Exercise 1

Exercise introduction

This is Exercise 1 in Part 1 of the course.

The purpose of the exercise is to introduce `R`, RStudio, and R Markdown to students.

Basic arithmetic operations

`R` can handle arithmetic.

```
15+25
```

```
## [1] 40
```

```
4^4
```

```
## [1] 256
```

Observing datasets available for loading

`R` comes with many datasets that can be loaded.

```
data()
```

Examining the `mtcars` dataset

This block loads the `mtcars` dataset, uses the `?` operator to access help files for the dataset as well as the `names` function, and then provides a summarizes the dataset using the `summary` function.

```
data(mtcars)
mtcars
```

```
##                    mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## Duster 360        14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## Merc 240D         24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
```

```
## Merc 230            22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## Merc 280            19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
## Merc 280C           17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
## Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
## Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
## Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
## Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
## Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
## Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
## Fiat 128            32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
## Honda Civic         30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
## Toyota Corolla      33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
## Toyota Corona       21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
## Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
## AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
## Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
## Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
## Fiat X1-9           27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## Porsche 914-2       26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
## Lotus Europa        30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
## Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
## Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
## Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
## Volvo 142E          21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

```
?mtcars
```

```
## starting httpd help server ... done
```

```
?names
names(mtcars)
```

```
##  [1] "mpg"  "cyl"  "disp" "hp"   "drat" "wt"   "qsec" "vs"   "am"   "gear"
## [11] "carb"
```

```
summary(mtcars)
```

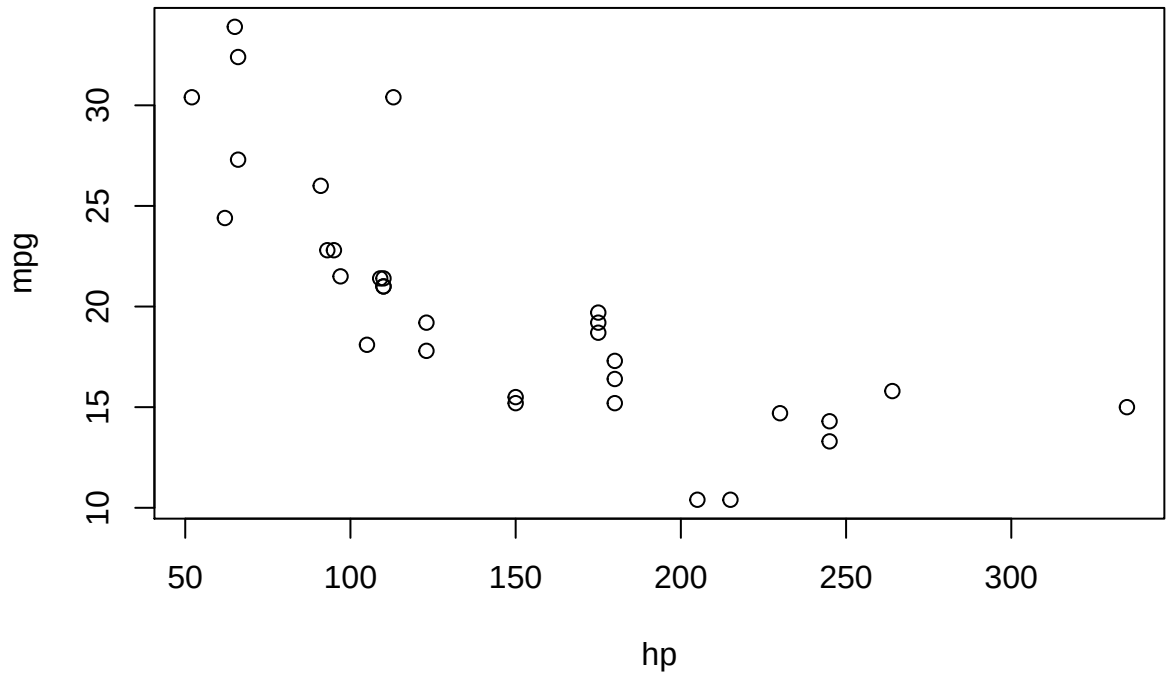
```
##       mpg             cyl             disp             hp
##  Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0
##  1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5
##  Median :19.20   Median :6.000   Median :196.3   Median :123.0
##  Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7
##  3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0
##  Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0
##       drat             wt             qsec             vs
##  Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000
##  1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000
##  Median :3.695   Median :3.325   Median :17.71   Median :0.0000
##  Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375
##  3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000
##  Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000
##        am              gear            carb
```

```
##  Min.   :0.0000   Min.   :3.000   Min.   :1.000
##  1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000
##  Median :0.0000   Median :4.000   Median :2.000
##  Mean   :0.4062   Mean   :3.688   Mean   :2.812
##  3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000
##  Max.   :1.0000   Max.   :5.000   Max.   :8.000
```

Plotting from `mtcars`

The default behavior of the `plot` function for two continuous variables is a scatter plot. Note the use of the `~` operator to indicate an implied dependent relationship; in this example, the implication is that miles per gallon (`mpg`) are dependent upon horsepower (`hp`), and thus we plot `hp` on the x-axis and `mpg` on the y-axis since this is typically how we visualize implied dependent relationships.

