



Learning to Draw!

Thomas Lumley

R Core Development Team
and University of Washington

Auckland — 2009-11-28

Principles

- A graph, like a paragraph, exists to convey a specific piece of information.
 - to the analyst (exploratory)
 - to other people (presentation)
- There are **facts** about the design of graphs, it's not all opinion
- There's also lots of useful expert opinion.

A graph needs to be designed, drafted, and revised, just like a paragraph. R's defaults are not bad, but usually benefit from customization.

Basic graphics competencies

- Simple R graphics
- Underused basic tools: scatterplot smoothers, forest plots
- Physiological constraints: color and preattentive perception
- Escape from flatland: conditioning plots
- Transparency and binning for large data.
- Pointy-clicky!!

Plenty more not included, eg maps, categorical data, etc

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)
```

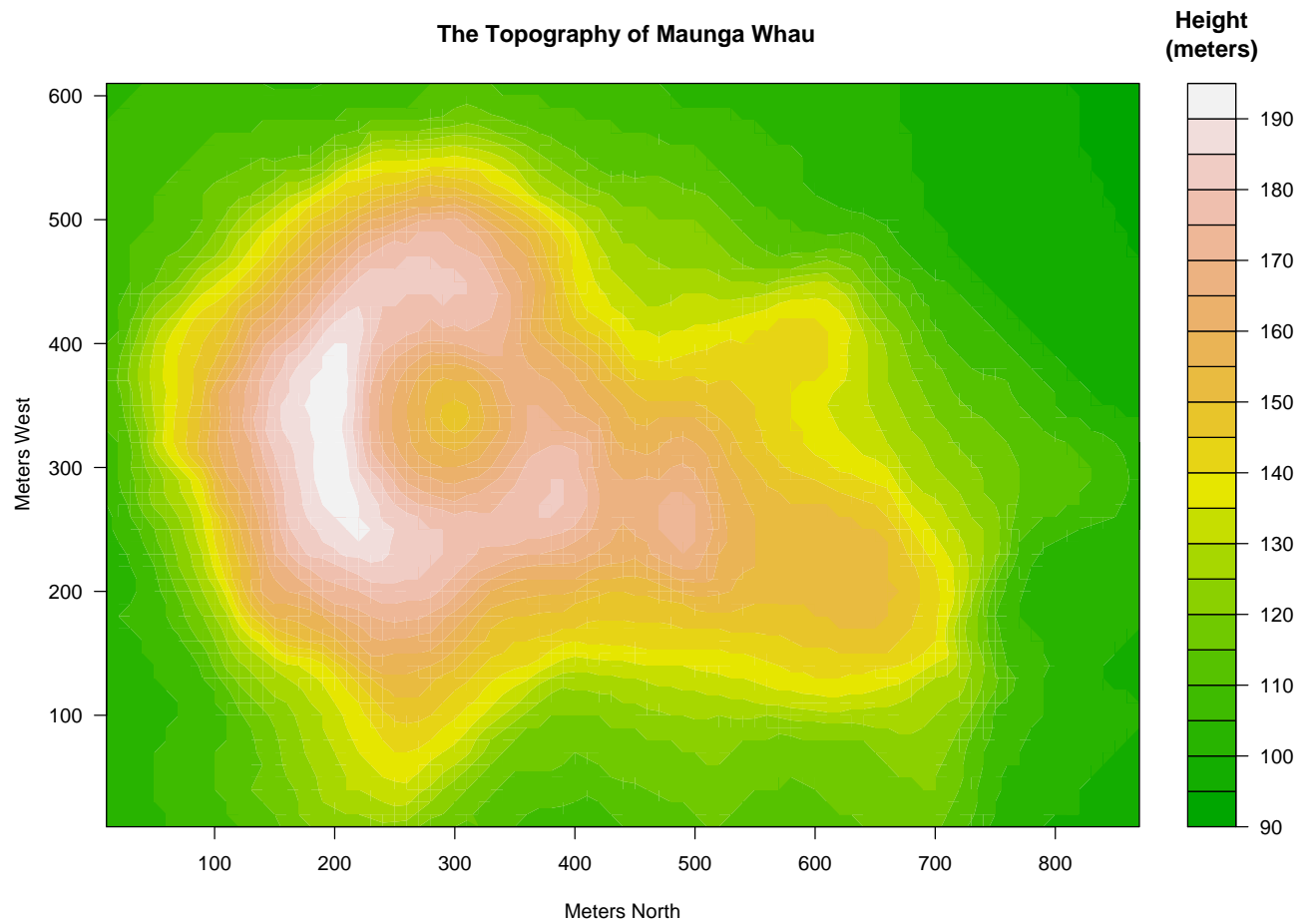
```
## on Unix
```

```
x11(height=4,width=6)
```

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

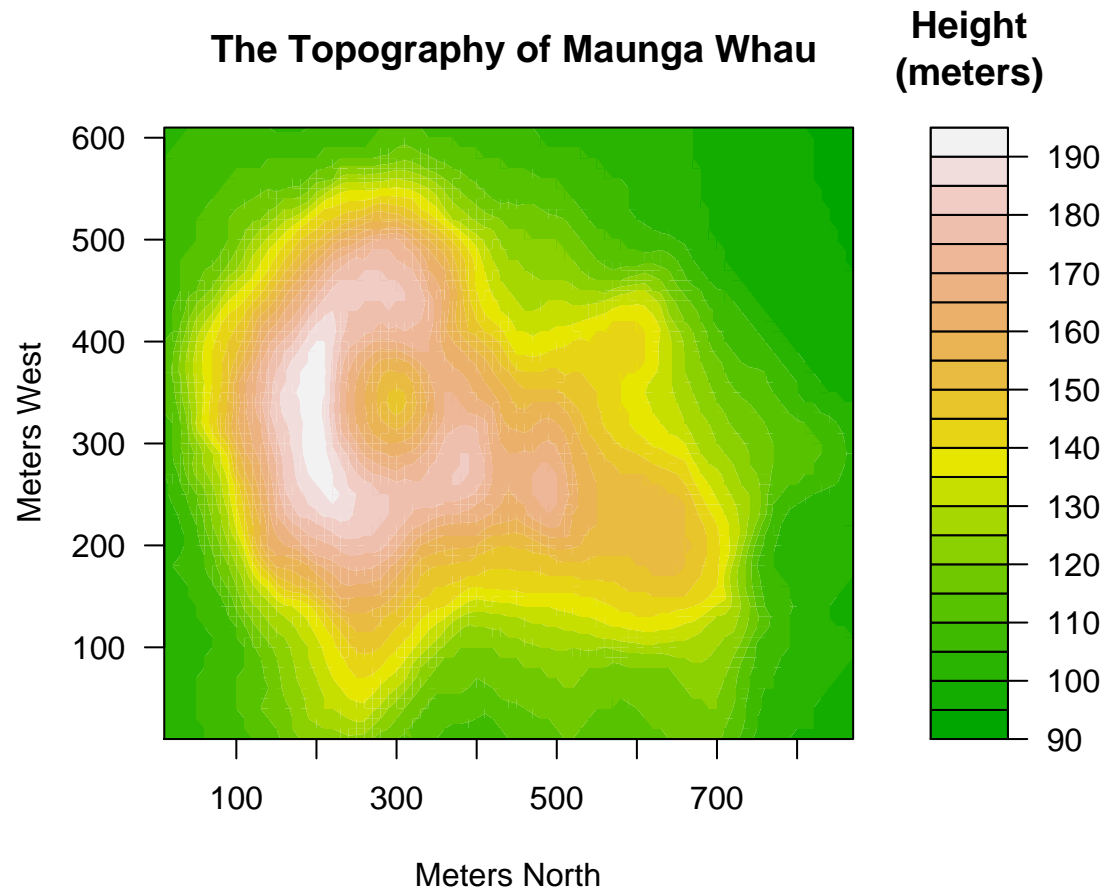
Setup

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



Setup

Created at 6×5 inches



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

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

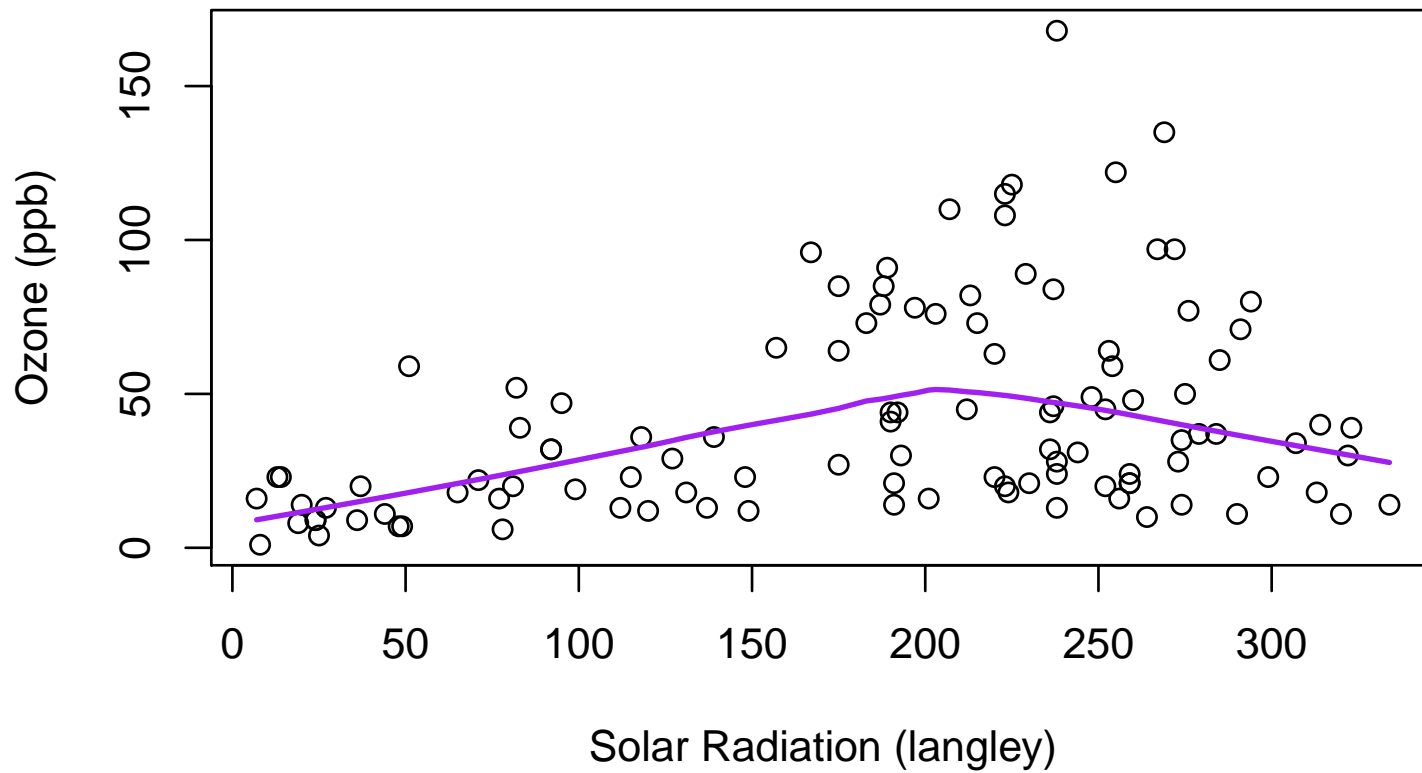
Set up a data set: daily ozone concentrations in New York, summer 1973

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

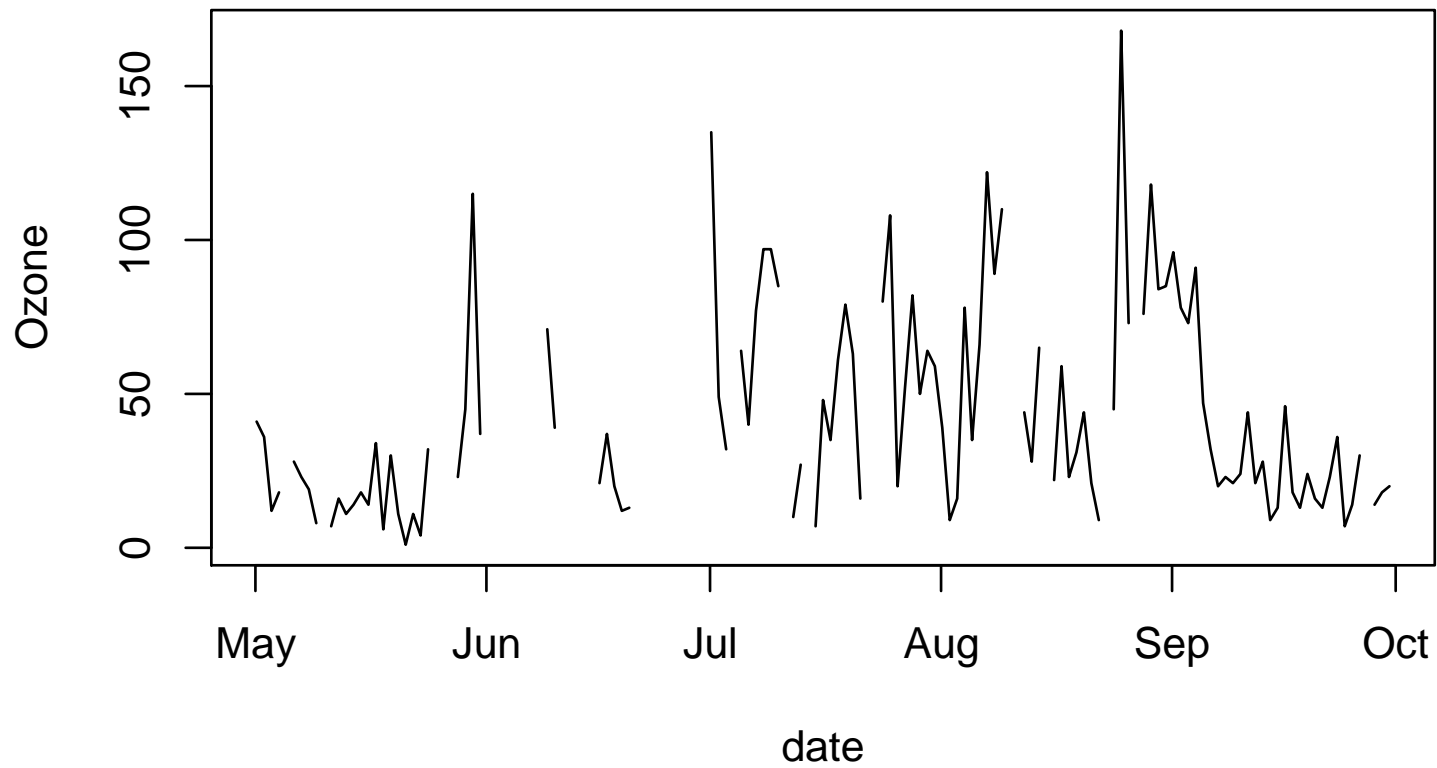
All these graphs were designed at 4in×6in and stored as PDF files

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

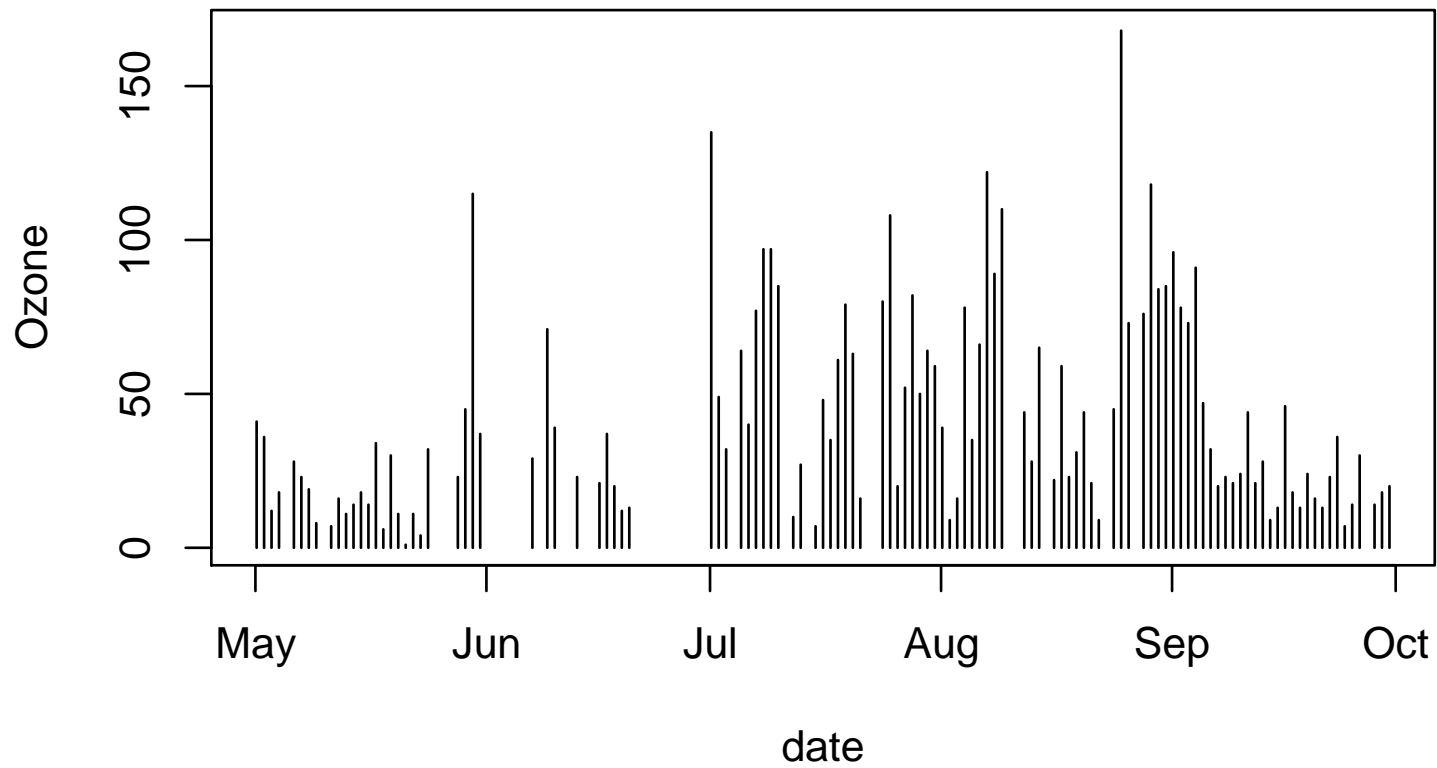
New York, Summer 1979



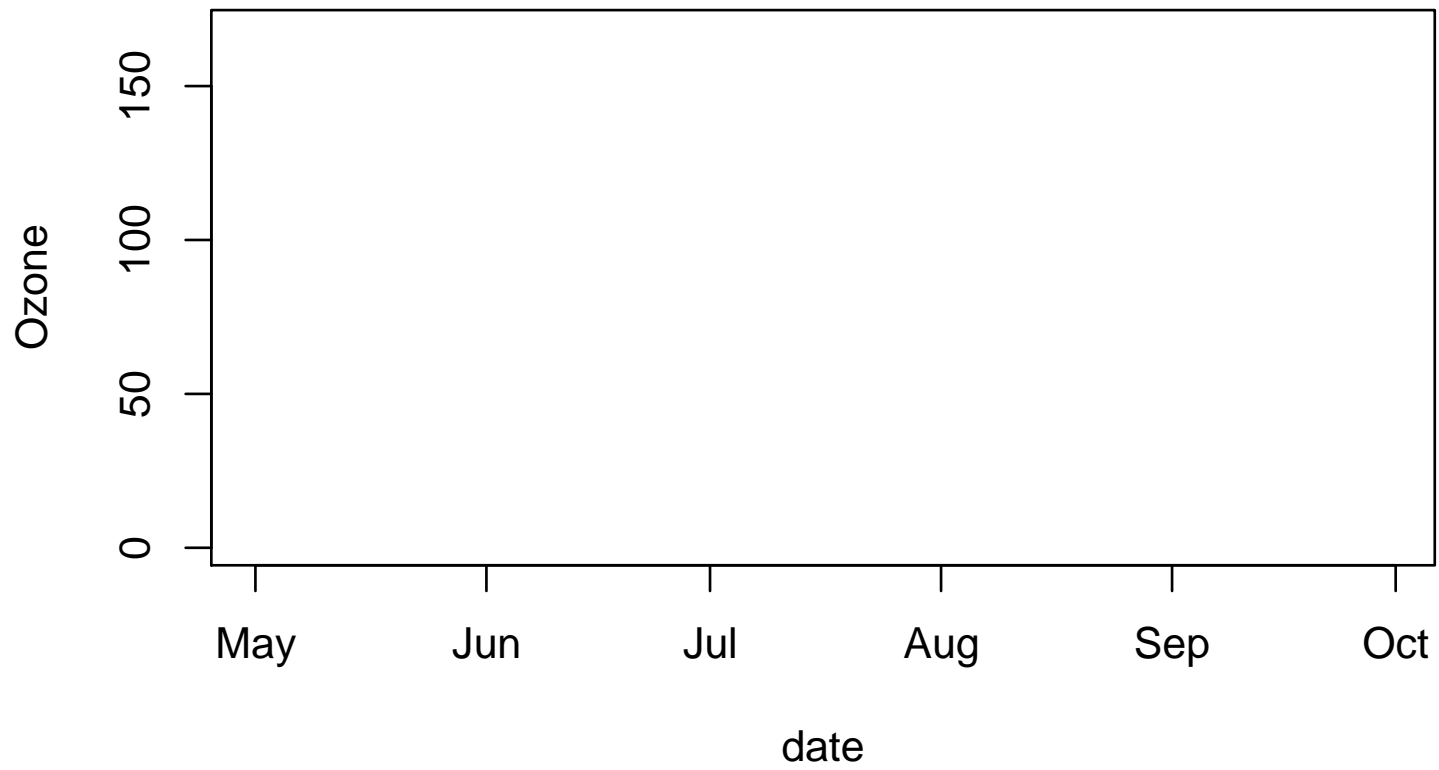
```
plot(Ozone~date, data=airquality,type="l")
```



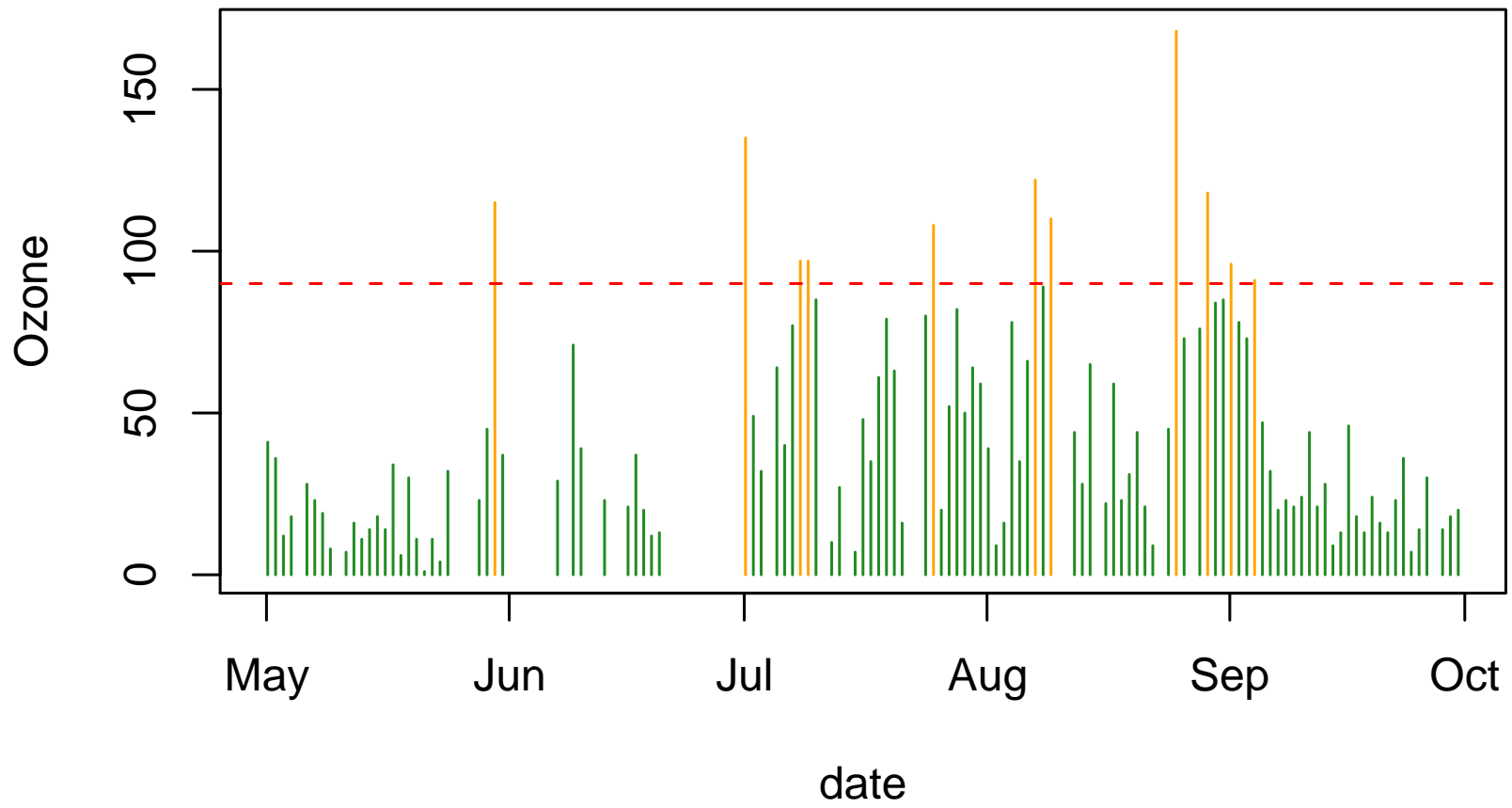
```
plot(Ozone~date, data=airquality,type="h")
```



