R in the Windows Lab: Startup

Get the correct R

Click start on lower left in search box: [r 2] and click on R 2.15.1

Get lattice.RData from course web site

http://ml.stat.purdue.edu/stat695t

Save to [Home Directory] which is [H:], the H drive

In R

```
> attach("H:/lattice.RData")
```

General Display Functions

```
xyplot()
bwplot()
stripplot()
qq()
dotplot()
qqmath()
histogram()
splom()
contourplot()
levelplot()
wireframe()
```

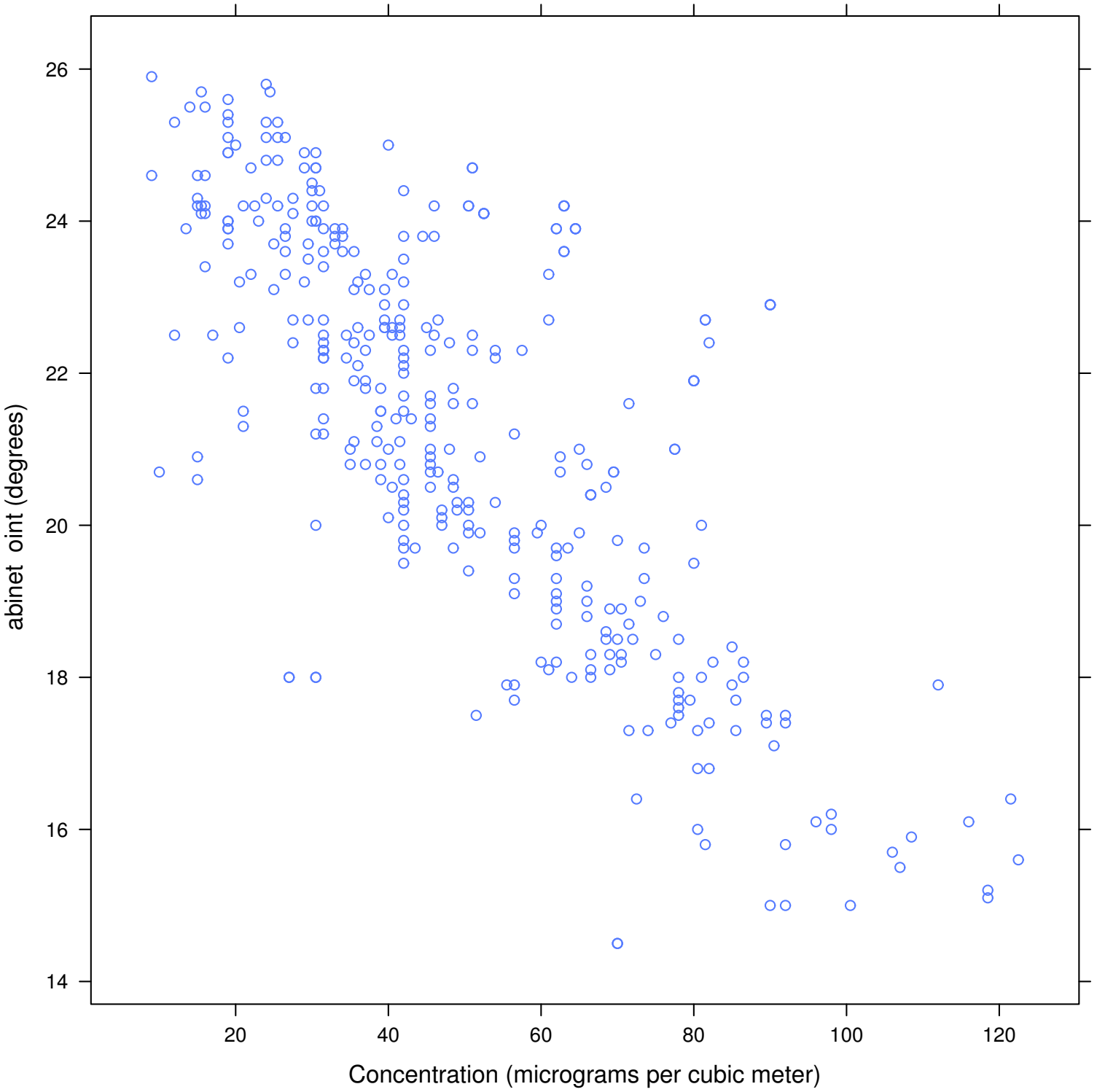
polarization

```
> dim(polarization)
[1] 355 2

> summary(polarization)
concentration      babinet
Min.   :  9.00  Min.   :14.50
1st Qu.: 31.50  1st Qu.:19.45
Median : 45.50  Median :21.60
Mean   : 49.58  Mean   :21.29
3rd Qu.: 66.00  3rd Qu.:23.60
Max.   :122.50  Max.   :25.90
```

xyplot()

```
xyplot(babinet ~ concentration,
       data = polarization,
       aspect = 1,
       xlab = "Concentration (micrograms per cubic meter)"
       ylab = "Babinet Point (degrees)")
```


26
24
22
20
18
16
14
abinet oint (degrees)
20
40
60
80
100
120
Concentration (micrograms per cubic meter)

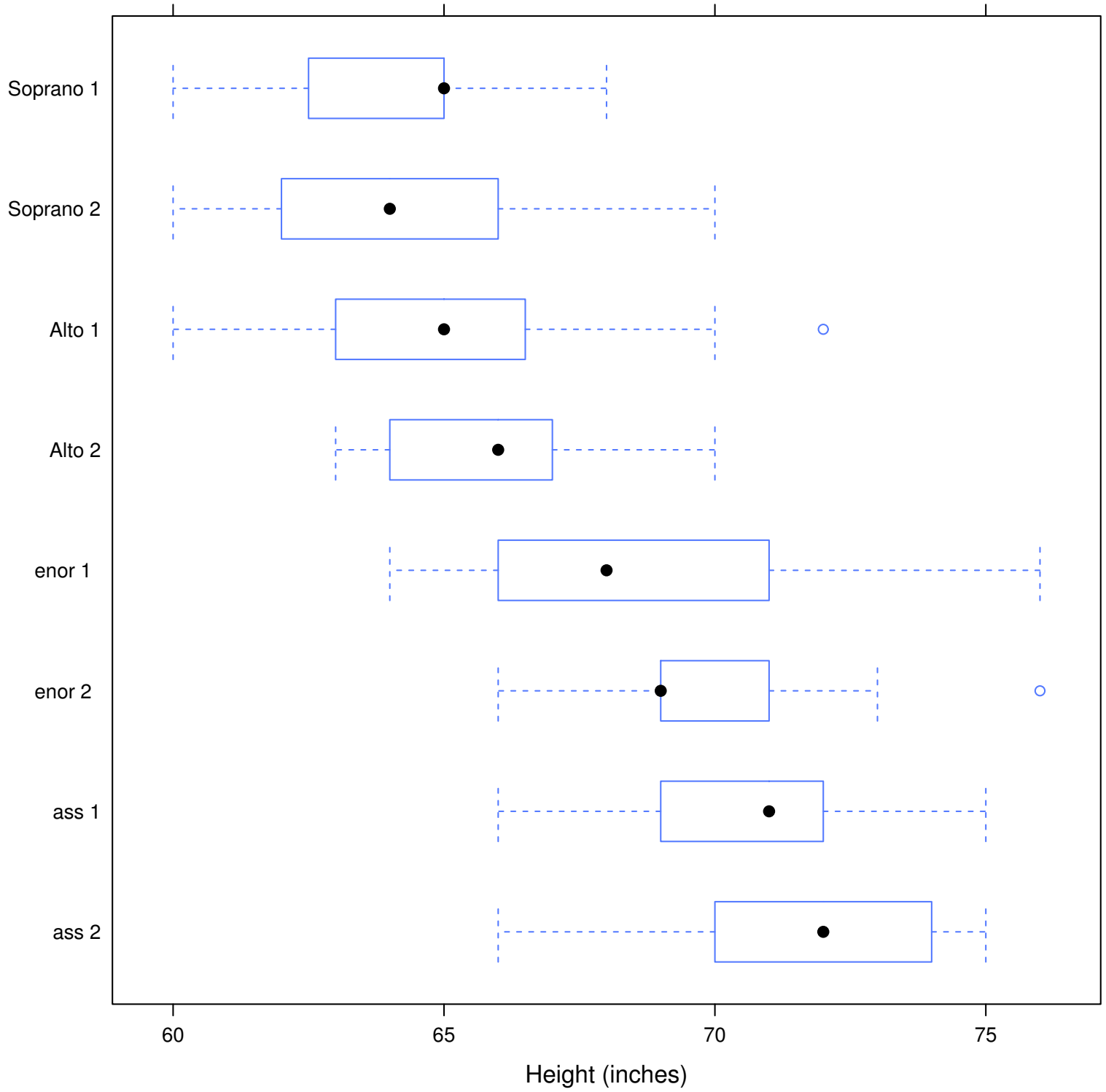
singer

```
> dim(singer)
[1] 235 2

> summary(singer)
     height          voice.part
Min.   :60.0   Bass 1   :39
1st Qu.:65.0   Soprano 1:36
Median :67.0   Alto 1   :35
Mean   :67.3   Soprano 2:30
3rd Qu.:70.0   Alto 2   :27
Max.   :76.0   Bass 2   :26
               (Other)  :42
```

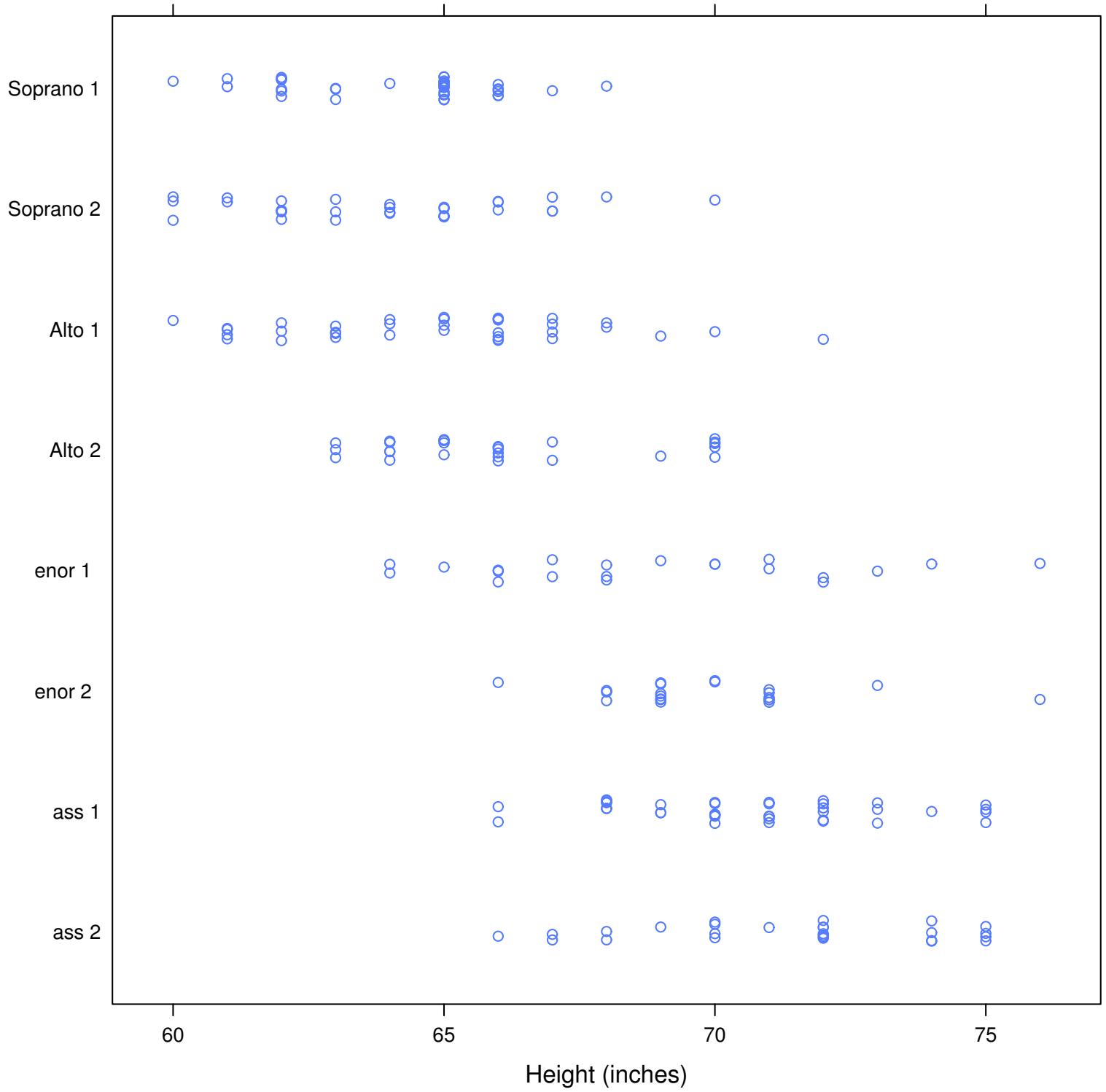
bwplot()

```
bwplot(voice.part ~ height,
       data = singer,
       aspect = 1,
       xlab = "Height (inches)")
```


Soprano 1
Soprano 2
Alto 1
Alto 2
enor 1
enor 2
ass 1
ass 2
60
65
70
75
Height (inches)

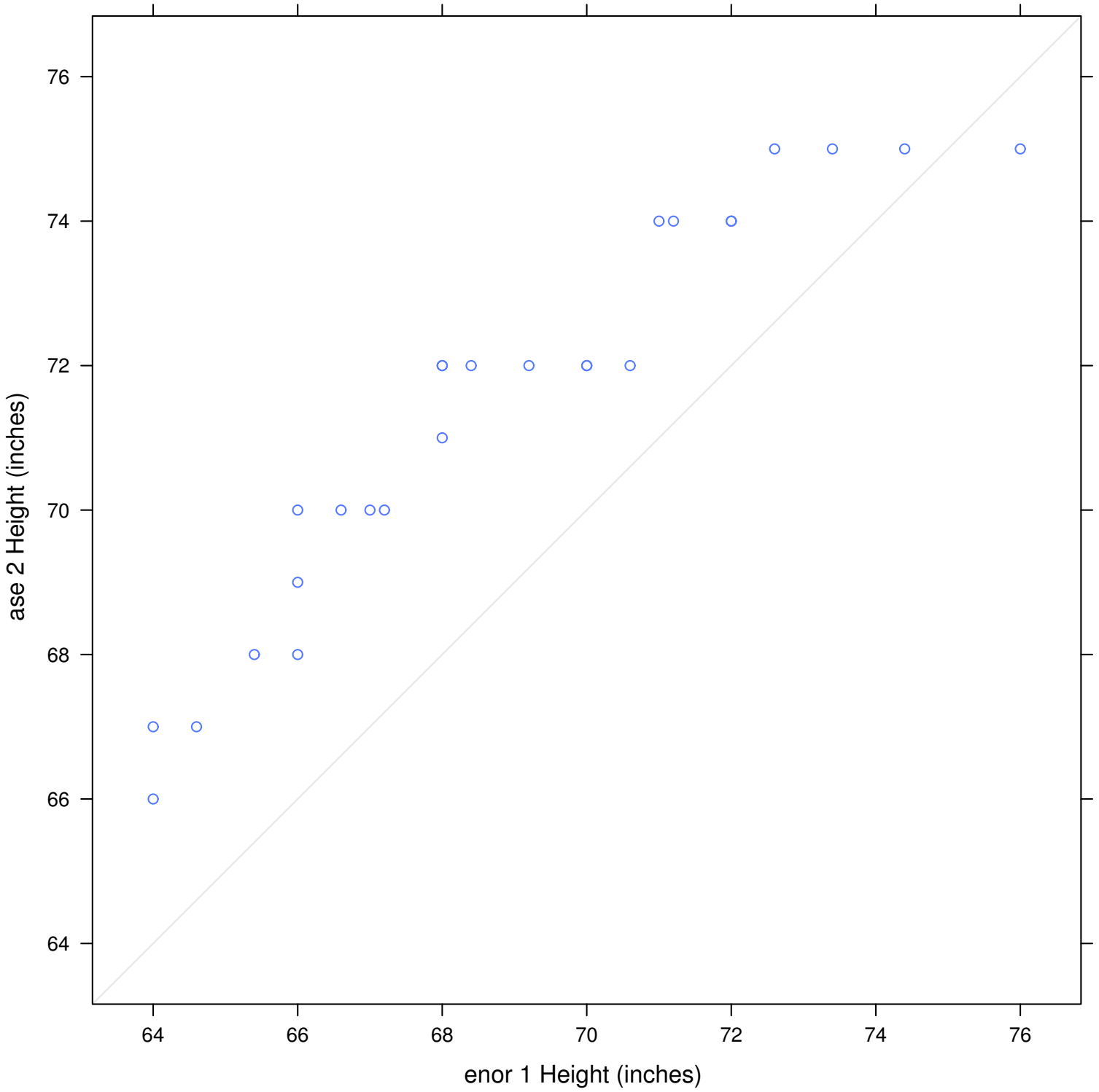
stripplot()

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


Soprano 1
Soprano 2
Alto 1
Alto 2
enor 1
enor 2
ass 1
ass 2
60
65
70
75
Height (inches)

qq()

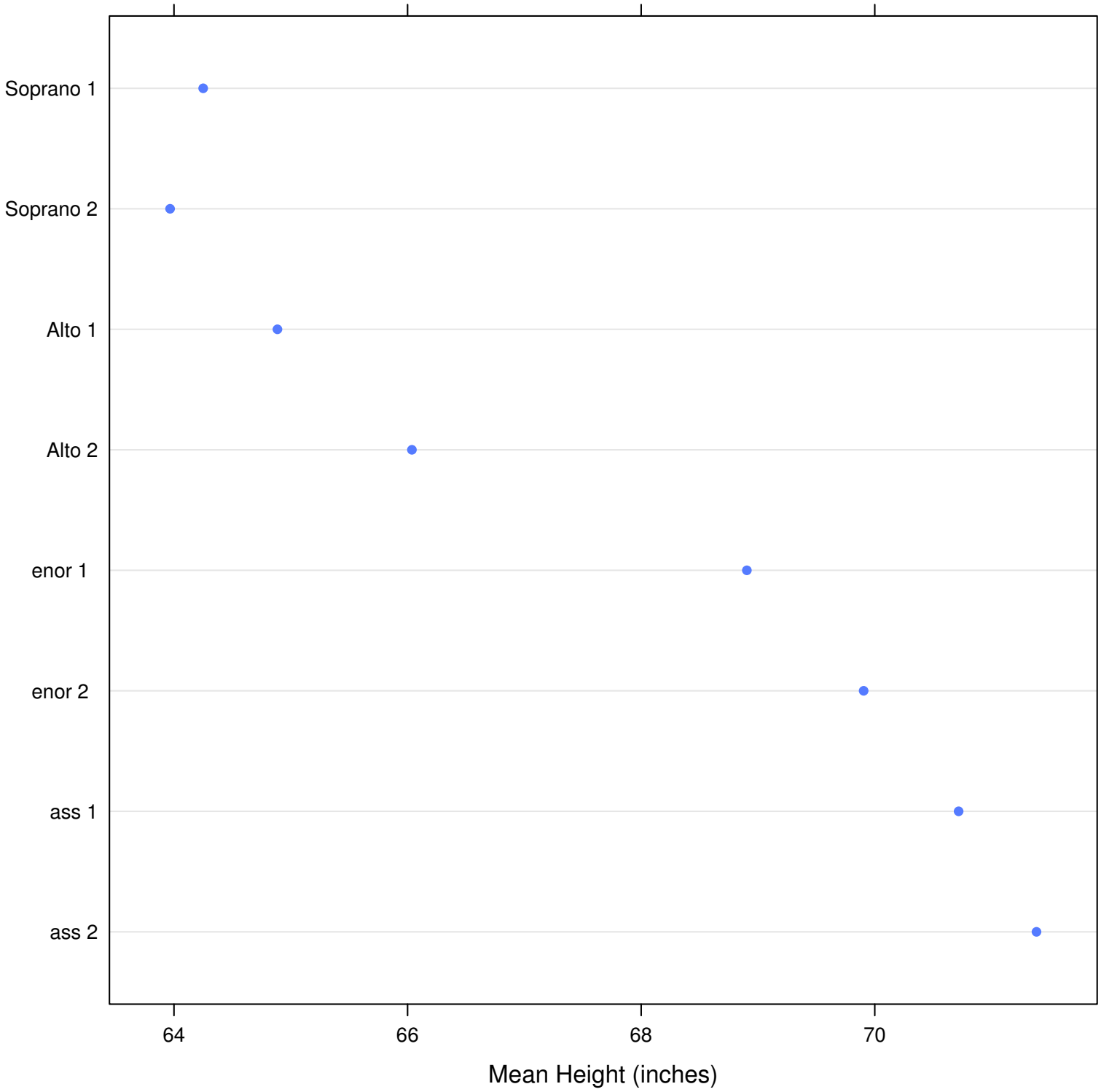
```
qq(voice.part ~ singer$height,
   data = singer,
   subset = voice.part=="Bass 2" | voice.part=="Tenor 1",
   aspect = 1,
   xlab = "Tenor 1 Height (inches)",
   ylab = "Base 2 Height (inches)"
)
```


76
74
72
70
68
66
64
ase 2 Height (inches)
64
66
68
70
72
74
76
enor 1 Height (inches)

dotplot()

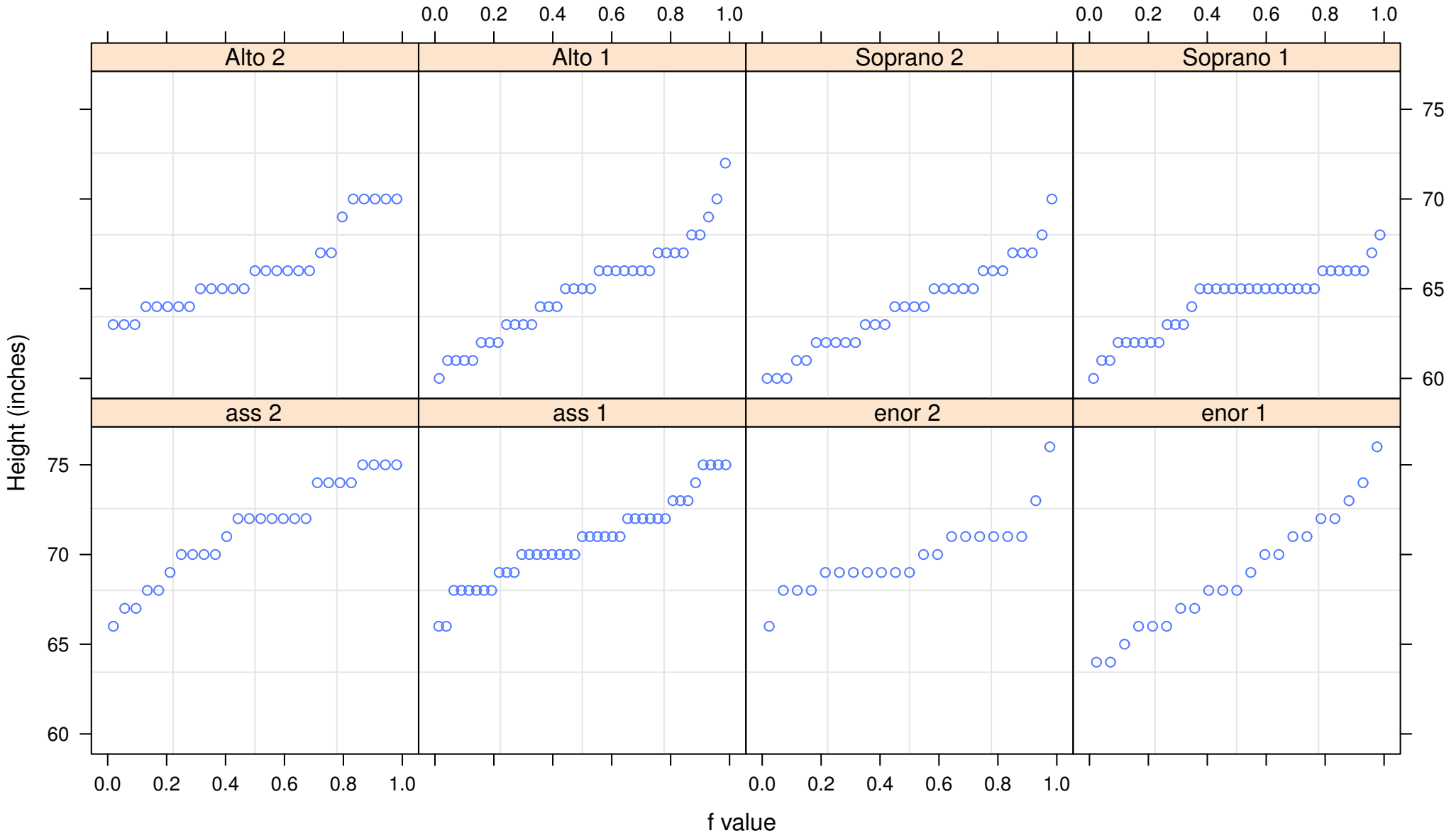
```
part.mean <- tapply(singer$height,
                    singer$voice.part,
                    mean)

dotplot(labels(part.mean) ~ part.mean,
        aspect = 1,
        ylab = "Voice Part",
        xlab = "Mean Height (inches)")
```


Soprano 1
Soprano 2
Alto 1
Alto 2
enor 1
enor 2
ass 1
ass 2
64
66
68
70
Mean Height (inches)

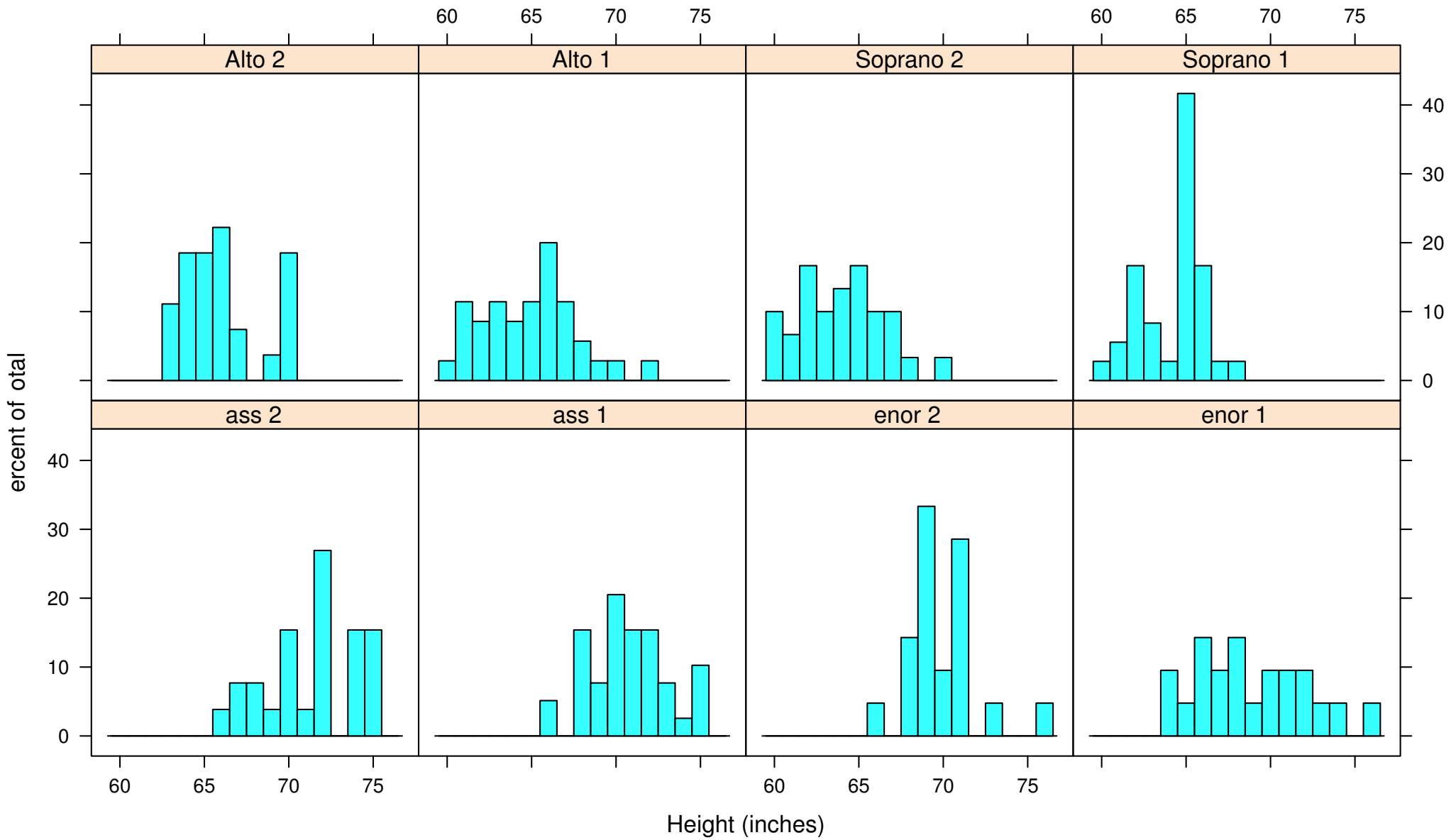
qqmath()

```
qqmath(~height | voice.part,
       distribution = qunif,
       data = singer,
       panel = function(x, ...) {
                panel.grid()
                panel.qqmath(x,...)
       },
       layout = c(4,2),
       aspect = 1,
       xlab = "f-value",
       ylab = "Height (inches)")
```


Alto 2
Alto 1
Soprano 2
Soprano 1
ass 2
ass 1
enor 2
enor 1
Height (inches)
f value
0.0
0.2
0.4
0.6
0.8
1.0
60
65
70
75

histogram()

```
histogram(~height | voice.part,
          data = singer,
          nint = 17,
          endpoints = c(59.5, 76.5),
          layout = c(4, 2),
          aspect = 1,
          xlab = "Height (inches)")
```


Alto 2
Alto 1
Soprano 2
Soprano 1
ass 2
ass 1
enor 2
enor 1
60
65
70
75
0
10
20
30
40
ercent of otal
Height (inches)

