Introduction to Programming with R - June 22

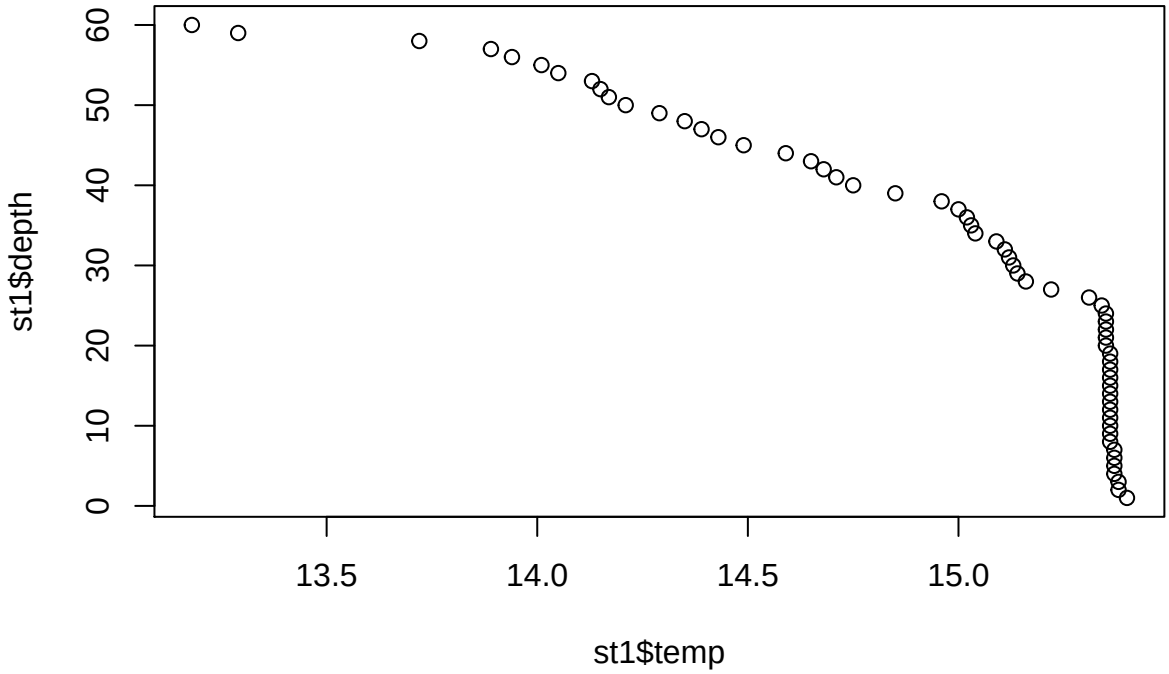
Eric Archer (eric.archer@noaa.gov, 858-546-7121)

base Graphics

Scatterplots

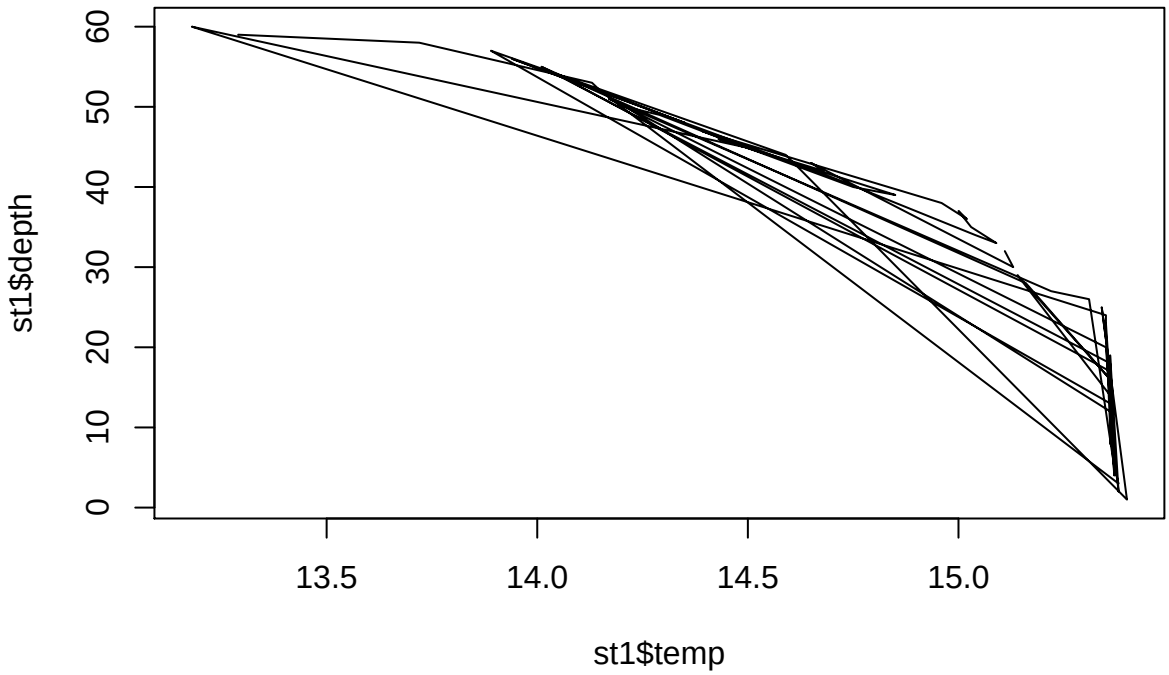
The most basic function for generating scatter and line plots is the function `plot`. The help for plot (`?plot`) is not that informative. You'll find more options with `?plot.default`. At its simplest it requires a vector of x values and a vector of y values. As an example, we'll plot the points representing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

```
# 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)
```



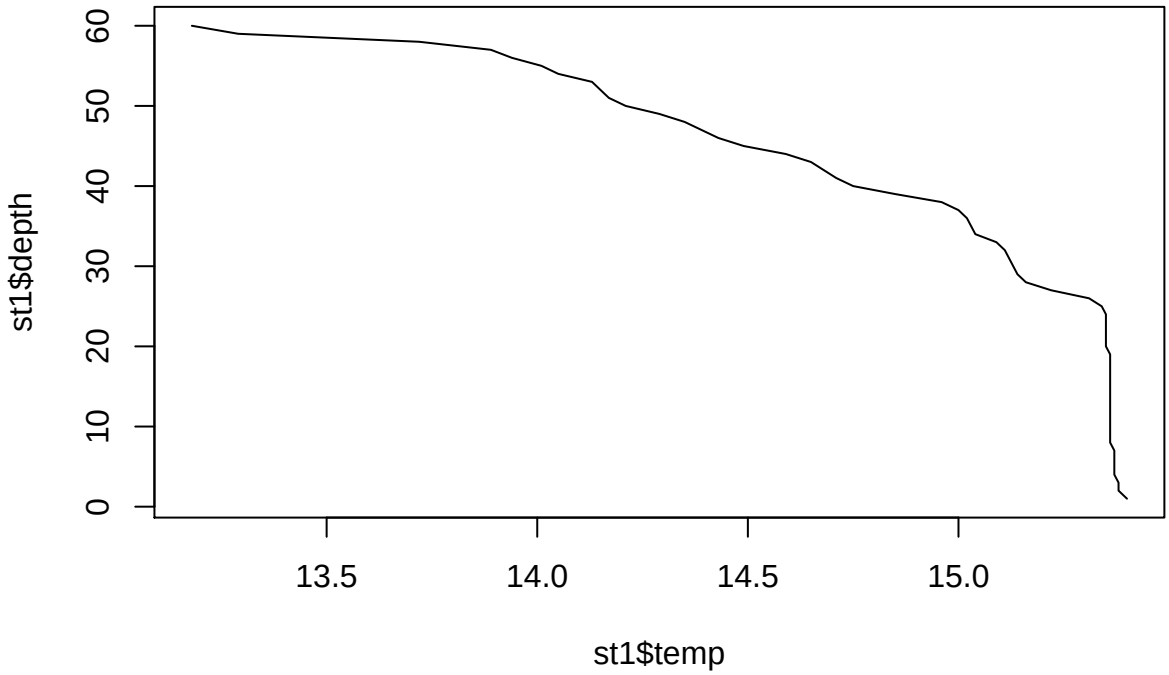
Changing the `type` argument selects (p)oints, (l)ines, or (b)oth.

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



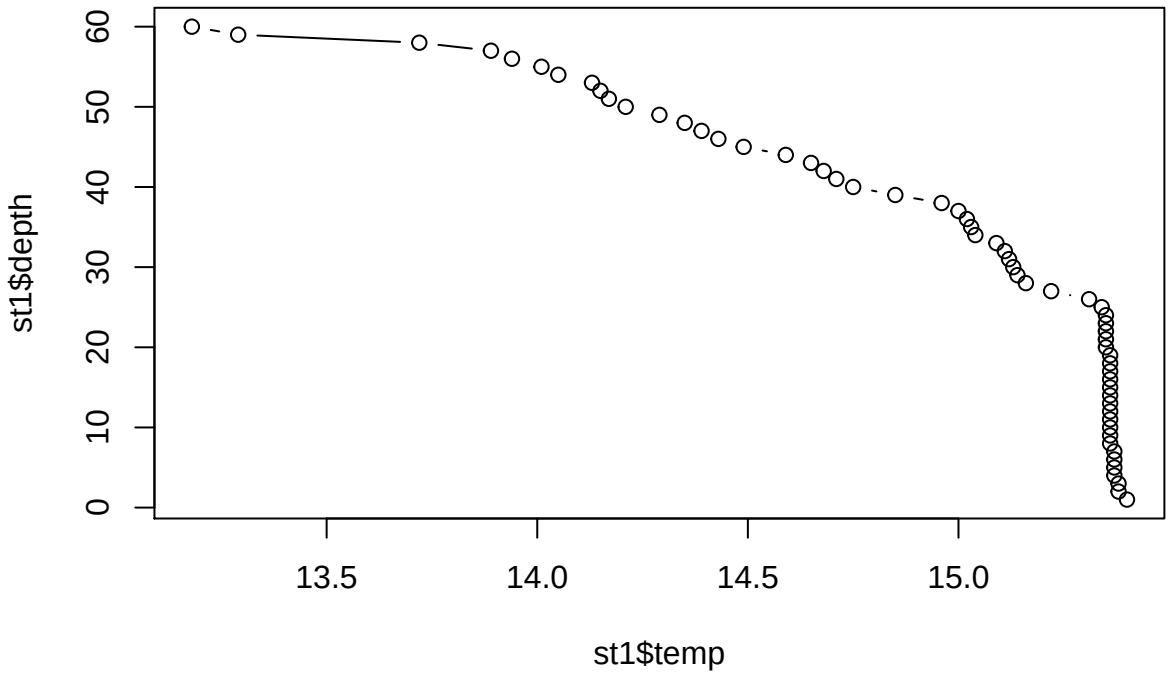
But if you plot lines, they are connectd in the order they occur in the vectors. In order to produce the trace properly, we have to sort the data frame by depth:

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



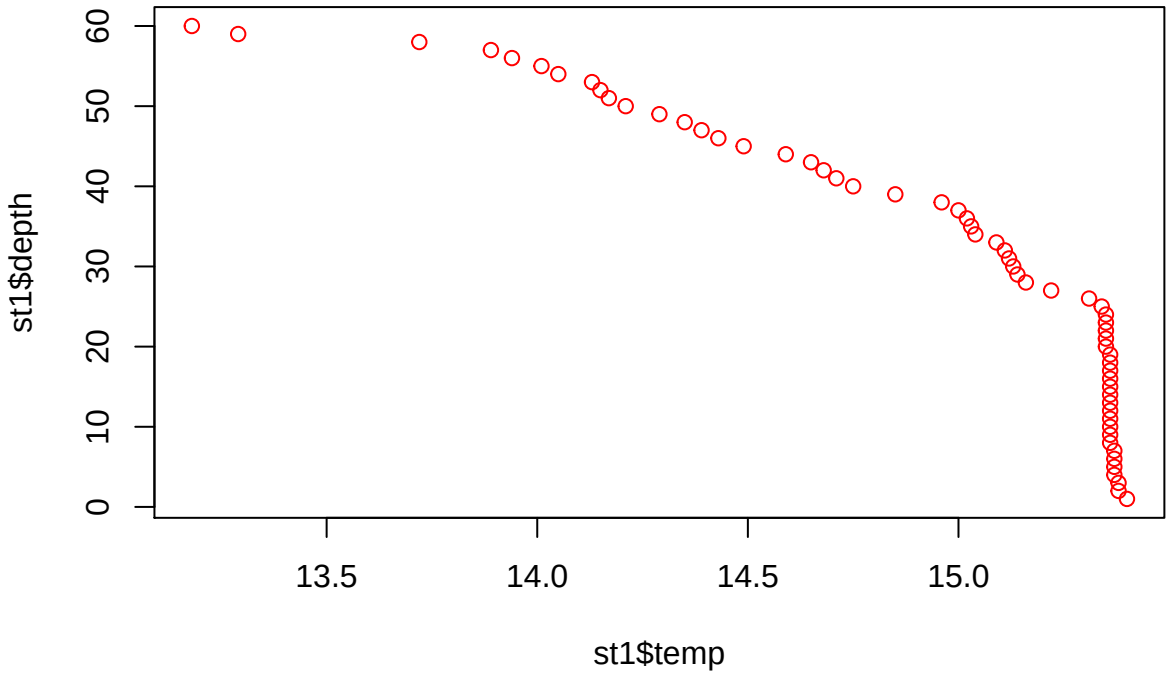
. . . and here is the same plot with both lines and points:

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



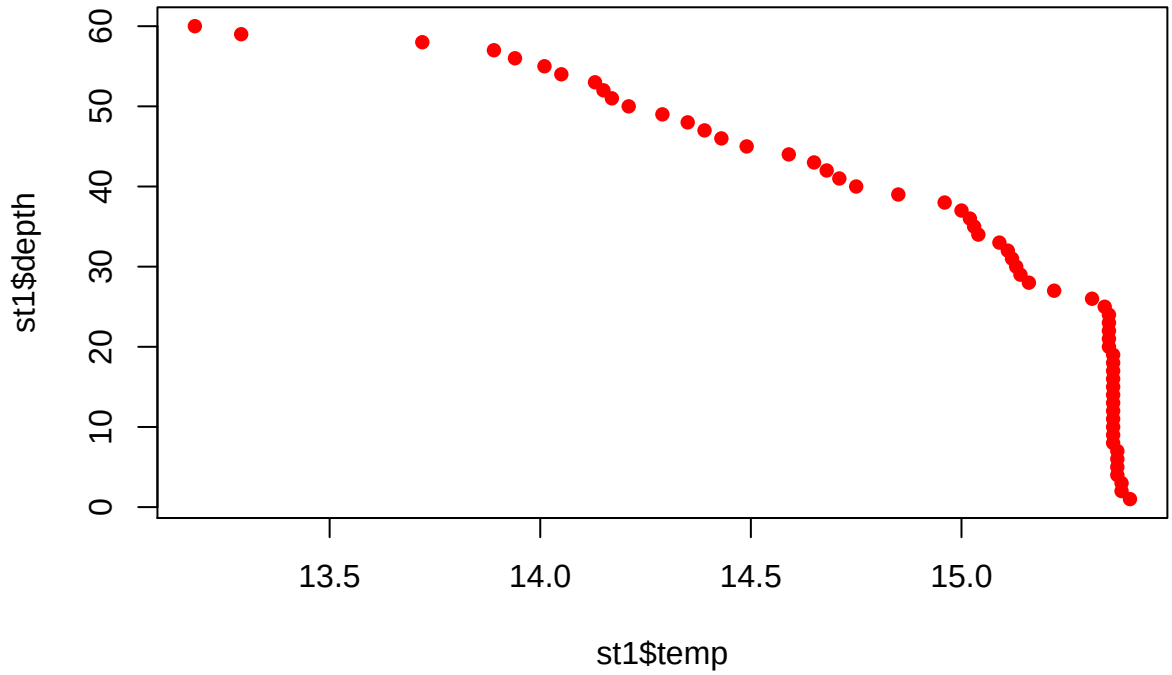
Colors can be changed using the `col` argument:

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



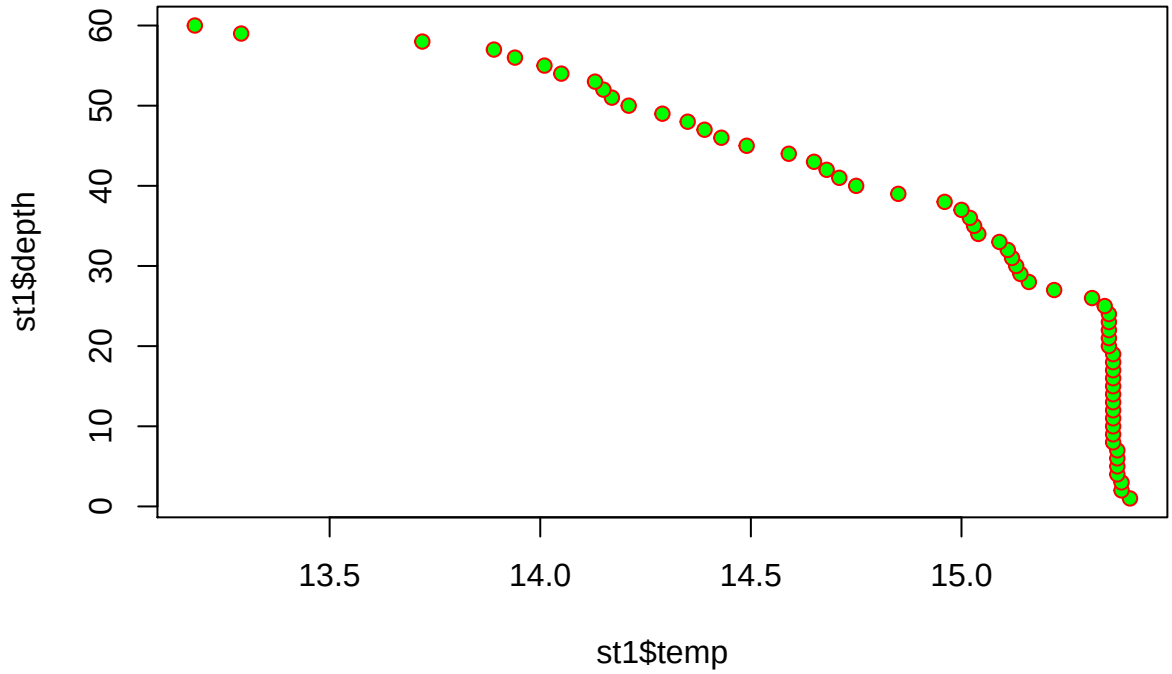
However, depending on the type of point, chosen, the `col` can refer to the outline color or the center color. The point type is chosen by assigning the `pch` argument with a number from 0 to 25. The coresponding shapes are given in `?points`. `pch = 16` produces a solid circle:

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



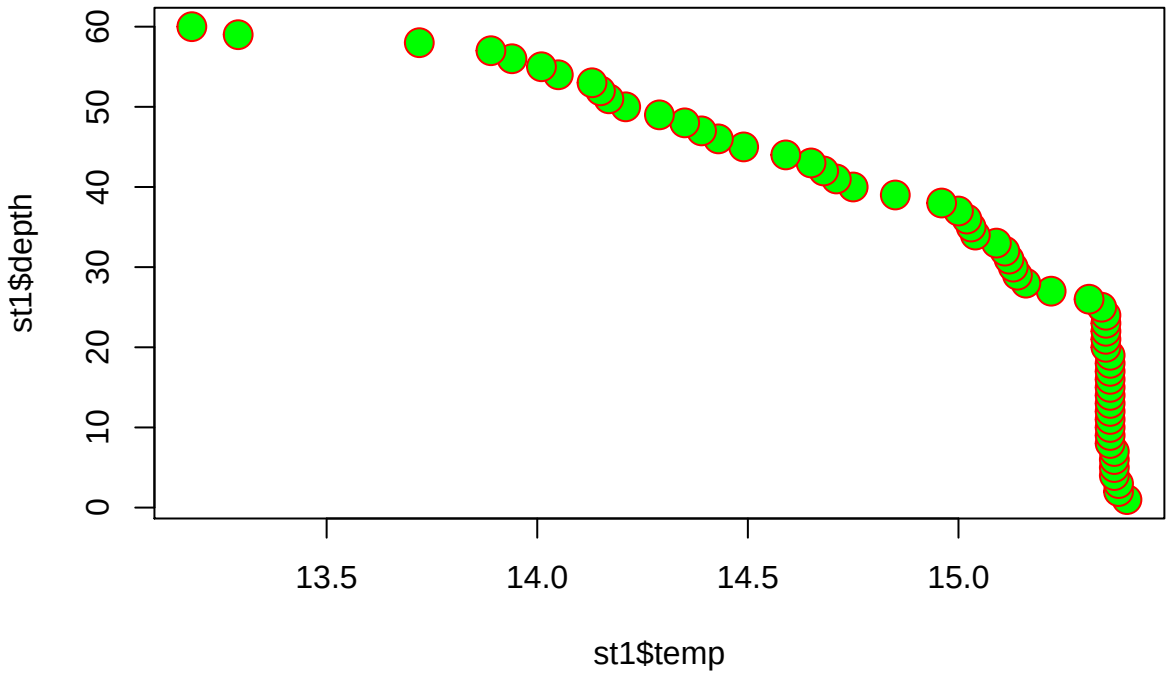
`pch = 21` produces a filled circle. In this case, `col` sets the outer color and `bg` sets the inner color:

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



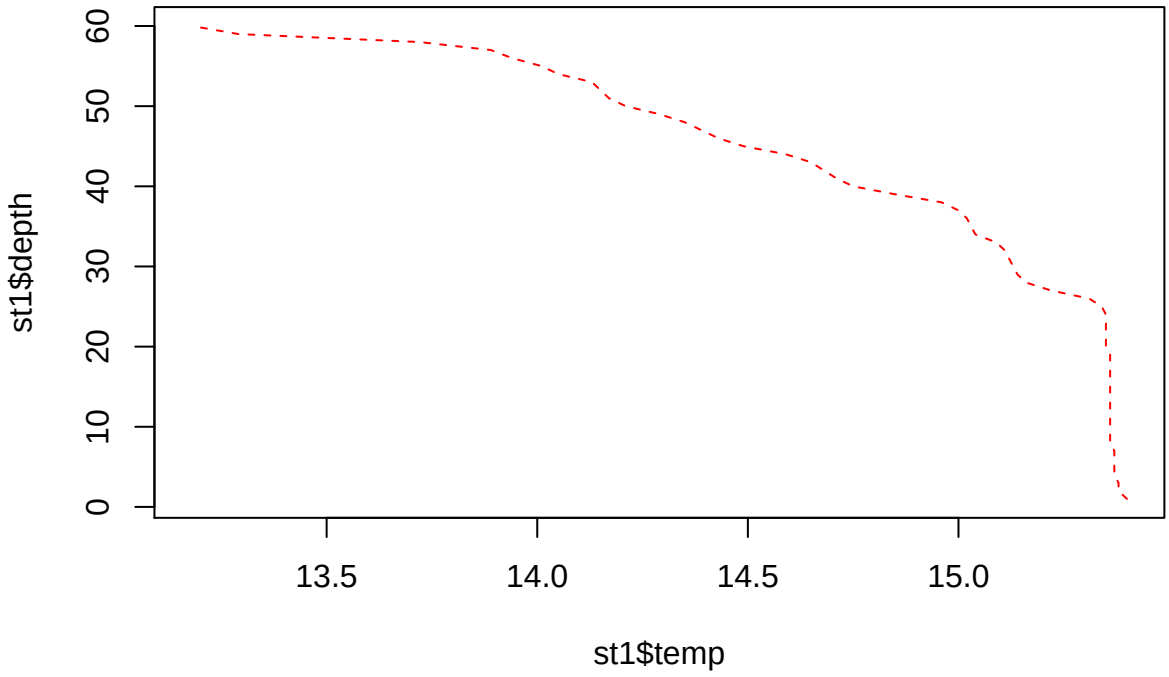
Point size is changed by `cex`. More information about `cex` and many other graphical parameters can be found in `?par`.

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



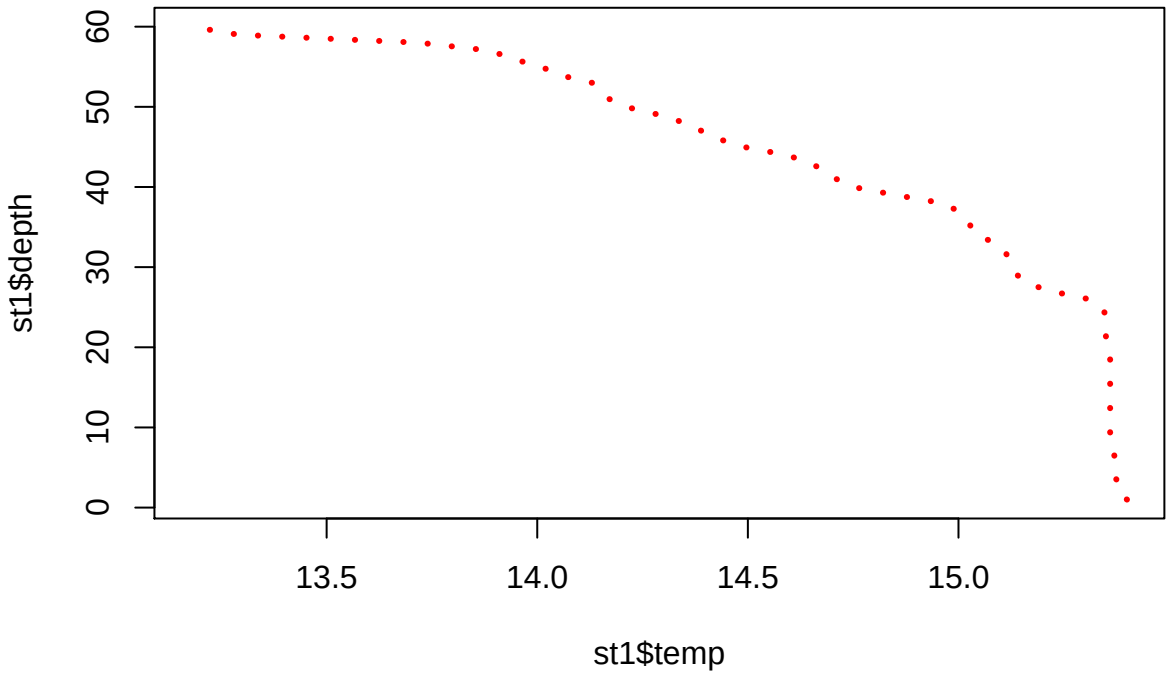
There are multiple line types as well, which can be specified with the graphical parameter `lty`.

