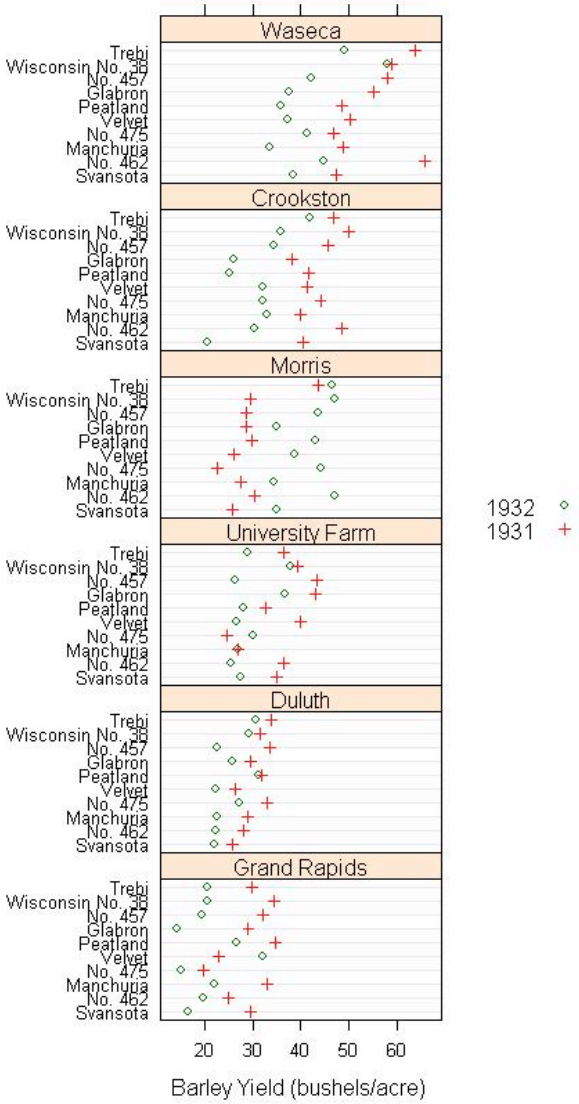
Trellis & Lattice

유 충현
블로그 모음 4탄(http://blog.naver.com/bdboys) • (주)오픈베이스 • 2010년 10월 3일



Trellis & Lattice

Trellis Graphics는 1993년 벨 연구소의 Bill Cleveland이 쓴 "Visualizing Data"라는 책에서 아이디어가 나타났고 이후 Rick Becker, Bill Cleveland 등이 개발한 데이터 시각화를 위한 Framework이다. 그리고 아이디어가 도출된 이후 1993-1996 동안 개발과 평가를 거쳐 마침내 S Language 기반으로 탄생하게 되었다.

Trellis Graphics는 panel이라는 여러 개의 독립된 Graph Chart를 가로, 세로 혹은 페이지의 배열로 나타내어서 비교 분석할 수 있다. 즉, 다변량 데이터에서의 변수들 간의 유기적인 관계나 특징을 파악할 수 있는 도구라 할 수 있다. 이들이 Trellis라고 명명한 것은 Trellis를 구현하는 기술이 정원의 격자짜기(garden trelliswork)를 추억케 하였기 때문이라 한다.

다음은 Trellis가 무엇이냐는 질문에 개발자들이 답한 내용이다.

"Trellis는 2차원과 3차원의 데이터를 시각적으로 표현하는 Framework이다. 이것은 단지 하나의 panel이나 혹은 많은 panel을 구성할 수 있다. 많은 panel이 있을 경우 거기에는 구조가 있다.: 행, 열, 페이지들의 세 가지 방법으로 나열된 panel의 직사각형의 배열 구조로 이것은 정원의 격자짜기(garden trelliswork)를 추억케 한다. (덩굴식물과 다른 식물들의 성장을 버팀목하는, 나무조각으로 교차해서 만는든 것)"

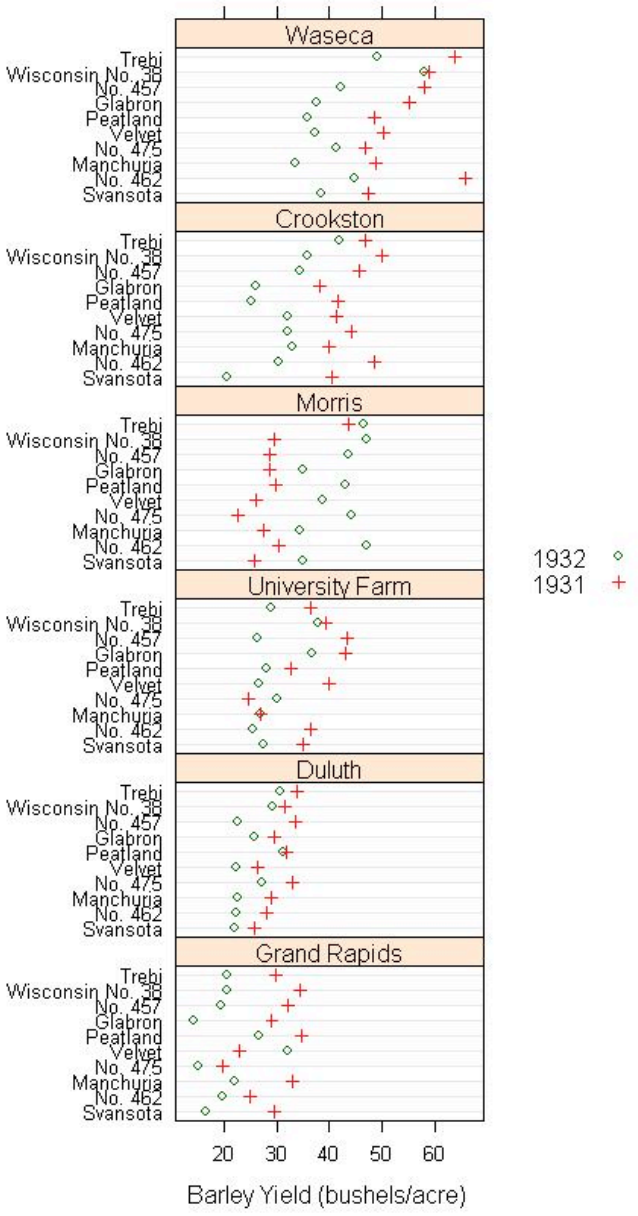
Trellis가 과연 어떻게 생겼는지 다음 그림을 보자.

Lattice는 Trellis Graphics를 R에 가장 잘 구현한 패키지로 인정받고 있다. 사전적으로 Trellis와 Lattice는 같다. 굳이 R에서 Trellis를 Lattice로 부르는 것은 S(S-PLUS) 바로 전의 알파벳이 R인점인 것과 유사하다. S 이전의 R처럼 명명자의 재치가 느껴지는 부분이기도 하다.

이 패키지의 interface는 S-PLUS의 것을 기본으로 하고 있지만 몇 가지의 차이점이 존재한다. 예를 들면 piechart를 제외하고는 S-PLUS에 구현된 High level의 Trellis function을 거의 사용할 수 있다.

그러면 Trellis의 진가를 한번 살펴보자.

```
〉 library(lattice)
〉 dotplot(variety ~ yield | site, data = barley, groups = year,
        key = simpleKey(levels(barley$year), space = "right"),
        xlab = "Barley Yield (bushels/acre) ",
        aspect=0.5, layout = c(1,6), ylab=NULL)
```



위 그림은 Trellis Chart로서 농업실험장에서 보리의 수확량을 테스트 하기위해 시도된 자료들을 도식화해 준다.

이 자료는 미국 미네소타주의 6개 농업실험장에서 10개의 품종에 대해서 두 해 간의 수확량의 수율의 증가를 보여준다. 또한 이 자료는 실험장, 품종, 년도의 모든 조합의 경우의 수 6*10*2 = 120개의 측정치로 이루어졌다.

그림에 의하면 20개의 개별 실험장에서의 수율이 각각의 Panel에 출력되어 있다.

참고로 수율은 단위 면적 당 수확된 보리의 양을 나타낸다.
- 수율 = bushels/acre
- bushel(부셸) : 건량 단위; 약 35리터
- acre(에이커) : 면적의 단위; 약 4046.8m2, 약 1,224평)

이 보리 경작 실험은 1930년대에 수행되었으며 Data는 실험자가 보고서로 출판한 1934년부터 처음으로 사용되었으며 그 후로 여러 사람들에 의해 분석되고 또 재분석되었다. R. A. Fisher는 실험계획(The Design of Experiments)이라는 책에 이 자료를 게재하였다. 이 출판물로 인해서 보리수확 실험의 자료는 유명해졌고, 그 후 많은 다른 사람들에 의해 분석되어졌으며 새로운 통계분석 방법의 예제 자료로 사용되었다. 그리고 1990년대 초, 이 자료가 Trellis Graphics의해 시각화되었다.

이 사건은 매우 놀랄만한 결과를 가져왔다. 60년이 넘는 시간과 많은 사람들의 자료분석을 통해서도 발견하지 못한 것을 발견한 것이다. 그림을 보면 Morris의 실험장에서 그것을 발견할 수 있다. 다른 모든 실험장에서는 1931년의 수율이 1932년 수율보다 전체적으로 현저하게 높게 나타나 있다. 그러나 Morris의 실험장에서는 정반대다.

Morris 실험장에서의 1932년의 수율이 1931년의 것을 초과하는 것과 비슷하게 다른 실험장에서는 1931년의 수율이 1932년의 것을 초과했다. 이 원인은 의도하지 않게 두 년도가 뒤바뀌었던가, 질병이나 이상기후 등의 예외사항이 동시에 발생하는 색다른 자연현상들 중에 하나일 것이다.

더 많은 Trellis displays이나 데이터분석의 모델링, 실험의 몇몇 배경을 점검하는 것들은 데이터에 오류가 있는것을 판단하게 도와준다. 특히 앞의 그림처럼 Trellis displays는 자료의 특성을 쉽게 간파할 수 있게 도와주며, 이것이 바로 Trellis displays의 장점이다.

