sb_rmd_ex_06

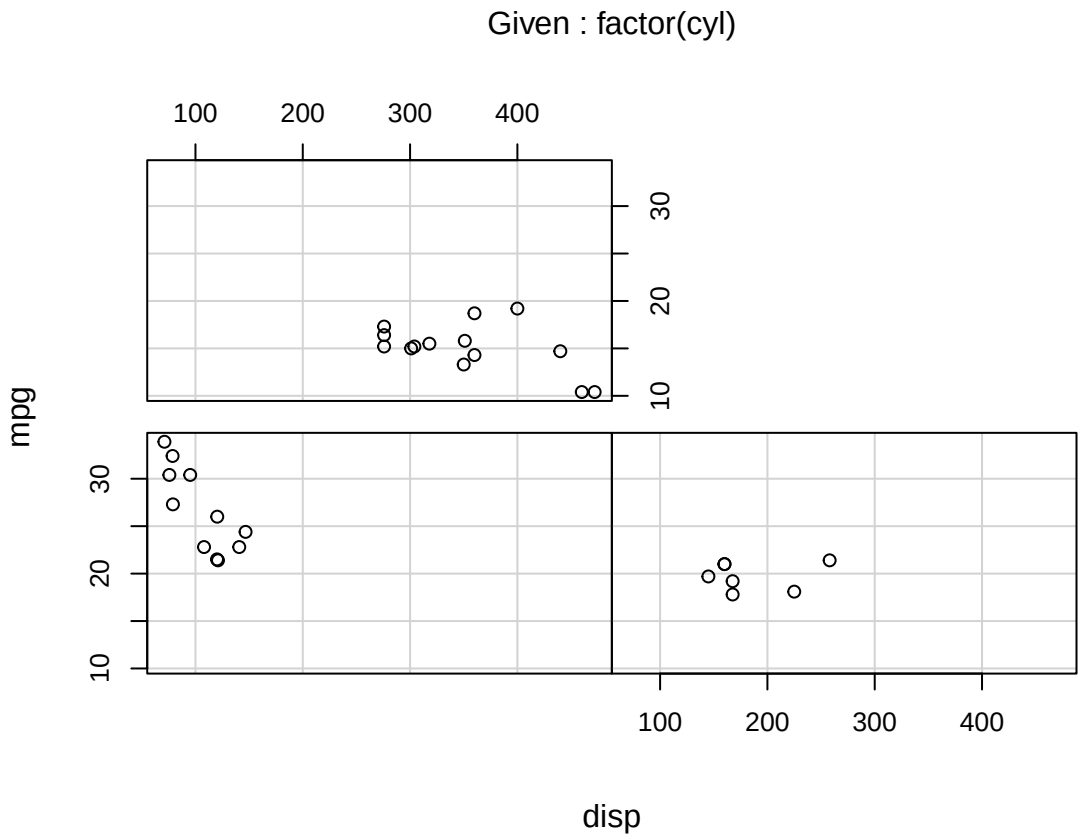
Sander Bossier

2024-01-15

1.

R has a core function `coplot()` to do multi-panel conditioning. Use it on the (core) dataset mtcars to create the following plot:

```
coplot(
  formula = mpg ~ disp | factor(cyl),
  data = mtcars,
  show.given = FALSE)
```