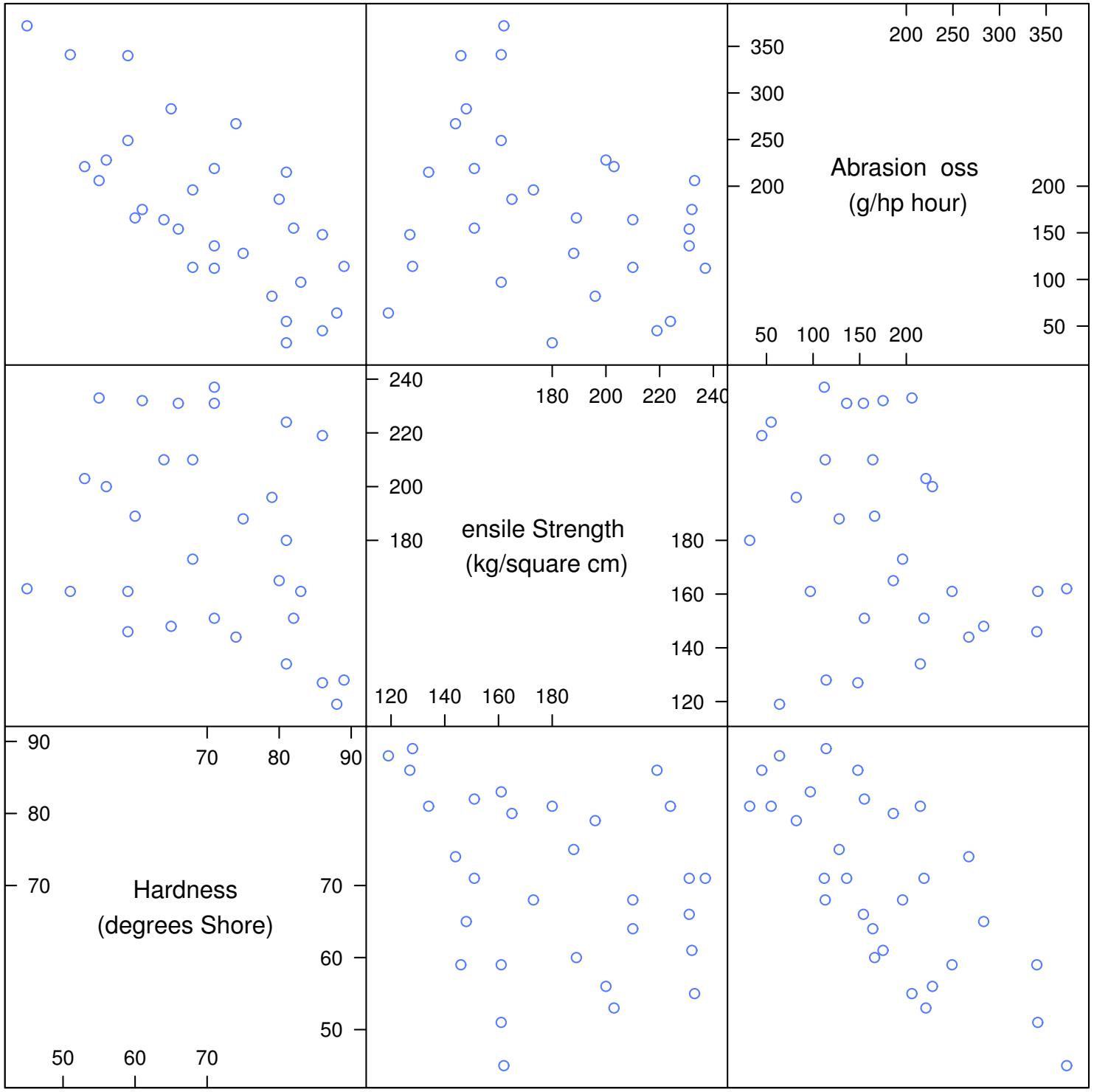
rubber

```
> dim(rubber)
[1] 30  5

> summary(rubber)
    hardness      tensile.strength abrasion.loss
Min.   :45.00   Min.   :119.0     Min.   : 32.0
1st Qu.:60.25   1st Qu.:151.0     1st Qu.:113.2
Median :71.00   Median :176.5     Median :165.0
Mean   :70.27   Mean   :180.5     Mean   :175.4
3rd Qu.:81.00   3rd Qu.:210.0     3rd Qu.:220.5
Max.   :89.00   Max.   :237.0     Max.   :372.0
     ts.low           ts.high
Min.   :-61.00   Min.   : 0.0
1st Qu.:-29.00   1st Qu.: 0.0
Median : -3.50   Median : 0.0
Mean   :-15.63   Mean   :16.1
3rd Qu.:  0.00   3rd Qu.:30.0
Max.   :  0.00   Max.   :57.0
```

splom()

```
splom(~rubber[,1:3],
      varnames = c("Hardness\n(degrees Shore)",
                   "Tensile Strength\n(kg/square cm)",
                   "Abrasion Loss\n(g/hp-hour)"))
```



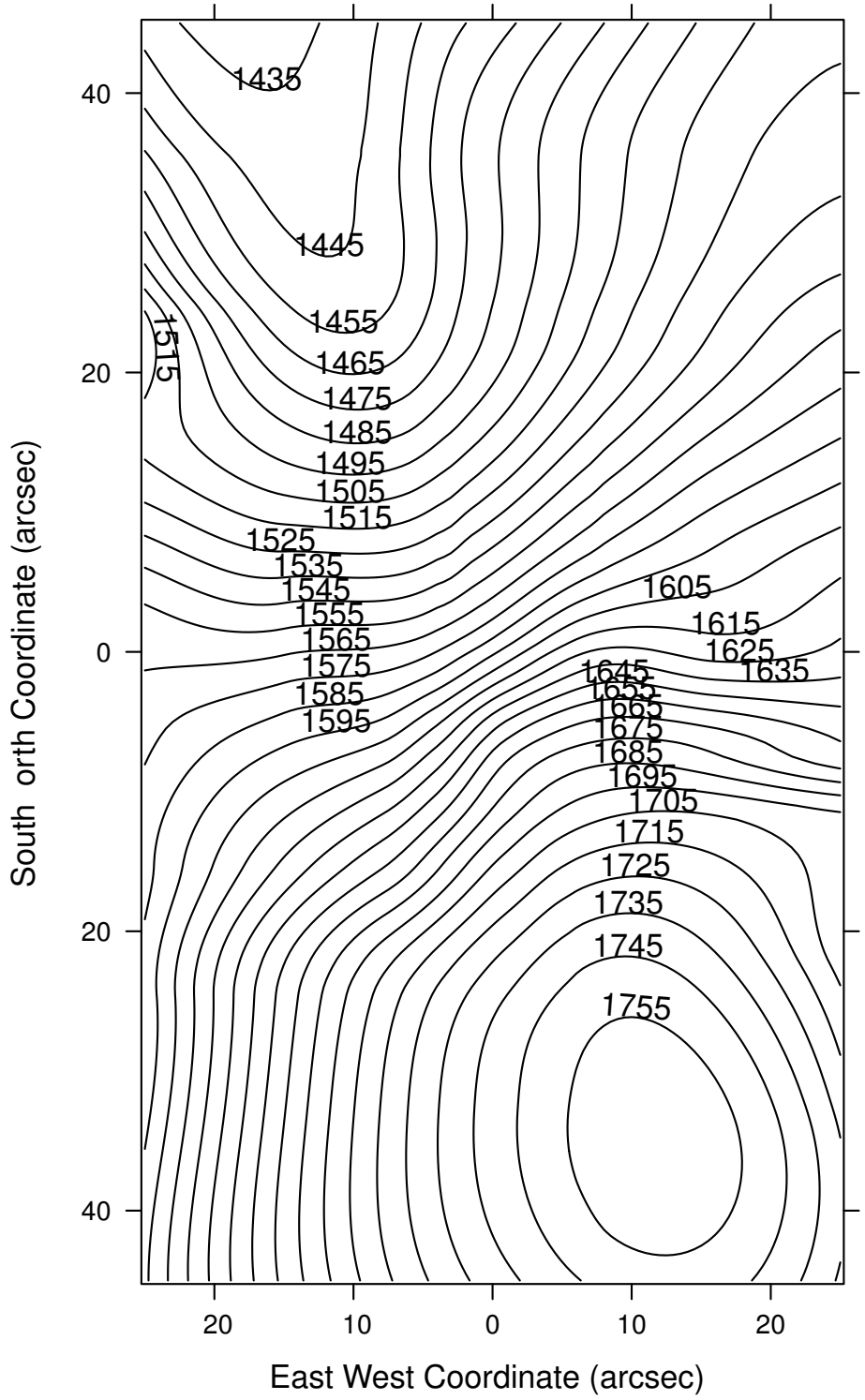
Scatter lot Matrix

contourplot()

```
attach(galaxy)
galaxy.marginal <- list(
east.west = seq(-25, 25,length=101),
        north.south = seq(-45, 45,length=181))
galaxy.grid <- expand.grid(galaxy.marginal)
gal.m <- loess(velocity ~ east.west * north.south,
span = 0.25, degree = 2, normalize = F,
family = "symmetric")
galaxy.fit <- predict(gal.m, galaxy.grid)
x <- range(galaxy.marginal$east.west)
y <- range(galaxy.marginal$north.south)

contourplot(
galaxy.fit ~ galaxy.grid$east.west * galaxy.grid$north.sou
        at = seq(1435, 1755, by = 10),
        aspect = diff(range(y))/diff(range(x)),
        xlab = "East-West Coordinate (arcsec)",
        ylab = "South-North Coordinate (arcsec)")

detach()
```

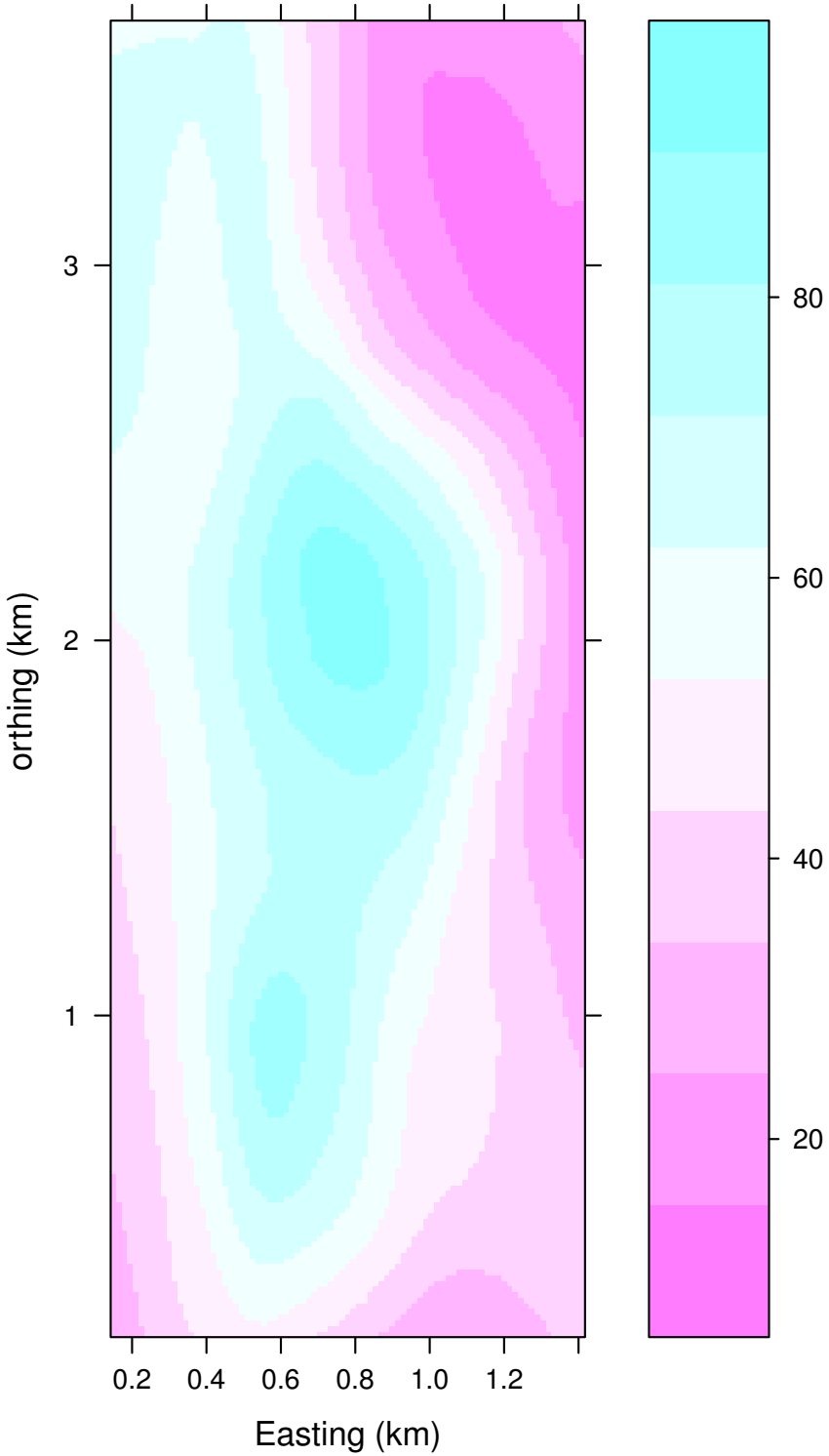

1435
1445
1455
1515
1465
1475
1485
1495
1505
1515
1525
1535
1545
1555
1565
1575
1585
1595
1605
1615
1625
1635
1645
1655
1665
1675
1685
1695
1705
1715
1725
1735
1745
1755
40
20
0
20
40
South orth Coordinate (arcsec)
20
10
0
10
20
East West Coordinate (arcsec)

levelplot()

```
attach(soil)
soi.grid <- expand.grid(
        easting = seq(.15, 1.410, by = .015),
        northing = seq(.150, 3.645, by = .015))
soi.fit <- predict(loess(resistivity~easting*northing,
        span = 0.25, degree = 2), soi.grid)

levelplot(soi.fit ~ soi.grid$easting * soi.grid$northing,
        cuts = 9,
        colorkey = list(
                labels = list(at = c(20, 40, 60, 80)),
                width = 5),
        aspect=diff(range(soi.grid$n))/diff(range(soi.grid
        xlab = "Easting (km)",
        ylab = "Northing (km)")

detach()
```

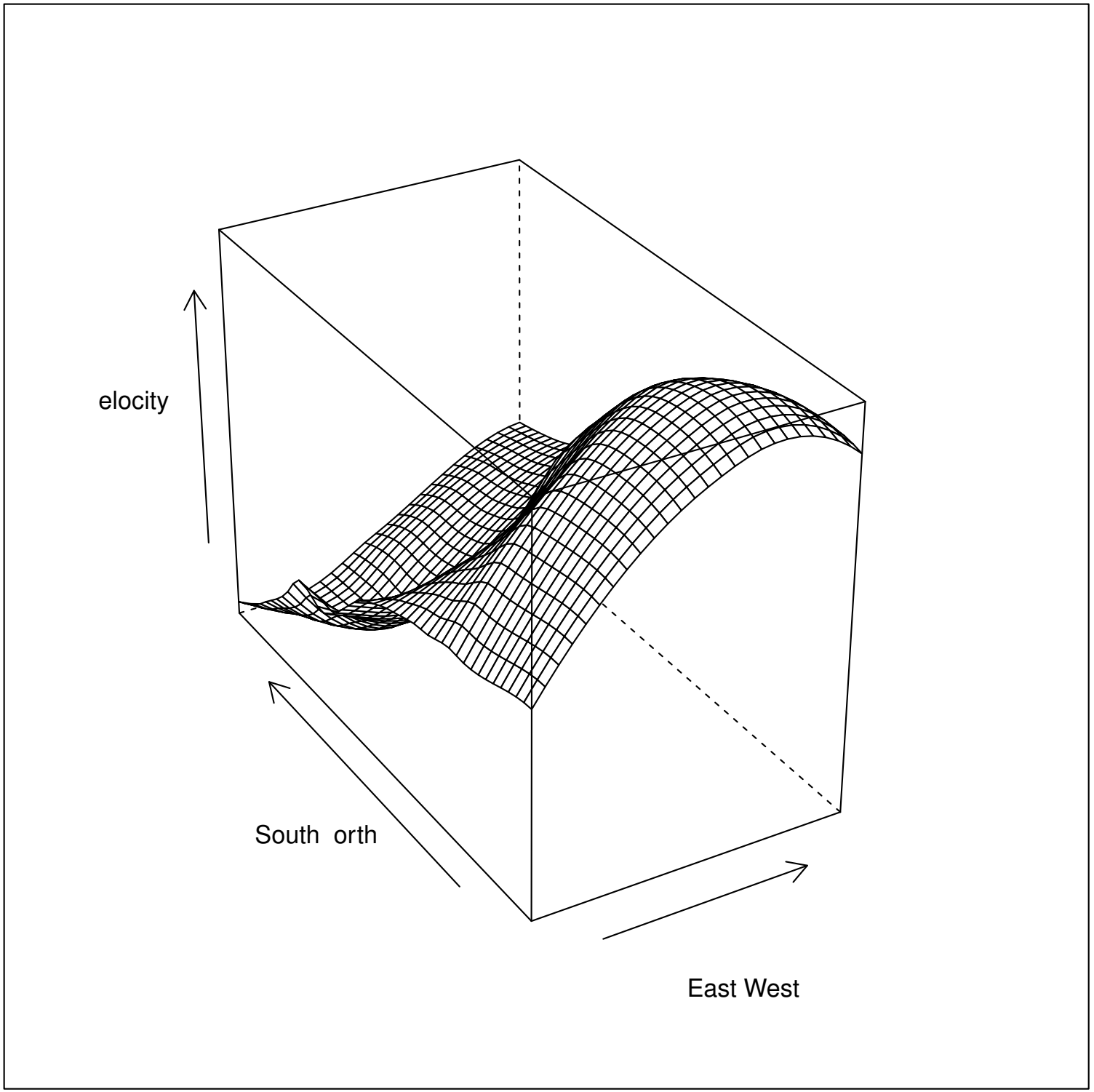

orthing (km)
3
2
1
0.2
0.4
0.6
0.8
1.0
1.2
Easting (km)
80
60
40
20

wireframe()

```
attach(galaxy)
galaxy.marginal <- list(east.west = seq(-25, 25, by = 2),
        north.south = seq(-45, 45, by = 2))
galaxy.grid <- expand.grid(galaxy.marginal)
galaxy.fit <- predict(loess(velocity ~ east.west*north.sou
        span = 0.25, degree = 2, normalize = F,
family = "symmetric"),
        galaxy.grid)
ar <- diff(range(galaxy.grid$north.south))/
        diff(range(galaxy.grid$east.west))

wireframe(galaxy.fit ~ galaxy.grid$east.west*galaxy.grid$n
        screen = list(z = 30, x = -60, y = 0),
        aspect = c(ar, 1.3),
        xlab = "East-West",
        ylab = "South-North",
        zlab = "Velocity")

detach()
```


elocity
South orth
East West

ethanol

```
> dim(ethanol)
[1] 88 3

> summary(ethanol)
      NOx                C                E         
 Min.   :0.370   Min.   : 7.500   Min.   :0.5350  
 1st Qu.:0.953   1st Qu.: 8.625   1st Qu.:0.7618  
 Median :1.754   Median :12.000   Median :0.9320  
 Mean   :1.957   Mean   :12.034   Mean   :0.9265  
 3rd Qu.:3.003   3rd Qu.:15.000   3rd Qu.:1.1098  
 Max.   :4.028   Max.   :18.000   Max.   :1.2320  
```

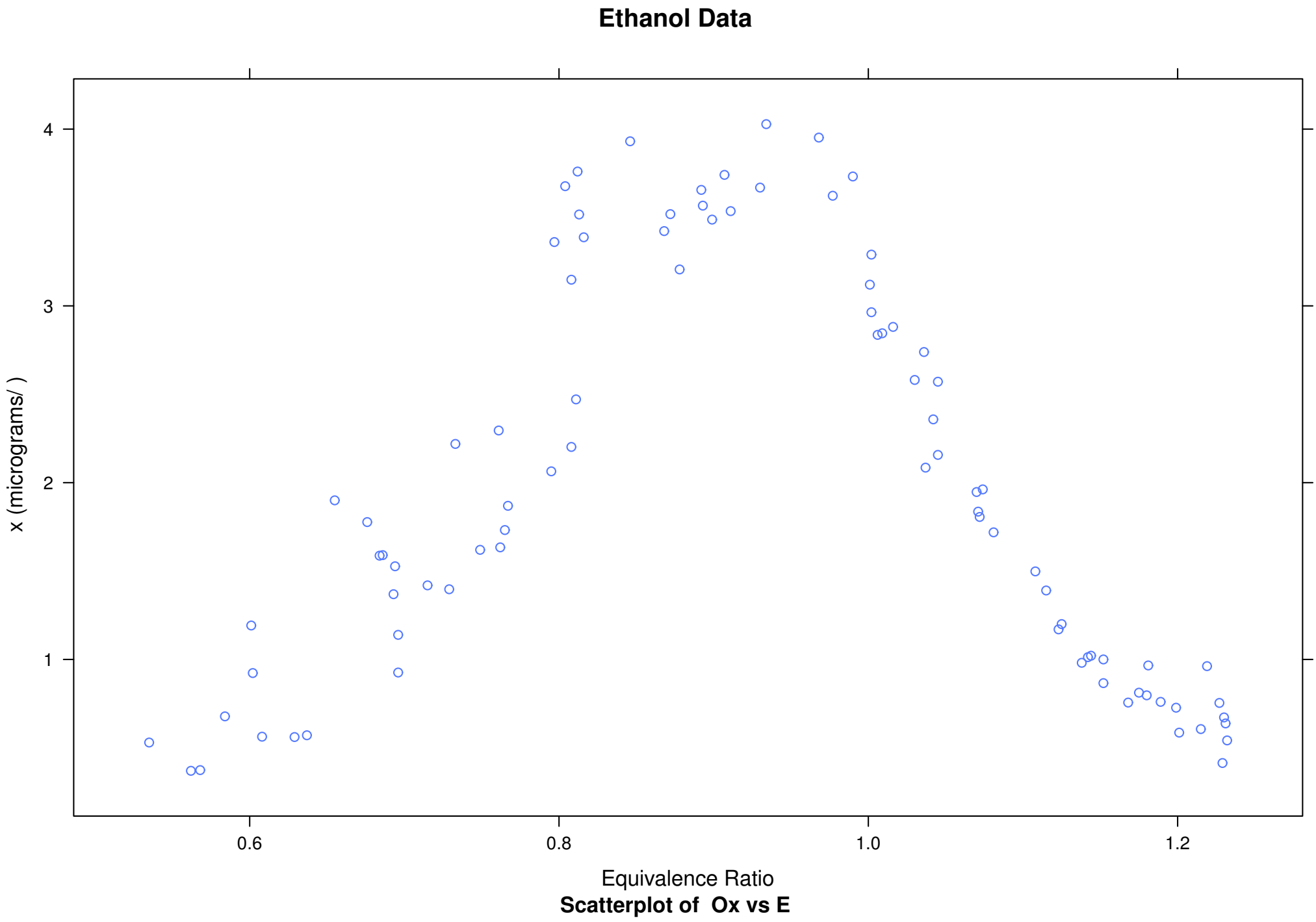
Captions and Labels

Most general display functions allow four standard labels:

`main`, `sub`, `xlab`, `ylab`

Apart from their positions, they are treated identically

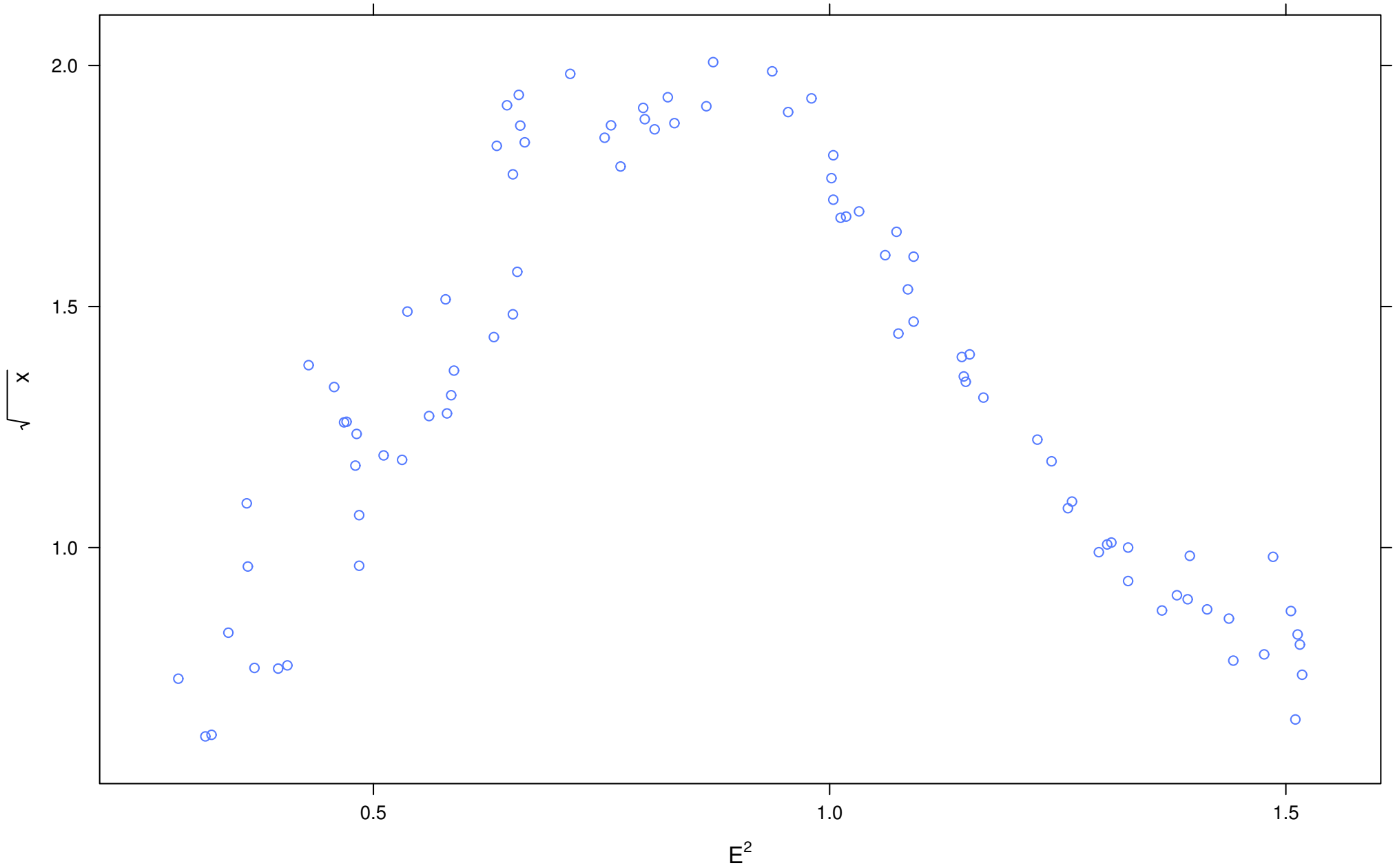
```
xyplot(NOx ~ E,
       data = ethanol,
       aspect = 0.6,
       main = "Ethanol Data",
       sub  = "Scatterplot of NOx vs E",
       xlab = "Equivalence Ratio",
       ylab = "NOx (micrograms/J)"
       )
```


Ethanol Data
x (micrograms/)
4
3
2
1
0.6
0.8
1.0
1.2
Equivalence Ratio
Scatterplot of Ox vs E

Captions and Labels: Mathematical Expressions

```
xyplot(sqrt(NOx) ~ E^2,
       data = ethanol,
       aspect = 0.6,
       xlab = expression(E^2),
       ylab = expression(sqrt(NOx))
       )
```

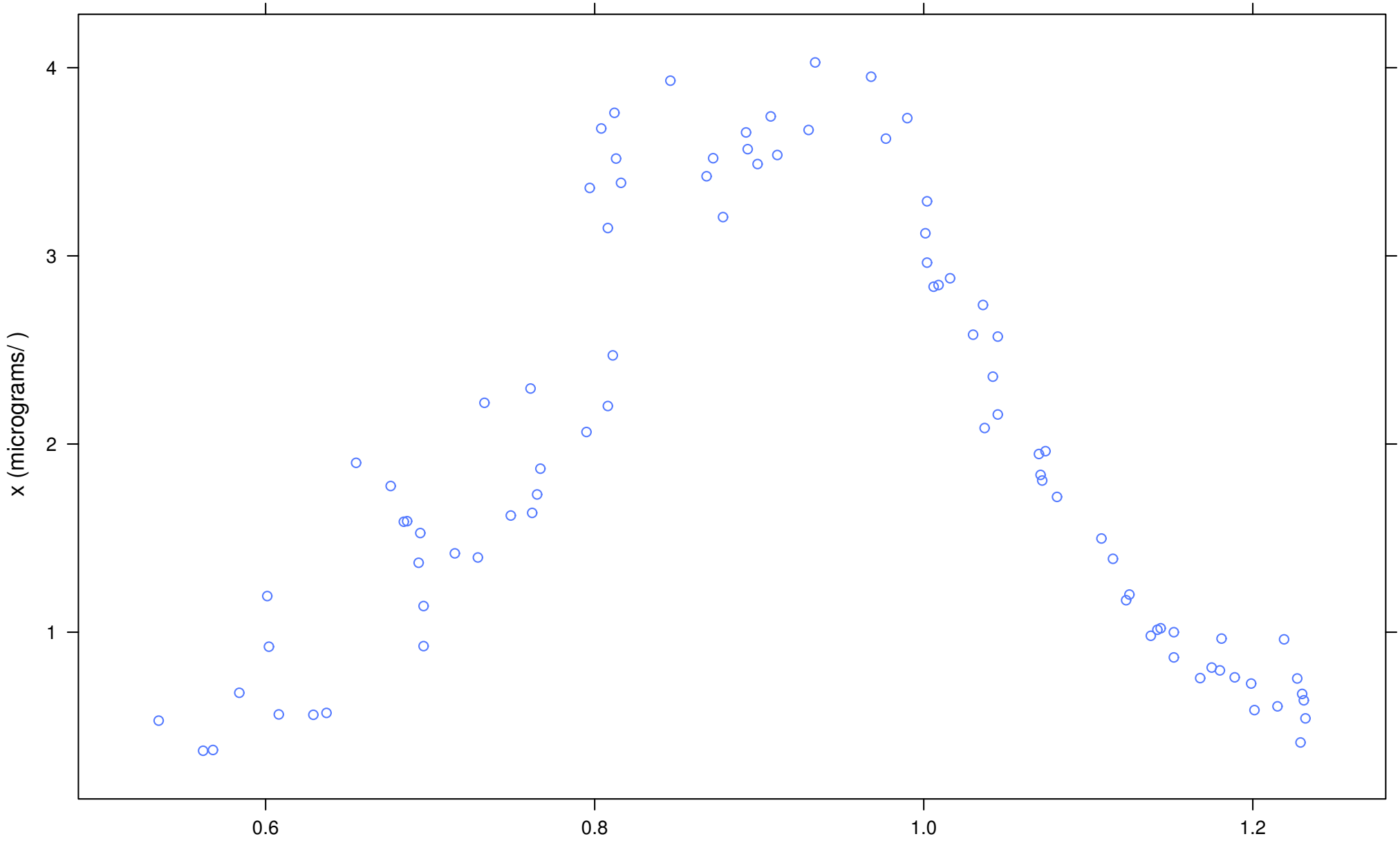
See `?plotmath` for more mathematical expressions


2.0
1.5
1.0
0.5
1.0
1.5
E^2
$\sqrt{x}$

Captions and Labels: Other Graphical Parameters

```
xyplot(NOx ~ E,
       data = ethanol,
       aspect = 0.6,
       xlab = list(
                   label = "Equivalence Ratio",
                   col = "red",
                   cex = 3,
                   font = "italic"
                  ),
       ylab = "NOx (micrograms/J)",
       )
```

See `?gpar` for a full list


x (micrograms/)
4
3
2
1
0.6
0.8
1.0
1.2
Equivalence Ratio

col Colour for lines and borders.

fill Colour for filling rectangles, polygons, ...

alpha Alpha channel for transparency

lty Line type

lwd: Line width

lex: Multiplier applied to line width

lineend: Line end style (round, butt, square)

linejoin: Line join style (round, mitre, bevel)

linemitre: Line mitre limit (number greater than 1)

fontsize: The size of text (in points)

cex: Multiplier applied to fontsize

fontfamily: The font family

fontface: The font face (bold, italic, ...)

lineheight: The height of a line as a multiple of the size of text

font: Font face (alias for fontface; for backward compatibility)

Scales: dataset `slope`

```
> dim(slope)
[1] 44  4

> slope[1:5, ]
      error percent distance resolution
1 18.90625      50 37.40523   5.010459
2 16.33333      55 37.15230   4.504590
3 19.43750      60 36.89979   3.999597
4 17.44791      65 36.64780   3.495549
5 17.03646      70 36.39627   2.992530

> summary(slope)
     error          percent         distance        resolution
Min.   : 3.104   Min.   : 50   Min.   : 0.265   Min.   : 0.00
1st Qu.: 6.262   1st Qu.: 60   1st Qu.: 9.875   1st Qu.: 1.95
Median : 7.984   Median : 75   Median :21.368   Median : 4.49
Mean   : 9.362   Mean   : 75   Mean   :20.685   Mean   : 5.87
3rd Qu.:11.962   3rd Qu.: 90   3rd Qu.:31.281   3rd Qu.: 8.66
Max.   :19.438   Max.   :100   Max.   :37.405   Max.   :19.47
```

Scales: relation="same"

Default of relation is "same".

