Piper Plot and Stiff Diagram Examples

Dave Lorenz

October 13, 2015

Abstract

This example demonstrates how to prepare data for a Piper plot and create a Piper plot from those data. It also demonstrates the ternary plot, also called trilinear or triangular diagram. The Piper diagram replicates figure 37 in Hem (1989). The trilinear example uses a randomly generated set of data. **NOTE:** to use the piperPlot function, you must first call a function to set up the graphics environment like `setPage` or `setPDF`, but these are not included here to use the graphics tools in `Sweave`.

Contents

1 Introduction

Some of the examples use data from the smwrData package. Other examples use randomly generated data. The data are retrieved or generated in the following code.

```
> # Load the smwrGraphs package
> library(smwrGraphs)
> # Generate a random sample for the ternary diagram
> set.seed(2727)
> # Ternary diagram data
> X <- runif(25, .1, 1.)
> Y <- runif(25, .1, .8)
> Z <- runif(25, .3, 1.)
> # Get the selected groundwqater quality date from Hem
> library(smwrData)
> data(MiscGW)
```

2 Piper Plot

The Piper plot assumes that the data are in similar units. The traditional units would be milli-equivalents per liter. Each column in the data set must be converted from milligrams per liter to milli-equivalents per liter. This can be accomplished by the `conc2meq` function in the smwrBase package, loaded by default when the smwrGraphs package is loaded. The data provided to the `piperPlot` function do not need to sum to 1 or 100.

```
> # Transform the data. This example will ignore potassium, fluoride, and nitrate
> # (carbonate is either 0 or missing and will also be ignored).
> PD <- transform(MiscGW, Ca.meq = conc2meq(Calcium, "calcium"),
+                     Mg.meq = conc2meq(Magnesium, "magnesium"),
+                     Na.meq = conc2meq(Sodium, "sodium"),
+                     Cl.meq = conc2meq(Chloride, "chloride"),
+                     SO4.meq = conc2meq(Sulfate, "sulfate"),
+                     HCO3.meq = conc2meq(Bicarbonate, "bicarb"))
> # abbreviations allowed in the call to conc2meq
> # The row name identifies the sample source, create a column
> PD$SS <- row.names(PD)
> # setSweave is a specialized function that sets up the graphics page for
> # Sweave scripts. It should be replaced by a call to setPage or setPDF
> # in a regular script.
> # The minimum page size for a Piper plot is 7 inches. No check is made,
> #  but the axis title spacings require a graph area of at least 6 inches.
> setSweave("piperplot01", 7, 7)
> # For this example, a separate graph area for an explanation is not needed
> #  because there are only 4 groups (individuals).
> AA.pl <- with(PD, piperPlot(Ca.meq, Mg.meq, Na.meq,
+     Cl.meq, HCO3.meq, SO4.meq,
+   Plot=list(name=SS, color=setColor(SS)),
+   zCat.title = "Sodium",
+   xAn.title = "Chloride",
+   yAn.title = "Bicarbonate"))
> addExplanation(AA.pl, where="ul", title="")
> # Required call to close PDF output graphics
> graphics.off()
```

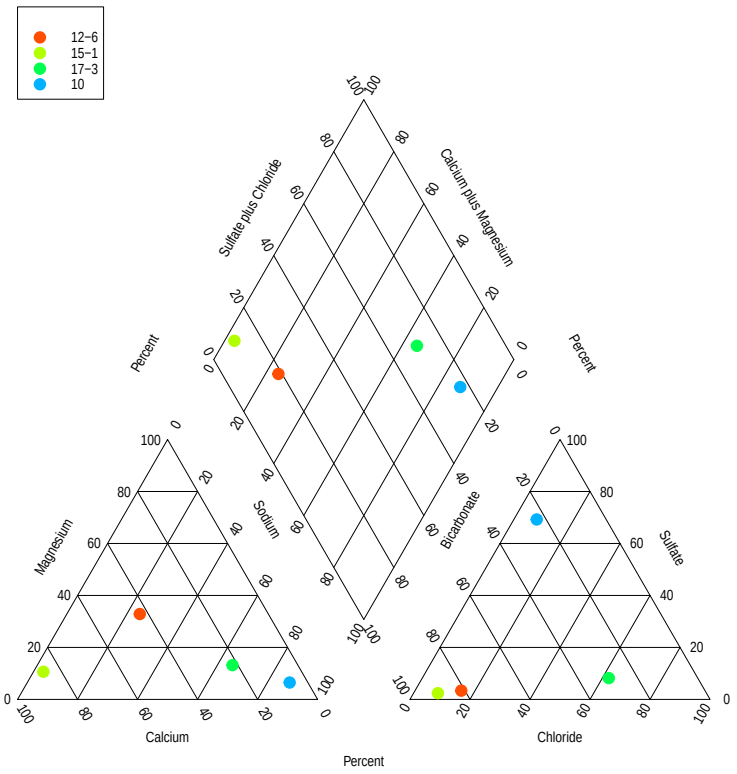


Figure 1. The Piper diagram.

