Real-Time Citizen Science Data Visualization

William Norfolk

11/1/2019

This Rmd serves as a cleaning and analysis script to produce simple water quality data visualizations collected through the MarineLab citizen science water quality monitoring program. The purplose of this script is to porduce small visualizations that can be used to educate participants about the data they collected in the field at the conclusion of their educational program.

This script functions using the excel template included in the *data* folder within the *individual\_program\_data* subfolder. Field data from individual programs should be entered into a copied version of the blank excel template and saved with a unique name to the *individual\_program\_data* folder. Following data entry, the newly saved data sheet must be loaded into this script by updating the relative path (detailed in the readme in the *data* folder).

Load the required packages to run the script. If an error occurs that reads *“there is no package called…”* install the named package in the error in the console (the bottom left hand screen in Rstudio) using the syntax *install.packages(“package name”)*.

library(readxl)  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(tidyverse)

## -- Attaching packages --------------------------------------------------------------- tidyverse 1.2.1 --

## v ggplot2 3.2.1 v readr 1.3.1  
## v tibble 2.1.3 v purrr 0.3.3  
## v tidyr 1.0.0 v stringr 1.4.0  
## v ggplot2 3.2.1 v forcats 0.4.0

## -- Conflicts ------------------------------------------------------------------ tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

library(forcats)  
library(ggthemes)  
library(knitr)  
library(naniar)  
library(gridExtra)

##   
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':  
##   
## combine

library(ggpubr)

## Loading required package: magrittr

##   
## Attaching package: 'magrittr'

## The following object is masked from 'package:purrr':  
##   
## set\_names

## The following object is masked from 'package:tidyr':  
##   
## extract

Load the data using the realitve path. *This is the only part of the script where any modification of code should occur!*

To update the script with the new data from a given program ensure that the program’s excel file is saved in the *individual\_program\_data* folder and change the file location in the path below. To change the path delete *only* the following text: *individual\_program\_data\_template\_example* replace this text with name you gave the new excel file for the program. Be aware R is case and space sensitive so if you named your file *VALPO Data* your new relative path will be *“../../data/individual\_program\_data/VALPO Data.xlsx”*

wqdata <- readxl::read\_excel("../../data/individual\_program\_data/individual\_program\_data\_template\_example.xlsx")

Once you have updated the relative path, you can run the script to produce a word document containing the generate visuals. To run click the *knit* buttom located in the upper left side of the script screen (the button has a yarn icon on it). Select *knit to Word* or simply click the button (the default setting is knit to word).

If the knit was successful, Rstudio should readily generate a word document with all of the code and commentary pasted directly into the files. Visualizations can be directly copied from the word document and pasted into a PowerPoint (or other presentation program) for viewing. *To avoid cluttering* of the folder, drag the generated docx files(it will be titled citizen\_science\_processing\_script.docx) into the *knitted\_documents* subfolder in the *processing code* folder. Once moved, rename the docx file with a unique identifier (i.e VALPO\_2018). If you do not rename the folder output files will be overwritten when moved ot the knitted documents folder because they will have identical names.

The code chunks from here down contain all of the processing code. I will provide commentary about each section, however no modifications are required to run the script (nor should they be made, unless you are familar with R). Feel free to read the commentarty to learn more about the function of each individual code chunk.

Start by taking a look at the data.

glimpse(wqdata)

## Observations: 5  
## Variables: 11  
## $ numeric\_date <chr> "11.20.12", "11.20.12", "11.20.12", "11.20.12...  
## $ program\_name <chr> "VALPO", "VALPO", "VALPO", "VALPO", "VALPO"  
## $ site\_name <chr> "Molasses", "Molasses", "Molasses", "Bush Poi...  
## $ military\_time <dbl> 900, 900, 1000, 1200, 1300  
## $ ph <dbl> 8.0, 8.0, 9.0, 7.5, 6.5  
## $ ammonia <dbl> 0.00, 0.25, 0.25, 1.00, 0.50  
## $ dissolved\_oxygen <dbl> 5.0, 6.0, 5.5, 5.0, 4.0  
## $ water\_temp <dbl> 26, 27, 29, 29, 28  
## $ salinity <dbl> 35, 35, 35, 20, 21  
## $ island\_side <chr> "ocean", "ocean", "ocean", "bay", "bay"  
## $ site\_type <chr> "Reef", "Reef", "Patch Reef/Hardbottom", "Sea...