```
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")
```



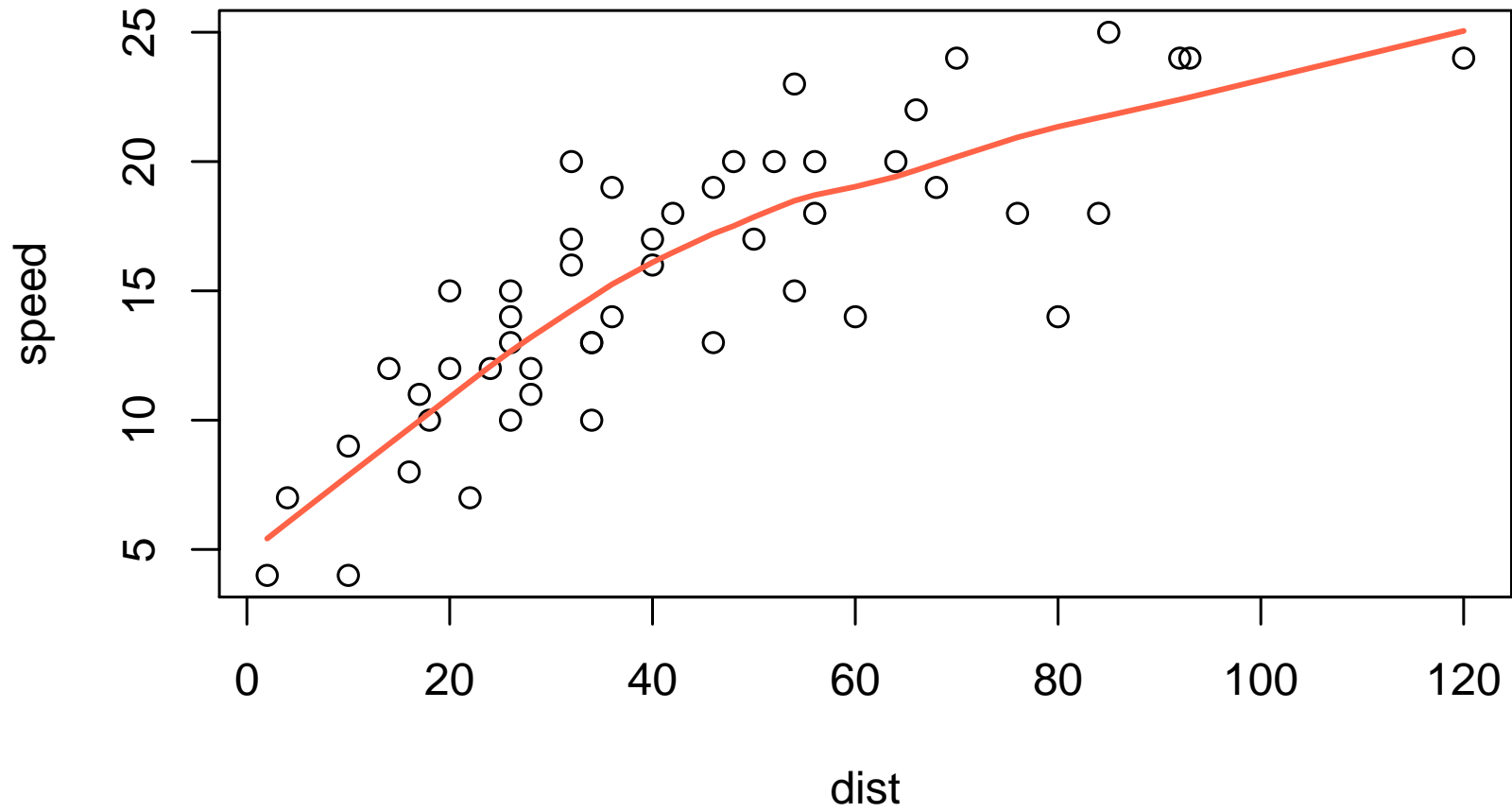
Notes

- `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.

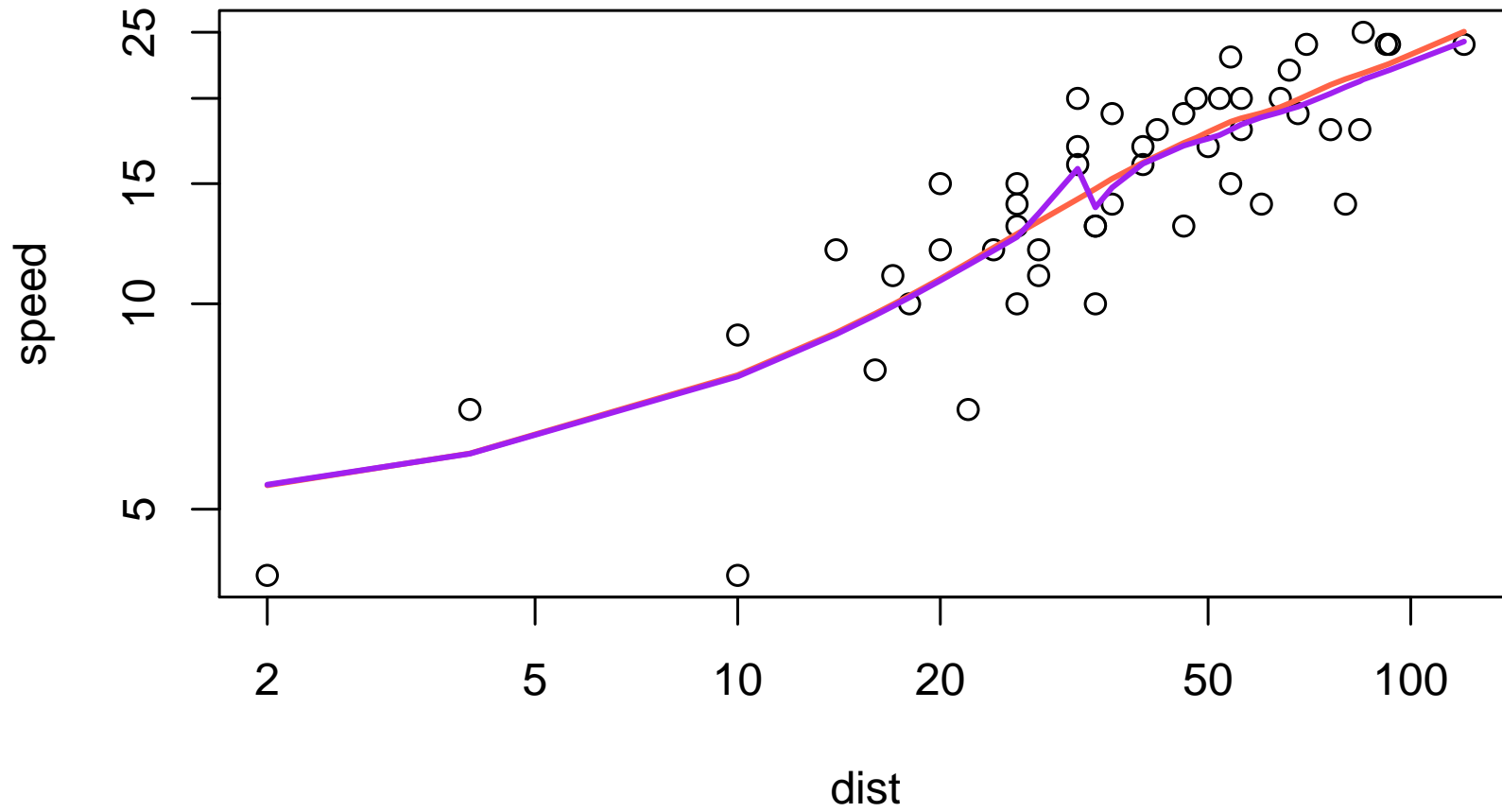
Adding to a plot

