Line/Scatter Examples

Dave Lorenz

October 24, 2024

Abstract

These examples demonstrate the basic high-level line/scatter plot functions in the smwrGraphs package. Most of the examples use randomly generated sets of data. **NOTE:** to use any of these functions, 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 random samples for the examples.
> set.seed(2576)
> X <- runif(33)
> Y <- runif(33)
> Z <- rep(c("A", "B", "C"), 11)
> X12 <- X[1:12]
> Z12 <- LETTERS[1:12]
> # Load the smwrData package
> library(smwrData)
> data(IonBalance)
> data(KlamathTP)
```

2 Scatter Plot

The `xyPlot` function plots paired x- and y-coordinate data. As of version 1.1.0, there are methods for factor and numeric x-coordinate data and numeric y-coordinate data. This example plots only numeric data.

```
> # 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.
> setSweave("lsplot01", 6 ,6)
> xyPlot(X, Y, Plot=list(color="darkblue"))
> # Required call to close PDF output graphics
> graphics.off()
```

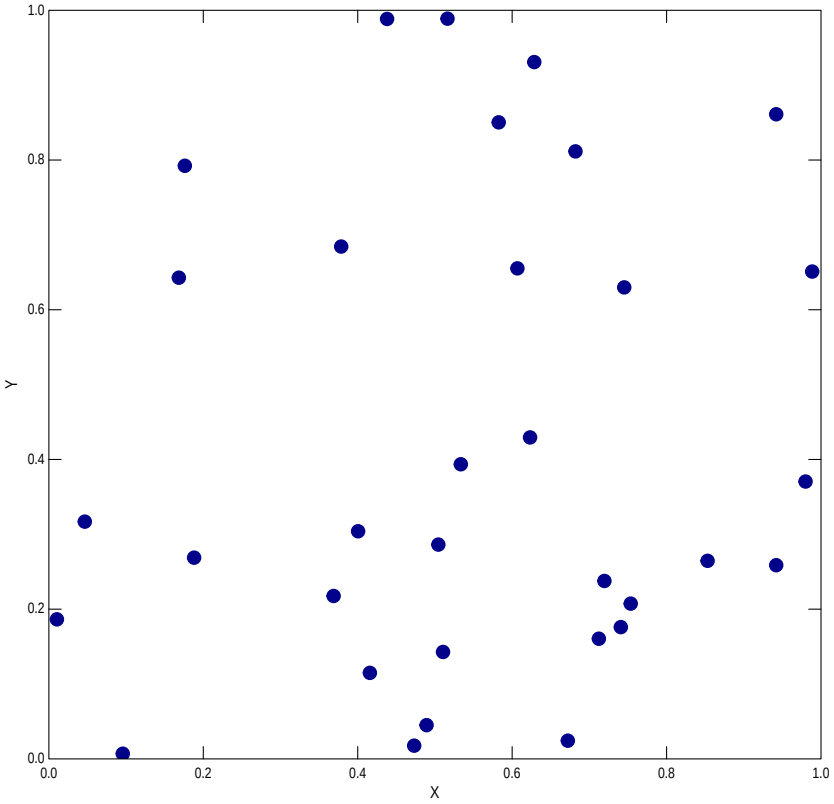


Figure 1. A simple scatter plot

3 Date/Time Plot

The `timePlot` function plots paired x- and y-coordinate data. As of version 0.7, there are methods for Date, POSIXt, numeric, and integer x-coordinate data and numeric y-coordinate data. This example plots total phosphorus concentrations over time.

