7. Fitting models

Ken Rice
Thomas Lumley

Universities of Washington and Auckland

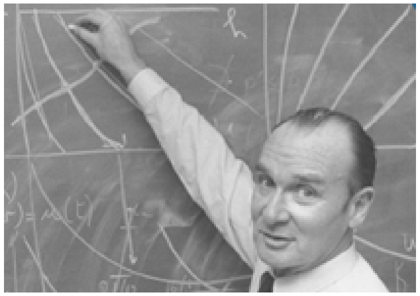
NYU Abu Dhabi, January 2017

Disclaimer/Warning

In statistics, as in fashion, a model is an idealization of reality.

Peter McCullagh
JRSSD (1999) 48:1

Models basically play the same role in economics as in fashion: they provide an articulated frame on which to show off your material to advantage ...; a useful role, but fraught with the dangers that the designer may get carried away by his personal inclination for the model, while the customers may forget that the model is more streamlined than reality.



Jacques Drèze
Economic Journal (1985) 95:380

In this session

... we will not attempt to teach `all` of statistical modeling. Instead, we'll cover;

- More about the formula syntax $(Y \sim X)$, and some functions that use it to fit models
- *Some* explanation of what these functions are doing, and why it might be useful
- Some 'helper' functions, used when fitting models

Example: the t-test

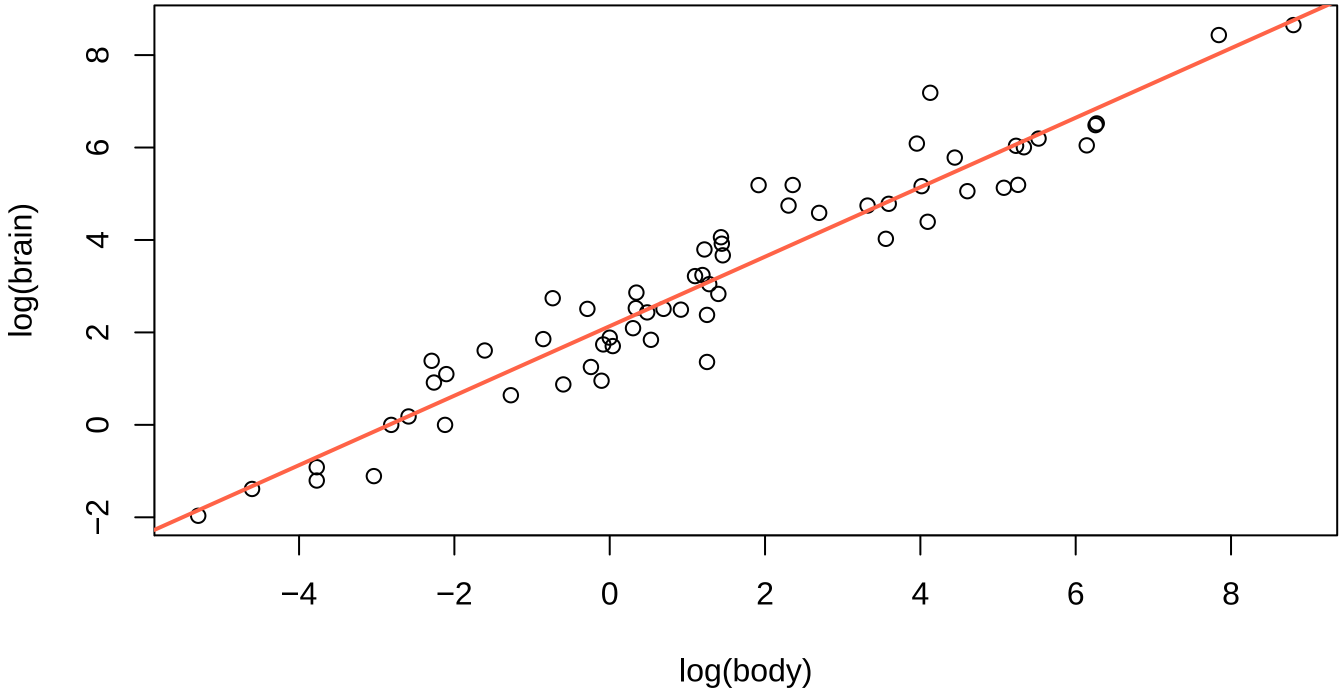
Recall the `sleep` example from Session 5. We want to compare mean levels of extra sleep, in Group 1 and 2. The full version of the code and output;

```
> t.test(extra˜group, data=sleep)
    Welch Two Sample t-test
data:  extra by group
t = -1.8608, df = 17.776, p-value = 0.07939
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -3.3654832  0.2054832
sample estimates:
mean in group 1 mean in group 2
           0.75            2.33
```

