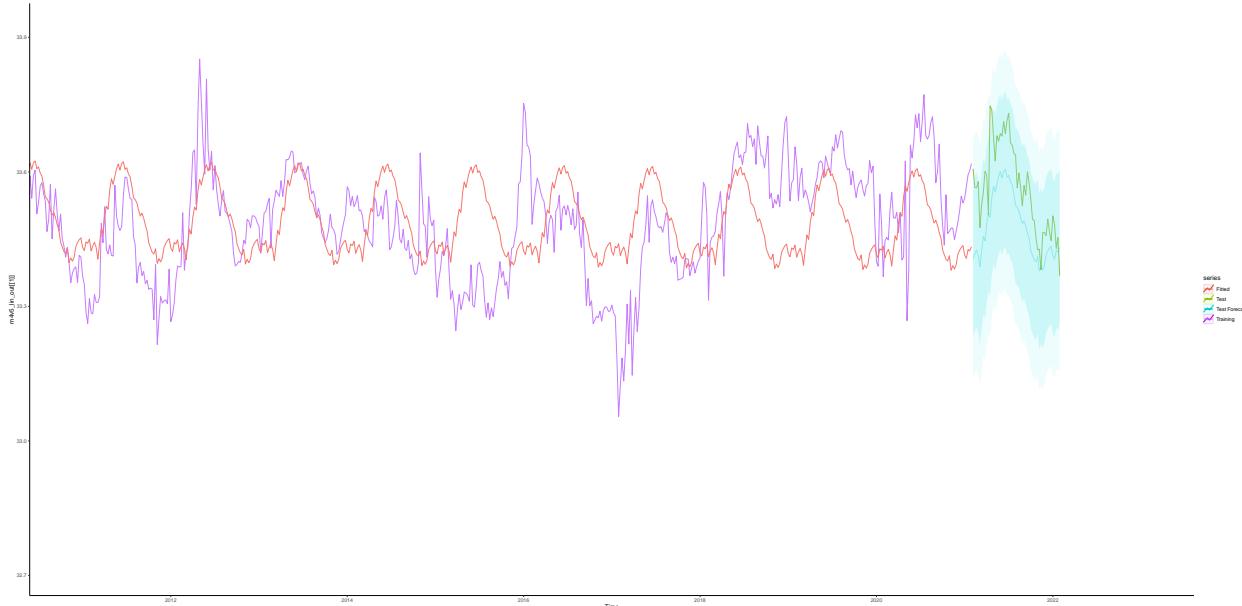
owt_df02_gb09_sboo_wkly02_train_tb03 %>%
  plot_stl_diagnostics(date_sample,
                        SALINITY,
                        .frequency = "auto",
                        .trend = "auto")

m4v5_lin_out <- lr_ts_mod(tb_train = owt_df02_gb09_sboo_wkly02_train_tb03,
                            tb_test = owt_df02_gb09_sboo_wkly02_test_tb03,
                            date = "date_sample",
                            param = "SALINITY",
                            ext_pred = c(),
                            train_start = c(1999, lubridate::isoweek(ymd(train_start))),
                            test_start = c(2021, 6),
                            freq = 52,
                            h = 52,
                            formula = "SALINITY ~ trend + season")

##               ME        RMSE       MAE        MPE       MAPE       MASE
## Training set -6.205923e-18 0.1289931 0.10160558 -0.001487309 0.3036048 2.764791
## Test set      8.702422e-02 0.1024643 0.08977283  0.258952595 0.2671888 2.442810
##               ACF1
## Training set 0.9099218
## Test set     NA

autoplot(m4v5_lin_out[[1]], series = 'Training') +
  autolayer(m4v5_lin_out[[2]], series = 'Test') +
  autolayer(m4v5_lin_out[[3]], series = 'Test Forecast', alpha = .3) +
  autolayer(m4v5_lin_out[[3]]$fitted, series = 'Fitted') +
  theme_classic() +
  coord_cartesian(#ylim = c(-50, 400),
                 xlim = c(2011, 2023))

```



```
summary(m4v5_lin_out[[3]])
```

```
##
## Forecast method: Linear regression model
##
## Model Information:
##
## Call:
## tslm(formula = as.formula(formula), data = train_ts01)
##
## Coefficients:
## (Intercept)      trend    season2    season3    season4    season5
## 3.345e+01 -3.326e-05 -7.255e-03 1.245e-02 9.157e-03 1.868e-02
##   season6    season7    season8    season9    season10   season11
## -7.978e-03 3.660e-03 1.141e-02 -4.467e-04 -2.599e-02 9.562e-03
##   season12   season13   season14   season15   season16   season17
## 4.175e-02 3.204e-02 7.053e-02 9.237e-02 8.538e-02 1.287e-01
##   season18   season19   season20   season21   season22   season23
## 1.536e-01 1.412e-01 1.592e-01 1.764e-01 1.867e-01 1.731e-01
##   season24   season25   season26   season27   season28   season29
## 1.875e-01 1.918e-01 1.747e-01 1.784e-01 1.647e-01 1.580e-01
##   season30   season31   season32   season33   season34   season35
## 1.375e-01 1.119e-01 1.076e-01 1.002e-01 8.406e-02 7.094e-02
##   season36   season37   season38   season39   season40   season41
## 7.706e-02 6.638e-02 5.009e-02 3.819e-02 1.112e-02 4.170e-04
##   season42   season43   season44   season45   season46   season47
## -9.674e-03 -1.335e-02 -1.778e-03 -3.414e-02 -2.373e-02 -3.116e-02
##   season48   season49   season50   season51   season52
## -2.158e-02 1.842e-03 1.049e-02 1.712e-02 2.109e-02
##
## Error measures:
##               ME        RMSE        MAE        MPE        MAPE        MASE
## Training set -6.205923e-18 0.1289931 0.1016056 -0.001487309 0.3036048 0.8655685
```

```

##          ACF1
## Training set 0.9099218
##
## Forecasts:
##           Point Forecast    Lo 80     Hi 80     Lo 95     Hi 95
## 2021.096      33.40632 33.23291 33.57974 33.14098 33.67167
## 2021.115      33.41793 33.24452 33.59134 33.15258 33.68327
## 2021.135      33.42565 33.25223 33.59906 33.16030 33.69099
## 2021.154      33.41375 33.24034 33.58717 33.14841 33.67910
## 2021.173      33.38818 33.21477 33.56159 33.12283 33.65353
## 2021.192      33.42370 33.25028 33.59711 33.15835 33.68904
## 2021.212      33.45586 33.28244 33.62927 33.19051 33.72120
## 2021.231      33.44611 33.27270 33.61952 33.18076 33.71145
## 2021.250      33.48456 33.31115 33.65798 33.21922 33.74991
## 2021.269      33.50638 33.33296 33.67979 33.24103 33.77172
## 2021.288      33.49935 33.32594 33.67276 33.23400 33.76469
## 2021.308      33.54267 33.36925 33.71608 33.27732 33.80801
## 2021.327      33.56751 33.39410 33.74092 33.30216 33.83286
## 2021.346      33.55503 33.38162 33.72844 33.28968 33.82037
## 2021.365      33.57301 33.39960 33.74643 33.30767 33.83836
## 2021.385      33.59018 33.41676 33.76359 33.32483 33.85552
## 2021.404      33.60043 33.42702 33.77384 33.33508 33.86577
## 2021.423      33.58688 33.41347 33.76029 33.32153 33.85223
## 2021.442      33.60118 33.42776 33.77459 33.33583 33.86652
## 2021.462      33.60542 33.43201 33.77884 33.34008 33.87077
## 2021.481      33.58835 33.41493 33.76176 33.32300 33.85369
## 2021.500      33.59204 33.41862 33.76545 33.32669 33.85738
## 2021.519      33.57832 33.40490 33.75173 33.31297 33.84366
## 2021.538      33.57157 33.39816 33.74498 33.30623 33.83692
## 2021.558      33.55098 33.37757 33.72439 33.28564 33.81633
## 2021.577      33.52537 33.35196 33.69879 33.26003 33.79072
## 2021.596      33.52099 33.34757 33.69440 33.25564 33.78633
## 2021.615      33.51361 33.34020 33.68702 33.24827 33.77896
## 2021.635      33.49743 33.32401 33.67084 33.23208 33.76277
## 2021.654      33.48427 33.31086 33.65769 33.21893 33.74962
## 2021.673      33.49036 33.31695 33.66378 33.22502 33.75571
## 2021.692      33.47965 33.30623 33.65306 33.21430 33.74499
## 2021.712      33.46333 33.28992 33.63674 33.19798 33.72868
## 2021.731      33.45140 33.27798 33.62481 33.18605 33.71674
## 2021.750      33.42429 33.25088 33.59771 33.15895 33.68964
## 2021.769      33.41355 33.24014 33.58697 33.14821 33.67890
## 2021.788      33.40343 33.23002 33.57684 33.13808 33.66878
## 2021.808      33.39972 33.22631 33.57313 33.13437 33.66506
## 2021.827      33.41126 33.23785 33.58467 33.14591 33.67660
## 2021.846      33.37886 33.20545 33.55227 33.11352 33.64421
## 2021.865      33.38924 33.21582 33.56265 33.12389 33.65458
## 2021.885      33.38178 33.20837 33.55519 33.11643 33.64712
## 2021.904      33.39132 33.21791 33.56474 33.12598 33.65667
## 2021.923      33.41471 33.24130 33.58813 33.14937 33.68006
## 2021.942      33.42333 33.24991 33.59674 33.15798 33.68867
## 2021.962      33.42993 33.25651 33.60334 33.16458 33.69527
## 2021.981      33.43387 33.26045 33.60728 33.16852 33.69921
## 2022.000      33.41274 33.23932 33.58615 33.14739 33.67808
## 2022.019      33.40545 33.23204 33.57886 33.14010 33.67079

```

```

## 2022.038      33.42512 33.25185 33.59839 33.15999 33.69025
## 2022.058      33.42180 33.24852 33.59507 33.15667 33.68692
## 2022.077      33.43128 33.25801 33.60455 33.16615 33.69641

```

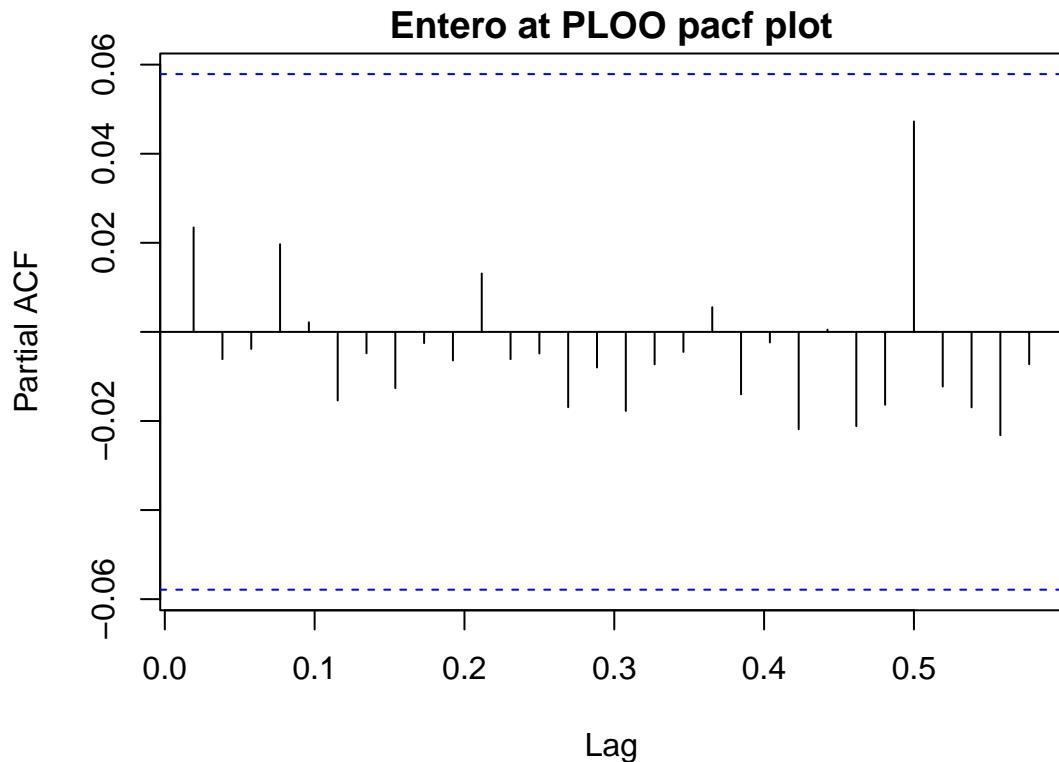
ARIMA Modeling

```

# PLOO Excluding Outliers - Training partition (01/18/1999-01/04/2021)
# owt_df02_gb09_ploo_wkly02_train_tb03 (imputed w/out outliers set)
gb09_ploo_wkly02_train_ts01_ent01 <- ts(owt_df02_gb09_ploo_wkly02_train_tb03$ENTERO,
                                         start = c(1999, 3),
                                         freq = 52)
# PLOO Excluding Outliers - Test partition (01/11/2021-01/03/2022)
# owt_df02_gb09_ploo_wkly02_test_tb03$ENTERO

# pacf plot of entero (ploo)
par(mar=c(5, 4, 0, 2)+1.5)
pacf(gb09_ploo_wkly02_train_ts01_ent01)
title("Enter at PLOO pacf plot")

```



ARIMA of only entero on PLOO data

```

# The plot shows no lines exceeding the boundaries so ar=p=0
# we aren't doing any differences so d=0
# From acf PLOO plot previously done, there are 3 or 4 leaving boundary so q= 3 or 4
# No seasonality (previous PLOO time series graph)

```

```

gb09_ploo_wkly03_train_ts01_ent01 <- ts(owt_df02_gb09_ploo_wkly03_train_tb02$ENTERO,
                                         start = c(1999, 3),
                                         freq = 52)
# calling arima function
arima_entero_ploo_q3 <- arima(gb09_ploo_wkly03_train_ts01_ent01, order= c(0, 0, 3))
summary(arima_entero_ploo_q3)

##
## Call:
## arima(x = gb09_ploo_wkly03_train_ts01_ent01, order = c(0, 0, 3))
##
## Coefficients:
##             ma1      ma2      ma3  intercept
##             0.0214 -0.0054 -0.0052    21.4321
## s.e.   0.0296  0.0290  0.0301     3.0887
##
## sigma^2 estimated as 10510:  log likelihood = -6805.09,  aic = 13620.17
##
## Training set error measures:
##               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -0.003363843 102.5163 27.91931 -424.0505 436.7107 0.8360876
## ACF1
## Training set 0.0002646514

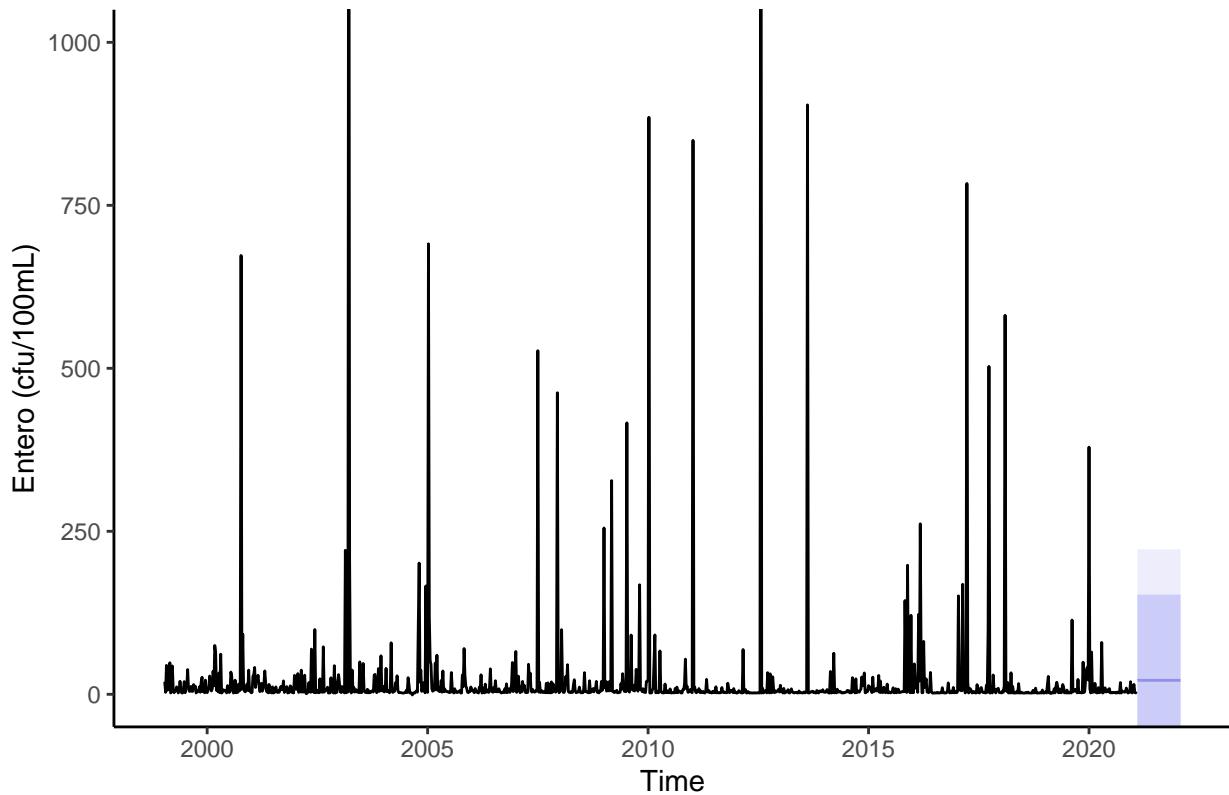
# saving training rmse value
arima_entero_ploo_q3_train_rmse <- 102.5163

# making forecasts and saving them
arima_entero_ploo_q3_forecast <- forecast(arima_entero_ploo_q3, h=52)

#
# getting plot ready
autoplot(gb09_ploo_wkly02_train_ts01_ent01,
         main = "ARIMA(0,0,3) for only Entero at PL00",
         ylab = "Entero (cfu/100mL)") +
  autolayer(arima_entero_ploo_q3_forecast, alpha=.3) +
  coord_cartesian(xlim=c(2015, 2025)) +
  coord_cartesian(ylim=c(0, 1000)) +
  theme_classic()

```

ARIMA(0,0,3) for only Enteroto at PLOO



```
# calling arima function
arima_entero_ploo_q4 <- arima(gb09_ploo_wkly03_train_ts01_ent01, order= c(0, 0, 4))

summary(arima_entero_ploo_q4)

##
## Call:
## arima(x = gb09_ploo_wkly03_train_ts01_ent01, order = c(0, 0, 4))
##
## Coefficients:
##             ma1      ma2      ma3      ma4  intercept
##             0.0211 -0.0048 -0.0044  0.0214    21.4330
## s.e.     0.0296  0.0296  0.0301  0.0308     3.1552
##
## sigma^2 estimated as 10505:  log likelihood = -6804.84,  aic = 13621.69
##
## Training set error measures:
##              ME      RMSE      MAE      MPE      MAPE      MASE
## Training set 0.005557505 102.4938 27.90147 -422.6641 435.3383 0.8355533
##                  ACF1
## Training set 0.0004429775