```
plot(mpg ~ hp, mtcars)
```



Installing and loading packages

The typical workflow when accessing a new package is to first use `install.packages` and then `library`. However, I find this a bit verbose and created a function `activatePkgs` to automate the workflow; if the package is not already installed, the function first installs the package before loading it.

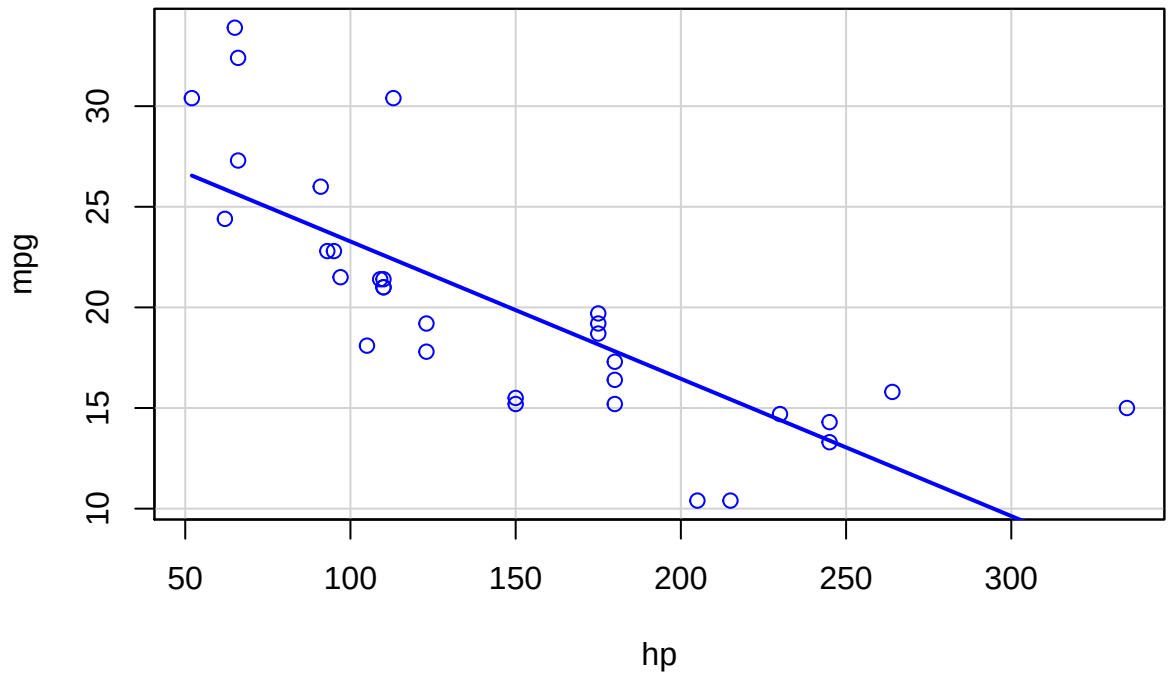
The rest of this code block creates a more-customized scatter plot with a simple linear-regression line. The chunk parameter `warning` is set to `FALSE` so as to silence warning messages from appearing in the final 'knit' document.

```
activatePkgs('car')
```

```
## Loading required package: car
```

```
## Loading required package: carData
```

```
scatterplot(mpg~hp, reg.line=lm, smooth=FALSE, spread=FALSE, boxplots=FALSE, span=0.5, ellipse=FALSE, da
```



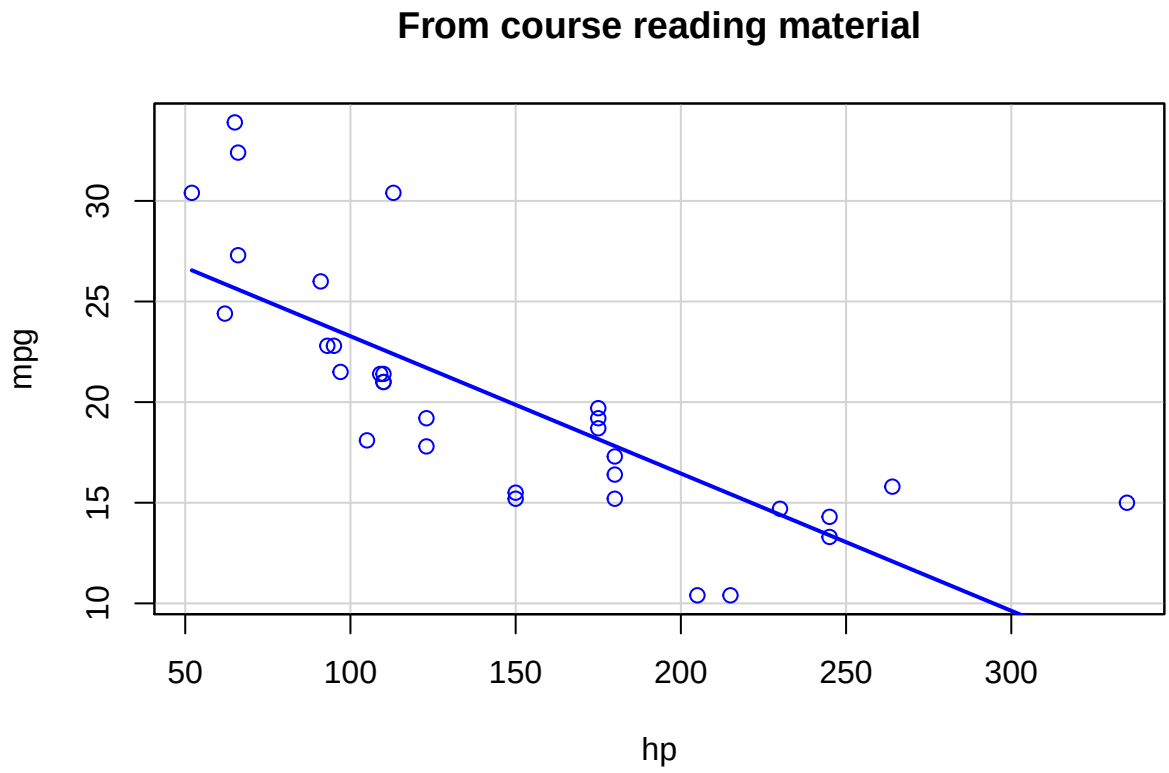
Modifying the scatter plot parameters

When I am exposed to a new function with parameters whose purpose is not immediately clear to me, I tend to reformat my code so that it is 'vertical' with 'leading commas' separating parameters. This makes it easy to quickly comment out individual lines with keyboard shortcuts (`ctrl` + `shift` + `c` in RStudio for Windows, `ctrl` + `/` in my VS Studio Code Insiders environment). After commenting out a line, I can re-run the code to see the effects of that parameter.

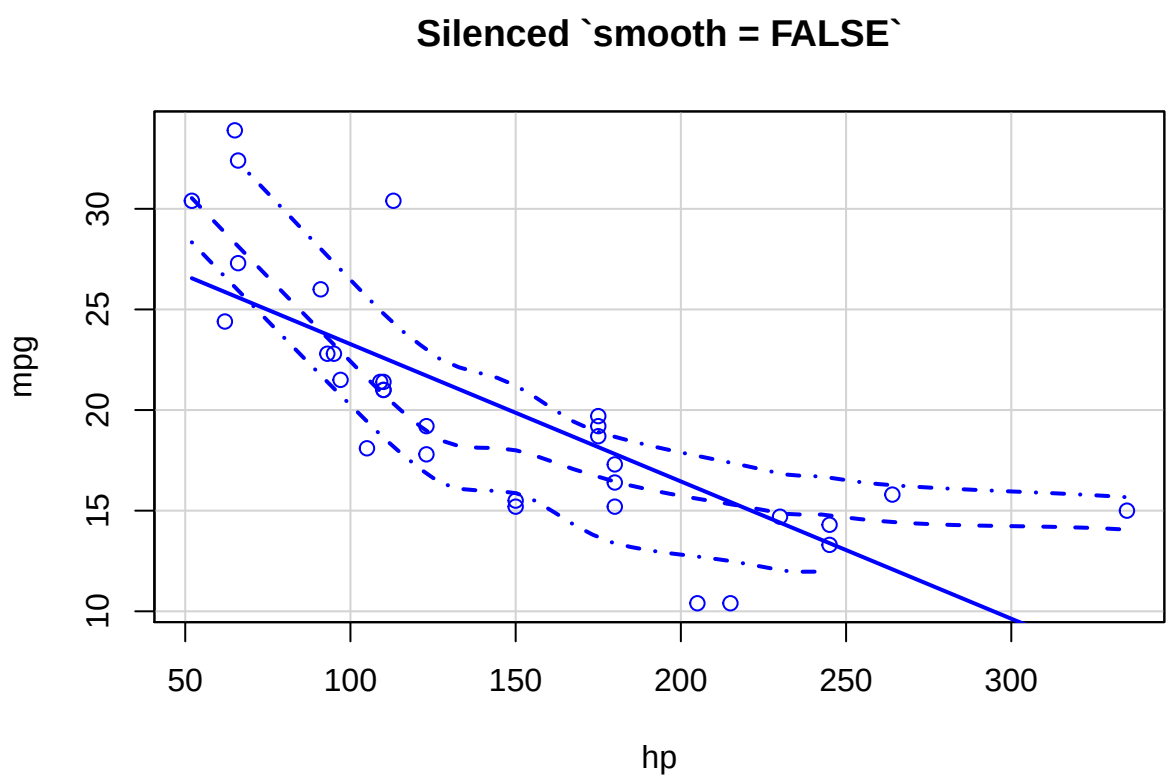
I added titles to each of the plots using the `main` parameter.

