# Learning to Draw

**Ken Rice**
**Thomas Lumley**

UW Biostatistics

*Seattle, June 2009*

# Graphics

R can produce graphics in many formats, including:

- on screen
- PDF files for LaTeX or emailing to people
- PNG or JPEG bitmap formats for web pages (or on non-Windows platforms to produce graphics for MS Office). PNG is also useful for graphs of large data sets.
- On Windows, metafiles for Word, Powerpoint, and similar programs

# Setup

Graphs should usually be designed on the screen and then may be replotted on eg a PDF file (for Word/Powerpoint you can just copy and paste)

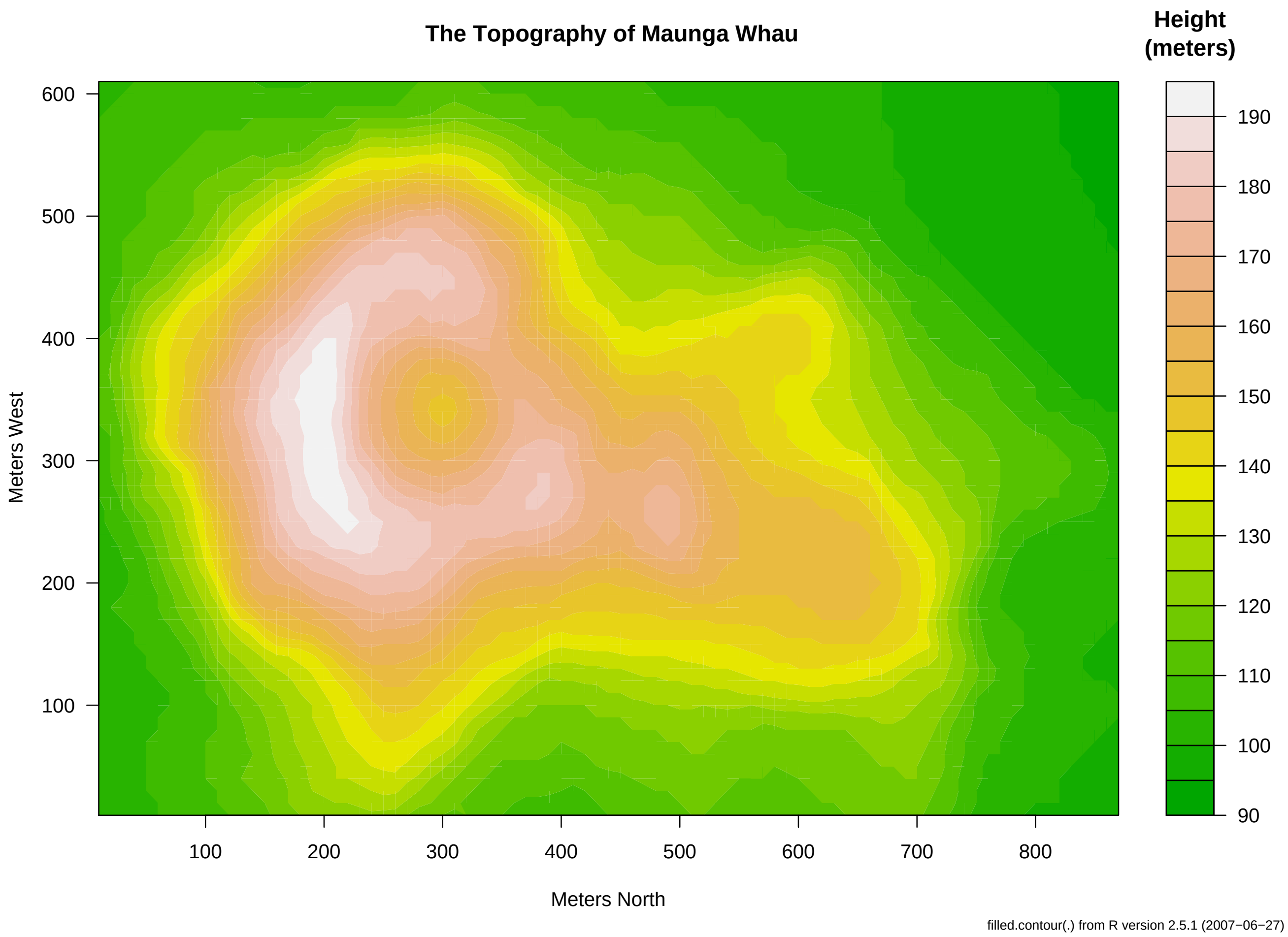
For printed graphs, you will get better results if you design the graph at the size it will end up, eg:

```
## on Windows
windows(height=4,width=6)
## on Unix
x11(height=4,width=6)
```

Word or LaTeX can rescale the graph, but when the graph gets smaller, so do the axis labels...

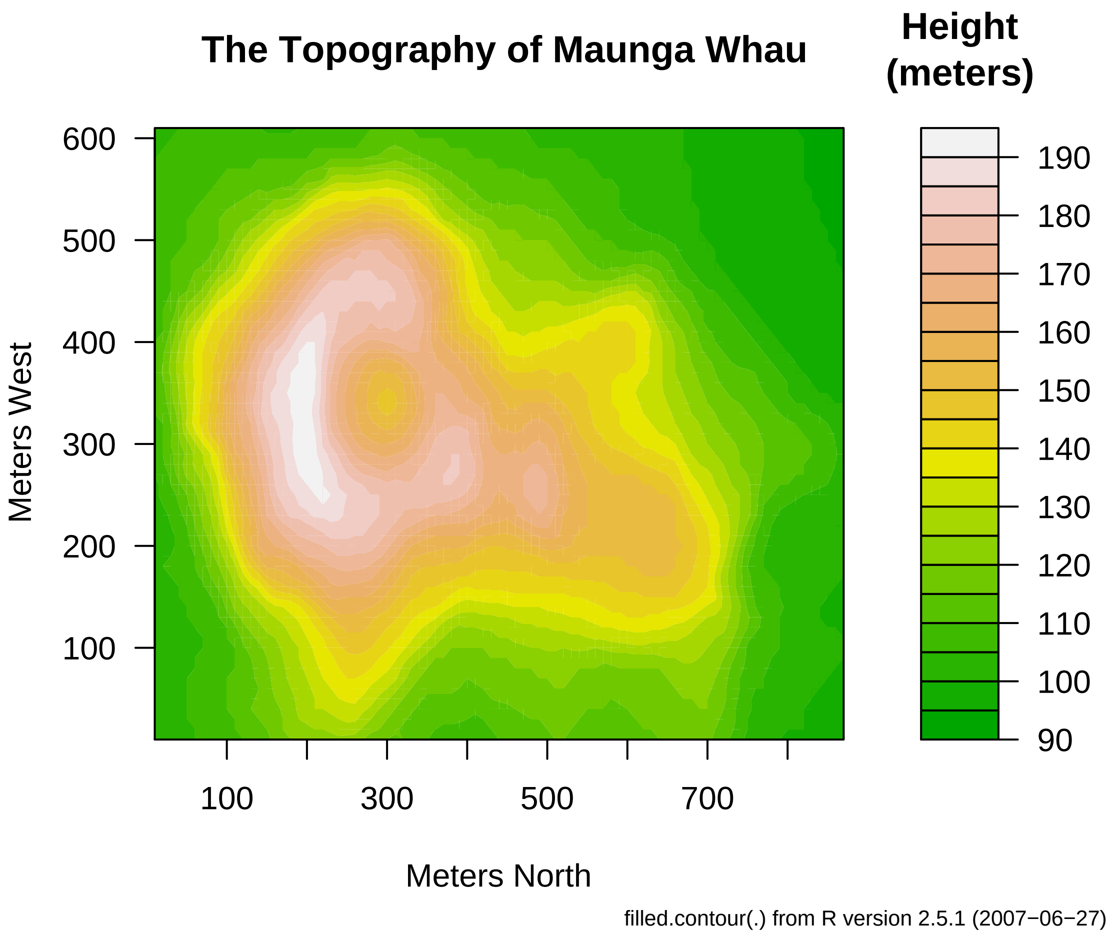
# Setup

Created at full-page size



# Setup

Created at 5x6 inches



# Finishing

After you have the right commands to draw the graph you can produce it in another format: eg

```
## start a PDF file
pdf("picture.pdf",height=4,width=6)
## your drawing commands here
...
### close the PDF file
dev.off()
```

# Drawing

Usually use plot() to create a graph and then lines(), points(), legend(), text(), and other commands to annotate it.

plot() is a generic function: it does appropriate things for different types of input

```
## scatterplot
plot(salary$year, salary$salary)
## boxplot
plot(salary$rank, salary$salary)
## stacked barplot
plot(salary$field, salary$rank)
```

and others for other types of input.

# Formula interface

The plot() command can be written

```
plot(salary~rank, data=salary)
```

introducing the formula system that is also used for regression models. The variables in the formula are automatically looked up in the data= argument.

# Designing graphs

Two important aspects of designing a graph

- It should have something to say
- It should be legible

Having something to say is your problem; software can help with legibility.

# Designing graphs

Important points

- Axes need labels (with units, large enough to read)
- Color can be very helpful (but not if the graph is going to be printed in black and white).
- Different line or point styles usually should be labelled.
- Points plotted on top of each other won't be seen

After these are satisfied, it can't hurt to have the graph look nice.

