Plotting with mobilizr

Gilbert Neuner

2022-10-07

Overview

The `mobilizr` package, used in the Intro to Data Science (IDS) curriculum, contains useful plotting functions, including `bargraph`, `bwplot`, `histogram`, and `xyplot`. Instructions for using these plotting functions can be found by accessing their help files, which can be found by running the following code (where `xyplot` may be substituted with the appropriate plotting function):

```
library(mobilizr)
help(xyplot, package="mobilizr")
```

The IDS curriculum also makes use of the `dotPlot` function from the mosaic package.

The help files for `mobilizr's` plotting functions contain notes indicating that additional options may be found in help files from the `lattice` and `mosaic` packages. The purpose of this vignette is to provide some helpful examples of these additional arguments.

The examples make use of the `cdc` dataset:

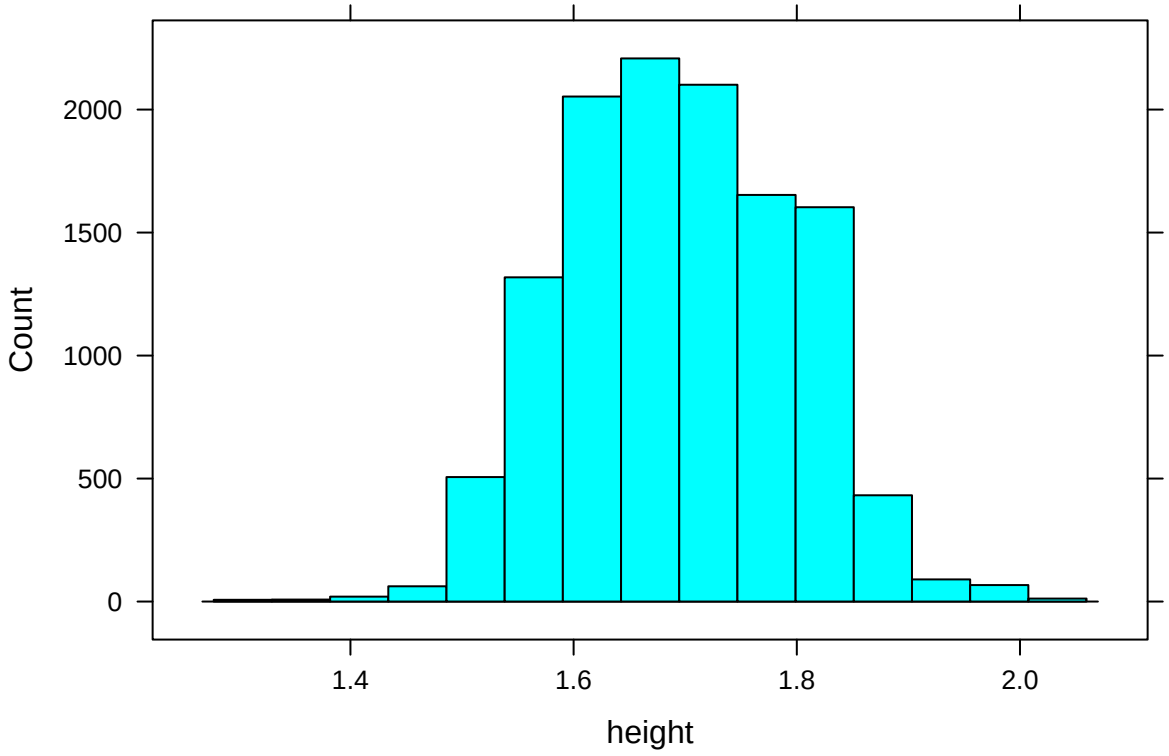
```
data(cdc)
```

Arguments

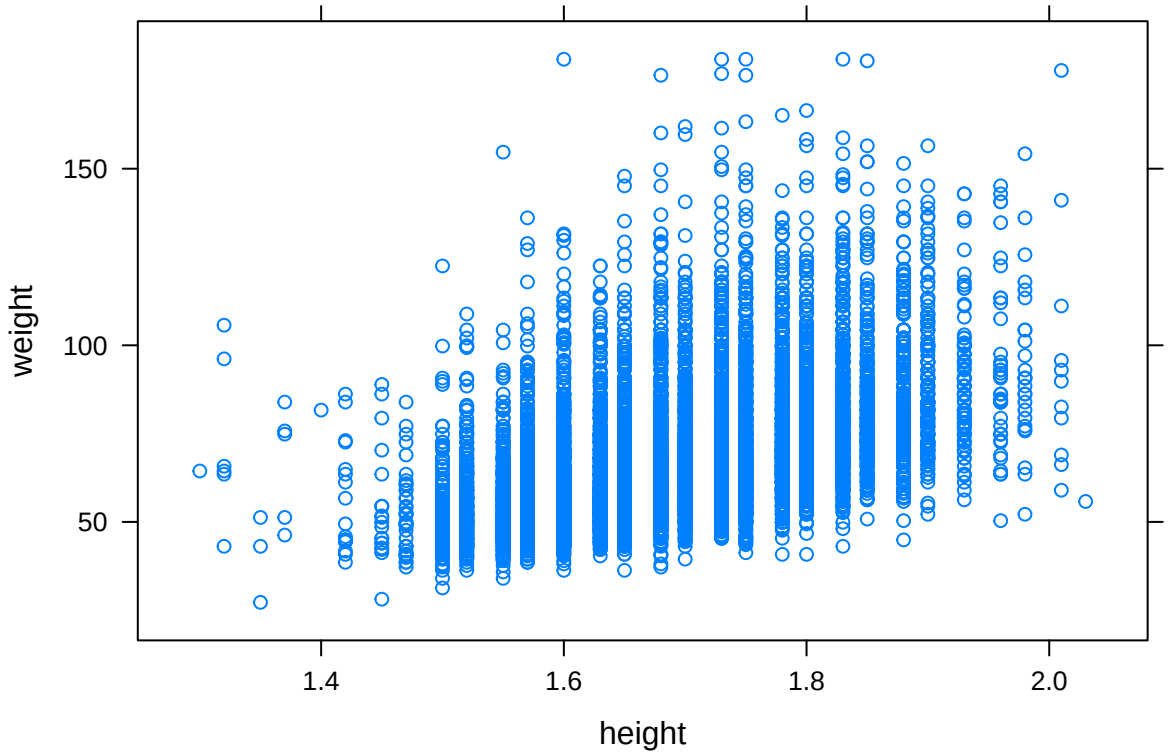
x and data

All of `mobilizr's` plotting functions take `x` and `data` as their first two arguments. `x` must be a formula of the form `~x` or `y ~ x` depending on the number of variables the plot requires. `data` is the data frame that contains the columns that were supplied to `x` as a formula. For example,

```
# Histograms take one variable
histogram(~height, data = cdc)
```

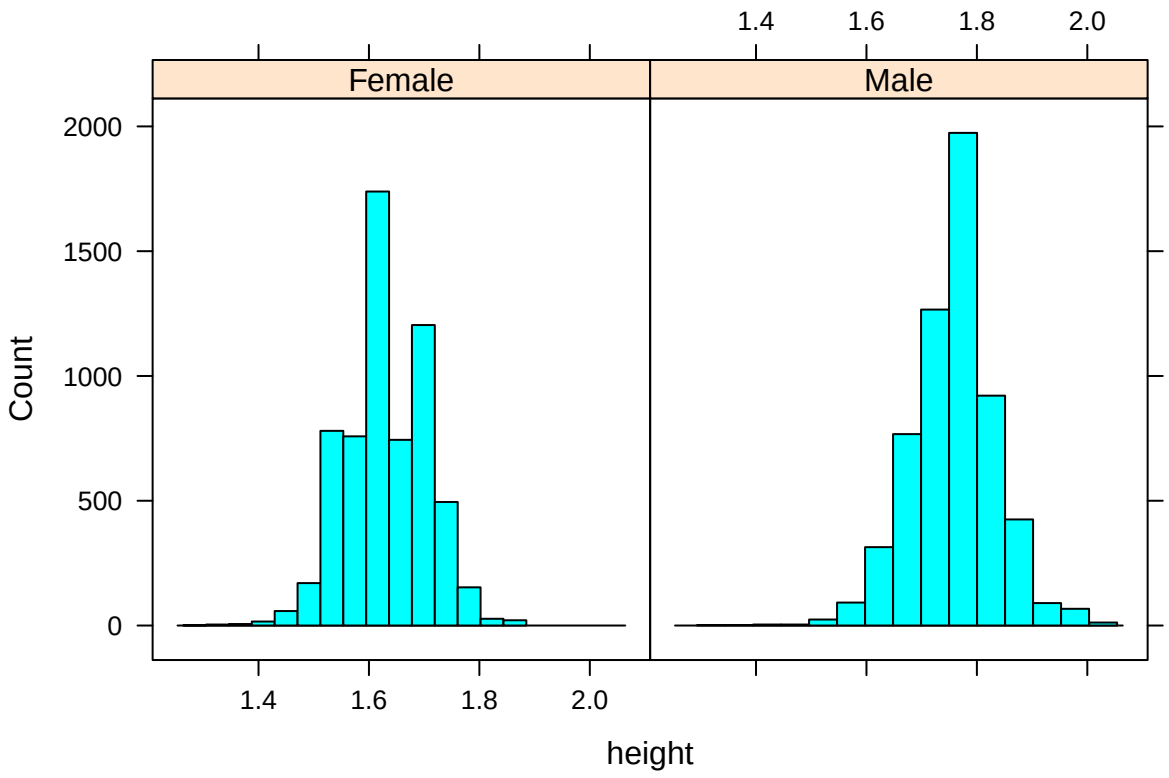


```r
# Scatterplots take two variables
xyplot(weight ~ height, data = cdc)
```