You should see our dataset contains 11 variables. The number of observations will be dependant on the amount of data collected. For this example there are 5 observations. Looking at the variables 5 of them are assigned as *character* variables and 6 are *double* (numeric). The class of variable is very important to the proper functioning of R. The *double* variables are just fine the way they are, however some of the character variables need to be converted to factors. Factor variables are essentially variables that group data into different categories.

We will recode the variables *site\_type and island\_side* to factors.

wqdata$site\_type <- as.factor(as.character(wqdata$site\_type))  
  
wqdata$island\_side <- as.factor(as.character(wqdata$island\_side))

Next we need to ensure there are no NA values in the data. Since each program is likely to have a low number of observations it is important that there are no NAs in the data, as such every effort should be taken in the field to ensure all measures are recorded. To keep the script simple to use we will have to drop measurements that have NA values for any of the variables so it is important to try to reduce this occurance. The code chunk below drops any observations that have an NA value for any of the variables. If the data contains no NA values the code chunk below should do nothing to the data (this is the ideal occurance).

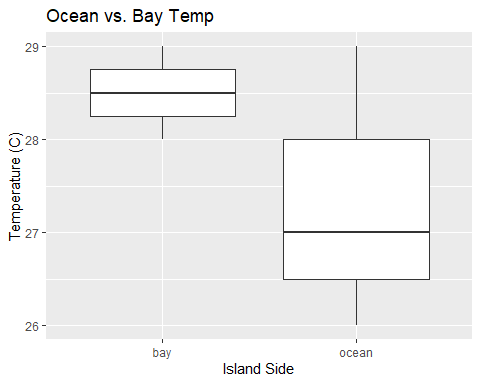
na\_check <- drop\_na(wqdata)

Now we can start making some visuals!

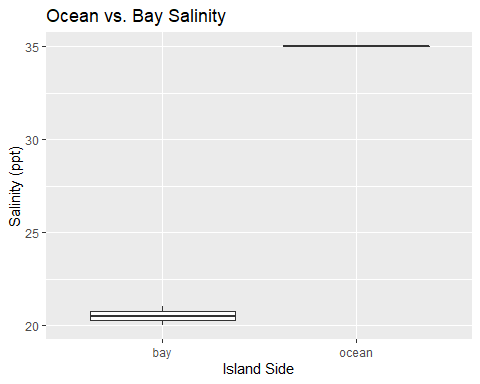
Lets start by comparing all of the water quality parameters with the ocean side and bay side.

This code chunk produces individual box plots so the individual water quality parameters can be compared between ocean and bay side.

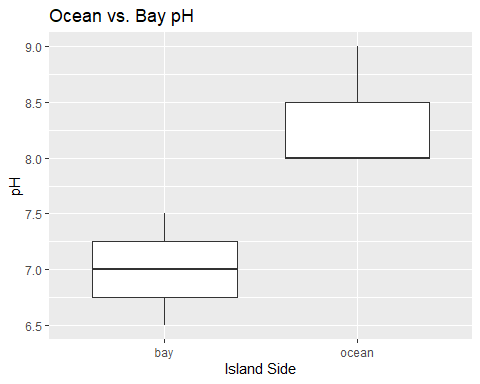
ocean\_vs\_bay\_temp <- ggplot(wqdata, aes(x = island\_side, y = water\_temp)) + geom\_boxplot() + ggtitle("Ocean vs. Bay Temp") + xlab("Island Side") + ylab("Temperature (C)")  
  
ocean\_vs\_bay\_temp



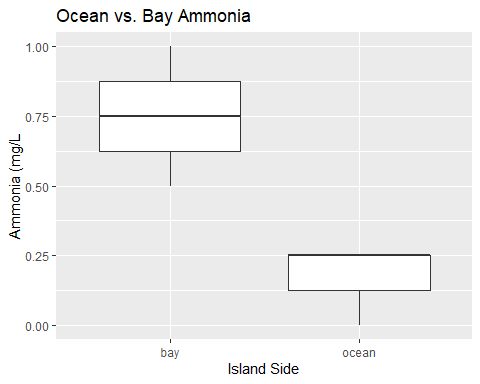
ocean\_vs\_bay\_salinity <- ggplot(wqdata, aes(x = island\_side, y = salinity)) + geom\_boxplot() + ggtitle("Ocean vs. Bay Salinity") + xlab("Island Side") + ylab("Salinity (ppt)")  
  