- `extra` is the outcome, it depends on `group` – for an analogous graphical comparison use `plot(extra∼group, data=sleep)`
- Confidence interval is for difference in means
- p-value: null hypothesis is of equal means (2-sided test)

Example: linear regression

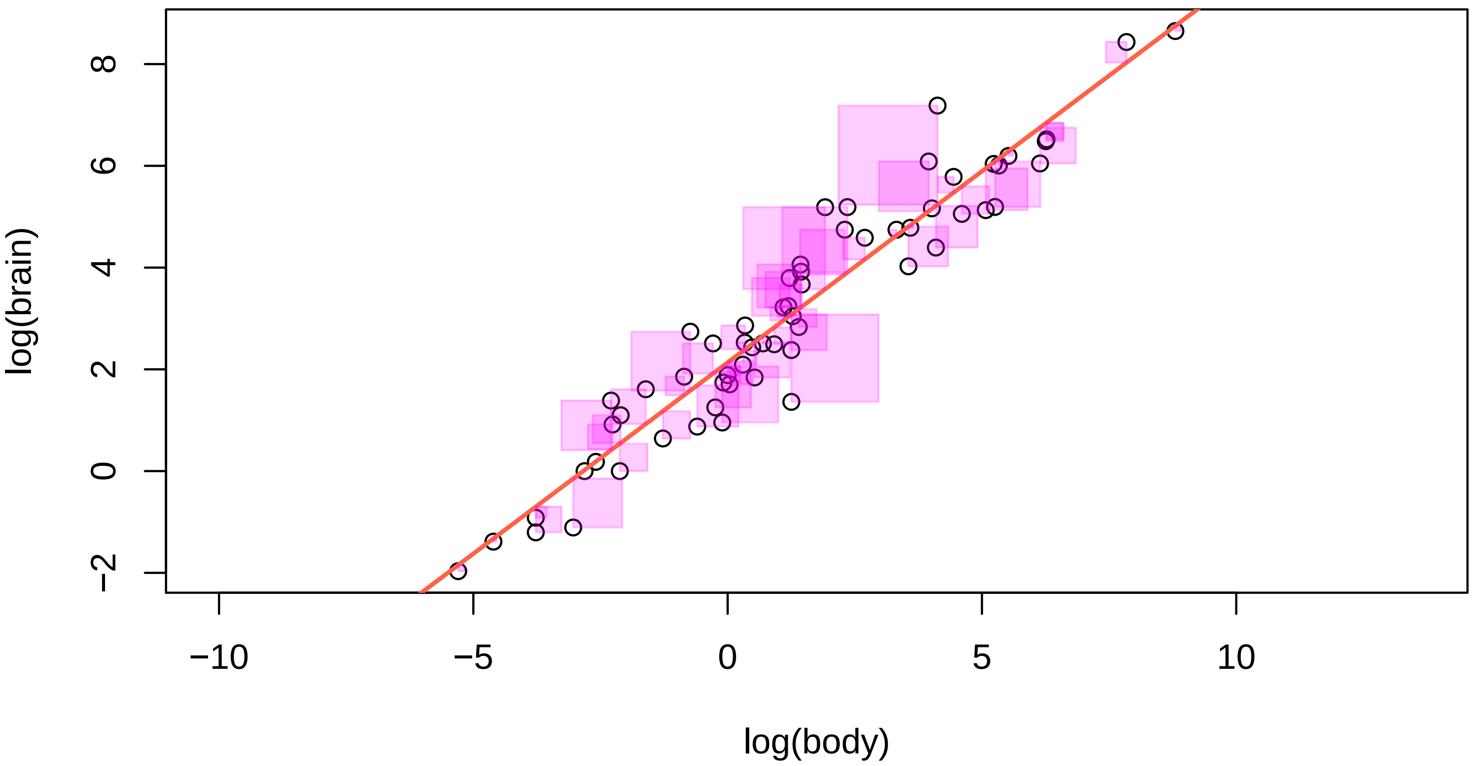
Another favorite example;



Straight line suggests `log(brain)` higher by ≈0.75 units, per 1-unit difference in `log(body)` – i.e. a power law, brain $\propto$ body$^{0.75}$.

Example: linear regression

Where does the straight line come from? One way* to justify it is as the *least squares* fit;



Any other choice of line would use more purple ink.

* there are several – too many to discuss here!

Example: linear regression

Finding the least-squares fit is known as ‘simple’ *linear regression*, or fitting a *linear model*. In R;

```
> mammals.reg <- lm(log(brain)˜log(body), data=mammals)
> summary(mammals.reg)

Call:
lm(formula = log(brain) ˜ log(body), data = mammals)

Residuals:
     Min       1Q   Median       3Q      Max
-1.71550 -0.49228 -0.06162  0.43597  1.94829

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.13479    0.09604   22.23   <2e-16 ***
log(body)    0.75169    0.02846   26.41   <2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 0.6943 on 60 degrees of freedom
Multiple R-squared:  0.9208,Adjusted R-squared:  0.9195
F-statistic: 697.4 on 1 and 60 DF,  p-value: < 2.2e-16
```

Example: linear regression

The `summary()` is fairly verbose; (SAS is a lot worse!)

