2. Graphics in R

Thomas Lumley
Ken Rice

Universities of Washington and Auckland

Seattle, July 2015

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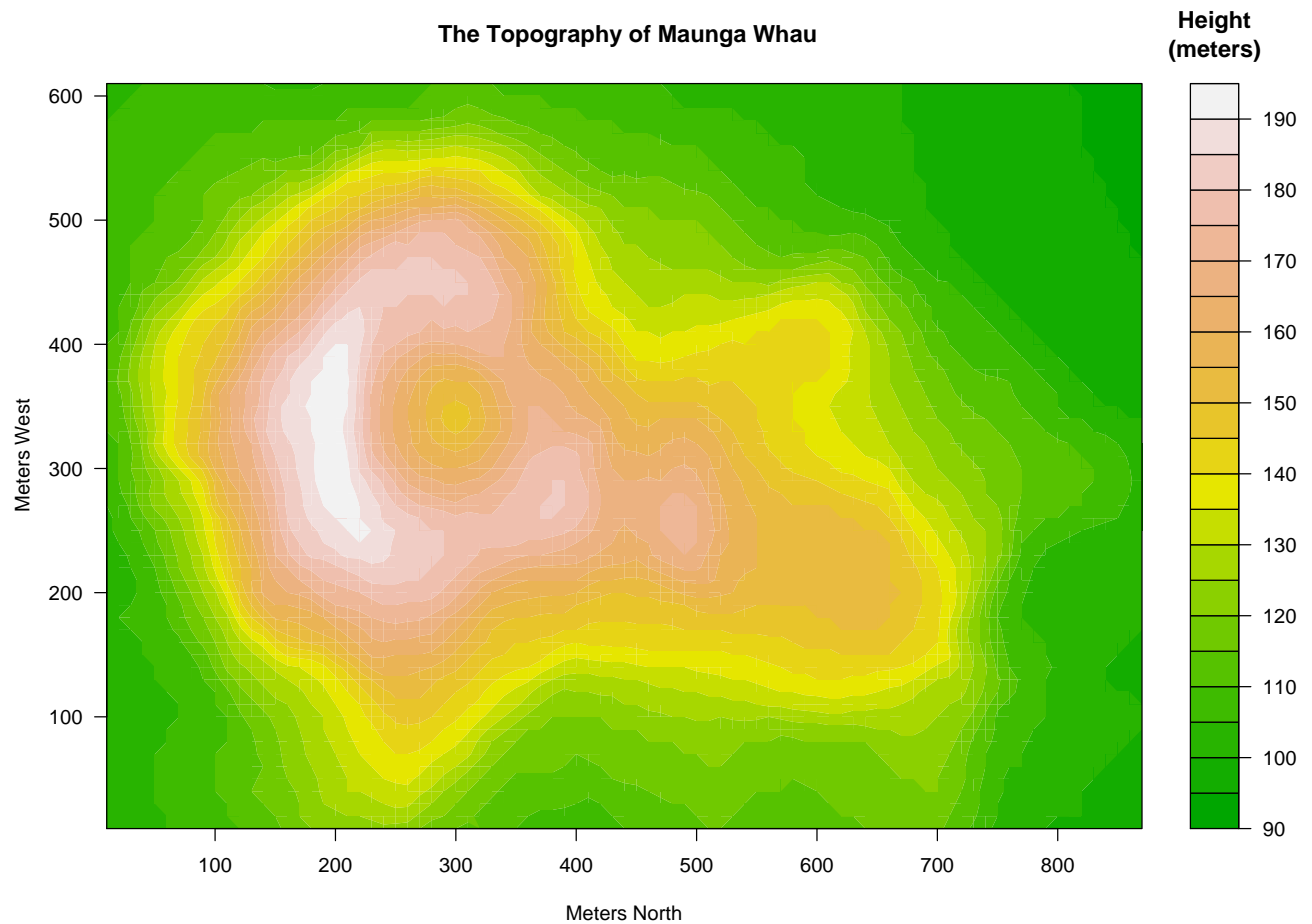
```
## on Windows
windows(height=4,width=6)
## on Unix
x11(height=4,width=6)
```

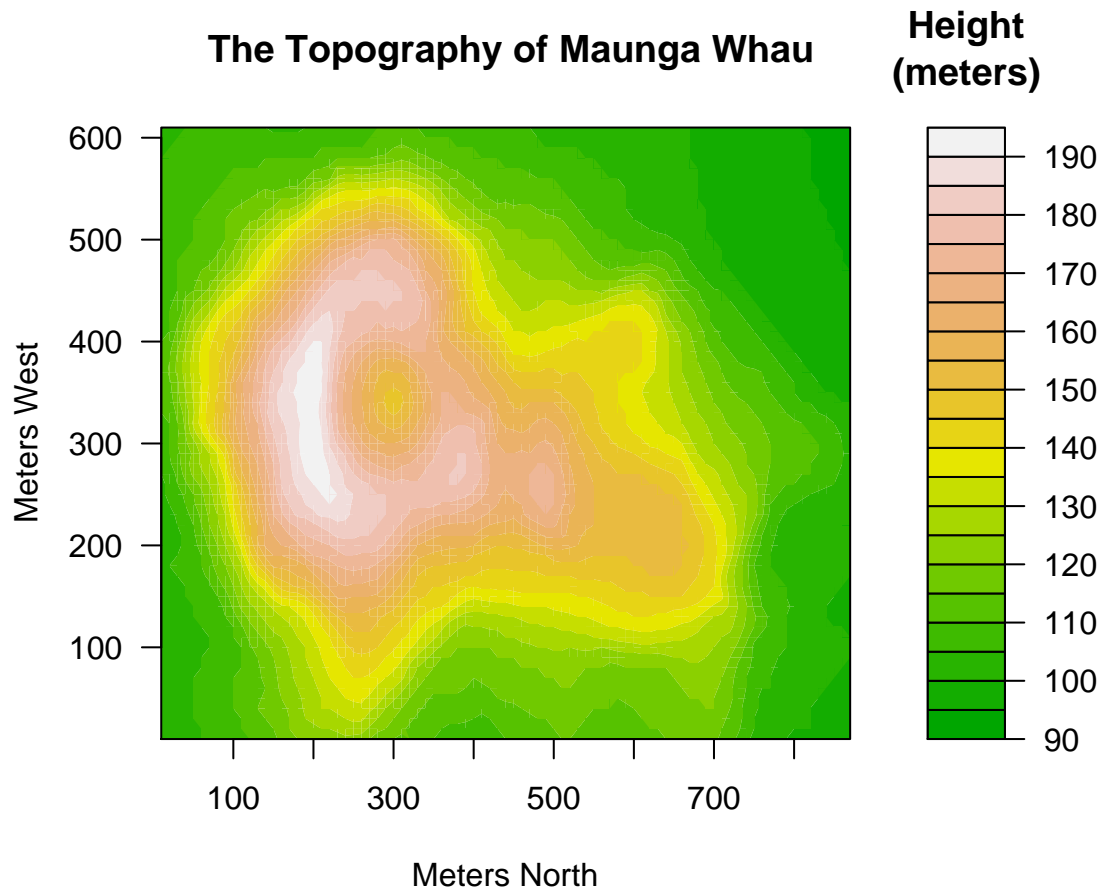Word or LaTeX can rescale the graph, but when the graph gets smaller, so do the axis labels...

