ILLINOIS

Visualization I

Lecture 17

Dirk Eddelbuettel

STAT 430: Data Science Programming Methods (Fall 2019)

Department of Statistics, University of Illinois

Outline

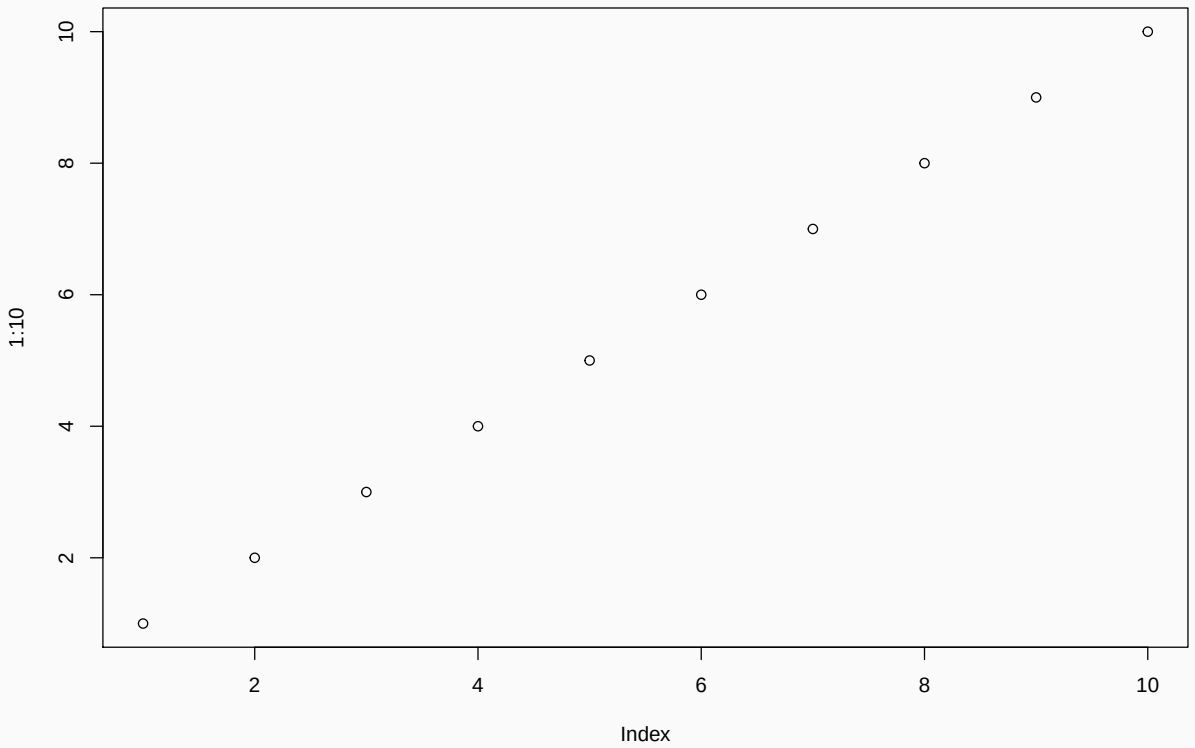
- base graphics
- lattice graphics
- base and lattice resources:
  - Lattice: Multivariate Data Visualization with R: an entire book by Deepayan Sarkar, author of the package (and R Core member)
  - Chapter 7 in R for Everyone: Adv. Analytics and Graphics, 2nd Ed
- ggplot2 resources (next lecture)
  - Kieran Healy 'SocViz' book and web site `socviz.co`
  - ggplot2 web site and numerous tutorials on the web
  - Chapter 22 in R for Data Science
  - Chapter 7 in R for Everyone: Adv. Analytics and Graphics, 2nd Ed

Visualization

Base plots

- R has excellent plotting facilities
- Base offers a wealth, built on top of packages
  - `grid` which we rarely ever use directly
  - `lattice` for Trellis-style panel plots
  - `ggplot2` (next lecture) now very popular too
- Focus today on base graphics
- Common for all those: graphs are *static*
  - no resizing or zooming
  - browser-based alternative plotting frameworks exists