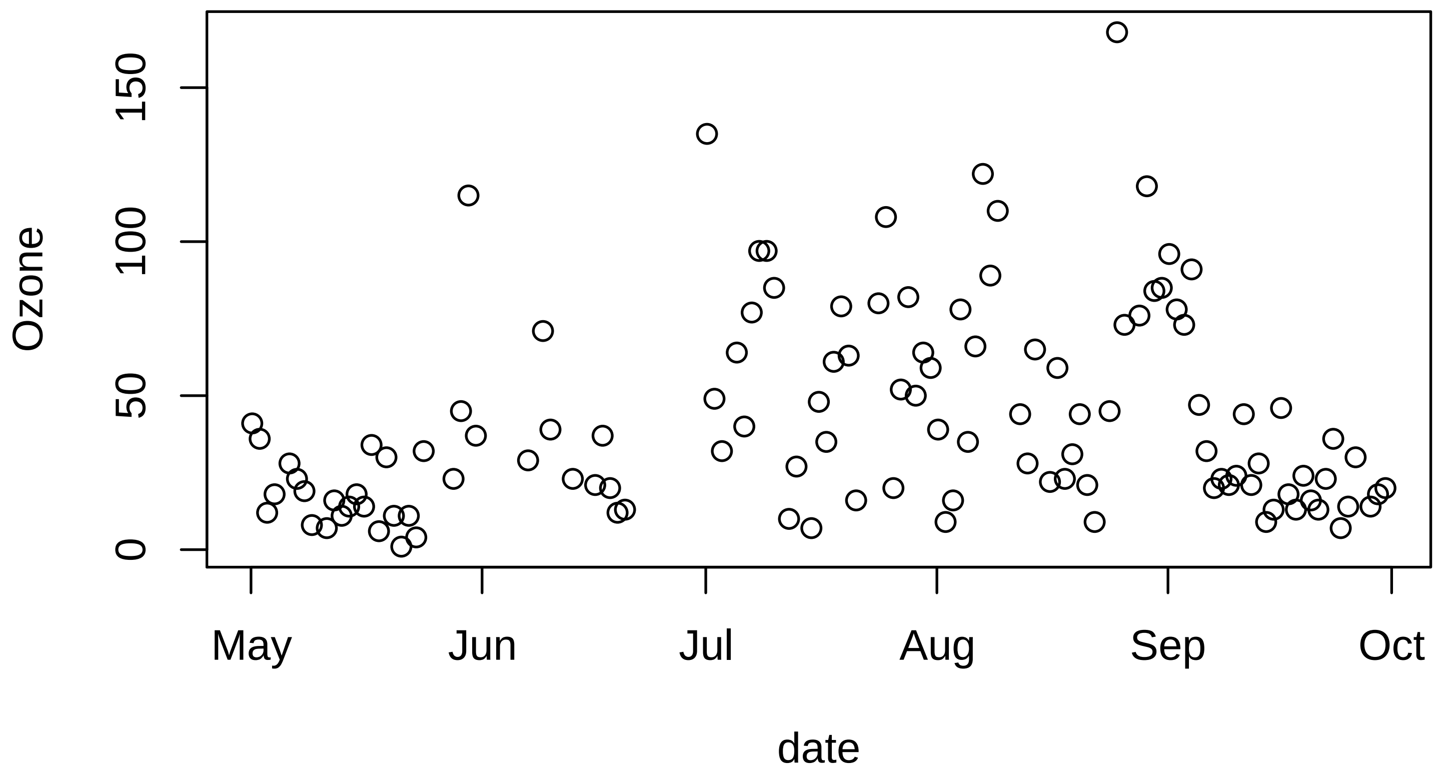
# Options

Set up a data set: daily ozone concentrations in New York, summer 1973

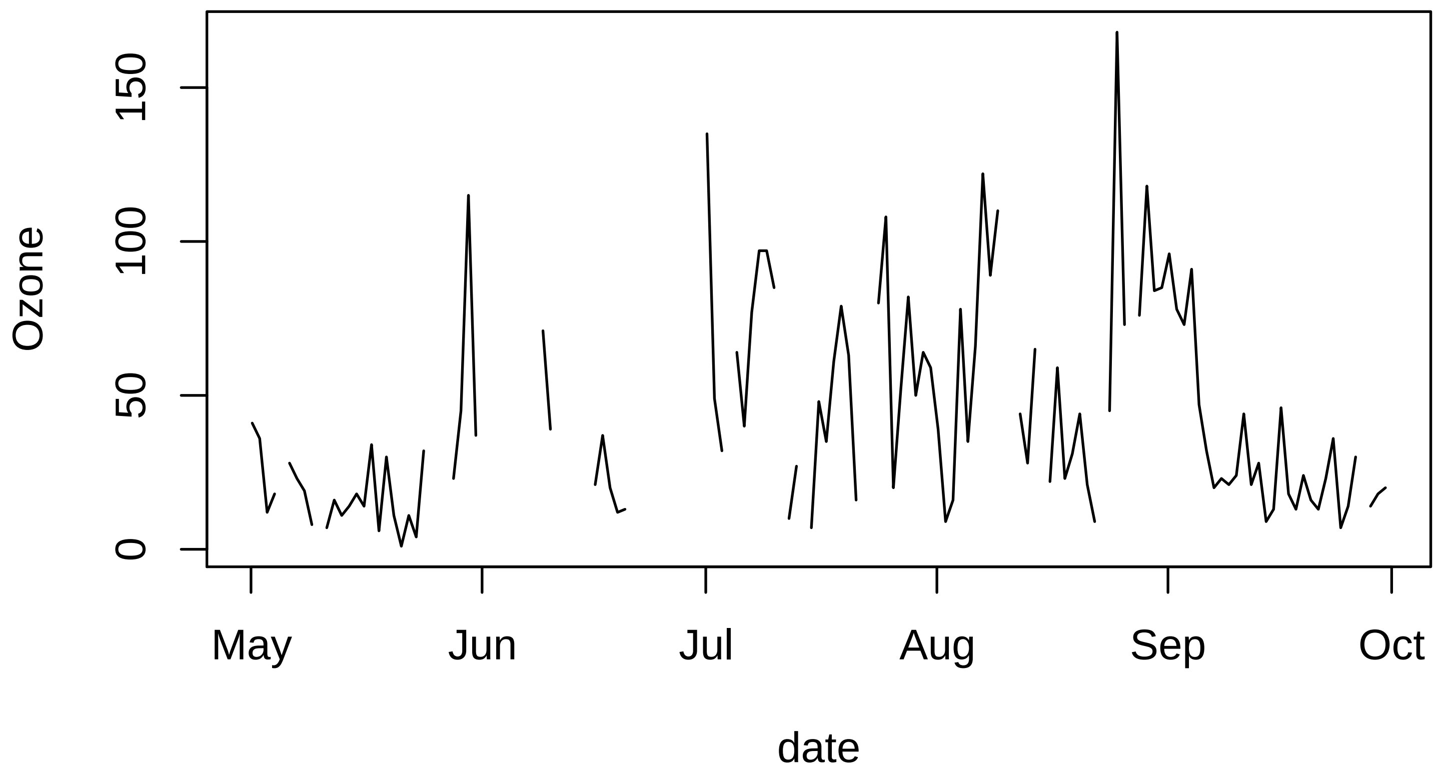
```
data(airquality)
names(airquality)
airquality$date<-with(airquality, ISOdate(1973,Month,Day))
```

All these graphs were designed at 4in$\times$6in and stored as PDF files

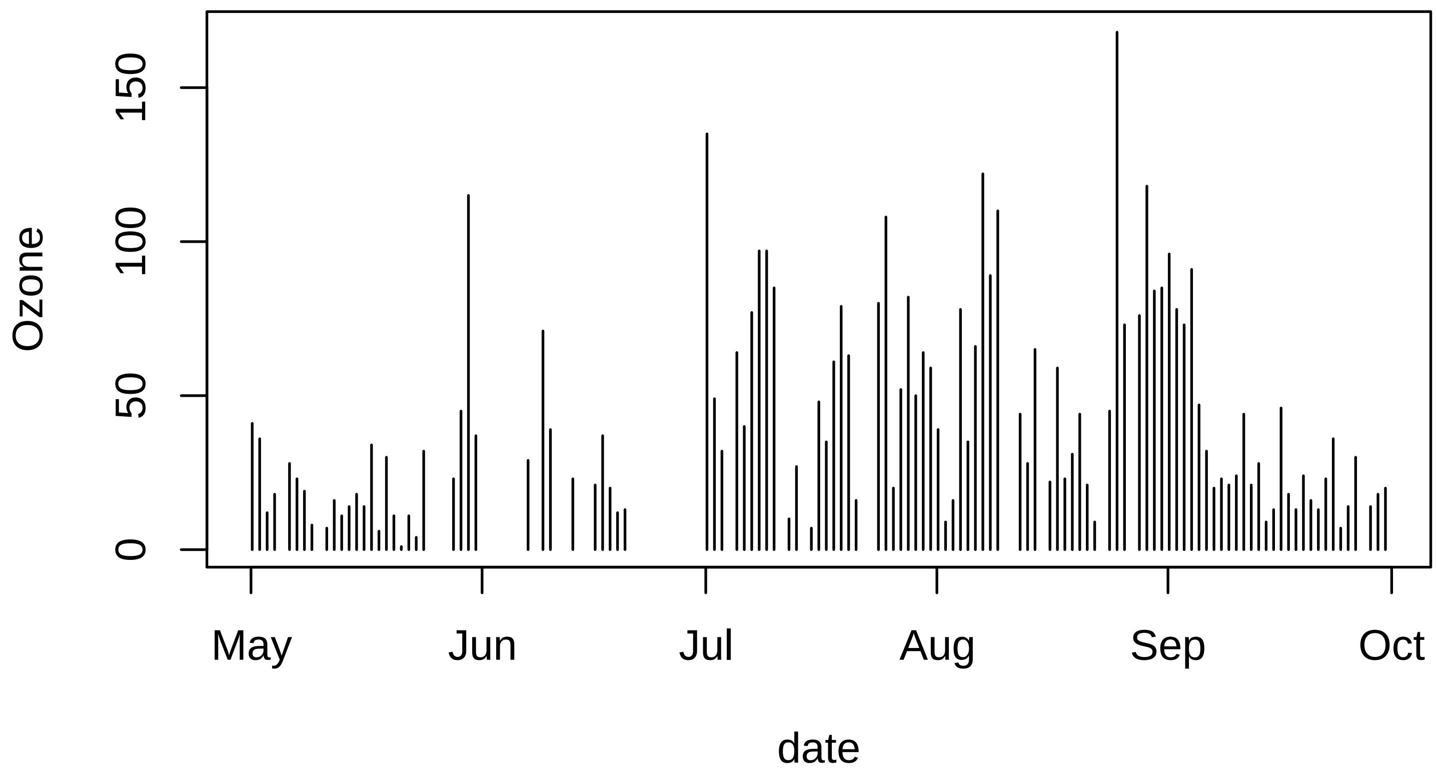
```
plot(Ozone~date, data=airquality)
```



