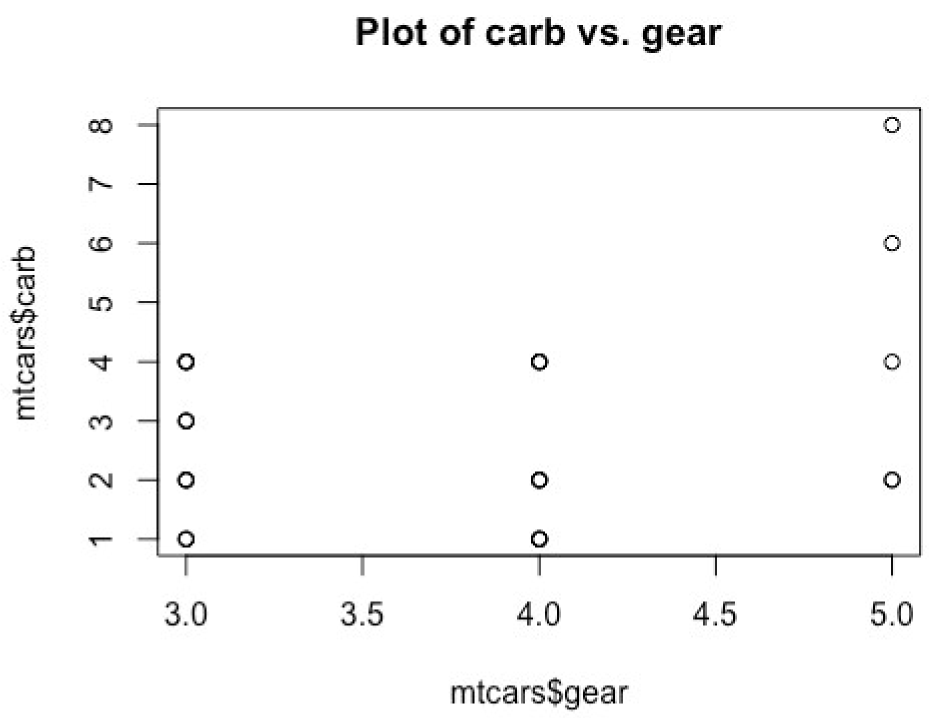
A ***sunflower plot*** is a type of scatterplot which tries to reduce overplotting. When there are multiple points that have the same `(x, y)` values, sunflower plots plot just one point there, but has little edges (or “petals”) coming out from the point to indicate how many points are really there.

It’s best to see this via an example. Here is a plot of `carb` vs. `gear` from the `mtcars` dataset:

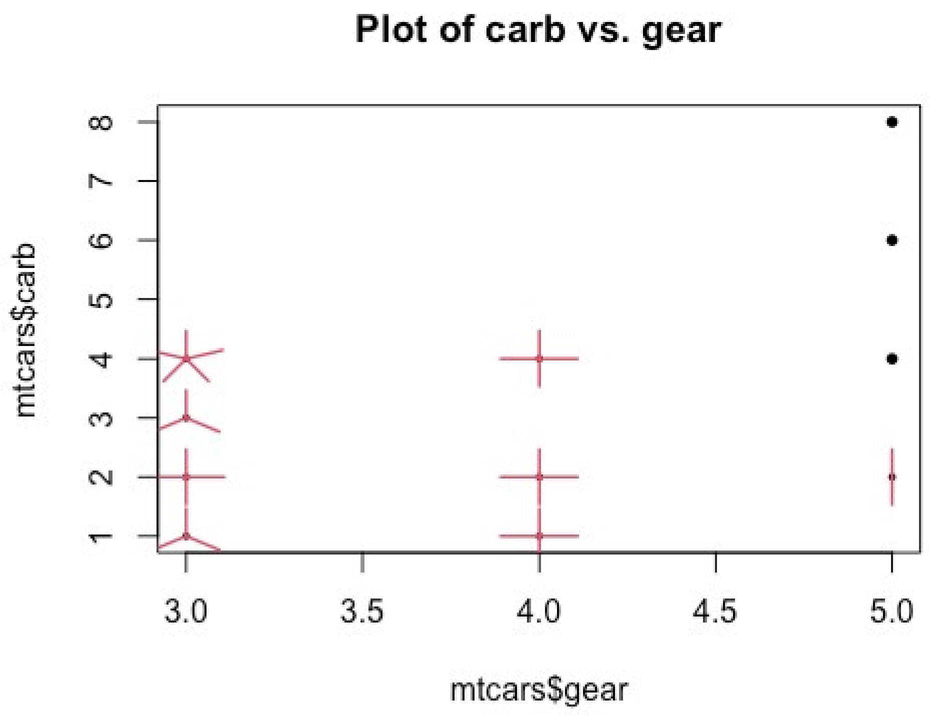
```
plot(mtcars$gear, mtcars$carb,
     main = "Plot of carb vs. gear")
```