어쩌면 60여년 동안 많은 사람들이 신중치 못하였다고 생각할 수도 있다. 아니면 과거에는 컴퓨터와 같은 도구를 사용할 수 없었고, 컴퓨터의 기술 발전에 따른 통계분석 도구의 성능 개선에 따른 쾌거일 수도 있다. 하지만 분명한 것은 단순 자료보다 도식화된 Chart가 자료의 특성을 쉽게 설명하고, 다변량 자료에 있어서는 변수들의 조합의 수의 Chart를 한 눈에 볼 수 있게 도식화하는 것이 한단계 더 진일보한 방법론일 것이다.

xyplot과 같은 High level Lattice 함수는 전통적인 S graphics 함수와 다르다. 그러므로 혼동되지 않게 주의를 기울여야만 한다. 예를 들면 par() 함수를 이용한 Setting의 변경은 lattice plots에서는 불가능하다. 그러나 다른 방법으로 Setting의 변경을 통해 고도의 커스트마이징이 가능하다.

간단하게, Lattice에 대해서 알아 보자. 우선 Trellis Plot을 그리기 위해서는 lattice 라이브러리를 가져와야 한다.

```
〉library(lattice)
```

그러면 High level Lattice 함수를 사용할 수 있다. 이 High level Lattice 함수들은 graphic device가 아닌 lattice device위에 그림을 그리게 되는데, 배경색이 검은색으로 나타날 것이다. 조금은 보기가 싫을지도 모른다. 이것이 Lattice의 기본 색상이다.

다른 고전적인 S Graphics의 함수 결과처럼 보기 위해서는 다음과 같은 두 가지 방법을 사용할 수 있다.

1. trellis.device를 이용하는 방법

```
trellis.device(color=TRUE, theme = "col.whitebg") ;
```

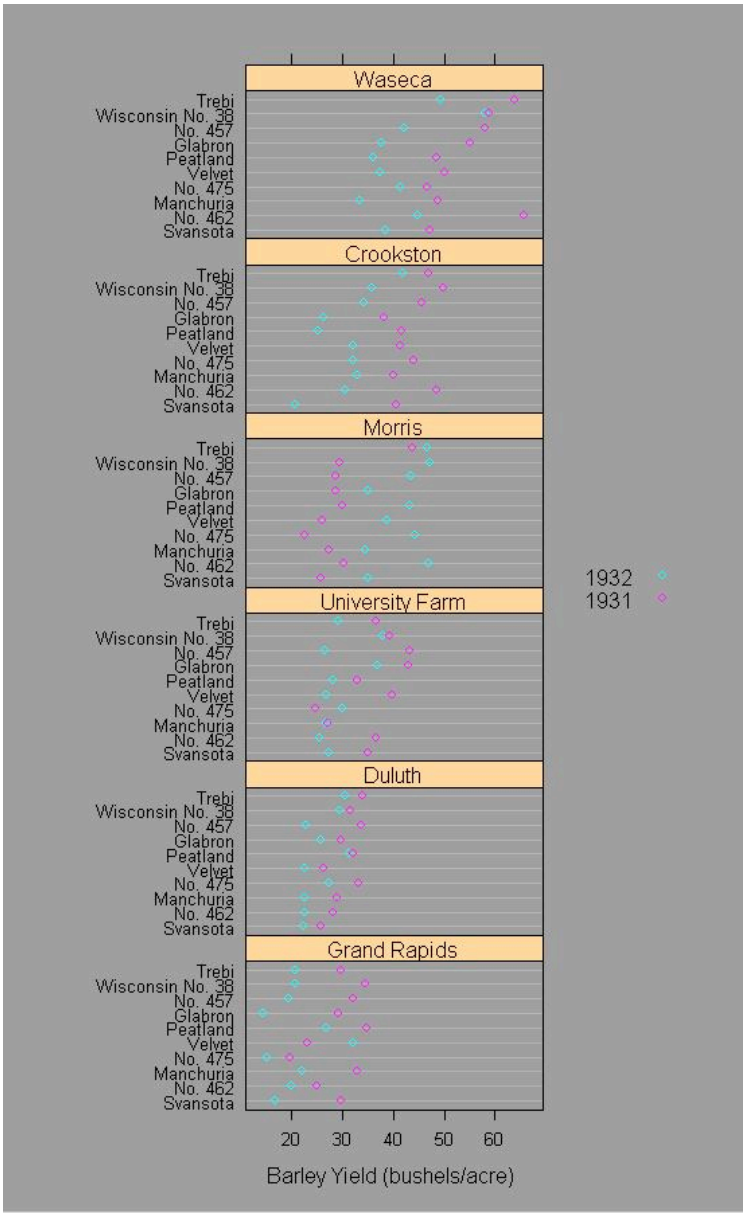
이 방법은 Trellis Device를 생성하는데, 테마(theme)를 배경 색상을 하얀색으로 바꾸는 것이다. 배경 색상외에 바뀌는 몇가지가 있다는 것도 염두에 두자. 이 방법은 dev.off()를 사용해서 Device를 닫아버리면 그 설정도 소멸한다. 그러므로 다시 High Level 함수를 사용하거나 Trellis Device를 생성할 때 theme = "col.whitebg" 옵션을 사용해야 하는 번거로움이 있다.

2. lattice.options을 이용하는 방법

```
lattice.options(default.theme ="col.whitebg");
```

이 방법은 Trellis Device의 옵션을 바꾸는 함수로서 R을 종료할 때까지 설정이 유효하다. 개인적으로 선호하는 방법이다.

그러면 Lattice의 기본 색상과 "col.whitebg" 테마의 색상을 비교해 보자. "col.whitebg" 테마의 색상은 앞서 보았던 보리 실험의 그림이다. 그러면 Lattice의 기본 색상은 어떤 모습일까? 다음과 같다. 많은 차이가 있다. 색상 외에 다른 점은 무엇일까? 1931년도의 Point가 기본 테마로는 원인데, "col.whitebg" 테마에서는 "+"로 출력된 것이다.



내친 김에 하나 더 알아보자.

Chart를 만들고 눈으로 보고 끝나지는 않을 것이다. 파일로 저장을 해서 보고서에 첨부하는 일이 있다. 개인적으로는 grDevices 패키지의 jpeg 함수를 사용하여 JPEG 형식의 그림 파일로 만들어서 보고서에 첨부하기도 한다.

그런데 Trellis의 최근 버전은 jpeg 함수에서 파일의 배경색을 지정하는 bg 옵션을 허용하지 않는다. 그래서 "col.whitebg"을 사용해도 결과물에는 배경색이 흰색이 아니라 회색에 가까운 색으로 출력된다. 물론 배경색만 차이가 난다.

```
> lattice.options(default.theme ="col.whitebg")
> jpeg("barley1.jpg",550,900)
> dotplot(variety ~ yield | site, data = barley, groups = year,
        key = simpleKey(levels(barley$year), space = "right"),
        xlab = "Barley Yield (bushels/acre) ",
        aspect=0.5, layout = c(1,6), ylab=NULL)
> dev.off()
```

이 script로 보리 실험의 dotplot은 화면이 아니라 "barley1.jpg" 파일에 550*900 크기의 파일로 저장된다. 참고로 여기 블로그에 첨부한 그림들은 이번 이야기 처음에 나온 보리 실험 그림을 빼고 전부 jpeg 함수를 이용해서 만들었다. 다른 방법으로 savePlot을 이용하는 방법이 있다. 이 함수도 grDevices 패키지에 포함된 함수이다. 특정 Device에 있는 Plot을 파일로 저장하는 함수이다.

함수의 원형은 다음과 같다.

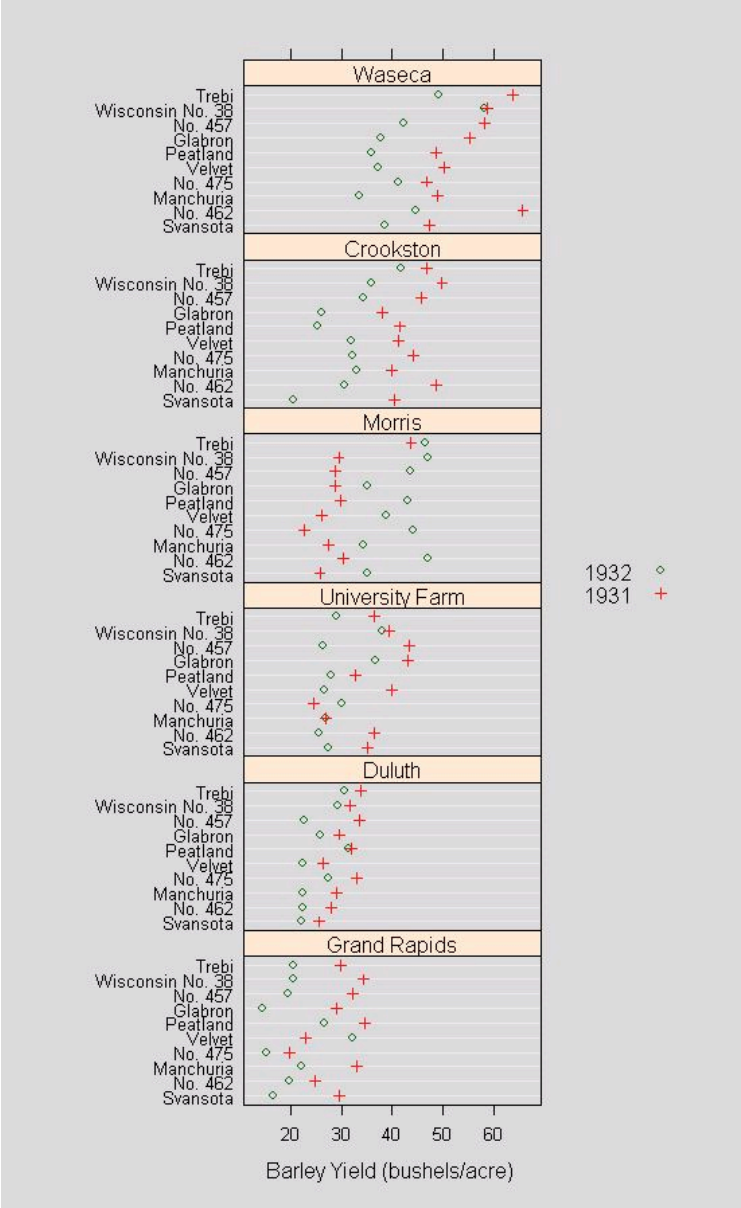
```
savePlot(filename="Rplot",
        type=c("wmf", "png", "jpeg", "jpg", "bmp", "ps", "pdf"),
        device=dev.cur())
> savePlot("barley.jpg", "jpeg")
```

이 명령어는 현재 Device에 출력된 그림을 파일로 저장하게 된다. 만약에 Trellis의 theme를 "col.whitebg"로 설정하였다면, jpeg 함수를 사용할 때와는 달리 배경색이 회색 계열이 아니라 흰색으로 화면 그대로의 색상으로 파일에 저장된다.