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



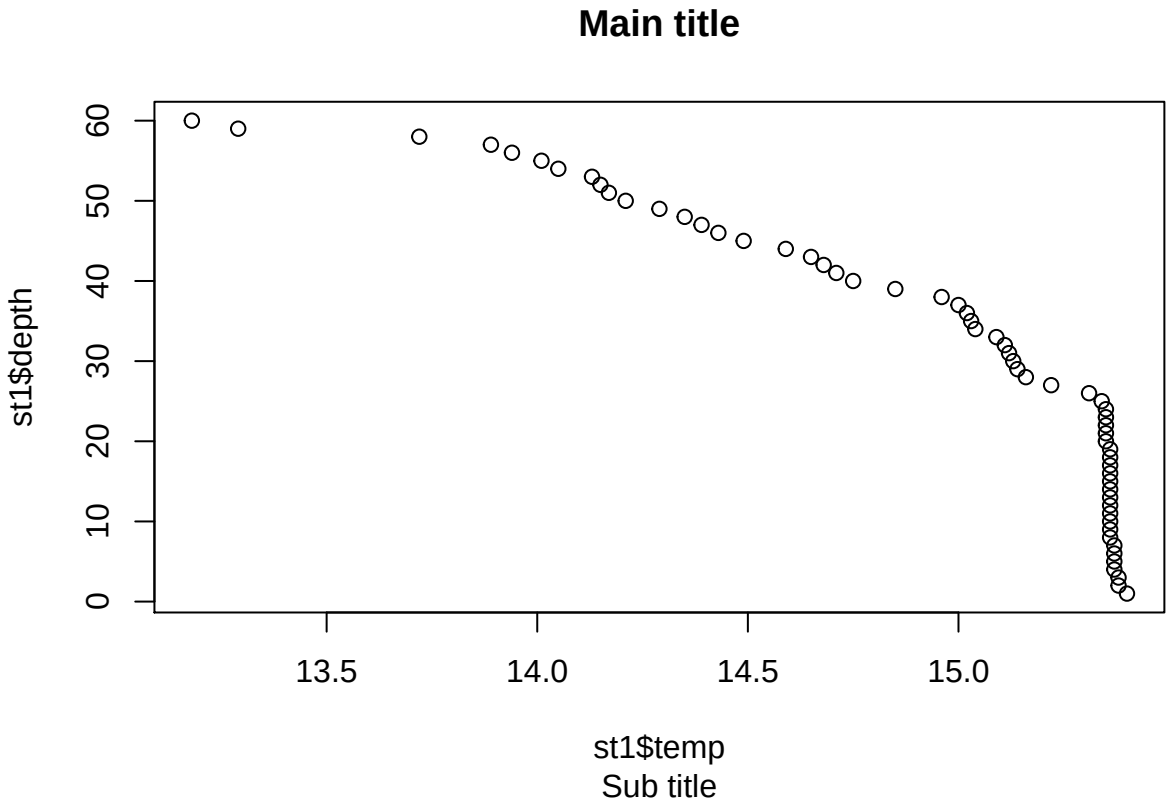
Line width is controlled with `lwd`:

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



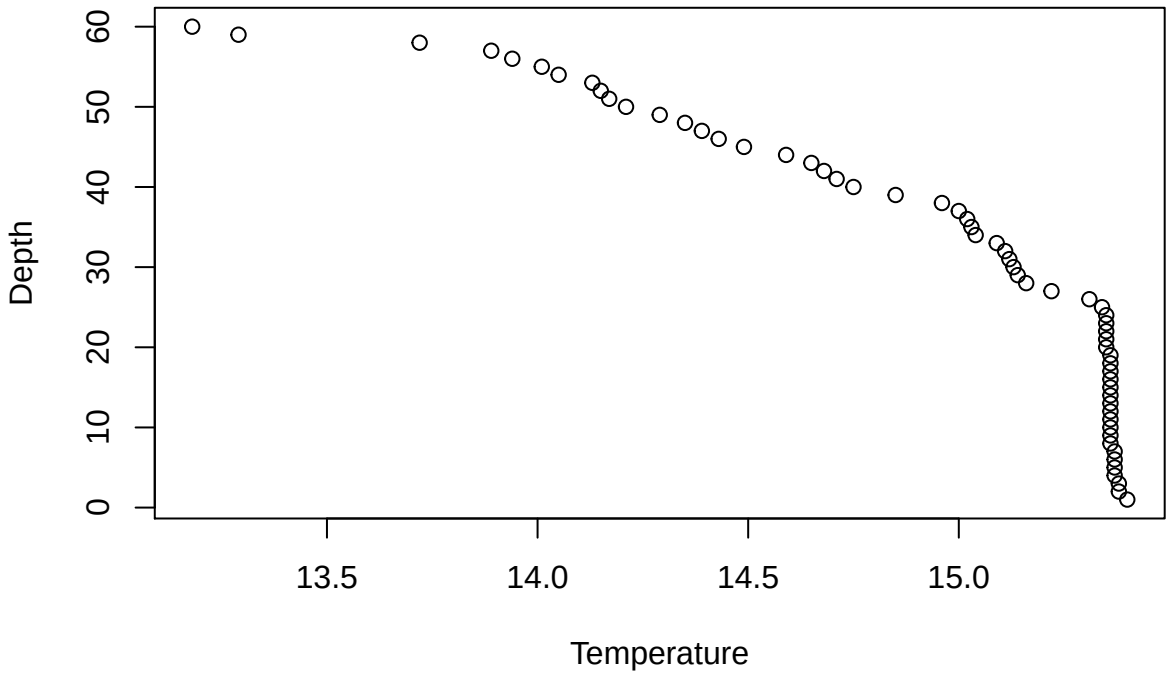
The `main` and `sub` arguments to plot allow you to specify main- and subtitles:

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



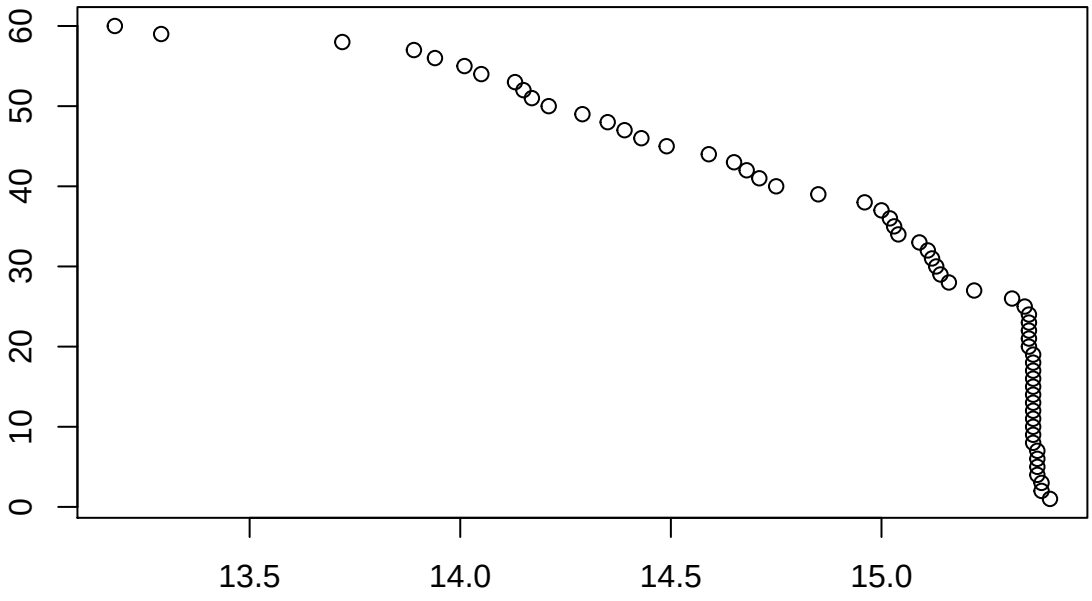
Labels for the axes are chosen by default, but can also be changed with the `xlab` and `ylab` arguments:

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



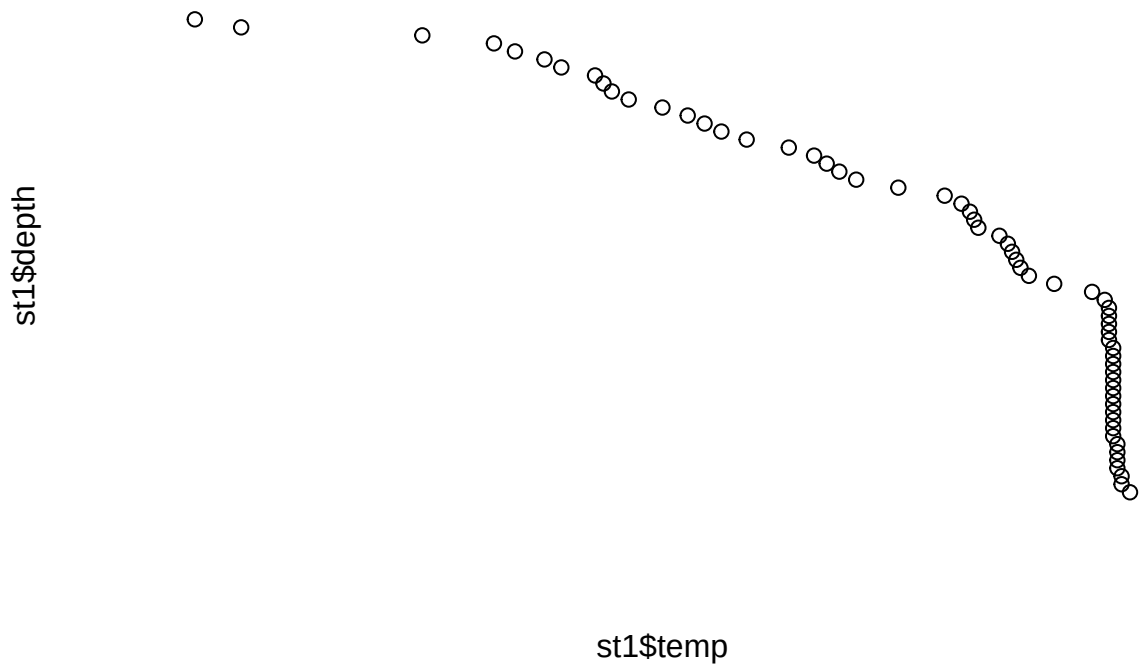
The default annotation (title and axis labels) can be turned off by setting `ann = FALSE`.

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



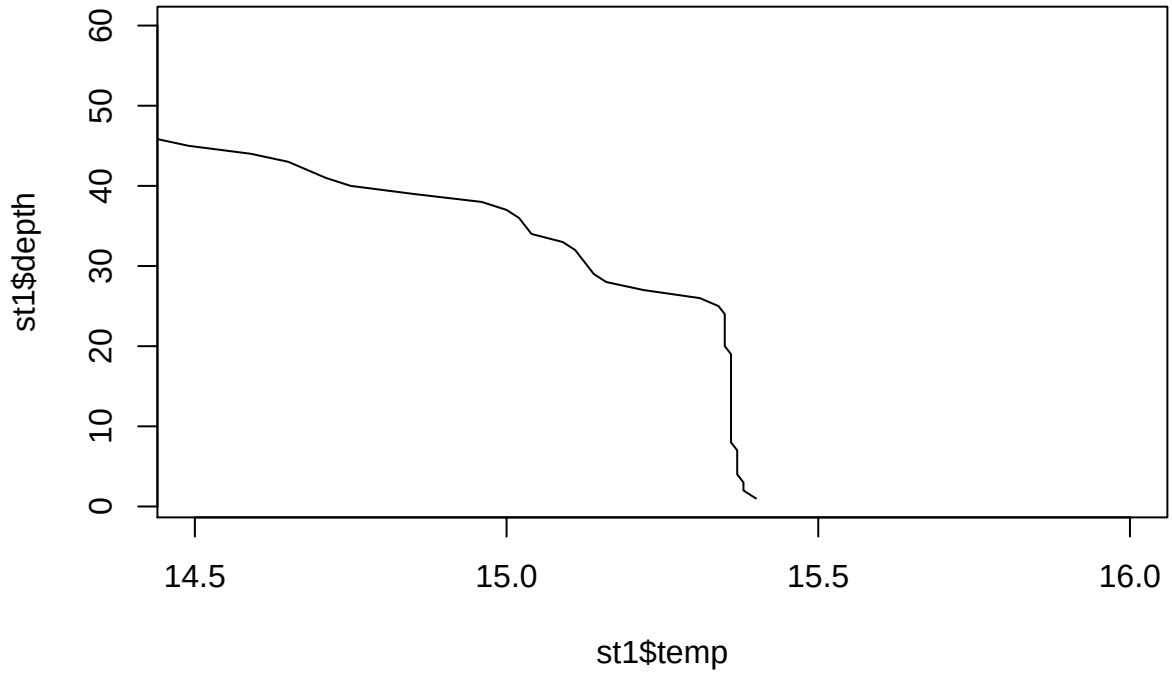
Likewise, the axes can be turned off with `axes = FALSE`. You may want to do this if you wish to customize axis or label placement or style, which we'll do below.

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



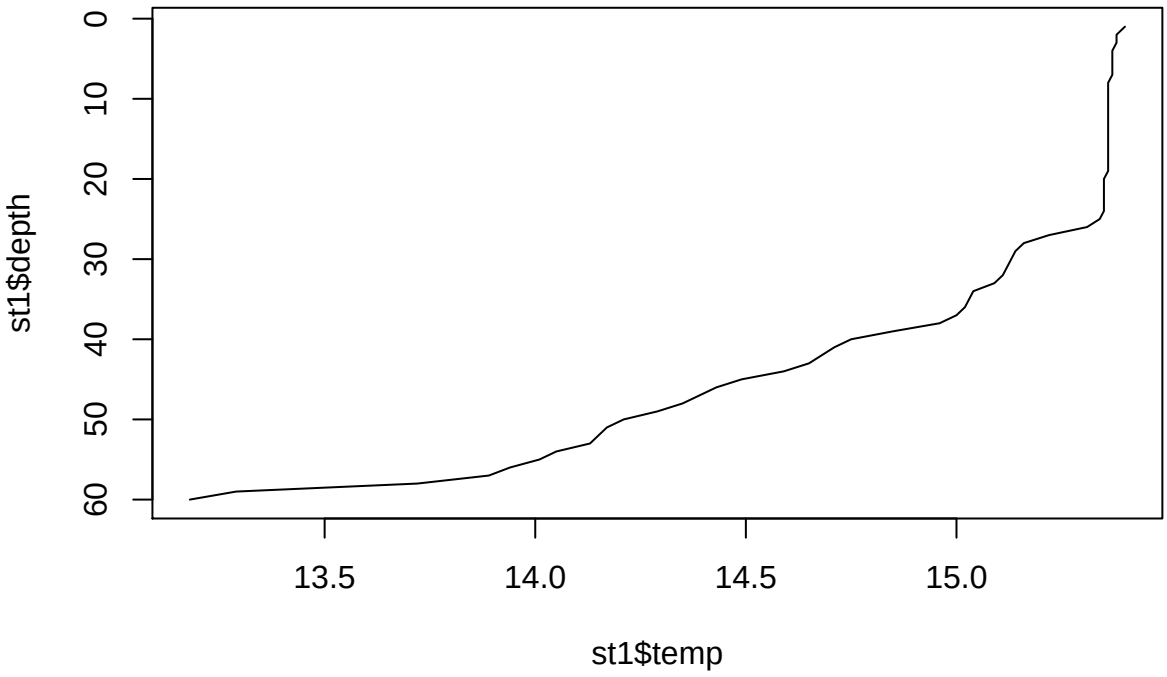
Axis limits can be controlled with `xlim` and `ylim`. Note that if either of these are specified, the excluded data will not be shown.

```
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`:

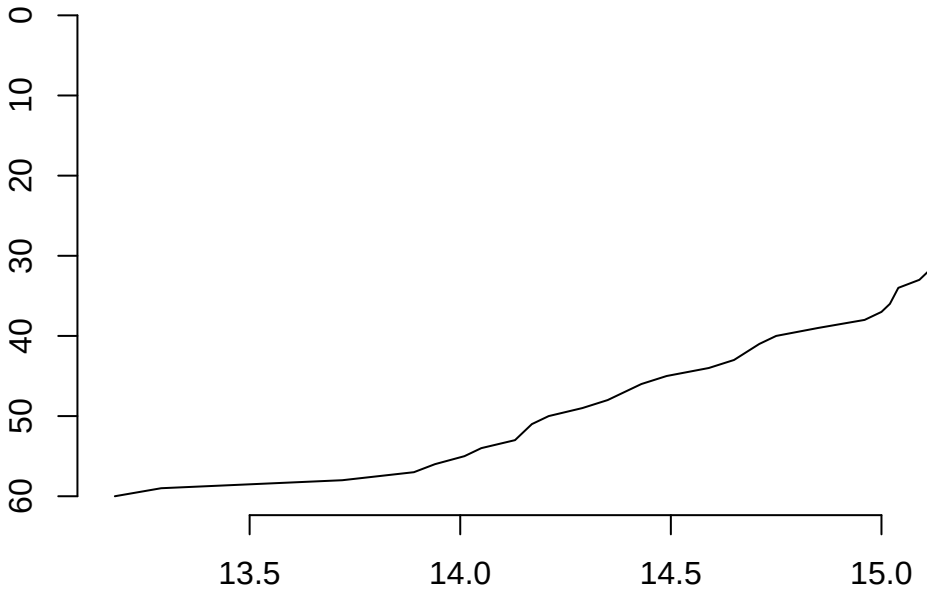
```
plot(st1$temp, st1$depth, type = "l", ylim = rev(range(st1$depth)))
```



Axes

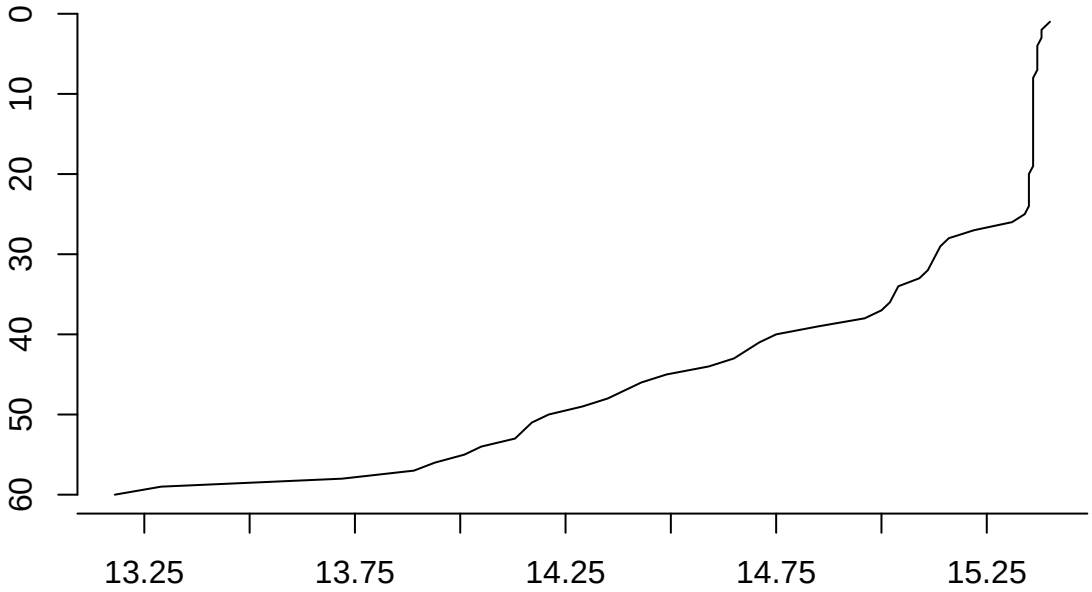
Axis ticks and labels can be customized, but they need to be first removed from the main plot, then added back individually with the `axis()` function. The first argument to the axis function (`side`) specifies the side the axis should be displayed on. This argument is a number from 1 to 4 where 1 =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

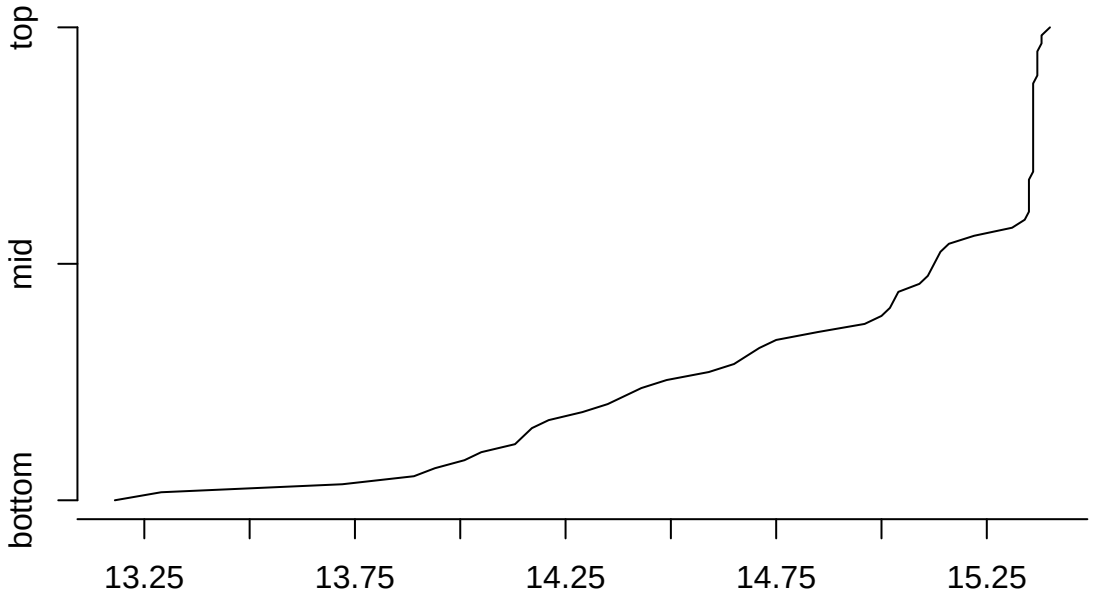
```
plot(
  st1$temp, st1$depth,
  type = "l", ylim = rev(range(st1$depth)),
  ann = FALSE, axes = FALSE
)
x.min <- floor(min(st1$temp))
x.max <- ceiling(max(st1$temp))
x.at <- seq(x.min, x.max, by = 0.25)
axis(1, at = x.at)
axis(2)
```



Axis labels are specified with the `label` argument:

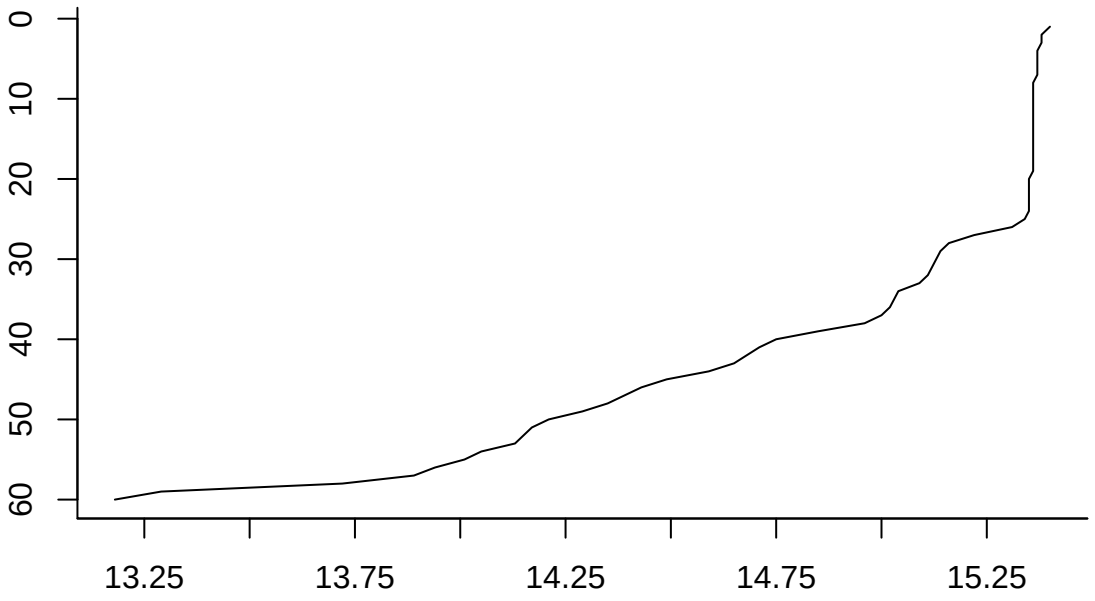
```
plot(
  st1$temp, st1$depth,
```

```
  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"))
```



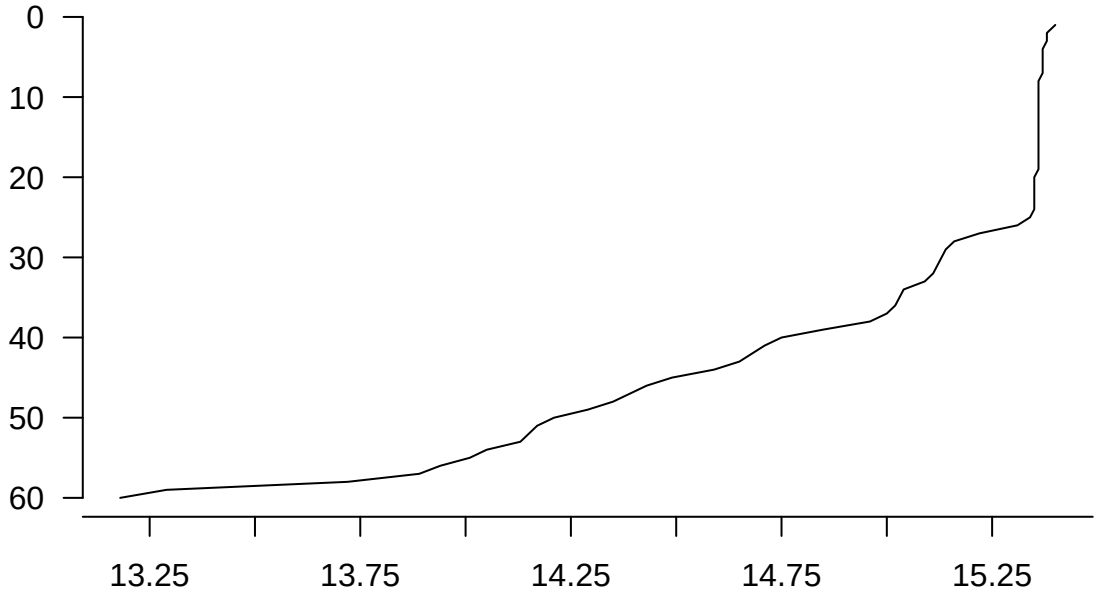
The plot can be surrounded by a box (`?box`). The `bty` argument (`?par`) will specify which sides are included in the box:

```
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`:

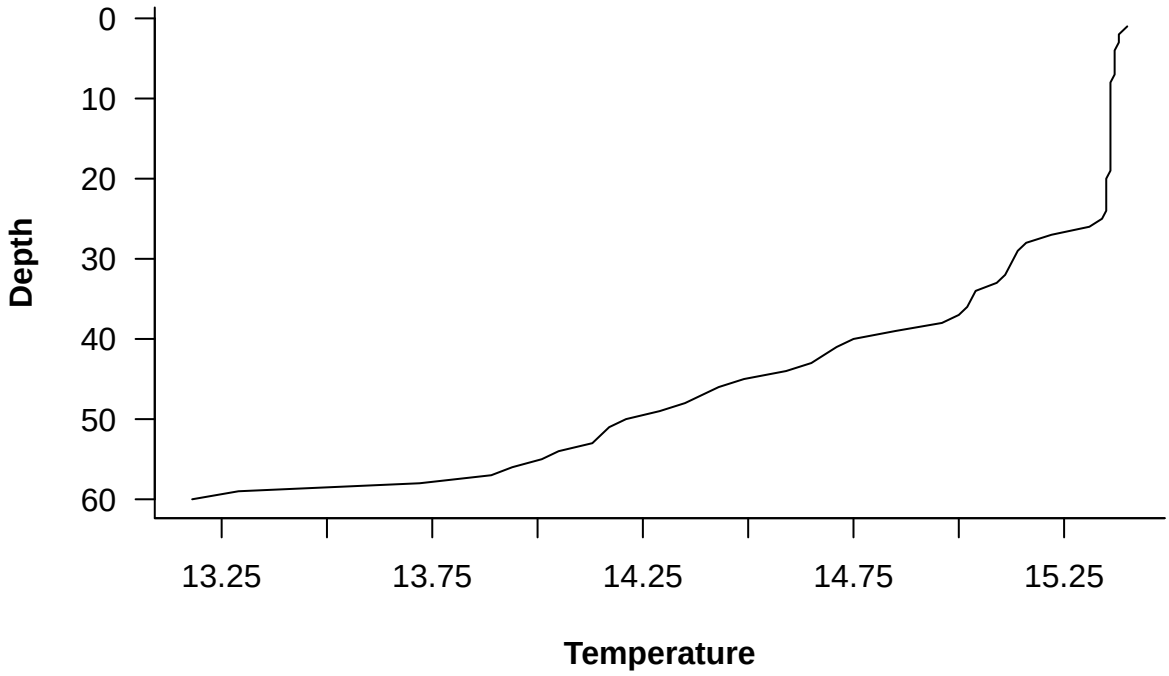
```
plot(
  st1$temp, st1$depth,
  type = "l", ylim = rev(range(st1$depth)),
  ann = FALSE, axes = FALSE
)
x.min <- floor(min(st1$temp))
x.max <- ceiling(max(st1$temp))
x.at <- seq(x.min, x.max, by = 0.25)
axis(1, at = x.at)
axis(2, las = 1)
```



