SIOB 296 Introduction to Programming with R

Eric Archer (eric.archer@noaa.gov)

Week 6 (May 9, 2017) - Graphics

Reading

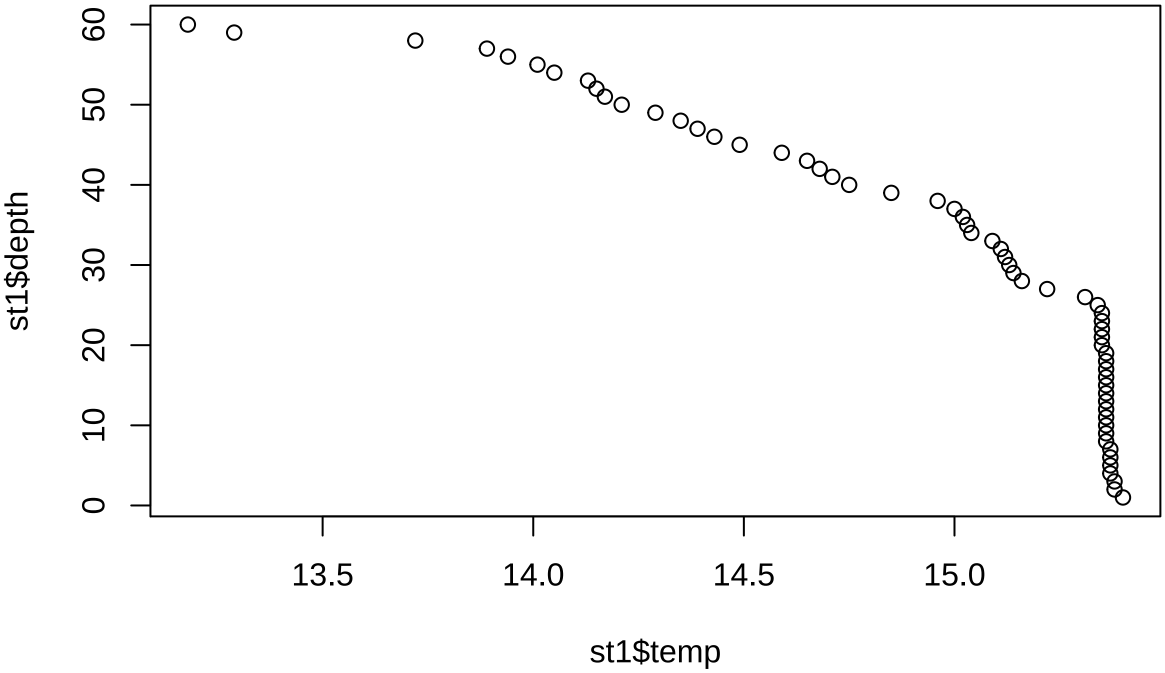
The Book of R:
Chapter 7.1, 7.2, 7.3
Chapter 14.1.1, 14.2, 14.3
Chapter 23.1.4, 23.2, 23.4, 23.5

The Art of R:
Chapter 12

Scatterplots

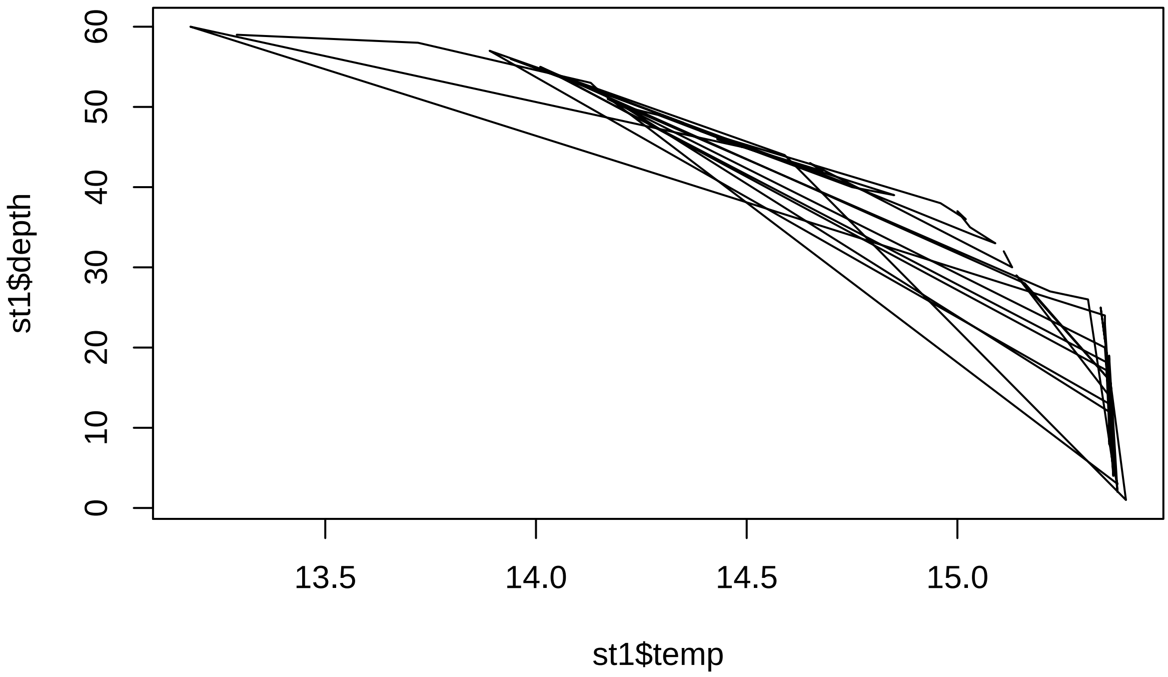
The most basic function for generating scatter and line plots is the function `plot`. The help for plot (`?plot`) is not that informative. You'll find more options with `?plot.default`. At its simplest it requires a vector of x values and a vector of y values. As an example, we'll plot the points representing temperature at depth for a single CTD cast:

```r
# read and subset data for a single cast
ctd <- read.csv("ctd.csv")
st1 <- subset(ctd, station == "Station.1" & sample_date == "2010-01-06")
# plot depth on the y axis and temperature on the x axis
plot(st1$temp, st1$depth)
```



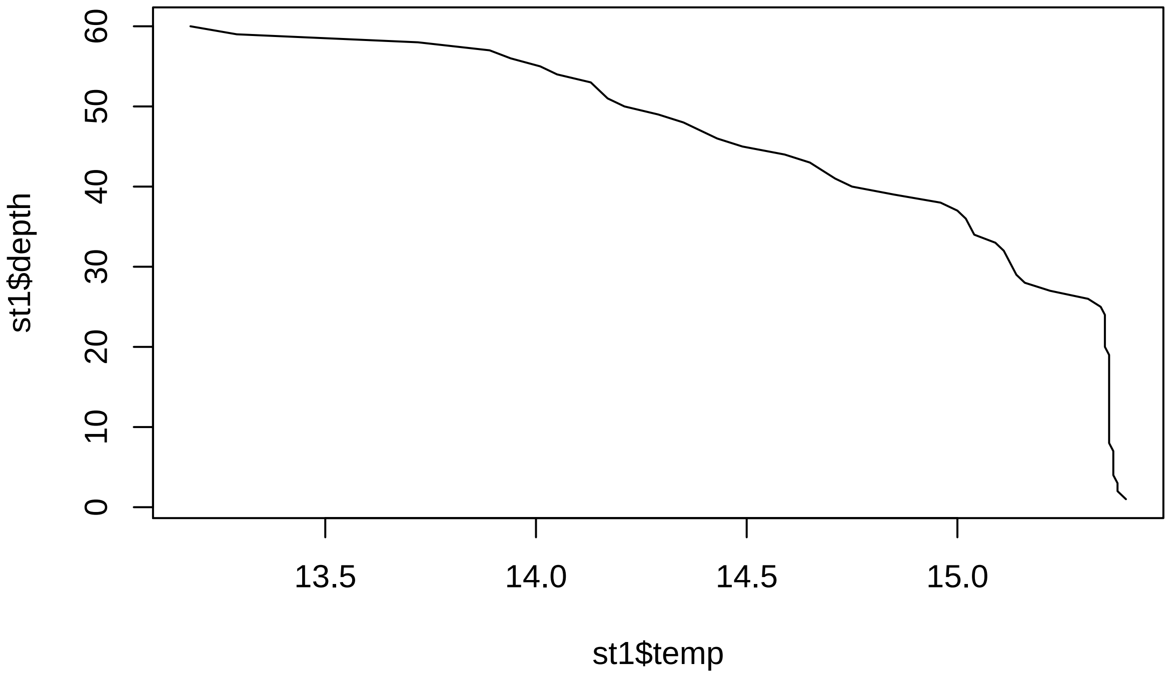
Changing the `type` argument selects (p)oints, (l)ines, or (b)oth.

```
plot(st1$temp, st1$depth, type = "l")
```



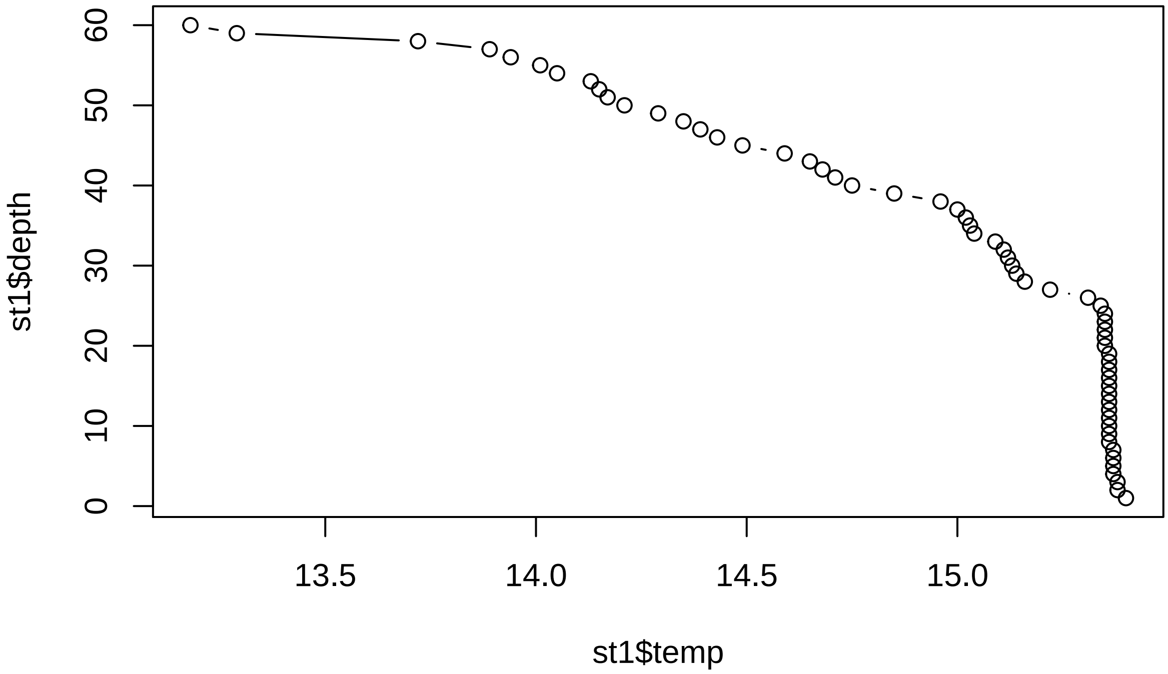
But if you plot lines, they are connectd in the order they occur in the vectors. In order to produce the trace properly, we have to sort the data frame by depth:

```
st1 <- st1[order(st1$depth), ]
plot(st1$temp, st1$depth, type = "l")
```



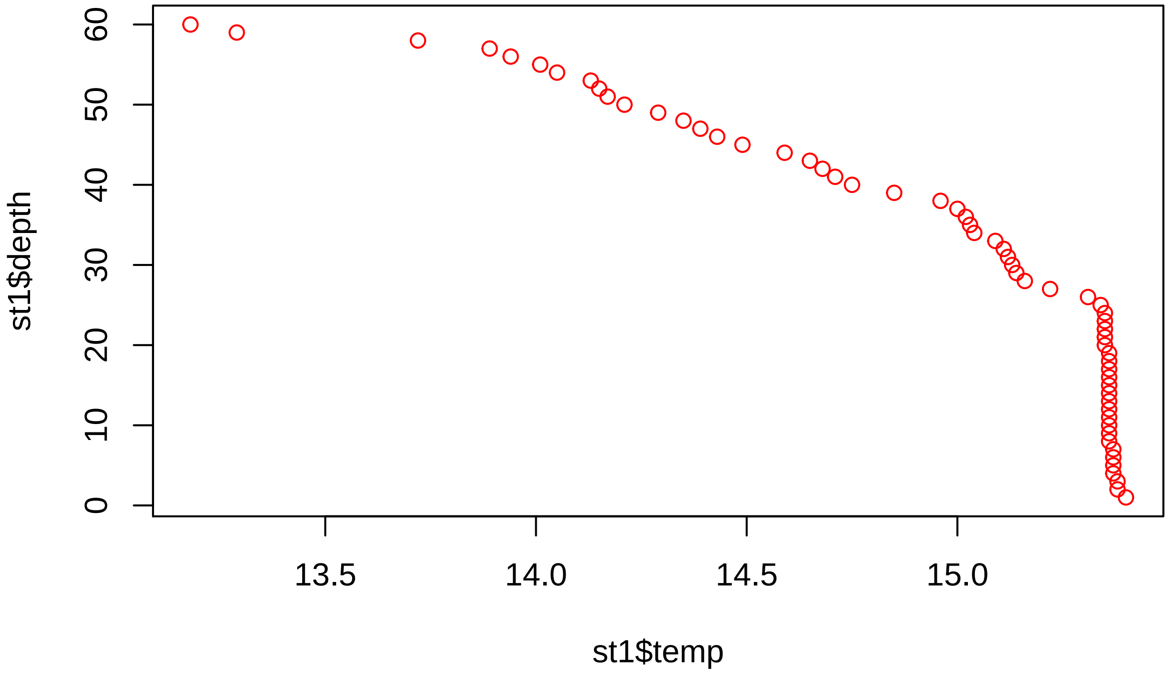
… and here is the same plot with both lines and points:

```
plot(st1$temp, st1$depth, type = "b")
```