- Function `lm()` make an `lm.object`, containing the output of the regression; try `str(mammals.reg)` to see that `summary()` picks out the most important bits
- `Call` restates the formula, `Residuals` summarizes how small our 'least' square edges are
- `Coefficient`; the fitted line is

  $$\log(\texttt{brain}) = 2.13 + 0.75 \times \log(\texttt{body})$$

  The intercept (2.13) is (sensibly) added by default.
- `Std. Error` describes the noise in each estimate – smaller when you have more data
- `Pr(>|t|)` is a two sided p-value, for the null hypothesis that the relevant coefficient is zero
- Other terms describe remaining 'noise'

Example: linear regression

The next-most useful summary; (recall bootstrap intervals)

```
> confint(mammals.reg, parm="log(body)", level=0.95)
              2.5 %    97.5 %
log(body) 0.6947503 0.8086215
> confint(mammals.reg, parm=1:2, level=0.975)
               1.25 %  98.75 %
(Intercept) 1.9139805 2.355597
log(body)   0.6862469 0.817125
```

- For `lm.objects`, `confint()` gives intervals based on point estimate ± `Std. Error` × the appropriate quantile of the appropriate t distribution
- `confint.default()` uses Normal quantiles instead
- `level` is the confidence level, default is 95%
- `parm` can be a vector of coefficient names, or a vector of numbers; the default gives intervals for all terms
- Like most software, R gives an insane number of decimal places – in the write-up round std errs to 2 significant figures, using `signif()`, then `round()` estimates/CIs to this precision

Example: linear regression

To ‘extract’ other parts of an `lm.object`, you can use the `$` (apostrophe-S) symbol, e.g. `mammals.reg$coef` is the point estimates. But R’s regression functions also have generic *extractor* functions, helpful for common jobs;

- `coef(mammals.reg)` – gives the fitted coefficients
- `fitted(mammals.reg)` returns the fitted `log(brain)` values (i.e. mean Y), for each data point (i.e. each X)
- `residuals(mammals.reg)` returns `log(body)` minus the fitted value – that we minimized the sum of, when squared
- `predict(mammals.reg, new.data.frame)` predicts the mean `log(brain)` (i.e. Y) for which you supply `log(body)`

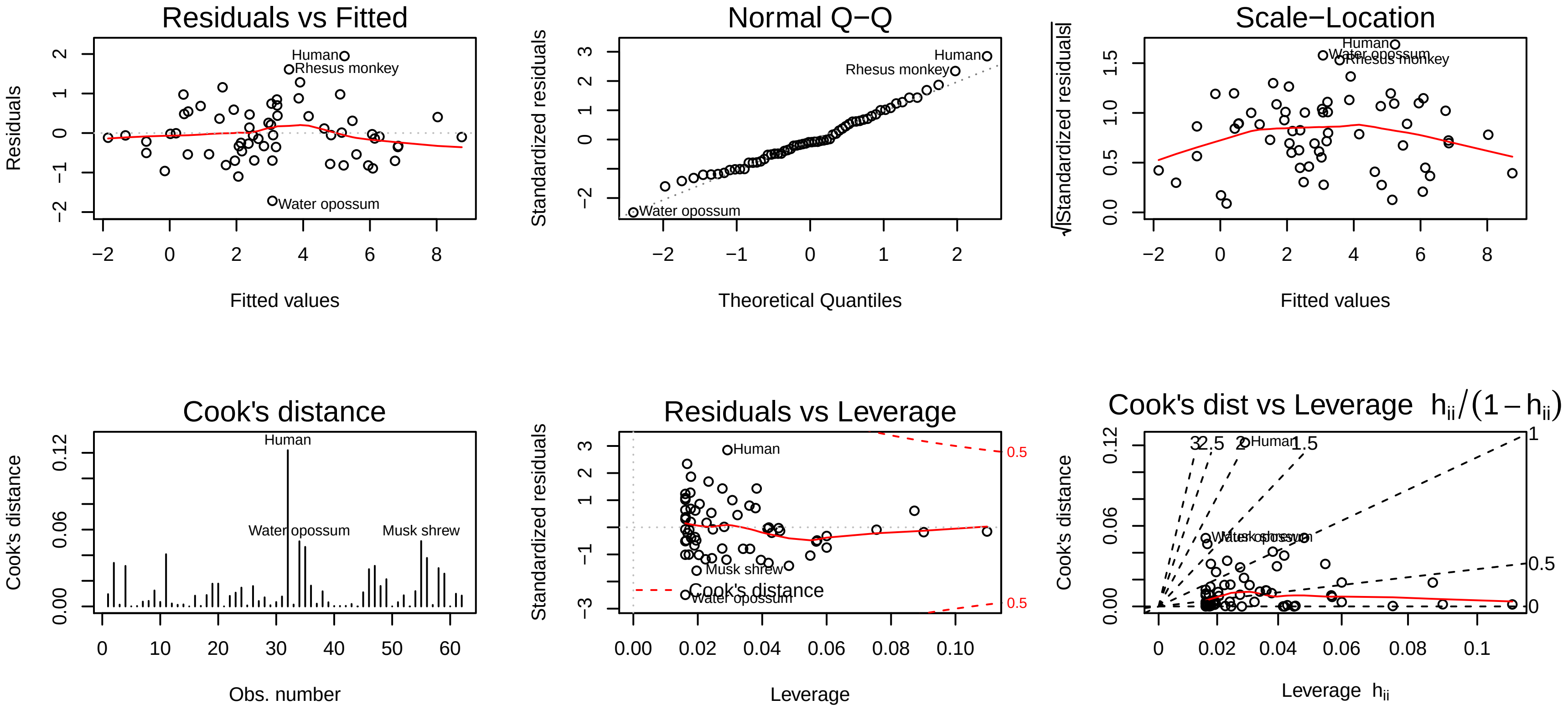
Experts: `vcov()` gives the variance-covariance matrix, describing the statistical noise in the coefficients; `sqrt(diag(vcov(mammals.reg)))` is the same as `Std. Error` column in `summary()` output.