```
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"))
```

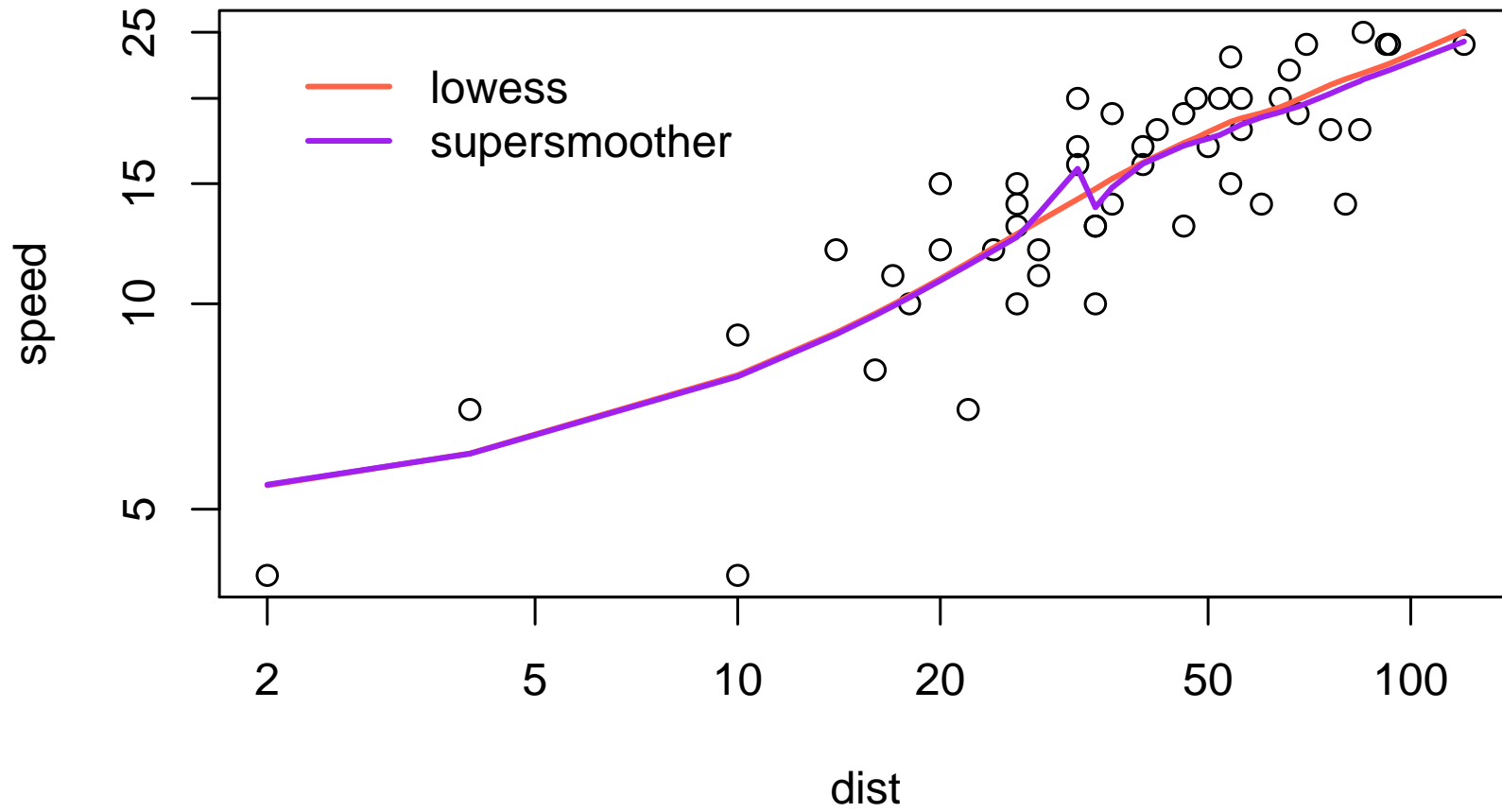
Adding to a plot



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

Scatterplot smoothing

Most scatterplots benefit from a indication of trend or spread.

- robust trend smoothers: lowess, median (R quantreg)
- mean trend smoothers: non-iterated lowess, kernel regression, splines, supersmoother
- conditional distribution: multiple quantile smoothers (eg 10%, 25%, 50%, 75%, 90%: R quantreg)

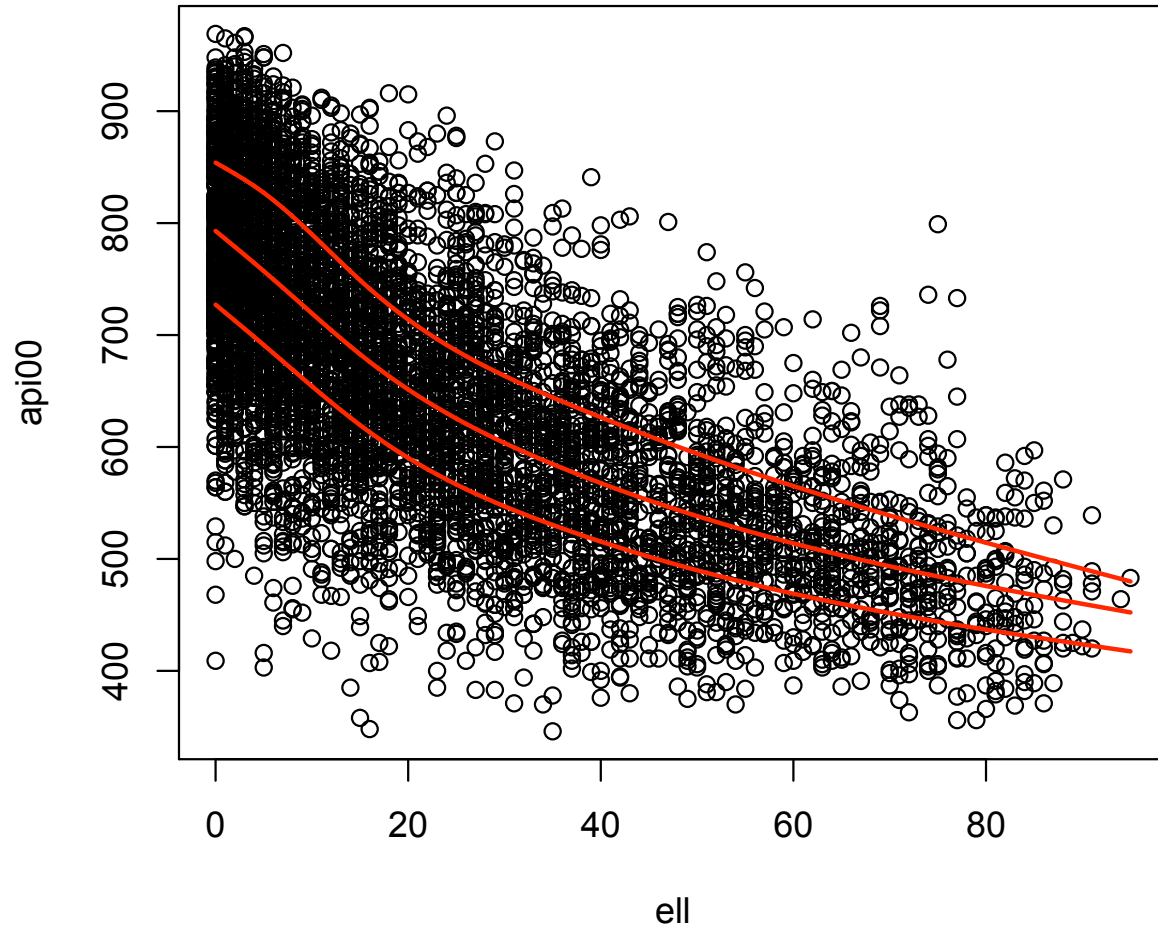
Remember that robust smoothers don't estimate the mean: eg smoothing squared residuals to get variance.

Scatterplot smoothing

Example: quantile smoothing with quantreg. School test results are lower for schools with lots of 'English language learners'—does this apply across the distribution or only at the mean?

```
library(quantreg)
library(splines)
data(api, package="survey")
plot(api00~ell, data=apipop)
xx<-apipop$ell
oo<-order(xx)
a<-rq(api00~ns(ell,4),data=apipop,tau=0.25)
yy<-fitted(a)
lines(xx[oo],yy[oo],col="red",lwd=2)
a<-rq(api00~ns(ell,4),data=apipop,tau=0.5)
yy<-fitted(a)
lines(xx[oo],yy[oo],col="red",lwd=2)
a<-rq(api00~ns(ell,4),data=apipop,tau=0.75)
yy<-fitted(a)
lines(xx[oo],yy[oo],col="red",lwd=2)
```

Scatterplot smoothing



Forest plots

The following are relative risks of bacterial colonisation from randomized trials of venous catheters with antimicrobial coating.

	RR	p
Ciresi	0.951	0.890
George	0.265	0.241
Hannan	0.630	0.408
Heard	0.866	0.810
Maki	0.208	0.043
Ramsay	0.237	0.197
Trazzera	0.644	0.503
Collins	0.355	0.350
Bach(b)	0.000	—
Tennenberg	0.588	0.330
Pemberton	0.833	0.836
Logghe	1.147	0.692

Forest plots

```
library(rmeta)
library(xtable)
data(catheter)
a <- meta.MH(n.trt, n.ctrl, inf.trt, inf.ctrl, data=catheter,
             names=Name, subset=c(1,2,3,4,6,8,10:15),
             statistic="RR")
output<-data.frame(RR=exp(a$logRR),
                   p=2*pnorm(abs(a$logRR/a$selogRR), lower.tail=FALSE))
row.names(output)<-a$names
xtable(output,digits=3)
```

Example: meta-analysis

A statistician would prefer this version:

	RR	(lower	95% upper)
Ciresi	0.95	0.47	1.94
George	0.27	0.03	2.44
Hannan	0.63	0.21	1.88
Heard	0.87	0.27	2.78
Maki	0.21	0.05	0.95
Ramsay	0.24	0.03	2.11
Trazzera	0.64	0.18	2.33
Collins	0.35	0.04	3.12
Bach(b)	0.00	0.00	—
Tennenberg	0.59	0.20	1.71
Pemberton	0.83	0.15	4.69
Logghe	1.15	0.58	2.26

In R: `xtable(summary(a)$stats)`

Example: meta-analysis

Several articles on this topic described the existing literature as inconsistent and as not showing strong support for the use of coated catheters.

The p -value table is almost useless for examining consistency, the confidence interval table is better but not ideal

A graph would help

Worst ever graph?

This is a graph design I saw at the American Heart Association conference on the epidemiology and prevention of heart disease.

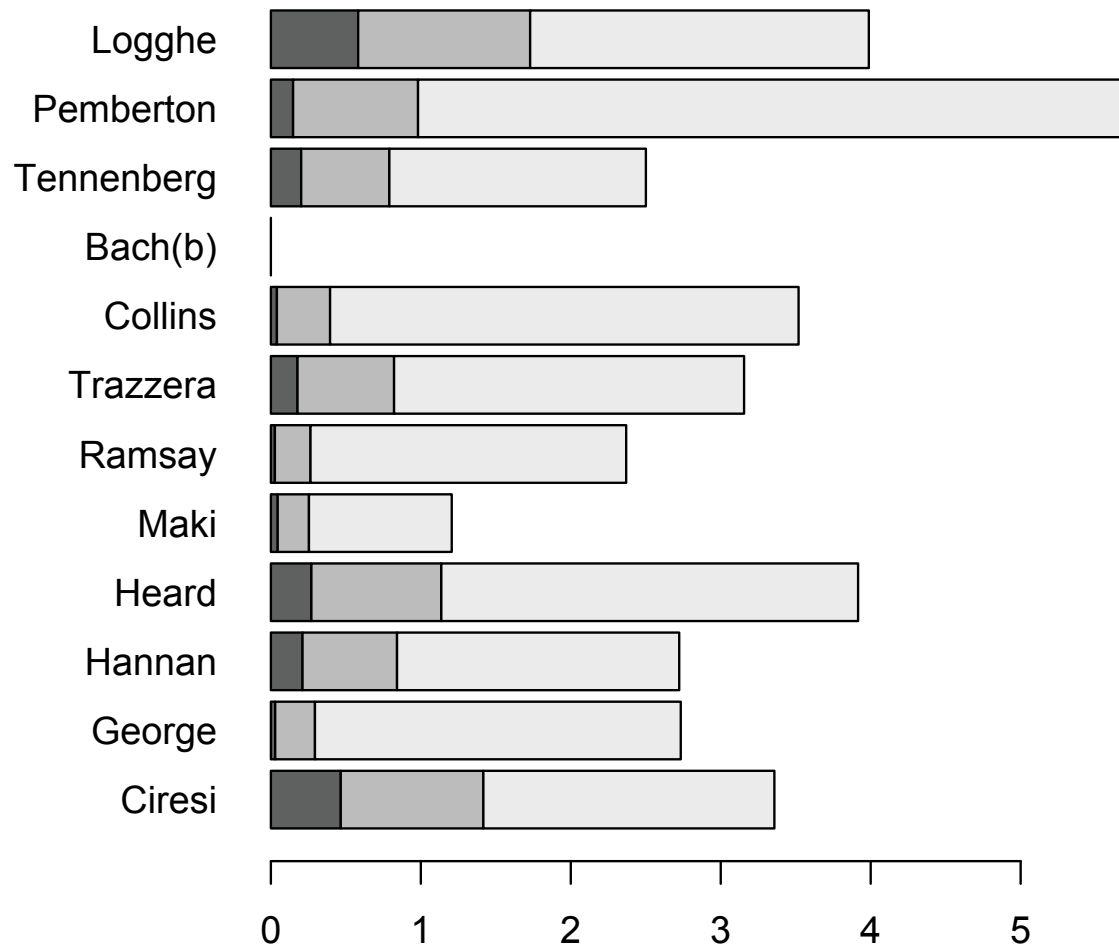
It was a display of confidence intervals as for a meta-analysis, but using a bar plot.

It's not easy to reproduce the bad color schemes and shading in R, but the basic design is straightforward.

```
par(las=1, mar=c(5.1,8.1,2.1,2.1))  
barplot(t(summary(a)$stats[,c(2,1,3)]),horiz=TRUE)
```

Why is this a bad graph?

Worst ever graph?



Better graph: forest plot

Show confidence interval and point estimate

Emphasize point estimate, by shading.

Increase visual weight of large trials relative to small trials, by increasing size of point estimate glyph

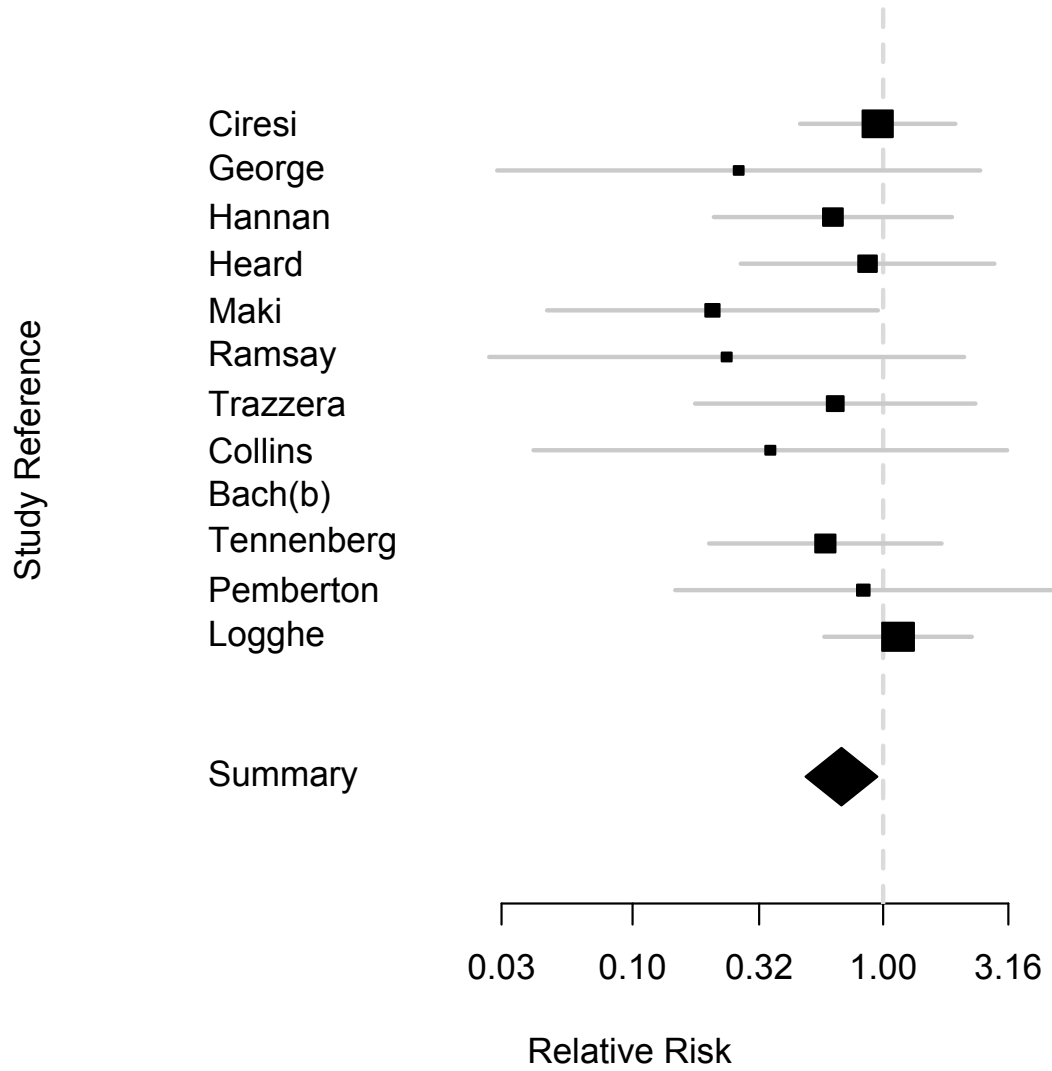
Conclusion is clear: the trials consistently show a substantial effect of treatment.

In R: `plot(a)`

or

