Pretty Pictures

Thomas Lumley

Biostatistics

2006-10-5

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
- 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...

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

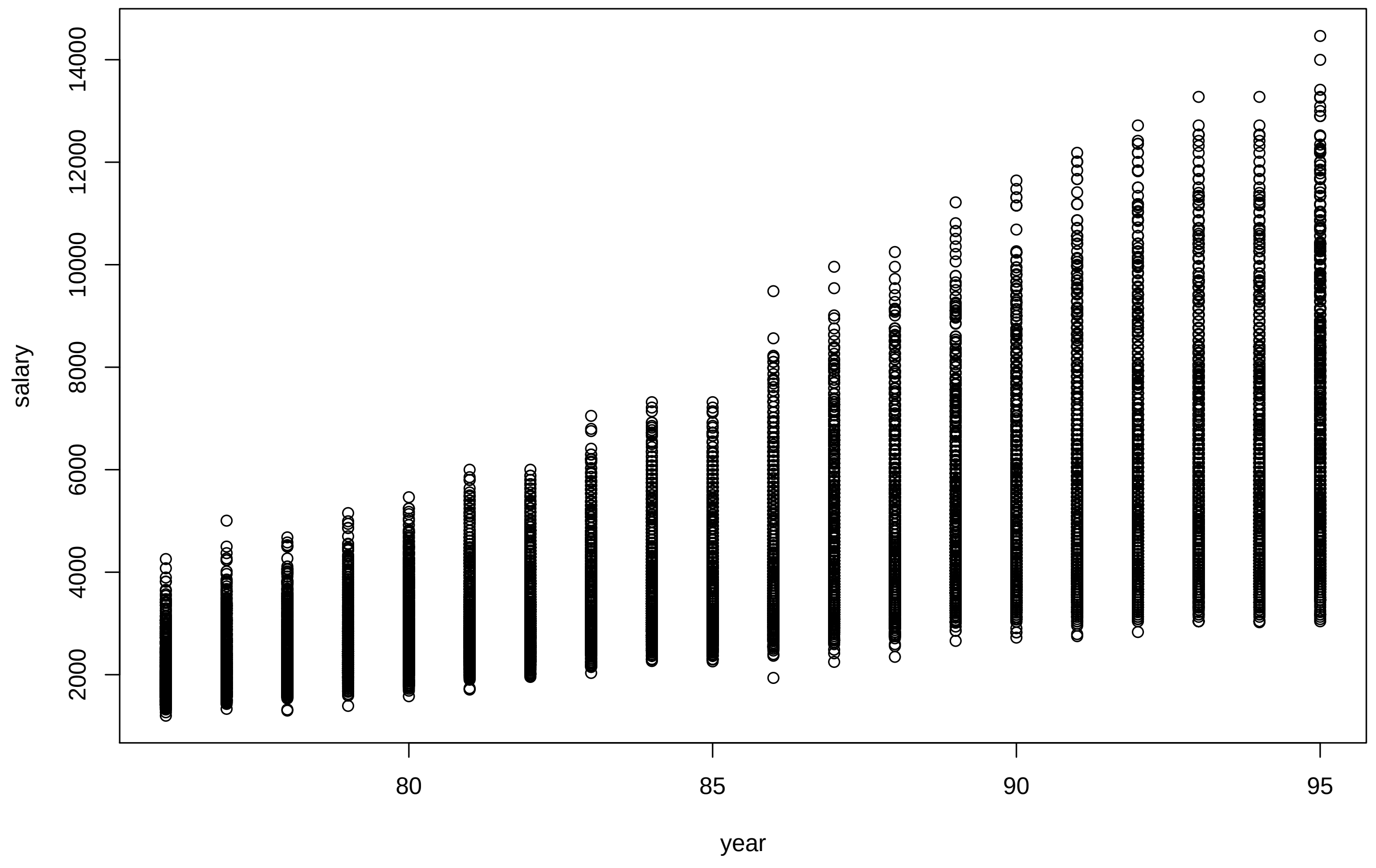
Important points

- 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.

Features of plot

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



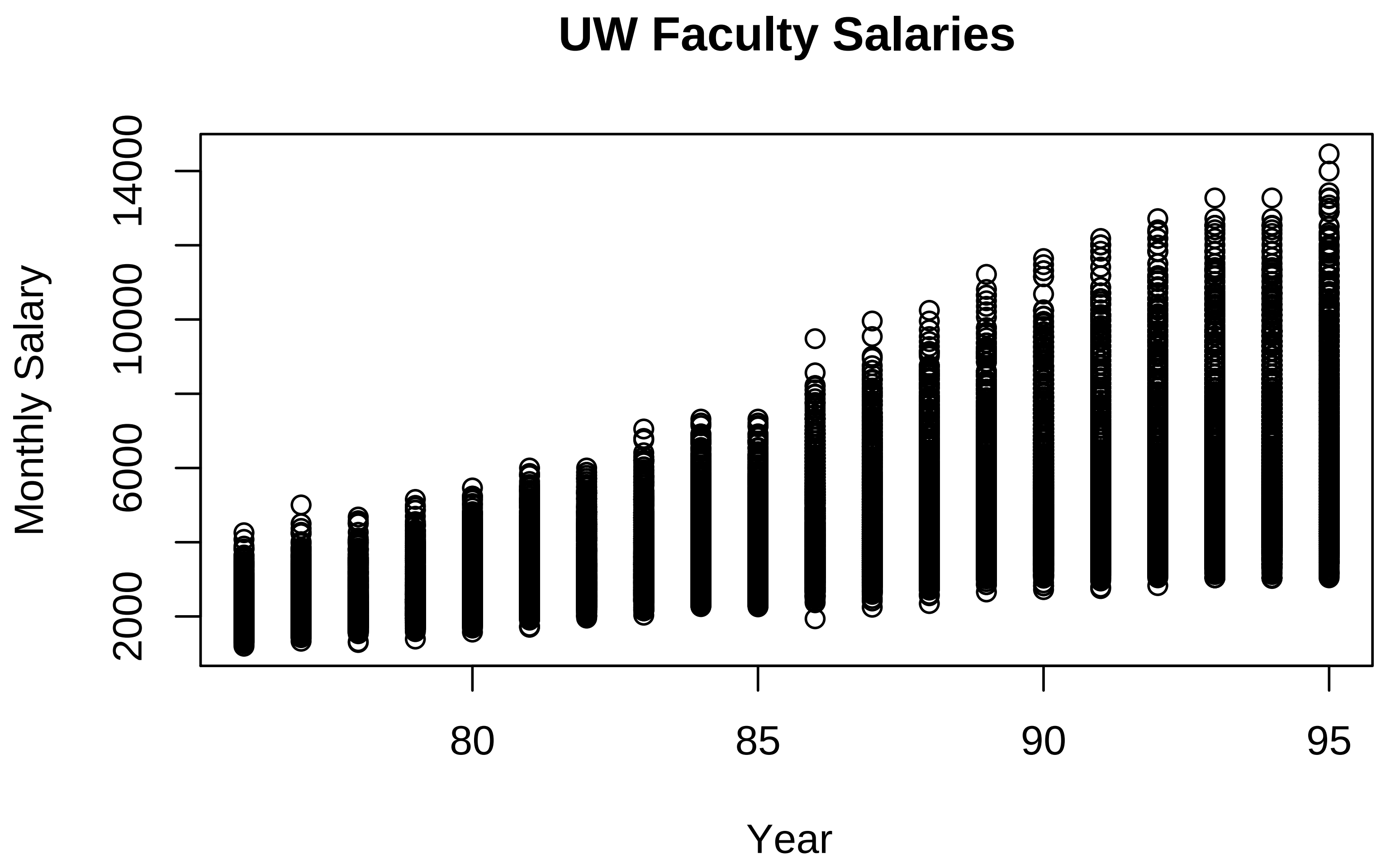
Notes

- There are automatic axis labels, taken from the variable names.
- The axis ranges are chosen to give nice round numbers that include all the data
- The labels are too small: this plot was drawn at full-page size and reduced.

Features of plot

```
plot(salary~year, data=salary, xlab="Year",ylab="Monthly salary",
    main="UW Faculty Salaries")
```

Features of plot



Notes

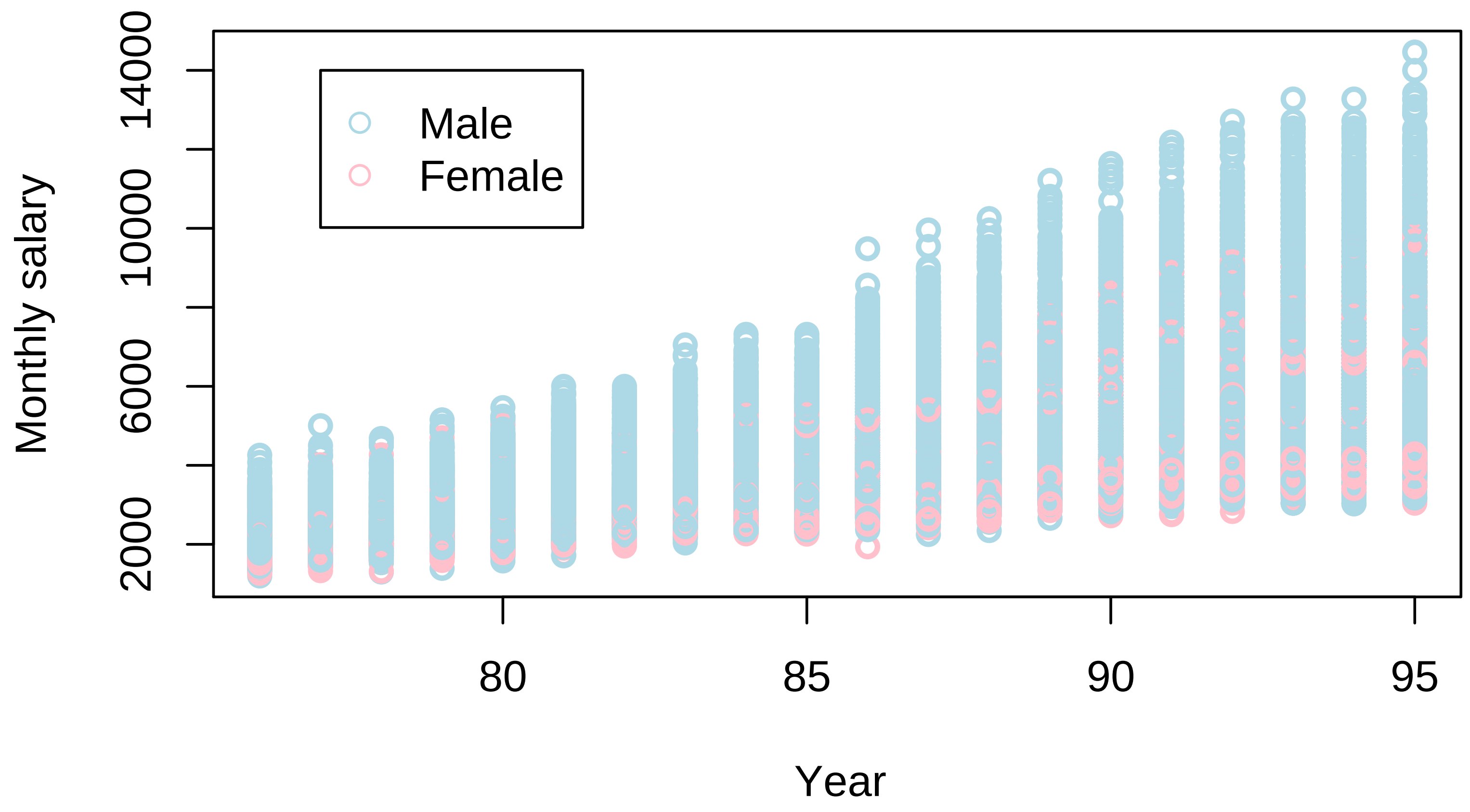
- This plot was drawn at 4×6 inches, a reasonable size for including in a report, and the labels are legible, though it wouldn't hurt to have them bigger.