그런데 jpeg 함수를 사용할 경우에는 Plot 명령어가 화면에 출력되는 것이 아니라 파일로 리다이렉션되기 때문에, 파일을 열어보기 전에는 그 모습을 볼 수 없다. 그러나 savePlot 함수는 일단 화면에 출력하고 파일로 저장할 수 있으며, Trellis Plot일 경우에 배경색도 흰색으로 저장할 수 있다는 장점이 있음에도 Plot의 크기(가로*세로)를 지정할 수 없어서 jpeg을 선호해 사용한다.

왜냐하면 Trellis에서만 색상의 문제가 있으며, 네이버 블르그에서는 그림의 가로 크기가 550을 넘어가면 웹브라우저에 보여줄 때, 축소가 되면서 Plot 모양이 지저분하게 보이기 때문이다. 단, window Device(Graphic 화면) Window를 마우스로 이용해서 조정하면 그 크기대로 그림의 크기가 저장되기는 하지만, 일관성이 없고 그 때 그때 다른 크기의 파일이 생기는 점때문에 블로그를 위해서는 jpeg를 사용한다. bmp, png등의 함수도 기능은 똑같고, 파일 형식만 다르지만 네이버에서 GIF, JPG만 지원하니 어쩔 수 없이 jpeg를 사용한다. 또한 압축된 포멧을로 저장되기 때문에 BMP나 GIF에 비해서 파일의 용량이 작은 점도 있다. 물론 압축시 화질이 떨어지는 경향이 있으나 사진 이미지가 아닌 Chart의 이미지에서는 큰 차이가 없다.

jpeg를 이용한 결과를 보면서 이번 이야기를 마친다.



Lattice Bivariate High Level Function

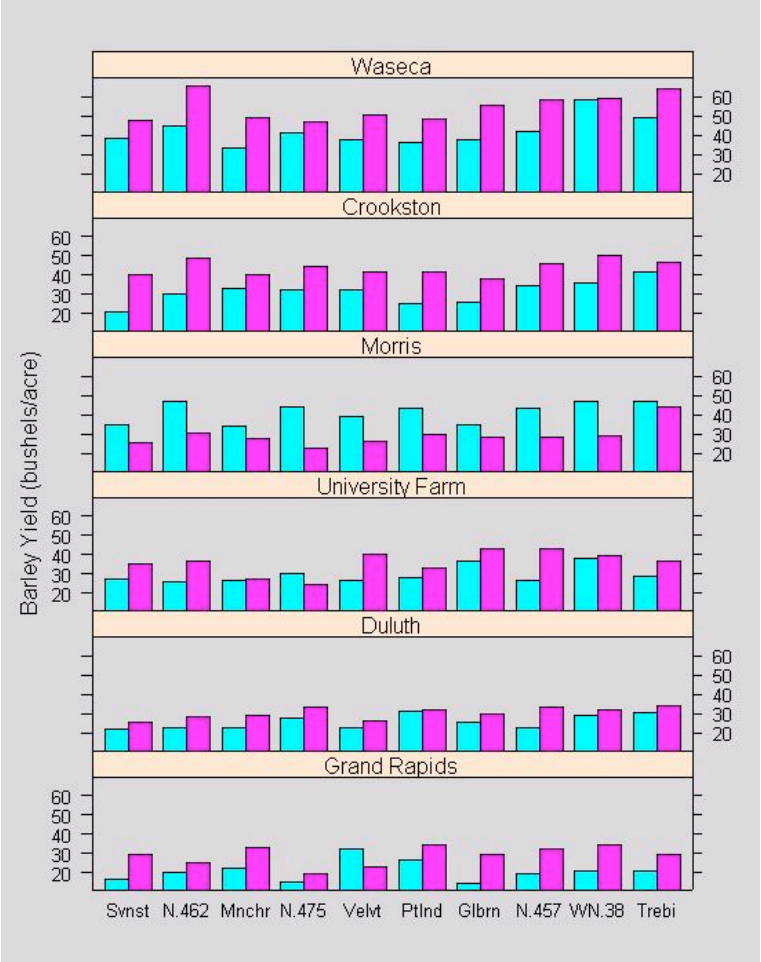
Lattice의 High Level 함수 중에서 단변량 Chart에 대해서 알아보자. 일단 여기서는 Low Level 함수의 사용법에 대해서는 다루지 않겠다. 다만 S-Plus의 Trellis와의 비교 차원에서 R에서도 S-Plus에서처럼 Trellis Plot을 사용할 수 있다는 것을 보이기 위함이다.

1. barcharts

S Graphics의 barplot의 Trellis 버젼으로 Bar Chart를 출력한다.

```
> barchart(yield ~ variety | site, data = barley,
        groups = year, layout = c(1,6),
        ylab = "Barley Yield (bushels/acre)",
        scales = list(x = list(abbreviate = TRUE,
                minlength = 5)))
```

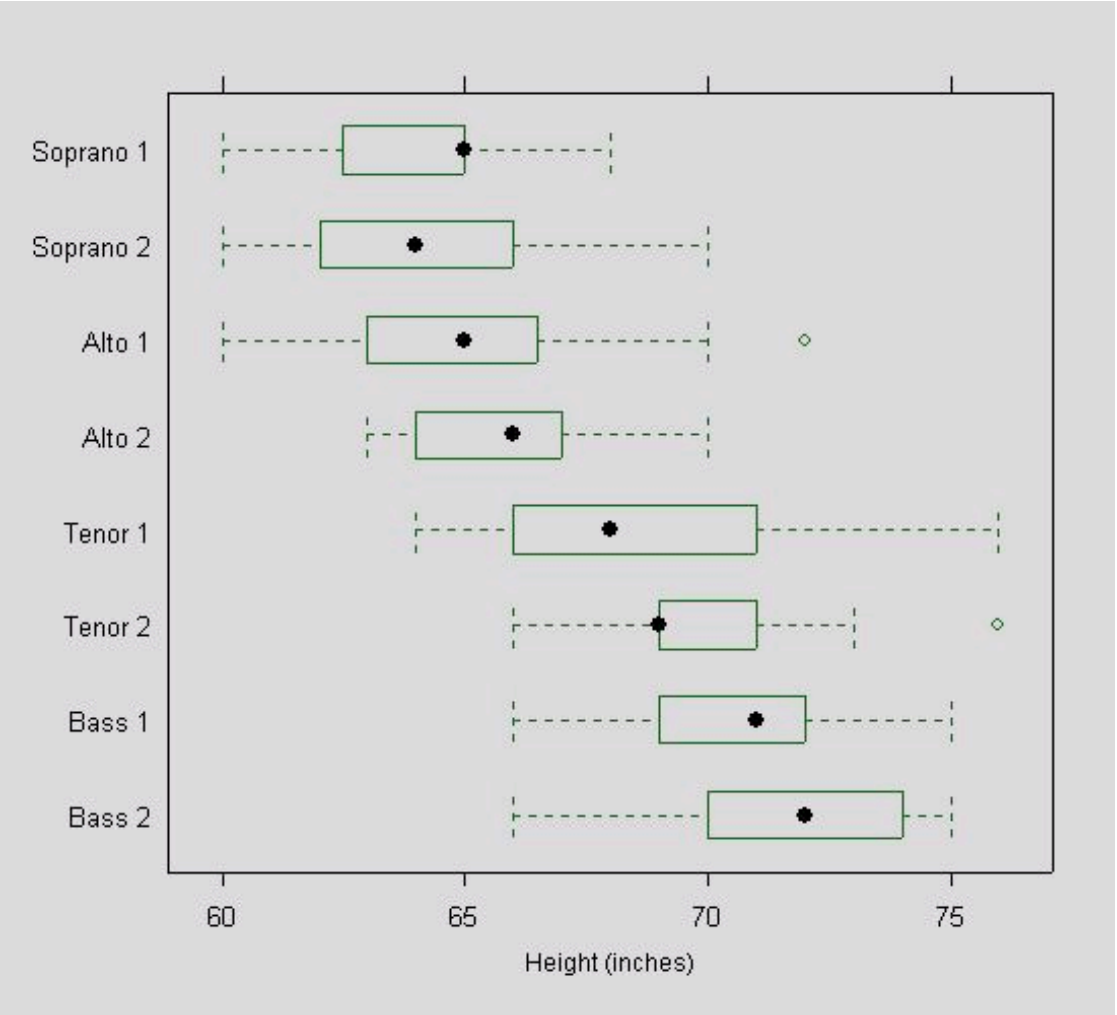
보리재배 실험의 데이터를 barchart로 표현하였다. Duluth 실험장에서는 품종별로 두 년도 간의 차이가 다른 실험장보다 적게 나타난다.



2. box and whisker plots

S Graphics의 boxplot의 Trellis 버젼으로 수염 상자그림을 출력한다.