Setup

Created at full-page size (11×8.5 inches)

Better scaling

Created at 6×5 inches

The Topography of Maunga Whau

Height (meters)

Meters West

Meters North

filled.contour(.) from R version 2.5.1 (2007−06−27)

The real thing

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()
```

You may find `dir()`, `getwd()` and `setwd()` helpful

Drawing

Usually, use `plot()` to create a graph and then `lines()`, `points()`, `legend()`, `text()`, `rect()`, `segments()`, `symbols()`, `arrows()` and other commands to annotate it. There is no 'erase' function, so keep your commands in a script.

`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 also be used this way;

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

where we introduce the formula system, that is also used for regression models. Here, think of `Y~X`.

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

First we set up the data — in this case, a built-in dataset, containing 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×6in and stored as PDF files;
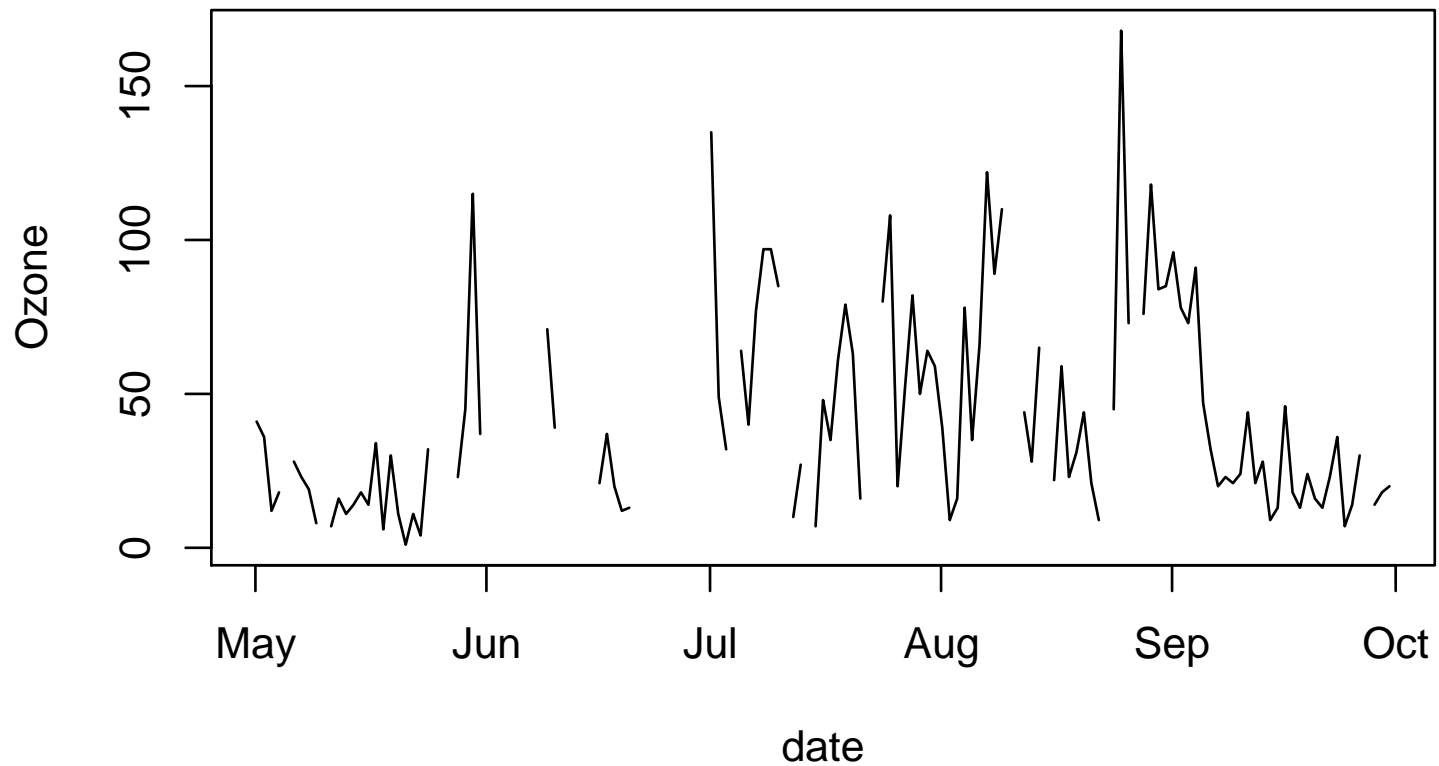
Options

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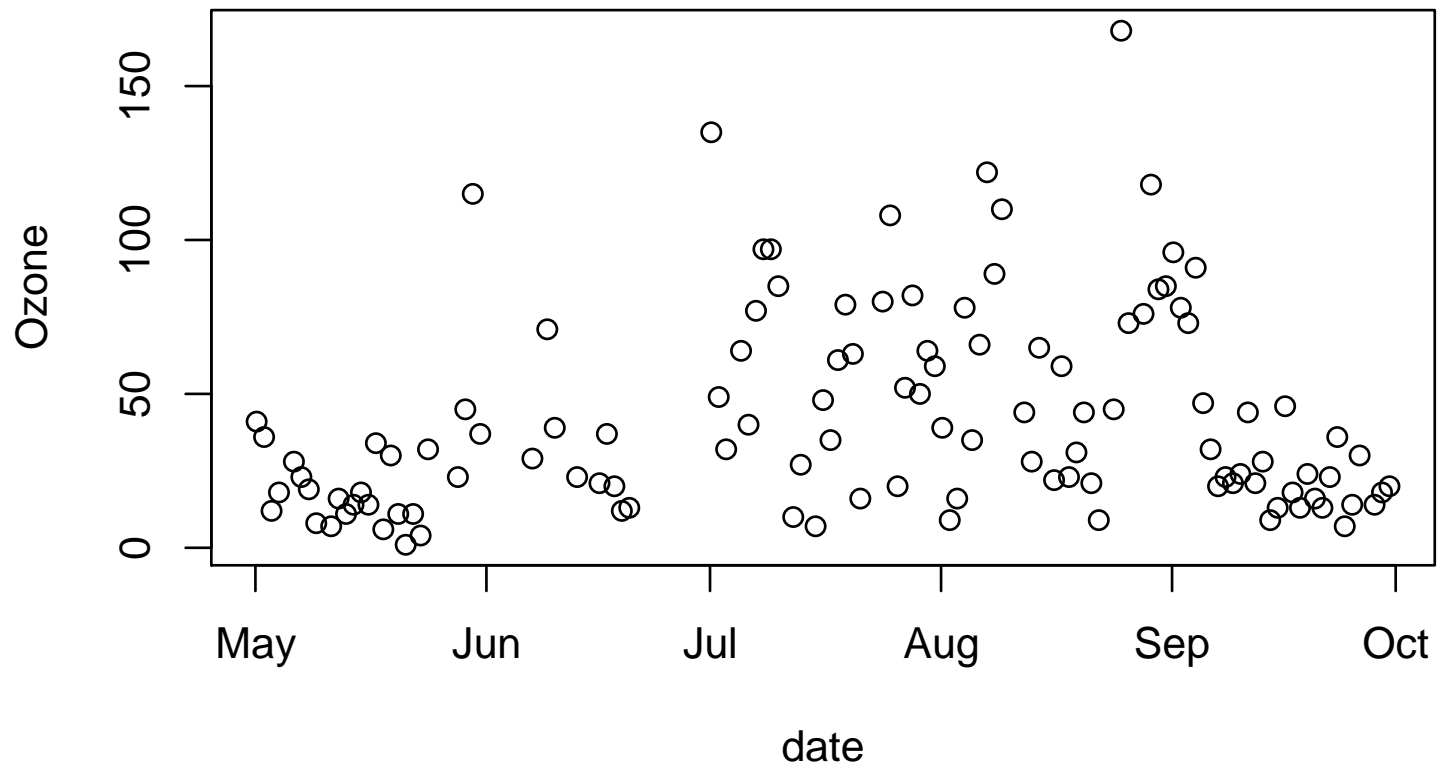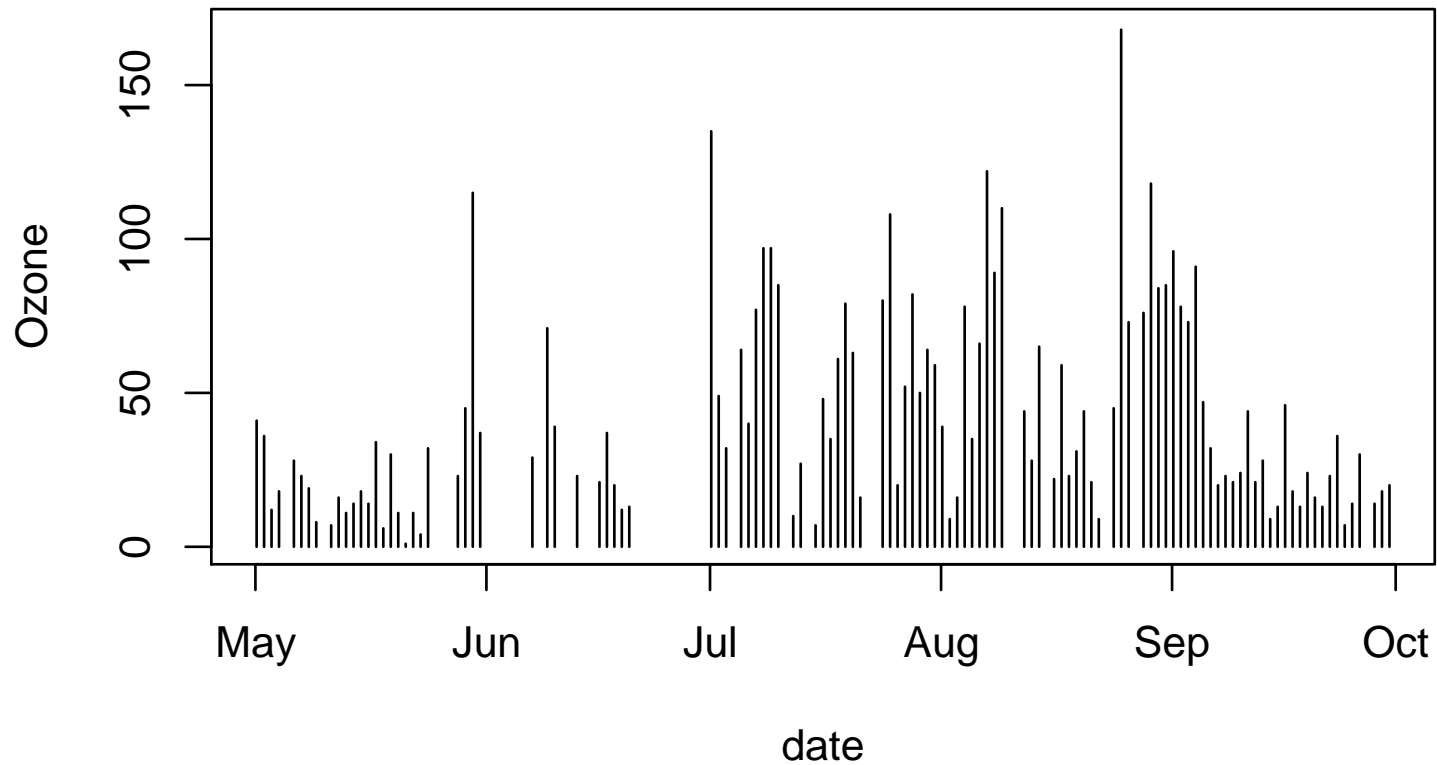
Options

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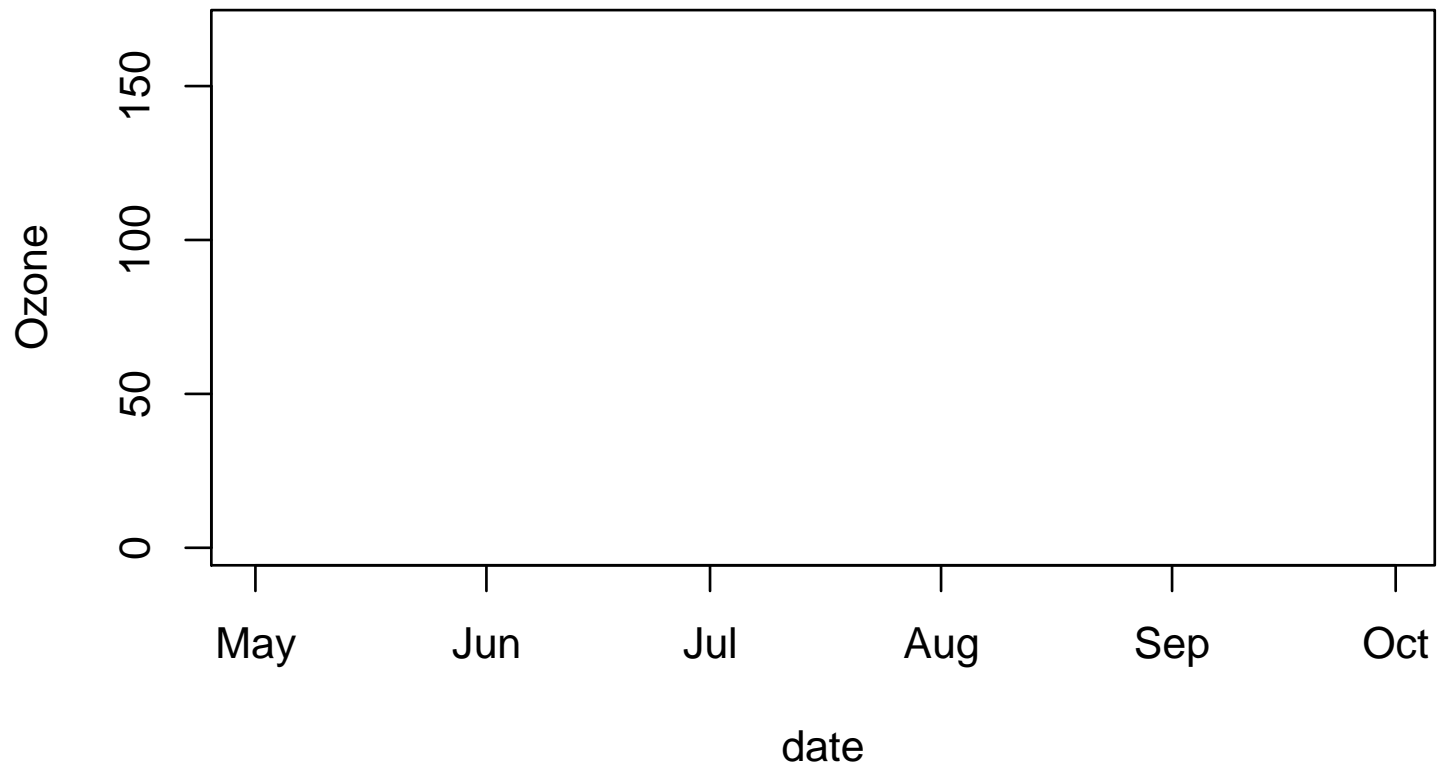
Options

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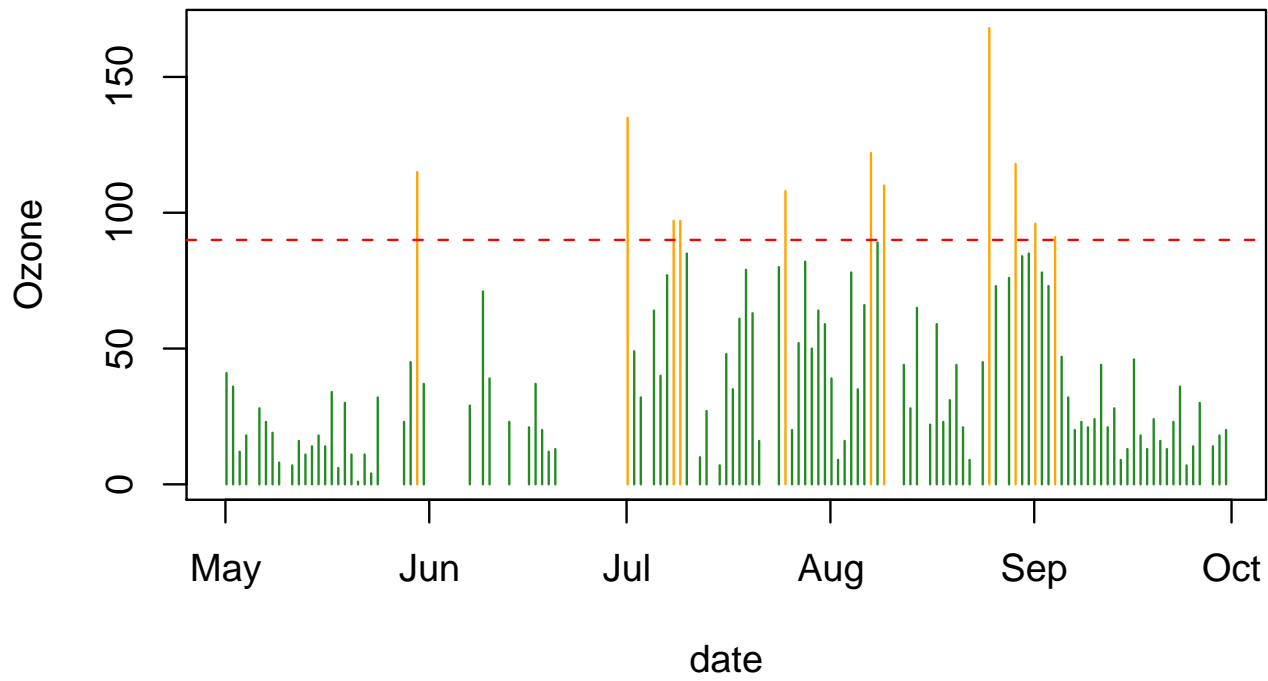
Options

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

Options

Commands to do a more complex plot;

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

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 (1:8) 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. See `?par`, then search for `lty`

Adding to a plot

More example commands