For more of these (some fairly esoteric) use `methods(class="lm")`.

Example: linear regression

Experts again: `plot()` has a method for `lm.object`s;

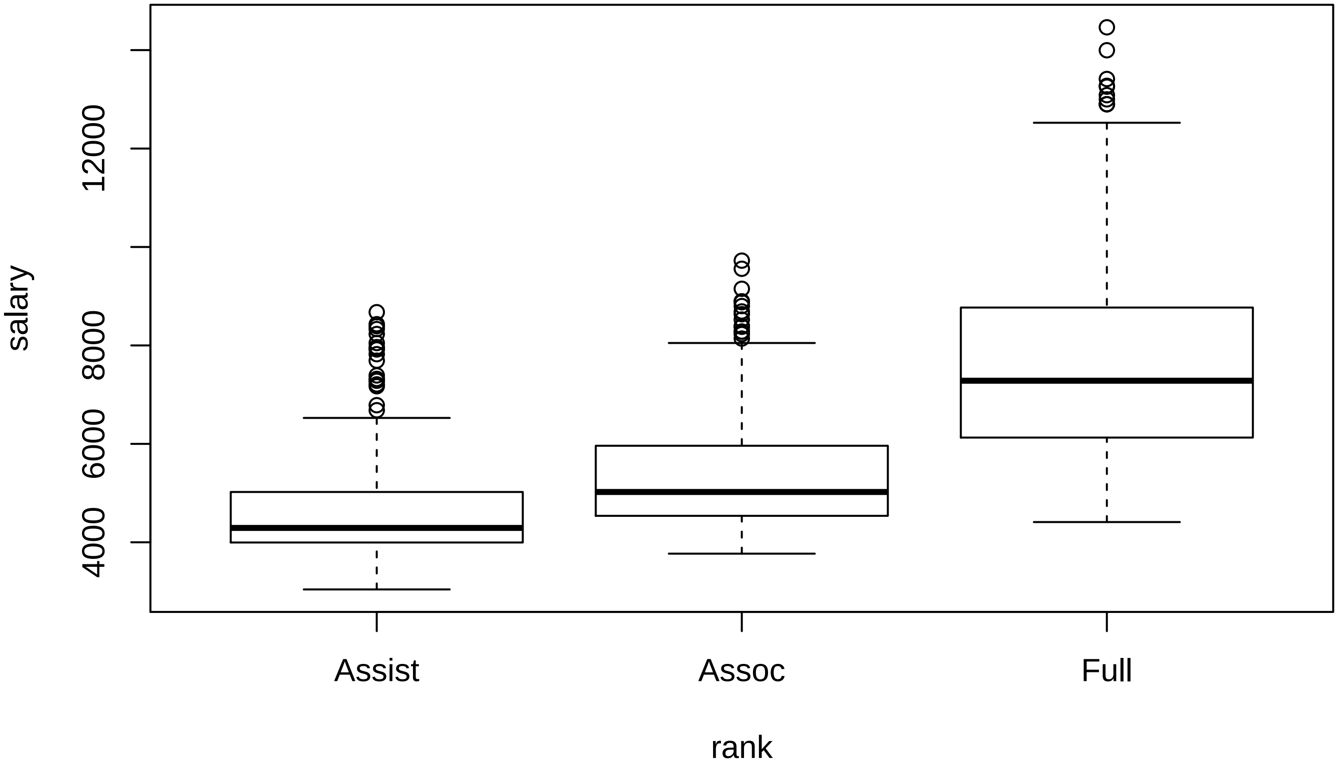
```
par(mfrow=c(2,3))
plot(mammals.reg, which=1:6)
```



Example: salaries again

Another familiar example; how does salary depend on rank?