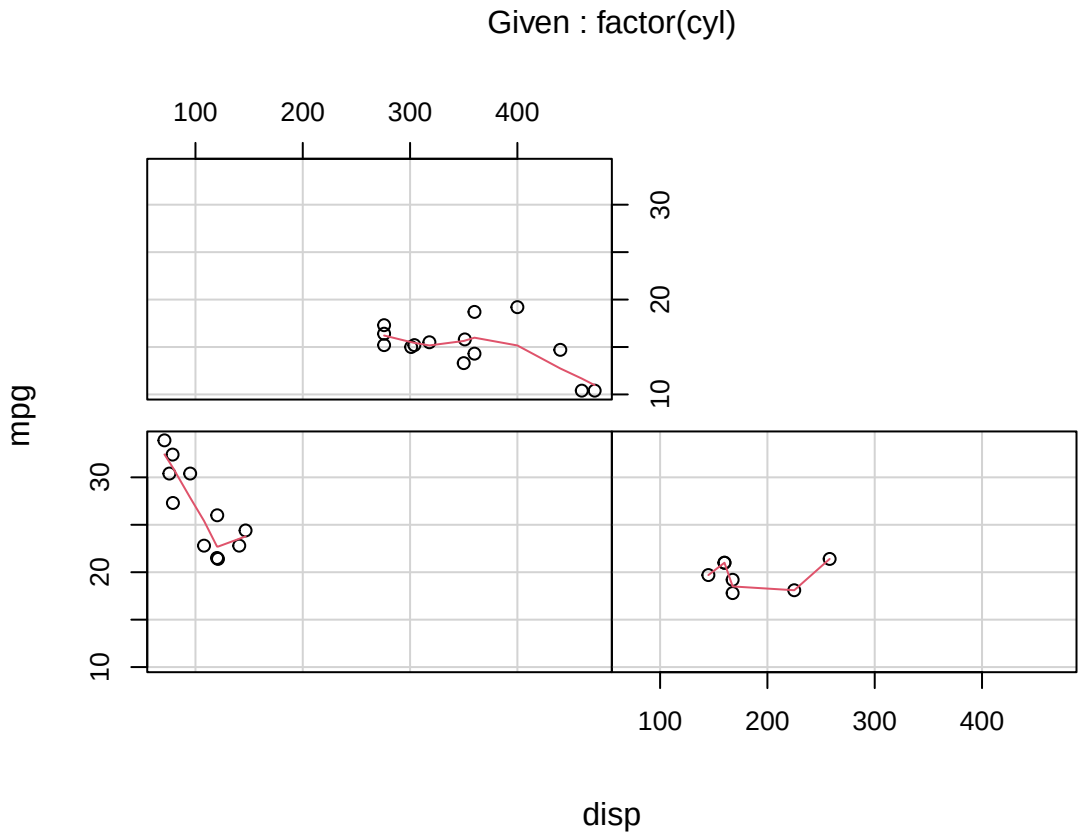


2.

Add a LOESS smoother to the previous conditioning plot:

```
coplot(
  formula = mpg ~ disp | factor(cyl),
```

```
  data = mtcars,
  panel = function(x, y, ...)
    panel.smooth(x, y, ...)
  ,
  show.given = FALSE)
```



3.

Use the `par()` function to check R's different coordinate systems: verify that changing one graphical parameter indeed affects other graphical parameters.

```
par()
```

```
## $xlog
## [1] FALSE
##
## $ylog
## [1] FALSE
##
## $adj
## [1] 0.5
##
## $ann
## [1] TRUE
##
## $ask
```

```
## [1] FALSE
##
## $bg
## [1] "transparent"
##
## $bty
## [1] "o"
##
## $cex
## [1] 1
##
## $cex.axis
## [1] 1
##
## $cex.lab
## [1] 1
##
## $cex.main
## [1] 1.2
##
## $cex.sub
## [1] 1
##
## $cin
## [1] 0.15 0.20
##
## $col
## [1] "black"
##
## $col.axis
## [1] "black"
##
## $col.lab
## [1] "black"
##
## $col.main
## [1] "black"
##
## $col.sub
## [1] "black"
##
## $cra
## [1] 10.8 14.4
##
## $crt
## [1] 0
##
## $csi
## [1] 0.2
##
## $cxy
## [1] 0.02851711 0.07518797
##
## $din
```

```
## [1] 6.5 4.5
##
## $err
## [1] 0
##
## $family
## [1] ""
##
## $fg
## [1] "black"
##
## $fig
## [1] 0 1 0 1
##
## $fin
## [1] 6.5 4.5
##
## $font
## [1] 1
##
## $font.axis
## [1] 1
##
## $font.lab
## [1] 1
##
## $font.main
## [1] 2
##
## $font.sub
## [1] 1
##
## $lab
## [1] 5 5 7
##
## $las
## [1] 0
##
## $lend
## [1] "round"
##
## $lheight
## [1] 1
##
## $ljoin
## [1] "round"
##
## $lmitre
## [1] 10
##
## $lty
## [1] "solid"
##
## $lwd
```

```
## [1] 1
##
## $mai
## [1] 1.02 0.82 0.82 0.42
##
## $mar
## [1] 5.1 4.1 4.1 2.1
##
## $mex
## [1] 1
##
## $mfcol
## [1] 1 1
##
## $mfg
## [1] 1 1 1 1
##
## $mfrow
## [1] 1 1
##
## $mgp
## [1] 3 1 0
##
## $mkh
## [1] 0.001
##
## $new
## [1] FALSE
##
## $oma
## [1] 0 0 0 0
##
## $omd
## [1] 0 1 0 1
##
## $omi
## [1] 0 0 0 0
##
## $page
## [1] TRUE
##
## $pch
## [1] 1
##
## $pin
## [1] 5.26 2.66
##
## $plt
## [1] 0.1261538 0.9353846 0.2266667 0.8177778
##
## $ps
## [1] 12
##
## $pty
```

```
## [1] "m"
##
## $smo
## [1] 1
##
## $srt
## [1] 0
##
## $tck
## [1] NA
##
## $tcl
## [1] -0.5
##
## $usr
## [1] 0 1 0 1
##
## $xaxp
## [1] 0 1 5
##
## $xaxs
## [1] "r"
##
## $xaxt
## [1] "s"
##
## $xpd
## [1] FALSE
##
## $yaxp
## [1] 0 1 5
##
## $yaxs
## [1] "r"
##
## $yaxt
## [1] "s"
##
## $ylbias
## [1] 0.2
```

