Class 2: Regression and classification

Andrew Parnell, School of Mathematics and Statistics,
University College Dublin

Learning outcomes

- Be able to understand the structure of regression and classification models
- Know how to read and interpret the output of a statistical model
- Be familiar with some of the extensions to basic regression and classification models

Why regression and classification?

- t-tests are only really useful when you have a continuous outcome variable and one discrete variable with two groups (e.g. treatment vs control)
- For almost any real life situation you have multiple variables of all different types
- For these situations you need a *statistical model*
- A statistical model allows to perform *probabilistic prediction* of the outcome variable from the remaining variable, and/or to explain how the other variables are causing the outcome variable to change

Regression vs Classification: what's the difference?

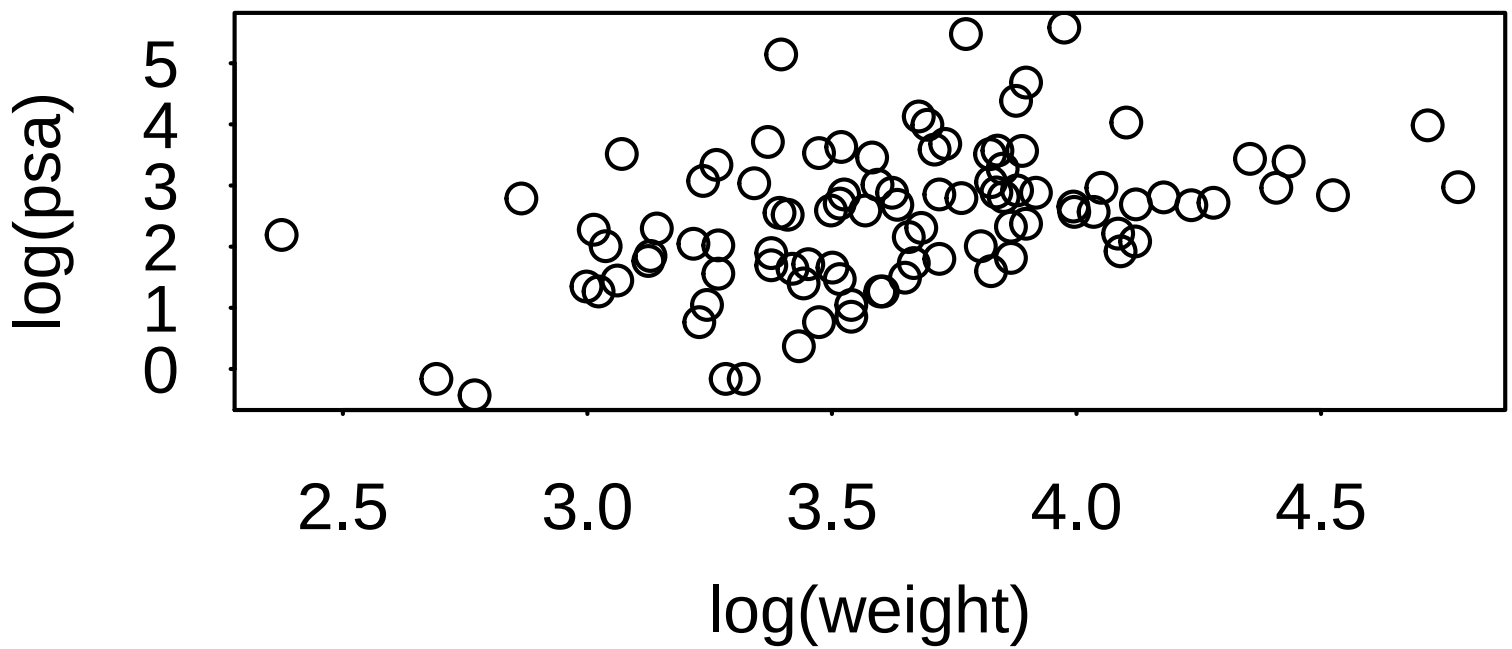
- In regression we have a single *continuous* outcome variable and lots of other variables which we think might be causing the outcome to change
- In classification we have a single *discrete* outcome variable and lots of other variables
- In the machine learning literature this is often known as *supervised learning*
- Situations where there are multiple outcome variables are beyond the scope of this course

Response and explanatory variables

- The outcome variable is more commonly known as the *response* variable
- The other variables which we think might be causing the response variable to change are called the *explanatory variables* (though be careful with causation)
- We will use these words from now on, but beware there are lots of other terms in the literature

A basic regression model

- Let's go back to the prostate cancer data
- Recall the key outcome variable is `lpsa` the log of the prostate specific antigen value. This is our response variable
- Suppose we had one explanatory variable `lweight`