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



As a regression, we could ask whether the mean salary is different at different ranks. NB With one *final* salary per person, it's reasonable to assume independent observations.

Example: salaries again

For independent outcomes, comparison of means is exactly what ‘analysis of variance’ does ...despite the name!

```
> salary.aov <- aov(salary~rank, data=finalsalary)
> summary( salary.aov )
              Df    Sum Sq   Mean Sq F value Pr(>F)
rank           2 2.642e+09 1.321e+09   529.6 <2e-16 ***
Residuals   1593 3.974e+09 2.495e+06
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
1 observation deleted due to missingness
> model.tables(salary.aov, type="means")
Tables of means
Grand mean
         6391.161
 rank
    Assist Assoc Full
      4650  5335 7584
rep    314   437  845
> table(finalsalary$rank)
Assist  Assoc    Full
   315     437     845       # spot the difference
```

Example: salaries again

‘Under the hood’, `aov()` runs group-specific linear regressions with just an intercept (`salary`$\sim$`1`, in the formula syntax) and recombines them. Here, least-squares $\equiv$ take each group’s mean.

A simpler approach* uses regression directly; (edited output)

```
> salary.lm <- lm(salary~rank, data=finalsalary)
> summary(salary.lm)
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  4650.43      89.13  52.173  < 2e-16 ***
rankAssoc     684.31     116.85   5.856 5.74e-09 ***
rankFull     2933.93     104.39  28.106  < 2e-16 ***
  (1 observation deleted due to missingness)
F-statistic: 529.6 on 2 and 1593 DF,  p-value: < 2.2e-16
```

- Same F statistic, and p-value, *equivalent* point estimates
- `Intercept` describes mean salary in Assist Profs (again)
- Other coefficients describe *differences* – so e.g. $p = 5.74 \times 10^{-9}$ is for testing Assist=Assoc. `Assist` is ‘reference’ level

* preferred by most statisticians, though not all

Multiple regression

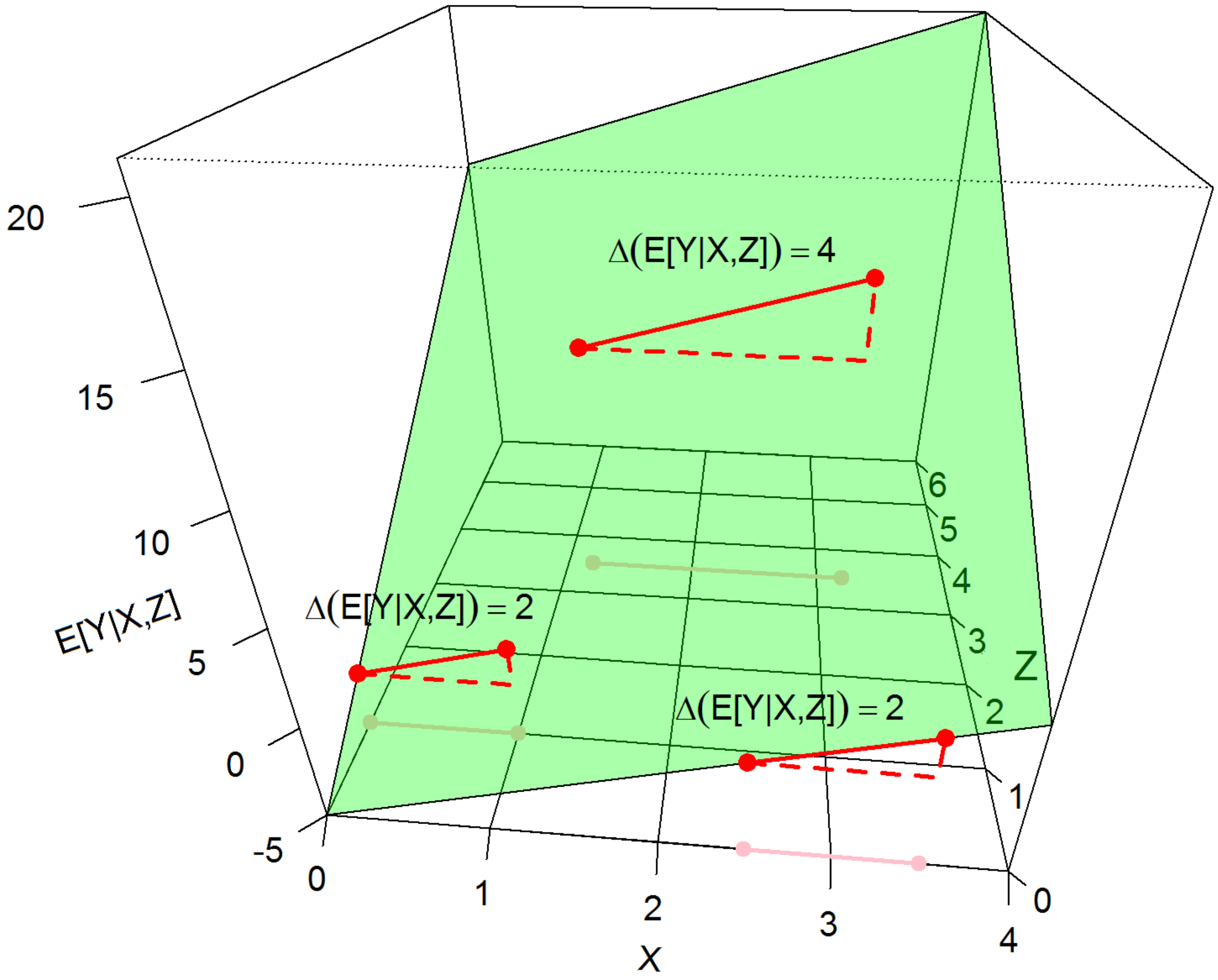
Say you wanted to know how salary depended on start year at UW, and on year of final degree ($\approx$ age, here)

```
> mreg <- lm(salary~ yrdeg + startyr, data=finalsalary)
> summary(mreg)
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 13583.596    375.936  36.133  < 2e-16 ***
yrdeg        -118.455      7.380 -16.051  < 2e-16 ***
startyr        22.438      7.275   3.084  0.00208 **
```