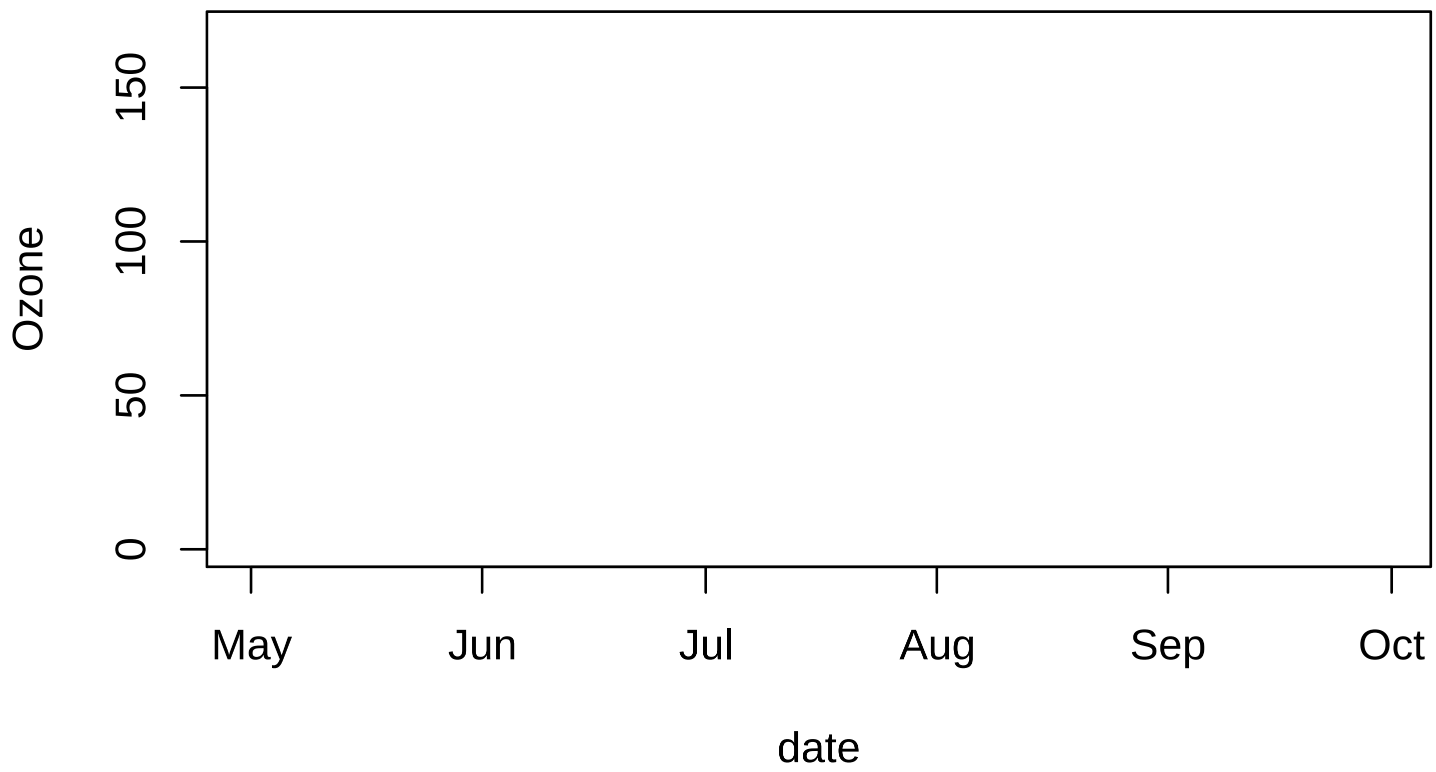
```
plot(Ozone~date, data=airquality,type="l")
```



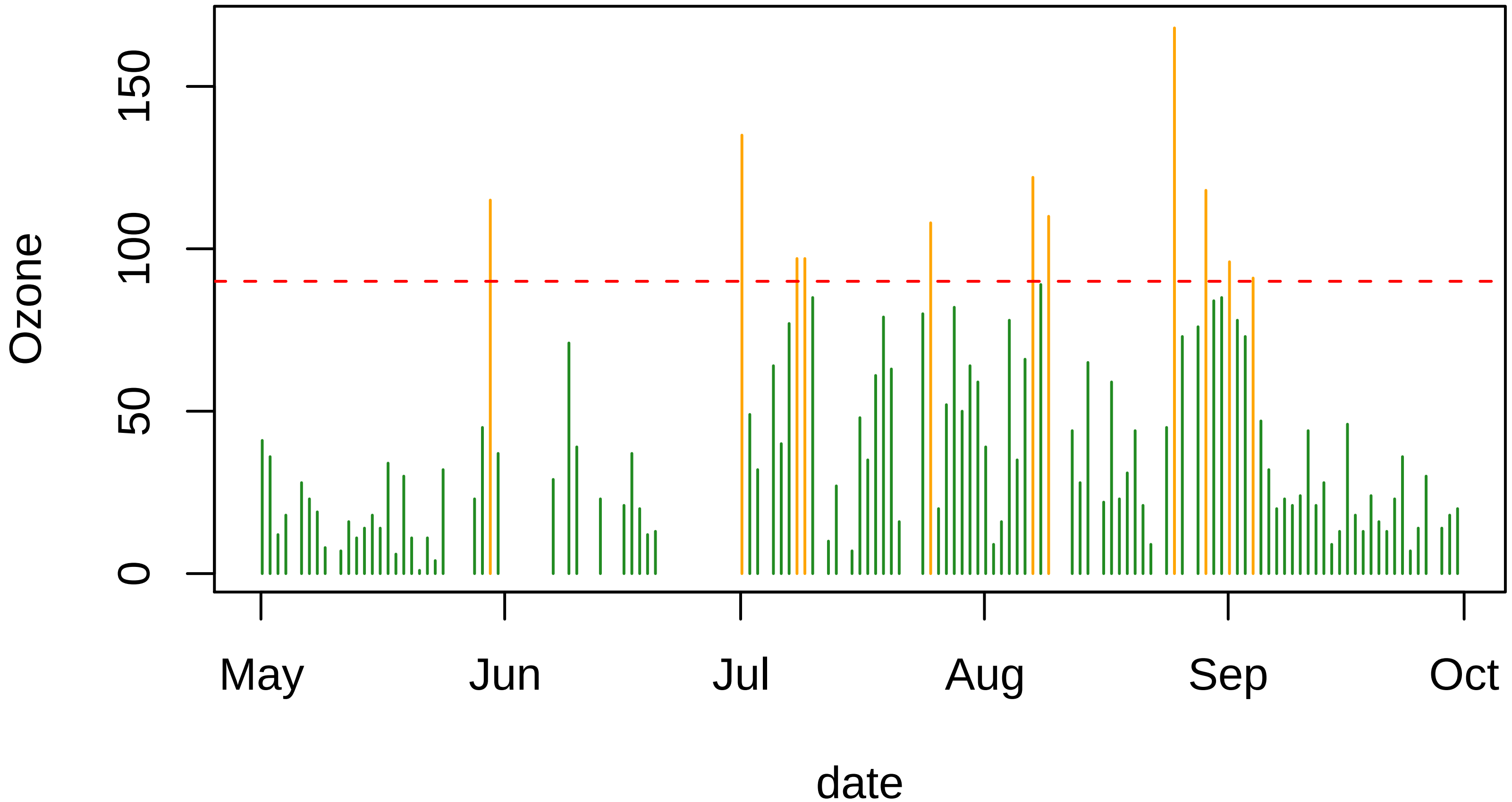
```
plot(Ozone~date, data=airquality,type="h")
```



```
plot(Ozone~date, data=airquality,type="n")
```



```
bad<-ifelse(airquality$Ozone>=90, "orange","forestgreen")
plot(Ozone~date, data=airquality,type="h",col=bad)
abline(h=90,lty=2,col="red")
```


