I'm delighted to announce that a new version of patchwork has been released on CRAN. This new version contains both a bunch of small bug fixes as well as some prominent features which will be showcased below.

If you are unaware of patchwork, it is a package that allows easy composition of graphics, primarily aimed at ggplot2, but with support for base graphics as well. You can read more about the package on its website.

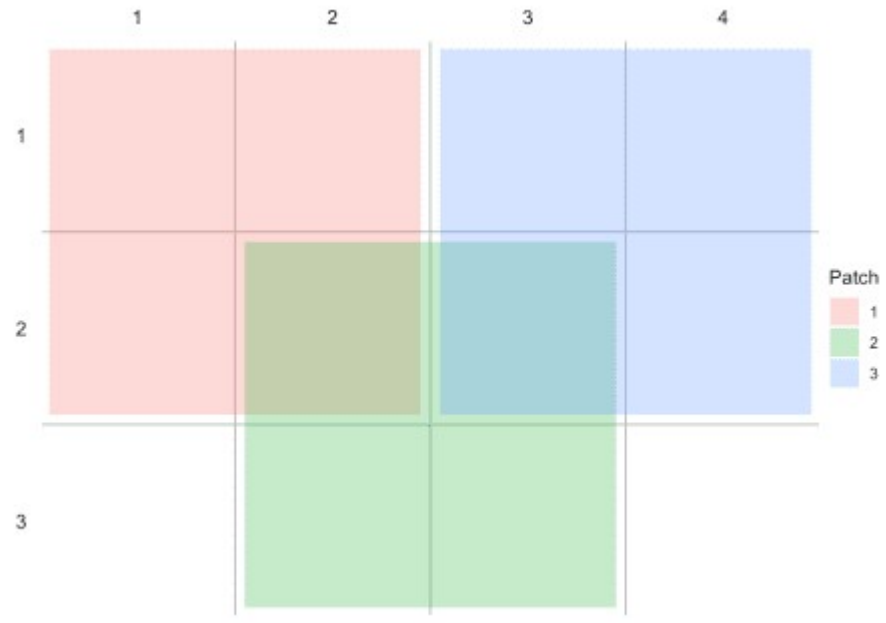
For the remainder of this post we'll use the following plots as examples:

```
library(ggplot2)
library(patchwork)
p1 <- ggplot(mtcars) +
  geom_point(aes(mpg, disp)) +
  ggtitle('Plot 1')

p2 <- ggplot(mtcars) +
  geom_boxplot(aes(gear, disp, group = gear)) +
  ggtitle('Plot 2')

p3 <- ggplot(mtcars) +
  geom_point(aes(hp, wt, colour = mpg)) +
  ggtitle('Plot 3')
```

Support for insets

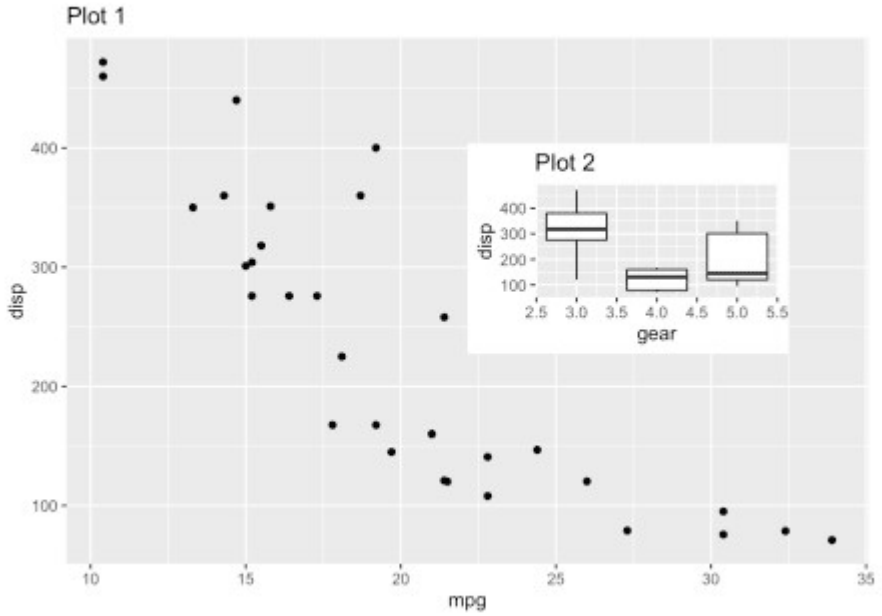
At it's inception patchwork was mainly designed to deal with alignment of plots displayed in a grid. This focus left out a small, but important for some, functionality for placing plots on top of each other. While it was possible to create a design with overlapping plots by combining different plotting areas:

```
design <- c(area(1, 1, 2, 2), area(2, 2, 3, 3), area(1, 3, 2, 4))
plot(design)
```