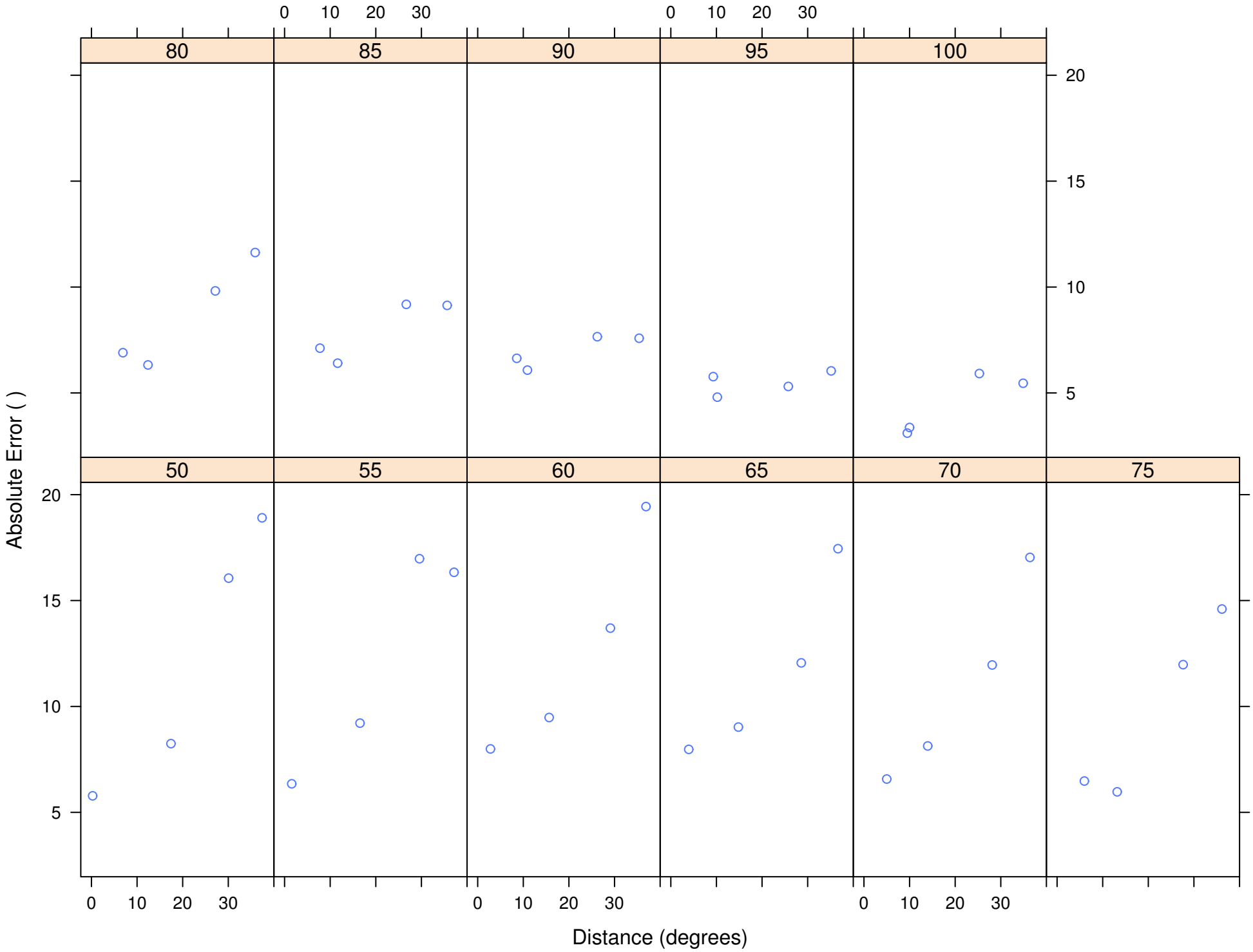
```
xyplot(error ~ distance | as.factor(percent),
       data = slope,
       layout = c(6,2),
       xlab="Distance (degrees)",
       ylab="Absolute Error (%)")
```

or equivalently,

```
xyplot(error ~ distance | as.factor(percent),
       data = slope,
       layout = c(6,2),
       scales = list(y="same"),
       xlab="Distance (degrees)",
       ylab="Absolute Error (%)")
```

or equivalently,

```
xyplot(error ~ distance | as.factor(percent),
       data = slope,
       layout = c(6,2),
       scales = list(y=list(relation="same")),
       xlab="Distance (degrees)",
       ylab="Absolute Error (%)")
```

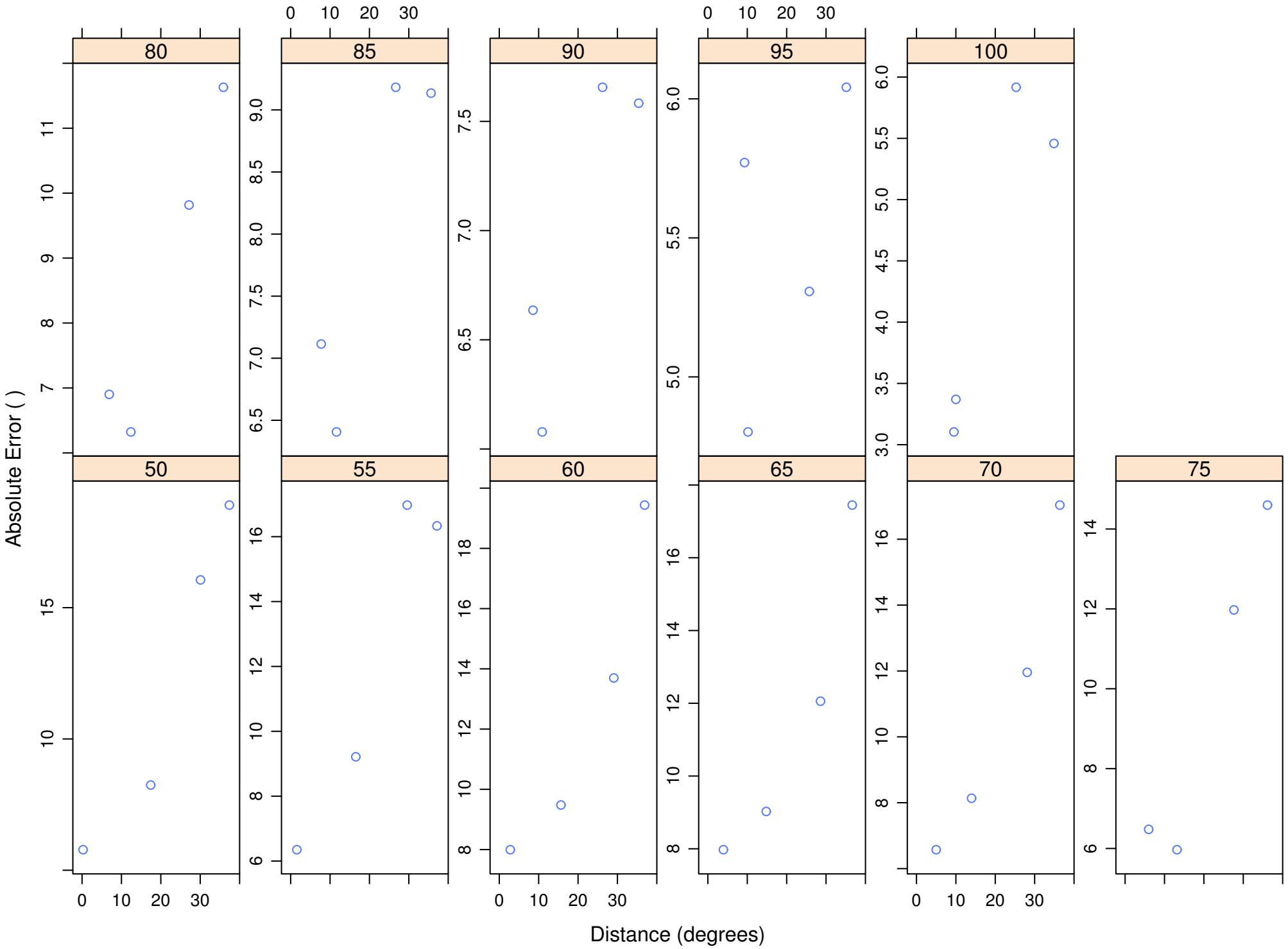

80
85
90
95
100
50
55
60
65
70
75
0
10
20
30
5
10
15
20
Absolute Error ( )
Distance (degrees)

Scales: relation="free"

```
xyplot(error ~ distance | as.factor(percent),
        data = slope,
        layout = c(6,2),
        scales = list(y="free"),
        xlab="Distance (degrees)",
        ylab="Absolute Error (%)")
```

or equivalently,

```
xyplot(error ~ distance | as.factor(percent),
        data = slope,
        layout = c(6,2),
        scales = list(y=list(relation="free")),
        xlab="Distance (degrees)",
        ylab="Absolute Error (%)")
```

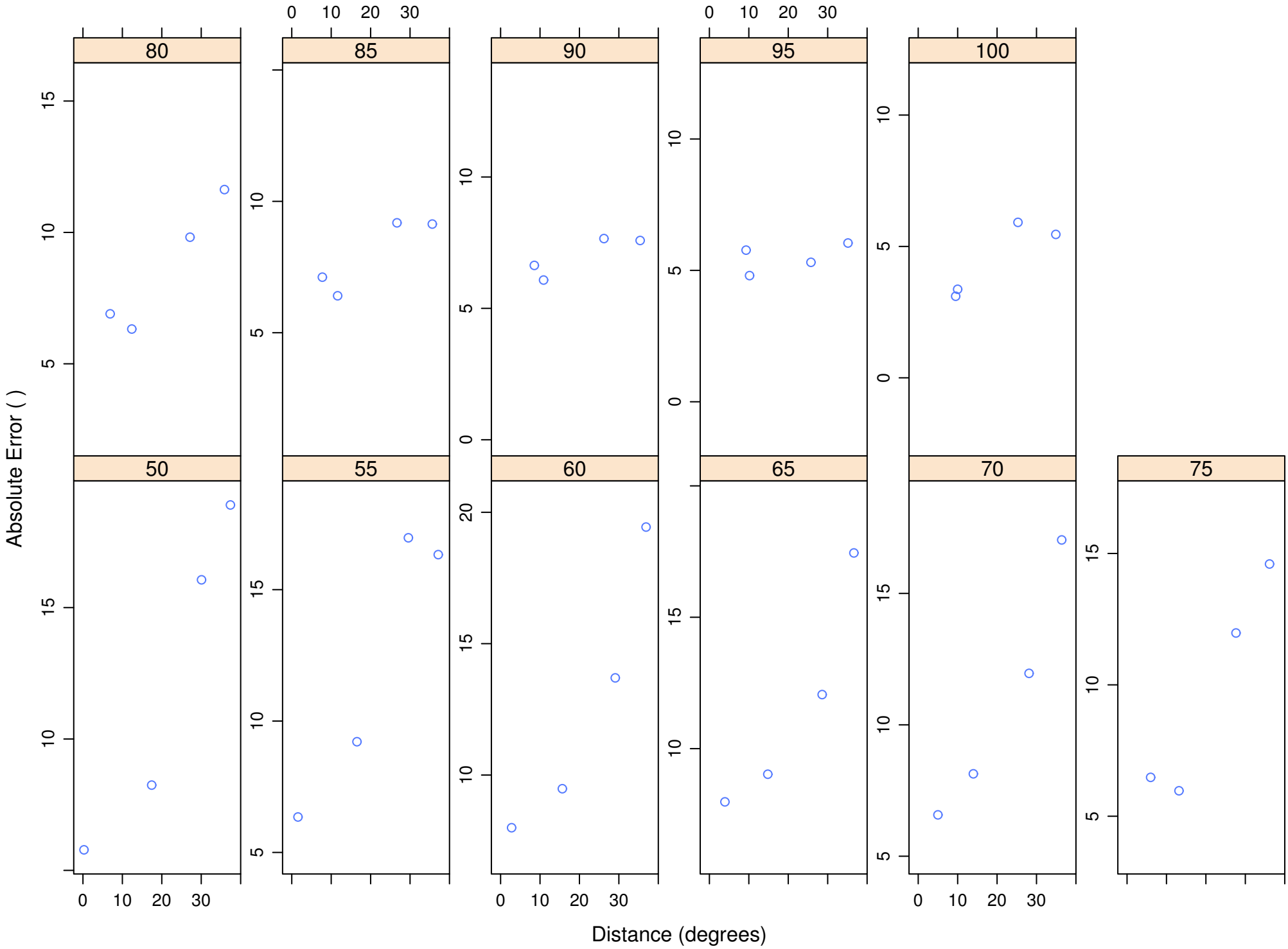

80
85
90
95
100
50
55
60
65
70
75
Absolute Error ( )
Distance (degrees)

Scales: relation="sliced"

```
xyplot(error ~ distance | as.factor(percent),
       data = slope,
       layout = c(6,2),
       scales = list(y="sliced"),
       xlab="Distance (degrees)",
       ylab="Absolute Error (%)")
```

or equivalently,

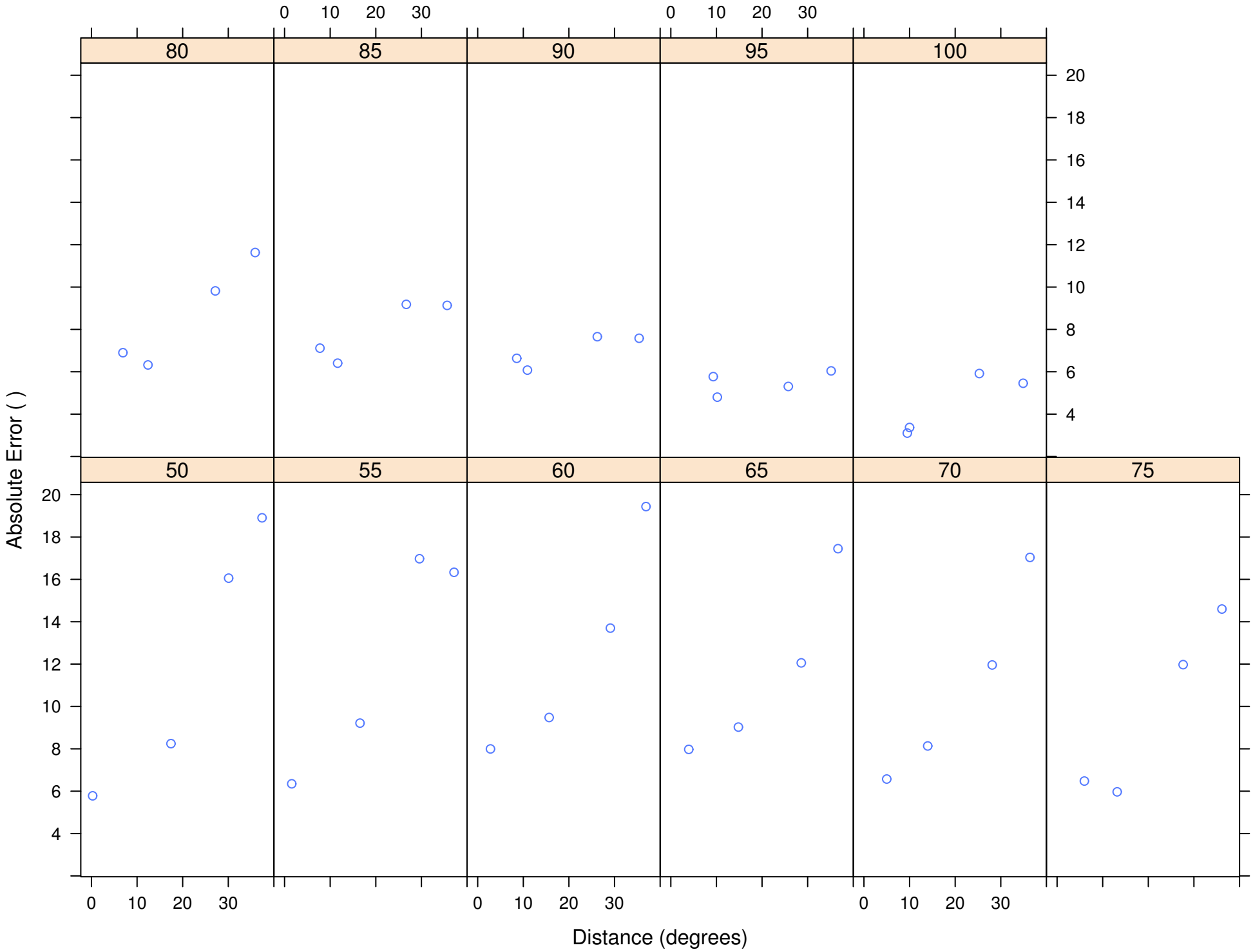
```
xyplot(error ~ distance | as.factor(percent),
       data = slope,
       layout = c(6,2),
       scales = list(y=list(relation="sliced")),
       xlab="Distance (degrees)",
       ylab="Absolute Error (%)")
```


80
85
90
95
100
50
55
60
65
70
75
Absolute Error ( )
Distance (degrees)

Scales: `tick.number`

`tick.number` gives the ”suggested” number of intervals between ticks.

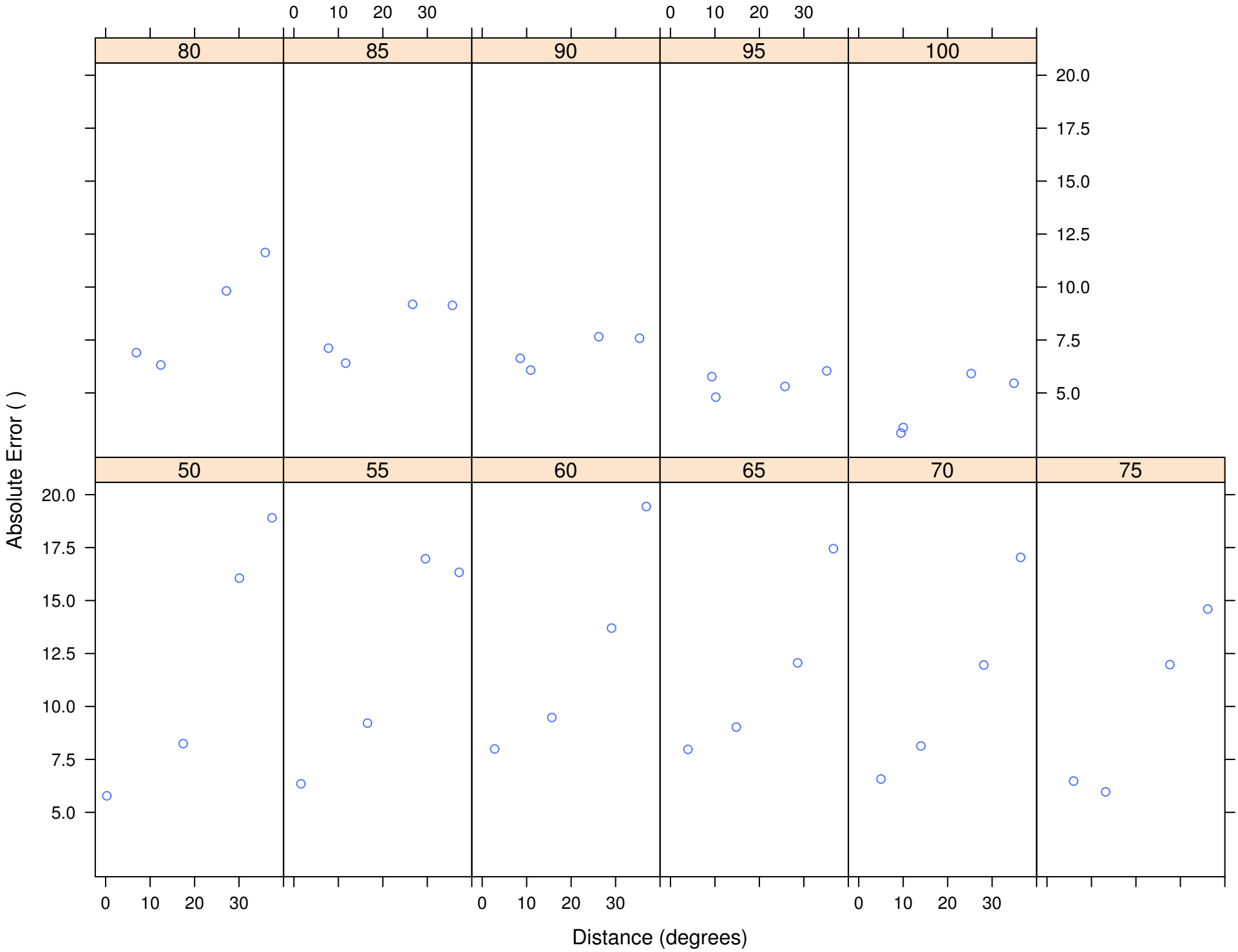
```
xyplot(error ~ distance | as.factor(percent),
         data = slope,
         layout = c(6,2),
         scales = list(y=list(tick.number=8)),
         xlab="Distance (degrees)",
         ylab="Absolute Error (%)")
```


80
85
90
95
100
50
55
60
65
70
75
Absolute Error ( )
Distance (degrees)
0
10
20
30
4
6
8
10
12
14
16
18
20

Scales: at

`at` gives the location of tick marks along the axis.

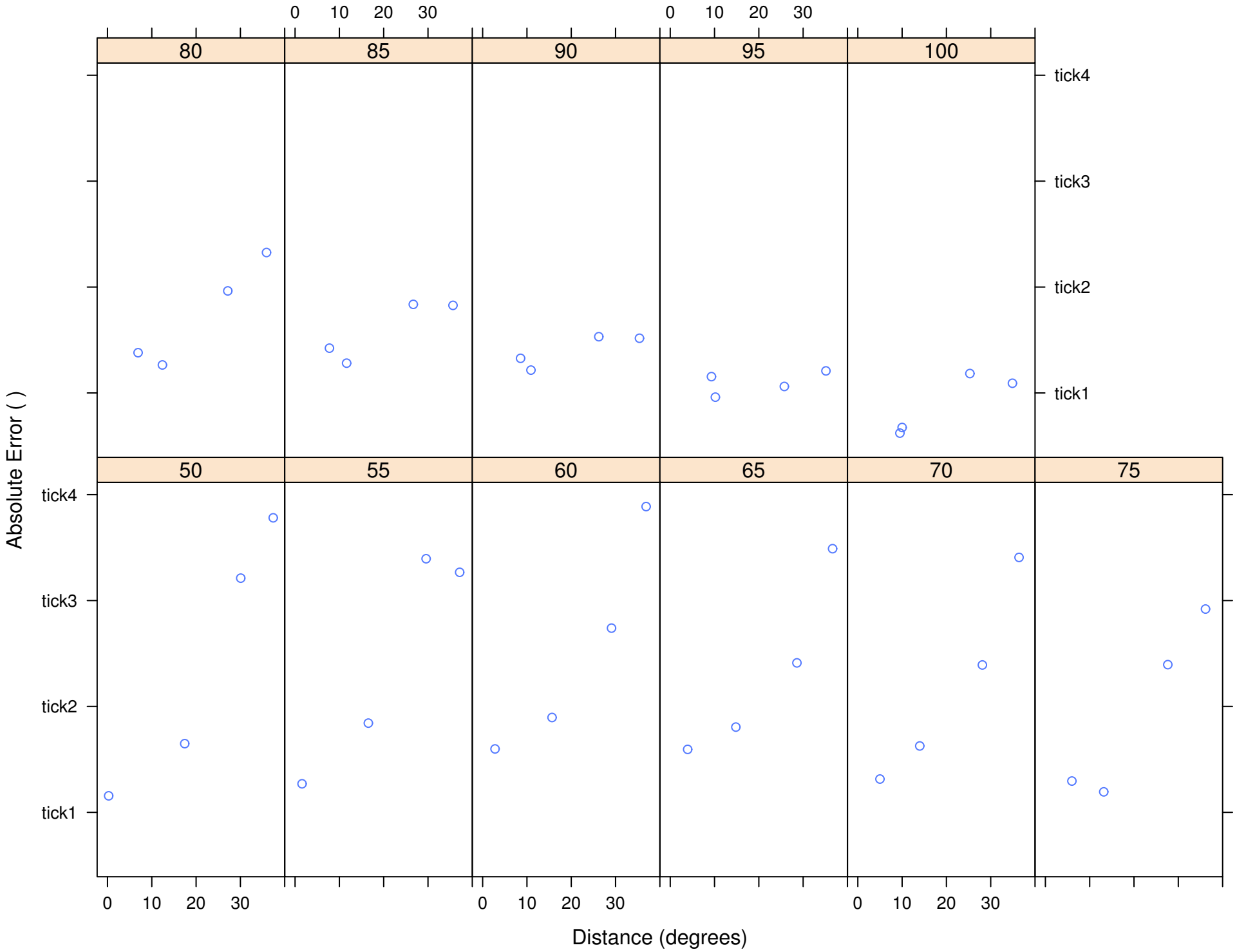
```
xyplot(error ~ distance | as.factor(percent),
         data = slope,
         layout = c(6,2),
         scales = list(
                   y=list(at=c(5,7.5,10,12.5,15,17.5,20))),
         xlab="Distance (degrees)",
         ylab="Absolute Error (%)")
```


80
85
90
95
100
50
55
60
65
70
75
0
10
20
30
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Absolute Error ( )
Distance (degrees)

Scales: `labels`

`labels` gives vector of labels to go along with `at`.

```
xyplot(error ~ distance | as.factor(percent),
         data = slope,
         layout = c(6,2),
         scales = list(
             y=list(at=c(5,10,15,20),
                 labels=c("tick1","tick2","tick3","tick4")
                 )
             ),
         xlab="Distance (degrees)",
         ylab="Absolute Error (%)")
```


80
85
90
95
100
50
55
60
65
70
75
0
10
20
30
tick1
tick2
tick3
tick4
Absolute Error ( )
Distance (degrees)

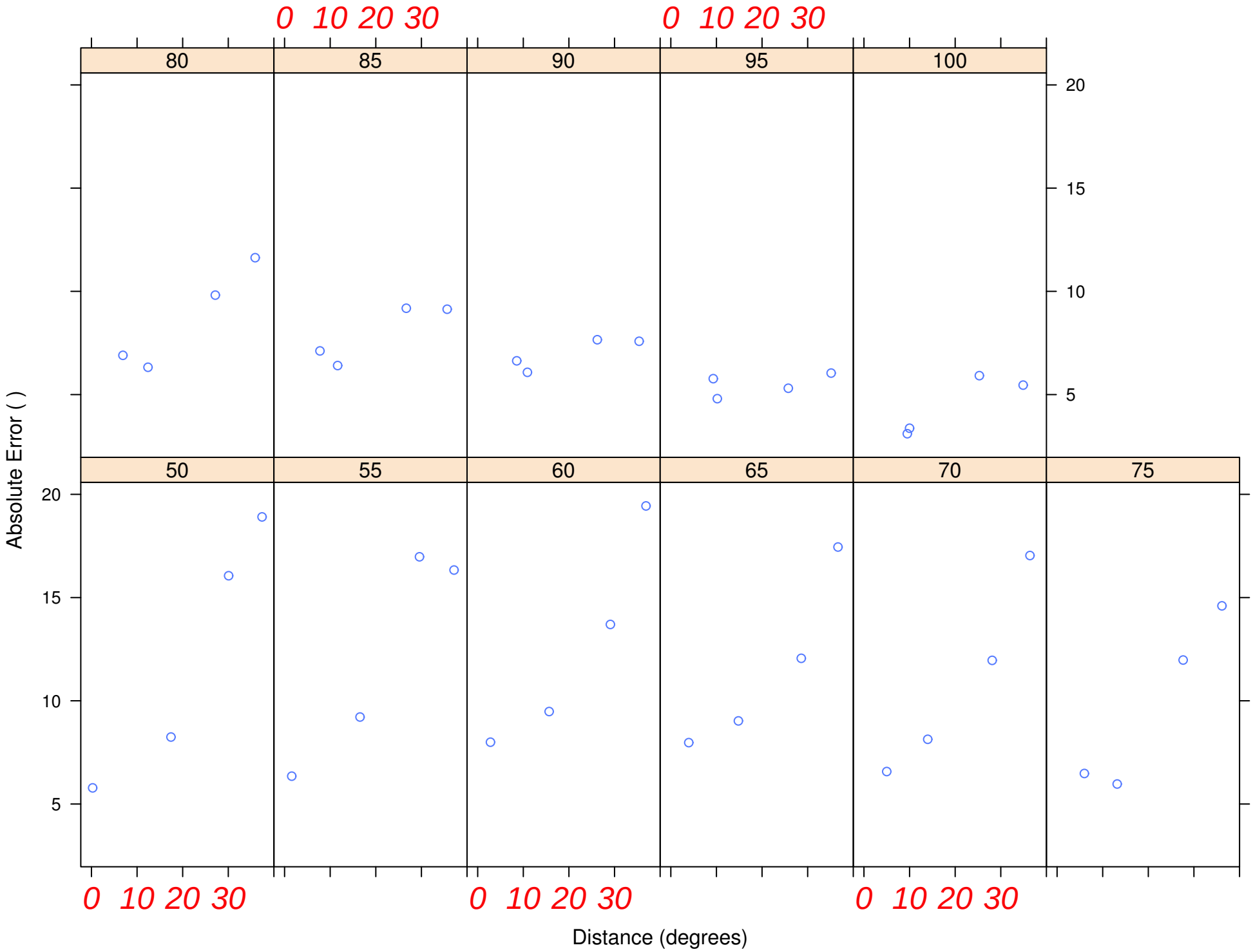
Scales: `cex, font, col`

`cex` controls character sizes for axis labels.