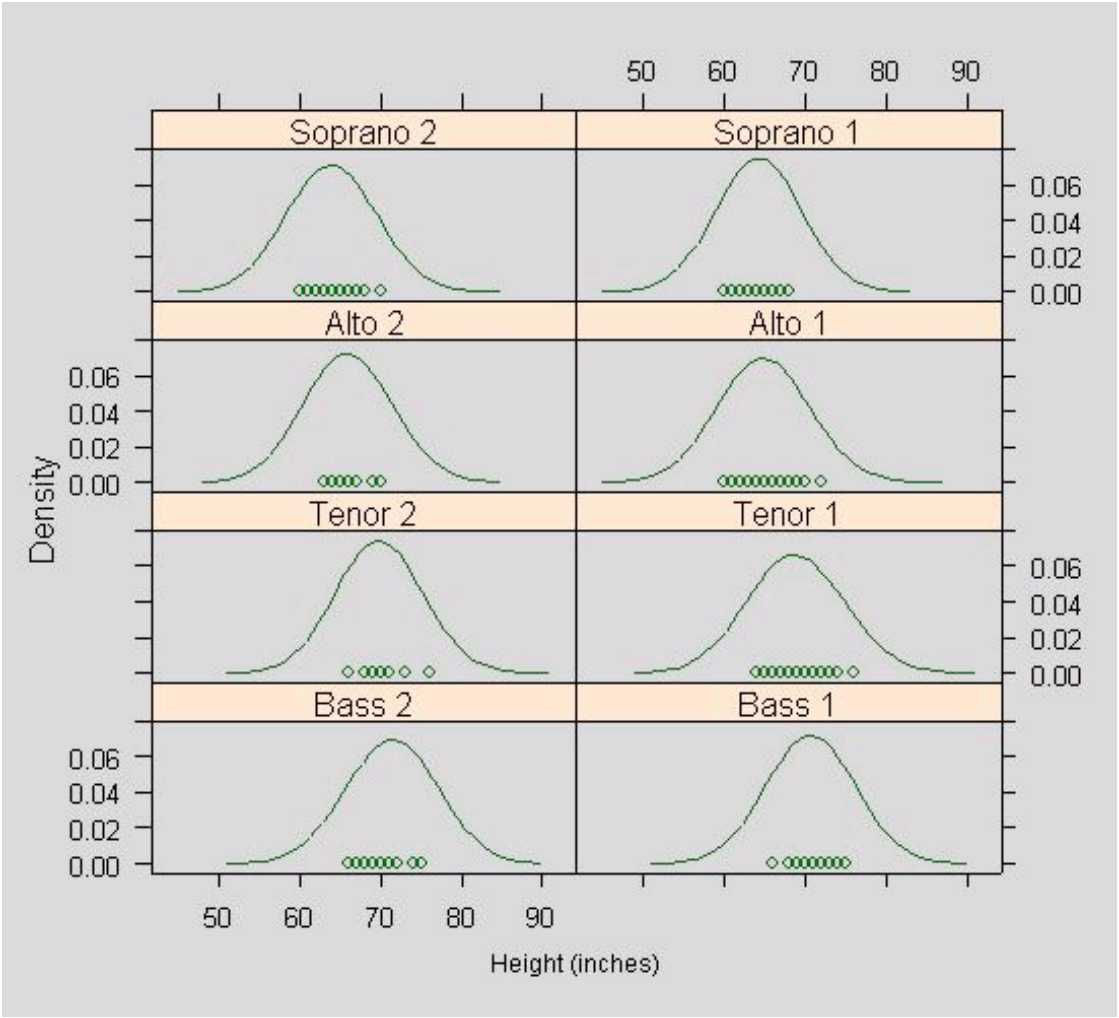
```
> bwplot(voice.part ~ height, data=singer, xlab="Height (inches)")
```



3. kernel density plots

S Graphics의 density의 Trellis 버젼이라고 할 수 있다. kernel density plots를 출력한다.

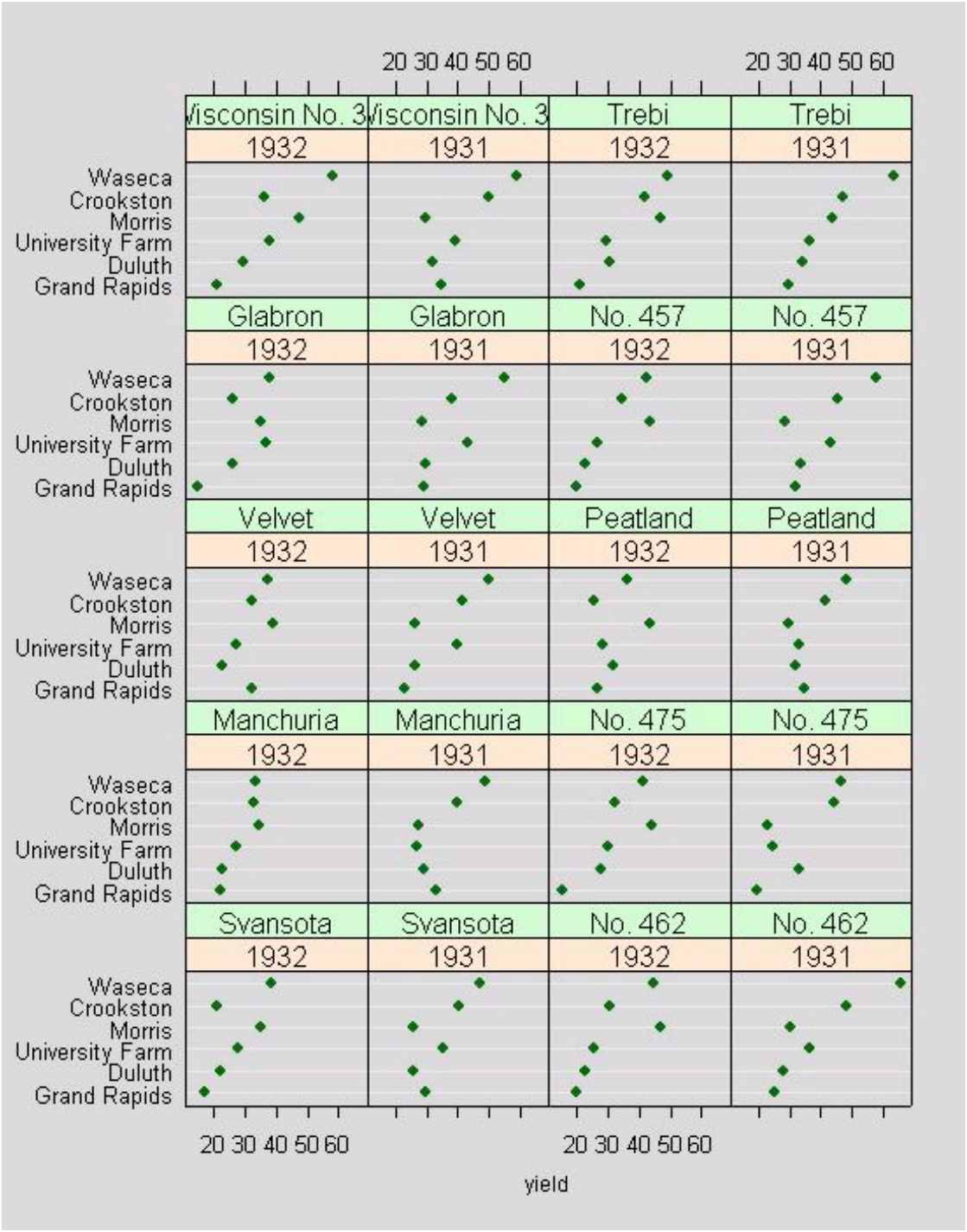
```
> densityplot( ~ height | voice.part, data = singer, layout = c(2, 4),
        xlab = "Height (inches)", bw = 5)
```



4. dot plots

S Graphics의 dotchart의 Trellis 버젼으로 dot plots를 출력한다.
4 columns, 5 rows, 1page로 배열하였다.

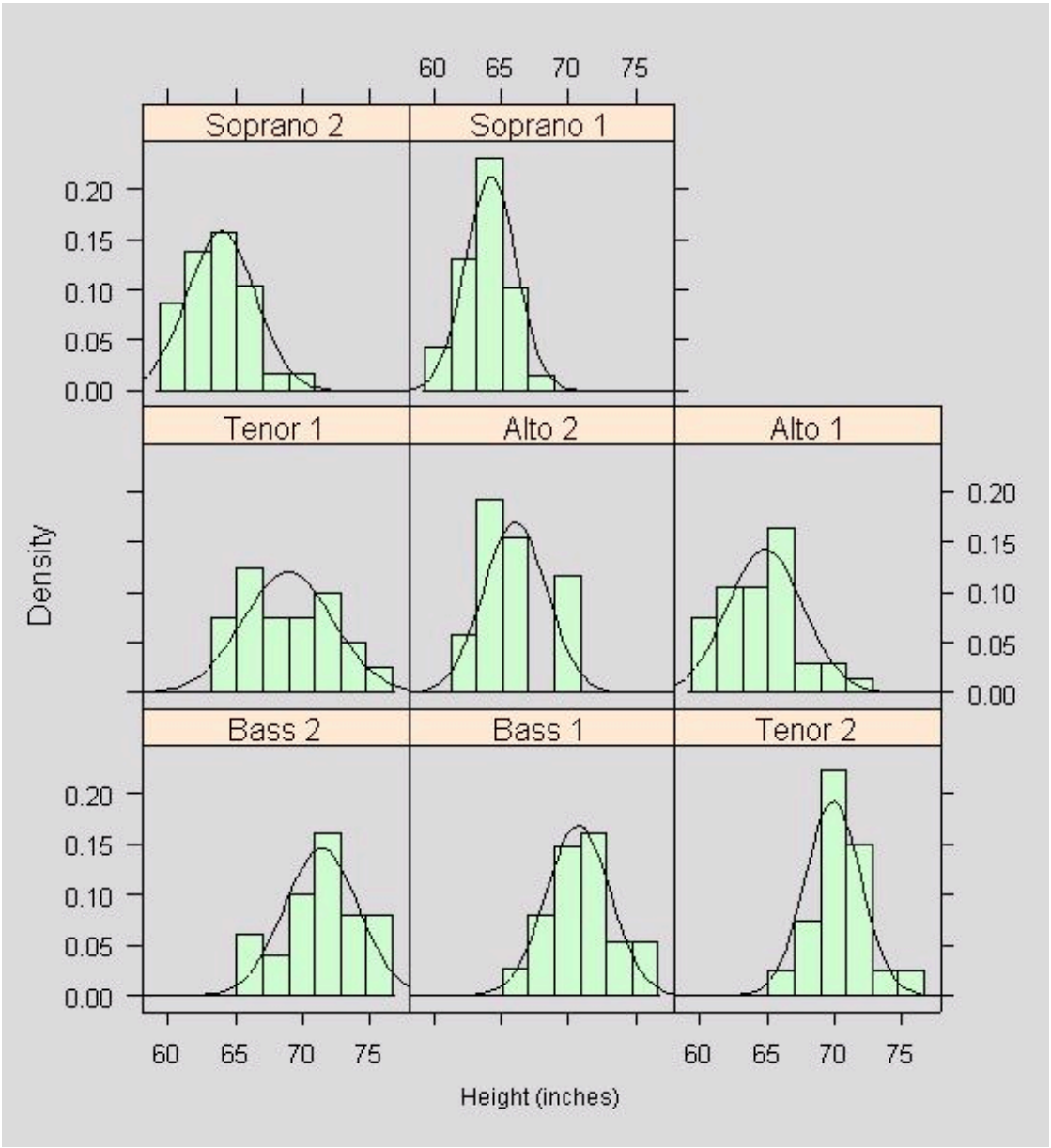
```
> dotplot(site ~ yield | year * variety, layout=c(4,5,1), data=barley)
```