```
metaplot(a$logRR, a$selogRR, nn=a$selogRR^-2, labels=a$names,
         xlab="Relative Risk", summn=a$logMH, sumse=a$selogMH,
         logeffect=TRUE, sumnn=a$selogMH^-2)
```

Meta-analysis



Preattentive perception

Some visual differences 'pop out' with no conscious effort — processed 'preattentively'

Preattentive perception allows groups of points to be seen as groups, rather than identified one-by-one, so patterns are visible.

Color, size, basic shape, angle, shading are preattentive. Letters and numbers are not.

Not possible to have **many** preattentively distinct groups even if they are pairwise distinct, even by combining dimensions. More than 4 groups is pushing it.

[demo of preattentive perception]

Color

Human color vision is three-dimensional: light–dark, red–green, blue–yellow.

The red–green dimension is weak or absent in about 7% of men.

The blue–yellow dimension has lower spatial resolution, and reproduces less accurately across media.

Some visual processing only works with light–dark differences.

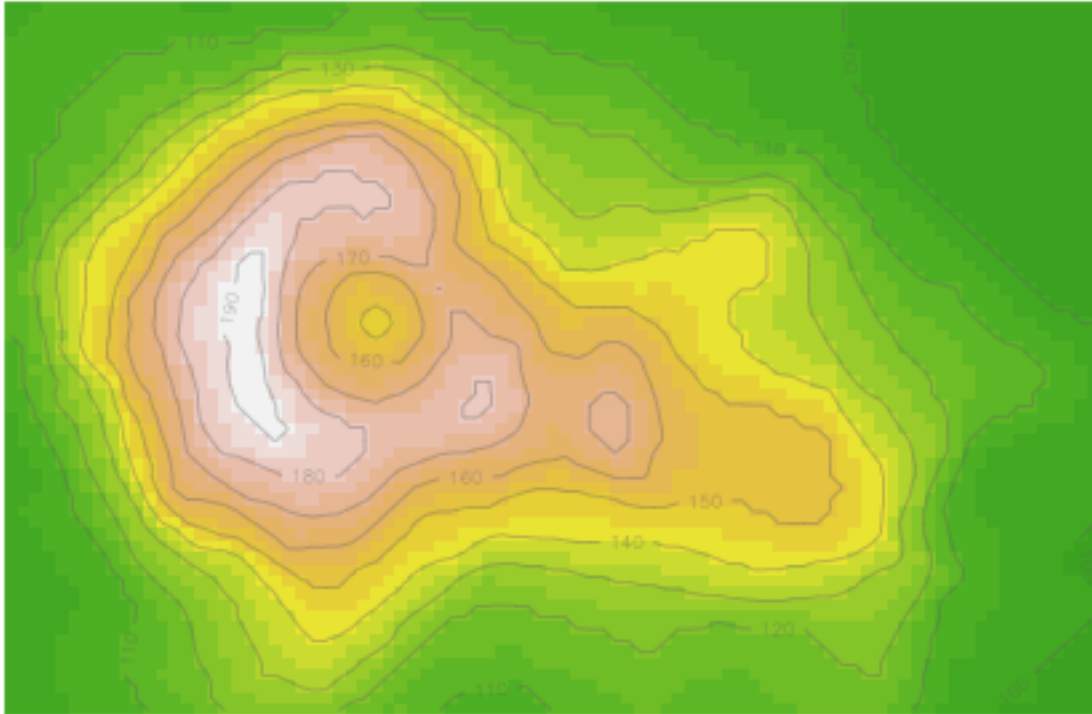
Color choice

Avoid graphs that are unreadable or misleading to dichromats.
(R dichromat package)

'Smooth' color shadings should be smooth in a metric relevant to perception, not RGB or CMYK (R colorspace, colorbrewer.org)

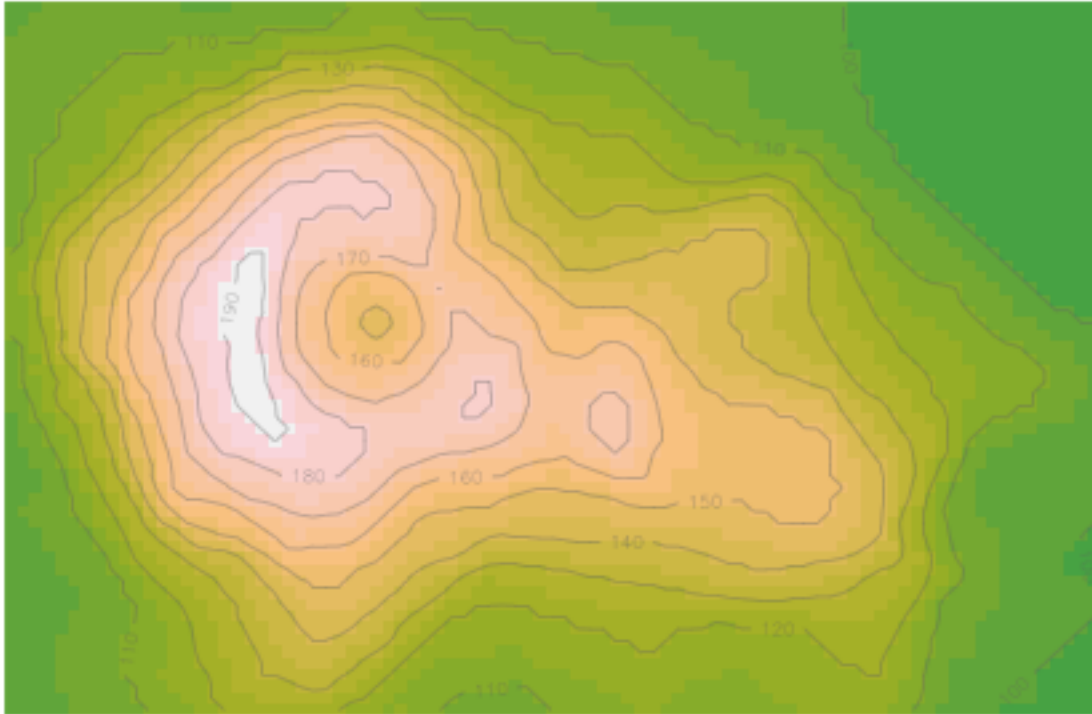
Sequential (low→high) or diverging ($- \leftarrow 0 \rightarrow +$) shadings should have luminance, chroma components, not just hue.

Mt Eden: RGB palette



```
filled.contour(volcano, col=terrain.colors)
```

Mt Eden: LUV palette



```
filled.contour(volcano, col=terrain_hcl)
```

'Manhattan' plots

```
stripe<-function(p,chromosome,main=""){
  chromcode <- c(1:22,"X","Y","XY")

  logp <- -log(p,10)

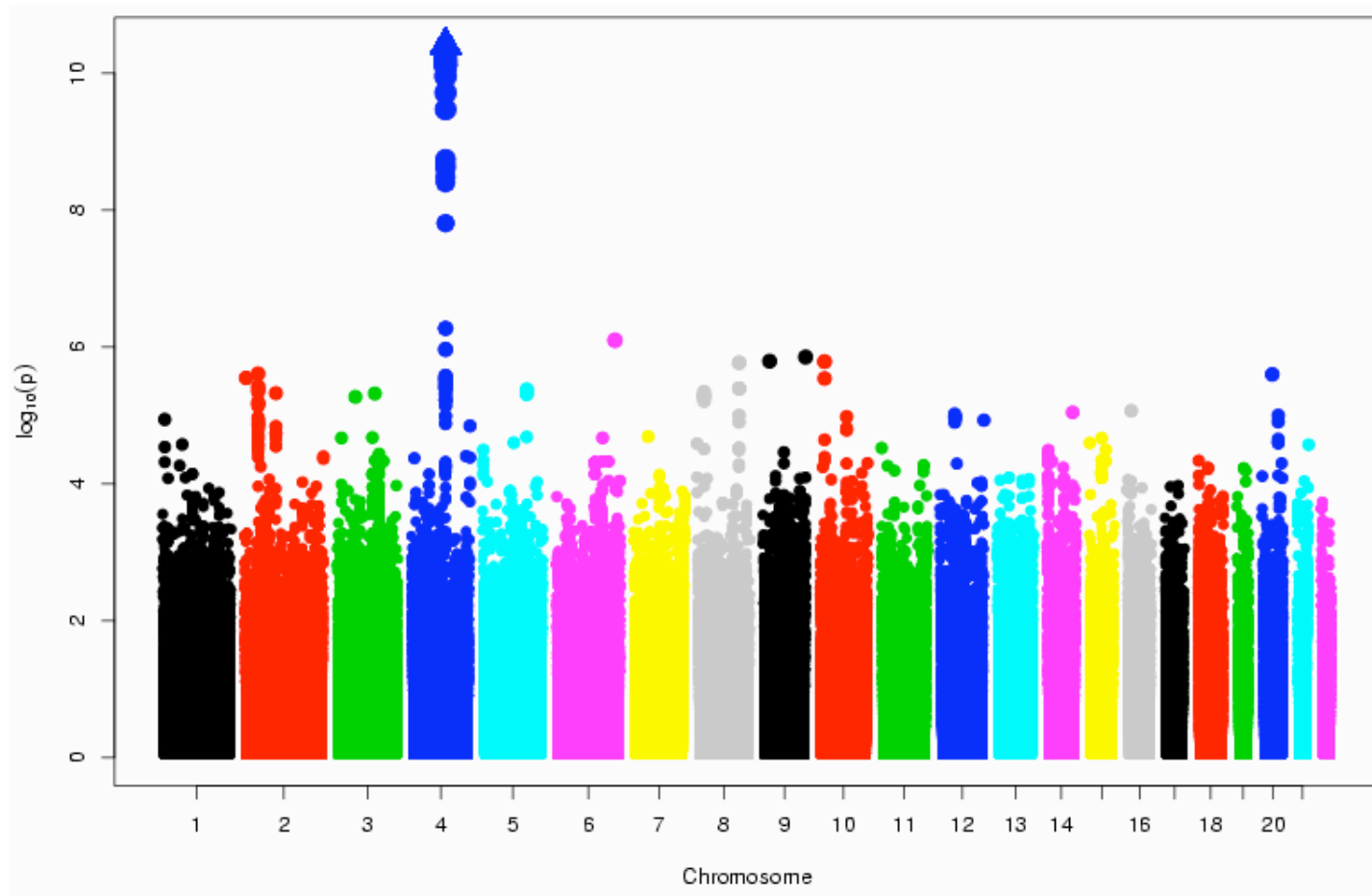
  N<-length(logp)
  ymax<-log(N,10)+4

  chromstart<-which(c(1,diff(chromosome))==1)
  chromend<-c(chromstart[-1],N)

  x<-(1:N)+chromosome*(chromend[1]/6)
  y<-pmin(ymax,logp)
  plot(x,y,cex=0.3+2*y/ymax, col=chromosome, pch=ifelse(y==ymax,24,19),
       bg=chromosome, xlab="Chromosome",ylab=quote(log[10](p)),
       xaxt="n",ylim=c(0,ymax),main=main)

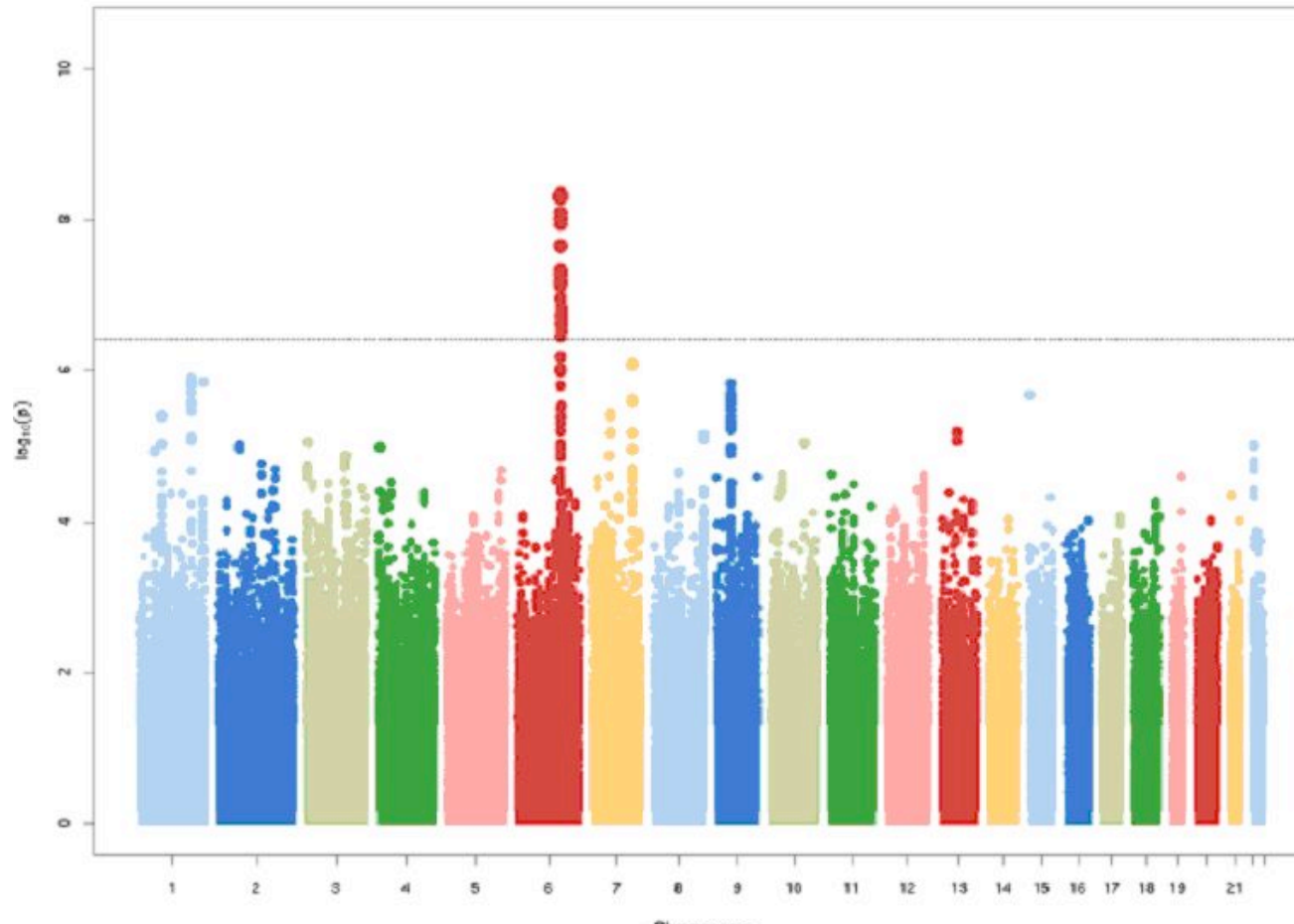
  centers<- (x[chromstart]+x[chromend]-(chromend[1]/6))/2
  centers[25]<-x[N]
  axis(1,at=centers,label=chromcode)
}
```

'Manhattan' plots



Standard R palette

'Manhattan' plots



ColorBrewer *Paired* palette

Better color choices

Cindy Brewer (Penn State) developed a set of color palettes for maps, and <http://www.colorbrewer.org> provides a tool for choosing them. The color schemes are in the RColorBrewer package.

The vcd package provides automated methods for constructing smooth color ramps: `sequential_hcl()`, `diverging_hcl()`. These give colors falling along straight lines in CIE Luv colorspace (nearly the same as CIE Lab).

`dichromat()` suppresses red-green distinctions to show whether a graph will be readable to people with red-green color blindness or anomalous color vision (about 7% of men; two variants 'protanopia', 'deuteranopia')

Better color choices

ColorBrewer - Selecting Good Color Schemes for Maps

http://www.personal.psu.edu/cab38/ColorBrewer/ColorBrewer.html

Most Visited ▾ BBC News ▾ NYT PubMed Home http://examinedlife... https://svn.r-projec... Washington Associa... Data Collections — S...

Schneier on Security ColorBrewer - Selecting Good C... WebPine - WebPine

number of classes Step1
◀ 5 ▶
[learn more](#)

legend type Step2
sequential diverging
qualitative [learn more](#)

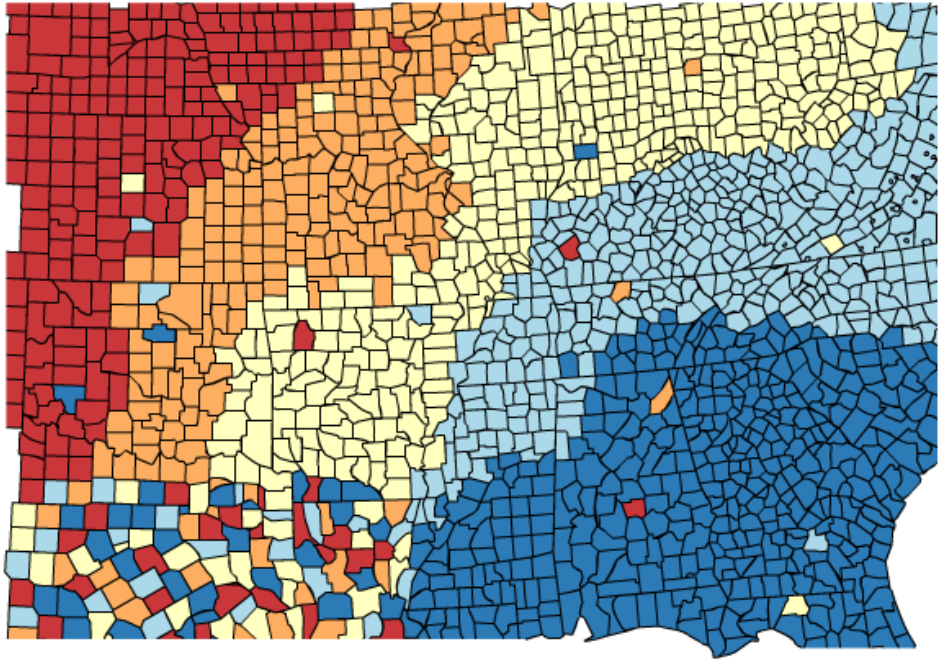
mini legends Step3
[learn more](#)

cmyk rgb hex Lab AV3
color specs print

ColorBrewer

5-class diverging RYB

dg QG
[how to use](#) [updates](#)
[credits](#) [reset view](#)
[about map](#)



map zoom map borders on city symbols on road network on
background color border color white black road network color
[learn more](#)

Done 1337

Image plots

`image()` plots colors to indicate values on a two-dimensional grid. A red-green spectrum is often used to code the values (especially in genomics). Use the `dichromat` package to compare red:green and blue:yellow spectra.

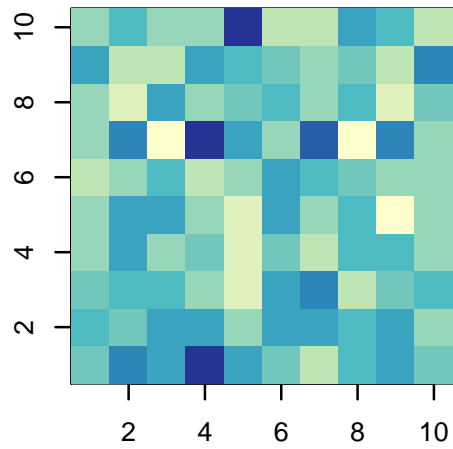
```
par(mfrow=c(2,3))
x<-matrix(rnorm(10*10),10)
bluescale<-colorRampPalette(c("#FFFFCC", "#C7E9B4", "#7FCDBB", "#40B6C4",
    "#2C7FB8" , "#253494"))
redgreen<-colorRampPalette(c("red", "green3"))

image(1:10,1:10,x, col=bluescale(10),
main="blue-yellow scale")
image(1:10,1:10,x, col=dichromat(bluescale(10)), main="deutan")
image(1:10,1:10,x,col=dichromat(bluescale(10),"protan"), main="protan")

