3. Plotting functions and formulas

Ken Rice
Ting Ye

University of Washington

Seattle, July 2021

In this session

R is known for having good graphics – good for data exploration and summary, as well as illustrating analyses. Here, we wil see;

- Some *generic* plotting commands

- Making graphics files

- Fine-tuning your plots (and why not to do too much of this)

- The formula syntax

NB more graphics commands will follow, in the next session.

Making a scatterplot with `plot()`

A first example, using the `mammals` dataset − and its output in the Plot window; (The preview button is recommended)

```
plot(x=mammals$body, y=mammals$brain)
```
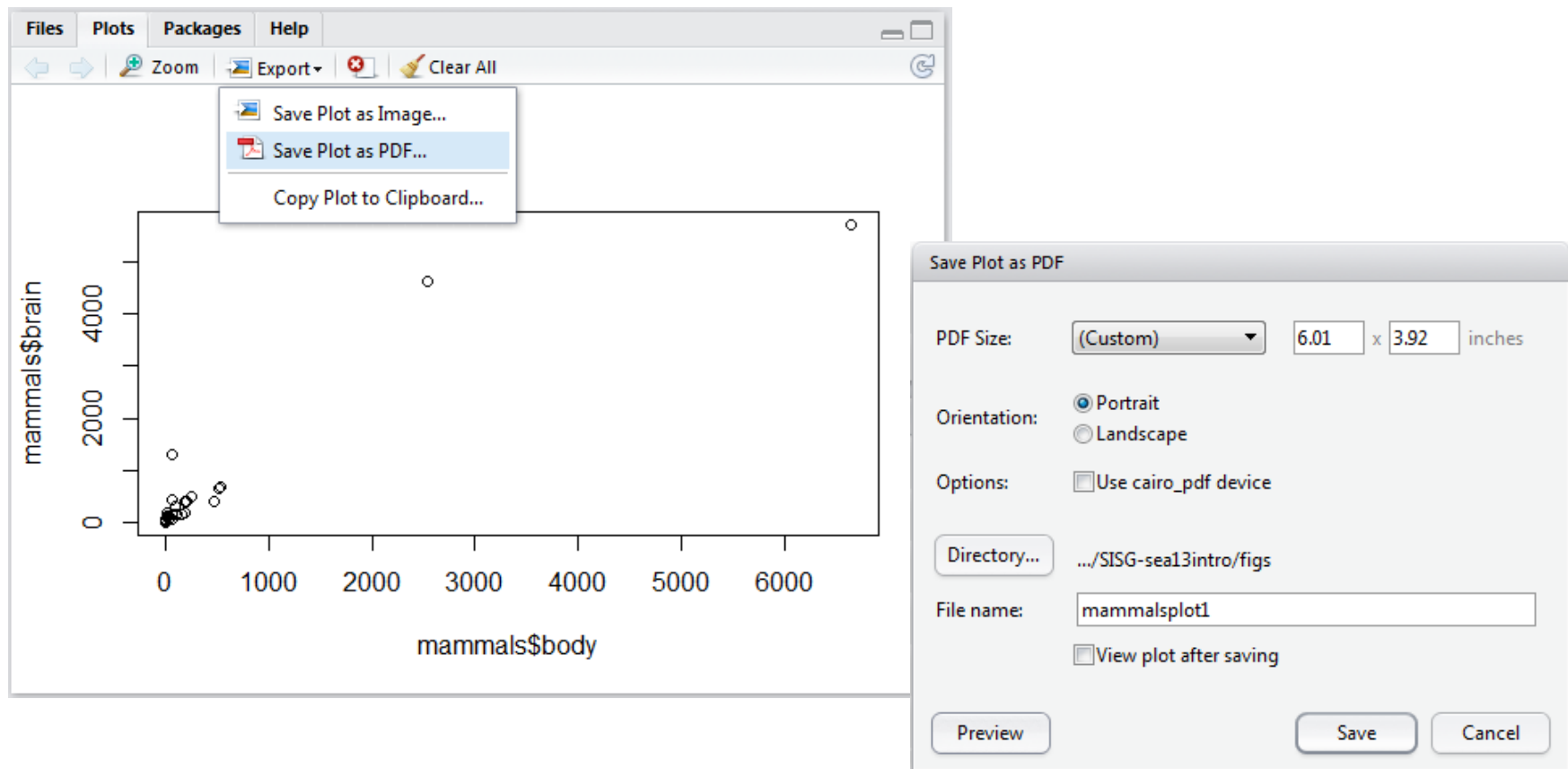
Making a scatterplot with `plot()`

Some other options for exporting;

- Copy directly to clipboard as a bitmap or editable (Windows) metafile - then paste into e.g. your Powerpoint slides
- With 'Save Plot as Image', PNG is a (good) bitmap format, suitable for line art, i.e. graphs. JPEG is good for photos, not so good for graphs
- For PNG/JPEG, previews disappear if they get too large!
- Many of the options (TIFF, EPS) are seldom used, today
- Handy hint; if too much re-sizing confuses your graphics device (i.e. the Plot window) enter `dev.off()` and just start over
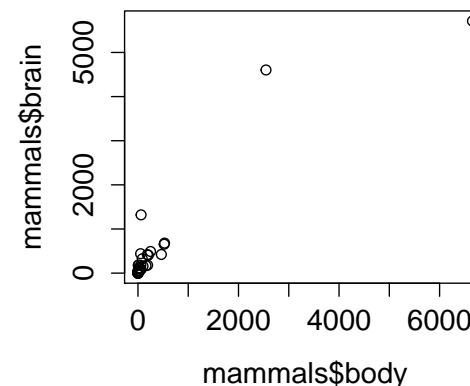
Making a scatterplot with `plot()`

A golden rule for exporting;

> Make the file the size it will be in the final document —
> because R is good at choosing font sizes

A 6:4 plot, saved
at 24×16 inches
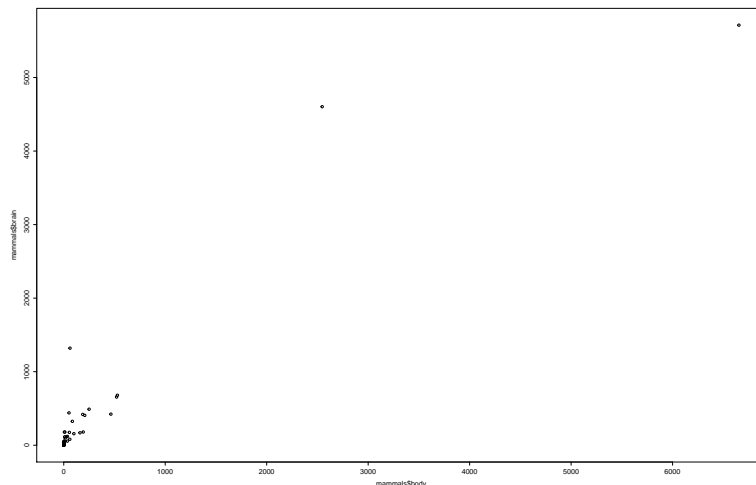
The same plot,
saved at 4×2.67 inches

- Not the same plot 'blown up' — note e.g. axes labels
- R likes to add white space around the edges — good in documents, less good in slides, depending on your software

Making a scatterplot with `plot()`

Better axes, better axis labels and a title would make the scatterplot better. But on looking up `?plot`...

"For simple scatter plots, `plot.default` will be used. However, there are `plot` methods for many R objects, including `functions`, `data.frame`s, `density` objects, etc. Use `methods(plot)` and the documentation for these."

`plot()` is a *generic* function — it does different things given different input; see `methods(plot)` for a full list. For our plot of `y` vs `x`, the details we need are in `?plot.default`...

```
plot(x, y = NULL, type = "p",  xlim = NULL, ylim = NULL,
     log = "", main = NULL, sub = NULL, xlab = NULL, ylab = NULL,
     ann = par("ann"), axes = TRUE, frame.plot = axes,
     panel.first = NULL, panel.last = NULL, asp = NA, ...)
```
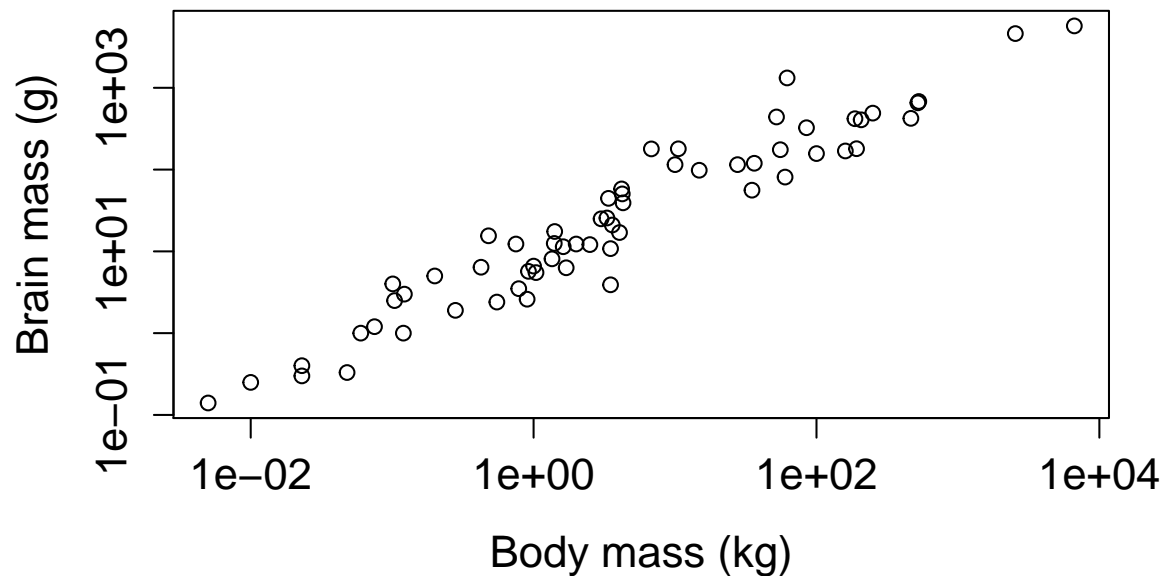
Making a scatterplot with `plot()`