- The axis labels were specified explicitly, as was an overall title.

Features of plot

```
plot(salary~year, data=salary, xlab="Year",ylab="Monthly salary",
    col=ifelse(gender=="M", "lightblue","pink"),lwd=2)
legend(77, 14000, col=c("lightblue","pink"), pch=1,
    legend=c("Male","Female"))
```

Features of plot



Notes

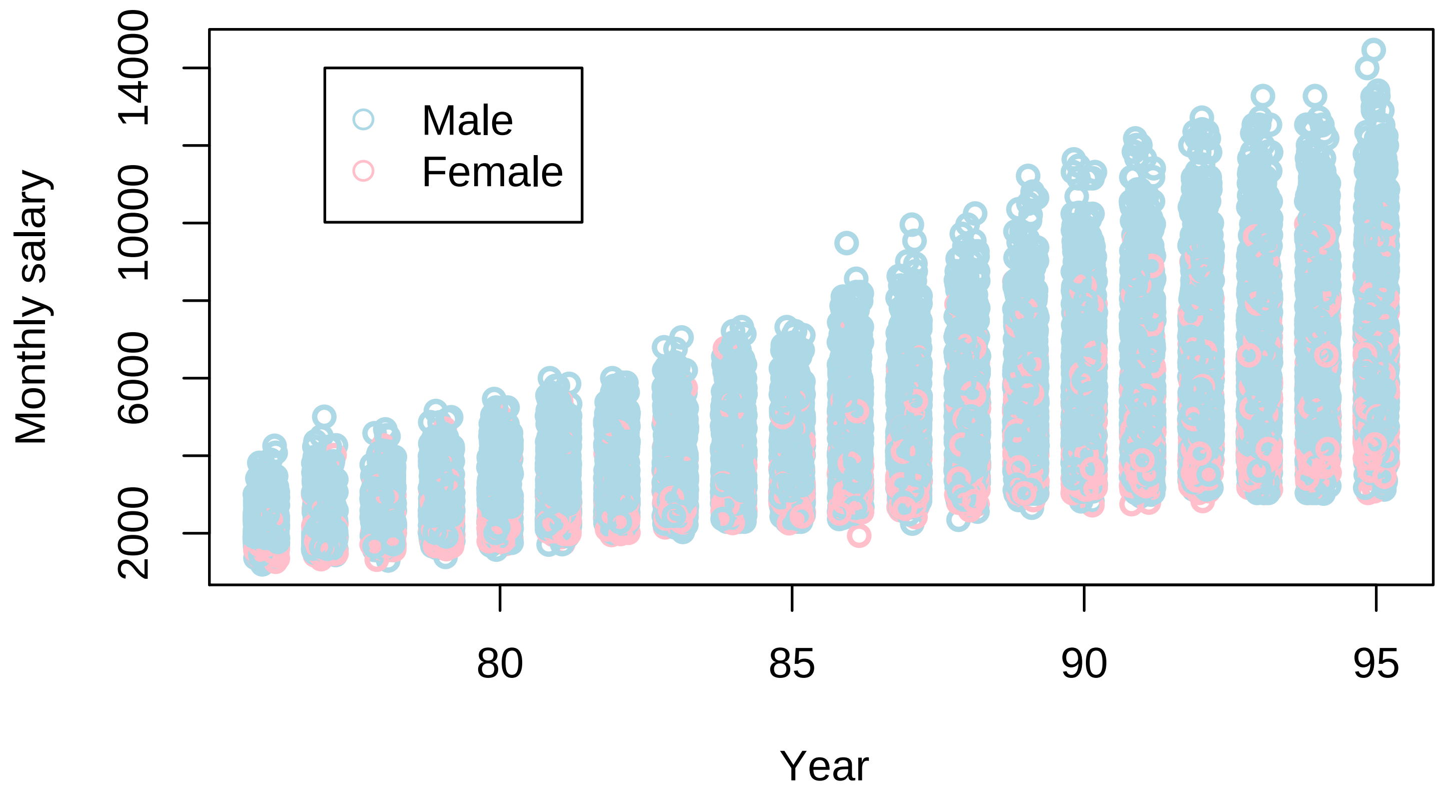
- col= specifies a plotting color that can be a single value or a vector. Colors can be specified by name, by number in the standard palette, or by RGB values (eg `"#FF0000"` for dark red)
- ifelse() selects from two vectors according to a logical condition
- lwd is line width
- legend() makes a legend. The first two arguments are the location

There is too much overplotting to see clearly what the patterns are.

Features of plot

```
plot(salary~jitter(year), data=salary,
    xlab="Year", ylab="Monthly salary",
    col=ifelse(gender=="M", "lightblue","pink"),lwd=2)
legend(77, 14000, col=c("lightblue","pink"), pch=1,
    legend=c("Male","Female"))
```

Features of plot

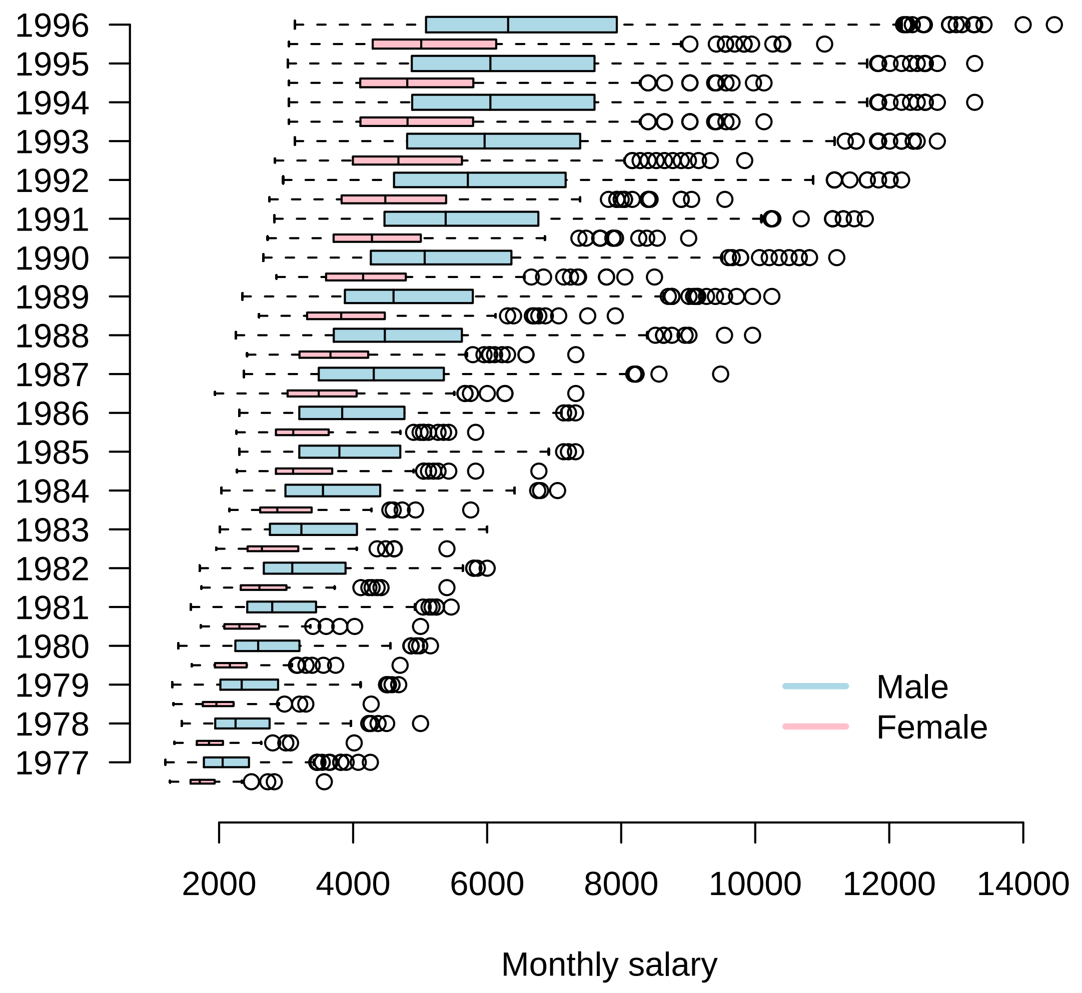


Notes

jitter() wiggles the values slightly, which helps with overplotting. The pattern still isn't very clear.

More complicated

```
plot(salary~interaction(gender,year), data=salary,
     col=c("pink","lightblue"), varwidth=TRUE,
     horizontal=TRUE,axes=F,xlab="",ylab="Monthly salary")
axis(1)
axis(2,at=(1:20)*2,label=1976+(1:20))
legend(10000,8,legend=c("Male","Female"),
    col=c("lightblue","pink"), lty=1,lwd=3,bty="n")
```


1996
1995
1994
1993
1992
1991
1990
1989
1988
1987
1986
1985
1984
1983
1982
1981
1980
1979
1978
1977
2000
4000
6000
8000
10000
12000
14000
Monthly salary
Male
Female

Notes

- interaction creates a factor with all combinations of values of the two variables
- When the predictor is a factor, plot does a boxplot
- A boxplot has a single color for each box
- varwidth=TRUE makes the box width proportional to sample size
- horizontal=TRUE says to rotate to have horizontal boxes
- axes=FALSE says not to draw axes

Notes

- axis() draws axes. Axis 1 is the lower x-axis, axis 2 is the left y-axis. There are default values, or specify the locations and the labels. We specified a label for every second y-value (every year).