4.

Use the **lattice** package to create a simple scatter plot of all the data for the `mpg` and `disp` variable of the `mtcars` dataset. Store the plot in a graphical objects called `plt1`. Compute the byte size of that graphical object.

```
library(lattice)
```

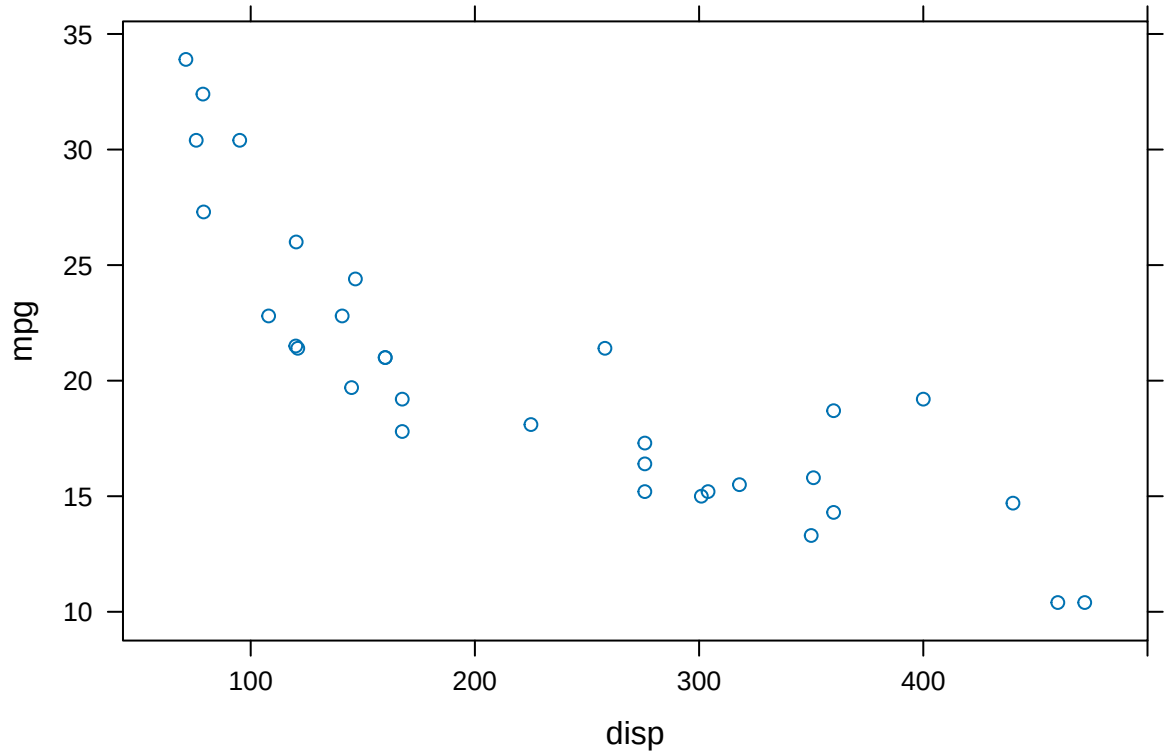
```
## Warning: package 'lattice' was built under R version 4.3.2
```

```
plt1 <- xyplot(
  mpg ~ disp,
  data = mtcars
)
```

```
object.size(plt1)
```

```
## 369784 bytes
```

```
plt1
```