```
data(cars)
plot(dist~speed,data=cars)
with(cars, lines(lowess(speed, dist), col="tomato", lwd=2))

plot(dist~speed,data=cars, log="xy")
with(cars, lines(lowess(speed, dist), col="tomato", lwd=2))
with(cars, lines(supsmu(speed, dist), col="purple", lwd=2))

legend("bottomright", legend=c("lowess","supersmoother"),bty="n",
  lwd=2, col=c("tomato","purple"))
```
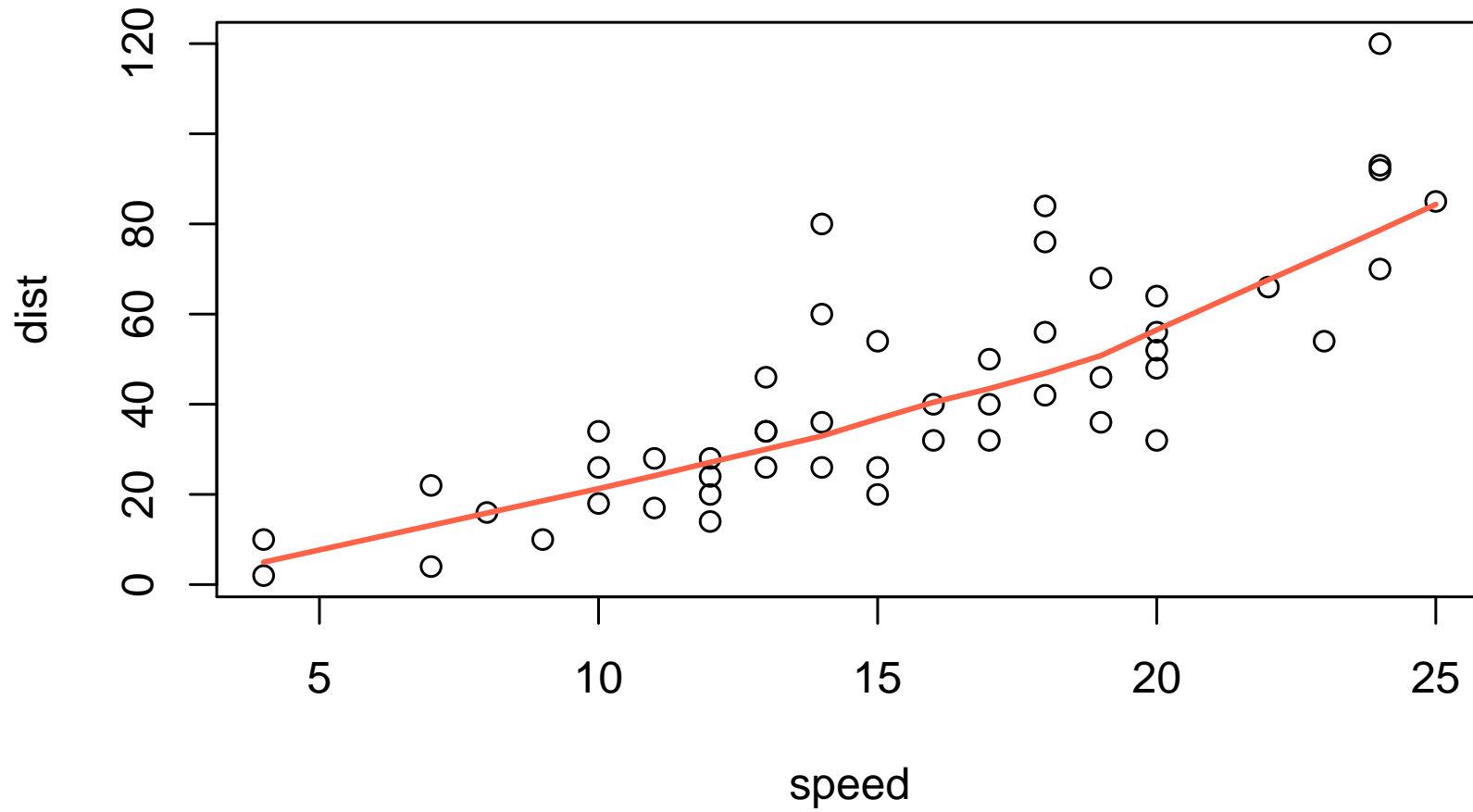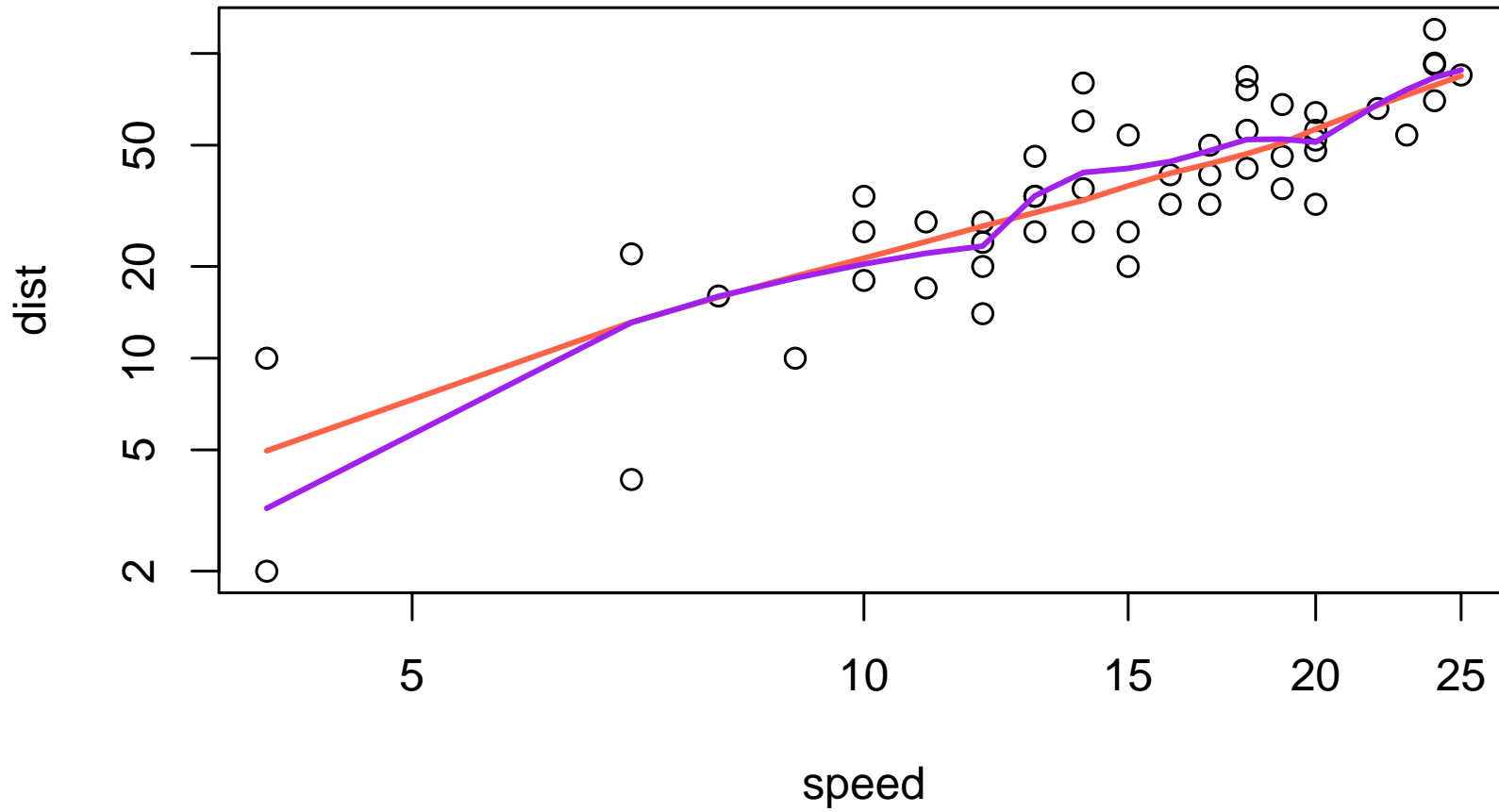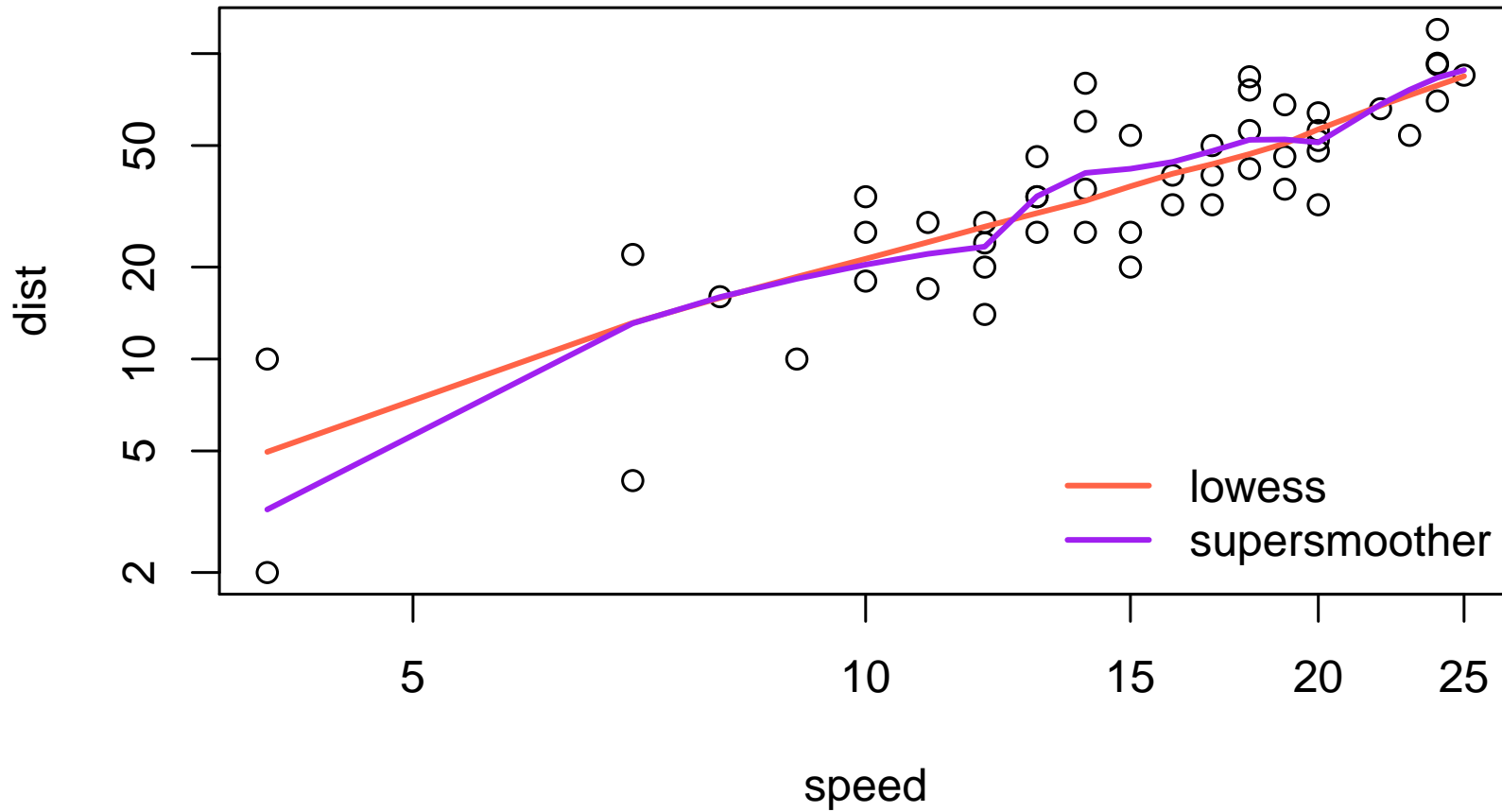
Adding to a plot

# Adding to a plot

Adding to a plot

Notes

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