5. histograms

S Graphics의 hist의 Trellis 버젼이라고 할 수 있다. 히스토그램을 출력한다.

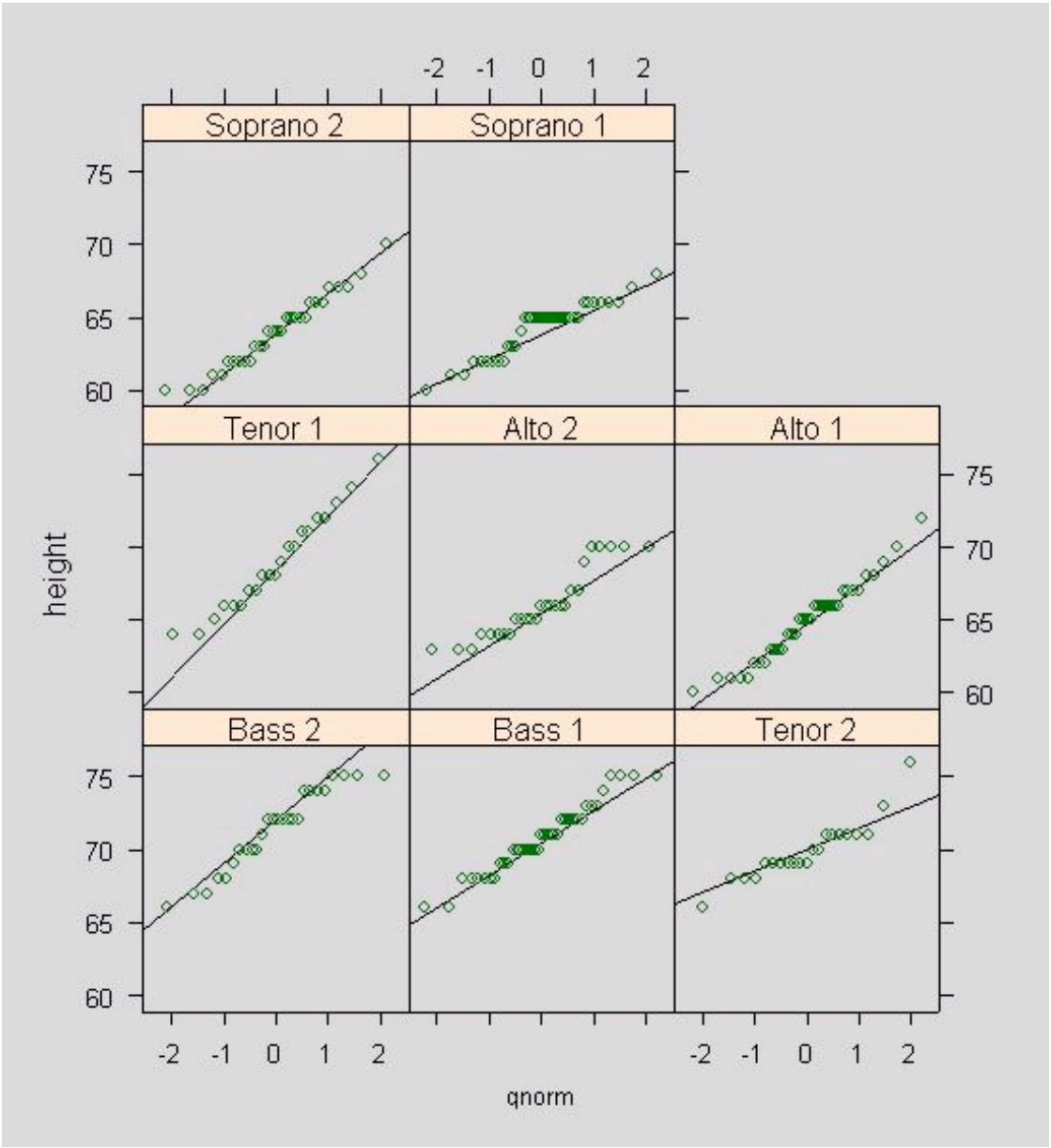
```
> histogram( ~ height | voice.part, data = singer,
        xlab = "Height (inches)", type = "density",
        panel = function(x, ...) {
            panel.histogram(x, ...)
            panel.mathdensity(dmath = dnorm, col = "black",
                              args = list(mean=mean(x),sd=sd(x)))
        } )
```



6. quantile plots against mathematical distributions

S Graphics의 qq의 Trellis 버전으로 Quantile plots를 출력한다.

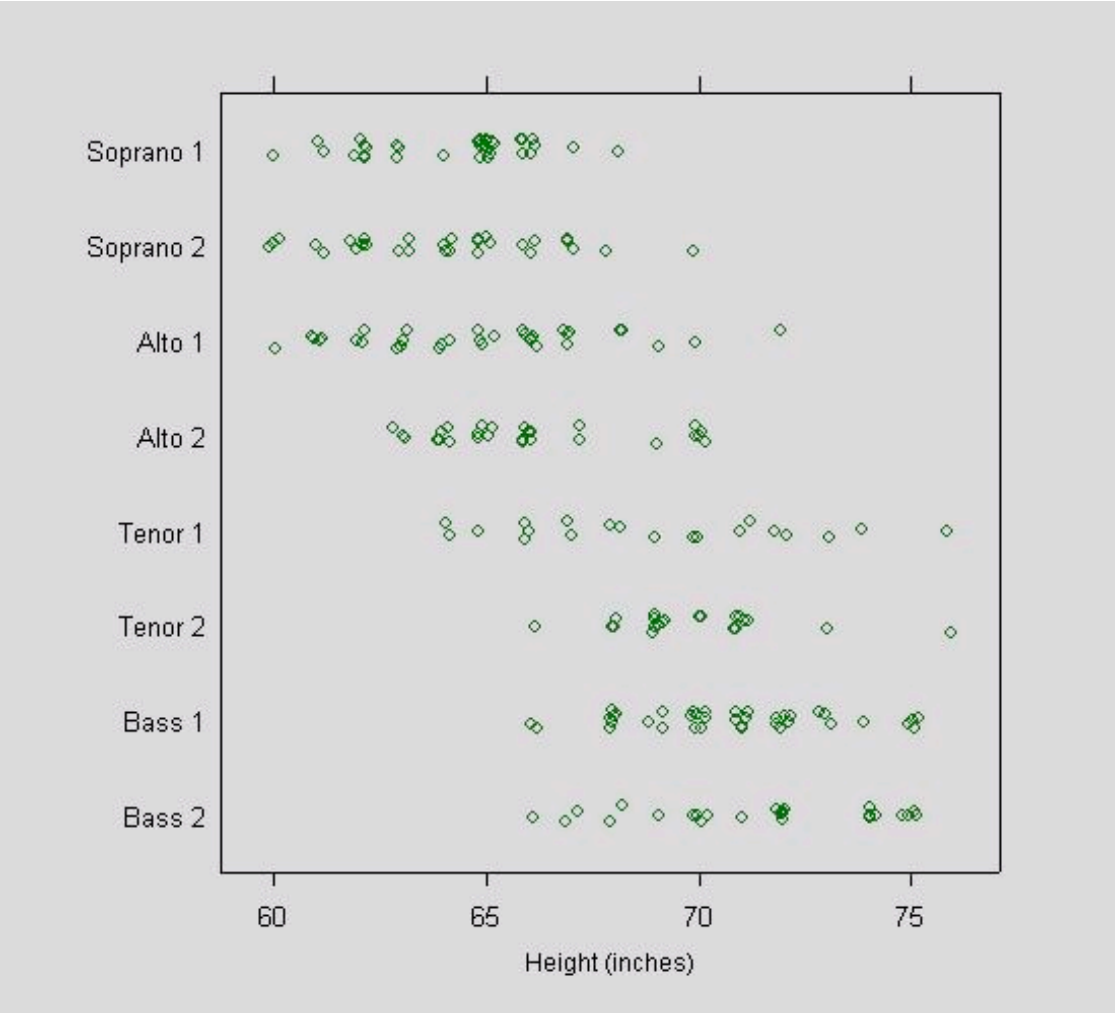
```
〉 qqmath(~ height | voice.part, aspect = 1, data = singer,
       prepanel = prepanel.qqmathline,
       panel = function(x, y) {
          panel.qqmathline(y, distribution = qnorm)
          panel.qqmath(x, y)
       })
```



7. 1-dimensional scatterplot

stripplot를 출력한다.

```
〉 stripplot(voice.part ~ jitter(height), data = singer, aspect = 1,
        jitter = TRUE, xlab = "Height (inches)")
```



지난 번에 언급한 바와 같이 Trellis와 비교하면 Pie Chart에 대한 Lattice만 없다.