After checking the help page to see what these mean, we use;

- `xlab, ylab` for the axis labels
- `main` for the main title
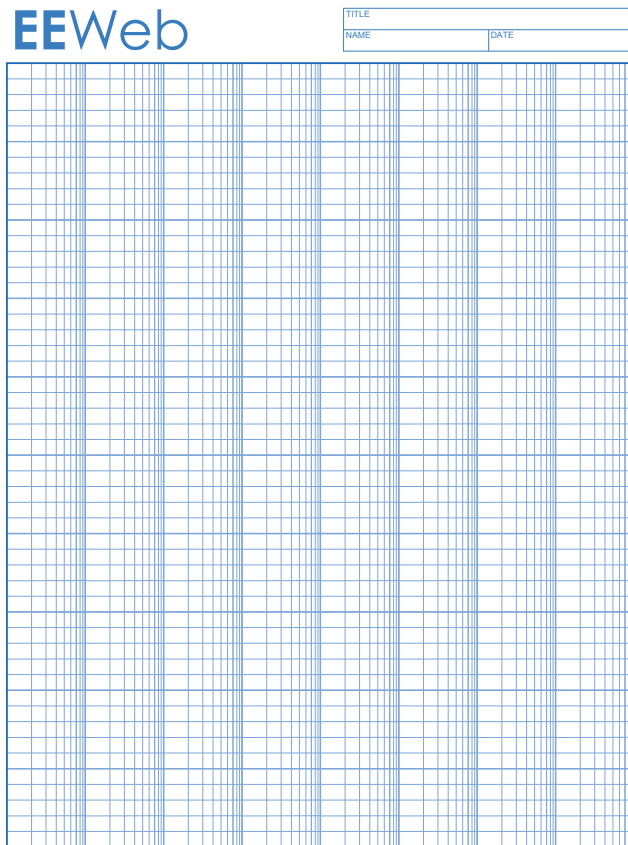- `log` to log the axes − `log="xy"`, to log them both

```
plot(x=mammals$body, y=mammals$brain, xlab="Body mass (kg)",
    ylab="Brain mass (g)", main="Brain and body mass, for 62 mammals",
    log="xy")
```
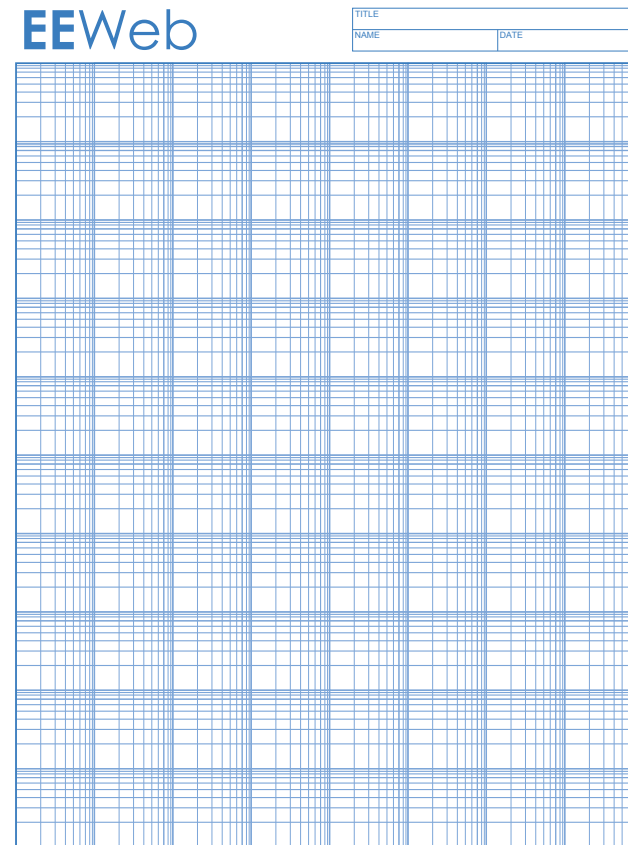
Brain and body mass, for 62 mammals

Making a scatterplot with `plot()`

For those with historical interests (or long memories);
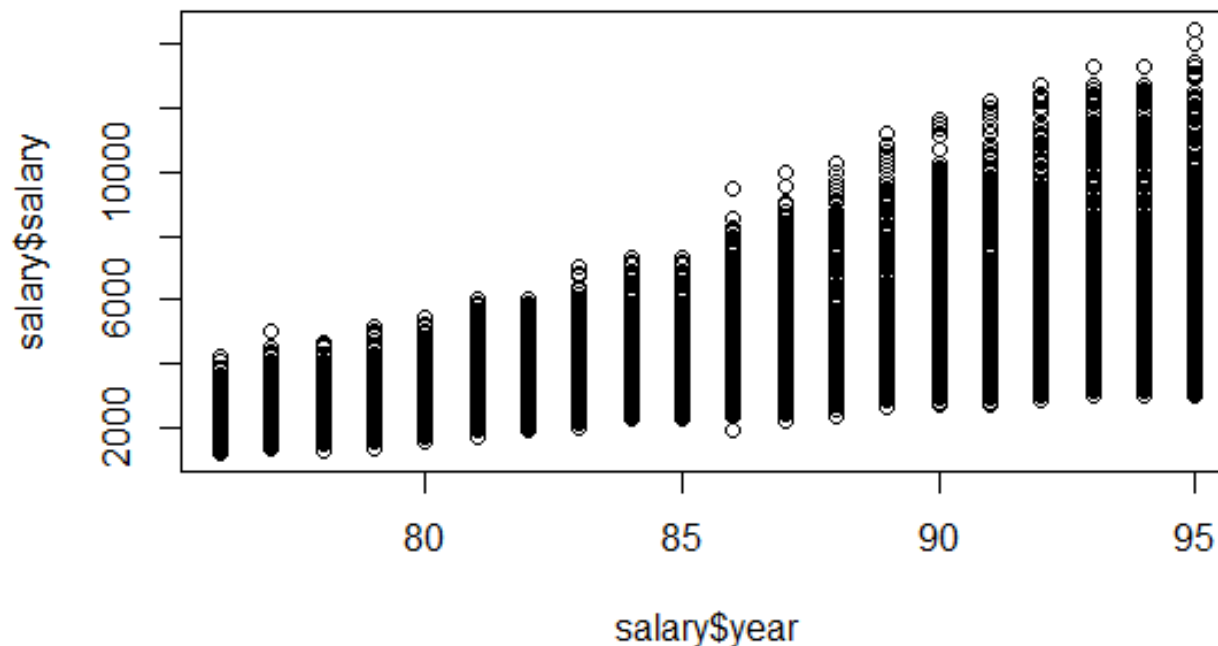

`log="x"`
Semi-log graph paper

`log="xy"`
Log-log graph paper

Other plots made with `plot()`

As the help file suggests, `plot()` gives different output for different types of input. First, another scatterplot;

```
plot(x=salary$year, y=salary$salary)
```


Tip: export graphs of large datasets as PNG, not PDF or JPEG.