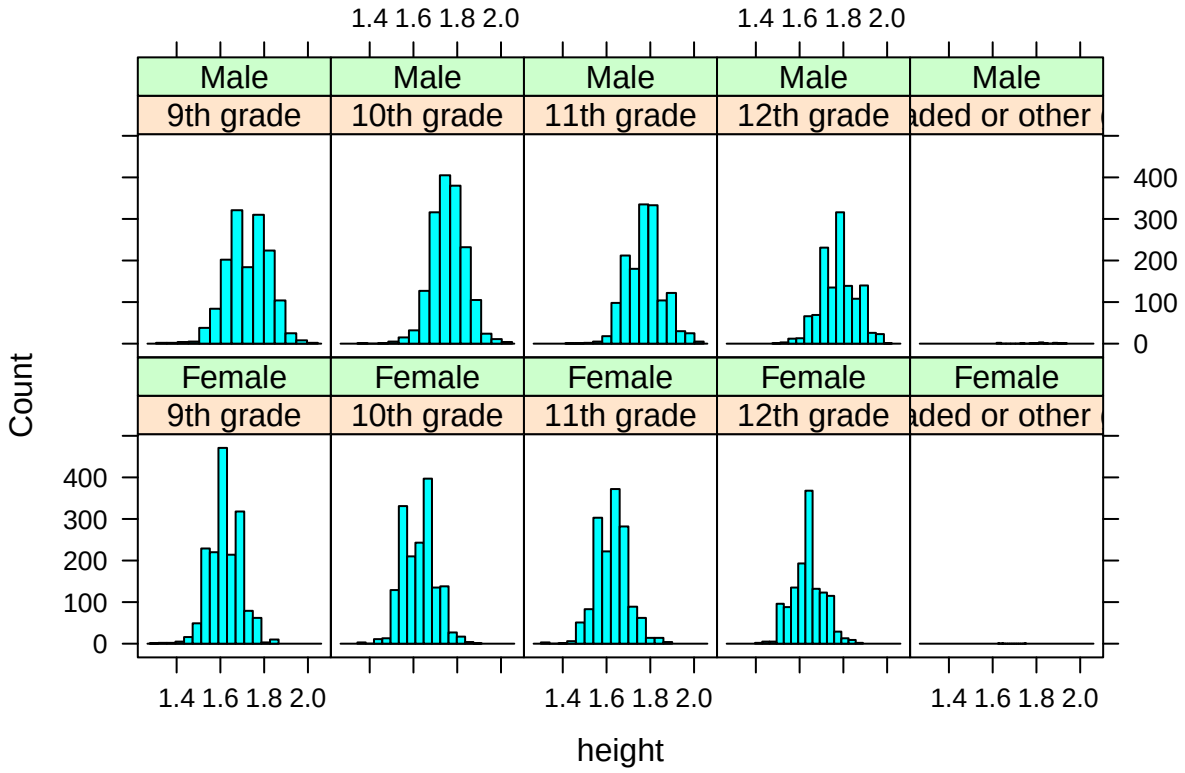


When plotting, one might want to facet by a categorical variable, i.e., subset the data by a faceted variable and create plots for each of subset seperately. The syntax for faceting is to supply `x` a formula of the form `x|g1 + g2 + ...` where `g1`, `g2`, ... are the faceted variables. For example,

```
# Facet by one variable
histogram(~height|gender, data = cdc)
```



```
# Facet by two variables
histogram(~height|grade + gender, data = cdc)
```

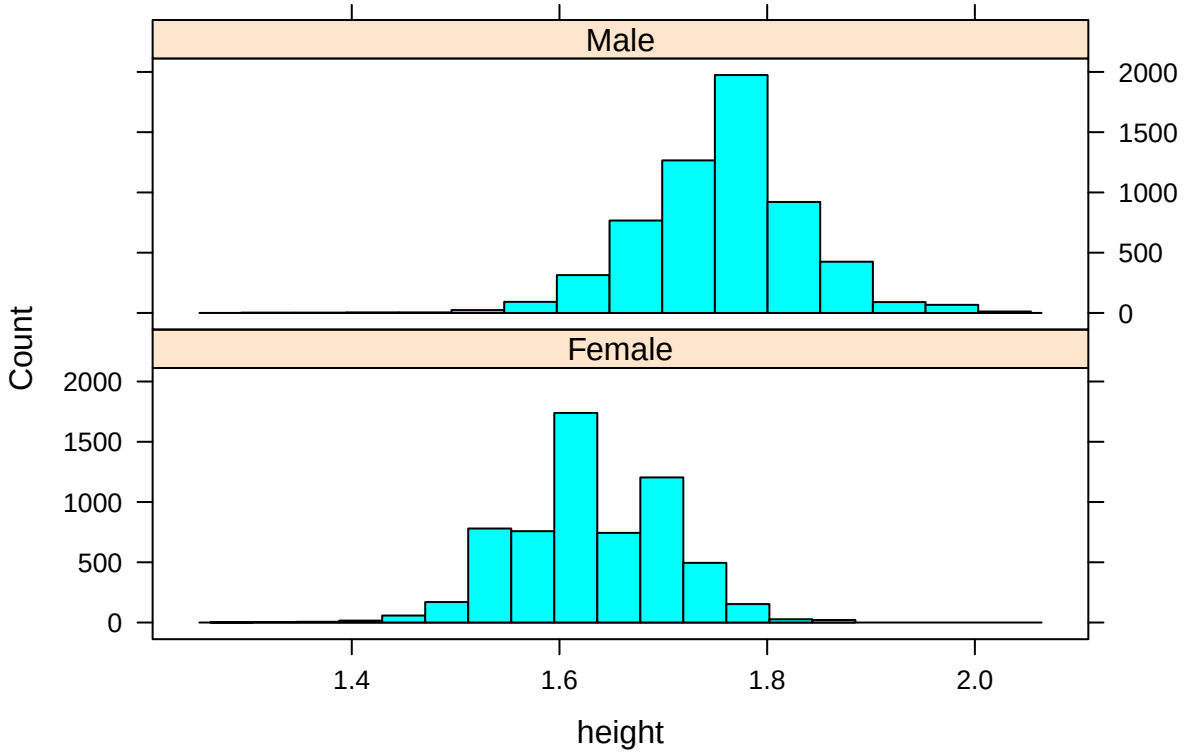


`x` and `data` are not optional, meaning they must be supplied every time a `mobilizr` plotting function is called. The rest of the arguments are optional, meaning that they take on default values (often `NULL`) if the user chooses not to supply them.

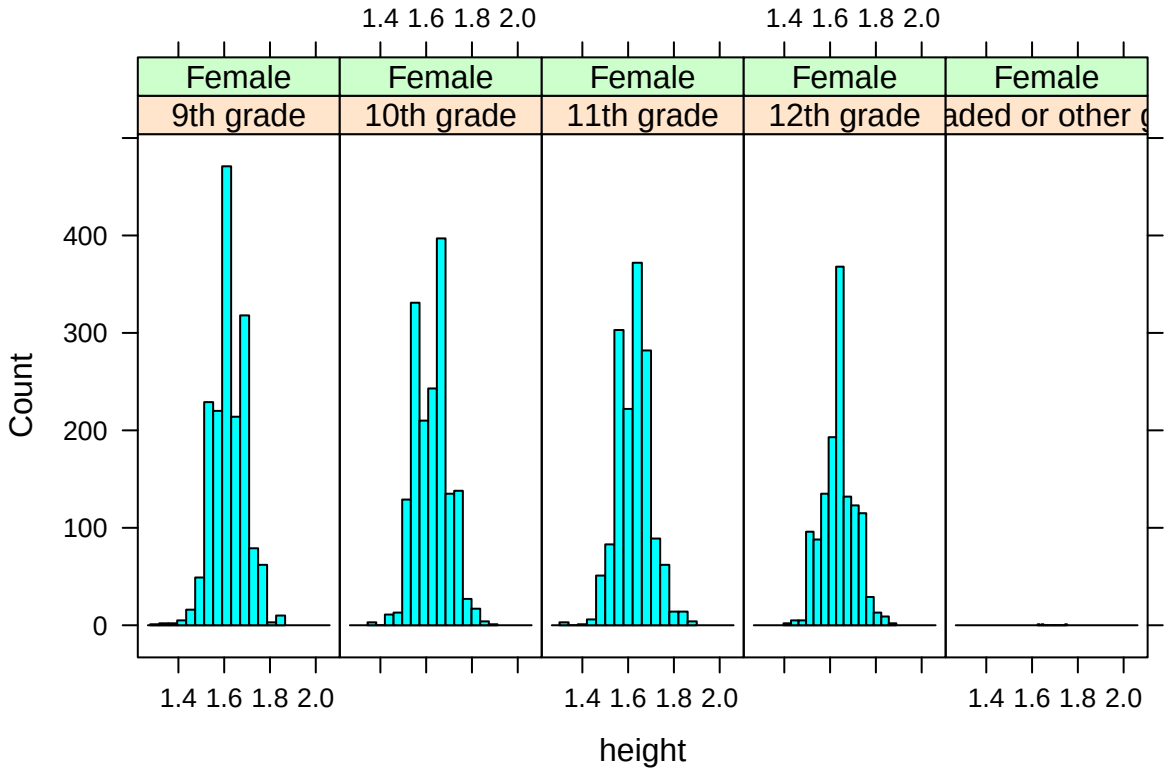
layout

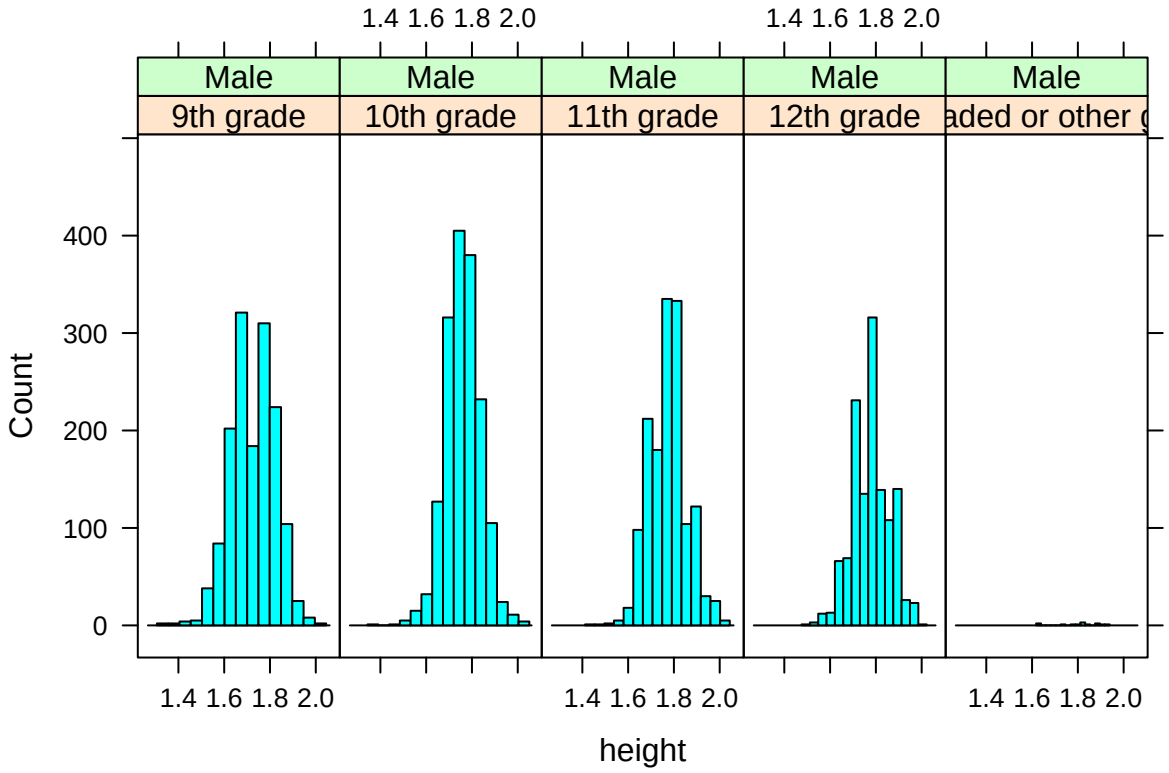
`layout` determines the arrangement of faceted plots. `layout` is a numeric vector of length 2 or 3 giving the number of columns, rows, and pages (optional).

```
# Put the faceted plots in a single column rather than a single row
histogram(~height|gender, data = cdc, layout = c(1,2))
```



```
# Specifying 2 pages produces 2 plots
histogram(~height|grade + gender, data = cdc, layout = c(5,1,2))
```