From the plot it looks like there are only 11 data points. However, if we check the Environments tab in RStudio we see that there are actually 32 observations in the dataset: it’s just that some of the observations have the same `(gear, carb)` values.

Let’s see how a sunflower plot deals with this overplotting. It turns out that base R comes with a `sunflower` plot function that does just this:

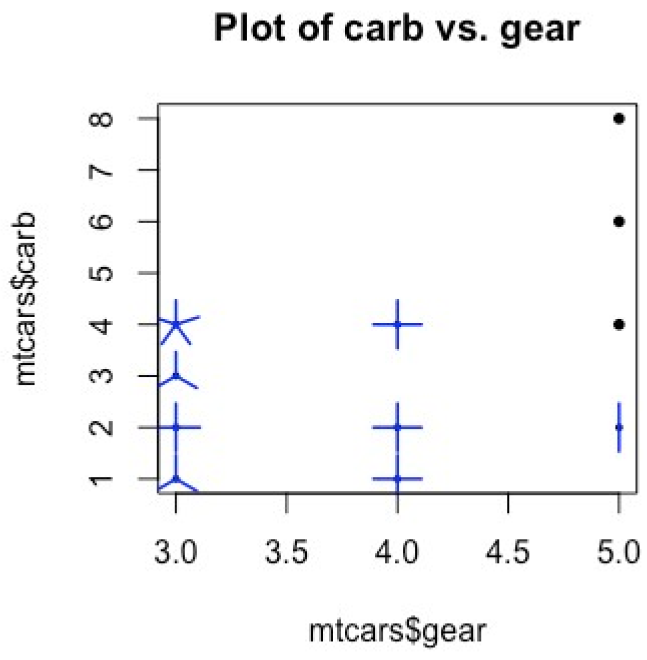
```
sunflowerplot(mtcars$gear, mtcars$carb,
              main = "Plot of carb vs. gear")
```



This tells us, for example, that there are 3 observations with `(gear, carb) = (3, 1).`