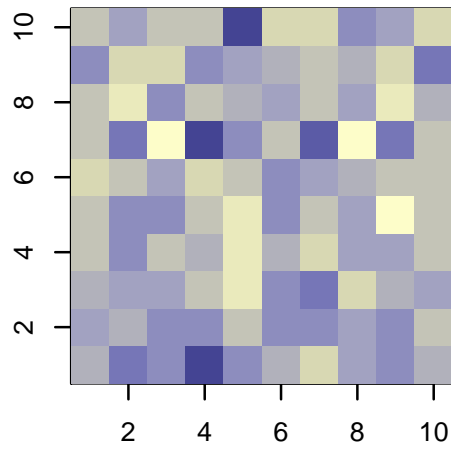
image(1:10,1:10,x, col=redgreen(10),
main="red-green scale")
image(1:10,1:10,x, col=dichromat(redgreen(10)), main="deutan")
image(1:10,1:10,x, col=dichromat(redgreen(10),"protan"), main="protan")
```

Image plots

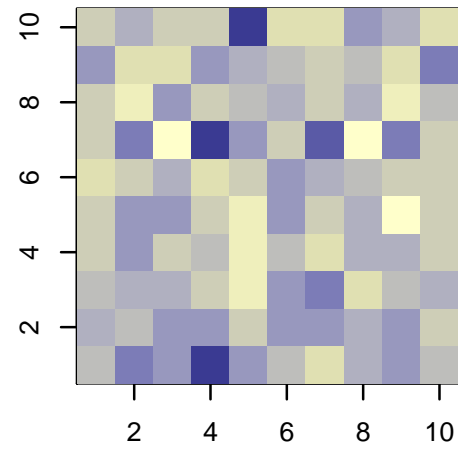
blue–yellow scale



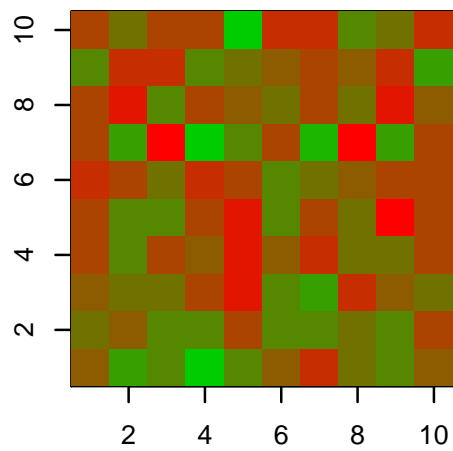
deutan



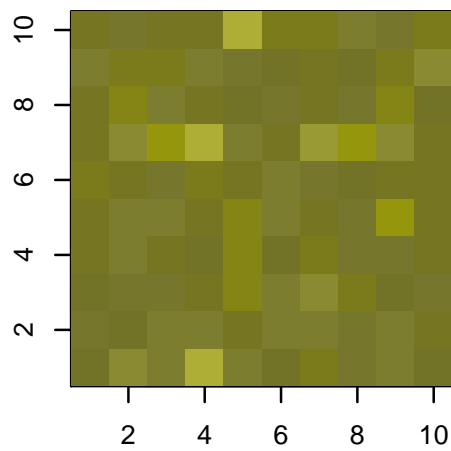
protan



red–green scale



deutan



protan

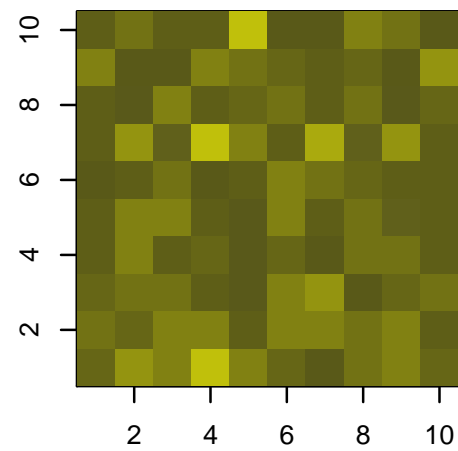


Image plots

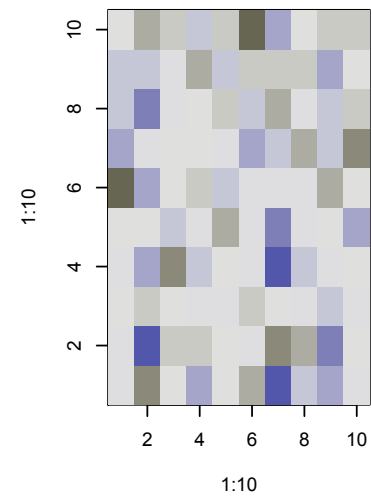
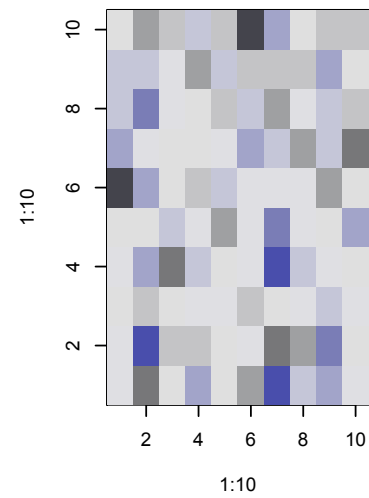
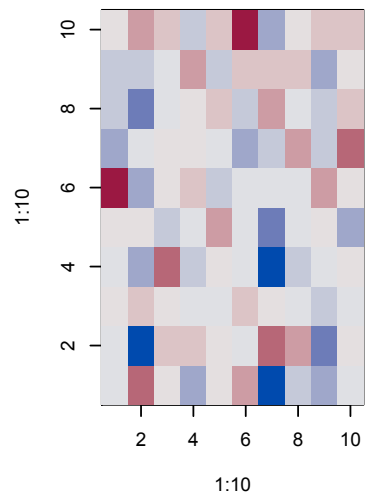
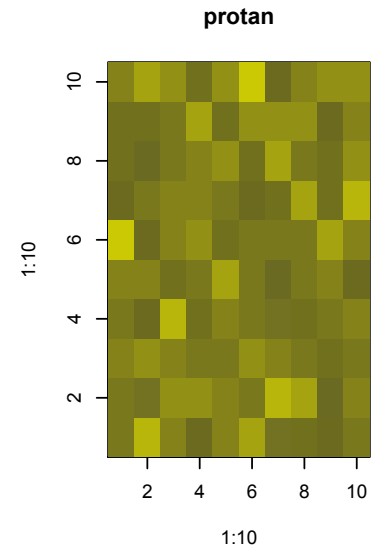
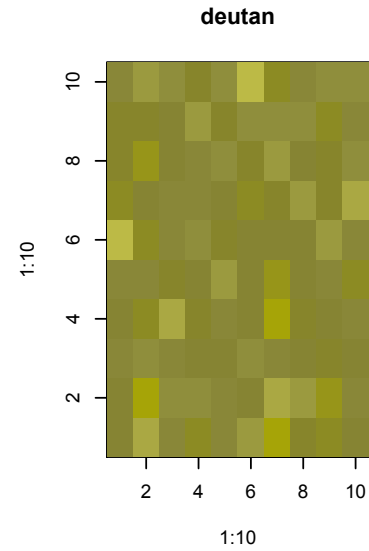
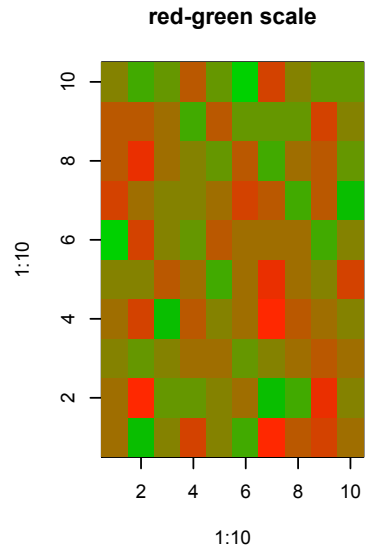
The `bluescale()` color scale is taken from ColorBrewer.

The 'colorspace' package also provides color scales using perceptually-based color spaces

Functions: `diverge_hcl` and `sequential_hcl`

The default `diverge_hcl` is a red-white-blue color scale

Image plots



Conditioning plots

Cleveland's Trellis graphics: a systematic model for conditioning plots, in R **lattice** package.

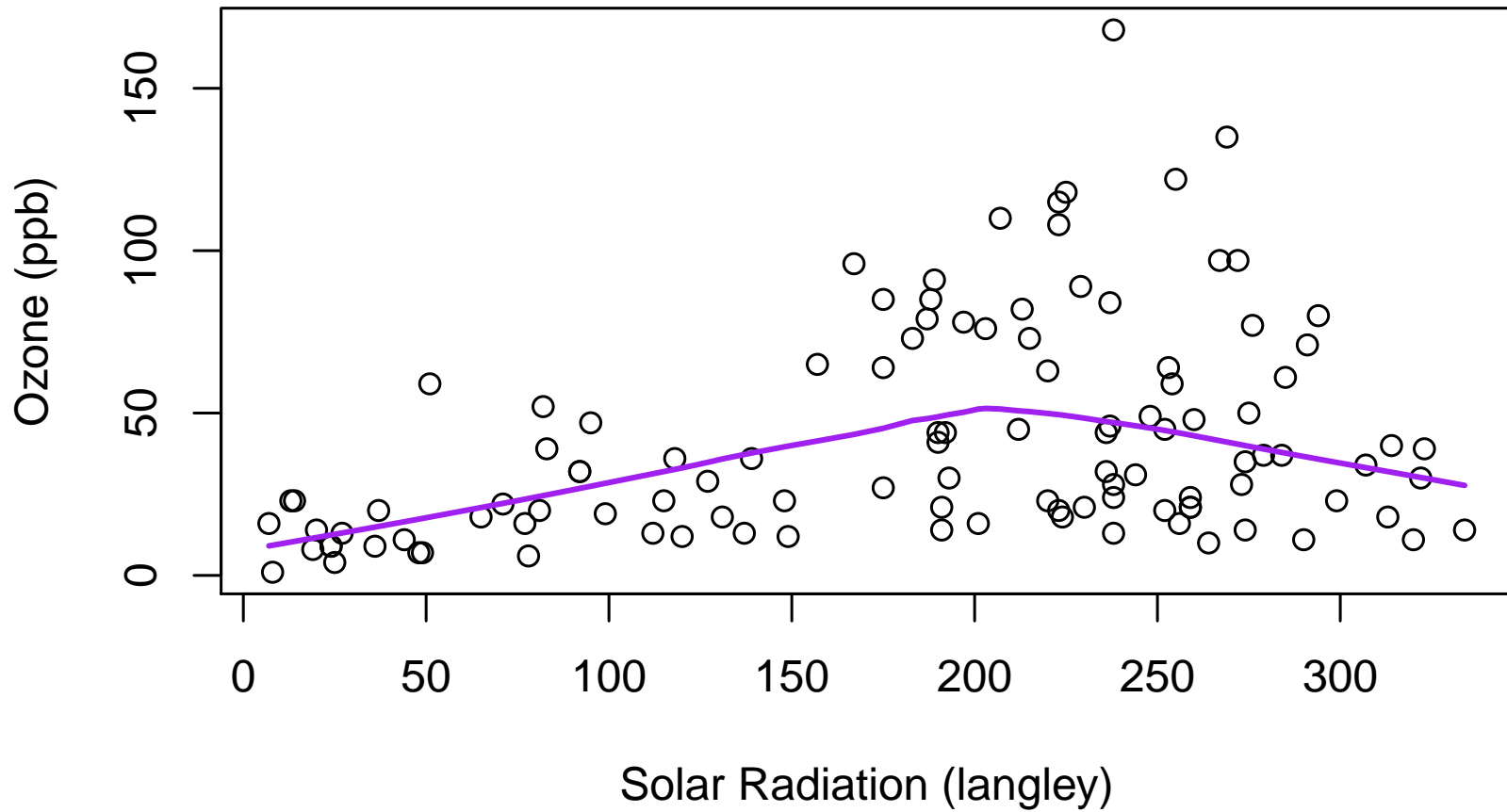
- Slicing: visualizing a 3 or 4 dimensional data cloud in slices
- Regression adjustment: relationship between y and x holding z constant

`coplot()` does scatterplots, the lattice package does everything.

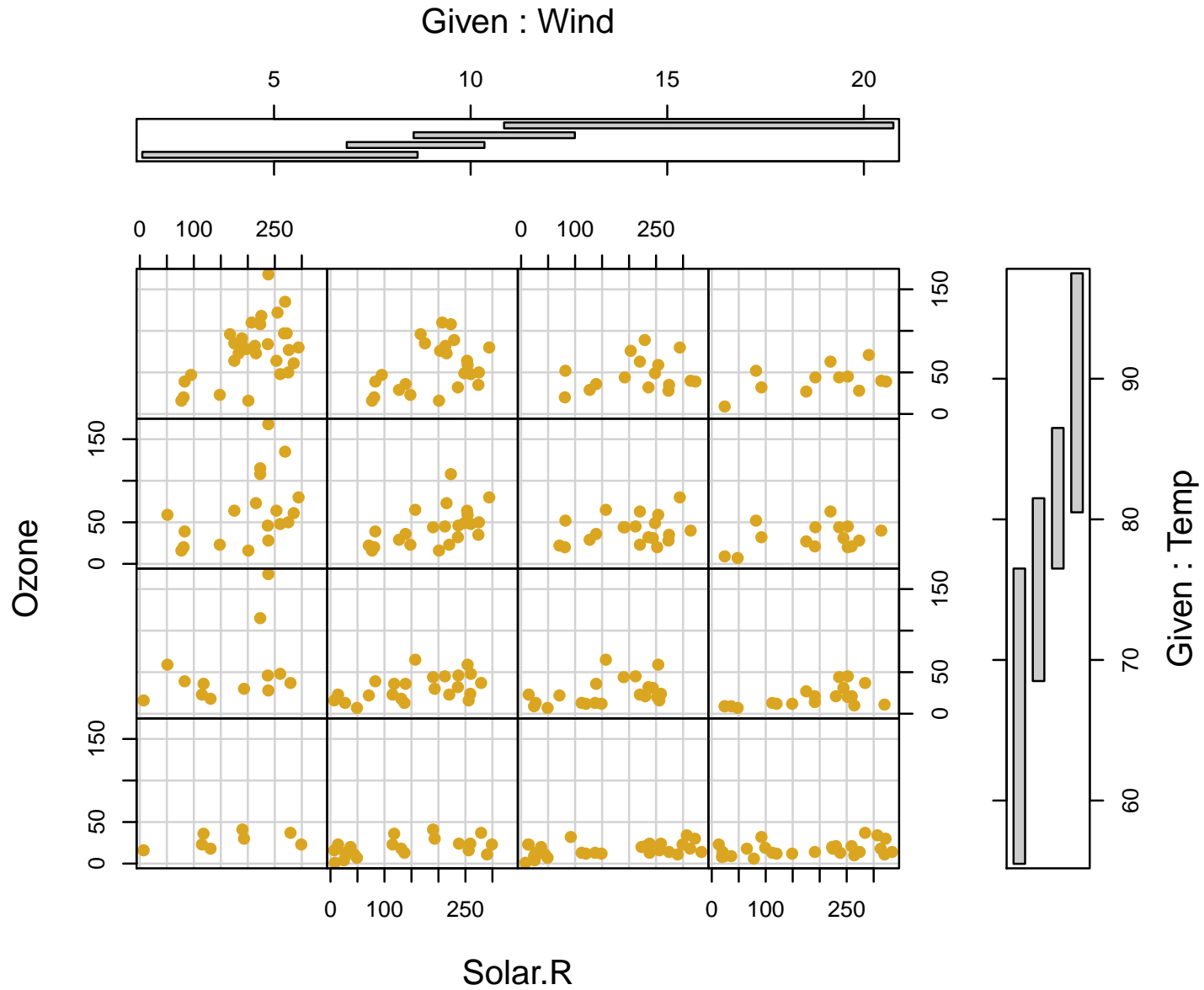
```
coplot(Ozone~Solar.R|Wind*Temp, data=airquality)
```

Example: adjustment

New York, Summer 1979



Example: adjustment

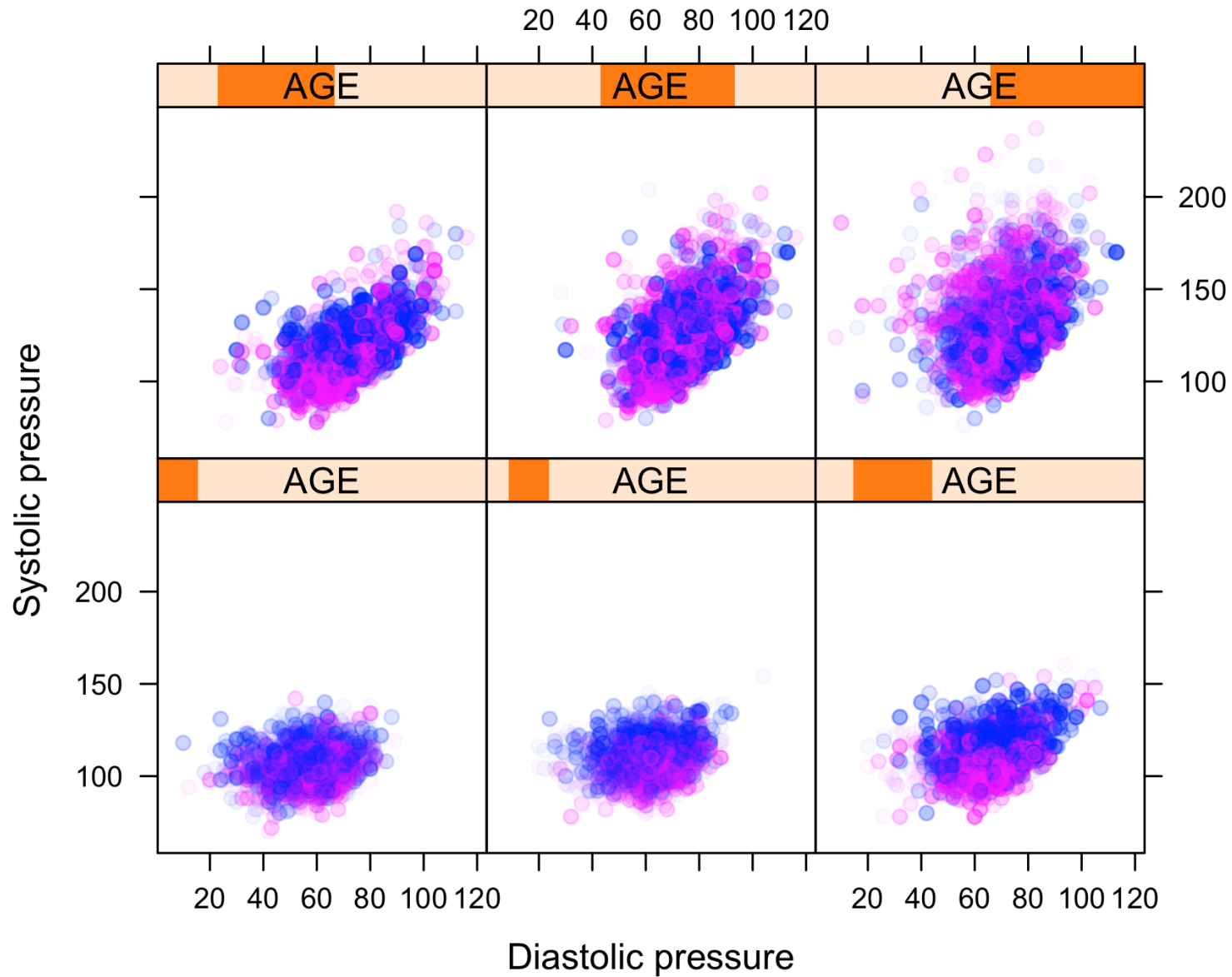


Aggregation and binning

Simple scatterplots don't work for large data sets: overplotting

- Transparent color: points are partially transparent. Works well on screen. Specify colors by "`#RRGGBBAA`" hexadecimal, or use `rgb()` to construct them
- Density estimation
- Hexagonal binning (Dan Carr; `hexbin` package): aggregate into hexagonal bins. Works well on paper.

Blood pressure by age and gender



Useful function

