Plotting Practice

2023-06-26

Let's go ahead and load a couple of libraries that may be useful.

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.3     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```
library(skimr)
```

Cool, now that we have a handful of packages that may come in handy, let's get started.

Book Examples

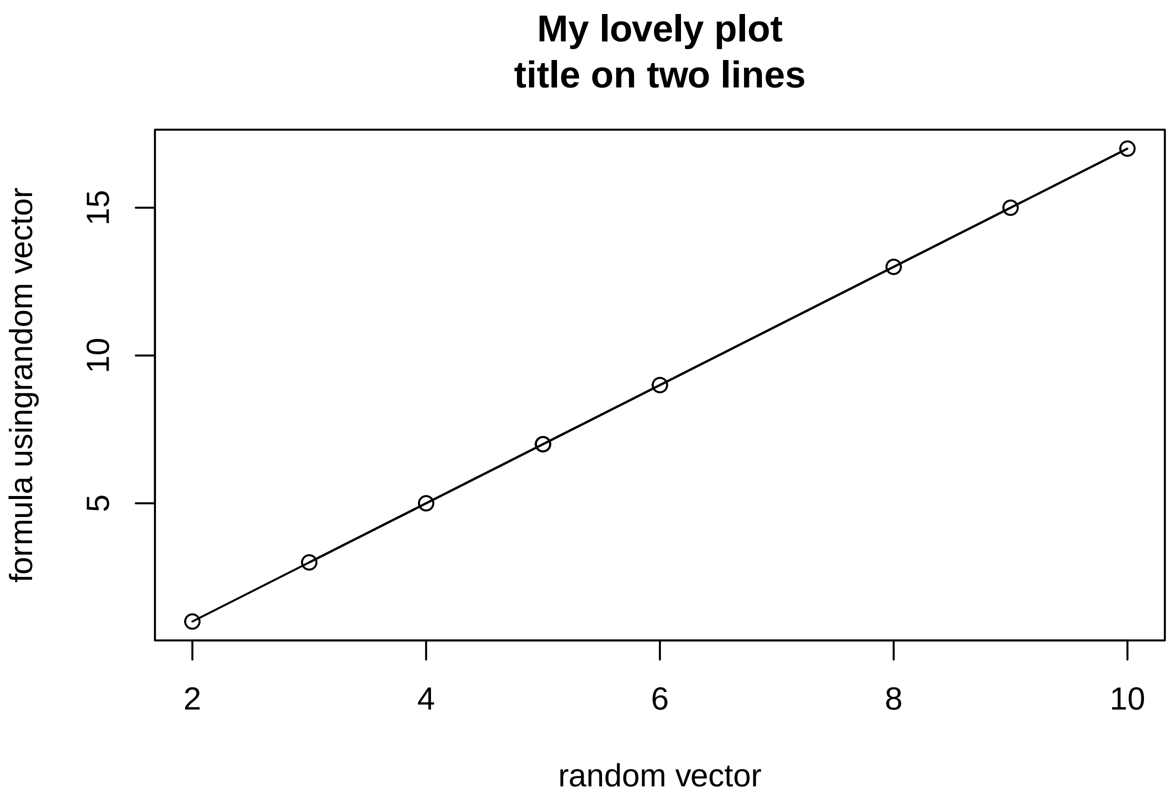
I'm just copying the code from The Book of R examples to see what the different colors and aesthetics look like for the various graphical parameters. First let's get some data to graph!

```
a <- c(2, 4, 5, 8, 10, 5, 3, 6, 9)
b <- a*2-3
c <- cbind(a,b)
c
```

```
##        a  b
##  [1,]  2  1
##  [2,]  4  5
##  [3,]  5  7
##  [4,]  8 13
##  [5,] 10 17
##  [6,]  5  7
##  [7,]  3  3
##  [8,]  6  9
##  [9,]  9 15
```

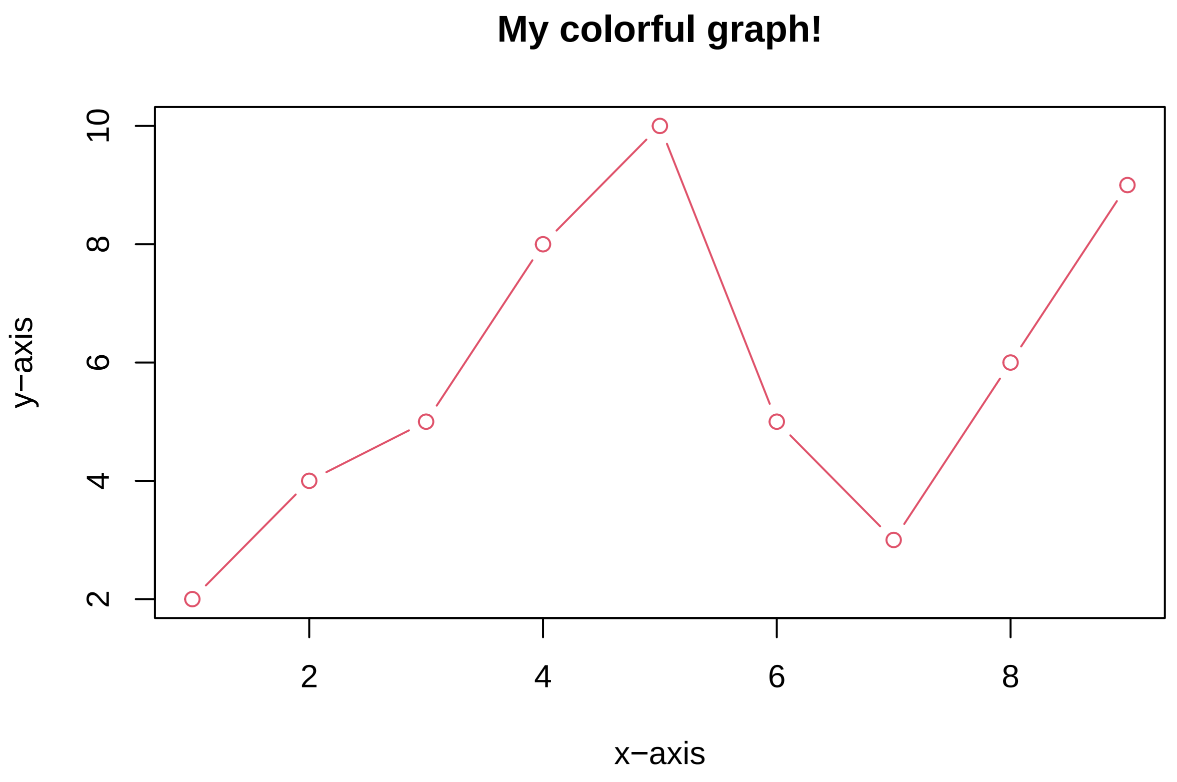
Cool, now let's plot it.

