6c. Line plots with error bars

6c.1 Line plot basics

Line plots can be generated in exactly the same way as scatter plots. They share the "plot" command and customizations (axes, box, labels, text, arrows, gridlines, colors, symbols, and legends) are applied in the same way. The difference between the two plot types is conceptual: If your x-axis is an ***ordered sequence*** or a ***time variable*** rather than a ***predictor variable***, then you should connect your data points with lines. Otherwise, never connect your data points with lines (i.e. in a scatter plot). With line-plots (unlike scatter plots), you may want to start your y-axis with a 0 value (e.g. to convey a sense of the scale of changes over time). However, this is not mandatory

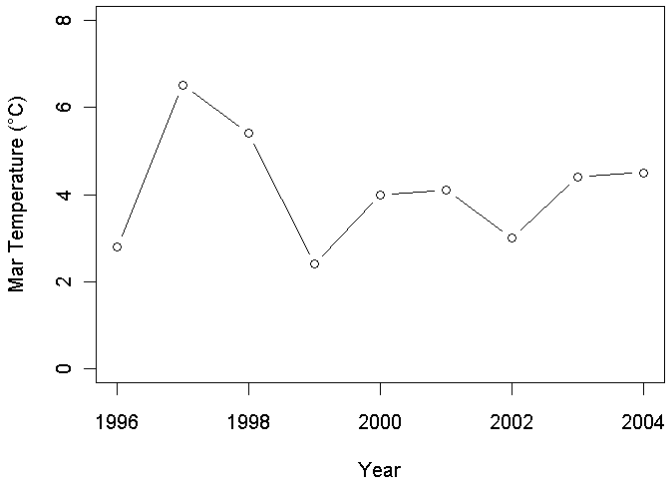
- Download or enter the following dataset in Excel and save as CSV. The dataset consists of March, April, and May temperature (independent variables) and the Julian date of bud break in aspen (dependent variable) over several years.

YEAR	T_MAR	T_APR	T_MAY	BUD	BUD_SE
1996	2.8	5.7	7	87	3.6
1997	6.5	7.8	8.8	98	2
1998	5.4	7.7	10	93	2.4
1999	2.4	4.6	6	85	4
2000	4	4.7	5.5	89	2.4
2001	4.1	6.2	7.6	91	2.4
2002	3	5.7	8.5	87	4
2003	4.4	5.9	7.3	92	2.4
2004	4.5	7	10.2	92	3

- Set yourself up as usual: (1) open R, (2) save an empty workspace in your working directory (e.g. C:/Lab6/), (3) close R and re-open R by double clicking the saved workspace, (4) in R, create and save a new script file (e.g. Lab6.r).

- As before, import the CSV file with the data and use three-liner that (1) re-sets your graphics parameters, (2) sets your window size to 6" wide and 5" high, and (3) sets the global graphics parameter to your preferences (size, font-size, font-type). Run and re-run each graphic starting with `graphics.off` command.

- Now, create a scatter plot of T_APR over YEAR with xlab and ylab to at the labels "Year" and "April Temp (°C)". The ASCII code to get "°" is "248". Once you have that scatter plot, change it to a line plot by including this option in the plot statement: `type="l"` Also try: `type="b"` (both: points and lines), `type="o"` (over-plotted: lines over points). Over-plotted looks fine if you use filled symbols, e.g. `pch=19`.



6c.2 Line plot customizations

- Let's customize this graph and bring in all temperature variables: Set the y-axis tick marks to 2-year intervals 1996-2004 (if R didn't already default to that), and the y-axis to 3° intervals (0 to 12°C). You need to use `xlim` and `ylim`, clear the plot with `ann`, `axes` and `type="n".` Also, change the y-label to "Temperature (°C)".

- Then, bring back the customized axes with the axis commands. Switch y-axis labels to horizontal with `las=2` and set your tick-marks with `axis` commands. If you execute this, you should see an empty plot. This should still yield just an empty canvas

- Now we need some new code to bring back the data rows. We did it before with the `points(…)` command (Reference: Exercise 3.5), now we use the equivalent `lines(…)` command. Since our data series are in separate columns we simply use multiple lines statements with customizations. Try those two options below or make up your own customizations:

```
lines(T_MAR~YEAR, lty=1)
lines(T_APR~YEAR, lty=2)
lines(T_MAY~YEAR, lty=3)
```

or:

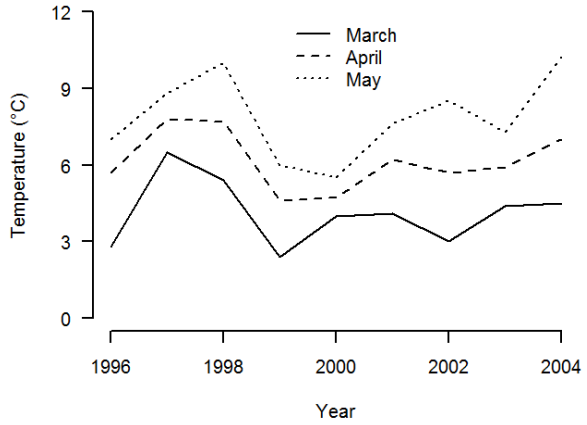
```
lines(T_MAR~YEAR, type="o", pch=19, col="green")
lines(T_APR~YEAR, type="o", pch=19, col="red")
lines(T_MAY~YEAR, type="o", pch=19, col="blue")
```