3.8

Other plots made with `plot()`

Plotting a numeric variable against a character gives an error;
```
> plot(x=salary$rank, y=salary$salary)
Error in plot.window(...) : need finite 'xlim' values
```

Could use `x=as.factor(salary$rank)`, but gets tiresome. Instead:
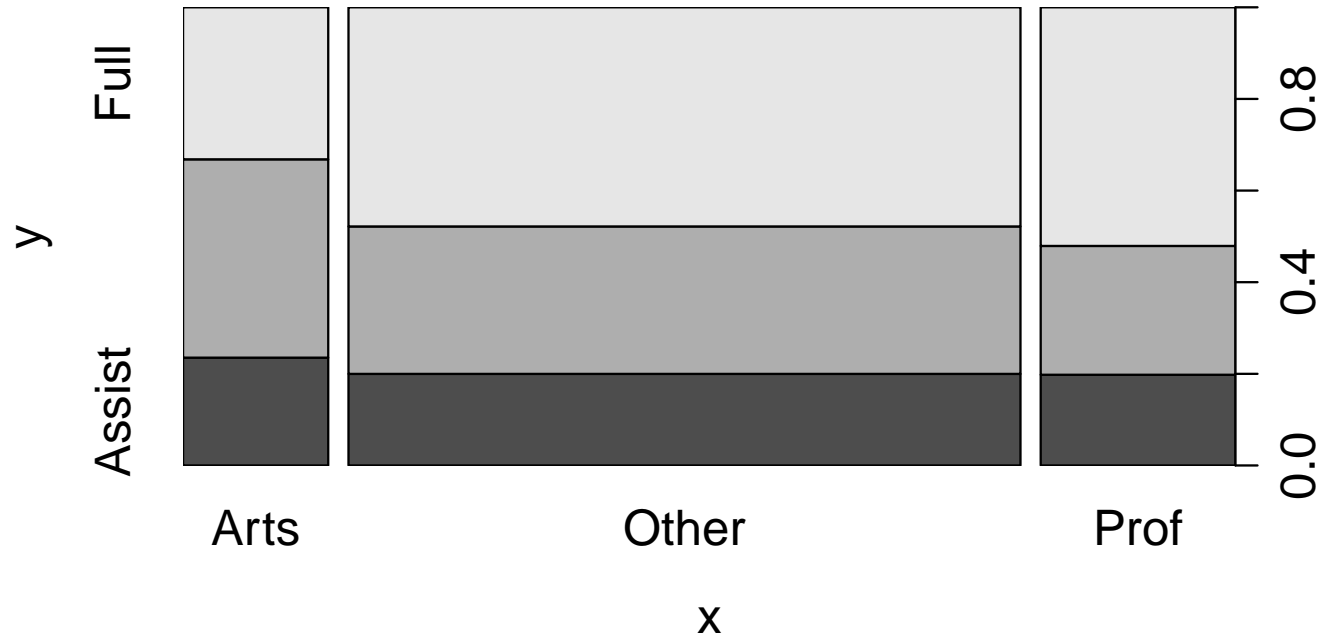
```
salary <- read.csv("http://faculty.washington.edu/kenrice/rintro/salary.csv",
                   stringsAsFactors = TRUE)
plot(x=salary$rank, y=salary$salary)
```
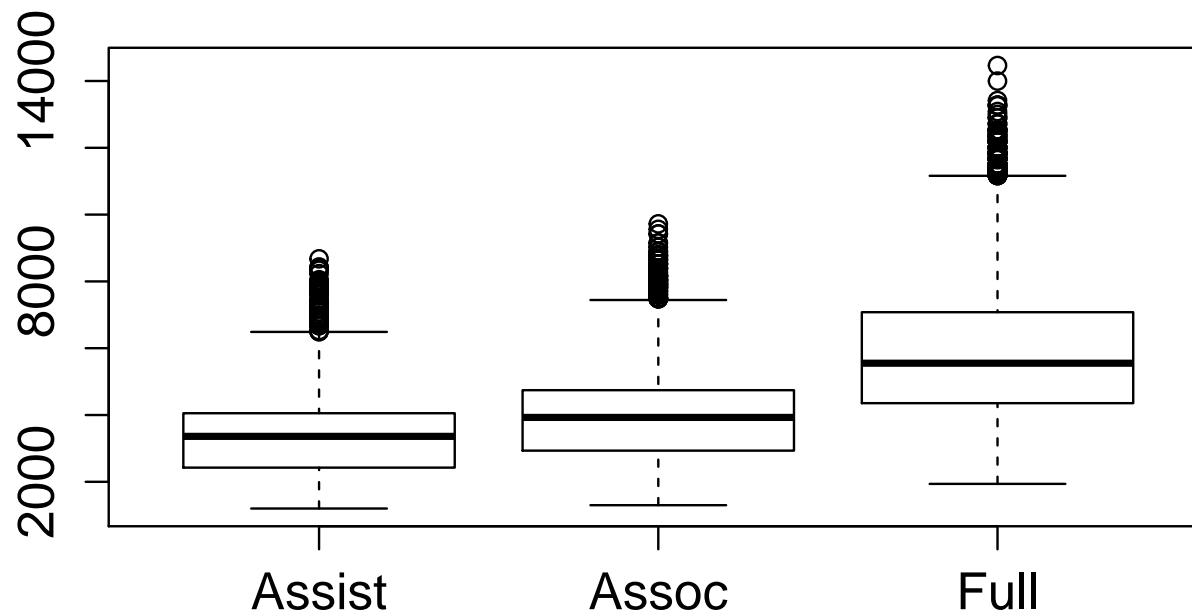

There is also a `boxplot()` function.

Other plots made with `plot()`

Plotting one factor variable against another;

```
plot(x=salary$field, y=salary$rank)
```
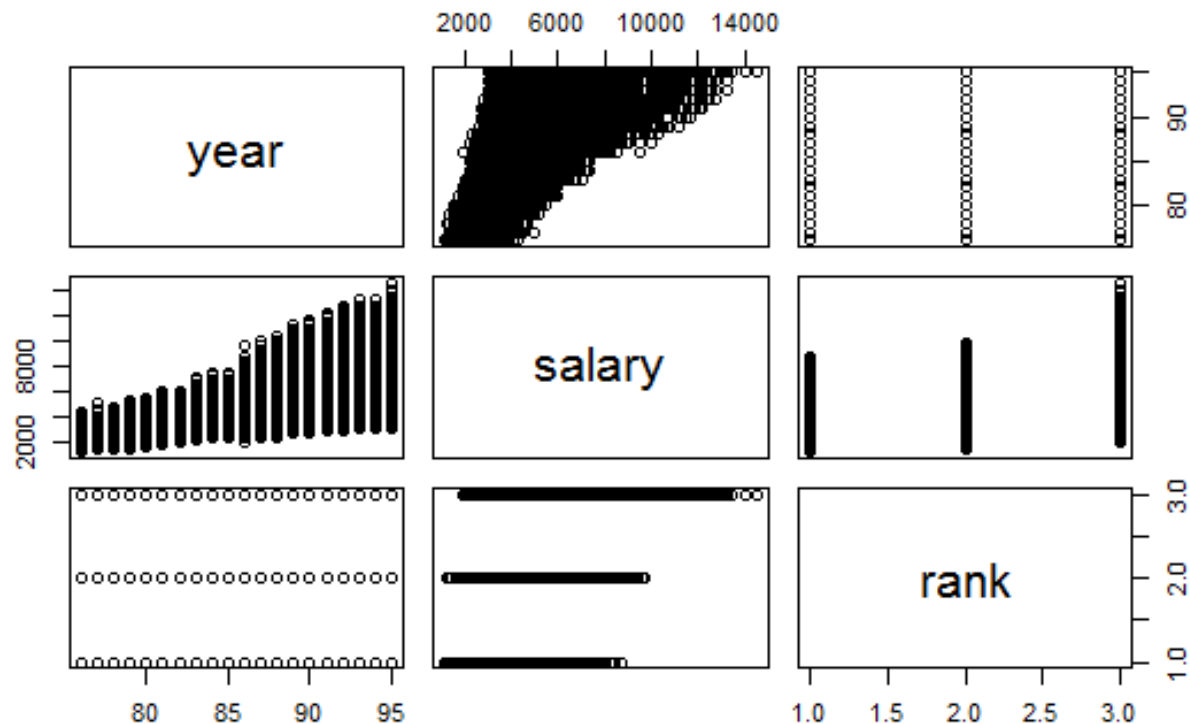

This is a *stacked barplot* — see also the `barplot()` function

Other plots made with `plot()`

Plotting an entire data frame (not too many columns)

```
smallsalary <- salary[,c("year","salary","rank")]
plot(smallsalary)
```


Not so clever! But quick, & okay if all numeric − see also `pairs()`.
NB Plotting functions for large datasets are in later sessions.

Other graphics commands

For histograms, use `hist()`;

```
hist(salary$salary, main="Monthly salary", xlab="salary")
```
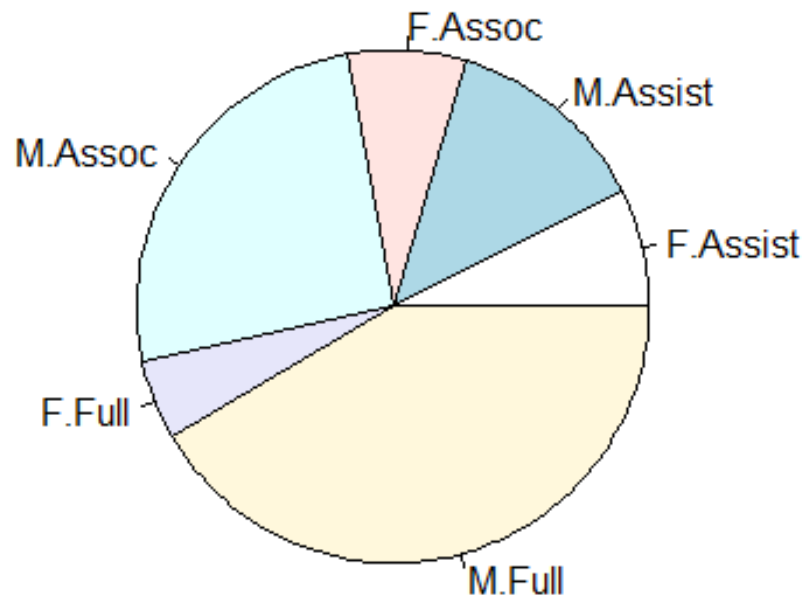
Monthly salary

For more control, set argument `breaks` to *either* a number, or a vector of the breakpoints.