```r
plot(c, type="o", main= "My lovely plot\ntitle on two lines", xlab= "random vector", ylab= "formula usi
```

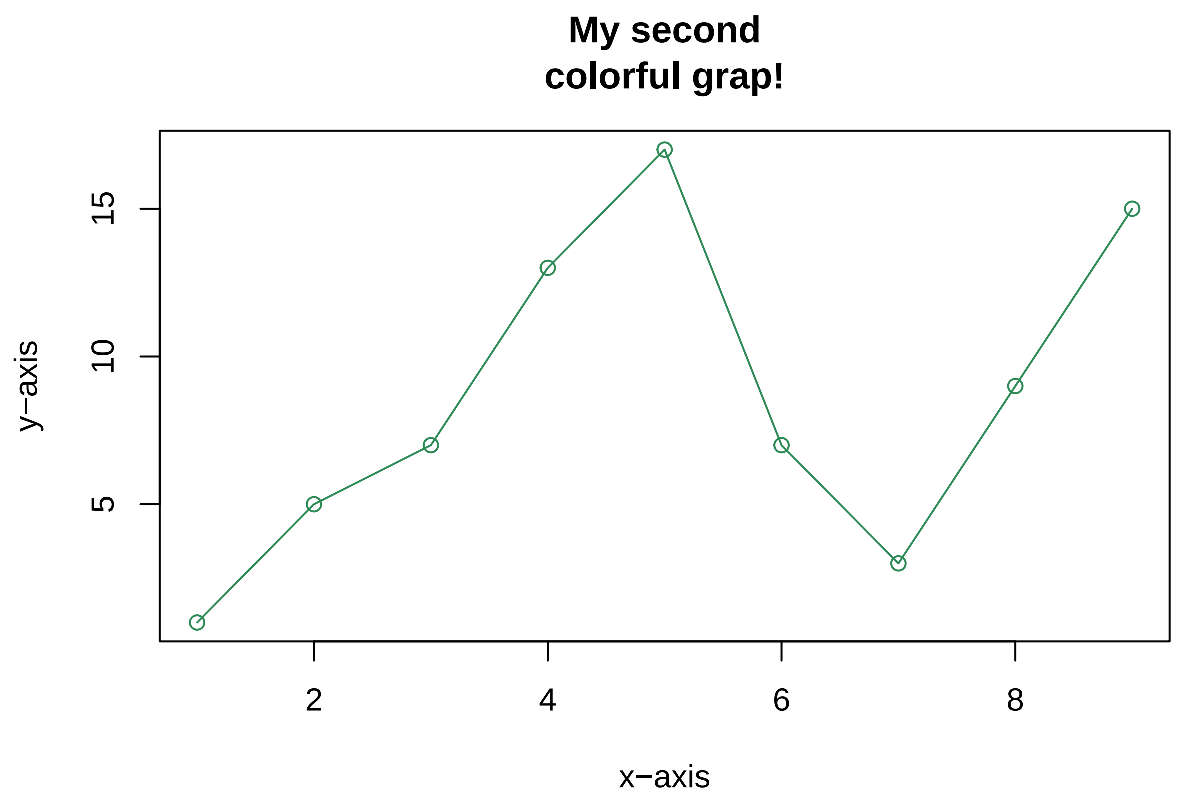


Cool, now let’s play around with colors.

```r
plot(a, type="b", main= "My colorful graph!", xlab= "x-axis", ylab= "y-axis", col= 2)
```

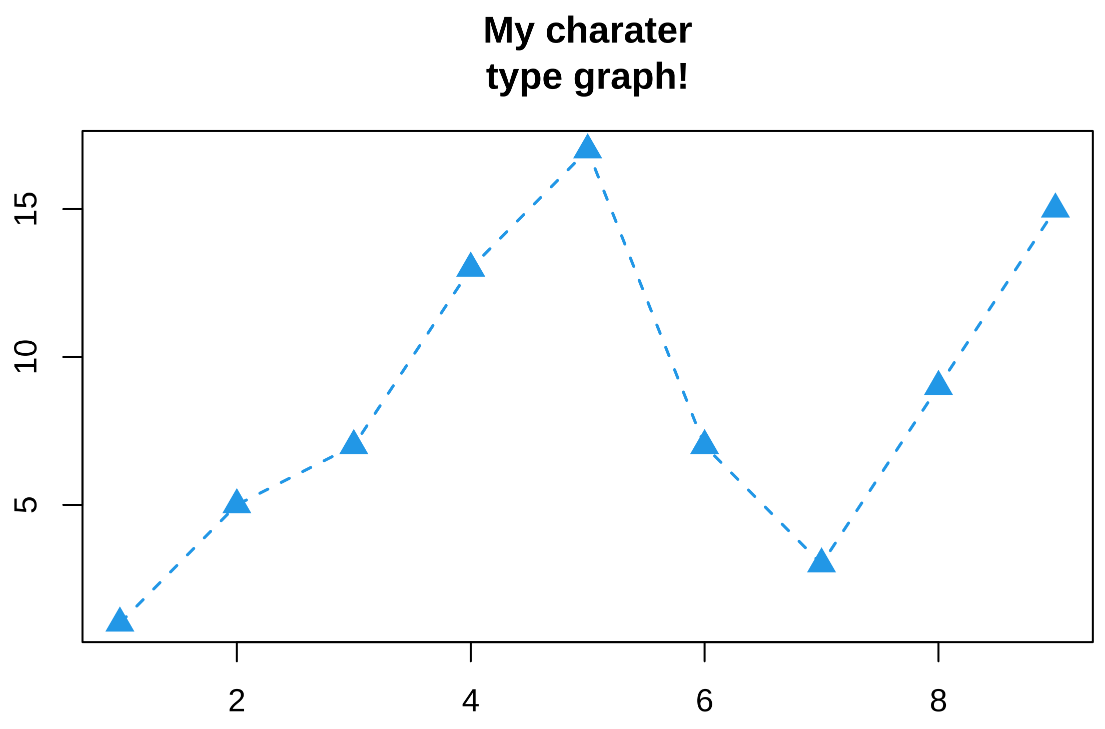


```
plot(b, type="o", main= "My second\ncolorful grap!", xlab= "x-axis", ylab= "y-axis", col= "seagreen4")
```

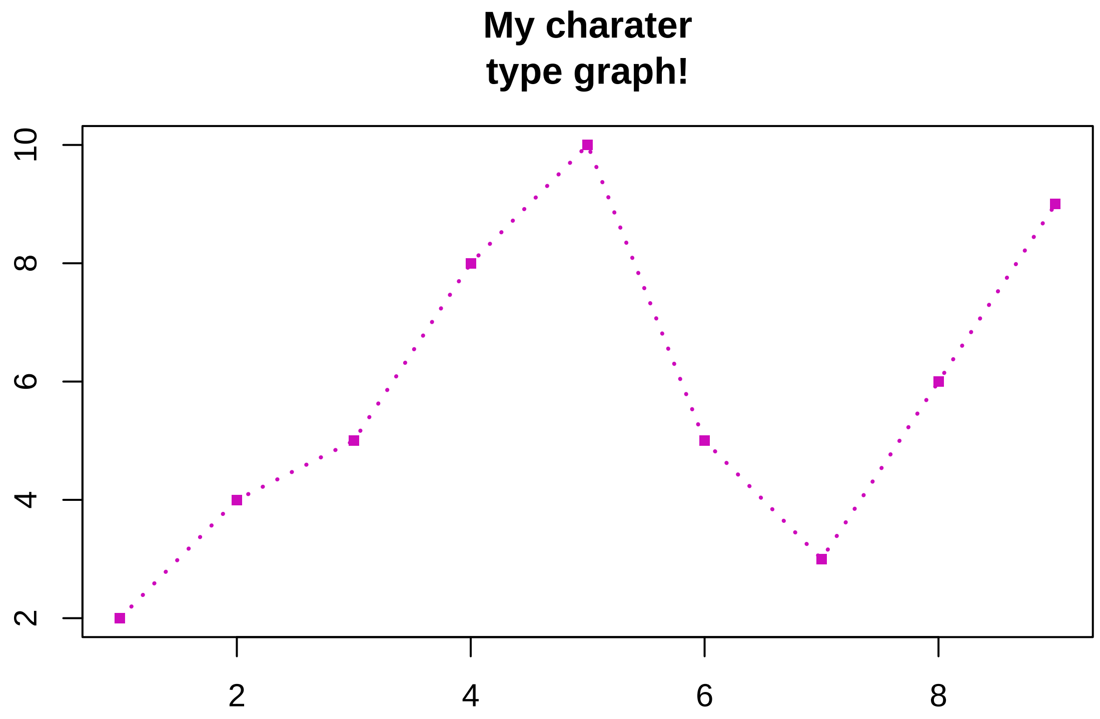


Awesome, now let’s play arouns with the other graphical parameters like pch, lty, cex, lwd.