```
fade<-function(colors, alpha){  
  rgbcols <- col2rgb(colors)  
  rgb(rgbcols[1,], rgbcols[2,], rgbcols[3,], alpha, max=255)  
}
```

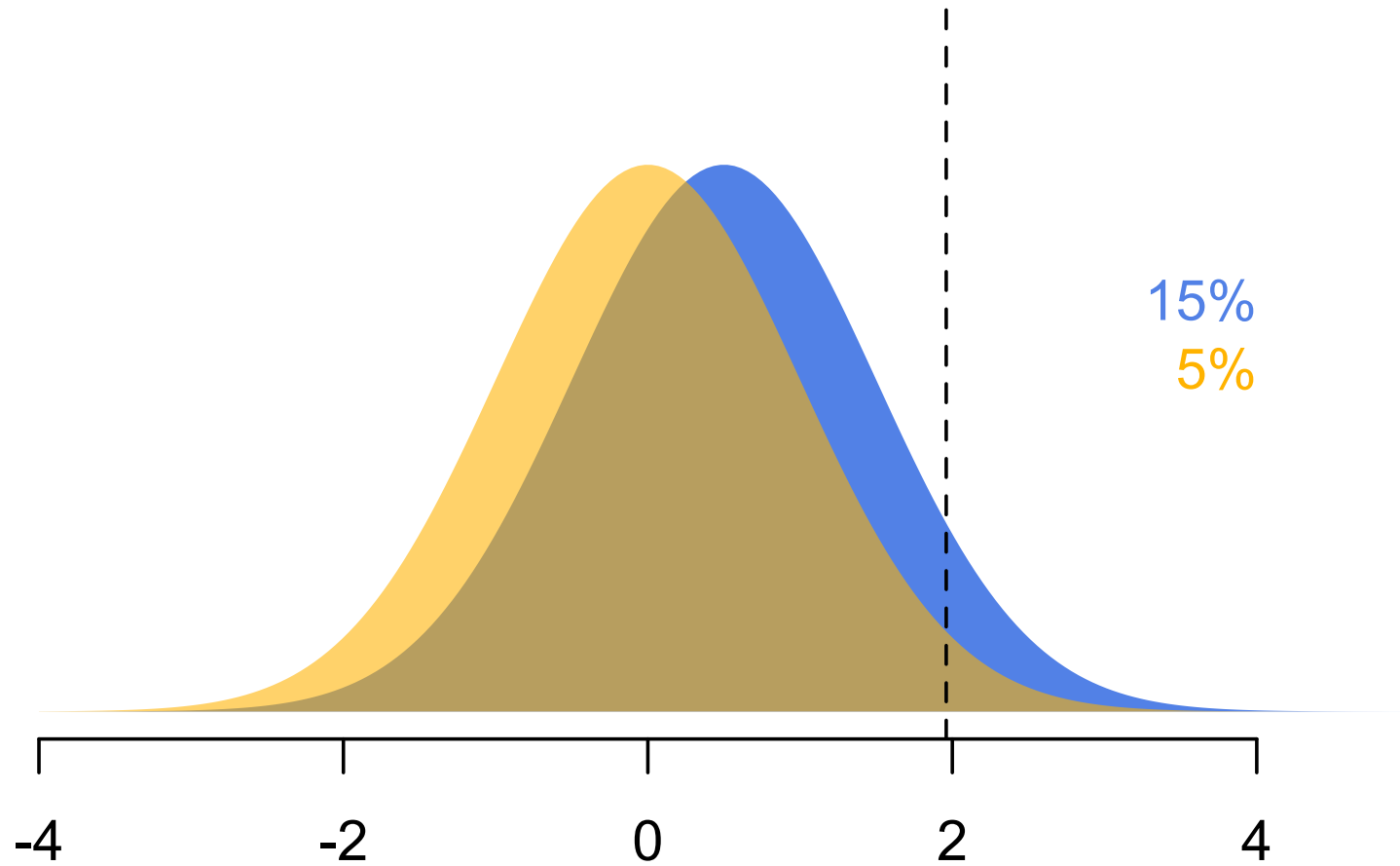
eg `fade("tomato",128)` to get a 50% transparent orange-red

Overlapping and transparency

```
quartz(height=3,width=5)
par(mar=c(3,2,1,1))
x<-seq(-4,5,length=200)
d1<-dnorm(x)
d2<-dnorm(x,m=0.5)

plot(x,d1,type="n",ylim=c(0,0.5),bty="n",yaxt="n",ylab="")
polygon(c(5,-4,x),y=c(0,0,d2),col=fade("royalblue"),border=NA)
polygon(c(5,-4,x),y=c(0,0,d1),col=fade("orange",150),border=NA)
abline(v=1.96,lty=2)
text(4,0.3,adj=1,"15%",col="royalblue")
text(4,0.25,adj=1," 5%",col="orange")
```

Overlapping and transparency



Hexagonal binning

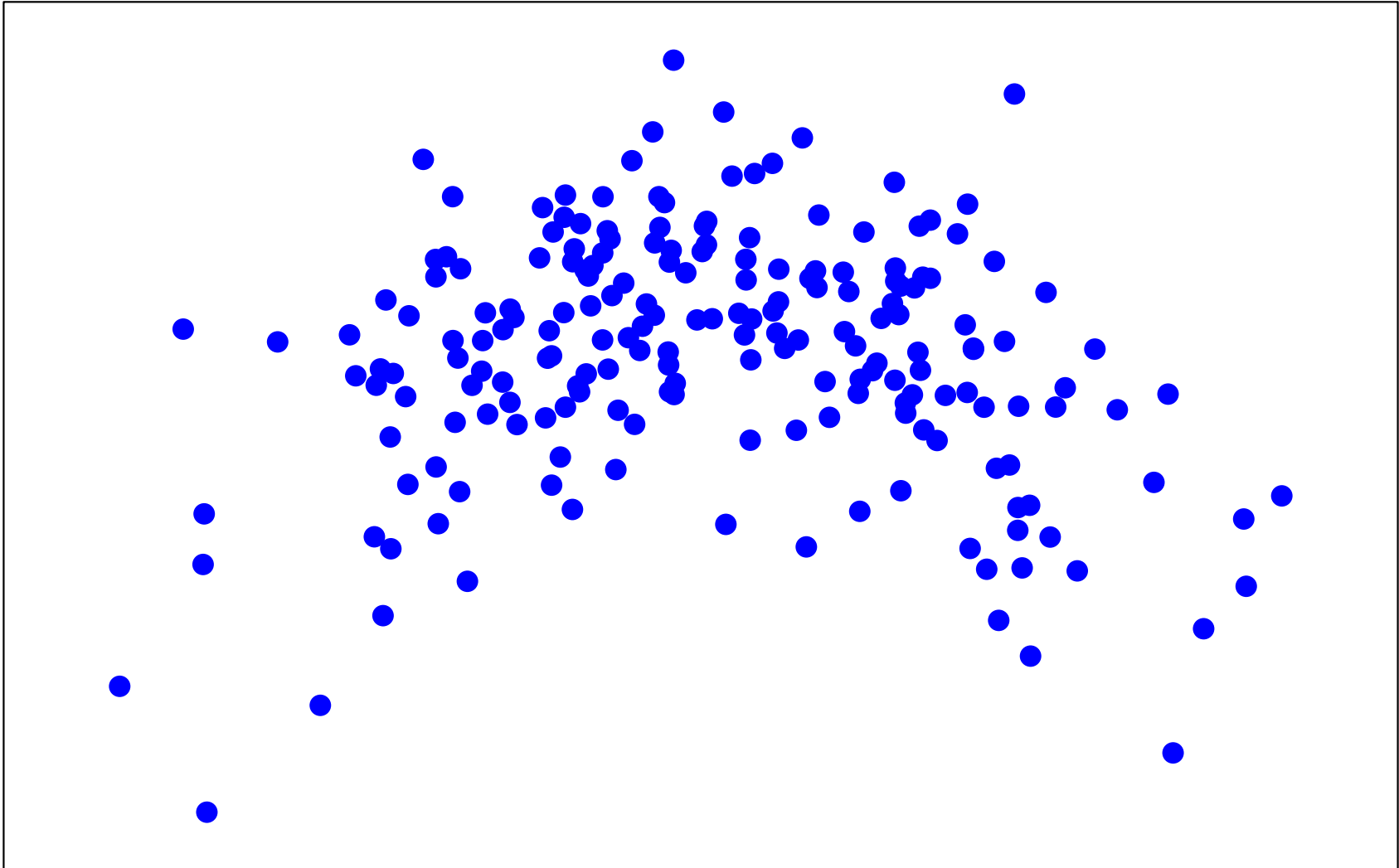
We don't *really* care about the exact location of every single point.

- How **many** points in one 'vicinity' compared to others?
- Any 'outliers' far from all other data points?

In one dimension, histograms answer these questions by **binning** the data

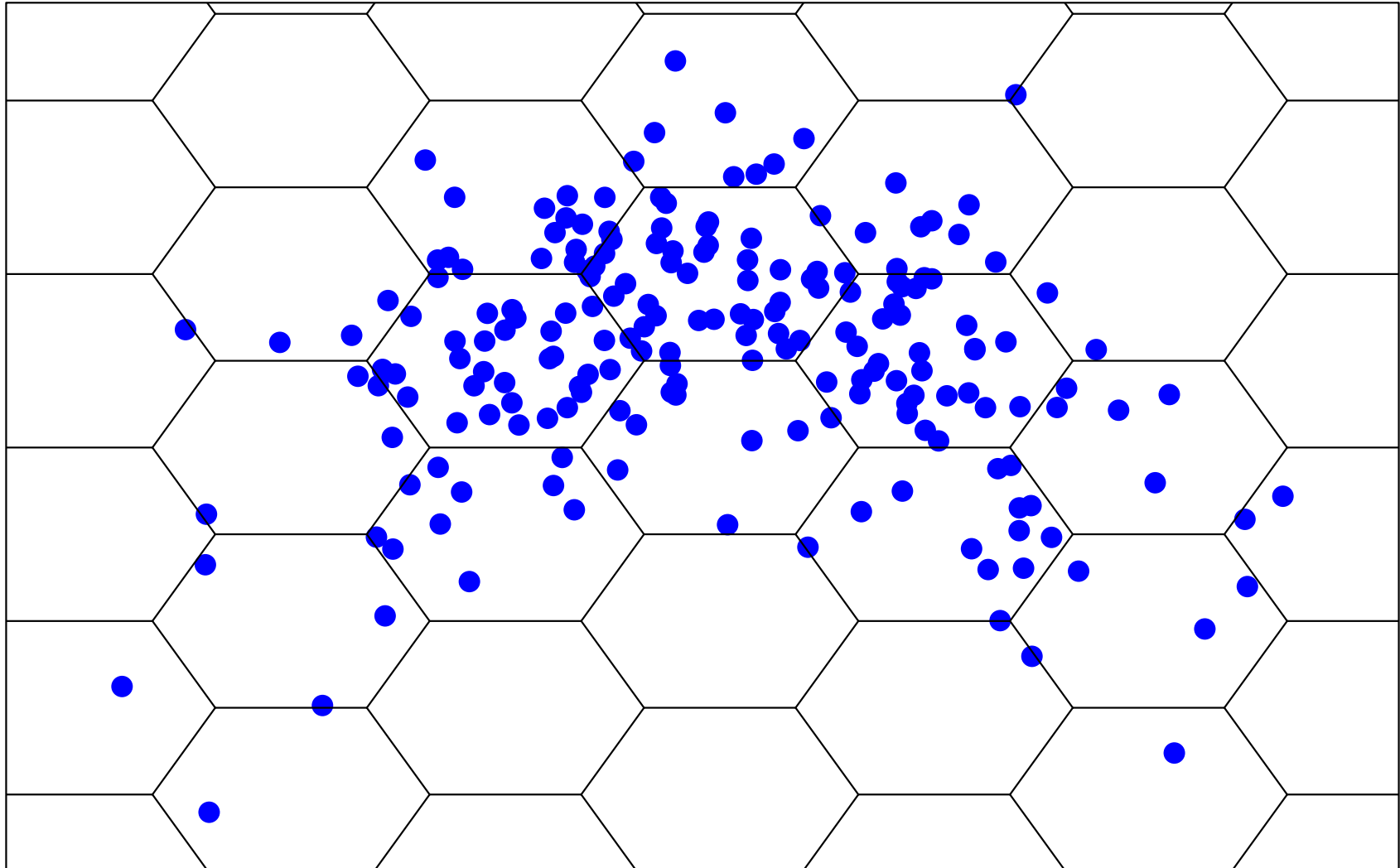
Hexagonal binning

Binning in two dimensions;



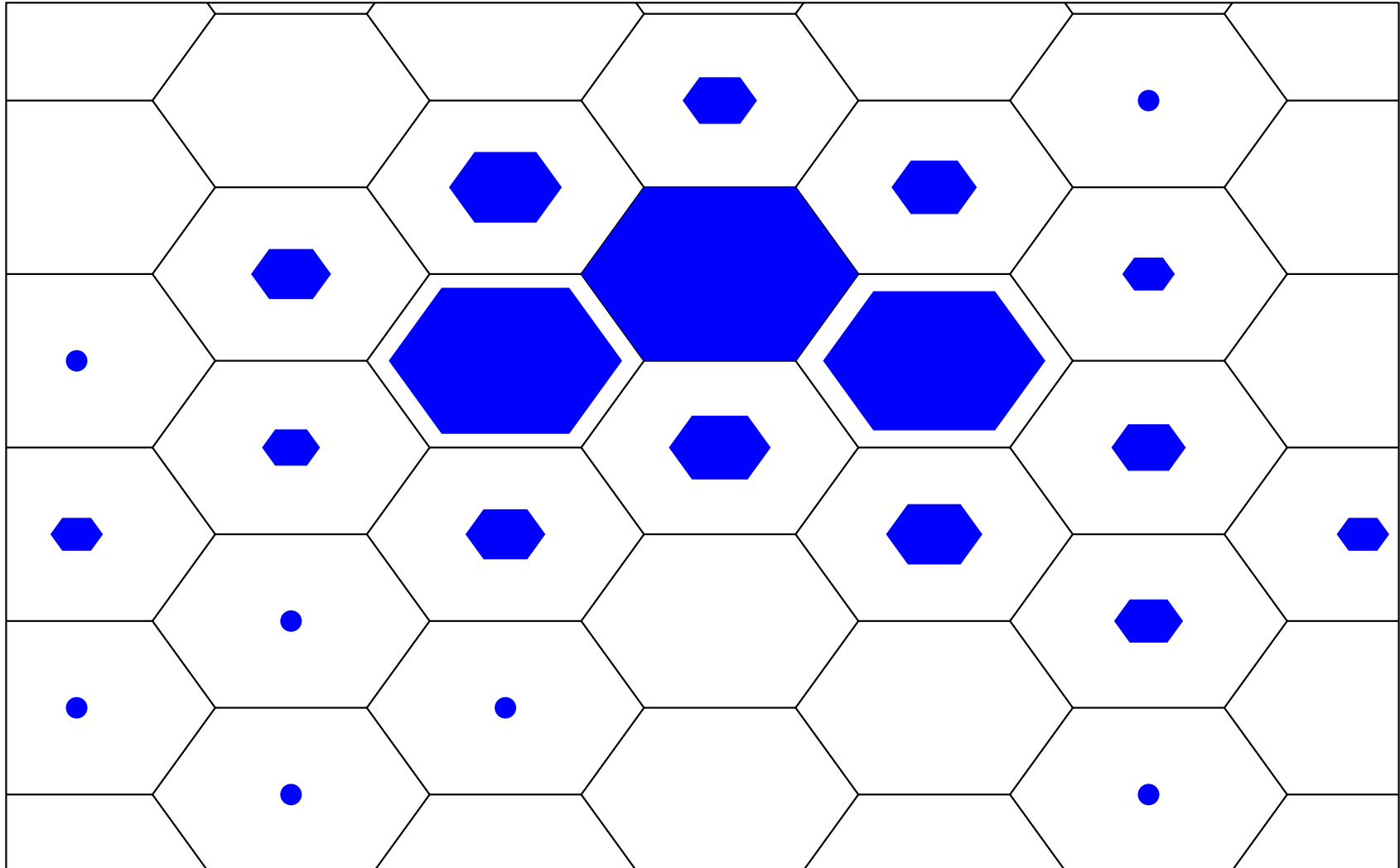
Hexagonal binning

Binning in two dimensions;



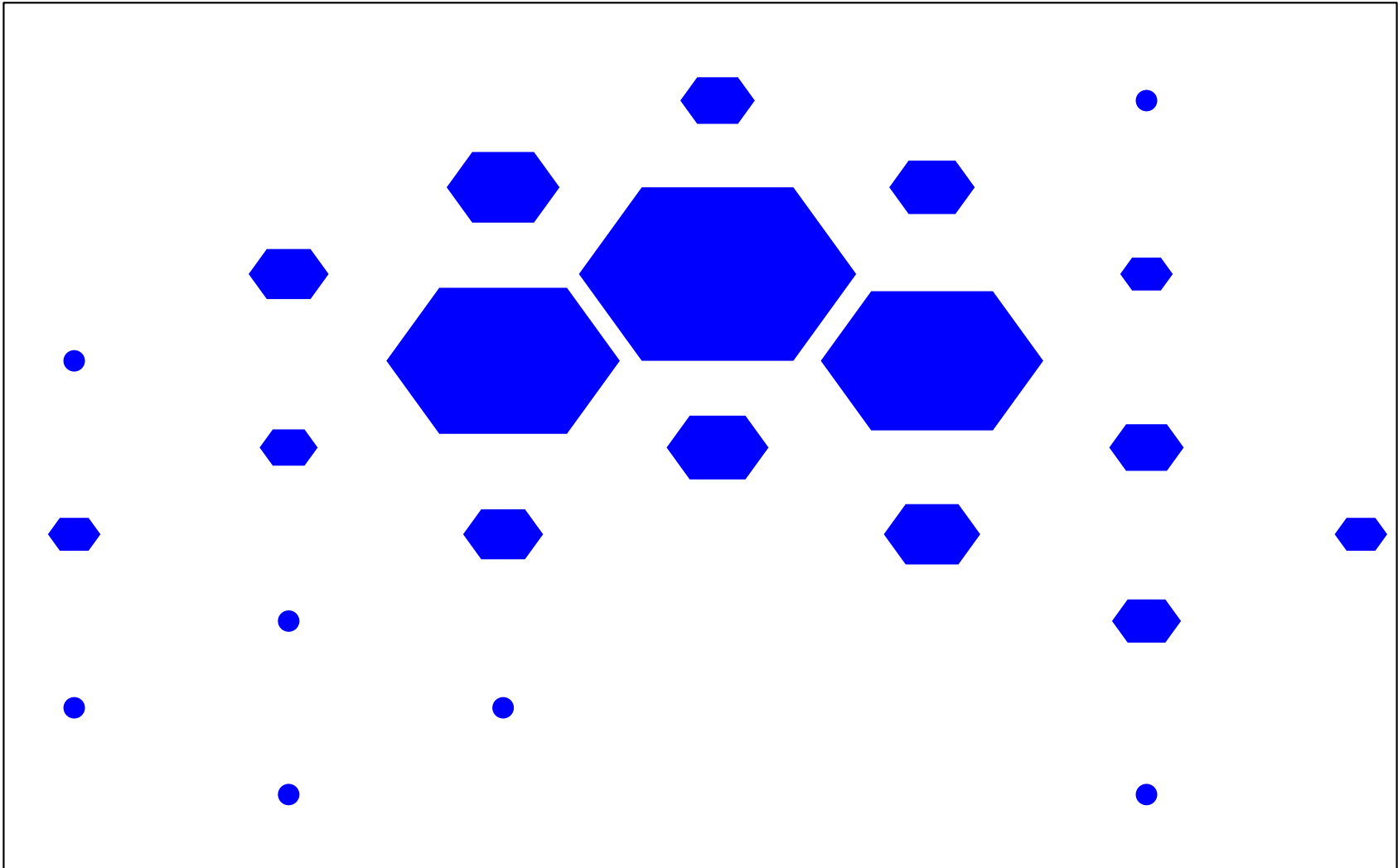
Hexagonal binning

Binning in two dimensions;



Hexagonal binning

Binning in two dimensions;



Example: California schools

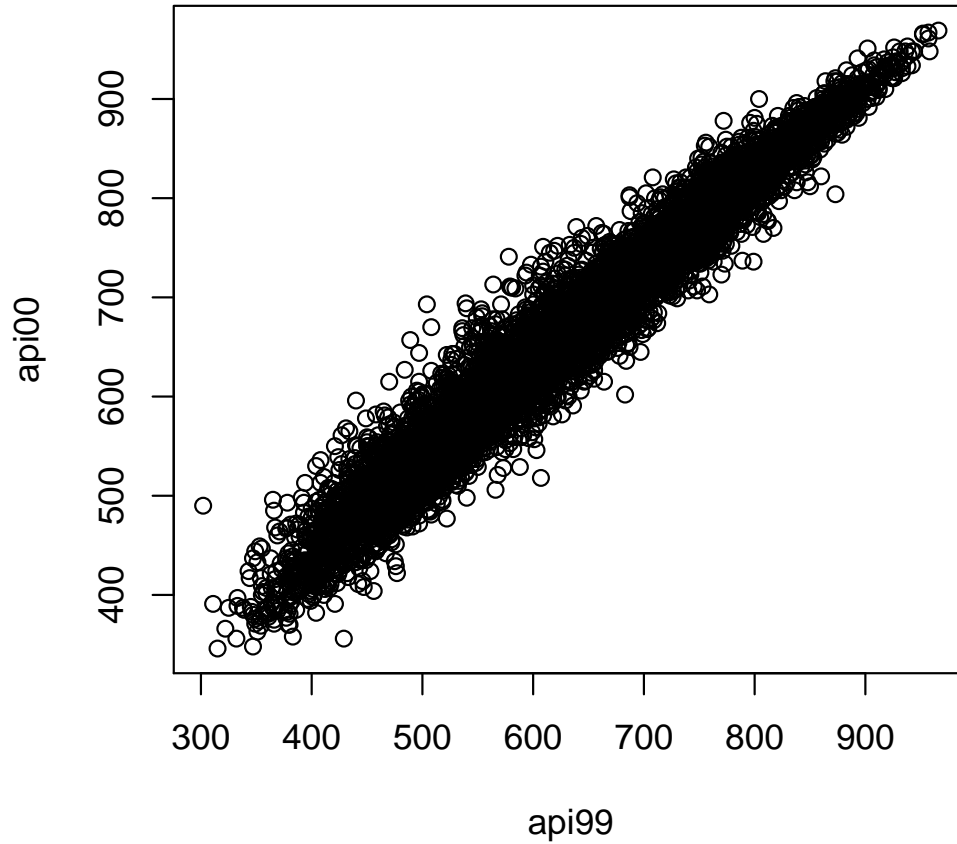
```
library(hexbin)
data(api, package="survey")