Other graphics commands

Please tell no-one I told you this one;

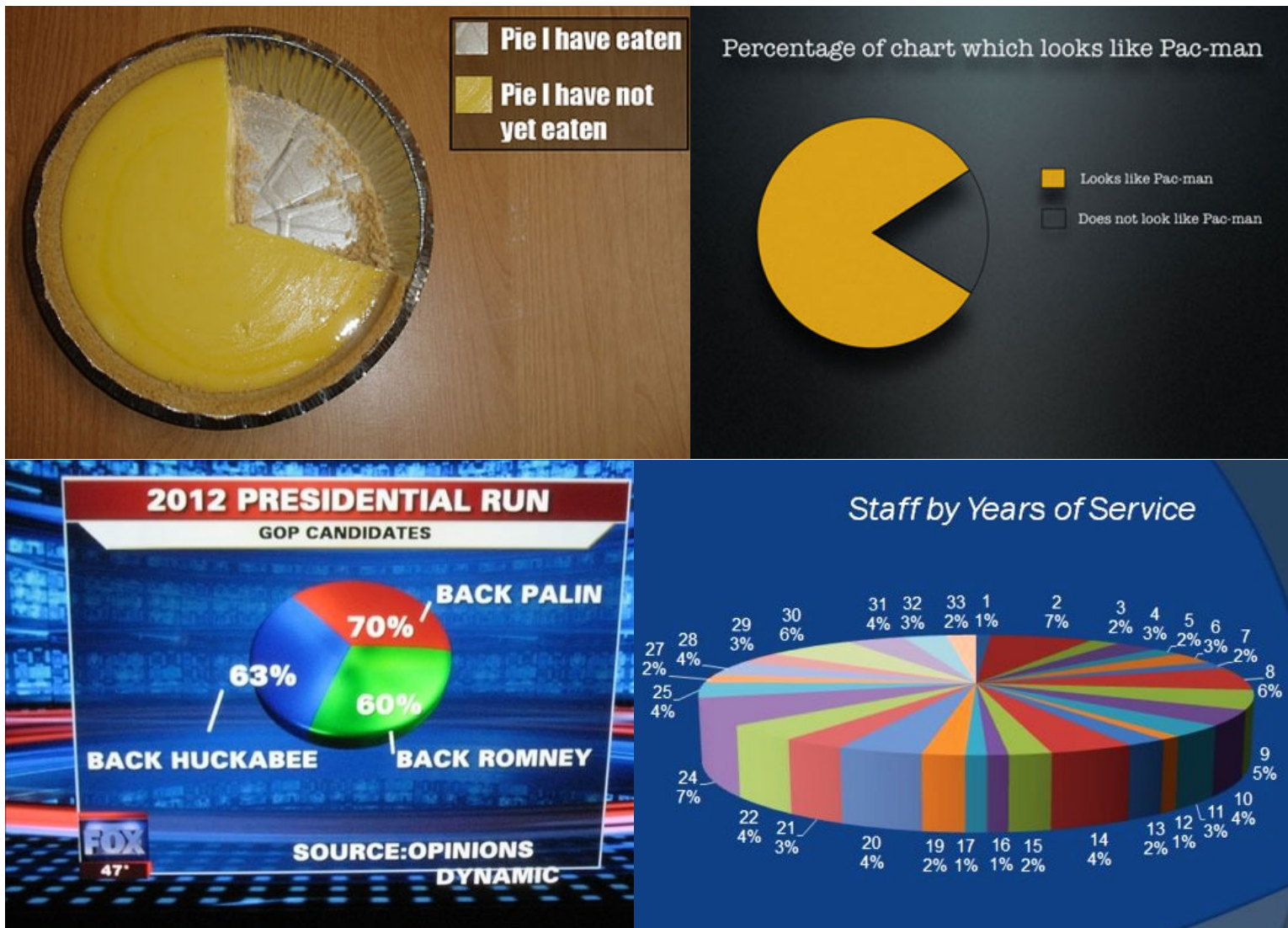```
> table(interaction(salary$gender, salary$rank))
F.Assist M.Assist F.Assoc M.Assoc   F.Full   M.Full
    1460   2588   1465   5064   1001   8210
> pie(table(interaction(salary$gender, salary$rank)))
```



Why do statisticians hate pie charts with such passion?
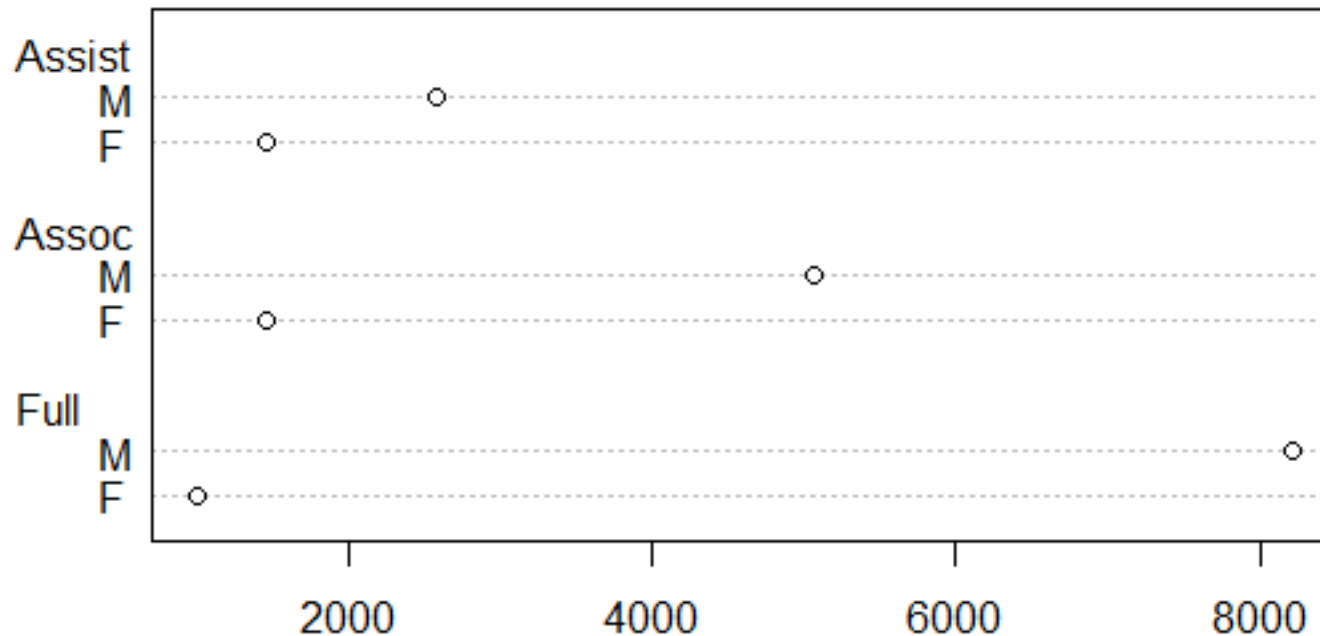
# Other graphics commands

... they really do!

# Other graphics commands

Because pie charts are usually a terrible way to present data. Dotcharts can be *much* better − and are also easy to code;

```
dotchart(table(salary$gender, salary$rank))
```



See also `stripchart()`; with multiple symbols per line, these are a good alternative to boxplots, for small samples.

# Changing plotting symbols

Suppose you want to highlight certain points on a scatterplot; other options to the `plot()` command change point style & color;

