

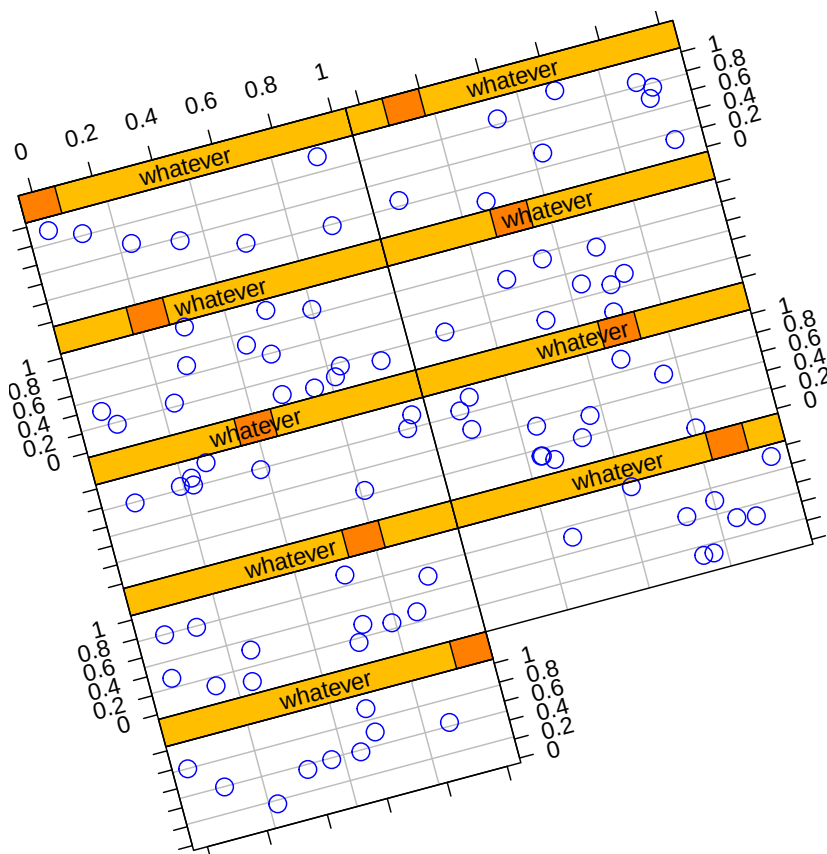
Rotated Viewports

Paul Murrell

October 14, 2026

It is possible to specify an angle of rotation for a Grid viewport. For example, the following code draws the example multipanel plot at an angle of 15° .

```
> pushViewport(viewport(h = .8, w = .8, angle = 15))  
> grid.multipanel(newpage = FALSE)  
> popViewport()
```



A more complicated example is now developed. First of all we generate some data to plot; an x and a y with an obvious correlation.

```
> x <- rnorm(50)
> y <- x + rnorm(50, 1, 2)
```

Next we generate some statistics from the data.

```
> # We will extend the axes over the entire region so
> # extrapolate scale from main data region
> scale <- extendrange(r = range(x,y))
> extscale <- c(min(scale), max(scale)+diff(scale)*1/3)
```

Now generate a layout of regions: a 3" by 3" region for a scatterplot, inside a 4" by 4" region.

```
> lay <- grid.layout(2, 2,
+                   widths = unit(c(3, 1), "inches"),
+                   heights = unit(c(1, 3), "inches"))
> vp1 <- viewport(width = unit(4, "inches"), height = unit(4, "inches"),
+                 layout = lay, xscale = extscale, yscale = extscale)
```

We draw a box around the outside and axes on the entire region.

```
> grid.newpage()
> pushViewport(vp1)
> grid.rect()
> grid.xaxis()
> grid.text("Test", y = unit(-3, "lines"))
> grid.yaxis()
> grid.text("Retest", x = unit(-3, "lines"), rot = 90)
```

We draw points within the interior region.

```
> vp2 <- viewport(layout.pos.row = 2, layout.pos.col = 1,
+                 xscale = scale, yscale = scale)
> pushViewport(vp2)
> grid.lines()
> grid.points(x, y, gp = gpar(col = "blue"))
> popViewport()
```

Now we use a rotated viewport to draw a boxplot which indicates the distribution of the distances between the points in the scatterplot and the line $y = x$ ¹.

The final output is shown on the last page.

¹This may look like a large amount of code, but that's mostly because its doing a boxplot by hand rather than using a predefined high-level function.

```

> diffs <- (y - x)
> rdiffs <- range(diffs)
> ddiffs <- diff(rdiffs)
> bxp <- boxplot(diffs, plot = FALSE)
> vp3 <- viewport(x = unit(3, "inches"),
+               y = unit(3, "inches"),
+               width = unit(.5, "inches"),
+               # NOTE that the axis on the boxplot represents
+               # actual (y - x) values BUT to make
+               # the bits of the boxplot line
+               # up with the data points we have to plot
+               # (y - x)/sqrt(2)
+               # Hence the sin(pi/4) below
+               height = unit(ddiffs*sin(pi/4)/diff(scale)*3, "inches"),
+               just = c("centre", "center"),
+               angle = 45,
+               gp = gpar(col = "red"),
+               yscale = c(-ddiffs/2, ddiffs/2))
> pushViewport(vp3)
> left <- -.3
> width <- .8
> middle <- left + width/2
> grid.rect(x = left, y = unit(bxp$conf[1,1], "native"),
+          width = width, height = unit(diff(bxp$conf[,1]), "native"),
+          just = c("left", "bottom"),
+          gp = gpar(col = NULL, fill = "orange"))
> grid.rect(x = left, y = unit(bxp$stats[4,1], "native"),
+          width = width, height = unit(diff(bxp$stats[4:3,1]), "native"),
+          just = c("left", "bottom"))
> grid.rect(x = left, y = unit(bxp$stats[3,1], "native"),
+          width = width, height = unit(diff(bxp$stats[3:2,1]), "native"),
+          just = c("left", "bottom"))
> grid.lines(x = c(middle, middle), y = unit(bxp$stats[1:2,1], "native"))
> grid.lines(x = c(middle, middle), y = unit(bxp$stats[4:5,1], "native"))
> grid.lines(x = c(middle-.1, middle+.1), y = unit(bxp$stats[1,1], "native"))
> grid.lines(x = c(middle-.1, middle+.1), y = unit(bxp$stats[5,1], "native"))
> np <- length(bxp$out)
> if (np > 0)
+   grid.points(x = rep(middle, np), y = unit(bxp$out, "native"))
> grid.yaxis(main = FALSE)
> popViewport(2)
>

```