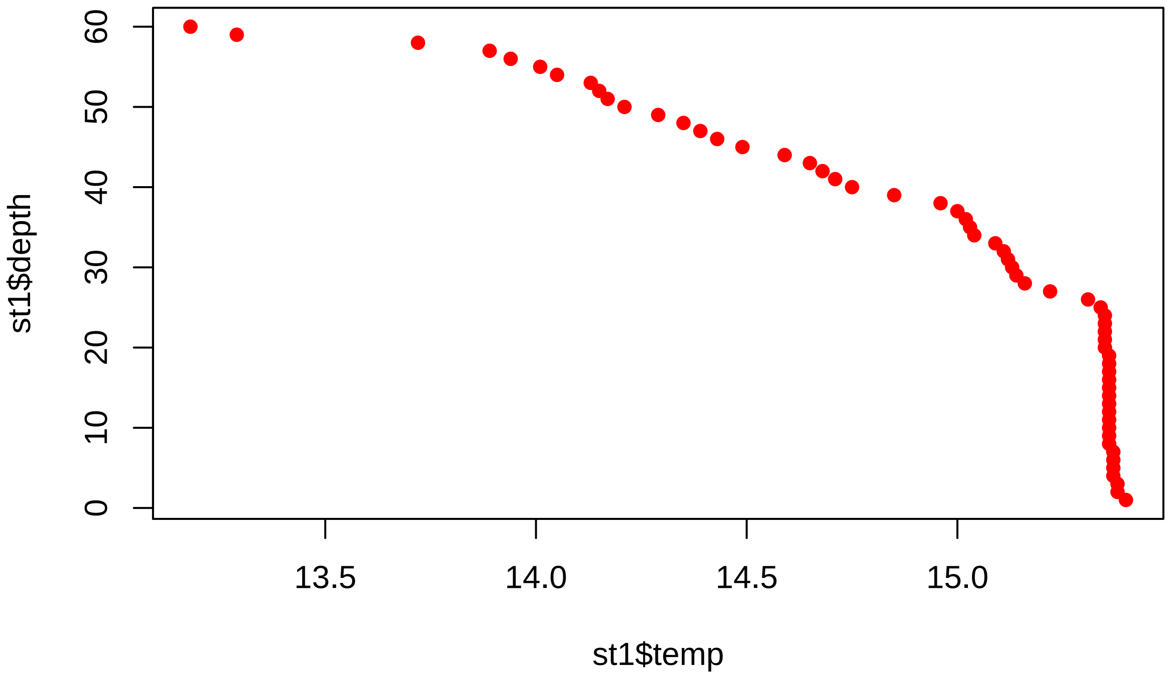
Colors can be changed using the `col` argument:

```
plot(st1$temp, st1$depth, col = "red")
```



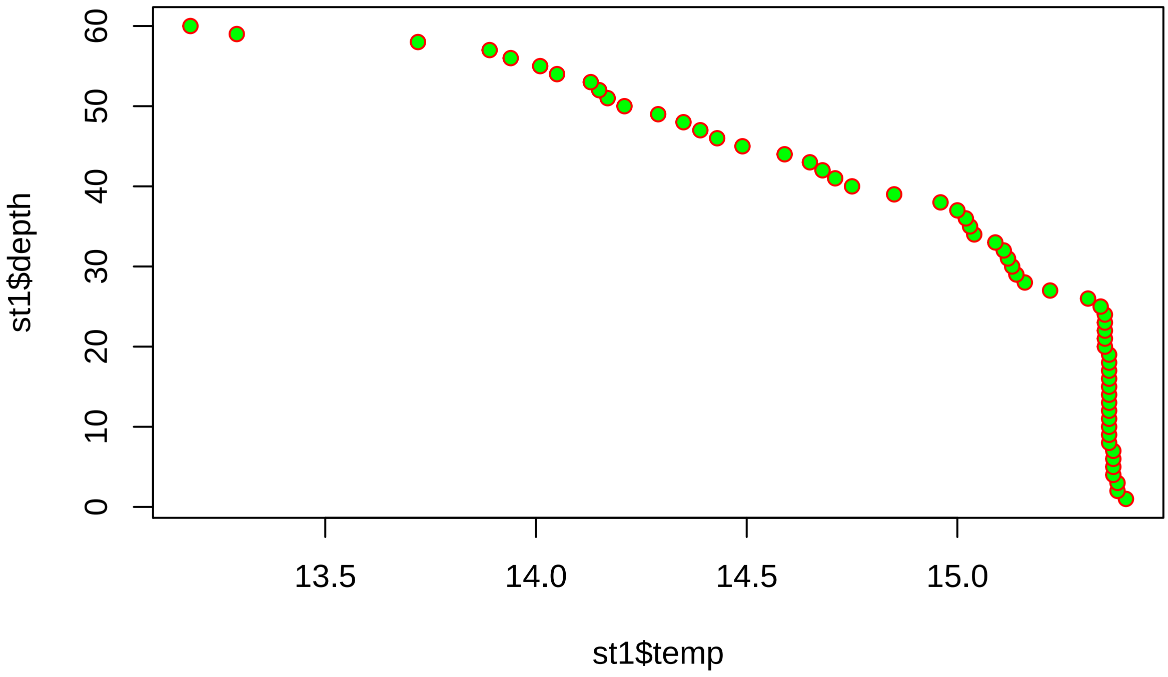
However, depending on the type of point, chosen, the `col` can refer to the outline color or the center color. The point type is chosen by assigning the `pch` argument with a number from 0 to 25. The coresponding shapes are given in `?points`. `pch = 16` produces a solid circle:

```r
plot(st1$temp, st1$depth, col = "red", pch = 16)
```



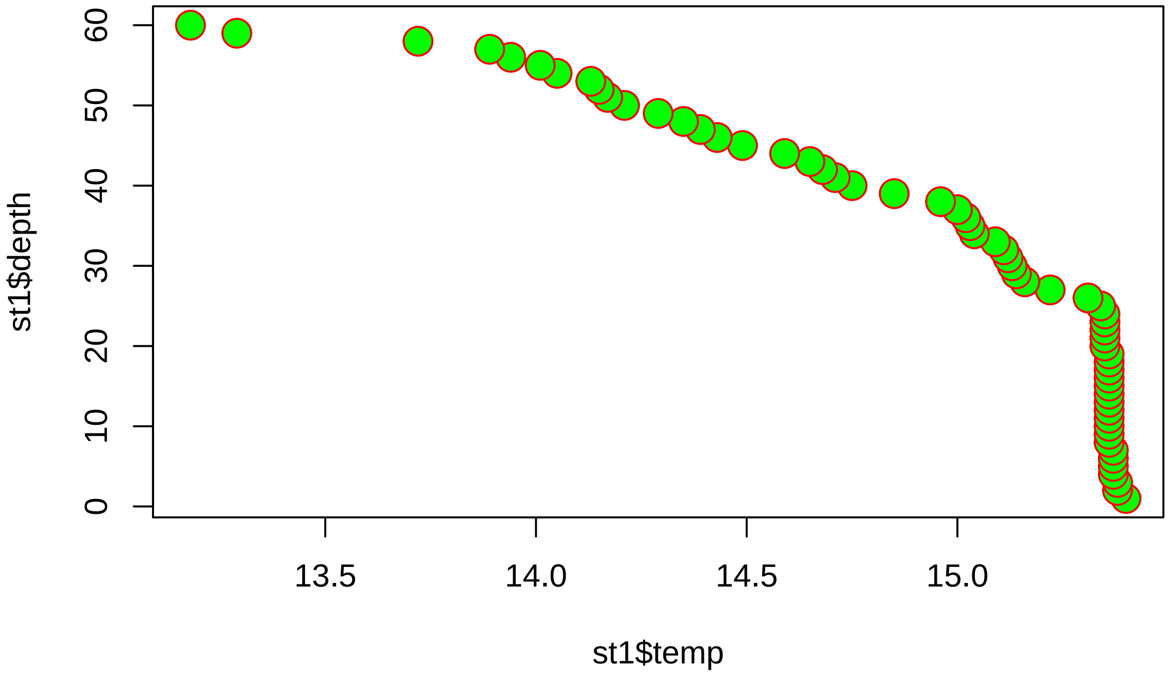
`pch = 21` produces a filled circle. In this case, `col` sets the outer color and `bg` sets the inner color:

```r
plot(st1$temp, st1$depth, col = "red", bg = "green", pch = 21)
```



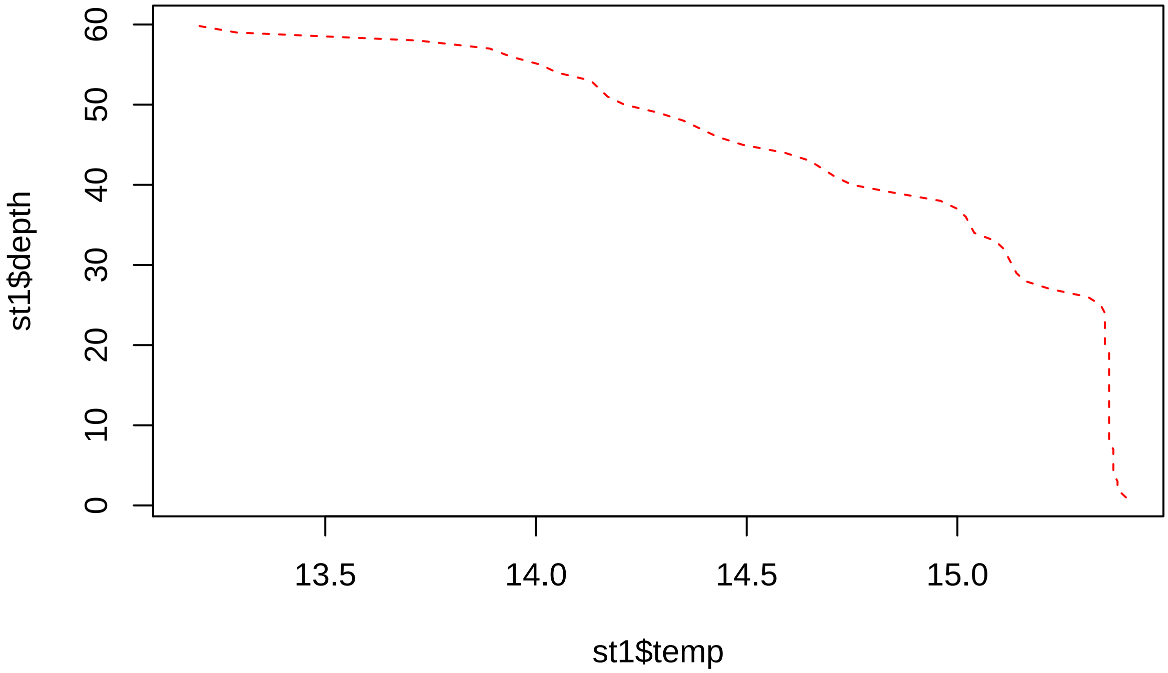
Point size is changed by `cex`. More information about `cex` and many other graphical parameters can be found in `?par`.

```r
plot(st1$temp, st1$depth, col = "red", bg = "green", pch = 21, cex = 2)
```



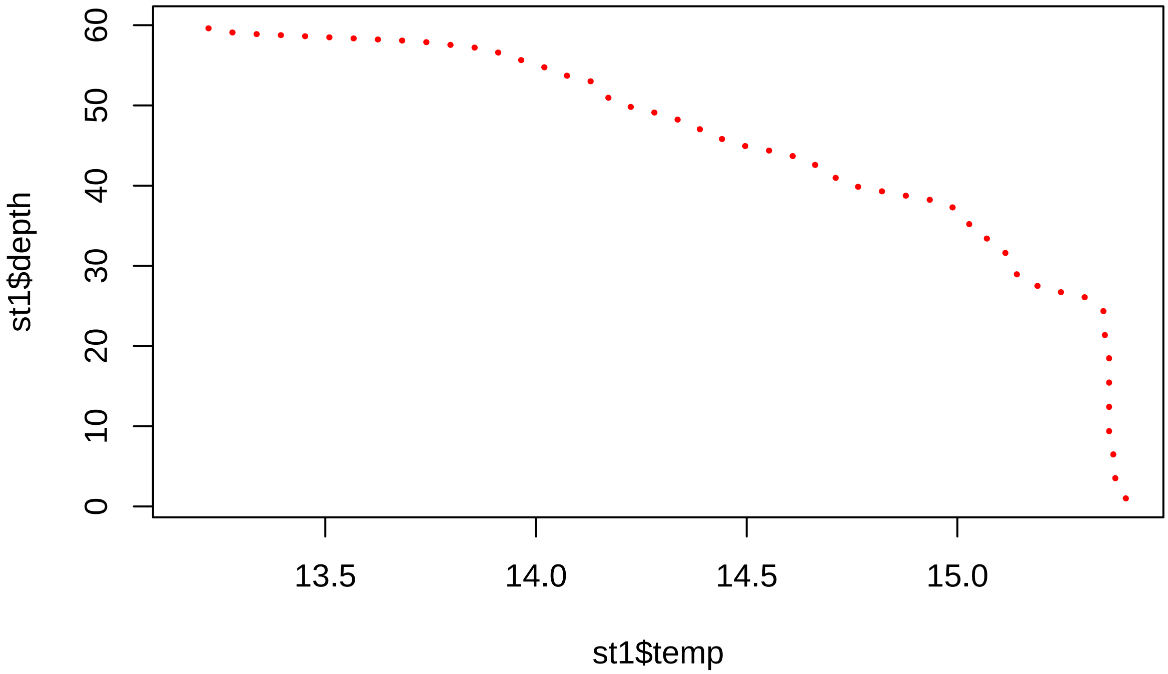
There are multiple line types as well, which can be specified with the graphical parameter `lty`.

```r
plot(st1$temp, st1$depth, type = "l", col = "red", lty = "dashed")
```