```
> grep("shrew", mammals$species) # or just look in Data viewer
[1] 14 55 61
> is.shrew <- 1:62 %in% c(14,55,61) # 3 TRUEs and 59 FALSEs
> plot(x=mammals$body, y=mammals$brain, xlab="Body mass (kg)",
+   ylab="Brain mass (g)",log="xy",
+   col=ifelse(is.shrew, "red", "gray50"), pch=19)
```
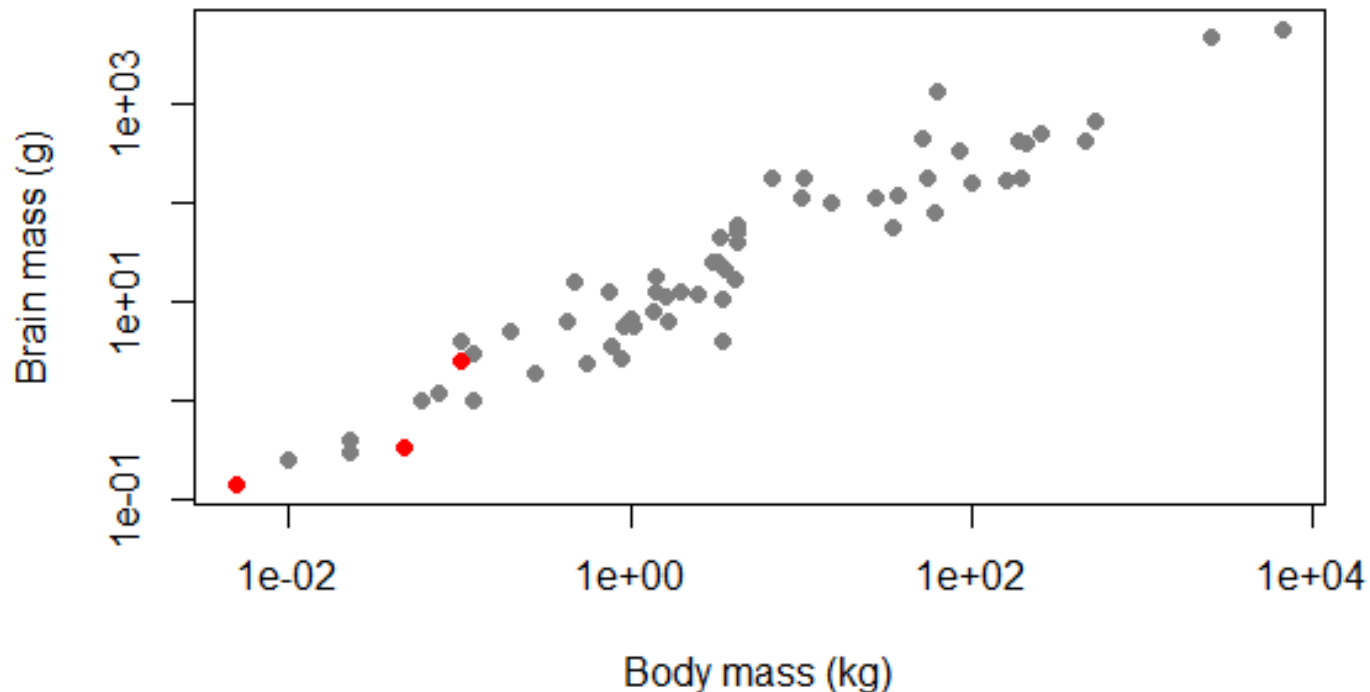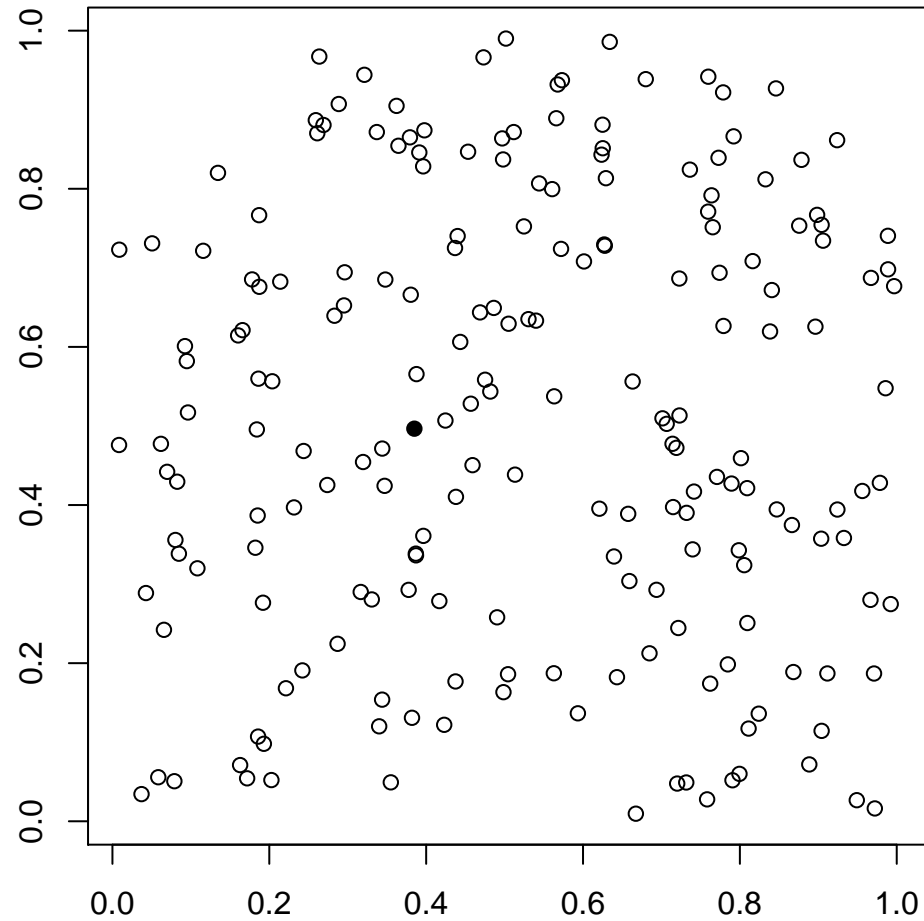
# Changing plotting symbols

We used `col=ifelse(is.shrew, "red", "gray50")` − a vector of 3 reds and 59 `gray50`s.

- If we supply fewer colors than datapoints, what we supplied is recycled
- You could probably guess `"red"`,`"green"`,`"purple"` etc, but not `"gray50"`. To find out the names of the (many) available R colors, use the `colors()` command − no arguments needed
- Can also specify colors by numbers; 1=black, 2=red, 3=green up to 8, then it repeats
- Or consult this online chart − or many others like it
- Can also supply colors by hexadecimal coding; #RRGGBB for red/green/blue − with #RRGGBBTT for transparency − or use `adjustcolor()`

NB legends will follow, in the next session.

# Changing plotting symbols

We also used `pch=19` − to obtain the same non-default plotting symbol, a filled circle.

The full range;

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
○ △ + × ◇ ▽ ⊠ ✳ ⬖ ⊕ ⬙ ⊞ ⊠ ◿ ■ ● ▲ ◆ ● ● ● ■ ◆ ▲ ▼

- Set the fill color for `21:25` with the `bg` argument
- The open circle (`pch=1`) is the default − because it makes it easiest to see points that nearly overlap.  Change it only if you have a good reason to