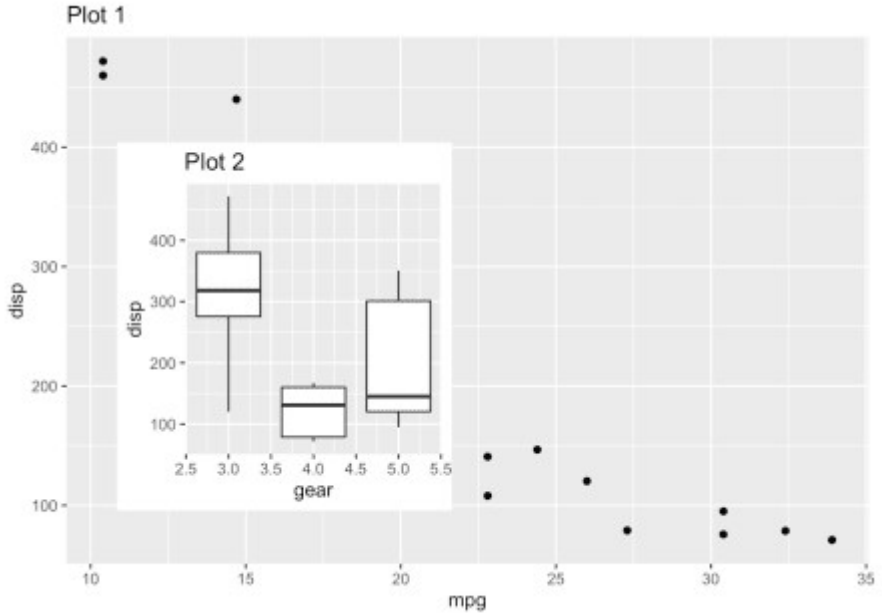
…this would still enforce an underlying grid, something that would come at odds with freely positioning insets. To make up for this patchwork has now gained an `inset_element()` function, which marks the given graphics as an inset to be added to the preceding plot. The function allows you to specify the exact location of the edges of the inset in any grid unit you want, thus giving you full freedom of the placement:

```
p1 + inset_element(p2, left = 0.5, bottom = 0.4, right = 0.9, top =
0.8)
```