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):

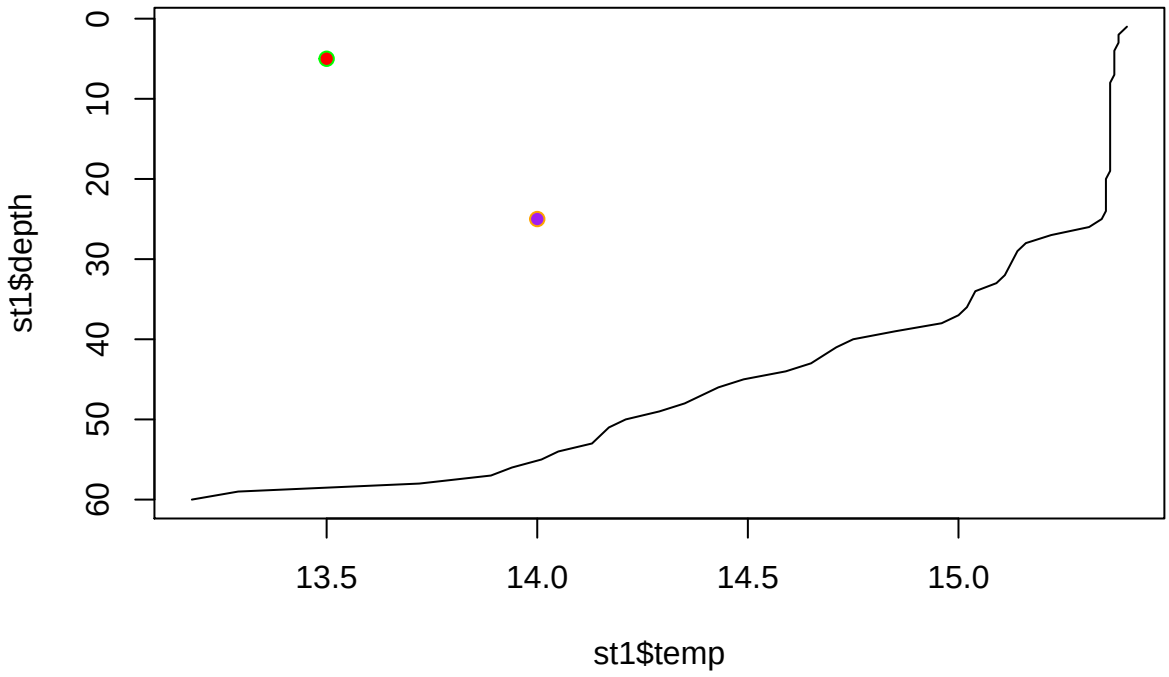
```
plot(
  st1$temp, st1$depth,
  type = "l", ylim = rev(range(st1$depth)),
  xlab = "Temperature", ylab = "Depth",
  family = "Helvetica", font.lab = 2,
  axes = FALSE
)
x.min <- floor(min(st1$temp))
x.max <- ceiling(max(st1$temp))
x.at <- seq(x.min, x.max, by = 0.25)
axis(1, at = x.at)
axis(2, las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

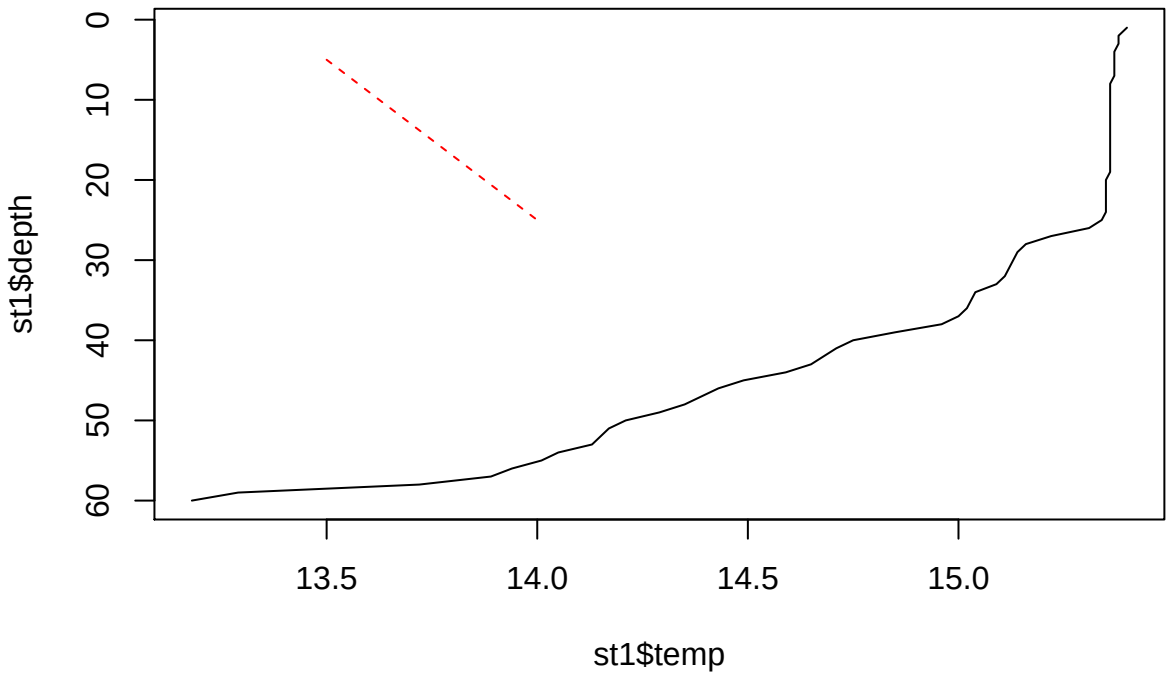
```
plot(
  st1$temp, st1$depth,
  type = "l", ylim = rev(range(st1$depth))
)
points(c(13.5, 14), c(5, 25), pch = 21, bg = c("red", "purple"), col = c("green", "orange"))
```



Lines

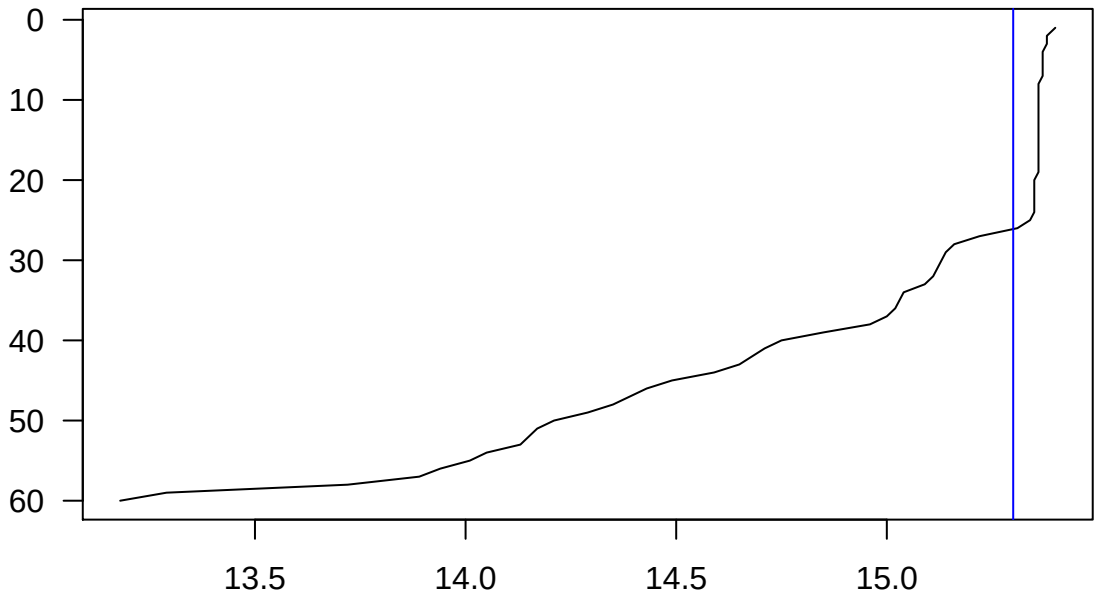
The same goes for lines to be added to a plot with the `lines` function.

```
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:

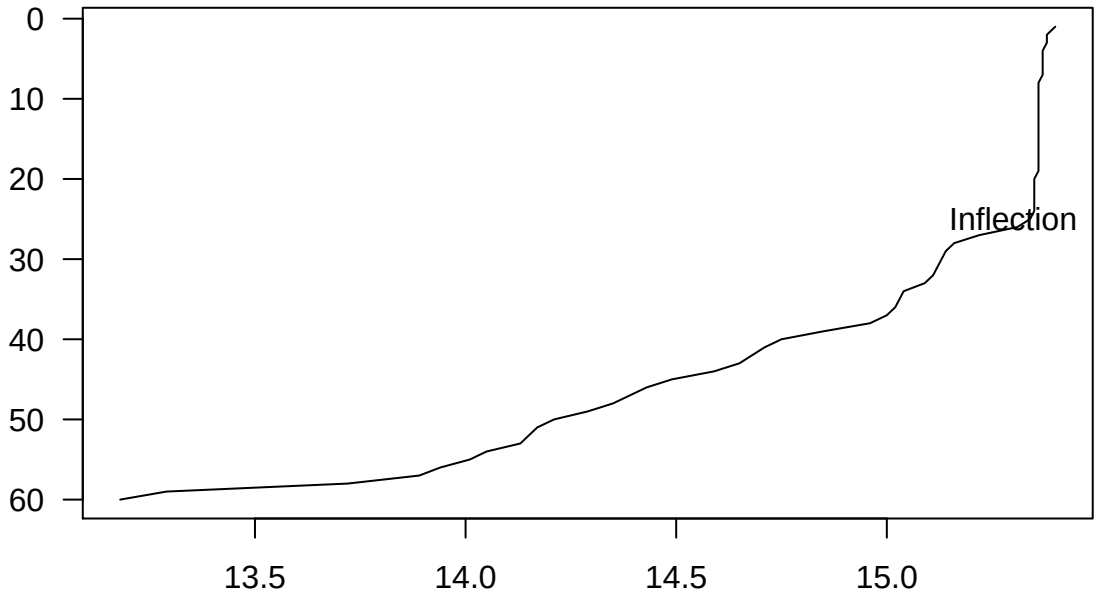
```
plot(
  st1$temp, st1$depth,
  type = "l", ylim = rev(range(st1$depth)),
  ann = FALSE, axes = FALSE
)
axis(1)
axis(2, las = 2)
box()
abline(v = 15.3, col = "blue")
```



Text

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

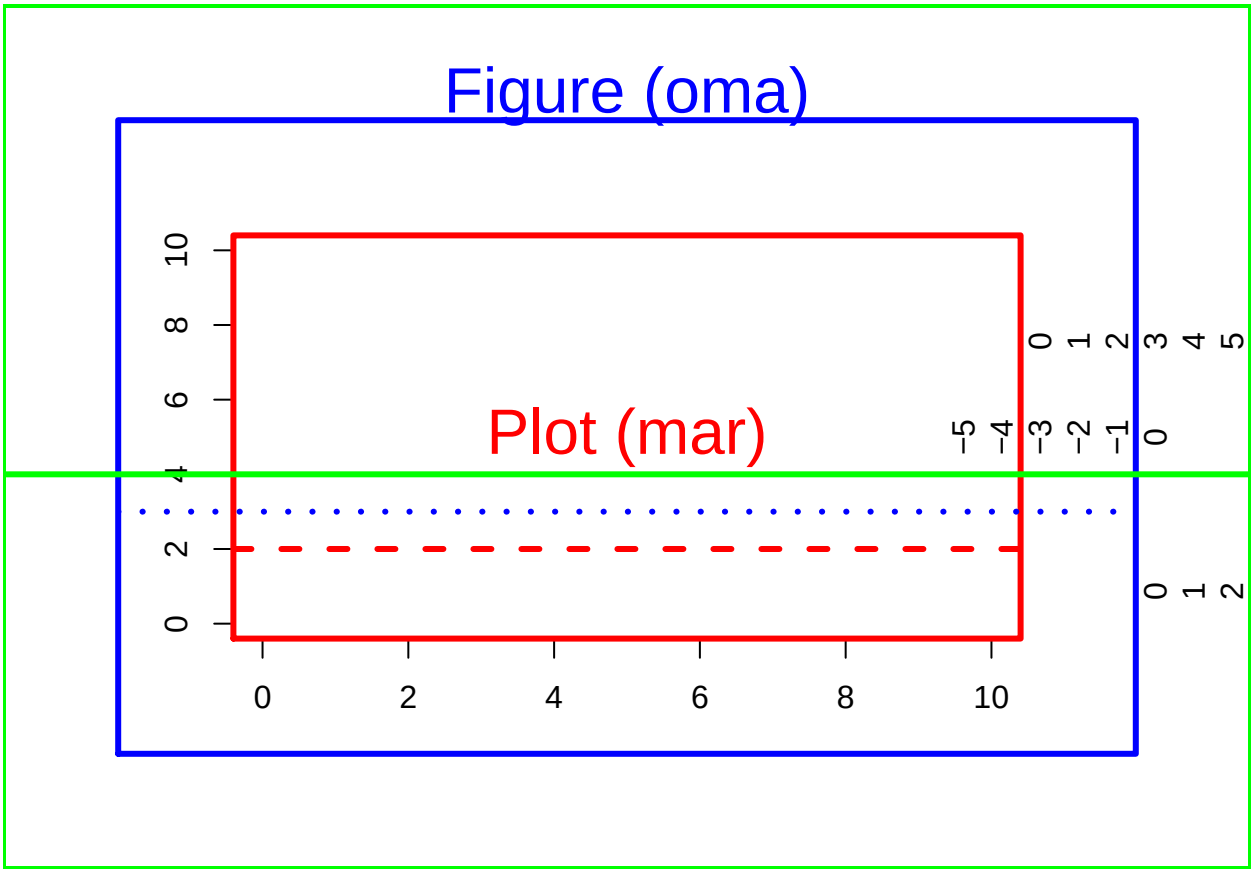
```
plot(
  st1$temp, st1$depth,
  type = "l", ylim = rev(range(st1$depth)),
  ann = FALSE, axes = FALSE
)
axis(1)
axis(2, las = 2)
box()
text(15.3, 25, "Inflection")
```



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.

```
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)
```

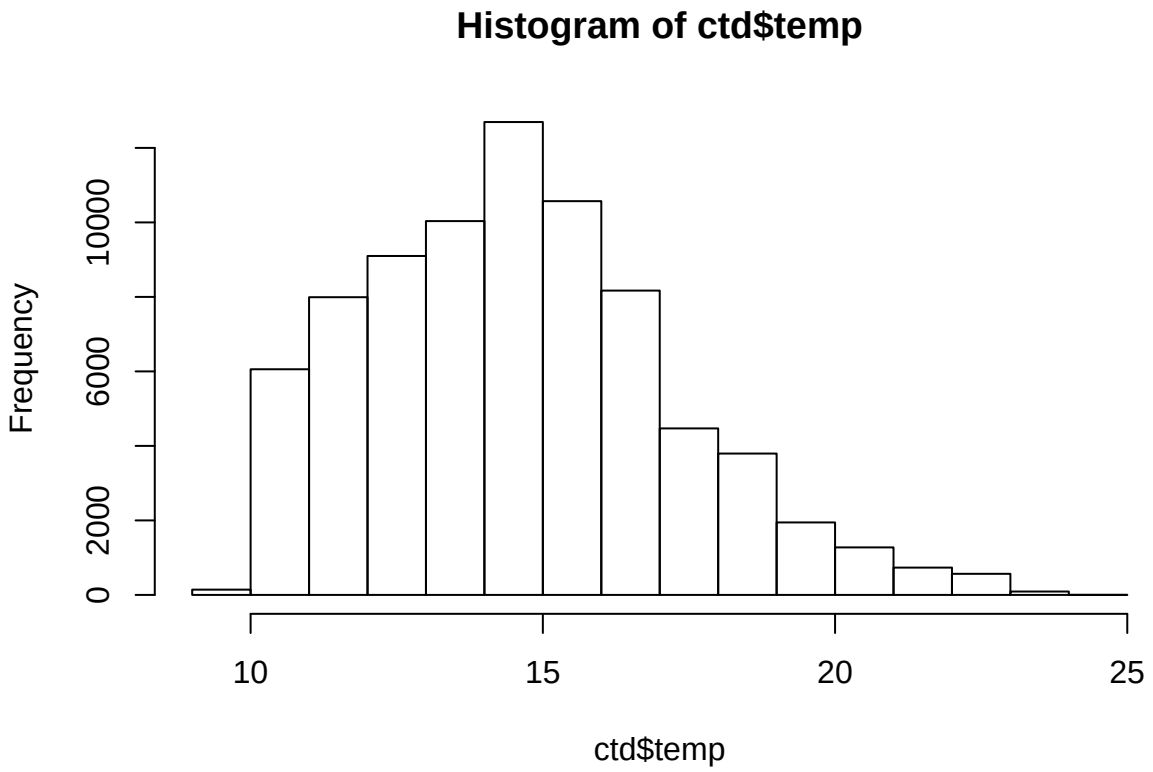



```
par(op)
```

Histograms

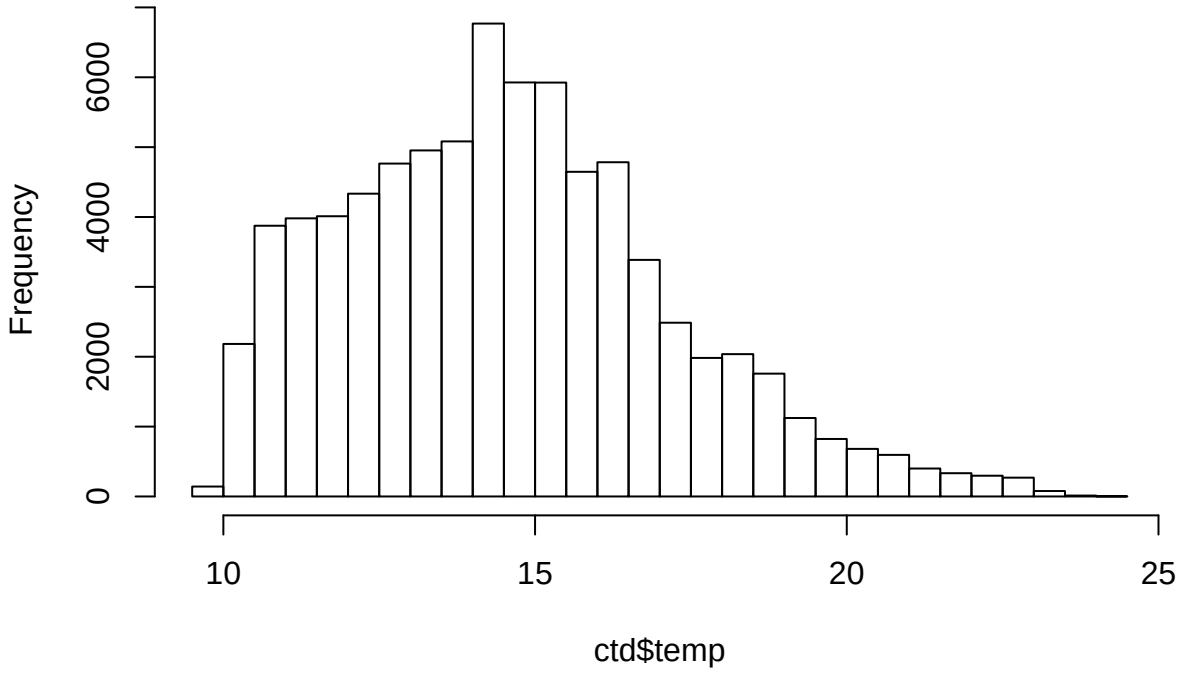
To plot frequencies of binned continuous variables, create a histogram with the `hist` function.

```
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 = "")
```

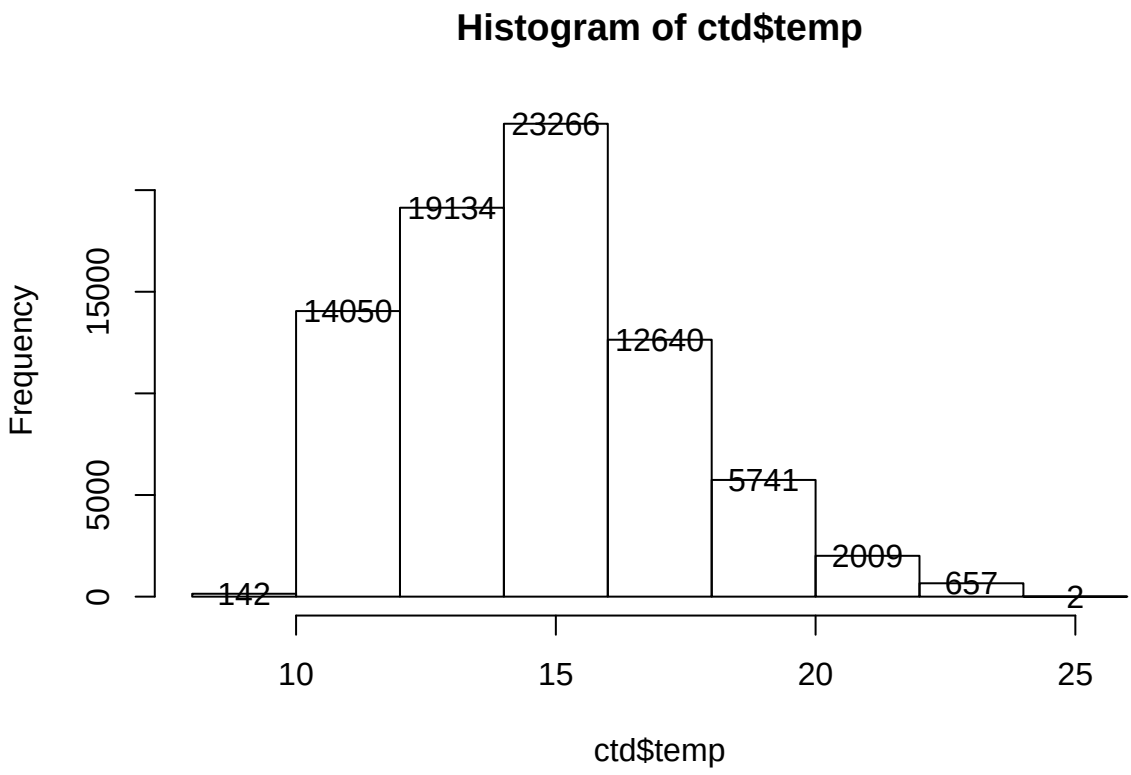


. . . or the actual breaks can be given. Also, if the result of `hist` is assigned to an object, information about the binning is stored in that object and can be used to annotate it:

```
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"
```

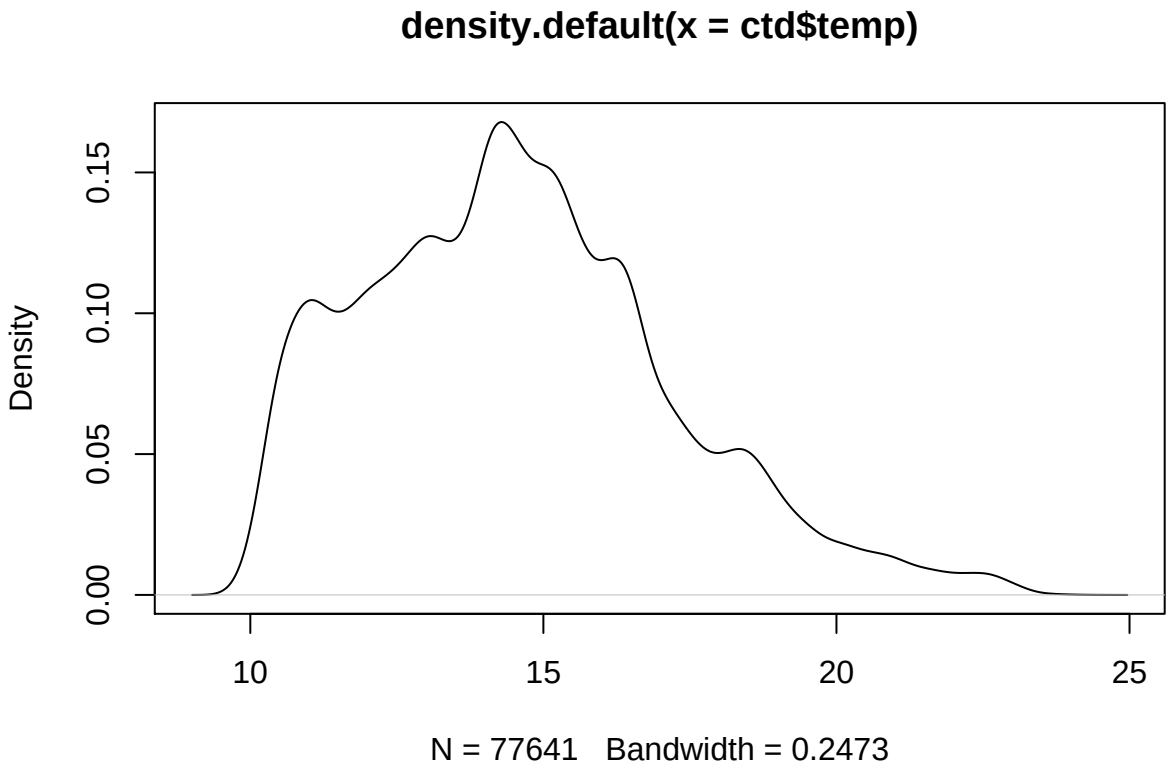
```
text(hist.vals$mids, hist.vals$counts, hist.vals$counts)
```