```r
plot(b, type= "o", main= "My charater\ntype graph!", xlab= "", ylab= "", col= 4, pch= 17,
     lty= 2, cex= 1.5, lwd= 1.5)
```



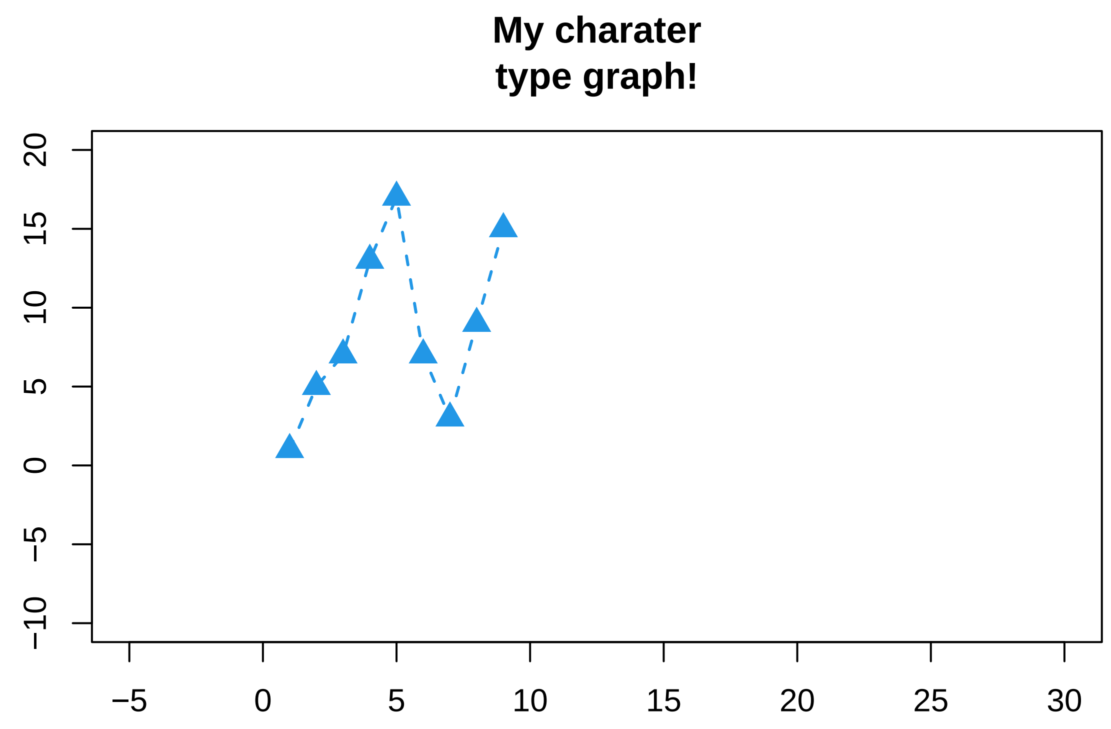
```r
plot(a, type= "o", main= "My charater\ntype graph!", xlab= "", ylab= "", col= 6, pch= 15,
     lty= 3, cex= 0.7, lwd= 2)
```



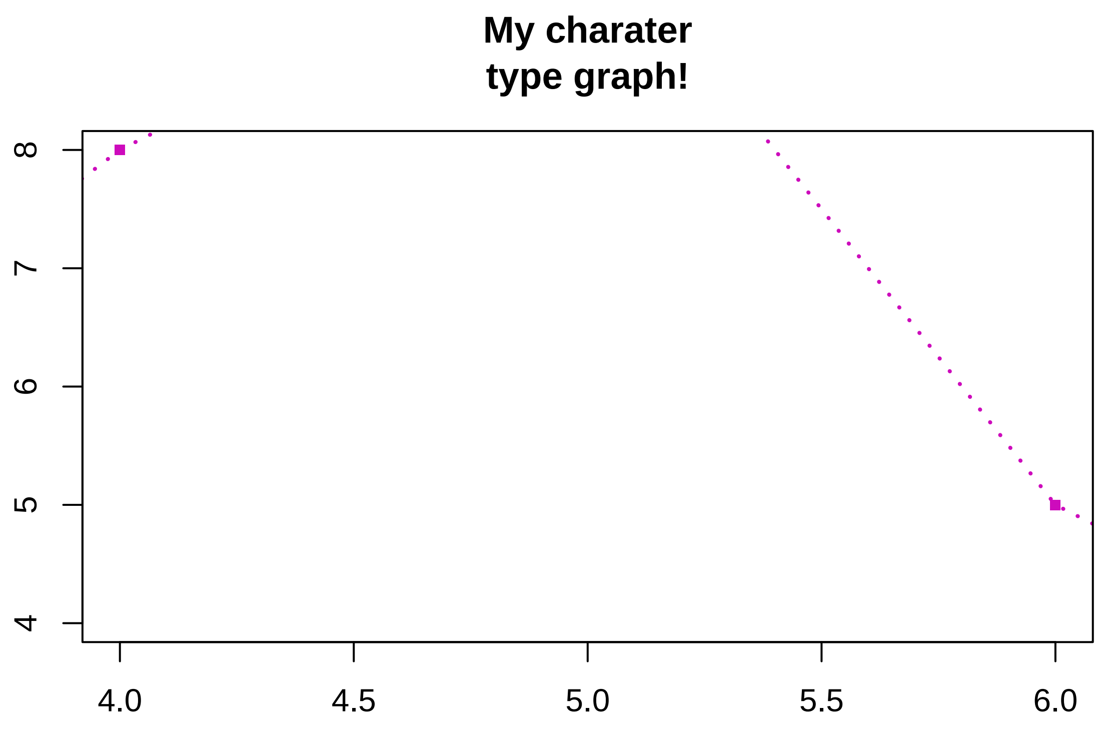
pch 8 is stars, 9-14 are weird shapes, pch 15-20 is filled shapes. cex is the size of the point, lwd is the size/boldness of the line, and lty is the amount of dashes. For instances, the higher the number, the more dashes there are.

Next, we can work with setting plot limits for the axis', so we'll just copy/paste the code from above and see how the graph changes, by adding the limits.