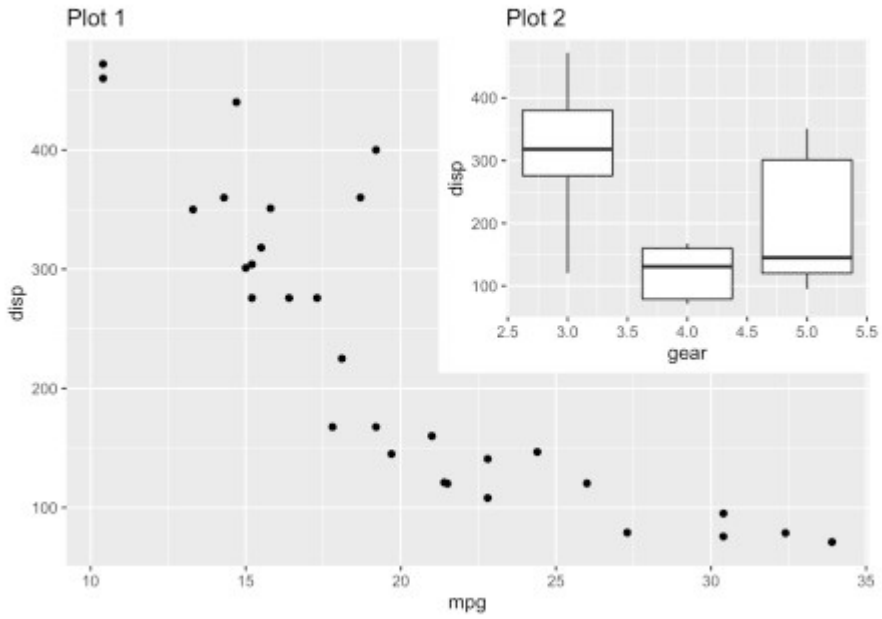
By default the positions use `npc` units which goes from 0 to 1 in the chosen area, other units can be used as well, by giving them explicitly:

```
p1 + inset_element(p2, left = unit(1, 'cm'), bottom = unit(30, 'pt'),
right = unit(3, 'in'),
                   top = 0.8)
```



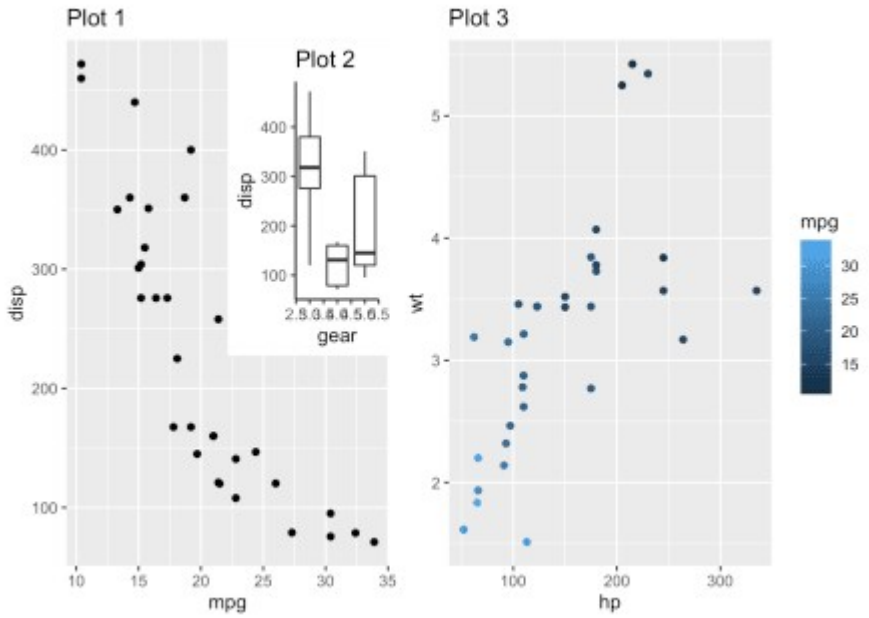
The default is to position the inset relative to the panel, but this can be changed with the `align_to` argument:

```
p1 + inset_element(p2, left = 0.5, bottom = 0.4, right = 1, top = 1,
align_to = 'full')
```