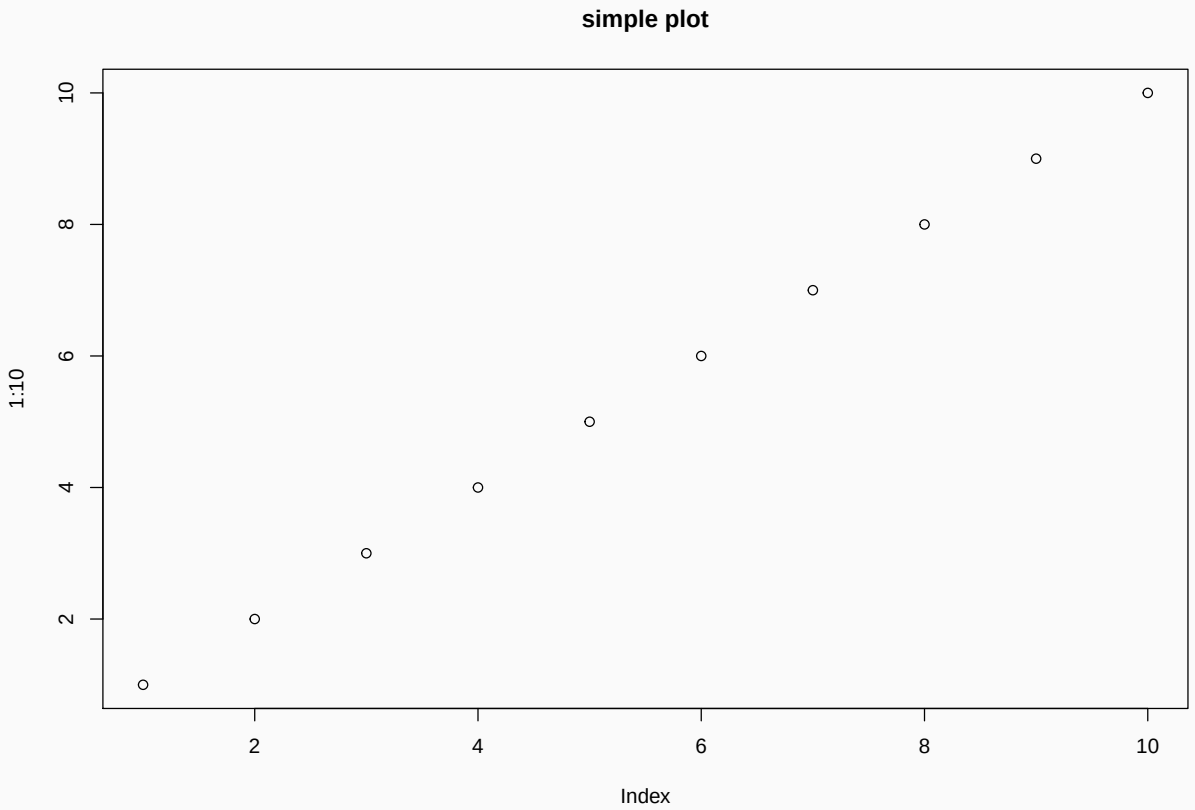
```r
plot(1:10)
```



Plots 10 points

Simple axis labels ‘guessed’ automatically

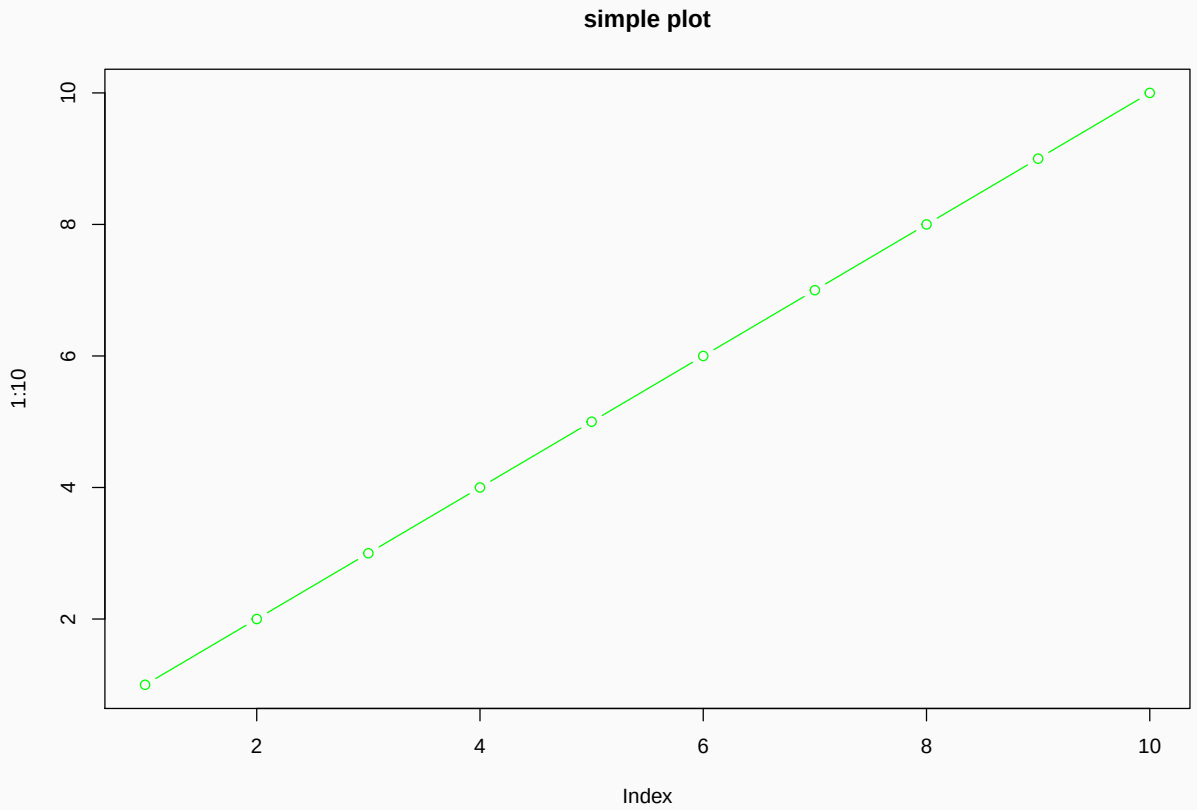
```r
plot(1:10, main="simple plot")
```



Plots 10 points

Sets a title

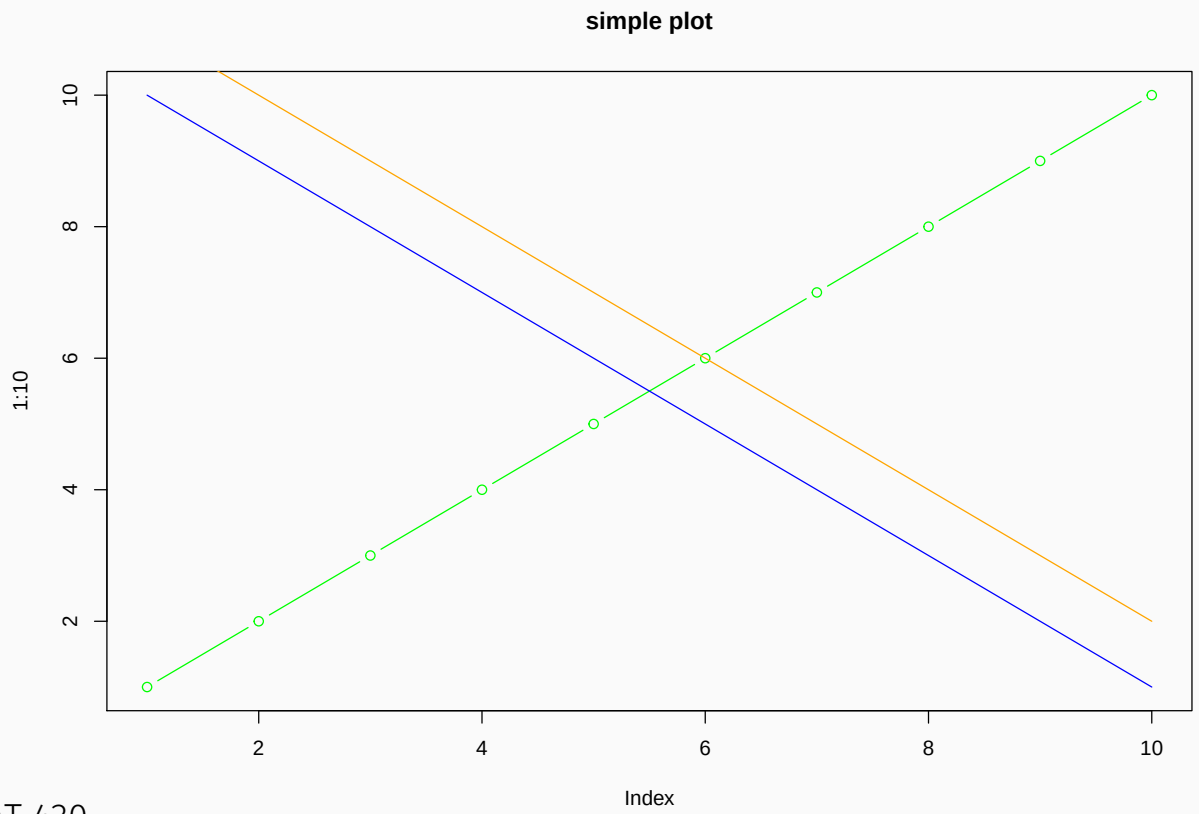
```
plot(1:10, main="simple plot", type="b", col="green")
```



Plots 10 points, sets a title

Modifies plot attributes

```
plot(1:10, main="simple plot", type="b", col="green")
lines(10:1, col="blue")
lines(1 + 10:1, col="orange")
```



Plots 10 points, sets a title

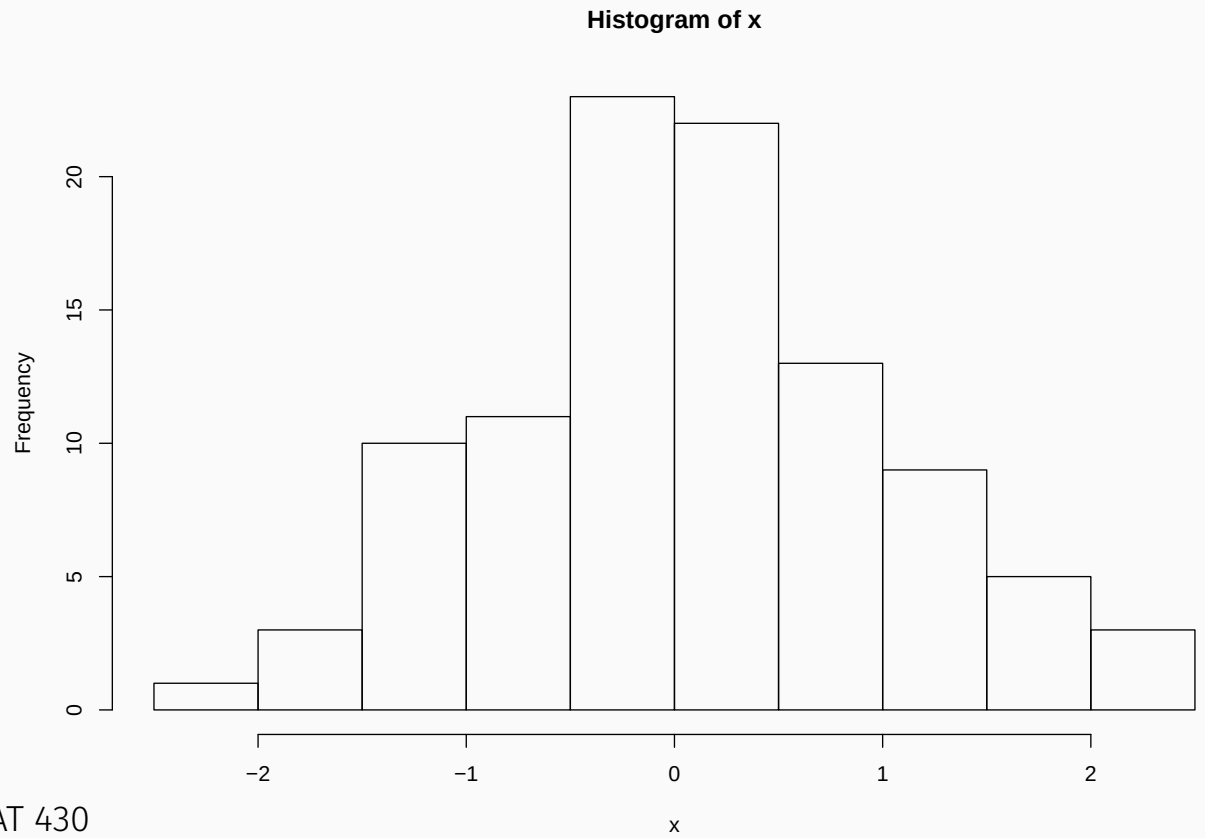
We can add element via command such as `lines` or `points`

Initial plot determines x and y axis limits so second line is clipped.