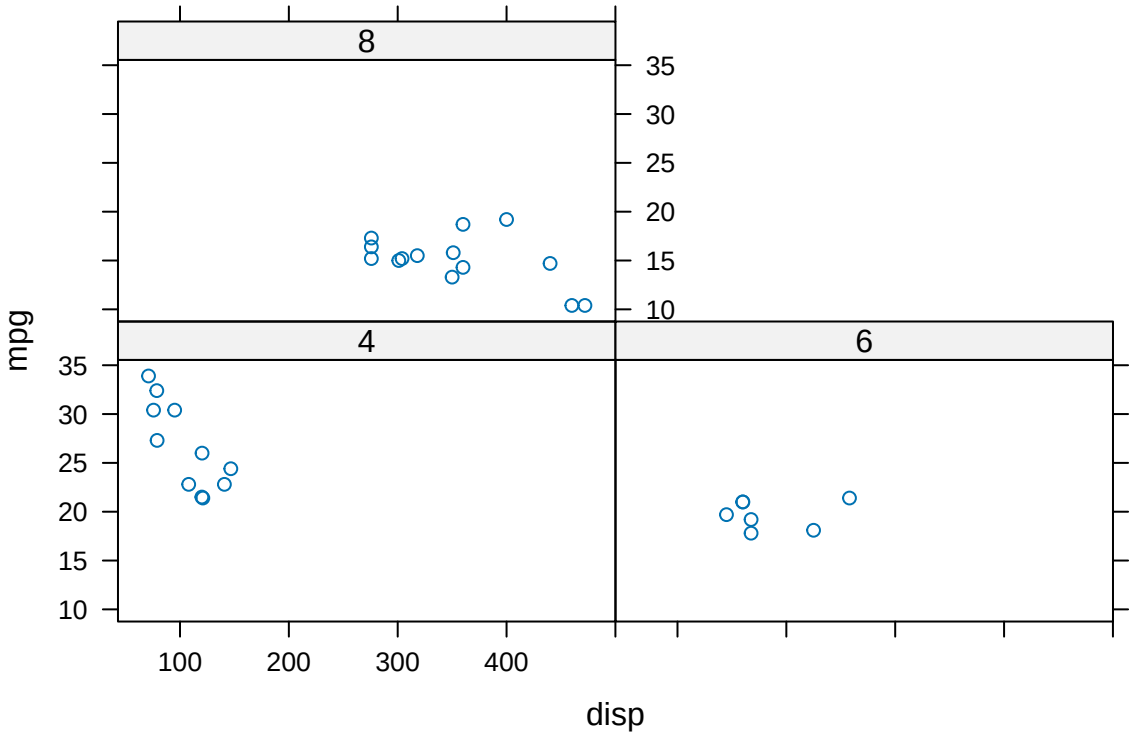
5.

Use the **lattice** package to create the same conditioning plot as in Exercise 1. Store it in a graphical object plt2 and compute the byte size of this object:

```
plt2 <- xyplot(
  mpg ~ disp | factor(cyl),
  data = mtcars,
  layout = c(2,2)
)
object.size(plt2)
```

```
## 373752 bytes
```

```
plt2
```



6.

Add a LOESS smoother to the previous **lattice** plot (in other words, create a similar plot as in Exercise 2). Store it in a graphical object `plt3` and compute the byte size of this object:

```
plt3 <- xyplot(
  mpg ~ disp | factor(cyl),
  data = mtcars,
  layout = c(2,2),
  panel = function(x, y, ...) {
    panel.xyplot(x, y, ...)
    panel.loess(x, y, ...)
  }
)
object.size(plt3)
```

```
## 386656 bytes
```

