Multiple plots in one page

In this post I will show you how to arrange multiple plots in single one page with:

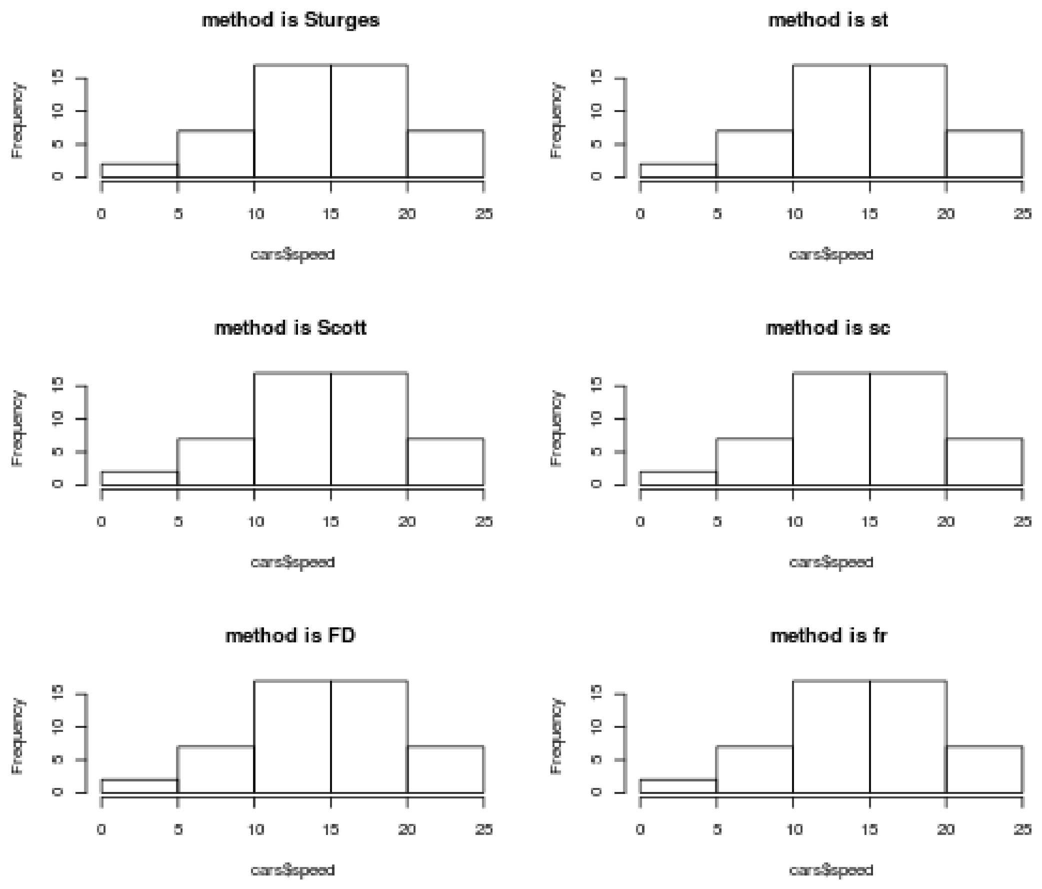
- Classic R command
- ggplot

Classic R command

Ploting multiple graphs in single one page (or canvas) with classic R command is straightforward and easy.

The key lies in `par`.

```
par(mfrow = c(3, 2))  # 3 rows and 2 columns
for (i in c("Sturges", "st", "Scott", "sc", "FD", "fr")) {
    hist(cars$speed, breaks = i, main = paste("method is",
i, split = ""))
}
```