- `lowess()` and `supsmu()` are scatterplot smoothers. They calculate smooth curves that fit the relationship between y and x locally. Their output has attributes `$x` and `$y`, that generic function `lines()` can cope with

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

- `legend()` adds a legend

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
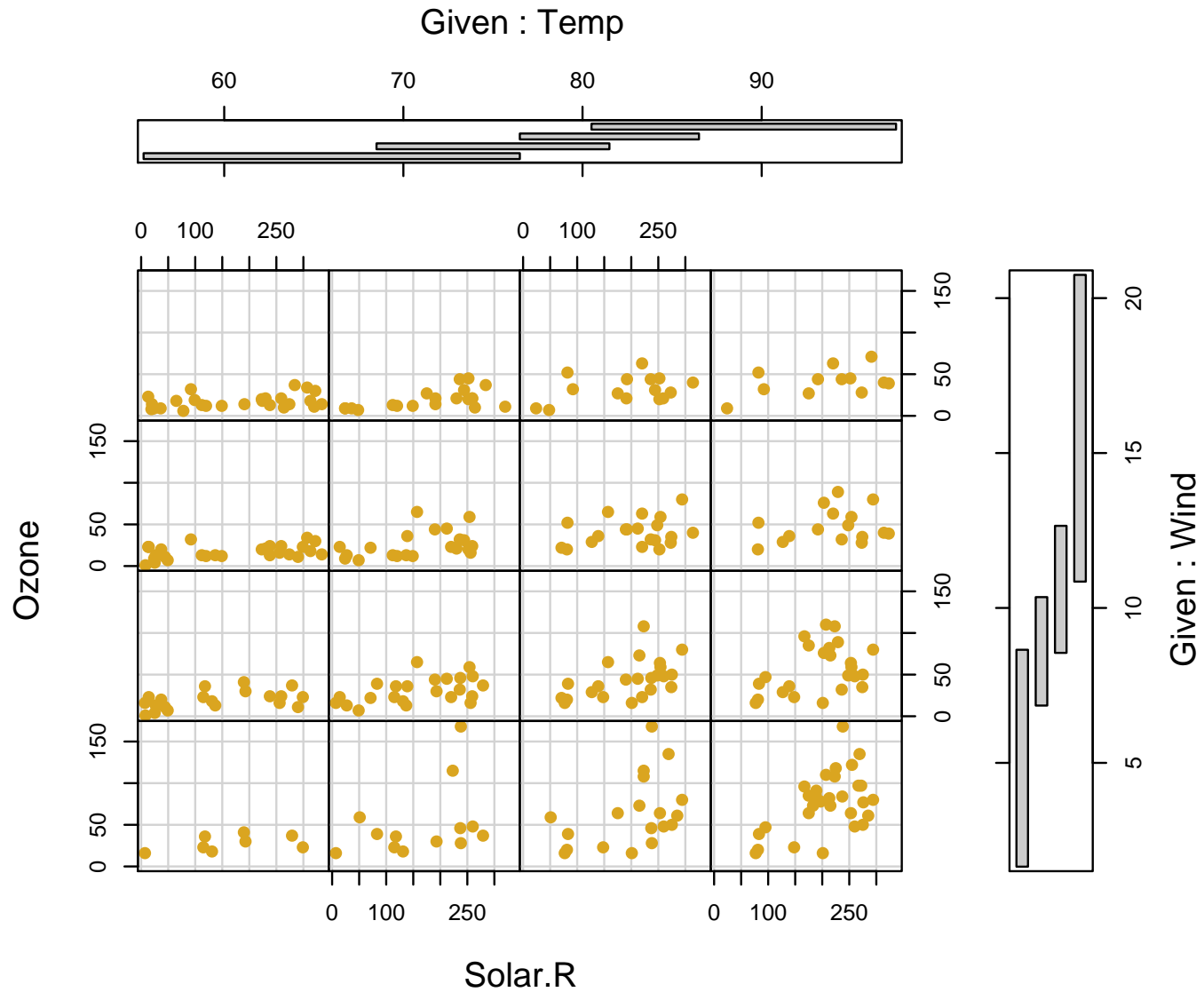
Conditioning plots
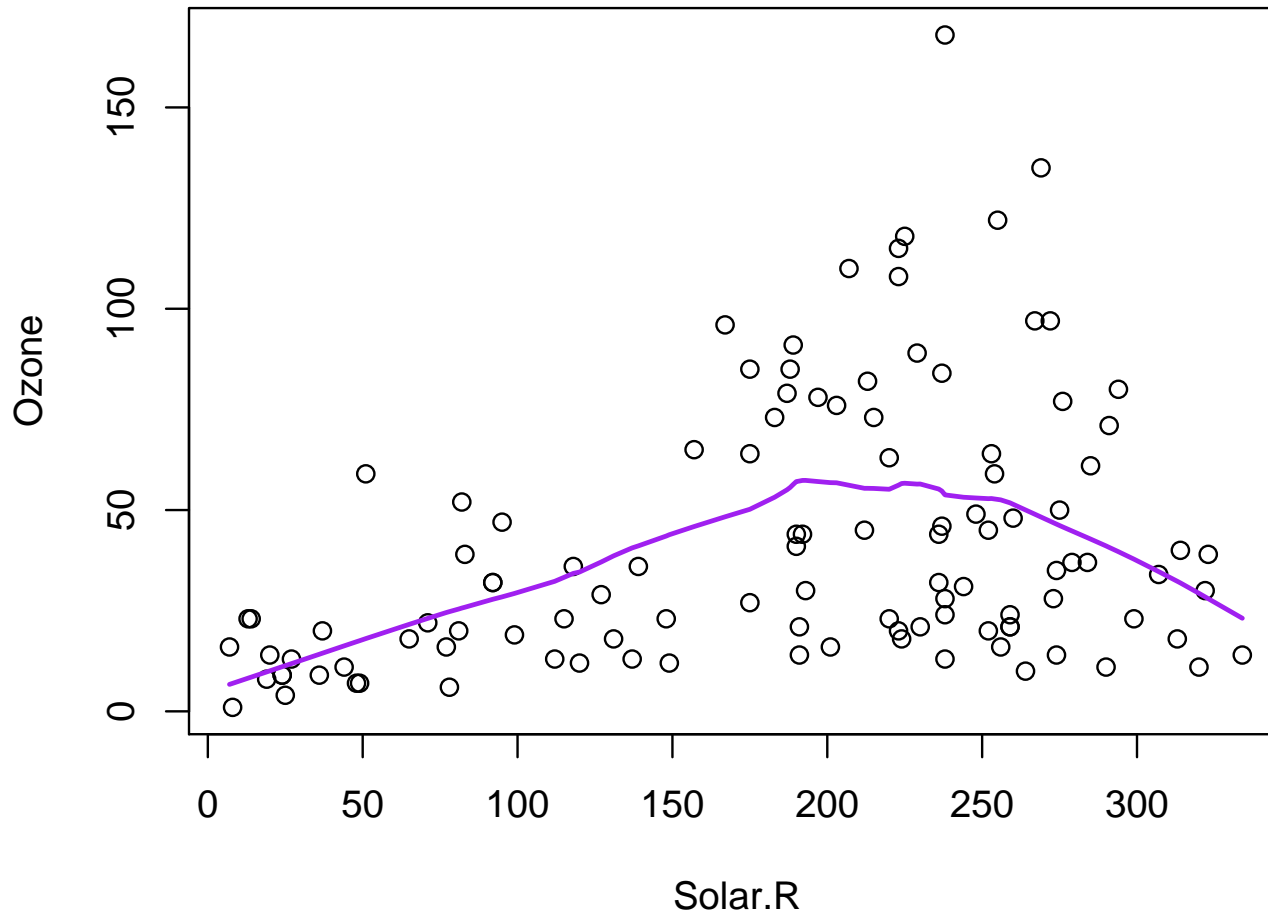
Conditioning plots

Here we draw a scatterplot of `Ozone` vs `Solar.R` for various subranges of `Temp` and `Wind`. For more examples like this, see the commands in the `lattice` package.

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

Conditioning plots

Conditioning plots