```
scatterplot(mpg~hp
            , reg.line=lm
            , smooth=FALSE
            , spread=FALSE
            , boxplots=FALSE
```

```
            , span=0.5
            , ellipse=FALSE
            , data=mtcars
            , main = 'From course reading material'
            )
```

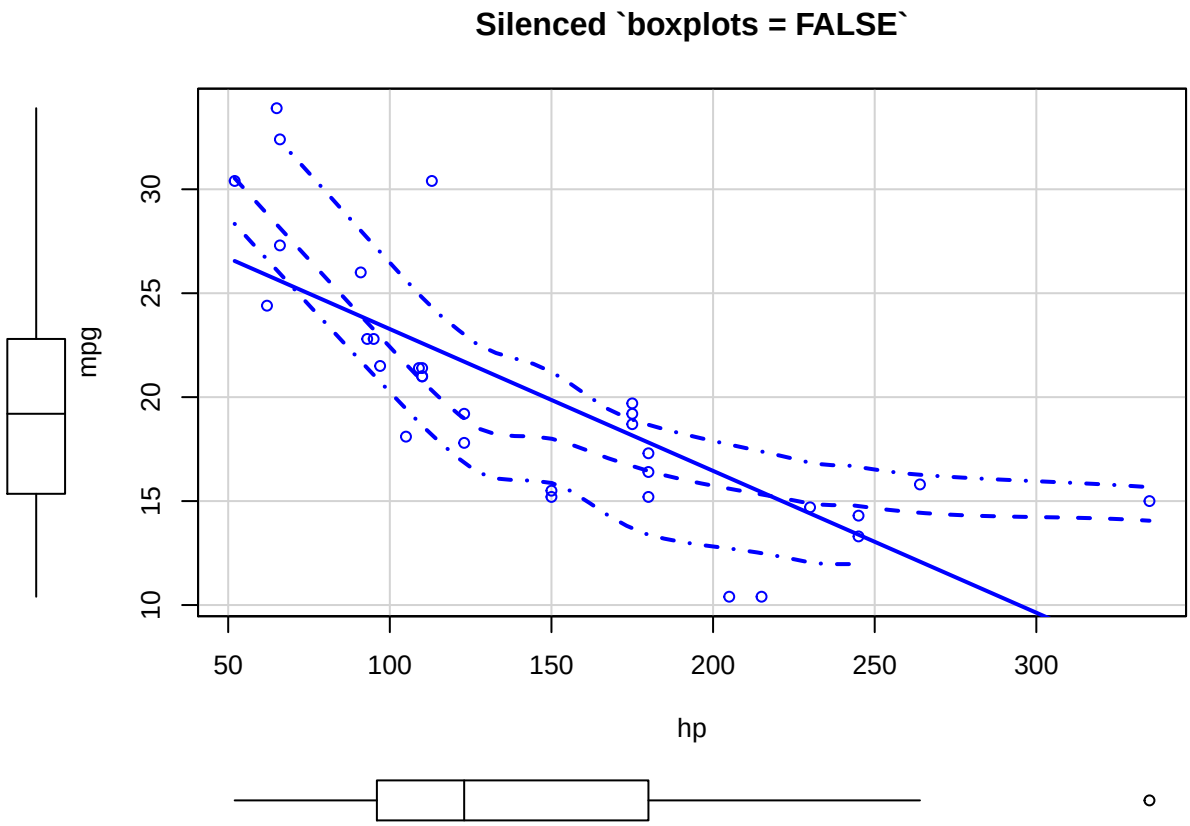