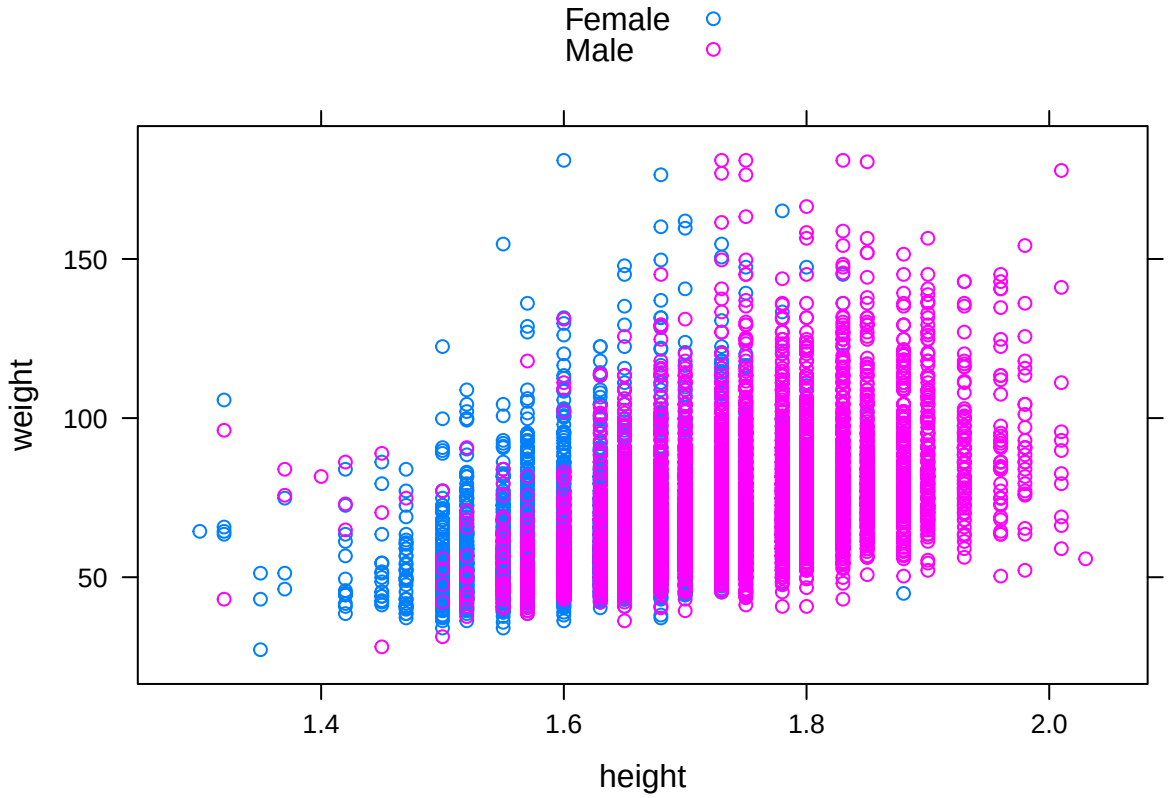



1.4 1.6 1.8 2.0
1.4 1.6 1.8 2.0
Male
Male
Male
Male
Male
9th grade
10th grade
11th grade
12th grade
aded or other
400
300
200
100
0
Count
1.4 1.6 1.8 2.0
1.4 1.6 1.8 2.0
1.4 1.6 1.8 2.0
height

groups

`groups` specifies a grouping variable, used to distinguish groups by color.

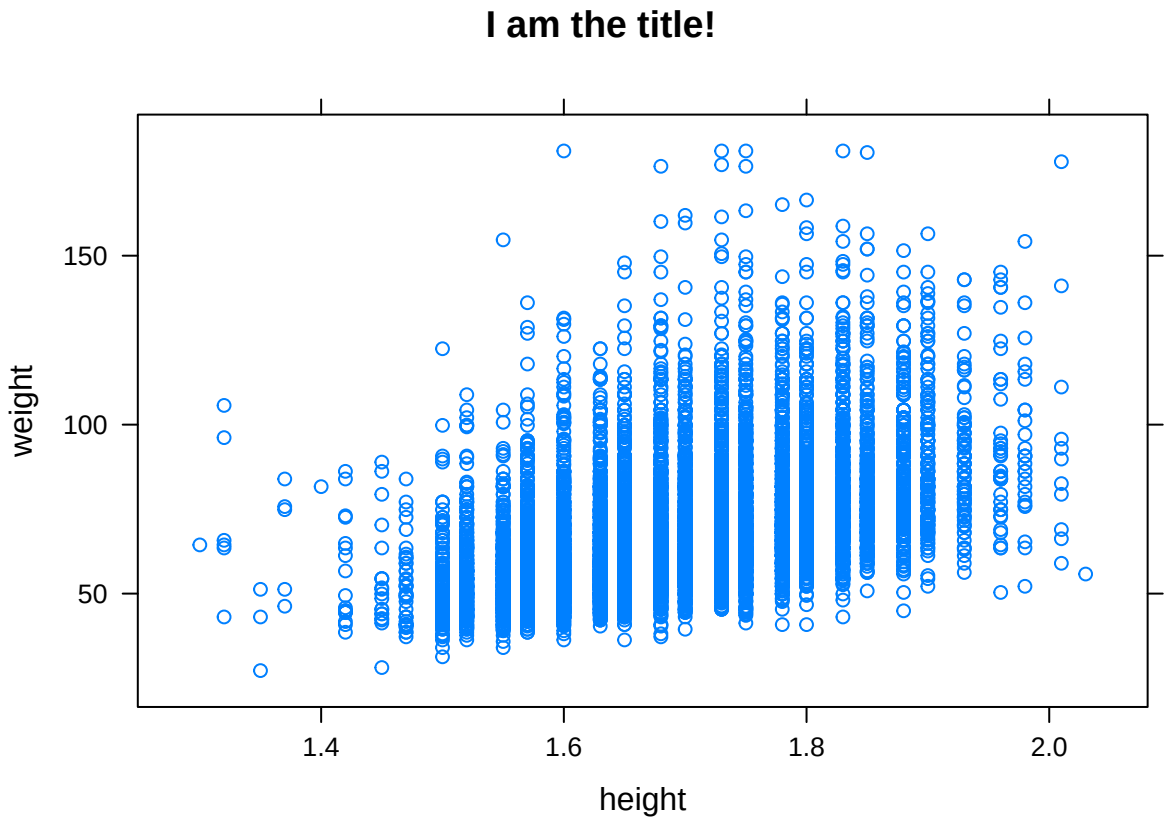
```
xyplot(weight ~ height, data = cdc, groups = gender)
```



main

`main` is typically a character string describing the main title. For example,

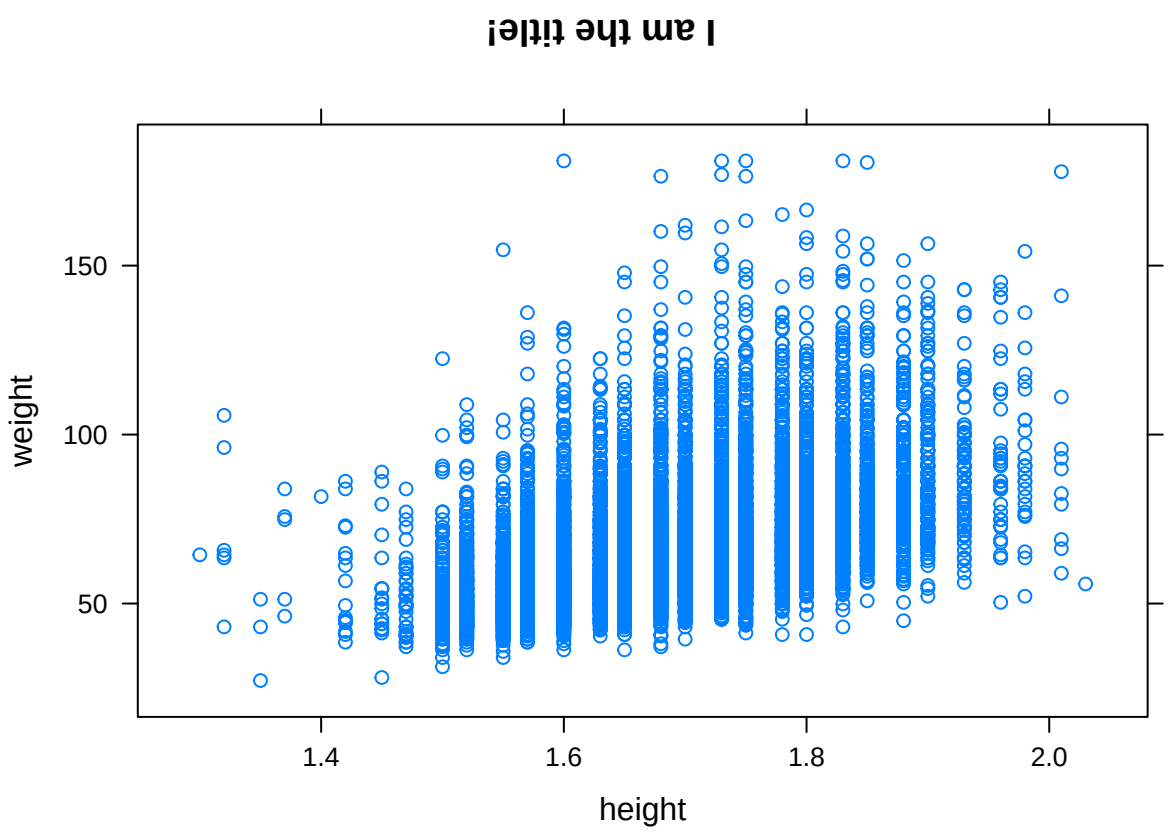
```
xyplot(weight ~ height, data = cdc, main = "I am the title!")
```



To control further details about the title, `main` can also be a list. When `main` is a list, the actual label should be specified as the `label` component (which may be unnamed if it is the first component). The label can be missing, in which case the default will be used. Further arguments include