Ozone
0
50
100
150
May
Jun
Jul
Aug
Sep
Oct
date

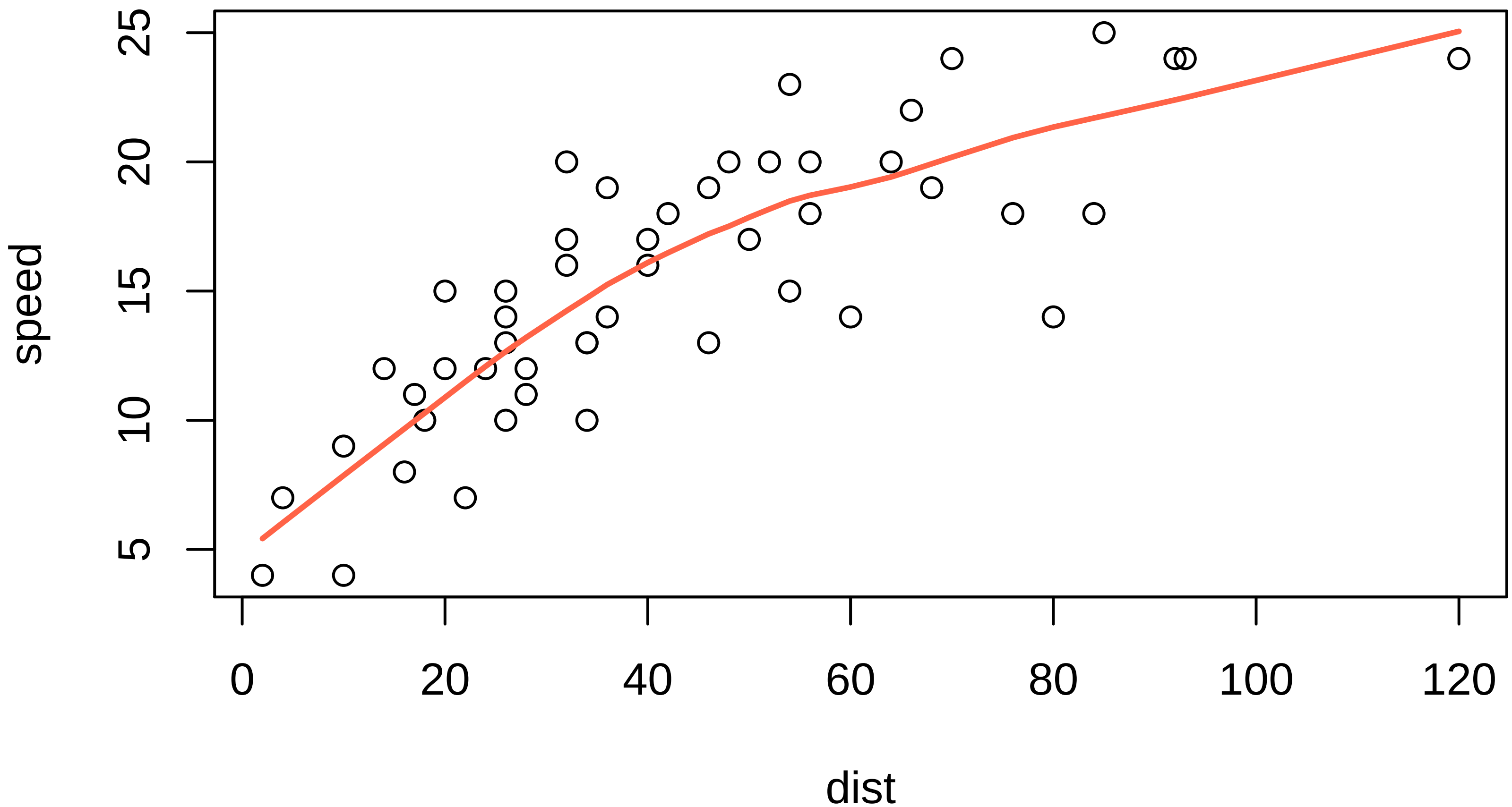
# Notes

- `type=` controls how data are plotted. `type="n"` is not as useless as it looks: it can set up a plot for latter additions.
- Colors can be specified by name (the `colors()` function gives all the names), by red/green/blue values (#rrggbb with six base-sixteen digits) or by position in the standard palette of 8 colors.
- `abline` draws a single straight line on a plot
- `ifelse()` selects between two vectors based on a logical variable.
- `lty` specifies the line type: 1 is solid, 2 is dashed, 3 is dotted, then it gets more complicated.

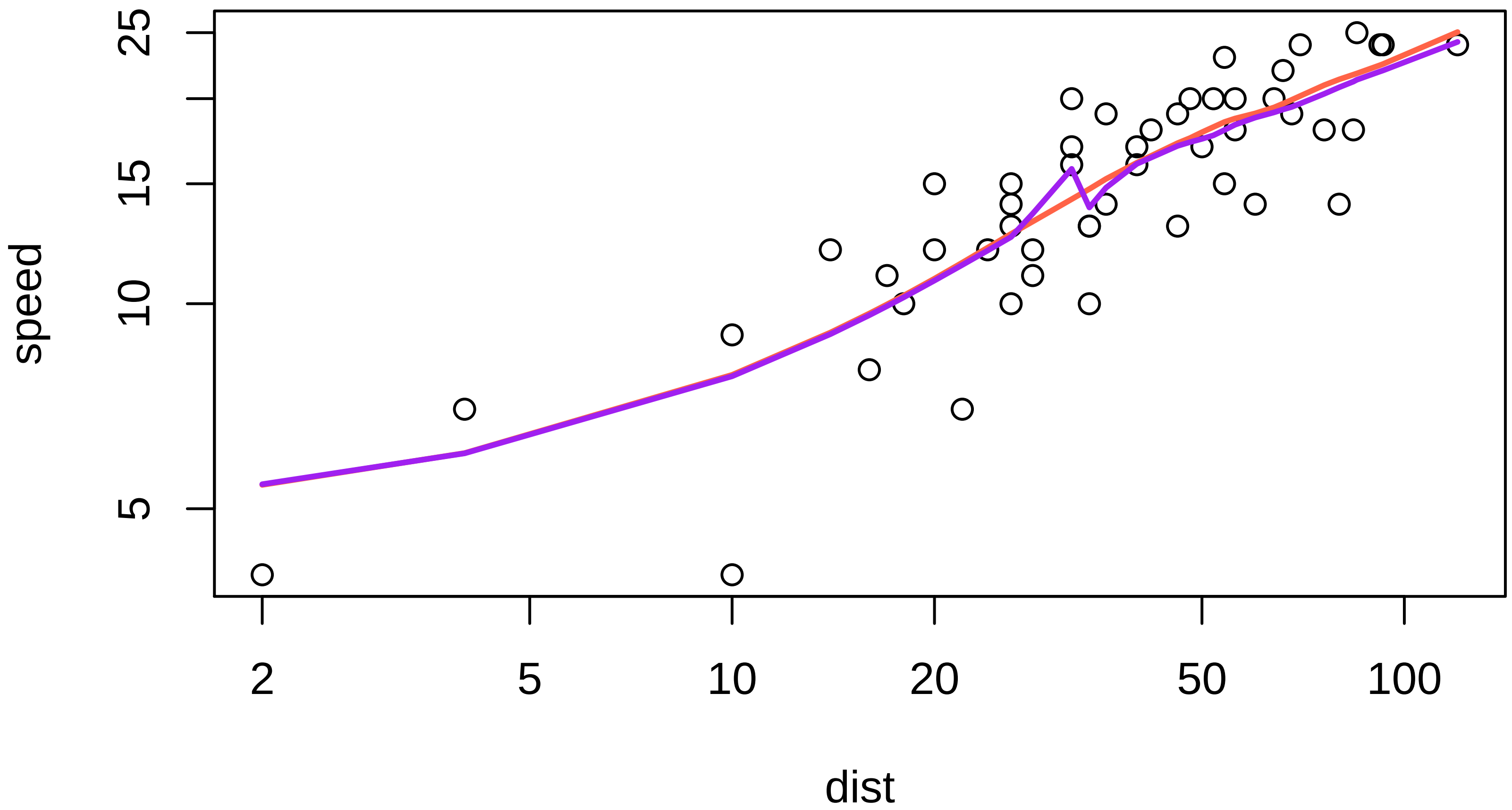
# Adding to a plot

```
data(cars)
plot(speed~dist,data=cars)
with(cars, lines(lowess(dist,speed), col="tomato", lwd=2))
plot(speed~dist,data=cars, log="xy")
with(cars, lines(lowess(dist,speed), col="tomato", lwd=2))
with(cars, lines(supsmu(dist,speed), col="purple", lwd=2))
legend(2,25, legend=c("lowess","supersmoother"),bty="n", lwd=2,
  col=c("tomato","purple"))
```

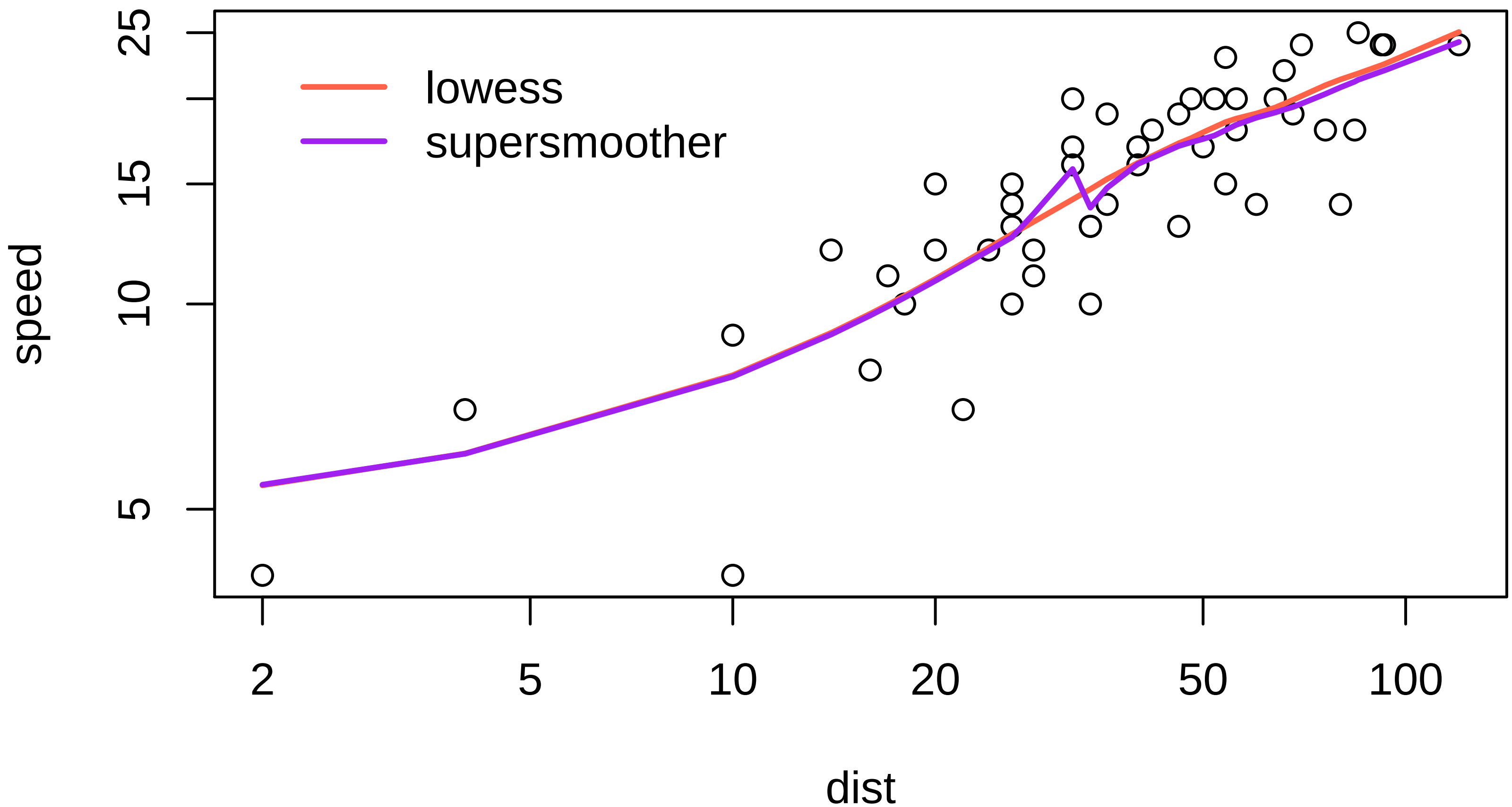
# Adding to a plot



# Adding to a plot



# Adding to a plot



# Notes

- `lines` adds lines to an existing plot (`points()` adds points).