3 Custom Piper Plot

This example demonstrates how to use the information that is contained in the output from `piperPlot` to customize the Piper plot. The output from `piperPlot` contains 3 data frames, cations, anions, and piper. The first 3 column names for cations and anions are taken from the axis labels, forced to be valid names in R. The values in those first 3 columns are the normalized (sum to 1 or 100) data. The last 2 columns in those data sets are named x and y and are the data projected onto the respective triangular simplex. The columns in the piper data frame are called x and y an are the data projected onto the middle (Piper) plot. The data in the columns named x and y can be supplied to the corresponding arguments in `addPiper`; cation$x corresponds to the `xCat` argument and piper$x corresponds to `xPip` for example. When that scheme is used for adding to the Piper plot, the arguments `zCat` and `zAn` must not be used.

The example below approximately recreates Figure 37 in Hem (1989). The size of the symbol in the middle (Piper) plot are related to the Ionic strength in this case, instead of the total dissolved solids in Hem (1989).

```
> # Use the data from the previous example
> setSweave("piperplot02", 7, 7)
> # Create the empty Piper plot
> AA.pl <- with(PD, piperPlot(Ca.meq, Mg.meq, Na.meq,
+     Cl.meq, HCO3.meq, SO4.meq,
+   Plot=list(what="none"),
+   ticks = TRUE,
+   zCat.title = "Sodium",
+   xAn.title = "Chloride",
+   yAn.title = "Bicarbonate"))
> # Fill in the symbols in the triangular graphs, do not overwrite AA.pl
> with(AA.pl, addPiper(xCat=cations$x, yCat=cations$y, xAn=anions$x, yAn=anions$y,
+   xPip=NA, yPip=NA, # Missing values are not plotted
+   Plot=list(size=.05), current=AA.pl))
> # Compute a measure of the ionic strength
> PD <- transform(PD, TotalCat=Ca.meq + Mg.meq + Na.meq)
> # Compute the symbol size (mean diameter is .2 inch)
> PD.size <- 0.2*sqrt(PD$TotalCat)/mean(sqrt(PD$TotalCat))
> # Now add the scaled circles to the middle plot
> with(AA.pl, addPiper(xCat=NA, yCat=NA, xAn=NA, yAn=NA,
+   xPip=piper$x, yPip=piper$y,
+   Plot=list(size=PD.size, filled=FALSE), current=AA.pl))
> # Required call to close PDF output graphics
> graphics.off()
```

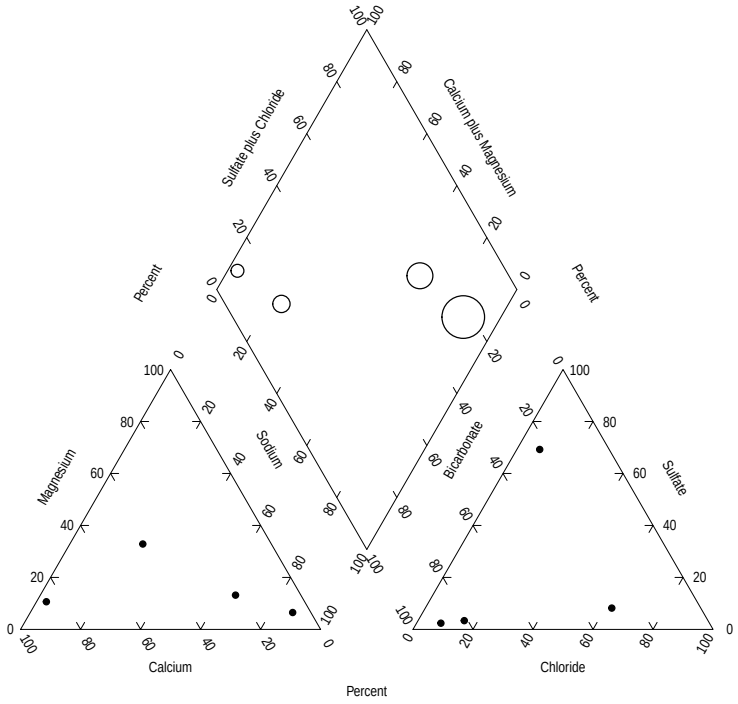


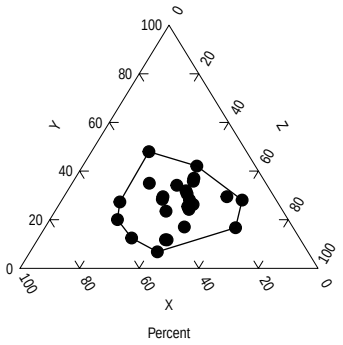
Figure 2. A customized Piper diagram.