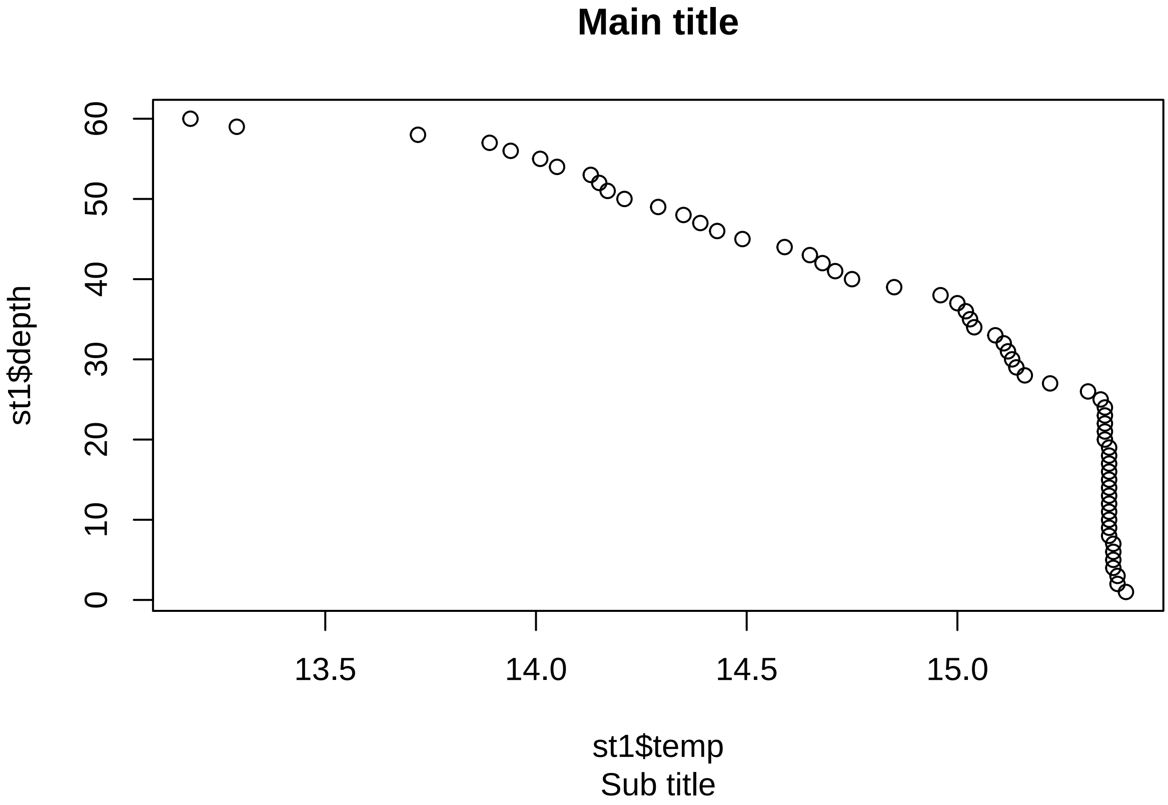
Line width is controlled with `lwd`:

```r
plot(st1$temp, st1$depth, type = "l", col = "red", lty = "dotted", lwd = 3)
```



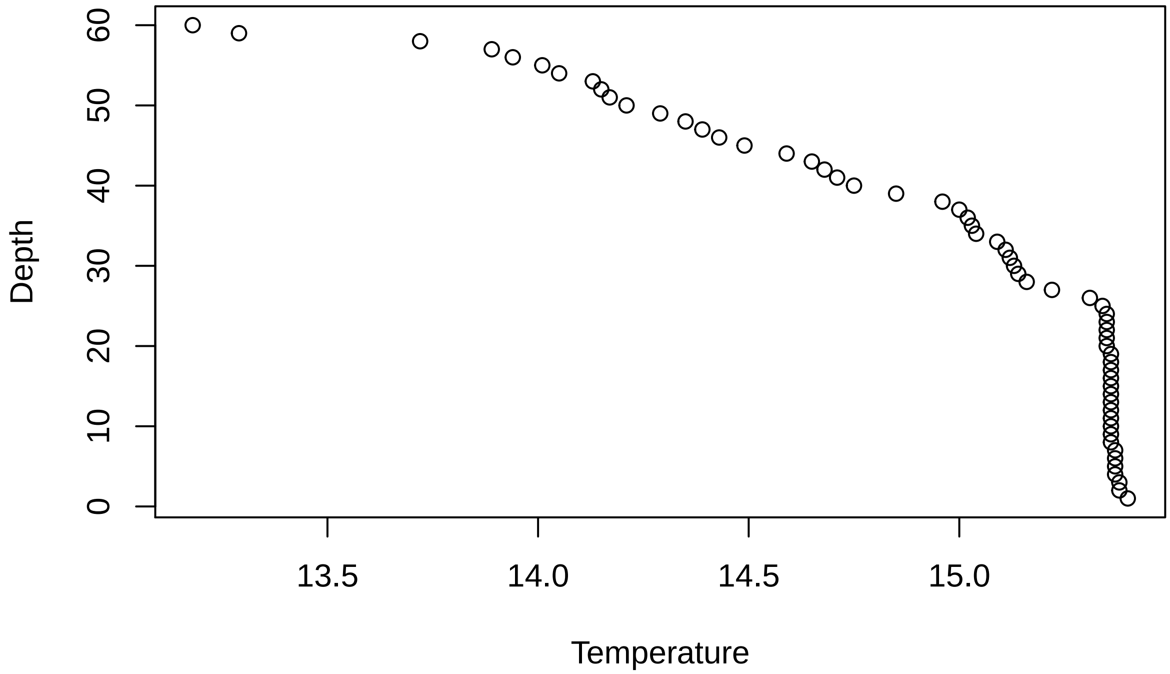
The `main` and `sub` arguments to plot allow you to specify main- and subtitles:

```
plot(st1$temp, st1$depth, main = "Main title", sub = "Sub title")
```



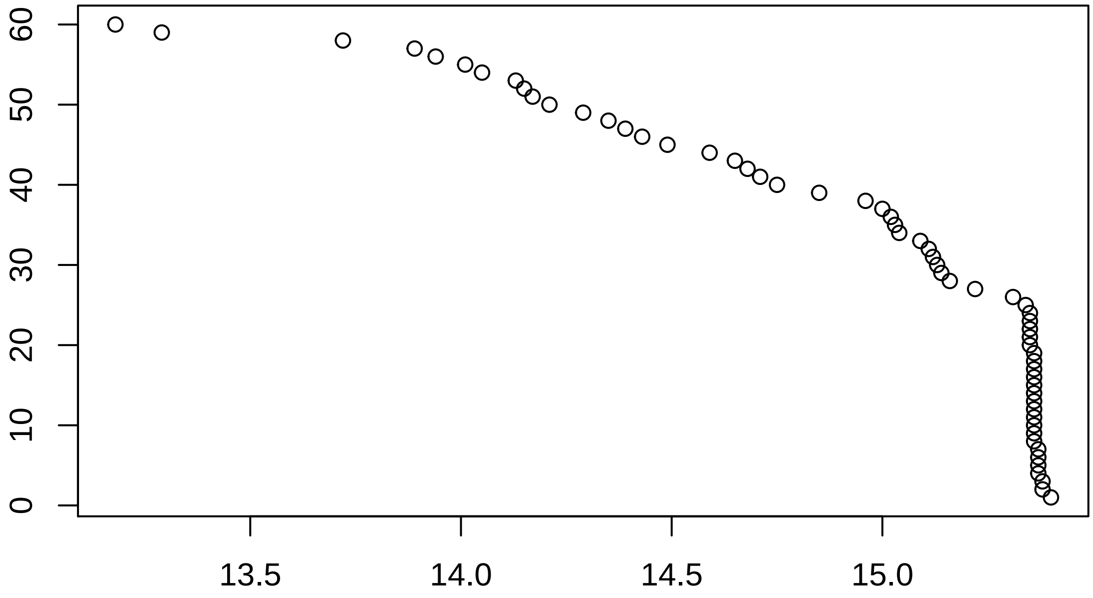
Labels for the axes are chosen by default, but can also be changed with the `xlab` and `ylab` arguments:

```
plot(st1$temp, st1$depth, xlab = "Temperature", ylab = "Depth")
```



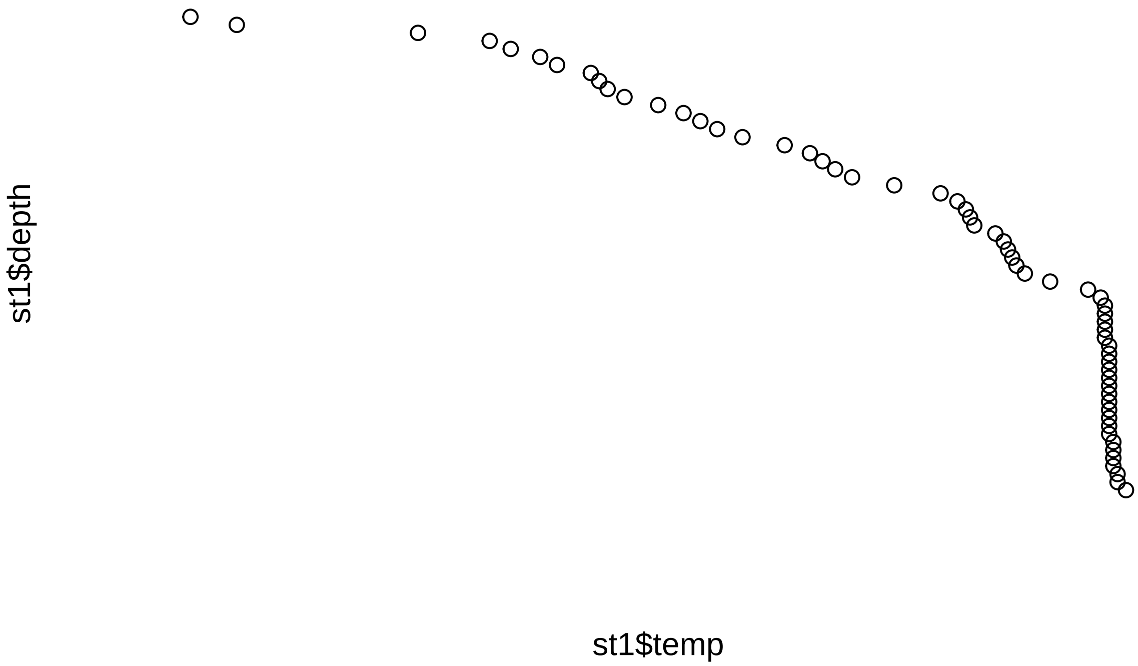
The default annotation (title and axis labels) can be turned off by setting `ann = FALSE`.

```
plot(st1$temp, st1$depth, ann = FALSE)
```



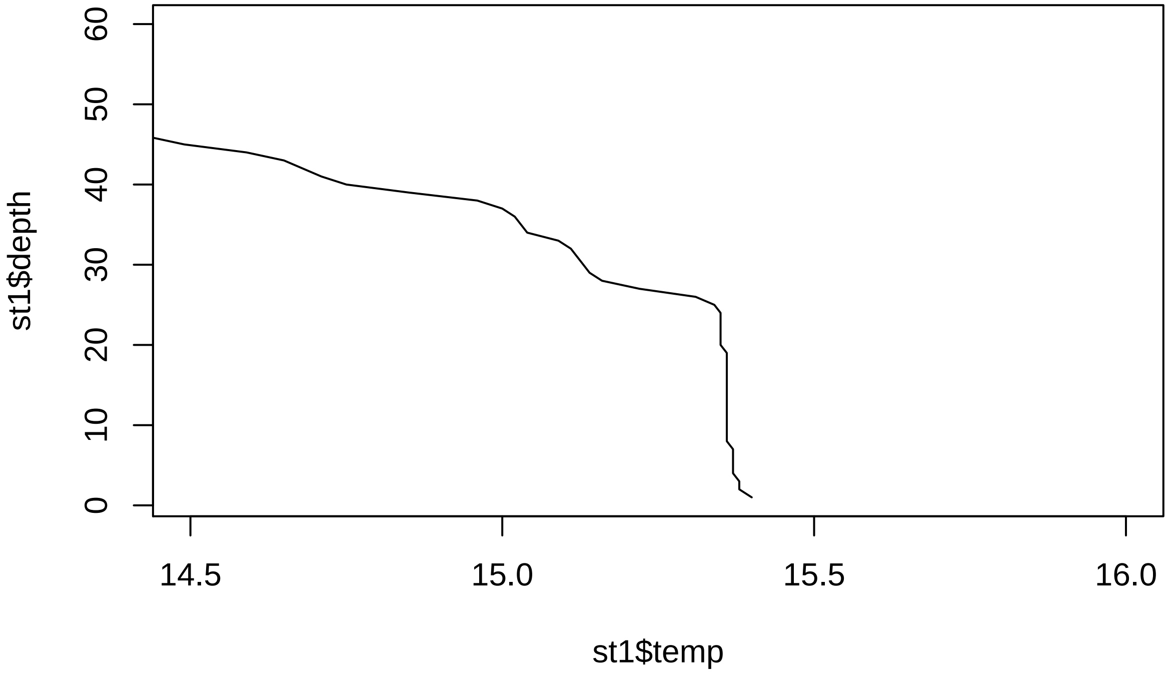
Likewise, the axes can be turned off with `axes = FALSE`. You may want to do this if you wish to customize axis or label placement or style, which we'll do below.

```
plot(st1$temp, st1$depth, axes = FALSE)
```