No, you wouldn't necessarily be expected to do this yourself.

Conditioning

Another approach is to use separate plots for men and women. A naive approach would be

```
men <- salary[gender=="M",]
women <- salary[gender=="F",]
plot(salary~year, data=men)
plot(salary~year, data=women)
```

The problem is that the y-axis scales would be different, making it hard to compare.

Axis limits

You could specify the axis limits

```
men <- subset(salary, gender=="M")
women <- subset(salary, gender=="F")
plot(salary~year, data=men, ylim=c(0,14000),
   main="Salaries for men")
plot(salary~year, data=women, ylim=c(0,14000),
   main="Salaries for women")
```

which gives reasonable plots. subset() is another way to subset data frames.

Coplots

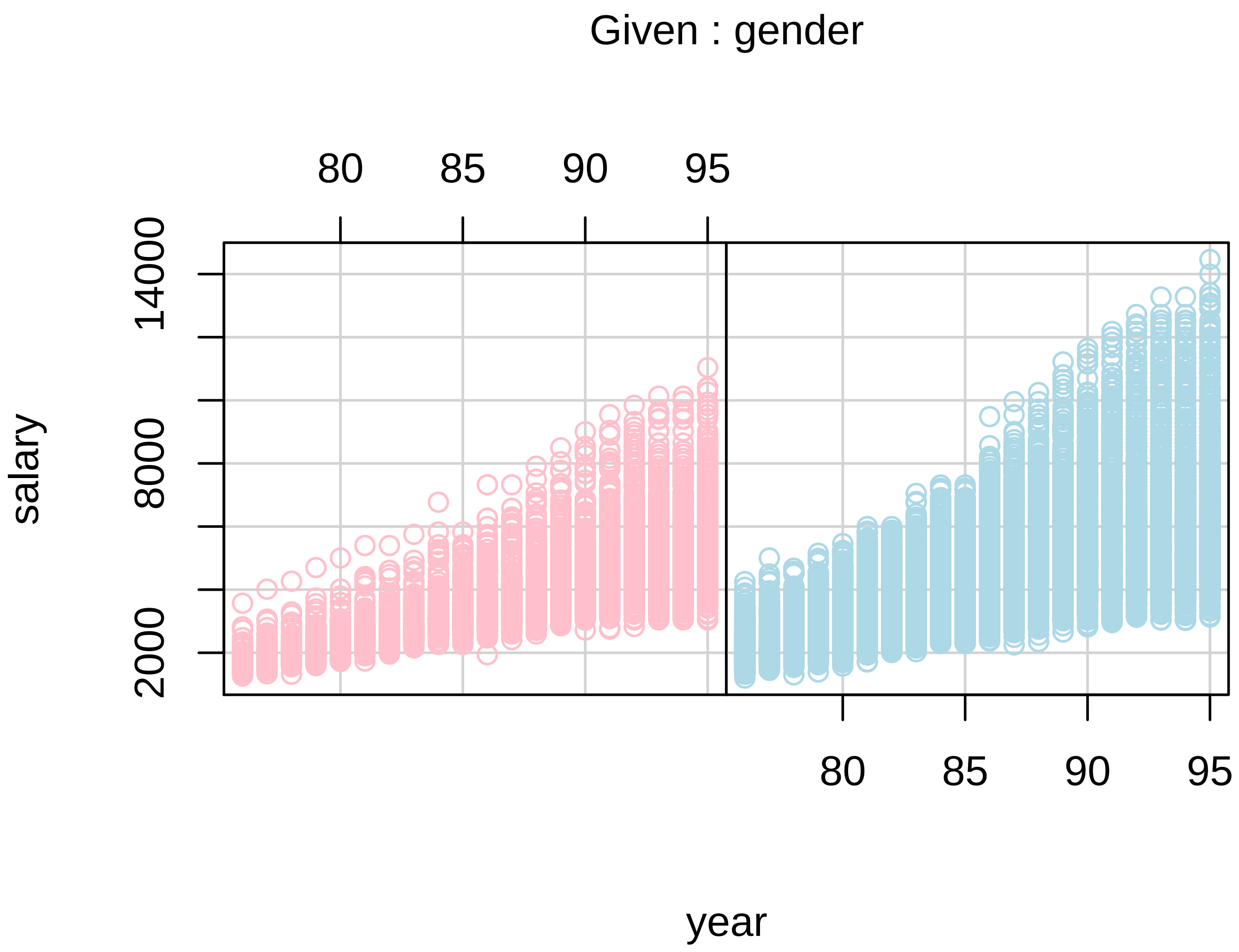
There is a more automated approach using coplot()

```
coplot(salary~year|gender,data=salary,
   col=ifelse(salary$gender=="M","lightblue","pink"),
   show.given=FALSE)
```

The `|gender` in the formula asks for a plot conditioned on gender.

More general versions of this are the Trellis graphics in the lattice package.

Coplots

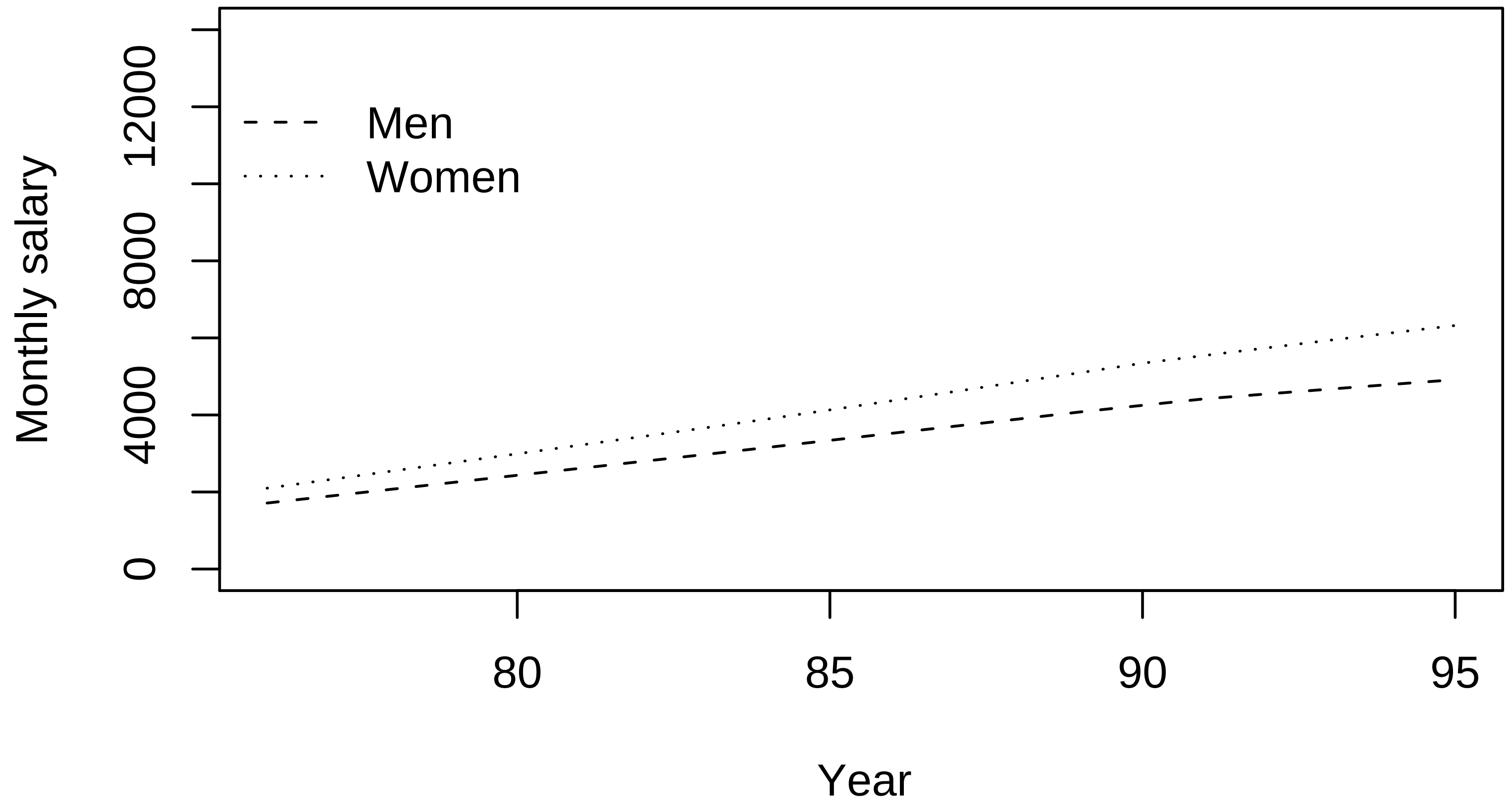


Scatterplot smoothers

Another approach is to stop trying to display all the data. A scatterplot smoother estimates the mean of y for each value of x using something like a mean of nearby points. Smoothers are useful as an addition or alternative to scatterplots.

```
men <- subset(salary, gender=="M")
women <- subset(salary, gender=="F")
with(men, plot(lowess(year,salary), xlab="Year",
   ylab="Monthly salary",ylim=c(0,14000),type="l",lty=3))
with(women, lines(lowess(year,salary),lty=2))
legend(75,13000,legend=c("Men","Women"),lty=c(2,3),bty="n")
```

Scatterplot smoothers



Notes

- lowess is a popular scatterplot smoother.
- type="l" means lines instead of points, lty specifies line type.
- lines() adds lines to an existing plot, so it doesn't need axis labels, ranges. For adding points use points()
- bty="n" means no box around the legend.