`font` specifies the font for axis labels.

`col` specifies the color for axis labels.

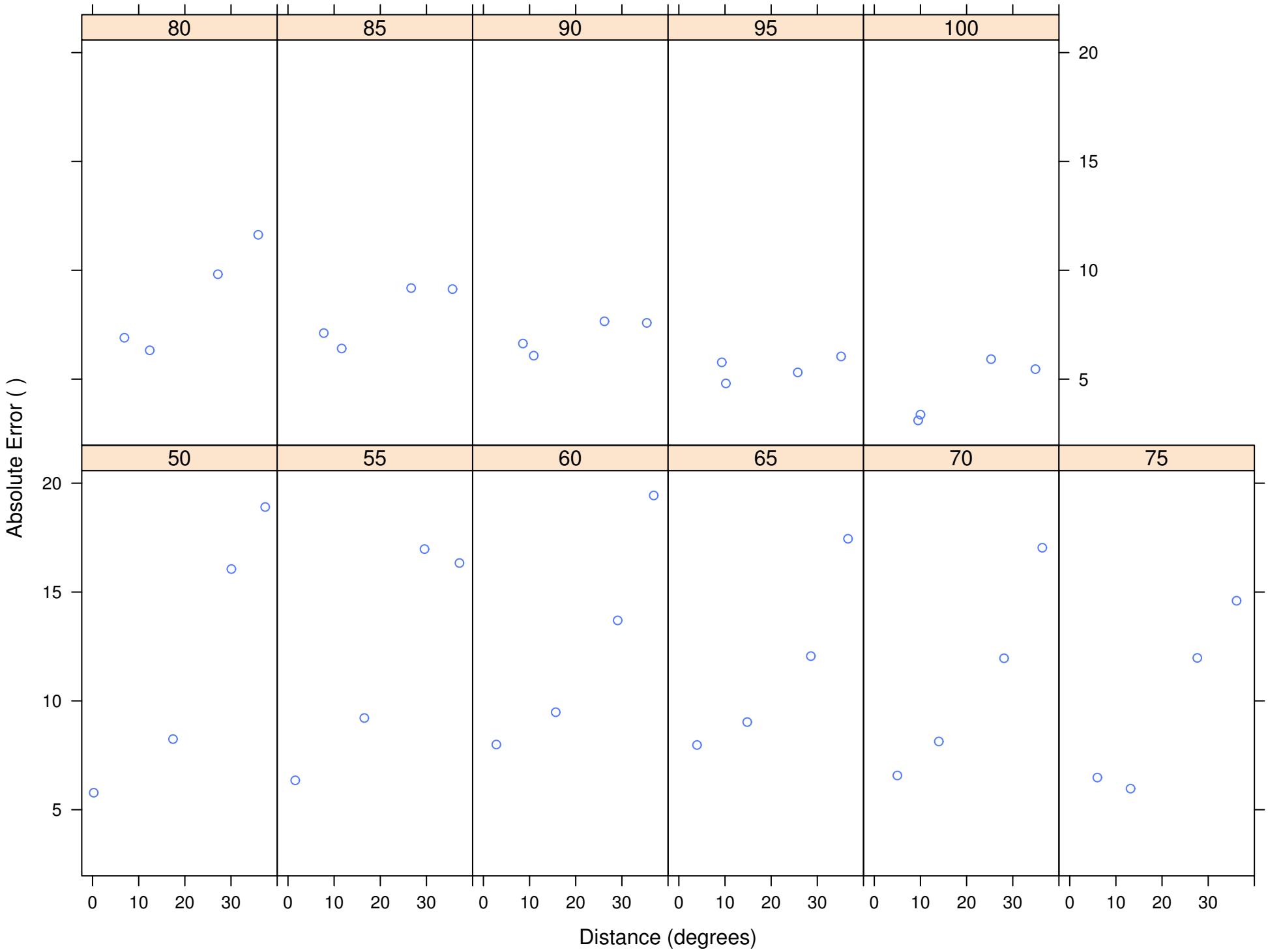
```
xyplot(error ~ distance | as.factor(percent),
       data = slope,
       layout = c(6,2),
       scales = list(
                x = list(
                         cex = 1.5,
                         font = "italic",
                         col = "red"
                         )
                ),
       xlab="Distance (degrees)",
       ylab="Absolute Error (%)")
```


80
85
90
95
100
50
55
60
65
70
75
0 10 20 30
Absolute Error ( )
Distance (degrees)
5
10
15
20

Scales: `alternating`

`alternating` specifies whether axis labels should alternate from one side of the group of panels to the other.

```
xyplot(error ~ distance | as.factor(percent),
        data = slope,
        layout = c(6,2),
        scales = list(x=list(alternating=FALSE)),
        xlab="Distance (degrees)",
        ylab="Absolute Error (%)")
```


80
85
90
95
100
50
55
60
65
70
75
Absolute Error ( )
Distance (degrees)
0
10
20
30
5
10
15
20

Scales: `alternating`

`alternating` can also be a vector (replicated to be as long as the number of rows or columns per page) consisting of the following numbers

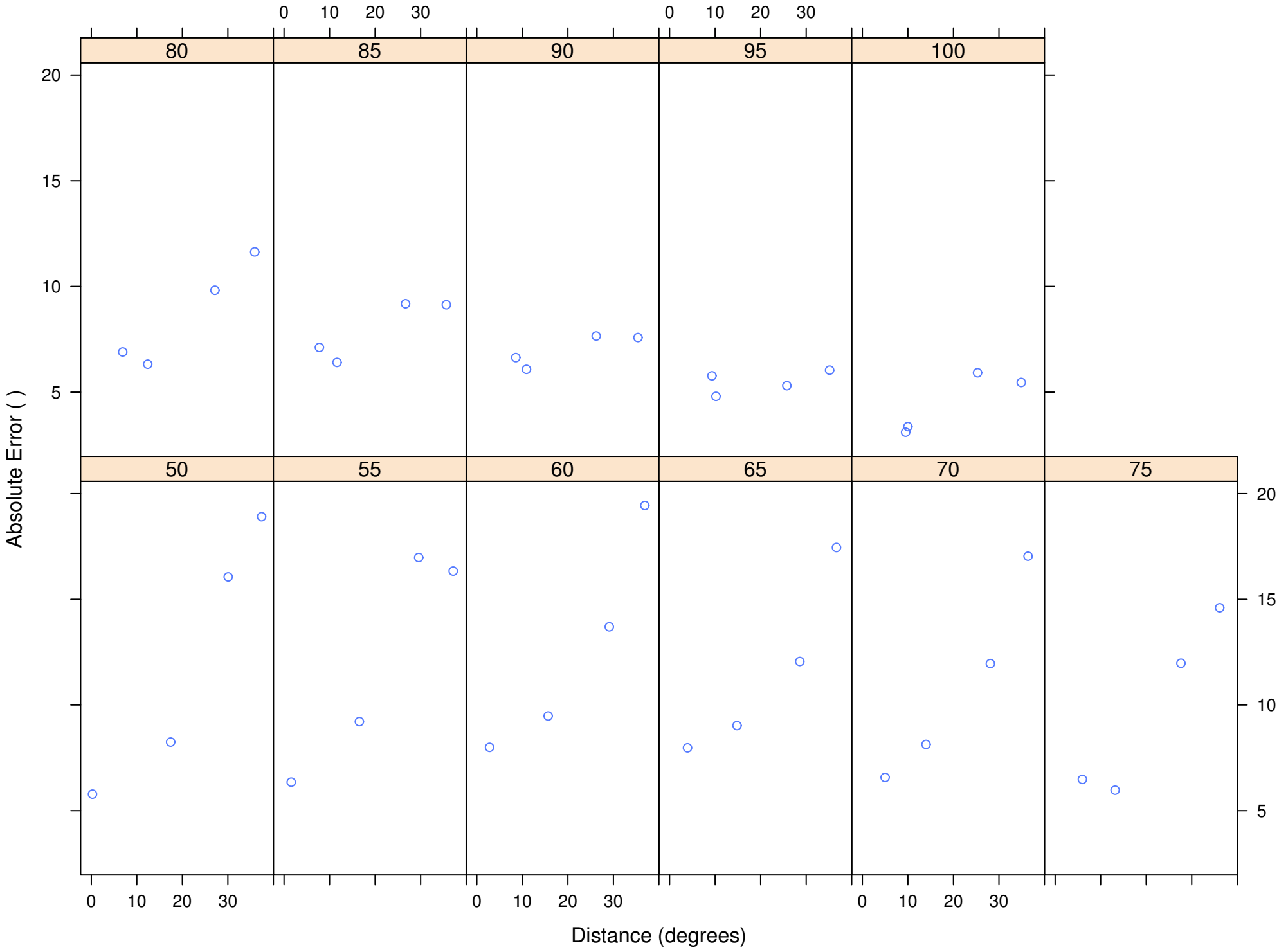
0: do not draw tick labels

1: bottom/left

2: top/right

3: both.

```
xyplot(error ~ distance | as.factor(percent),
                  data = slope,
                  layout = c(6,2),
                  scales = list(x=list(alternating=c(1,2)),
                                y=list(alternating=c(2,1))),
                  xlab="Distance (degrees)",
                  ylab="Absolute Error (%)")
```


80
85
90
95
100
50
55
60
65
70
75
0
10
20
30
5
10
15
20
Absolute Error ( )
Distance (degrees)

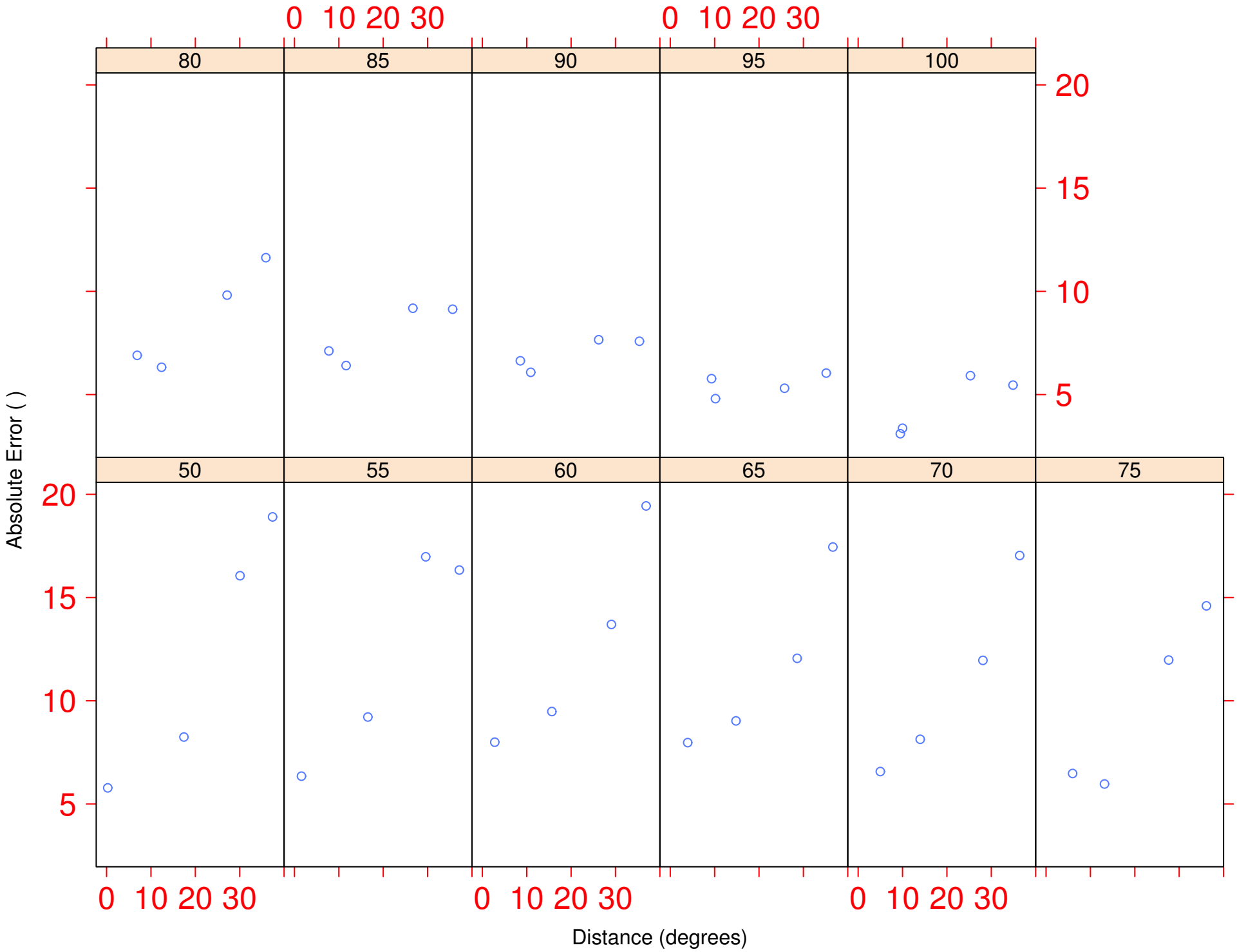
Scales

Without specifying x or y-axis, the arguments in `scales` are applied to both x and y-axis.

```
xyplot(error ~ distance | as.factor(percent),
               data = slope,
               layout = c(6,2),
               scales = list(cex=1.5, col="red"),
               xlab="Distance (degrees)",
               ylab="Absolute Error (%)")
```

or equivalently,

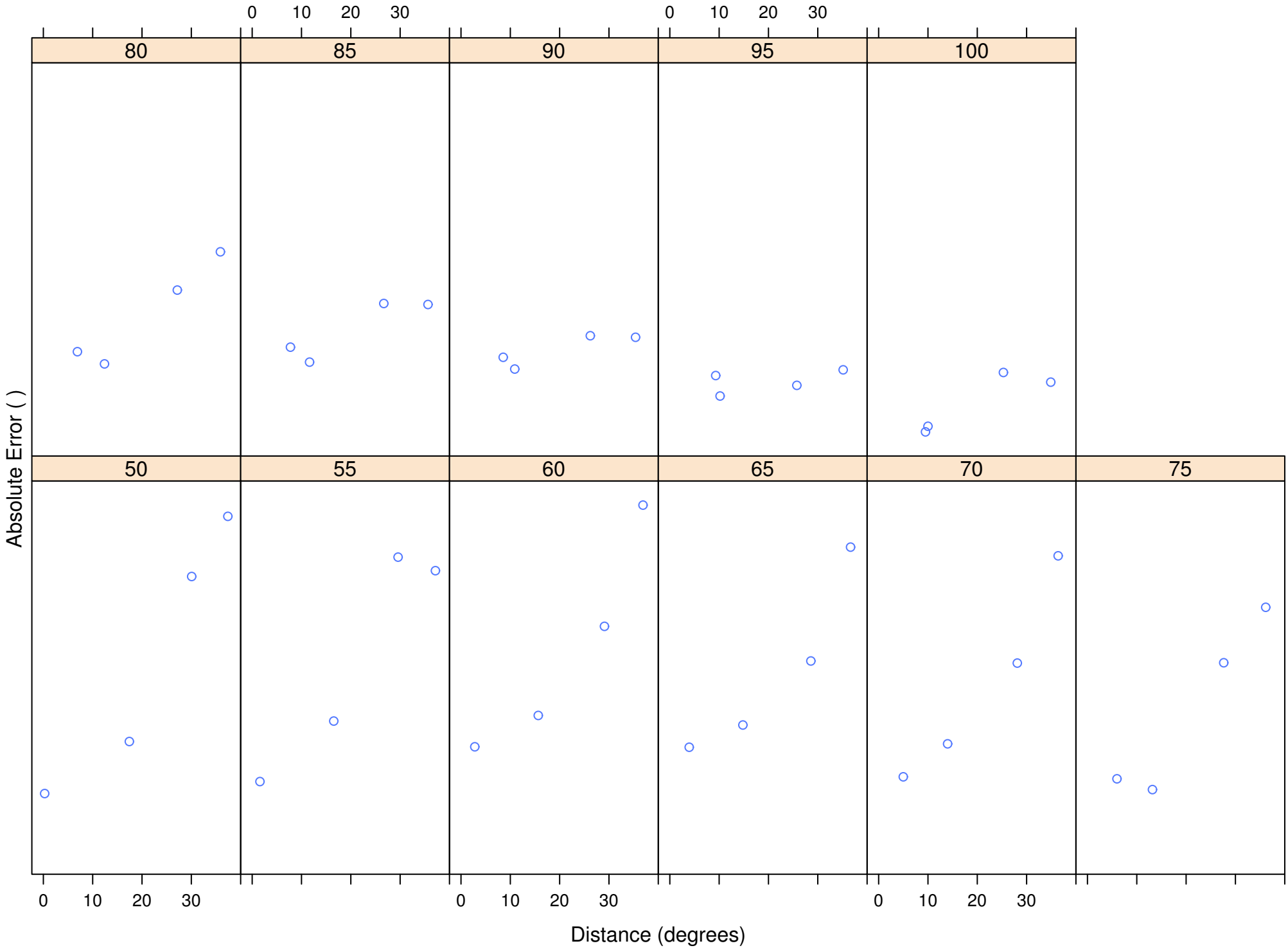
```
xyplot(error ~ distance | as.factor(percent),
               data = slope,
               layout = c(6,2),
               scales = list(
                        x = list(cex=1.5, col="red"),
                        y = list(cex=1.5, col="red")
                        )
               xlab="Distance (degrees)",
               ylab="Absolute Error (%)")
```


80
85
90
95
100
50
55
60
65
70
75
0 10 20 30
5
10
15
20
Absolute Error ( )
Distance (degrees)

Scales: draw

`draw` specifies whether axis labels should be drawn.

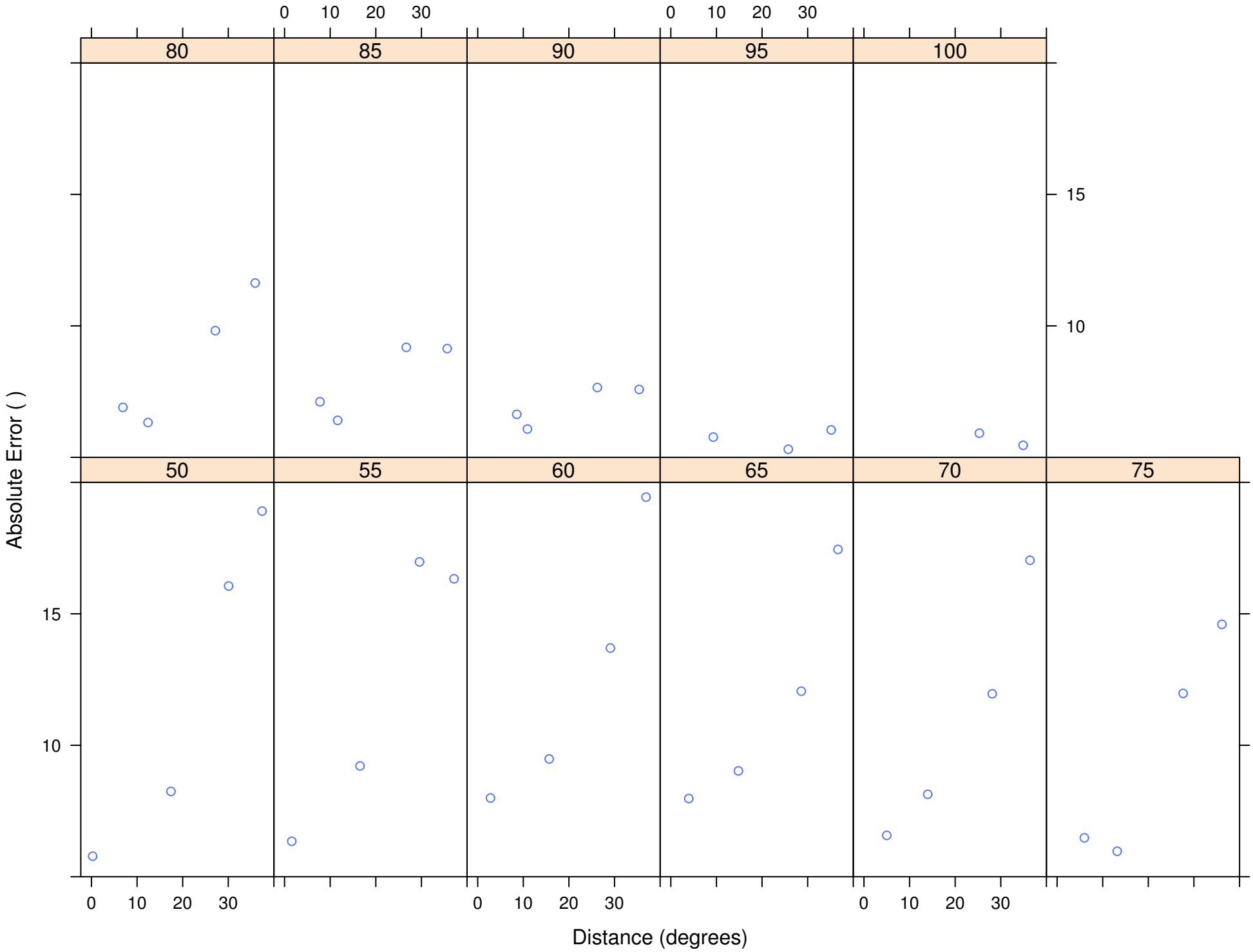
```
xyplot(error ~ distance | as.factor(percent),
                  data = slope,
                  layout = c(6,2),
                  scales = list(y=list(draw=FALSE)),
                  xlab="Distance (degrees)",
                  ylab="Absolute Error (%)")
```


80
85
90
95
100
50
55
60
65
70
75
0
10
20
30
Absolute Error ( )
Distance (degrees)

Scales: `limits`

`limits` same as xlim and ylim, normally a numeric vector of length 2 giving left and right limits for the axis.

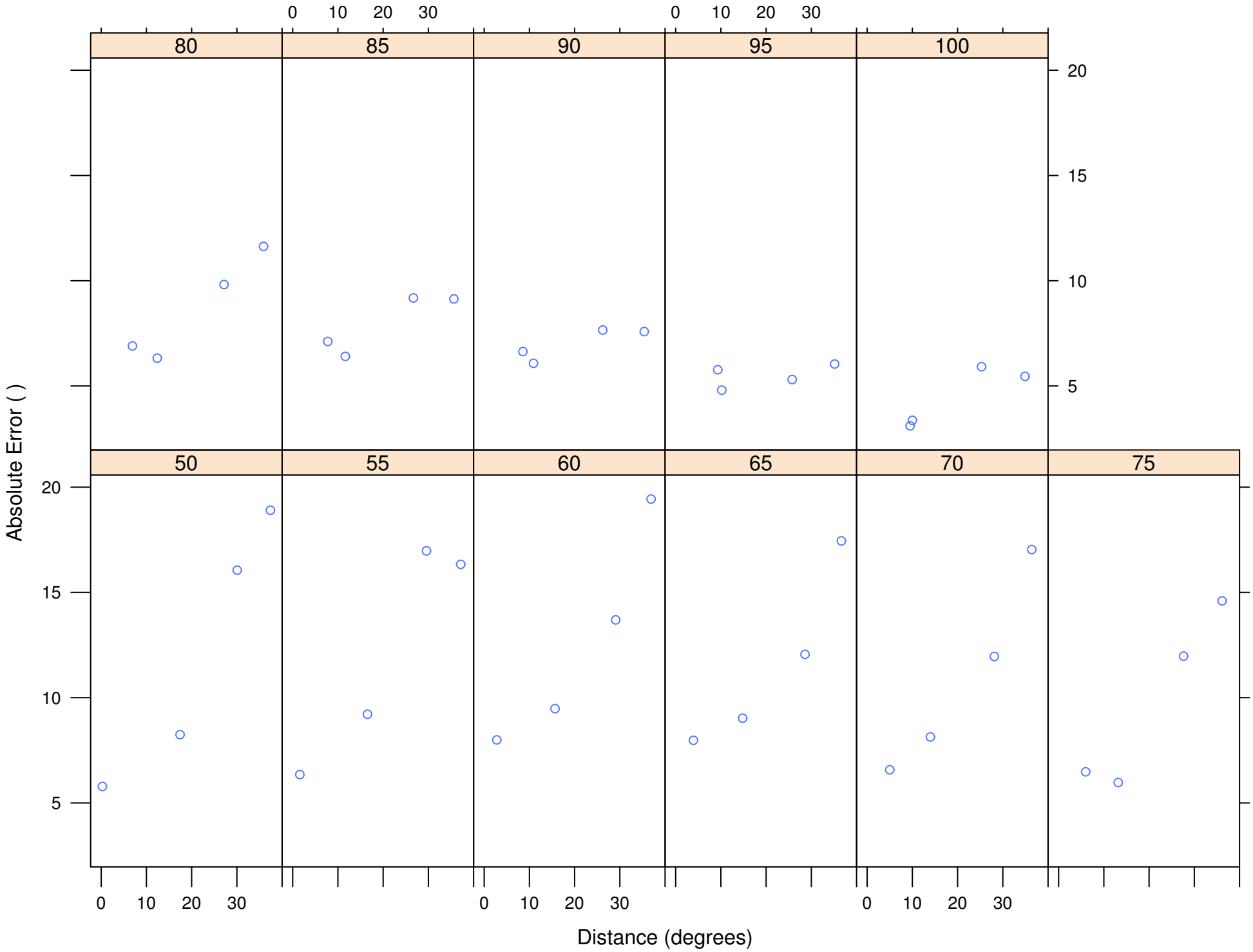
```
xyplot(error ~ distance | as.factor(percent),
                  data = slope,
                  layout = c(6,2),
                  scales = list(y=list(limits=c(5,20))),
                  xlab="Distance (degrees)",
                  ylab="Absolute Error (%)")
```


80
85
90
95
100
50
55
60
65
70
75
0
10
20
30
15
10
Absolute Error ( )
Distance (degrees)

Scales: tck

`tck` a numeric scalar controlling the length of tick marks. Can also be a vector of length 2, to control the length of left/bottom and right/top tick marks separately.

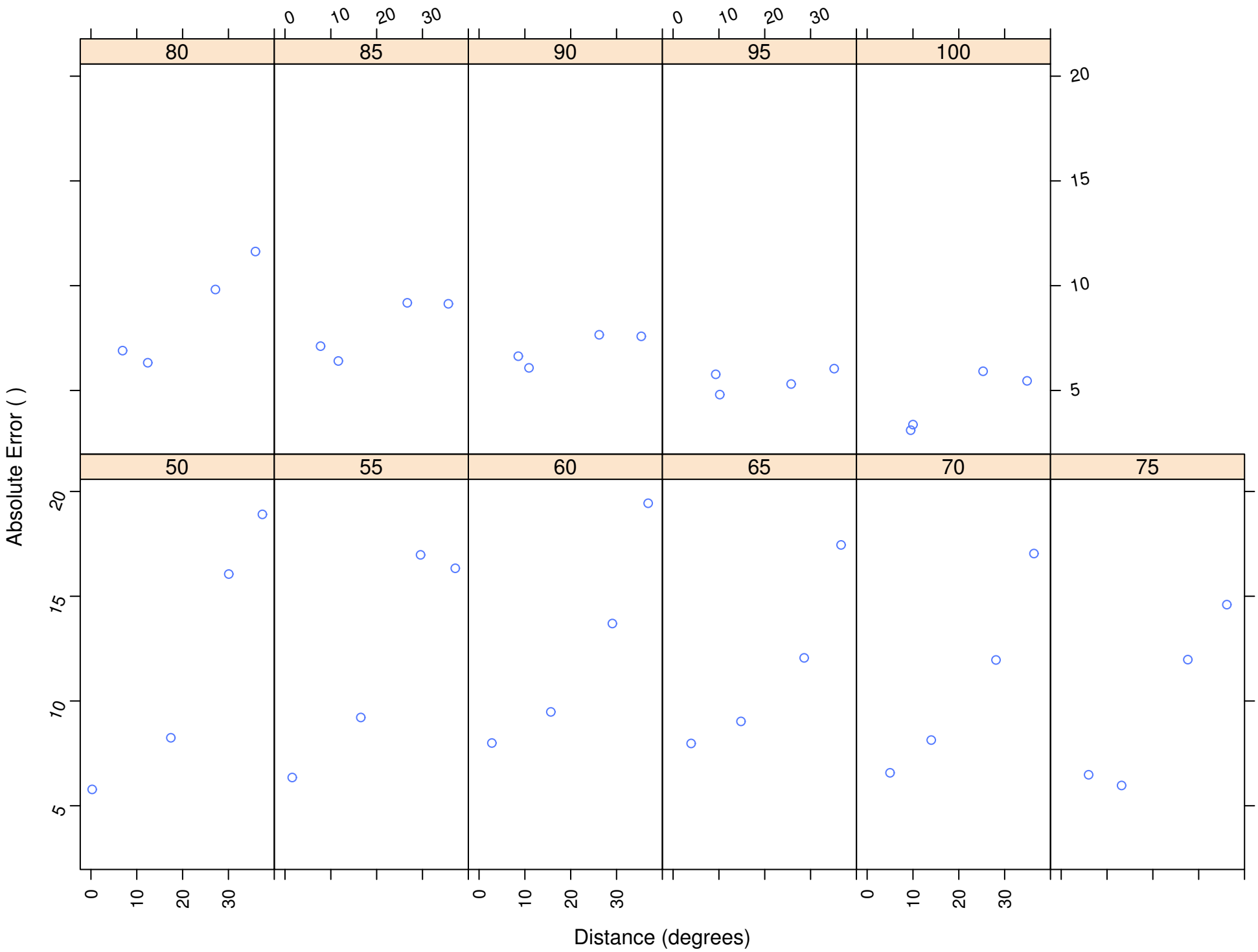
```
xyplot(error ~ distance | as.factor(percent),
                  data = slope,
                  layout = c(6,2),
                  scales = list(y=list(tck=c(2,1)),
                                x=list(tck=c(2,0.5))),
                  xlab="Distance (degrees)",
                  ylab="Absolute Error (%)")
```


80
85
90
95
100
50
55
60
65
70
75
Absolute Error ( )
Distance (degrees)
0
10
20
30
5
10
15
20

Scales: `rot`

`rot` Angle (in degrees) by which the axis labels are to be rotated. Can be a vector of length 2, to control left/bottom and right/top axes separately.