Density plots

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

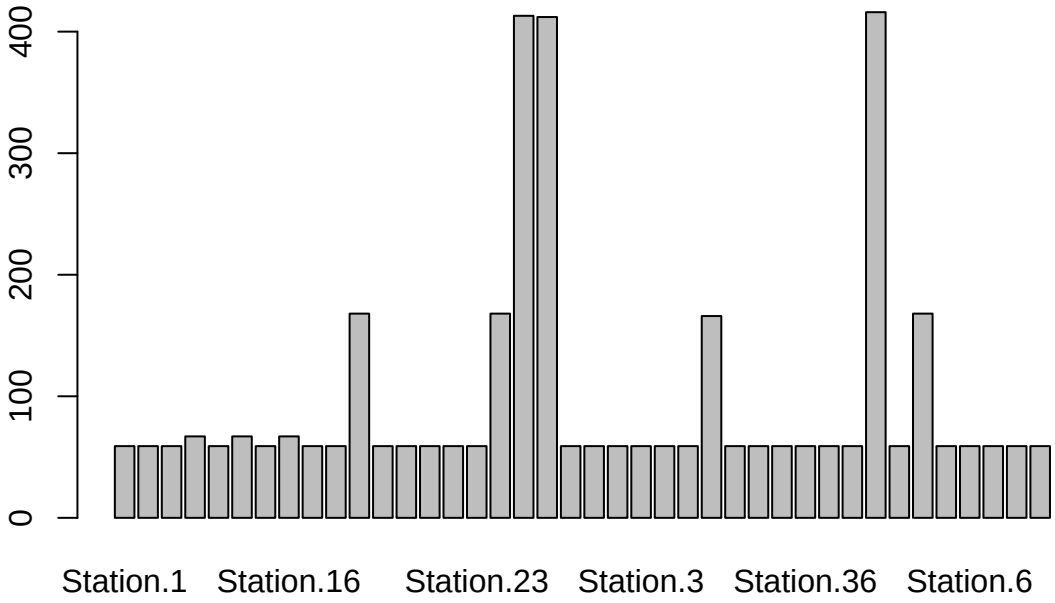
```
plot(density(ctd$temp))
```



Barplots

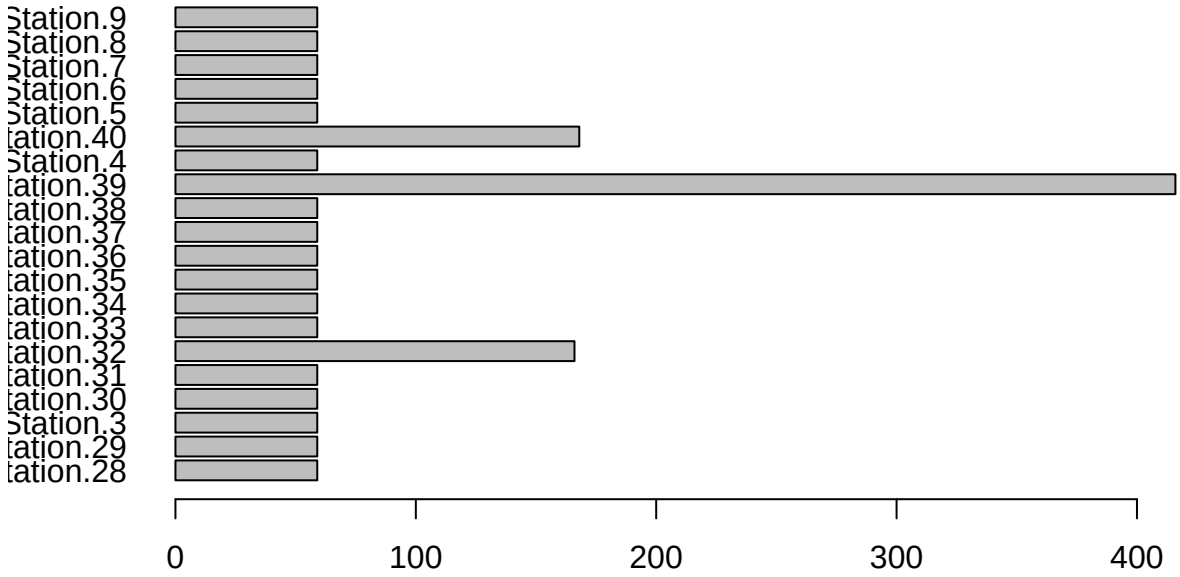
To plot a set values from a vector, like precalculated frequencies, use `barplot`:

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



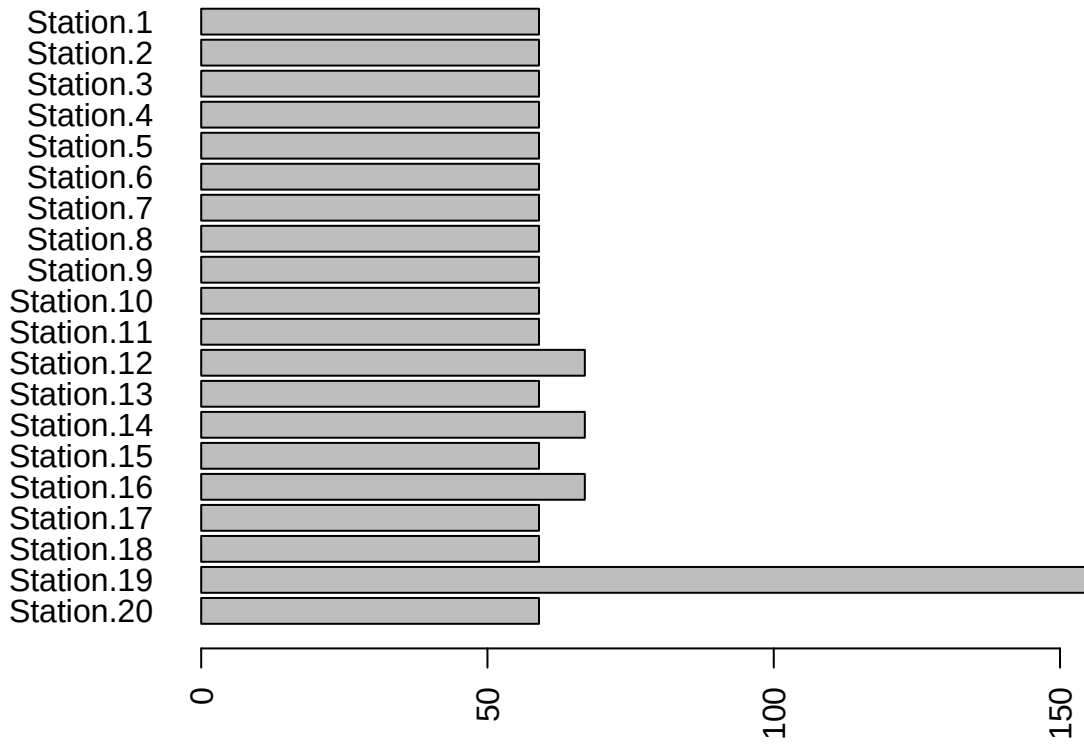
Barplots can also be plotted horizontally by setting `horiz = TRUE`:

```
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)
```

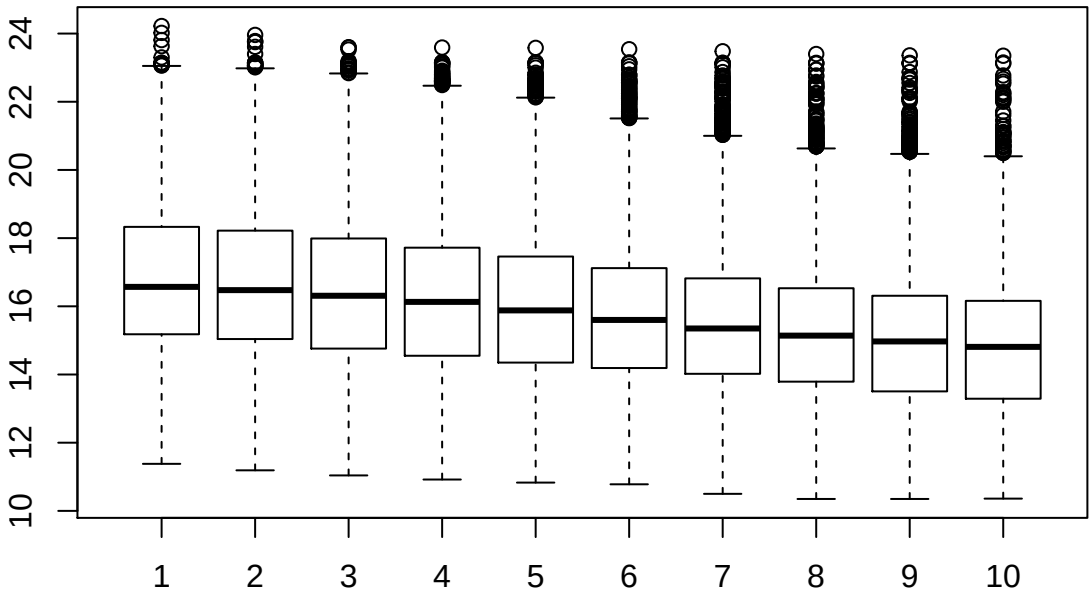


```
par(op)
```

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:

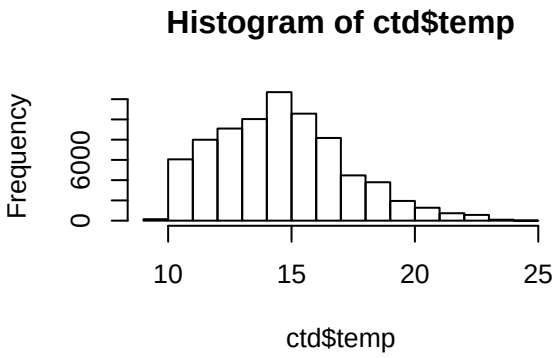
```
top.10 <- subset(ctd, depth <= 10)
boxplot(temp ~ depth, top.10)
```

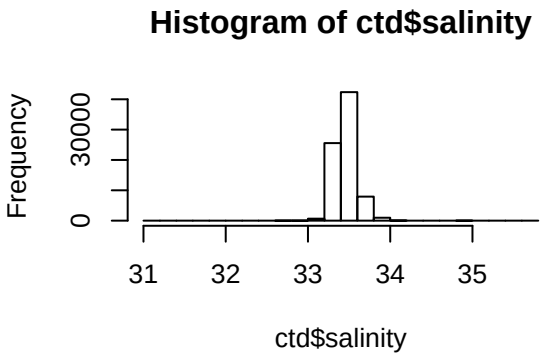


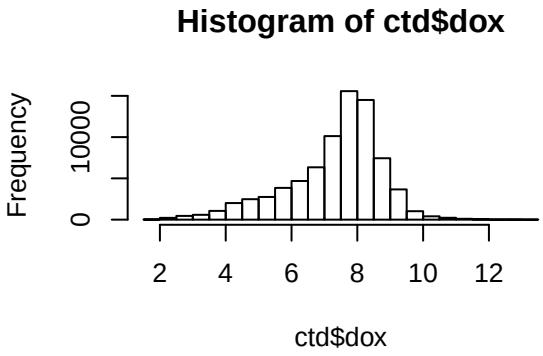
Multple panels: mfrow/mfcol

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)
```







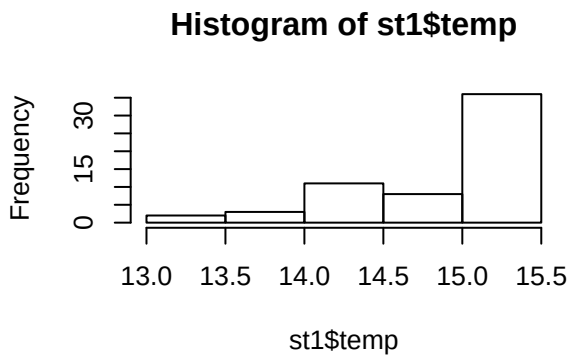
Multiple panels: layout

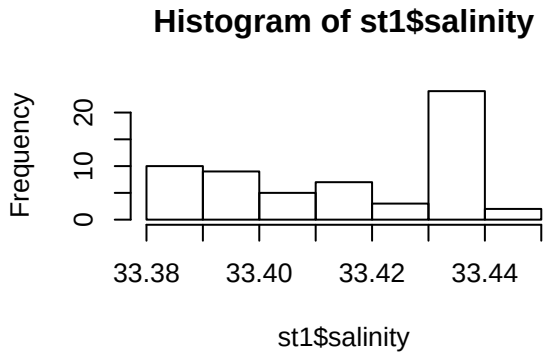
Another way is to use `layout` which requires mapping specified through a matrix. The values in the matrix correspond to the locations of the plots on the page:

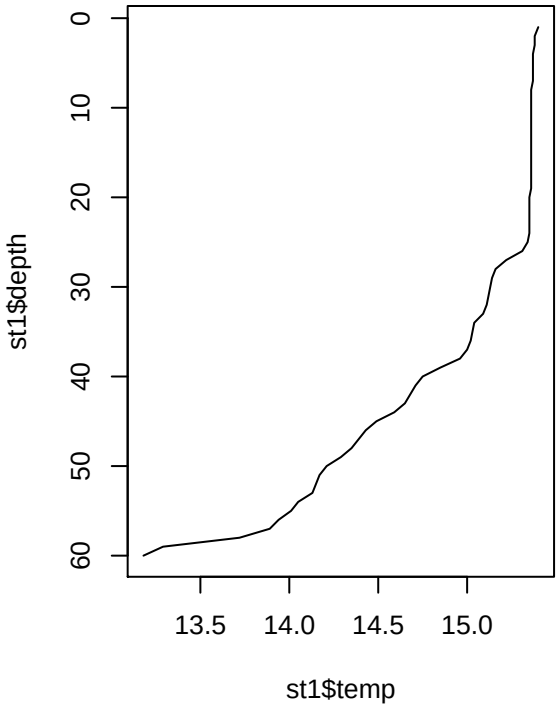
```
lm <- matrix(c(1, 2, 3, 3), nrow = 2)
lm
```

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

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



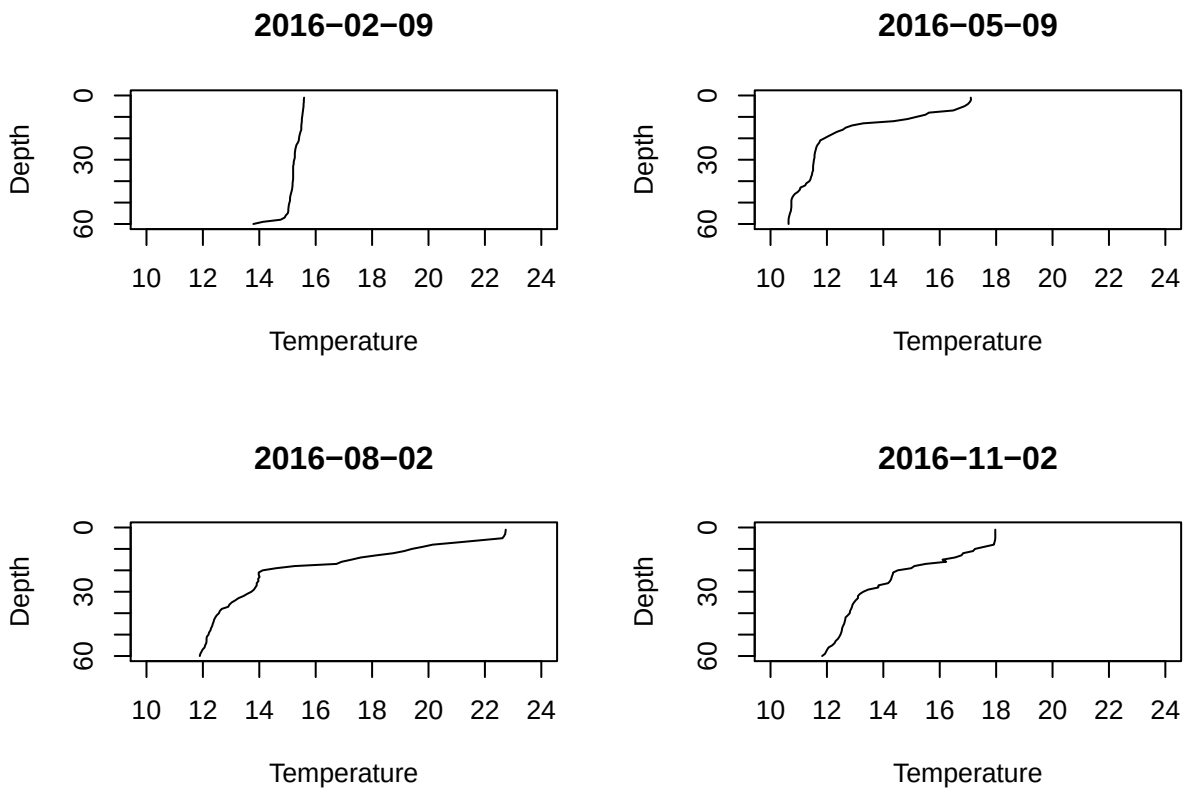





```
layout(matrix(1))
```

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
  )
}
```




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

---

ggplot

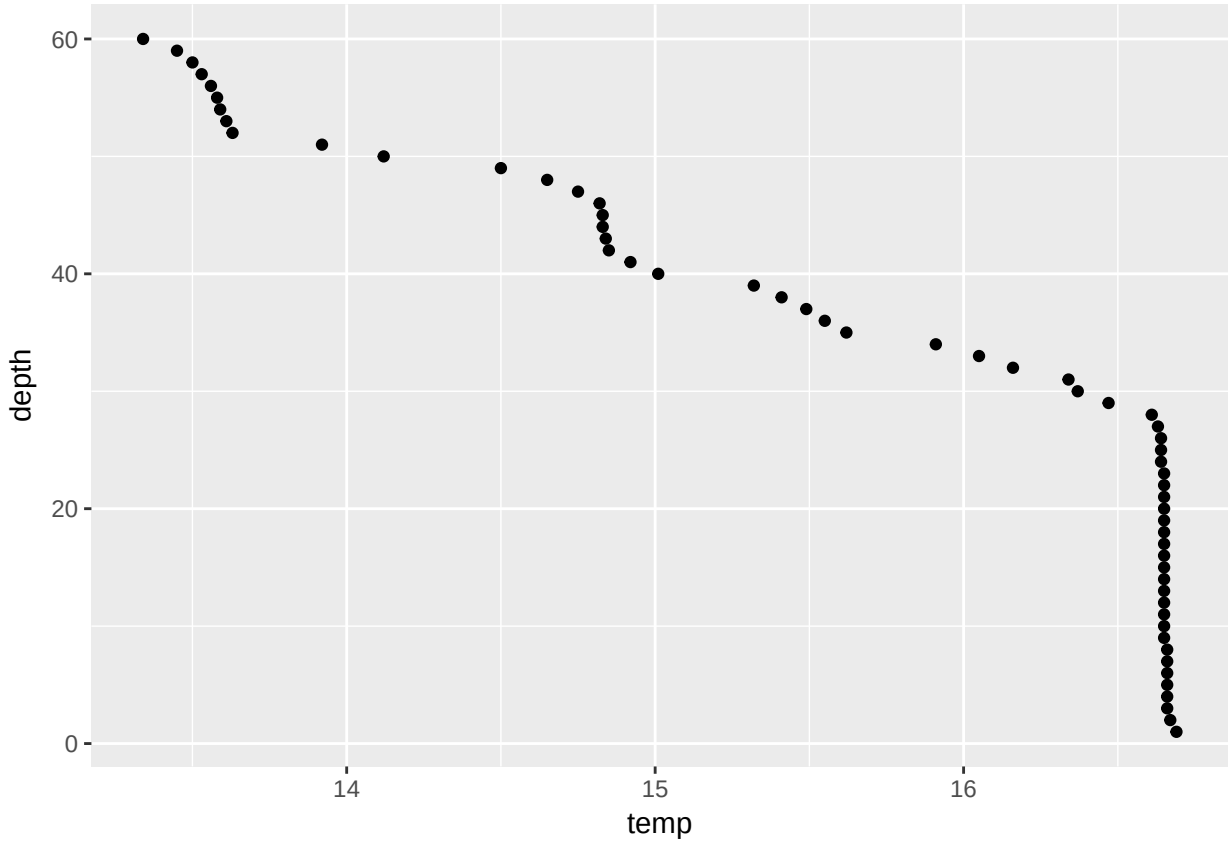
ggplot is based entirely around constructing a properly formed data.frame of what needs to be plotted and then constructing a plotting statement with the components linked with `+`. The values in the data.frame that are to be plotted are identified with the appropriate "mapping" that is referred to as an "aesthetic". They are specified with the `aes` function with a ggplot object. The first item in a ggplot figure is the `ggplot` function that sets up the data and aesthetic. Here we are setting up a base ggplot object that has the `x` value mapped to `temperature` and the `y` value mapped to `depth`:

```
# read CTD data and format date columns
ctd <- read.csv("ctd.csv", stringsAsFactors = FALSE)
ctd$date <- as.POSIXct(ctd$sample_date, format = "%Y-%m-%d")
ctd$month <- months(as.Date(ctd$date))
ctd$month <- factor(ctd$month, levels = month.name)
ctd$year <- as.numeric(format(ctd$date, "%Y"))
ctd$quarter <- factor(quarters(as.Date(ctd$date)))

df <- ctd[ctd$station == "Station.1" & grepl("2015", ctd$sample_date) & ctd$month == "February", ]

library(ggplot2)
p <- ggplot(df, mapping = aes(x = temp, y = depth))
```

You can see that nothing happens because we haven't specified how to use that mapping. To do that, we have

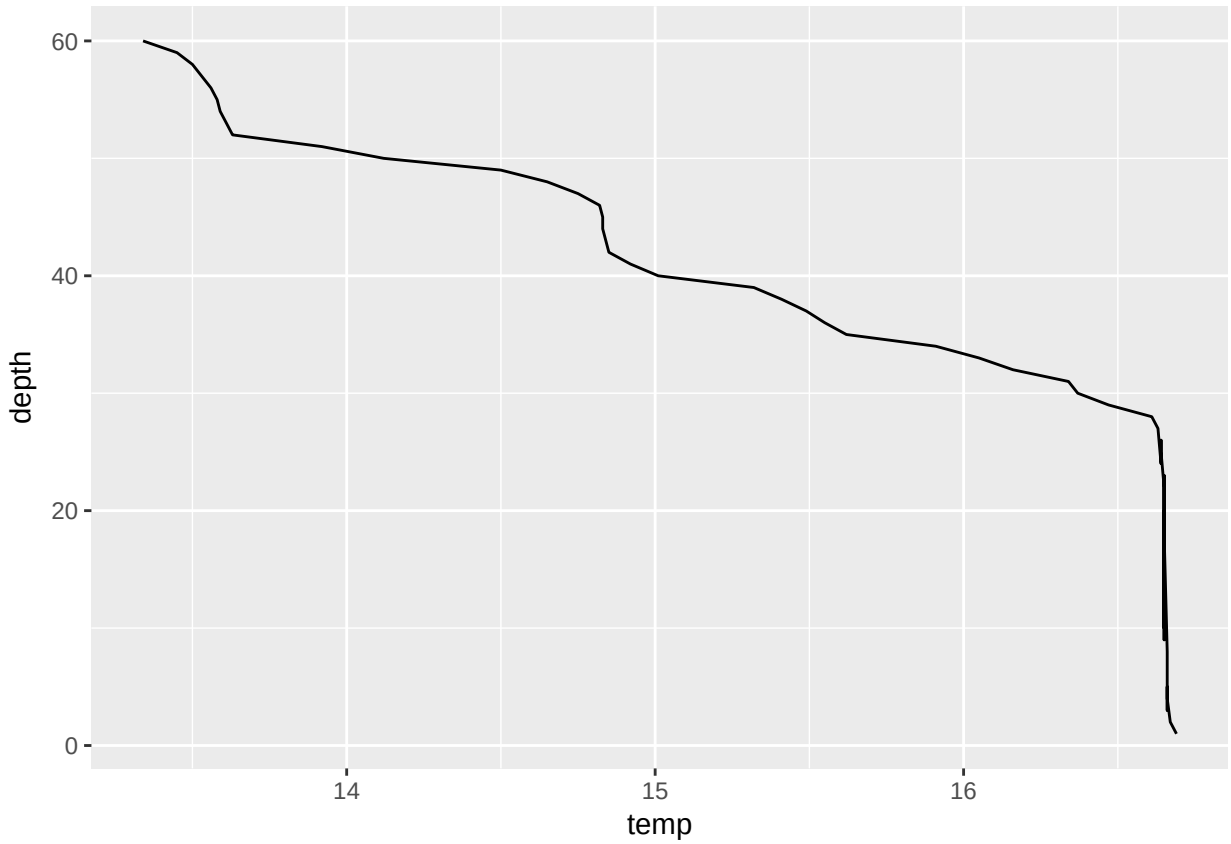
to specify a "geometry" which usually begins with `geom_`. Let's plot some simple points with `geom_point`:

```
p <- ggplot(df, mapping = aes(x = temp, y = depth)) +
  geom_point()
# we have to use `print` to see the result...
print(p)
```



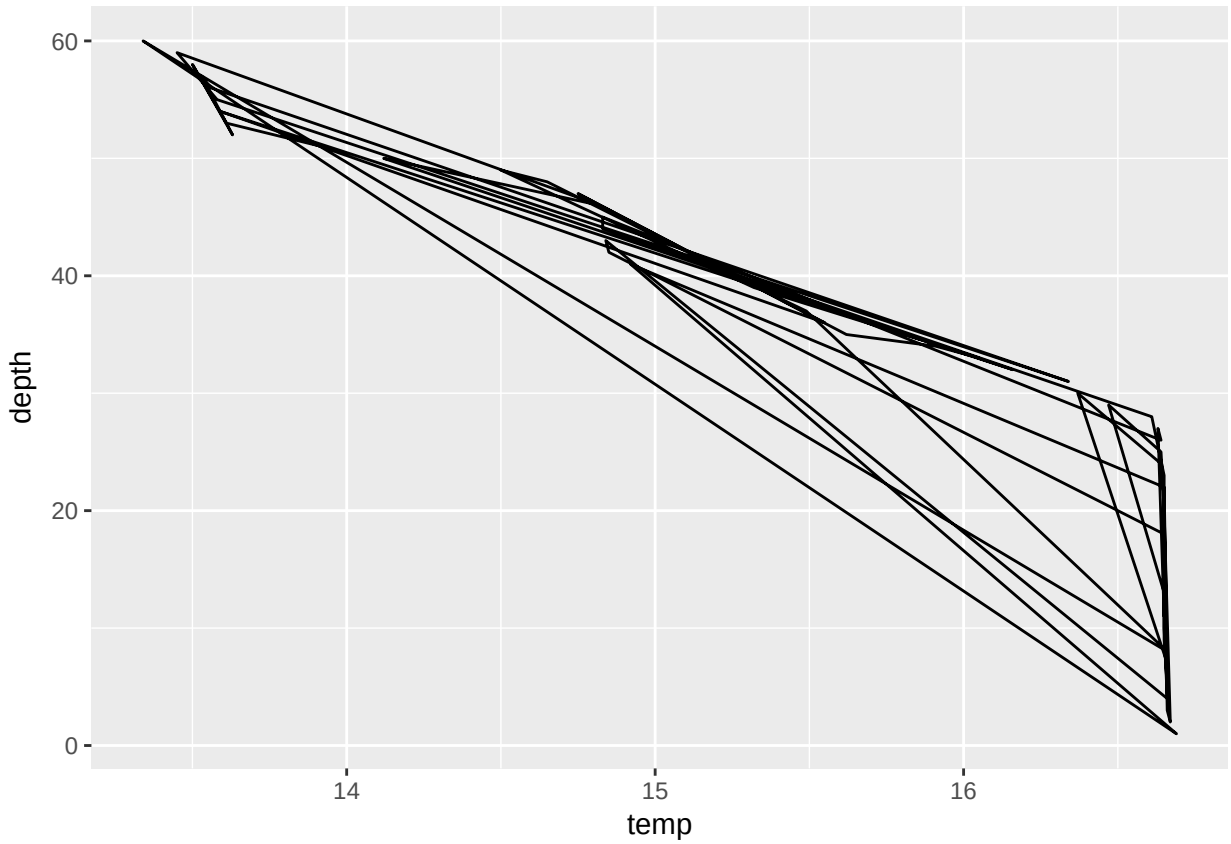
There are two geometries for lines, `geom_line` and `geom_path`. `geom_line` connects the points in the order of the x-axis:

```
p <- ggplot(df, mapping = aes(x = temp, y = depth)) +
  geom_line()
print(p)
```