- Filled symbols `15:20` work well with transparent colors, e.g. `col="#FF000033"` for translucent pink

For different size symbols, there is a `cex` option; `cex=1` is standard size, `cex=1.5` is 50% bigger, etc.

But beware!  These options should be used *sparingly*...
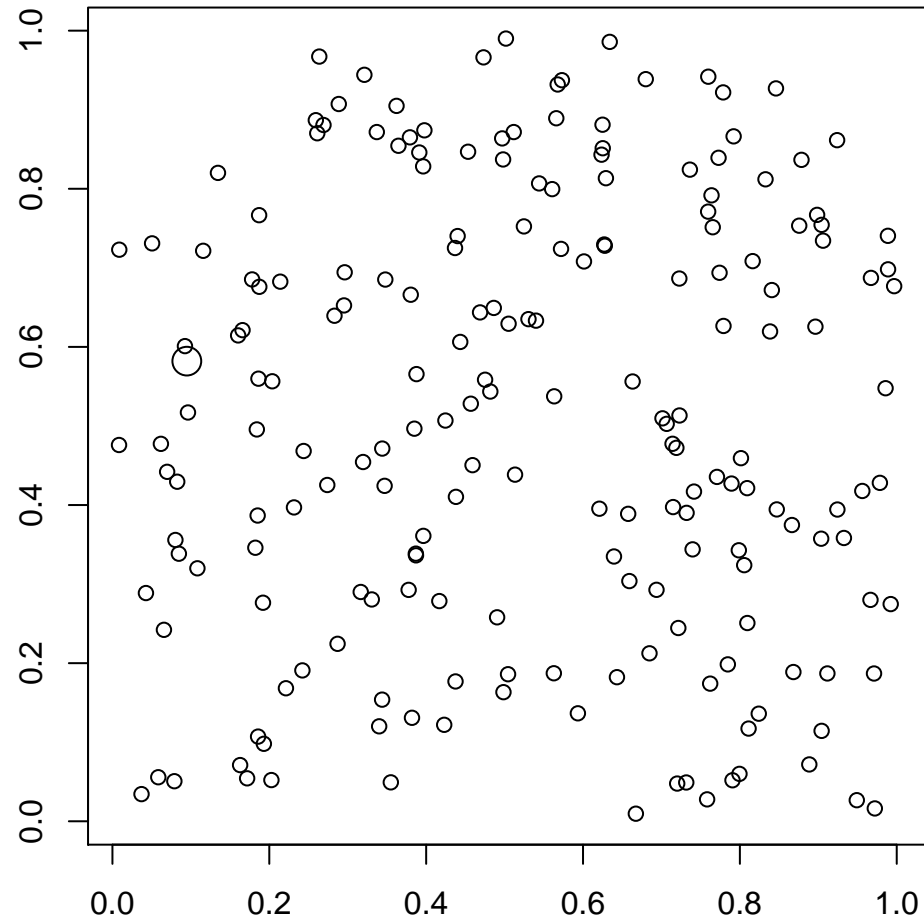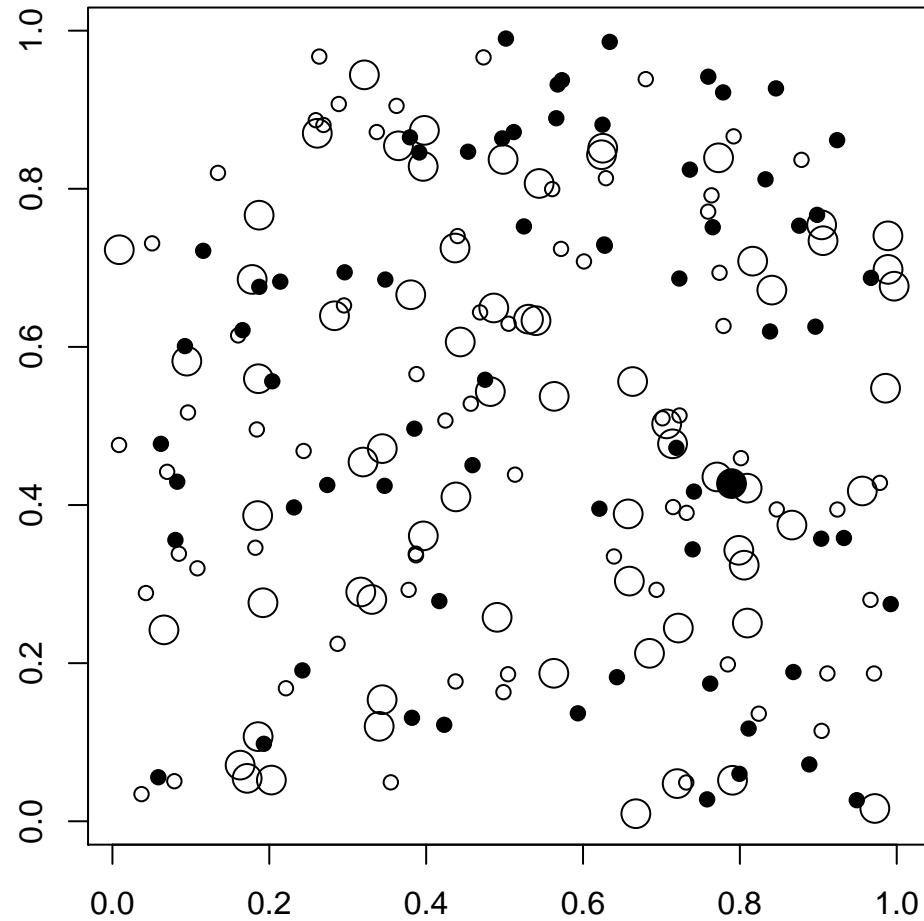
# Changing plotting symbols

One of these points is not like the others...

# Changing plotting symbols

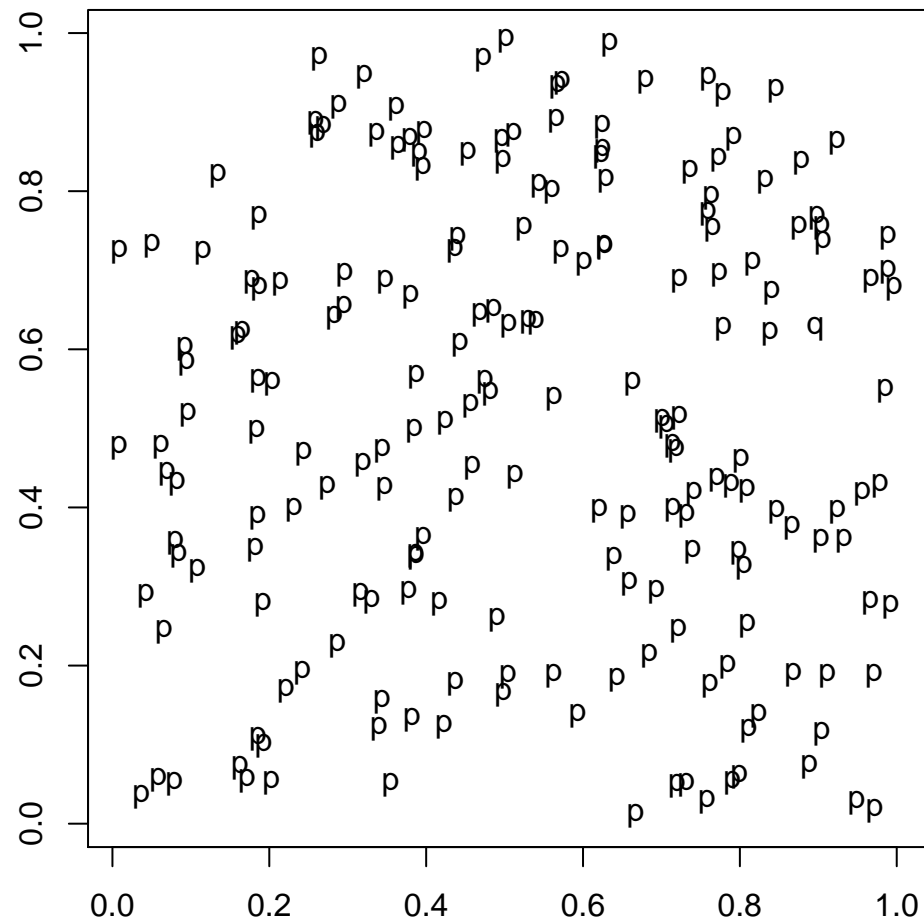One of these points is not like the others...

# Changing plotting symbols

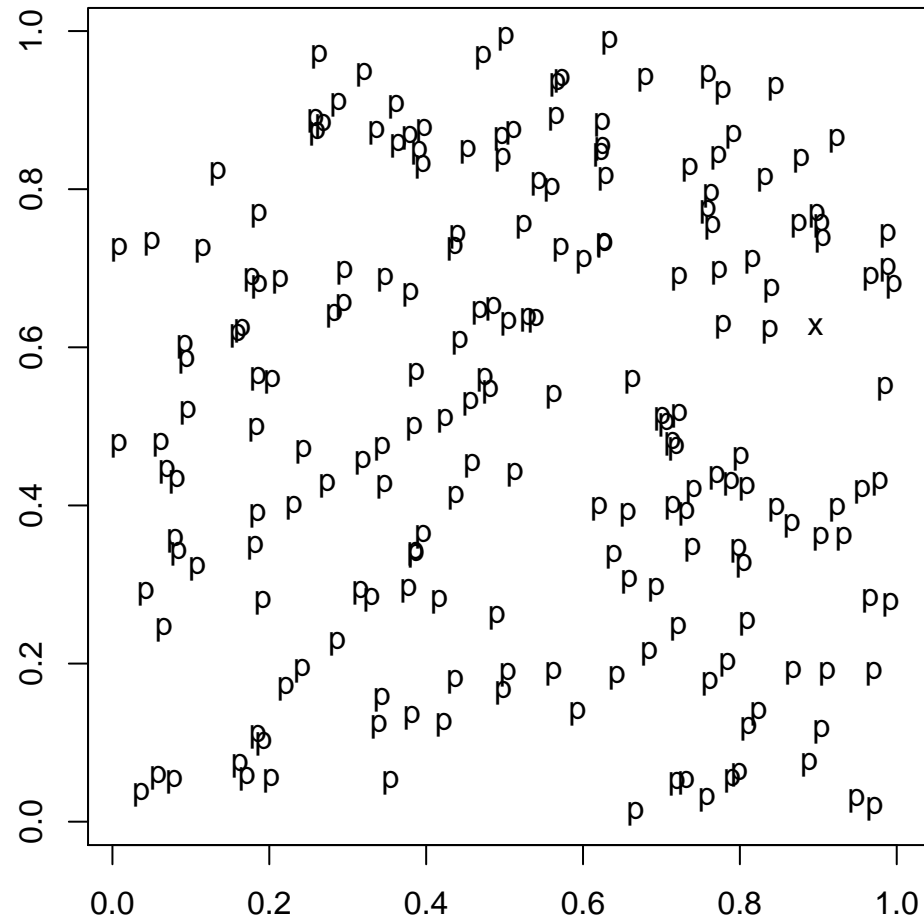One of these points is not like the others...

# Changing plotting symbols

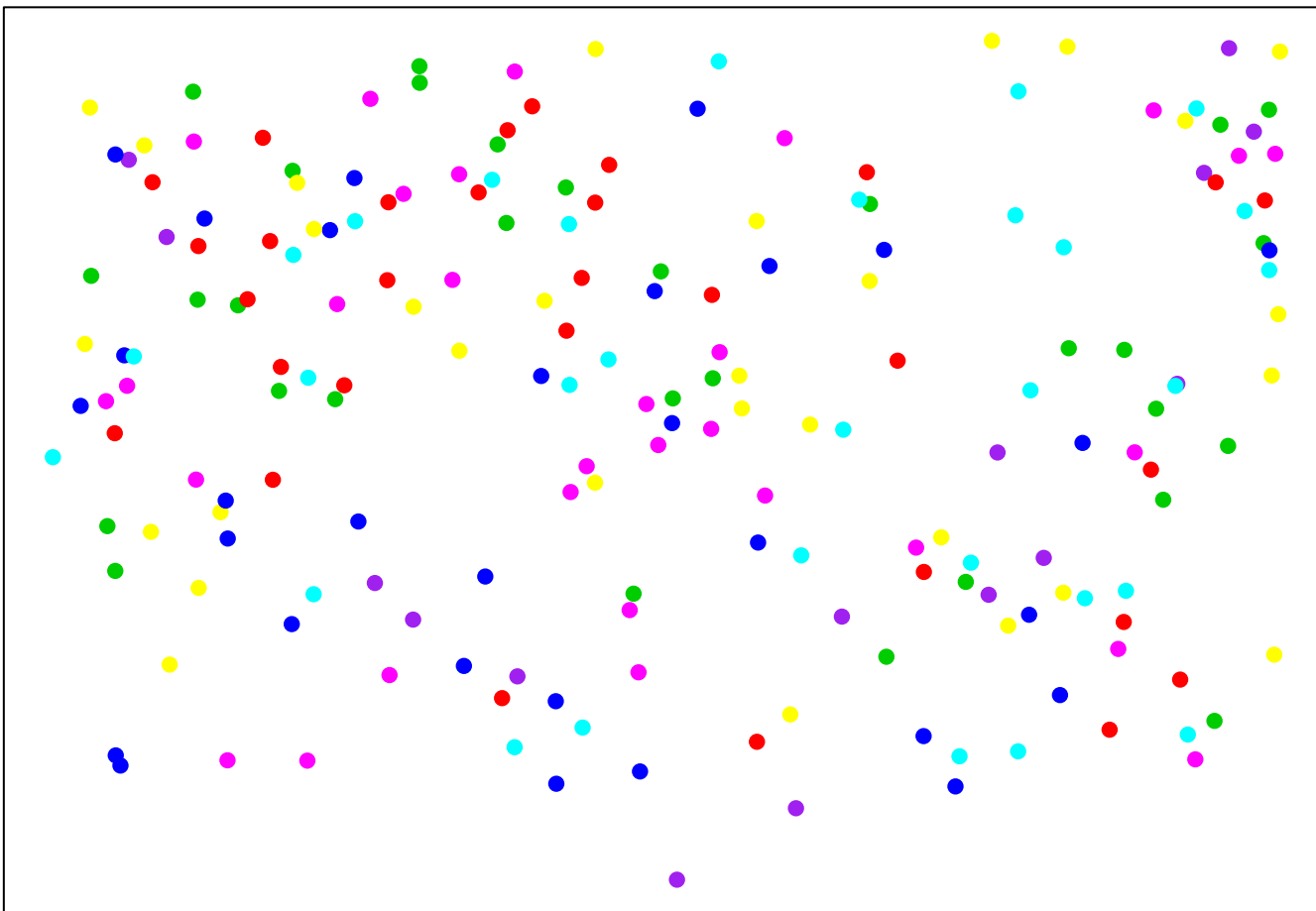One of these points is not like the others... (`pch="p"`)

# Changing plotting symbols

One of these points is not like the others... (`pch="p"`)
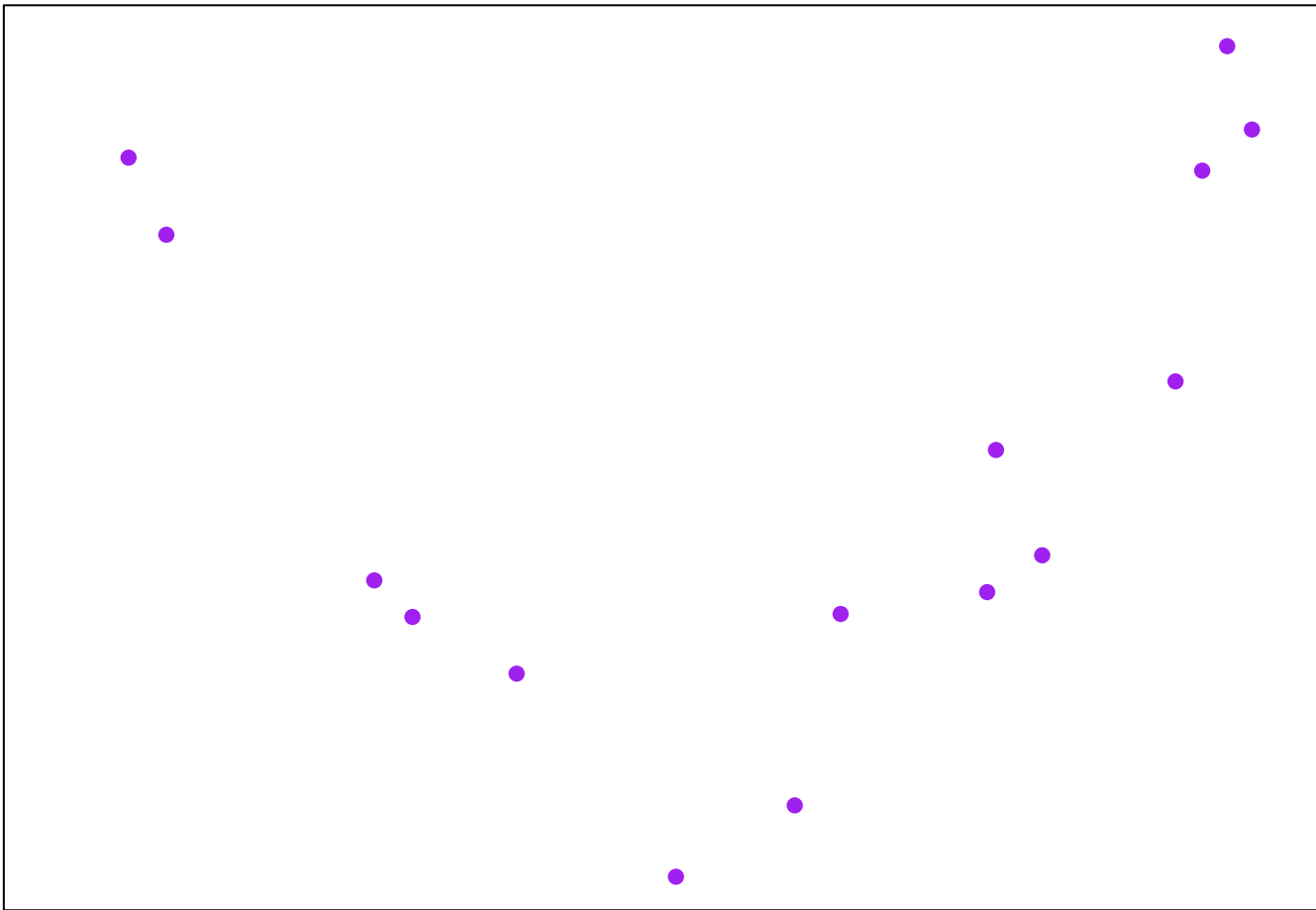
# Changing plotting symbols

Too many colors ($> 4$, say) requires too much attention; what pattern is illustrated here?
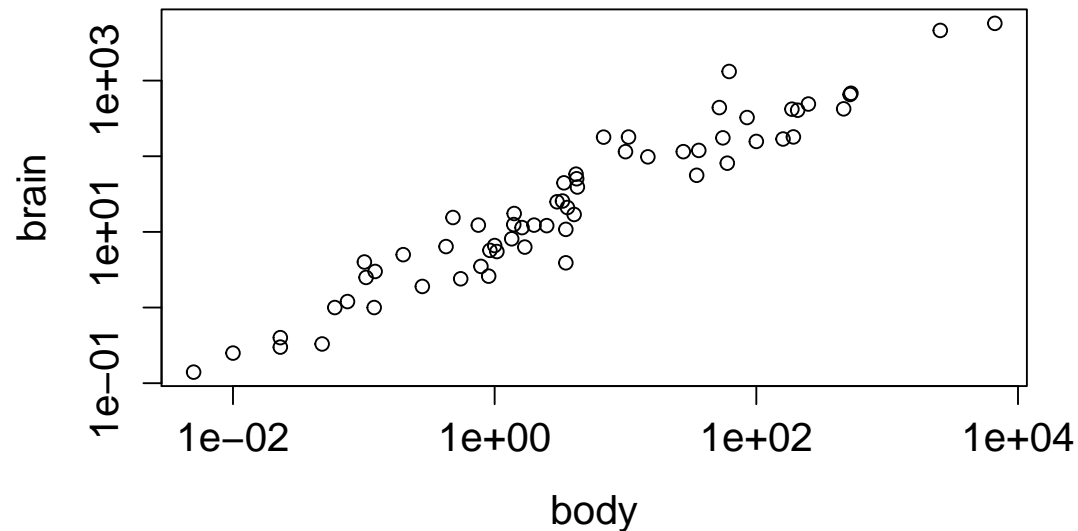
# Changing plotting symbols

Too many colors ($> 4$, say) requires too much attention; what pattern is illustrated here?

# Plots via the formula syntax

To make plots, we've used arguments `x` (on the X-axis) and `y` (on the Y-axis). But another method makes a stronger connection to *why* we're making the plot;