```
scatterplot(mpg~hp
            , reg.line=lm
            # , smooth=FALSE
            , spread=FALSE
            , boxplots=FALSE
            , span=0.5
            , ellipse=FALSE
            , data=mtcars
            , main = 'Silenced `smooth = FALSE`'
            )
```

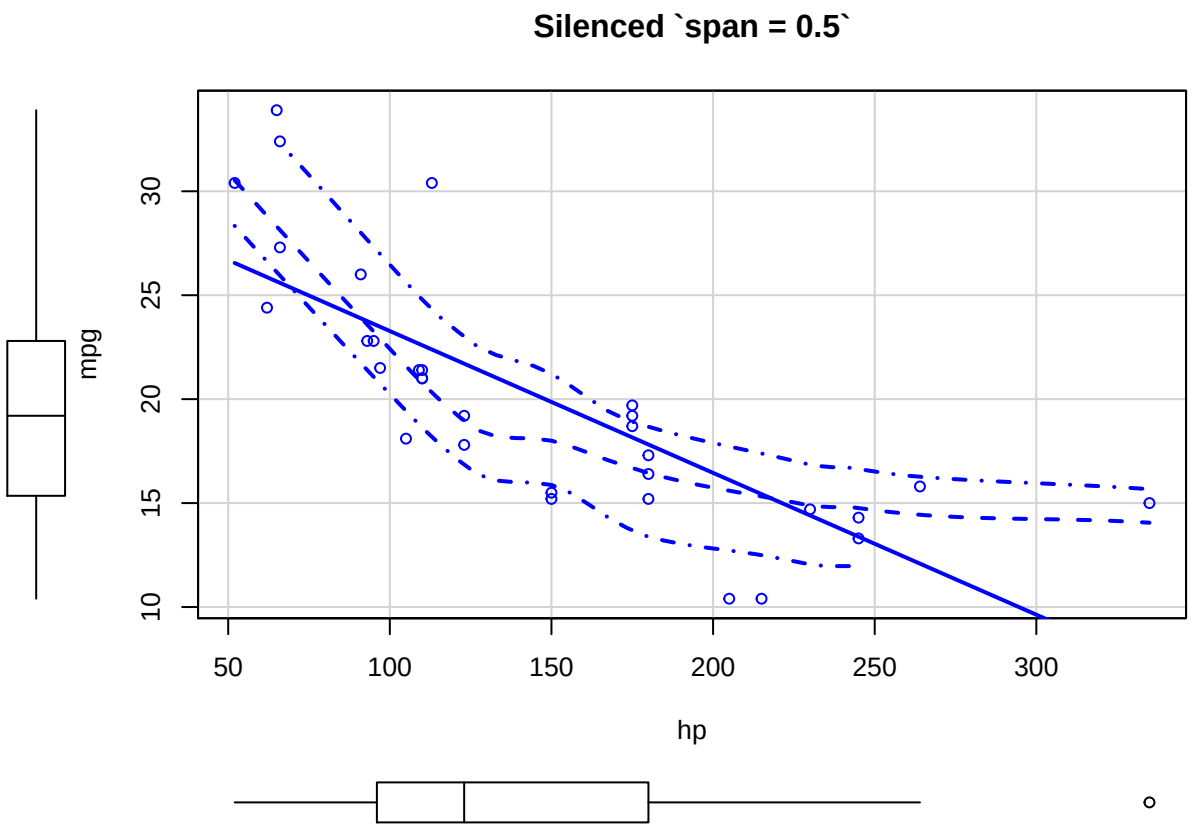



```
scatterplot(mpg~hp
            , reg.line=lm
            # , smooth=FALSE
            # , spread=FALSE
            , boxplots=FALSE
            , span=0.5
            , ellipse=FALSE
            , data=mtcars
            , main = 'Silenced `spread = FALSE`'
            )
```

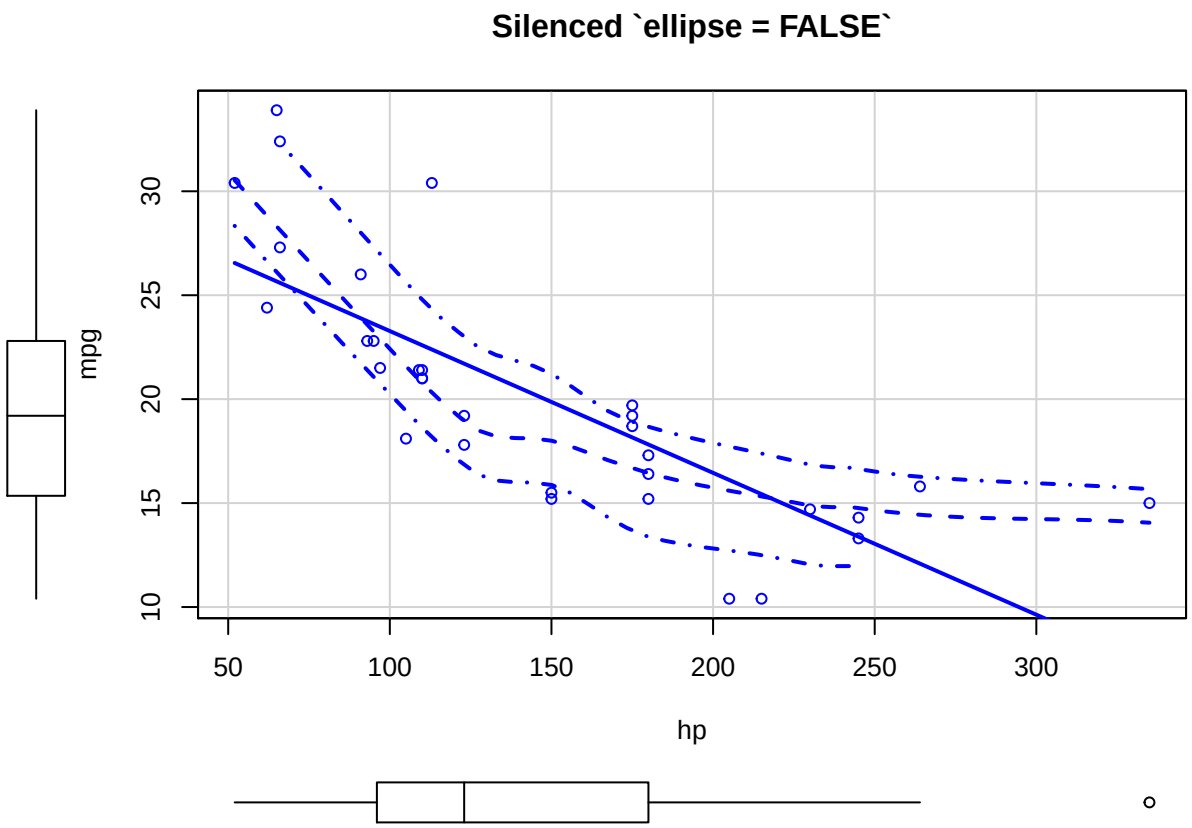