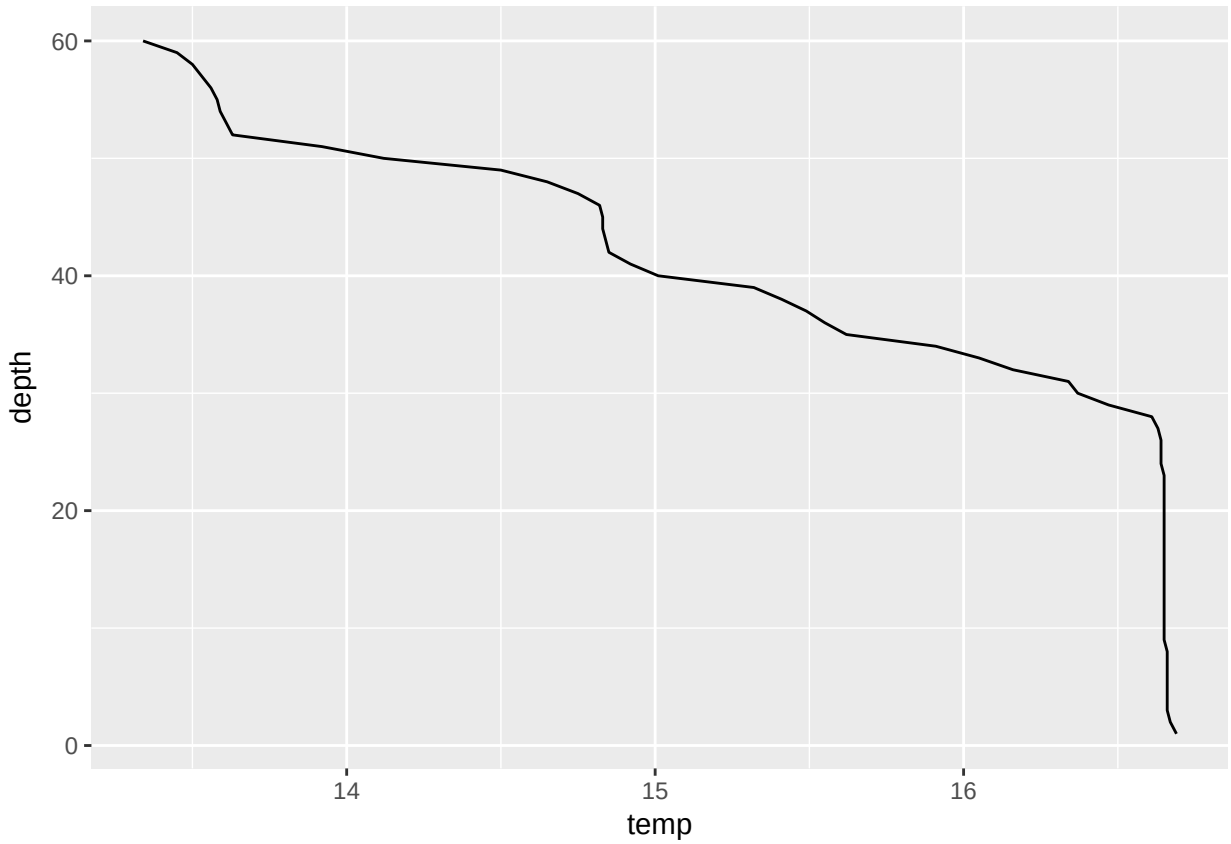
while `geom_path` connects them in order they are found:

```
p <- ggplot(df, mapping = aes(x = temp, y = depth)) +
  geom_path()
print(p)
```



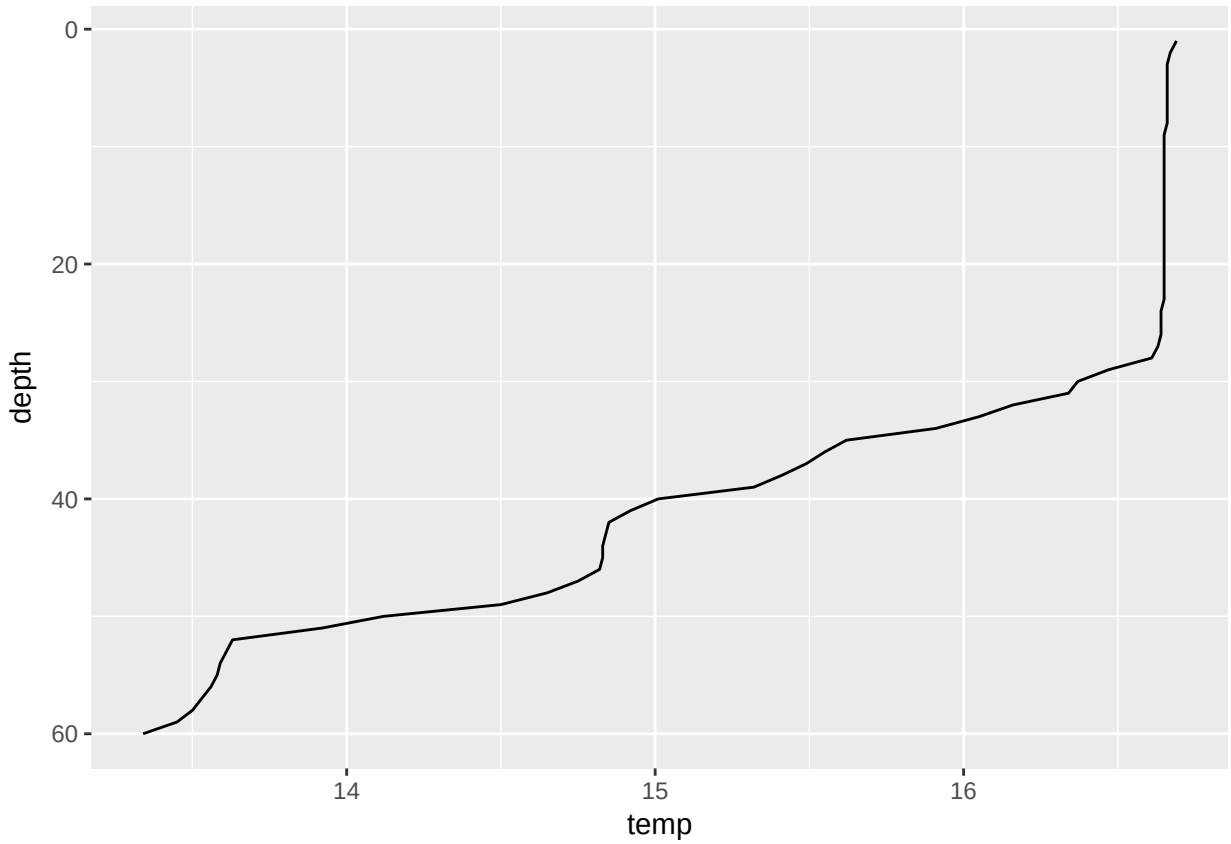
We need to first sort our data in order of depth, then connect them in order with `geom_path`:

```
df <- df[order(df$depth), ]
p <- ggplot(df, mapping = aes(x = temp, y = depth)) +
  geom_path()
print(p)
```



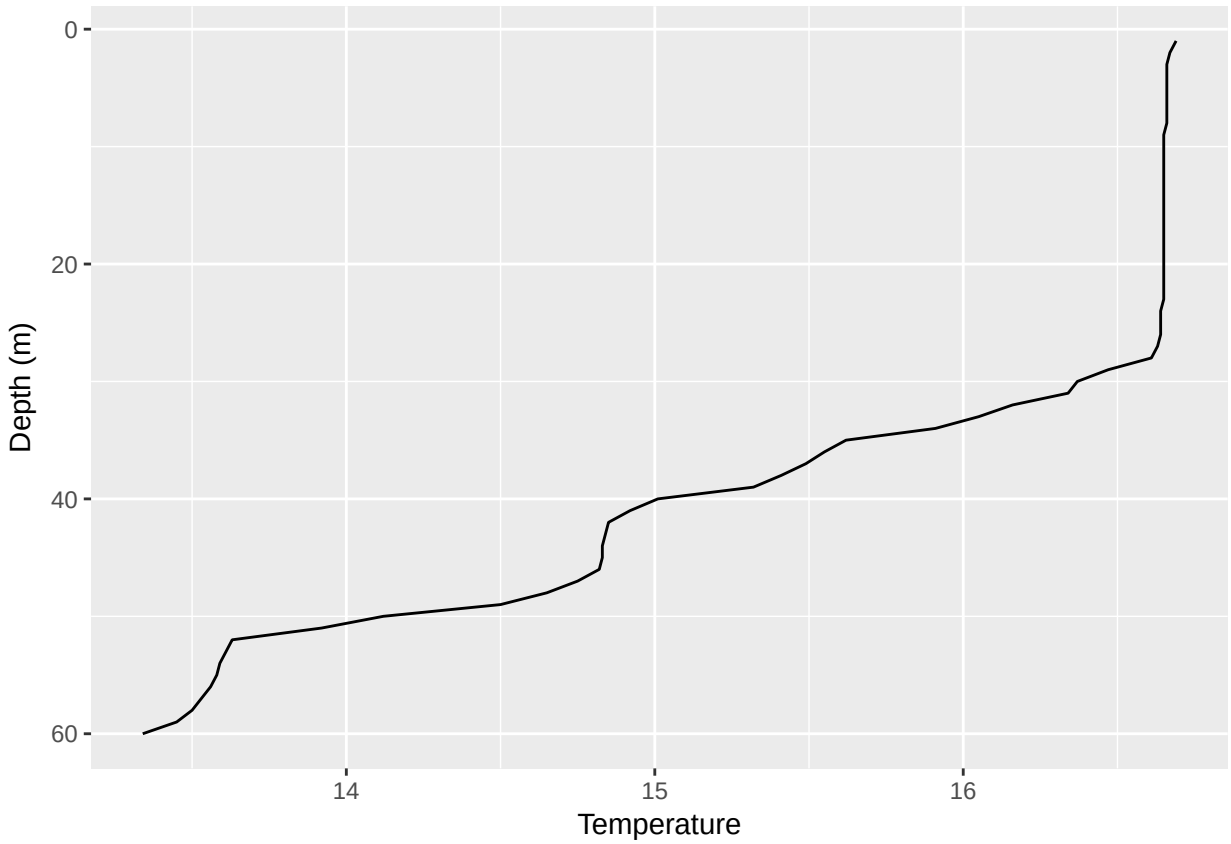
We still have the same problem as before, where we need to reverse the scale of our y-axis to get depth to go from small values at top to high values on the bottom. We can do this by adding 'scale_x_reverse':

```
df <- df[order(df$depth), ]
p <- ggplot(df, mapping = aes(x = temp, y = depth)) +
  geom_path() +
  scale_y_reverse()
print(p)
```



We can specify our axis labels individually with `xlab` and `ylab`, or together with `labs`:

```
df <- df[order(df$depth), ]
p <- ggplot(df, mapping = aes(x = temp, y = depth)) +
  geom_path() +
  scale_y_reverse() +
  labs(x = "Temperature", y = "Depth (m)")
print(p)
```

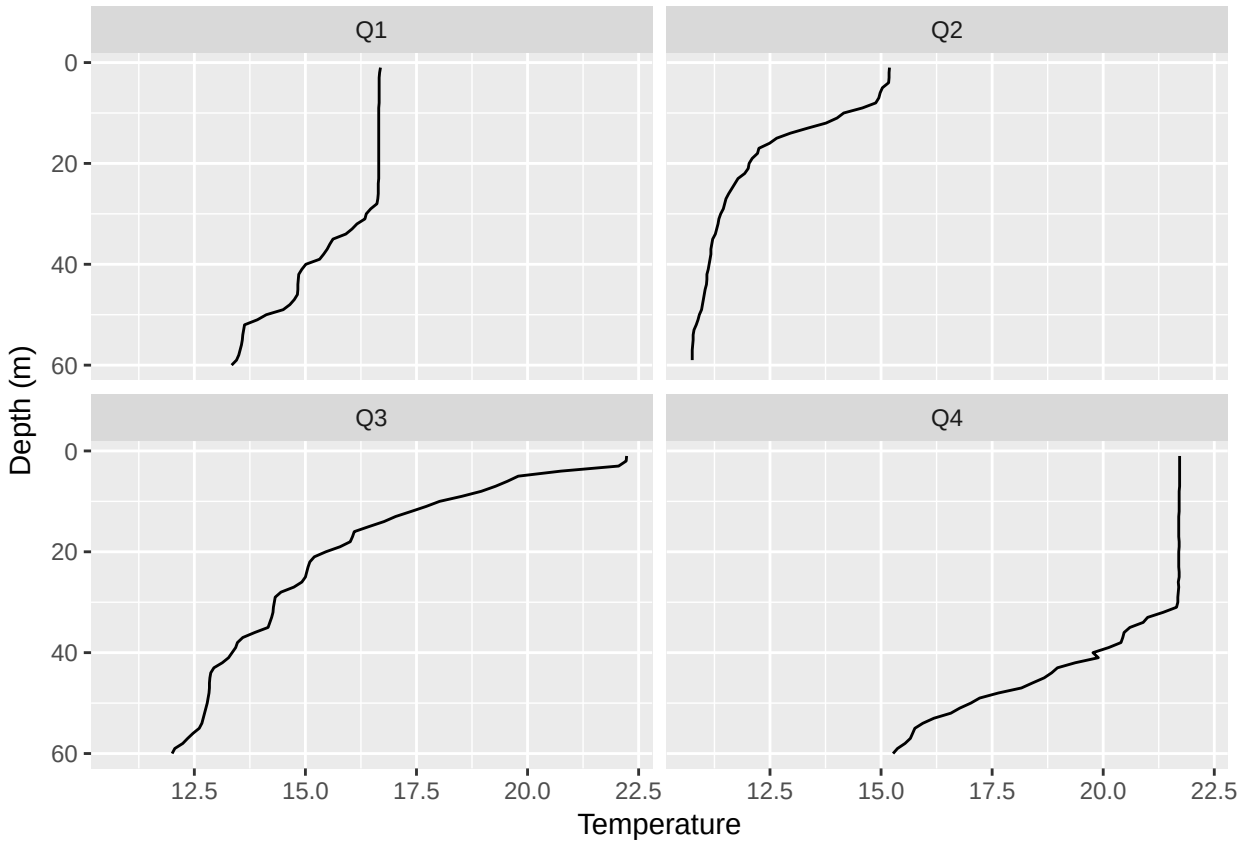


Facetting

Multiple panels in ggplot are created using "facets". There are two primary ways to do this: creating a facet for sequential levels of a factor, where the panels are placed in a specified number of rows and/or columns (`facet_wrap`), or two-dimensional facets where one factor is represented by rows and the other by the columns (`facet_grid`).

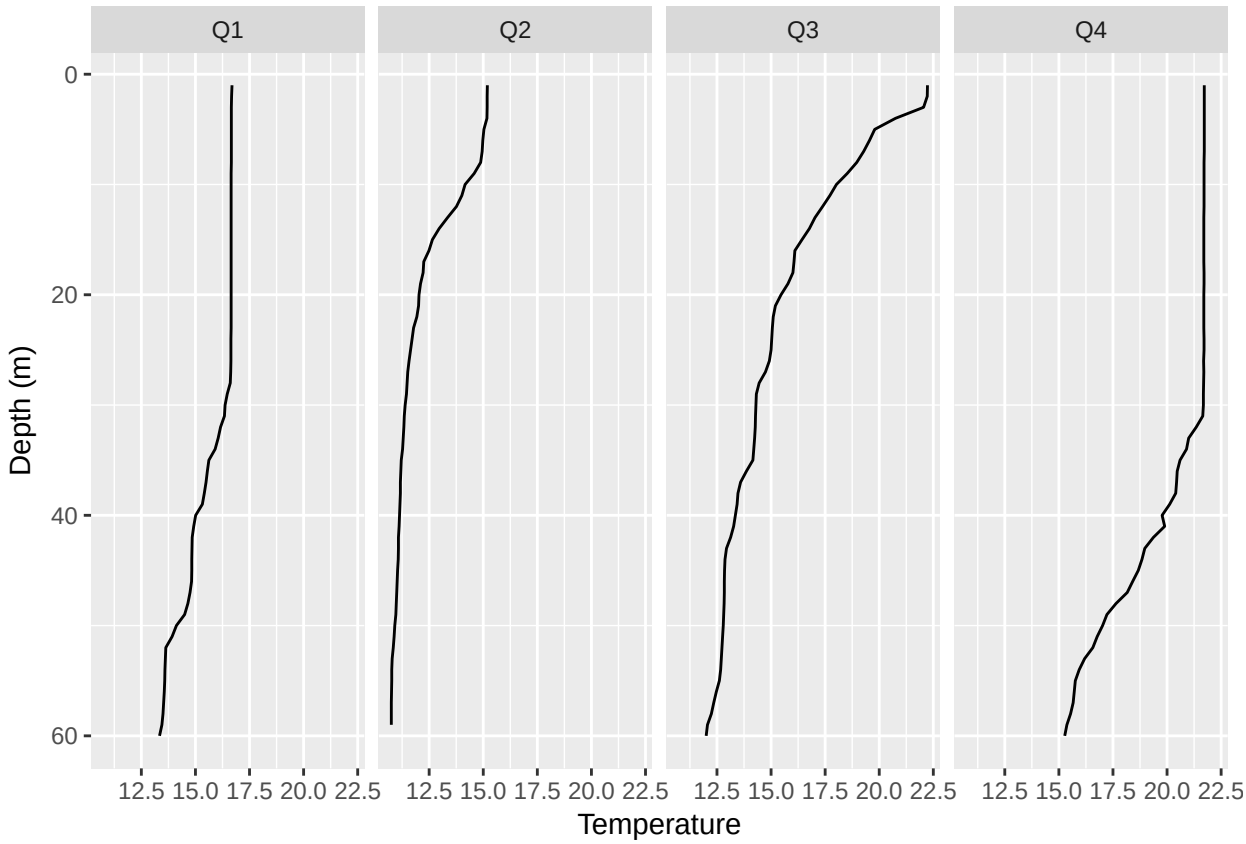
Here is an example of `facet_wrap` to plot the temperature profile for every quarter at Station.1 in 2015:

```
df <- subset(ctd, station == "Station.1" & year == 2015)
df <- df[order(df$quarter, df$depth), ]
p <- ggplot(df, mapping = aes(x = temp, y = depth)) +
  geom_path() +
  scale_y_reverse() +
  facet_wrap(~ quarter) +
  labs(x = "Temperature", y = "Depth (m)")
print(p)
```



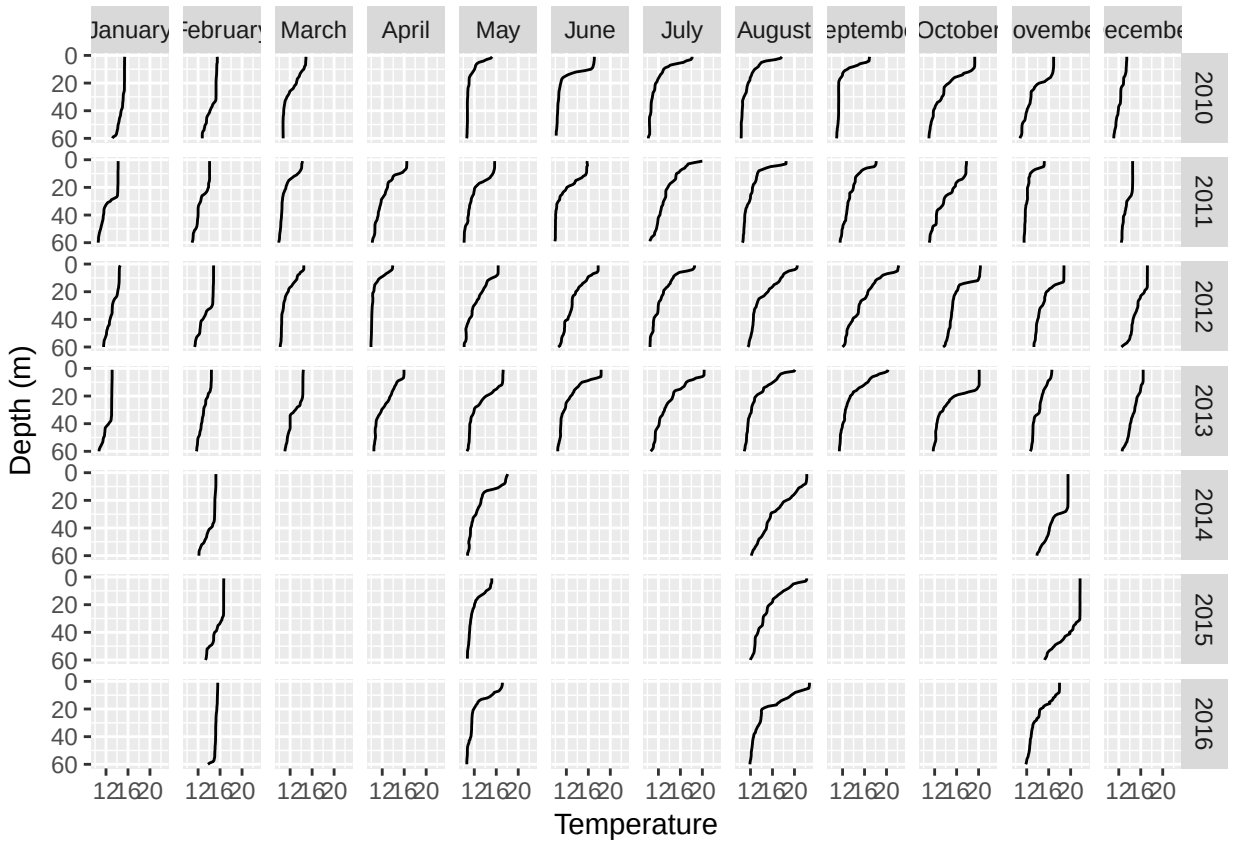
We can make this a single row or single column:

```
df <- subset(ctd, station == "Station.1" & year == 2015)
df <- df[order(df$quarter, df$depth), ]
p <- ggplot(df, mapping = aes(x = temp, y = depth)) +
  geom_path() +
  scale_y_reverse() +
  facet_wrap(~ quarter, nrow = 1) +
  labs(x = "Temperature", y = "Depth (m)")
print(p)
```



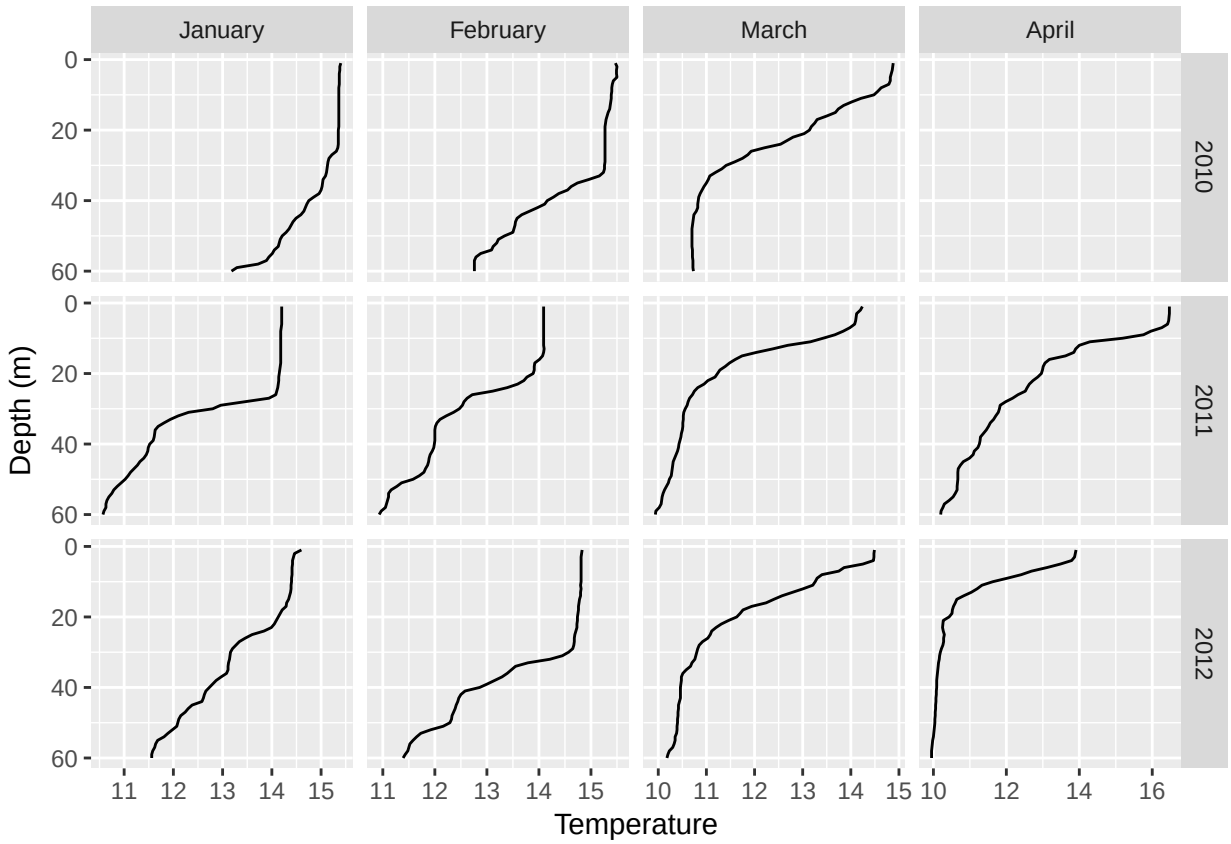
To demonstrate `facet_grid`, let's look at the temperature profiles for Station.39 across months(rows) and years (columns):

```
df <- subset(ctd, station == "Station.1")
df <- df[order(df$year, df$quarter, df$depth), ]
p <- ggplot(df, mapping = aes(x = temp, y = depth)) +
  geom_path() +
  scale_y_reverse() +
  facet_grid(year ~ month) +
  labs(x = "Temperature", y = "Depth (m)")
print(p)
```