# saving training rmse value
arima_entero_ploo_q4_train_rmse <- 102.4938

# making forecasts and saving them
```

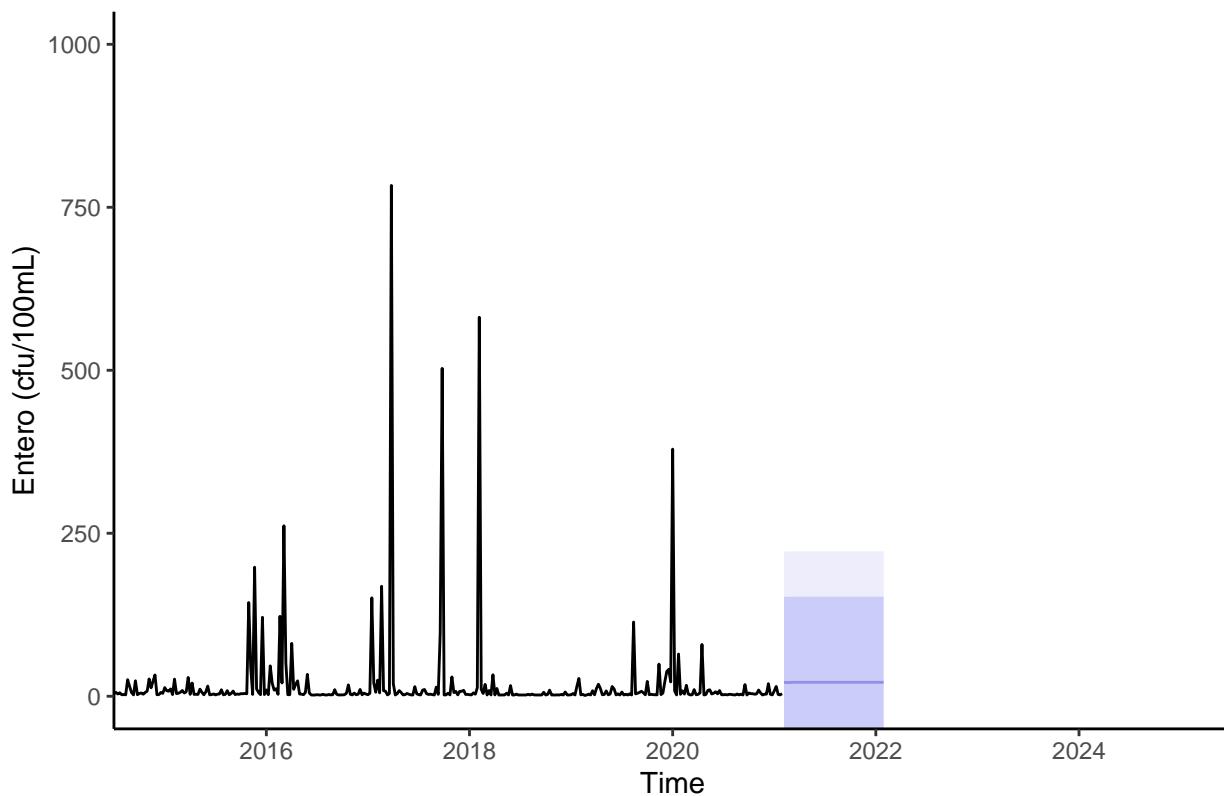
```

arima_entero_ploo_q4_forecast <- forecast(arima_entero_ploo_q4, h=52)

# getting plot ready
autoplot(gb09_ploo_wkly02_train_ts01_ent01,
         main = "ARIMA(0,0,4) for only Entero at PLOO",
         ylab = "Entero (cfu/100mL)") +
  autolayer(arima_entero_ploo_q4_forecast, alpha=.3) +
  coord_cartesian(xlim=c(2015, 2025),
                  ylim=c(0, 1000)) +
  theme_classic()

```

ARIMA(0,0,4) for only Entero at PLOO



```

print(paste("ARIMA(0,0,3) RMSE of ", arima_entero_ploo_q3_train_rmse))

## [1] "ARIMA(0,0,3) RMSE of 102.5163"

print(paste("ARIMA(0,0,4) RMSE of ", arima_entero_ploo_q4_train_rmse))

## [1] "ARIMA(0,0,4) RMSE of 102.4938"

# the RMSE values are very close (they both round to 102.5) so going with simpler one
# (q=3)

# looking at performance of the better (lower RMSE train) or ARIMA(0,0,3)
arima_entero_ploo_acc <- accuracy(arima_entero_ploo_q3_forecast,
                                    owt_df02_gb09_ploo_wkly02_test_tb03$ENTERO)

```

```
arima_entero_ploo_acc
```

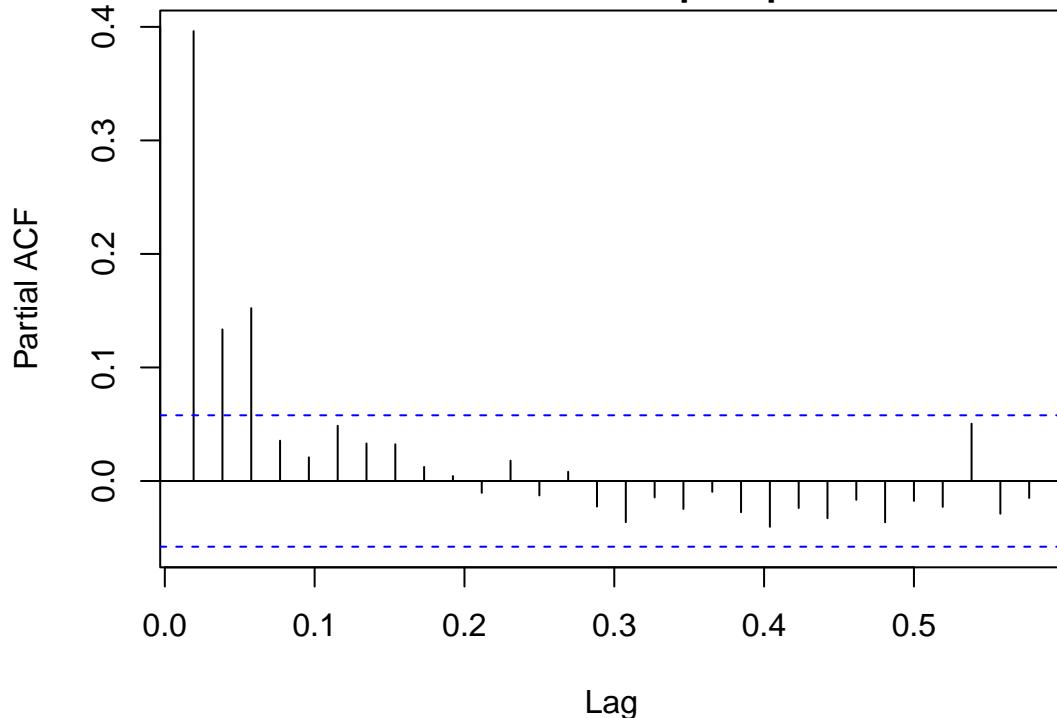
```
##               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -0.003363843 102.51625 27.91931 -424.0505 436.7107 0.8360876
## Test set     -14.377836809 19.72948 18.07501 -569.9495 576.5190 0.5412846
##                   ACF1
## Training set 0.0002646514
## Test set      NA
```

ARIMA of only entero on SBOO data

```
# SBOO Excluding Outliers - Training partition (01/18/1999-01/04/2021)
# owt_df02_gb09_sboo_wkly02_train_tb03
gb09_sboo_wkly02_train_ts01_ent01 <- ts(owt_df02_gb09_sboo_wkly02_train_tb03$ENTERO,
                                         start = c(1999, 3),
                                         freq = 52)
# SBOO Excluding Outliers - Test partition (01/11/2021-01/03/2022)
# owt_df02_gb09_sboo_wkly02_test_tb03

# pacf plot of entero (ploo)
par(mar=c(5, 4, 0, 2)+1.5)
pacf(gb09_sboo_wkly02_train_ts01_ent01)
title("Enter at SBOO pacf plot")
```

Enter at SBOO pacf plot



```

# based on pacf plot with 1 set out of bounds, p=1
# no differencing done so d=0
# based on the acf plot with 2 or 3 sets out of bound, q=2 or 3
# seasonality in time series plot so need second set
# P= seasonal AR= 0 since only out of bounds at beginning
# D=0 (still no differencing)
# Q= Seasonal MA (looking at year points on acf): repeated spikes corresponding to year
# ahead
# therefore Q=1

# calling arima model for sboo
arima_entero_sboo_q2 <- arima(gb09_sboo_wkly02_train_ts01_ent01,
                               order = c(1, 0, 2),
                               seasonal = c(0, 0, 1))
summary(arima_entero_sboo_q2)

##
## Call:
## arima(x = gb09_sboo_wkly02_train_ts01_ent01, order = c(1, 0, 2), seasonal = c(0,
##       0, 1))
##
## Coefficients:
##             ar1      ma1      ma2      sma1  intercept
##             0.8706 -0.5585 -0.0937  0.0577   141.4872
## s.e.    0.0344  0.0460  0.0365  0.0311   25.3017
##
## sigma^2 estimated as 92009:  log likelihood = -8182.65,  aic = 16377.31
##
## Training set error measures:
##               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set 0.05817302 303.3306 160.5064 -1082.045 1104.841 0.993147
##               ACF1
## Training set -0.002314864

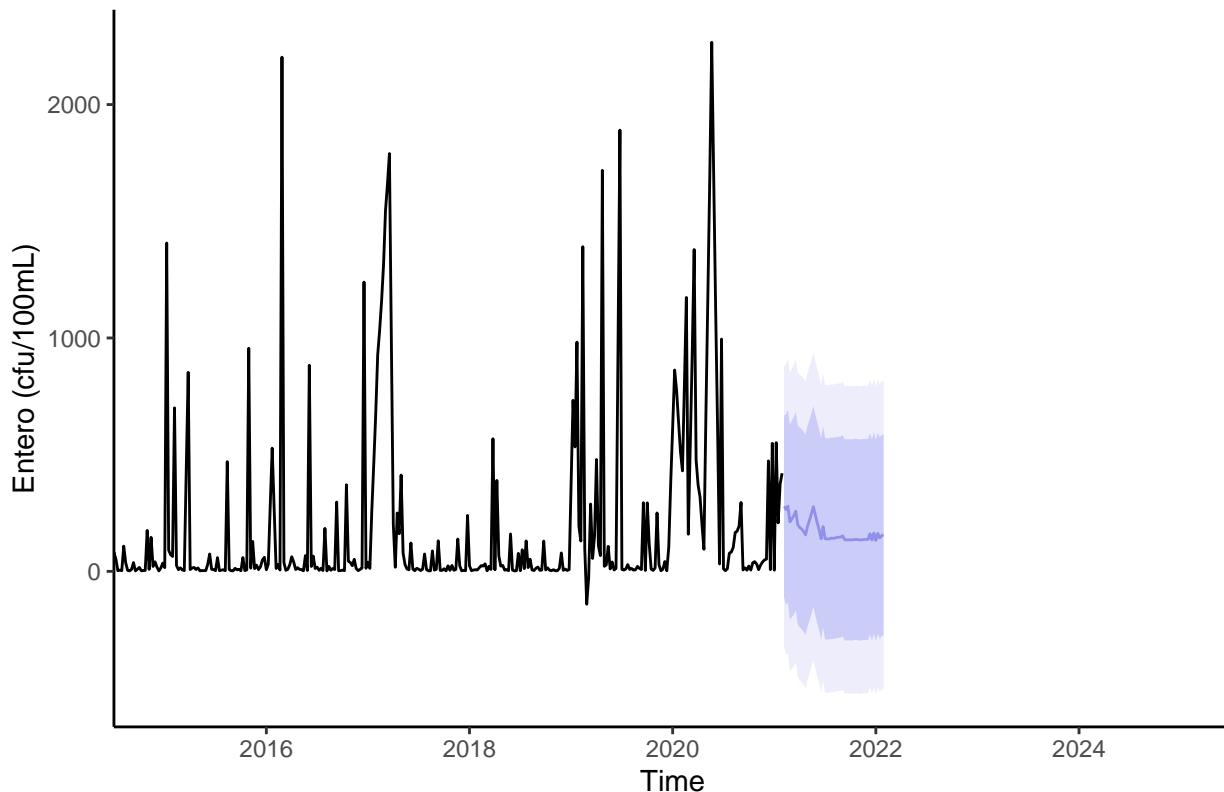
# saving training rmse value
arima_entero_sboo_q2_train_rmse <- sqrt(mean(arima_entero_sboo_q2$residuals^2))

# making forecasts and saving them
arima_entero_sboo_q2_forecast <- forecast(arima_entero_sboo_q2, h=52)

# getting plot ready
autoplot(gb09_sboo_wkly02_train_ts01_ent01,
          main = "ARIMA(1,0,2)(0,0,1) for only Entero at SBOO",
          ylab = "Enter (cfu/100mL)") +
  autolayer(arima_entero_sboo_q2_forecast, alpha=.3) +
  coord_cartesian(xlim=c(2015, 2025)) +
  theme_classic()

```

ARIMA(1,0,2)(0,0,1) for only Enteroto at SBOO



```
# calling arima model for sboo with q=3
arima_entero_sboo_q3 <- arima(gb09_sboo_wkly02_train_ts01_ent01,
                               order = c(1, 0, 3),
                               seasonal = c(0, 0, 1))
summary(arima_entero_sboo_q3)

##
## Call:
## arima(x = gb09_sboo_wkly02_train_ts01_ent01, order = c(1, 0, 3), seasonal = c(0,
##       0, 1))
##
## Coefficients:
##             ar1      ma1      ma2      ma3     sma1  intercept
##             0.8487 -0.5364 -0.1022  0.0339  0.0539   141.7728
## s.e.    0.0482  0.0574  0.0387  0.0330  0.0315    24.5176
##
## sigma^2 estimated as 91923:  log likelihood = -8182.11,  aic = 16378.22
##
## Training set error measures:
##                  ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -0.0849465 303.1888 160.6197 -1082.38 1105.38 0.993848
##                  ACF1
## Training set -0.001032968

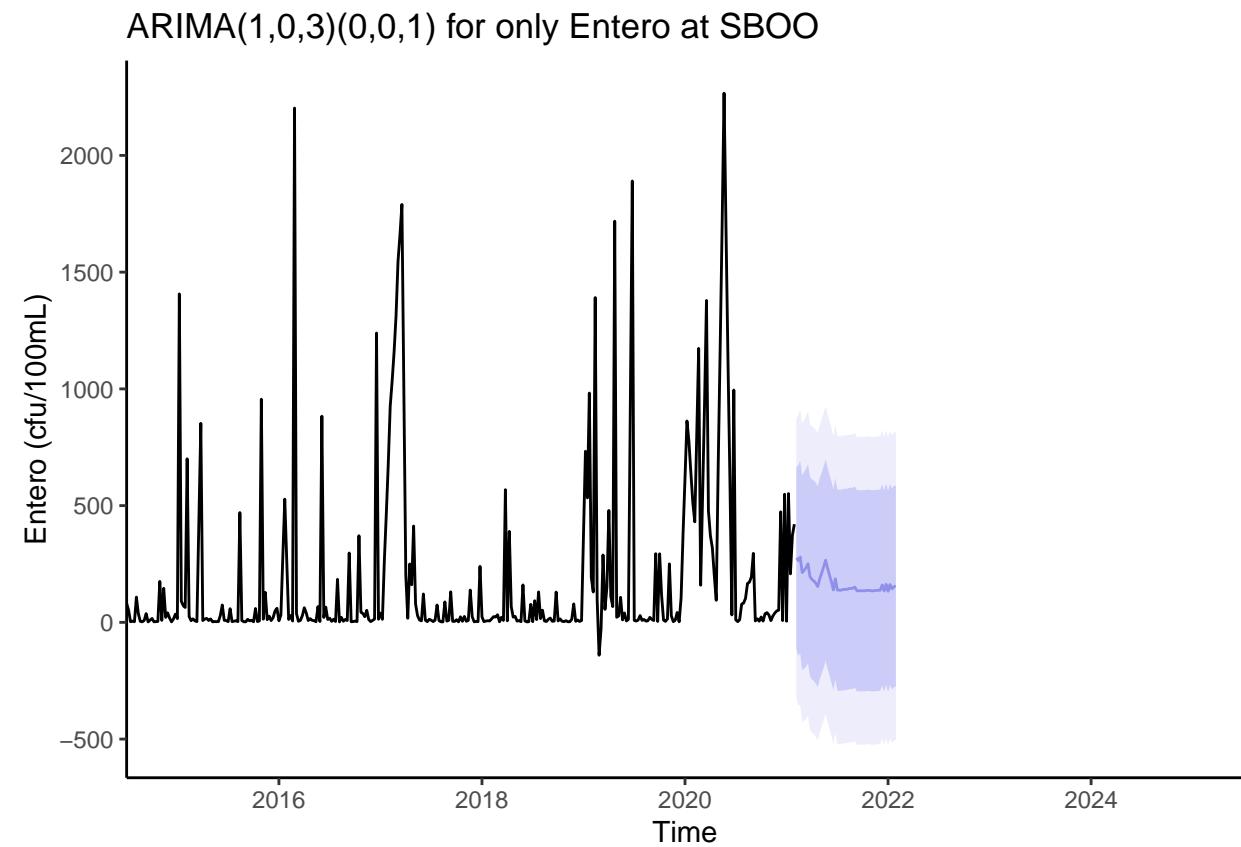
# saving training rmse value
arima_entero_sboo_q3_train_rmse <- sqrt(mean(arima_entero_sboo_q3$residuals^2))
```

```

# making forecasts and saving them
arima_entero_sboo_q3_forecast <- forecast(arima_entero_sboo_q3, h=52)

# getting plot ready
autoplot(gb09_sboo_wkly02_train_ts01_ent01,
         main = "ARIMA(1,0,3)(0,0,1) for only Entero at SBOO",
         ylab = "Entero (cfu/100mL)") +
  autolayer(arima_entero_sboo_q3_forecast, alpha=.3) +
  coord_cartesian(xlim=c(2015, 2025)) +
  theme_classic()

```



```

print(paste("ARIMA(1,0,2)(0,0,1) RMSE of ", arima_entero_sboo_q2_train_rmse))

## [1] "ARIMA(1,0,2)(0,0,1) RMSE of 303.330637745971"

print(paste("ARIMA(1,0,3)(0,0,1) RMSE of ", arima_entero_sboo_q3_train_rmse))

## [1] "ARIMA(1,0,3)(0,0,1) RMSE of 303.188753331435"

# q=2 RMSE of 303.3 vs. q=3 RMSE of 303.2; going with q=3 since lower.

# looking at performance of the better (lower RMSE train) or ARIMA(1,0,3)(0,0,1).
arima_entero_sboo_acc <- accuracy(arima_entero_sboo_q3_forecast,
                                    owt_df02_gb09_sboo_wkly02_test_tb03$ENTERO)

```

```
arima_entero_sboo_acc
```

```
##               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -0.0849465 303.1888 160.6197 -1082.3797 1105.3801 0.993848
## Test set     142.0636657 465.5062 293.9374 -838.8908 884.7925 1.818763
##                   ACF1
## Training set -0.001032968
## Test set       NA
```

ARIMA Regression Models:

ARIMA Regression 1-PLOO: Using all but SUSO and OG from PLOO

Use: chlorophyll, fecal, ph, salinity, temp, density, DO, XMS) to get: Entero

```
# set up ARIMA regression
# PLOO Excluding Outliers - Training partition (01/18/1999-01/04/2021)

# training set: all predictors (minus the OG and SUSO) from PLOO
ploo_arima_train <- owt_df02_gb09_ploo_wkly02_train_tb03 %>% select(-date_sample)
# test set: all predictors (minus the OG and SUSO) from PLOO
ploo_arima_test <- owt_df02_gb09_ploo_wkly02_test_tb03 %>% select(-date_sample)