```
scatterplot(mpg~hp
            , reg.line=lm
            # , smooth=FALSE
            # , spread=FALSE
            # , boxplots=FALSE
            , span=0.5
            , ellipse=FALSE
            , data=mtcars
            , main = 'Silenced `boxplots = FALSE`'
            )
```




```
scatterplot(mpg~hp
            , reg.line=lm
            # , smooth=FALSE
            # , spread=FALSE
            # , boxplots=FALSE
            # , span=0.5
            , ellipse=FALSE
            , data=mtcars
            , main = 'Silenced `span = 0.5`'
            )
```



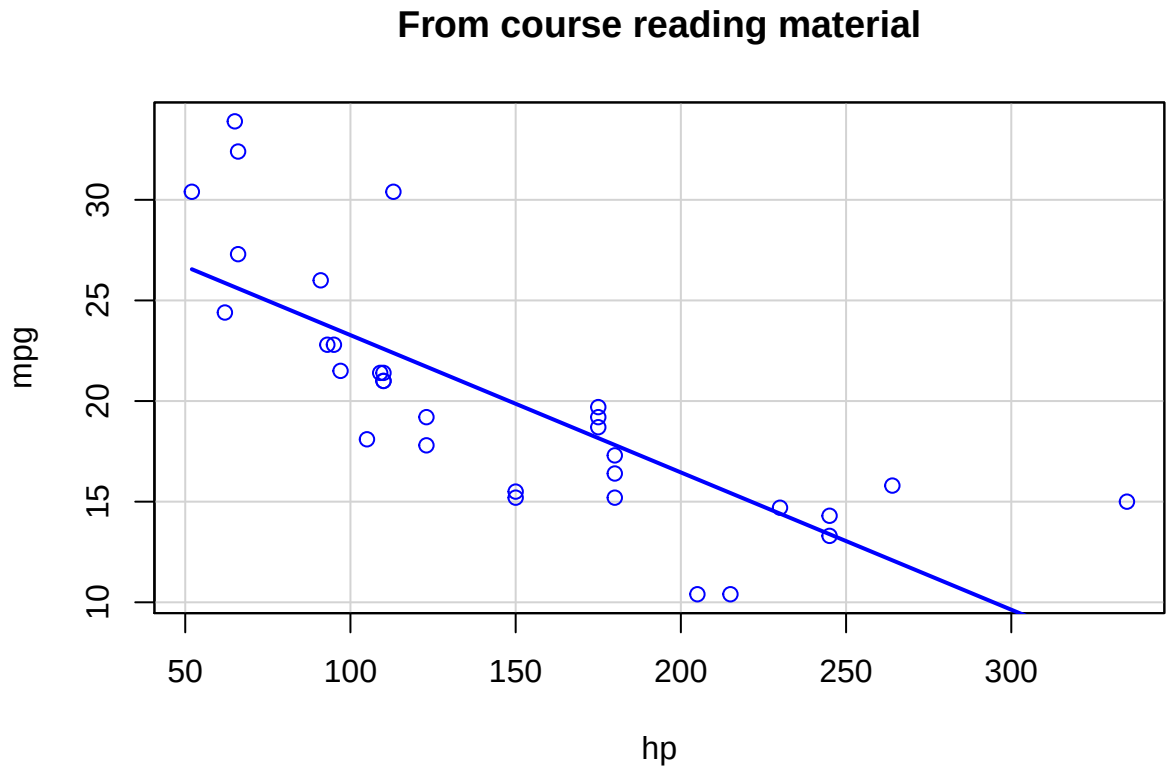