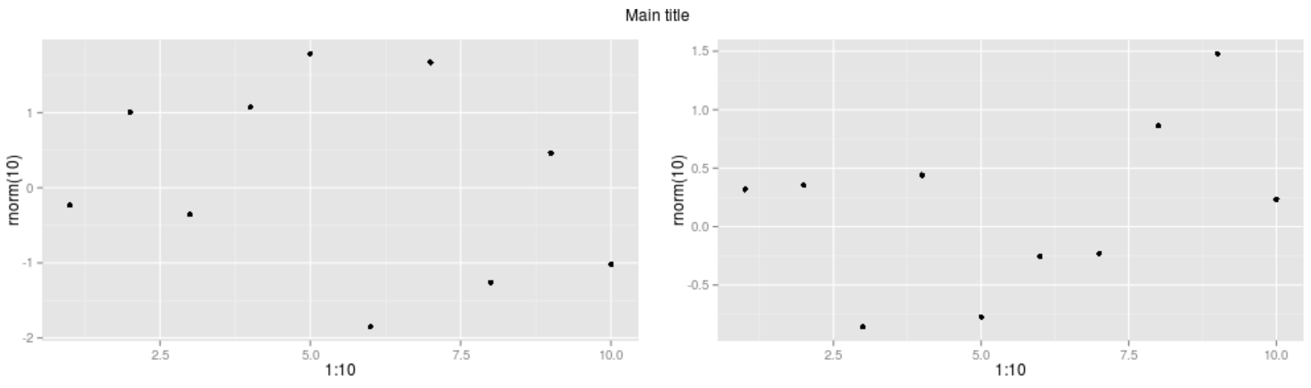


ggplot2

gridExtra

We all know `ggplot2` is elegant. And It cannot easily plotted in sigle one page just with defined `par`. That is why we need `gridExtra`. Just like the official wiki introduces, `grid_arrage` will make it work.

```
library(ggplot2)
library(grid)
library(gridExtra)
p1 = qplot(1:10, rnorm(10))
p2 = qplot(1:10, rnorm(10))
grid.arrange(p1, p2, ncol = 2, main = "Main title")
```



gridExtra-like custom funcions

From the Cookbook for R/), `multiplot` function is costumer designed function. You can copy the codes, and it can work like `gridExtra`.

```
# muliplot(p1,p2,p3,...,cols=5)
```