Lattice Otherwise High Level Function

Lattice의 High Level 함수중에서 이변량, 삼변량, 복합변량, 그외 잡다한 Chart에 대해서 알아보자.

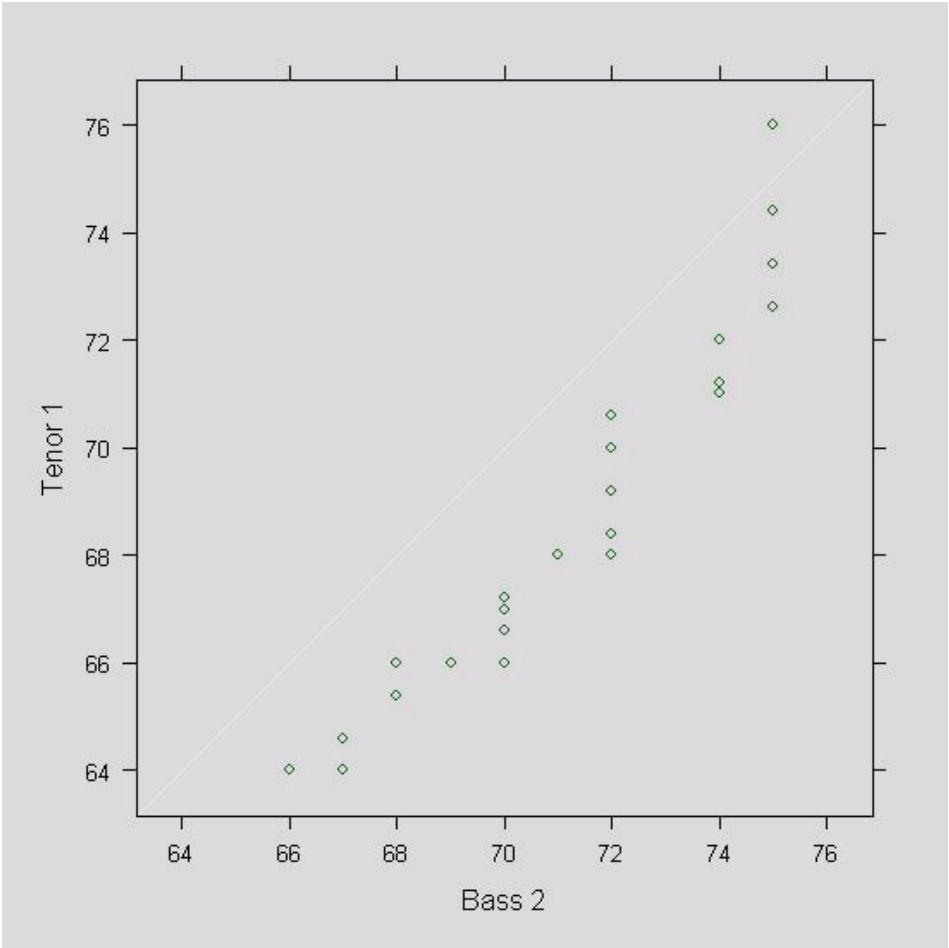
Bivariate

X축과 Y축을 가지는 2차원 변량의 Chart.

1. qq

S Graphics의 qqplot의 Trellis 버전이라고 할 수 있다. Quantile-Quantile Plots를 출력한다.

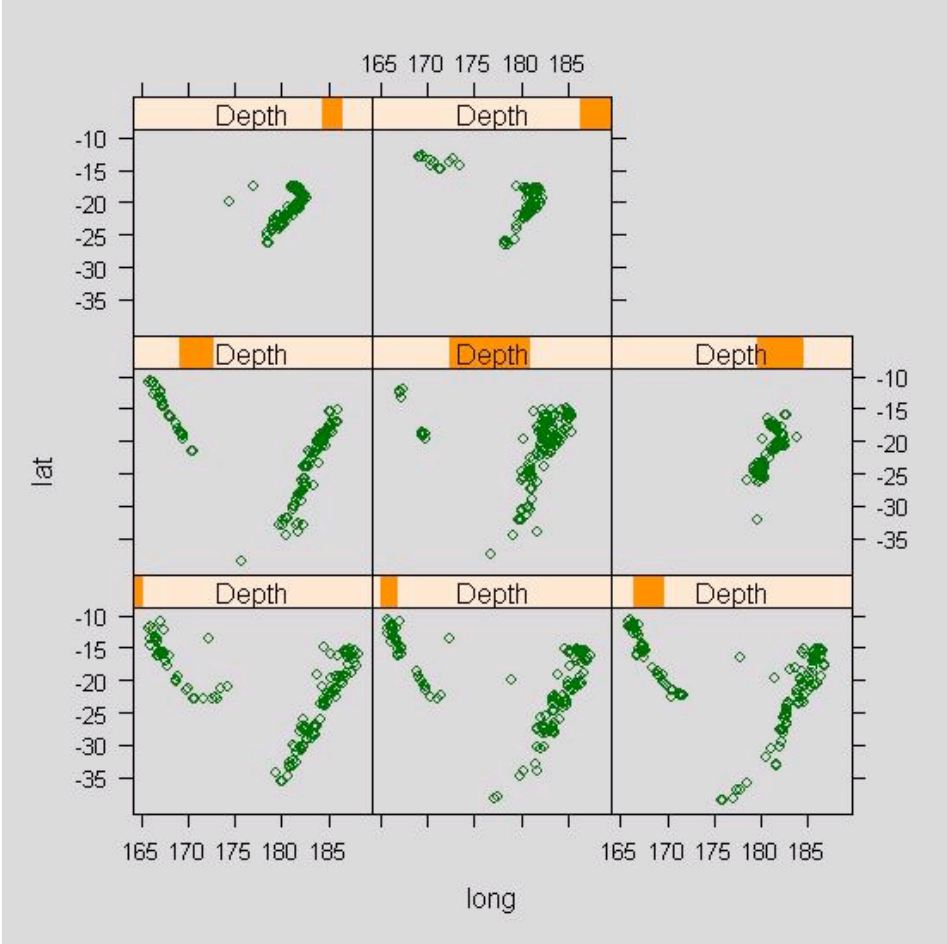
```
〉 qq(voice.part ~ height, aspect = 1, data = singer,
   subset = (voice.part == "Bass 2" | voice.part == "Tenor 1"))
```



2. xyplot

S Graphics의 plot의 Trellis 버전이라고 할 수 있다. plot이 S Graphics의 기본인 것처럼 Trellis Plots의 기본이다.

```
> Depth <- equal.count(quakes$depth, number=8, overlap=.1)
> xyplot(lat ~ long | Depth, data = quakes)
```



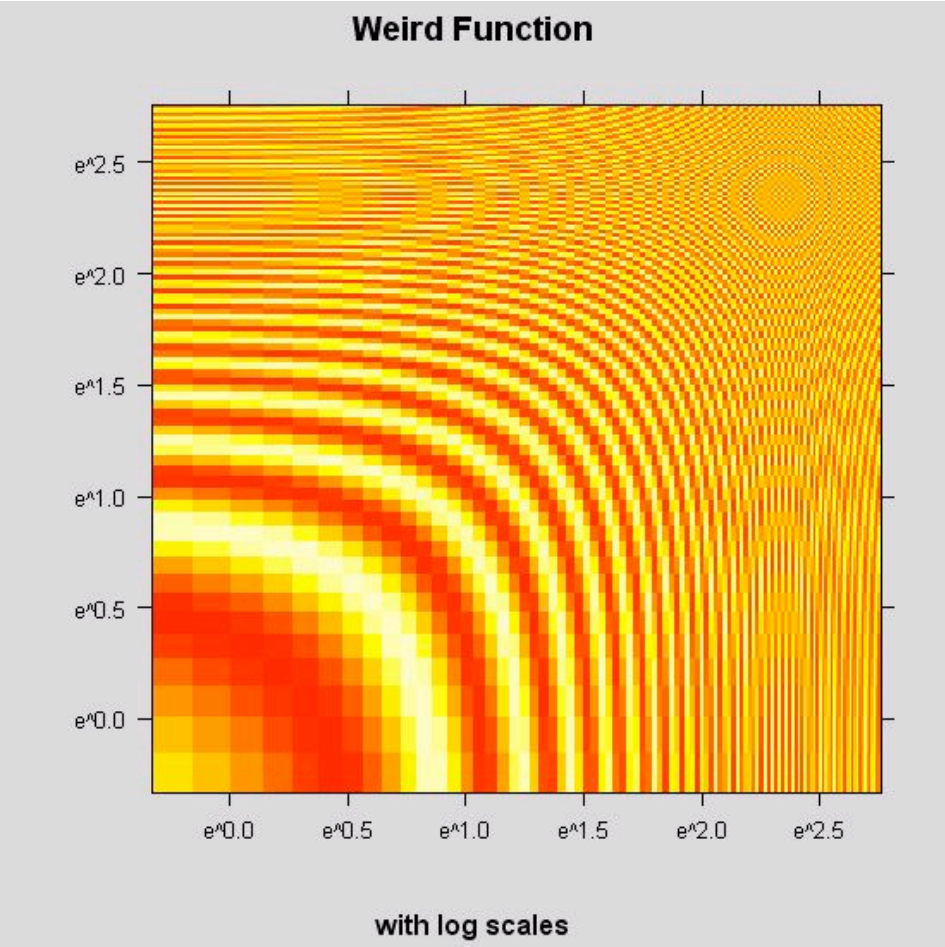
Trivariate

X축과 Y축, Z축을 가지는 3차원 변량의 Chart.