```
scatterplot(mpg~hp
            , reg.line=lm
            # , smooth=FALSE
            # , spread=FALSE
            # , boxplots=FALSE
            # , span=0.5
            # , ellipse=FALSE
            , data=mtcars
            , main = 'Silenced `ellipse = FALSE`'
            )
```



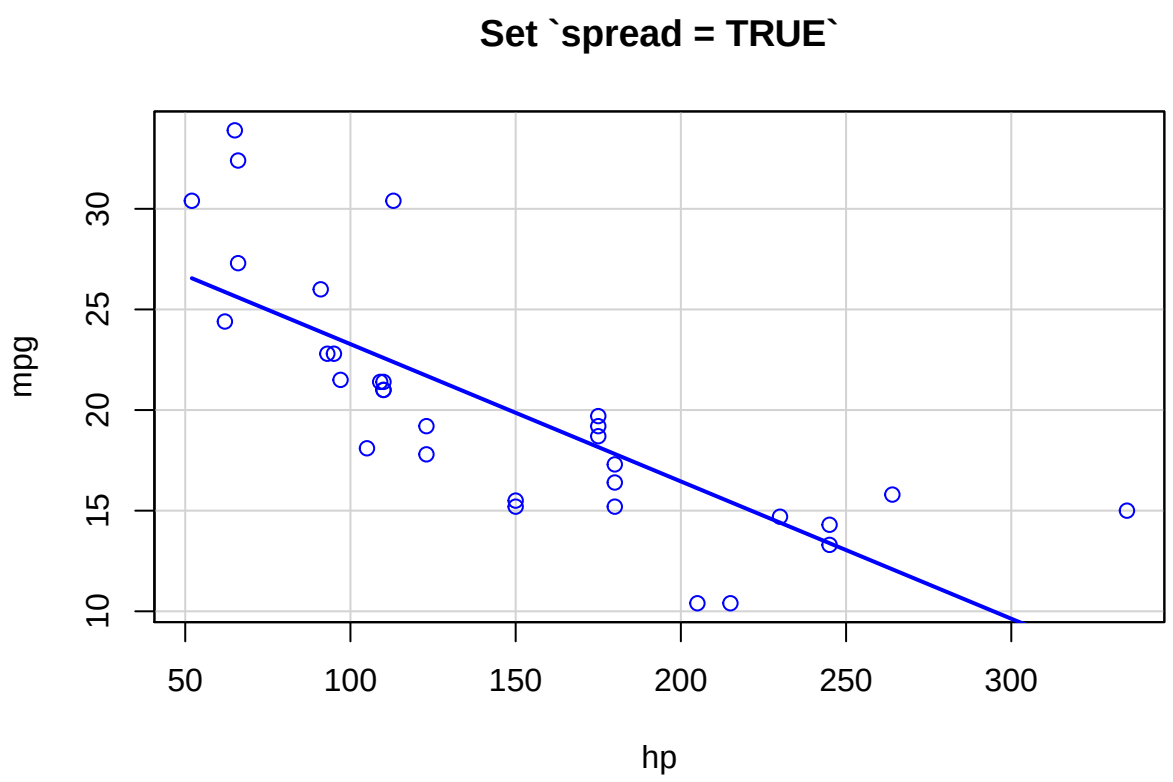
Further exploration of `spread`, `span`, and `ellipse`

The parameters `spread`, `span`, and `ellipse` did not seem to affect the plot output in this context. Perhaps the values provided for these parameters in the course's reading material are their default values, and thus commenting them out did not make a difference. Here, I will try to alter their values from those provided in the reading material to try to discover how they work.

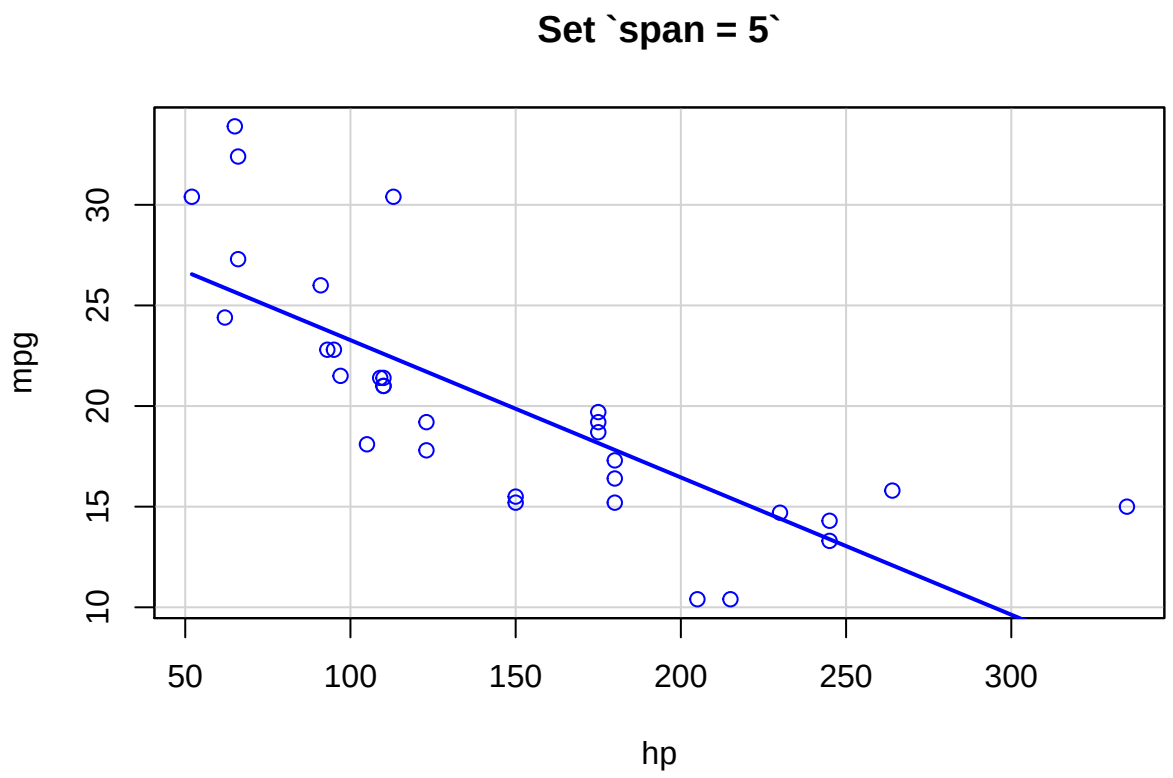
```
scatterplot(mpg~hp
            , reg.line=lm
            , smooth=FALSE
            , spread=FALSE
            , boxplots=FALSE
            , span=0.5
            , ellipse=FALSE
            , data=mtcars
            , main = 'From course reading material'
            )
```



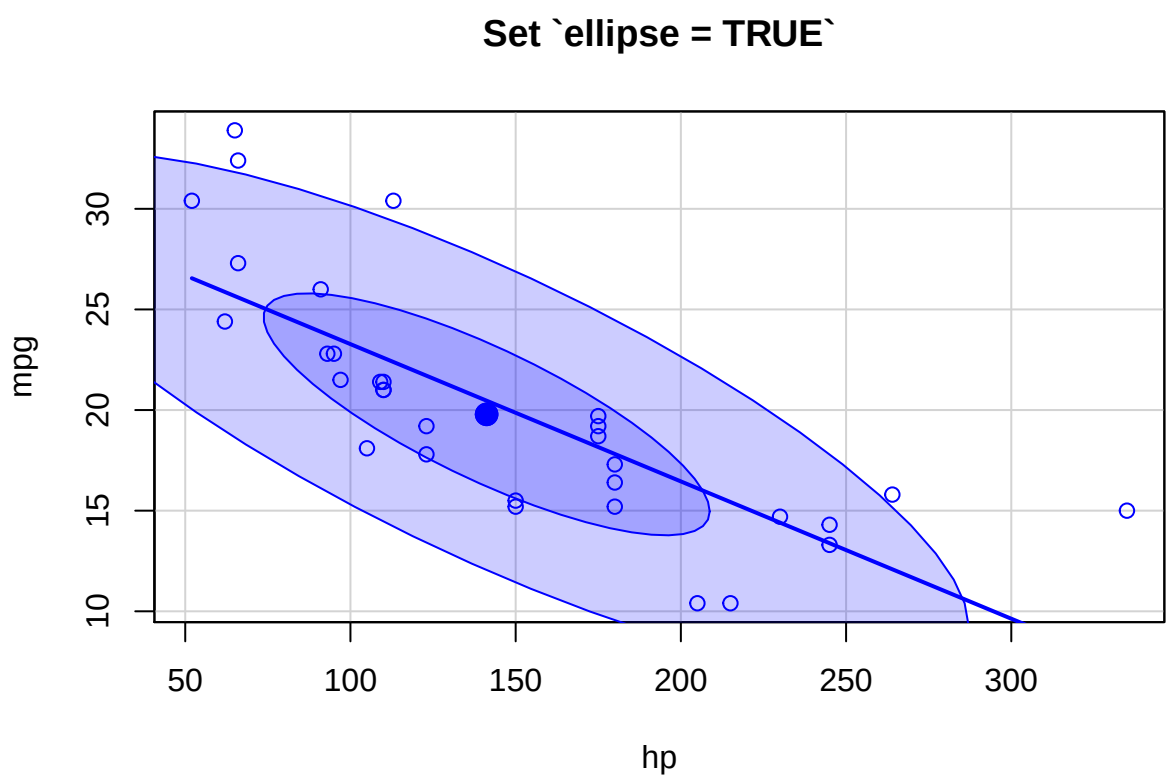