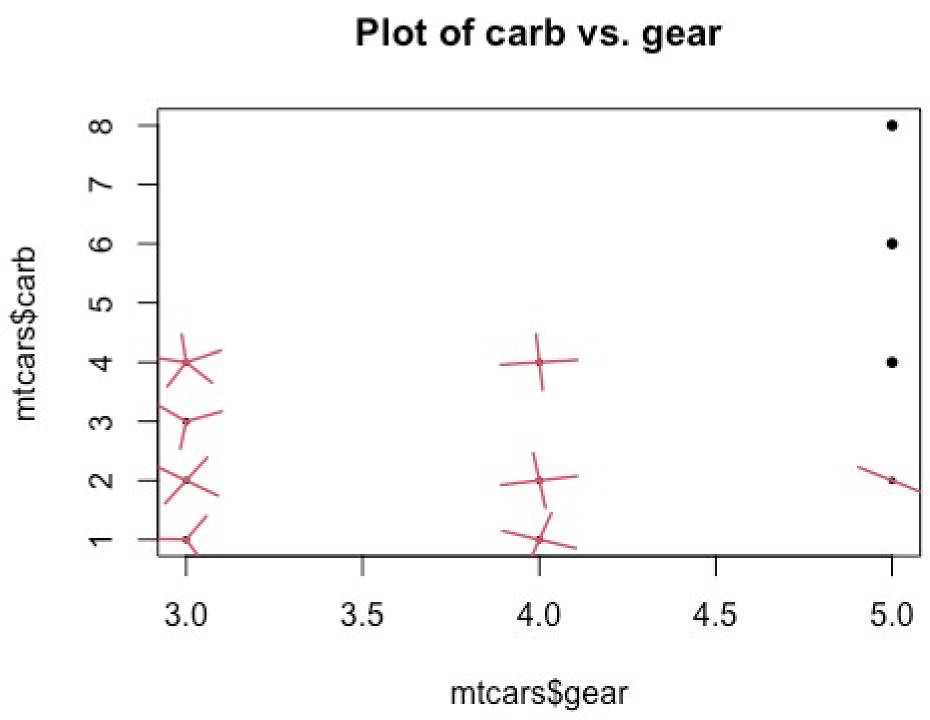
We can change the color of the “petals” by specifying `seg.col`:

```
sunflowerplot(mtcars$gear, mtcars$carb,
              seg.col = "blue",
              main = "Plot of carb vs. gear")
```



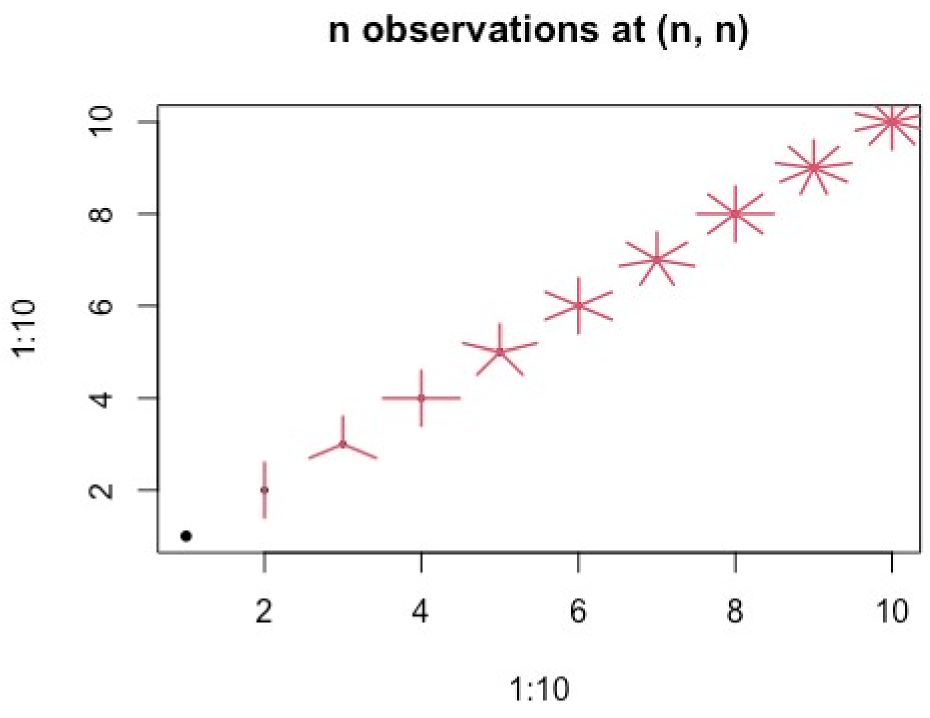
By default, the first petal always points up. To have the first petal point in random directions, specify `rotate = TRUE`:

```
set.seed(1)
sunflowerplot(mtcars$gear, mtcars$carb,
              rotate = TRUE,
              main = "Plot of carb vs. gear")
```