```
plot(b, type= "o", main= "My charater\ntype graph!", xlab= "", ylab= "", col= 4, pch= 17,
     lty= 2, cex= 1.5, lwd= 1.5, xlim= c(-5,30), ylim= c(-10, 20))
```



```
plot(a, type= "o", main= "My charater\ntype graph!", xlab= "", ylab= "", col= 6, pch= 15,
     lty= 3, cex= 0.7, lwd= 2, xlim= c(4,6), ylim= c(4,8))
```



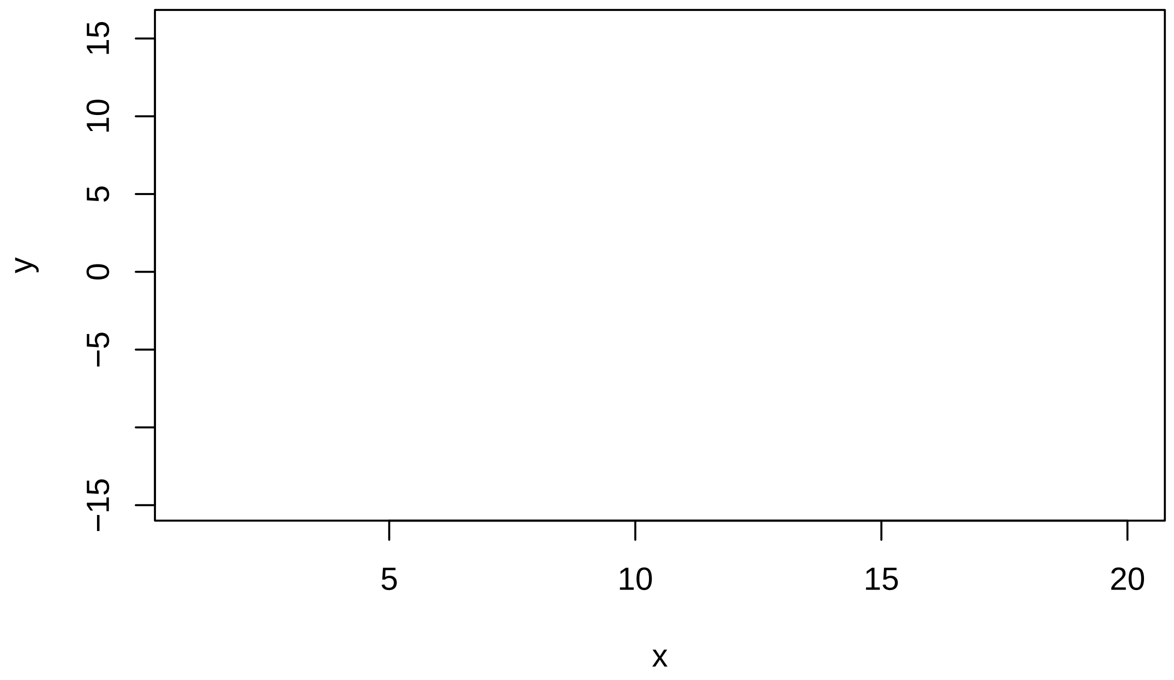
Crafting a Plot with Various Elements

Okay, now we're going to follow along with a book example for crafting a more complex plot that includes segments, arrows, a legend, and more. First, we need to enter the data for the plot.

```
x <- 1:20
y <- c(-1.49, 3.37, 2.59, -2.78, -3.94, -0.92, 6.43, 8.51, 3.41, -8.23, -12.01, -6.58,
       2.87, 14.12, 9.63, -4.58, -14.78, -11.67, 1.17, 15.62)
```

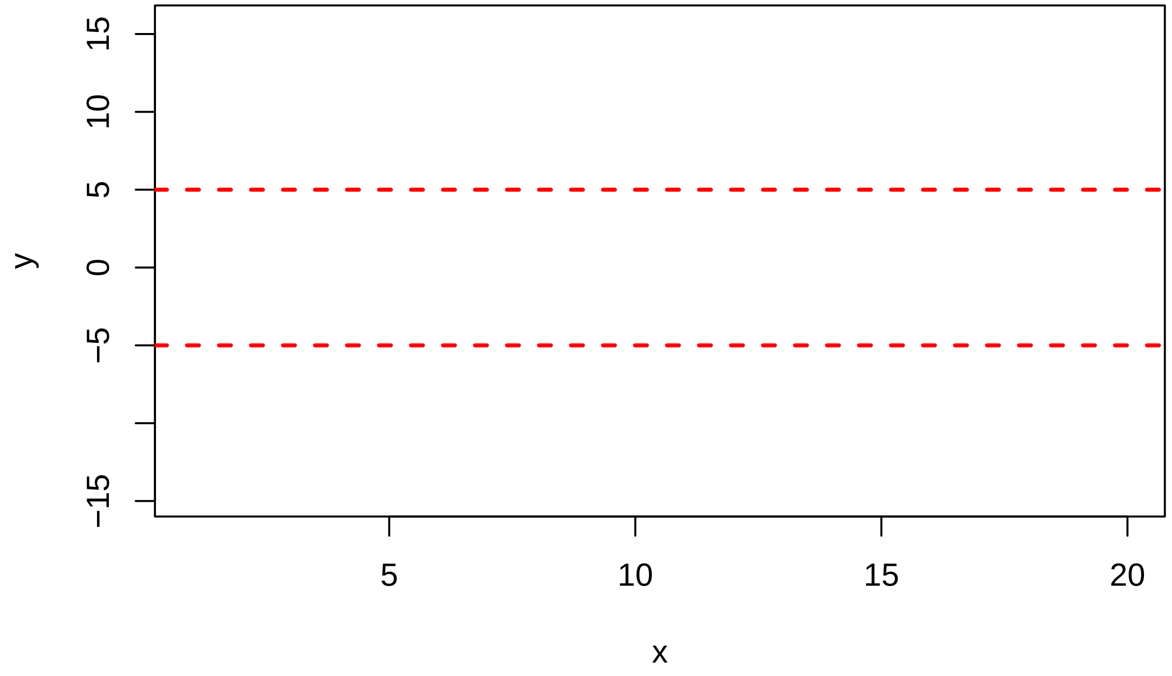
Step 1: We're going to set up an empty plotting region to work in.

```
plot(x,y, type= "n", main= "")
```



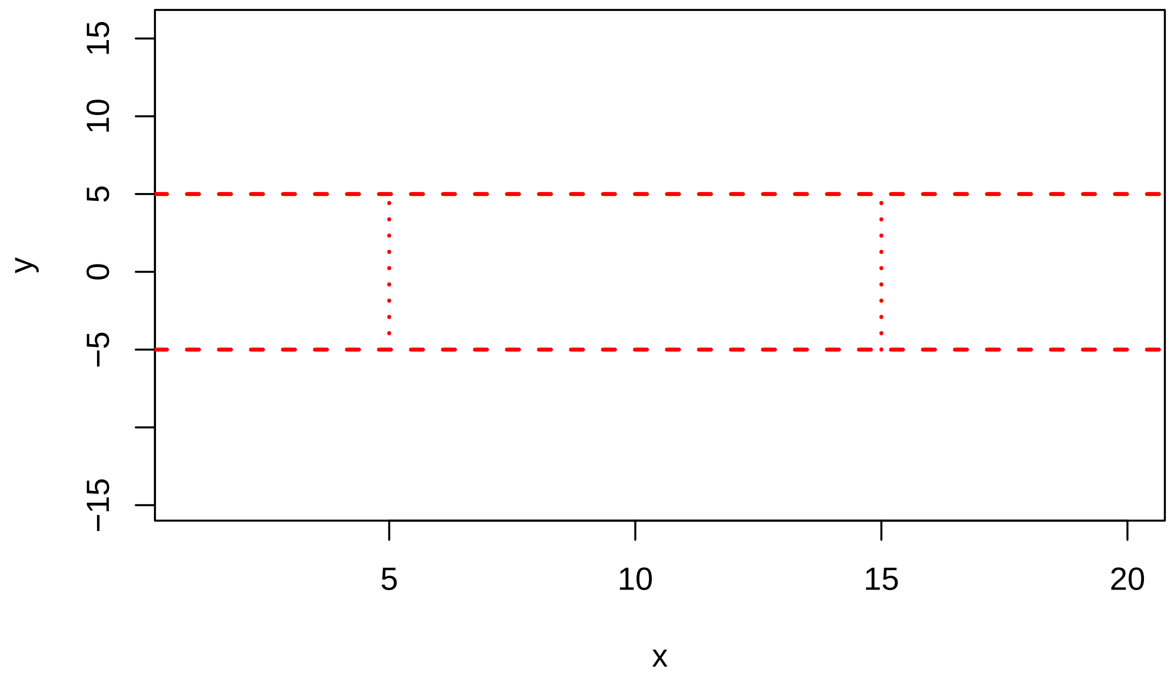
Step 2: Now we're going to begin adding the lines to denote the plot's 'sweet spot'