- `lowess()` and `supsmu()` are scatterplot smoothers. They draw smooth curves that fit the relationship between $y$ and $x$ locally.

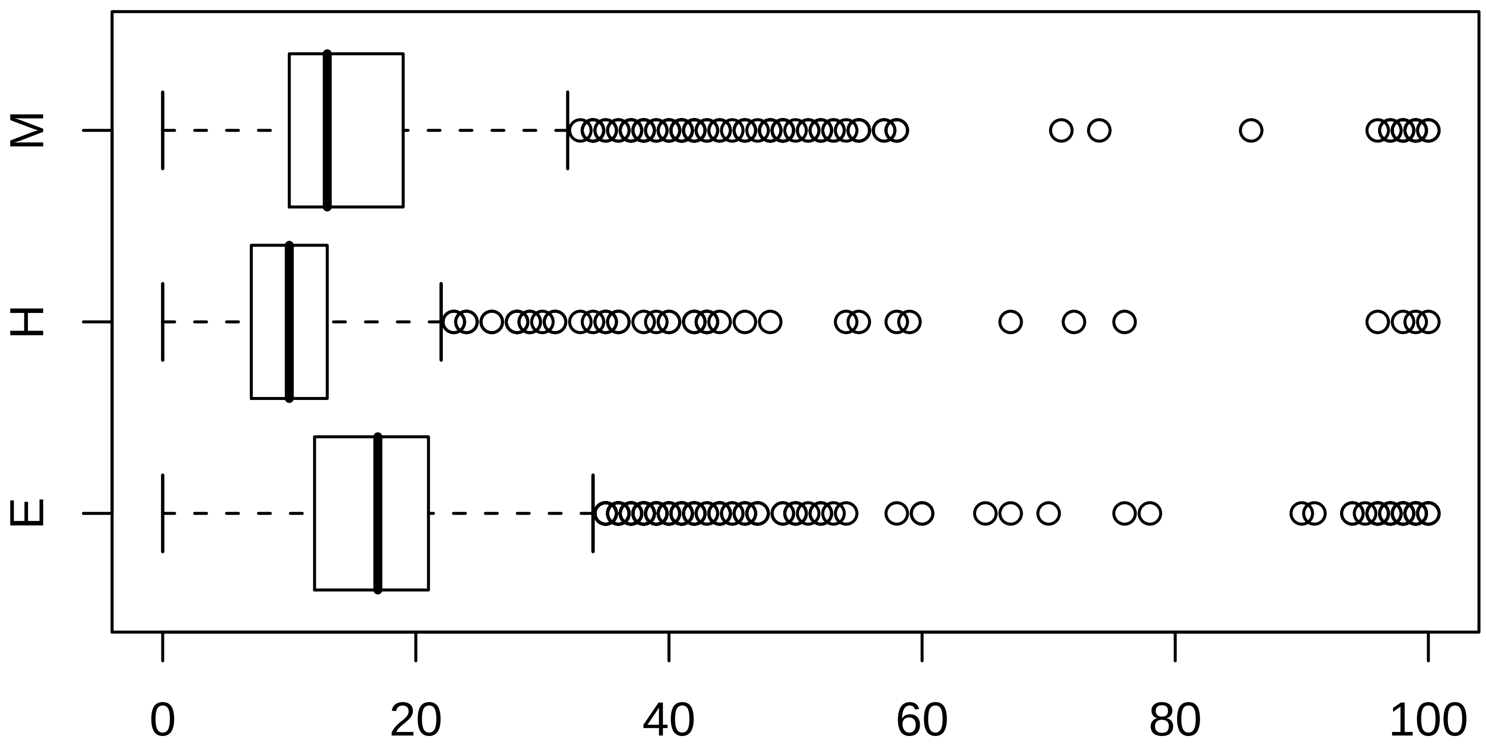
- `log="xy"` asks for both axes to be logarithm (`log="x"` would just be the x-axis)

- `legend()` adds a legend

# Boxplots

```
data(api, package="survey")
boxplot(mobility~stype,data=apipop, horizontal=TRUE)
```

# Boxplots



# Notes

- `boxplot` computes and draws boxplots.

- `horizontal=TRUE` turns a boxplot sideways

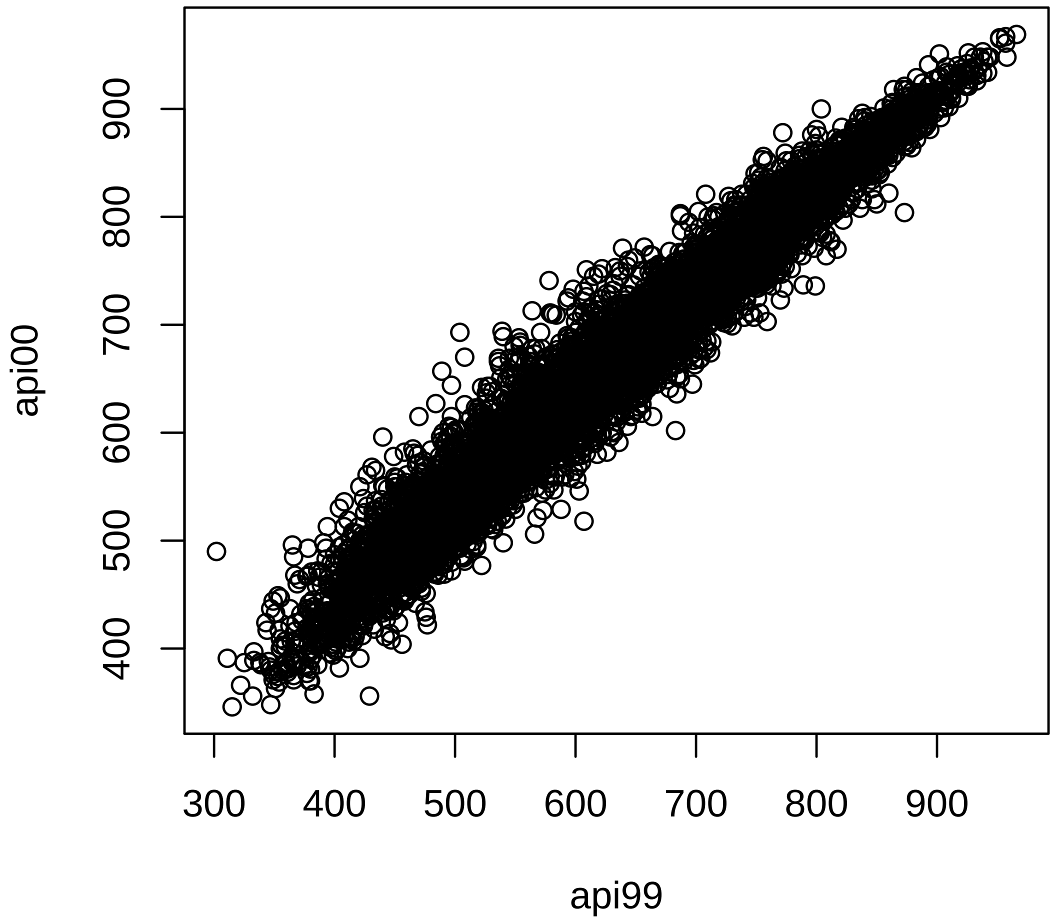
- `xlab` and `ylab` are general options for x and y axis labels.

# Large data sets

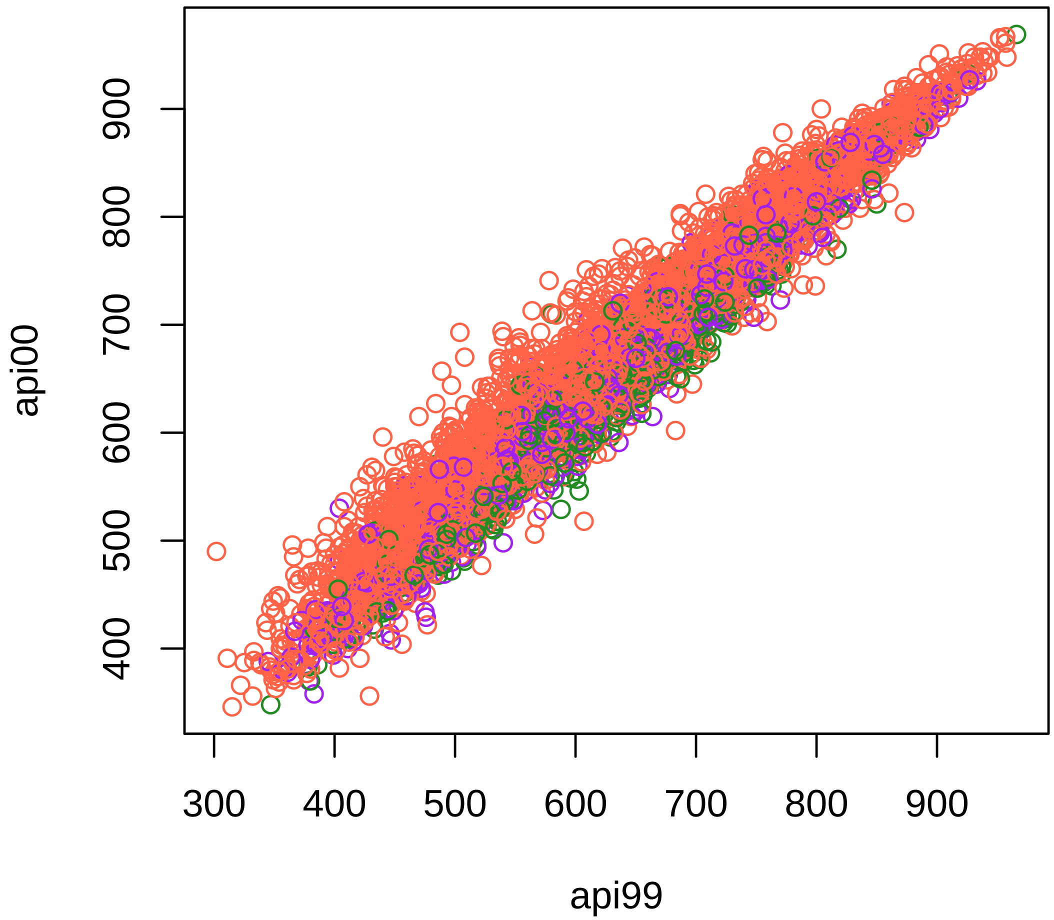
Scatterplots quickly get crowded. For example, the California Academic Performance Index is reported on 6194 schools

```
> plot(api00~api99,data=apipop)
> colors<-c("tomato","forestgreen","purple")[apipop$stype]
> plot(api00~api99,data=apipop,col=colors)
```