```
plot(brain~body, data=mammals, log="xy")
```


"Plot how `brain` *depends on* `body`, using the `mammals` dataset, with logarithmic x and y axes"

Plots via the formula syntax

A few more examples, using the salary dataset;

```
plot(salary~year, data=salary) # scatterplot
plot(salary~rank, data=salary) # boxplot
plot(rank~field,  data=salary) # stacked barplot
```

In all of these, $Y \sim X$ can be interpreted as Y *depends on* X — the 'tilde' symbol is R's shorthand for 'depends on'.

Statisticians (and scientists) like to think this way;

- How does some outcome (Y) depend on a covariate (X)? (a.k.a. a predictor)
- How does a dependent variable (Y) depend on an independent variable (X)?

And how does Y depend on X in observations with the same Z?

Plots via the formula syntax

To help us illustrate how scientists think, a bit of science;

Ozone is a *secondary pollutant*, produced from organic compounds and atmostpheric oxygen, in reactions catalyzed by nitrogen oxides and powered by sunlight. But for ozone concentrations in NY in summer (Y) a smoother (code later) shows a non-monotone relationship with sunlight (X) …
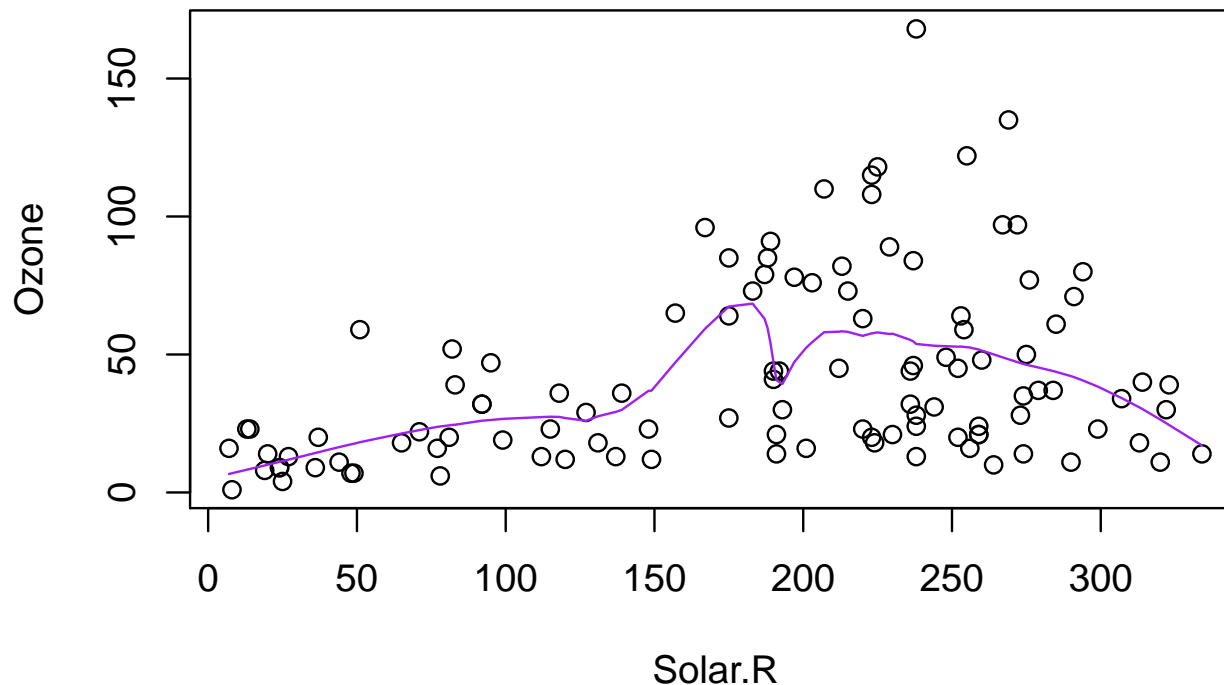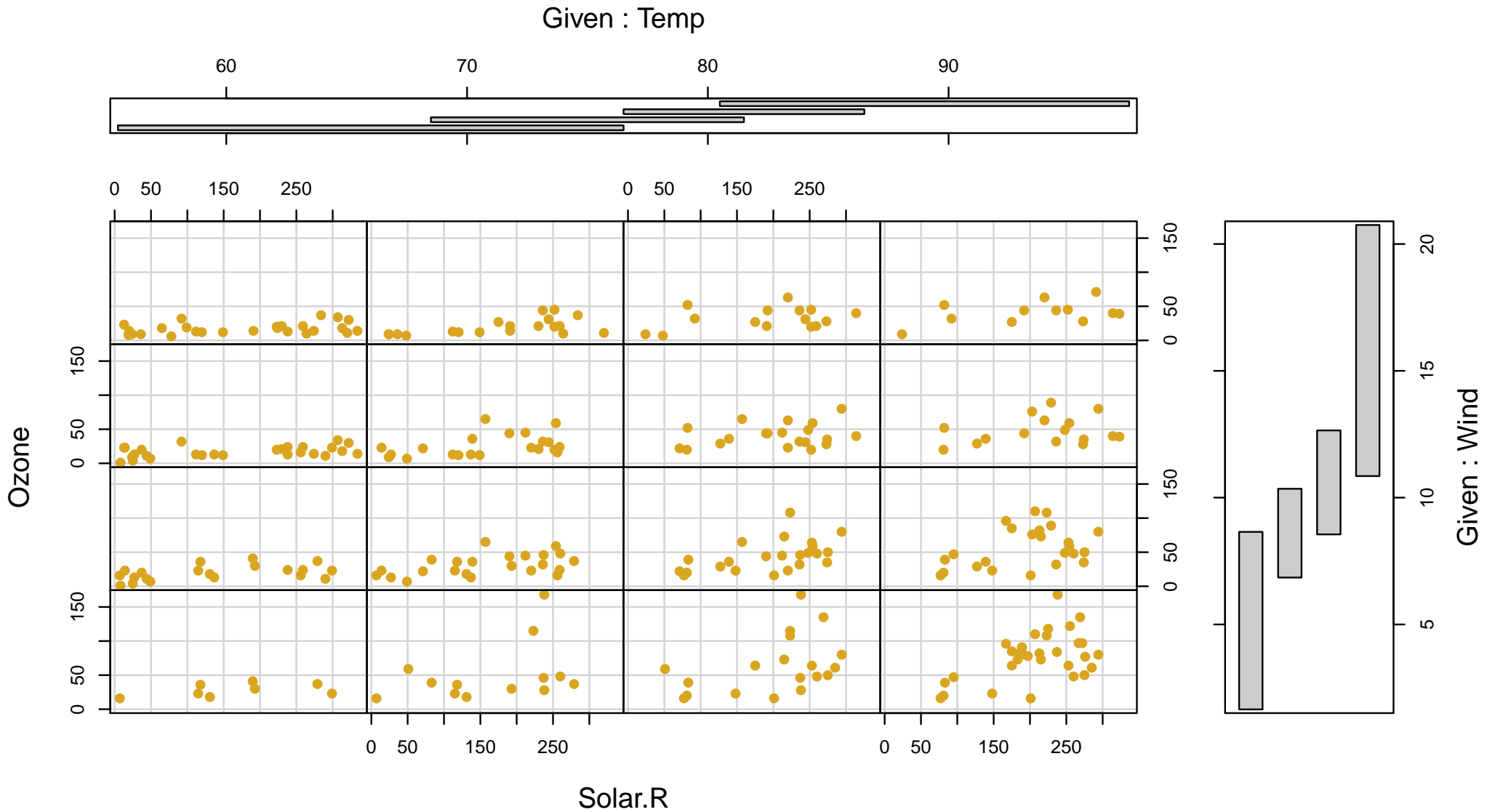
Plots via the formula syntax

Now draw a scatterplot of `Ozone` vs `Solar.R` for various subranges of `Temp` and `Wind`.