```
> # 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.
> setSweave("lsplot02", 6 ,6)
> with(KlamathTP, timePlot(sample_dt, TP_ss, Plot=list(what="points", color="darkblue"),
+   yaxis.range=c(0,1.5)))
> # Required call to close PDF output graphics
> graphics.off()
```

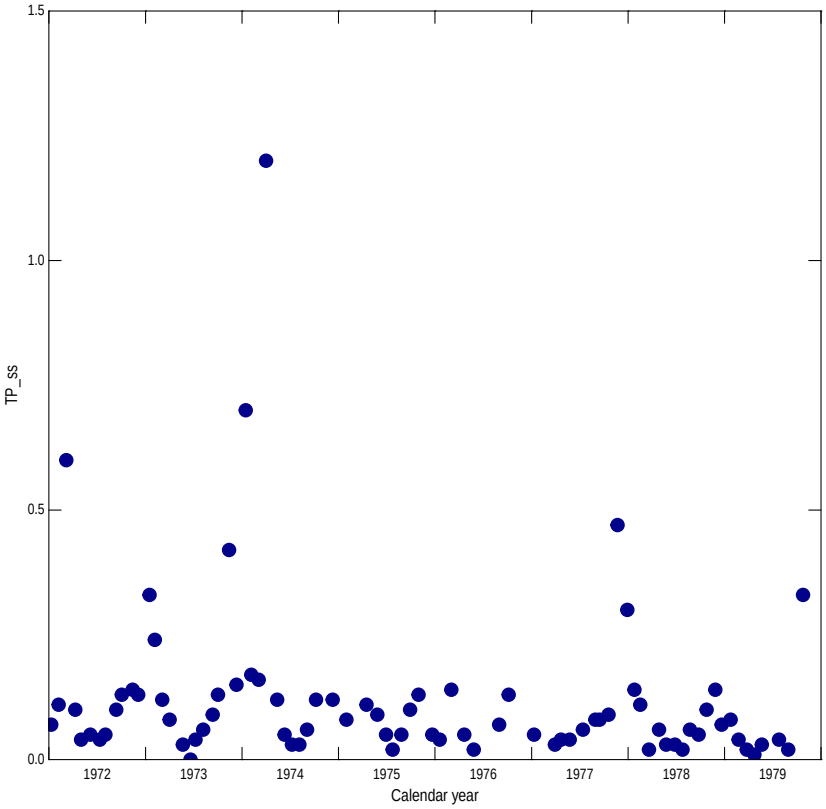


Figure 2. A time-series plot

4 Season Plot

The `seasonPlot` function plots paired x- and y-coordinate data along an x axis that spans one year. The x-coordinate data must be in a form that can be converted to decimal time—representing the date as the year and fractional part of the year. The function `dectime` in `smwrBase` describes that format. This example shows the variation of total phosphorus at a site.

```
> # 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.
> setSweave("lsplot03", 6 ,6)
> with(KlamathTP, seasonPlot(sample_dt, TP_ss, yaxis.range=c(0,1.5)))
> # Required call to close PDF output graphics
> graphics.off()
```

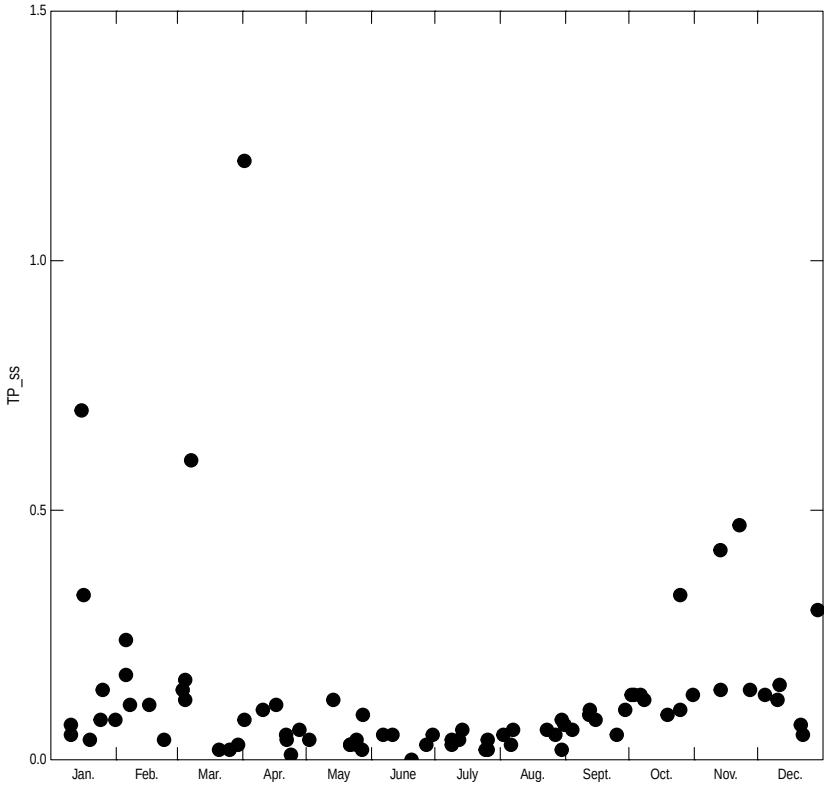


Figure 3. A seasonal plot

5 Series Plot

The `seriesPlot` function plots y-coordinate data arranged within a season associated with each observation. The graph shows the sequence of observations within each season over time. The x-axis data is generated from the sequence, so the data to be plotted must be a regular series. The function `regularSeries` in `smwrBase` can be used to generate the regular series from data that have irregular sampling. The example plots the total phosphorus data as a regular series. Note that the default method for `seriesPlot` assumes no missing values and if there are any missing values within a season, that season is not plotted.