# Large data sets



# Large data sets



# Density plots

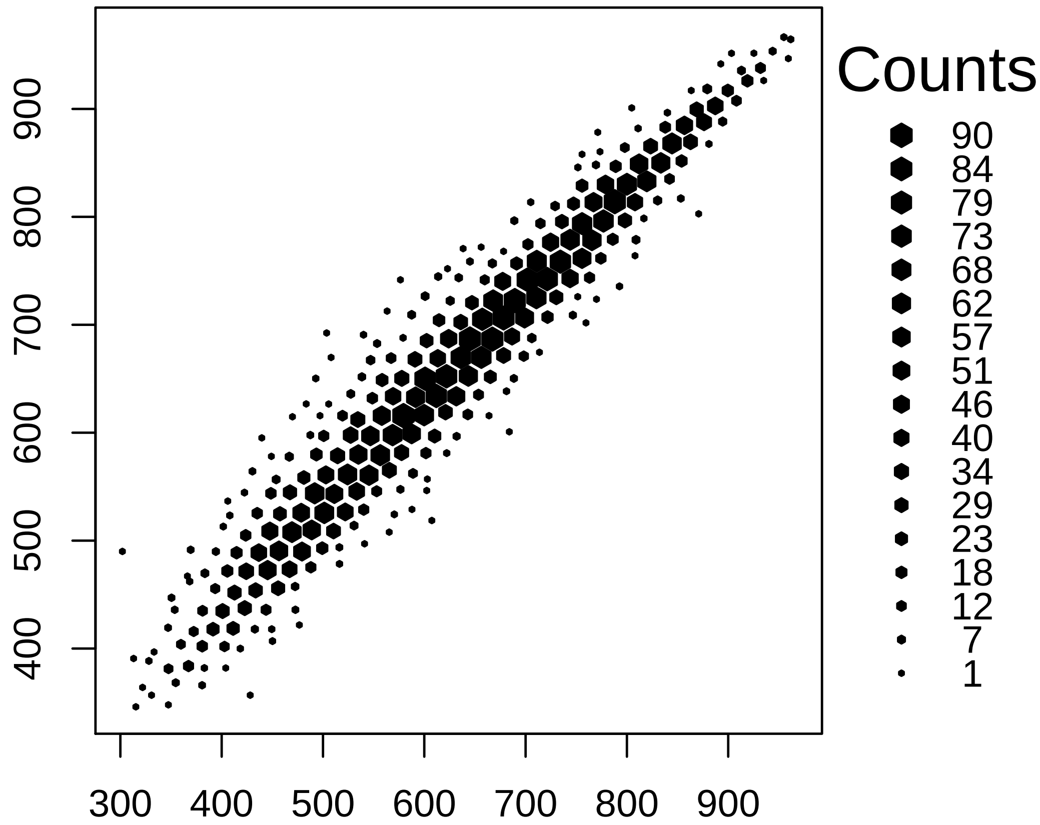
For a single large scatterplot some form of aggregation is useful

```
library(hexbin)
with(apipop, plot(hexbin(api99,api00), style="centroids"))
```

`hexbin` is in the hexbin package from the Bioconductor project. It computes the number of points in each hexagonal bin.

The `style=centroids` plot puts a filled hexagon with size depending on the number of points at the centroid of the points in the bin.

# Density plots

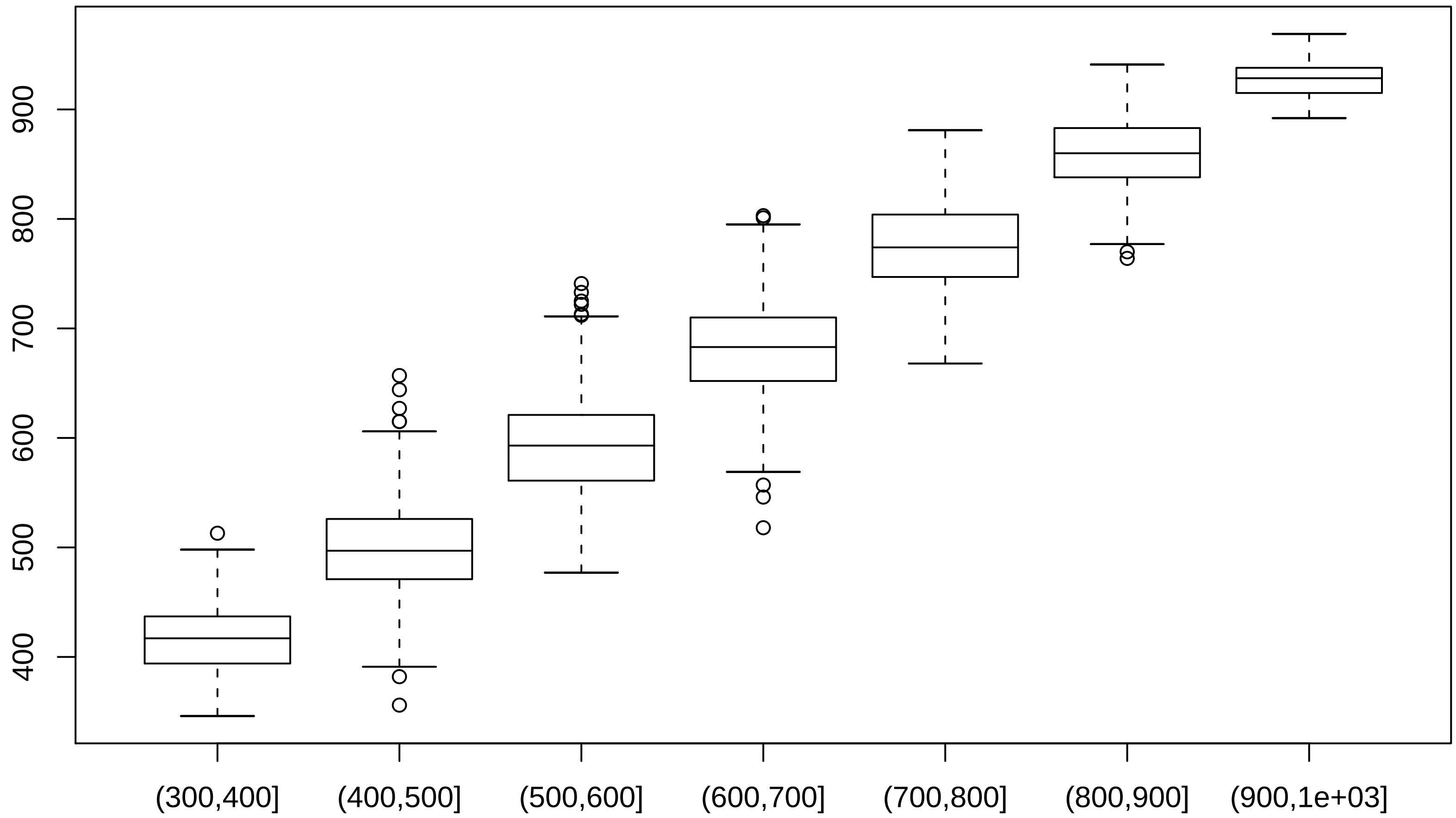


# Smoothers

For showing multiple groups a scatterplot smoother or perhaps boxplots would be better.