- Starting later is associated with greater salary (+22.44) in people with same year of degree
- Getting a degree earlier associated with less salary (-118.46) in those who started in same year
- In the formula, '+' means 'and'. To regress on multiple covariates, use `y` $\sim$ `x + z + u + v + ...`
- To use 'plus' in a formula (or minus) `I()` is for insulate; `y` $\sim$ `x + I(z + u)` regresses Y on X and the sum of $Z + U$

Multiple regression

Regressing Y on X and Z fits a plane;



Multiple regression

To test hypotheses involving more than one parameter at a time, use `anova()` to compare the fitted models with and without those parameters;

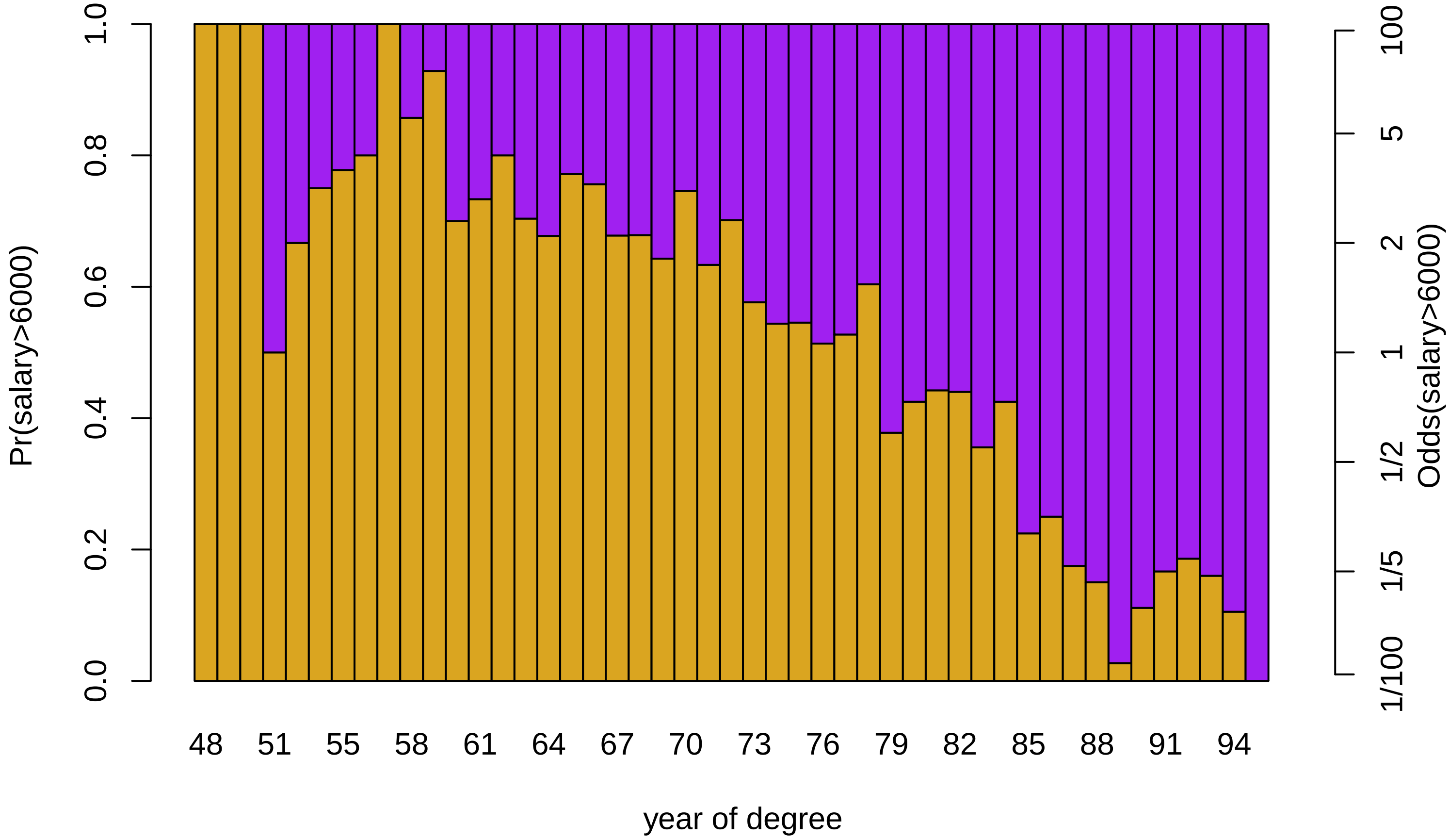
```
> mreg  <- lm(salary~ yrdeg + startyr, data=finalsalary)
> mreg0 <- lm(salary~ yrdeg + startyr + rank, data=finalsalary)
> anova(mreg, mreg0)
Analysis of Variance Table

Model 1: salary ~ yrdeg + startyr