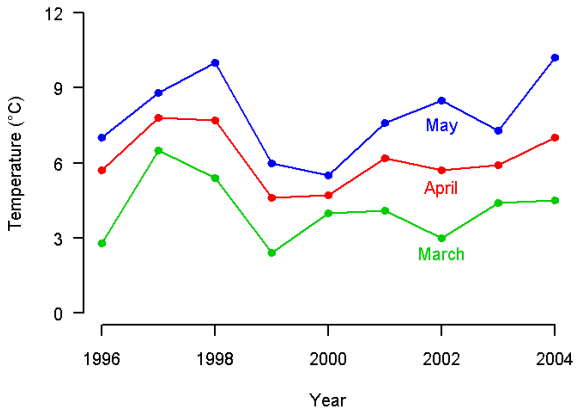
- You can add a legend as we've learned before, or you can directly label the lines with text commands. Try to get the plots below on your own based on this code for legends or labels:

```
legend(1999,12, bty="n",
   lty    = c(1,1,1),
   lwd    = c(2,2,2),
   pch    = c(19,19,19),
   col    = c("green","red","blue"),
   legend = c("March","April","May"))
```

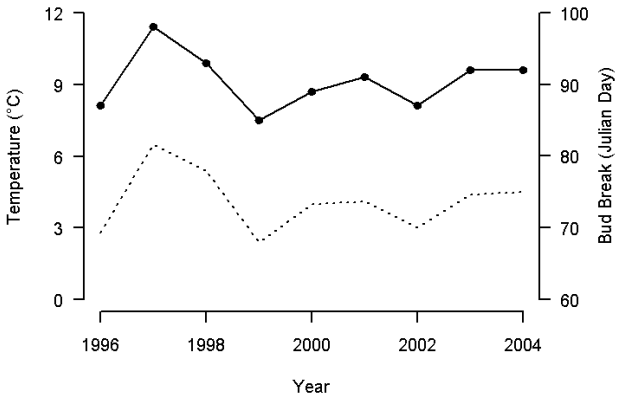
or:

```
text(2002,2.4, "March", col="green")
text(2002,5.1, "April", col="red")
text(2002,7.6, "May",   col="blue")
```





6c.3 Two y-axes with different scales



A good use of line plots is to show how a dependent tracks an independent variable over time. In contrast to a scatter plot (where I only see a relationship of un-ordered points), the line plot allows you to see an additional trend over time, or allows to see and explain a breakdown of the relationship for particular times.

- Since our dependent and dependent variables, are rarely measured on the same scale, we usually have to introduce another y-axis on the right. In R, we do this by overlaying a second line-plot on top of the first one. The command `par(new=T)` indicates that we want to do an overlay. So, run your plot from the previous section followed by this:

```
par(new=T)
plot(BUD~YEAR, type="l", ann=F, axes=F)
axis(4)
```

- You may sometimes run out of space on the right side. Then, modify your global graphics parameters (window and right margin half an inch wider):

```
dev.off()
windows(width=6.5, height=5)
par (cex=1, family="sans", mar=c(5,5,5,5.5))
```

- Now we have space to add label for the second y-axis with a margin text (`mtext`) command, and also customize the line graph a bit. You can use `adj` and `padj` with positive and negative numbers to adjust the position of the mtext label vertically and horizontally:

```
par(new=T)
plot(BUD~YEAR, type="o", ann=F, axes=F, pch=19, ylim=c(60,100))
axis(4, las=2)
mtext("Bud Break (Julian Day)", side=4, padj=4)
```

- Now the graph is a little busy and you probably notice that that the day of bud break tracks March temperature best. So, let's clean up the code and throw April and May temperatures out of our plot. Your simplified, cleaned-up code may look like this:

```
dev.off()
windows(width=6.5, height=5)
par (cex=1, family="sans", mar=c(5,5,5,5.5))

plot(T_MAR~YEAR, type="n", pch=19, ann=F, axes=F,
     xlim=c(1996,2004), ylim=c(0,12))
title(ylab="Temperature (°C)",xlab="Year")

axis(1, at=seq(1996, 2004, 2))
axis(2, at=c(0,3,6,9,12), las=2)
lines(T_MAR~YEAR, type="l", lty=2)

par(new=T)
plot(BUD~YEAR, type="o", ann=F, axes=F, pch=19, ylim=c(60,100))
axis(4, las=2)
mtext("Bud Break (Julian Day)", side=4, padj=4)
```

6c.4 Adding error bars to line (or scatter) plots

In line plots, arrow bars are commonly used to indicate the spread of your data (standard deviation) or spread of the estimates of a mean (standard error). In this example, the date of bud break was measured on multiple trees, and we have an estimate of a mean (BUD) and a standard error of the mean (BUD_SE).

- We cleverly use the arrow function to add error bars with the angle of the arrow set to 90°. Recall that the first four positions after **`arrows(`** indicate the start x-coordinate, start y-coordinate, end x-coordinate, end y-coordinate. (This is not a hack, it's how it's officially done in R :-)

```
arrows(YEAR,BUD,  YEAR,BUD+BUD_SE,  length=0.05, angle=90)
arrows(YEAR,BUD,  YEAR,BUD-BUD_SE,  length=0.05, angle=90)
```

- If you just want just lines, use this:

```
arrows(YEAR,BUD+BUD_SE,  YEAR,BUD-BUD_SE,  length=0)
```

- As an exercise, see if you can re-create these graphs.

- For the multi-panel graph, you have to control the upper and lower margins for the first and second graph individually. Try `par(mar=c(0,0,0,0))` before the first plot, and `par(mar=c(3,0,0,0))` before the second plot. This way you can stack up to three line plots with individually adjusted heights.

- You also need to apply points and line commands separately to overlay white dots on a black line.