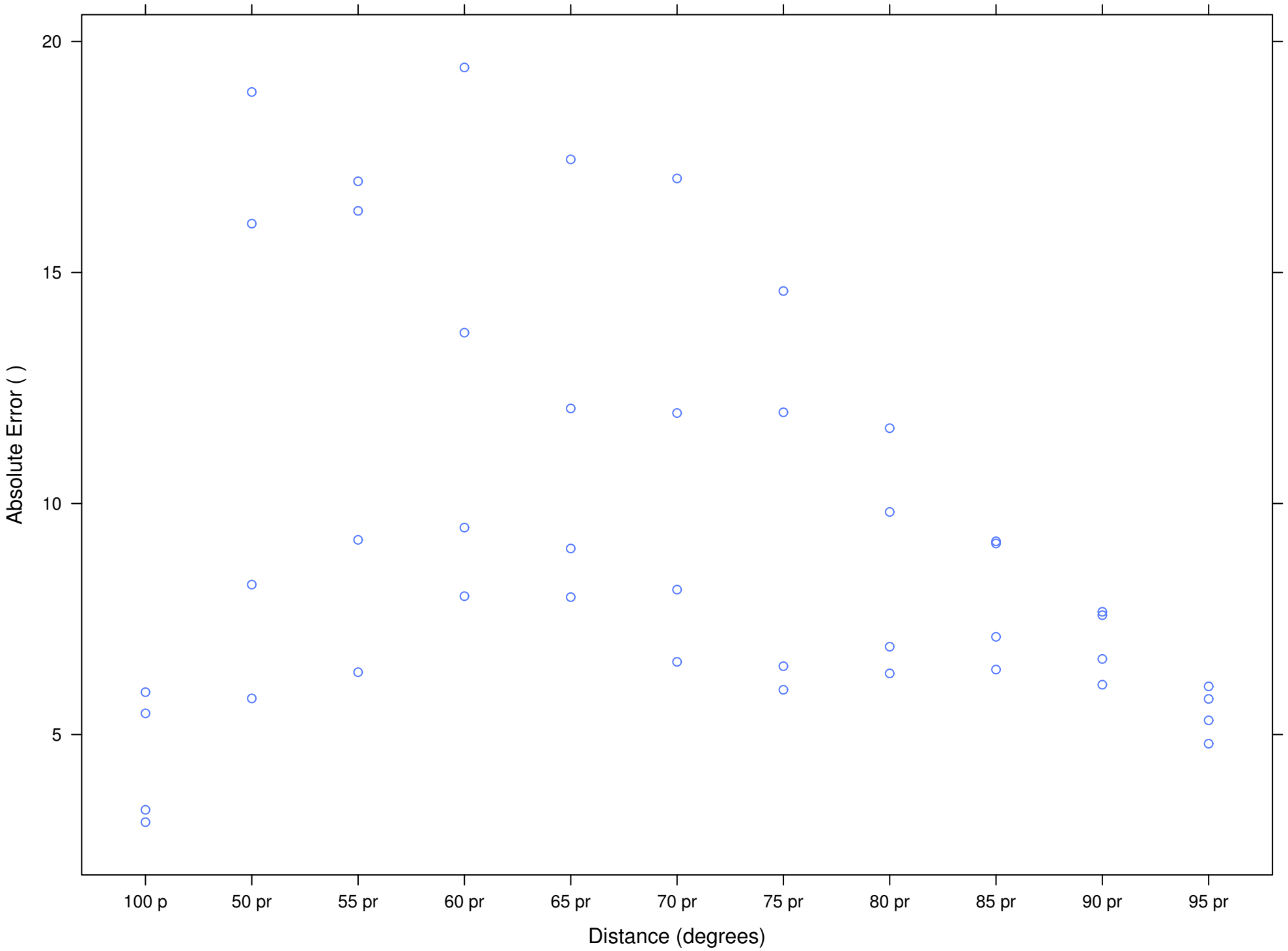
```
xyplot(error ~ distance | as.factor(percent),
                  data = slope,
                  layout = c(6,2),
                  scales = list(y=list(rot=c(70, 10)),
                                x=list(rot=c(90, 20))),
                  xlab="Distance (degrees)",
                  ylab="Absolute Error (%)")
```


80
85
90
95
100
50
55
60
65
70
75
0
10
20
30
5
10
15
20
Absolute Error ( )
Distance (degrees)

Scales: abbreviate

`abbreviate` A logical flag, indicating whether to abbreviate the labels using the abbreviate function. Can be useful for long labels (e.g., in factors), especially on the x-axis.

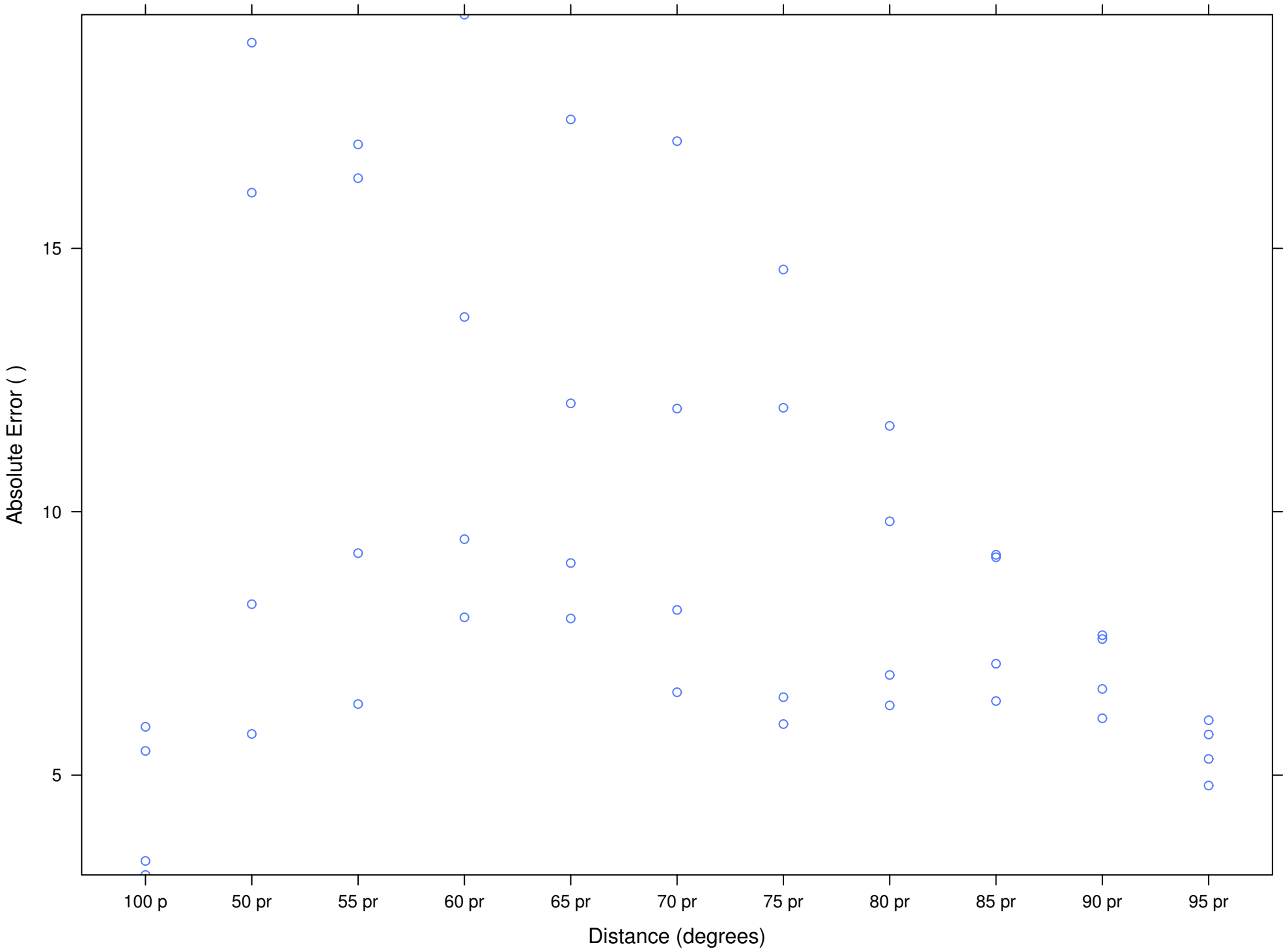
```
slope$pc <- paste(slope$percent, "% percent", sep="")
xyplot(error ~ as.factor(pc),
                 data = slope,
                 scales = list(x=list(abbreviate=TRUE,
                         minlength=5)),
                 xlab="Distance (degrees)",
                 ylab="Absolute Error (%)")
```


20
15
10
5
Absolute Error ( )
100 p
50 pr
55 pr
60 pr
65 pr
70 pr
75 pr
80 pr
85 pr
90 pr
95 pr
Distance (degrees)

Scales: axs

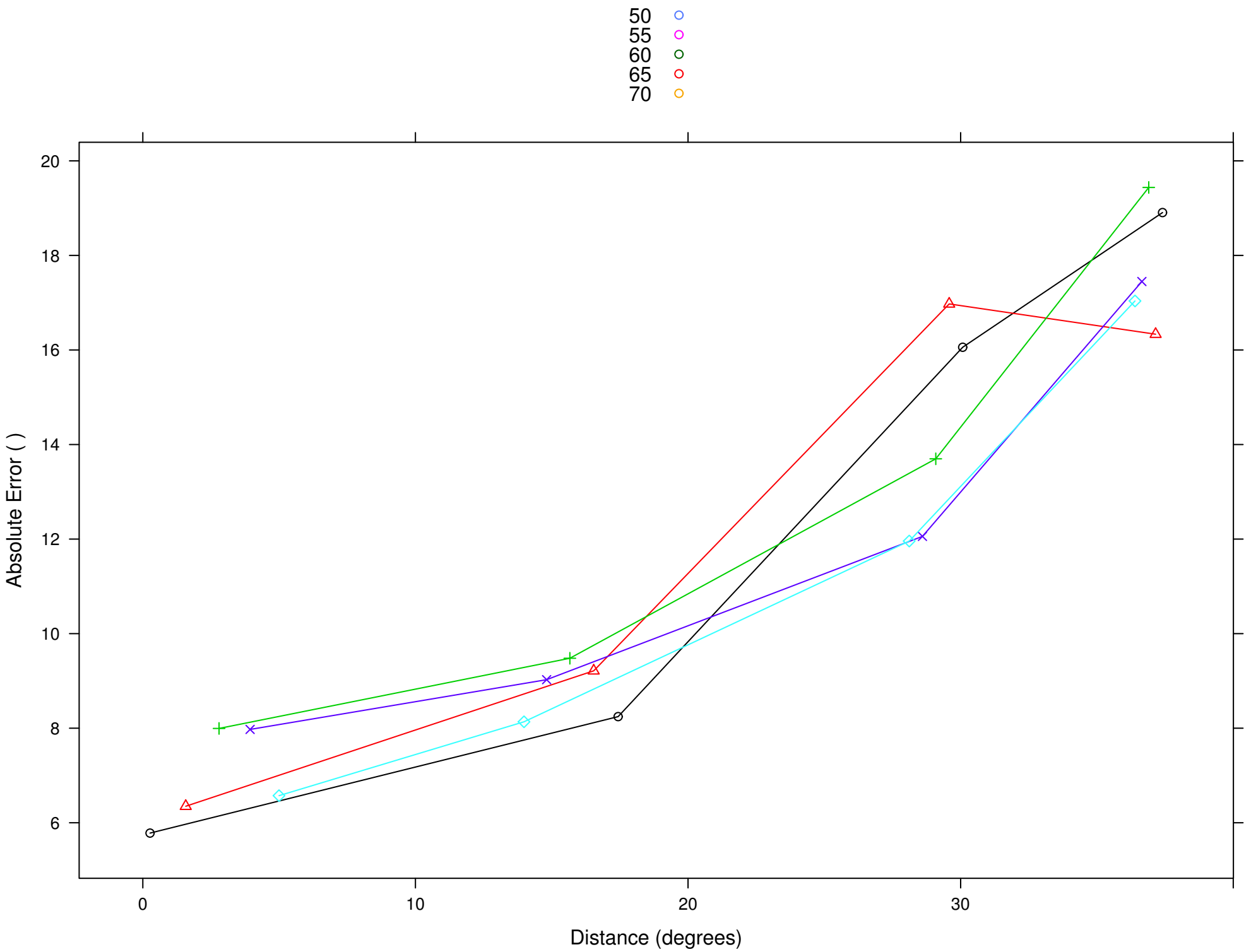
axsA character string, ”r” (default) or ”i”. In the latter case, the axis limits are calculated as the exact data range, instead of being padded on either side.

```
slope$pc <- paste(slope$percent, "% percent", sep="")
xyplot(error ~ as.factor(pc),
                 data = slope,
                 scales = list(x=list(abbreviate=TRUE,
                                      minlength=5),
                               y=list(axs="i")),
                 xlab="Distance (degrees)",
                 ylab="Absolute Error (%)")
```


Absolute Error ( )
15
10
5
100 p
50 pr
55 pr
60 pr
65 pr
70 pr
75 pr
80 pr
85 pr
90 pr
95 pr
Distance (degrees)

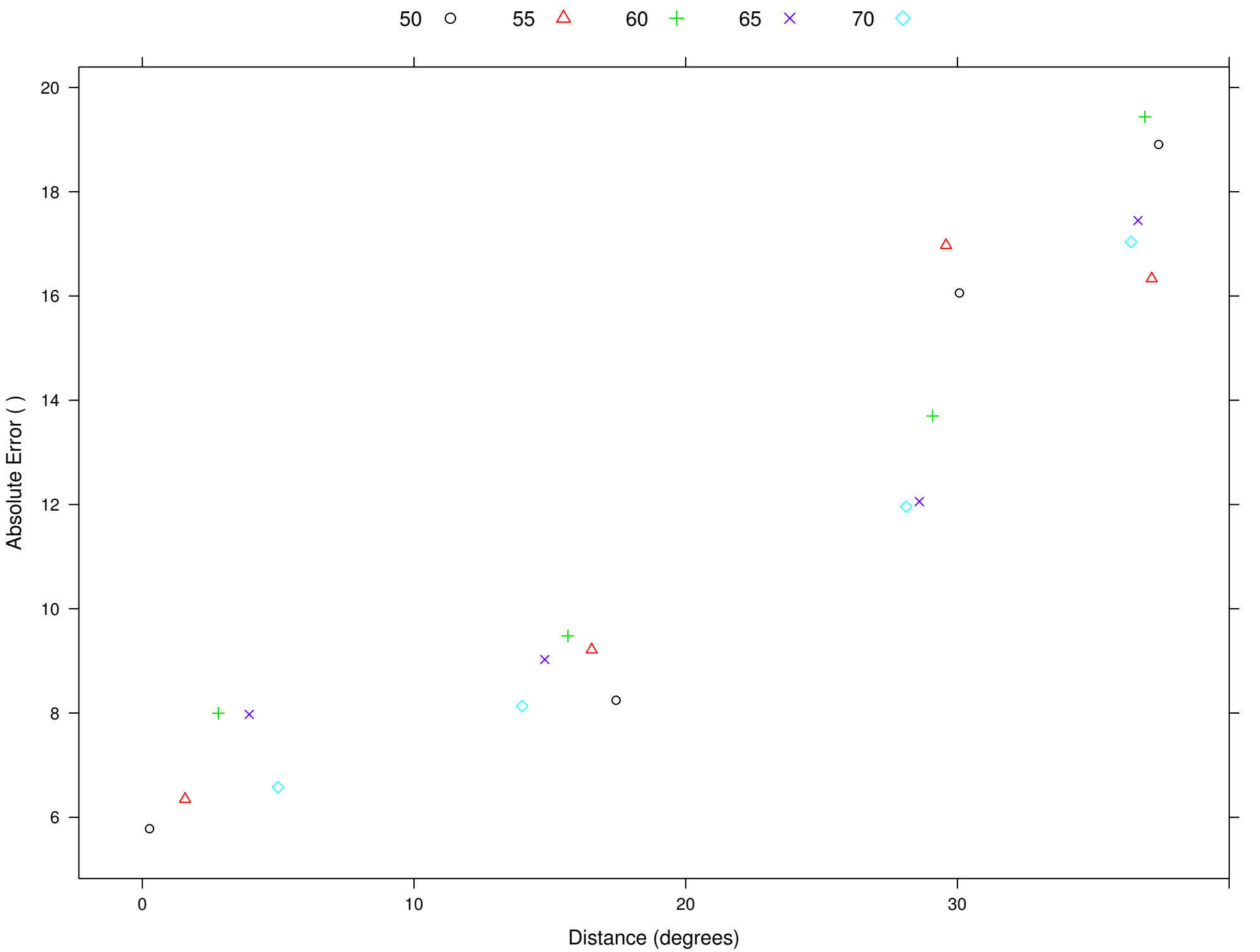
Key: auto.key

```
xyplot(error ~ distance,
        groups = as.factor(percent),
        data = subset(slope, percent <= 70),
        auto.key = TRUE,
        type = "b",
        scales = list(y=list(tick.number=8)),
        xlab="Distance (degrees)",
        ylab="Absolute Error (%)"
)
```


50
55
60
65
70
Absolute Error ( )
Distance (degrees)
20
18
16
14
12
10
8
6
0
10
20
30

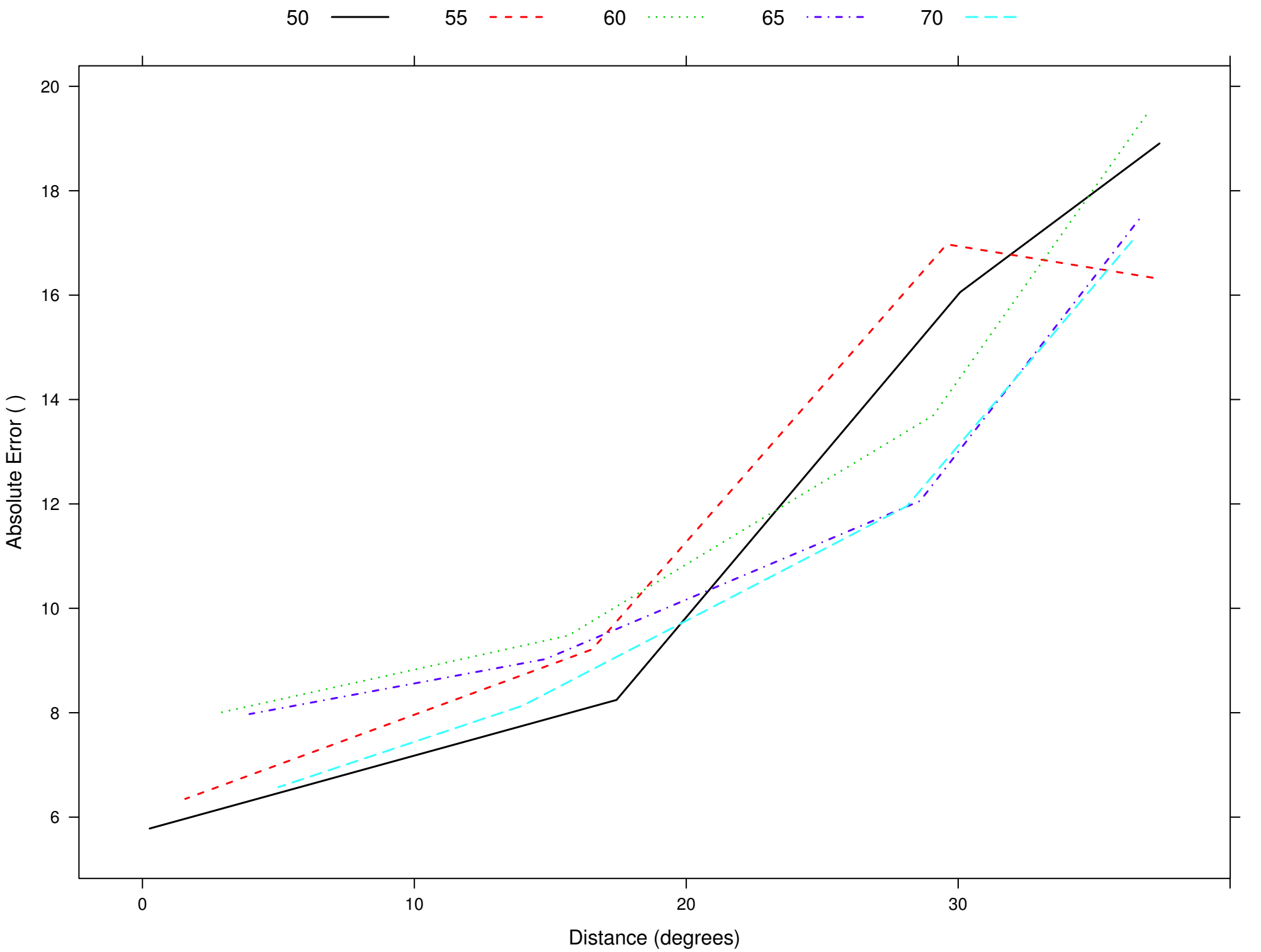
Key: points

```
label <- levels(
           as.factor(subset(slope, percent <=70)$percent)
)
xyplot(error ~ distance,
         groups = as.factor(percent),
         data = subset(slope, percent <= 70),
         auto.key = TRUE,
         type = "p",
         pch = 1:5,
         col = 1:5,
         key = list(type = c("p"),
              text = list(label = label, cex = 1.2),
              points = list(col=1:5, pch = 1:5),
              column = 5,
              space = "top"),
         scales = list(y=list(tick.number=8)),
         xlab="Distance (degrees)",
         ylab="Absolute Error (%)")
```


50
55
60
65
70
Absolute Error ( )
Distance (degrees)
6
8
10
12
14
16
18
20
0
10
20
30

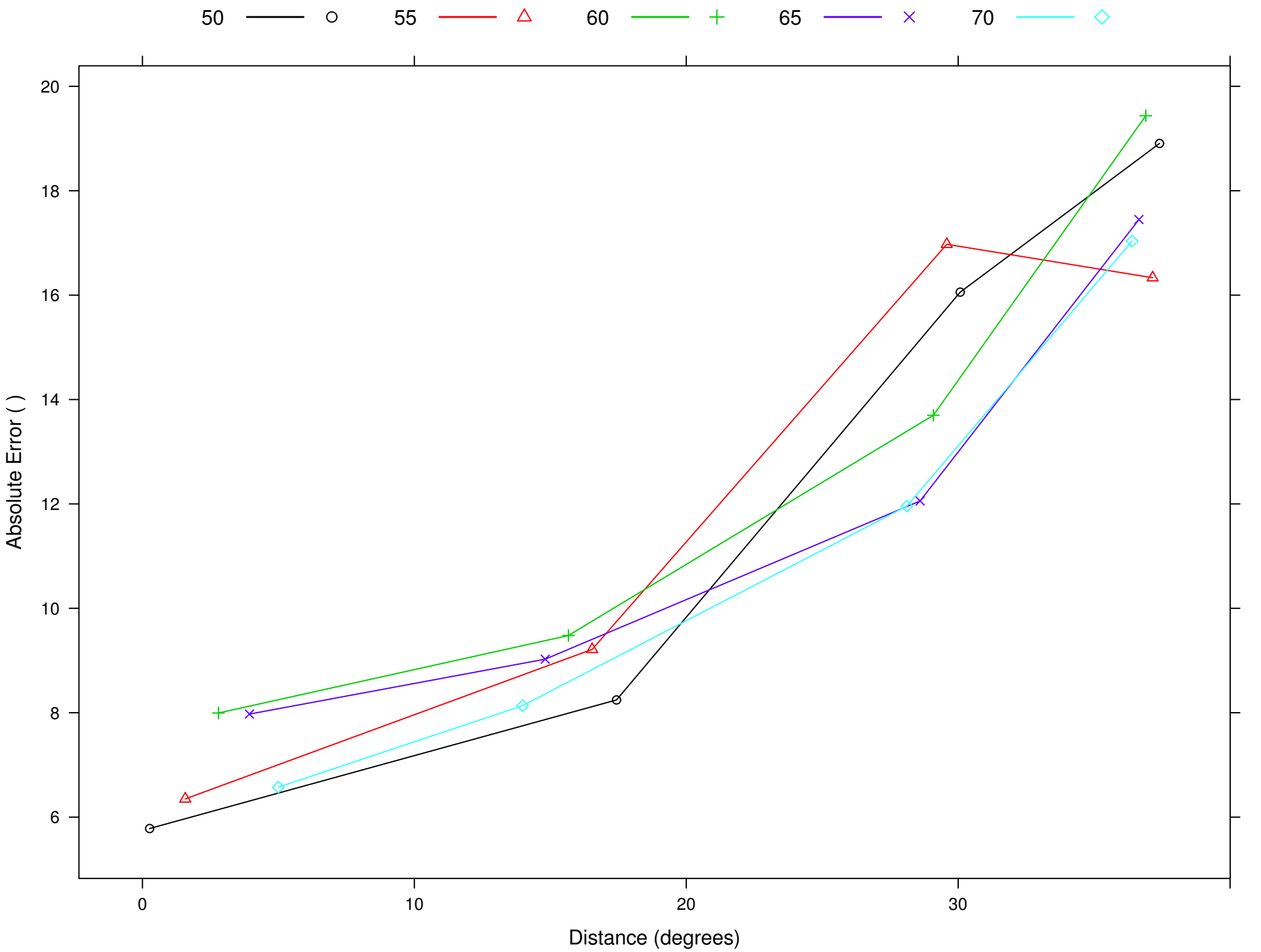
Key: lines

```
xyplot(error ~ distance,
       groups = as.factor(percent),
       data = subset(slope, percent <= 70),
       auto.key = TRUE,
       type = "l",
       lty = 1:5,
       lwd = 1.5,
       col = 1:5,
       key = list(type = "l",
            text = list(label = label),
            lines = list(col=1:5, lty=1:5, lwd=1.5),
            column = 5,
            space = "top"),
       scales = list(y=list(tick.number=8)),
       xlab="Distance (degrees)",
       ylab="Absolute Error (%)")
```


50
55
60
65
70
20
18
16
14
12
10
8
6
Absolute Error ( )
0
10
20
30
Distance (degrees)

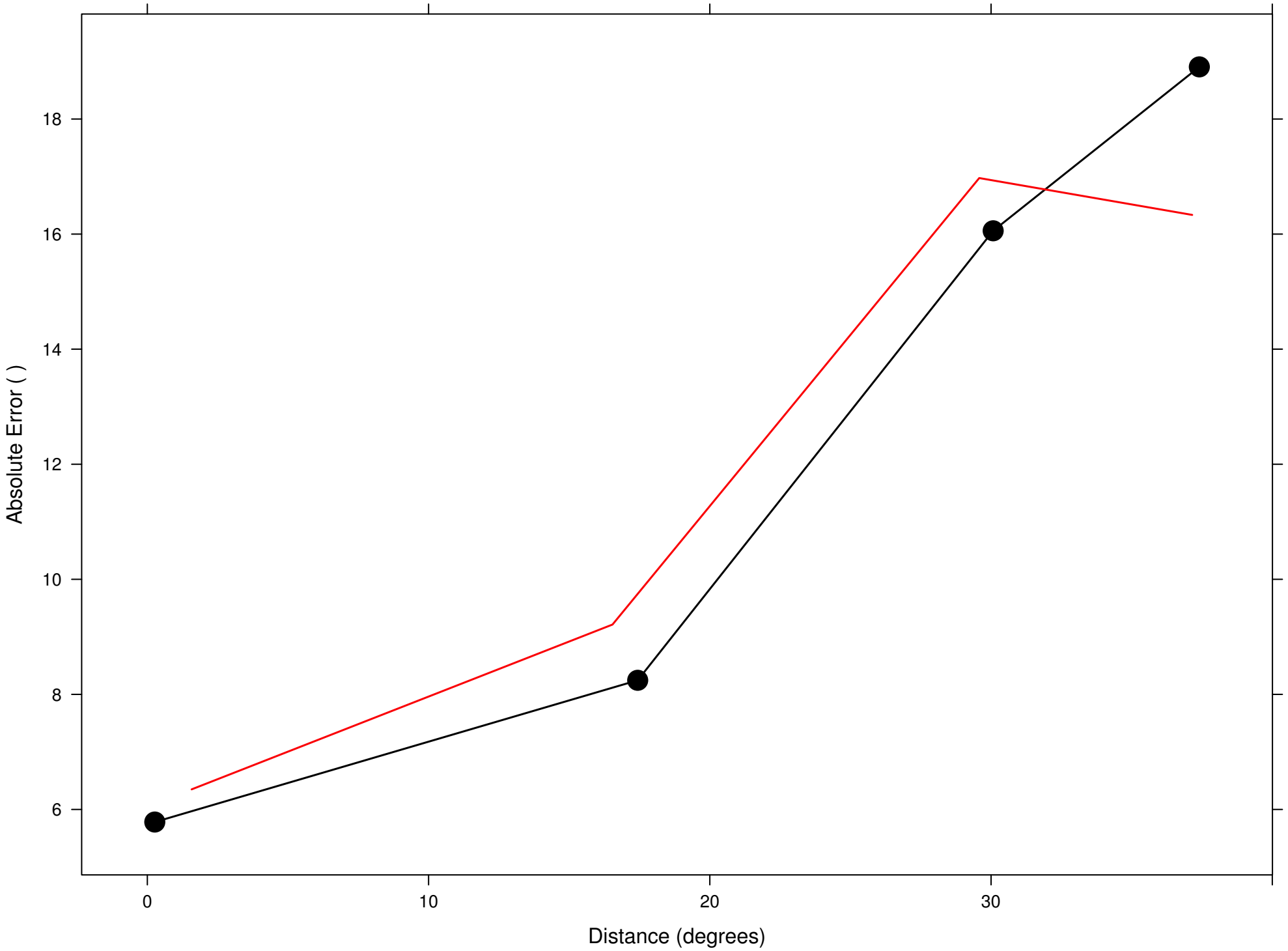
Key: both

```
xyplot(error ~ distance,
        groups = as.factor(percent),
        data = subset(slope, percent <= 70),
        auto.key = TRUE,
        type = "b",
        pch = 1:5,
        col = 1:5,
        key = list(type = c("l"),
             text = list(label = label),
             lines = list(col =1:5, lwd =1.5),
             points = list(col=1:5, pch = 1:5),
             column = 5,
             space = "top"),
        scales = list(y=list(tick.number=8)),
        xlab="Distance (degrees)",
        ylab="Absolute Error (%)")
```


50
55
60
65
70
Absolute Error ( )
Distance (degrees)
20
18
16
14
12
10
8
6
0
10
20
30