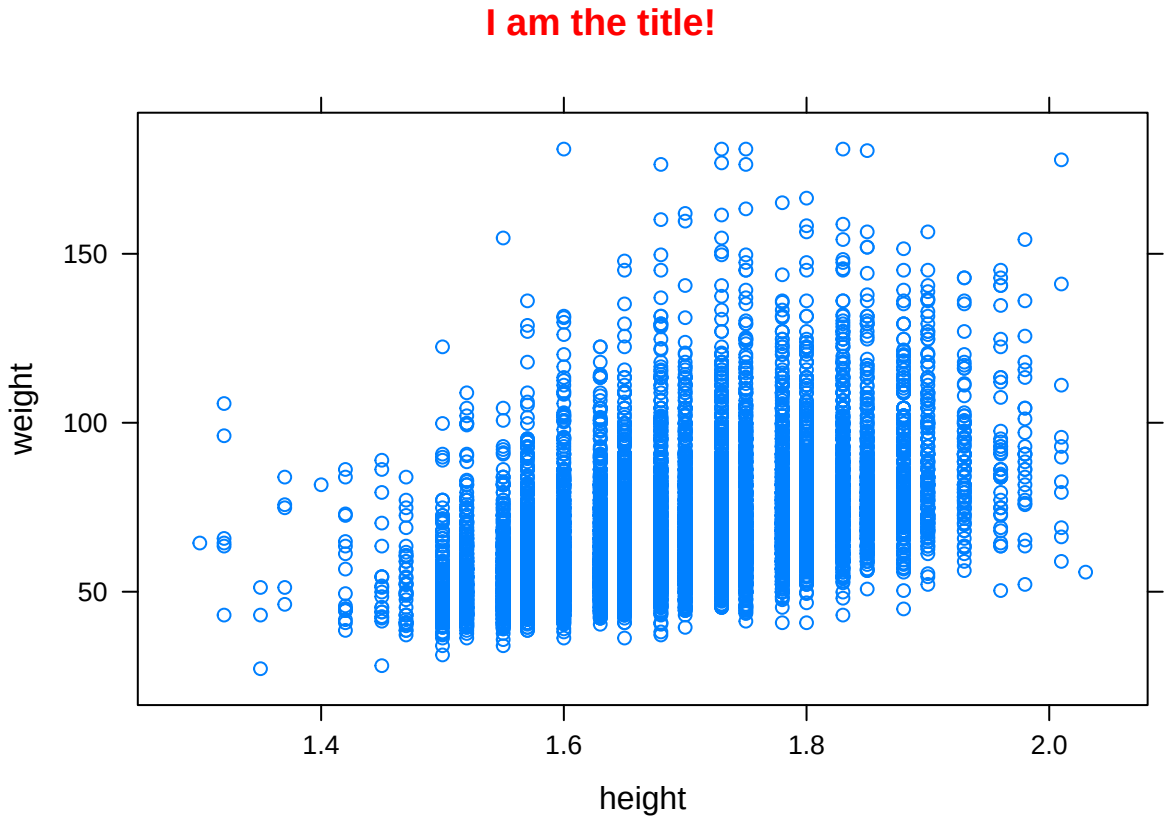
rot: rotation

```
xyplot(weight ~ height, data = cdc, main = list("I am the title!", rot = 180))
```



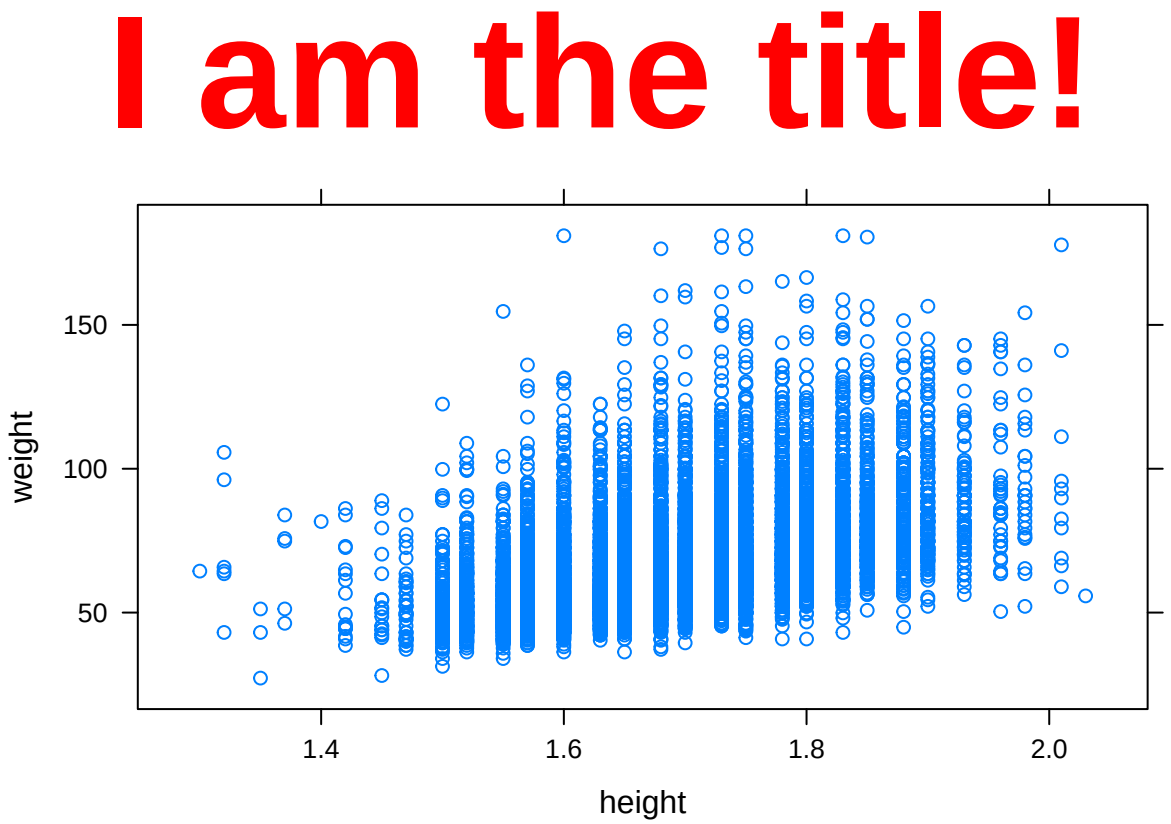
col: color

```
xyplot(weight ~ height, data = cdc, main = list("I am the title!", col = "red"))
```



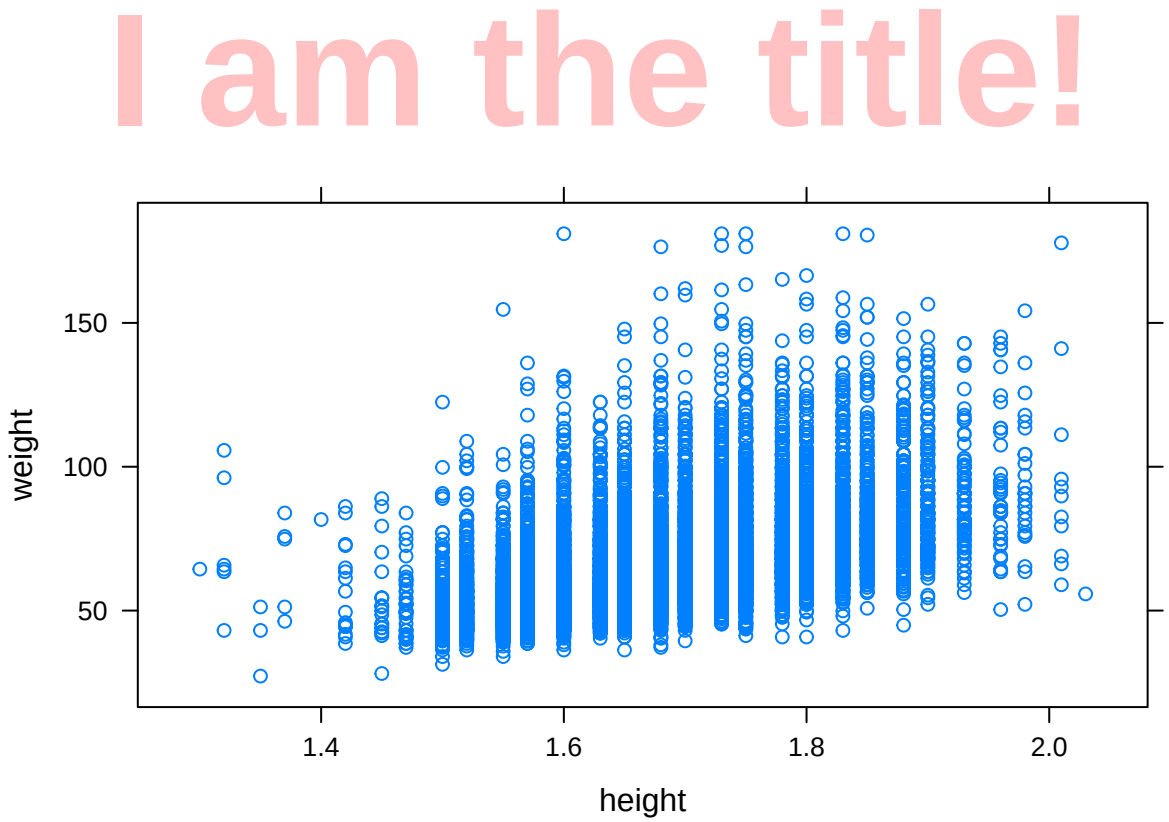
cex: size

```
xyplot(weight ~ height, data = cdc, main = list("I am the title!", col = "red", cex = 5))
```