```
plot(x,y, type= "n", main= "")
abline(h=c(-5,5), col="red", lty=2, lwd=2)
```



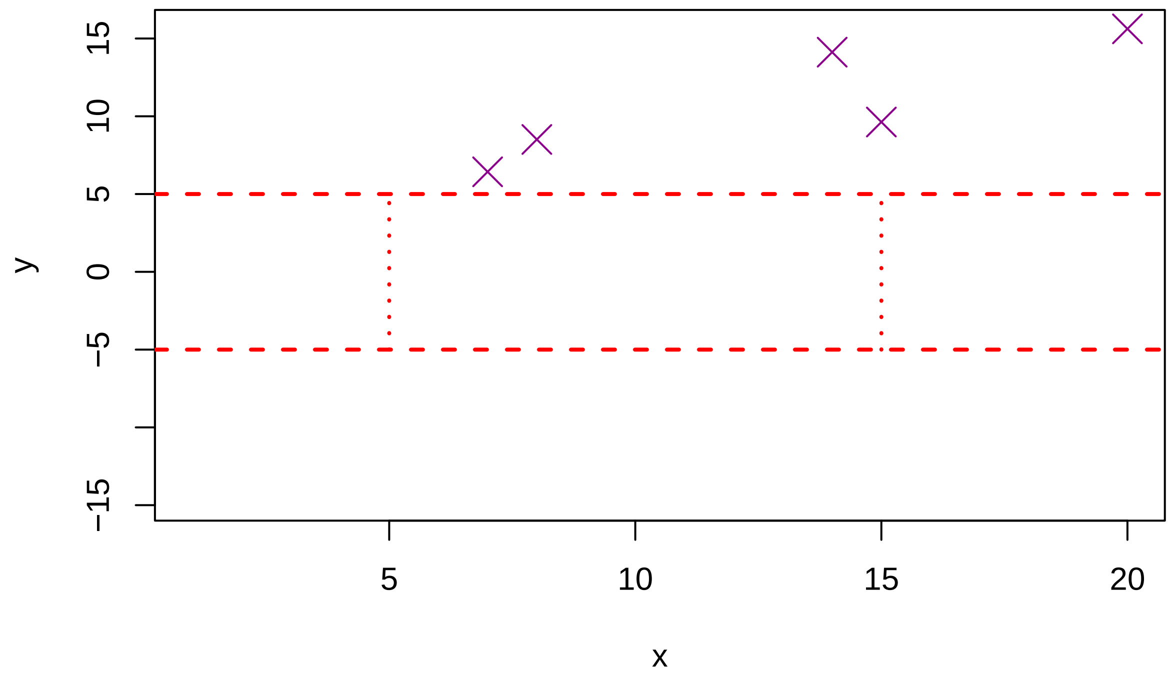
Step 3:

```
plot(x,y, type= "n", main= "")
abline(h=c(-5,5), col="red", lty=2, lwd=2)
segments(x0=c(5,15), y0=c(-5,-5), x1=c(5,15), y1=c(5,5), col="red", lty=3, lwd=2)
```



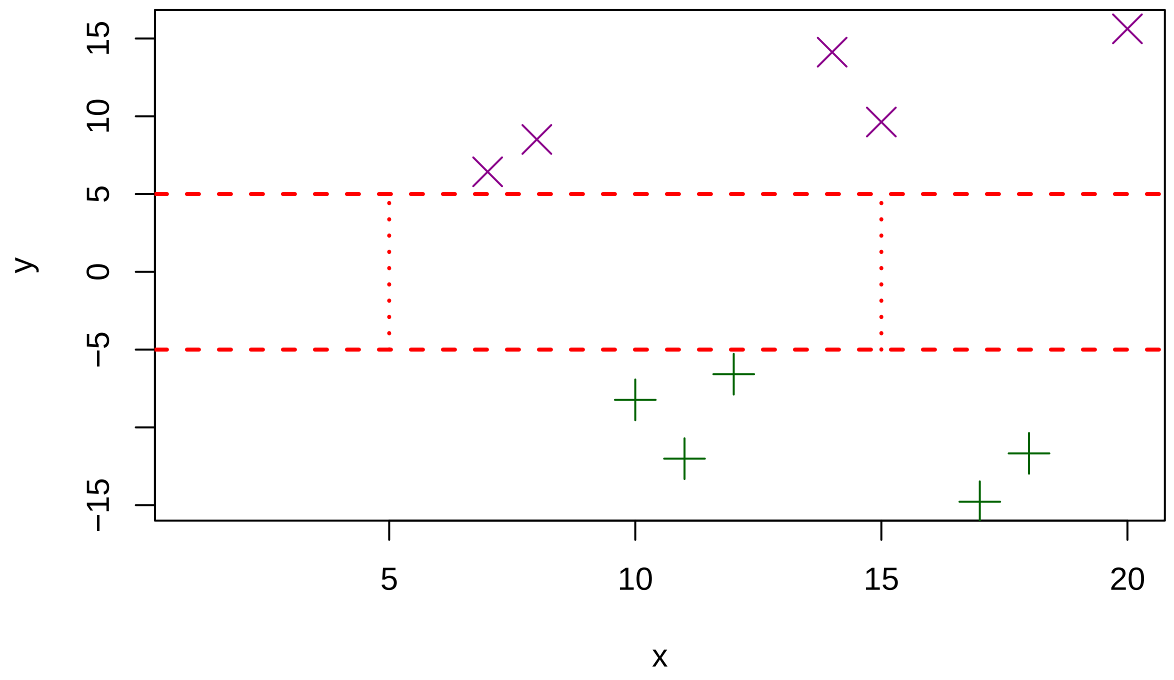
Step 4: Now we're going to be adding points by their range (using logical subsetting) to specify their point's asthetics.