```
scatterplot(mpg~hp
            , reg.line=lm
            , smooth=FALSE
            , spread=TRUE
            , boxplots=FALSE
            , span=0.5
            , ellipse=FALSE
            , data=mtcars
            , main = 'Set `spread = TRUE`'
            )
```




```
scatterplot(mpg~hp
            , reg.line=lm
            , smooth=FALSE
            , spread=TRUE
            , boxplots=FALSE
            , span=5
            , ellipse=FALSE
            , data=mtcars
            , main = 'Set `span = 5`'
            )
```



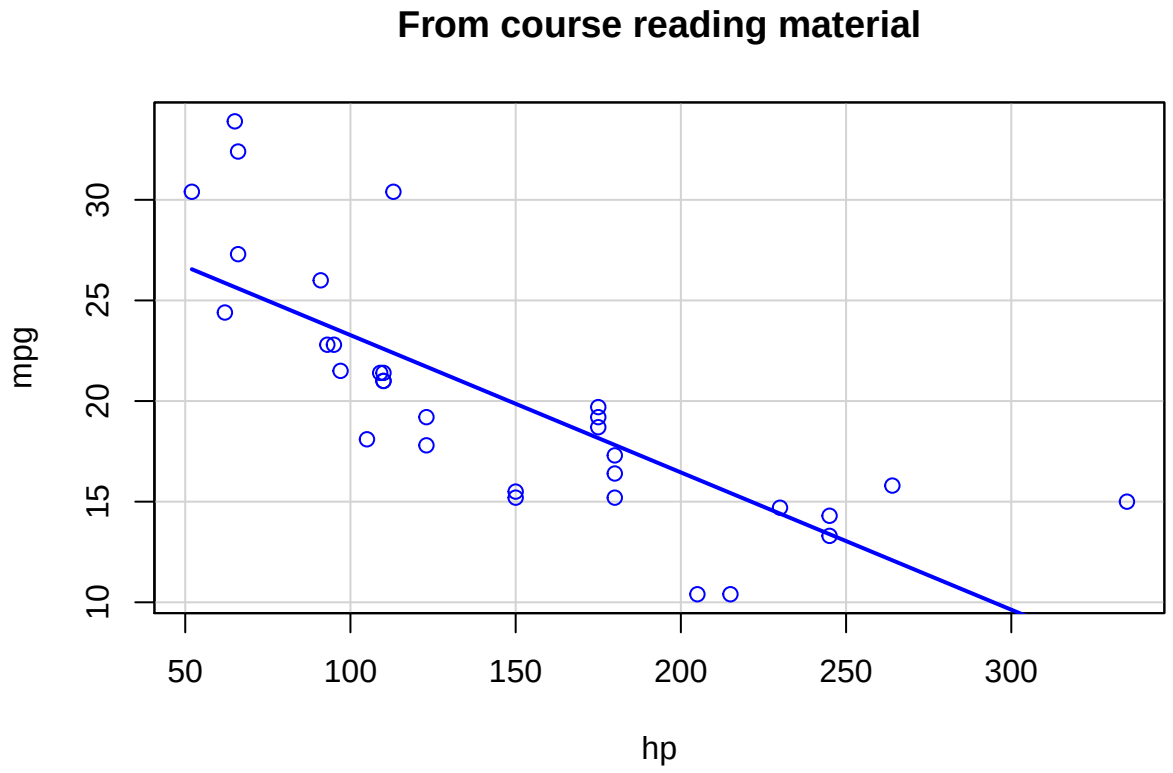