Plot styling

- The `par()` function offers numerous arguments for styling plots
- There are too many for us to consider all, but a selection is
  - `par(mfrow=c(1,2))` sets a one by two row-wise grid
  - `par(mfcol=c(2,1))` sets a one by two col-wise grid
  - `par(mar=c(2,1,1,1))` sets bottom/left/top/right margins
- Can assign return value of call to `par()` to restore values (see below for example)
- There is a good overview of `par()` settings somewhere...
- Fallback: `help(par)`

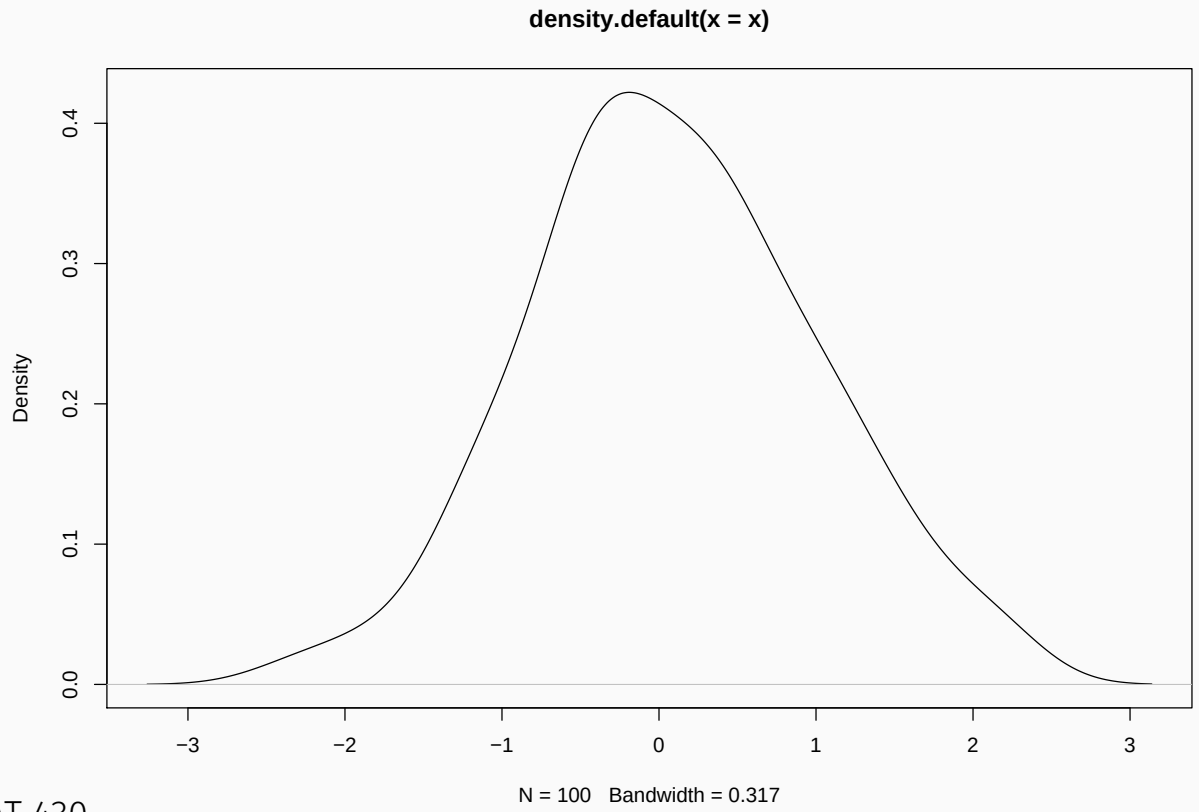
```r
set.seed(123)
x <- rnorm(100)
hist(x)
```



Histogram

You may want to set binning options.

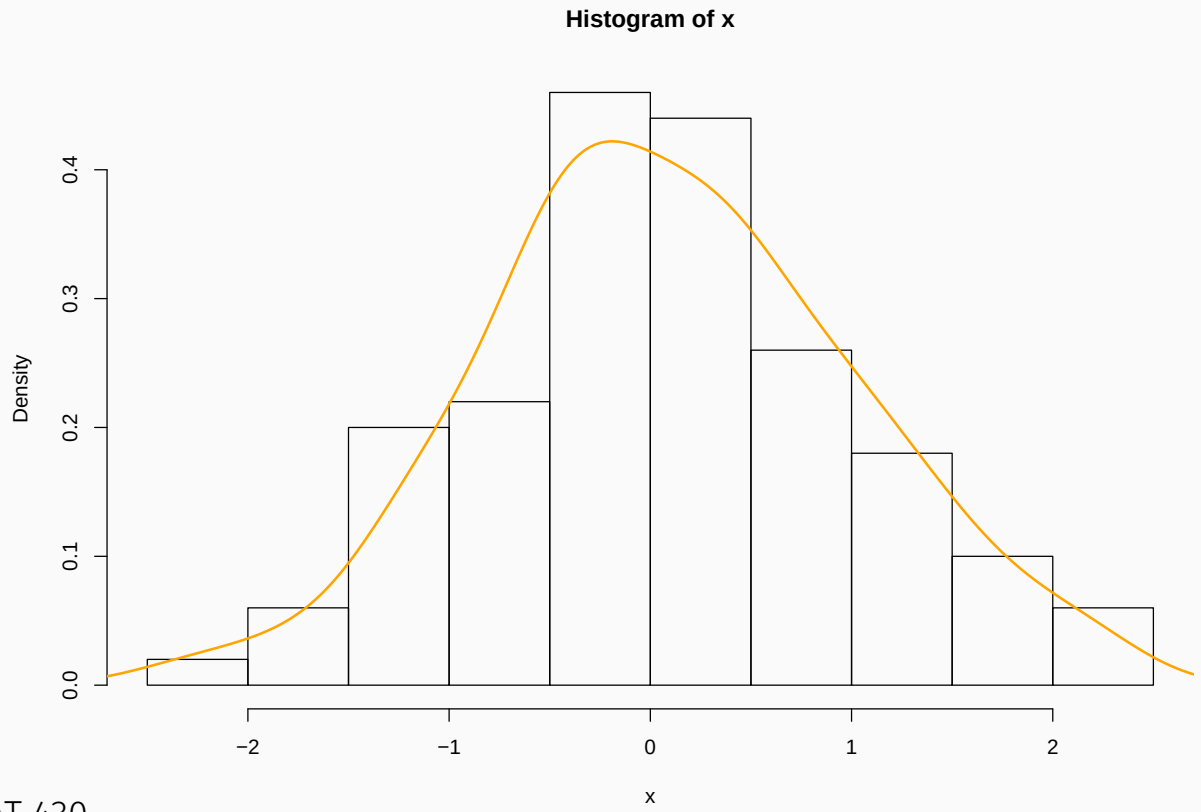
```r
set.seed(123)
x <- rnorm(100)
plot(density(x))
```



Density

You may want to set smoothing options.

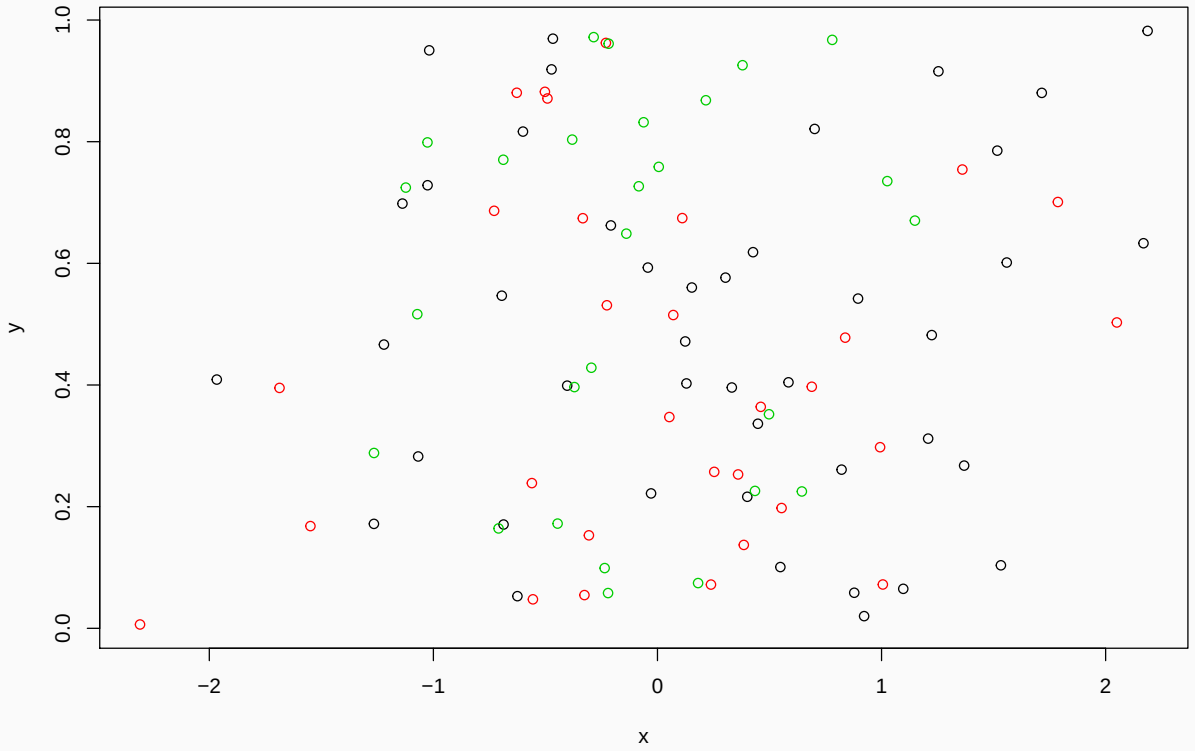
```
set.seed(123); x <- rnorm(100)
hist(x, freq=FALSE)
lines(density(x), col="orange", lwd=2)
```



Histogram and density

We select probabilities for the *y*-axis of the histogram so that the density can be added on the same scale.