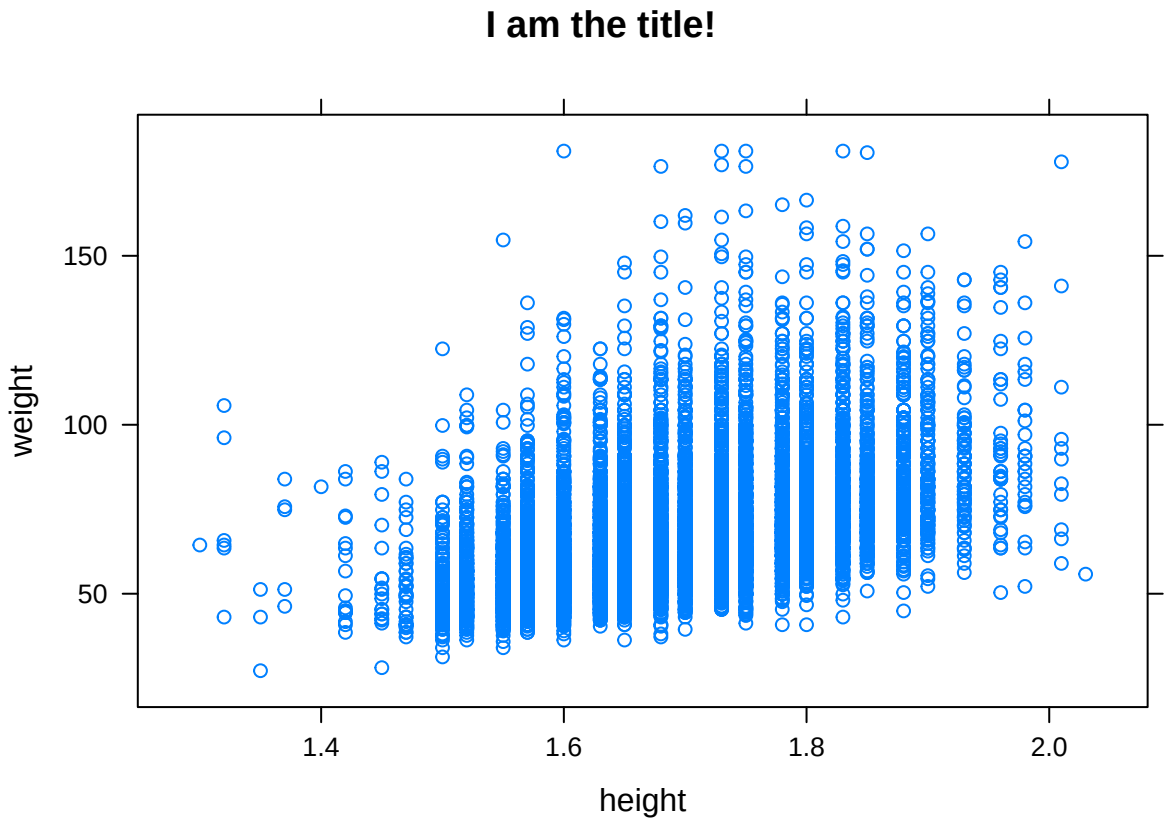
alpha: transparency

```
xyplot(weight ~ height, data = cdc, main = list("I am the title!", col = "red", cex = 5, alpha = 0.25))
```

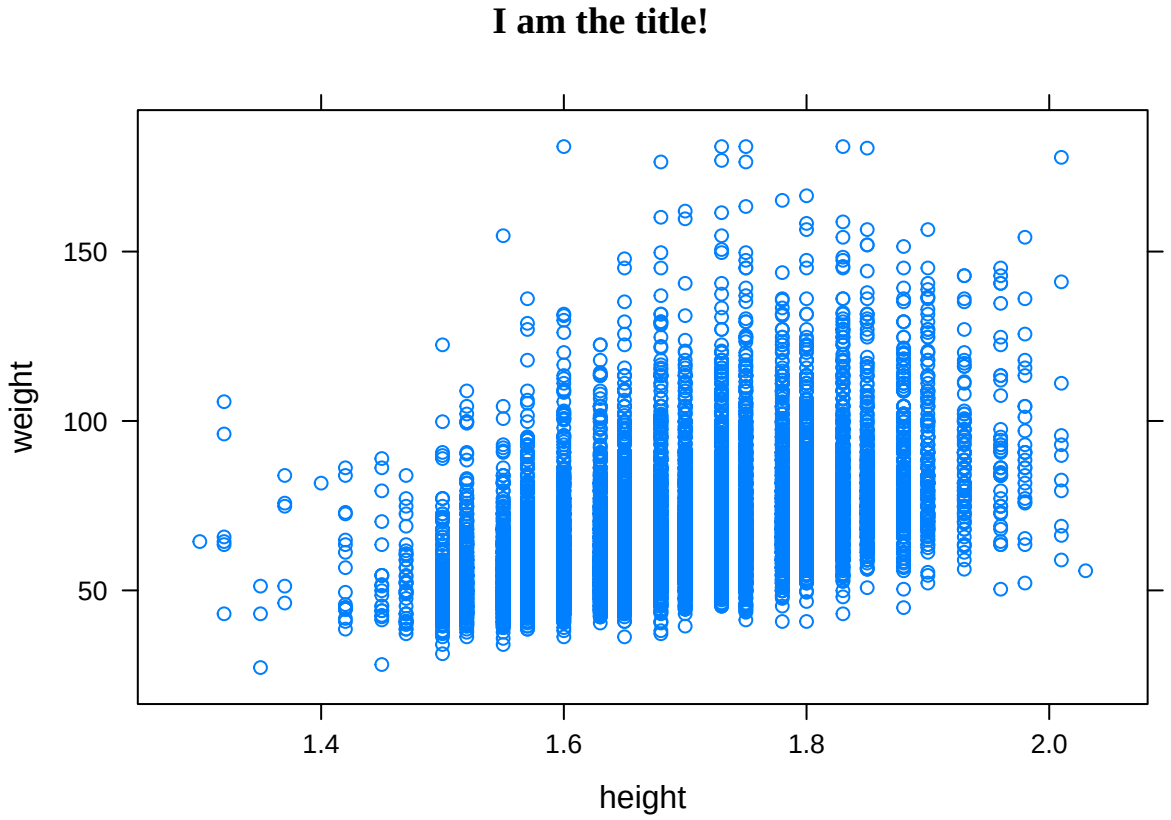


fontfamily

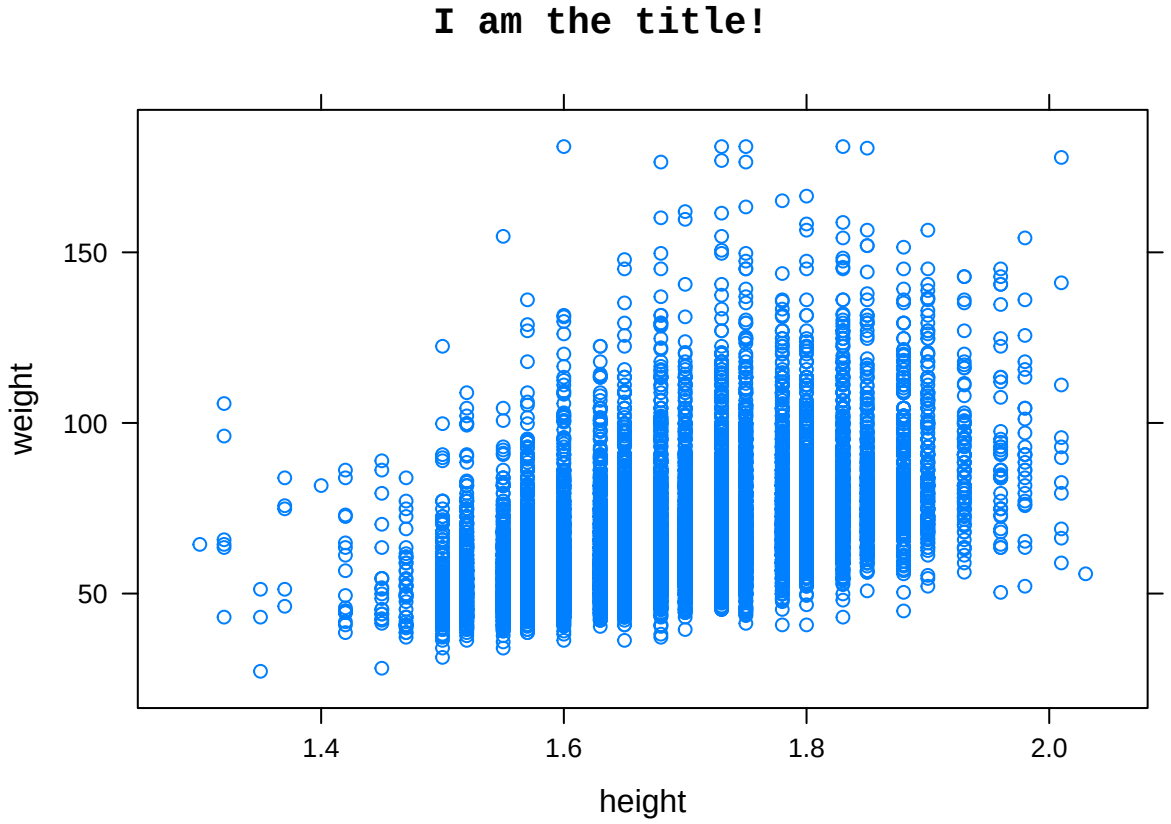
```
xyplot(weight ~ height, data = cdc, main = list("I am the title!", fontfamily = "sans"))
```



```
xyplot(weight ~ height, data = cdc, main = list("I am the title!", fontfamily = "serif"))
```

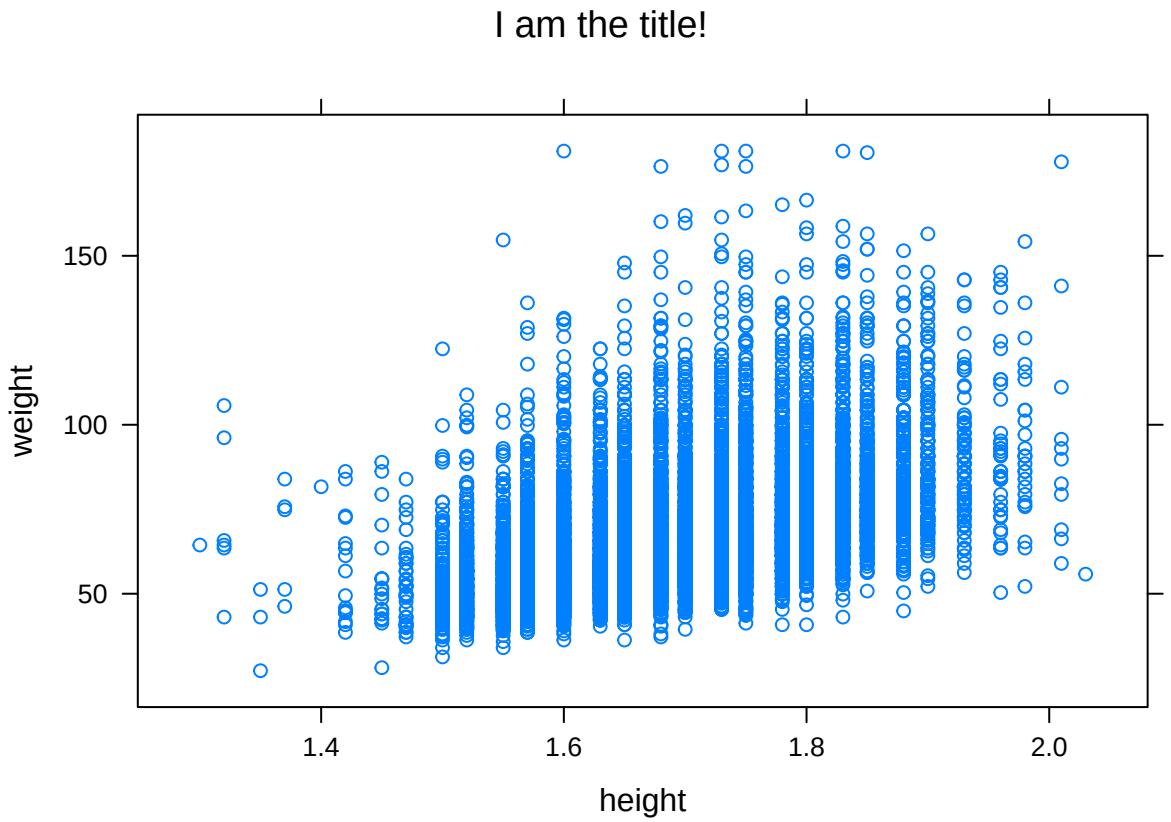


```
xyplot(weight ~ height, data = cdc, main = list("I am the title!", fontfamily = "mono"))
```