Model 2: salary ~ yrdeg + startyr + rank
  Res.Df        RSS Df  Sum of Sq      F    Pr(>F)
1   1593 5025366914
2   1591 3834775234  2 1190591680 246.98 < 2.2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

- Here testing any difference between ranks, adjusted for the other two variables – order doesn't matter
- Not the same as `aov()`!
- With only one model, `anova()` tests each coefficient, in order of appearance – order *does* matter

Logistic regression

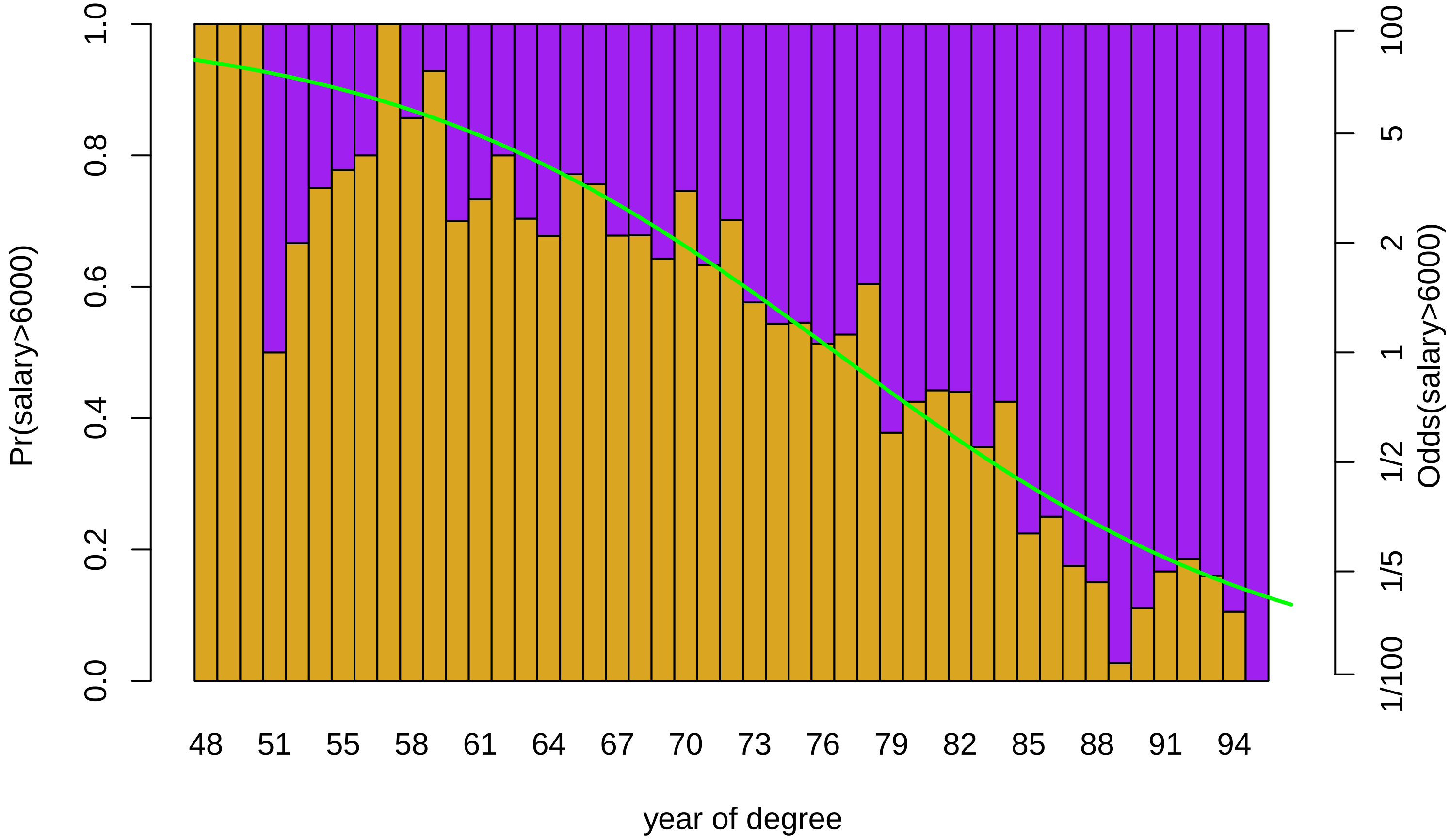
When Y is binary (e.g. 1/0, yes/no, dead/alive) the expected value of Y is the probability that $Y = 1$.



Linear regression's straight line might give a poor summary.

Logistic regression

Instead of a straight line, *logistic regression* fits a curve through the data;



The fitted odds shrink by ≈10%, for each extra year.

Logistic regression

The `glm()` command does this (close relative of `lm()`)