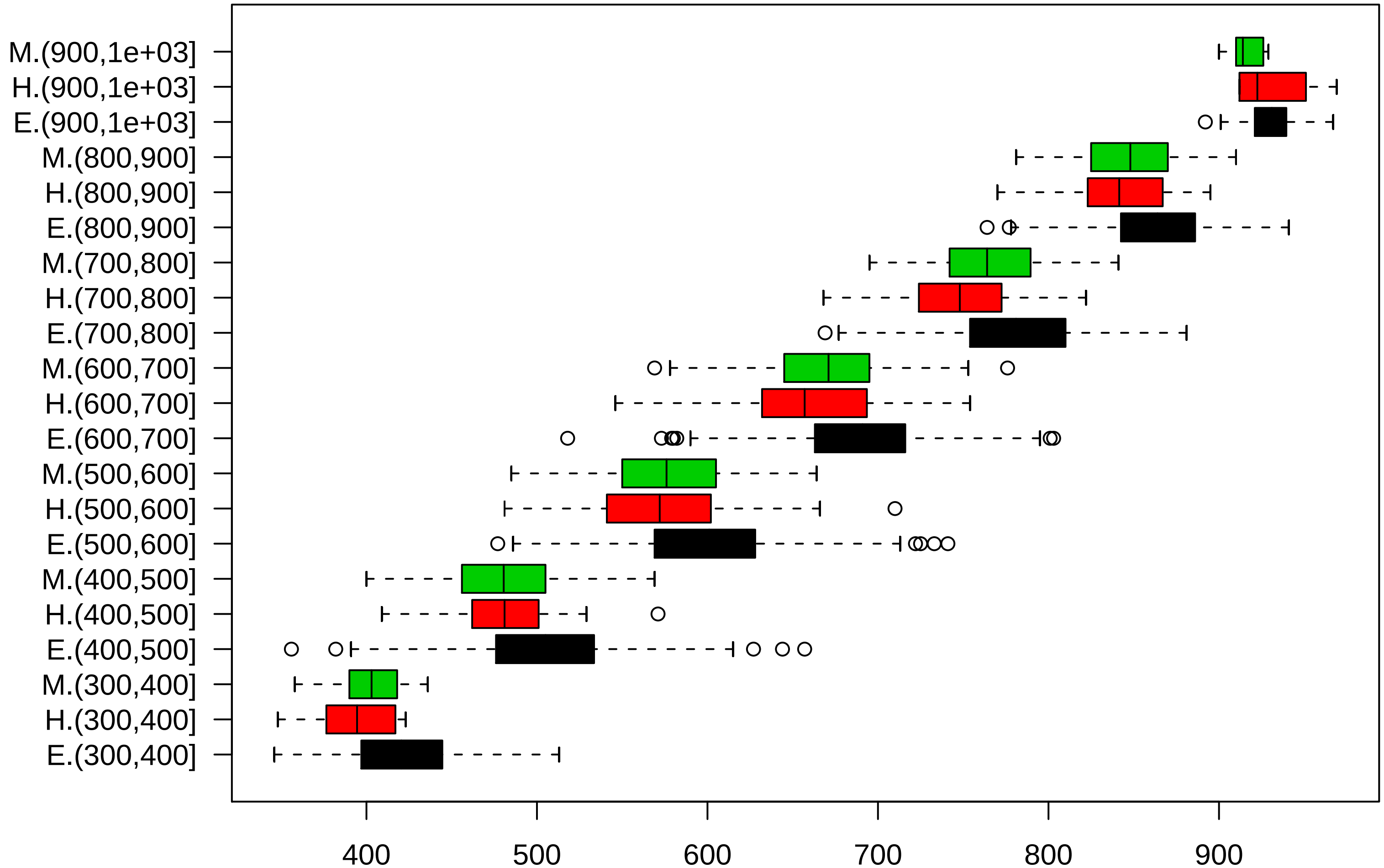
```
> boxplot(api00~cut(api99,(3:10)*100), data=apipop)
> par(las=1)
> par(mar=c(5.1,10.1,2.1,2.1))
> boxplot(api00~interaction(stype,
                           cut(api99,(3:10)*100)),
        data=apipop, horizontal=TRUE,col=1:3)
 plot(api00~api99,data=apipop,type="n")
 with(subset(apipop, stype=="E"),
      lines(lowess(api99, api00), col="tomato"))
 with(subset(apipop, stype=="H"),
      lines(lowess(api99, api00), col="forestgreen"))
 with(subset(apipop, stype=="M"),
      lines(lowess(api99, api00), col="purple"))
```

# Smoothers



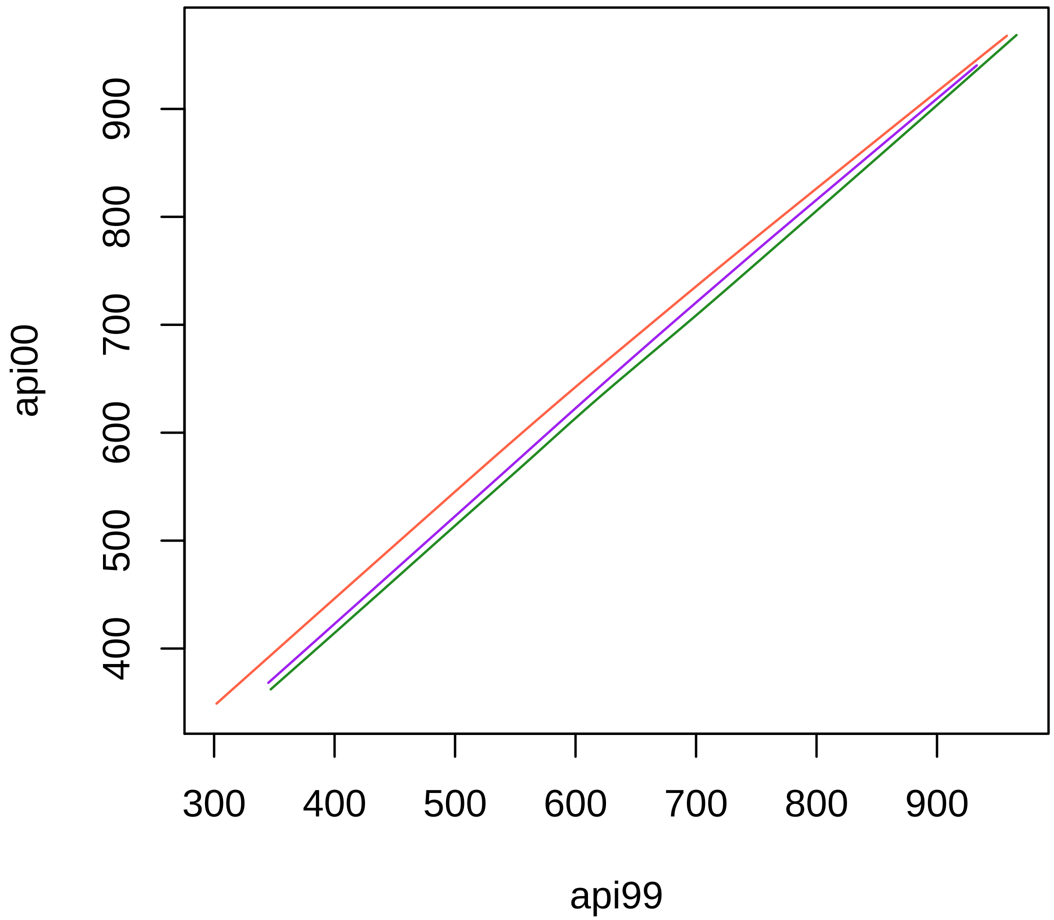
900
800
700
600
500
400
(300,400]
(400,500]
(500,600]
(600,700]
(700,800]
(800,900]
(900,1e+03]

# Smoothers



M.(900,1e+03]
H.(900,1e+03]
E.(900,1e+03]
M.(800,900]
H.(800,900]
E.(800,900]
M.(700,800]
H.(700,800]
E.(700,800]
M.(600,700]
H.(600,700]
E.(600,700]
M.(500,600]
H.(500,600]
E.(500,600]
M.(400,500]
H.(400,500]
E.(400,500]
M.(300,400]
H.(300,400]
E.(300,400]
400
500
600
700
800
900

# Smoothers



# Notes

- `cut` turns a variable into a factor by cutting it at the specified points.

- Note the use of `type="n"`

- `par(mar=)` sets the margins around the plot. We need a large left margin for the labels.

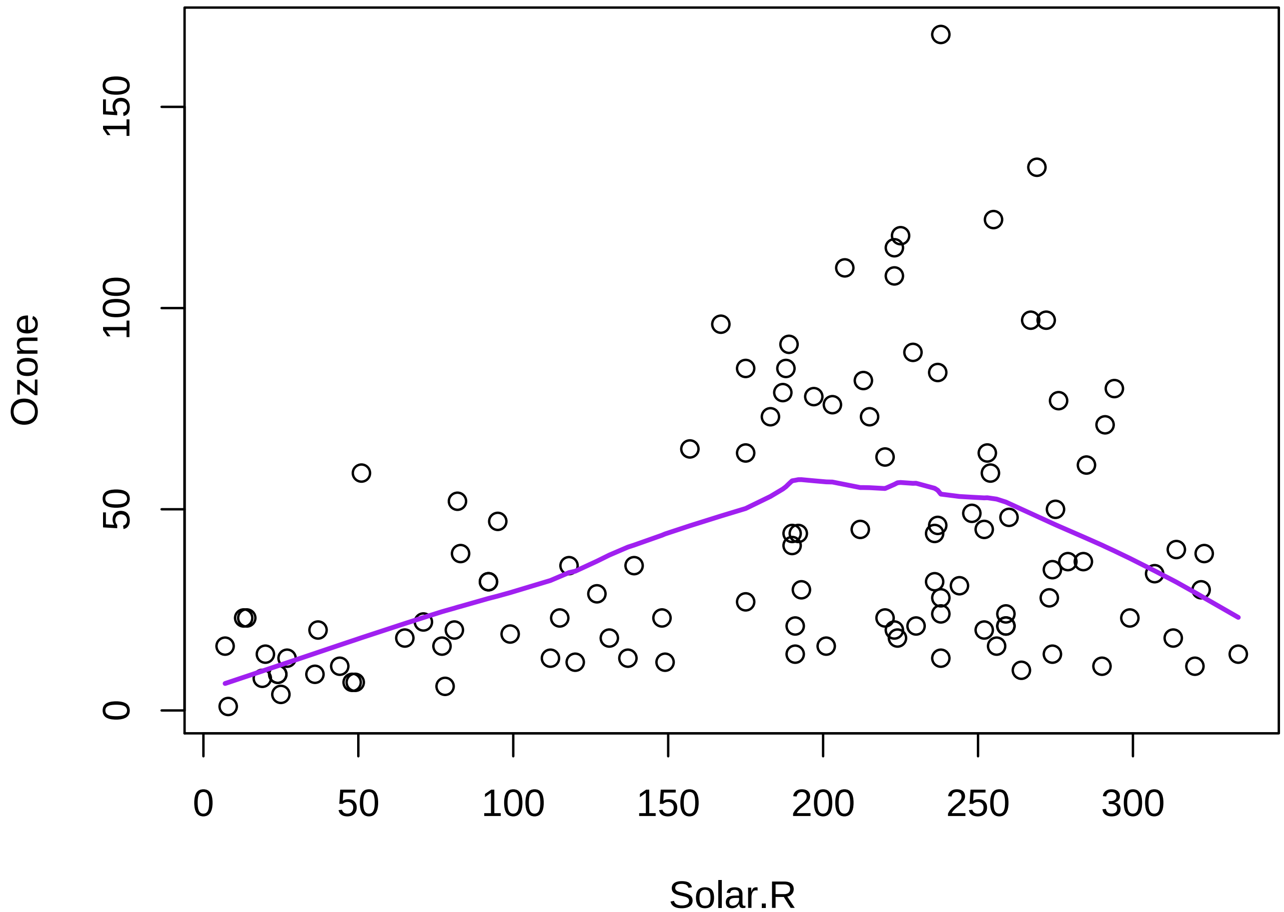
- `subset` takes a subset of a data frame.