- A 4-D relationship is illustrated; the Ozone/sunlight relationship changes in strength depending on both the Temperature and Wind

- The vertical bar | is statistician-speak for 'conditioning on' (nb this is different to use of |'s meaning as Boolean 'OR')

- The horizontal/vertical 'shingles' tell you which data appear in which plot. The overlap can be set to zero, if preferred

- `coplot()`'s default layout is a bit odd; try setting `rows`, `columns` to different values

- For more plotting commands that support conditioning, see `library(help="lattice")`
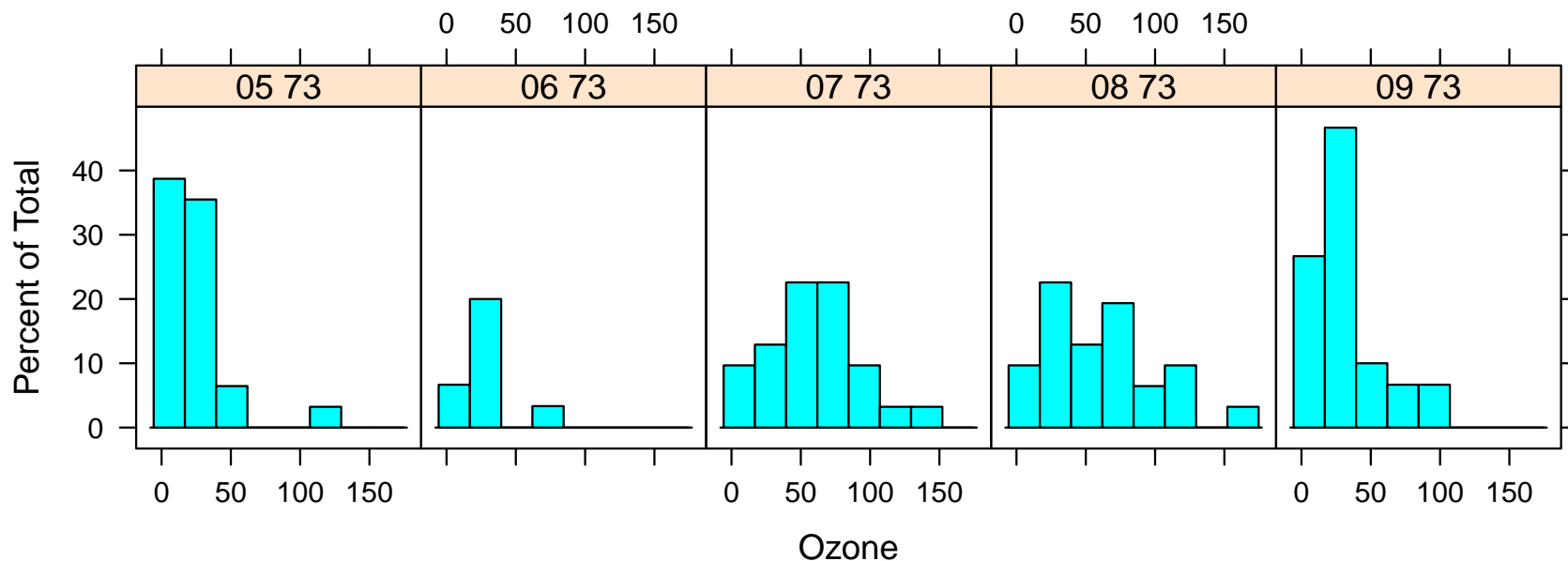
Conditioning plots: general

Functions in the `lattice` package allow various forms of conditioning plot — here for histograms;