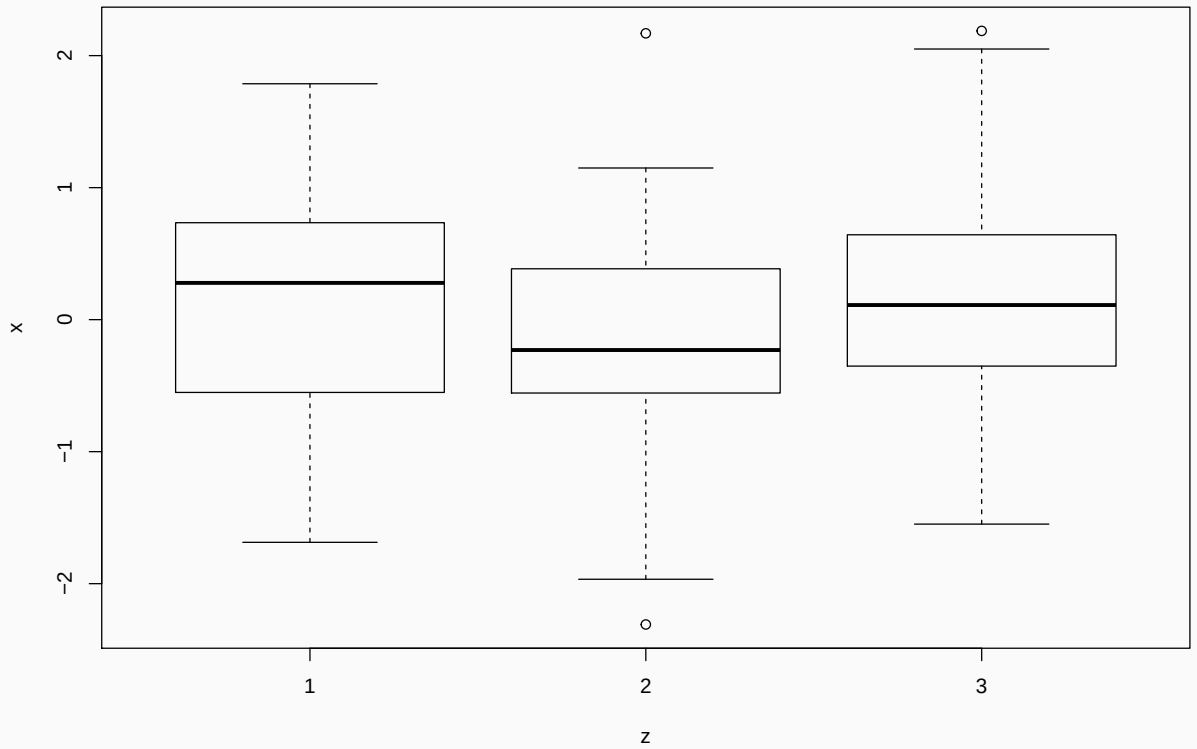
```r
set.seed(123); x <- rnorm(100); y <- runif(100)
z <- sample(1:3, 100, replace=TRUE)
plot(x, y, col=z)
```



Scatterplot

With `par()` arguments `cex` (for size) and `pch` (for plot symbol) more information can be conveyed.

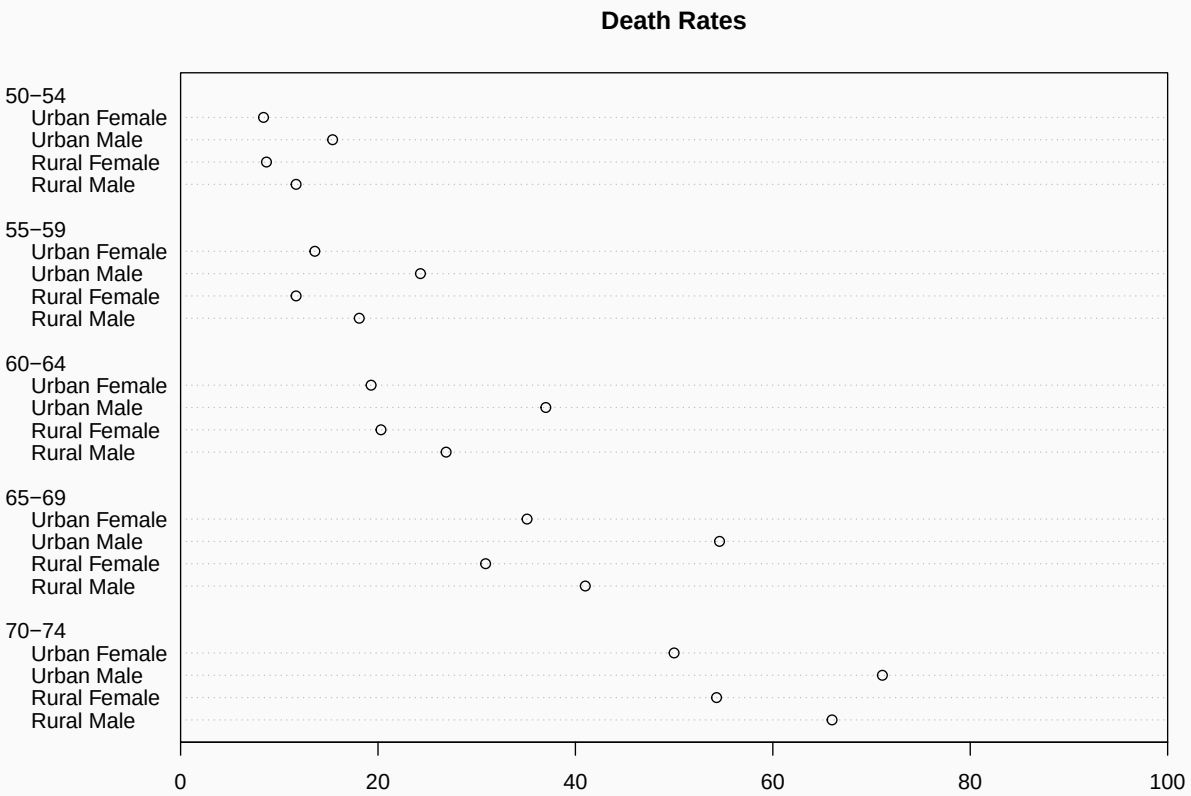
```r
set.seed(123); x <- rnorm(100)
z <- sample(1:3, 100, replace=TRUE)
boxplot(x ~ z)
```



Boxplot

Summarises distribution: the first and third quartile form "box" containing the median; "hinges" extended it (see `help(boxplot.stat)` for details).

```
# from example(dotchart)
dotchart(t(VADeaths), xlim = c(0,100),
         main = "Death Rates")
```

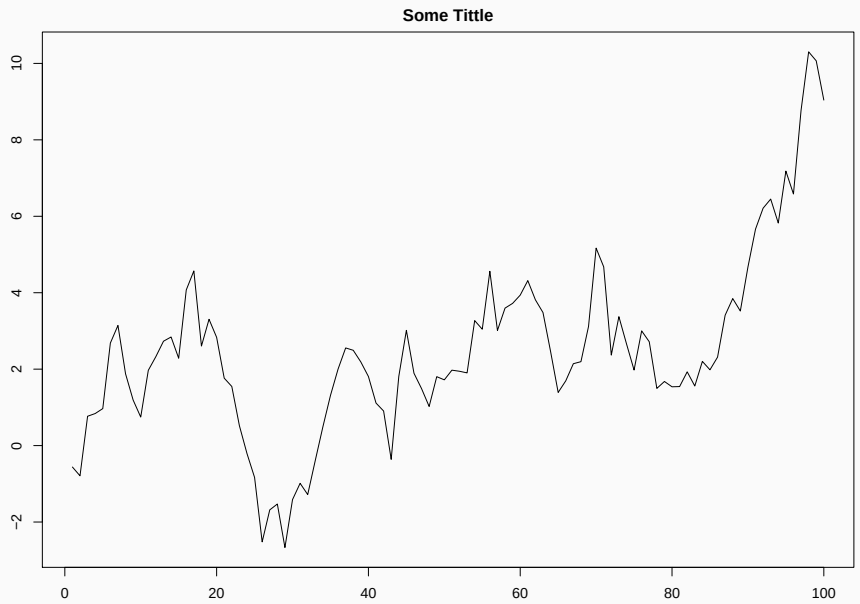


Dotchart

Popularized by Bill Cleveland as an alternative to barplots.