```
> # 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.
> setSweave("lsplot05", 6 ,4)
> # Create the regular series of observations
> AA.rs <- with(KlamathTP, regularSeries(TP_ss, sample_dt, begin="1972-01-01",
+   end="1980-01-10"))
> seriesPlot(AA.rs$Value, yaxis.range=c(0,1.5), xlabels=month.USGS)
> # Required call to close PDF output graphics
> graphics.off()
```

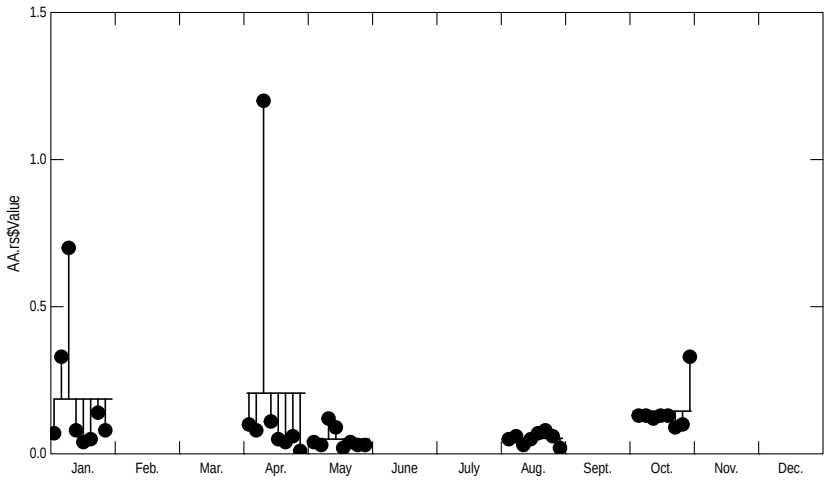


Figure 5. A series plot

6 Color Plot

The `colorPlot` function plots paired x- and y-coordinate data, with symbol color controlled by another variable. As of version 0.7, there is only one method, for numeric x and y data. The color variable can be type character or factor, numeric, or integer. If the type is character or factor, then the values can be color names (see the documentation for colors), or group names that are converted to bright colors. If the type is numeric, then the colors are created using the color, groups, and ramp components of the Plot argument. If the type is integer, the the color component of Plot should be "Index" to set the color to the index color of the palette (see the documentation for palette).