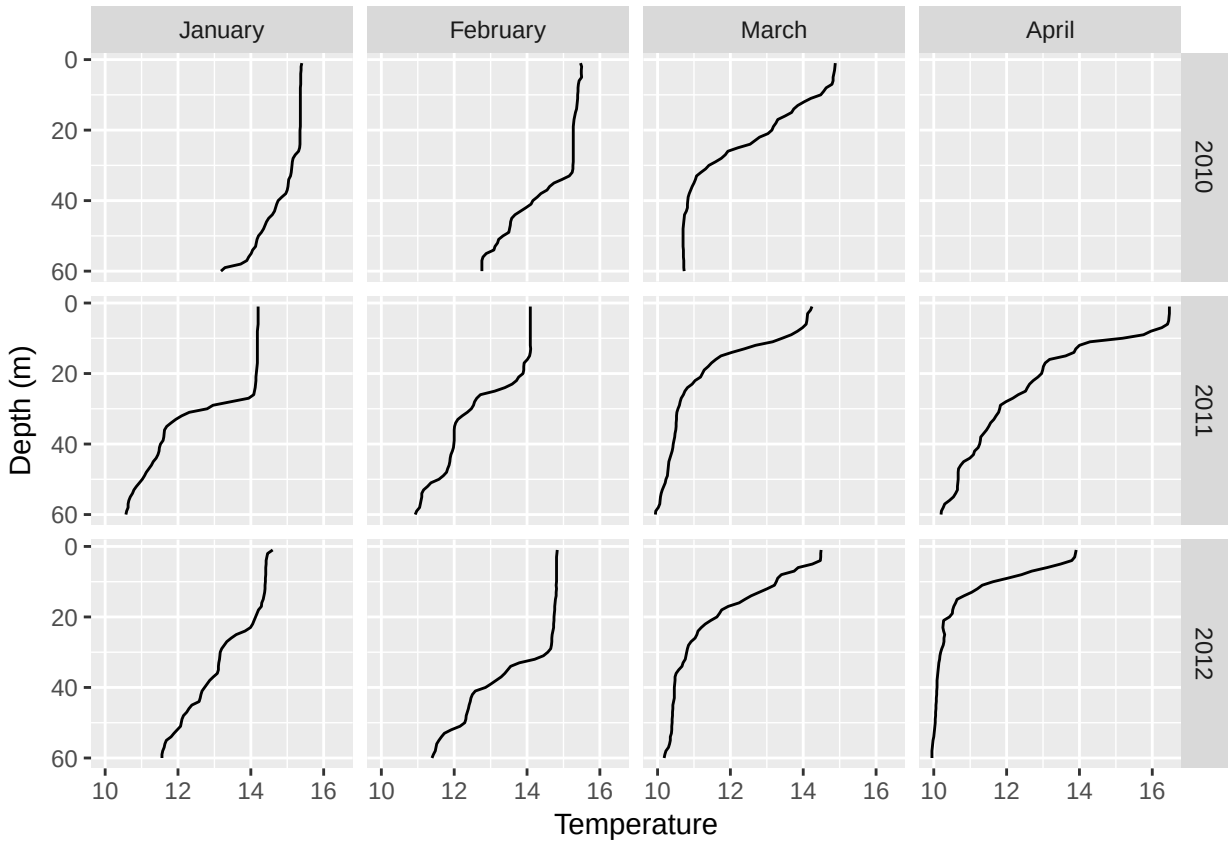


You can control how the axis scales are set in the facets. By default all facets share the same axis ranges. You can let axes have their own ranges for rows or columns by specifying the `scales` argument as either `free_x`, `free_y`, or `free`:

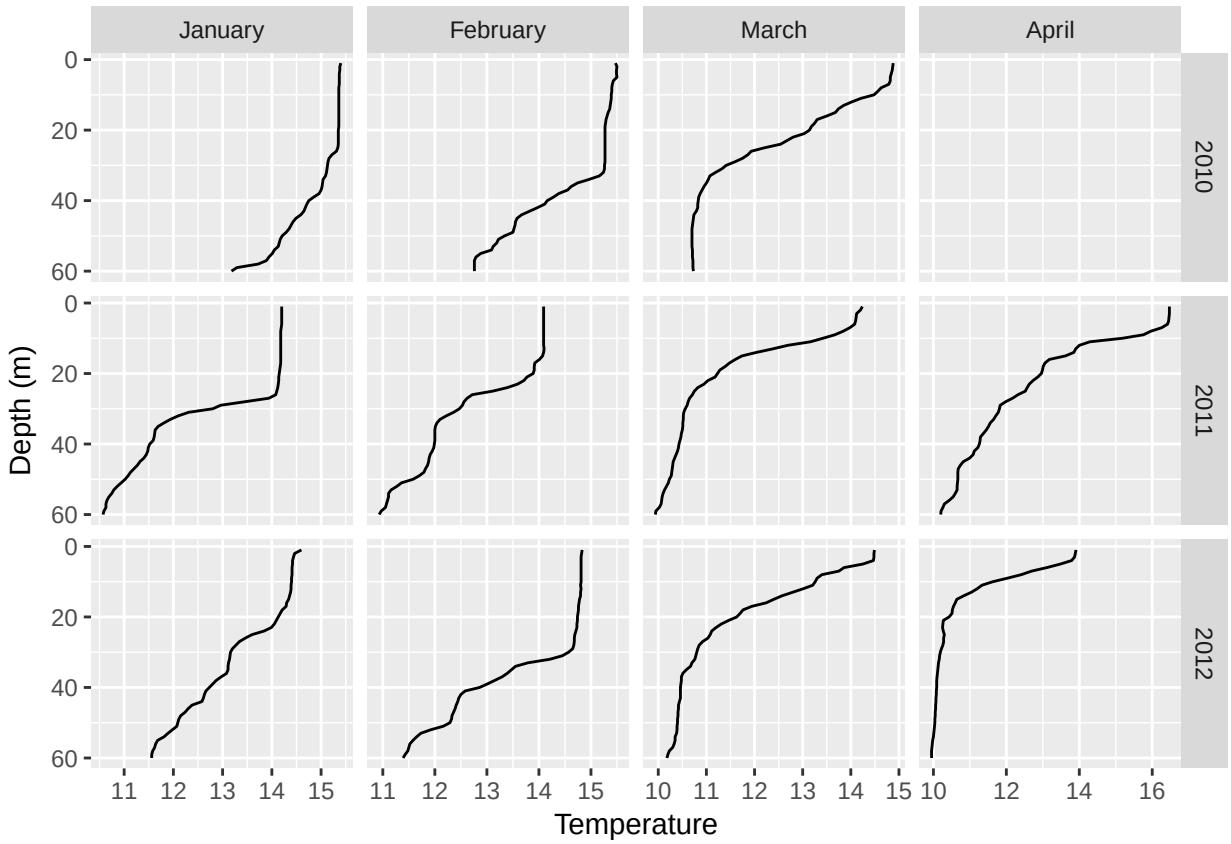
```
df <- subset(ctd, year %in% 2010:2012 & month %in% c("January", "February", "March", "April") & station
df <- df[order(df$year, df$month, df$depth), ]
p <- ggplot(df, mapping = aes(x = temp, y = depth)) +
  geom_path() +
  scale_y_reverse() +
  facet_grid(year ~ month, scales = "free_x") +
  labs(x = "Temperature", y = "Depth (m)")
print(p)
```




```
p <- ggplot(df, mapping = aes(x = temp, y = depth)) +
  geom_path() +
  scale_y_reverse() +
  facet_grid(year ~ month, scales = "free_y") +
  labs(x = "Temperature", y = "Depth (m)")
print(p)
```



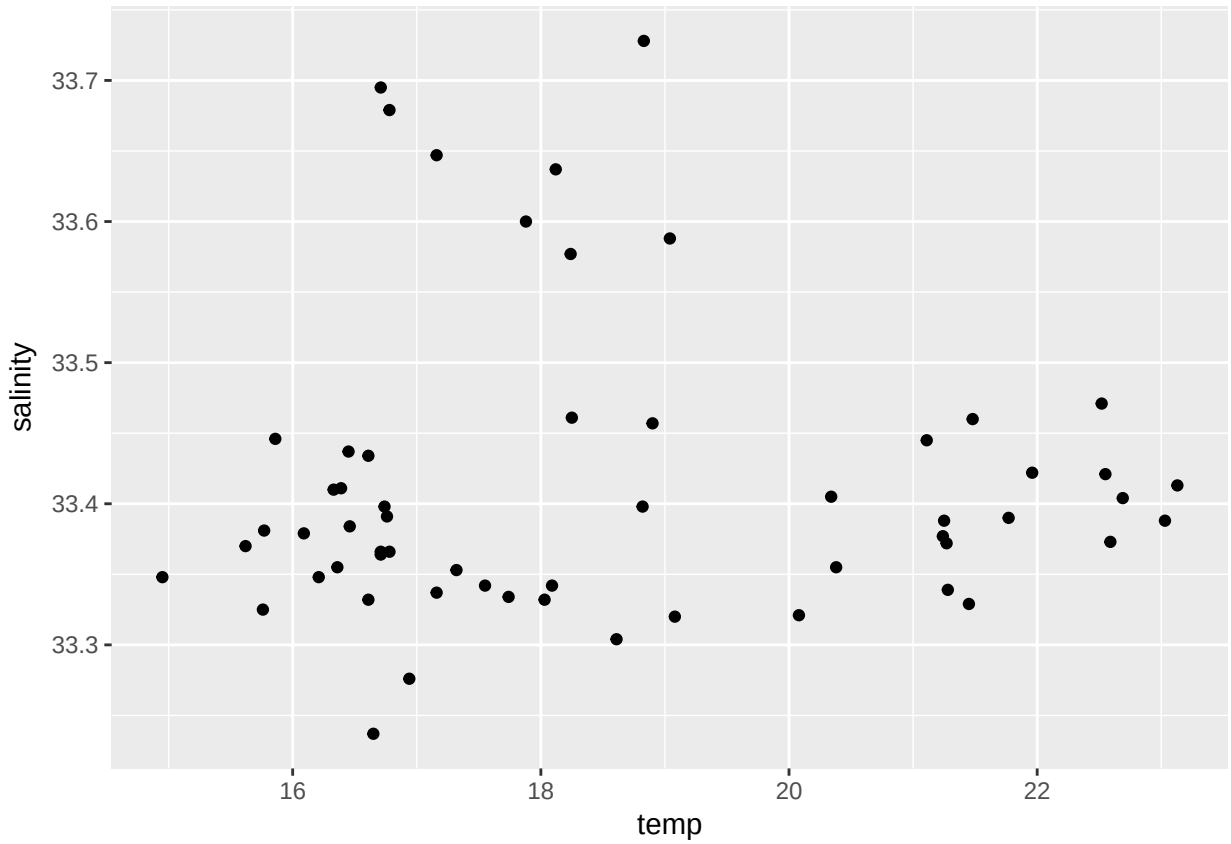

```r
p <- ggplot(df, mapping = aes(x = temp, y = depth)) +
  geom_path() +
  scale_y_reverse() +
  facet_grid(year ~ month, scales = "free") +
  labs(x = "Temperature", y = "Depth (m)")
print(p)
```



Grouping

Within a single plot, groups can be specified with the `group`, `color`, or `fill` arguments. Which one is used depends on the type of plot being created. For example, let's look at the surface temperatures and salinities for Station.39 in 2015:

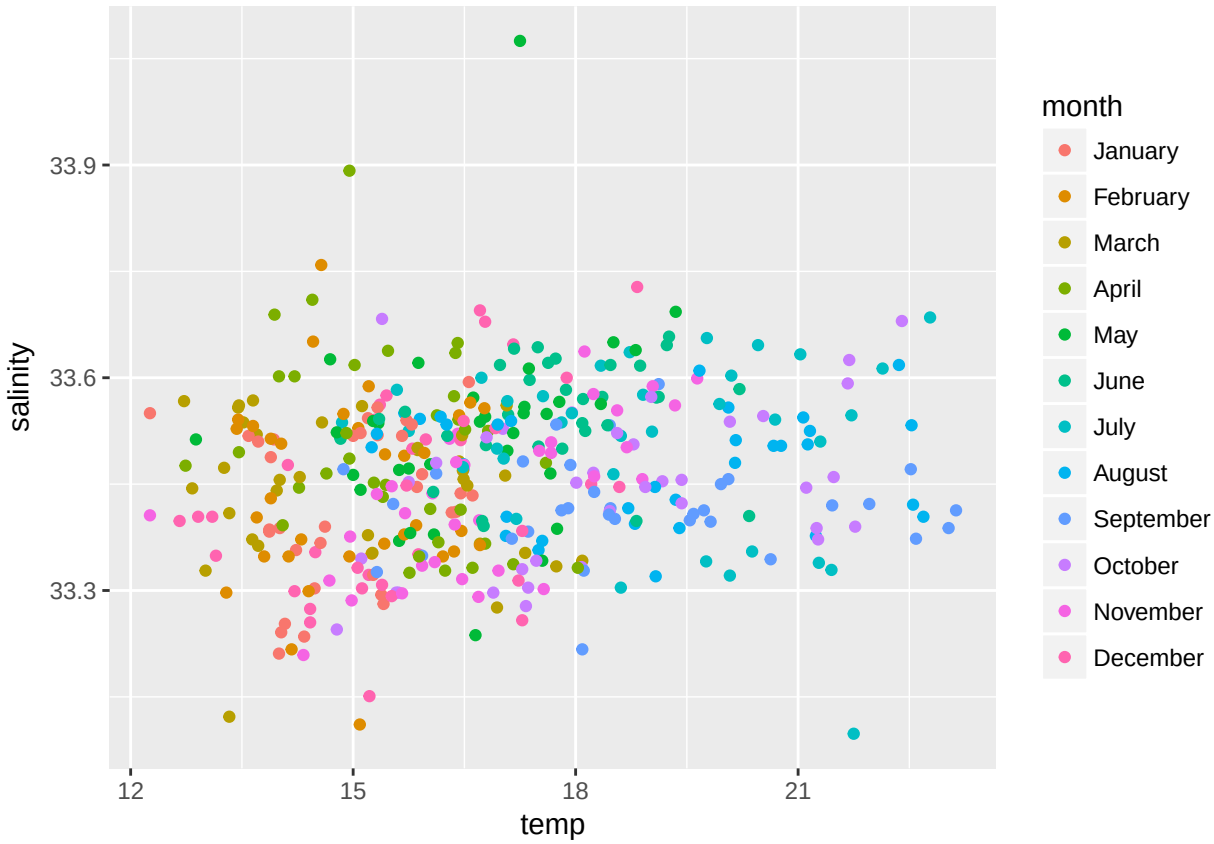
```
df <- subset(ctd, station == "Station.39" & depth == 1 & year == 2015)
p <- ggplot(df, aes(temp, salinity)) +
  geom_point()
print(p)
```



Let's color these by month:

```
df <- subset(ctd, station == "Station.39" & depth == 1)
p <- ggplot(df, aes(temp, salinity, color = month)) +
  geom_point()
print(p)
```

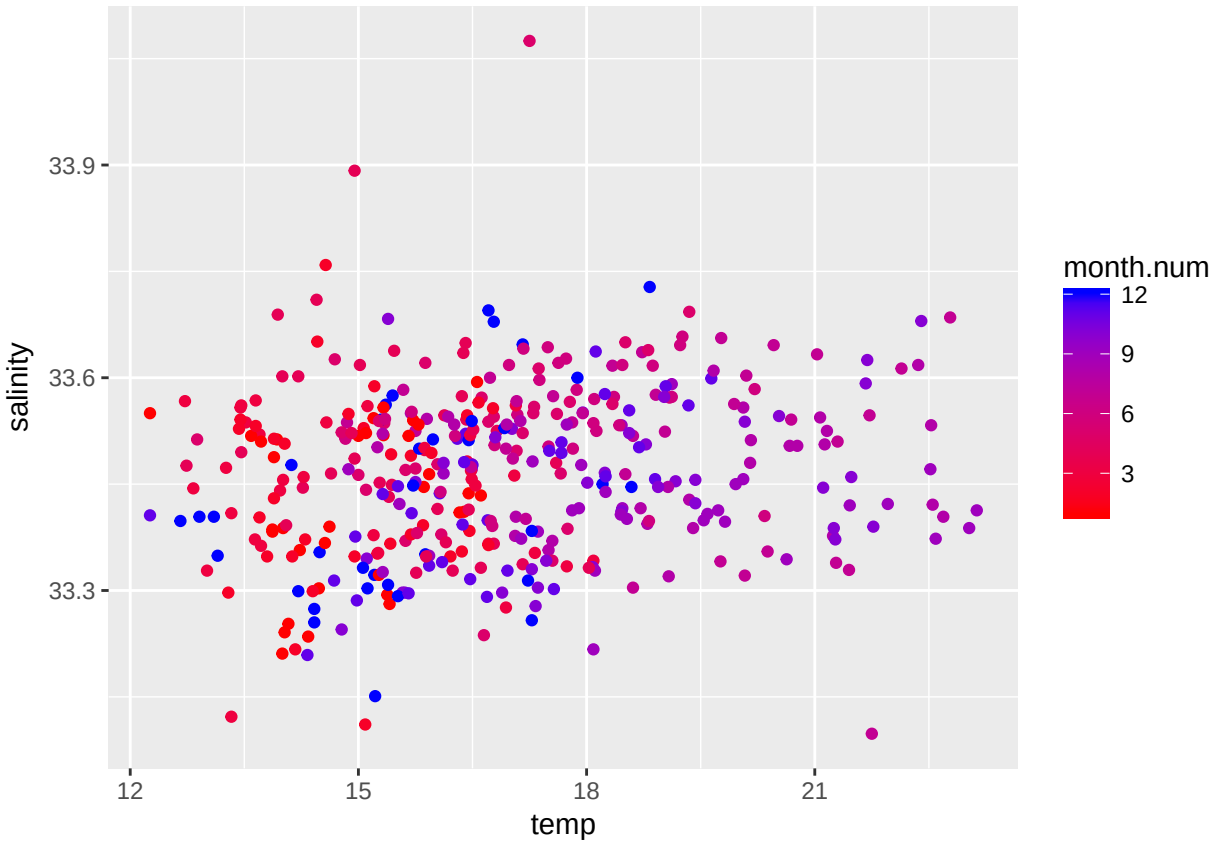
```
Warning: Removed 4 rows containing missing values (geom_point).
```



Because month is a number, a continuous scale is used. We can chane the nature of this scale using `scale_color_gradient`:

```
df$month.num <- as.numeric(df$month)
p <- ggplot(df, aes(temp, salinity, color = month.num)) +
  geom_point() +
  scale_color_gradient(low = "red", high = "blue")
print(p)
```

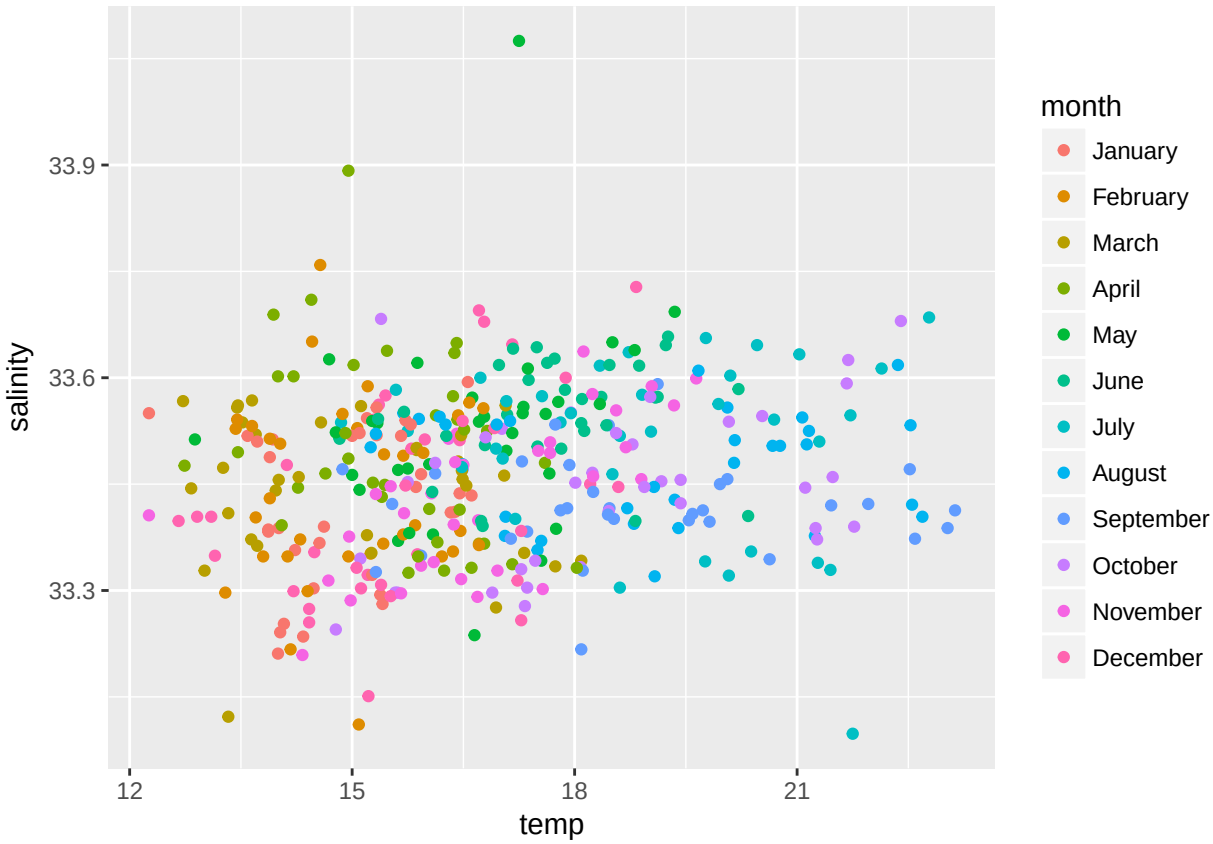
```
Warning: Removed 4 rows containing missing values (geom_point).
```



If month is a factor, then the default plot looks like this, where each month gets its own color:

```
p <- ggplot(df, aes(temp, salinity, color = month)) +
  geom_point()
print(p)
```

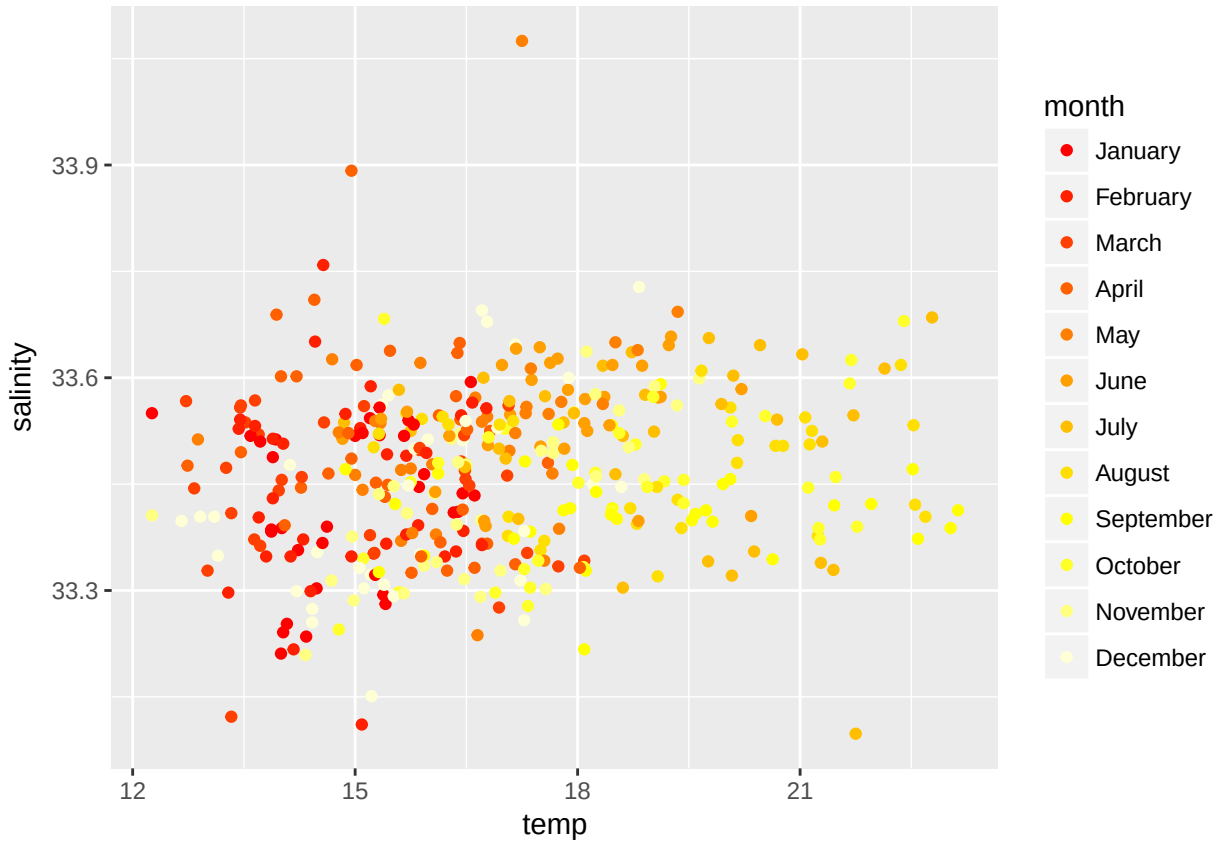
```
Warning: Removed 4 rows containing missing values (geom_point).
```



To change the colors of a factor, I recommend using the RColorBrewer palettes (see http://colorbrewer2.org) and `scale_color_brewer`. However, for 12 months, let's use the built-in `heat.colors` palette and specify it with `scale_color_manual`:

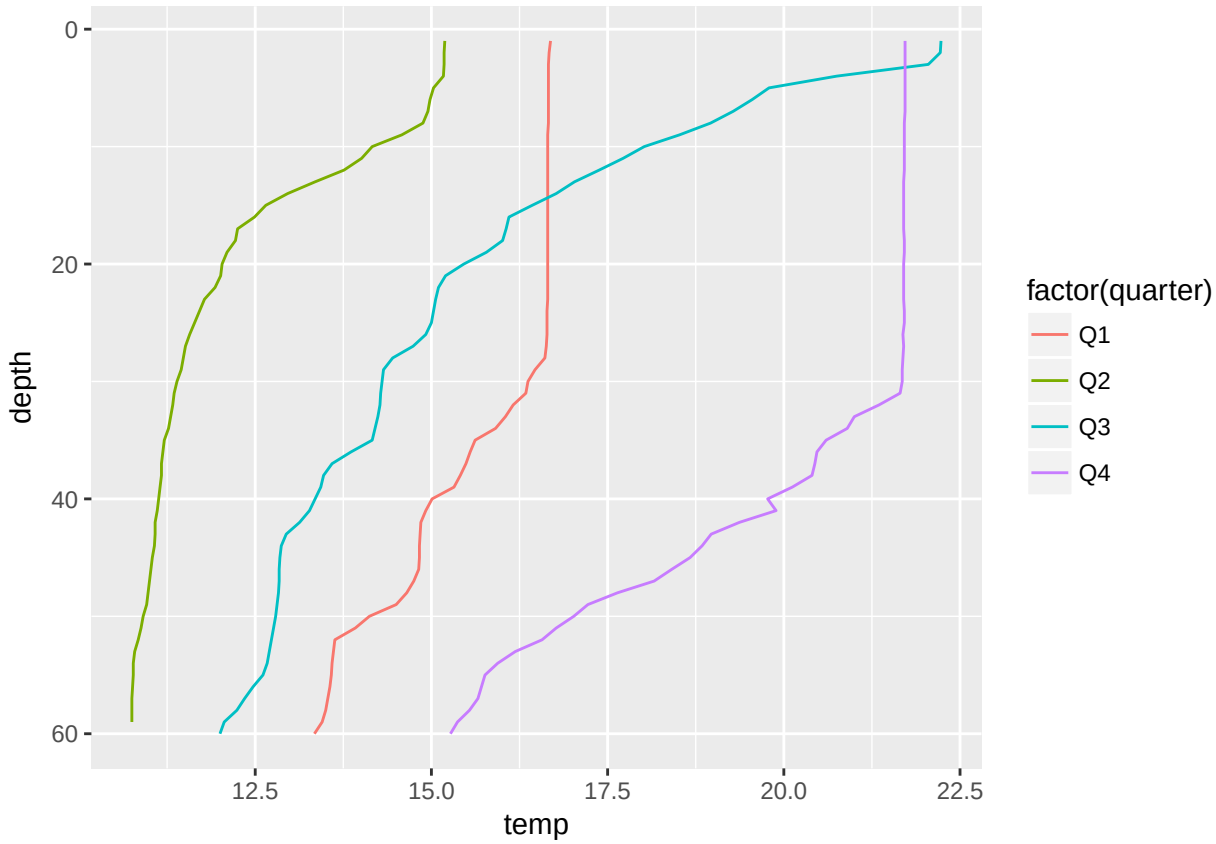
```
p <- ggplot(df, aes(temp, salinity, color = month)) +
  geom_point() +
  scale_color_manual(values = heat.colors(12))
print(p)
```

```
Warning: Removed 4 rows containing missing values (geom_point).
```