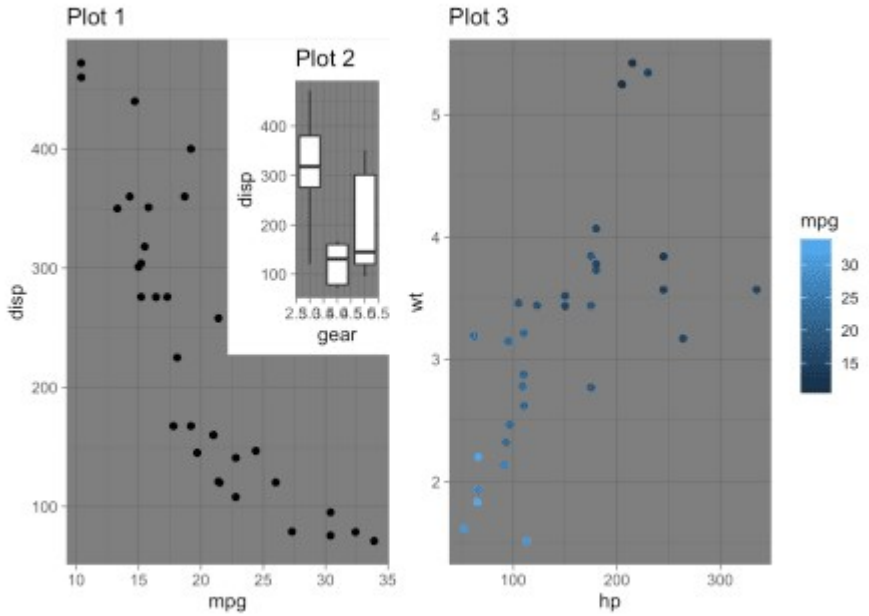


When it comes to all other functionality in patchwork, insets behaves as regular plots. This means that they are amenable to change after the composition:

```
p_all <- p1 + inset_element(p2, left = 0.5, bottom = 0.4, right = 1,
top = 1) + p3
p_all[[2]] <- p_all[[2]] + theme_classic()
p_all
```