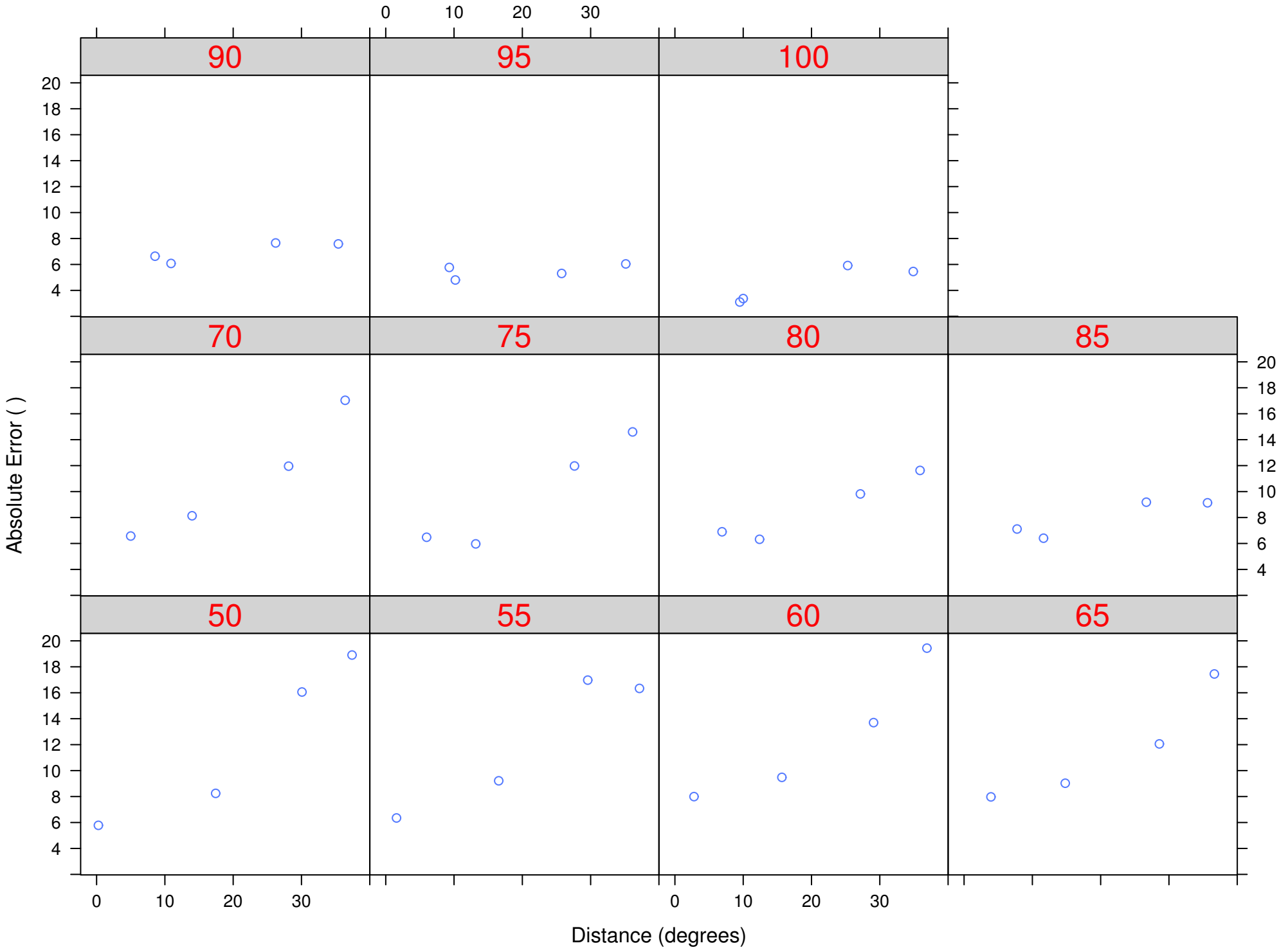
Type: distribute.type

```
xyplot(error ~ distance,
        groups = as.factor(percent),
        data = subset(slope, percent < 60),
        type = c("b","l"),
        distribute.type = TRUE,
        pch = 16,
        col = 1:2,
        lwd = 1.5,
        cex = 2,
        scales = list(y=list(tick.number=8)),
        xlab="Distance (degrees)",
        ylab="Absolute Error (%)")
```


Absolute Error ( )
18
16
14
12
10
8
6
0
10
20
30
Distance (degrees)

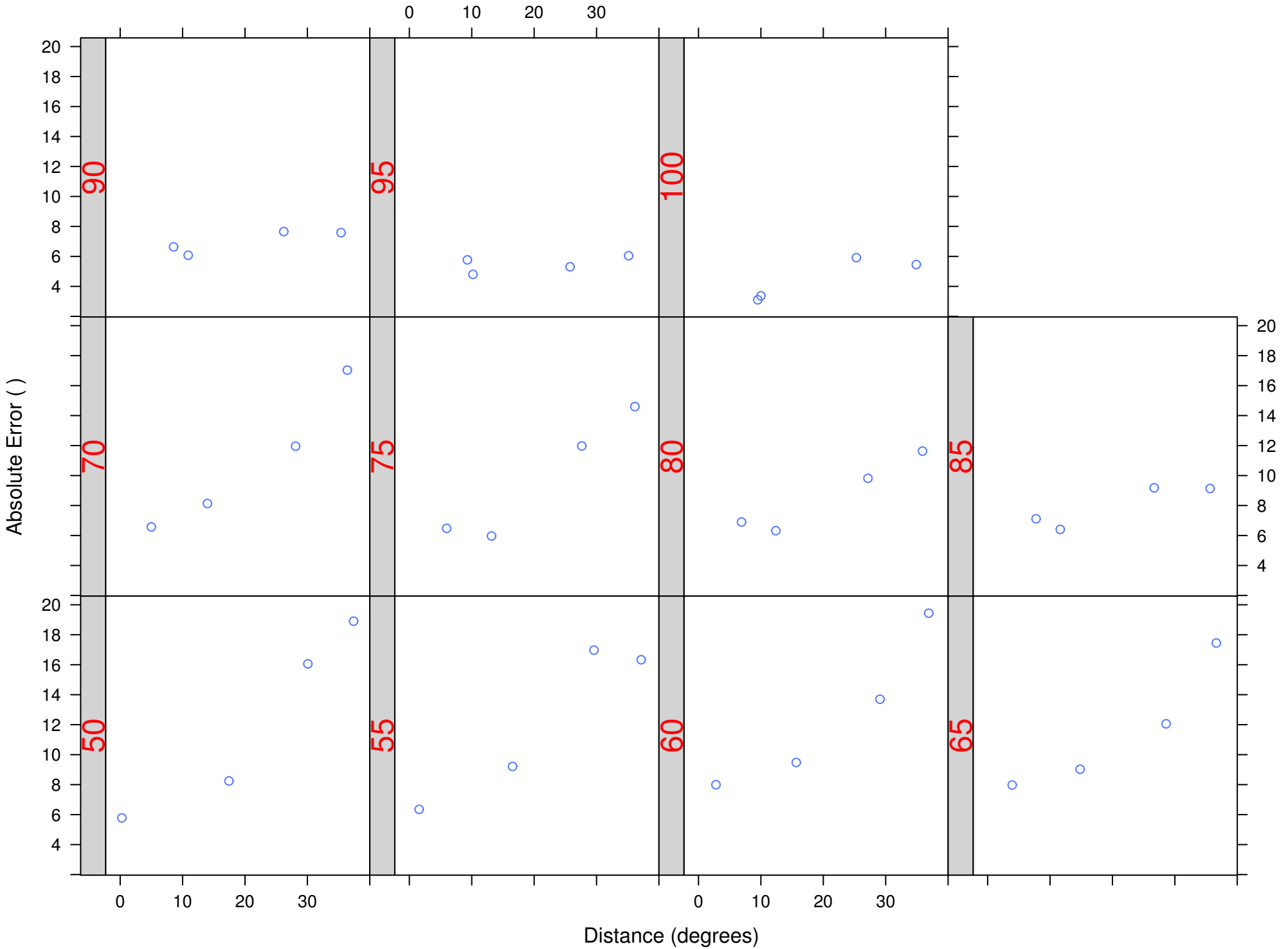
Strip

```
xyplot(error ~ distance | as.factor(percent),
        data = slope,
        strip = strip.custom(
                par.strip.text= list(
                     cex = 1.5, col = "red")),
        par.settings = list(
                layout.heights = list(strip = 1.5),
                strip.background=list(col="lightgrey")),
        scales = list(y=list(tick.number=8)),
        xlab="Distance (degrees)",
        ylab="Absolute Error (%)")
```


90
95
100
70
75
80
85
50
55
60
65
Absolute Error ( )
Distance (degrees)
0
10
20
30
4
6
8
10
12
14
16
18
20

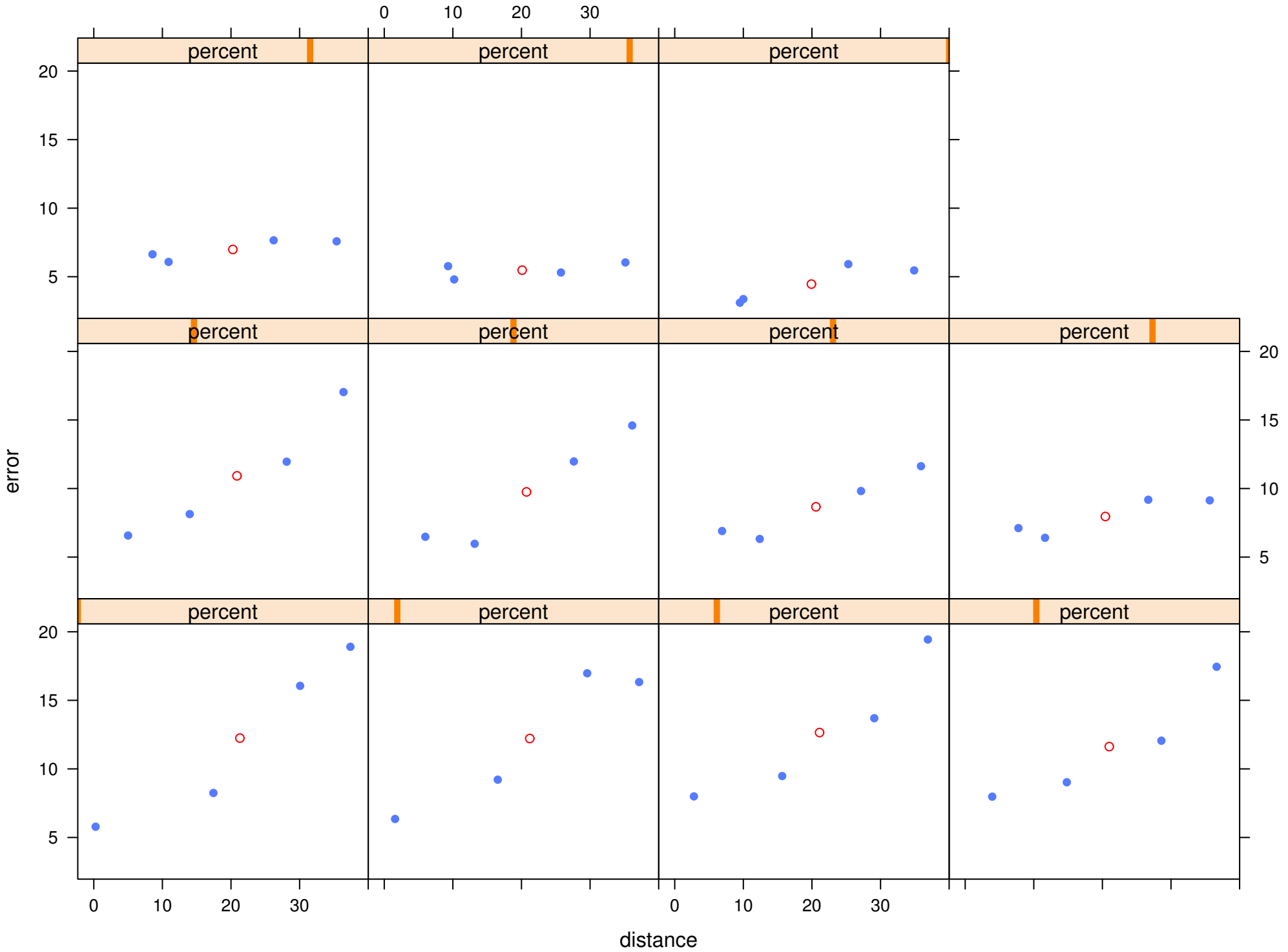
Strip: strip.left

```
xyplot(error ~ distance | as.factor(percent),
        data = slope,
        strip = FALSE,
        strip.left =
                strip.custom(par.strip.text=list(
                    cex = 1.5, col = "red")),
        par.settings = list(
                layout.heights = list(strip = 1.5),
                strip.background=list(col="lightgrey")),
        scales = list(y=list(tick.number=8)),
        xlab="Distance (degrees)",
        ylab="Absolute Error (%)")
```


90
95
100
70
75
80
85
50
55
60
65
Absolute Error ( )
Distance (degrees)
0
10
20
30
4
6
8
10
12
14
16
18
20

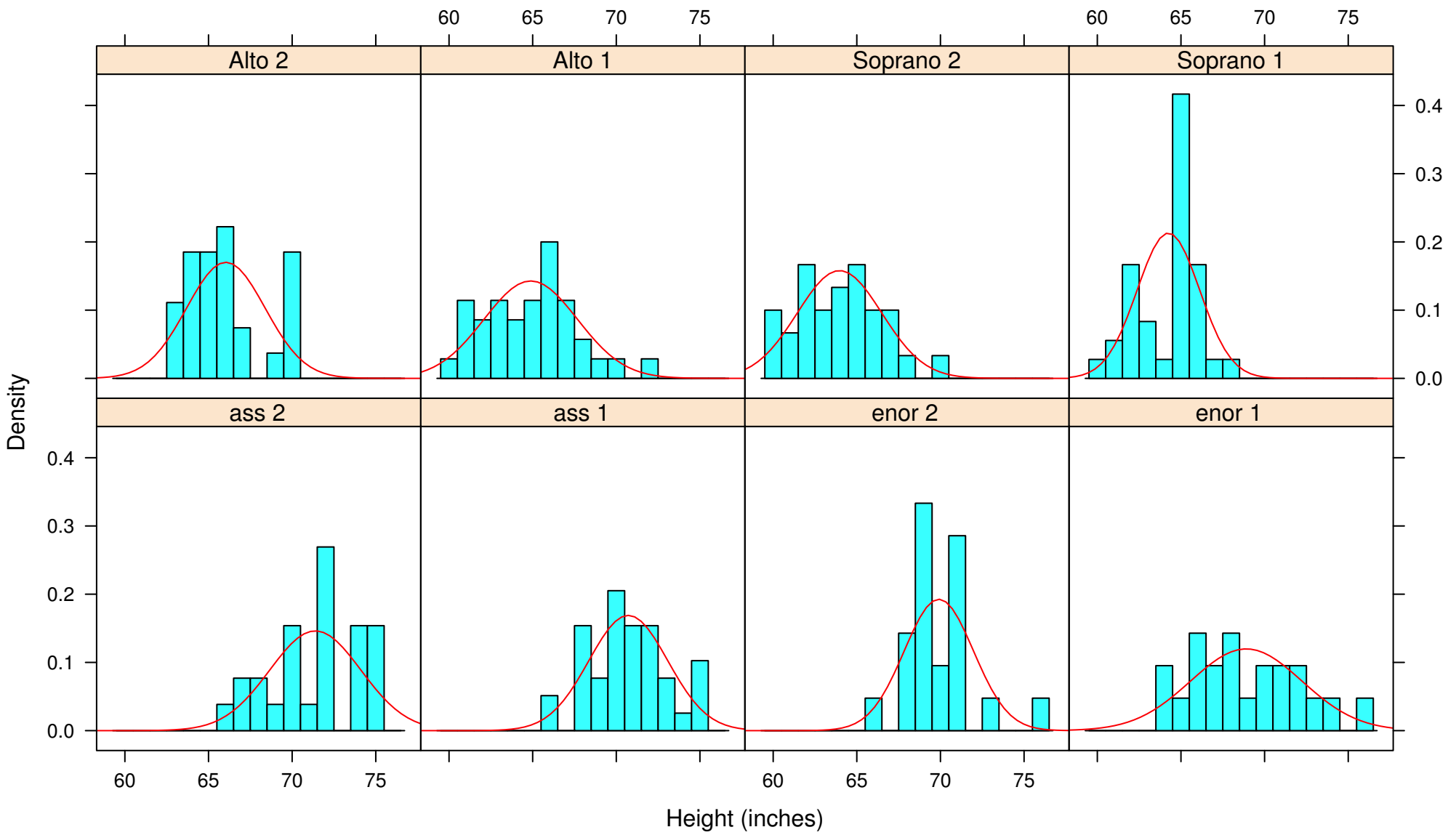
Panel Function: panel.points

```
xyplot(error ~ distance | percent,
        data = slope,
        pch = 16,
        panel = function(x,y,...){
                panel.xyplot(x,y,...)
                panel.points(mean(x), mean(y), col="red")
        }
)
```


percent
percent
percent
percent
percent
percent
percent
percent
percent
percent
percent
percent
error
distance
0
10
20
30
5
10
15
20

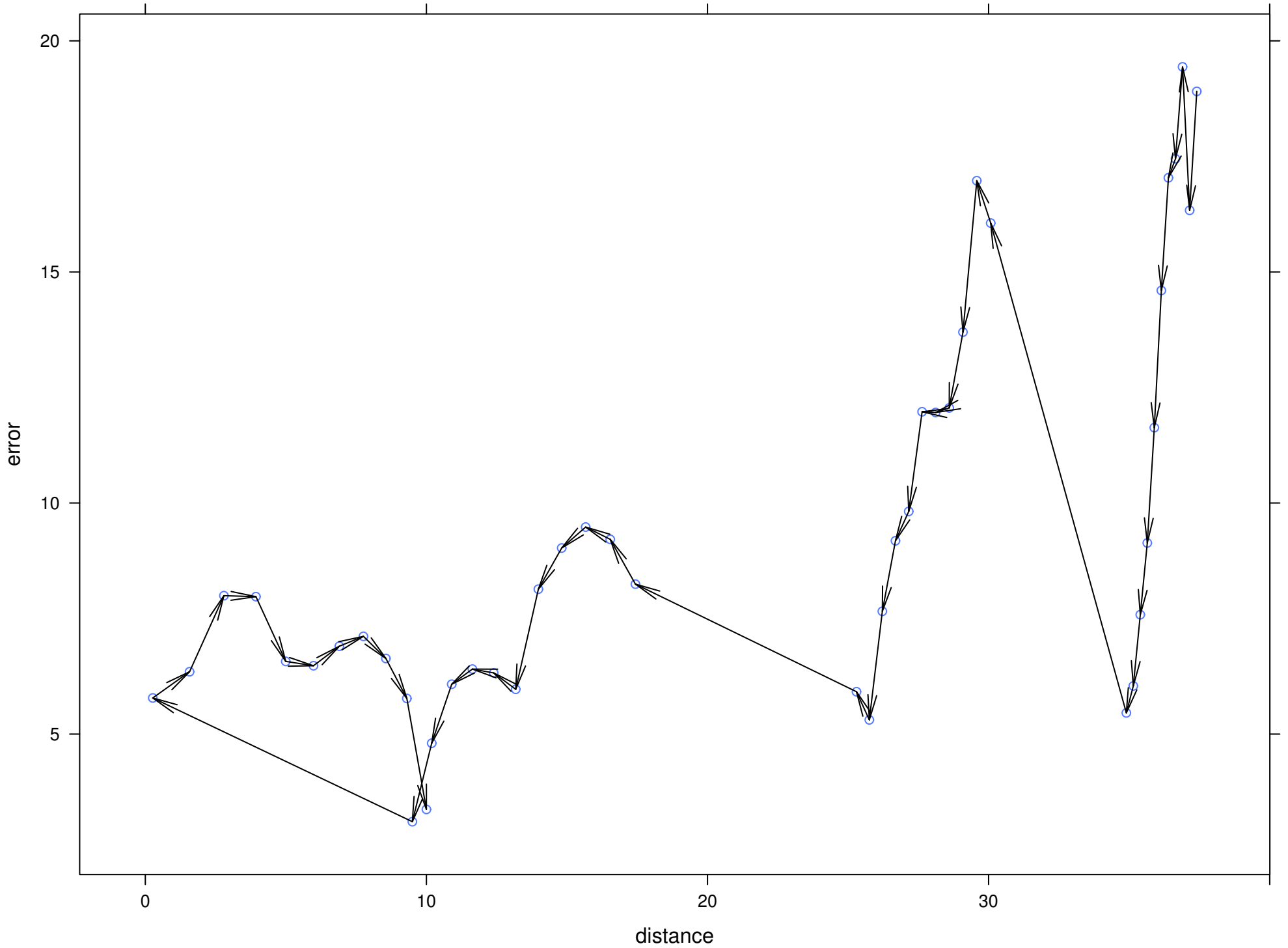
Panel Function: panel.mathdensity

```
histogram( ~ height | voice.part,
         data = singer,
         nint = 17,
         type = "density",
         endpoints = c(59.5, 76.5),
         layout = c(4, 2),
         aspect = 1,
         xlab = "Height (inches)",
         panel = function(x,...){
                 panel.histogram(x,...)
                 panel.mathdensity(dmath = dnorm,
                                   col = "red",
                                   lwd = 1,
                                   args = list(
                                         mean=mean(x),
                                         sd=sd(x)), ...)
         }
)
```


Alto 2
Alto 1
Soprano 2
Soprano 1
ass 2
ass 1
enor 2
enor 1
Density
Height (inches)
60
65
70
75
0.0
0.1
0.2
0.3
0.4

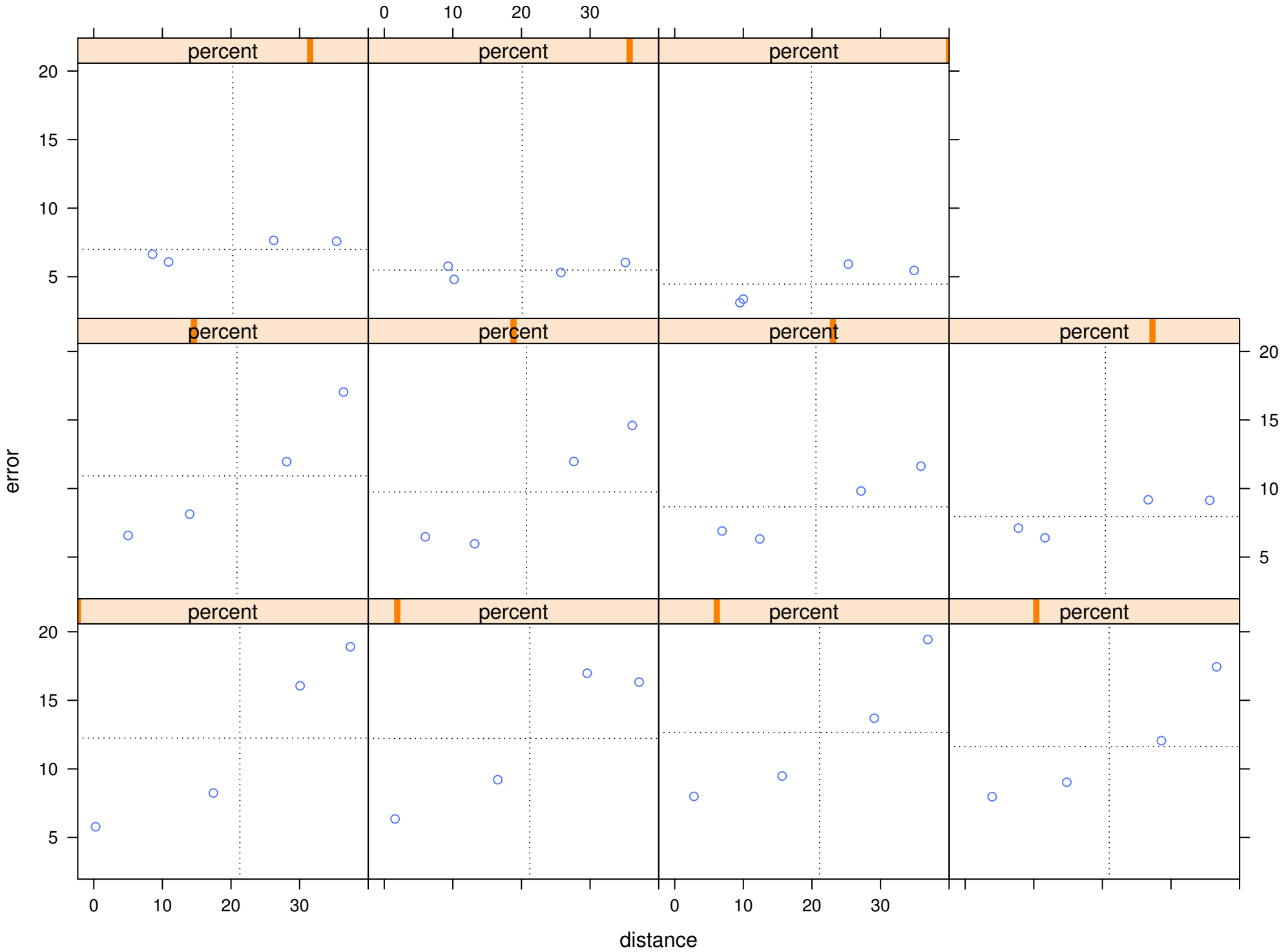
Panel Function: panel.arrows

```
xyplot(error ~ distance,
        data=slope,
        panel=function(x,y,...){
           panel.xyplot(x,y,...)
           for(i in 1:(length(x)-1)){
             panel.arrows(x[i],y[i],x[i+1],y[i+1],
                angle=10, code=2, length=0.2,...)
           }
        }
)
```


20
15
10
5
error
0
10
20
30
distance

Panel Function: panel.abline

```
xyplot(error ~ distance | percent,
        data = slope,
        panel = function(x,y,...){
                panel.abline(h=mean(y), v=mean(x),
                                        lty=3,...)
                panel.xyplot(x,y,...)
        }
)
```


percent
percent
percent
percent
percent
percent
percent
percent
percent
percent
error
distance
0
10
20
30
5
10
15
20

Panel Function: panel.abline

```
xyplot(error ~ distance | percent,
        data = slope,
        panel = function(x,y,...){
                cof <- lm(y ~ x)$coefficients
                panel.abline(a=cof[1], b=cof[2],
                        col="red", ...)
                panel.xyplot(x,y,...)
        }
)
```

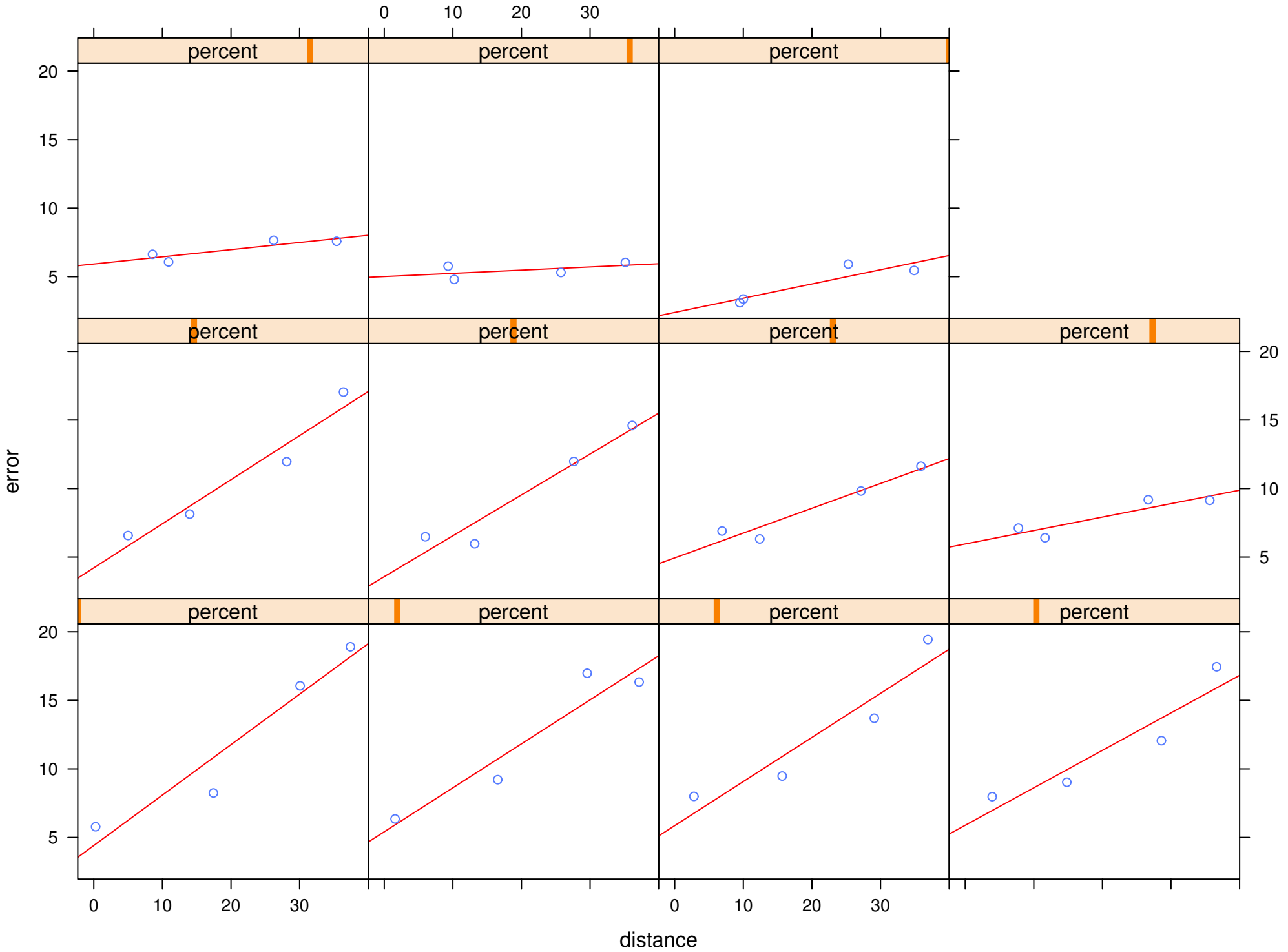
or equivalently,

```
xyplot(error ~ distance | percent,
        data = slope,
        panel = function(x,y,...){
                cof <- lm(y ~ x)
                panel.abline(reg = cof, col="red", ...)
                panel.xyplot(x,y,...)
        }
)
```