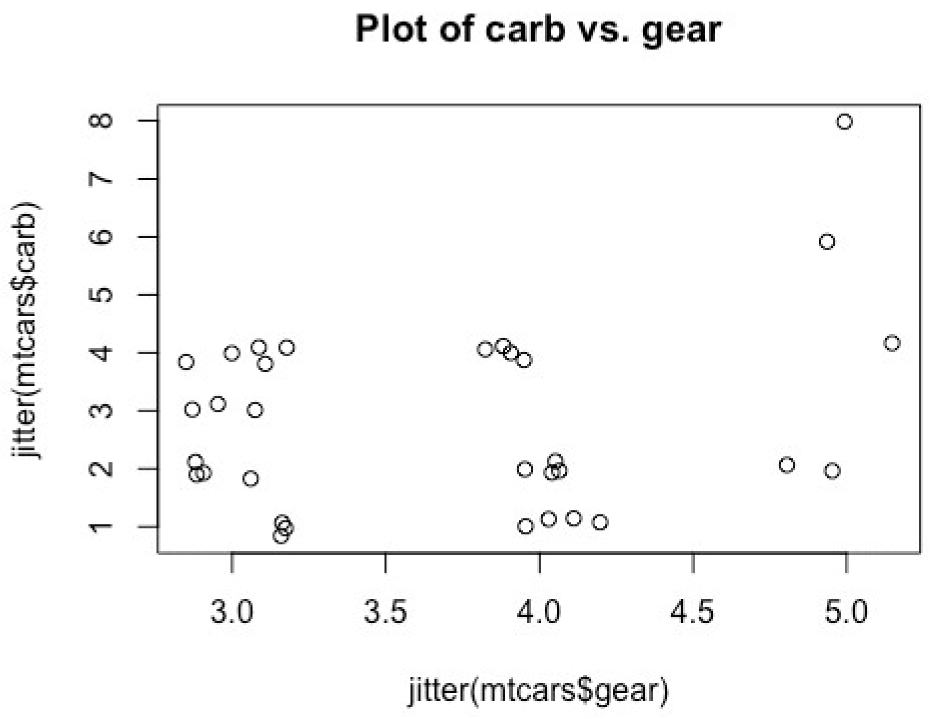
When given `(x, y)` values, `sunflowerplot` counts the number of times each `(x, y)` value appears to determine the number of petals it needs to draw. It is possible to override this behavior by passing a `number` argument, as the next code snippet shows:

```
set.seed(1)
sunflowerplot(1:10, 1:10, number = 1:10,
              main = "n observations at (n, n)")
```



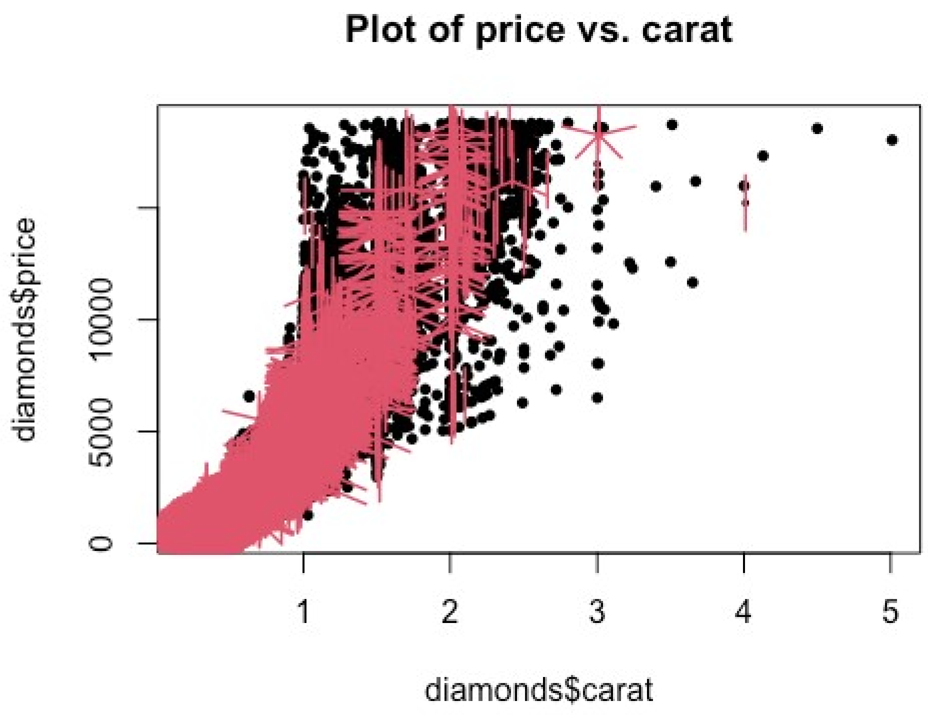
Sunflower plots aren't the only way to reduce overplotting. Another common technique is ***jittering***, where random noise is added to each point. The code below shows how you can do this in base R. Of course, a drawback of this is that the points are not plotted at the exact location of the data.

```
set.seed(1)
plot(jitter(mtcars$gear), jitter(mtcars$carb),
     main = "Plot of carb vs. gear")
```