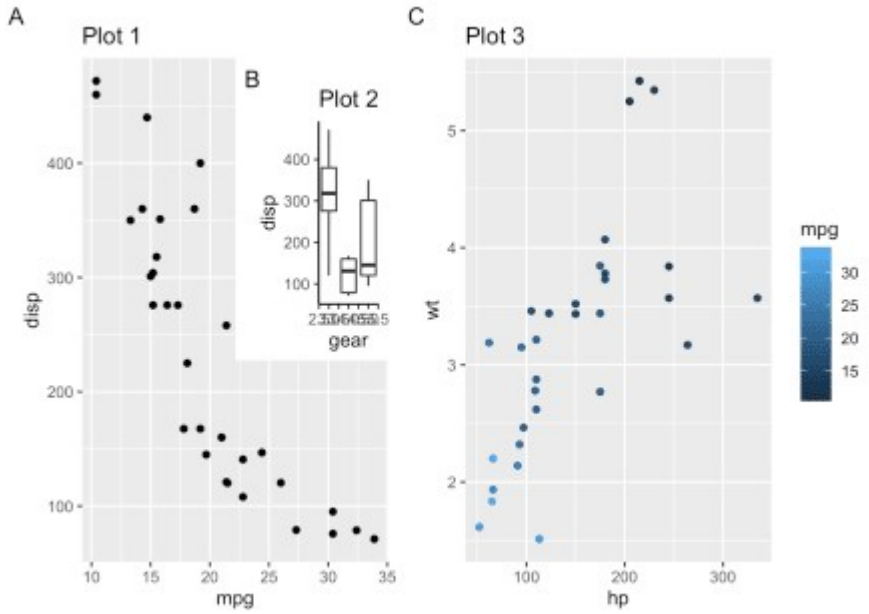



```
p_all & theme_dark()
```



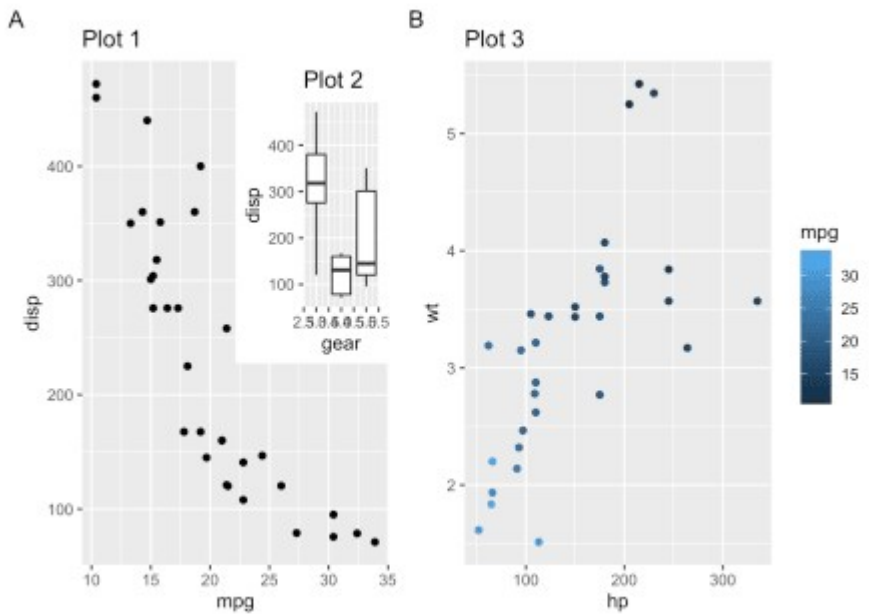
It can also get tagged automatically:

```
p_all + plot_annotation(tag_levels = 'A')
```



which can be turned off in the same manner as for `wrap_elements()`:

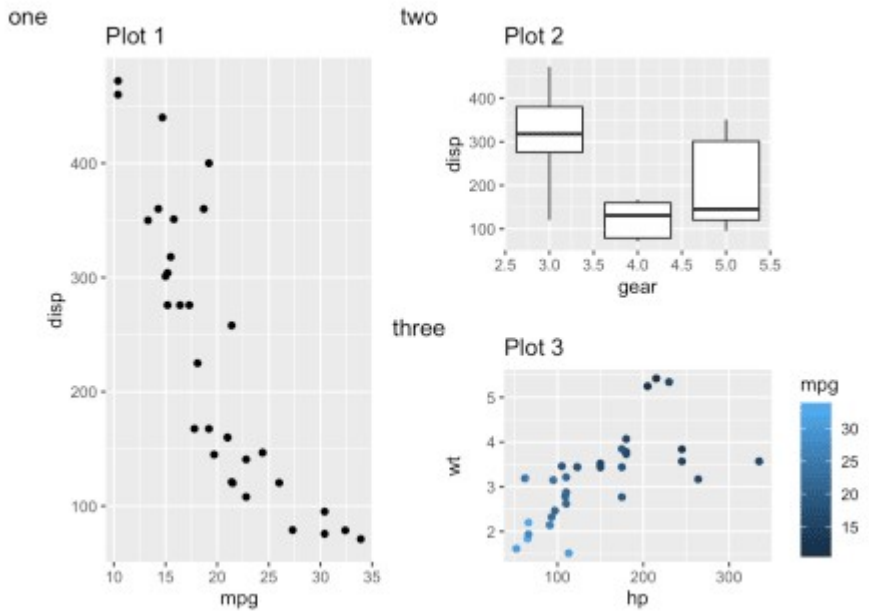
```
p_all <- p1 +
  inset_element(p2, left = 0.5, bottom = 0.4, right = 1, top = 1,
ignore_tag = TRUE) +
  p3
p_all + plot_annotation(tag_levels = 'A')
```