Panel Function: panel.abline

or equivalently,

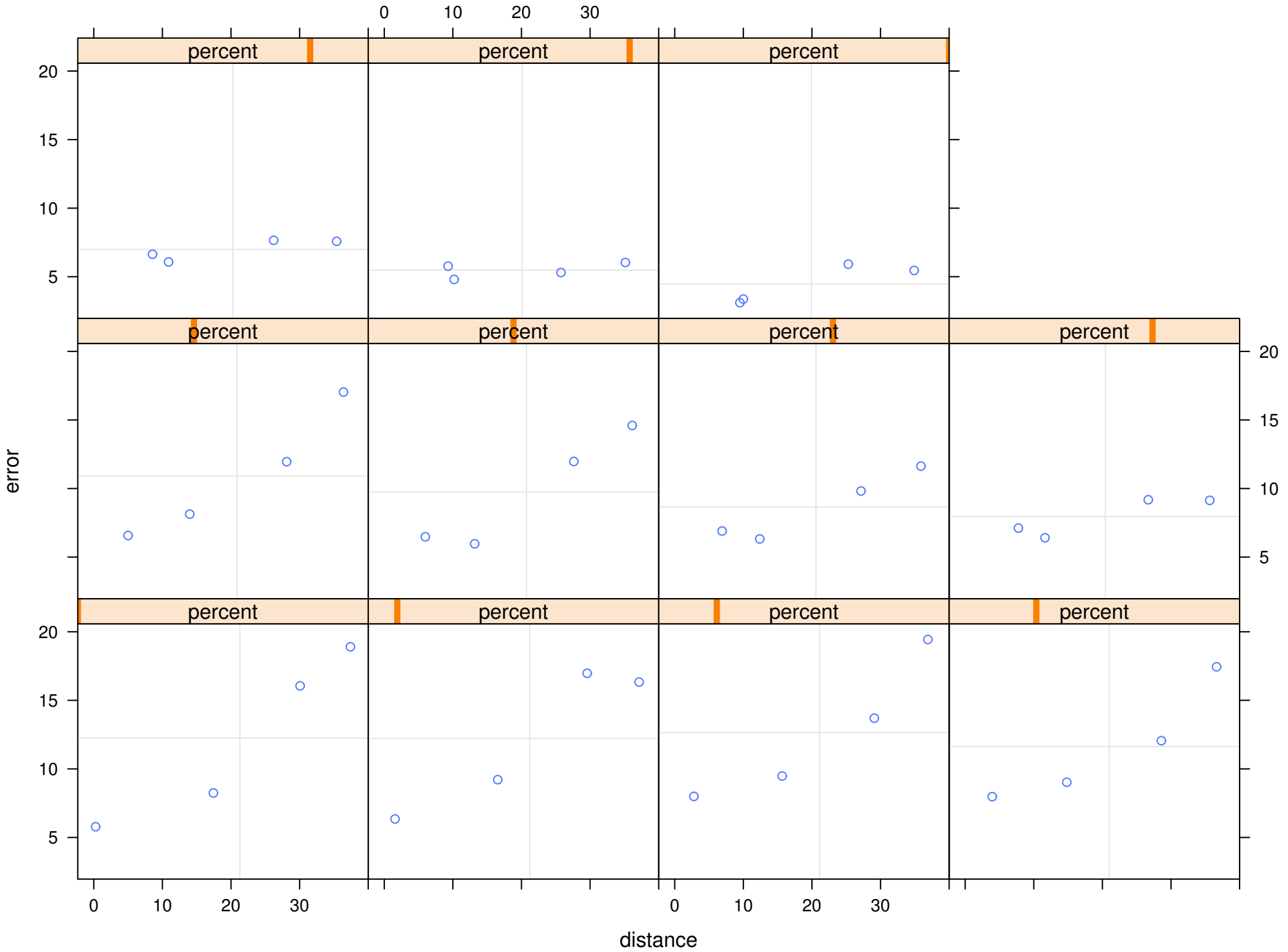
```
xyplot(error ~ distance | percent,
         data = slope,
         panel = function(x,y,...){
                 cof <- lm(y ~ x)
                 panel.lmline(x,y,col="red", ...)
                 panel.xyplot(x,y,...)
         }
)
```


percent
percent
percent
percent
percent
percent
percent
percent
percent
percent
percent
percent
error
distance
0
10
20
30
5
10
15
20

Panel Function: panel.refline

or equivalently,

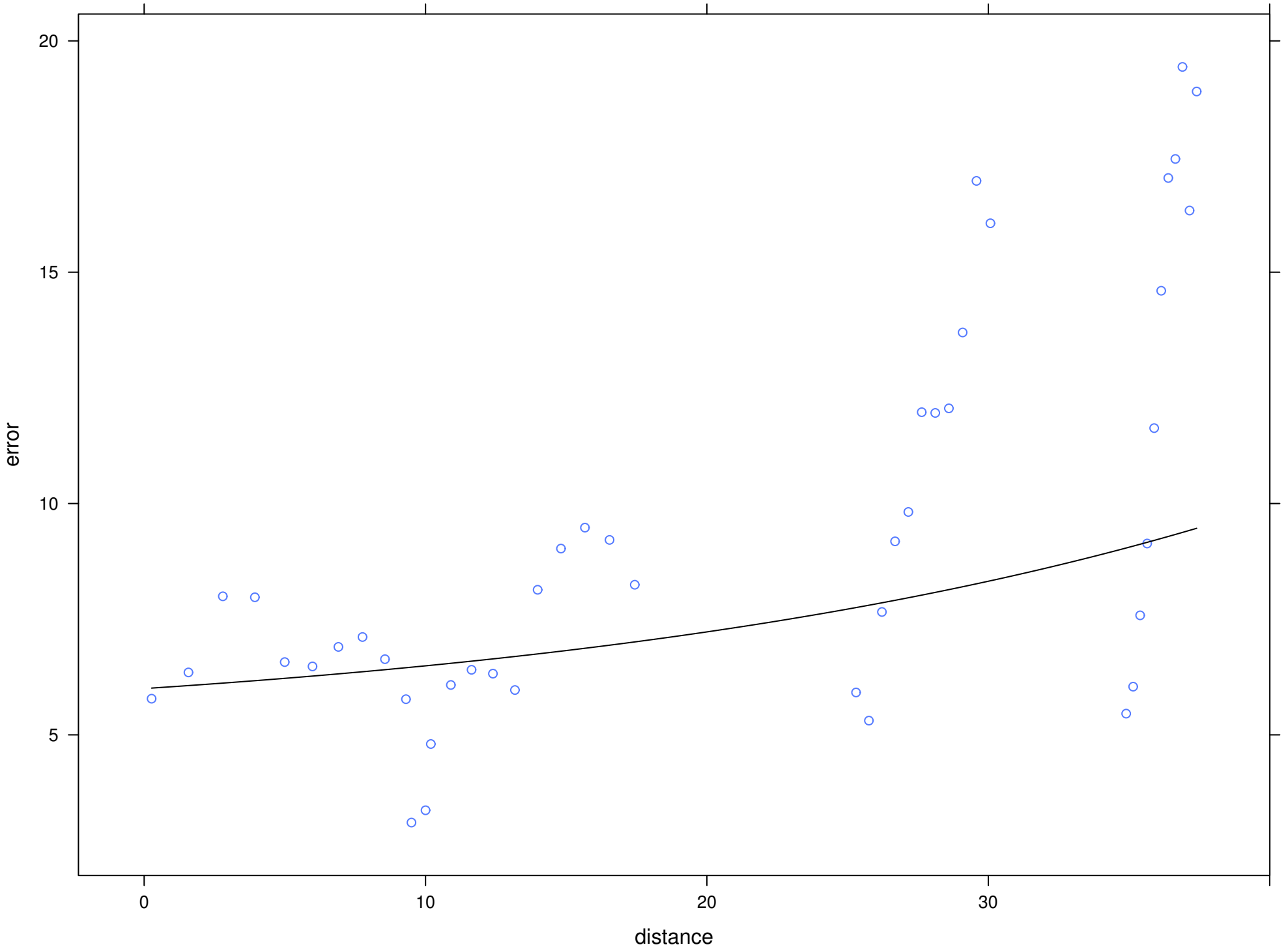
```
xyplot(error ~ distance | percent,
         data = slope,
         panel = function(x,y,...){
                 cof <- lm(y ~ x)
                 panel.refline(v=mean(x), h=mean(y),...)
                 panel.xyplot(x,y,...)
         }
)
```


percent
percent
percent
percent
percent
percent
percent
percent
percent
percent
percent
percent
error
distance
0
10
20
30
5
10
15
20

Panel Function: panel.curve

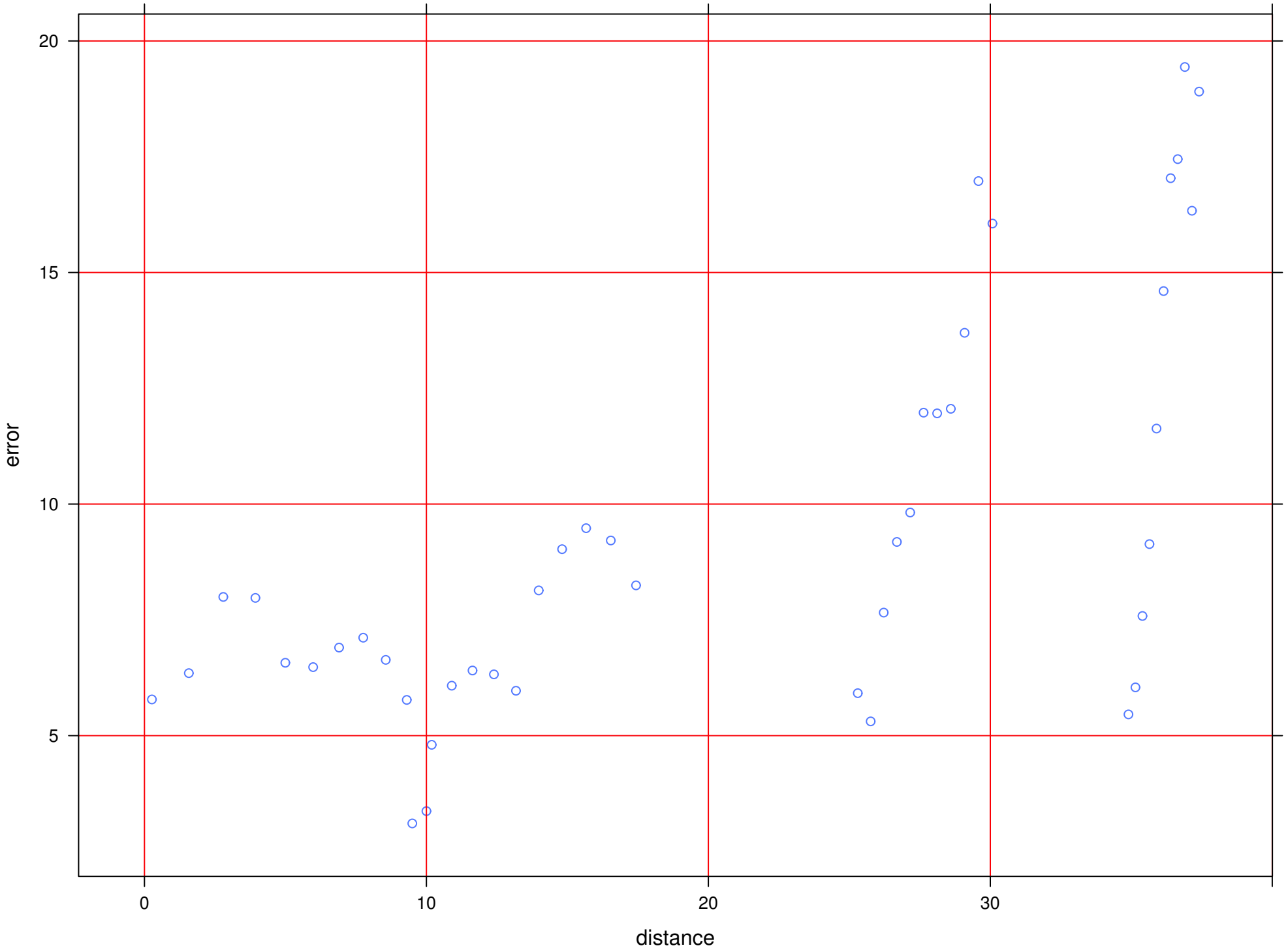
or equivalently,

```
xyplot(error ~ distance | percent,
         data = slope,
         panel = function(x,y,...){
                 panel.curve(exp(x/25)+5,
                      from=min(x), to=max(x), ...)
                 panel.xyplot(x,y,...)
         }
)
```


20
15
10
5
error
0
10
20
30
distance

Panel Function: panel.grid

```
xyplot(error ~ distance,
       data = slope,
       panel = function(x,y,...){
               panel.grid(h=-1, v=-1, col="red", ...)
               panel.xyplot(x,y,...)
       }
)
```

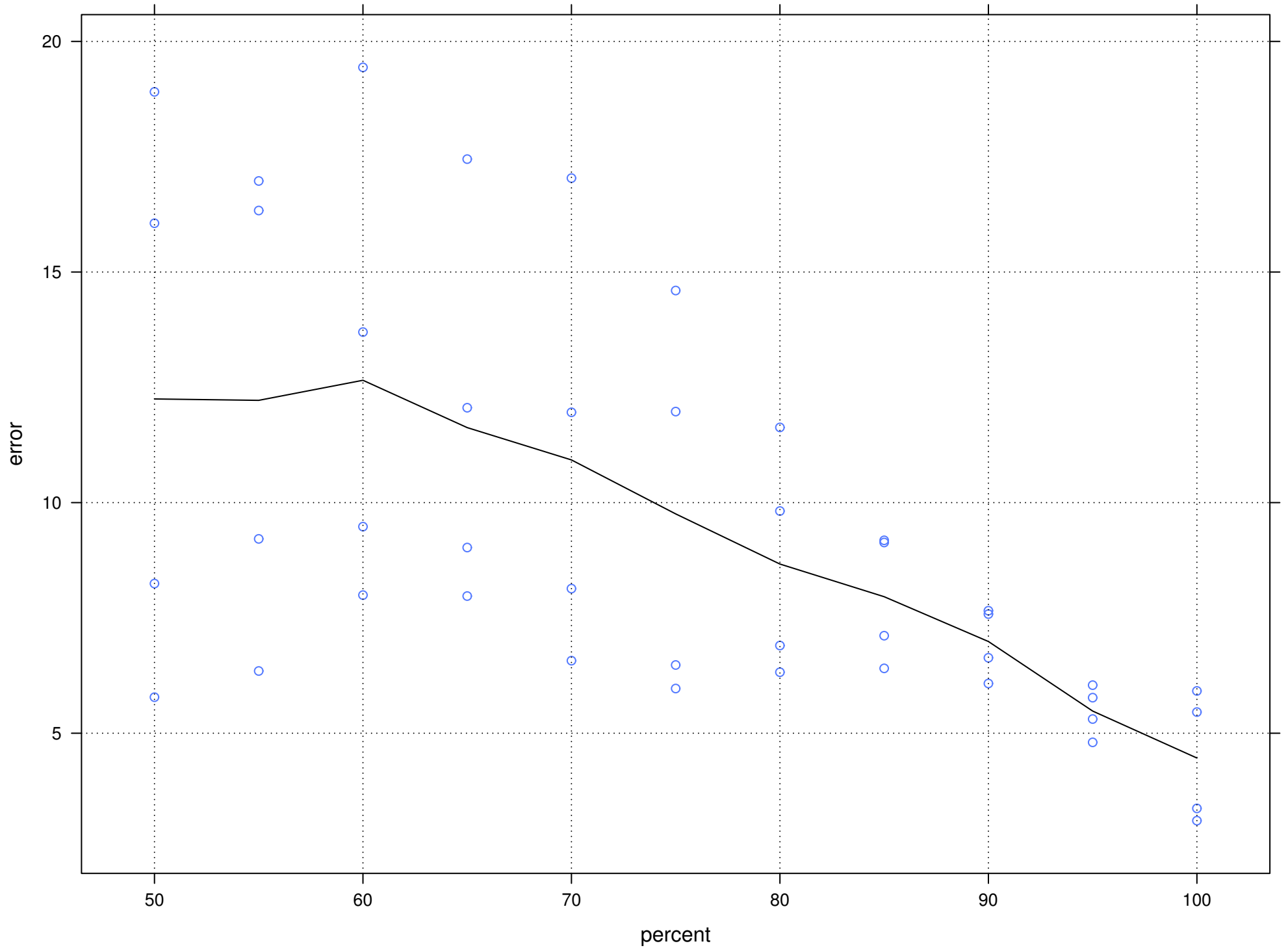

20
15
10
5
error
0
10
20
30
distance

Panel Function: panel.average

```
xyplot(error ~ percent,
        data = slope,
        panel = function(x,y,...){
                panel.xyplot(x,y,...)
                panel.grid(h=-1, v=-1,col="black",
                        lty=3,...)
                panel.average(x,y,fun = mean,
                        horizontal = FALSE,
                        col="black",...)
        }
)
```

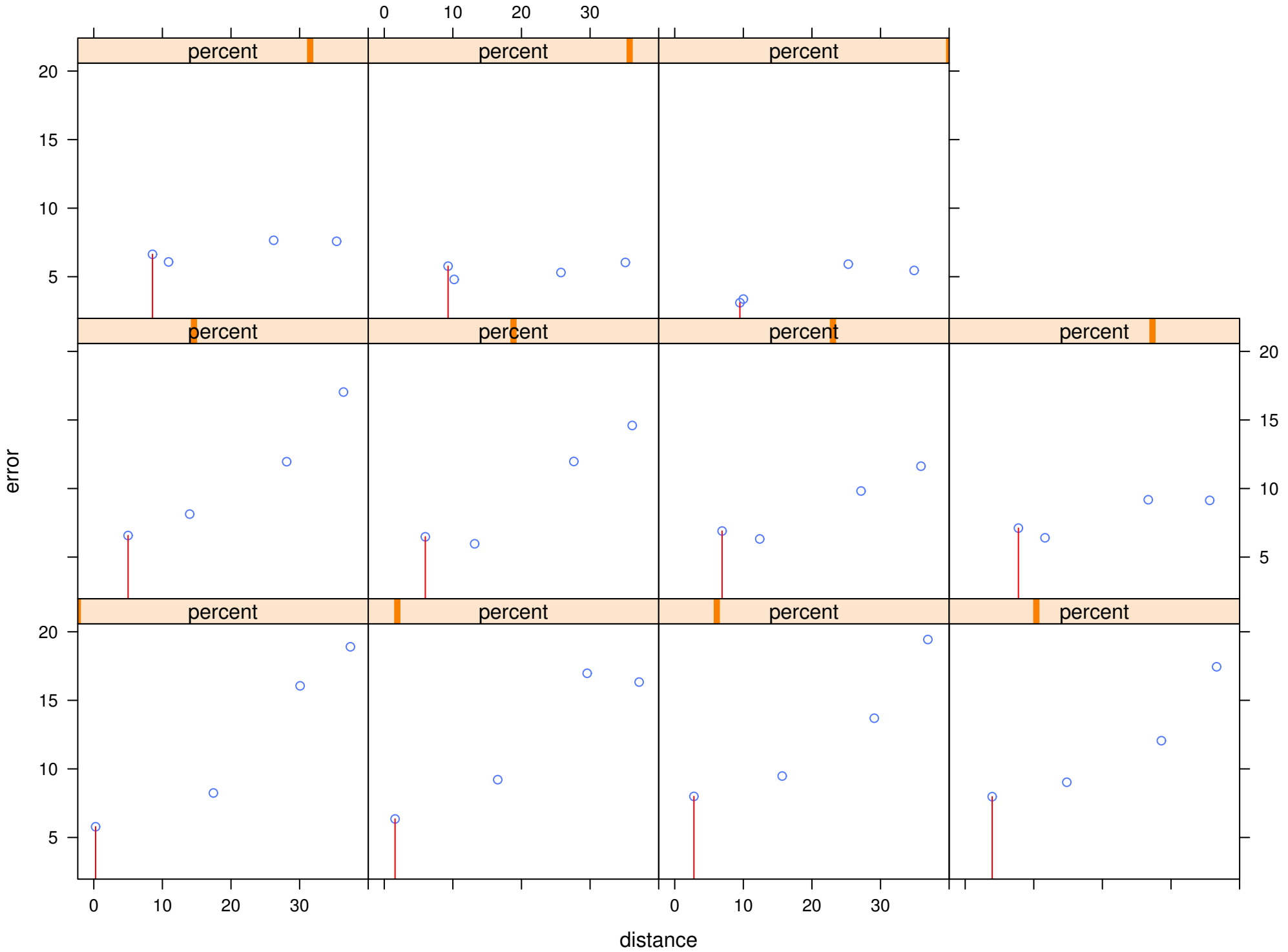
or equivalently,

```
xyplot(error ~ percent,
        data = slope,
        panel = function(x,y,...){
                panel.xyplot(x,y,...)
                panel.grid(h=-1, v=-1,col="black",
                        lty=3,...)
                panel.linejoin(x, y, fun = mean,
                        horizontal = FALSE,
                        col="black", ...)
        }
)
```


20
15
10
5
error
50
60
70
80
90
100
percent

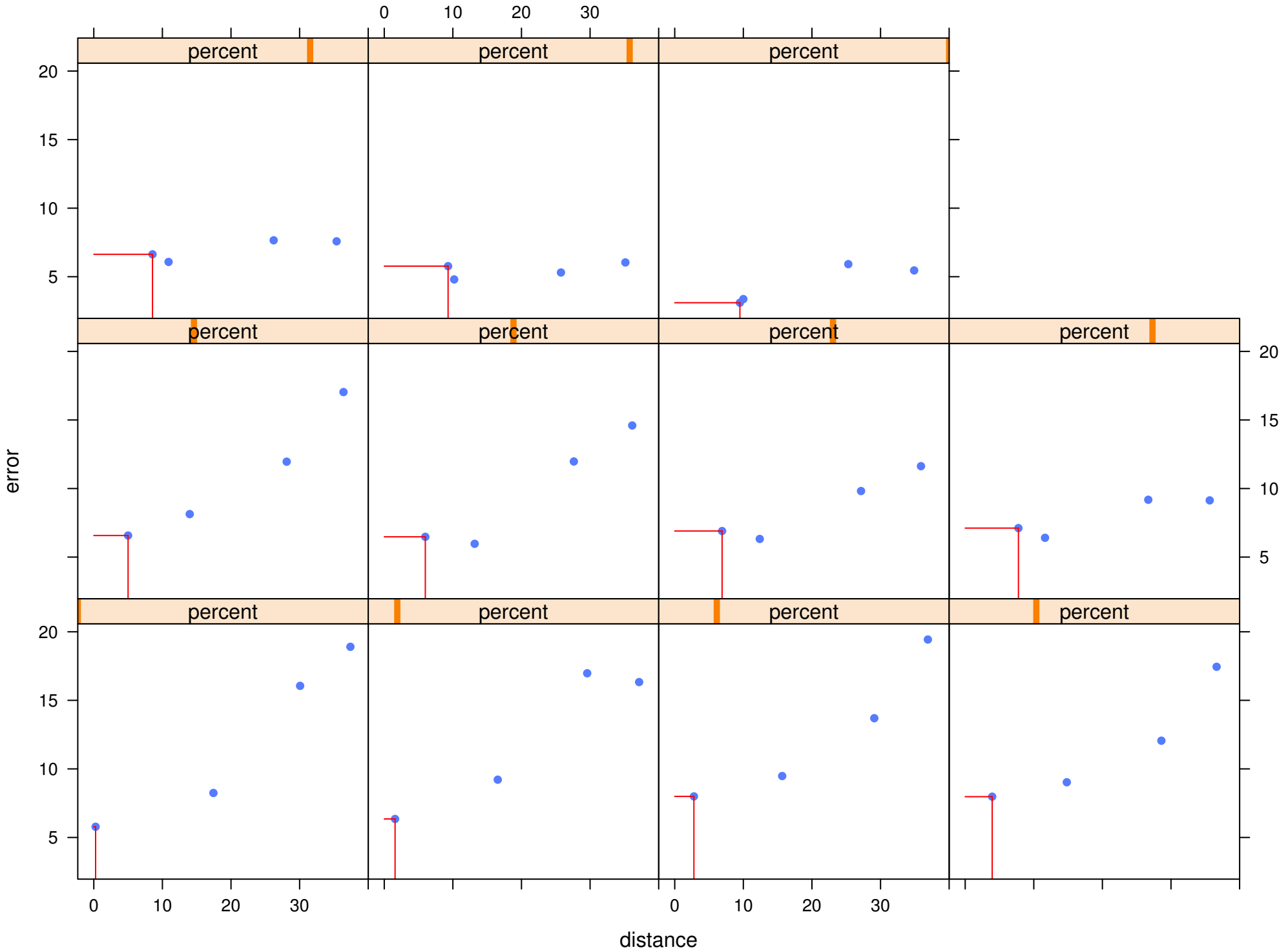
Panel Function: panel.segments

```
xyplot(error ~ distance | percent,
        data = slope,
        panel = function(x,y,...){
                panel.segments( min(x), y[which.min(x)],
                        min(x), 0, col="red", ...)
                panel.xyplot(x,y,...)
        }
)
```


percent
percent
percent
percent
percent
percent
percent
percent
percent
percent
error
distance
0
10
20
30
5
10
15
20

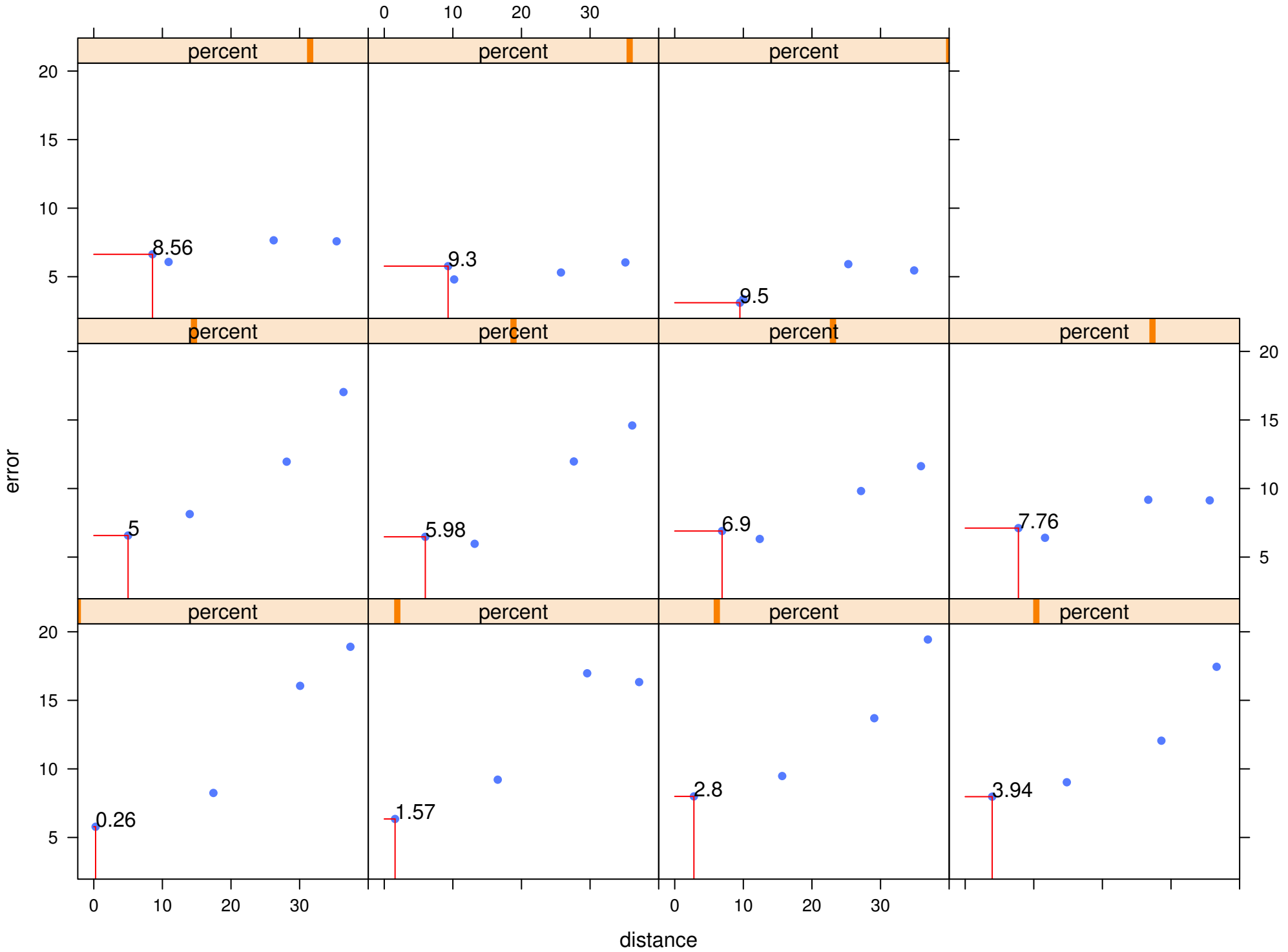
Panel Function: panel.segments

```
xyplot(error ~ distance | percent,
         data = slope,
         panel = function(x,y,...){
            panel.segments(
                  rep(min(x),2), rep(y[which.min(x)],2),
                  c(0, min(x)), c(y[which.min(x)], 0),
                  col="red", ...)
            panel.xyplot(x,y,...)
         }
)
```


percent
percent
percent
percent
percent
percent
percent
percent
percent
percent
percent
percent
0
10
20
30
5
10
15
20
error
distance

Panel Function: panel.text

```
xyplot(error ~ distance | percent,
         data = slope,
         panel = function(x,y,...){
            panel.segments(
                  rep(min(x),2), rep(y[which.min(x)],2),
                  c(0, min(x)), c(y[which.min(x)], 0),
                  col="red", ...)
            panel.text(min(x), y[which.min(x)],
                  round(min(x),2), adj=c(0,0))
            panel.xyplot(x,y,...)
         }
)
```


percent
8.56
9.3
9.5
5
5.98
6.9
7.76
0.26
1.57
2.8
3.94
error
distance

Aspect Ratio: aspect

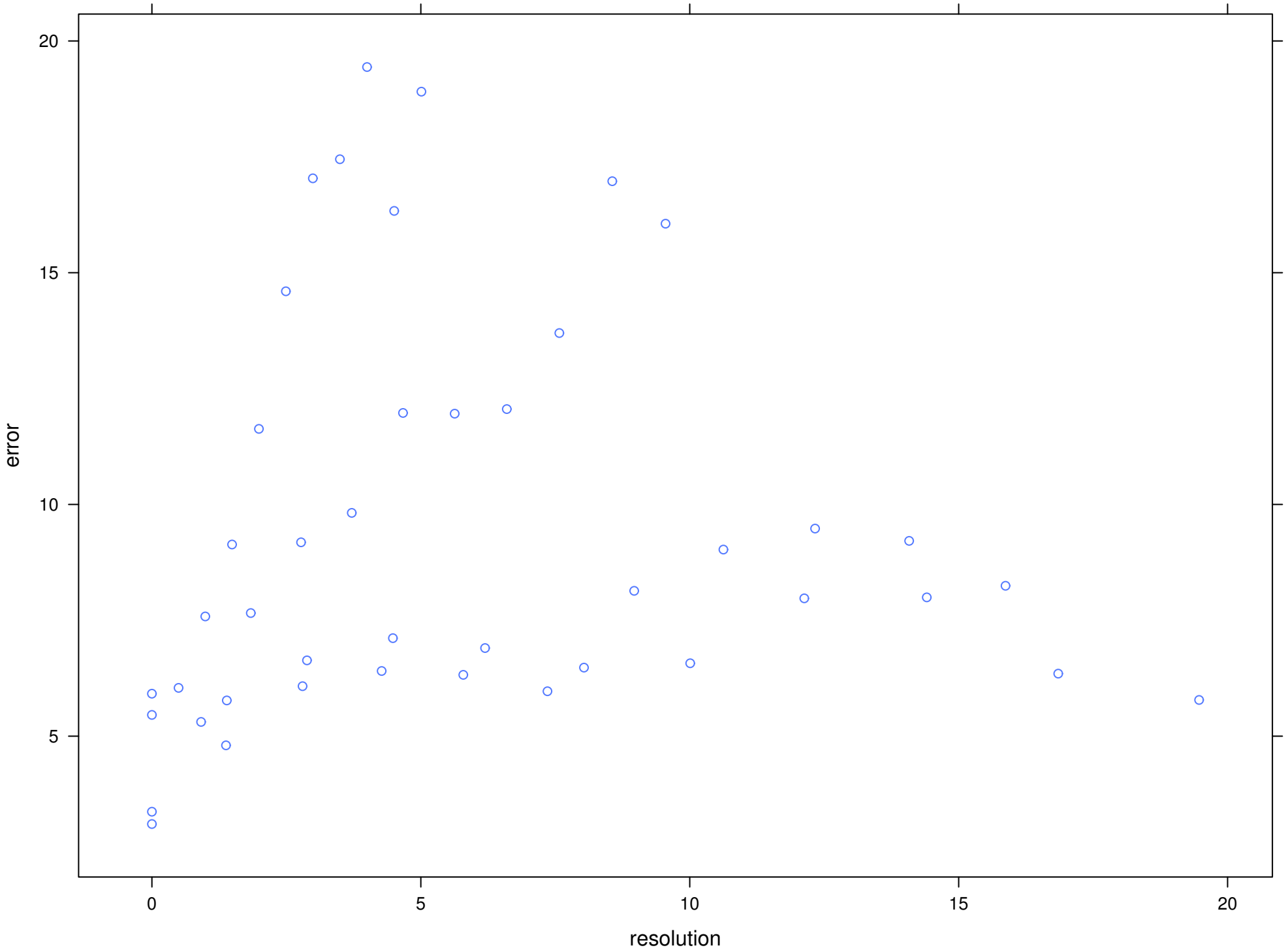
`number` controls the ratio of physical height to physical length.

