**Routine Data Graphing in R: FGS**

**R prep**

1. Install R and RStudio
2. Install ggplot2 package

**Dataset Prep**

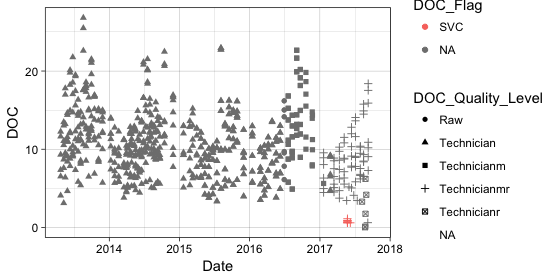
1. Download full FGS dataset
2. Delete all tabs except the data tab
3. Select header row, and use ‘search-replace’ function to replace all the spaces (“ “) with underscores (“\_”).
4. Import to RStudio
5. Begin work in R.
   1. Use code below but you’ll need to update the name of the data frame. This can be done in one step with the ‘search-replace’ tool in Word.

**Graphing in R – Using DOC as example variable**

**1. Status of QC work:**

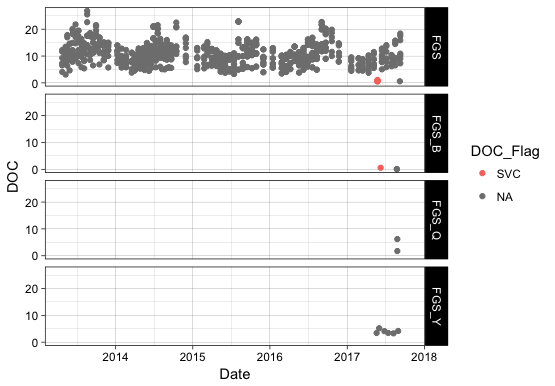
* 1. **What has and hasn’t been QC’d? (QC level)**
  2. **What are the known problem values? (QC flags)**

> ggplot(X2017\_10\_20\_142151\_HakaiData\_fgs, aes(x=Date, y=DOC)) + geom\_point(aes(shape=DOC\_Quality\_Level, color=DOC\_Flag)) + theme\_linedraw()



**2. View by survey: does each survey have the expected range of values?**

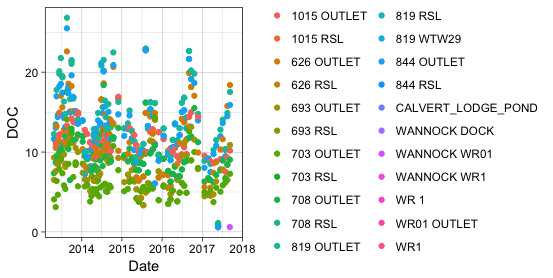
> ggplot(X2017\_10\_20\_142151\_HakaiData\_fgs, aes(x=Date, y=DOC)) + theme\_linedraw() + geom\_point(aes(color=DOC\_Flag)) + facet\_grid(Survey~.)



**3. River by river: does each river show the expected range of values?**

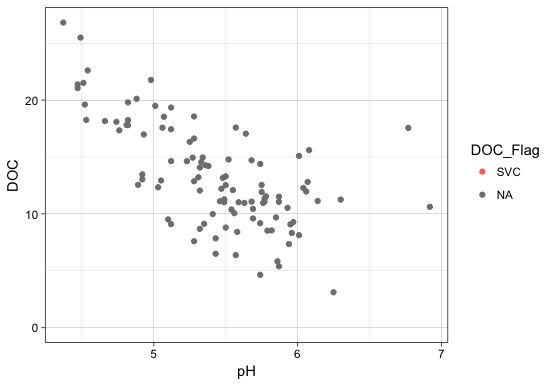
#Subset by Survey within graphing command

ggplot(subset(X2017\_10\_20\_142151\_HakaiData\_fgs, Survey == "FGS"), aes(x=Date, y=DOC)) + geom\_point(aes(colour=Site\_ID)) + theme\_linedraw()

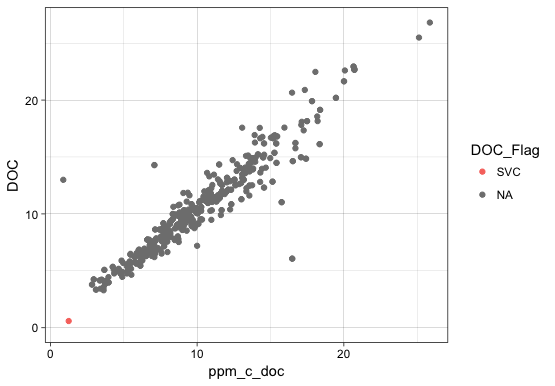


**4. Do we see the expected relationships between variables: pH and DOC?**

> ggplot(X2017\_10\_20\_142151\_HakaiData\_fgs, aes(x=pH, y=DOC)) + geom\_point(aes(color=DOC\_Flag)) + theme\_linedraw()



> ggplot(X2017\_10\_20\_142151\_HakaiData\_fgs, aes(x=ppm\_c\_doc, y=DOC)) + geom\_point(aes(color=DOC\_Flag)) + theme\_linedraw()



**Optional: combine multiple plots in one**

#1. Define the individual plots

> p1 <- ggplot(X2017\_10\_20\_142151\_HakaiData\_fgs, aes(x=Date, y=DOC)) + geom\_point(aes(shape=DOC\_Quality\_Level, color=DOC\_Flag)) + theme\_linedraw()

> p2 <- ggplot(X2017\_10\_20\_142151\_HakaiData\_fgs, aes(x=Date, y=DOC)) + theme\_linedraw() + geom\_point(aes(color=DOC\_Flag)) + facet\_grid(Survey~.)