Arbitrary tagging sequences

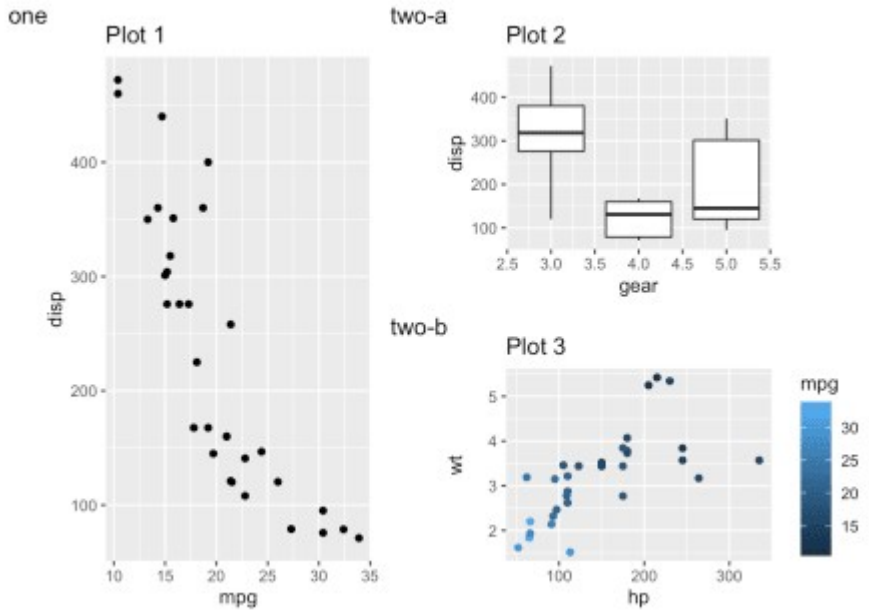
While we're discussing tagging, patchwork now allows you to provide your own sequence to use, instead of relying on the Latin character, Roman, or Arabic numerals that patchwork understands. This can be used by supplying a list of character vectors to the `tag_levels` argument instead of a single vector:

```
p_all <- p1 | (p2 / p3)
p_all + plot_annotation(tag_levels = list(c('one', 'two', 'three')))
```



When working with multiple tagging levels, custom sequences can be mixed with the automatic ones:

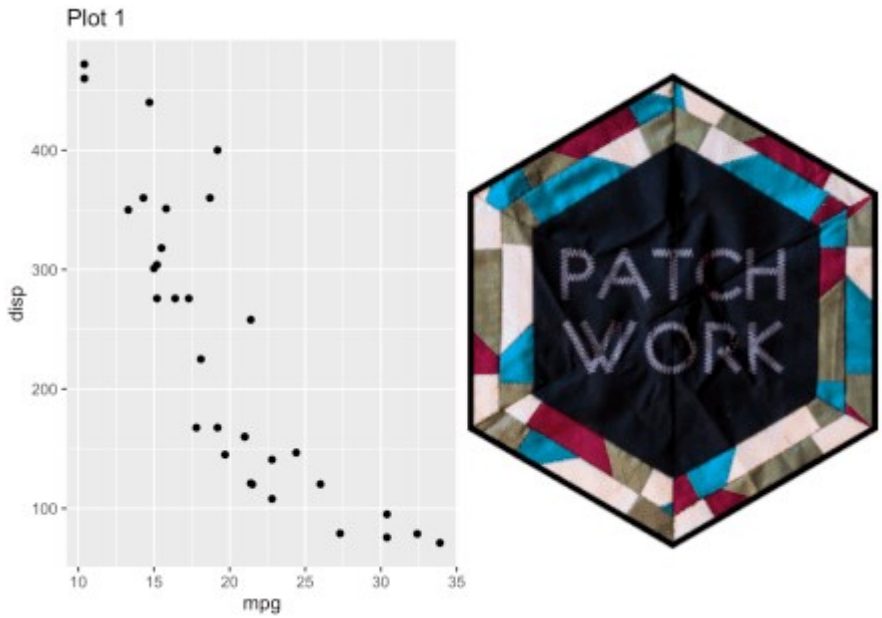
```
p_all[[2]] <- p_all[[2]] + plot_layout(tag_level = 'new')
p_all + plot_annotation(tag_levels = list(c('one', 'two', 'three'),
'a'), tag_sep = '-')
```