plot(api00~api99,data=apipop) # plain plot

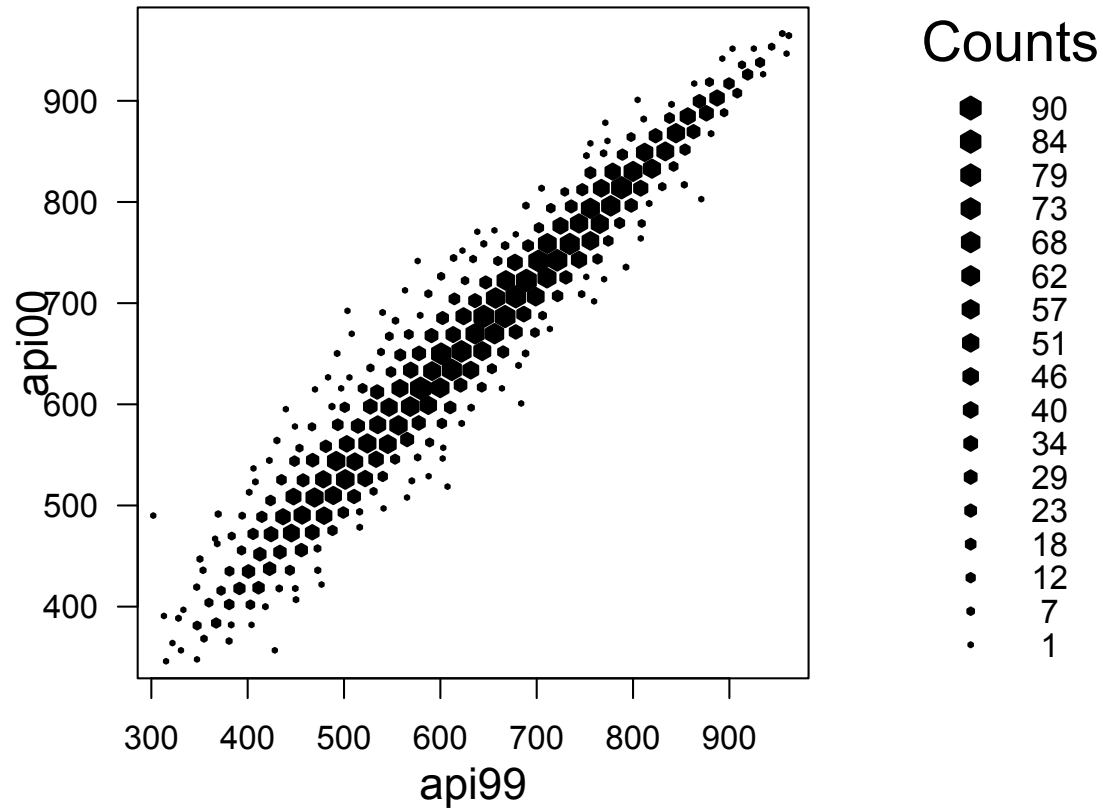
with(apipop, plot(hexbin(api99,api00), style="centroids"))

hexbinplot(api99~api00|equal.count(emer,4),
  style="centroids",data=apipop,xbins=20,colorkey=FALSE)
```

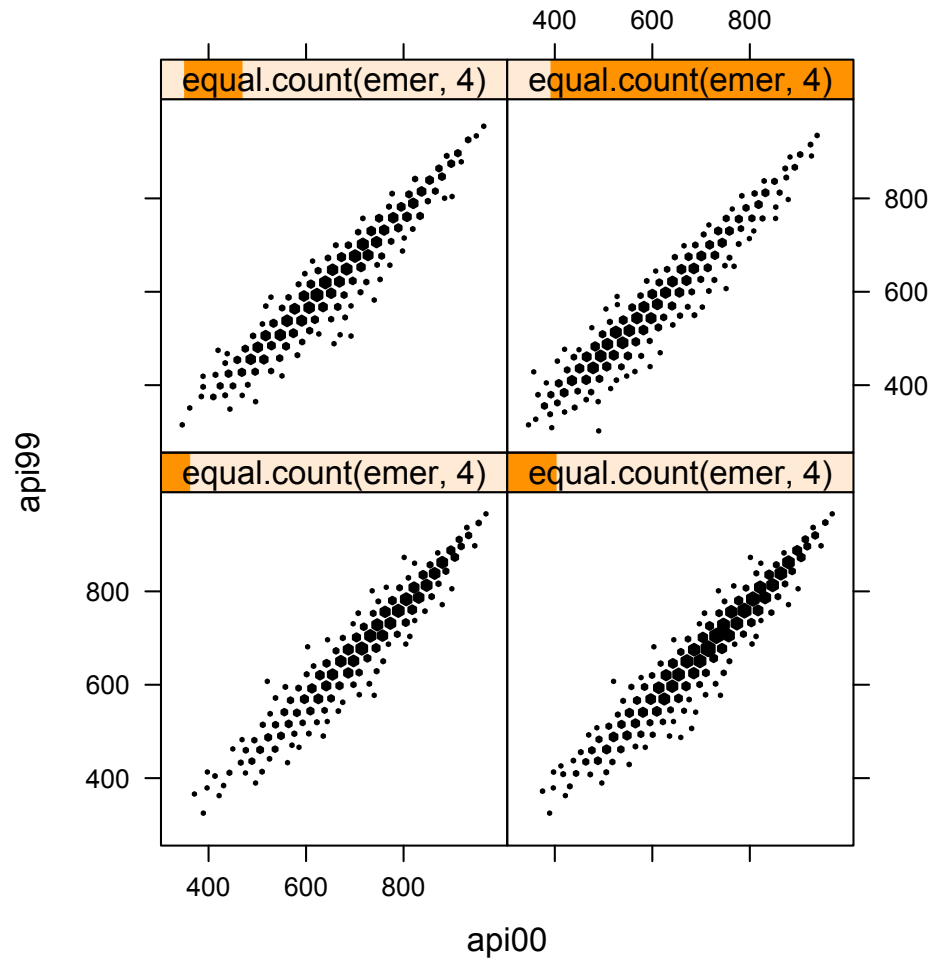
Example: California schools



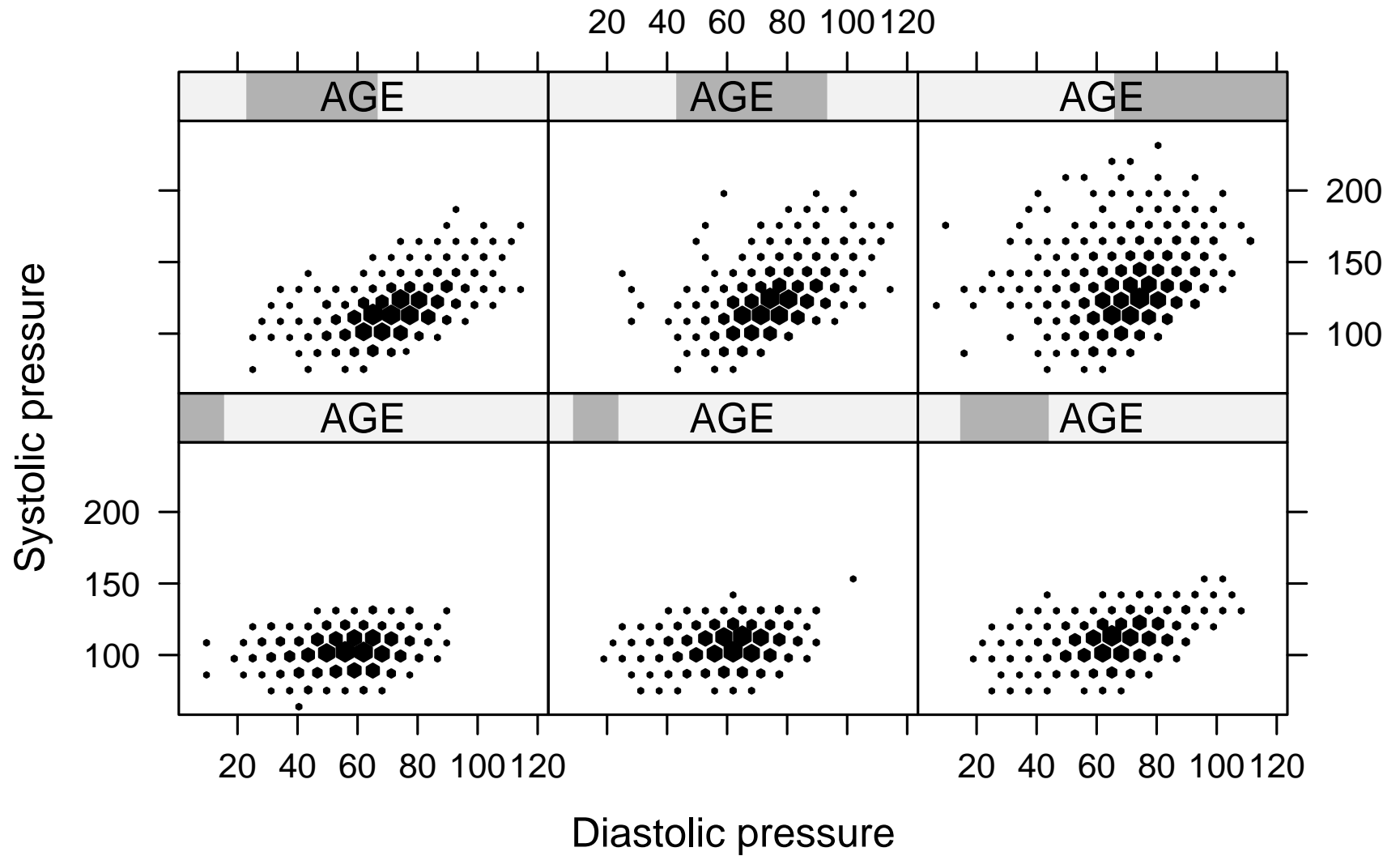
Example: California schools



Example: California schools



Blood pressure by age



High dimensions

1986 Data Expo (sponsored by ASA Graphics and Computing sections)

Data set was artificial data on "pollen grain measurements": 5 dimensions, 3848 points

```
> str(pollen)
```

```
'data.frame': 3848 obs. of 6 variables:
```

```
$ ridge : num -2.35 -1.15 -2.52 5.75 8.75 ...
```

```
$ nub : num 3.63 1.48 -6.86 -6.51 -3.9 ...
```

```
$ crack : num 5.03 3.24 -2.8 -5.15 -1.38 ...
```

```
$ weight : num 10.872 -0.594 8.463 4.348 -14.878 ...
```

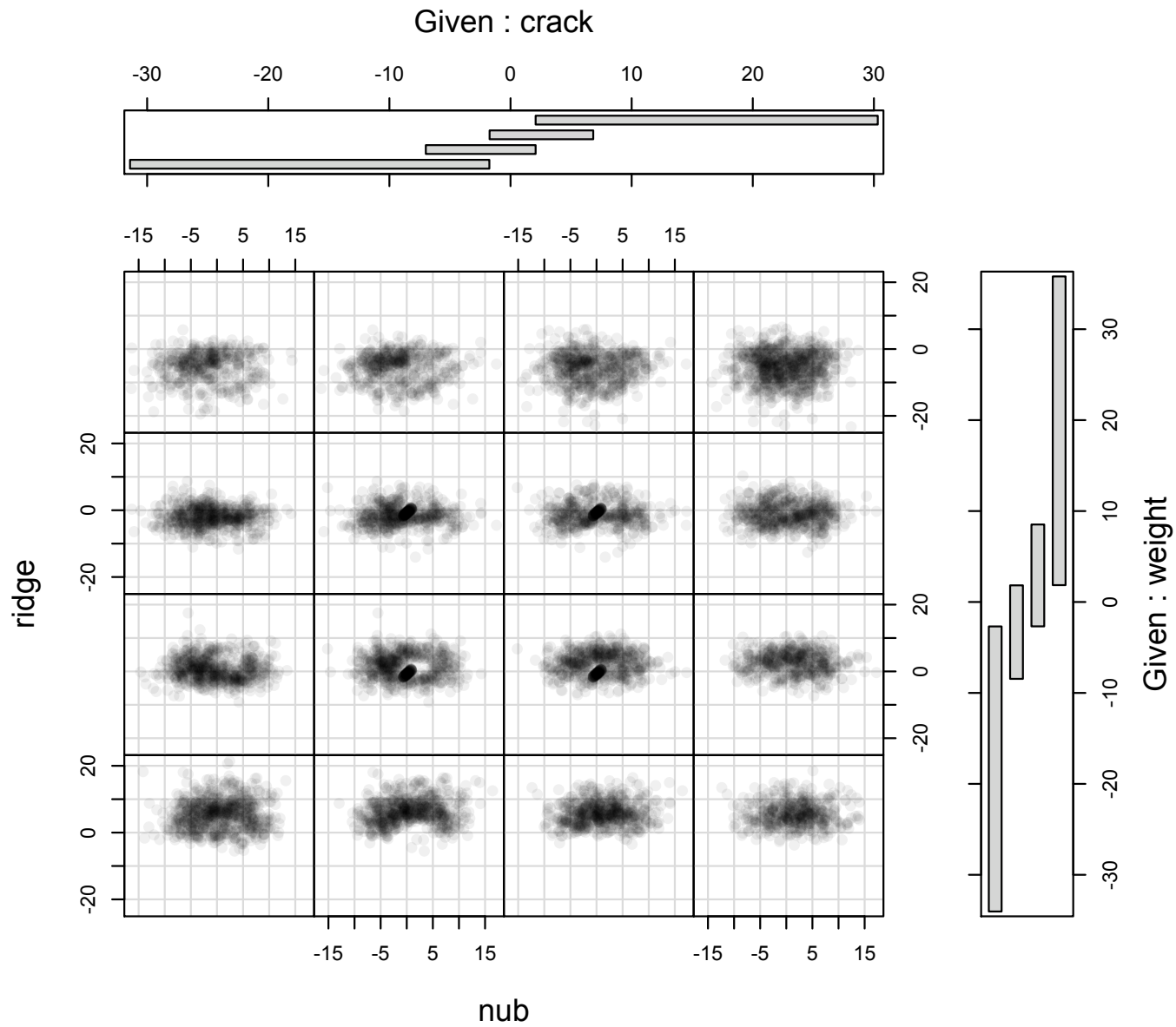
```
$ density: num -1.39 2.12 -3.41 -10.33 -2.42 ...
```