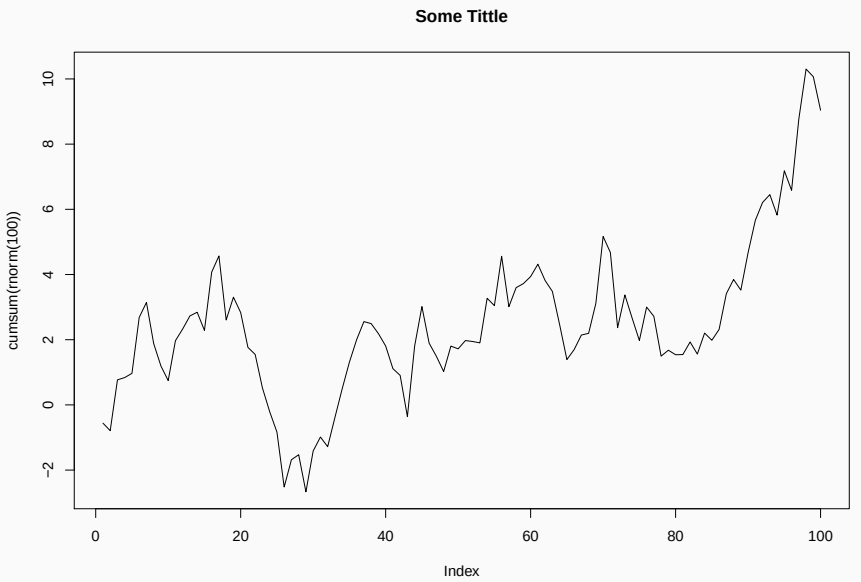
Note the grouping here: `VADeaths` is already in table form.

```
## bottom/left/top/right margin
par(mar=c(2,2,2,1))
set.seed(123)
plot(cumsum(rnorm(100)), type='l',
     main="Some Tittle")
```



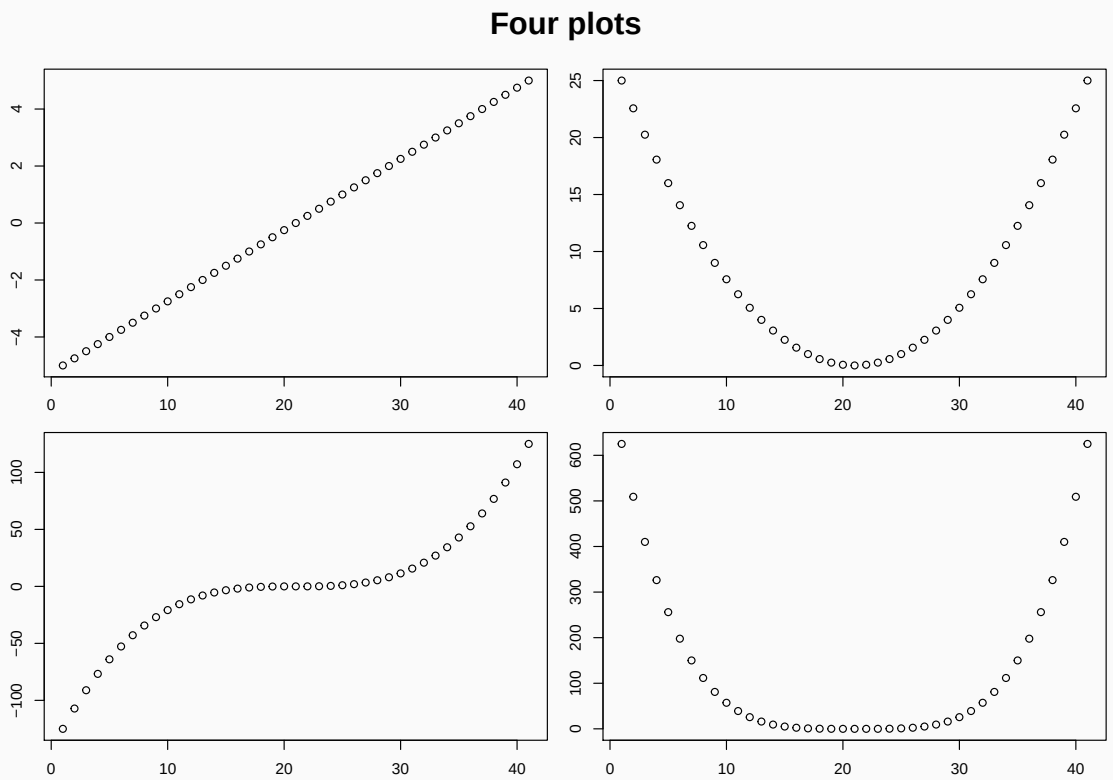
More focus on plot by setting narrower margins.

```
## default values
## ie no changes
set.seed(123)
plot(cumsum(rnorm(100)), type='l',
     main="Some Tittle")
```



Default leaves a lot of whitespace.

```
par(mfrow=c(2,2), mar=c(2,2,1,1),  oma=c(0,0,3,0))
x <- seq(-5,5,by=0.25)
plot(x);   plot(x^2); plot(x^3); plot(x^4)
title("Four plots", outer=TRUE, cex.main=2) # larger
```

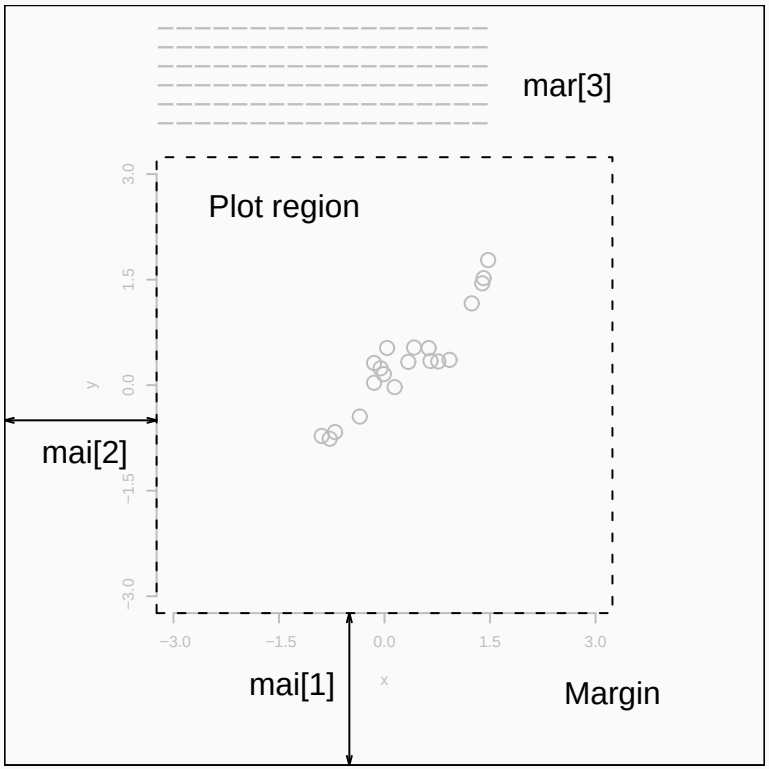


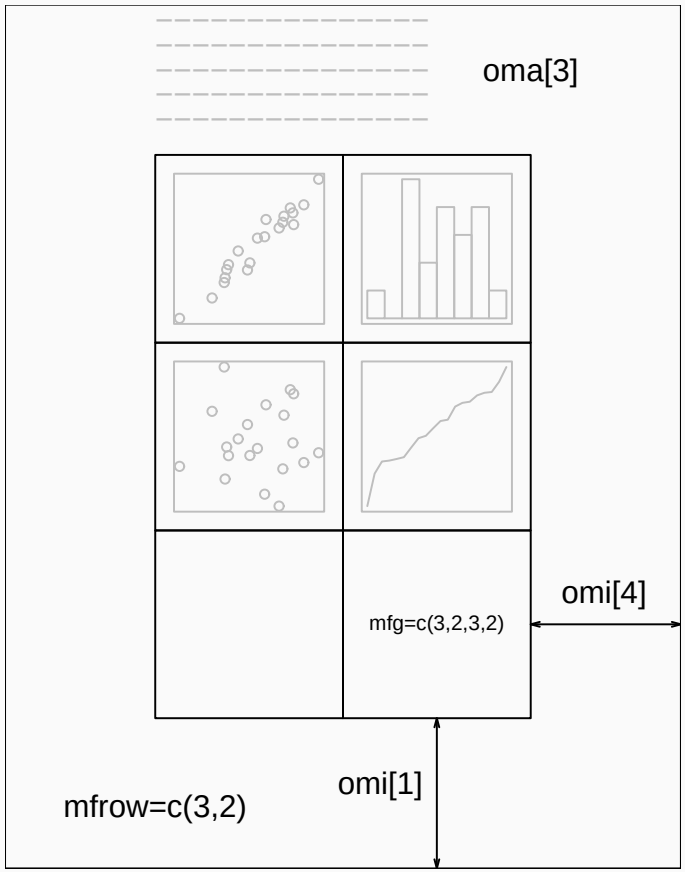
`mfrow()` (and `mfcol()`) easy way for multiple same-size charts.

Can be combined with `par(oma=c(...))` to set *outer margin*.

`title()` uses options to be placed in outer margin.