ocean\_vs\_bay\_salinity



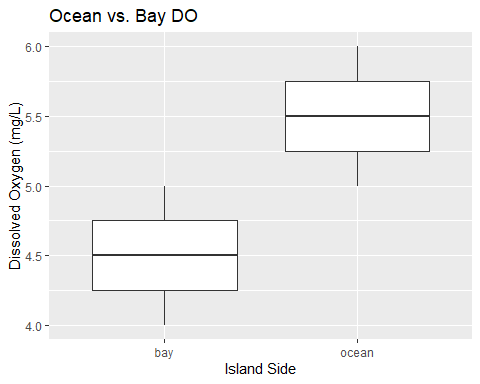
ocean\_vs\_bay\_ph <- ggplot(wqdata, aes(x = island\_side, y = ph)) + geom\_boxplot() + ggtitle("Ocean vs. Bay pH") + xlab("Island Side") + ylab("pH")  
  
ocean\_vs\_bay\_ph



ocean\_vs\_bay\_ammonia <- ggplot(wqdata, aes(x = island\_side, y = ammonia)) + geom\_boxplot() + ggtitle("Ocean vs. Bay Ammonia") + xlab("Island Side") + ylab("Ammonia (mg/L")  
  
ocean\_vs\_bay\_ammonia

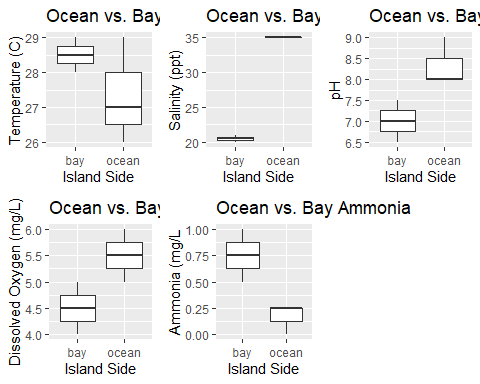


ocean\_vs\_bay\_do <- ggplot(wqdata, aes(x = island\_side, y = dissolved\_oxygen)) + geom\_boxplot() + ggtitle("Ocean vs. Bay DO") + xlab("Island Side") + ylab("Dissolved Oxygen (mg/L)")  
  
ocean\_vs\_bay\_do



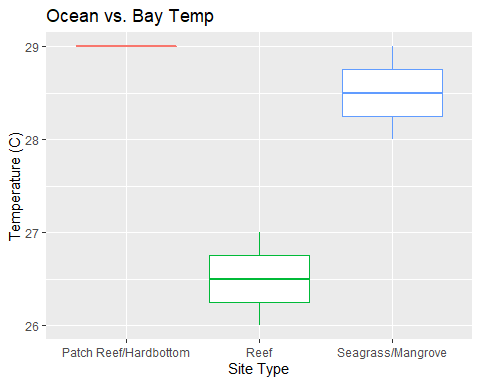
The code chunk below take the individual plots produced above places them onto a combined viewing pane if this visual is more informative.

grid.arrange(ocean\_vs\_bay\_temp, ocean\_vs\_bay\_salinity, ocean\_vs\_bay\_ph, ocean\_vs\_bay\_do, ocean\_vs\_bay\_ammonia, nrow = 2)

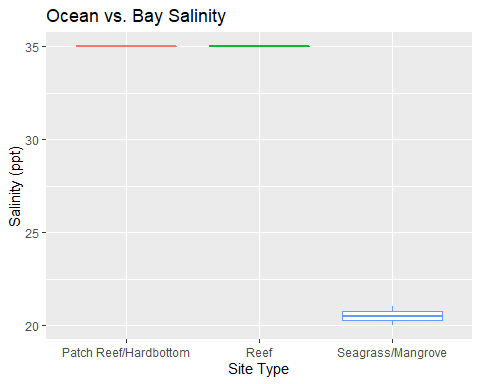


The code chunk below generates the same boxplots as above, but instead seperated parameters by site type.

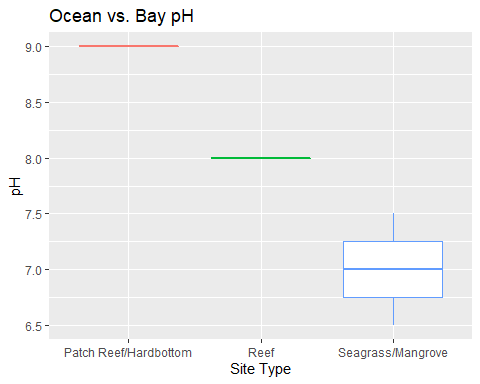
site\_temp <- ggplot(wqdata, aes(x = site\_type, y = water\_temp, color = site\_type)) + geom\_boxplot() + ggtitle("Ocean vs. Bay Temp") + xlab("Site Type") + ylab("Temperature (C)") + theme(legend.position = "none")  
  