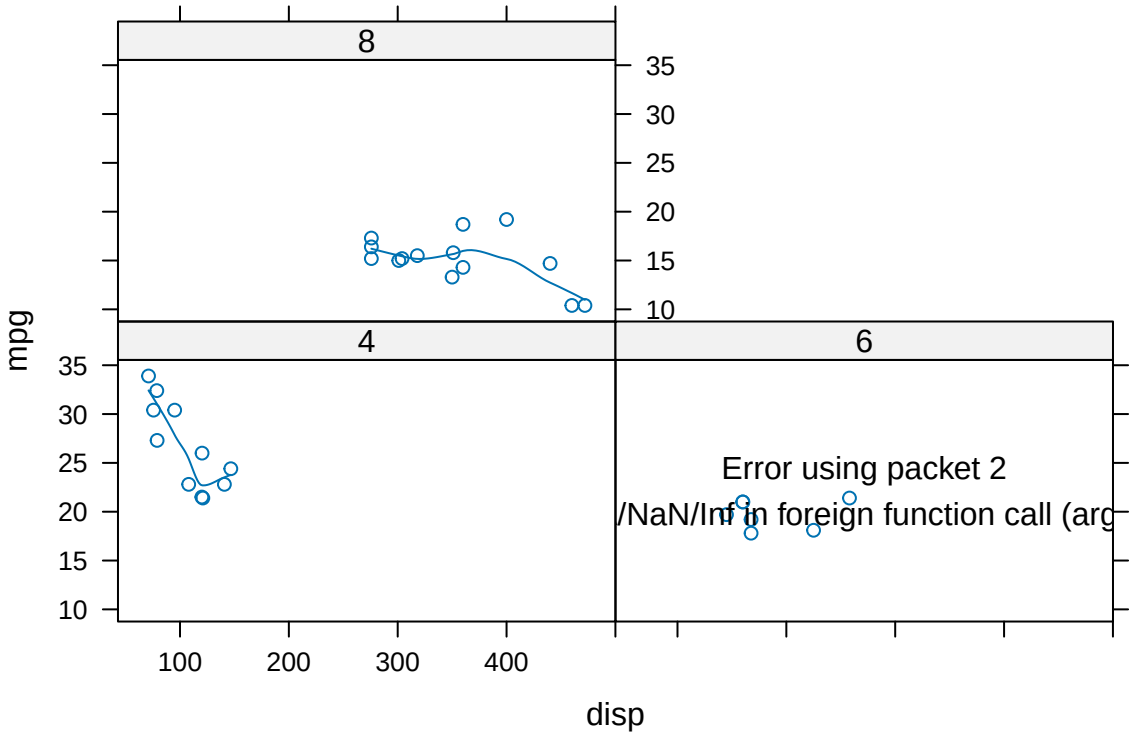
```
plt3
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## pseudoinverse used at 167.6
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## neighborhood radius 7.6
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## reciprocal condition number 0
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## There are other near singularities as well. 57.76
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## pseudoinverse used at 167.6
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## neighborhood radius 7.6
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## reciprocal condition number 0
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## There are other near singularities as well. 57.76
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## pseudoinverse used at 167.6
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## neighborhood radius 7.6
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## reciprocal condition number 0
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## There are other near singularities as well. 57.76
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## pseudoinverse used at 167.6
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## neighborhood radius 7.6
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## reciprocal condition number 0
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = FALSE, :
## There are other near singularities as well. 57.76
```



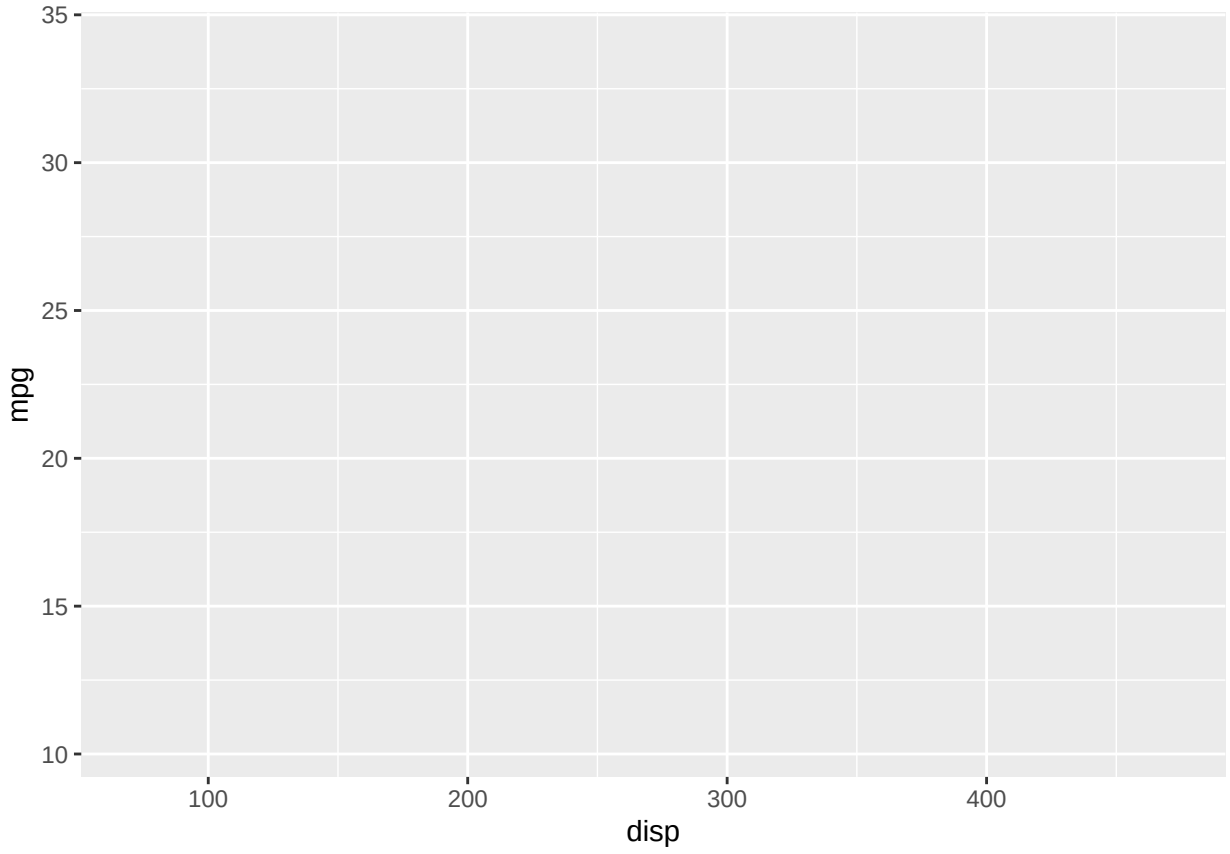
7.

