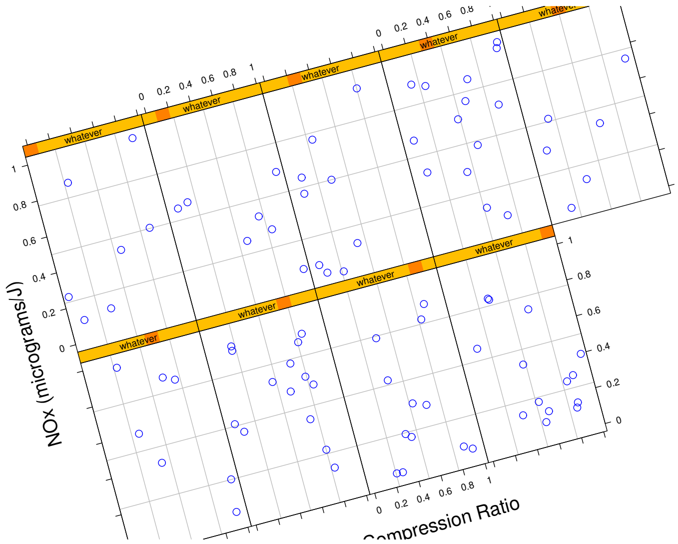
Rotated Viewports

August 29, 2001

⟨*⟩≡

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◦.

⟨*rotmulti*⟩≡
```
push.viewport(viewport(h=.8, w=.8, angle=15))
grid.multipanel(newpage=F)
pop.viewport()
```



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

⟨*ttest*⟩≡
```
x <- rnorm(50)
y <- x + rnorm(50, 1, 2)
```

Next we generate some statistics from the data.

⟨*ttest*⟩+≡

```
rx <- range(x)
dx <- diff(rx)
ry <- range(y)
dy <- diff(ry)
max <- max(rx, ry)
min <- min(rx, ry)
r <- c(min(rx, ry), max(rx, ry))
d <- diff(r)
# We will extend the axes over the entire region so
# extrapolate scale from main data region
scale <- r + c(-1, 1)*d*.05
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.

⟨*ttest*⟩+≡

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

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

⟨*ttest*⟩+≡

```
push.viewport(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.

⟨*ttest*⟩+≡

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

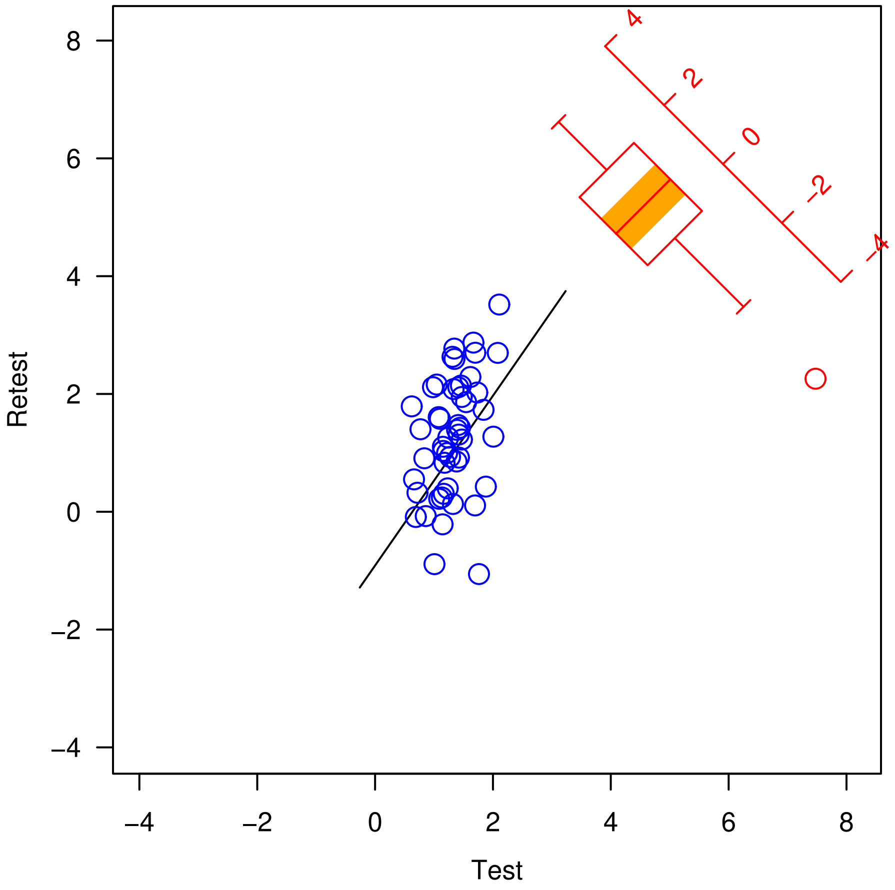
Now we use a rotated viewport to draw a boxplot which indicates the distribution of the distances between the points in the scatterplot and the line $y = x$[1].

The final output is shown on the last page.

$\langle ttest \rangle{+}{\equiv}$

```
diffs <- (y - x)
rdiffs <- range(diffs)
ddiffs <- diff(rdiffs)
bxp <- boxplot(diffs, plot=F)
vp3 <- viewport(x=unit(3, "inches"),
                y=unit(3, "inches"),
                w=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
                h=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))
push.viewport(vp3)
left <- -.3
width <- .8
middle <- left + width/2
grid.rect(x=left, y=unit(bxp$conf[1,1], "native"),
          w=width, h=unit(diff(bxp$conf[,1]), "native"),
          just=c("left", "bottom"),
          gp=gpar(col=NULL, fill="orange"))
grid.rect(x=left, y=unit(bxp$stats[4,1], "native"),
          w=width, h=unit(diff(bxp$stats[4:3,1]), "native"),
          just=c("left", "bottom"))
grid.rect(x=left, y=unit(bxp$stats[3,1], "native"),
          w=width, h=unit(diff(bxp$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=F)
pop.viewport(2)
```

[1] This 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.



Problems

1. Data symbols will not be affected by the angle of rotation. For round data symbols this does not matter, but it will make just about everything else look pretty odd.