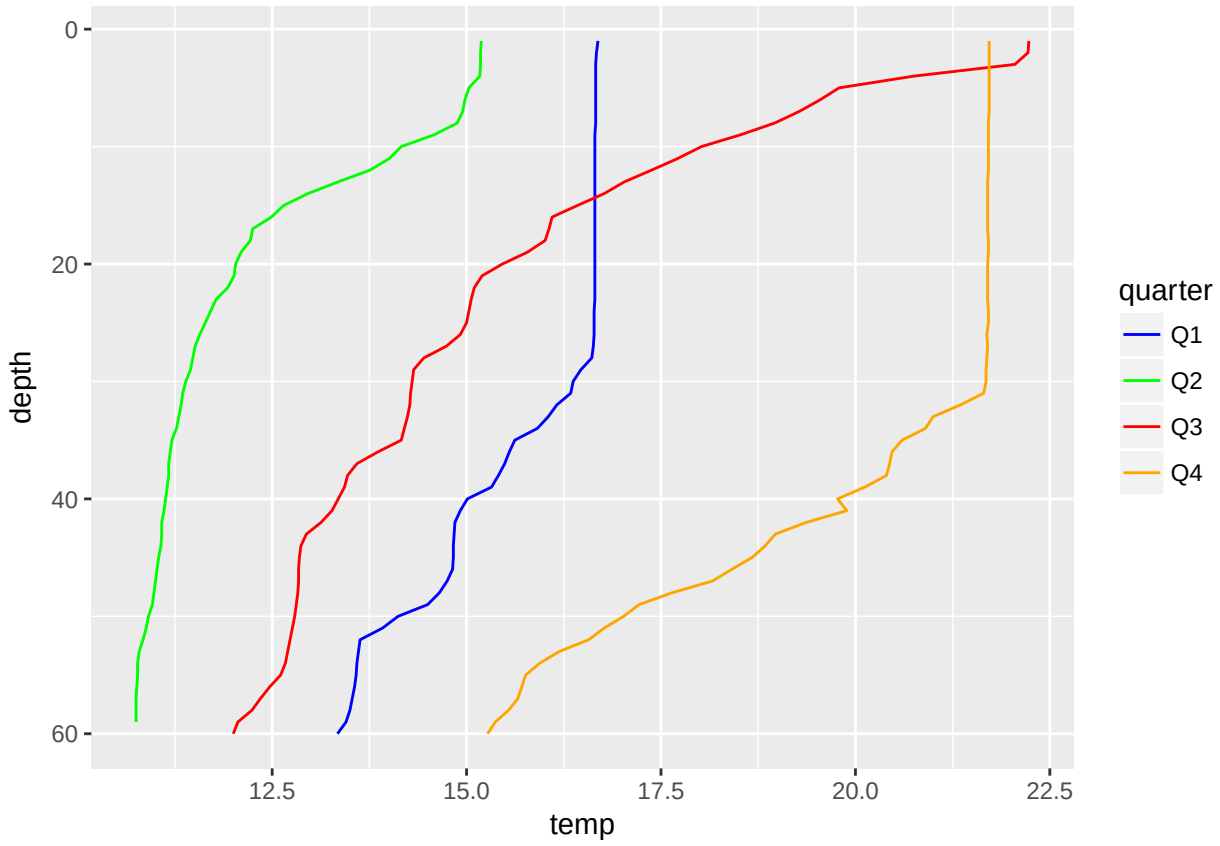
Colors can also be specified by category. For instance, let's plot temperature profiles for Station.39 in 2015 for each quarter:

```
df <- subset(ctd, station == "Station.1" & year == 2015)
df <- df[order(df$month, df$depth), ]
p <- ggplot(df, aes(temp, depth, color = factor(quarter))) +
  geom_path() +
  scale_y_reverse()
print(p)
```



Let's make each quarter a factor and manually set the colors:

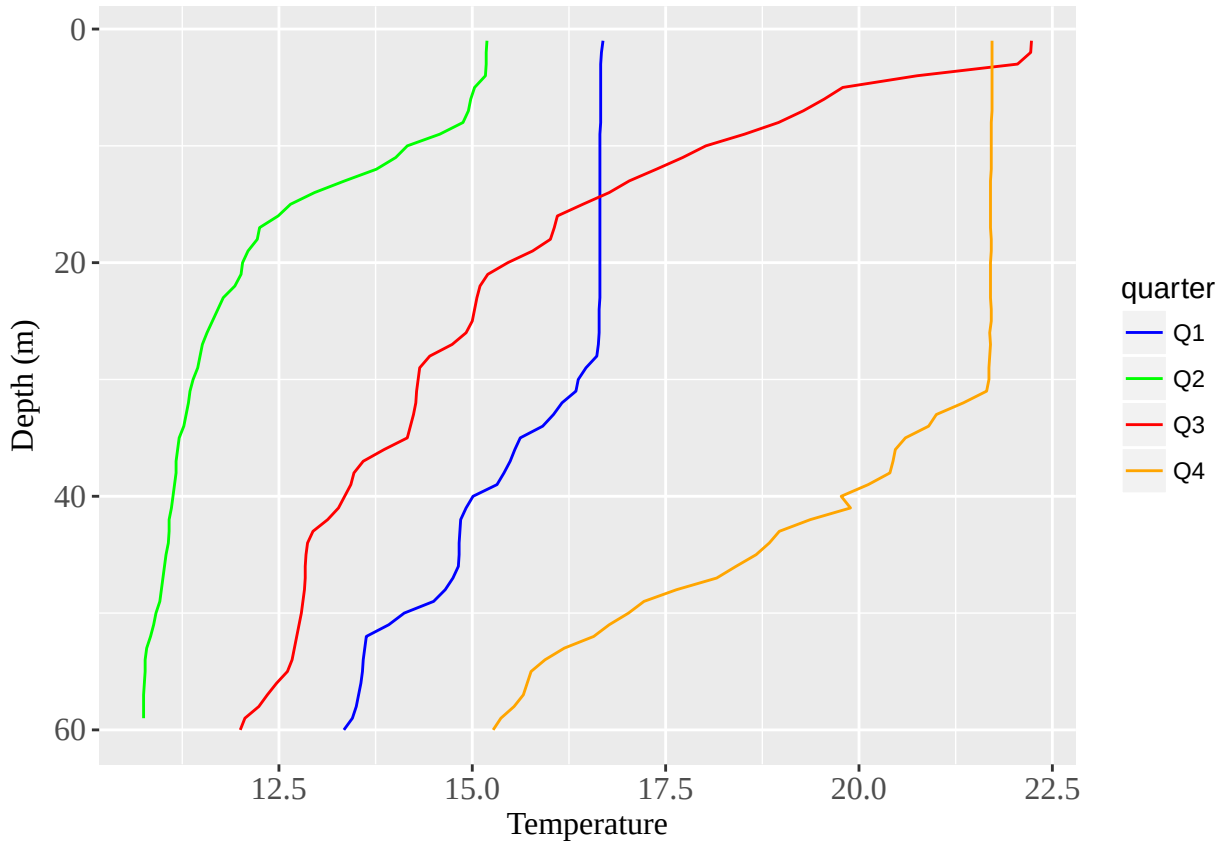
```
q.colors <- c(Q3 = "red", Q4 = "orange", Q1 = "blue", Q2 = "green")
p <- ggplot(df, aes(temp, depth, color = quarter)) +
  geom_path() +
  scale_y_reverse() +
  scale_color_manual(values = q.colors)
print(p)
```



Themes

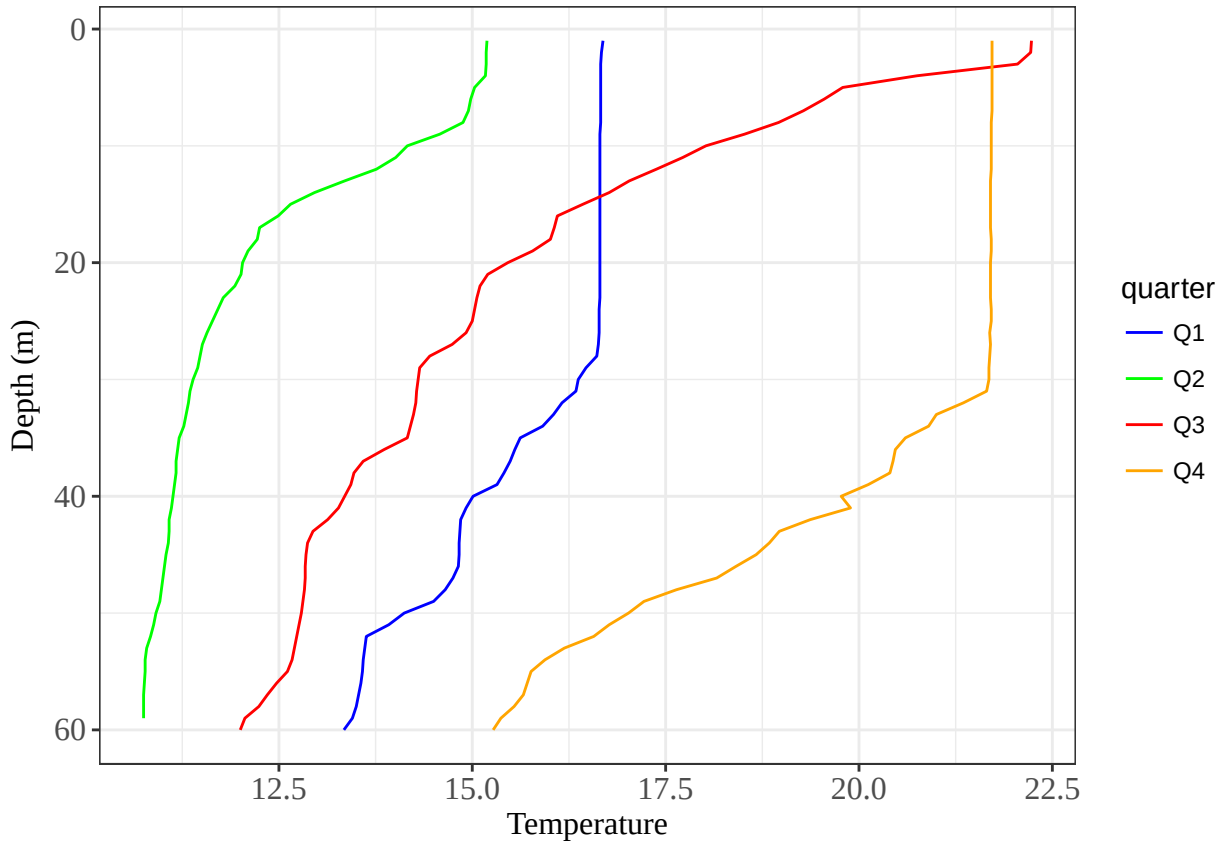
Features of the plot like font sizes, legend position, etc. can be specified with `theme`. For example, we'll set the tick labels and axis labels to Times New Roman 12 point:

```
p <- ggplot(df, aes(temp, depth, color = quarter)) +
  geom_path() +
  scale_y_reverse() +
  labs(x = "Temperature", y = "Depth (m)") +
  scale_color_manual(values = q.colors) +
  theme(
    axis.title = element_text(family = "Times", size = 12),
    axis.text = element_text(family = "Times", size = 12)
  )
print(p)
```

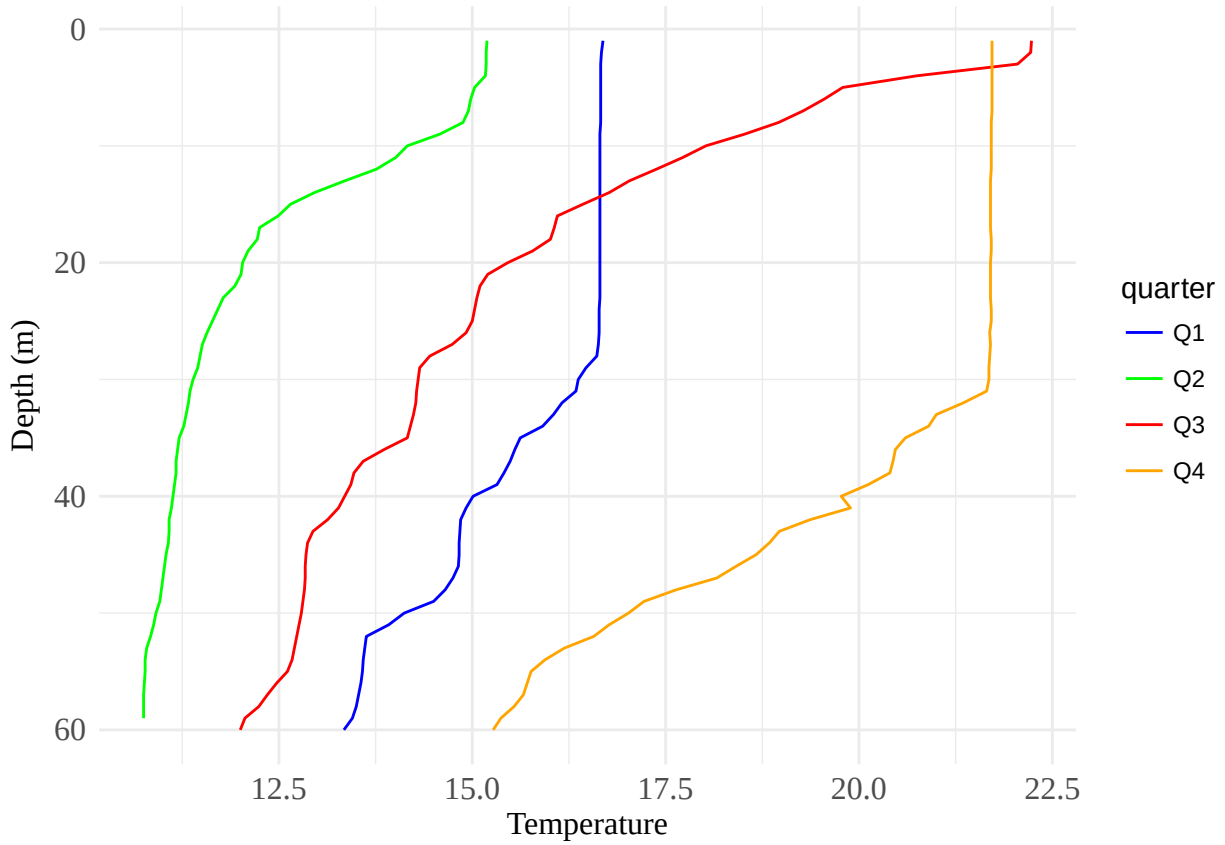


If you don't like the default background shadng and lines and you don't want to hand specify all components, you can try a few of the alternate themes:

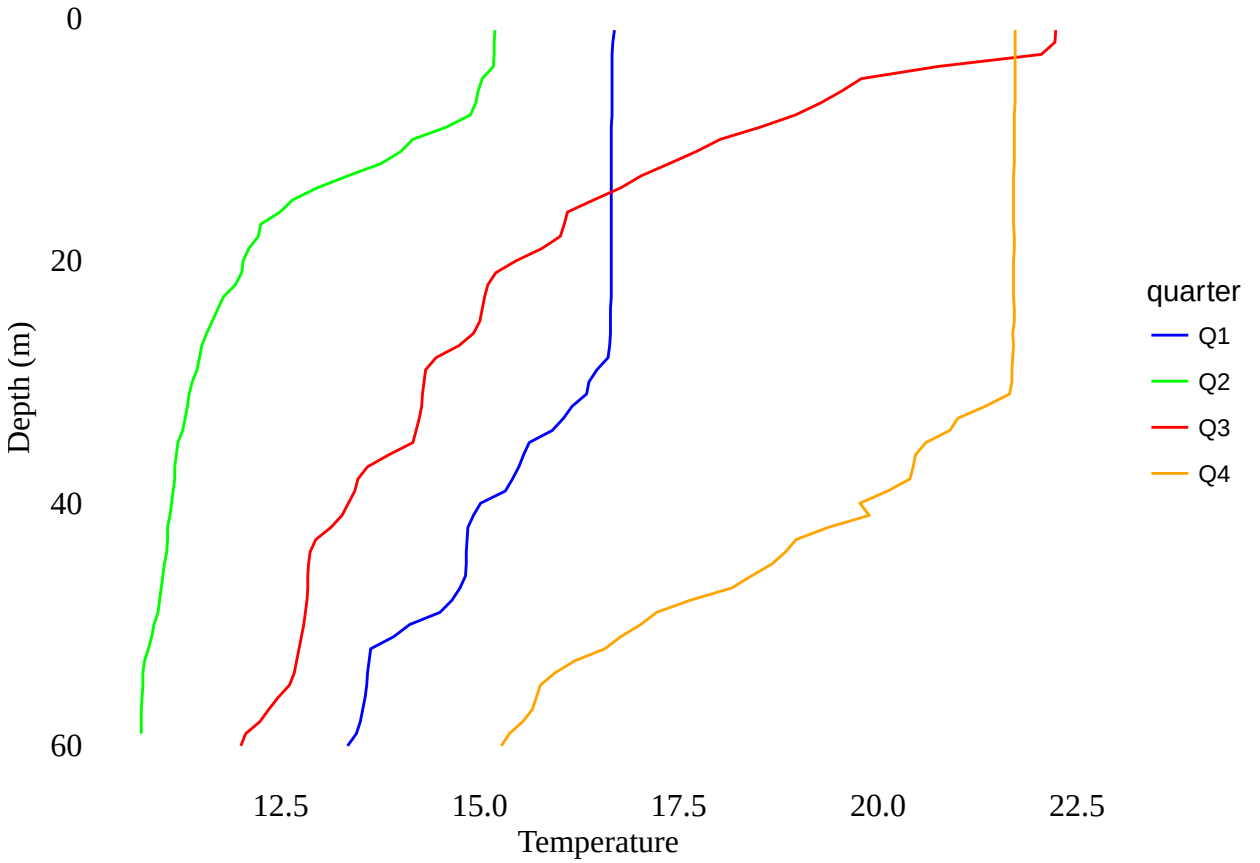
```
p <- ggplot(df, aes(temp, depth, color = quarter)) +
  geom_path() +
  scale_y_reverse() +
  scale_color_manual(values = q.colors) +
  labs(x = "Temperature", y = "Depth (m)") +
  theme_bw() +
  theme(
    axis.title = element_text(family = "Times", size = 12),
    axis.text = element_text(family = "Times", size = 12)
  )
print(p)
```




```
p <- ggplot(df, aes(temp, depth, color = quarter)) +
  geom_path() +
  scale_y_reverse() +
  scale_color_manual(values = q.colors) +
  labs(x = "Temperature", y = "Depth (m)") +
  theme_minimal() +
  theme(
    axis.title = element_text(family = "Times", size = 12),
    axis.text = element_text(family = "Times", size = 12)
  )
print(p)
```




```
p <- ggplot(df, aes(temp, depth, color = quarter)) +
  geom_path() +
  scale_y_reverse() +
  scale_color_manual(values = q.colors) +
  labs(x = "Temperature", y = "Depth (m)") +
  theme_void() +
  theme(
    axis.title = element_text(family = "Times", size = 12),
    axis.text = element_text(family = "Times", size = 12)
  )
print(p)
```



---

dplyr and tidyr

Piping

```
# method one of doing three steps (sequential)
x <- runif(100)
x.q <- quantile(x, c(0.025, 0.975))
x.q.diff.1 <- diff(x.q)

# method two (nested)
x.q.diff.2 <- diff(quantile(runif(100), c(0.025, 0.975)))
```

Piping (magrittr) - %>%

```
library(magrittr)
runif(10)
```

```
 [1] 0.08342796 0.04200117 0.10965631 0.82979056 0.93038856 0.12707832
 [7] 0.15356792 0.38120820 0.52753004 0.55908553
```

```
10 %>% runif()
```

```
 [1] 0.75213576 0.14476405 0.02620817 0.50441439 0.33466675 0.20197012
 [7] 0.40722347 0.34252022 0.66795336 0.16848838
```

```r
# no parentheses needed if left side is all that is going into function
10 %>% runif
```

```
 [1] 0.97632142 0.80558091 0.13460272 0.62249252 0.84254622 0.64118755
 [7] 0.08908386 0.92919316 0.10543350 0.88223604
```

```r
# using arguments
10 %>% runif(100, 200)
```

```
 [1] 118.6036 160.5716 178.8871 179.5806 161.9392 140.7162 102.4584
 [8] 144.0024 178.8518 103.4132
```

```r
# pipe to second argument (must name arguments)
100 %>% runif(n = 5, max = 200)
```

```
[1] 158.4218 123.2533 168.1198 111.5263 116.0822
```

```r
# vs...
100 %>% runif(5, 200)
```

```
  [1] 194.886039 149.546862 177.484848 152.453773 175.084856 148.140255
  [7]  41.043549 106.752346 154.213134 147.134667 178.957743   9.181060
 [13] 194.045230 183.290765 137.340737  60.426010  21.895349 168.263214
 [19] 186.665495 153.855491 194.371116  53.384942 188.551015  21.340513
 [25]  68.357917  94.324325  58.971277 192.264827 169.223939 135.200443
 [31] 114.897691 156.652961  54.814202 104.188834 178.170826  15.728422
 [37] 117.880833  64.433474  60.997774  83.586533 162.576283  24.372688
 [43]   8.416461  82.631455 191.034426  92.660237 115.630536 143.078044
 [49] 154.543380  81.001755 169.741674 193.770896  42.433962  61.927923
 [55] 107.747835 155.841948 192.412448  75.741526 146.121962  46.465702
 [61]  85.247862 126.781763  11.002255 110.453120 192.765424  84.339113
 [67]  47.689127  16.615393 162.688038 123.980377  73.284572 186.389238
 [73]  34.240899  50.237139 130.254195 143.430311 165.308153 163.864565
 [79] 114.207094 199.504269  33.494407 143.083192 163.769215 156.618886
 [85]  83.650480 156.313297  39.602187 165.989817 190.723507  64.874633
 [91] 127.579466   7.991954 187.266318 150.976815  79.908858 195.989918
 [97]  17.558056  36.798338 167.556485  41.238267
```

pipe version of first example

```r
q.diff.pipe <- 100 %>%
  runif %>%
  quantile(c(0.025, 0.975)) %>%
  diff
```

dplyr

filter and select

```r
suppressMessages(library("tidyverse"))
library(tidyverse)

# base R indexing to select males
#starwars[starwars$gender == "male", ]
#subset(starwars, gender == "male")
```

```r
# dplyr way - filter
filter(starwars, gender == "male")
```

```
# A tibble: 62 x 13
   name     height  mass hair_color skin_color eye_color birth_year gender
   <chr>     <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr>
 1 Luke Sk~    172    77 blond      fair       blue            19   male
 2 Darth V~    202   136 none       white      yellow          41.9 male
 3 Owen La~    178   120 brown, gr~ light      blue            52   male
 4 Biggs D~    183    84 black      light      brown           24   male
 5 Obi-Wan~    182    77 auburn, w~ fair       blue-gray       57   male
 6 Anakin ~    188    84 blond      fair       blue            41.9 male
 7 Wilhuff~    180    NA auburn, g~ fair       blue            64   male
 8 Chewbac~    228   112 brown      unknown    blue           200   male
 9 Han Solo    180    80 brown      fair       brown           29   male
10 Greedo      173    74 <NA>       green      black           44   male
# ... with 52 more rows, and 5 more variables: homeworld <chr>,
#   species <chr>, films <list>, vehicles <list>, starships <list>
```

```r
# pipeline version
starwars %>%
  filter(gender == "male" & height > 190)
```

```
# A tibble: 20 x 13
   name    height  mass hair_color skin_color  eye_color birth_year gender
   <chr>    <int> <dbl> <chr>      <chr>       <chr>          <dbl> <chr>
 1 Darth ~    202   136 none       white       yellow          41.9 male
 2 Chewba~    228   112 brown      unknown     blue           200   male
 3 Qui-Go~    193    89 brown      fair        blue            92   male
 4 Nute G~    191    90 none       mottled gr~ red             NA   male
 5 Jar Ja~    196    66 none       orange      orange          52   male
 6 Roos T~    224    82 none       grey        orange          NA   male
 7 Rugor ~    206    NA none       green       orange          NA   male
 8 Ki-Adi~    198    82 white      pale        yellow          92   male
 9 Kit Fi~    196    87 none       green       black           NA   male
10 Yarael~    264    NA none       white       yellow          NA   male
11 Mas Am~    196    NA none       blue        blue            NA   male
12 Dooku      193    80 white      fair        brown          102   male
13 Bail P~    191    NA black      tan         brown           67   male
14 Dexter~    198   102 none       brown       yellow          NA   male
15 Lama Su    229    88 none       grey        black           NA   male
16 Wat Ta~    193    48 none       green, grey unknown         NA   male
17 San Hi~    191    NA none       grey        gold            NA   male
18 Grievo~    216   159 none       brown, whi~ green, y~       NA   male
19 Tarfful    234   136 brown      brown       blue            NA   male
20 Tion M~    206    80 none       grey        black           NA   male
# ... with 5 more variables: homeworld <chr>, species <chr>, films <list>,
#   vehicles <list>, starships <list>
```

```r
# "select" columns to return
select(starwars, name, height, mass, gender)
```

```
# A tibble: 87 x 4
   name               height  mass gender
   <chr>               <int> <dbl> <chr>
```

```
 1 Luke Skywalker        172    77 male
 2 C-3PO                 167    75 <NA>
 3 R2-D2                  96    32 <NA>
 4 Darth Vader           202   136 male
 5 Leia Organa           150    49 female
 6 Owen Lars             178   120 male
 7 Beru Whitesun lars    165    75 female
 8 R5-D4                  97    32 <NA>
 9 Biggs Darklighter     183    84 male
10 Obi-Wan Kenobi        182    77 male
# ... with 77 more rows
```

```
select(starwars, height, gender, name, mass)
```

```
# A tibble: 87 x 4
   height gender name                mass
    <int> <chr>  <chr>              <dbl>
 1    172 male   Luke Skywalker        77
 2    167 <NA>   C-3PO                 75
 3     96 <NA>   R2-D2                 32
 4    202 male   Darth Vader          136
 5    150 female Leia Organa           49
 6    178 male   Owen Lars            120
 7    165 female Beru Whitesun lars    75
 8     97 <NA>   R5-D4                 32
 9    183 male   Biggs Darklighter     84
10    182 male   Obi-Wan Kenobi        77
# ... with 77 more rows
```

```
# extend pipeline above
starwars %>%
  filter(gender == "male" & height > 190) %>%
  select(name, height, mass)
```

```
# A tibble: 20 x 3
   name                height  mass
   <chr>                <int> <dbl>
 1 Darth Vader            202   136
 2 Chewbacca              228   112
 3 Qui-Gon Jinn           193    89
 4 Nute Gunray            191    90
 5 Jar Jar Binks          196    66
 6 Roos Tarpals           224    82
 7 Rugor Nass             206    NA
 8 Ki-Adi-Mundi           198    82
 9 Kit Fisto              196    87
10 Yarael Poof            264    NA
11 Mas Amedda             196    NA
12 Dooku                  193    80
13 Bail Prestor Organa    191    NA
14 Dexter Jettster        198   102
15 Lama Su                229    88
16 Wat Tambor             193    48
17 San Hill               191    NA
18 Grievous               216   159
```

```
19 Tarfful                234   136
20 Tion Medon             206    80
```

```
# helper functions for select

# select range of columns
starwars %>%
  filter(gender == "male" & height > 190) %>%
  select(eye_color:homeworld)
```

```
# A tibble: 20 x 4
   eye_color     birth_year gender homeworld
   <chr>              <dbl> <chr>  <chr>
 1 yellow              41.9 male   Tatooine
 2 blue               200   male   Kashyyyk
 3 blue                92   male   <NA>
 4 red                 NA   male   Cato Neimoidia
 5 orange              52   male   Naboo
 6 orange              NA   male   Naboo
 7 orange              NA   male   Naboo
 8 yellow              92   male   Cerea
 9 black               NA   male   Glee Anselm
10 yellow              NA   male   Quermia
11 blue                NA   male   Champala
12 brown              102   male   Serenno
13 brown               67   male   Alderaan
14 yellow              NA   male   Ojom
15 black               NA   male   Kamino
16 unknown             NA   male   Skako
17 gold                NA   male   Muunilinst
18 green, yellow       NA   male   Kalee
19 blue                NA   male   Kashyyyk
20 black               NA   male   Utapau
```

```
# select columns that start with string
starwars %>%
  filter(gender == "male" & height > 190) %>%
  select(starts_with("h"))
```

```
# A tibble: 20 x 3
   height hair_color homeworld
    <int> <chr>      <chr>
 1    202 none       Tatooine
 2    228 brown      Kashyyyk
 3    193 brown      <NA>
 4    191 none       Cato Neimoidia
 5    196 none       Naboo
 6    224 none       Naboo
 7    206 none       Naboo
 8    198 white      Cerea
 9    196 none       Glee Anselm
10    264 none       Quermia
11    196 none       Champala
12    193 white      Serenno
13    191 black      Alderaan
14    198 none       Ojom
```

```
15    229 none       Kamino
16    193 none       Skako
17    191 none       Muunilinst
18    216 none       Kalee
19    234 brown      Kashyyyk
20    206 none       Utapau
```

```
# select columns that contain a string
starwars %>%
  filter(gender == "male" & height > 190) %>%
  select(contains("color"))
```

```
# A tibble: 20 x 3
   hair_color skin_color    eye_color
   <chr>      <chr>         <chr>
 1 none       white         yellow
 2 brown      unknown       blue
 3 brown      fair          blue
 4 none       mottled green red
 5 none       orange        orange
 6 none       grey          orange
 7 none       green         orange
 8 white      pale          yellow
 9 none       green         black
10 none       white         yellow
11 none       blue          blue
12 white      fair          brown
13 black      tan           brown
14 none       brown         yellow
15 none       grey          black
16 none       green, grey   unknown
17 none       grey          gold
18 none       brown, white  green, yellow
19 brown      brown         blue
20 none       grey          black
```

```
# select columns excluding certain ones
starwars %>%
  filter(gender == "male" & height > 190) %>%
  select(-name, -gender, -height)
```

```
# A tibble: 20 x 10
    mass hair_color skin_color   eye_color   birth_year homeworld  species
   <dbl> <chr>      <chr>        <chr>            <dbl> <chr>      <chr>
 1   136 none       white        yellow            41.9 Tatooine   Human
 2   112 brown      unknown      blue             200   Kashyyyk   Wookiee
 3    89 brown      fair         blue              92   <NA>       Human
 4    90 none       mottled gre~ red               NA   Cato Neim~ Neimod~
 5    66 none       orange       orange            52   Naboo      Gungan
 6    82 none       grey         orange            NA   Naboo      Gungan
 7    NA none       green        orange            NA   Naboo      Gungan
 8    82 white      pale         yellow            92   Cerea      Cerean
 9    87 none       green        black             NA   Glee Anse~ Nautol~
10    NA none       white        yellow            NA   Quermia    Quermi~
11    NA none       blue         blue              NA   Champala   Chagri~
12    80 white      fair         brown            102   Serenno    Human
```

```
13    NA black      tan          brown             67   Alderaan   Human
14   102 none       brown        yellow            NA   Ojom       Besali~
15    88 none       grey         black             NA   Kamino     Kamino~
16    48 none       green, grey  unknown           NA   Skako      Skakoan
17    NA none       grey         gold              NA   Muunilinst Muun
18   159 none       brown, white green, yel~       NA   Kalee      Kaleesh
19   136 brown      brown        blue              NA   Kashyyyk   Wookiee
20    80 none       grey         black             NA   Utapau     Pau'an
# ... with 3 more variables: films <list>, vehicles <list>,
#   starships <list>
```