`"fill"` default, tries to make the panels as big as possible to fill the available space.

`"xy"` computes the aspect ratio based on the 45 degree banking rule.

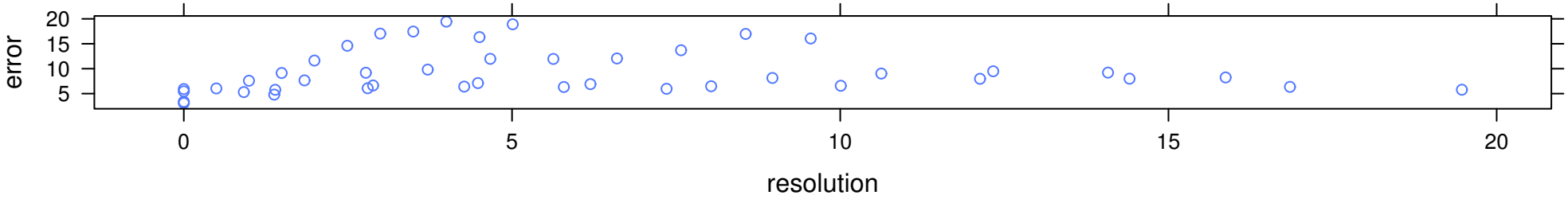
`"iso"` forces relation between physical distance and distance in the data scale are the same for both axes.

```
xyplot(error ~ resolution,
       data=slope,
       aspect = "fill",
)
```


20
15
10
5
error
0
5
10
15
20
resolution

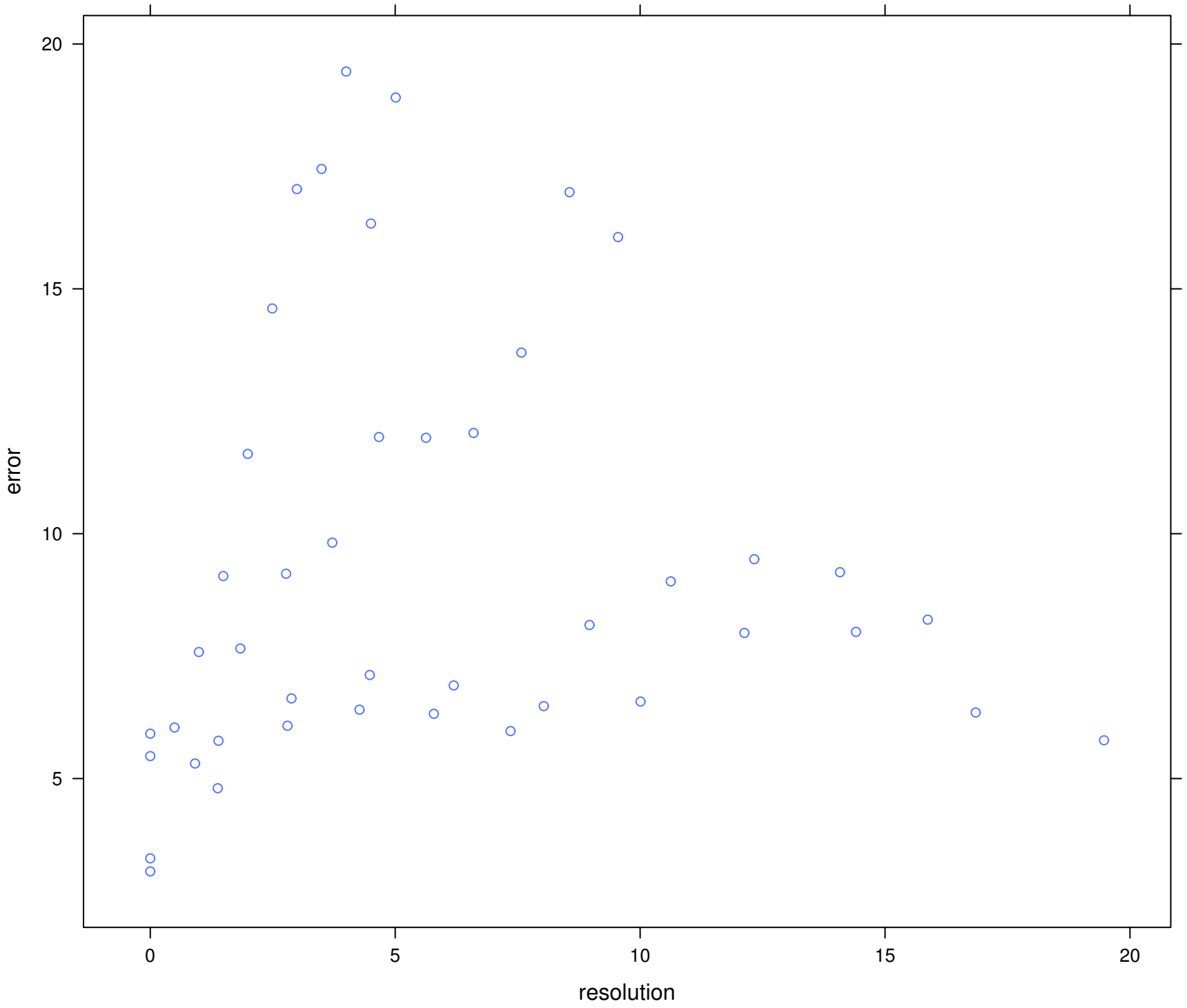
Aspect: ”xy”

```
xyplot(error ~ resolution,
        data=slope,
        aspect = "xy",
)
```


error
20
15
10
5
0
5
10
15
20
resolution

Aspect: ”iso”

```
xyplot(error ~ resolution,
        data=slope,
        aspect = "iso",
)
```


error
resolution
20
15
10
5
0
5
10
15
20

Prepanel: Prepanel.default

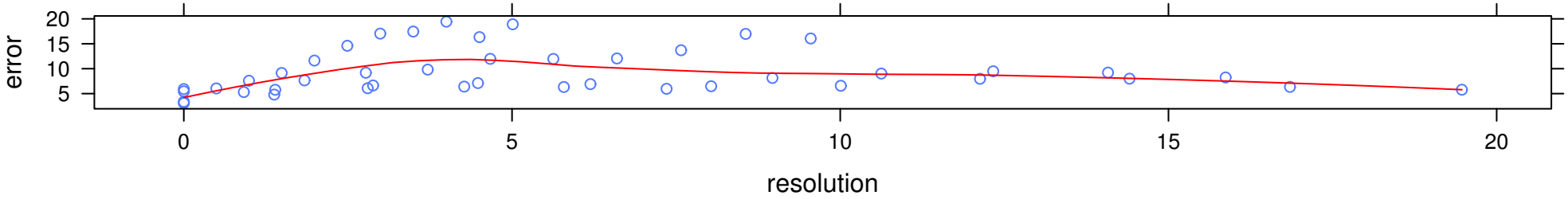
`Prepanel` A function that takes the same arguments as the panel function and returns a list, containing components named xlim, ylim, dx, and dy.

Every lattice graphic function has a default prepanel function named as:

```
prepanel.default.functionname
```

Prepanel: prepandel.default.xyplot

```
xyplot(error ~ resolution,
        data=slope,
        aspect = "xy",
        prepanel = function(x,y,...){
                prepanel.default.xyplot(x,y,...)
        },
        panel = function(x,y,...){
                panel.xyplot(x,y,...)
                panel.loess(x,y,span=2/3, degree=2,
                                col="red", ...)
        }
)
```


error
20
15
10
5
0
5
10
15
20
resolution

Prepanel: prepandel.default.xyplot

```
xyplot(error ~ resolution,
        data=slope,
        aspect = "xy",
        prepanel = function(x,y,...){
                a <- prepanel.default.xyplot(x,y,...)
                print(a)
        },
        panel = function(x,y,...){
                panel.xyplot(x,y,...)
                panel.loess(x,y,span=2/3, degree=2,
                                col="red", ...)
        }
)
```

Prepanel: prepandel.default.xyplot

```
$xlim
[1]  0.00000 19.47035

$ylim
[1]  3.104167 19.437500

$dx
 [1] 0.0000000 0.0000000 0.0000000 0.4953526 0.4189504
 [6] 0.0778560 0.3851590 0.0151460 0.0978860 0.3482900
[11] 0.1512160 0.5007510 0.2820770 0.0283220 0.0822070
[16] 0.1093170 0.5030190 0.2205400 0.2835080 0.2720060
[21] 0.2091870 0.0238000 0.1638890 0.3419800 0.6189900
[26] 0.1599810 0.4049100 0.4042260 0.7559930 0.2208100
[31] 0.4581610 0.5258490 0.4074310 0.5832790 0.4583910
[36] 0.6171700 1.5039100 0.2006800 1.7491200 0.3277300
[41] 1.4643000 0.9796300 2.6193300
```

Prepanel: prepandel.default.xyplot

```
$dy
 [1]   0.458336  -2.812499   0.265623   2.671876
 [5]  -0.734376   2.276040  -2.781250   0.968750
 [9]   3.364590  -1.479170   3.973960   2.968750
[13]   5.416669  -3.104166   0.557295  10.401040
[17]   0.411450  -7.630200   9.619790 -13.031250
[21]   0.708330   9.218750  -4.359370   6.932290
[25]  -6.947920  -5.635414   0.578124   5.156250
[29]  -6.088540   7.729170  -7.218754  10.494794
[33]  -8.838540   7.921860  -9.484360   2.453121
[37]  -1.052081   1.505210  -0.265629  -1.218751
[41]   0.250001  -1.895831  -0.567710

$xat
NULL

$yat
NULL
```

Prepanel: prepandel.default.xyplot

$$\frac{v_i}{h_i} = \frac{\ddot{v}_i}{\ddot{h}_i} \cdot (\frac{v/\ddot{v}}{h/\ddot{h}}) = \frac{v}{h} \cdot (\frac{\ddot{v}_i/\ddot{v}}{\ddot{h}_i/\ddot{h}})$$

v_i: length of vertical side in physical unit for i'th segment

h_i: length of horizontal side in physical unit for i'th segment

$\ddot{v}_i$: length of vertical side in data unit for i'th segment

$\ddot{h}_i$: length of horizontal side in data unit for i'th segment

v: length of vertical side in physical unit for whole data

h: length of horizontal side in physical unit for whole data

$\ddot{v}$: length of vertical side in data unit for whole data

$\ddot{h}$: length of horizontal side in data unit for whole data

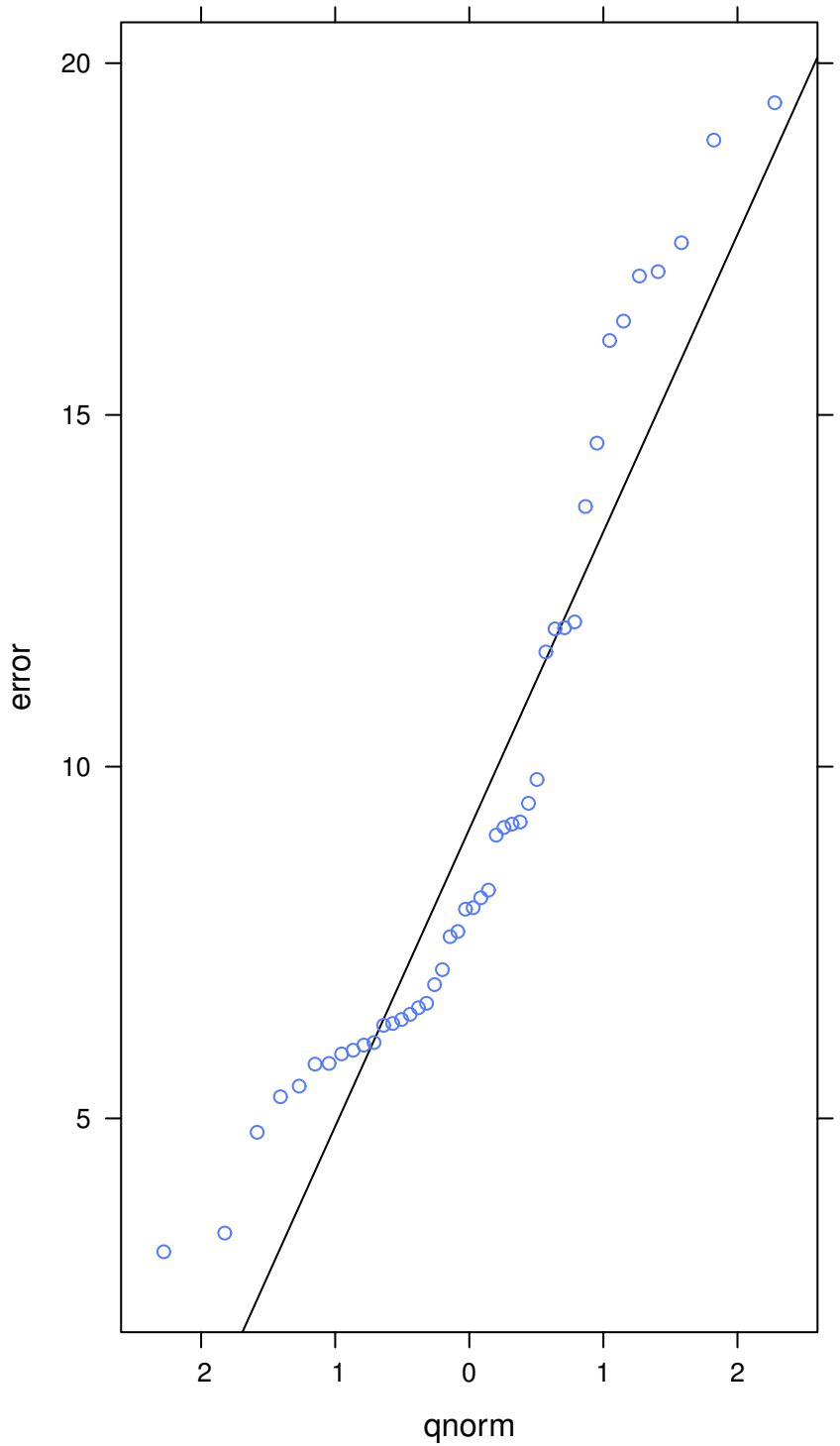
Prepanel: prepandel.default.xyplot

```
xyplot(error ~ resolution,
        data=slope,
        aspect = "xy",
        prepanel = function(x,y,...){
                a <- prepanel.default.xyplot(x,y,...)
                ydif <- diff(a$ylim)
                xdif <- diff(a$xlim)
                b <- abs((a$dy/a$dx)/(ydif/xdif))
                print(1/median(b))
        },
        panel = function(x,y,...){
                panel.xyplot(x,y,...)
                panel.loess(x,y,span=2/3, degree=2,
                        col="red", ...)
        }
)
```

```
[1] 0.05324232
```

Prepanel: prepandel.default.qqmath

```
qqmath(~ error,
        data=slope,
        aspect = "xy",
        prepanel = function(x,...){
                a <- prepanel.default.qqmath(x,...)
                print(a)
        },
        panel = function(x,...){
                panel.qqmathline(x,...)
                panel.qqmath(x,...)
        }
)
```


20
15
10
5
error
2
1
0
1
2
qnorm

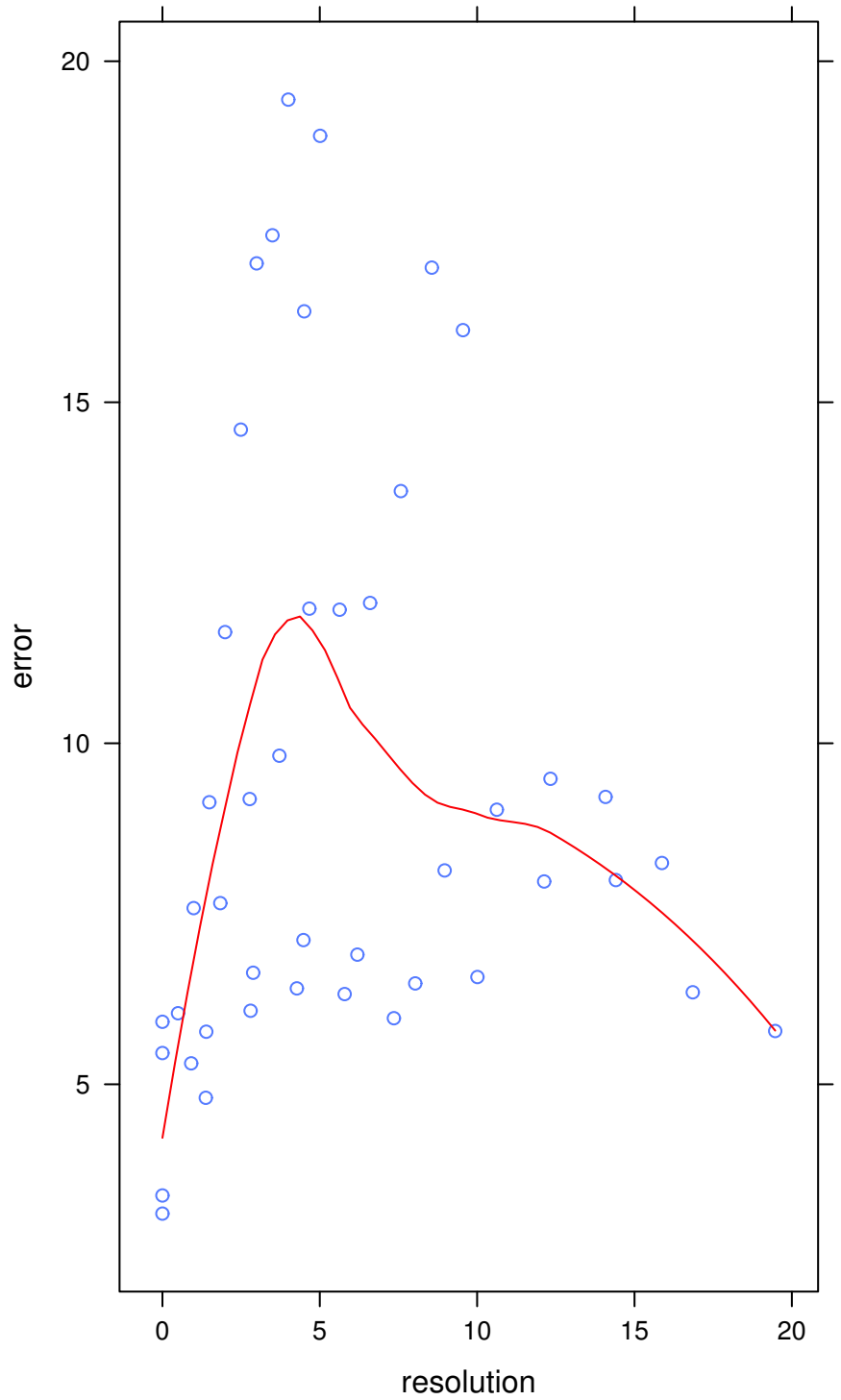
Prepanel: prepandel.default.qqmath

```
qqmath(~ error,
        data=slope,
        aspect = "xy",
        prepanel = function(x,...){
                a <- prepanel.default.qqmath(x,...)
                print(a)
                ydif <- diff(a$ylim)
                xdif <- diff(a$xlim)
                b <- abs((a$dy/a$dx)/(ydif/xdif))
                print(1/median(b))
        },
        panel = function(x,...){
                panel.qqmathline(x,...)
                panel.qqmath(x,...)
        }
)

[1] 1.879839
```

Prepanel: prepandel.loess

```
xyplot(error ~ resolution,
        data=slope,
        aspect = "xy",
        prepanel = function(x,y,...){
           a <- prepanel.loess(x,y,span=2/3,
                degree=2,...)
           print(a)
        },
        panel = function(x,y,...){
           panel.xyplot(x,y,...)
           panel.loess(x,y, span=2/3, degree=2,
                       col="red", ...)
        }
)
```


20
15
10
5
error
0
5
10
15
20
resolution

Prepanel: prepandel.loess

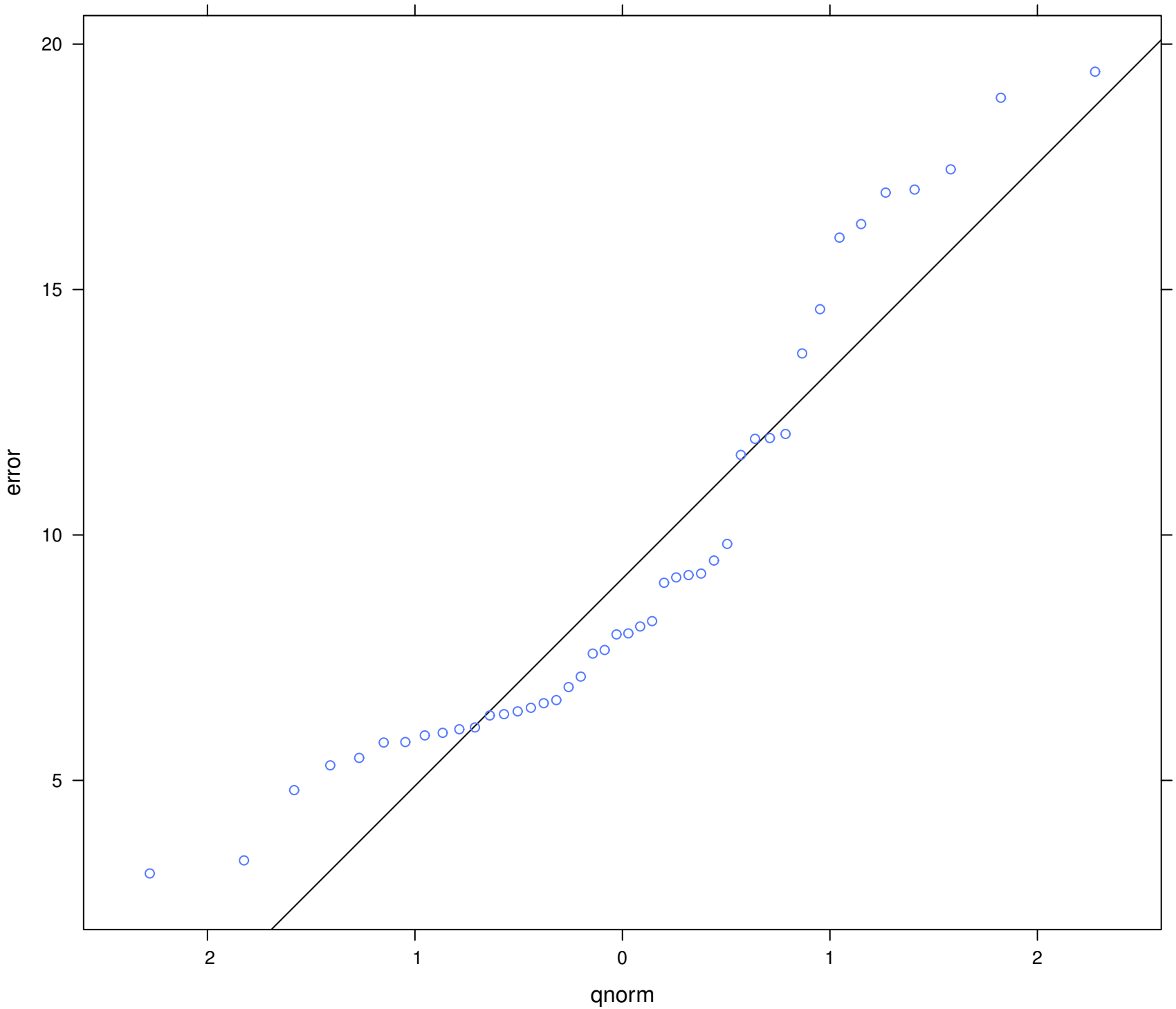
```
xyplot(error ~ resolution,
        data=slope,
        aspect = "xy",
        prepanel = function(x,y,...){
                a <- prepanel.loess(x,y,span=2/3,
                        degree=2,...)
                ydif <- diff(a$ylim)
                xdif <- diff(a$xlim)
                b <- abs((a$dy/a$dx)/(ydif/xdif))
                print(1/median(b))
        },
        panel = function(x,y,...){
                panel.xyplot(x,y,...)
                panel.loess(x,y,span=2/3,
                        degree=2, col="red", ...)
        }
)
```

```
[1] 1.818469
```

Prepanel: prepandel.qqmathline

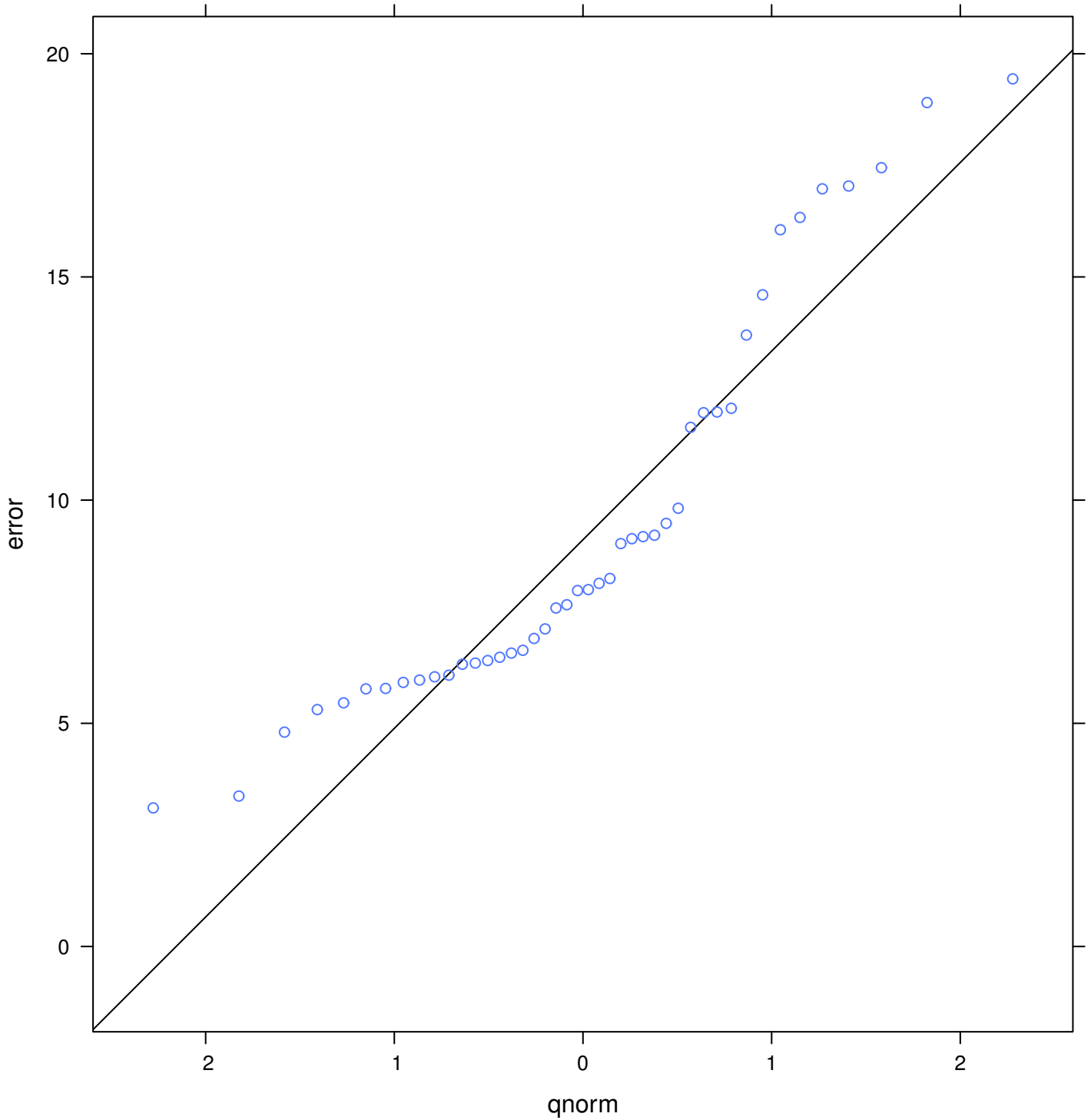
```
qqmath(~ error,
        data=slope,
        aspect = "xy",
        prepanel = function(x,...){
                a <- prepanel.qqmathline(x,...)
                print(a)
        },
        panel = function(x,...){
                panel.qqmathline(x,...)
                panel.qqmath(x,...)
        }
)

$xlim
[1] -2.277988  2.277988
$ylim
[1]  3.104167 19.437500
$dx
[1] 1.34898
$dy
[1] 5.700519
```


20
15
10
5
error
2
1
0
1
2
qnorm

Custom prepanel function

```
prepanel <- function(x,...){
        ans <- prepanel.qqmathline(x,...)
        theo.quant <- qnorm(c(.25,.75))
        data.quant <- quantile(x, c(.25,.75), names = F)
        slope <- ans$dy / ans$dx
        intercept <- data.quant[1] - slope*theo.quant[1]
        yylim <-  slope*(ans$xlim)+ intercept
        ans$ylim <- range(ans$ylim, yylim)
        ans
}
qqmath(~ error,
        data=slope,
        aspect = "xy",
        prepanel = function(x,...){
                prepanel(x)
},
        panel = function(x,...){
                panel.qqmathline(x,...)
                panel.qqmath(x,...)
        }
)
```


20
15
10
5
0
error
2
1
0
1
2
qnorm