site\_temp



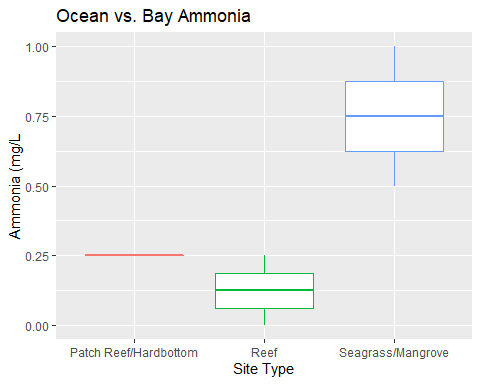
site\_salinity <- ggplot(wqdata, aes(x = site\_type, y = salinity, color = site\_type)) + geom\_boxplot() + ggtitle("Ocean vs. Bay Salinity") + xlab("Site Type") + ylab("Salinity (ppt)") + theme(legend.position = "none")  
  
site\_salinity



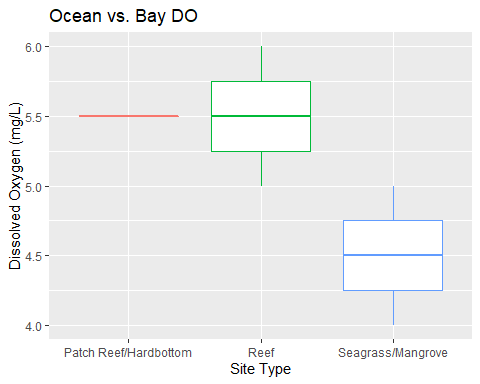
site\_ph <- ggplot(wqdata, aes(x = site\_type, y = ph, color = site\_type)) + geom\_boxplot() + ggtitle("Ocean vs. Bay pH") + xlab("Site Type") + ylab("pH") + theme(legend.position = "none")  
  
site\_ph



site\_ammonia <- ggplot(wqdata, aes(x = site\_type, y = ammonia, color = site\_type)) + geom\_boxplot() + ggtitle("Ocean vs. Bay Ammonia") + xlab("Site Type") + ylab("Ammonia (mg/L") + theme(legend.position = "none")  
  
site\_ammonia



site\_do <- ggplot(wqdata, aes(x = site\_type, y = dissolved\_oxygen, color = site\_type)) + geom\_boxplot() + ggtitle("Ocean vs. Bay DO") + xlab("Site Type") + ylab("Dissolved Oxygen (mg/L)") + theme(legend.position = "none")  
  
site\_do

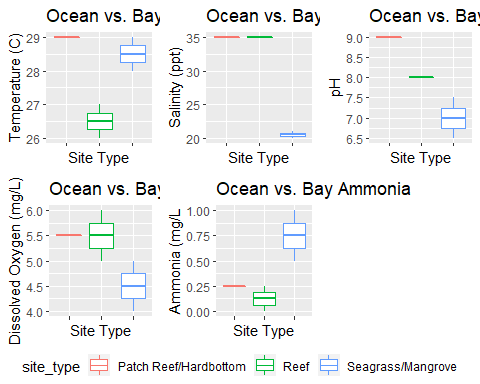


#site\_do\_2

Now the same as above for the combined view. This structure requires a bit more coding to account for overlapping x-axis tick names. Also we will use garrange instead of grid.Extra to place the plots because it is friendlier to legend positions.

site\_temp\_2 <- site\_temp + theme(axis.text.x = element\_blank())  
  
site\_sal\_2 <- site\_salinity + theme(axis.text.x = element\_blank())  
  
site\_ammonia\_2 <- site\_ammonia + theme(axis.text.x = element\_blank())  
  
site\_ph\_2 <- site\_ph + theme(axis.text.x = element\_blank())  
  
site\_do\_2 <- site\_do + theme(axis.text.x = element\_blank())

ggarrange(site\_temp\_2, site\_sal\_2, site\_ph\_2, site\_do\_2, site\_ammonia\_2, ncol = 3, nrow = 2, common.legend = TRUE, legend = "bottom")



Thus this script produces a very simple collection of plots that are readily producable and presentable for quick visualization of data gathered in the field. The hope is that this script will provide students visuals the actual data they have collected in the field that are quick to produce and easy to understand. Through the usage of data visualization we can inspire future citizen scientists and continued/collaborative data collection.

For troubleshooting, errors, explicit explinations of code contents, or further code development please contact William Norfolk at [william.norfolk@uga.edu](mailto:william.norfolk@uga.edu).