3. levelplot

S Graphics의 image의 Trellis 버전이라고 할 수 있다. Level Plots를 출력한다.

```
> x <- seq(pi/4, 5 * pi, length = 100)
> y <- seq(pi/4, 5 * pi, length = 100)
> r <- as.vector(sqrt(outer(x^2, y^2, "+")))
> grid <- expand.grid(x=x, y=y)
> grid$z <- cos(r^2) * exp(-r/(pi^3))
> levelplot(z~x*y, grid, cuts = 50, scales=list(log="e"), xlab="",
        ylab="", main="Weird Function", sub="with log scales",
        colorkey = FALSE, region = TRUE)
```

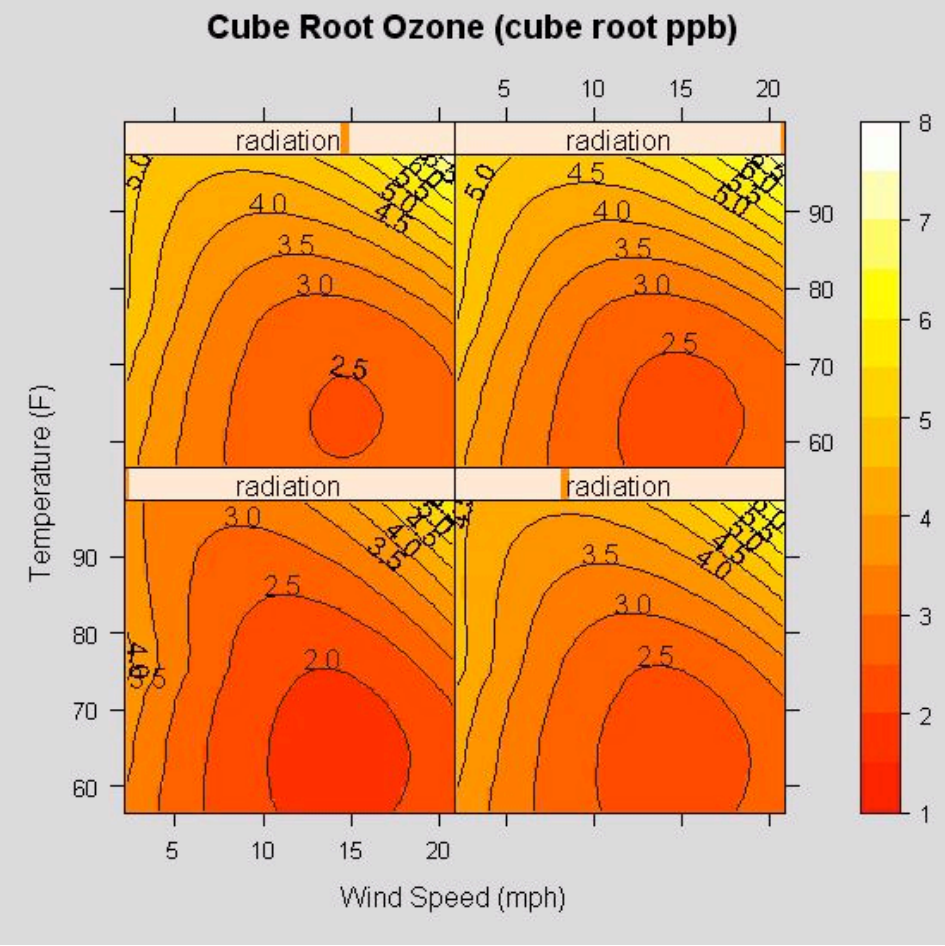


4. contourplot

S Graphics의 contour의 Trellis 버전이라고 할 수 있다. Contour plots를 출력한다.

```
> require(stats)
> attach(environmental)
> ozo.m <- loess((ozone^(1/3)) ~ wind * temperature * radiation,
      parametric = c("radiation", "wind"), span = 1, degree = 2)
> w.marginal <- seq(min(wind), max(wind), length = 50)
```

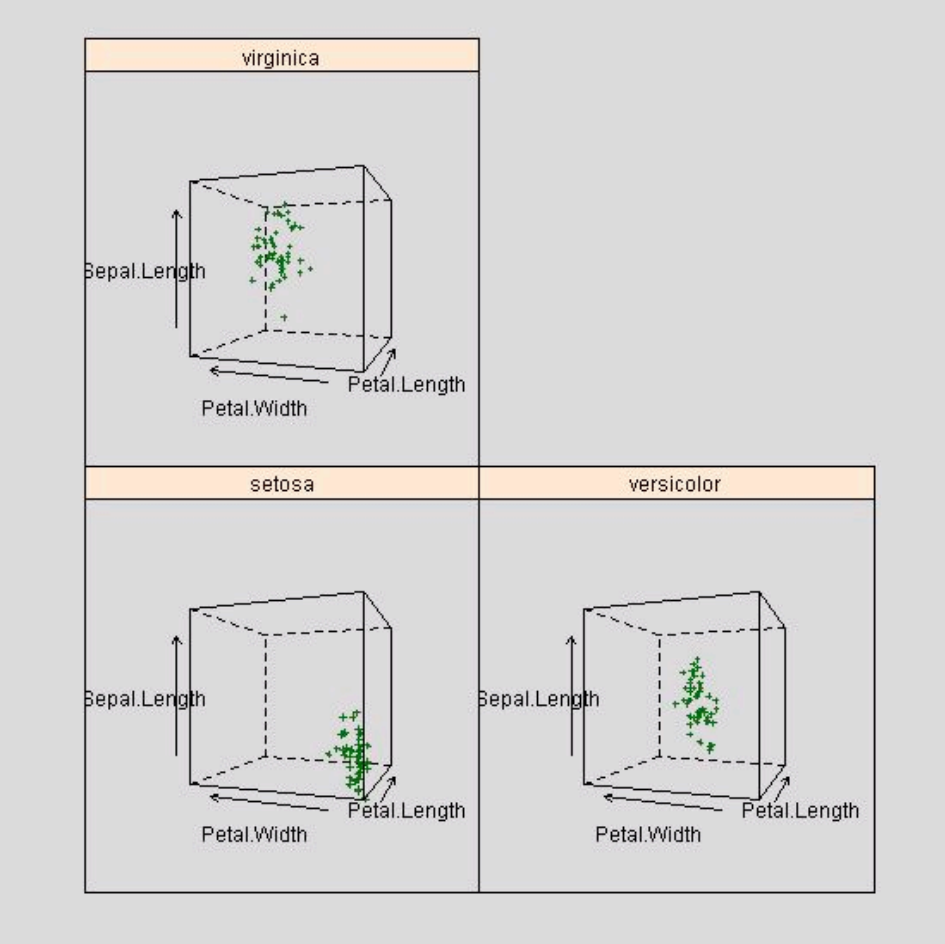
```
> t.marginal <- seq(min(temperature), max(temperature), length = 50)
> r.marginal <- seq(min(radiation), max(radiation), length = 4)
> wtr.marginal <- list(wind = w.marginal, temperature = t.marginal,
      radiation = r.marginal)
> grid <- expand.grid(wtr.marginal)
> grid[, "fit"] <- c(predict(ozo.m, grid))
> contourplot(fit ~ wind * temperature | radiation, data = grid,
          cuts = 10, region = TRUE,
          xlab = "Wind Speed (mph)",
          ylab = "Temperature (F)",
          main = "Cube Root Ozone (cube root ppb)")
> detach()
```



5. cloud

3-D scatter plots를 출력한다.

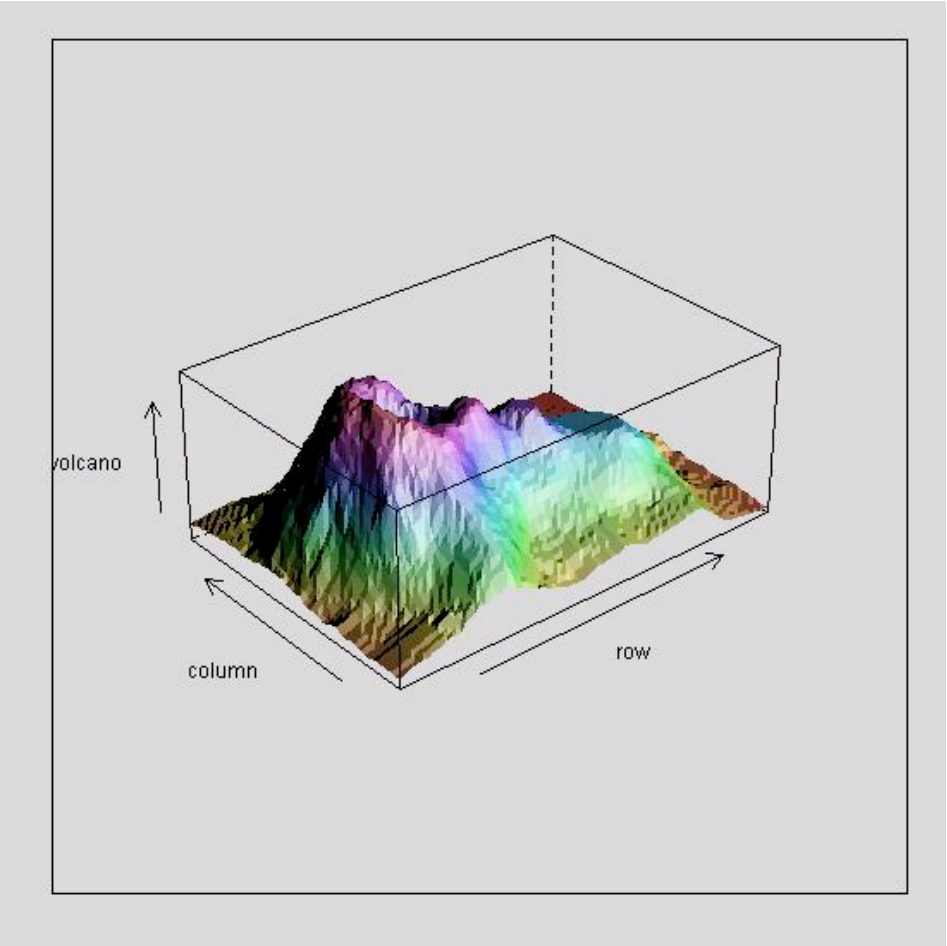
```
> cloud(Sepal.Length ~ Petal.Length * Petal.Width | Species, data=iris,
      screen = list(x = -90, y = 70), distance = .4, zoom = .6)
```