# predictors as matrix
# predictors as matrix (1-9 columns)
ploo_arima_predictors_train_df <- ploo_arima_train %>% select(-ENTERO)
ploo_arima_predictors_train <- as.matrix(ploo_arima_predictors_train_df)

ploo_arima_predictors_test_df <- ploo_arima_test %>% select(-ENTERO)
ploo_arima_predictors_test <- as.matrix(ploo_arima_predictors_test_df)

# outcome
ploo_arima_train_outcome <- ploo_arima_train$ENTERO
ploo_arima_test_outcome <- ploo_arima_test$ENTERO

# using the same model set up PLOO-entero only: ARIMA(0,0,3)
ploo_arimaReg <- Arima(ploo_arima_train_outcome, order = c(0,0,3), xreg =
  ploo_arima_predictors_train)

summary(ploo_arimaReg)

## Series: ploo_arima_train_outcome
## Regression with ARIMA(0,0,3) errors
##
## Coefficients:
##             ma1      ma2      ma3  intercept CHLOROPHYLL    DENSITY      DO      FECAL
##             0.0103   -0.0133   -0.0019    353.0589    -2.3532   37.9092   3.6846   0.4184
## s.e.      0.0297    0.0295    0.0297    750.2709     1.1970   17.3877   4.6965   0.0229
##             PH  SALINITY    TEMP      XMS
##             72.7709  -57.9220   7.4265   -0.7637
## s.e.      41.9715   23.0405   3.6873    0.5800
##             sigma^2 = 8011: log likelihood = -6776.44
## AIC=13578.88  AICc=13579.2  BIC=13644.46
##
```

```

## Training set error measures:
##               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -4.273099e-06 89.03495 25.55855 -191.1968 295.9924 0.7858751
##               ACF1
## Training set -1.932063e-06

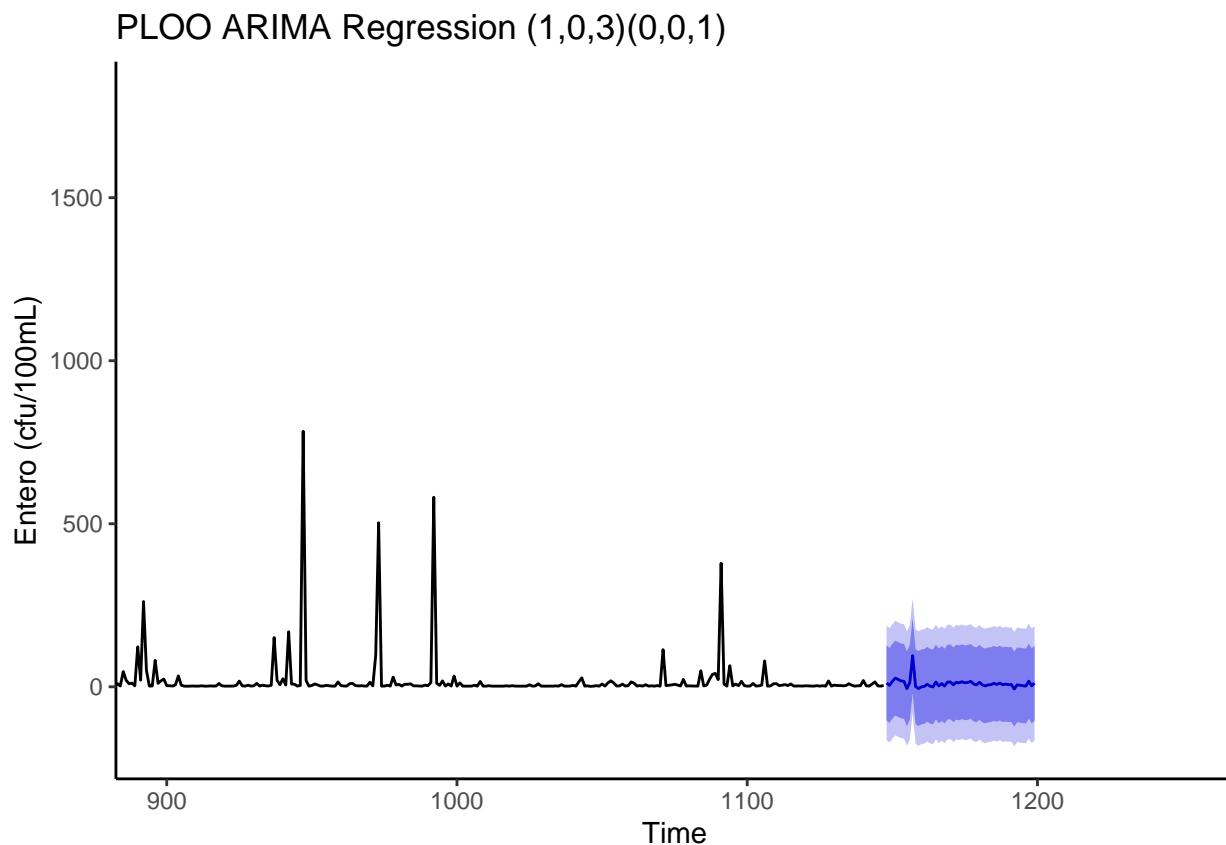
ploo_arimaReg_train_rmse <- sqrt(mean(ploo_arimaReg$residuals^2))
print(paste("PLOO ARIMA Regression RMSE: ", ploo_arimaReg_train_rmse))

## [1] "PLOO ARIMA Regression RMSE: 89.0349452478337"

# saving forecast using test set
ploo_arimaReg_forecast <- forecast(ploo_arimaReg, xreg= ploo_arima_predictors_test, h=52)

# plotting the forecasts
autoplot(ploo_arimaReg_forecast,
         main = "PLOO ARIMA Regression (1,0,3)(0,0,1)",
         ylab = "Enterobacter (cfu/100mL)" +
         coord_cartesian(xlim = c(900, 1250))+
         theme_classic()

```



```

ploo_arimaReg_coefs <- as.data.frame(round(ploo_arimaReg$coef, 3))
colnames(ploo_arimaReg_coefs)[colnames(ploo_arimaReg_coefs) == "round(ploo_arimaReg$coef,
← 3)"] <- "coef_value"
ploo_arimaReg_coefs

##           coef_value

```

```

## ma1          0.010
## ma2         -0.013
## ma3        -0.002
## intercept   353.059
## CHLOROPHYLL -2.353
## DENSITY     37.909
## DO           3.685
## FECAL        0.418
## PH           72.771
## SALINITY    -57.922
## TEMP         7.427
## XMS         -0.764

# the three strongest absolute coeffecients (most important) at PLOO are:
# pH (.73), density (.38), temp (.7.43)
# the next are DO (.69), chlorophyll (2.35), XMS (.764), and fecal (.418)

# looking at performance of PLOO ARima Regression.
arimaReg_entero_ploo_acc <- accuracy(ploo_arimaReg_forecast,
                                         owt_df02_gb09_ploo_wkly02_test_tb03$ENTERO)

arimaReg_entero_ploo_acc

```

```

##               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -4.273099e-06 89.03495 25.55855 -191.1968 295.9924 0.7858751
## Test set      -2.978892e+00 19.60083 11.46303 -198.9662 250.0864 0.3524655
##                      ACF1
## Training set -1.932063e-06
## Test set       NA

```

ARIMA Regression 1-SBOO: Using all but SUSO and OG from SBOO

Use: chlorophyll, fecal, ph, salinity, temp, density, DO, XMS) to get: Entero

```

# set up ARIMA regression for SBOO

# training set: all predictors (minus the OG and SUSO) from SBOO
sboo_arima_train <- owt_df02_gb09_sboo_wkly02_train_tb03 %>% select(-date_sample)
# test set: all predictors (minus the OG and SUSO) from SBOO
sboo_arima_test <- owt_df02_gb09_sboo_wkly02_test_tb03 %>% select(-date_sample)

# predictors as matrix
sboo_arima_predictors_train_df <- sboo_arima_train %>% select(-ENTERO)
sboo_arima_predictors_train <- as.matrix(sboo_arima_predictors_train_df)

sboo_arima_predictors_test_df <- sboo_arima_test %>% select(-ENTERO)
sboo_arima_predictors_test <- as.matrix(sboo_arima_predictors_test_df)
# outcome
sboo_arima_train_outcome <- sboo_arima_train$ENTERO
sboo_arima_test_outcome <- sboo_arima_test$ENTERO

# using the same model set up SBOO-entero only: ARIMA(1,0,3)(0,0,1)
sboo_arimaReg <- Arima(sboo_arima_train_outcome, order = c(1,0,3), seasonal = c(0,0,1),
                       xreg = sboo_arima_predictors_train)

```

```

summary(sboo_arimaReg)

## Series: sboo_arima_train_outcome
## Regression with ARIMA(1,0,3) errors
##
## Coefficients:
##             ar1      ma1      ma2      ma3  intercept  CHLOROPHYLL   DENSITY
##             0.4531 -0.1253 -0.0761  0.0961  10328.437     -4.5984 -68.1285
## s.e.    0.1607  0.1603  0.0598  0.0339   2879.752      3.4067  68.4621
##             D0    FECAL          PH  SALINITY      TEMP       XMS
##             22.1975  0.3890 -223.7577 -181.5957  -16.9649  -7.7588
## s.e.    14.2382  0.0175  135.1713   93.5847   14.4394   1.2690
##
## sigma^2 = 60253: log likelihood = -7933.17
## AIC=15894.35  AICc=15894.72  BIC=15964.98
##
## Training set error measures:
##             ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -0.02501764 244.069 121.7442 -362.0654 647.8616 0.7533024
##             ACF1
## Training set 0.001202772

sboo_arimaReg_train_rmse <- sqrt(mean(sboo_arimaReg$residuals^2))
print(paste("SBOO ARIMA Regression RMSE: ", sboo_arimaReg_train_rmse))

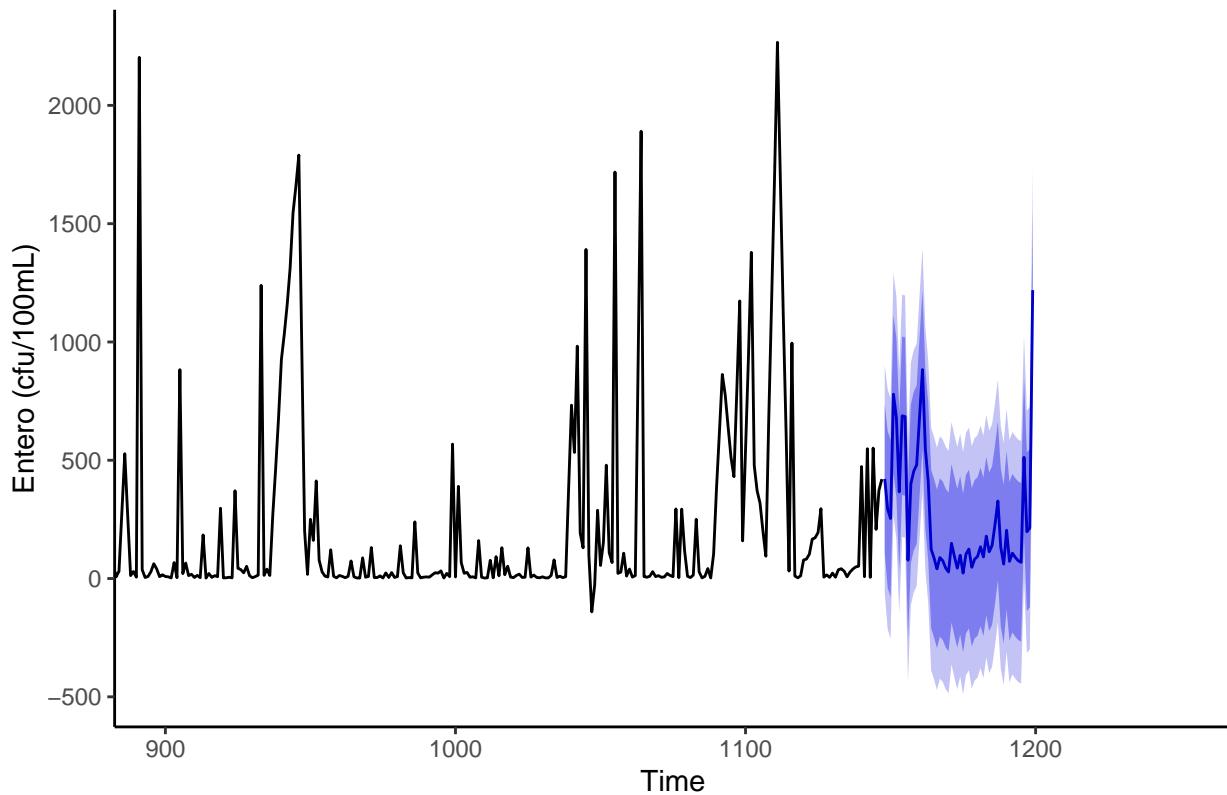
## [1] "SBOO ARIMA Regression RMSE: 244.068957667935"

# saving forecast using test set
sboo_arimaReg_forecast <- forecast(sboo_arimaReg, xreg= sboo_arima_predictors_test)

# plotting the forecasts
autoplot(sboo_arimaReg_forecast,
         main = "SBOO ARIMA Regression (1,0,3)(0,0,1)",
         ylab = "Enterobacter (cfu/100mL)" +
coord_cartesian(xlim = c(900, 1250))+ theme_classic()

```

SBOO ARIMA Regression (1,0,3)(0,0,1)



```

sboo_arimaReg_coefs <- as.data.frame(round(sboo_arimaReg$coef, 3))
colnames(sboo_arimaReg_coefs)[colnames(sboo_arimaReg_coefs) == "round(sboo_arimaReg$coef,
  ↪ 3)"] <- "coef_value"
sboo_arimaReg_coefs

##           coef_value
## ar1          0.453
## ma1         -0.125
## ma2         -0.076
## ma3          0.096
## intercept   10328.437
## CHLOROPHYLL -4.598
## DENSITY     -68.129
## DO           22.198
## FECAL        0.389
## PH          -223.758
## SALINITY    -181.596
## TEMP        -16.965
## XMS          -7.759

# three strongest absolute coefficients (most important) at SBOO are:
# pH (224), salinity (182), density (68),
# then it drops to DO (22), temp (17), xms (7.76), chlorophyll (4.60), fecal (.389)

# looking at performance of the ARIMA Regression at SBOO.
arimaReg_entero_sboo_acc <- accuracy(sboo_arimaReg_forecast,

```

```

        owt_df02_gb09_sboo_wkly02_test_tb03$ENTERO)

arimaReg_entero_sboo_acc

##               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -0.02501764 244.069 121.7442 -362.0654 647.8616 0.7533024
## Test set      50.51999458 230.856 150.8782 -421.8009 445.9124 0.9335716
##               ACF1
## Training set 0.001202772
## Test set      NA

```

ARIMA models of the top coefficients of the regression models

ARIMA at PLOO

```

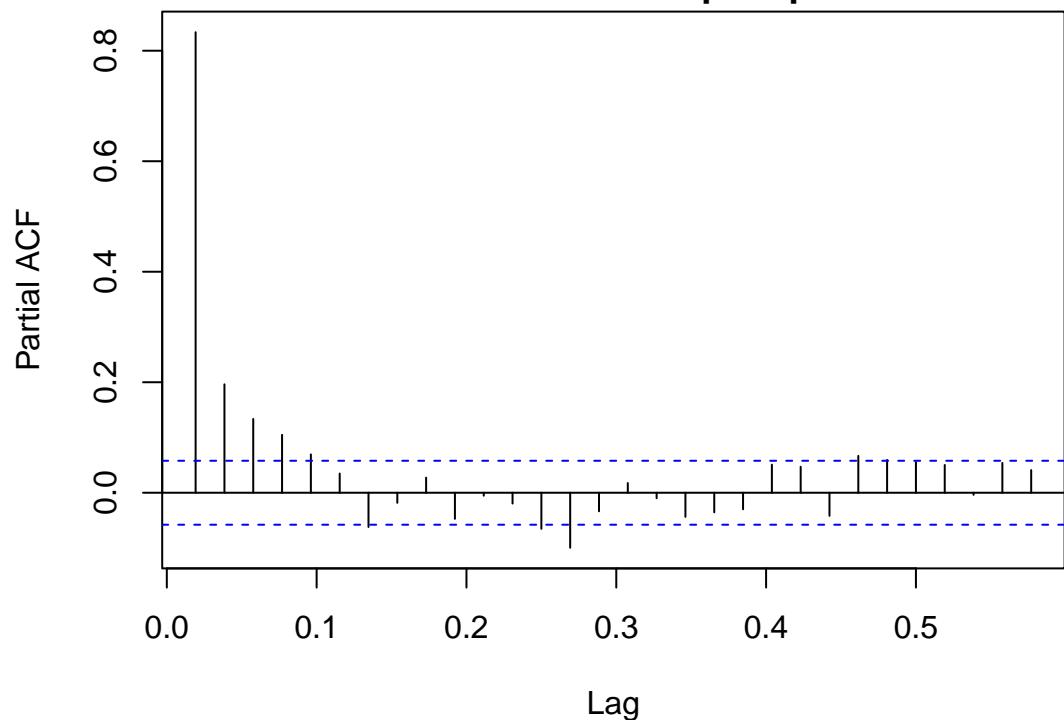
# PLOO Excluding Outliers - Training partition (01/18/1999-01/04/2021)
# owt_df02_gb09_ploo_wkly02_train_tb03 (imputed w/out outliers set)

# DENSITY
ploo_arima_train_density <- ts(owt_df02_gb09_ploo_wkly02_train_tb03$DENSITY,
                                start = c(1999, 3),
                                freq = 52)
# PLOO Excluding Outliers - Test partition (01/11/2021-01/03/2022)
ploo_arima_test_density <- owt_df02_gb09_ploo_wkly02_test_tb03$DENSITY

# pacf plot of density (ploo)
par(mar=c(5, 4, 0, 2)+1.5)
pacf(ploo_arima_train_density)
title("DENSITY at PLOO pacf plot")