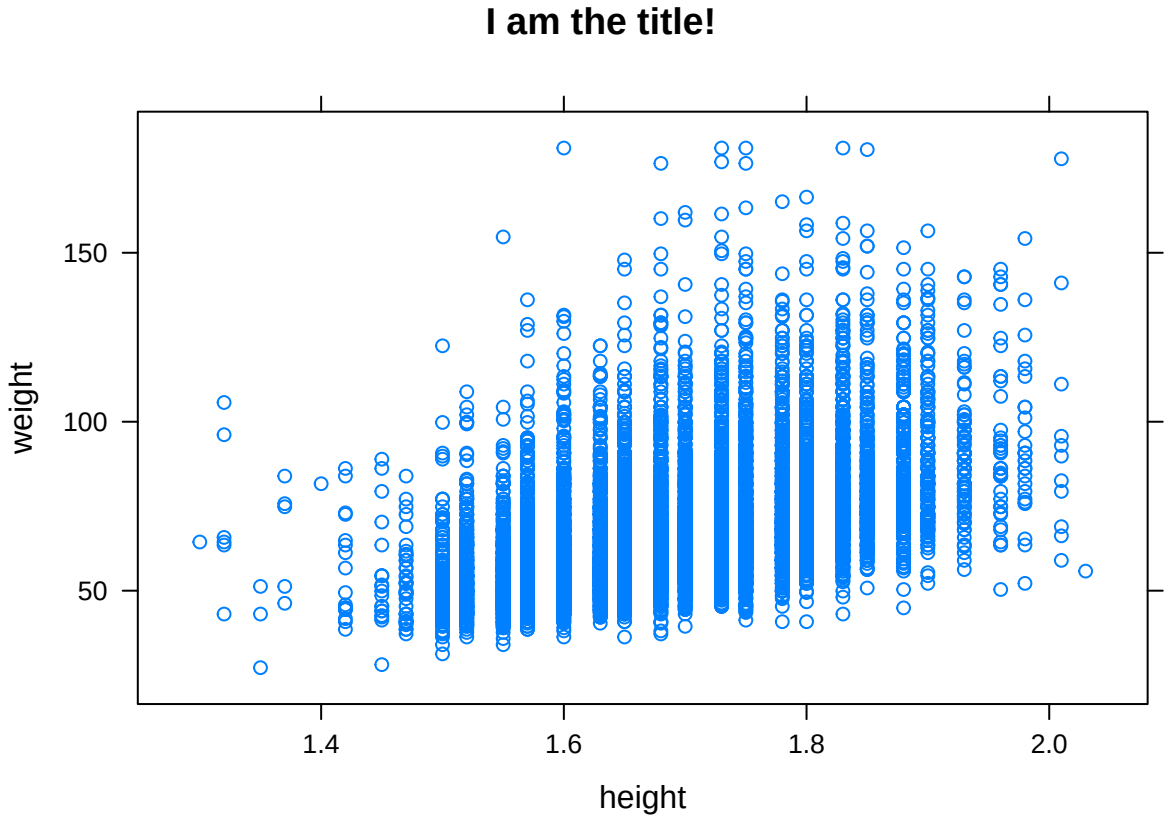


fontface

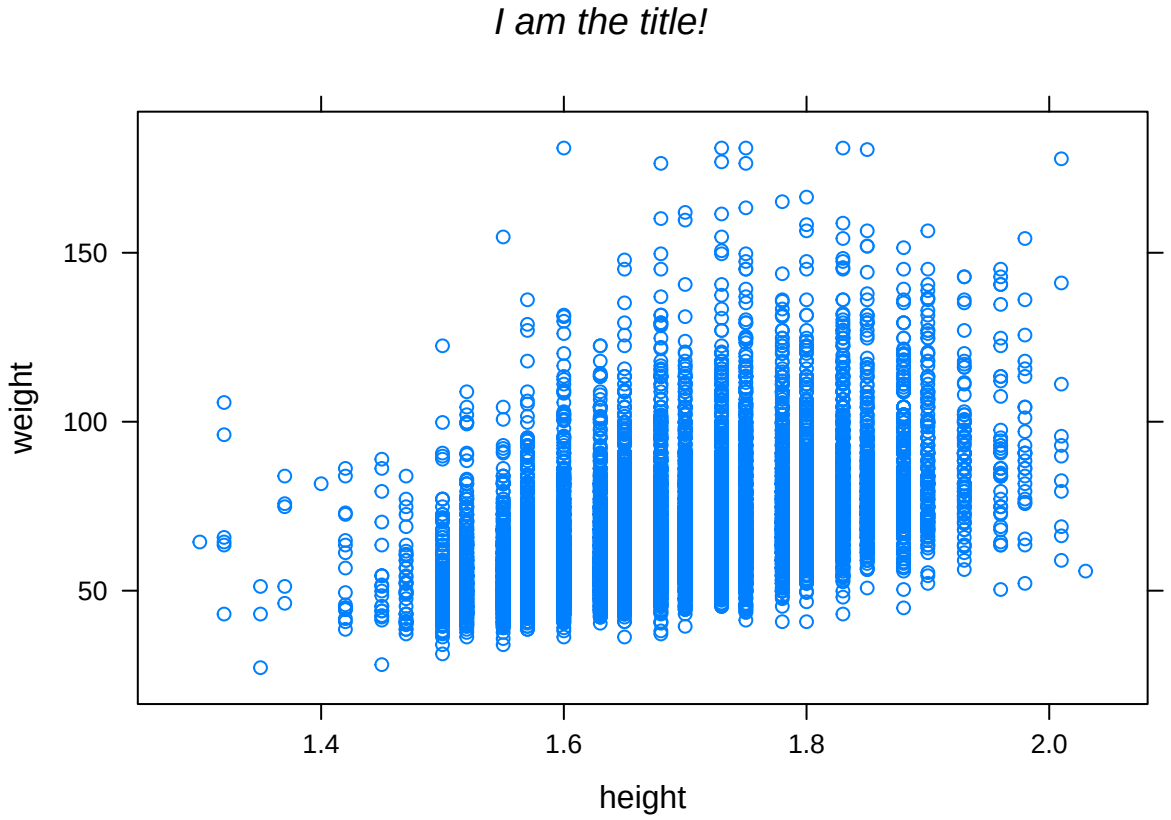
```
xyplot(weight ~ height, data = cdc, main = list("I am the title!", fontface = "plain"))
```



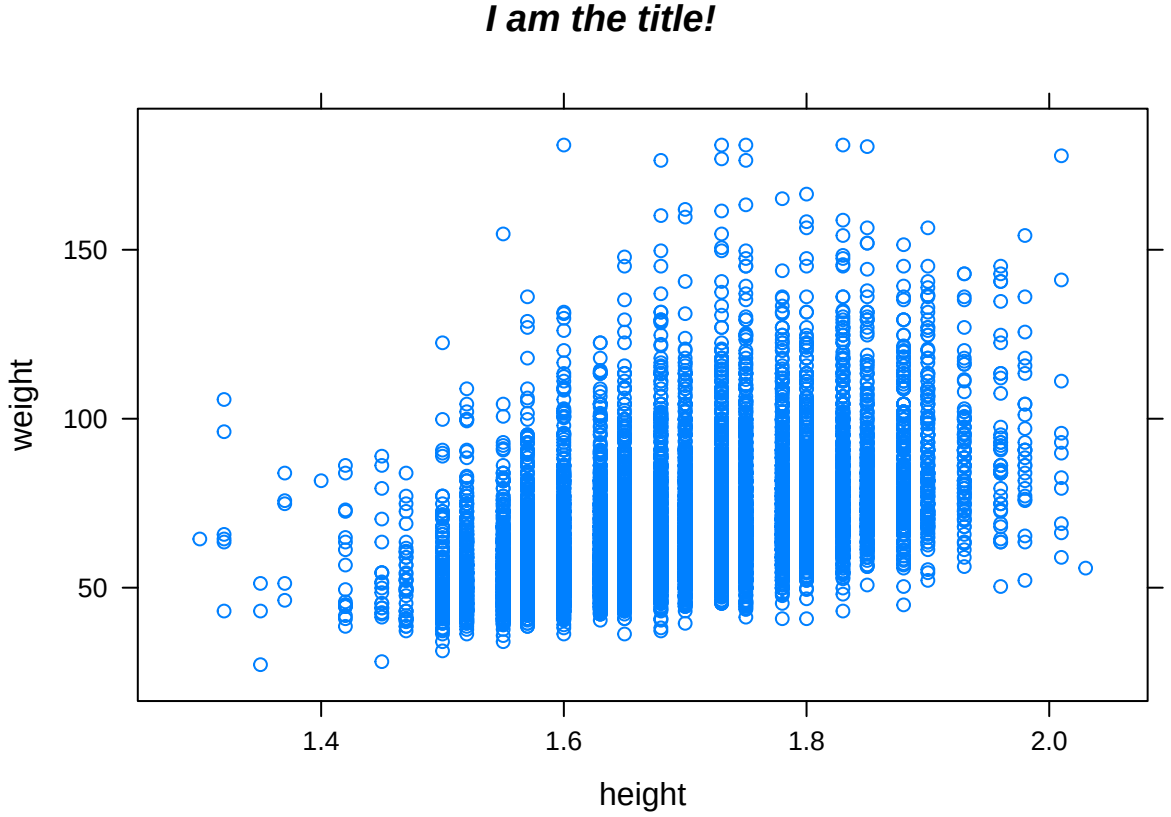
```
xyplot(weight ~ height, data = cdc, main = list("I am the title!", fontface = "bold"))
```



```
xyplot(weight ~ height, data = cdc, main = list("I am the title!", fontface = "italic"))
```