6. wireframe

S Graphics의 persp의 Trellis 버전이라고 할 수 있다. 3-D surfaces를 출력한다.

```
> wireframe(volcano, shade = TRUE, aspect = c(61/87, 0.4),
        light.source = c(10,0,10))
```



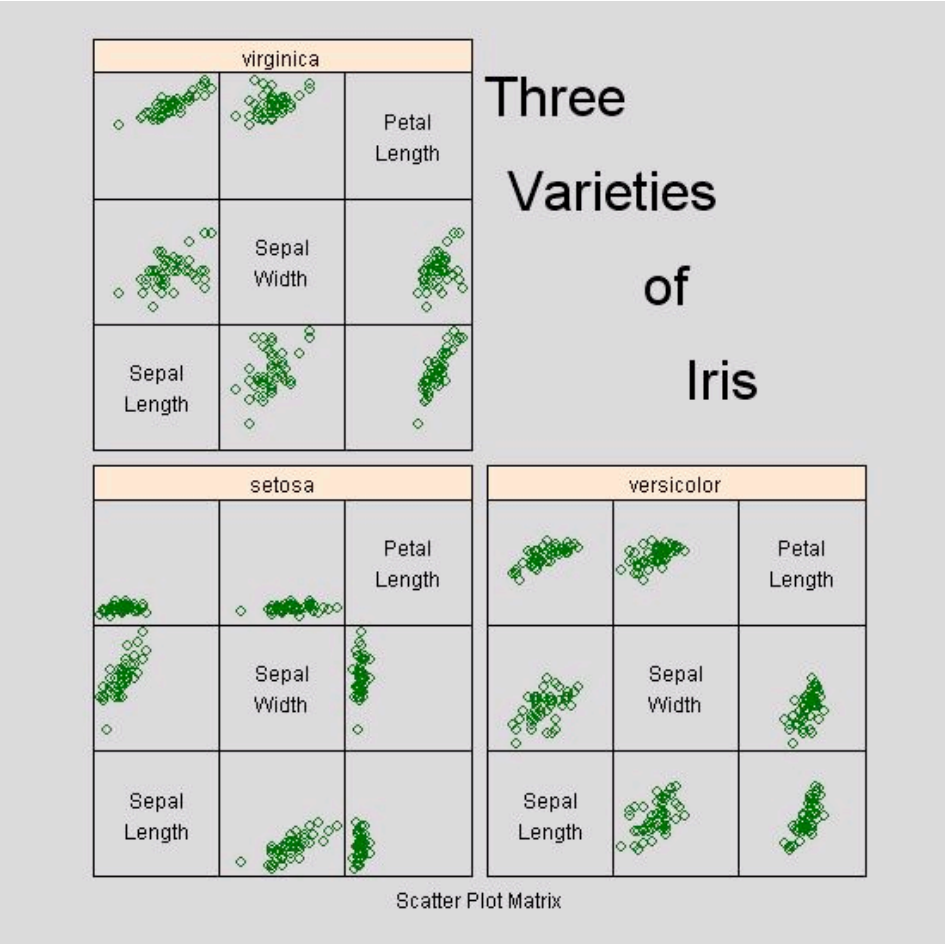
Hypervariate

복합된 형태의 Chart.

7. splom

Scatter Plot의 Matrices를 출력한다. 굳이 비교하자면 S Graphics의 pairs와 유사하다고 할 수 있다.

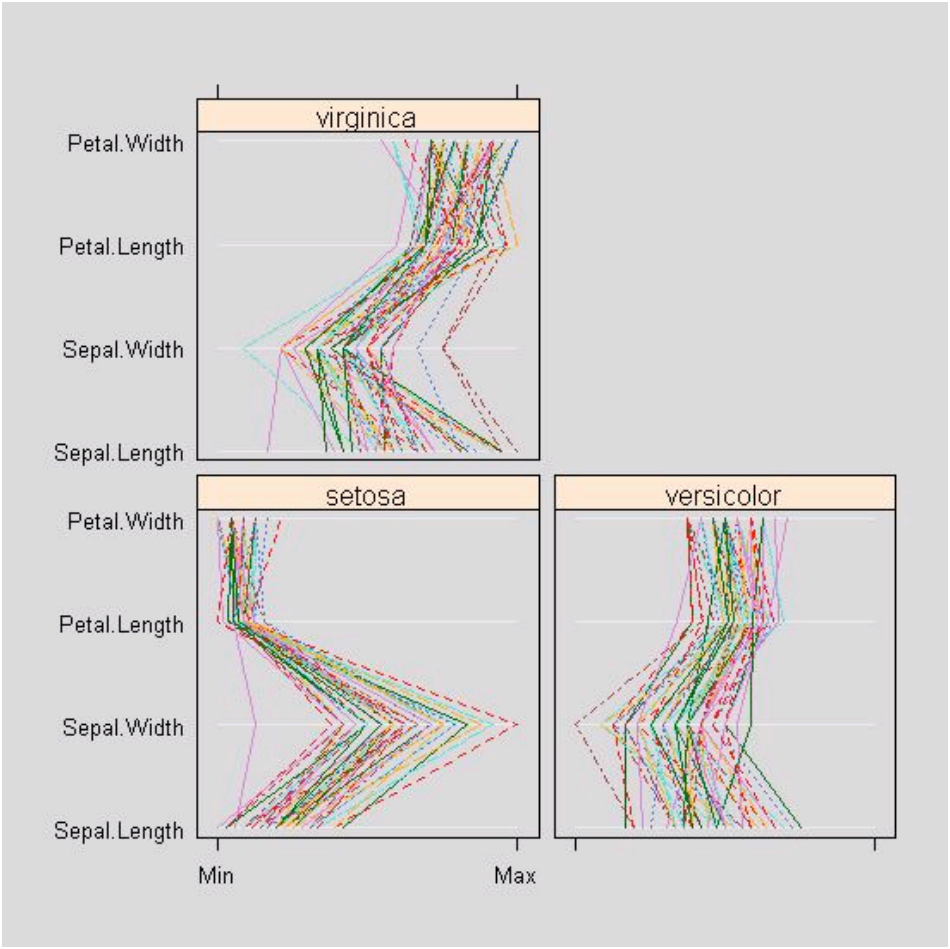
```
〉 splom(~iris[1:3]|Species, data = iris,
      layout=c(2,2), pscales = 0,
      varnames = c("Sepal\nLength", "Sepal\nWidth", "Petal\nLength"),
      page = function(...) {
         ltext(x = seq(.6, .8, len = 4),
               y = seq(.9, .6, len = 4),
               lab = c("Three", "Varieties", "of", "Iris"),
               cex = 2)
      })
```



8. parallel

Parallel Coordinate Plots를 출력한다. 다변량 자료의 Classification 분석 등에 주로 사용한다.

> parallel(~iris[1:4] | Species, iris)



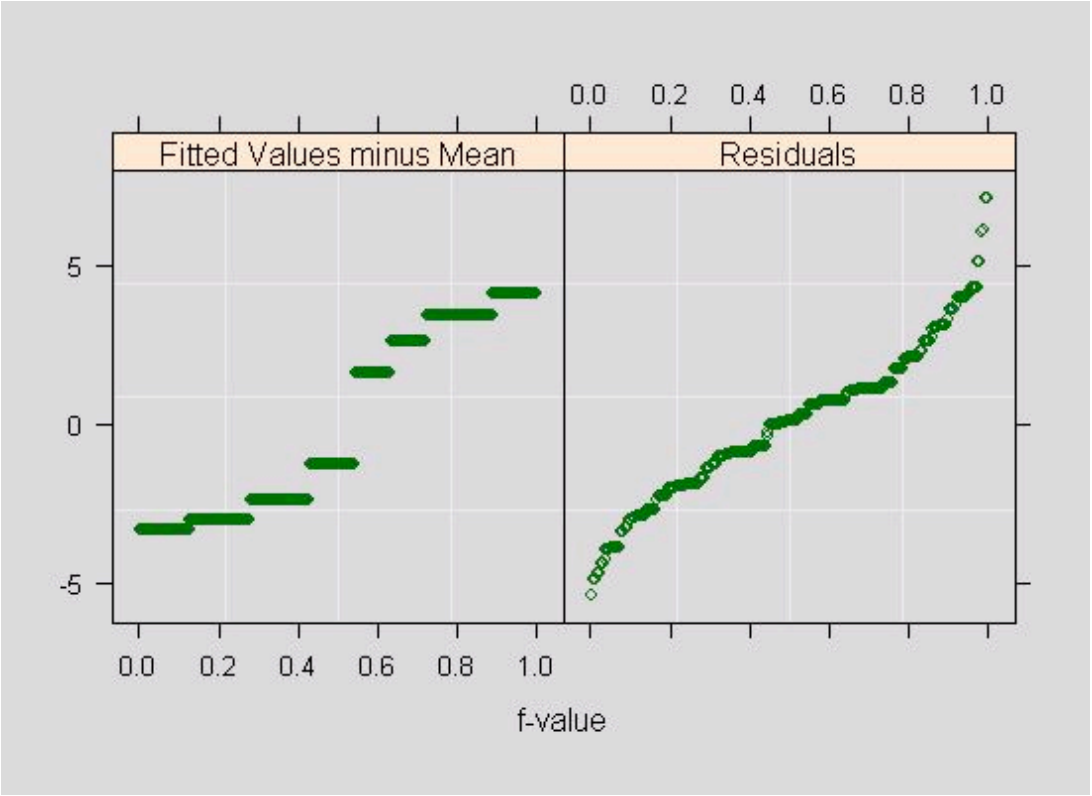
Miscellaneous

그 밖의 Chart.

9. rfs

residual and fitted value plot을 출력한다.

〉 rfs(oneway(height ~ voice.part, data = singer, spread = 1), aspect = 1)



10. tmd

Tukey Mean-Difference plots를 출력한다.

> tmd(qqmath(~height|voice.part, data = singer))