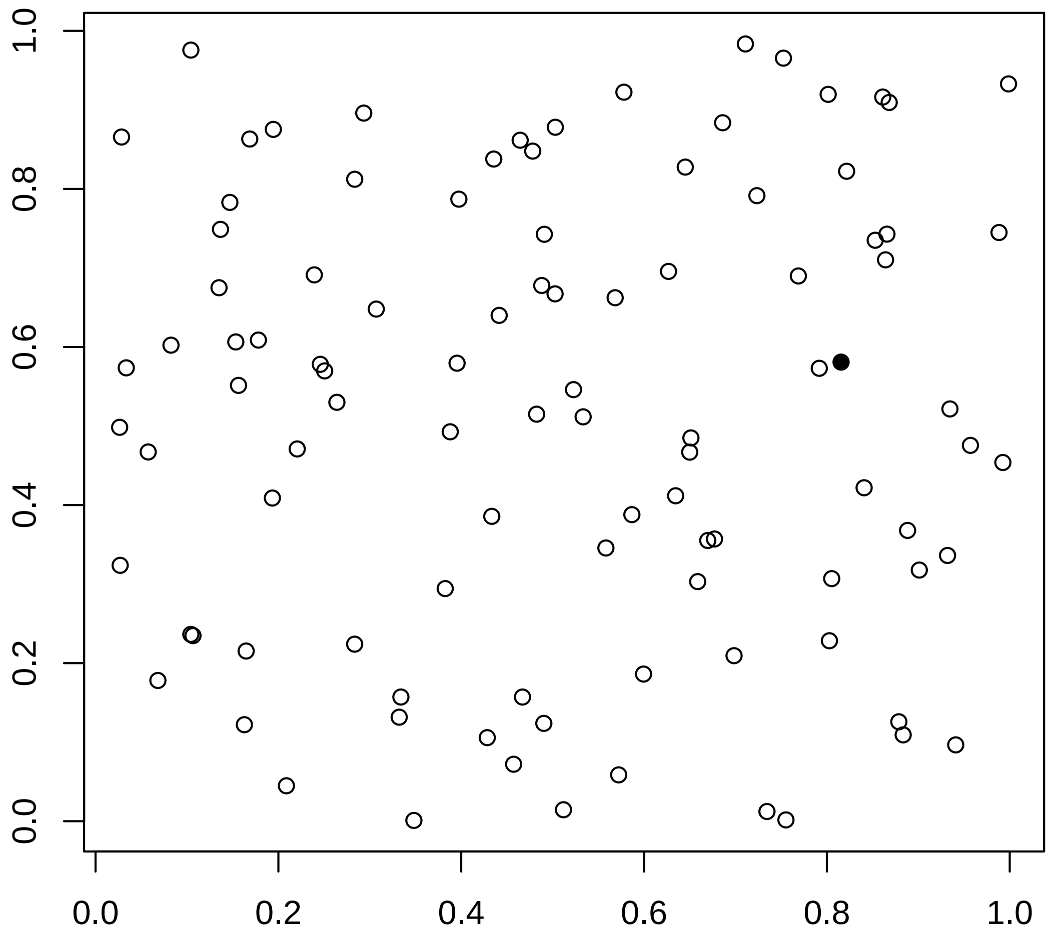
Theory

Design of graphics is supported by both statistical theory (what information is informative about the question of interest) and perceptual theory (how do we code the information so that it is easy to see).

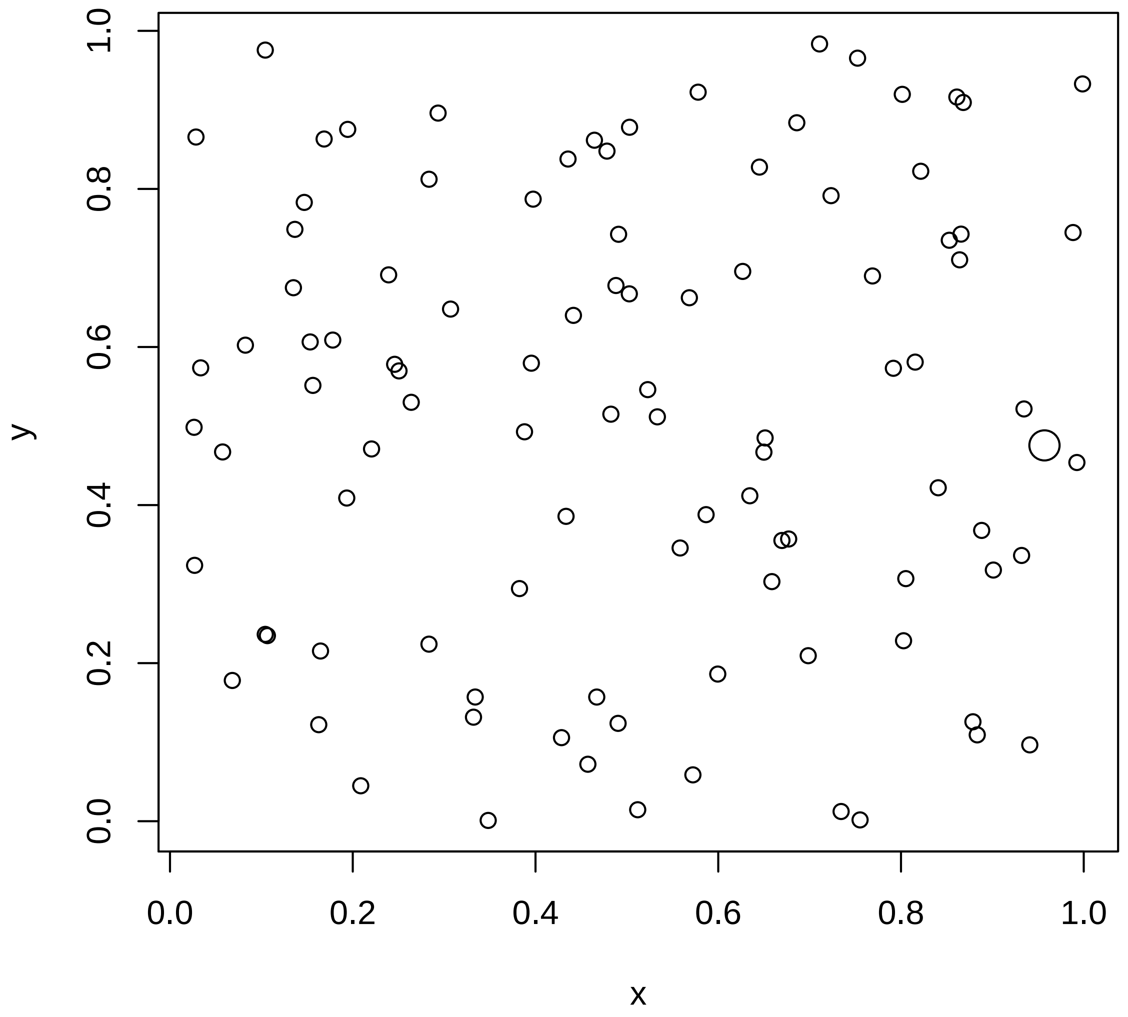
Three excellent books:

- WS Cleveland The Elements of Graphing Data
- Edward Tufte Visual Display of Quantitative Information
- Colin Ware Information Visualization

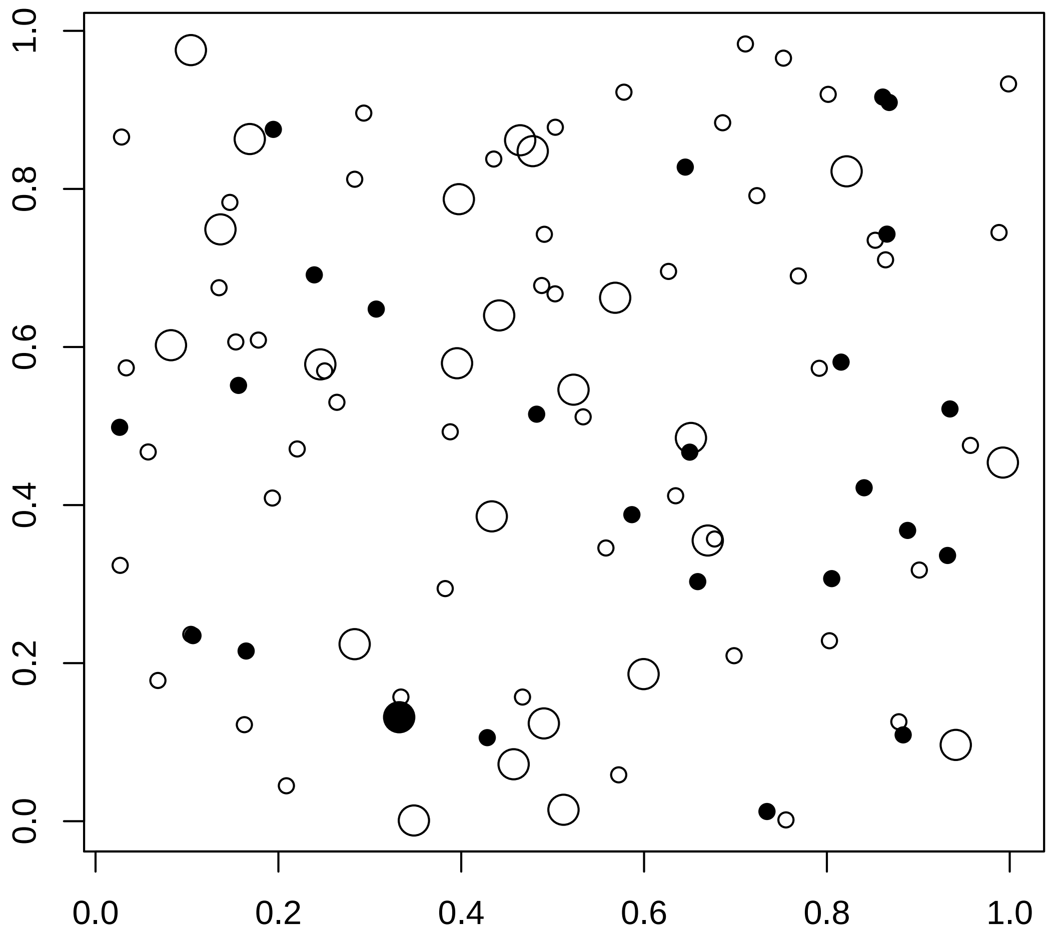
Which point is different?