```
plot(x,y, type= "n", main= "")
abline(h=c(-5,5), col="red", lty=2, lwd=2)
segments(x0=c(5,15), y0=c(-5,-5), x1=c(5,15), y1=c(5,5), col="red", lty=3, lwd=2)
points(x[y>=5],y[y>=5], pch=4, col= "darkmagenta", cex=2)
```



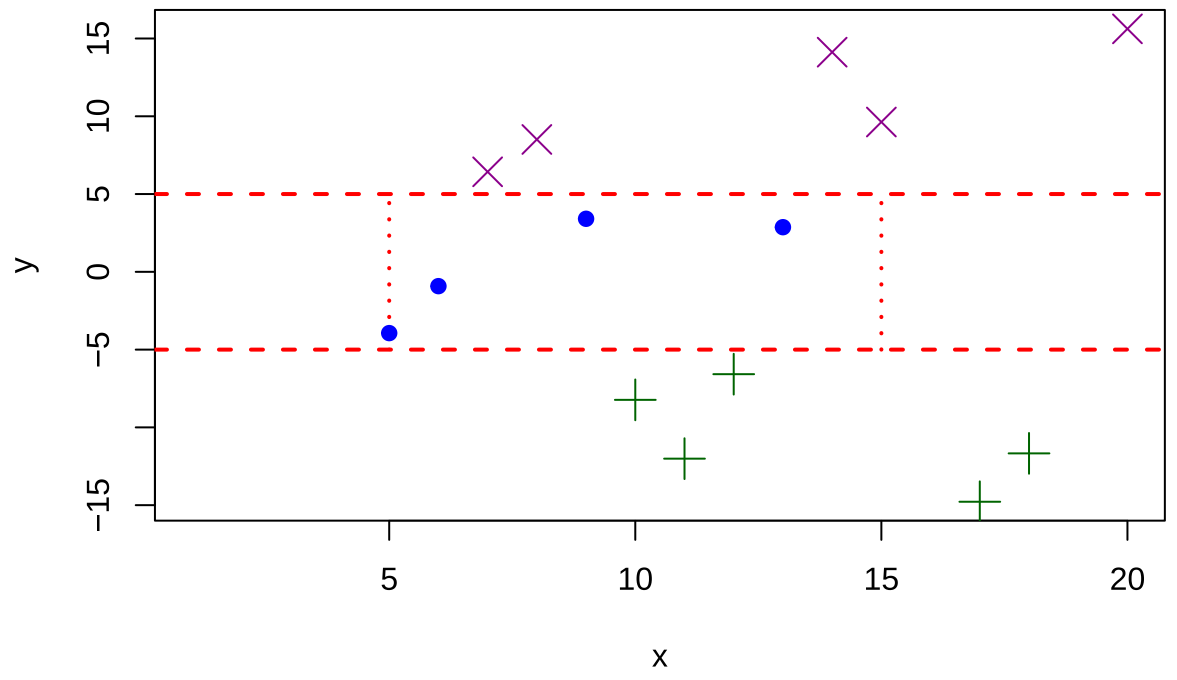
Step 5: Now we're going to add another group of points that fall within a specified range as green '+'

```
plot(x,y, type= "n", main= "")
abline(h=c(-5,5), col="red", lty=2, lwd=2)
segments(x0=c(5,15), y0=c(-5,-5), x1=c(5,15), y1=c(5,5), col="red", lty=3, lwd=2)
points(x[y>=5],y[y>=5], pch=4, col= "darkmagenta", cex=2)
points(x[y<=-5], y[y<=-5], pch=3, col="darkgreen", cex=2)
```



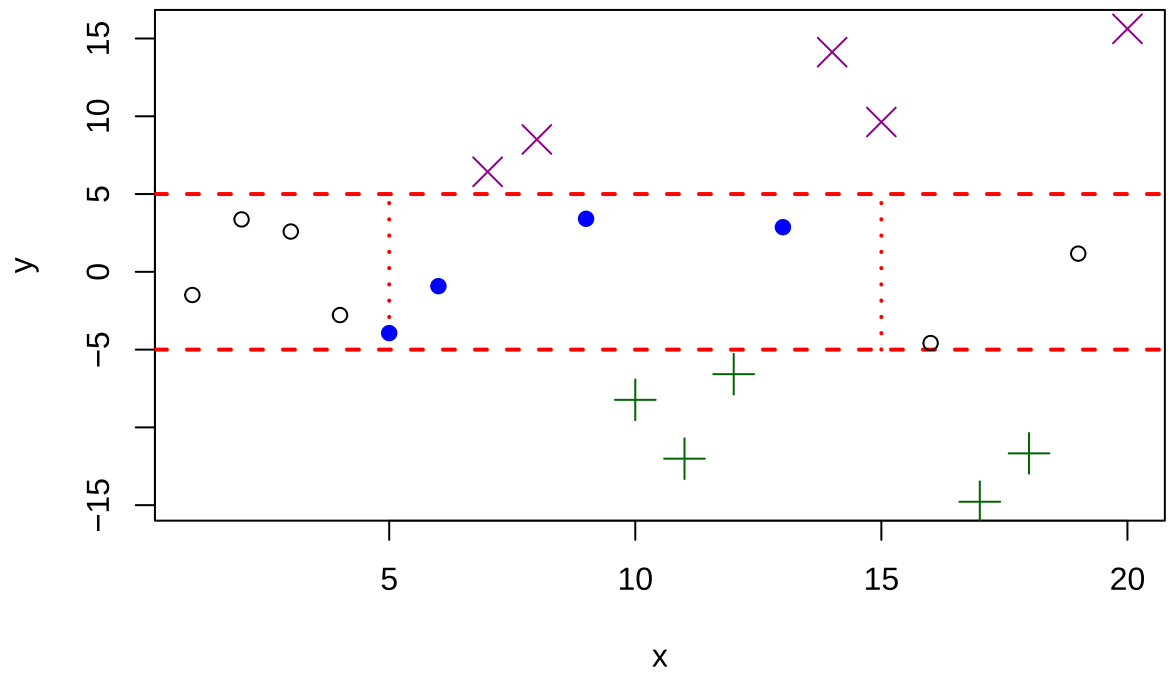
Step 6: Awesome, now we're going to add in the points that fall within the plot's 'sweet spot'