Use the **ggplot2** package to create a graphical object `plt4` in which the `mpg` and `disp` variables of the `mtcars` dataset are mapped onto the `y` and `x` aesthetic, respectively. Compute the byte size of that graphical object.

```
library(ggplot2)
plt4 <- ggplot(
  data = mtcars,
  mapping = aes(
    x = disp,
    y = mpg
    )
  )
object.size(plt4)
```

```
## 12528 bytes
```

```
plt4
```



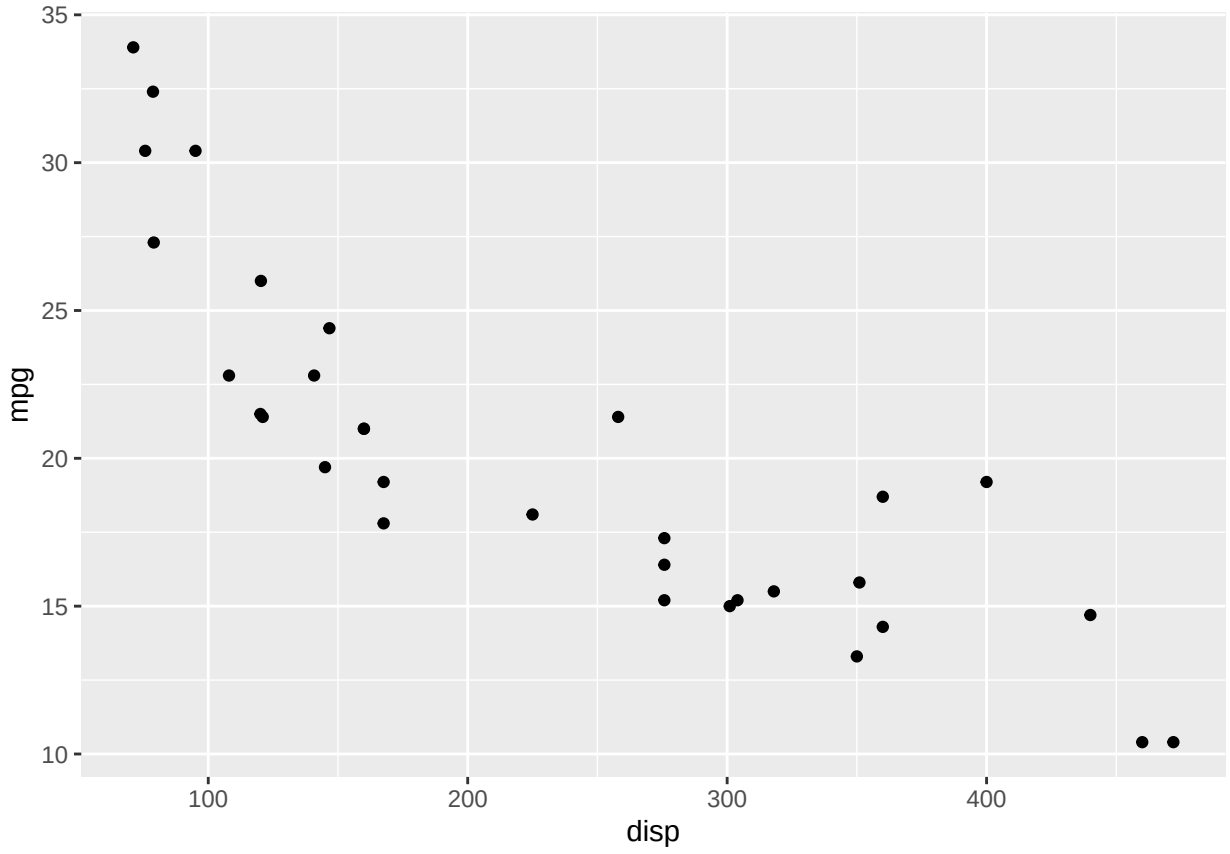
8.