Raster support

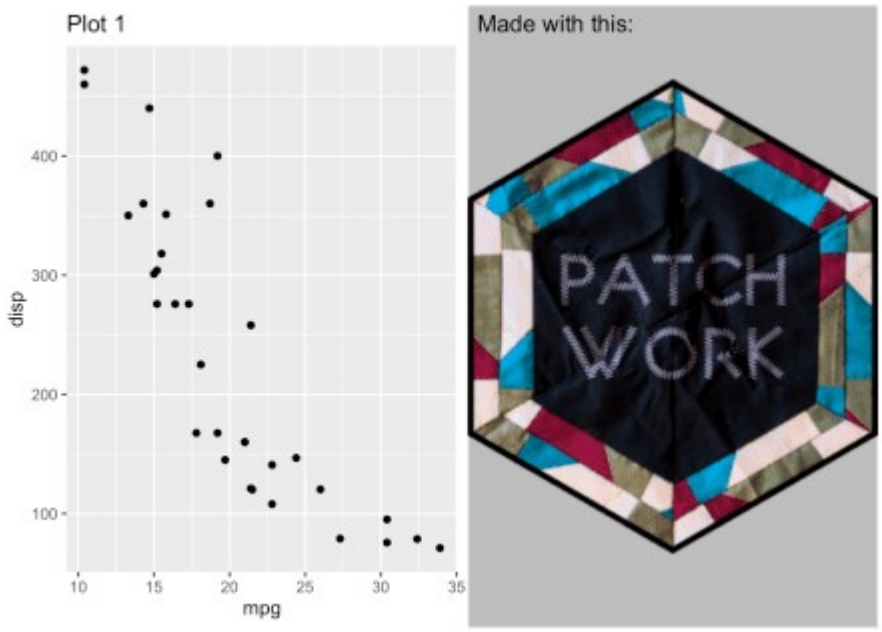
While patchwork was designed with ggplot2 in mind it has always supported additional graphic types such as grobs and base graphics (by using formula notation). This release adds support for an additional type: raster. The raster class (and nativeRaster class) are bitmap representation of images and they are now recognized directly and with the `wrap_elements()` function:

```
logo <- system.file('help', 'figures', 'logo.png', package =
'patchwork')
logo <- png::readPNG(logo, native = TRUE)

p1 + logo
```



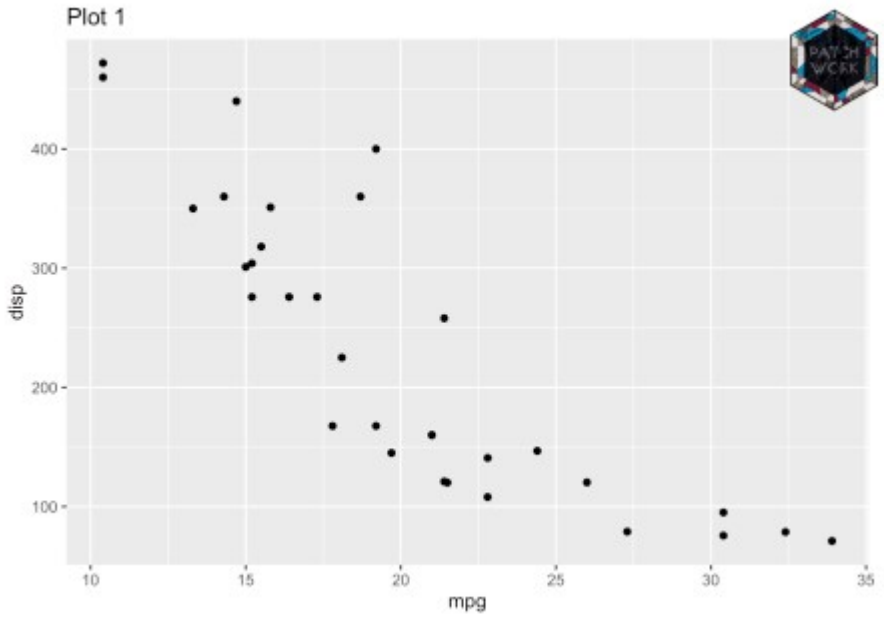
Since they are implemented as wrapped elements they can still be titled etc:

```
p1 + logo + ggtitle('Made with this:') + theme(plot.background =
element_rect('grey'))
```



They can of course also be used with the new inset feature to easily add watermarks etc.

```
p1 + inset_element(logo, 0.9, 0.8, 1, 1, align_to = 'full') +
theme_void()
```



The future

That's it for this release. …