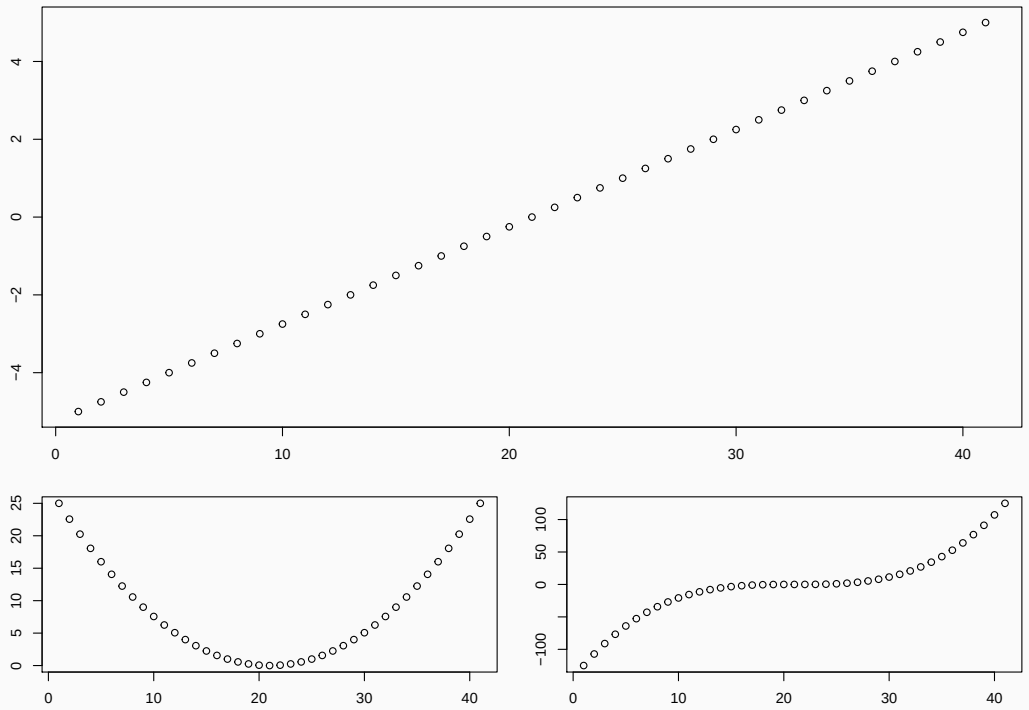
From `help(par)` and the *Introduction to R* manual:





Source: Figures 11 and 12 from Section 12 on *An Introduction to R*; pdf files included in the R source directory `doc/manual/images/`.

```
M <- matrix(c(1,1,2,3), 2, 2, byrow=TRUE)
par(mar=c(3,3,1,1))
layout(M, heights=c(0.66,0.33,0.33))
x <- seq(-5,5,by=0.25)
plot(x); plot(x^2); plot(x^3)
```



Use `layout()` for richer arrangements

The matrix `M` determines plot order sequence via its integer values.

Plot 1 fills the entire first two, 2 and 3 in next row.

`widths()` (not use) and `heights()` give additional scaling.

Cheatsheets and overviews

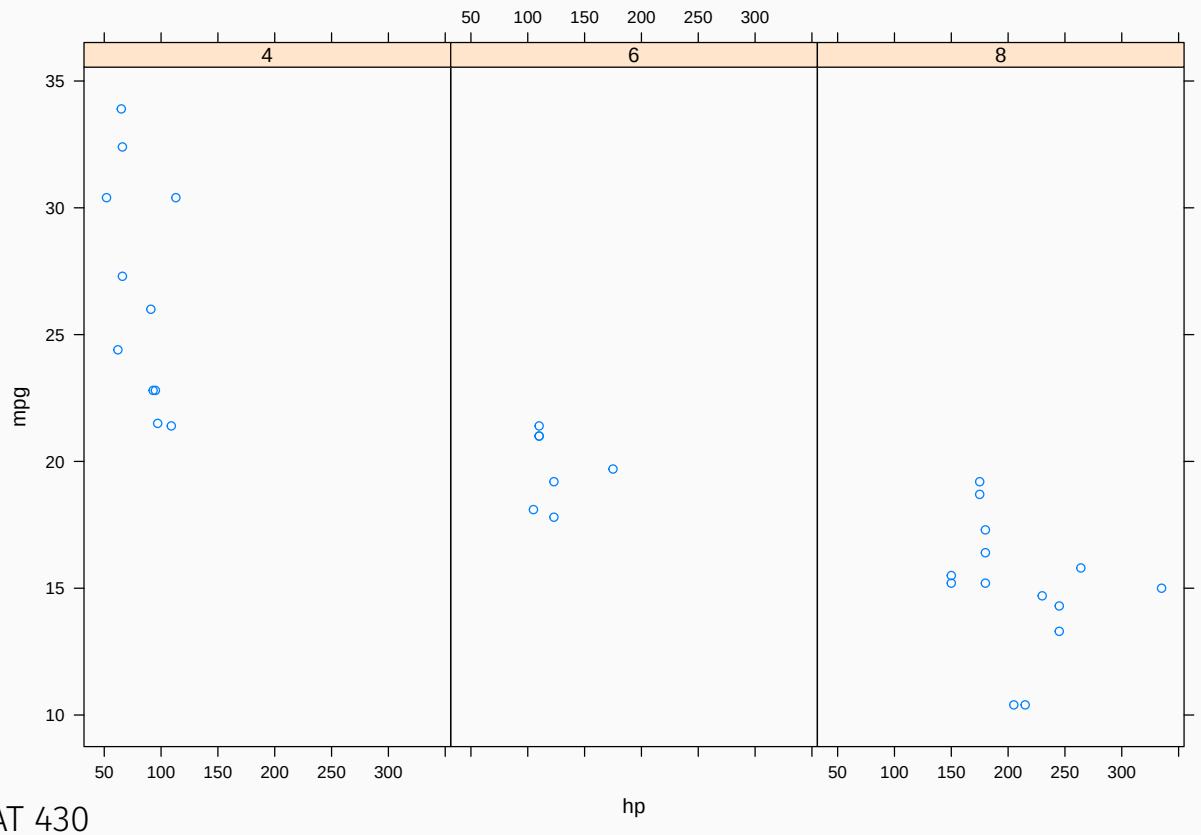
- Generally more examples and usage patterns that we can cover
- Many resources out here
  - One-page Base R Graphics cheatsheet
  - Nice Base R Graphics Cheatsheet website

Lattice

Another Family

- `lattice` is a very powerful system
- It is also somewhat complex
- Written by Deepayan Sarkar while at U Wisc
- Predecessor S / S-Plus had 'Trellis'
- ... which `lattice` reimplements
- ... now somewhat overshadowed by `ggplot2`
- ... but still worth knowing about

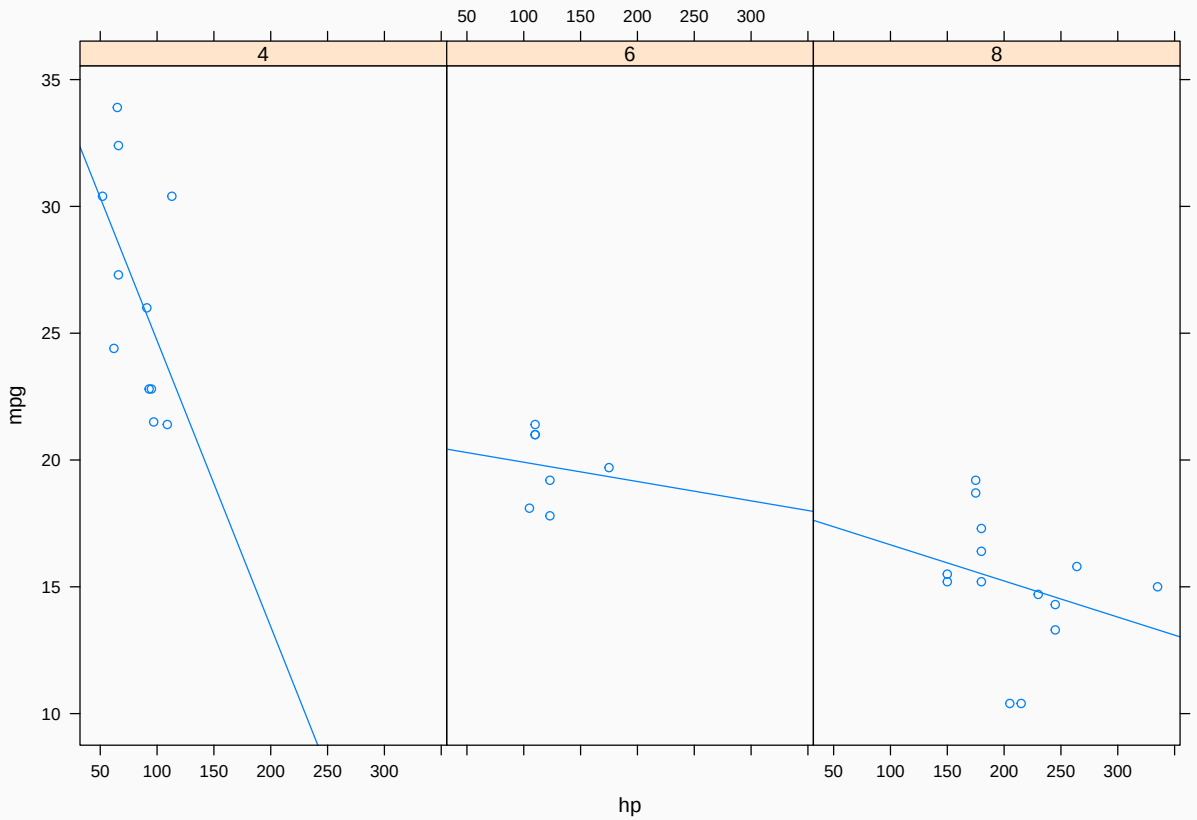
```
library(lattice)
xyplot(mpg ~ hp | factor(cyl),
       data=mtcars)
```



xyplot

Scatter of miles-per-gallon and horsepower, conditioned on cylinders

```
xyplot(mpg ~ hp | factor(cyl), type=c("p", "r"),
       data=mtcars)
```