```
data("airquality") # using a dataset supplied with R
coplot(Ozone ~ Solar.R | Temp + Wind, number = c(4, 4),
       data = airquality,
       pch = 21, col = "goldenrod", bg = "goldenrod")
```

- The vertical dash ("|") means 'given particular values of', i.e. 'conditional on'
- Here, "+" means 'and', not 'plus' − see `?formula`, and later sessions
- How does Ozone depend on Solar Radiation, on days with (roughly) the same Temperature *and* Wind Speed?
- ...using the `airquality` data, with a 4 × 4 layout, with solid dark yellow circular symbols

Plots via the formula syntax

Plots via the formula syntax

What does this show?

- A 4-D relationship is illustrated; the Ozone/sunlight relationship changes in strength depending on both the Temperature and Wind
- 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
- Almost any form of plot can be 'conditioned' in this way − but the commands are in the non-default `lattice` package

NB it is possible to produce 'fake 3D' plots in R − but (on 2D paper) conditioning plots work better!

Summary

- R makes publication-quality graphics, as well as graphics for data exploration and summary

- `plot()` is generic, and adapts to what you give it. There are (necessarily) lots of arguments to consider; colors, plotting symbols, labels, etc

- `hist()`, `boxplot()`, `dotplot()` and `coplot()` offer more functionality

- The formula syntax is a (more) natural way from translate scientific aims to choice of what to plot

- Much more to come! In the next section we'll build up more complex plots