> p3 <- ggplot(subset(X2017\_10\_20\_142151\_HakaiData\_fgs, Survey == "FGS"), aes(x=Date, y=DOC)) + geom\_point(aes(colour=Site\_ID)) + theme\_linedraw()

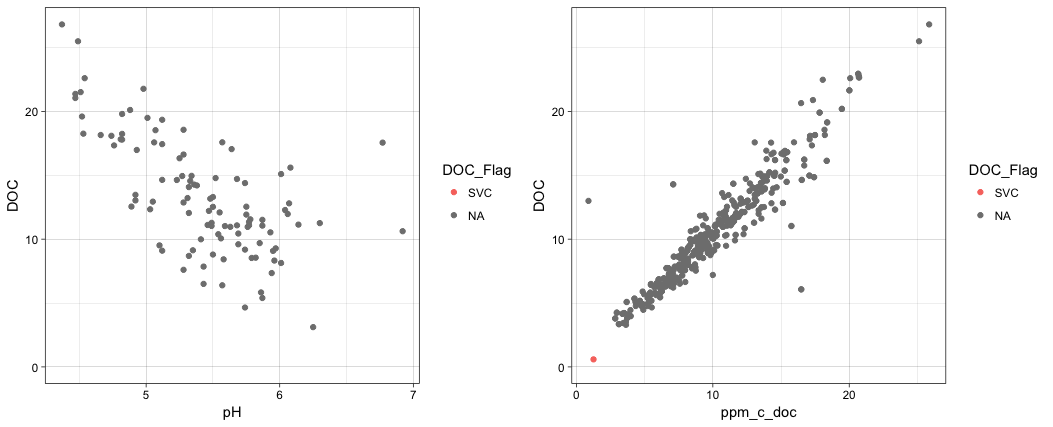
> p4 <- ggplot(X2017\_10\_20\_142151\_HakaiData\_fgs, aes(x=pH, y=DOC)) + geom\_point(aes(color=DOC\_Flag)) + theme\_linedraw()

p5 <- ggplot(X2017\_10\_20\_142151\_HakaiData\_fgs, aes(x=ppm\_c\_doc, y=DOC)) + geom\_point(aes(color=DOC\_Flag)) + theme\_linedraw()

#2. Create multiplot function. See Appendix 1.

#3. Graph any combination of those individual plots. Here is one example.

> multiplot(p4, p5, cols=2)

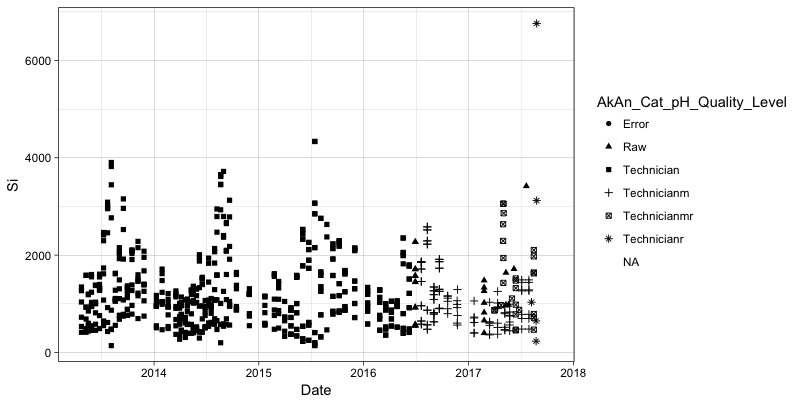


**Repeat for cations (here with Si and Ca examples)**

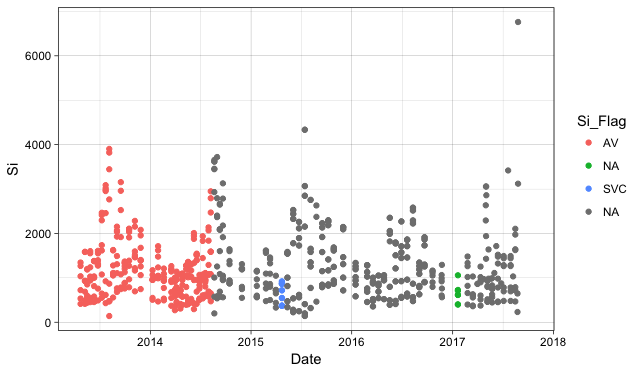
**1. Status of QC work:**

1. **What has and hasn’t been QC’d? (QC level)**
2. **What are the known problem values? (QC flags)**