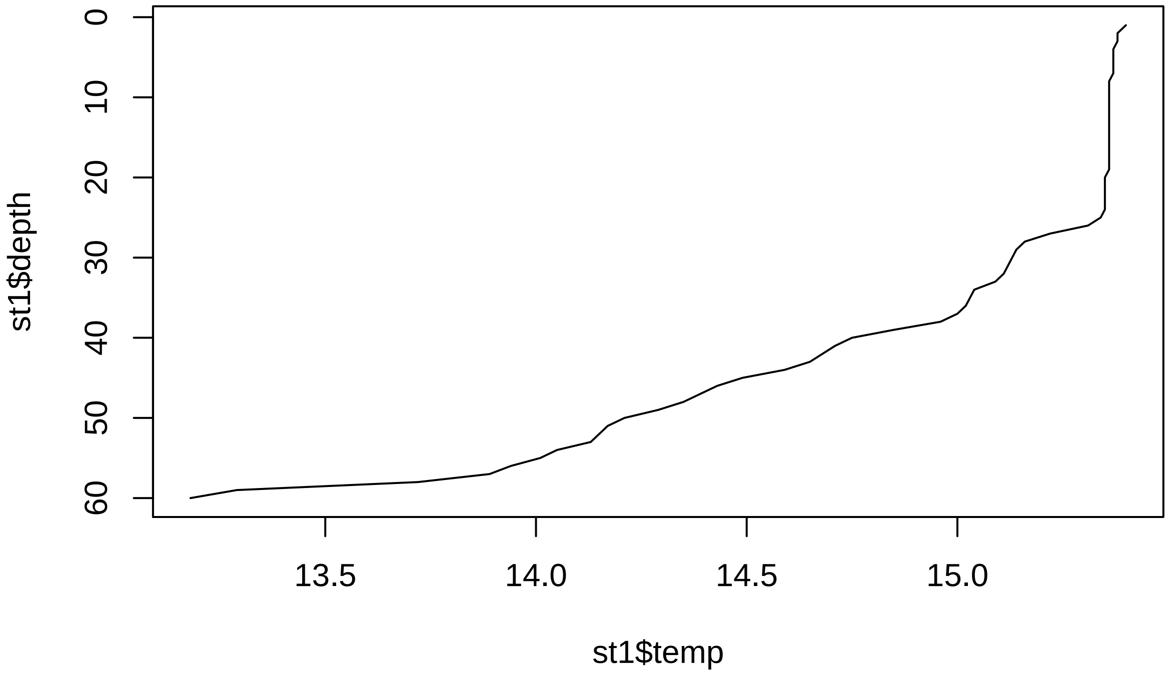
Axis limits can be controlled with `xlim` and `ylim`. Note that if either of these are specified, the excluded data will not be shown.

```r
plot(st1$temp, st1$depth, type = "l", xlim = c(14.5, 16))
```



Because we want depth on the y-axis to be displayed with zero at the top and increasing as it goes down, we can supply a reversed range for `ylim`:

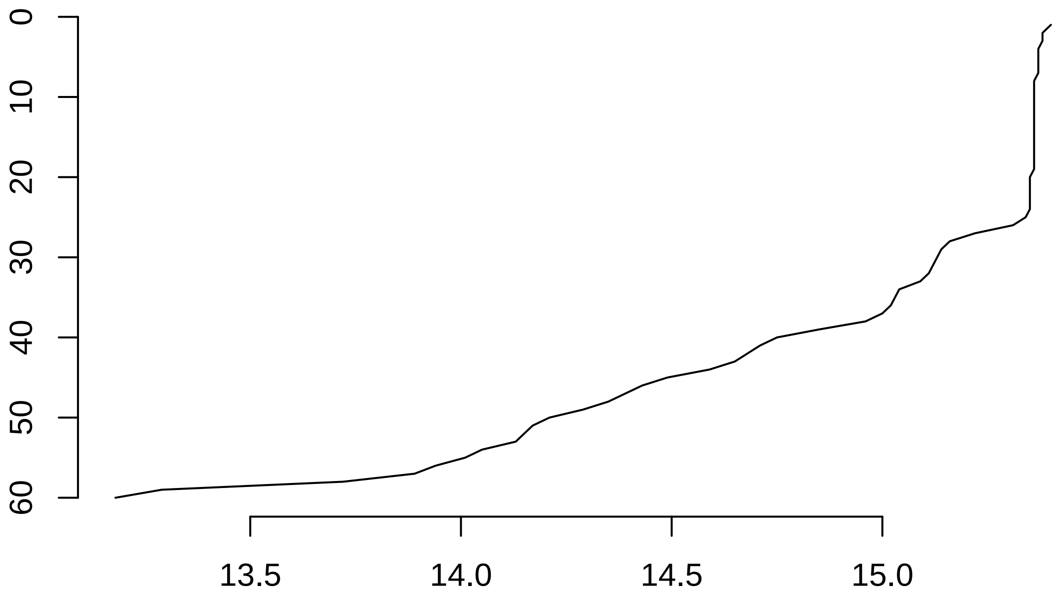
```r
plot(st1$temp, st1$depth, type = "l", ylim = rev(range(st1$depth)))
```



Axes

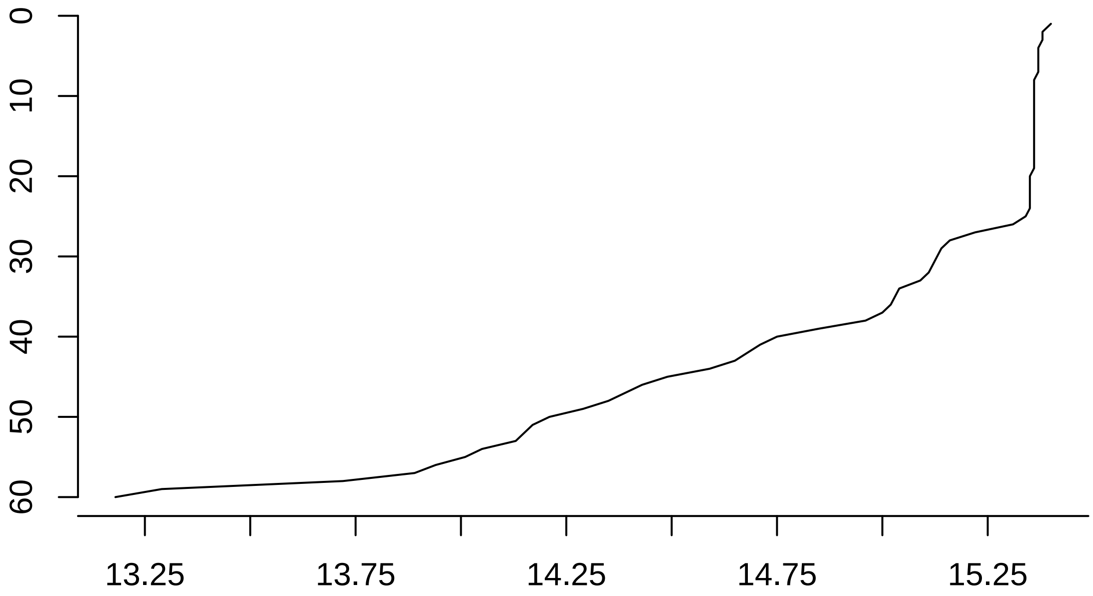
Axis ticks and labels can be customized, but they need to be first removed from the main plot, then added back individually with the `axis()` function. The first argument to the axis function (`side`) specifies the side the axis should be displayed on. This argument is a number from 1 to 4 where 1 = bottom, 2 = left, 3 = top, and 4 = right.

```
plot(
  st1$temp, st1$depth,
  type = "l", ylim = rev(range(st1$depth)),
  ann = FALSE, axes = FALSE
)
axis(1)
axis(2)
```



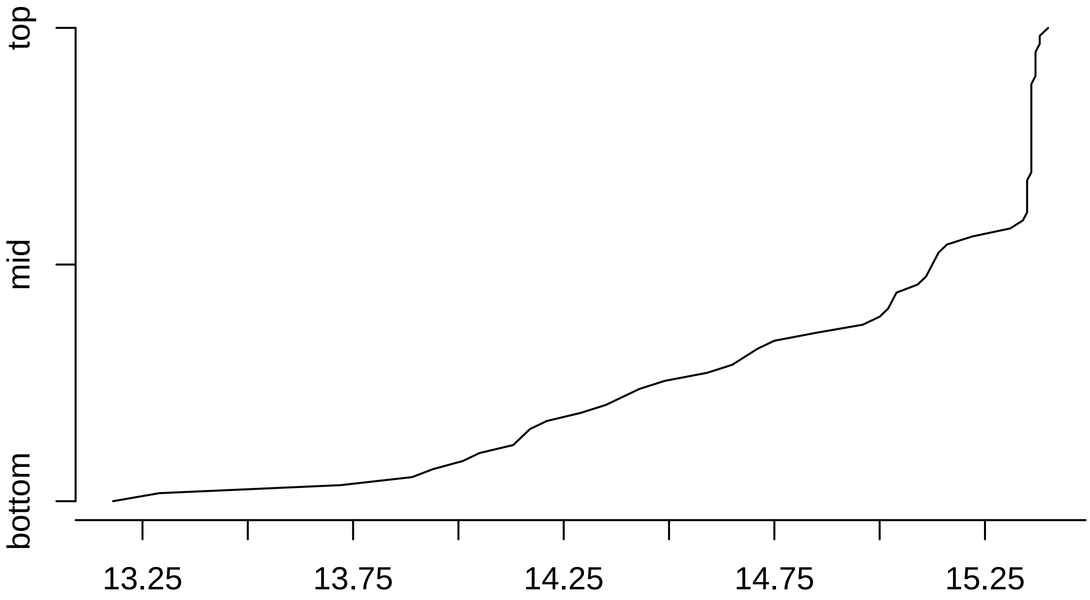
The axis ticks can be specified by setting the `at` argument of `axis`. Below, we set the x-axis ticks to be from the minimum integer to the maximum integer stepping by 0.25:

```
plot(
  st1$temp, st1$depth,
  type = "l", ylim = rev(range(st1$depth)),
  ann = FALSE, axes = FALSE
)
x.min <- floor(min(st1$temp))
x.max <- ceiling(max(st1$temp))
x.at <- seq(x.min, x.max, by = 0.25)
axis(1, at = x.at)
axis(2)
```



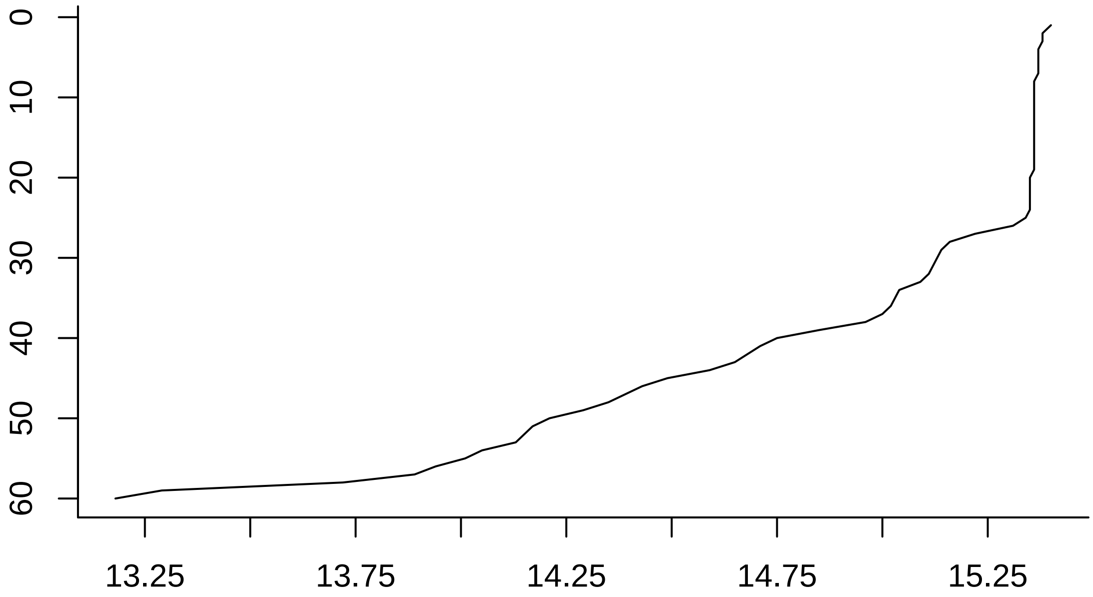
Axis labels are specified with the `label` argument:

```
plot(
  st1$temp, st1$depth,
  type = "l", ylim = rev(range(st1$depth)),
  ann = FALSE, axes = FALSE
)
x.min <- floor(min(st1$temp))
x.max <- ceiling(max(st1$temp))
x.at <- seq(x.min, x.max, by = 0.25)
axis(1, at = x.at)
axis(2, at = quantile(st1$depth, p = c(0, 0.5, 1)), labels = c("top", "mid", "bottom"))
```