Add a point geom to the graphical object, so you get a similar scatter plot as in Exercise 4. Store this in a graphical object `plt5` and compute its byte size.

```
plt5 <- plt4 +
  geom_point()
object.size(plt5)
```

```
## 13016 bytes
```

```
plt5
```



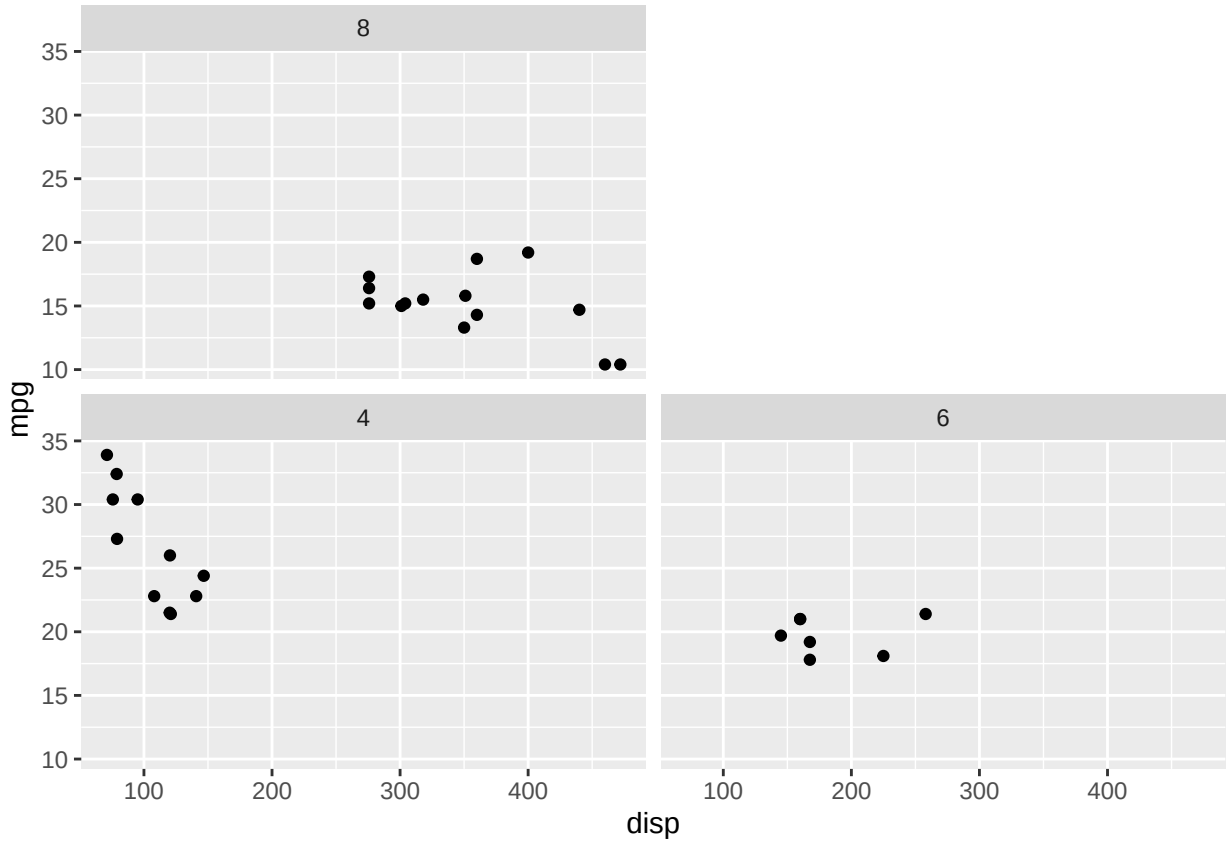
9.