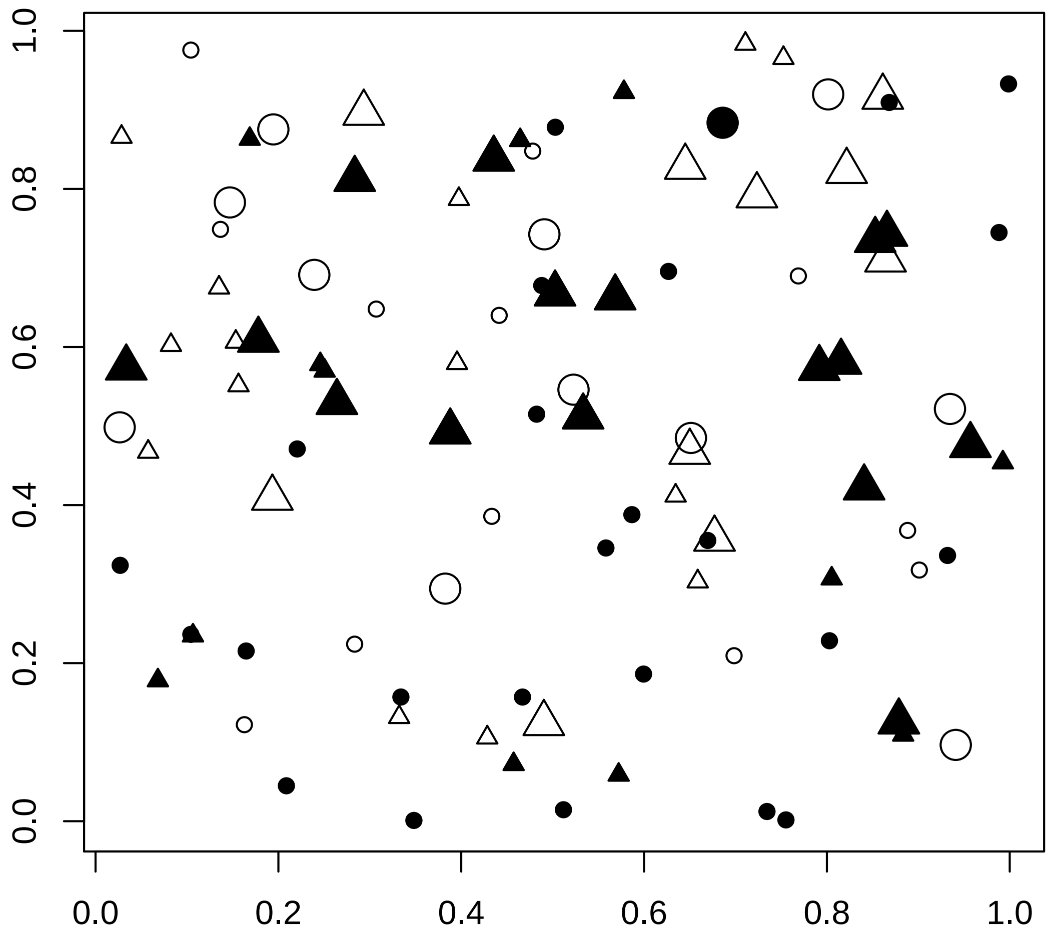
Which point is different?



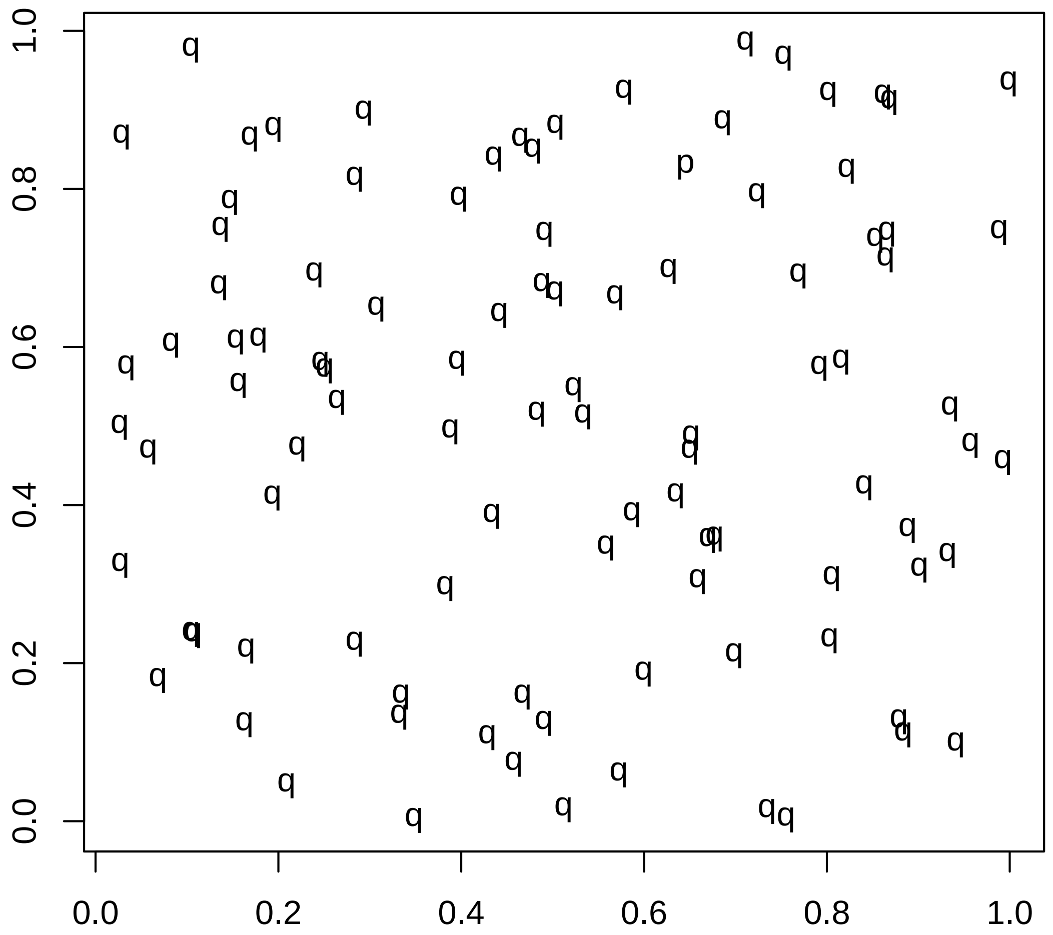
Which point is different?



Which point is different?



Which point is different?

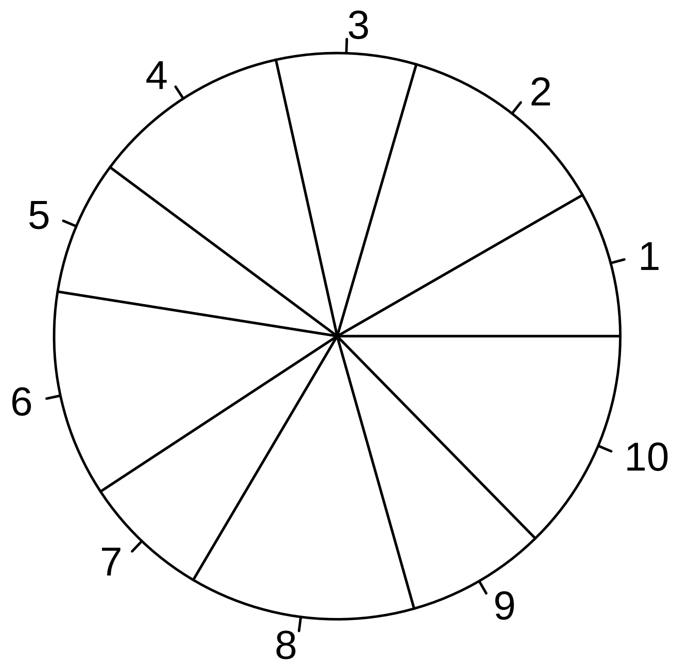


Preattentive perception

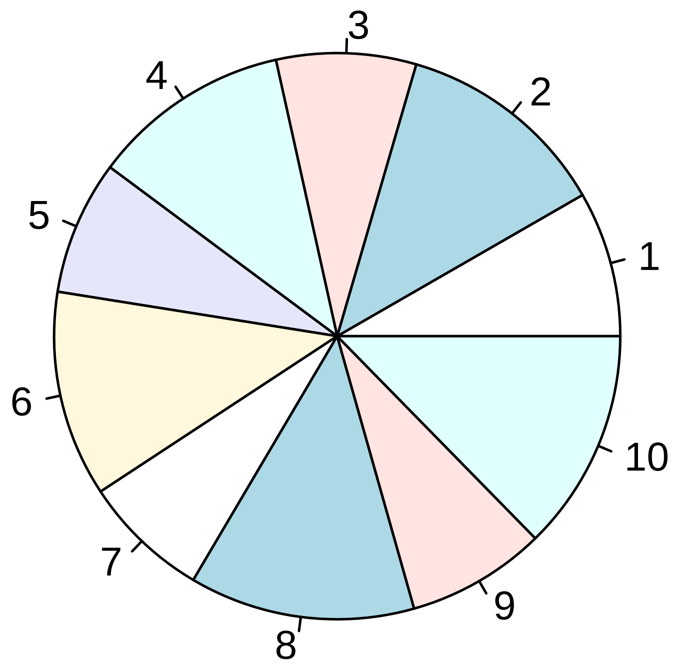
Some differences are processed by the brain before conscious perception and others rely on conscious effort.

Preattentively distinct plotting symbols allow individual points to be identified and allow a subset of points to be seen as a single entity.