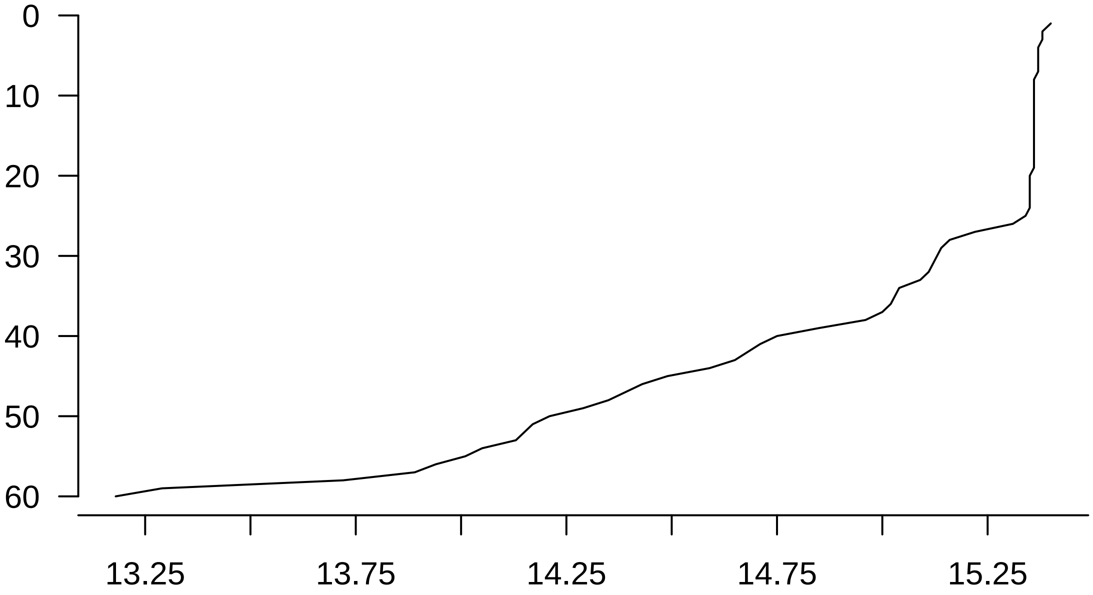
The plot can be surrounded by a box (`?box`). The `bty` argument (`?par`) will specify which sides are included in the box:

```
plot(
  st1$temp, st1$depth,
  type = "l", ylim = rev(range(st1$depth)),
  ann = FALSE, axes = FALSE
)
x.min <- floor(min(st1$temp))
x.max <- ceiling(max(st1$temp))
x.at <- seq(x.min, x.max, by = 0.25)
axis(1, at = x.at)
axis(2)
box(bty = "l")
```



The `las` argument will determine the orientation of the axis labels. To make them all horizontal, set `las = 1`:

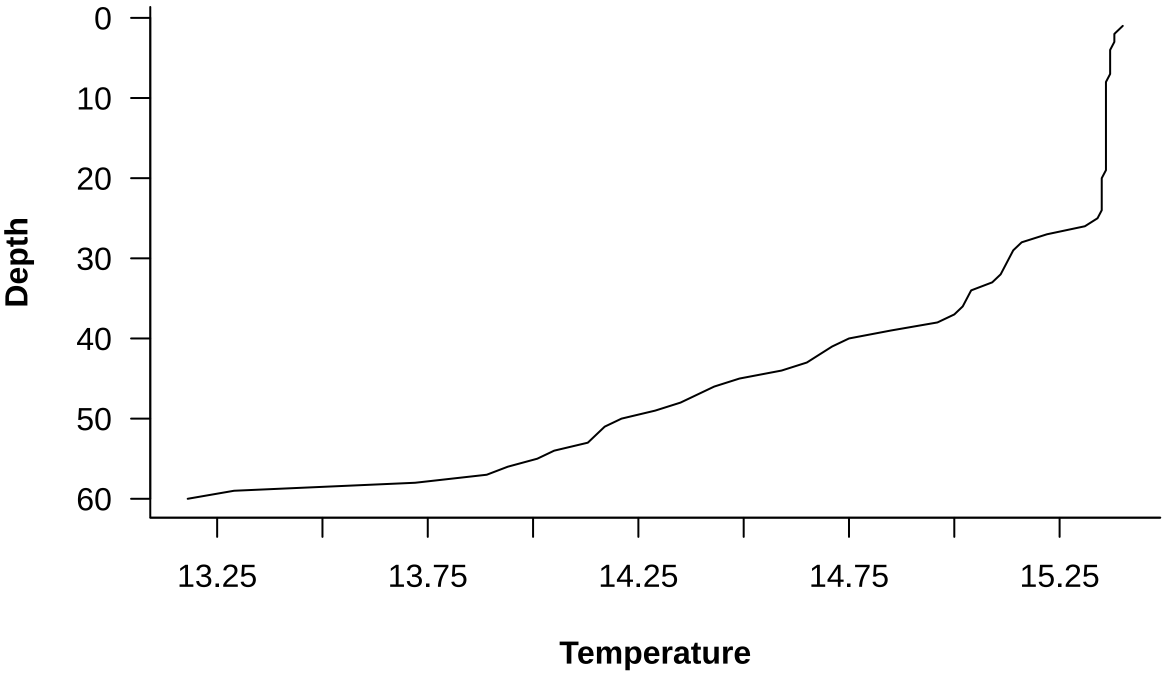
```
plot(
  st1$temp, st1$depth,
  type = "l", ylim = rev(range(st1$depth)),
  ann = FALSE, axes = FALSE
)
x.min <- floor(min(st1$temp))
x.max <- ceiling(max(st1$temp))
x.at <- seq(x.min, x.max, by = 0.25)
axis(1, at = x.at)
axis(2, las = 1)
```



Fonts

The font style is changed with `family`, and the type of font with `font`. The type is a number that specifies 1 = plain text, 2 = bold, 3 = italic, and 4 = bold & italic. The text that is to be modified follows `font` (e.g., `font.lab` for x and y axis labels, or `font.main` for main title text):

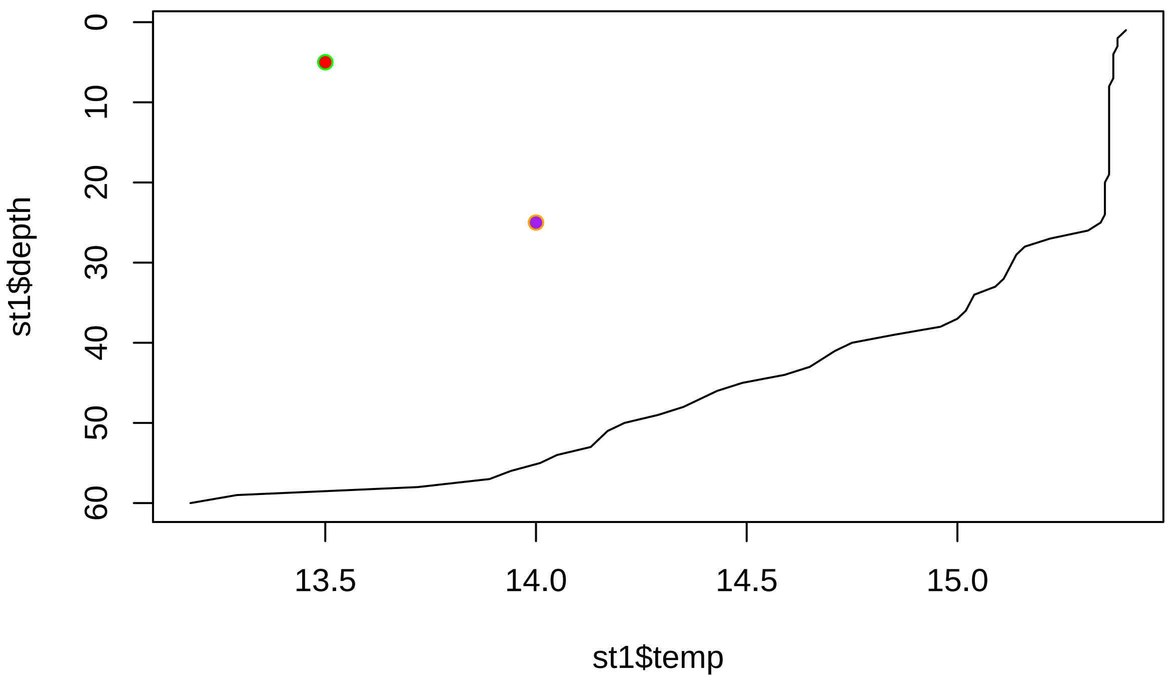
```r
plot(
  st1$temp, st1$depth,
  type = "l", ylim = rev(range(st1$depth)),
  xlab = "Temperature", ylab = "Depth",
  family = "Helvetica", font.lab = 2,
  axes = FALSE
)
x.min <- floor(min(st1$temp))
x.max <- ceiling(max(st1$temp))
x.at <- seq(x.min, x.max, by = 0.25)
axis(1, at = x.at)
axis(2, las = 1)
box(bty = "l")
```



Points

Points can be added to a plot with the `points` function. A vector of x and y values are specified along with vectors for point shapes and colors if desired:

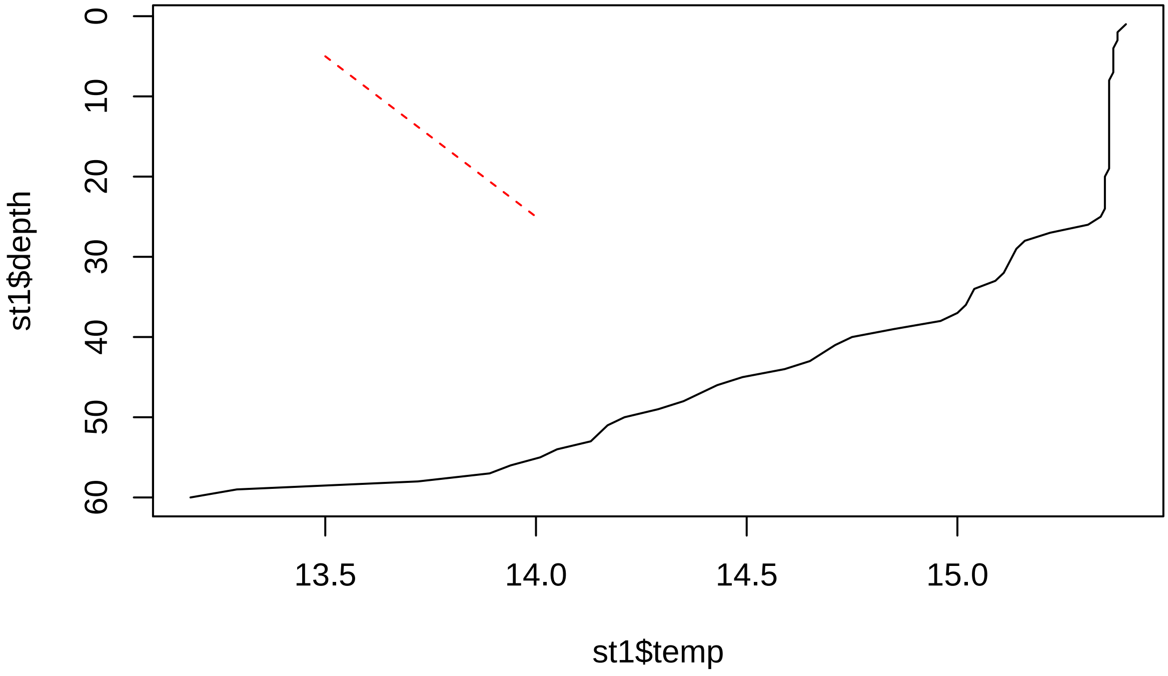
```r
plot(
  st1$temp, st1$depth,
  type = "l", ylim = rev(range(st1$depth))
)
points(c(13.5, 14), c(5, 25), pch = 21, bg = c("red", "purple"), col = c("green", "orange"))
```



Lines

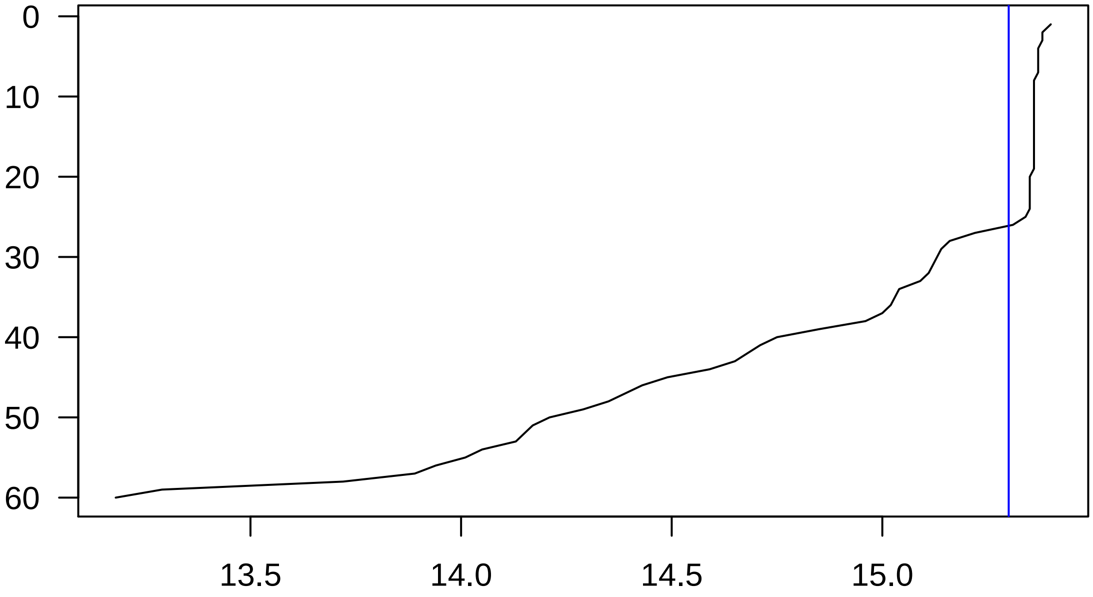
The same goes for lines to be added to a plot with the `lines` function.

```
plot(
  st1$temp, st1$depth,
  type = "l", ylim = rev(range(st1$depth))
)
lines(c(13.5, 14), c(5, 25), lty = "dashed", col = "red")
```



Horizontal and vertical lines, along with lines derived from a slope and intercept can be added with the `ablines` function:

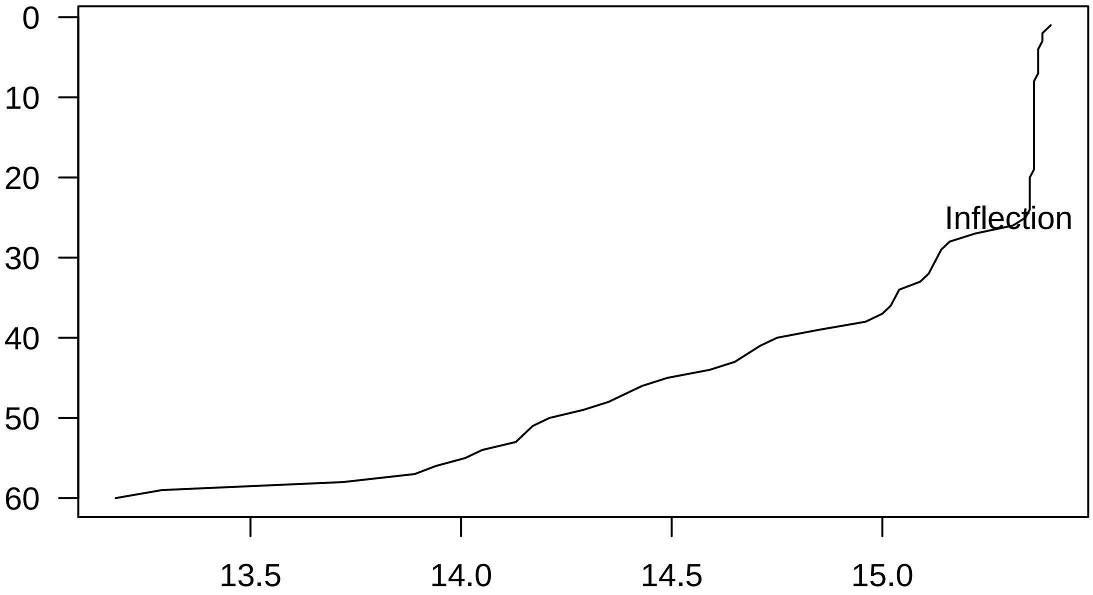
```
plot(
  st1$temp, st1$depth,
  type = "l", ylim = rev(range(st1$depth)),
  ann = FALSE, axes = FALSE
)
axis(1)
axis(2, las = 2)
box()
abline(v = 15.3, col = "blue")
```



Text

Text can also be placed in a plot similar to points and lines:

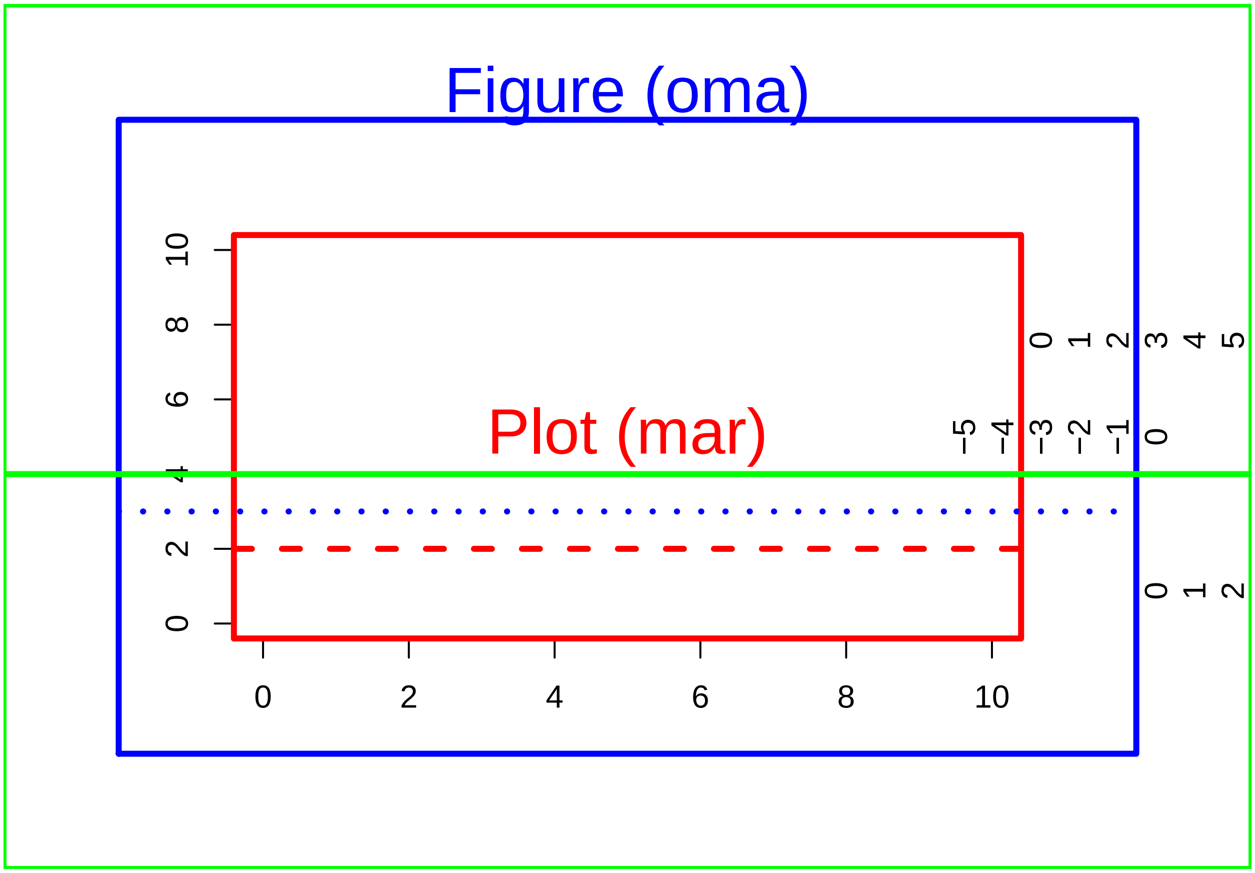
```
plot(
  st1$temp, st1$depth,
  type = "l", ylim = rev(range(st1$depth)),
  ann = FALSE, axes = FALSE
)
axis(1)
axis(2, las = 2)
box()
text(15.3, 25, "Inflection")
```



Margins

Margins are controlled with `par`. There are three regions of the plot (device, figure, plot) to be aware of and two margins that can be controlled. The plot margin (between plot and figure) is controlled by `mar` and the outer margin (between figure and device) by `oma`. Within each margin are a series of lines going from the section outwards or negative values go inwards. These lines are usually used to specify text where text goes with the `mtext` function. The vector for `mar` and `oma` is also based on the number of lines for each side. The `outer` argument for `mtext` controls whether or not the outer margins are used, and the `xpd` argument (defined in `?par`) determins if plotting is clipped to the plot, figure, or device region.

```r
op <- par(mar = c(3, 3, 3, 3), oma = c(3, 3, 3, 3))

plot(0:10, 0:10, type = "n", xlab = "X", ylab = "Y")

box("plot", lwd = 3, col = "red")
text(5, 5, "Plot (mar)", cex = 2, col = "red")

box("figure", lwd = 3, col = "blue")
mtext("Figure (oma)", side = 3, line = 3, cex = 2, col = "blue")

box("outer", lwd = 3, col = "green")

for(i in 0:5) mtext(i, 4, line = i, adj = 0.75 )
for(i in 0:3) mtext(i, 4, line = i, outer = TRUE, adj = 0.25)
for(i in -(5:0)) mtext(i, 4, line = i, outer = TRUE)

abline(h = 2, lty = "dashed", lwd = 3, col = "red", xpd = FALSE)
abline(h = 3, lty = "dotted", lwd = 3, col = "blue", xpd = TRUE)
abline(h = 4, lty = "solid", lwd = 3, col = "green", xpd = NA)
```

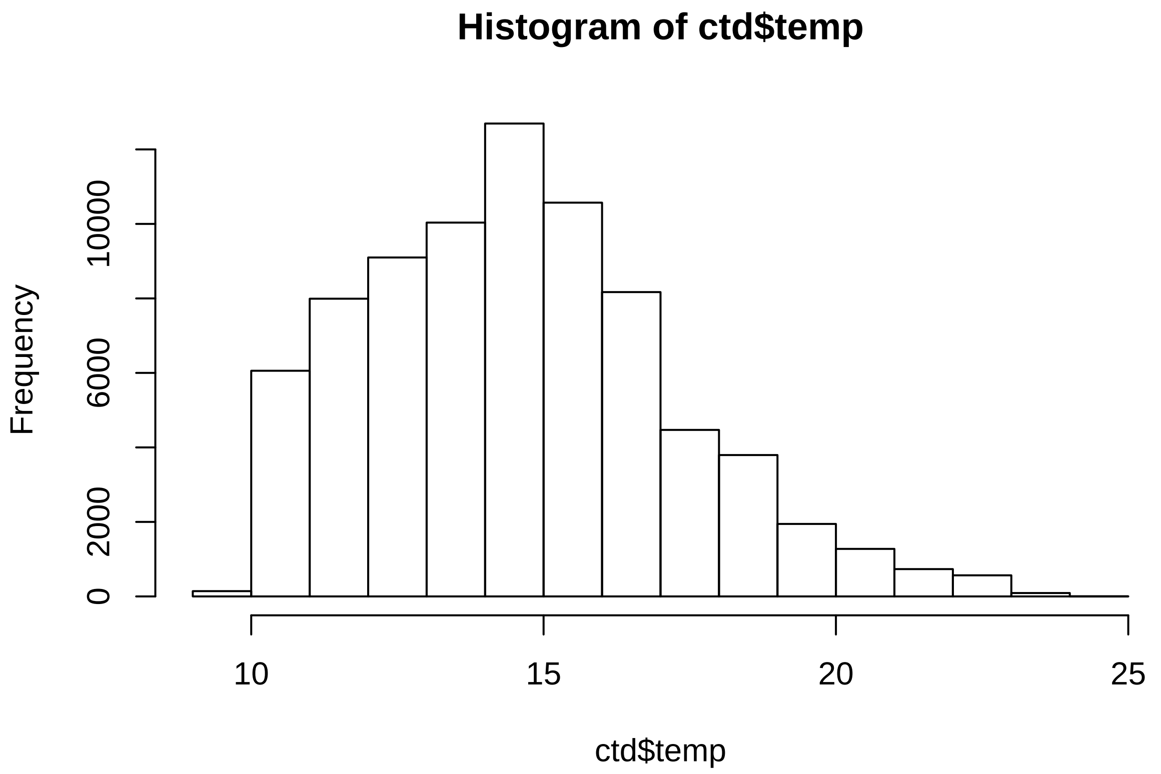


```
par(op)
```

Histograms

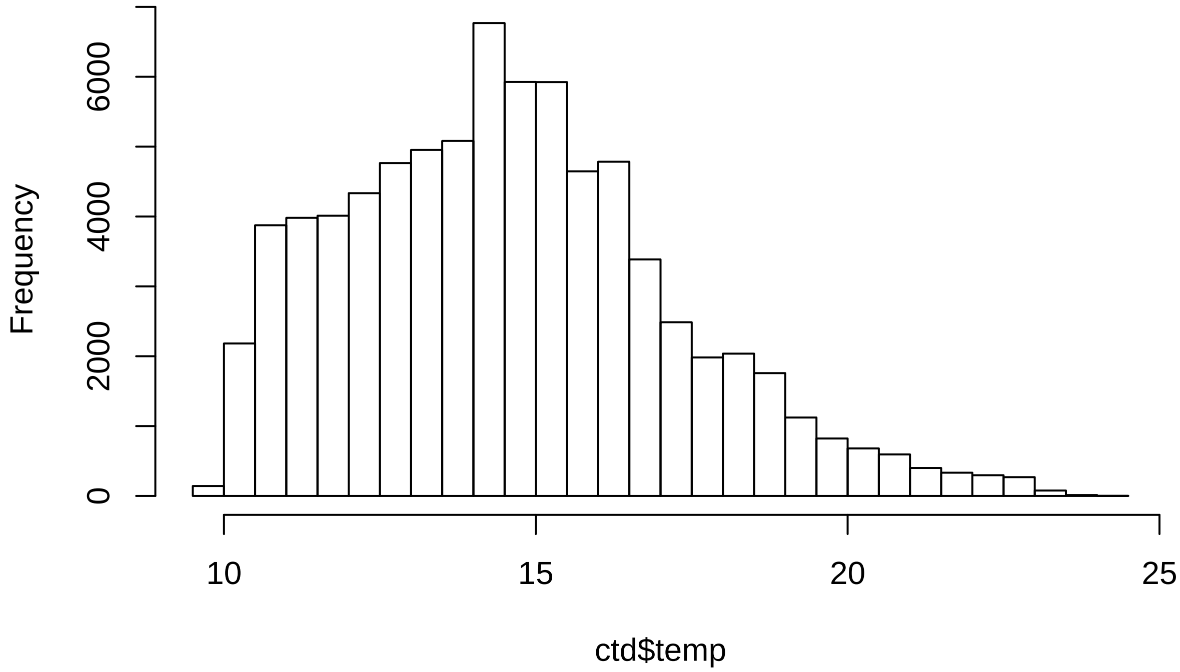
To plot frequencies of binned continuous variables, create a histogram with the `hist` function.

```
hist(ctd$temp)
```



The `breaks` argument determines how the binning is done. If it is a single number, then the data is split into that many bins:

```
hist(ctd$temp, breaks = 30, main = "")
```