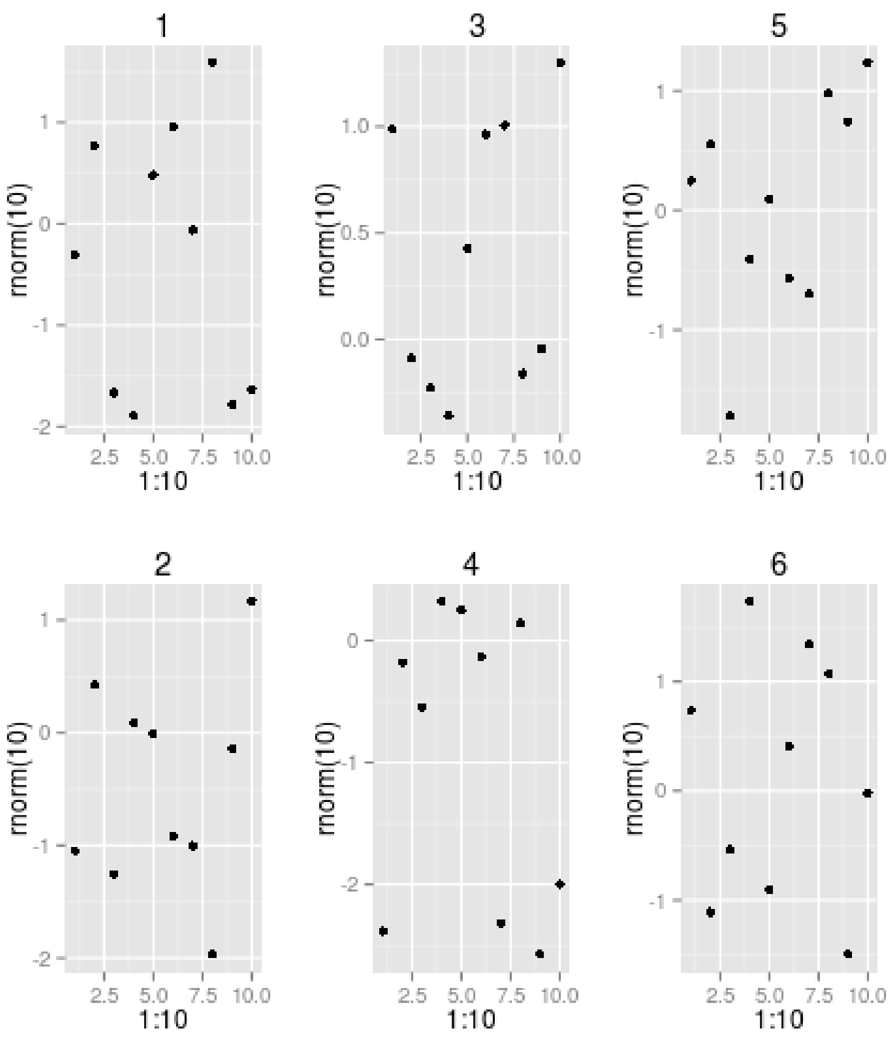
par-like custom functions

List all plots

But here is a disadvantage. What if all the plots have tiny differences between each of them? The classic R command `for` loop can make it. It is a huge waste of time to establish every plot named in p1, p2, p3, etc, let alone input them one by one.

Still use `multiplot`. But here we use `list` function. Inside the `multiplot` function, there is a flag, named: **plotlist**.

```
plots <- list()  # new empty list
for (i in 1:6) {
    p1 = qplot(1:10, rnorm(10), main = i)
    plots[[i]] <- p1  # add each plot into plot list
}
multiplot(plotlist = plots, cols = 3)
```

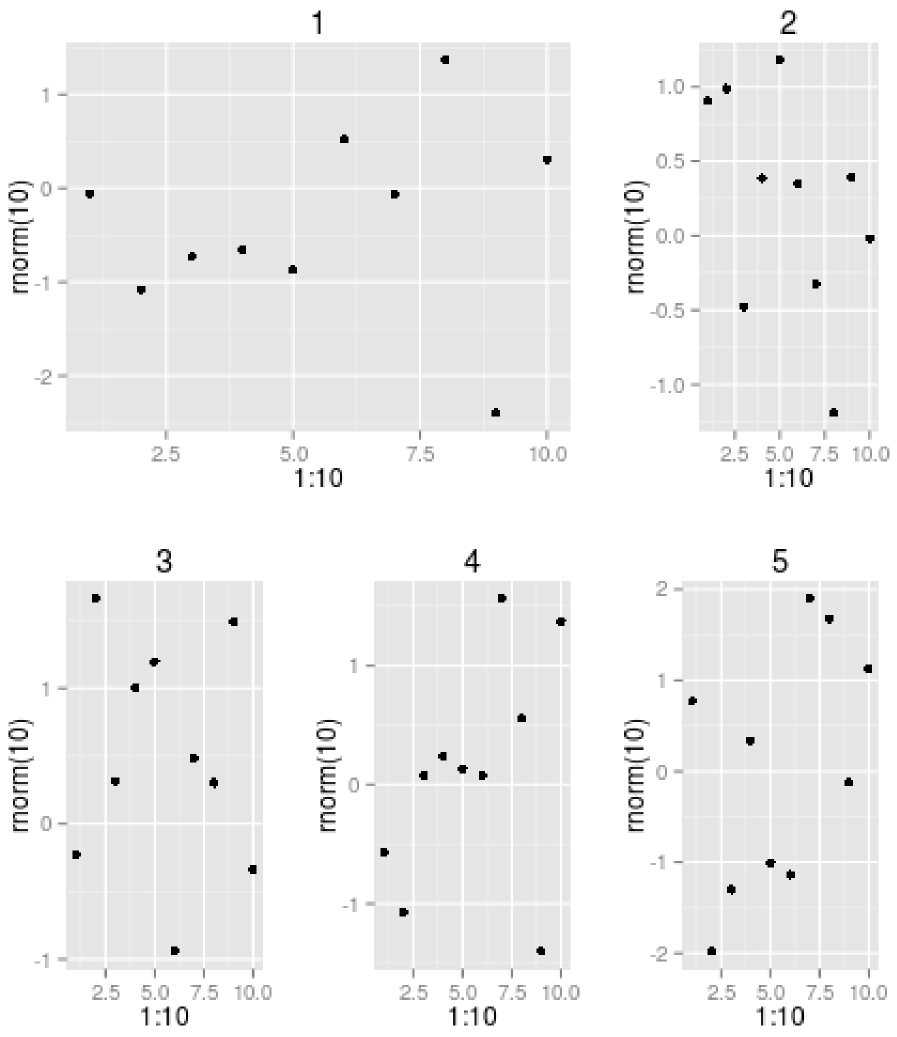


Sensational!!!