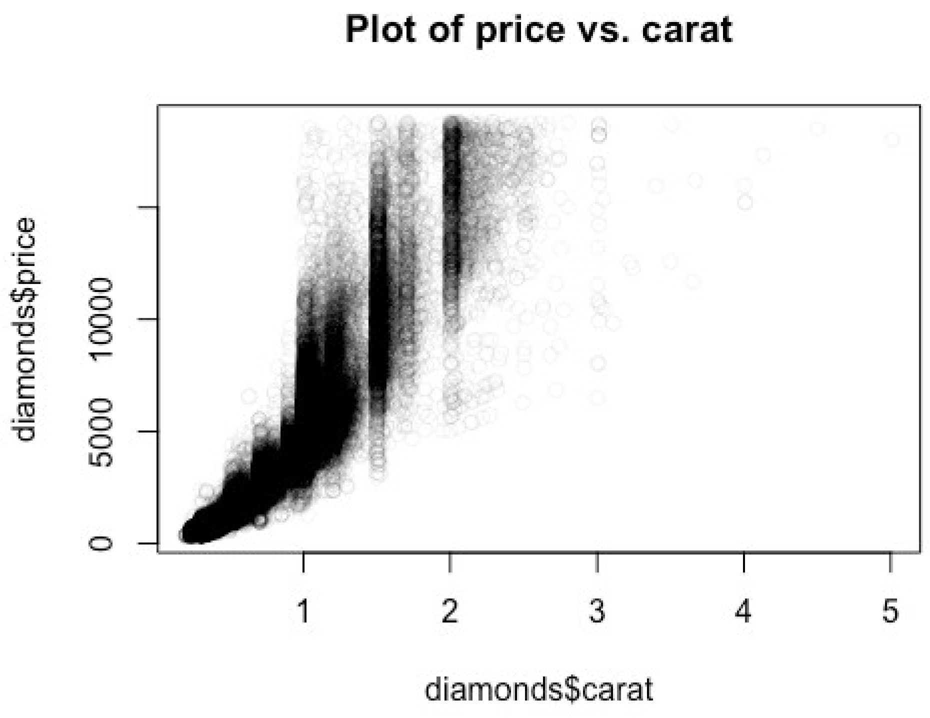
Sunflower plots will not solve all your overplotting issues. Here is an example (on the `diamonds` dataset) where it does a horrendous job:

```
library(ggplot2)
data(diamonds)
sunflowerplot(diamonds$carat, diamonds$price,
              main = "Plot of price vs. carat")
```



A better solution here would be to change the transparency of the points. The code snippet below shows how this can be done in base R:

```
plot(diamonds$carat, diamonds$price,
     col = rgb(0, 0, 0, alpha = 0.05),
     main = "Plot of price vs. carat")
```


Plot of price vs. carat
diamonds$price
0
5000
10000
diamonds$carat
1
2
3
4
5