```
> glm1 <- glm(salary>6000 ~ yrdeg, data=finalsalary, family=binomial)
> summary(glm1)
Call:
glm(formula = salary > 6000 ~ yrdeg, family = binomial, data = finalsalary)
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2923  -0.9342  -0.5215   0.9674   1.9871
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  7.743524   0.490335   15.79   <2e-16 ***
yrdeg       -0.101791   0.006403  -15.90   <2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
(Dispersion parameter for binomial family taken to be 1)
    Null deviance: 2212.5  on 1595  degrees of freedom
Residual deviance: 1895.9  on 1594  degrees of freedom
  (1 observation deleted due to missingness)
AIC: 1899.9
Number of Fisher Scoring iterations: 4
```

NB turn those #&*%ing stars off! `options(show.signif.stars=FALSE)`

Logistic regression

The coefficients here are *log* odds (for the Intercept) and *log* odds ratios. So, for a confidence interval around the '10% smaller' result;

```
> confint(glm1, "yrdeg", level=0.95)
Waiting for profiling to be done...
      2.5 %      97.5 %
-0.11454628 -0.08943648
> confint.default(glm1, "yrdeg", level=0.95)
           2.5 %     97.5 %
yrdeg -0.1143396 -0.0892416
> round(exp(confint.default(glm1, "yrdeg", level=0.95)), 3)
      2.5 % 97.5 %
yrdeg 0.892  0.915
```

- The default is fairly sophisticated; for typical symmetric intervals use `confint.default()`
- ... then exponentiate to get interval for the odds ratio

All the extractor functions we saw before are available – and use the formula syntax to regress on multiple covariates.

Other regressions, other tests

In glm(), other family arguments provide other forms of regressions – too many for our course. Some other tests;

```
> tab1 <- with(droplevels(subset(finalsalary, yrdeg>87 & rank!="Full")),
+              table( salary>6000, rank) )
> tab1
       rank
        Assist Assoc
  FALSE    199    24
  TRUE      25     8
> chisq.test( tab1 )
Pearson's Chi-squared test with Yates' continuity correction
X-squared = 3.6229, df = 1, p-value = 0.05699
Warning message:
In chisq.test(tab1) : Chi-squared approximation may be incorrect
> fisher.test(tab1)
Fisher's Exact Test for Count Data
p-value = 0.04413
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
 0.9243447 6.9357450
sample estimates:
odds ratio
  2.640338
```

Summary

- There are R implementations of almost every regression method
- Most use the formula syntax, also used for plotting – naturally, because both describe how outcome Y depends on some covariates
- The default in `lm()` and `glm()` is to drop cases with NAs – without warning you
- Extractor functions save time, and make code easier to read

There are many more regression methods available, beyond what ‘plain vanilla’ R provides – in the next session we’ll discuss use of ‘packages’, to extend R.