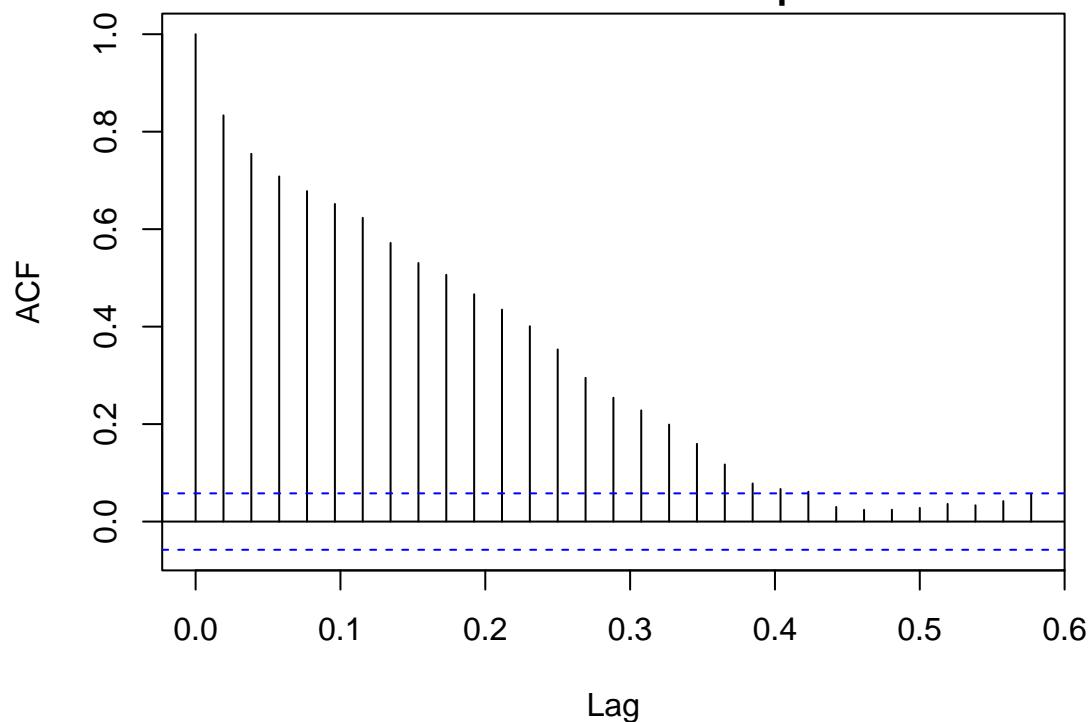
```

DENSITY at PLOO pacf plot



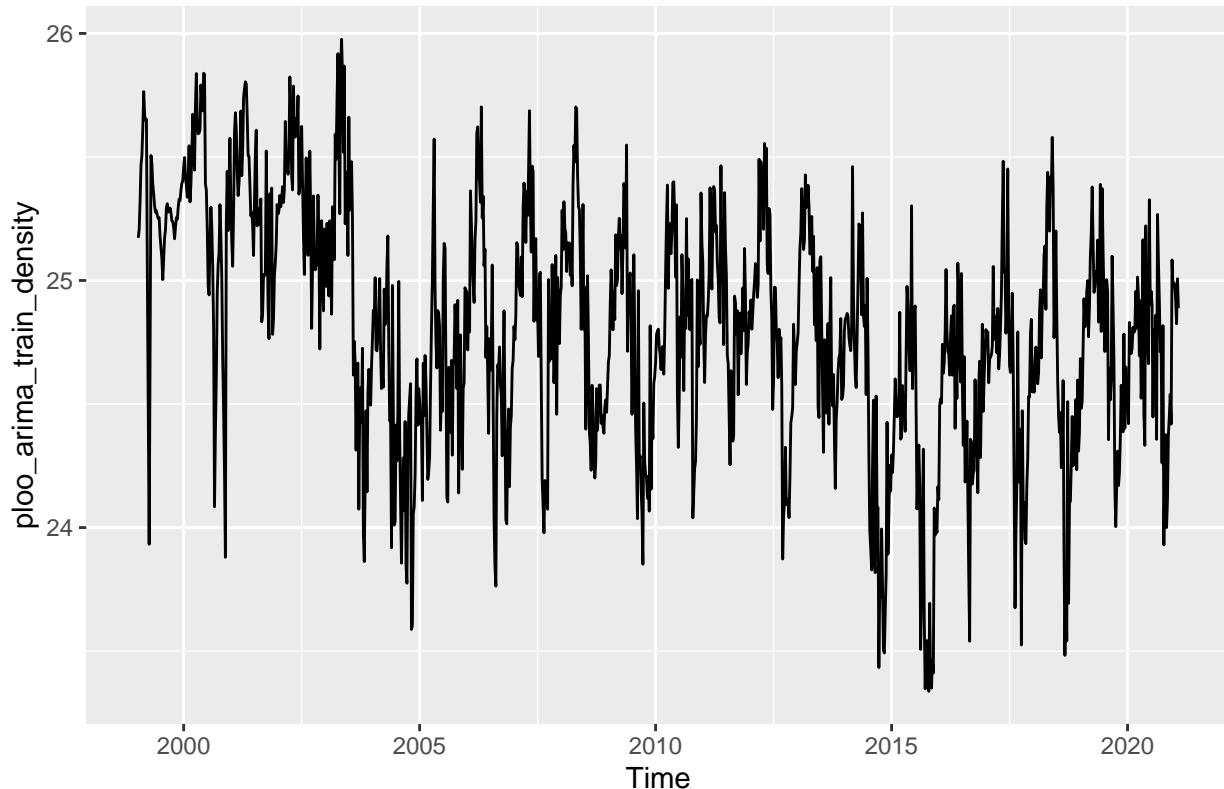
```
# p=2  
  
# acf plot of density (ploo)  
acf(ploo_arima_train_density)  
title("DENSITY at PLOO acf plot")
```

DENSITY at PLOO acf plot



```
# q value at least 1  
  
# ts plot of density (ploo)  
autoplot(ploo_arima_train_density,  
         main = "DENSITY at PLOO ts plot")
```

DENSITY at PLOO ts plot



```

# definite seasonality so 2nd part needed
# d and D = 0
# P= 1 (pacf plot out of bounds at more than beginning, barely)
# Q (acf): Q= 0 since acf had no repeating patterns of out of bounds

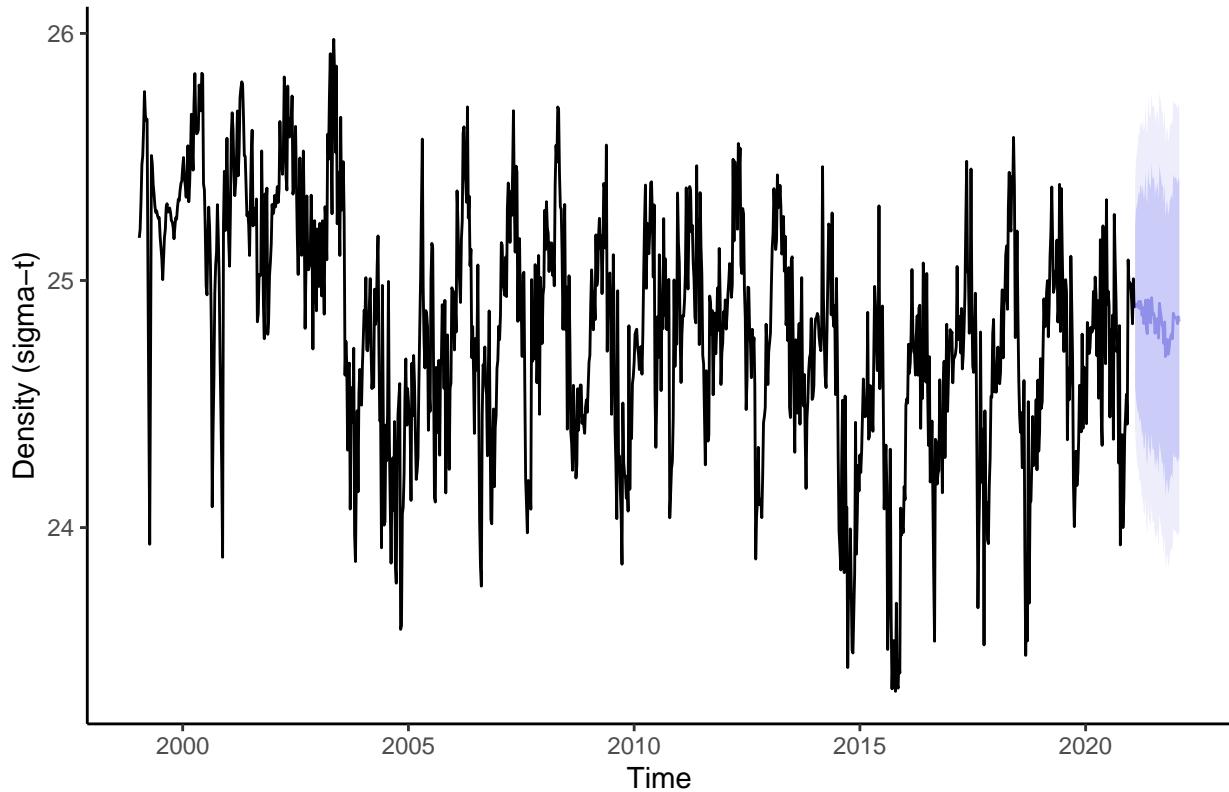
ploo_arima_density <- arima(ploo_arima_train_density,
                             order=c(2, 0, 1), seasonal = c(1, 0, 0))

# saving forecast using test set
ploo_arima_density_forecast <- forecast(ploo_arima_density, h=52)

# plotting the forecasts
autoplot(ploo_arima_train_density,
         main = "PLOO ARIMA Density",
         ylab = "Density (sigma-t)") +
  autolayer(ploo_arima_density_forecast, alpha=.3) +
  theme_classic()

```

PLOO ARIMA Density



```
# looking at performance of the ARIMA Regression at SBOO.
ploo_arima_density_acc <- accuracy(ploo_arima_density_forecast, ploo_arima_test_density)

ploo_arima_density_acc

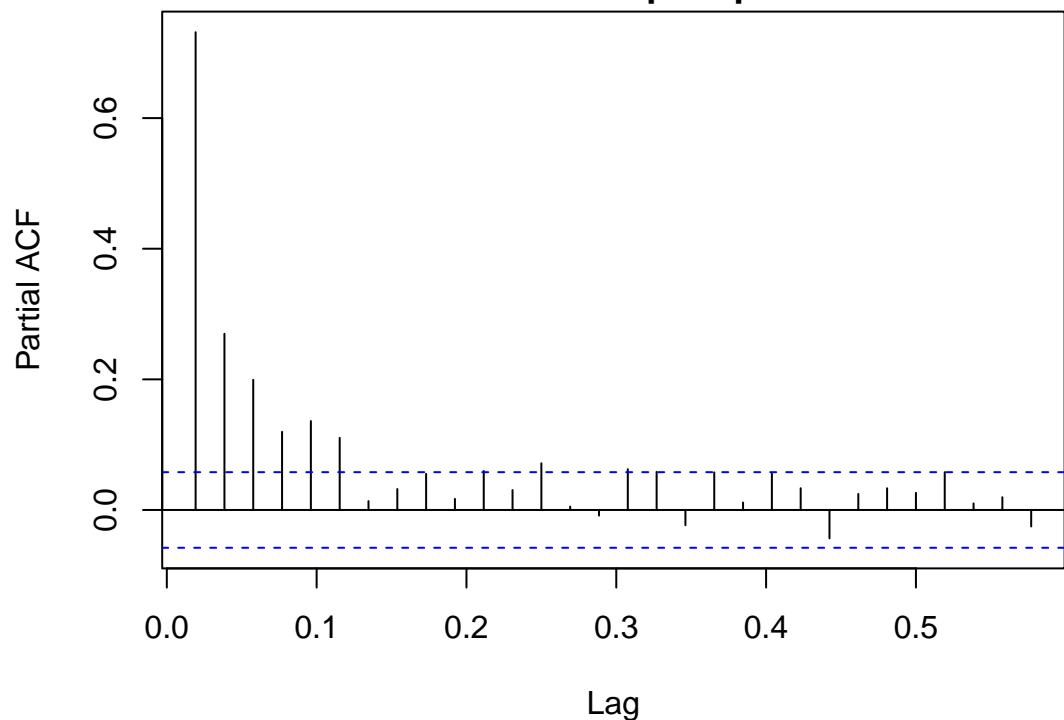
##                               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -0.001313540 0.2542205 0.1927933 -0.01617478 0.7797282 0.9308101
## Test set       0.002642859 0.2953983 0.2388208 -0.00427795 0.9603224 1.1530320
##                               ACF1
## Training set 0.006691562
## Test set      NA

# PLOO Excluding Outliers - Training partition (01/18/1999-01/04/2021)
# owt_df02_gb09_ploo_wkly02_train_tb03 (imputed w/out outliers set)

# PH
ploo_arima_train_ph <- ts(owt_df02_gb09_ploo_wkly02_train_tb03$PH,
                           start = c(1999, 3),
                           freq = 52)
# PLOO Excluding Outliers - Test partition (01/11/2021-01/03/2022)
ploo_arima_test_ph <- owt_df02_gb09_ploo_wkly02_test_tb03$PH

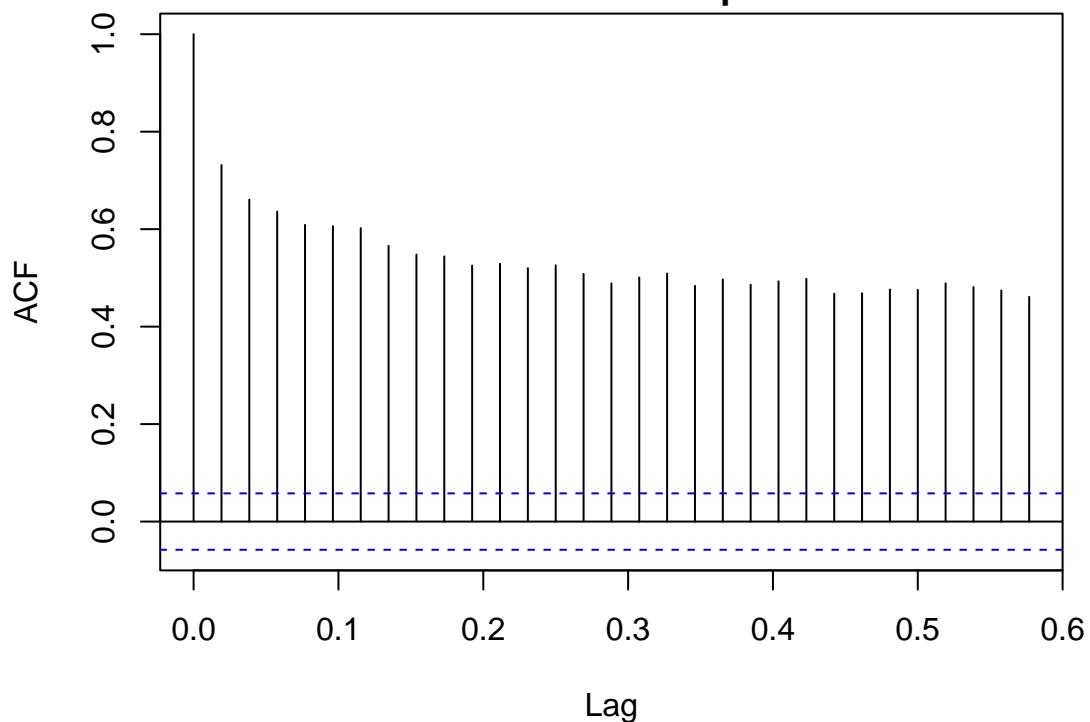
# pacf plot of ph (ploo)
par(mar=c(5, 4, 0, 2)+1.5)
pacf(ploo_arima_train_ph)
title("PH at PL00 pacf plot")
```

PH at PLOO pacf plot



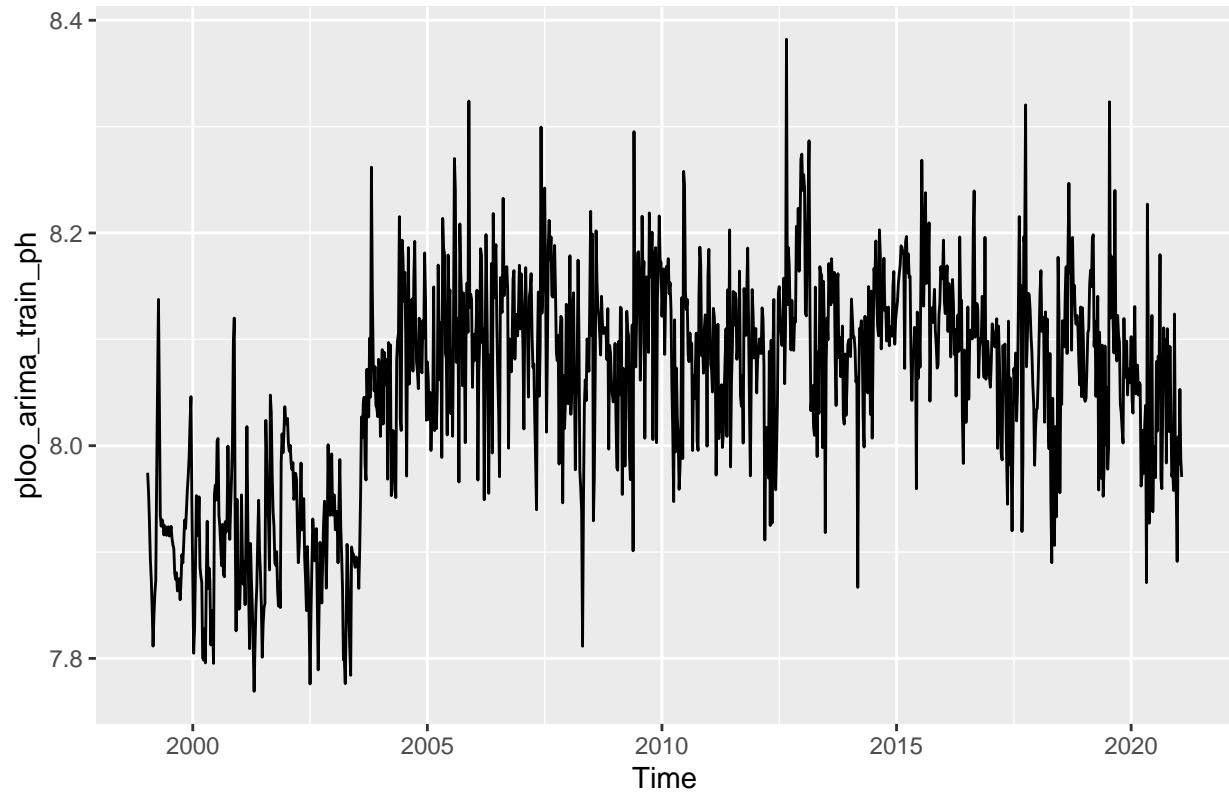
```
# p= 1  
  