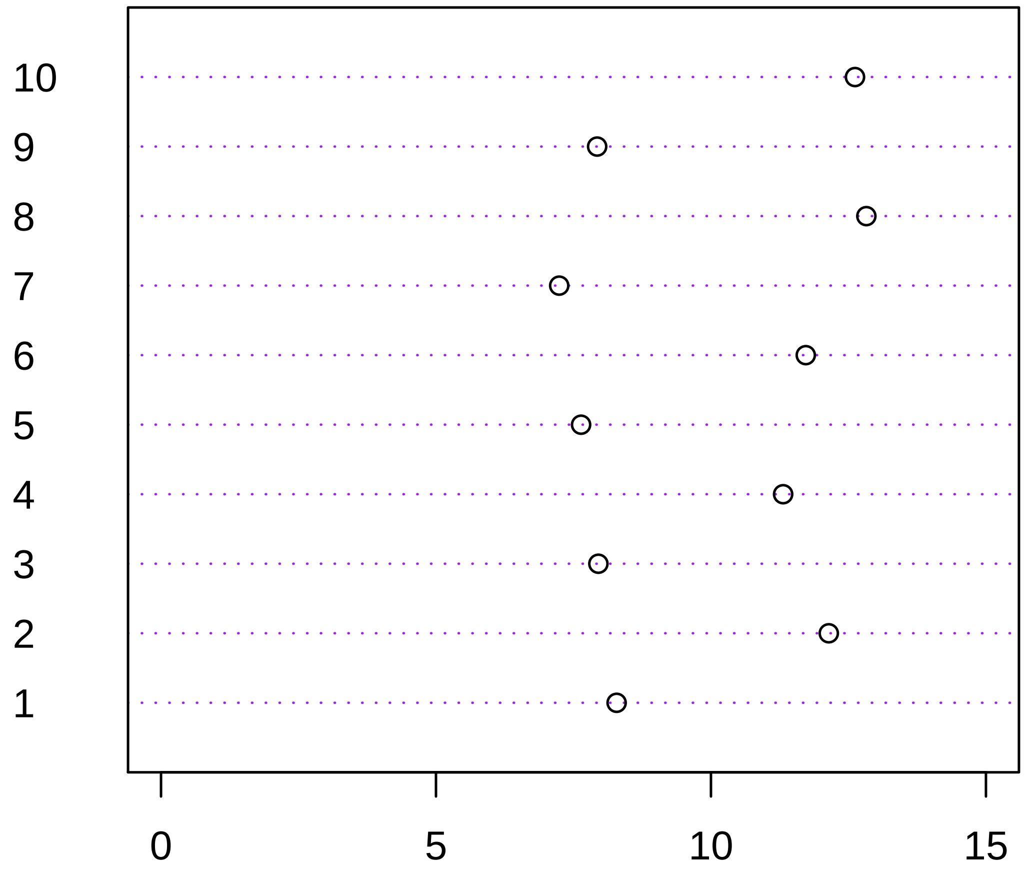
Example: angle vs position



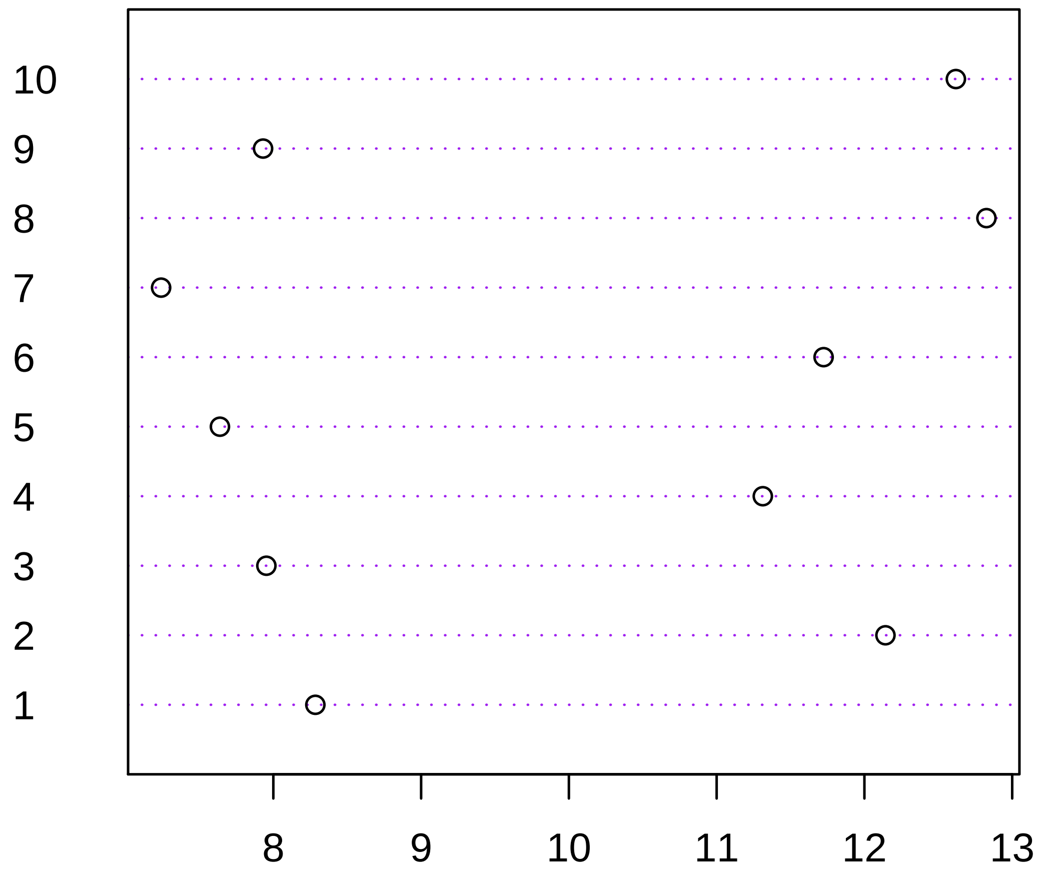
Example: angle vs position



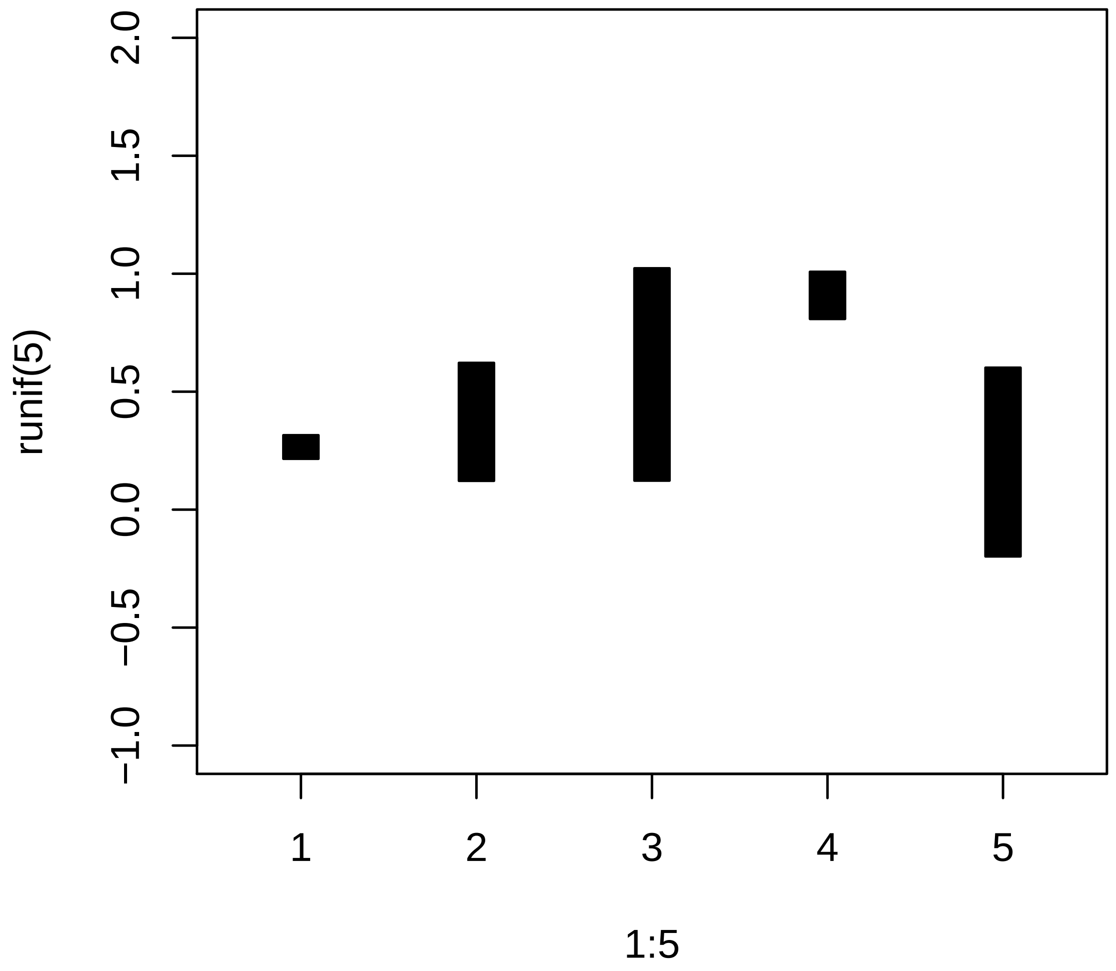
Example: angle vs position



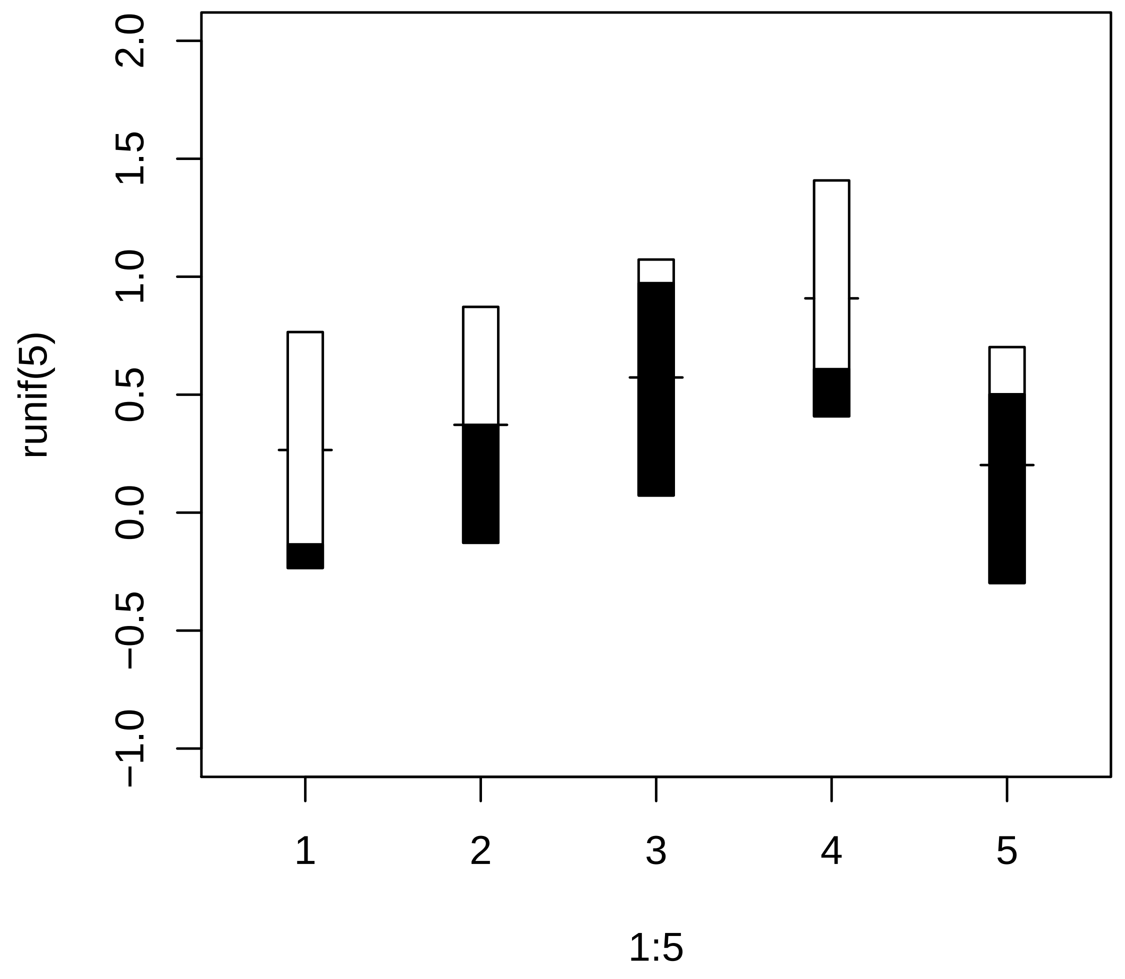
Example: angle vs position



Example: Weber's law



Example: Weber's law



Estimation

The best way to compare quantities reliably is to have them lined up on an identical scale.