```
scatterplot(mpg~hp
            , reg.line=lm
            , smooth=FALSE
            , spread=TRUE
            , boxplots=FALSE
            , span=5
            , ellipse=TRUE
            , data=mtcars
            , main = 'Set `ellipse = TRUE`'
            )
```



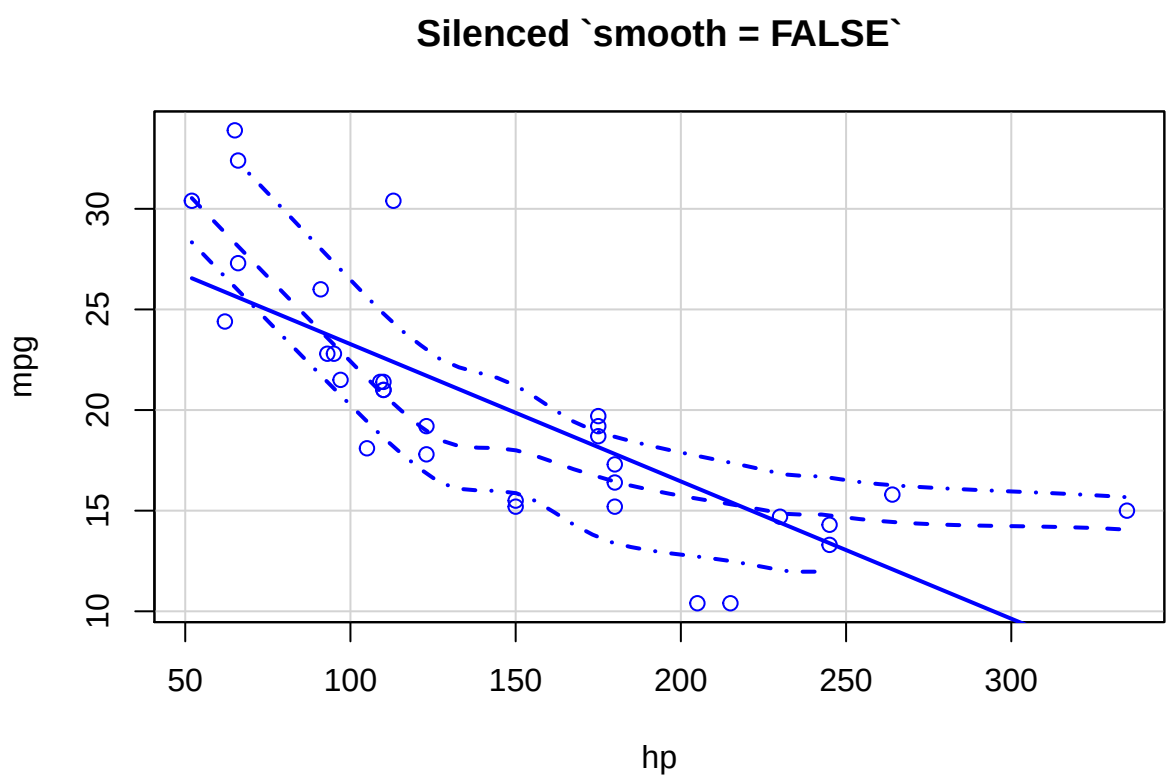
Discovering purpose of the parameters `spread` and `span` is elusive!

The help page called forth by `?scatterplot` indicates that `spread` is a sub-argument of the `smooth` argument. Perhaps it's behavior is only observable when `spread = TRUE`.

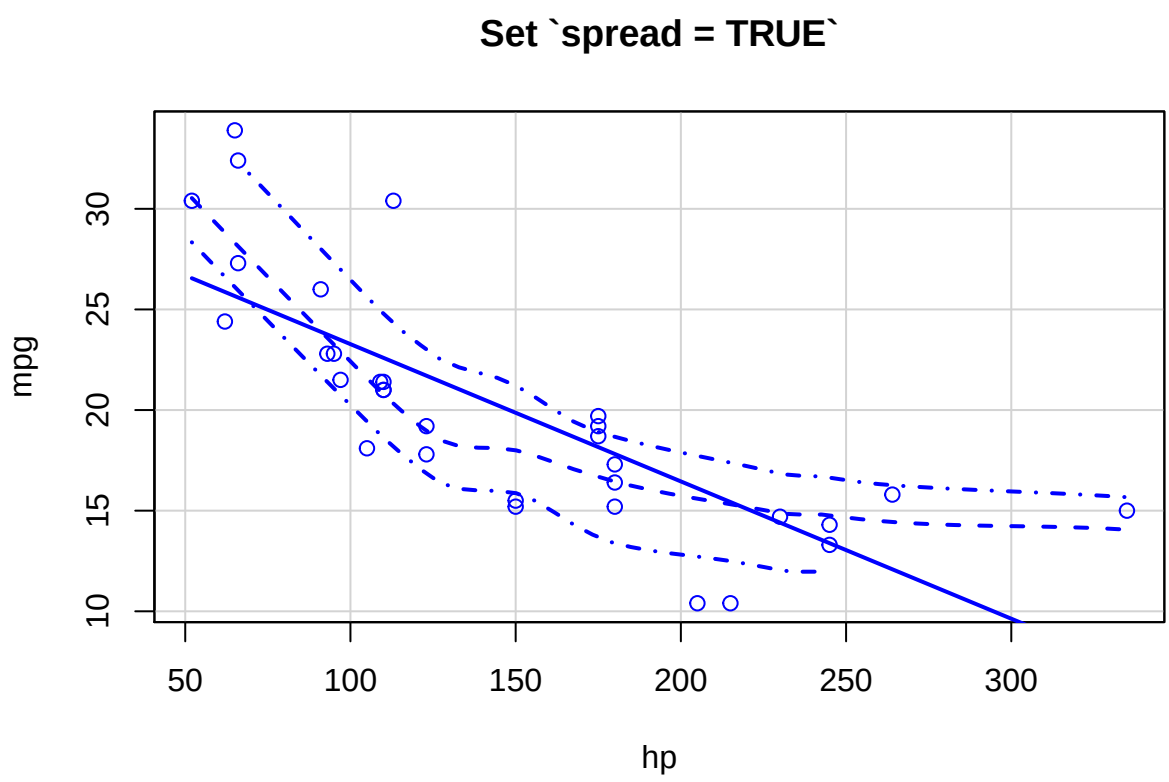
```
?scatterplot
scatterplot(mpg~hp
            , reg.line=lm
            , smooth=FALSE
            , spread=FALSE
            , boxplots=FALSE
            , span=0.5
            , ellipse=FALSE
            , data=mtcars
            , main = 'From course reading material'
            )
```




```
scatterplot(mpg~hp
            , reg.line=lm
            # , smooth=FALSE
            , spread=FALSE
            , boxplots=FALSE
            , span=0.5
            , ellipse=FALSE
            , data=mtcars
            , main = 'Silenced `smooth = FALSE`'
            )
```




```
scatterplot(mpg~hp
            , reg.line=lm
            # , smooth=FALSE
            , spread=TRUE
            , boxplots=FALSE
            , span=0.5
            , ellipse=FALSE
            , data=mtcars
            , main = 'Set `spread = TRUE`'
            )
```



Quit while you're ahead

Even the help documentation for `scatterplot` did not help me discover the effect of the `spread` or `span` parameters, unfortunately.

Key learnings

- There are many datasets in `R` that can be used for experimenting with code. Use `data()` to see an overview and `data(<dataset>)` to load a dataset of interest.
- From the reading material for this lesson I learned that one can use the cogwheel in code chunks to modify chunk-specific options such as `include` and `echo`.
- From some explorations on the internet, I figured out how to direct 'knit' R Markdown files to my desired output folder `p03_outputs` by modifing the `YAML` header as such:

  ```
  knit: (function(inputFile, encoding) { rmarkdown::render(inputFile, encoding = encoding, output_dir = paste0(getwd(), "p03_outputs")) })
  ```

- The `car::scatterplot` function can plot linear models, smoothed regions of confidence, accompanying boxplots, and even an ellipse over the data points.
- There is also the user interface `RCommander`, but it did not seem as useful as RStudio and so I will not be using it.

Unresolved questions

- I was unable to find a way to 'knit' the same R Markdown file to multiple outputs simultaneously. Specifically, I would like to generate word, pdf, and html files in a single 'knit' operation, but it seems I have to manually change the `output` parameter in the `YAML` header and then re-'knit' the document to get different outputs.
- Although I tried I did not figure out how parameters `spread` and `span` in the `car::scatterplot` function modify the function's output. I simply observed that in this specific use case, including them or commenting them out made no difference.
- I did not figure out how to send the results of `?<R object>` to the output of a code chunk so that these results are also included in the final 'knit' document.