Add a facet layer for the `cyl` variable to the previous graphical object. Store it in an object `plt6` and compute the byte size of this object.

```
plt6 <- plt5 +
  facet_wrap(
    facets = ~ factor(cyl),
    nrow = 2,
    as.table = FALSE
  )
object.size(plt6)
```

```
## 13016 bytes
```

```
plt6
```



10.

Add a layer with a LOESS smoother to the graphical object of the previous exercise (this may mean that the geom layer is automatically adjusted). Store this into a graphical object `plt7` and compute its byte size.

```
plt7 <- plt6 +
  geom_smooth(
    method = "loess",
    se = FALSE
  )
object.size(plt7)
```

```
## 13504 bytes
```

```
plt7
```

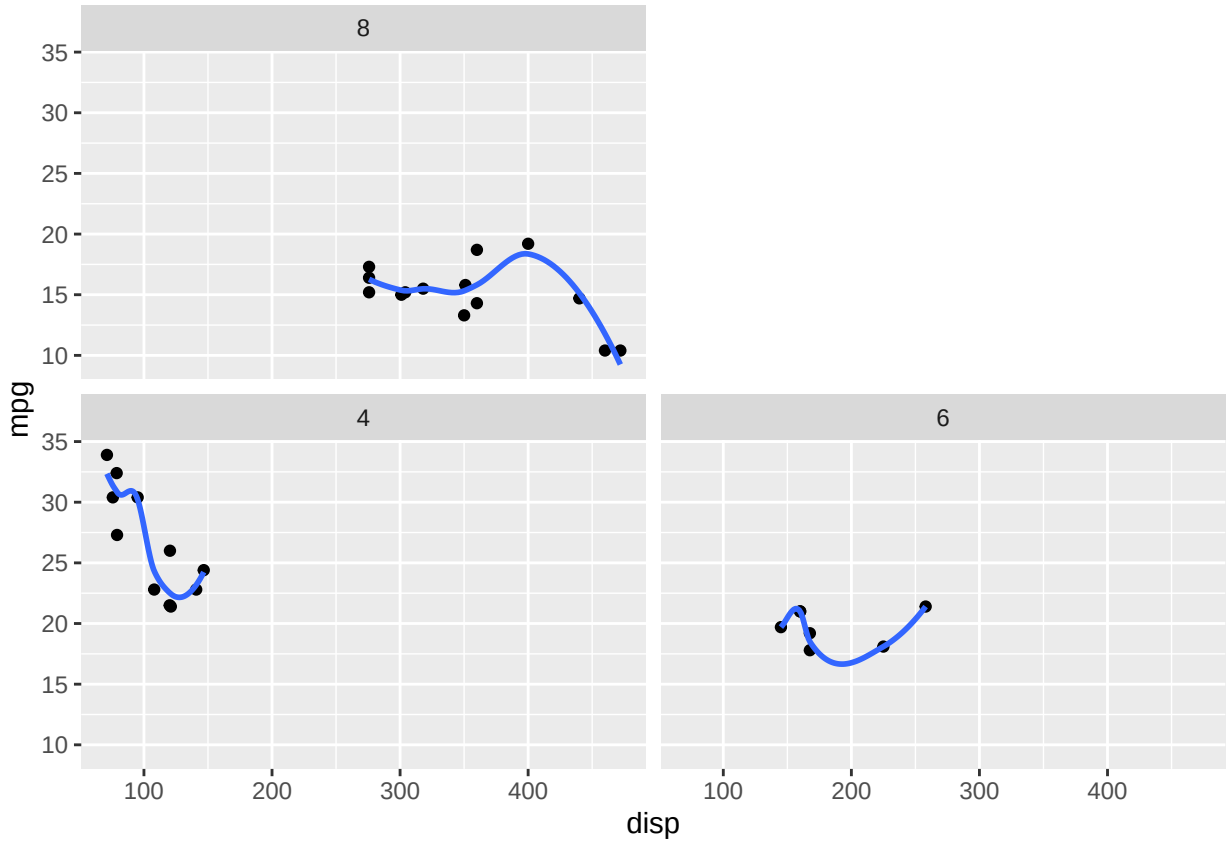
```
## `geom_smooth()` using formula = 'y ~ x'
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : pseudoinverse used at 144.44
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : neighborhood radius 23.165
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : reciprocal condition number 0
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric = parametric,
## : There are other near singularities as well. 510.76
```


8
4
6
mpg
disp
35
30
25
20
15
10
100
200
300
400