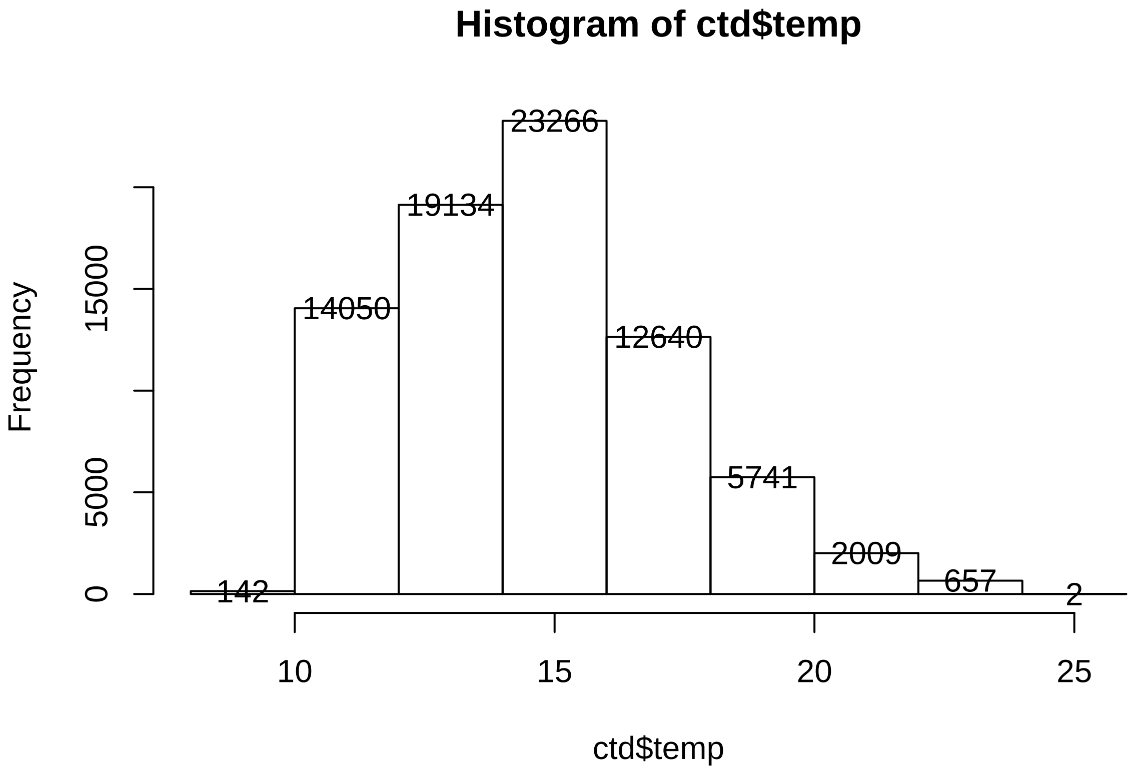


. . . or the actual breaks can be given. Also, if the result of `hist` is assigned to an object, information about the binning is stored in that object and can be used to annotate it:

```
hist.vals <- hist(ctd$temp, breaks = seq(8, 26, by = 2))
str(hist.vals)
```

```
List of 6
 $ breaks  : num [1:10] 8 10 12 14 16 18 20 22 24 26
 $ counts  : int [1:9] 142 14050 19134 23266 12640 5741 2009 657 2
 $ density : num [1:9] 0.000914 0.090481 0.123221 0.149831 0.0814 ...
 $ mids    : num [1:9] 9 11 13 15 17 19 21 23 25
 $ xname   : chr "ctd$temp"
 $ equidist: logi TRUE
 - attr(*, "class")= chr "histogram"
```

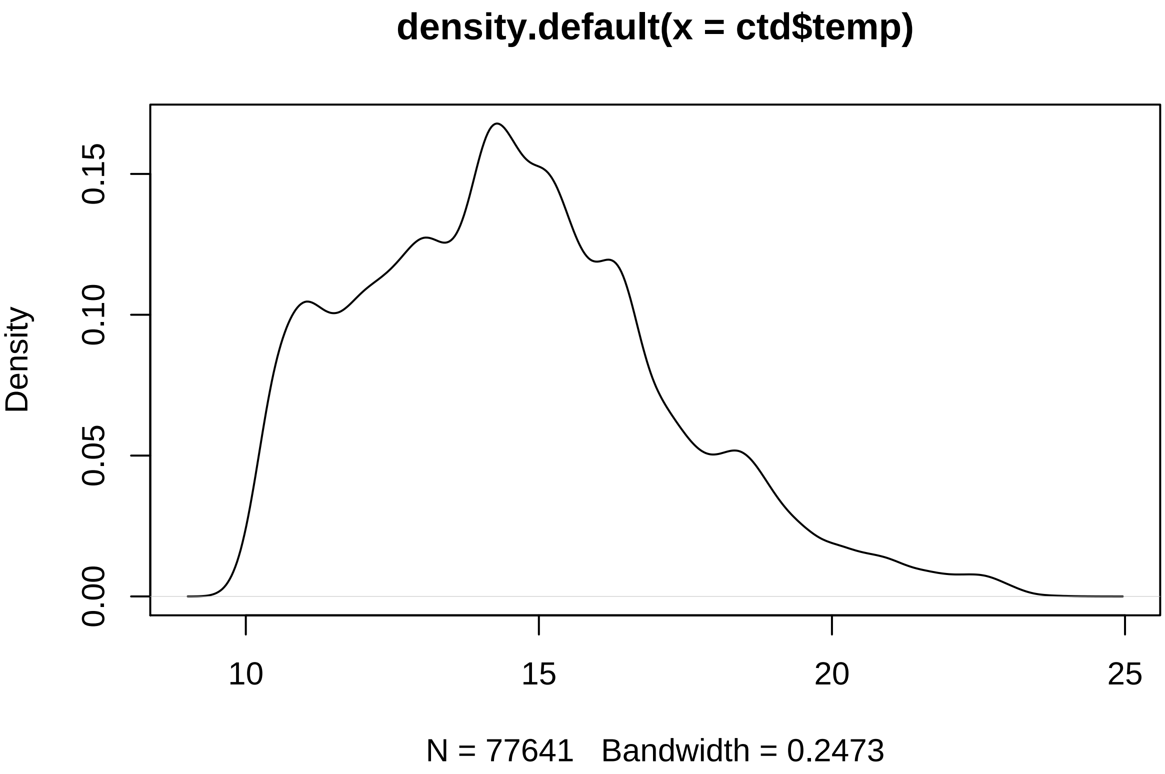
```
text(hist.vals$mids, hist.vals$counts, hist.vals$counts)
```



Density plots

A smoothed version of the histogram is the density plot. Here, you plot the result of a call to `density`:

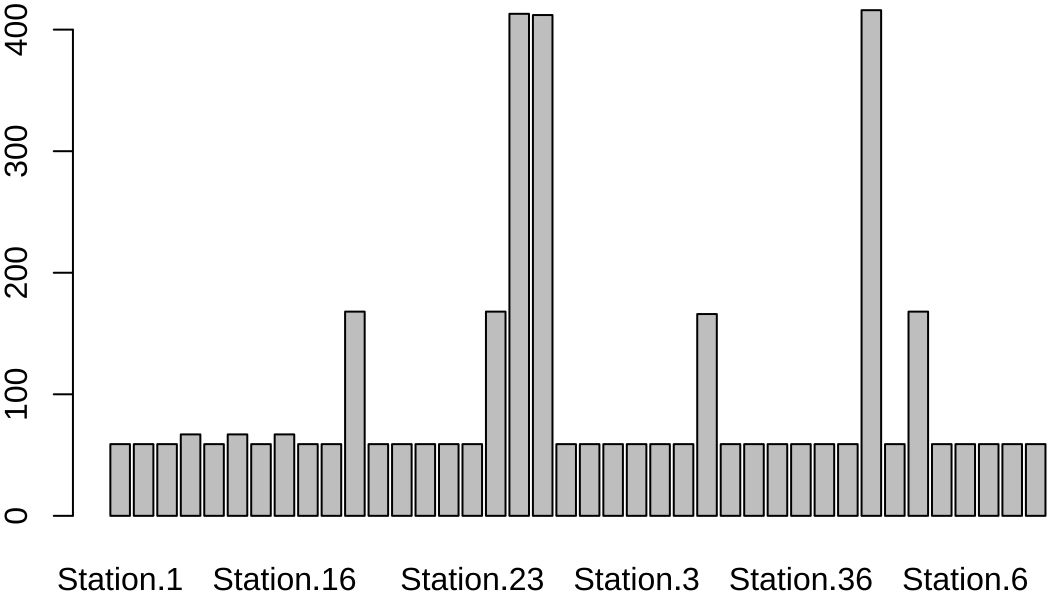
```
plot(density(ctd$temp))
```



Barplots

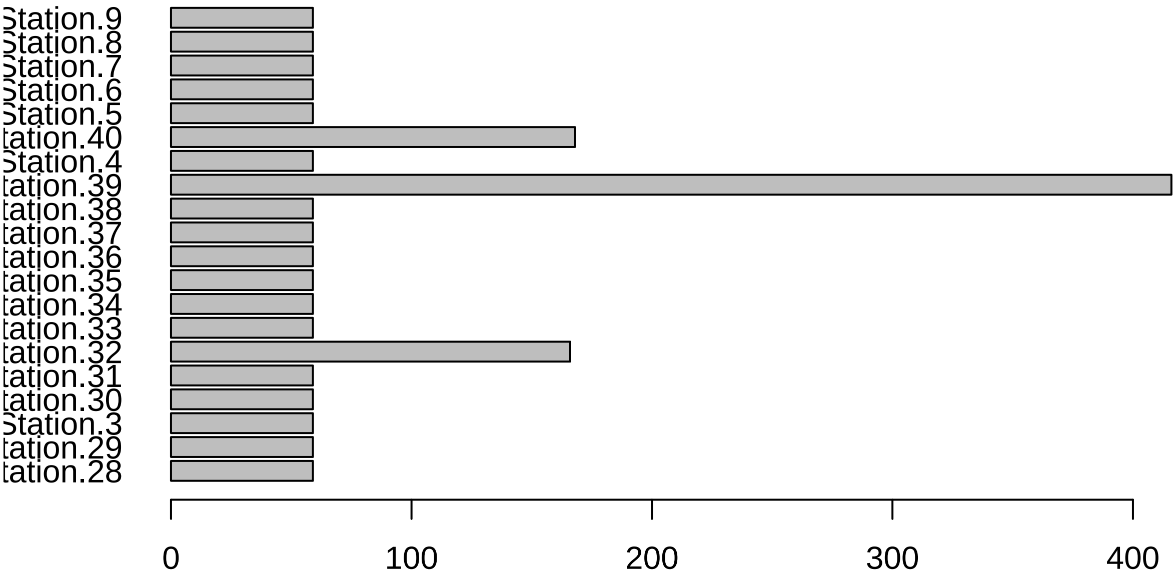
To plot a set values from a vector, like precalculated frequencies, use `barplot`:

```
freq <- table(ctd$station, ctd$sample_date)
num.casts <- apply(freq, 1, function(x) sum(x > 0))
barplot(num.casts)
```



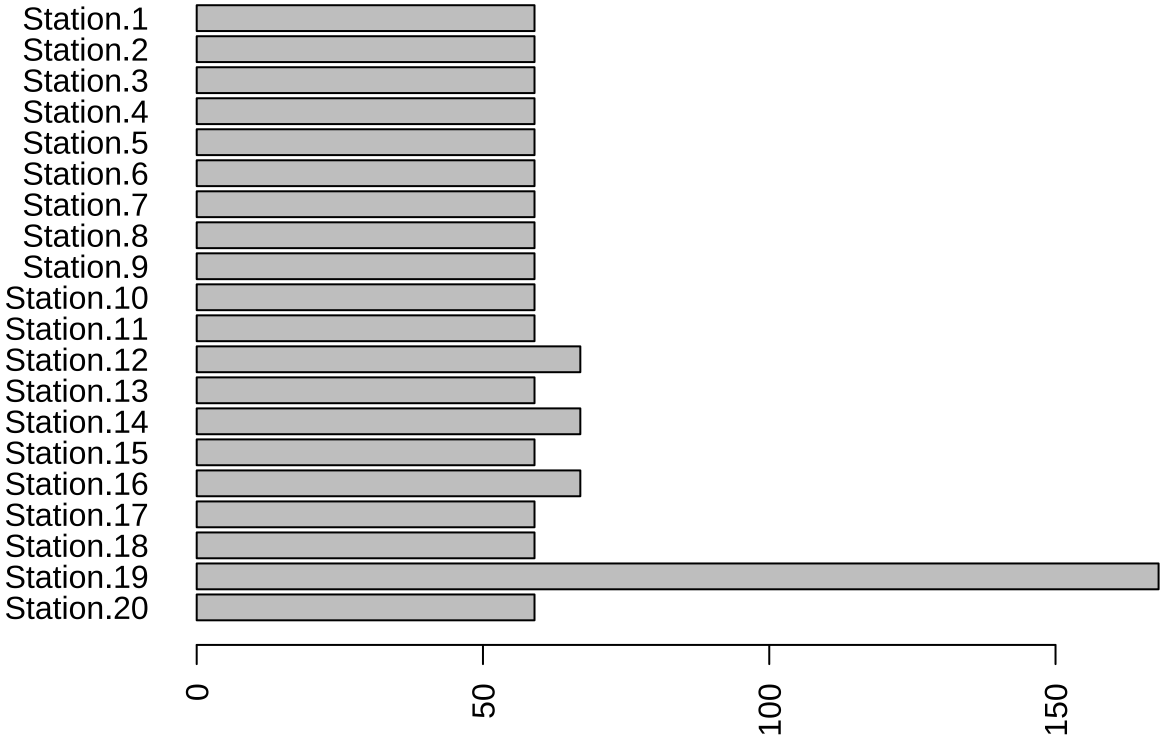
Barplots can also be plotted horizontally by setting `horiz = TRUE`:

```
barplot(num.casts[21:40], horiz = T, las = 1)
```



Because the labels are long, we should expand the left side margin:

```
st <- names(num.casts)
num.casts <- num.casts[order(nchar(st), st, decreasing = TRUE)]
op <- par(mar = c(4, 6, 1, 1) + 0.1)
barplot(num.casts[21:40], horiz = T, las = 2)
```