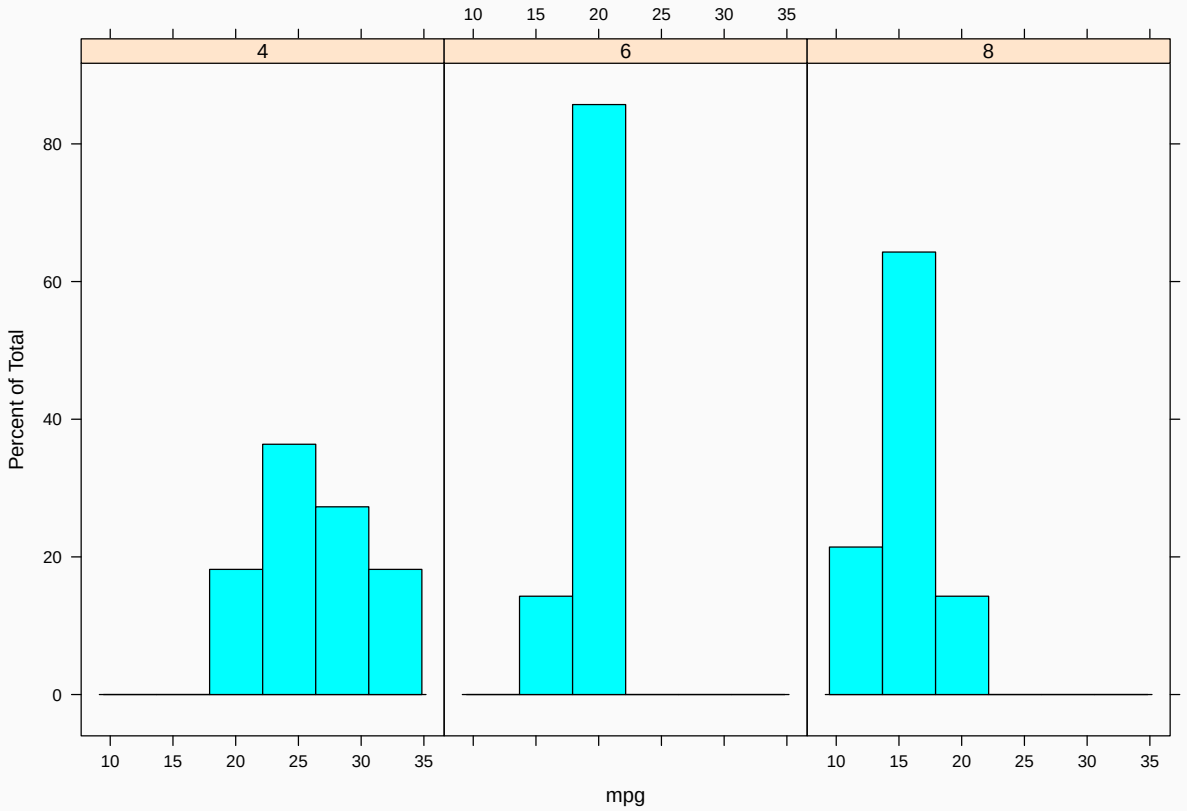


xyplot

Scatter of miles-per-gallon and horsepower, conditioned on cylinders

Now with added 'p'oints and 'r'egression.

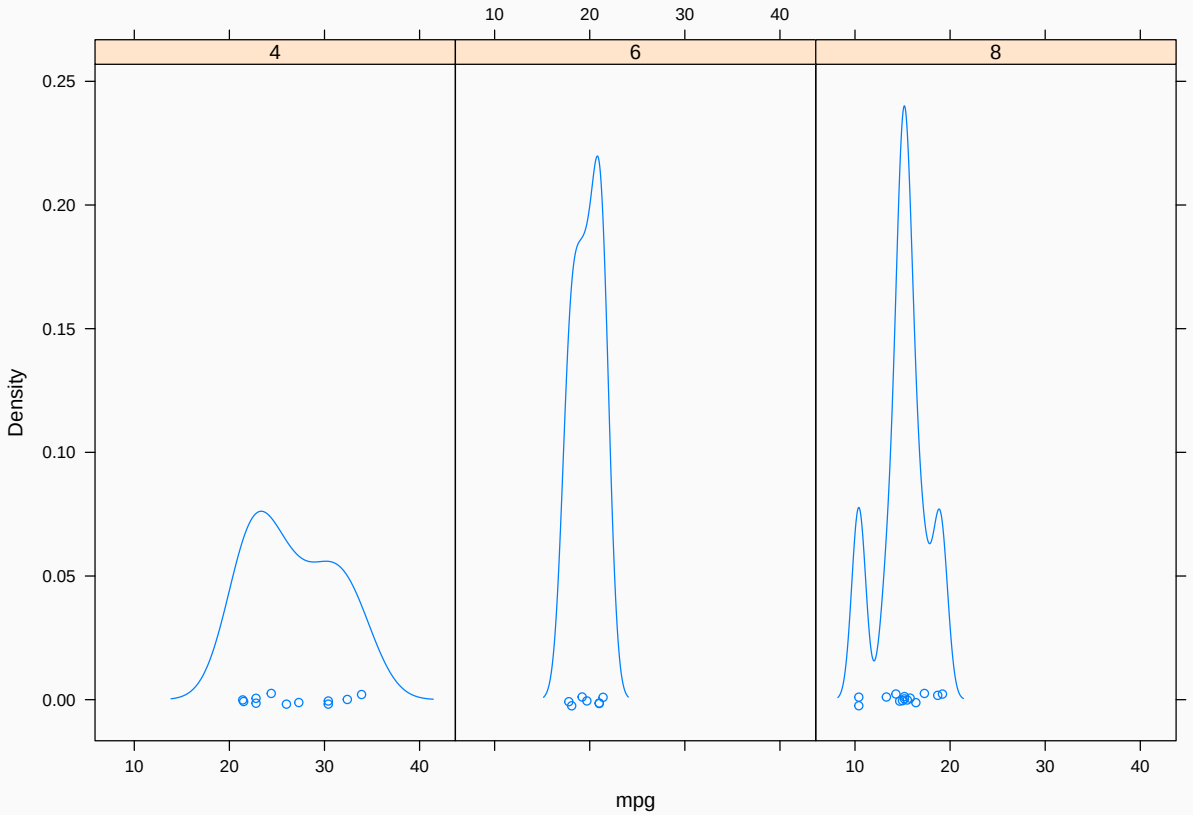
```
histogram(~ mpg | factor(cyl),
          data=mtcars)
```



histogram

Conditional on (factor variable) cylinders

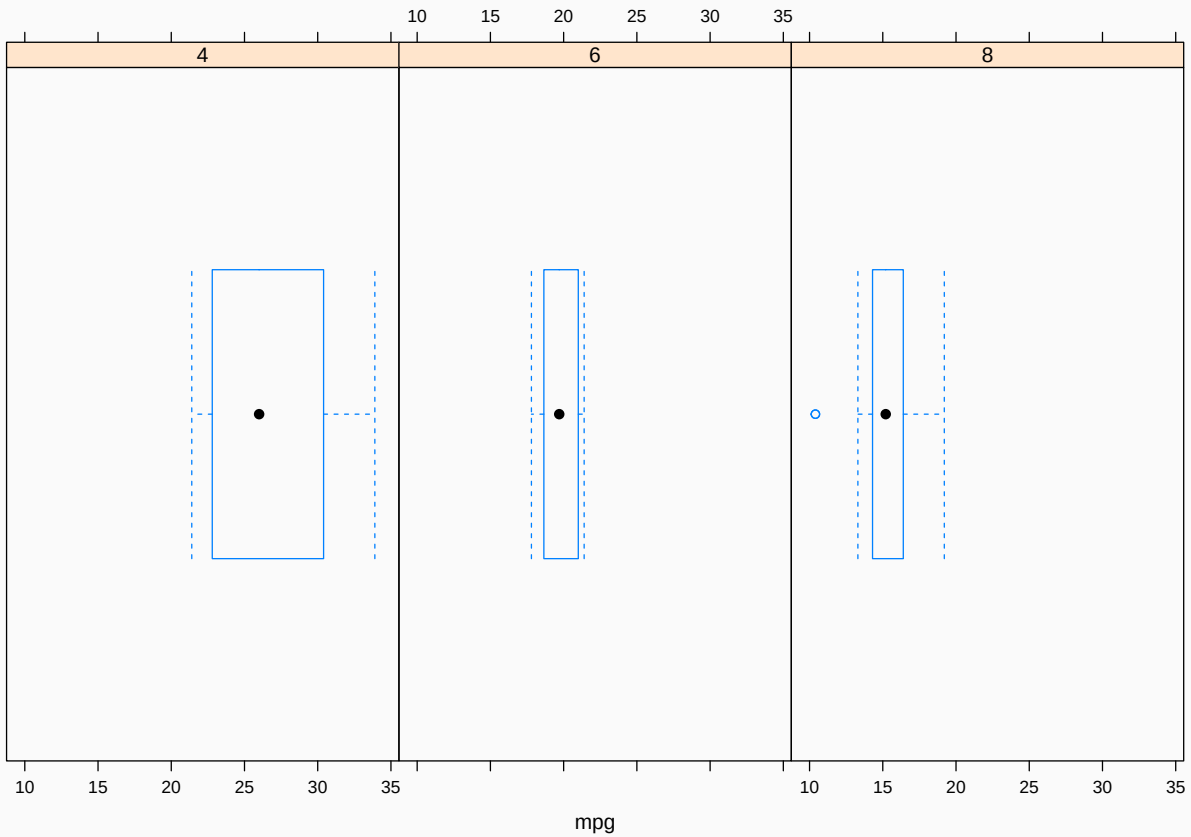
```
densityplot(~ mpg | factor(cyl),
            data=mtcars)
```