```
$ id : int 1 2 3 4 5 6 7 8 9 10 ...
```

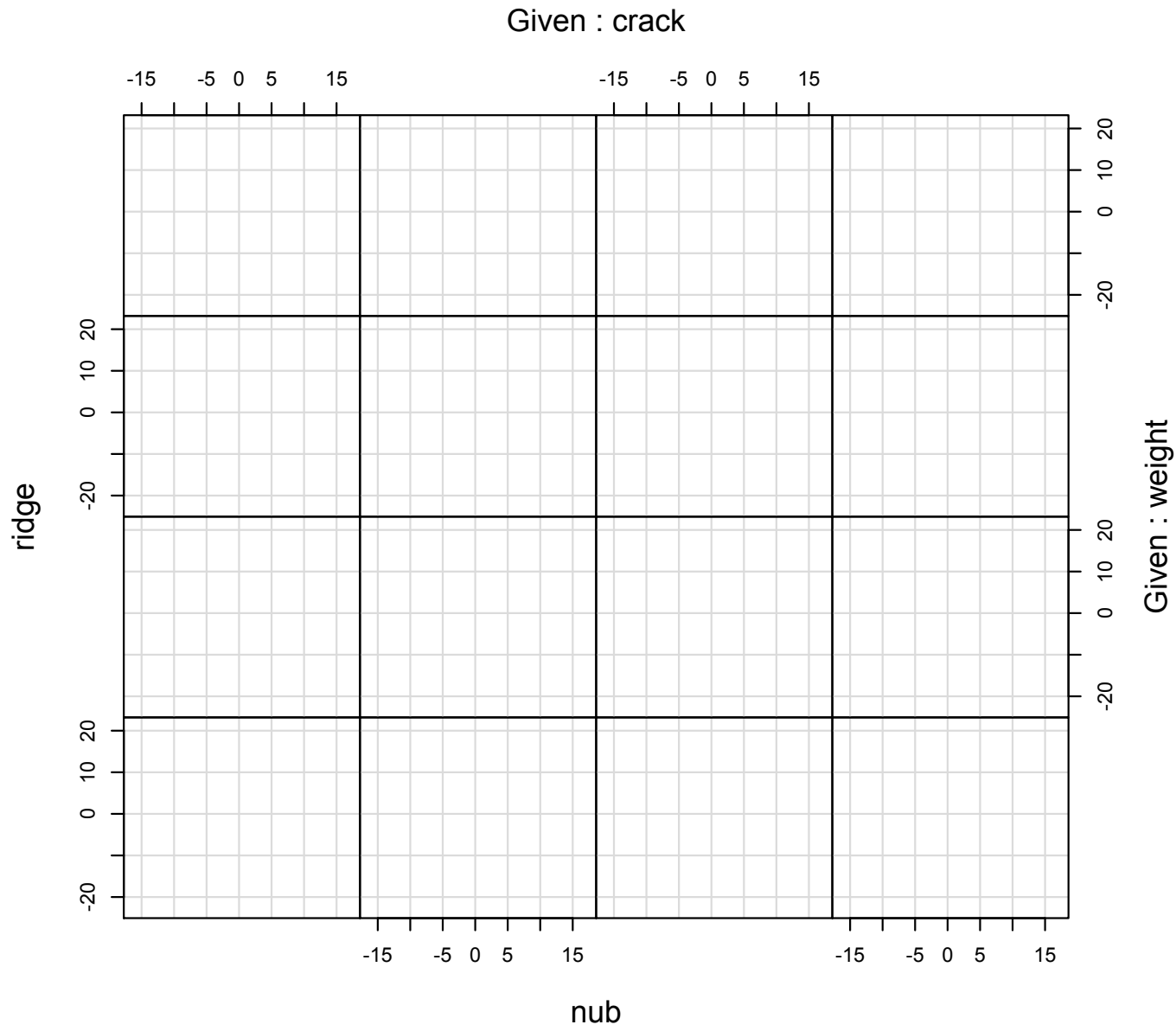
High dimensions

- > `coplot(ridge~nub|crack*weight,data=pollen,pch=19,
col="#00000010",n=4)`
- > `coplot(ridge~nub|crack*weight,data=pollen,pch=19,
col="#00000001",n=4)`
- > `hexbinplot(ridge~nub|equal.count(crack,6)*equal.count(weight,4),
data=pollen,strip=FALSE)`
- > `plot(density~ridge, data=pollen,
subset=abs(crack<2) & abs(weight<2) & abs(nub<2) &
abs(ridge<2) & abs(density)<2)`

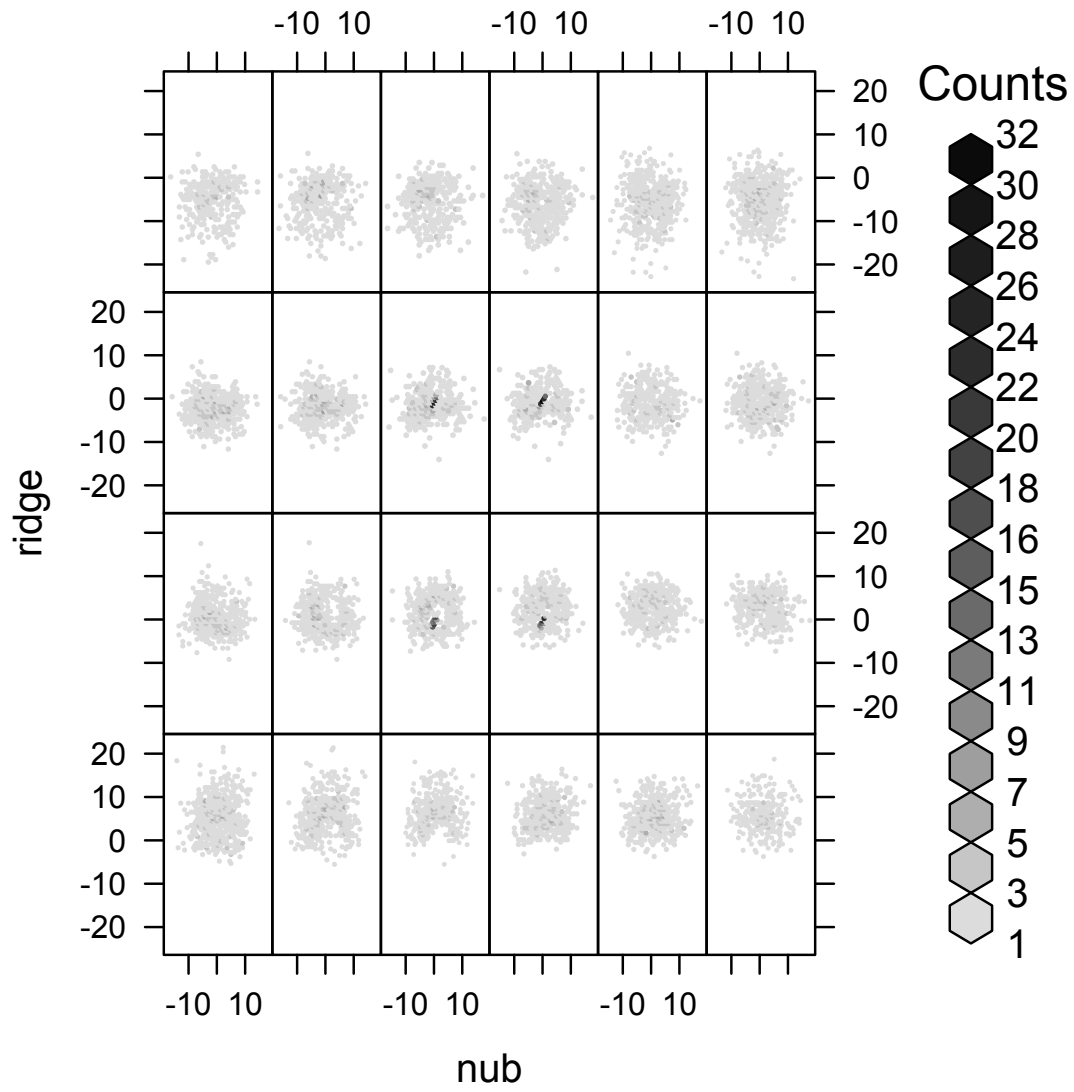
High dimensions



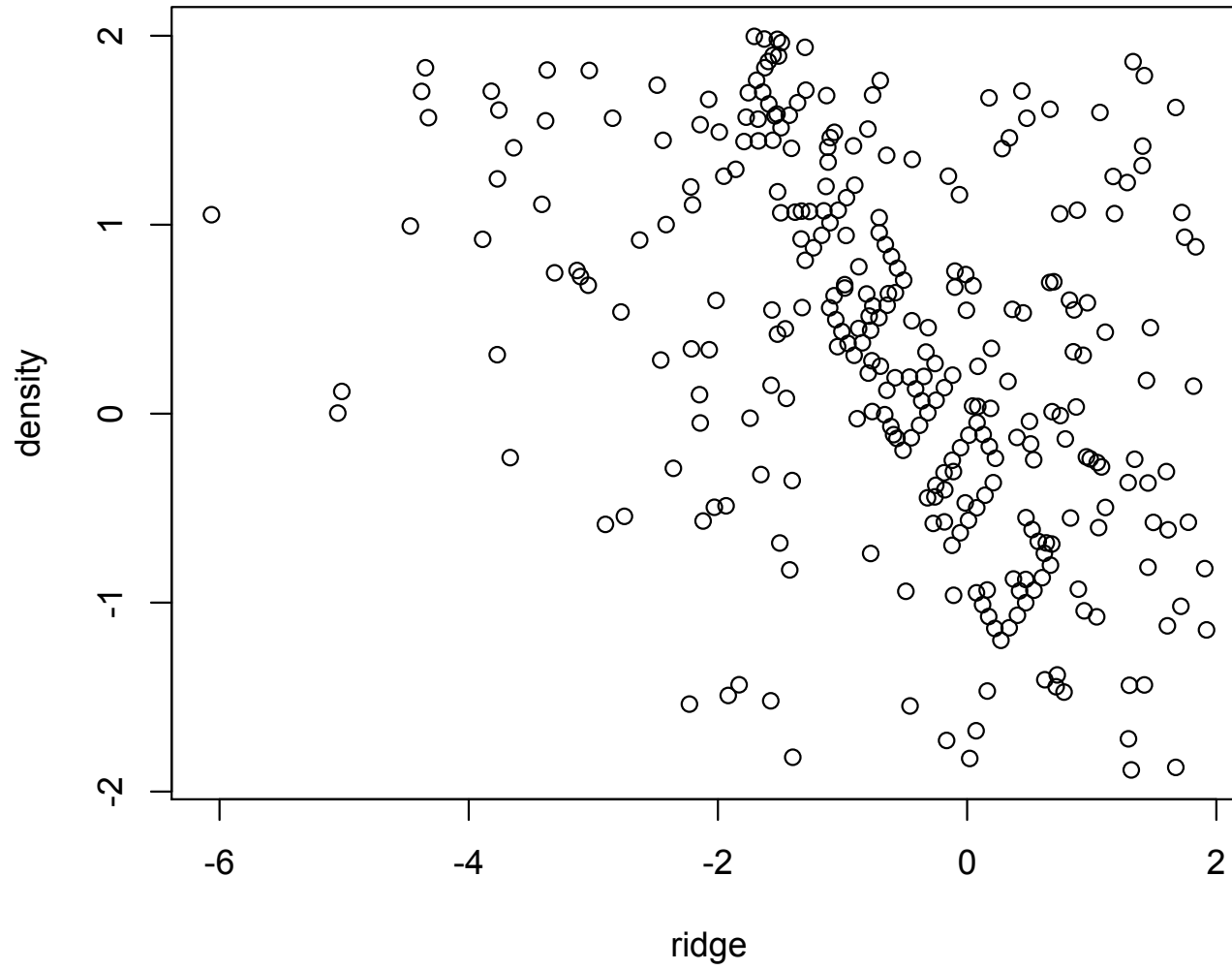
High dimensions



High dimensions



Eureka



SVG+tooltips

SVG (Scalable Vector Graphics) is a non-bitmap graphics format for the web.

The `RSvgDevice` (all platforms) and the `svg()` device (Linux, Mac) allow SVG output.

`RSVGTipsDevice` and `SVGAnnotation` (Linux, Mac) help create interactive plots.

We can use this to create graphs with links and tooltips. For example, a funnelplot showing associations between a large number of SNPs and VTE.

Point at a dot to see the SNP it represents, and click to go to information about the gene.

SVG+tooltips

```
for(i in 1:length(or)) {
  setSVGShapeToolTip(title=gene[i],
    desc1=snp[i],
    desc2=if(abs(lor[i]/se[i])>qnorm(0.5/n,lower.tail=FALSE))
              qvals[i] else NULL
  )

  setSVGShapeURL(paste("http://pga.gs.washington.edu/data",
                        tolower(gene[i]),
                        sep="/")
  )
  points(prec[i],lor[i], cex=1, pch=19, col='grey')
}
```


Google Earth

Google Earth is controlled by KML files specifying locations. KML is another plain text format.

We can write a KML file

```
<?xml version="1.0" encoding="UTF-8"?>
  <kml xmlns="http://earth.google.com/kml/2.1">
    <Placemark>
      <name> 1 </name>
      <Point> <coordinates>-118.0256,34.11619,400</coordinates>
    </Point>
  </Placemark>
</kml>
```

and then send it to Google Earth with the `shell.exec(filename)` function, which opens a file using whatever is the appropriate program.

Google Earth

The `identify()` function lets the user select a point on a scatterplot.

In this example the points are locations where air pollution was measured, and we can call Google Earth to look at the location.

```
kml<-function(conn,lat,lon,name){
  ss<-gsub(" ","",paste("<coordinates>",lat,",",lon,",400</coordinates>"))
  cat("<?xml version=\"1.0\" encoding=\"UTF-8\"?>
  <kml xmlns=\"http://earth.google.com/kml/2.1\">
  <Placemark>
      <name>",name,"</name>
  <Point>",
  ss,"
  </Point>
  </Placemark>
</kml>\n", file=conn)
}
```

Google Earth

```
ogle<-function(lat,lon,name){
  ff<-paste(tempfile(),"kml",sep=".")
  cc<-file(ff,"w")
  kml(cc,lat,lon,name)
  close(cc)
  system(paste("open",ff))
}

mesa<-read.table("noxmonitors.dat",header=TRUE)
mesa<-na.omit(mesa)

with(mesa,plot(Dist_nearestmajor,air_nox_corr))
repeat({
  a<-with(mesa,
    identify(Dist_nearestmajor,air_nox_corr,n=1,labels=rownames(mesa)))
  with(mesa, ogle(LONG[a],LAT[a],rownames(mesa)[a]))
})
```

A few useful books

General

Edward Tufte. *The Visual Display of Quantitative Information*

William S. Cleveland. *Elements of Graphing Data*

(and his other books)

Colin Ware *Information Visualization*

Unwin, Theus, Hoffman *Visualizing a million*

Leland Wilkinson *Grammar of Graphics*

R implementation

Paul Murrell. *R Graphics*.

Deepayan Sarkar: *Lattice* (Springer useR series)

Hadley Wickham. *ggplot* (Springer useR series)

Final notes

In addition to the tools in the standard R distribution, there are many useful things in packages.

CRAN Task views help you find them



CRAN

[Mirrors](#)

[What's new?](#)

[Task Views](#)

[Search](#)

About R

[R Homepage](#)

Software

[R Sources](#)

[R Binaries](#)

[Packages](#)

[Other](#)

Documentation

[Manuals](#)

[FAQs](#)

[Contributed](#)

[Bayesian](#)

[ChemPhys](#)

[Cluster](#)

[Distributions](#)

[Econometrics](#)

[Environmetrics](#)

[ExperimentalDesign](#)

[Finance](#)

[Genetics](#)

[Graphics](#)

[gR](#)

[MachineLearning](#)

[Multivariate](#)

[NaturalLanguageProcessing](#)

[Optimization](#)

[Pharmacokinetics](#)

[Psychometrics](#)

[Robust](#)

CRAN Task Views

Bayesian Inference

Chemometrics and Computational Physics

Cluster Analysis & Finite Mixture Models

Probability Distributions

Computational Econometrics

Analysis of Ecological and Environmental Data

Design of Experiments (DoE) & Analysis of Experimental Data

Empirical Finance

Statistical Genetics

Graphic Displays & Dynamic Graphics & Graphic Devices & Visualization

gRaphical Models in R

Machine Learning & Statistical Learning

Multivariate Statistics

Natural Language Processing

Optimization and Mathematical Programming

Analysis of Pharmacokinetic Data

Psychometric Models and Methods

Robust Statistical Methods