# acf plot of density (ploo)  
acf(ploo_arima_train_ph)  
title("PH at PLOO acf plot")
```

PH at PLOO acf plot



```
# q (acf plot) all out of bounds so 1  
  
# ts plot of density (ploo)  
autoplot(ploo_arima_train_ph,  
         main = "PH at PLOO ts plot")
```

PH at PLOO ts plot



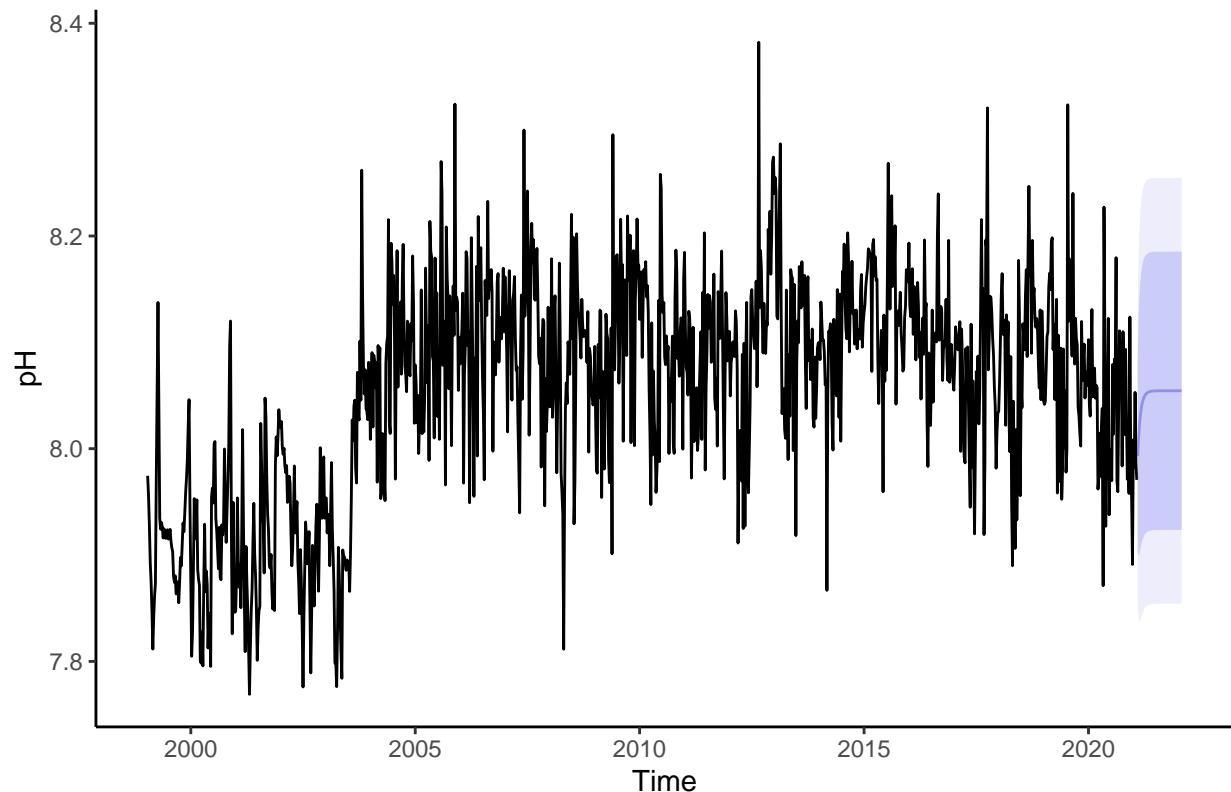
```
# hard to tell on seasonality maybe none

ploo_arima_ph <- arima(ploo_arima_train_ph,
                        order=c(1, 0, 0))

# saving forecast using test set
ploo_arima_ph_forecast <- forecast(ploo_arima_ph, h=52)

# plotting the forecasts
autoplot(ploo_arima_train_ph,
         main = "PLOO ARIMA PH",
         ylab = "pH") +
  autolayer(ploo_arima_ph_forecast, alpha=.3) +
  theme_classic()
```

PLOO ARIMA PH



```
# looking at performance of the ARIMA Regression at SBOO.
ploo_arima_ph_acc <- accuracy(ploo_arima_ph_forecast, ploo_arima_test_ph)

ploo_arima_ph_acc

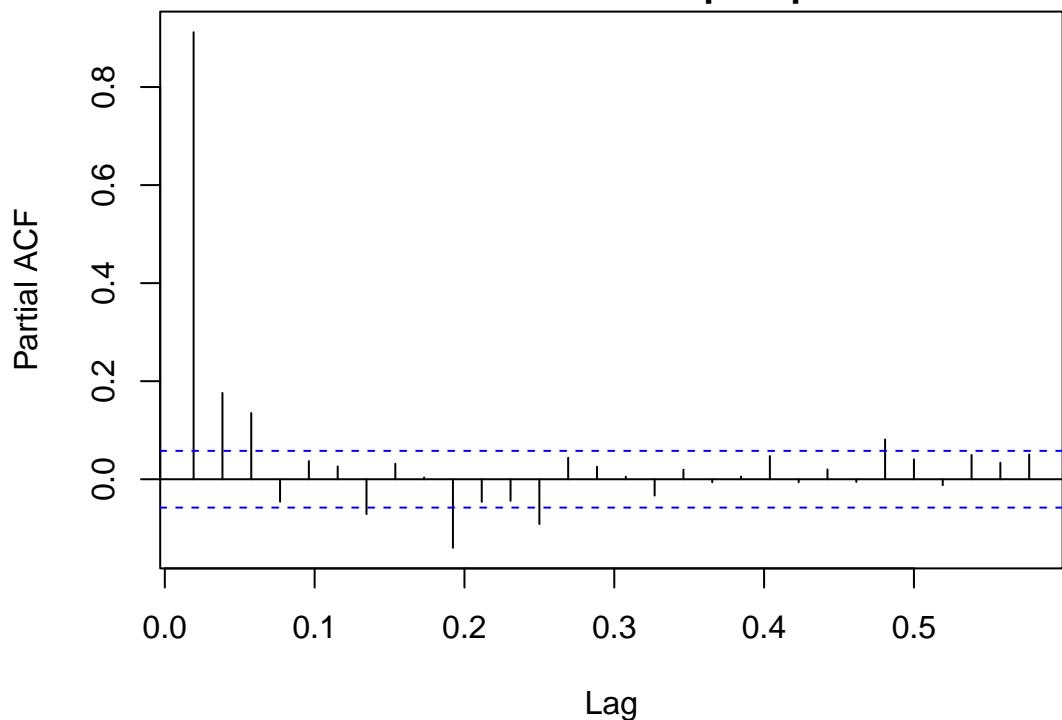
##                               ME      RMSE      MAE      MPE      MAPE
## Training set  3.604196e-05 0.0695939 0.05312883 -0.007032612 0.6595602
## Test set      -3.153926e-02 0.0864575 0.07483995 -0.403072769 0.9360303
##                         MASE      ACF1
## Training set 0.9789913 -0.1976876
## Test set      1.3790567       NA

# PLOO Excluding Outliers - Training partition (01/18/1999-01/04/2021)
# owt_df02_gb09_ploo_wkly02_train_tb03 (imputed w/out outliers set)

# SALINITY
ploo_arima_train_sal <- ts(owt_df02_gb09_ploo_wkly02_train_tb03$SALINITY,
                           start = c(1999, 3),
                           freq = 52)
# PLOO Excluding Outliers - Test partition (01/11/2021-01/03/2022)
ploo_arima_test_sal <- owt_df02_gb09_ploo_wkly02_test_tb03$SALINITY

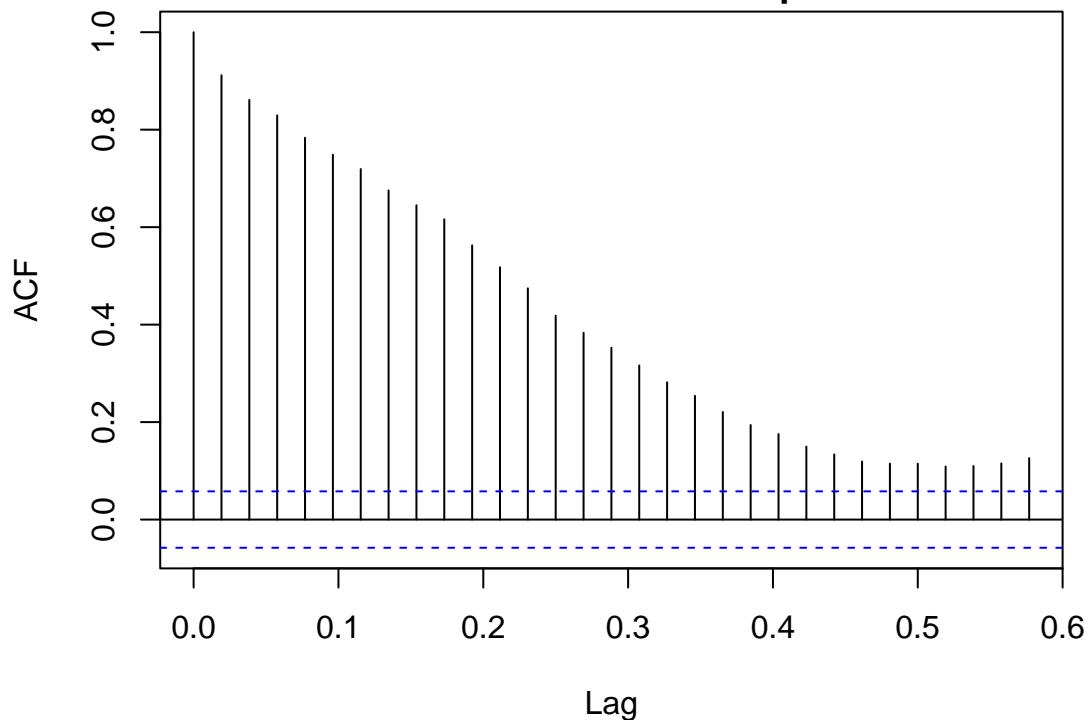
# pacf plot
par(mar=c(5, 4, 0, 2)+1.5)
pacf(ploo_arima_train_sal)
title("SALINITY at PLOO pacf plot")
```

SALINITY at PLOO pacf plot



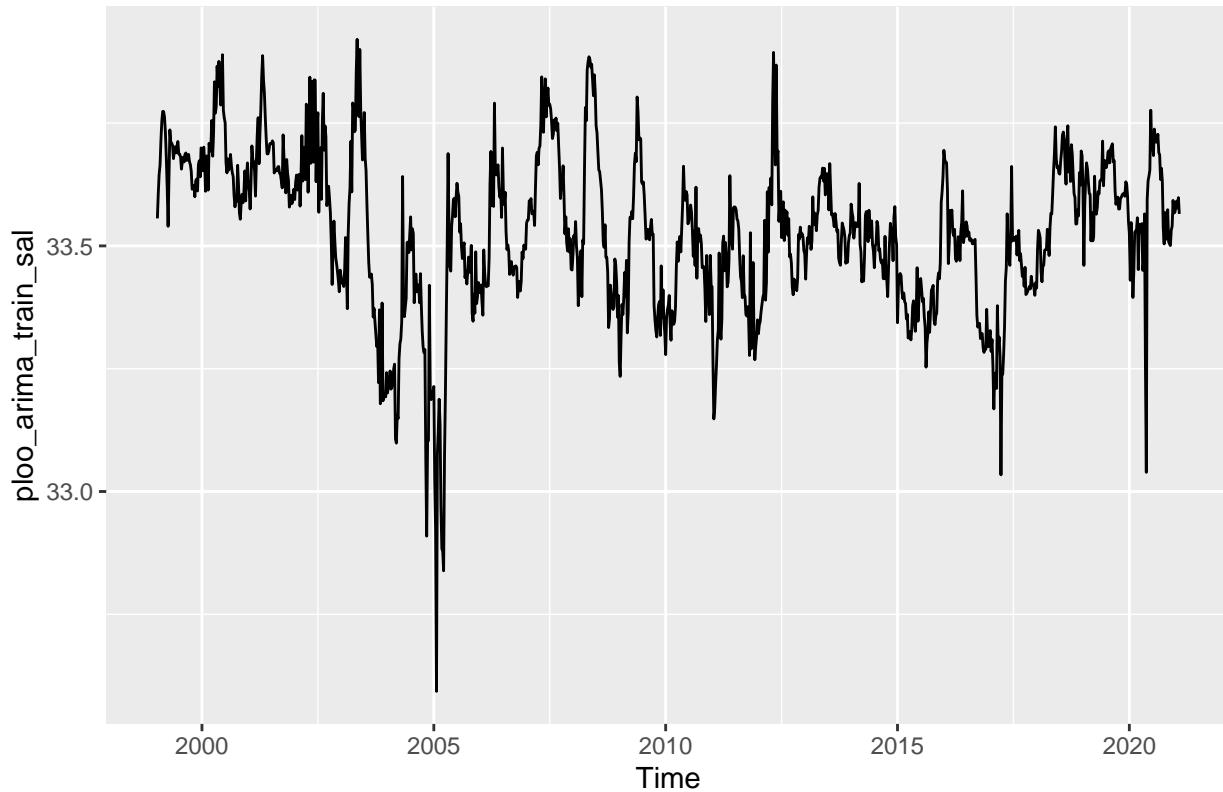
```
# p= 2  
  
# acf plot  
acf(ploo_arima_train_sal)  
title("SALINITY at PLOO acf plot")
```

SALINITY at PLOO acf plot



```
# q (acf plot) all out of bounds so 1  
  
# ts plot  
autoplot(ploo_arima_train_sal,  
         main = "SALINITY at PLOO ts plot")
```

SALINITY at PLOO ts plot



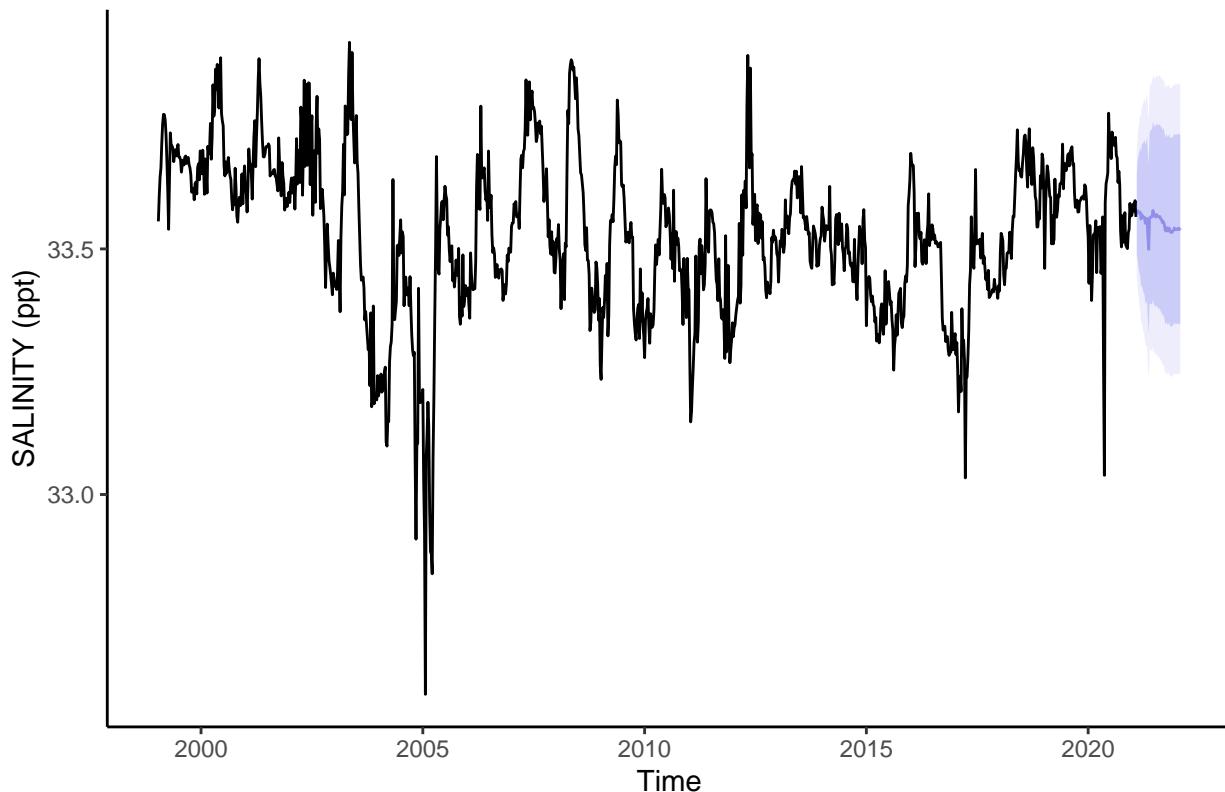
```
# seasonality, need P, D, Q
# P (pacf plot repeat out of bounds): 1
# D, like d, is 0
# Q (acf plot patterns), no pattern so 0

ploo_arima_sal <- arima(ploo_arima_train_sal,
                         order=c(2, 0, 1), seasonal = c(1,0,0))

# saving forecast using test set
ploo_arima_sal_forecast <- forecast(ploo_arima_sal, h=52)

# plotting the forecasts
autoplot(ploo_arima_train_sal,
         main = "PLOO ARIMA SALINITY",
         ylab = "SALINITY (ppt)") +
  autolayer(ploo_arima_sal_forecast, alpha=.3) +
  theme_classic()
```

PLOO ARIMA SALINITY



```
# looking at performance of the ARIMA .
ploo_arima_sal_acc <- accuracy(ploo_arima_sal_forecast, ploo_arima_test_sal)

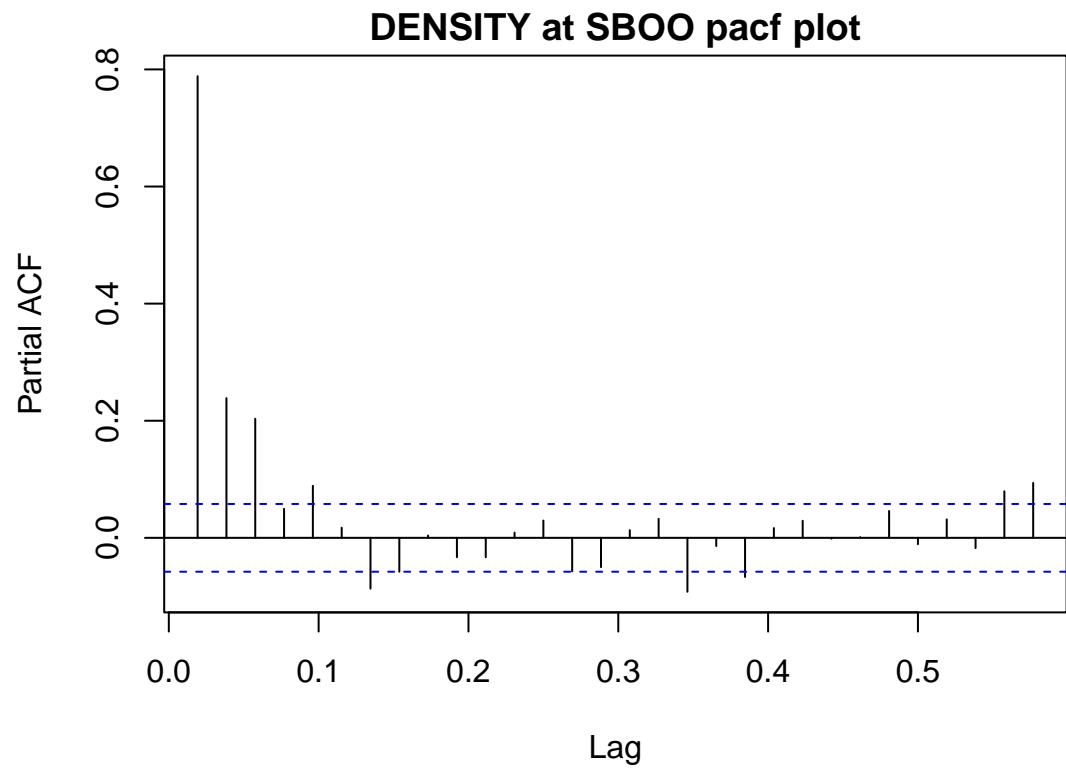
ploo_arima_sal_acc
```

```
##               ME      RMSE      MAE      MPE      MAPE
## Training set -2.878516e-05 0.06417251 0.04354409 -0.000461302 0.1299821
## Test set      2.807745e-02 0.08837713 0.06777939  0.082954959 0.2015667
##             MASE      ACF1
## Training set 0.9753143 0.005495697
## Test set      1.5181441          NA
```

ARIMA at SBOO

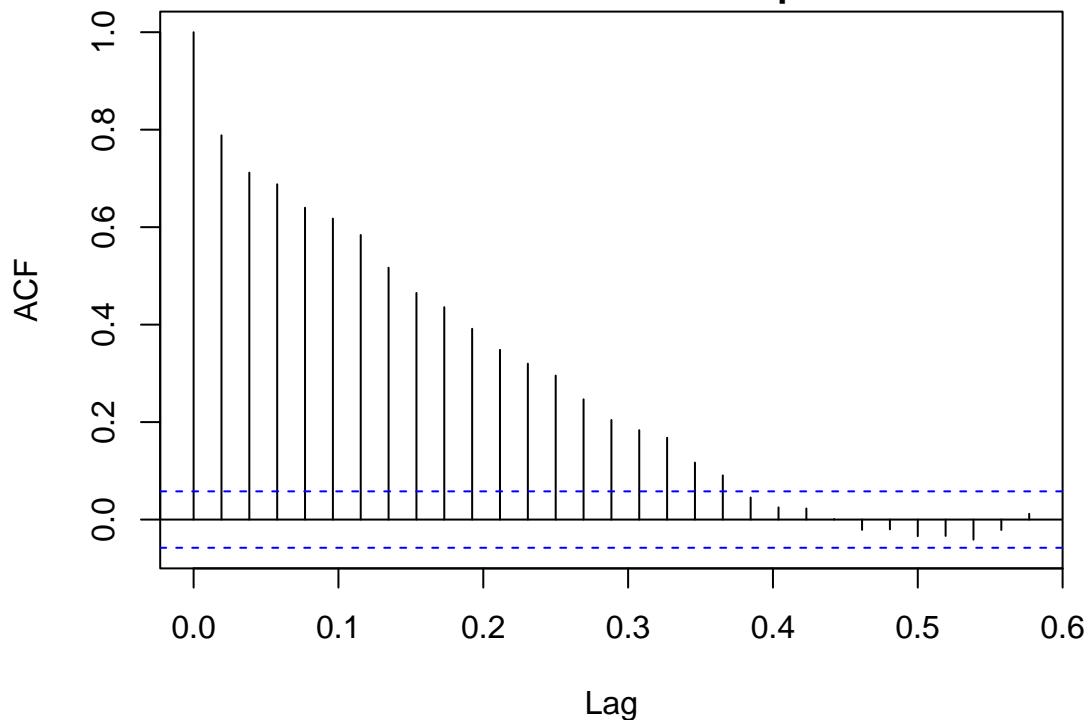
```
# DENSITY
sboo_arima_train_density <- ts(owt_df02_gb09_sboo_wkly02_train_tb03$DENSITY,
                                start = c(1999, 3),
                                freq = 52)
# SBOO Excluding Outliers - Test partition (01/11/2021-01/03/2022)
sboo_arima_test_density <- owt_df02_gb09_sboo_wkly02_test_tb03$DENSITY