arrange to sort data

```
# base R sorting a data.frame
starwars[order(starwars$species, starwars$height), ]
```

```
# A tibble: 87 x 13
   name    height  mass hair_color skin_color  eye_color birth_year gender
   <chr>    <int> <dbl> <chr>      <chr>       <chr>           <dbl> <chr>
 1 Ratts ~     79    15 none       grey, blue  unknown            NA male
 2 Dexter~    198   102 none       brown       yellow             NA male
 3 Ki-Adi~    198    82 white      pale        yellow             92 male
 4 Mas Am~    196    NA none       blue        blue               NA male
 5 Zam We~    168    55 blonde     fair, gree~ yellow             NA female
 6 R2-D2       96    32 <NA>       white, blue red                33 <NA>
 7 R5-D4       97    32 <NA>       white, red  red                NA <NA>
 8 C-3PO      167    75 <NA>       gold        yellow            112 <NA>
 9 IG-88      200   140 none       metal       red                15 none
10 BB8         NA    NA none       none        black              NA none
# ... with 77 more rows, and 5 more variables: homeworld <chr>,
#   species <chr>, films <list>, vehicles <list>, starships <list>
```

```
# arrange
starwars %>%
  arrange(species, desc(height)) %>%
  select(name, height, species)
```

```
# A tibble: 87 x 3
   name            height species
   <chr>            <int> <chr>
 1 Ratts Tyerell       79 Aleena
 2 Dexter Jettster    198 Besalisk
 3 Ki-Adi-Mundi       198 Cerean
 4 Mas Amedda         196 Chagrian
 5 Zam Wesell         168 Clawdite
 6 IG-88              200 Droid
 7 C-3PO              167 Droid
 8 R5-D4               97 Droid
 9 R2-D2               96 Droid
10 BB8                 NA Droid
# ... with 77 more rows
```

new columns

```
sw <- starwars %>%
  mutate(
    height.m = height / 100,
```

```
    bmi = mass / height.m ^ 2
  )

# takes place of
# sw <- starwars
# sw$height.m <- sw$height / 100
# sw$bmi <- sw$mass / sw$height.m ^ 2
```

change name of column

```
sw <- starwars %>%
  rename(handle = "name")
colnames(starwars)
```

```
 [1] "name"       "height"     "mass"       "hair_color" "skin_color"
 [6] "eye_color"  "birth_year" "gender"     "homeworld"  "species"
[11] "films"      "vehicles"   "starships"
```

```
colnames(sw)
```

```
 [1] "handle"     "height"     "mass"       "hair_color" "skin_color"
 [6] "eye_color"  "birth_year" "gender"     "homeworld"  "species"
[11] "films"      "vehicles"   "starships"
```

create new column and drop all others

```
sw <- starwars %>%
  transmute(
    name = name,
    height.m = height / 100,
    bmi = mass / height.m ^ 2
  )
sw
```

```
# A tibble: 87 x 3
   name               height.m   bmi
   <chr>                 <dbl> <dbl>
 1 Luke Skywalker         1.72  26.0
 2 C-3PO                  1.67  26.9
 3 R2-D2                  0.96  34.7
 4 Darth Vader            2.02  33.3
 5 Leia Organa            1.5   21.8
 6 Owen Lars              1.78  37.9
 7 Beru Whitesun lars     1.65  27.5
 8 R5-D4                  0.97  34.0
 9 Biggs Darklighter      1.83  25.1
10 Obi-Wan Kenobi         1.82  23.2
# ... with 77 more rows
```

```
# same as
sw <- starwars %>%
  mutate(
    height.m = height / 100,
    bmi = mass / height.m ^ 2
  ) %>%
  select(height.m, bmi)
sw
```

```
# A tibble: 87 x 2
   height.m   bmi
      <dbl> <dbl>
 1     1.72  26.0
 2     1.67  26.9
 3     0.96  34.7
 4     2.02  33.3
 5     1.5   21.8
 6     1.78  37.9
 7     1.65  27.5
 8     0.97  34.0
 9     1.83  25.1
10     1.82  23.2
# ... with 77 more rows
```

complete data set (no missing data)

```
# in base R
#sw.complete <- starwars[complete.cases(starwars), ]

sw.complete <- starwars %>%
  select(-(films:starships), -mass) %>%
  filter(complete.cases(.))

nrow(starwars)
```

```
[1] 87
```

```
nrow(sw.complete)
```

```
[1] 35
```

```
sw.complete
```

```
# A tibble: 35 x 9
   name          height hair_color  skin_color eye_color birth_year gender
   <chr>          <int> <chr>       <chr>      <chr>          <dbl> <chr>
 1 Luke Skywalk~    172 blond       fair       blue            19   male
 2 Darth Vader      202 none        white      yellow          41.9 male
 3 Leia Organa      150 brown       light      brown           19   female
 4 Owen Lars        178 brown, grey light      blue            52   male
 5 Beru Whitesu~    165 brown       light      blue            47   female
 6 Biggs Darkli~    183 black       light      brown           24   male
 7 Obi-Wan Keno~    182 auburn, wh~ fair       blue-gray       57   male
 8 Anakin Skywa~    188 blond       fair       blue            41.9 male
 9 Wilhuff Tark~    180 auburn, gr~ fair       blue            64   male
10 Chewbacca        228 brown       unknown    blue           200   male
# ... with 25 more rows, and 2 more variables: homeworld <chr>,
#   species <chr>
```

removing duplicates

```
# what are the observed combinations of gender and species
starwars %>%
  select(gender, species) %>%
  distinct() %>%
  arrange(species, gender)
```

```
# A tibble: 43 x 2
   gender species
   <chr>  <chr>
 1 male   Aleena
 2 male   Besalisk
 3 male   Cerean
 4 male   Chagrian
 5 female Clawdite
 6 none   Droid
 7 <NA>   Droid
 8 male   Dug
 9 male   Ewok
10 male   Geonosian
# ... with 33 more rows
```

select random rows

```
# without replacement
starwars %>%
  sample_n(10)
```

```
# A tibble: 10 x 13
   name    height  mass hair_color  skin_color eye_color birth_year gender
   <chr>    <int> <dbl> <chr>       <chr>      <chr>          <dbl> <chr>
 1 Plo Ko~    188  80   none        orange     black             22 male
 2 Tarfful    234 136   brown       brown      blue              NA male
 3 Shaak ~    178  57   none        red, blue~ black             NA female
 4 Han So~    180  80   brown       fair       brown             29 male
 5 Darth ~    175  80   none        red        yellow            54 male
 6 Biggs ~    183  84   black       light      brown             24 male
 7 Lumina~    170  56.2 black       yellow     blue              58 female
 8 Jango ~    183  79   black       tan        brown             66 male
 9 Wilhuf~    180  NA   auburn, gr~ fair       blue              64 male
10 Lama Su    229  88   none        grey       black             NA male
# ... with 5 more variables: homeworld <chr>, species <chr>, films <list>,
#   vehicles <list>, starships <list>
```

```
# with replacement
starwars %>%
  sample_n(10, weight = sample(1:10, nrow(.), replace = T))
```

```
# A tibble: 10 x 13
   name    height  mass hair_color skin_color  eye_color birth_year gender
   <chr>    <int> <dbl> <chr>      <chr>       <chr>          <dbl> <chr>
 1 Bail P~    191    NA black      tan         brown             67 male
 2 Greedo     173    74 <NA>       green       black             44 male
 3 Gregar~    185    85 black      dark        brown             NA male
 4 Kit Fi~    196    87 none       green       black             NA male
 5 Chewba~    228   112 brown      unknown     blue             200 male
 6 Plo Ko~    188    80 none       orange      black             22 male
 7 Nien N~    160    68 none       grey        black             NA male
 8 Han So~    180    80 brown      fair        brown             29 male
 9 Zam We~    168    55 blonde     fair, gree~ yellow            NA female
10 Darth ~    175    80 none       red         yellow            54 male
# ... with 5 more variables: homeworld <chr>, species <chr>, films <list>,
#   vehicles <list>, starships <list>
```

group_by

```
sw <- starwars %>%
  group_by(species) %>%
  summarize(
    mean.height = mean(height, na.rm = T),
    mean.mass = mean(mass, na.rm = T),
    bmi.mean = mean.mass / (mean.height / 100) ^ 2
  )
sw
```

```
# A tibble: 38 x 4
   species   mean.height mean.mass bmi.mean
   <chr>           <dbl>     <dbl>    <dbl>
 1 Aleena             79        15       24.0
 2 Besalisk          198       102       26.0
 3 Cerean            198        82       20.9
 4 Chagrian          196       NaN      NaN
 5 Clawdite          168        55       19.5
 6 Droid             140        69.8     35.6
 7 Dug               112        40       31.9
 8 Ewok               88        20       25.8
 9 Geonosian         183        80       23.9
10 Gungan            209.       74       17.0
# ... with 28 more rows
```

```
sw <- starwars %>%
  group_by(species, gender) %>%
  summarize(
    mean.height = mean(height, na.rm = T),
    mean.mass = mean(mass, na.rm = T),
    bmi.mean = mean.mass / (mean.height / 100) ^ 2
  )
sw
```

```
# A tibble: 43 x 5
# Groups:   species [?]
   species   gender mean.height mean.mass bmi.mean
   <chr>     <chr>        <dbl>     <dbl>    <dbl>
 1 Aleena    male            79        15       24.0
 2 Besalisk  male           198       102       26.0
 3 Cerean    male           198        82       20.9
 4 Chagrian  male           196       NaN      NaN
 5 Clawdite  female         168        55       19.5
 6 Droid     none           200       140       35
 7 Droid     <NA>           120        46.3     32.2
 8 Dug       male           112        40       31.9
 9 Ewok      male            88        20       25.8
10 Geonosian male           183        80       23.9
# ... with 33 more rows
```

```
# same summaries, but with mutate on grouped tibble
sw <- starwars %>%
  group_by(species, gender) %>%
  mutate(
    mean.height = mean(height, na.rm = T),
```

```
    mean.mass = mean(mass, na.rm = T),
    bmi.mean = mean.mass / (mean.height / 100) ^ 2
  )
sw
```

```
# A tibble: 87 x 16
# Groups:   species, gender [43]
   name     height  mass hair_color skin_color eye_color birth_year gender
   <chr>     <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr>
 1 Luke Sk~    172    77 blond      fair       blue            19   male
 2 C-3PO       167    75 <NA>       gold       yellow         112   <NA>
 3 R2-D2        96    32 <NA>       white, bl~ red             33   <NA>
 4 Darth V~    202   136 none       white      yellow          41.9 male
 5 Leia Or~    150    49 brown      light      brown           19   female
 6 Owen La~    178   120 brown, gr~ light      blue            52   male
 7 Beru Wh~    165    75 brown      light      blue            47   female
 8 R5-D4        97    32 <NA>       white, red red             NA   <NA>
 9 Biggs D~    183    84 black      light      brown           24   male
10 Obi-Wan~    182    77 auburn, w~ fair       blue-gray       57   male
# ... with 77 more rows, and 8 more variables: homeworld <chr>,
#   species <chr>, films <list>, vehicles <list>, starships <list>,
#   mean.height <dbl>, mean.mass <dbl>, bmi.mean <dbl>
```

```
# same summaries, but with mutate on grouped tibble
sw <- starwars %>%
  group_by(species, gender) %>%
  mutate(
    mean.height = mean(height, na.rm = T),
    mean.mass = mean(mass, na.rm = T),
    bmi.mean = mean.mass / (mean.height / 100) ^ 2,
    bmi = mass / (height / 100) ^ 2
  )
sw
```

```
# A tibble: 87 x 17
# Groups:   species, gender [43]
   name     height  mass hair_color skin_color eye_color birth_year gender
   <chr>     <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr>
 1 Luke Sk~    172    77 blond      fair       blue            19   male
 2 C-3PO       167    75 <NA>       gold       yellow         112   <NA>
 3 R2-D2        96    32 <NA>       white, bl~ red             33   <NA>
 4 Darth V~    202   136 none       white      yellow          41.9 male
 5 Leia Or~    150    49 brown      light      brown           19   female
 6 Owen La~    178   120 brown, gr~ light      blue            52   male
 7 Beru Wh~    165    75 brown      light      blue            47   female
 8 R5-D4        97    32 <NA>       white, red red             NA   <NA>
 9 Biggs D~    183    84 black      light      brown           24   male
10 Obi-Wan~    182    77 auburn, w~ fair       blue-gray       57   male
# ... with 77 more rows, and 9 more variables: homeworld <chr>,
#   species <chr>, films <list>, vehicles <list>, starships <list>,
#   mean.height <dbl>, mean.mass <dbl>, bmi.mean <dbl>, bmi <dbl>
```

```
# count number of rows in group
num.sp.gend <- starwars %>%
  group_by(species, gender) %>%
```

```
  summarize(num = n())

# fraction of mass of each character
fr.mass <- starwars %>%
  group_by(species) %>%
  mutate(pct.mass = mass / sum(mass, na.rm = TRUE)) %>%
  ungroup %>%
  select(name, pct.mass)
```

Joining

```
bmi <- starwars %>%
  group_by(species) %>%
  summarize(bmi = mean(mass / (height / 100) ^ 2, na.rm = TRUE))

num.tall.characters <- starwars %>%
  filter(height > 150) %>%
  group_by(species) %>%
  summarize(num = n()) %>%
  rename(spp = "species")

num.tall.characters %>%
  left_join(bmi, by = c("spp" = "species"))
```

```
# A tibble: 31 x 3
   spp         num   bmi
   <chr>     <int> <dbl>
 1 Besalisk      1  26.0
 2 Cerean        1  20.9
 3 Chagrian      1 NaN
 4 Clawdite      1  19.5
 5 Droid         2  32.7
 6 Geonosian     1  23.9
 7 Gungan        3  16.8
 8 Human        29  25.5
 9 Hutt          1 443.
10 Iktotchi      1 NaN
# ... with 21 more rows
```

```
final <- starwars %>%
  group_by(species) %>%
  summarize(bmi = mean(mass / (height / 100) ^ 2, na.rm = TRUE)) %>%
  left_join(
    starwars %>%
      filter(height > 150) %>%
      group_by(species) %>%
      summarize(num = n()),
    by = "species"
  )
```

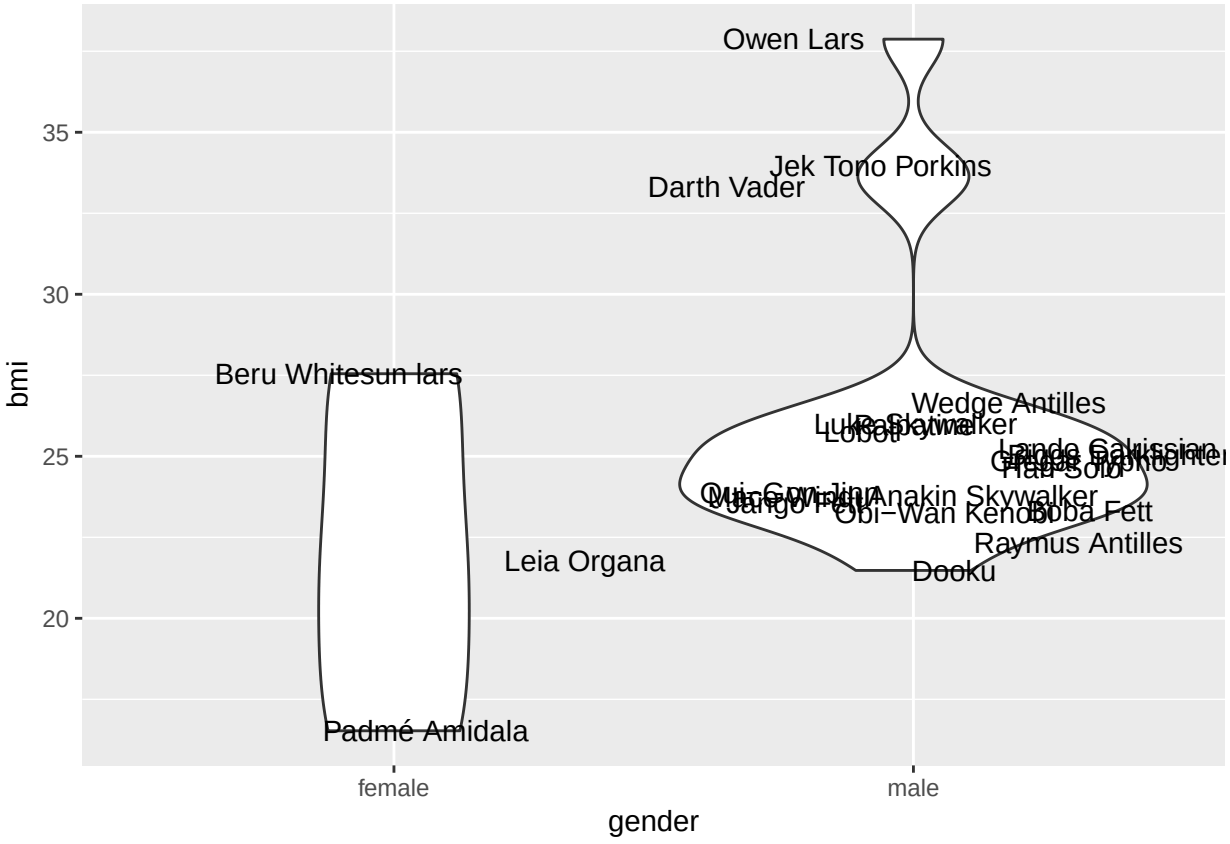
tidyr : gather, spread

```
sw <- select(starwars, -(films:starships))

body.colors <- starwars %>%
  select(name, contains("color"))
```

```
colors.gathered <- body.colors %>%
  gather(color_type, color, -name) %>%
  arrange(name, color_type, color)

colors.spread <- colors.gathered %>%
  spread(color_type, color) %>%
  as.data.frame
```

pipeline to ggplot

```
starwars %>%
  mutate(bmi = mass / (height / 100) ^ 2) %>%
  select(name, bmi, species, gender) %>%
  filter(complete.cases(.) & species == "Human") %>%
  ggplot(aes(gender, bmi)) +
  geom_violin() +
  geom_text(aes(label = name), position = "jitter")
```



RMarkdown

Code Chunks

To create a simple code chunk, you can use:

- keyboard shortcuts
  - Windows: CTL+ALT+i
  - Mac: Cmd+Option+i
- Insert from toolbar
- type in by hand

This chunk executes and shows the command:

```
z <- 1
```

The code in this chunk is not visible, but the result is:

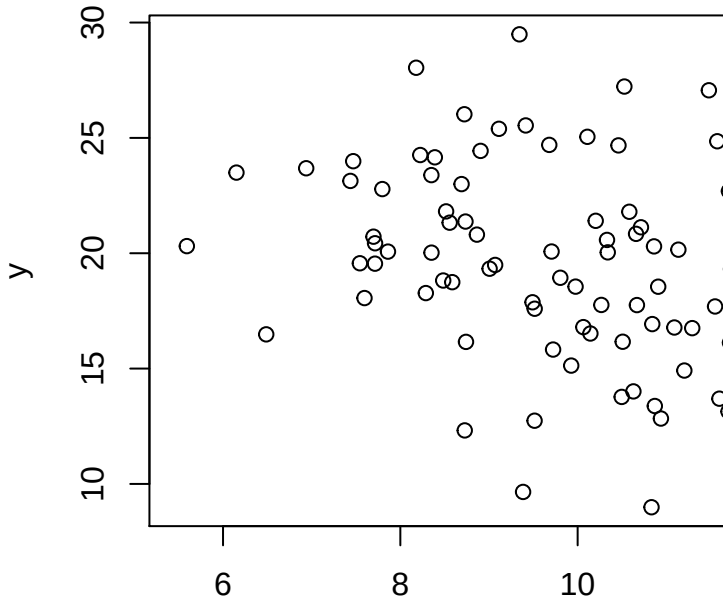
```
[1] 5
```

The code and result are visible from this chunk:

```
print(z)
```

```
[1] 5
```

Neither the code nor result is visible from this chunk, but the code is still run:



The plot above shows that this code and the code above was run.

This code is visible, but is never run:

```
z <- 250
print(z)
```

We can see that `z` has the same value as when first set:

```
print(z)
```

```
[1] 5
```

The comment character can be changed (or eliminated), and a console prompt shown:

```
> a <- 4
> b <- 8
> # Here is the result
> a + b ^ 2
```

```
[1] 68
```

Defaults can be changed by setting options to `knitr::opts_chunk$set`:

```
knitr::opts_chunk$set(comment = "*", echo = FALSE)
```

```
# Hello there!!
print(5 * 6)
```

```
[1] 30
```

If you are creating a computationally-intensive document, you should cache the results of your chunks:

```
knitr::opts_chunk$set(cache = TRUE)
```

When chunks are cached, it can be good to name them:

```
x <- runif(1e5)
y <- mean(x)
cat(y)
```

```
0.4988337
```

Full list of knitr chunk options: https://yihui.name/knitr/options/#other-chunk-options

---

Inline Code

Code can be placed inline so that results are embedded directly in the text. For instance, a * b ^ 2 = 256. Multiple steps can be done in chunks, then results embedded inline:

```
p <- 0.6
odds.p <- p / (1 - p)
log.odds.p <- log(odds.p)
```

For p = 0.6, the odds(p) = 1.5, and the log-odds(p) = 0.405.

Markdown Formatting Syntax

This is a sentence. There is one space following this sentence. The third sentence ends up here.

This is another sentence. If I follow this sentence with two spaces, the third sentence will be a new paragraph. Like this.

Text can be *italics* or **bold** or ***italics-bold***.

Text can have$^{\text{superscripts}}$ and$_{\text{subscripts}}$.

See cheatsheet and reference guide for other formatting syntax.

Tables

You can use the vertical separators (i.e., pipe, "|") and horizontal separator (i.e., dash, "-") to create tables. It doesn't have to be neatly aligned as in the examples:

Header 1	Header 2
Row1, Col1	Row1, Col2
Row2, Col1	Row2, Col2

Here are example default tables for a `data.frame` using `knitr`, `xtable`, and `pander`:

```
df <- mtcars[1:5, ]
library(knitr)
kable(df, caption = "From knitr")
```

Table 2: From knitr

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2

```
library(xtable)
print(xtable(df, caption = "From xtable"), comment = F)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.00	6.00	160.00	110.00	3.90	2.62	16.46	0.00	1.00	4.00	4.00
Mazda RX4 Wag	21.00	6.00	160.00	110.00	3.90	2.88	17.02	0.00	1.00	4.00	4.00
Datsun 710	22.80	4.00	108.00	93.00	3.85	2.32	18.61	1.00	1.00	4.00	1.00
Hornet 4 Drive	21.40	6.00	258.00	110.00	3.08	3.21	19.44	1.00	0.00	3.00	1.00
Hornet Sportabout	18.70	8.00	360.00	175.00	3.15	3.44	17.02	0.00	0.00	3.00	2.00

Table 3: From xtable

```
library(pander)
pander(df, caption = "From pander")
```

Table 4: From pander (continued below)

	mpg	cyl	disp	hp	drat	wt	qsec	vs
Mazda RX4	21	6	160	110	3.9	2.62	16.46	0
Mazda RX4 Wag	21	6	160	110	3.9	2.875	17.02	0
Datsun 710	22.8	4	108	93	3.85	2.32	18.61	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1
Hornet Sportabout	18.7	8	360	175	3.15	3.44	17.02	0

	am	gear	carb
Mazda RX4	1	4	4
Mazda RX4 Wag	1	4	4
Datsun 710	1	4	1
Hornet 4 Drive	0	3	1
Hornet Sportabout	0	3	2