```
> # 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.
> setSweave("lsplot06", 6 ,4)
> # Accept the default colors for groups.
> AA.pl <- colorPlot(X, Y, color=Z)
> addExplanation(AA.pl, where="ul", title="", box.off=FALSE)
> # Required call to close PDF output graphics
> graphics.off()
```

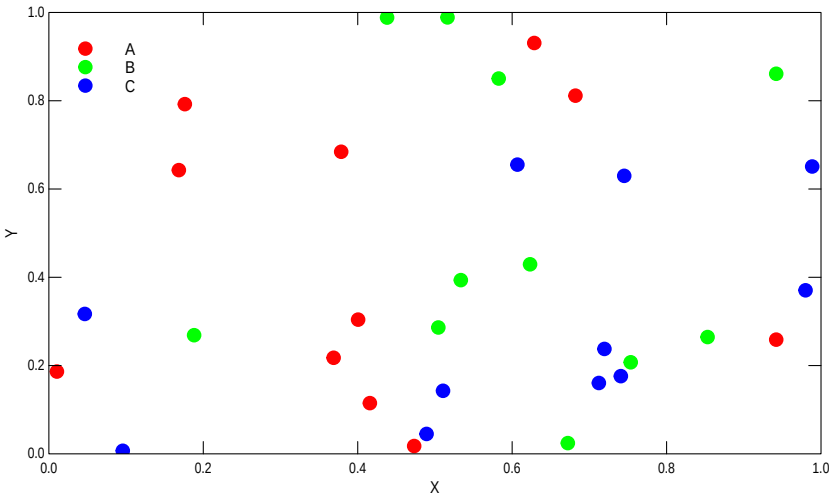
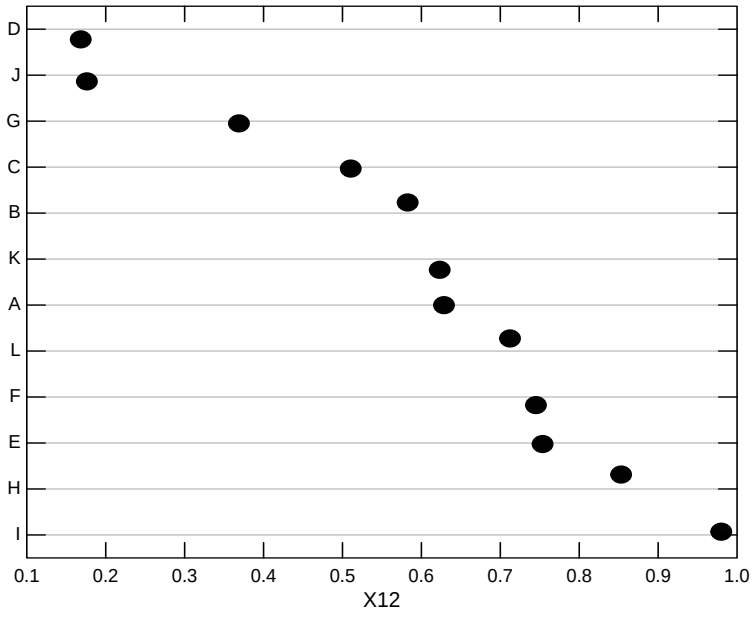


Figure 6. A color plot

7 Dot Plot

The `dotPlot` function plots continuous x-coordinate data for discrete y-coordinate data. Often, dot plots are used to display single values for each discrete value and often sorted as done in this example.

```
> # 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.
> setSweave("lsplot07", 4, 4)
> # Create the y-axis data as a factor with levels sorted by x
> Z12f <- factor(Z12, levels=Z12[order(X12)])
> # Plot the results
> dotPlot(X12, Z12f)
> # Required call to close PDF output graphics
> graphics.off()
```



.

Figure 7. A dot plot

8 Scaled Plot

The `scalePlot` function plots x- and y-coordinate data on the same scale. This can be used to assess the relation between x and y in the context of their true magnitudes or for geographic data. This example plots some data in the IonBalance dataset to asses the relative ionic strengths.

```
> # 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.
> setSweave("lsplot08", 6 ,6)
> # Plot Calcium and Magnesium
> with(IonBalance, scalePlot(Ca, Mg, Plot=list(what="points")))
> # Required call to close PDF output graphics
> graphics.off()
```

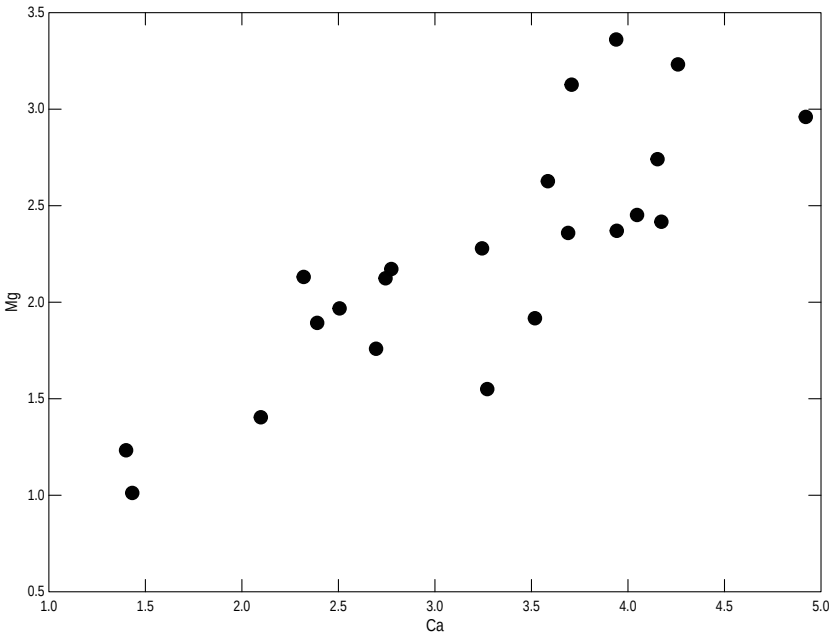


Figure 8. A scaled plot

9 Scatter Plot Matrix

A scatter plot matrix shows multiple pairs of x and y data in separate graphs. This example plots some data in the IonBalance dataset to assess some of the relations among selected cations. The function `setSplom` must be used to set up the graphics device. For many pairs of data, a very large graphics device can be created. The `Panel` argument is a very flexible way to add statistics or selected lines to each plot.