# pacf plot
par(mar=c(5, 4, 0, 2)+1.5)
pacf(sboo_arima_train_density)
title("DENSITY at SBOO pacf plot")
```



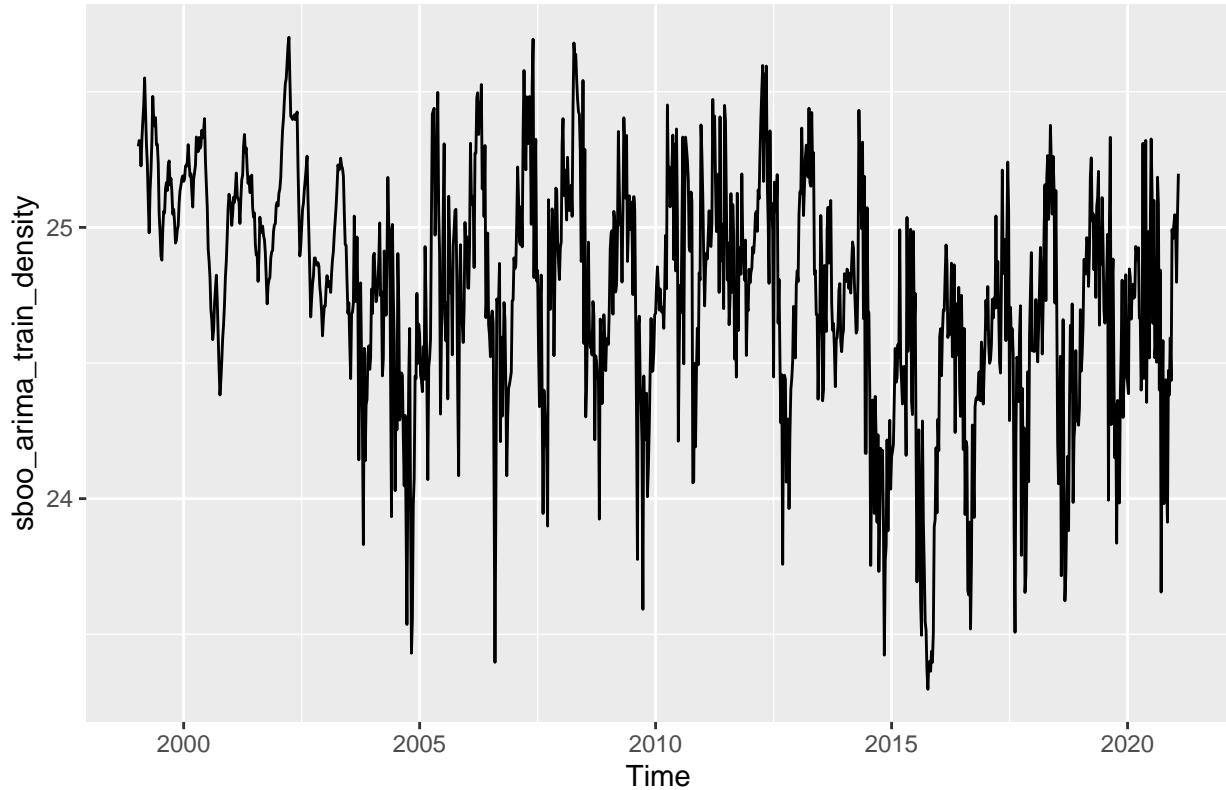
```
# p=2?  
  
# acf plot  
acf(sboo_arima_train_density)  
title("DENSITY at SBOO acf plot")
```

DENSITY at SBOO acf plot



```
# q value 1  
  
# ts plot  
autoplot(sboo_arima_train_density,  
         main = "DENSITY at SBOO ts plot")
```

DENSITY at SBOO ts plot



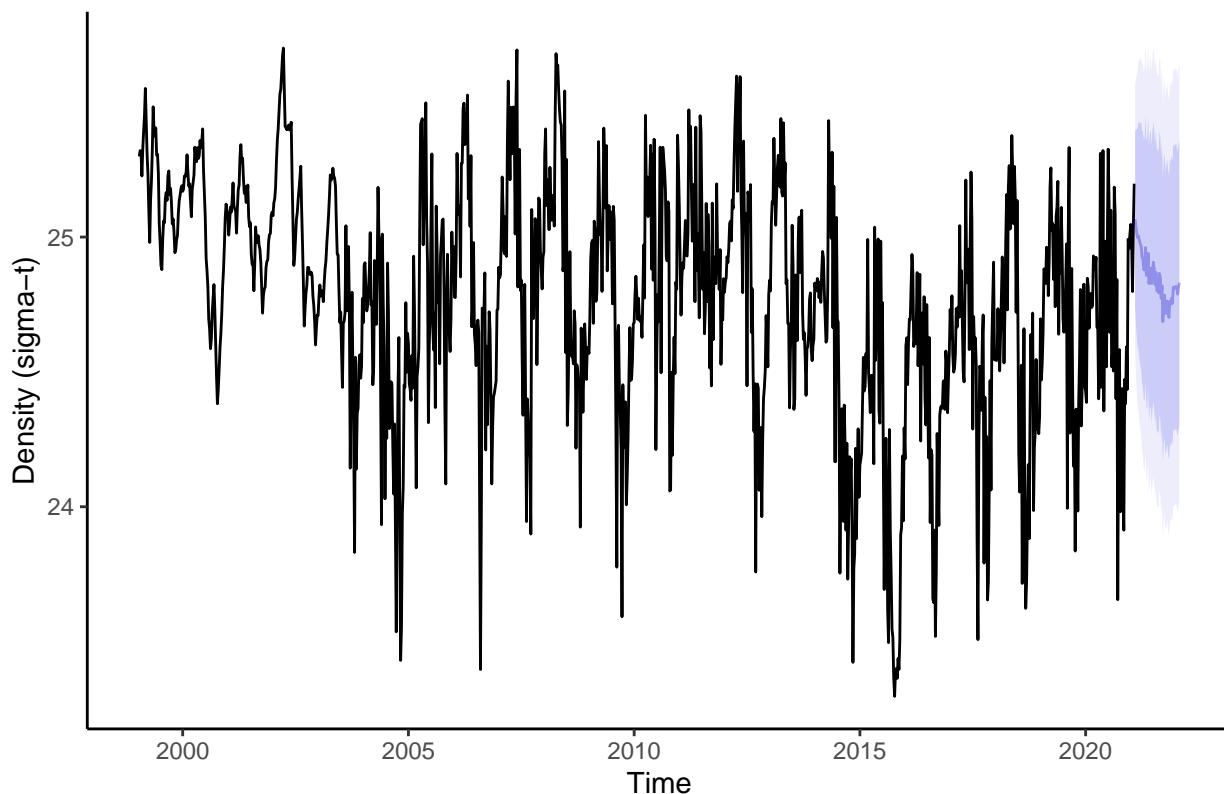
```
# definite seasonality so 2nd part needed
# d and D = 0
# P= 1 (pacf plot out of bounds not only beginning)
# Q (acf): Q= 0

sboo_arima_density <- arima(sboo_arima_train_density,
                             order=c(2, 0, 1), seasonal = c(1, 0, 0))

# saving forecast using test set
sboo_arima_density_forecast <- forecast(sboo_arima_density, h=52)

# plotting the forecasts
autoplot(sboo_arima_train_density,
         main = "SBOO ARIMA Density",
         ylab = "Density (sigma-t)") +
  autolayer(sboo_arima_density_forecast, alpha=.3) +
  theme_classic()
```

SBOO ARIMA Density



```
# looking at performance of the ARIMA
sboo_arima_density_acc <- accuracy(sboo_arima_density_forecast, sboo_arima_test_density)

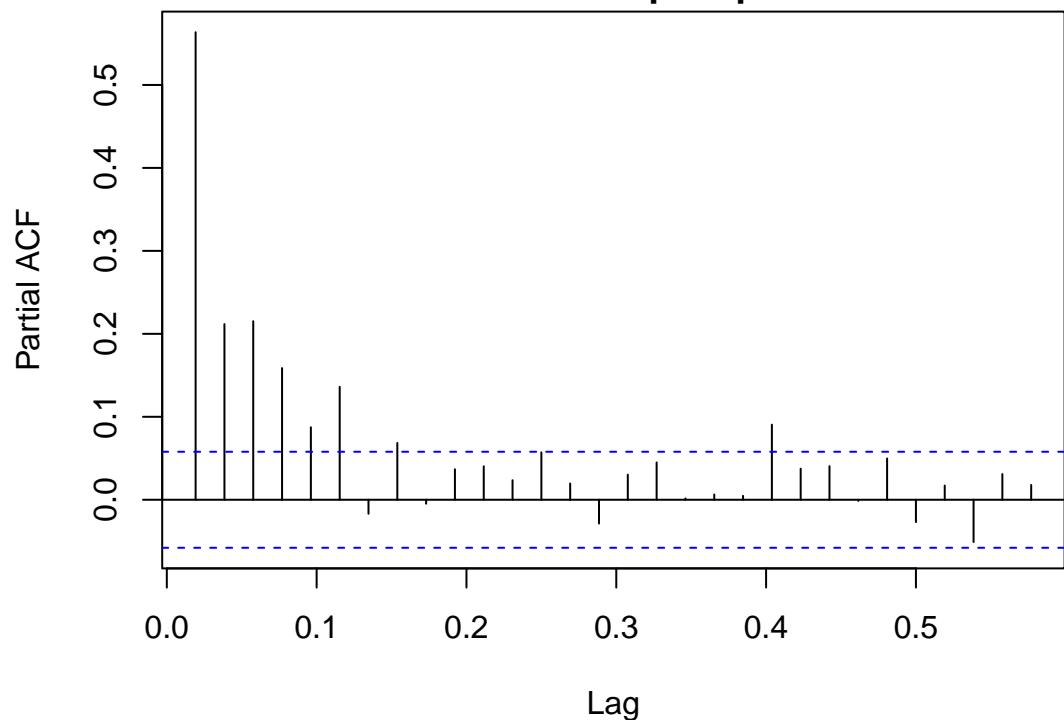
sboo_arima_density_acc

##               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -0.001087979 0.2519350 0.1829165 -0.01502515 0.7422643 0.9220391
## Test set      -0.054286261 0.3555327 0.2765508 -0.24064085 1.1193129 1.3940275
##                  ACF1
## Training set 0.004775348
## Test set      NA

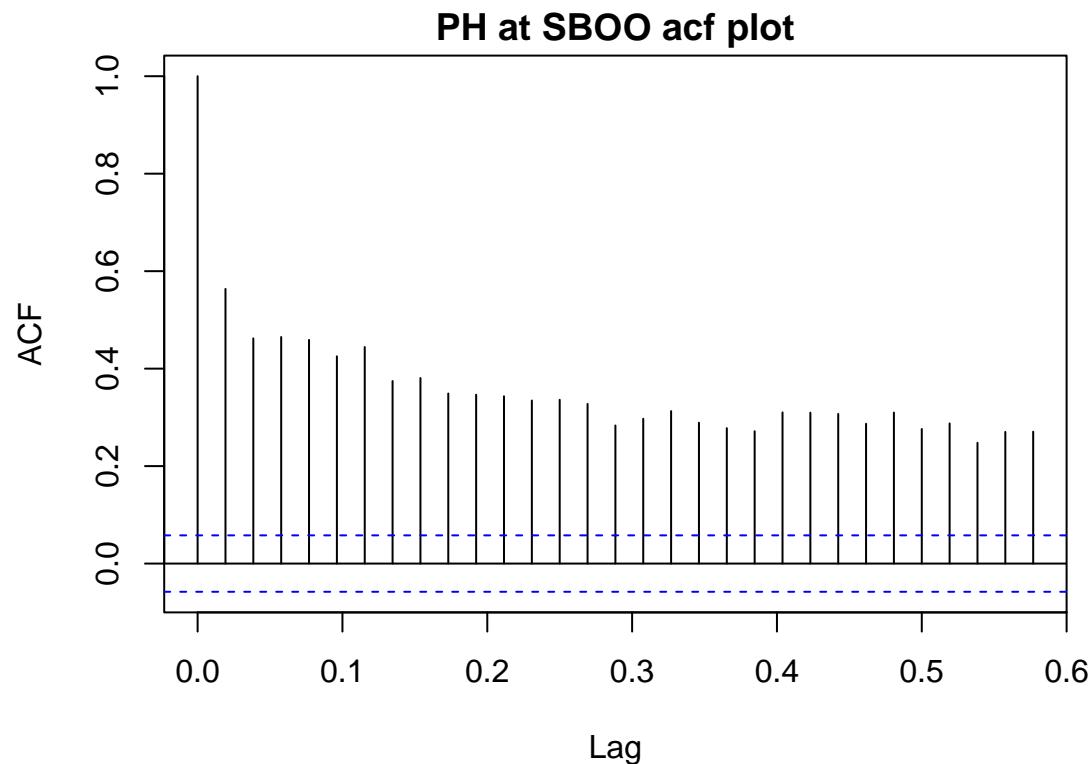
# PH
sboo_arima_train_ph <- ts(owt_df02_gb09_sboo_wkly02_train_tb03$PH,
                           start = c(1999, 3),
                           freq = 52)
# SBOO Excluding Outliers - Test partition (01/11/2021-01/03/2022)
sboo_arima_test_ph <- owt_df02_gb09_sboo_wkly02_test_tb03$PH

# pacf plot
par(mar=c(5, 4, 0, 2)+1.5)
pacf(sboo_arima_train_ph)
title("PH at SBOO pacf plot")
```

PH at SBOO pacf plot

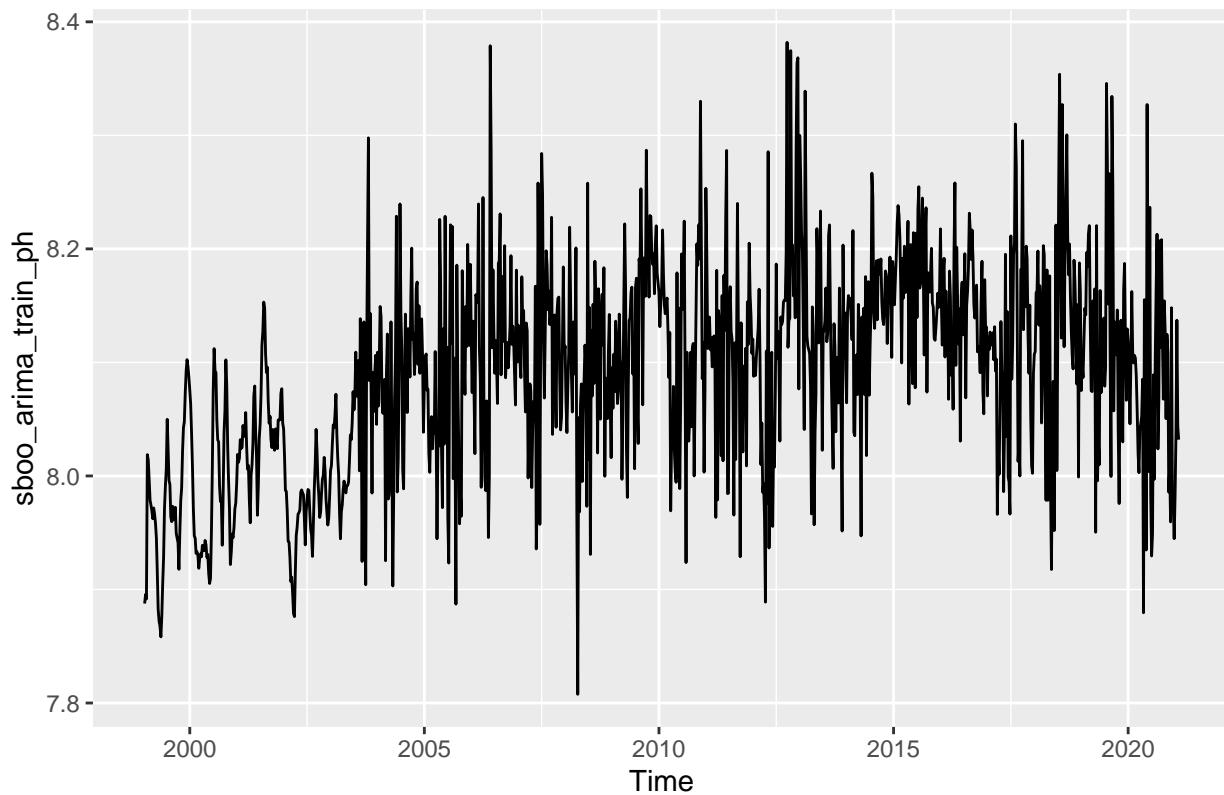