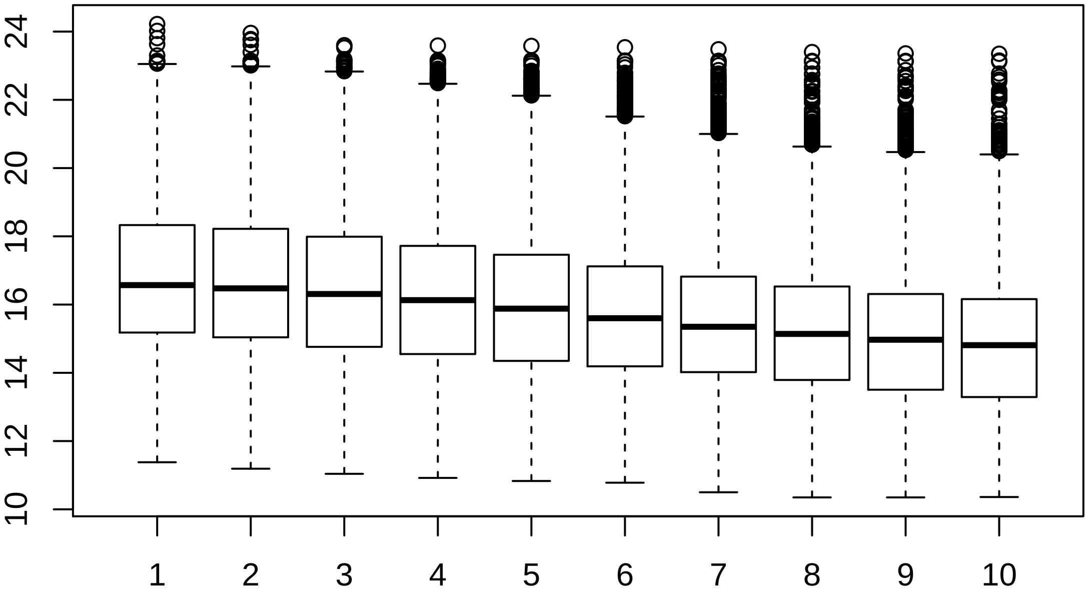


```
par(op)
```

Boxplots

To summarize distributions of continuous variables by some grouping factor, use a boxplot, which shows medians, quartiles, and outliers. The most common form uses the `formula` interfaces which is expressed as `y ~ x`. Here we plot the distribution of temperature for the top 10 meters:

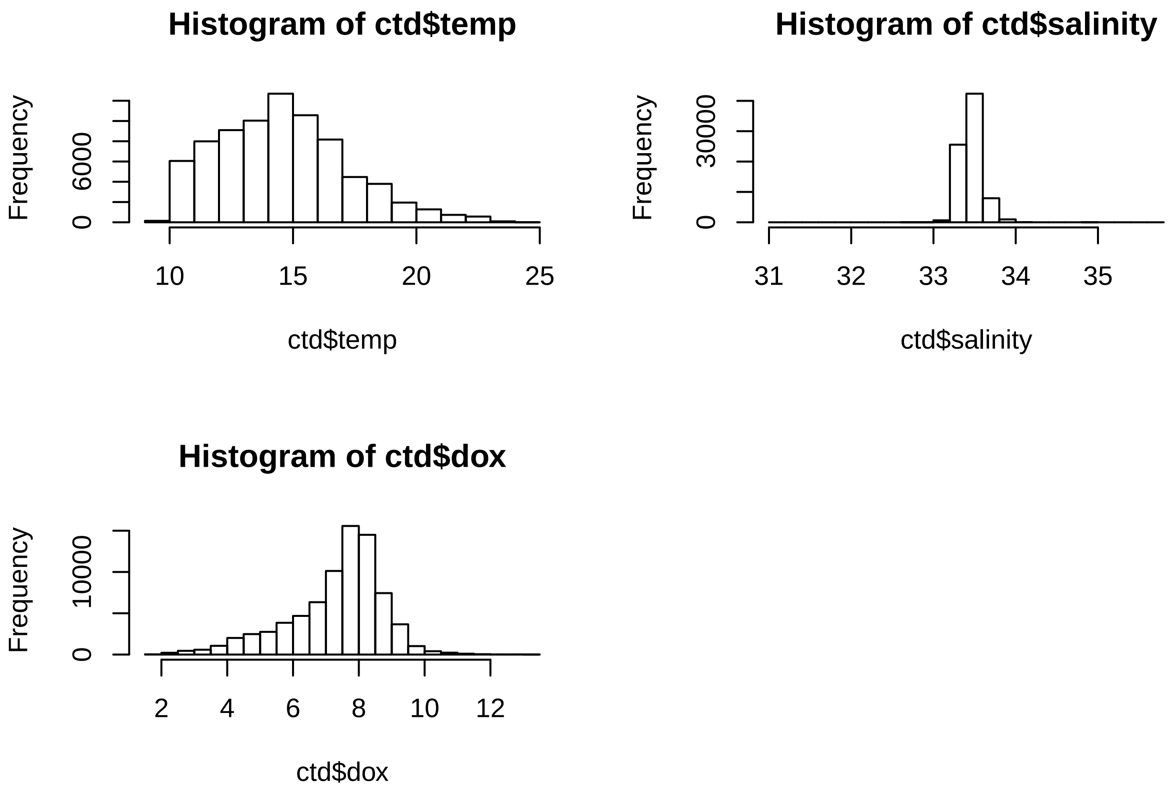
```
top.10 <- subset(ctd, depth <= 10)
boxplot(temp ~ depth, top.10)
```



Multple panels: mfrow/mfcol

One common way to create multiple panels is to specify `mfrow` or `mfcol` as a `par` parameter. The vector for either of these arguments specifies the number of rows and columns. `mfrow` will lay out the plots by row, while `mfcol` lays them out by column.

```
op <- par(mfrow = c(2, 2))
hist(ctd$temp)
hist(ctd$salinity)
hist(ctd$dox)
par(op)
```



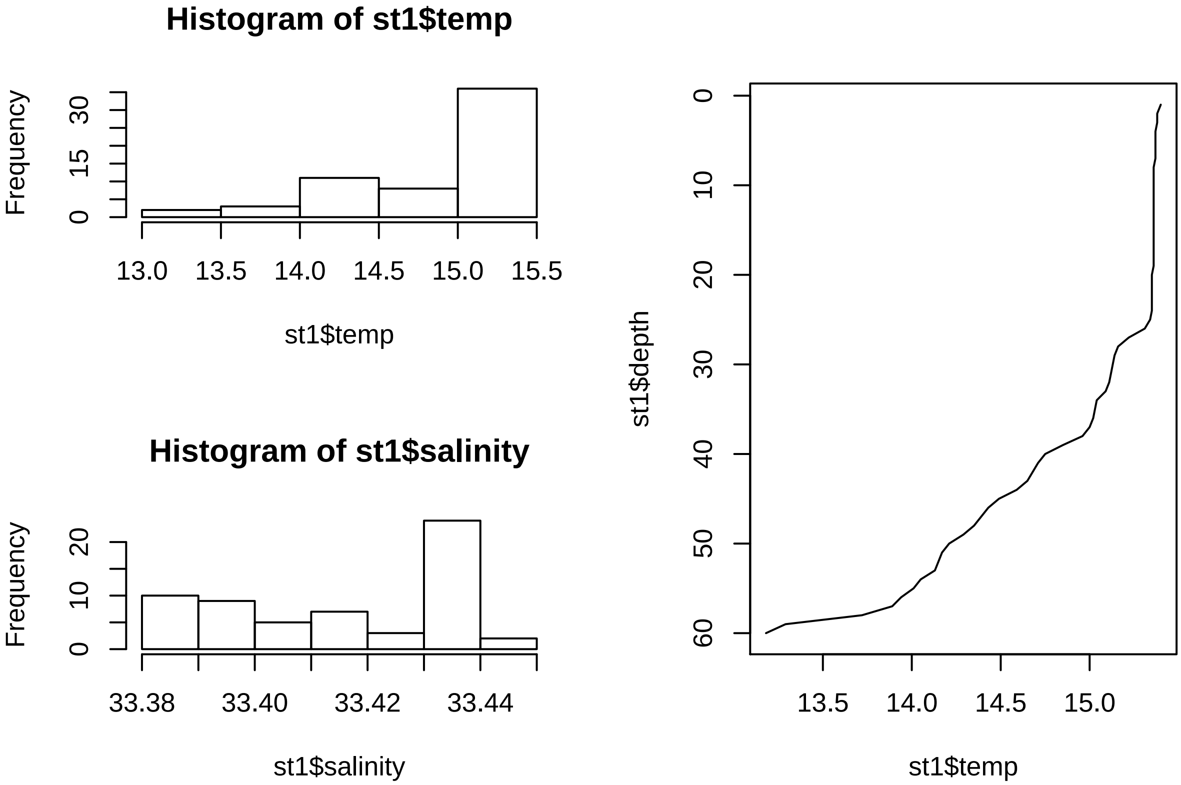
Multiple panels: layout

Another way is to use `layout` which requires mapping specified through a matrix. The values in the matrix correspond to the locations of the plots on the page:

```r
lm <- matrix(c(1, 2, 3, 3), nrow = 2)
lm
```

```
     [,1] [,2]
[1,]    1    3
[2,]    2    3
```

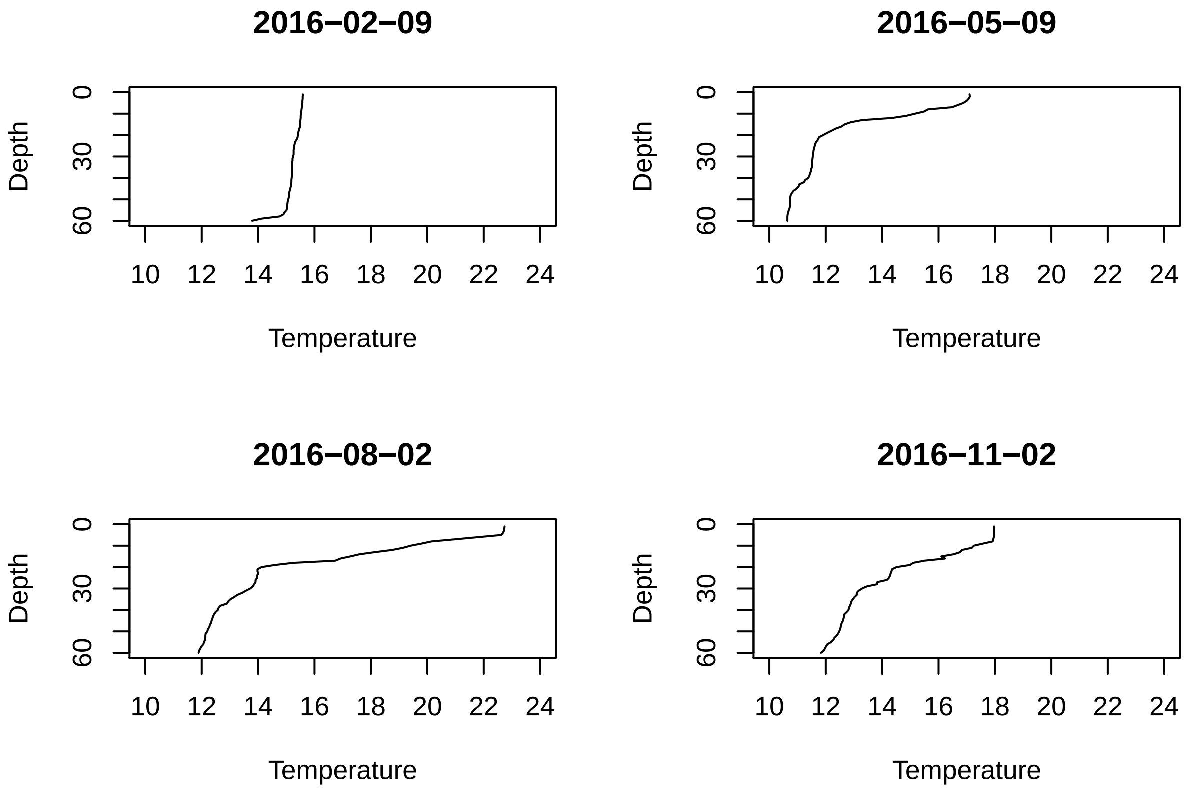
```r
layout(lm)
hist(st1$temp)
hist(st1$salinity)
plot(
  st1$temp, st1$depth,
  type = "l", ylim = rev(range(st1$depth))
)
```



```
layout(matrix(1))
```

Here, we use layout and a `for` loop and `layout` to temperature/depth traces for 4 casts from Station 1 in 2016:

```
# extract year to a column in the ctd data frame
ctd$year <- as.numeric(substr(ctd$sample_date, 1, 4))
# select all data for station 1 in 2016
st1.2016 <- subset(ctd, station == "Station.1" & year == 2016, drop = TRUE)
# set the x and y axis limits so they'll be consistent across panels
xlim <- range(pretty(st1.2016$temp))
ylim <- rev(range(pretty(st1.2016$depth)))
# get the dates of unique casts
casts <- sort(unique(st1.2016$sample_date))
# set a 2x2 matrix
layout(matrix(1:4, nrow = 2, byrow = TRUE))
# loop through the casts
for(dt in casts) {
  # subset the data for each cast
  cast.df <- subset(st1.2016, sample_date == dt)
  # sort by depth
  cast.df <- cast.df[order(cast.df$depth), ]
  # plot the temperature/depth trace
  plot(
    cast.df$temp, cast.df$depth, type = "l",
    xlim = xlim, ylim = ylim,
    xlab = "Temperature", ylab = "Depth", main = dt
  )
}
```



```r
# reset the layout to a single plot
layout(matrix(1))
```