densityplot

Conditional on (factor variable) cylinders

```
bwplot(~ mpg | factor(cyl),
       data=mtcars)
```



bwplot

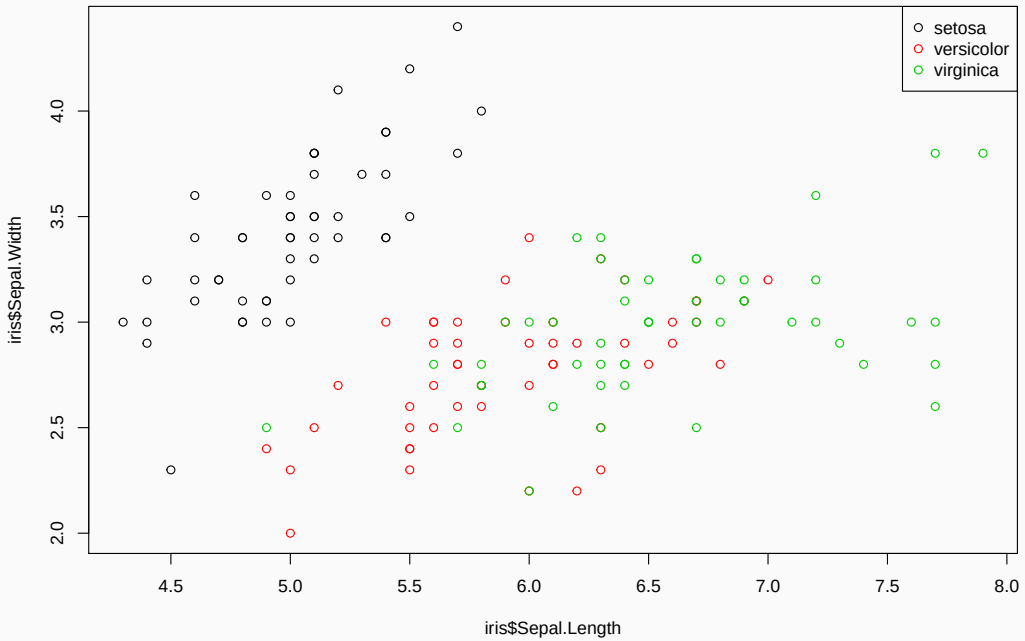
Conditional on (factor variable) cylinders

Much more to explore

- Several extra packages with themes
  - `latticeExtra` is quite nice
- Several key features we have not explore
  - notably panel function
  - applied to each panel
  - permit fine-grained visualization

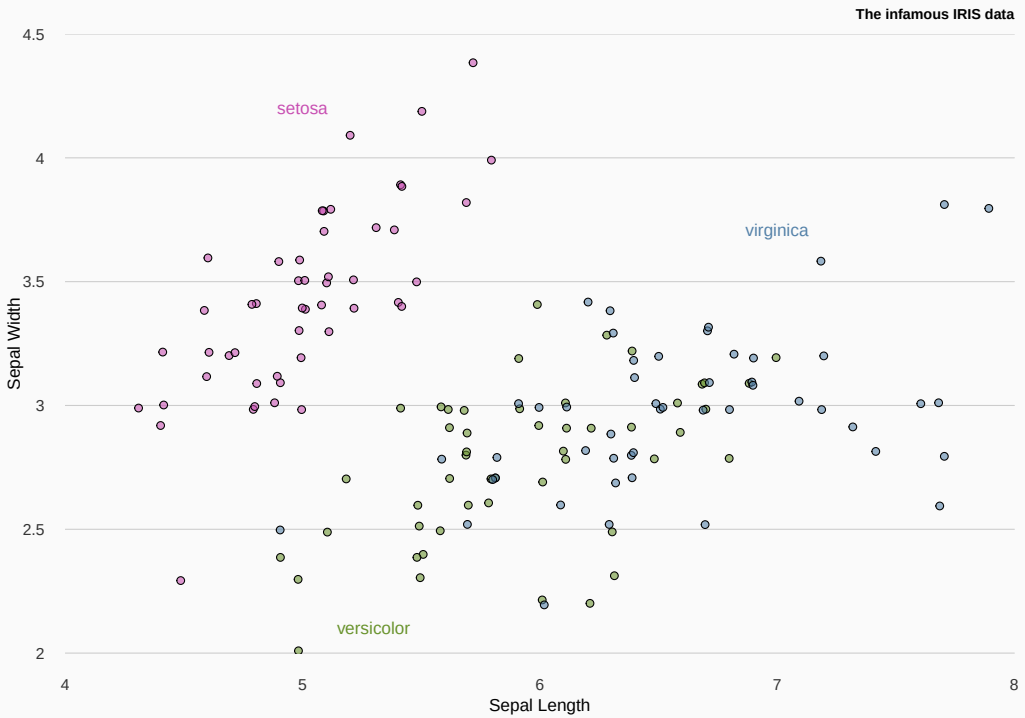
Styling

```
plot(iris$Sepal.Length, iris$Sepal.Width, col = iris$Species)
legend("topright", legend = levels(iris$Species),
       col = 1:3, pch = 21)
```



On the JumpingRivers blog, the co-author of Efficient R gives a fine example.

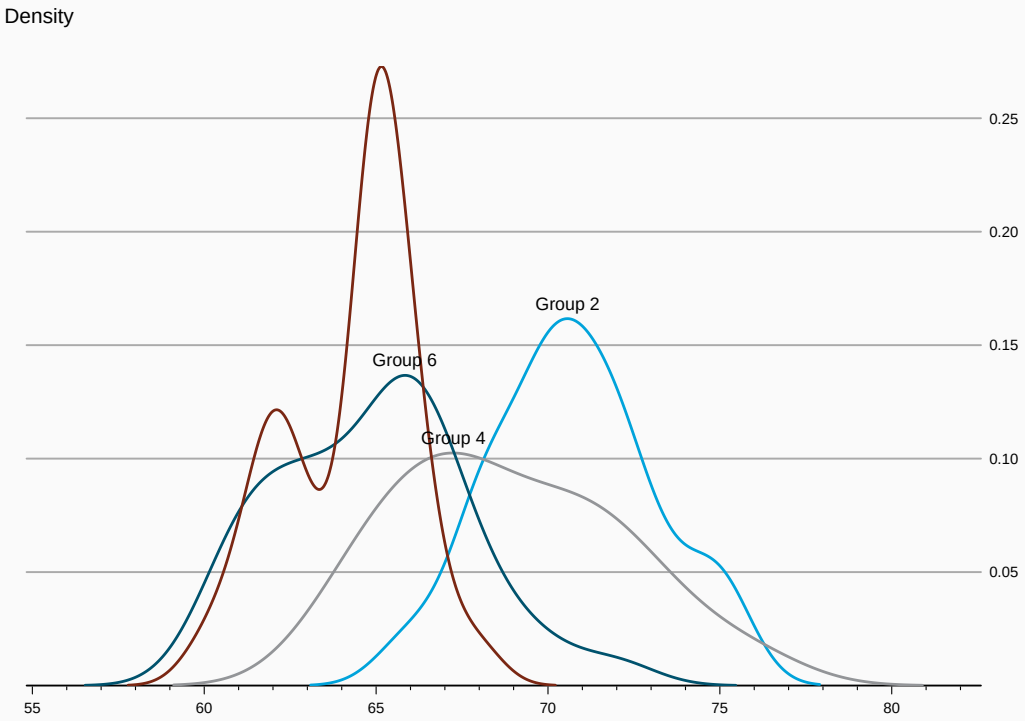
On the left is a standard plot with default values for many style elements. It is passable, but not too dressed up.



This updated version of the same data set looks more refreshed, and involves just a few lines of specific code.

One Option: Package `latticeExtra`

```
suppressMessages(library(latticeExtra))
asTheEconomist(
  densityplot(~ height, groups = voice.part,
              data = singer,
              subset = grep("1", voice.part),
              plot.points = FALSE)) +
  glayer(d <- density(x), i <- which.max(d$y),
         ltext(d$x[i], d$y[i],
               paste("Group",
                     group.number), pos = 3))
```



See the two packages for details.

Base Graphics

- Always static: initial plot sets up 'canvas'
- Can add with `lines()`, `points()`
- Parameterisation via `par()`
- Richer placement via `layout()`
- `lattice` graphics offer rich layering

Next lecture: ggplot2