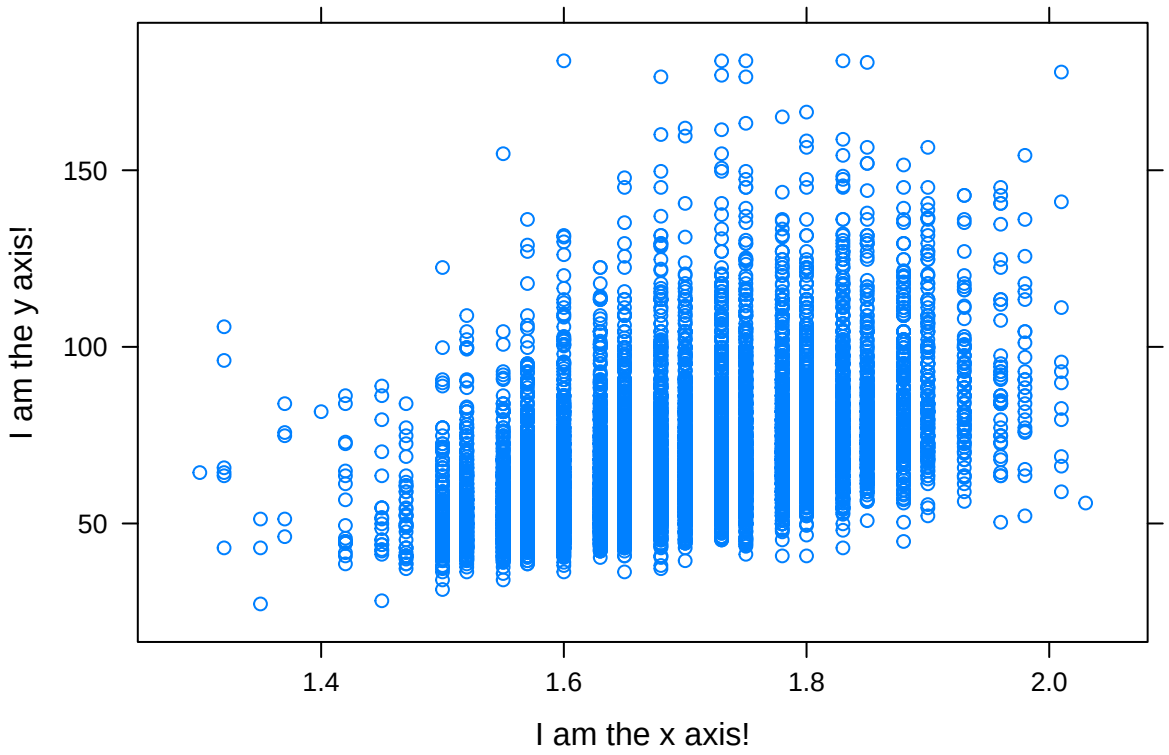
```
xyplot(weight ~ height, data = cdc, main = list("I am the title!", fontface = "bold.italic"))
```



xlab and ylab

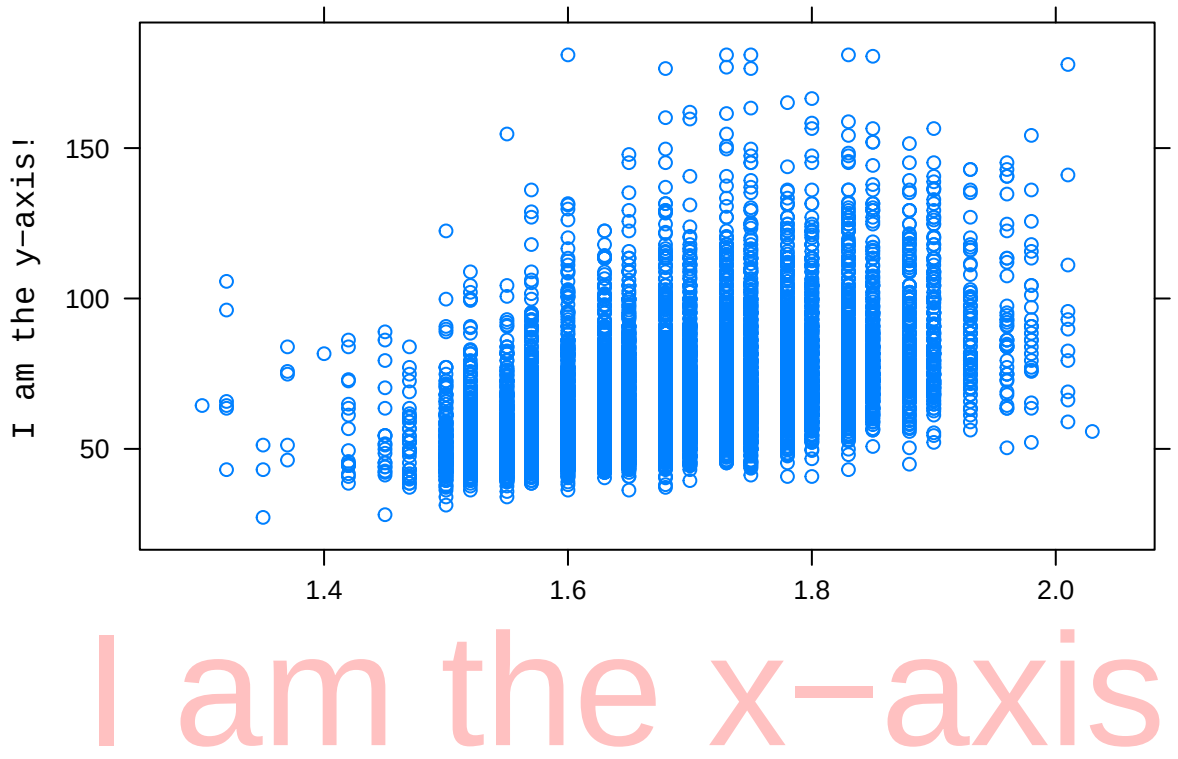
`xlab` and `ylab` are used to label the x and y axes.

```
xyplot(weight ~ height, data = cdc, xlab = "I am the x axis!", ylab = "I am the y axis!")
```



Finer control is possible, and is done in the same way as `main`.

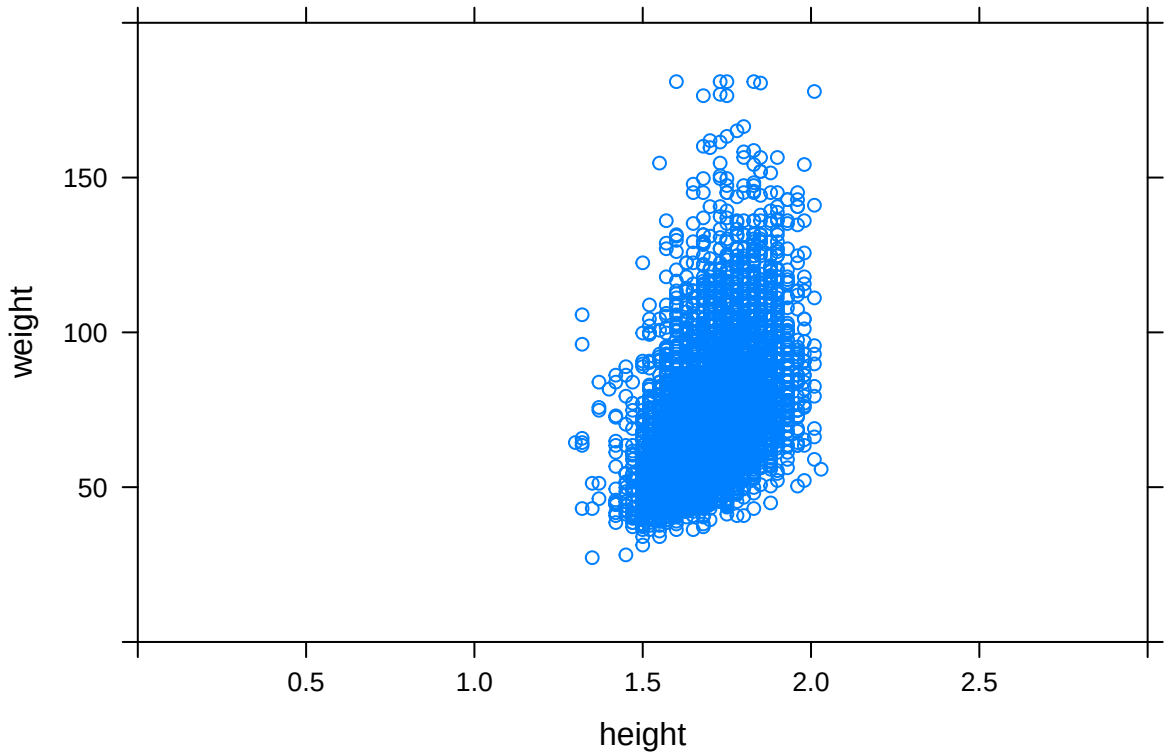
```
xyplot(weight ~ height, data = cdc,
       xlab = list("I am the x-axis!", col = "red", cex = 5, alpha = 0.25),
       ylab = list("I am the y-axis!", fontfamily = "mono"))
```



xlim and ylim

`xlim` and `ylim` are numeric vectors specifying the left and right limits for the x and y axis, respectively.

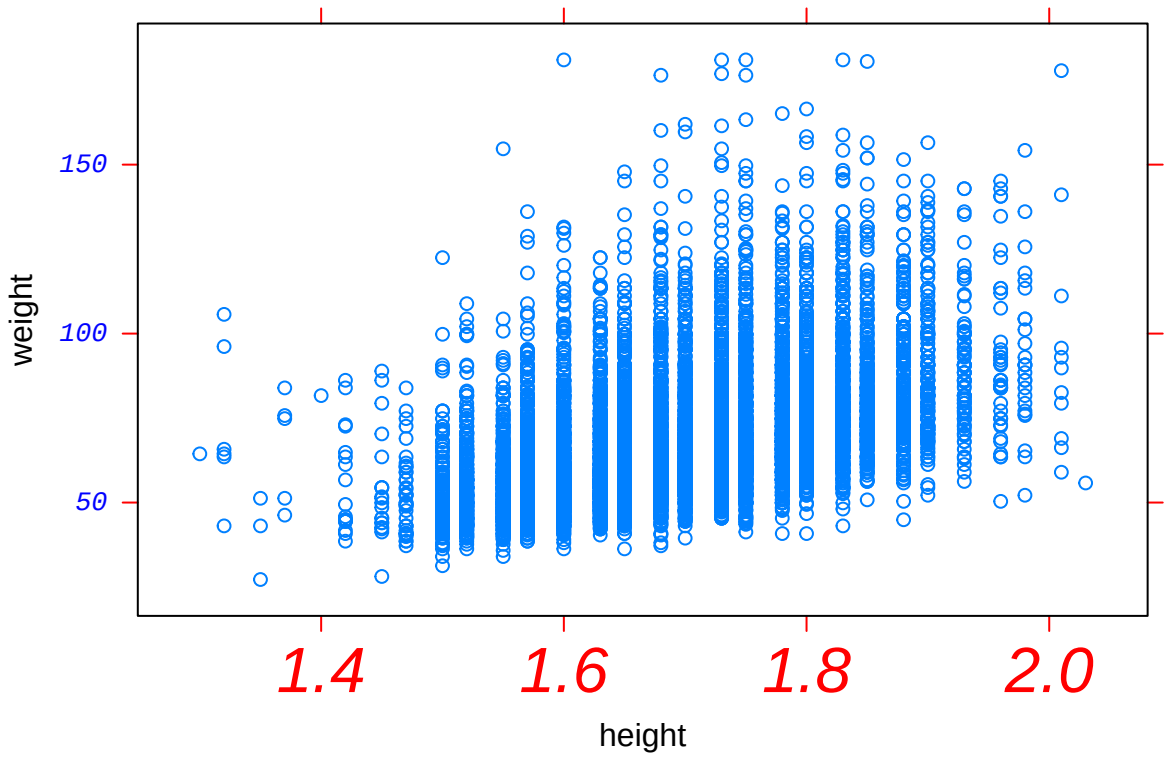
```
xyplot(weight ~ height, data = cdc, xlim = c(0, 3), ylim = c(0, 200))
```



scales

To control further details about the axes, use `scales`. `scales` is a list of arguments, just like `main`. All of the arguments that worked for `main` also work for `scales`. `scales` may contain two other lists called `x` and `y` of the same form (described below). Components of `x` and `y` affect the respective axes only, while those in scales affect both. When parameters are specified in both lists, the values in `x` or `y` are used. For example:

```
# Contrived example
xyplot(weight ~ height, data = cdc, scales = list(col = "red", fontface = "italic",
                                                  x = list(cex = 2),
                                                  y = list(col = "blue", fontfamily = "mono")))
```