```
plot(x,y, type= "n", main= "")
abline(h=c(-5,5), col="red", lty=2, lwd=2)
segments(x0=c(5,15), y0=c(-5,-5), x1=c(5,15), y1=c(5,5), col="red", lty=3, lwd=2)
points(x[y>=5],y[y>=5], pch=4, col= "darkmagenta", cex=2)
points(x[y<=-5], y[y<=-5], pch=3, col="darkgreen", cex=2)
points(x[(x>=5 & x<=15)&(y>-5 & y<5)], y[(x>=5 & x<=15)&(y>-5 & y<5)], pch=19, col="blue")
```



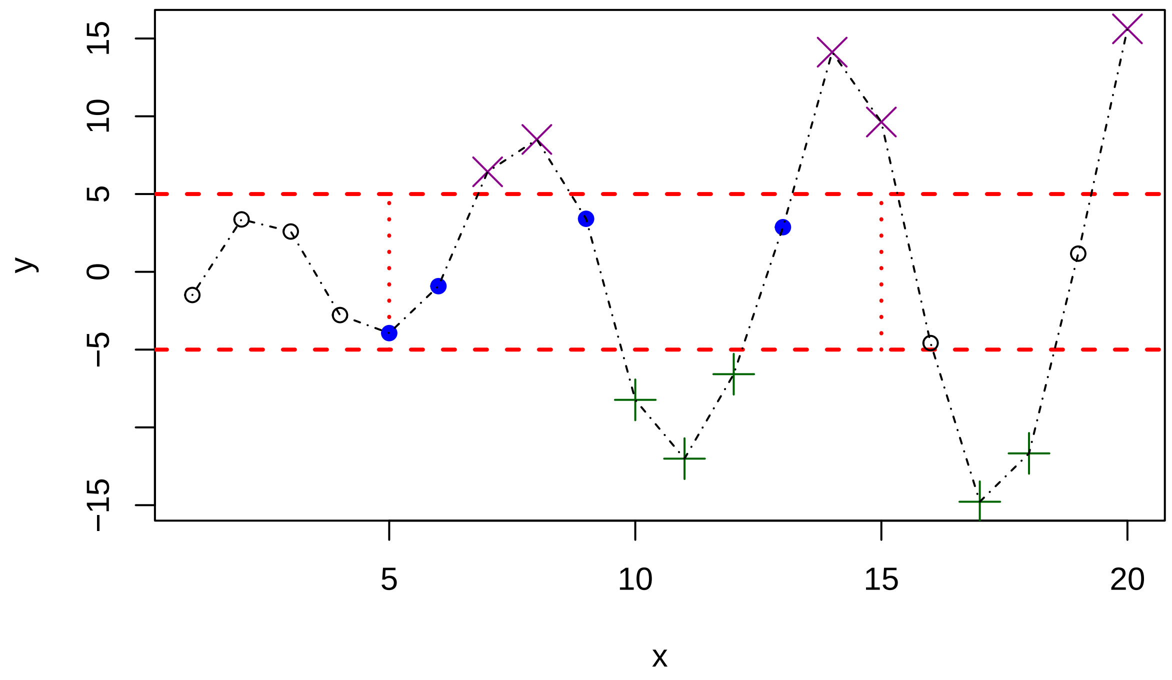
Step 7: Now we'll add the remaining data points with no graphical parameters specified, so they'll be just empty black circles plotted.

```r
plot(x,y, type= "n", main= "")
abline(h=c(-5,5), col="red", lty=2, lwd=2)
segments(x0=c(5,15), y0=c(-5,-5), x1=c(5,15), y1=c(5,5), col="red", lty=3, lwd=2)
points(x[y>=5],y[y>=5], pch=4, col= "darkmagenta", cex=2)
points(x[y<=-5], y[y<=-5], pch=3, col="darkgreen", cex=2)
points(x[(x>=5 & x<=15)&(y>-5 & y<5)], y[(x>=5 & x<=15)&(y>-5 & y<5)], pch=19, col="blue")
points(x[(x<5 | x>15) & (y>-5 & y<5)],y[(x<5 | x>15) & (y>-5 & y<5)])
```



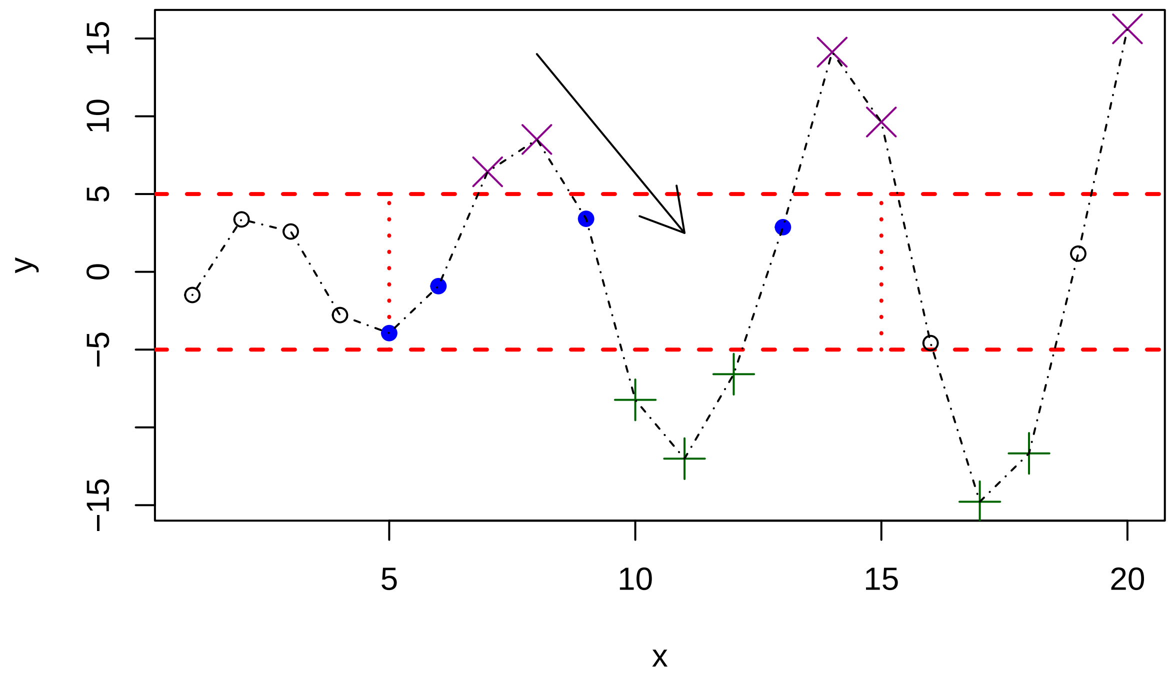
Step 8: Now we’re going to add a dotted line connecting all the points

```
plot(x,y, type= "n", main= "")
abline(h=c(-5,5), col="red", lty=2, lwd=2)
segments(x0=c(5,15), y0=c(-5,-5), x1=c(5,15), y1=c(5,5), col="red", lty=3, lwd=2)
points(x[y>=5],y[y>=5], pch=4, col= "darkmagenta", cex=2)
points(x[y<=-5], y[y<=-5], pch=3, col="darkgreen", cex=2)
points(x[(x>=5 & x<=15)&(y>-5 & y<5)], y[(x>=5 & x<=15)&(y>-5 & y<5)], pch=19, col="blue")
points(x[(x<5 | x>15) & (y>-5 & y<5)],y[(x<5 | x>15) & (y>-5 & y<5)])
lines(x,y,lty=4)
```



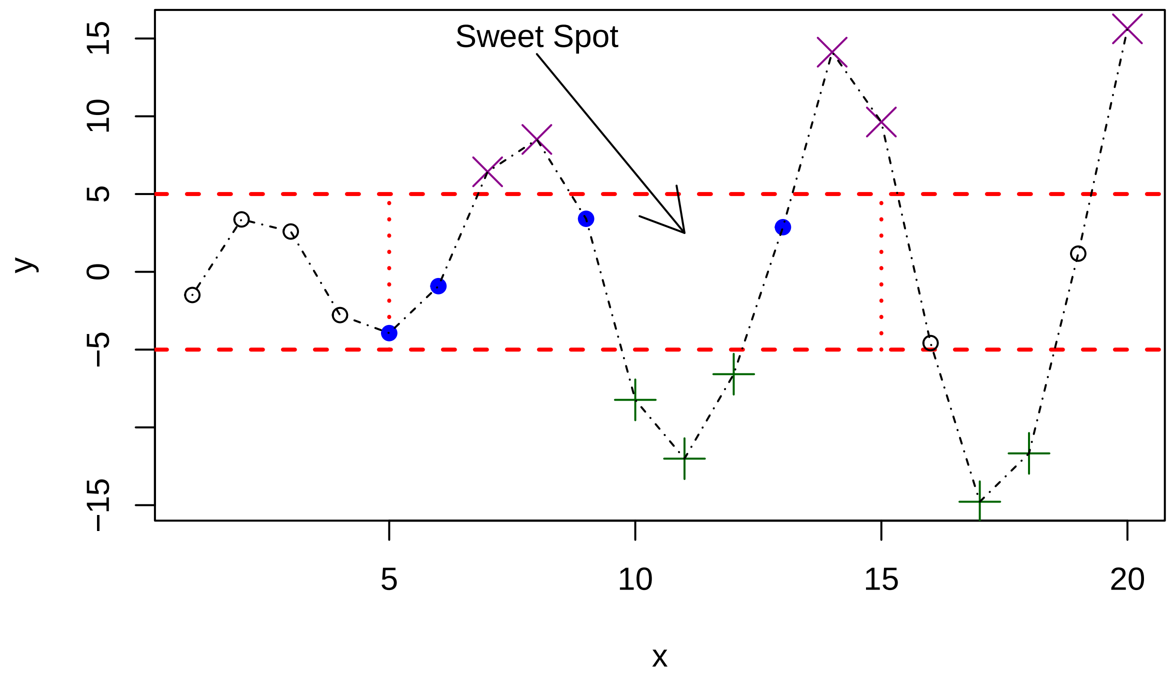
Step 9: Let's add an arrow now, pointing to the 'sweet spot' of the plot.