4 Ternary Diagram

The ternary diagram also assumes that the data are in similar units. The traditional use would be milli-equivalents per liter for water-chemistry data, or sand-slit-clay for soil data, but other units are possible. The data provided to the `ternaryPlot` function do not need to sum to 1 or 100.

This example simply plots arbitrary x, y, and z data and provides an example of the use of `addTernary`. In contrast to `addPiper`, `addTernary` requires values for all of the `x`, `y`, and `z` arguments. The data returned in the output from `ternaryPlot` are components called x and y, which are the transformed data for plotting in the triangular plot.

```
> setSweave("piperplot03", 3.5, 3.5)
> # Accept all defaults
> AA.pl <- ternaryPlot(X, Y, Z)
> # Use the chull function to extract the points that define the
> # convex hull of the data.
> AA.pts <- chull(AA.pl$x, AA.pl$y)
> # Close it
> AA.pts[length(AA.pts) + 1] <- AA.pts[1]
> # Select those points and draw the hull
> addTernary(X[AA.pts], Y[AA.pts], Z[AA.pts],
+   Plot=list(what="lines"), current=AA.pl)
> # Required call to close PDF output graphics
> graphics.off()
```



..

Figure 3. A very simple ternary diagram.

5 Stiff Diagram

The Stiff diagram also assumes that the data are in similar units. The traditional use would be milli-equivalents per liter for water-chemistry data, but other units are possible.

```
> setSweave("piperplot04", 6, 6)
> AA.lo <- setLayout(height=3.5, explanation=list(bottom=1.1))
> setGraph(1, AA.lo)
> # Accept all defaults, but subset the data for the small graph size
> AA.pl <- with(PD, stiffPlot(cbind(Ca.meq, Mg.meq, Na.meq),
+           cbind(Cl.meq, SO4.meq, HCO3.meq), ylabels=SS))
> setGraph("explanation", AA.lo)
> addExplanation(AA.pl)
> # Required call to close PDF output graphics
> graphics.off()
```

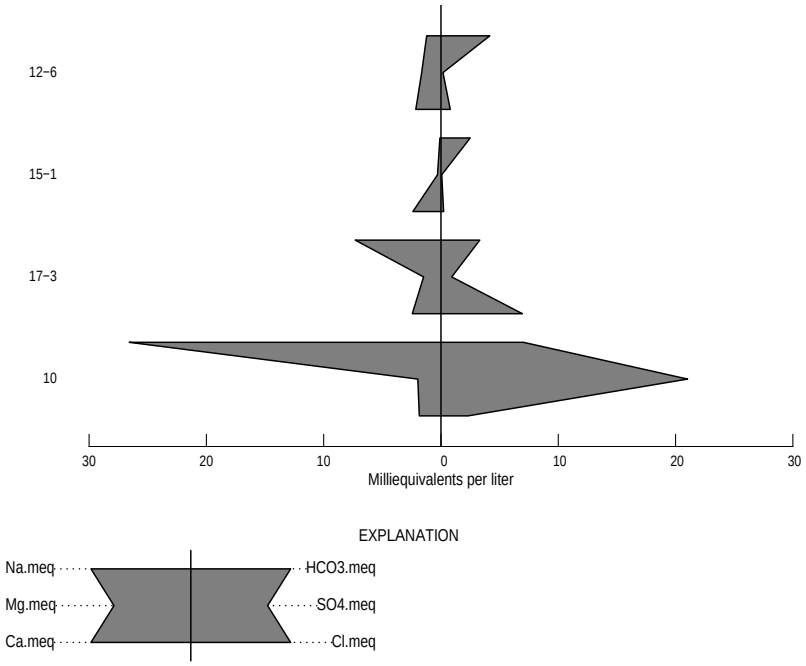


Figure 4. The Stiff diagram.

References

[1] Hem J.D., 1989, Study and interpretation of the chemical characteristics of natural water: U.S. Geological Survey Water-Supply Paper 2254, 263 p.