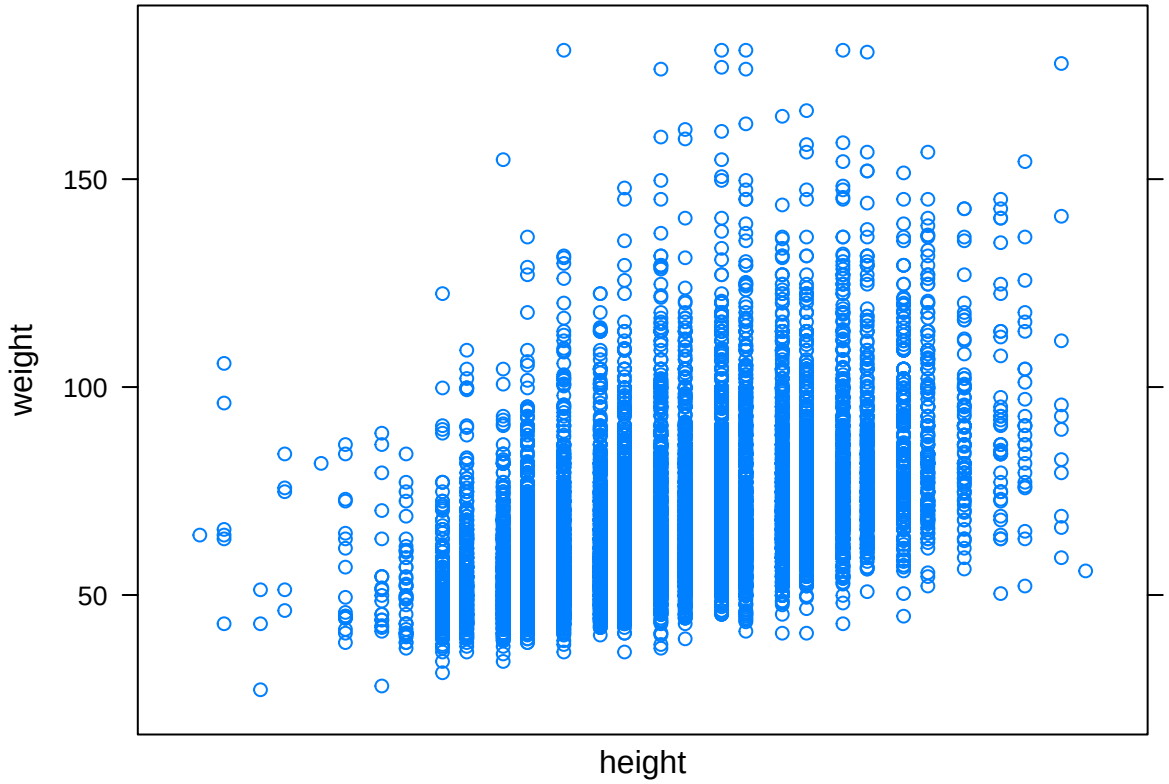
- Components of `x` and `y` affect the respective axes only: note that only the x scales have `cex = 2` and only the y scales have `fontfamily = mono`
- Components in `scales` affect both axes: note that both scales are italicized
- When parameters are specified in both lists, the values in `x` or `y` are used: despite the fact that `col = "red"` is included in scales, only the x axis is red. This is because `col = "blue"` is also included in `y`.

Some other arguments that can be included in `scales`, `x`, and `y` include:

draw

Determines whether the axis is drawn at all.

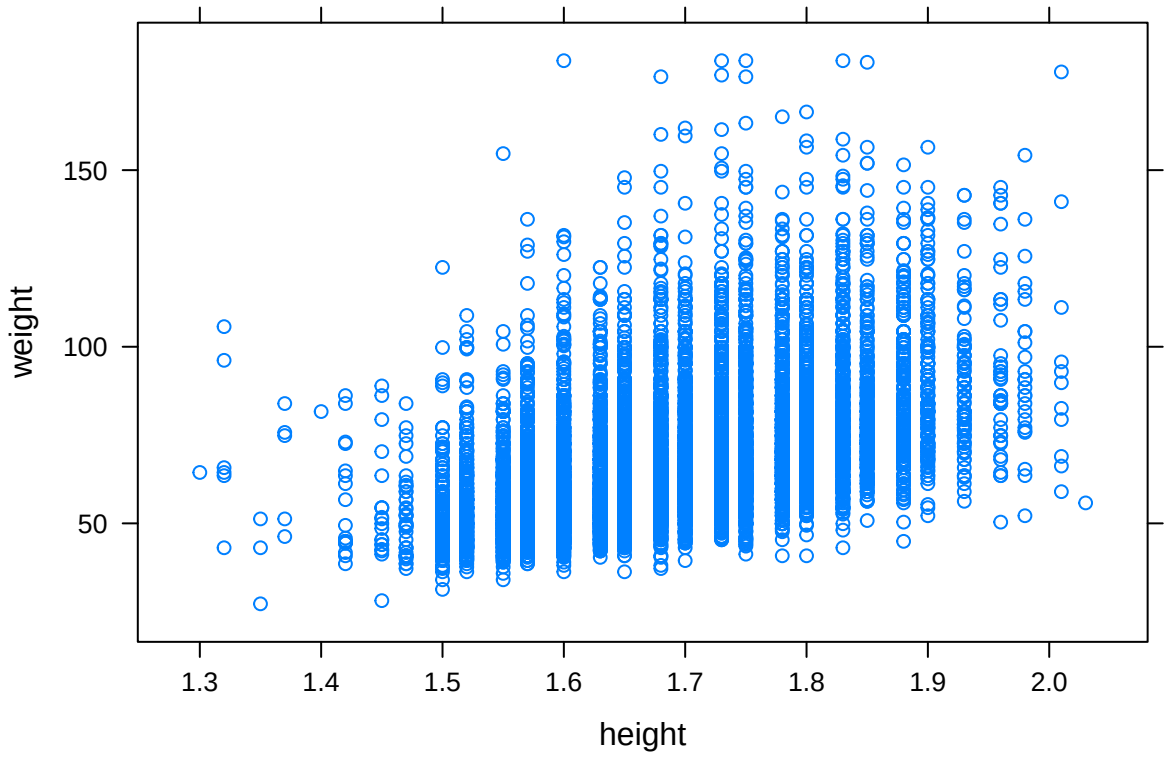
```
xyplot(weight ~ height, data = cdc, scales = list(x = list(draw = FALSE)))
```



tick.number

Determines (roughly) how many tick marks should be on the axis.

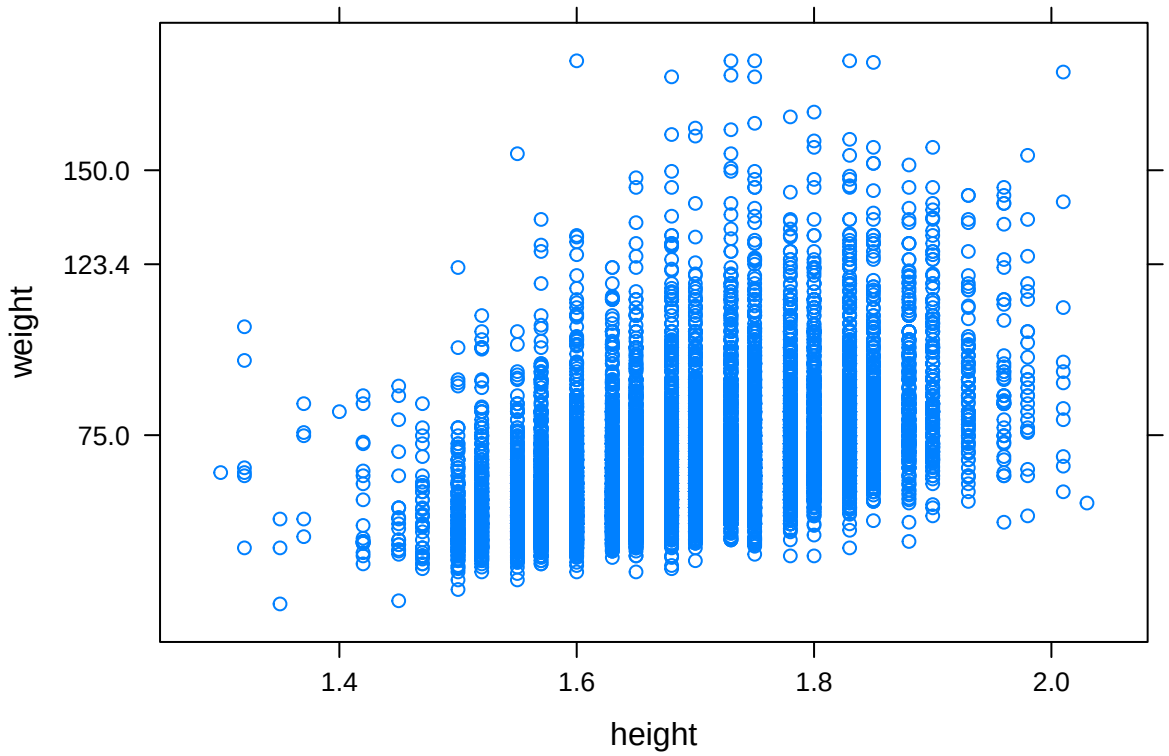
```
xyplot(weight ~ height, data = cdc, scales = list(x = list(tick.number = 8),
                                                  y = list(tick.number = 3)))
```



at

Determines where the tick marks are drawn.

```
xyplot(weight ~ height, data = cdc, scales = list(y = list(at = c(75, 123.4, 150))))
```



labels

Vector of labels to go along with at.

```
xyplot(weight ~ height, data = cdc,
       scales = list(x = list(at = c(1.4, 1.9), labels = c("silly", "example!")),
                     y = list(at = c(150, 100, 50), labels = c("This", "is", "a"))))
```