```
library("lattice")
airquality$date  <- with(airquality, ISOdate(1973,Month,Day))
airquality$month <- strftime(airquality$date, "%m %y")
histogram(~Ozone|month, data=airquality)
```

Conditioning plots: general

Some other plots that can be conditioned;

- `xyplot()` for scatterplots − more flexible version of `coplot()`
- `barchart()` for boxplots
- `dotplot()` and `stripchart()`
- `qq()` and `qqmath()`, e.g. comparing to $N(0,1)$
- `levelplot()` − heatmaps, see also `image()`
- `contourplot()` − see also `contour()`
- `cloud()` and `wireframe()`, for fake 3D

For still others see `?Lattice` after loading the `lattice` package ...
or `demo(lattice)`.

For fine control, these functions all permit `panel` arguments, i.e. you can supply a function implemented in each subplot. `?panel.abline` describes some pre-written versions.

persp() and friends

Should you *ever* need them; (and you may not!)

```
with(trees,{
per <- persp(x=1:2, y=1:2, z=matrix(NA,2,2), theta=35,
    xlim=range(Girth), ylim=range(Height), zlim=range(Volume),
    xlab="Girth",ylab="Height", zlab="Volume", axes=TRUE, ticktype="detailed")
points(trans3d(x=Girth, y=Height, z=Volume, per), col="red", pch=19)
})
```
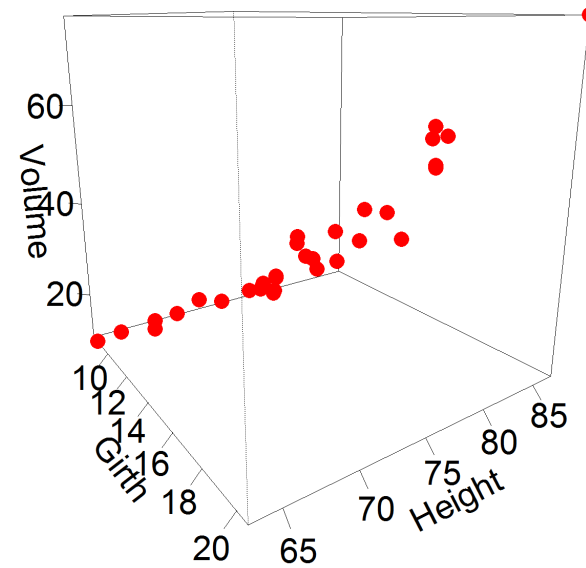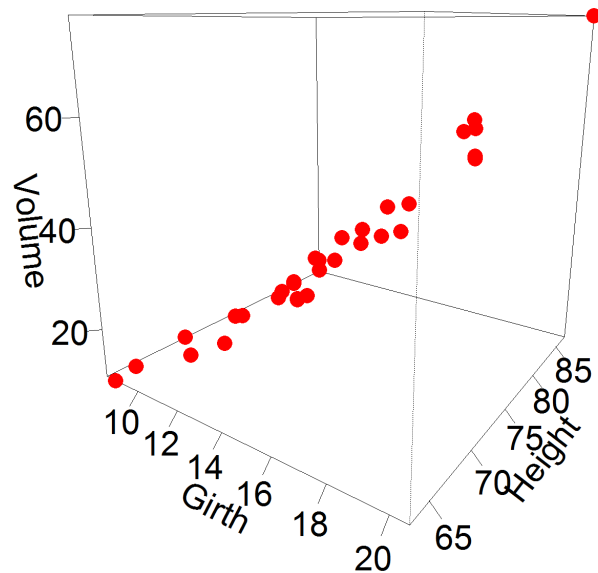

persp() is okay for wireframes, otherwise try cloud() first.

Toys: Mathematical annotation

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

Toys: 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))
```