```
plot(x,y, type= "n", main= "")
abline(h=c(-5,5), col="red", lty=2, lwd=2)
segments(x0=c(5,15), y0=c(-5,-5), x1=c(5,15), y1=c(5,5), col="red", lty=3, lwd=2)
points(x[y>=5],y[y>=5], pch=4, col= "darkmagenta", cex=2)
points(x[y<=-5], y[y<=-5], pch=3, col="darkgreen", cex=2)
points(x[(x>=5 & x<=15)&(y>-5 & y<5)], y[(x>=5 & x<=15)&(y>-5 & y<5)], pch=19, col="blue")
points(x[(x<5 | x>15) & (y>-5 & y<5)],y[(x<5 | x>15) & (y>-5 & y<5)])
lines(x,y,lty=4)
arrows(x0=8, y0=14, x1=11, y1=2.5)
```



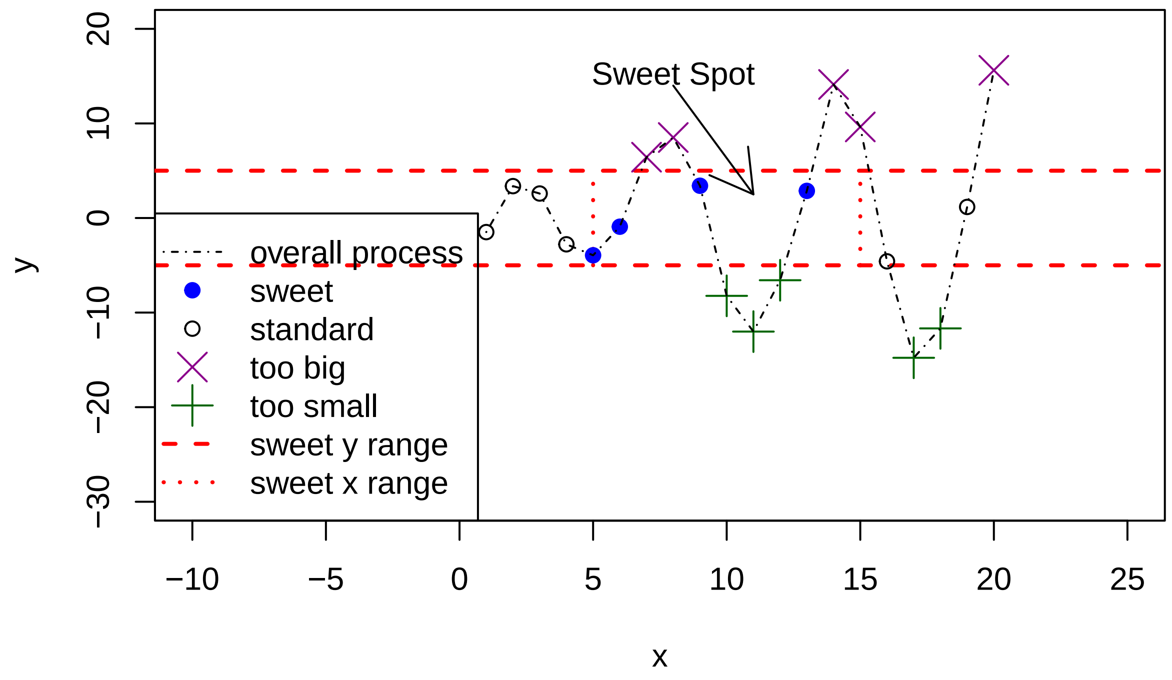
Step 10: Let's add a label for the arrow, letting the reader know what the arrow is pointing out.

```
plot(x,y, type= "n", main= "")
abline(h=c(-5,5), col="red", lty=2, lwd=2)
segments(x0=c(5,15), y0=c(-5,-5), x1=c(5,15), y1=c(5,5), col="red", lty=3, lwd=2)
points(x[y>=5],y[y>=5], pch=4, col= "darkmagenta", cex=2)
points(x[y<=-5], y[y<=-5], pch=3, col="darkgreen", cex=2)
points(x[(x>=5 & x<=15)&(y>-5 & y<5)], y[(x>=5 & x<=15)&(y>-5 & y<5)], pch=19, col="blue")
points(x[(x<5 | x>15) & (y>-5 & y<5)],y[(x<5 | x>15) & (y>-5 & y<5)])
lines(x,y,lty=4)
arrows(x0=8, y0=14, x1=11, y1=2.5)
text(x=8, y=15, labels="Sweet Spot")
```



Step 11: We'll add a legend as the final touch, to let the reader know what the various point styles signify.

```
plot(x,y, type= "n", main= "", xlim = c(-10, 25), ylim = c(-30, 20))
abline(h=c(-5,5), col="red", lty=2, lwd=2)
segments(x0=c(5,15), y0=c(-5,-5), x1=c(5,15), y1=c(5,5), col="red", lty=3, lwd=2)
points(x[y>=5],y[y>=5], pch=4, col= "darkmagenta", cex=2)
points(x[y<=-5], y[y<=-5], pch=3, col="darkgreen", cex=2)
points(x[(x>=5 & x<=15)&(y>-5 & y<5)], y[(x>=5 & x<=15)&(y>-5 & y<5)], pch=19, col="blue")
points(x[(x<5 | x>15) & (y>-5 & y<5)],y[(x<5 | x>15) & (y>-5 & y<5)])
lines(x,y,lty=4)
arrows(x0=8, y0=14, x1=11, y1=2.5)
text(x=8, y=15, labels="Sweet Spot")
legend("bottomleft",
       legend = c("overall process", "sweet", "standard", "too big", "too small", "sweet y range",
                  "sweet x range"),
       pch = c(NA,19, 1, 4, 3, NA, NA), lty = c(4, NA, NA, NA, NA, 2, 3),
       col = c("black", "blue", "black", "darkmagenta", "darkgreen", "red", "red"),
       lwd = c(1, NA, NA, NA, NA, 2, 2), pt.cex = c(NA, 1, 1, 2, 2, NA, NA))
```



I added 'x' and 'y' axis limits since without any, the legend prints over a lot of the data. I would like to see if there's a way to reduce the overall size of the legend itself, rather than needing to manipulate the size of the plot.