```
> # 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.
> setSweave("lsplot09", 6 ,8)
> # Plot Calcium and Magnesium and Sodium, and add explanation at bottom
> AA.lo <- with(IonBalance, setSplom(num.variables=3, touching=FALSE,
+   explanation=list(bottom=2)))
> # Do not call setGraph with splomPlot
> AA.pl <- with(IonBalance, splomPlot(cbind(Ca, Mg, Na), Plot=list(name="Ionic Strength"),
+   Panel=list(line="slr"), layout=AA.lo))
> # setGraph needed for explanation
> setGraph("explanation", AA.lo)
> addExplanation(AA.pl)
> # Required call to close PDF output graphics
> graphics.off()
```

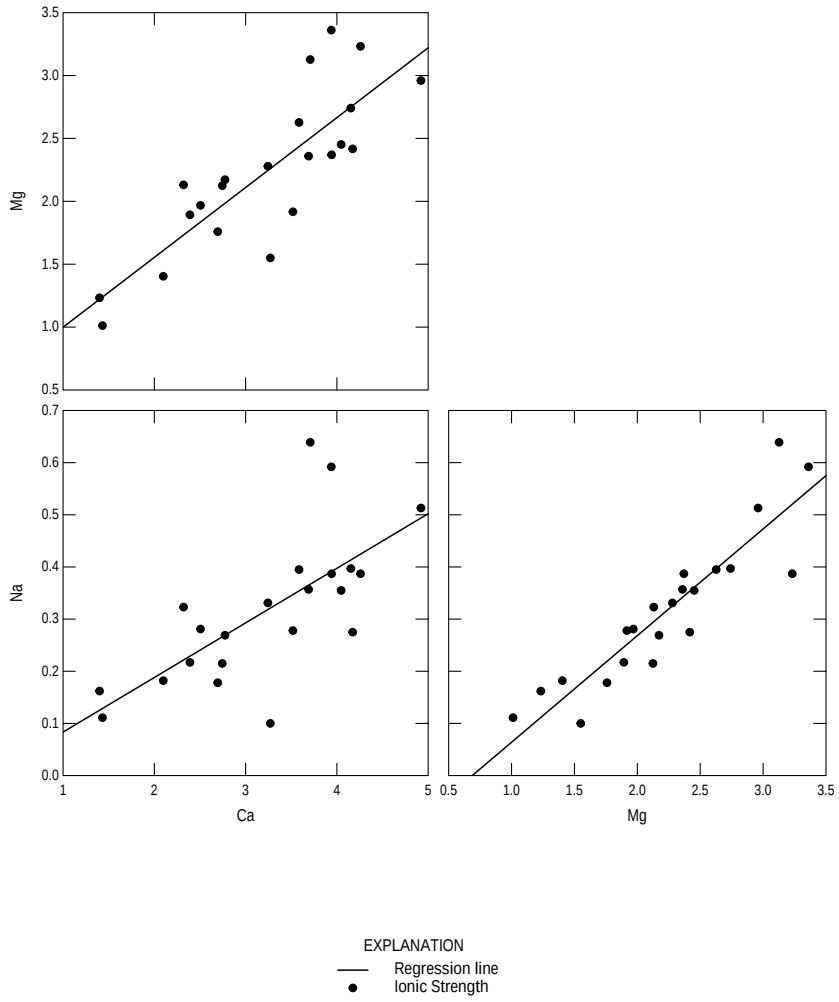


Figure 9. A scatter plot matrix with an explanation.