# Conditioning plots

Ozone is a secondary pollutant, it is produced from organic compounds and atmostpheric oxygen in reactions catalyzed by nitrogen oxides and powered by su nlight.

However, looking at ozone concentrations in NY in summer we see a non-monotone relationship with sunlight

# Conditioning plots

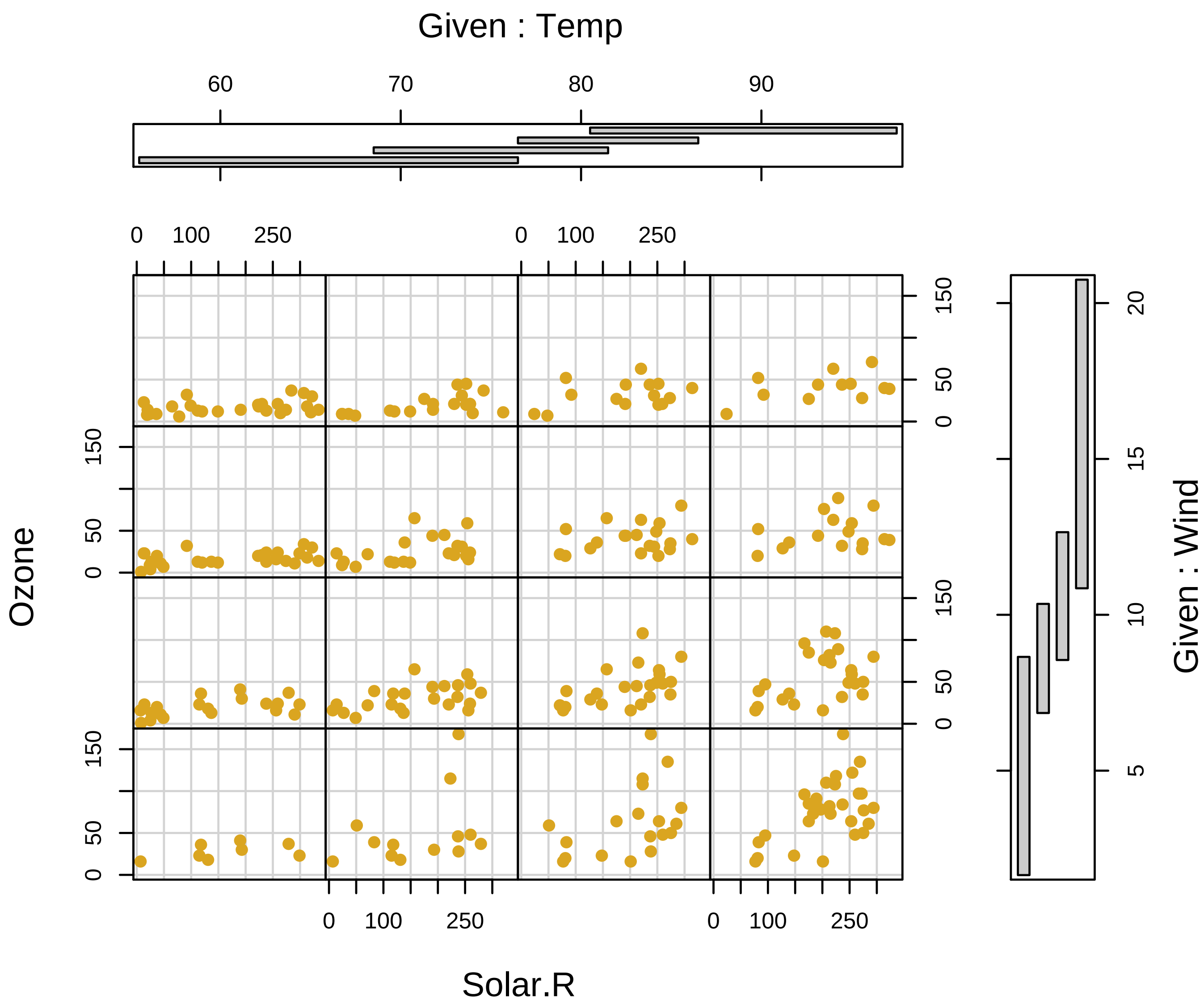


# Conditioning plots

Here we draw a scatterplot of `Ozone` vs `Solar.R` for various subranges of `Temp` and `Wind`. A simple version of what is possible with the Trellis system.

```
data(airquality)
coplot(Ozone ~ Solar.R | Temp * Wind, number = c(4, 4),
     data = airquality,
    pch = 21, col = "goldenrod", bg = "goldenrod")
```

# Conditioning plots



# Mathematical annotation

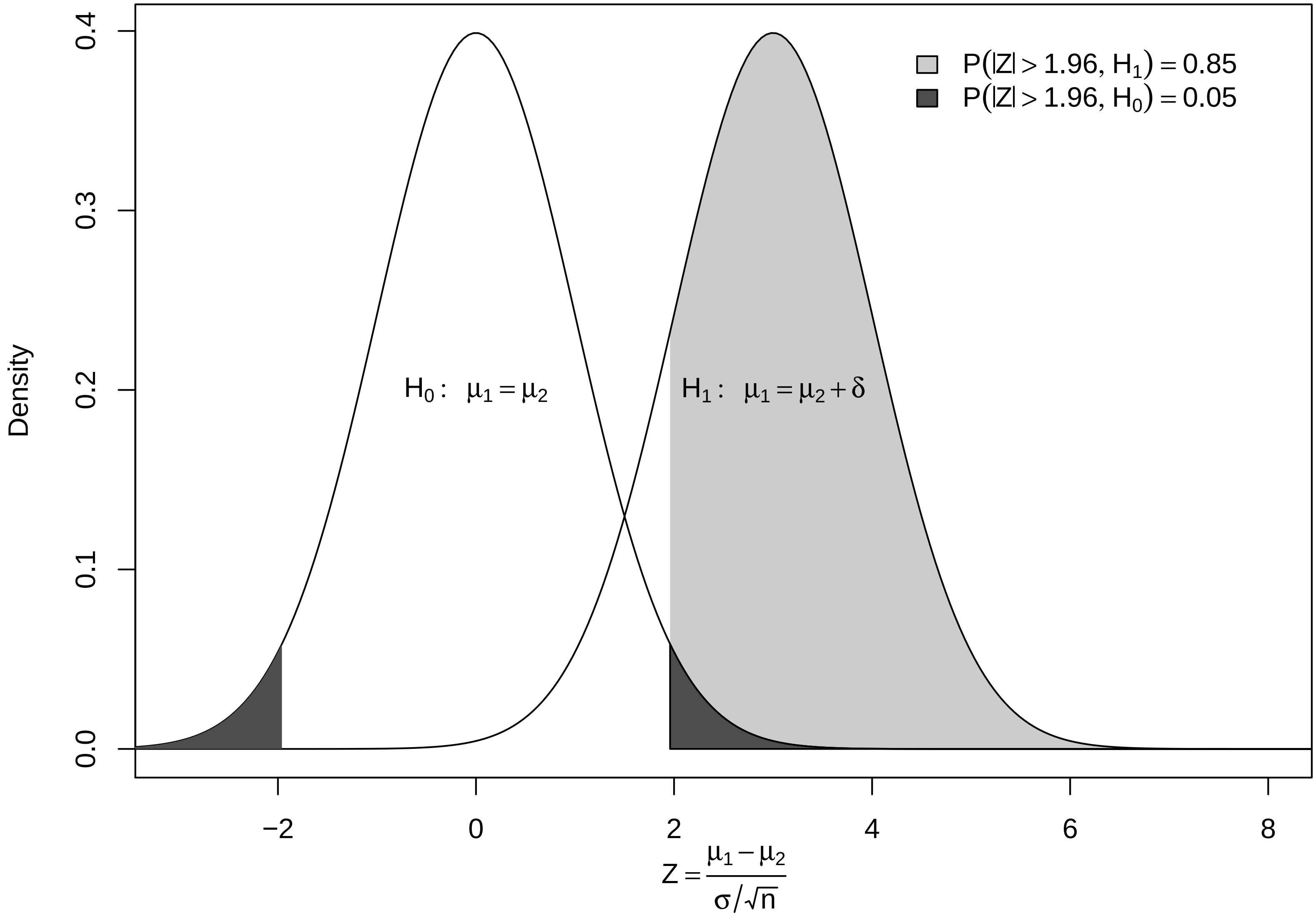
An expression can be specified in R for any text in a graph (`help(plotmath)` for details). Here we annotate a graph drawn with `polygon`.

```
x<-seq(-10,10,length=400)
y1<-dnorm(x)
y2<-dnorm(x,m=3)
par(mar=c(5,4,2,1))
plot(x,y2,xlim=c(-3,8),type="n",
    xlab=quote(Z==frac(mu[1]-mu[2],sigma/sqrt(n))),
    ylab="Density")
polygon(c(1.96,1.96,x[240:400],10),
        c(0,dnorm(1.96,m=3),y2[240:400],0),
        col="grey80",lty=0)
lines(x,y2)
lines(x,y1)
polygon(c(-1.96,-1.96,x[161:1],-10),
        c(0,dnorm(-1.96,m=0),y1[161:1],0),
        col="grey30",lty=0)
polygon(c(1.96,1.96,x[240:400],10),
        c(0,dnorm(1.96,m=0),y1[240:400],0),
        col="grey30")
```

# Mathematical annotation

```
legend(4.2,.4,fill=c("grey80","grey30"),
    legend=expression(P(abs(Z)>1.96,H[1])==0.85,
         P(abs(Z)>1.96,H[0])==0.05),bty="n")
text(0,.2,quote(H[0]:~~mu[1]==mu[2]))
text(3,.2,quote(H[1]:~~mu[1]==mu[2]+delta))
```

# Mathematical annotation



# Maps

```
> library(maps)
> map('county', 'washington', fill = TRUE,
        col = grey(sqrt(wa[,10]/(wa[,1]))) )
> title(main="Proportion Hispanic")
```

# Maps

Proportion Hispanic