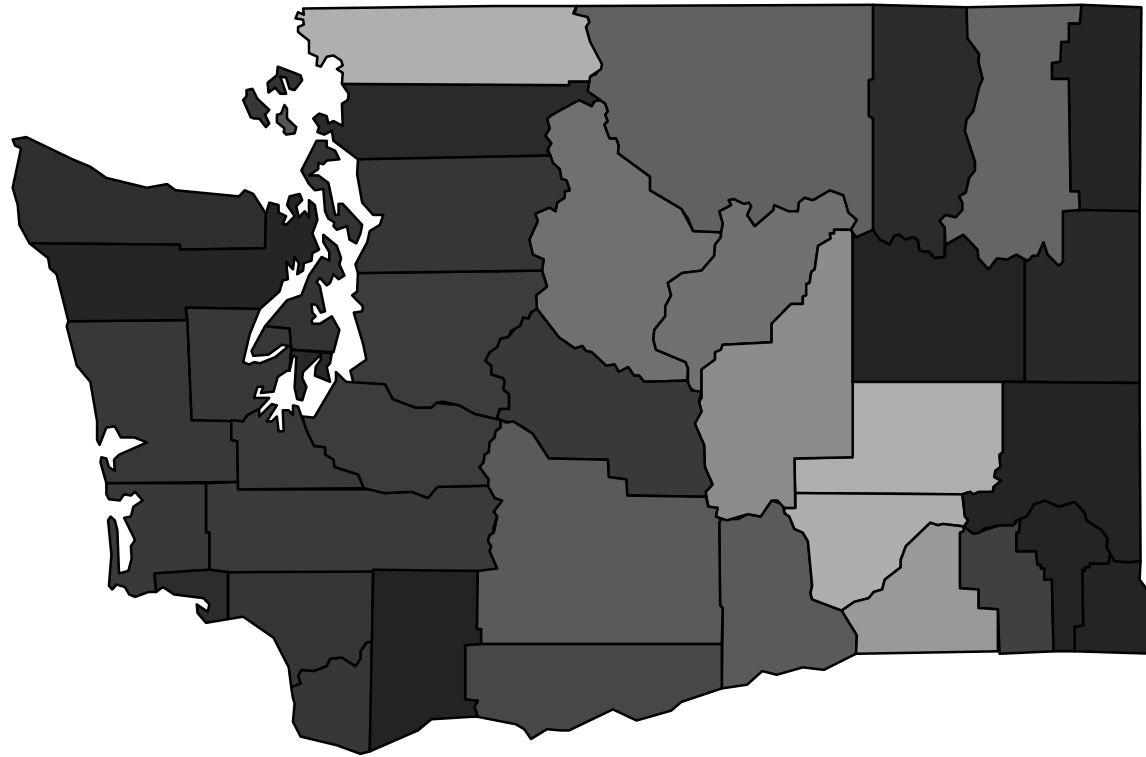# Toys: Mathematical annotation
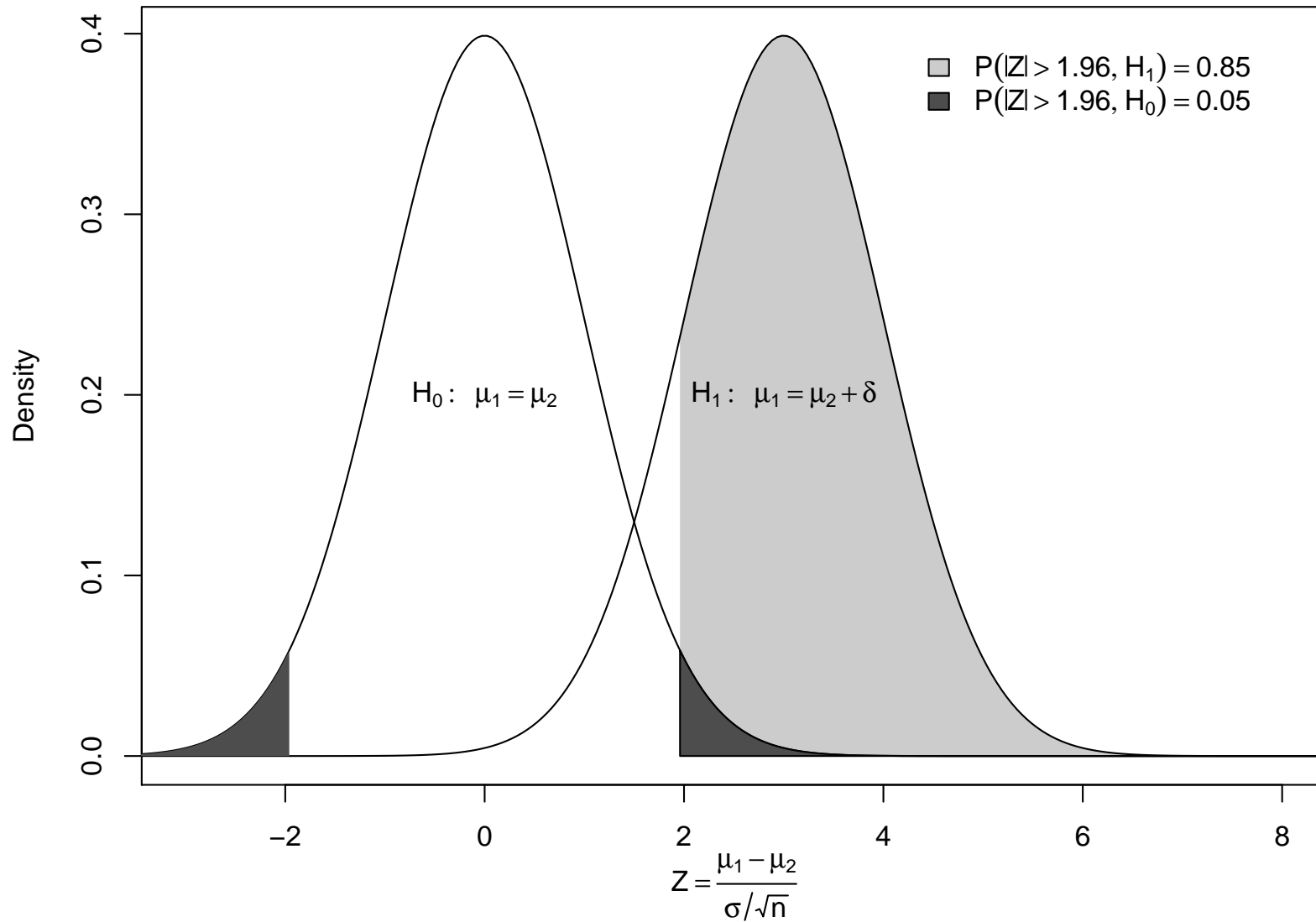
Toys: Maps

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

Proportion Hispanic

Even conditioned maps!

Health insurance coverage by age and state