```
# p=2  
  
# acf plot  
acf(sboo_arima_train_ph)  
title("PH at SBOO acf plot")
```



```
# q 1  
  
# ts plot  
autoplot(sboo_arima_train_ph,  
         main = "PH at SBOO ts plot")
```

PH at SBOO ts plot



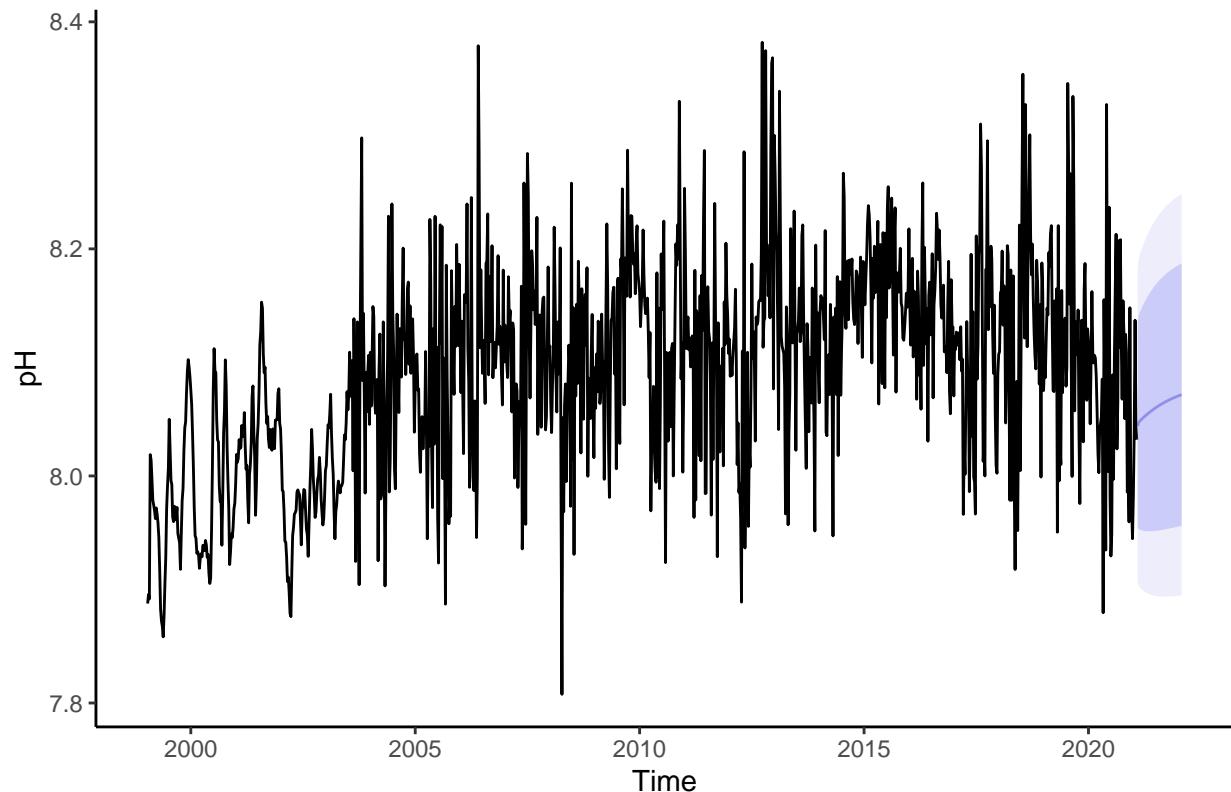
```
# hard to tell on seasonality maybe none
# d and D = 0

sboo_arima_ph <- arima(sboo_arima_train_ph,
                        order=c(2, 0, 1))

# saving forecast using test set
sboo_arima_ph_forecast <- forecast(sboo_arima_ph, h=52)

# plotting the forecasts
autoplot(sboo_arima_train_ph,
          main = "SBOO ARIMA pH",
          ylab = "pH") +
  autolayer(sboo_arima_ph_forecast, alpha=.3) +
  theme_classic()
```

SBOO ARIMA pH



```
# looking at performance of the ARIMA
sboo_arima_ph_acc <- accuracy(sboo_arima_ph_forecast, sboo_arima_test_ph)

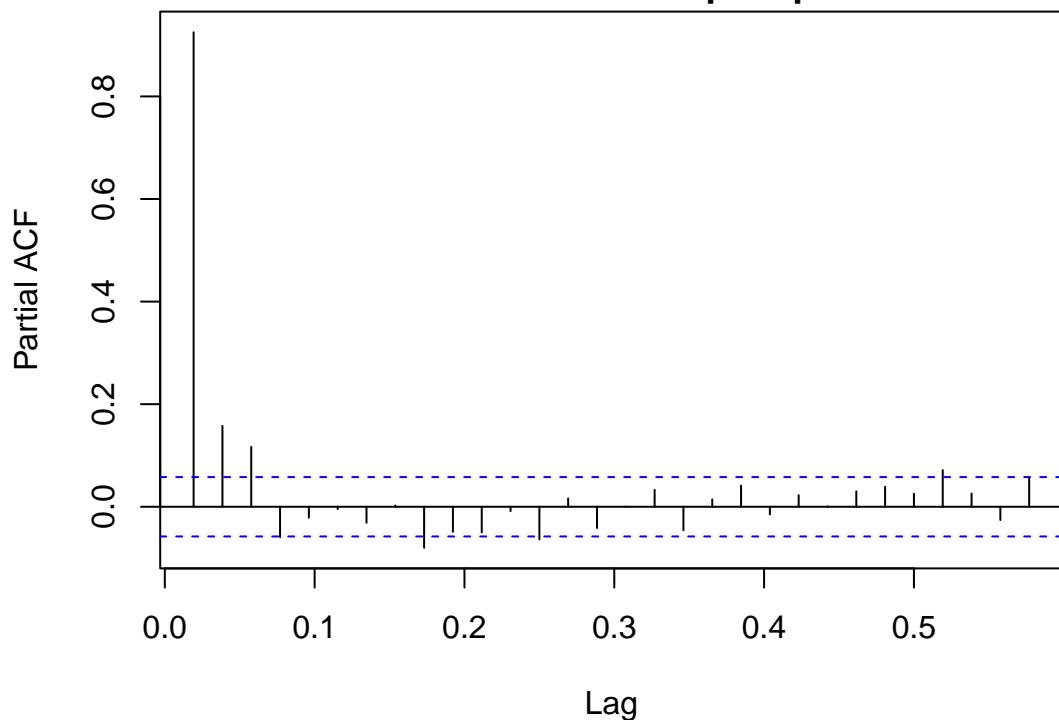
sboo_arima_ph_acc

##               ME      RMSE      MAE      MPE      MAPE
## Training set 0.000807955 0.06957934 0.05169269 0.002683095 0.6382645
## Test set     -0.005279801 0.11030998 0.08935559 -0.083728450 1.1090916
##             MASE      ACF1
## Training set 0.8672183 0.001676008
## Test set     1.4990670          NA

# SALINITY
sboo_arima_train_sal <- ts(owt_df02_gb09_sboo_wkly02_train_tb03$SALINITY,
                           start = c(1999, 3),
                           freq = 52)
# SBOO Excluding Outliers - Test partition (01/11/2021-01/03/2022)
sboo_arima_test_sal <- owt_df02_gb09_sboo_wkly02_test_tb03$SALINITY

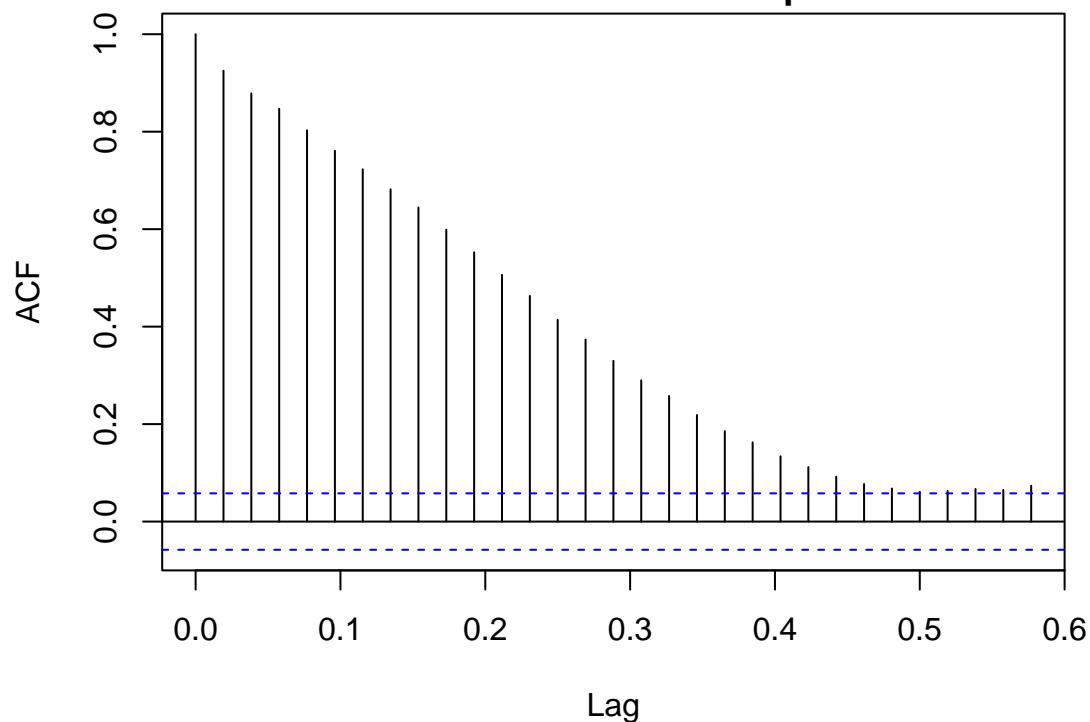
# pacf plot
par(mar=c(5, 4, 0, 2)+1.5)
pacf(sboo_arima_train_sal)
title("SALINITY at SBOO pacf plot")
```

SALINITY at SBOO pacf plot



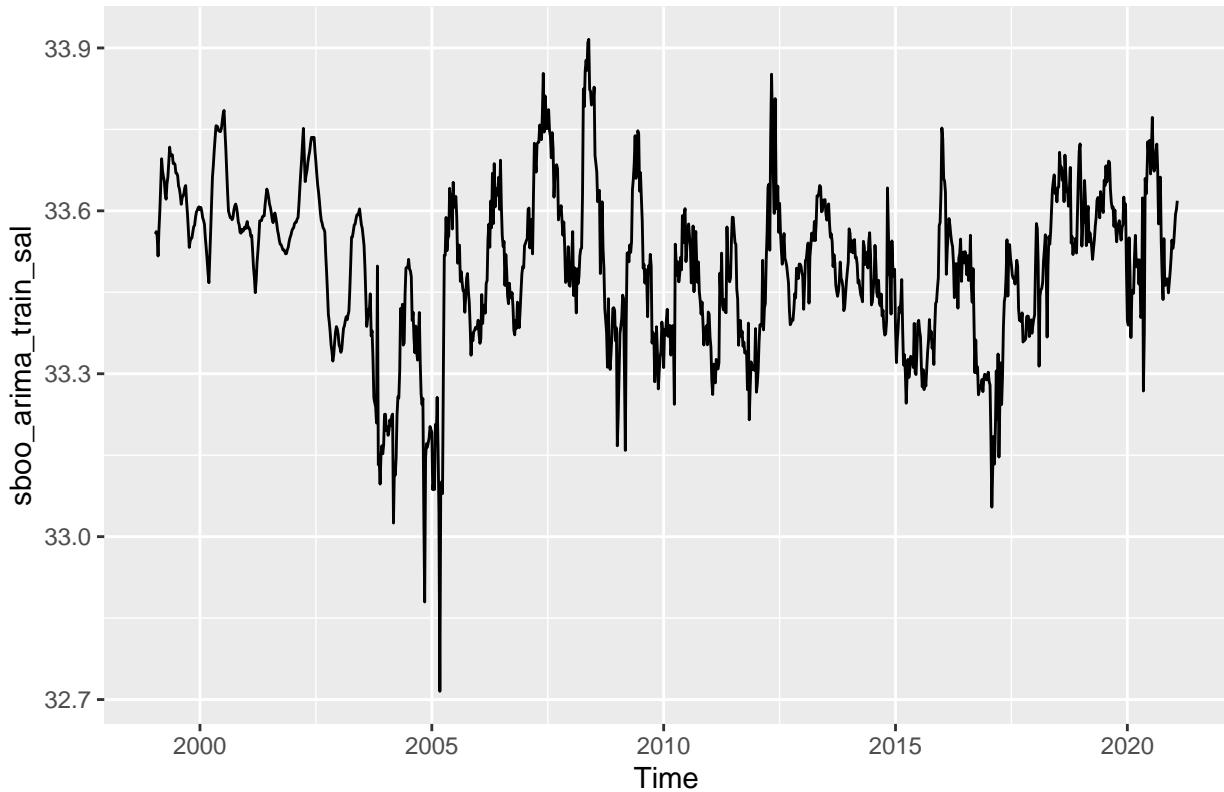
```
# p=1  
  
# acf plot  
acf(sboo_arima_train_sal)  
title("SALINITY at SBOO acf plot")
```

SALINITY at SBOO acf plot



```
# q 1  
  
# ts plot  
autoplot(sboo_arima_train_sal,  
         main = "SALINITY at SBOO ts plot")
```

SALINITY at SBOO ts plot



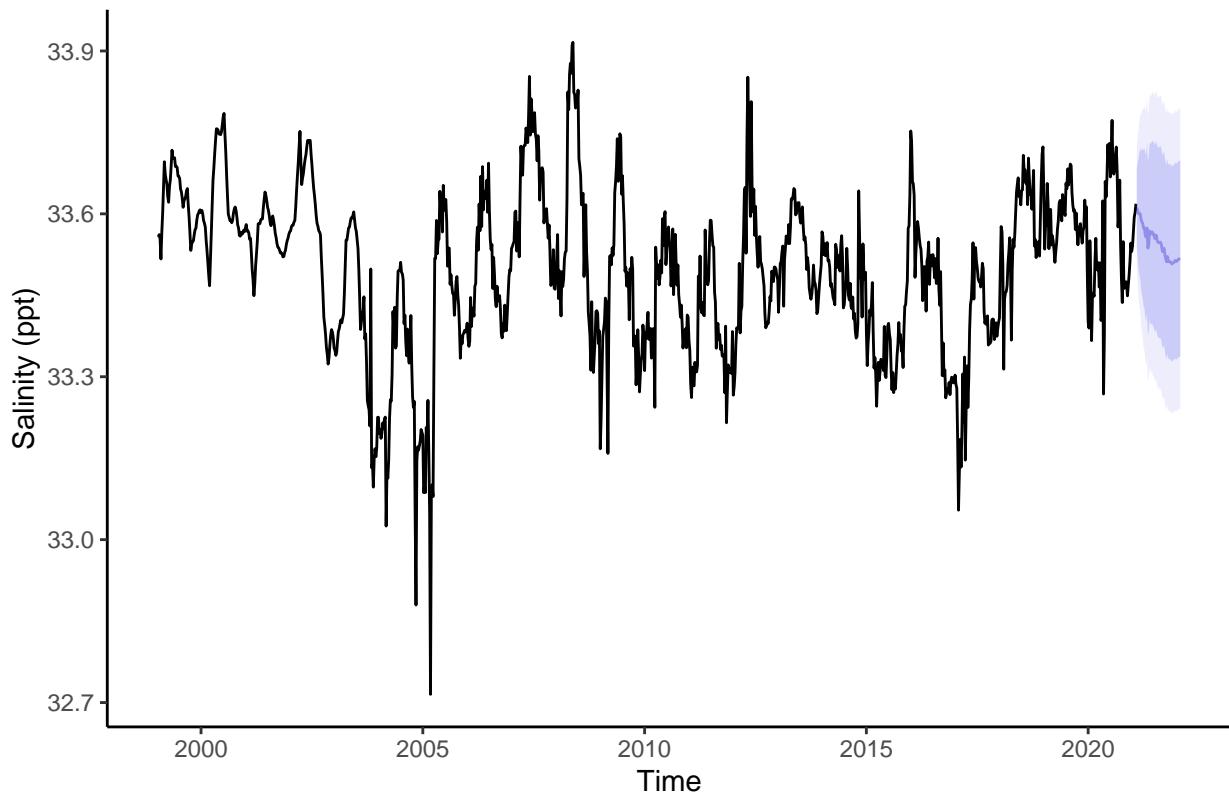
```
# seasonal plot
# P (pacf plot) = 1
# d and D = 0
# Q (acf plot) = 0

sboo_arima_sal <- arima(sboo_arima_train_sal,
                         order=c(1, 0, 1), seasonal = c(1,0,0))

# saving forecast using test set
sboo_arima_sal_forecast <- forecast(sboo_arima_sal, h=52)

# plotting the forecasts
autoplot(sboo_arima_train_sal,
         main = "SBOO ARIMA SALINITY",
         ylab = "Salinity (ppt)") +
  autolayer(sboo_arima_sal_forecast, alpha=.3) +
  theme_classic()
```

SBOO ARIMA SALINITY



```
# looking at performance of the ARIMA
sboo_arima_sal_acc <- accuracy(sboo_arima_sal_forecast, sboo_arima_test_sal)

sboo_arima_sal_acc

##                               ME      RMSE      MAE      MPE      MAPE
## Training set 2.058847e-05 0.05498850 0.03628011 -0.0002137381 0.1084051
## Test set      1.760262e-02 0.08751071 0.07237851  0.0517296153 0.2154528
##                               MASE      ACF1
## Training set 0.9872188 0.01400696
## Test set      1.9694928      NA

z <- get("ENTER0", owt_df02_gb09_ploo_wkly03_train_tb03)
ts_owt_df02_gb09_sboo_wkly02_train_tb03 <- ts(z, start = c(1999, 1), freq = 52)

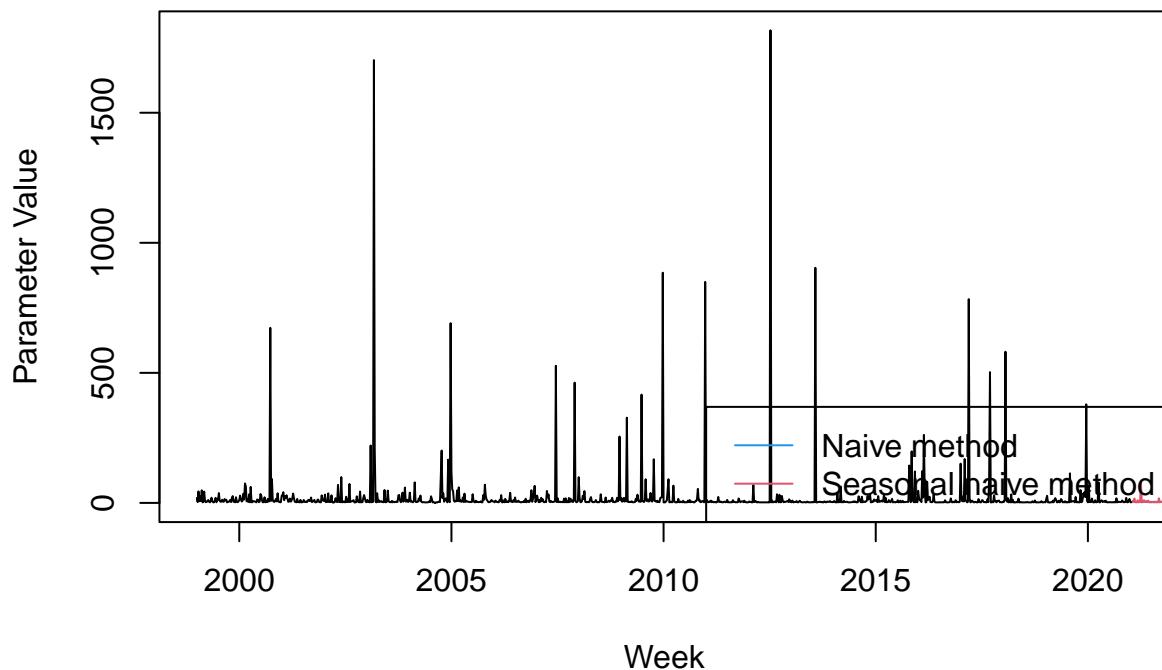
naive.rose <- naive(ts_owt_df02_gb09_sboo_wkly02_train_tb03, h=104)
plot(ts_owt_df02_gb09_sboo_wkly02_train_tb03, main="Forecast for weekly PL00 dataset",
     xlab="Week", ylab="Parameter Value")

snaive.rose <- snaive(ts_owt_df02_gb09_sboo_wkly02_train_tb03, h=104)
plot(ts_owt_df02_gb09_sboo_wkly02_train_tb03, main="Forecast for weekly PL00 dataset",
     xlab="Week", ylab="Parameter Value")

lines(naive.rose$mean, col=4)
lines(snaive.rose$mean, col=2)
```

```
legend("bottomright", lty=1, col=c(4,2),
      legend=c("Naive method", "Seasonal naive method"))
```

Forecast for weekly PLOO dataset