Identical, non-aligned scales are next best.

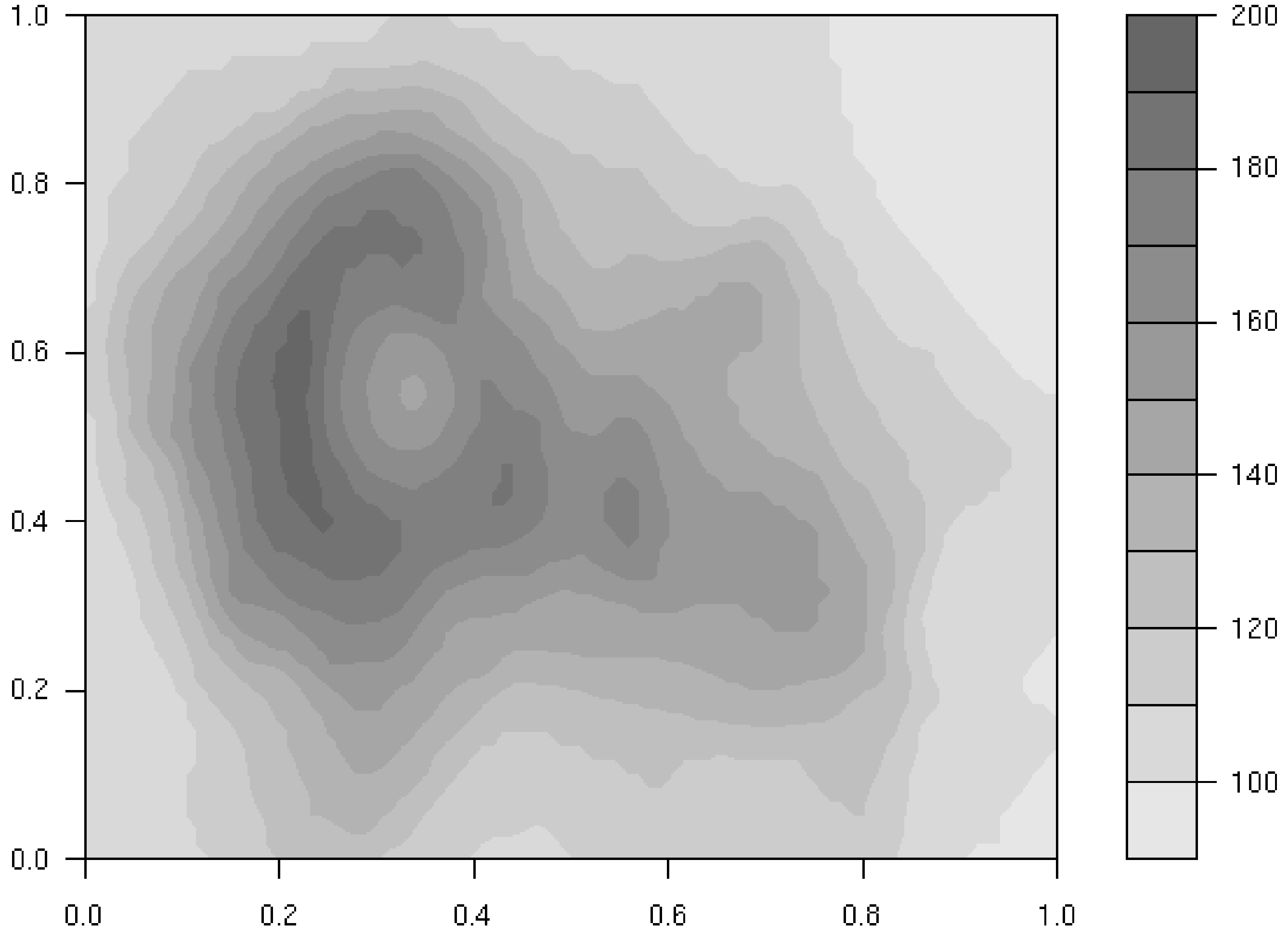
Seeing patterns in angles or areas is hard.

Color and shade

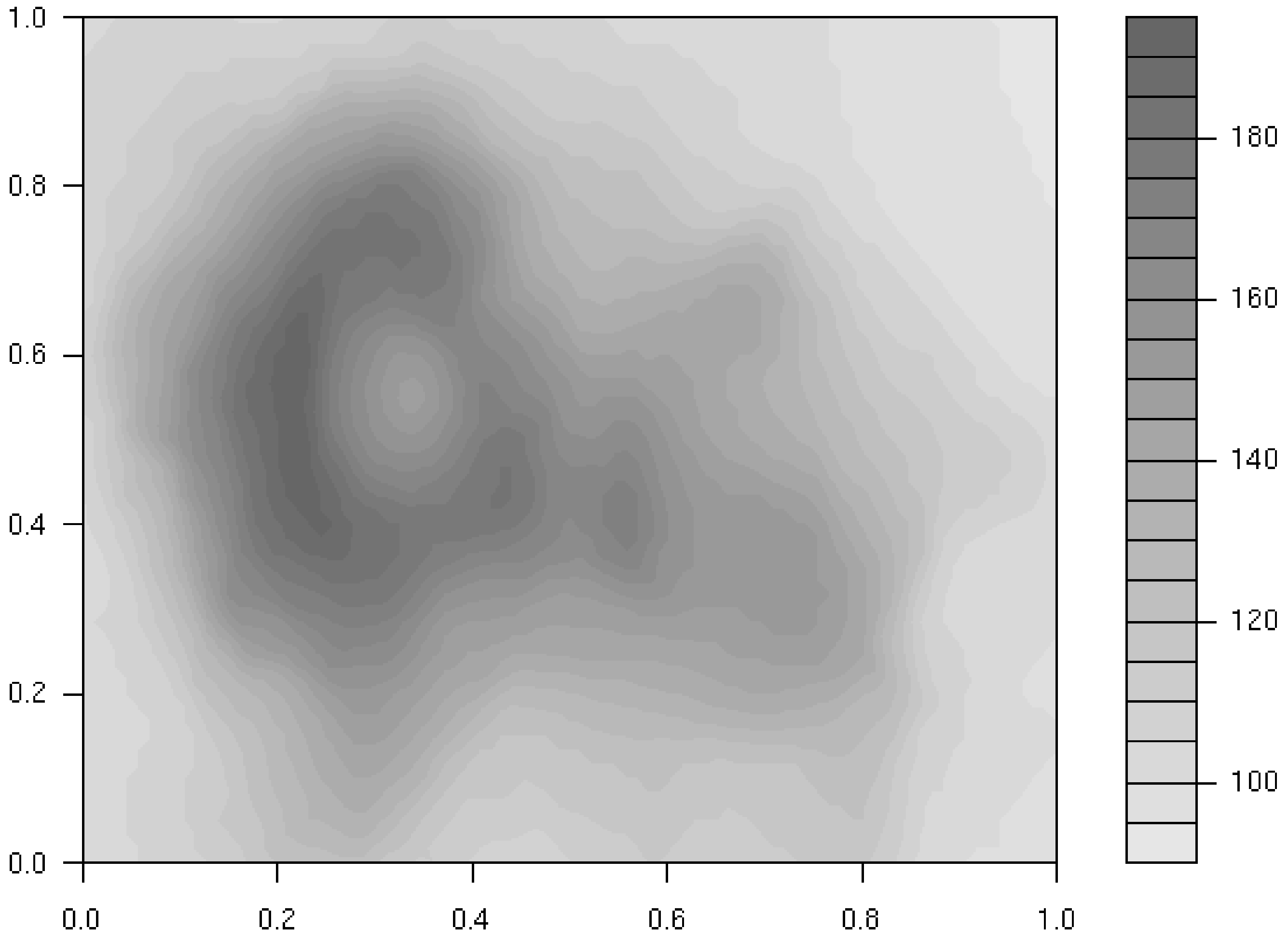
Using gray scales for quantative values is unreliable because the perception of lightness is sensitive to the lightness of nearby points. Using a small number of gray levels causes Mach banding, which clearly makes lightness estimation less reliable.