> ggplot(X2017\_10\_20\_142151\_HakaiData\_fgs, aes(x=Date, y=Si)) + geom\_point(aes(shape=AkAn\_Cat\_pH\_Quality\_Level)) + theme\_linedraw()

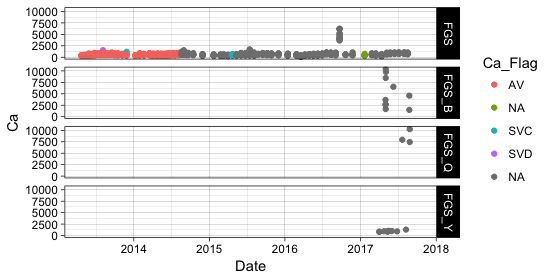


> ggplot(X2017\_10\_20\_142151\_HakaiData\_fgs, aes(x=Date, y=Si)) + geom\_point(aes(colour=Si\_Flag)) + theme\_linedraw()



**2. View by survey: does each survey have the expected range of values?**

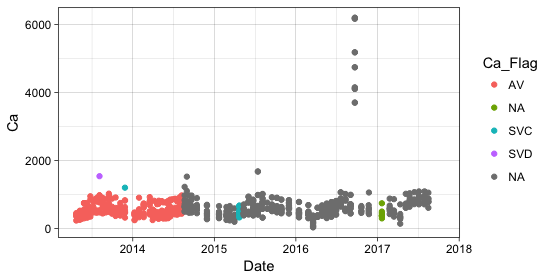
> ggplot(X2017\_10\_20\_142151\_HakaiData\_fgs, aes(x=Date, y=Ca)) + theme\_linedraw() + geom\_point(aes(color=Ca\_Flag)) + facet\_grid(Survey~.)



**View Kwakshua alone because y-axis quite different from other surveys**

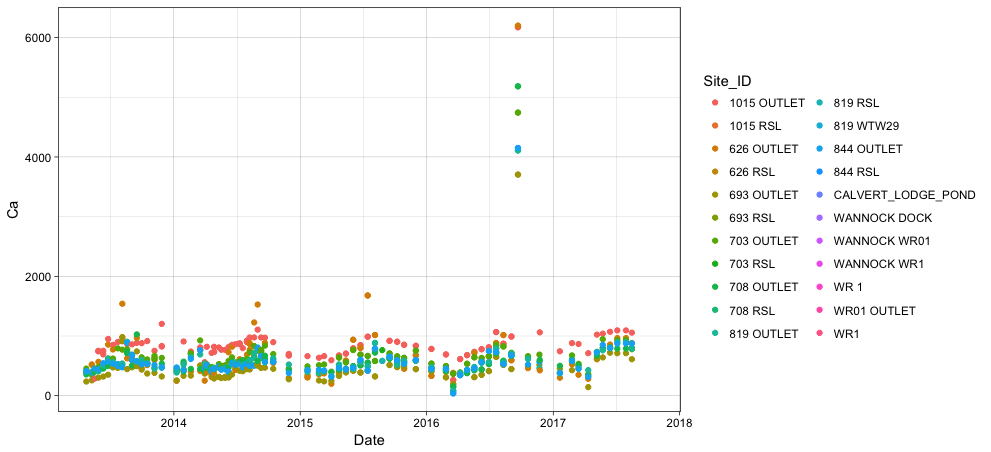
#Subset by Survey within graphing command

> ggplot(subset(X2017\_10\_20\_142151\_HakaiData\_fgs, Survey == "FGS"), aes(x=Date, y=Ca)) + geom\_point(aes(colour=Ca\_Flag)) + theme\_linedraw()



**3. River by river: does each river show the expected range of values?**

> ggplot(subset(X2017\_10\_20\_142151\_HakaiData\_fgs, Survey == "FGS"), aes(x=Date, y=Ca)) + geom\_point(aes(colour=Site\_ID)) + theme\_linedraw()



**APPENDIX 1. MULTIPLOT CODE**

From http://www.cookbook-r.com/Graphs/Multiple\_graphs\_on\_one\_page\_%28ggplot2%29/

multiplot <- function(..., plotlist=NULL, file, cols=1, layout=NULL) { library(grid) *# Make a list from the ... arguments and plotlist* plots <- c(list(...), plotlist) numPlots = length(plots) *# If layout is NULL, then use 'cols' to determine layout* if (is.null(layout)) { *# Make the panel* *# ncol: Number of columns of plots* *# nrow: Number of rows needed, calculated from # of cols* layout <- matrix(seq(1, cols \* ceiling(numPlots/cols)), ncol = cols, nrow = ceiling(numPlots/cols)) } if (numPlots==1) { print(plots[[1]]) } else { *# Set up the page* grid.newpage() pushViewport(viewport(layout = grid.layout(nrow(layout), ncol(layout)))) *# Make each plot, in the correct location* for (i in 1:numPlots) { *# Get the i,j matrix positions of the regions that contain this subplot* matchidx <- as.data.frame(which(layout == i, arr.ind = TRUE)) print(plots[[i]], vp = viewport(layout.pos.row = matchidx$row, layout.pos.col = matchidx$col)) } } }