Layouts

What if you wish to define territorries for each plot? You should define `Layout` flag.

```
library(ggplot2)
for (i in 1:5) {
    p1 = qplot(1:10, rnorm(10), main = i)
    plots[[i]] <- p1
}
layout <- matrix(c(1, 1, 2, 3, 4, 5), nrow = 2, byrow =
TRUE)
multiplot(plotlist = plots, layout = layout)
```

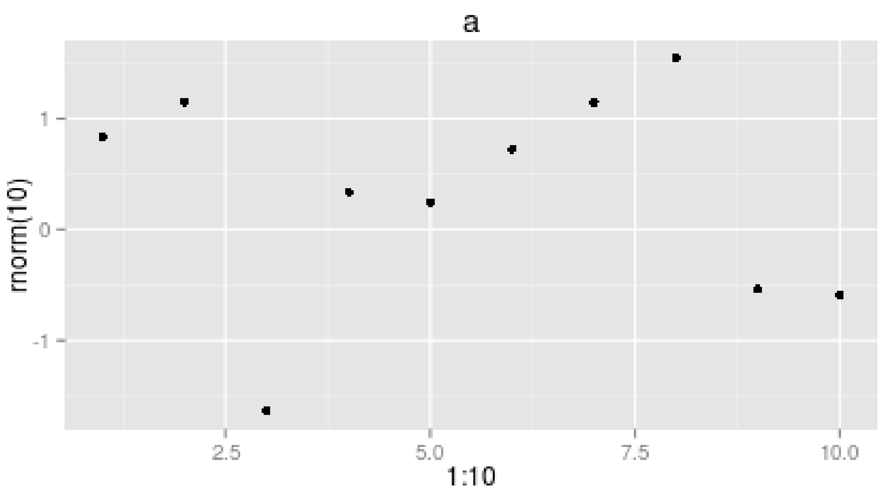


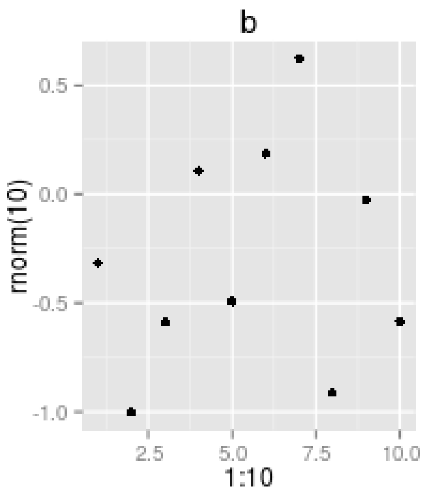
```
FALSE Error: Invalid 'layout.pos.row' in viewport
```

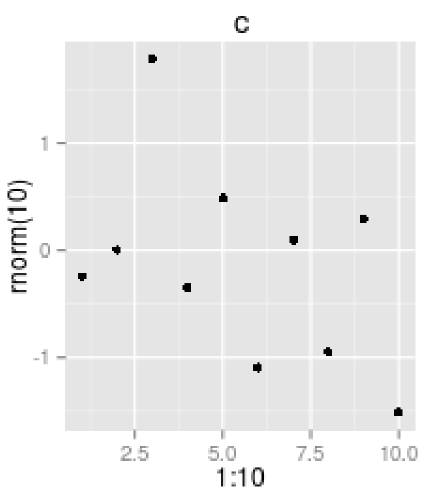
How does it work?

The key is to define the area. The following code will show you how:

```
a <- qplot(1:10, rnorm(10), main = "a")
b <- qplot(1:10, rnorm(10), main = "b")
c <- qplot(1:10, rnorm(10), main = "c")
grid.newpage()
pushViewport(viewport(layout = grid.layout(2, 2)))
vplayout <- function(x, y) viewport(layout.pos.row = x,
layout.pos.col = y)
print(a, vp = vplayout(1, 1:2))  # key is to define
vplayout
print(b, vp = vplayout(2, 1))
print(c, vp = vplayout(2, 2))
```


a
rnorm(10)
1
0
-1
2.5
5.0
7.5
10.0
1:10


b
rnorm(10)
0.5
0.0
-0.5
-1.0
2.5
5.0
7.5
10.0
1:10


c
rnorm(10)
1
0
-1
2.5
5.0
7.5
10.0
1:10