A smooth sequence of levels removes the Mach banding but creates a blurry image

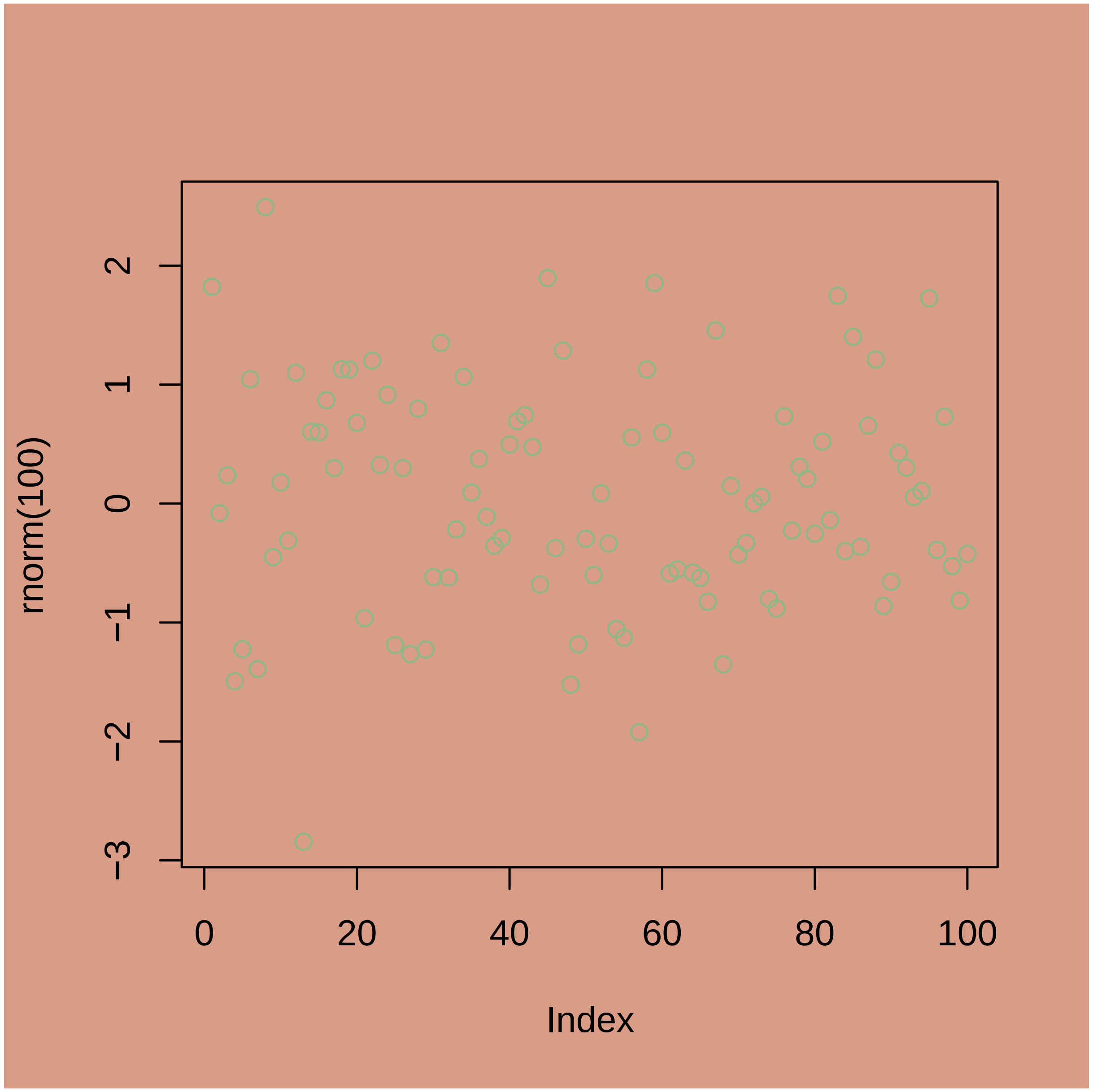
Color and shade



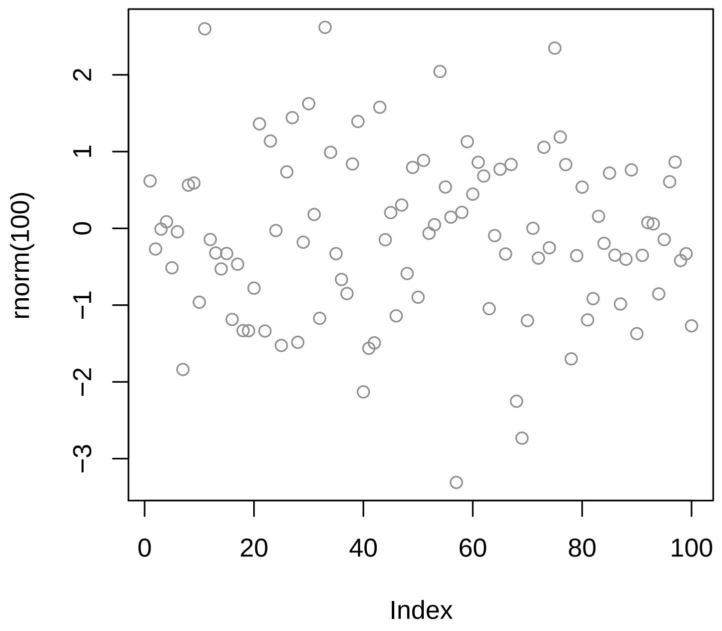
Color and shade

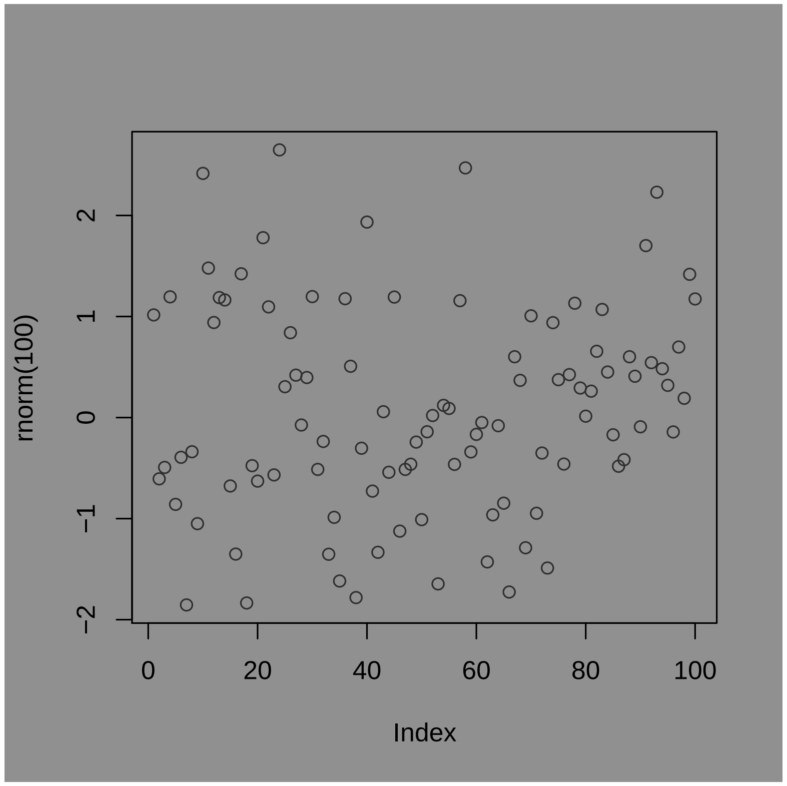


Isoluminant colors



Luminance differences





All three plots have the same amount of color difference between points and background.

Note that intense blues all have very low luminance and so do not contrast well with black, and that yellows have high luminance and do not contrast well with white.

Daltonism

Daltonism, or red:green anomalous vision, results when the long or medium wavelength cones are either absent or are producing the wrong color-sensitive pigment. The relevant genes are on the X chromosome so it is much more common in men.

About 6% of men have anomalous red-green vision and about 2.5% have no red:green vision.

The possibility for mutations producing the wrong pigment occurs because the long and medium wavelength cone pigments are coded by adjacent genes with a single upstream control region and have very similar sequences.

There are two variants, proteranopia and deuteranopia, depending on which pigment is missing. Both produce red:green color blindness, but they produce different distortions in luminance.

Safe color schemes

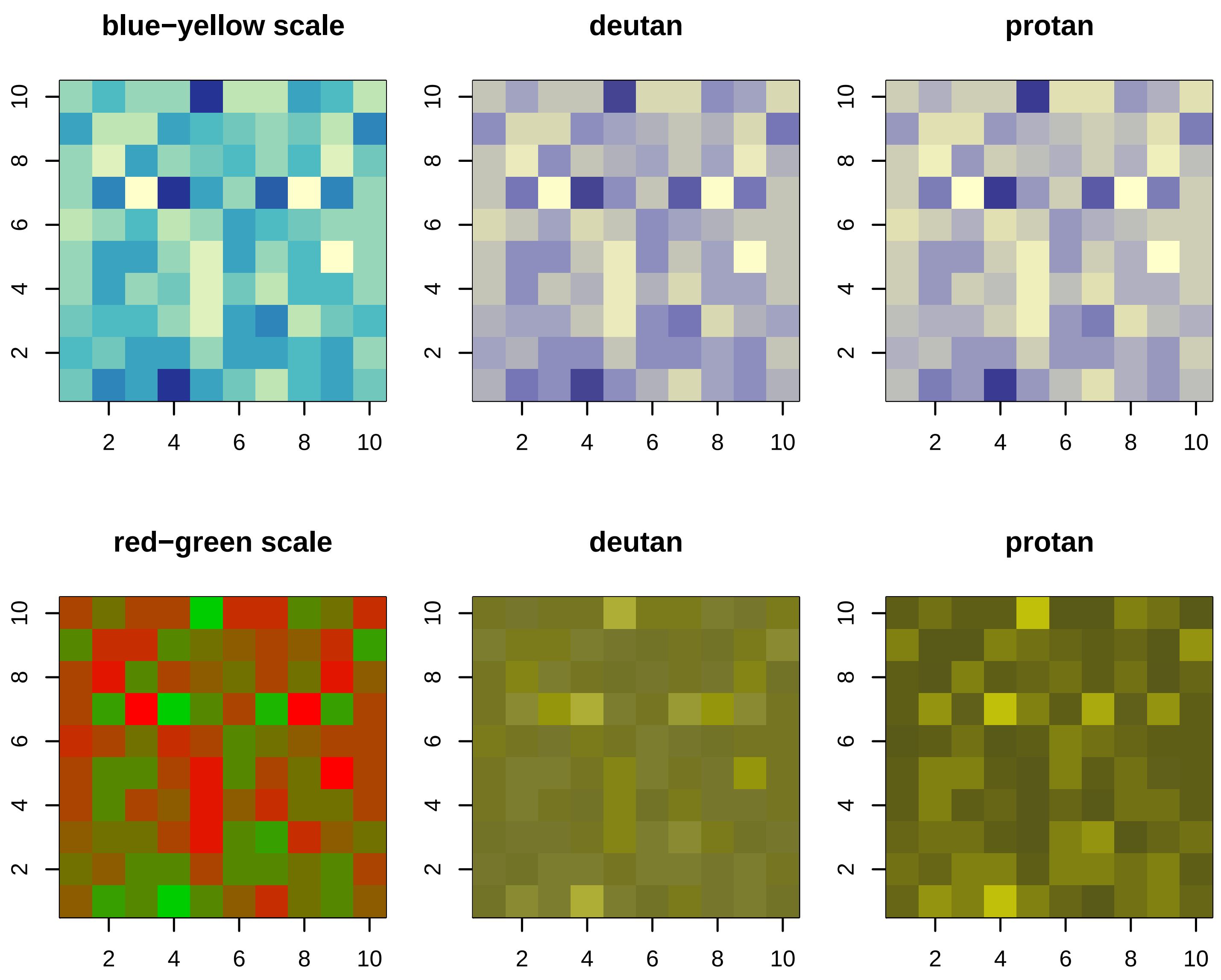
One strategy is to ensure that all your colors have, eg, 0 for the red component in an RGB specification. The disadvantage is that this restricts luminance variation.

`http://www.colorbrewer.com` gives many good colour palettes and says which ones are useful for red:green colorblind viewers.

Experiments on color perception (F. Vinot, H. Brettel and J. D. Mollon (1999) Digital video colourmaps for checking the legibility of displays by dichromats. Color Research and Application 24, 243-252.) give a mapping from RGB space to what is seen with dichromatic vision.

`http://www.vischeck.com` and the dichromat package in R implement this mapping. Use it to check visibility in graphs.

Example


blue−yellow scale
deutan
protan
red−green scale
deutan
protan
2
4
6
8
10