```
print(accuracy(snaive.rose))
```

```
##               ME      RMSE      MAE      MPE      MAPE MASE      ACF1
## Training set -0.2189141 143.2925 32.63567 -461.2263 511.0516    1 0.02976892
```

DENSITY

```
z_den <- get("DENSITY", owt_df02_gb09_ploo_wkly03_train_tb03)
ts_owt_df02_gb09_sboo_wkly02_train_tb03 <- ts(z_den, start = c(1999, 1), freq = 52)
snaive.den <- snaive(ts_owt_df02_gb09_sboo_wkly02_train_tb03, h=104)
print("DENSITY:")
```

```
## [1] "DENSITY:"
```

```
print(accuracy(snaive.den))
```

```
##               ME      RMSE      MAE      MPE      MAPE MASE      ACF1
## Training set -0.02520021 0.4384758 0.3346855 -0.1167214 1.35569    1 0.6311523
```

PH

```
z_ph <- get("PH", owt_df02_gb09_ploo_wkly03_train_tb03)
ts_owt_df02_gb09_sboo_wkly02_train_tb03 <- ts(z_ph, start = c(1999, 1), freq = 52)
snaive.ph <- snaive(ts_owt_df02_gb09_sboo_wkly02_train_tb03, h=104)
print("PH:")
```

```

## [1] "PH:"

print(accuracy(snaive.ph))

##               ME      RMSE      MAE      MPE      MAPE MASE
## Training set 0.005067217 0.1020739 0.07956865 0.0554673 0.9879985    1
##               ACF1
## Training set 0.4673687

# SALINITY
z_sal <- get("SALINITY", owt_df02_gb09_ploo_wkly03_train_tb03)
ts_owt_df02_gb09_sboo_wkly02_train_tb03 <- ts(z_sal, start = c(1999, 1), freq = 52)
snaive.sal <- snaive(ts_owt_df02_gb09_sboo_wkly02_train_tb03, h=104)
print("SALINITY:")

## [1] "SALINITY:"

print(accuracy(snaive.sal))

##               ME      RMSE      MAE      MPE      MAPE MASE
## Training set -0.004825046 0.1763696 0.1230404 -0.01577362 0.3677022    1
##               ACF1
## Training set 0.7814462

z <- get("ENTERO", owt_df02_gb09_ploo_wkly03_test_tb03)
ts_owt_df02_gb09_ploo_wkly03_test_tb03 <- ts(z, start = c(2021, 1), freq = 52)

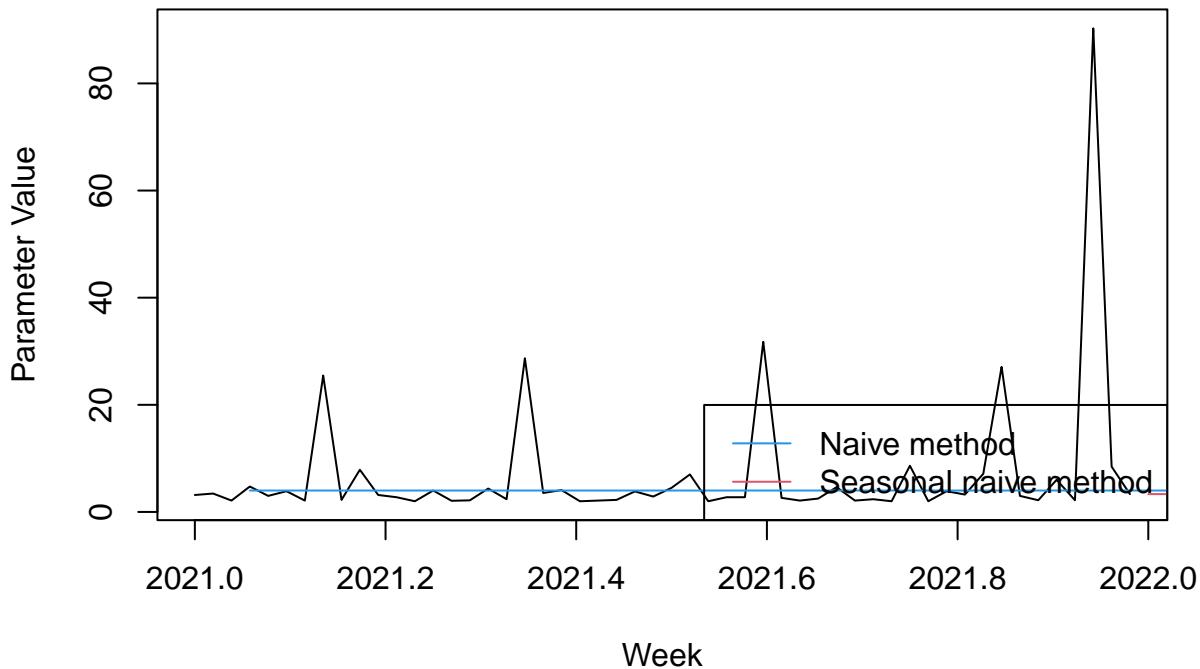
snaive.rose <- naive(ts_owt_df02_gb09_ploo_wkly03_test_tb03, h=104)
plot(ts_owt_df02_gb09_ploo_wkly03_test_tb03, main="Forecast for weekly PL00 dataset",
  ~ xlab="Week", ylab="Parameter Value")

lines(naive.rose$mean, col=4)
lines(snaive.rose$mean, col=2)

legend("bottomright", lty=1, col=c(4,2),
       legend=c("Naive method", "Seasonal naive method"))

```

Forecast for weekly PLOO dataset



```

print(accuracy(snaive.rose))

##               ME      RMSE      MAE      MPE      MAPE MASE      ACF1
## Training set 0.003267974 19.72166 8.882983 -103.4933 150.049  NaN -0.500478

# DENSITY
z_den <- get("DENSITY", owt_df02_gb09_ploo_wkly03_test_tb03)
ts_owt_df02_gb09_ploo_wkly03_test_tb03 <- ts(z_den, start = c(2021, 1), freq = 52)
snaive.den <- naive(ts_owt_df02_gb09_ploo_wkly03_test_tb03, h=104)
print("DENSITY:")

## [1] "DENSITY:"

print(accuracy(snaive.den))

##               ME      RMSE      MAE      MPE      MAPE MASE      ACF1
## Training set -0.007789891 0.3026302 0.2493652 -0.03865236 1.004043  NaN
##               ACF1
## Training set -0.4839433

# PH
z_ph <- get("PH", owt_df02_gb09_ploo_wkly03_test_tb03)
ts_owt_df02_gb09_ploo_wkly03_test_tb03 <- ts(z_ph, start = c(2021, 1), freq = 52)
snaive.ph <- naive(ts_owt_df02_gb09_ploo_wkly03_test_tb03, h=104)
print("PH:")

## [1] "PH:"

```

```

print(accuracy(snaive.ph))

##               ME        RMSE       MAE        MPE       MAPE MASE
## Training set -0.0002490054 0.08047558 0.06232518 -0.00815208 0.7769789  NaN
##                  ACF1
## Training set -0.4204429

# SALINITY
z_sal <- get("SALINITY", owt_df02_gb09_ploo_wkly03_test_tb03)
ts_owt_df02_gb09_ploo_wkly03_test_tb03 <- ts(z_sal, start = c(2021, 1), freq = 52)
snaive.sal <- naive(ts_owt_df02_gb09_ploo_wkly03_test_tb03, h=104)
print("SALINITY:")

## [1] "SALINITY:"

print(accuracy(snaive.sal))

##               ME        RMSE       MAE        MPE       MAPE MASE
## Training set -0.003188264 0.05168319 0.03787635 -0.009628083 0.1127649  NaN
##                  ACF1
## Training set -0.2564732

z <- get("ENTERO", owt_df02_gb09_sboo_wkly03_train_tb03)
ts_owt_df02_gb09_sboo_wkly03_train_tb03 <- ts(z, start = c(1999, 1), freq = 52)

naive.rose <- naive(ts_owt_df02_gb09_sboo_wkly03_train_tb03, h=104)
plot(ts_owt_df02_gb09_sboo_wkly03_train_tb03, main="Forecast for weekly PL00 dataset",
  ~ xlab="Week", ylab="Parameter Value")

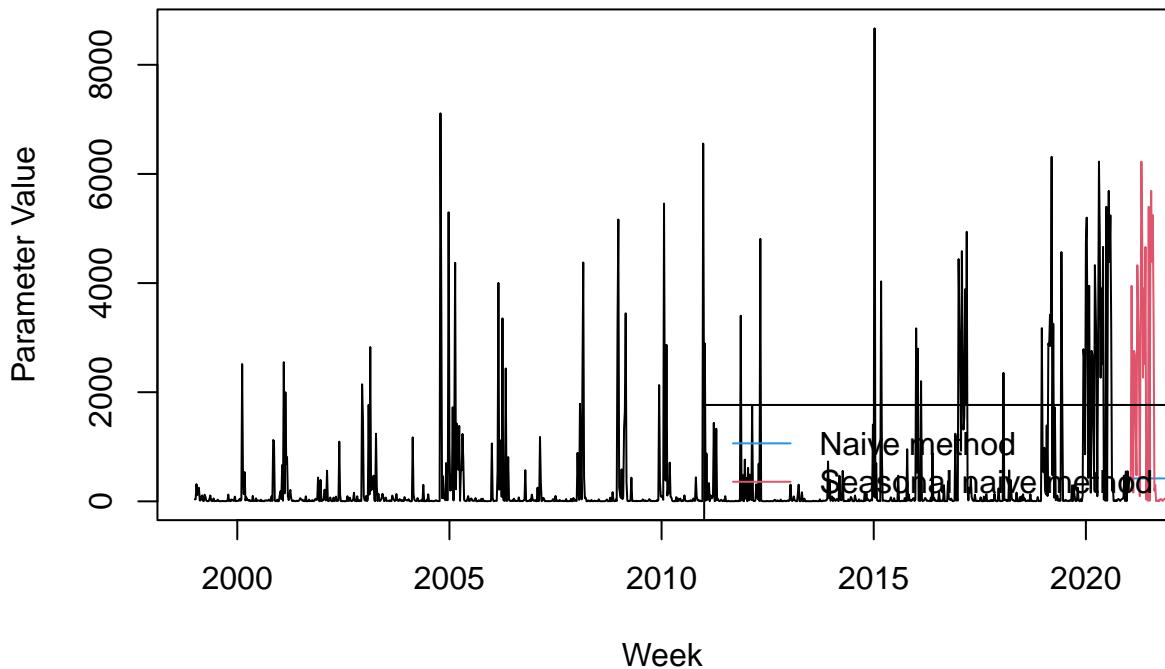
snaive.rose <- snaive(ts_owt_df02_gb09_sboo_wkly03_train_tb03, h=104)
plot(ts_owt_df02_gb09_sboo_wkly03_train_tb03, main="Forecast for weekly PL00 dataset",
  ~ xlab="Week", ylab="Parameter Value")

lines(naive.rose$mean, col=4)
lines(snaive.rose$mean, col=2)

legend("bottomright", lty=1, col=c(4,2),
      legend=c("Naive method", "Seasonal naive method"))

```

Forecast for weekly PLOO dataset



```

print(accuracy(snaive.rose))

##               ME      RMSE      MAE      MPE      MAPE MASE      ACF1
## Training set 74.85486 1257.591 489.9936 -1843.094 1913.031    1 0.2416428

# DENSITY
z_den <- get("DENSITY", owt_df02_gb09_sboo_wkly03_train_tb03)
ts_owt_df02_gb09_sboo_wkly03_train_tb03 <- ts(z_den, start = c(1999, 1), freq = 52)
snaive.den <- snaive(ts_owt_df02_gb09_sboo_wkly03_train_tb03, h=104)
print("DENSITY:")

## [1] "DENSITY:"
print(accuracy(snaive.den))

##               ME      RMSE      MAE      MPE      MAPE MASE      ACF1
## Training set -0.02267708 0.4369493 0.3392177 -0.1072116 1.377332    1 0.5948643

# PH
z_ph <- get("PH", owt_df02_gb09_sboo_wkly03_train_tb03)
ts_owt_df02_gb09_sboo_wkly03_train_tb03 <- ts(z_ph, start = c(1999, 1), freq = 52)
snaive.ph <- snaive(ts_owt_df02_gb09_sboo_wkly03_train_tb03, h=104)
print("PH:")

## [1] "PH:"
print(accuracy(snaive.ph))

```

```

##               ME      RMSE      MAE      MPE      MAPE MASE      ACF1
## Training set 0.004397847 0.1056579 0.08178623 0.04620683 1.009717    1 0.353987

# SALINITY
z_sal <- get("SALINITY", owt_df02_gb09_sboo_wkly03_train_tb03)
ts_owt_df02_gb09_sboo_wkly03_train_tb03 <- ts(z_sal, start = c(1999, 1), freq = 52)
snaive.sal <- snaive(ts_owt_df02_gb09_sboo_wkly03_train_tb03, h=104)
print("SALINITY:")

## [1] "SALINITY:"
print(accuracy(snaive.sal))

##               ME      RMSE      MAE      MPE      MAPE MASE      ACF1
## Training set -0.003068609 0.1492933 0.117386 -0.01015136 0.3507278    1
##               ACF1
## Training set 0.8635462

z <- get("ENTERO", owt_df02_gb09_sboo_wkly03_test_tb03)
ts_owt_df02_gb09_sboo_wkly03_test_tb03 <- ts(z, start = c(2021, 1), freq = 52)

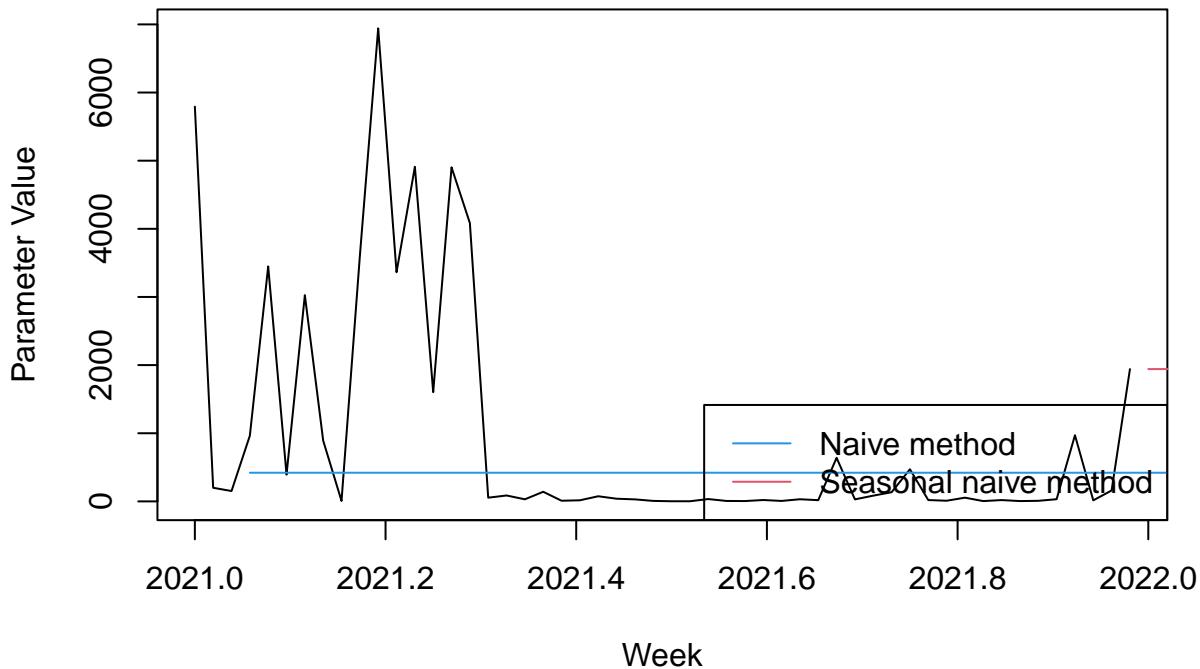
snaive.rose <- naive(ts_owt_df02_gb09_sboo_wkly03_test_tb03, h=104)
plot(ts_owt_df02_gb09_sboo_wkly03_test_tb03, main="Forecast for weekly PL00 dataset",
  ~ xlab="Week", ylab="Parameter Value")

lines(naive.rose$mean, col=4)
lines(snaive.rose$mean, col=2)

legend("bottomright", lty=1, col=c(4,2),
       legend=c("Naive method", "Seasonal naive method"))

```

Forecast for weekly PLOO dataset



```

print(accuracy(snaive.rose))

##               ME      RMSE      MAE      MPE      MAPE MASE      ACF1
## Training set -75.5289 1679.732 938.639 -757.3371 828.1079  NaN -0.2973567

# DENSITY
z_den <- get("DENSITY", owt_df02_gb09_sboo_wkly03_test_tb03)
ts_owt_df02_gb09_sboo_wkly03_test_tb03 <- ts(z_den, start = c(2021, 1), freq = 52)
snaive.den <- naive(ts_owt_df02_gb09_sboo_wkly03_test_tb03, h=104)
print("DENSITY:")

## [1] "DENSITY:"
print(accuracy(snaive.den))

##               ME      RMSE      MAE      MPE      MAPE MASE      ACF1
## Training set -0.008203548 0.3977213 0.3230145 -0.0458566 1.304932  NaN
##               ACF1
## Training set -0.5195655

# PH
z_ph <- get("PH", owt_df02_gb09_sboo_wkly03_test_tb03)
ts_owt_df02_gb09_sboo_wkly03_test_tb03 <- ts(z_ph, start = c(2021, 1), freq = 52)
snaive.ph <- naive(ts_owt_df02_gb09_sboo_wkly03_test_tb03, h=104)
print("PH:")

## [1] "PH:"

```

```

print(accuracy(snaive.ph))

##               ME      RMSE      MAE      MPE      MAPE MASE
## Training set -0.0008870215 0.1294695 0.1006093 -0.02398872 1.251453  NaN
##                      ACF1
## Training set -0.5671807

# SALINITY
z_sal <- get("SALINITY", owt_df02_gb09_sboo_wkly03_test_tb03)
ts_owt_df02_gb09_sboo_wkly03_test_tb03 <- ts(z_sal, start = c(2021, 1), freq = 52)
snaive.sal <- naive(ts_owt_df02_gb09_sboo_wkly03_test_tb03, h=104)
print("SALINITY:")

## [1] "SALINITY:"

print(accuracy(snaive.sal))

##               ME      RMSE      MAE      MPE      MAPE MASE
## Training set -0.00467507 0.05398767 0.03857742 -0.01408978 0.1149218  NaN
##                      ACF1
## Training set -0.2877513

```