Creating the model

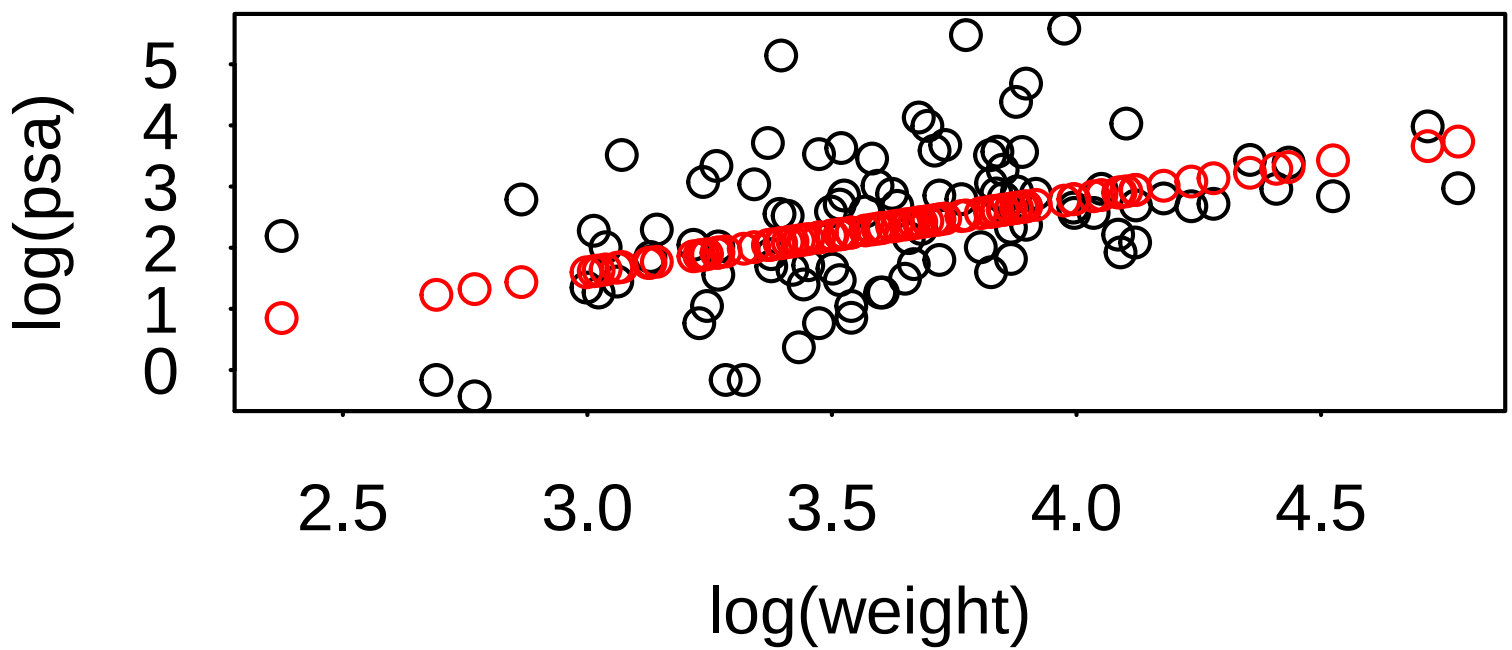
- Looking at the plot, there may be a positive, linear relationship between log(weight) and log(psa)
- Perhaps we can create a prediction model that allows us to predict log(psa) from log(weight)
- Suppose, for each patient we multiplied the log(weight) value by 1.2 and then subtracted the value 2 so:

$$prediction = 1.2 \times \log(weight) - 2$$

- If we do this repeatedly for every value in the data set we get ...

A first model

```
par(mar=c(3,3,2,1), mgp=c(2,.7,0), tck=-.01,las=1)
prediction = 1.2 * prostate$lweight - 2
plot(prostate$lweight, prostate$lpsa, xlab = 'log(weight)'
points(prostate$lweight, prediction, col='red')
```



Refining the model

- Is this model any good?
- How might we measure how close our predictions are to the truth?
- How can we choose the values (here 1.2 and -2) better?

Getting R to do the work

- Luckily the R function `lm` will do the work for us

```
model_1 = lm(formula = lpsa ~ lweight, data = prostate)
summary(model_1)
```

```
##
## Call:
## lm(formula = lpsa ~ lweight, data = prostate)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.27976 -0.67507 -0.03503  0.53984  2.93649
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.7586     0.9103  -1.932   0.0564 .
## lweight       1.1676     0.2491   4.686 9.28e-06 ***
## ---
```

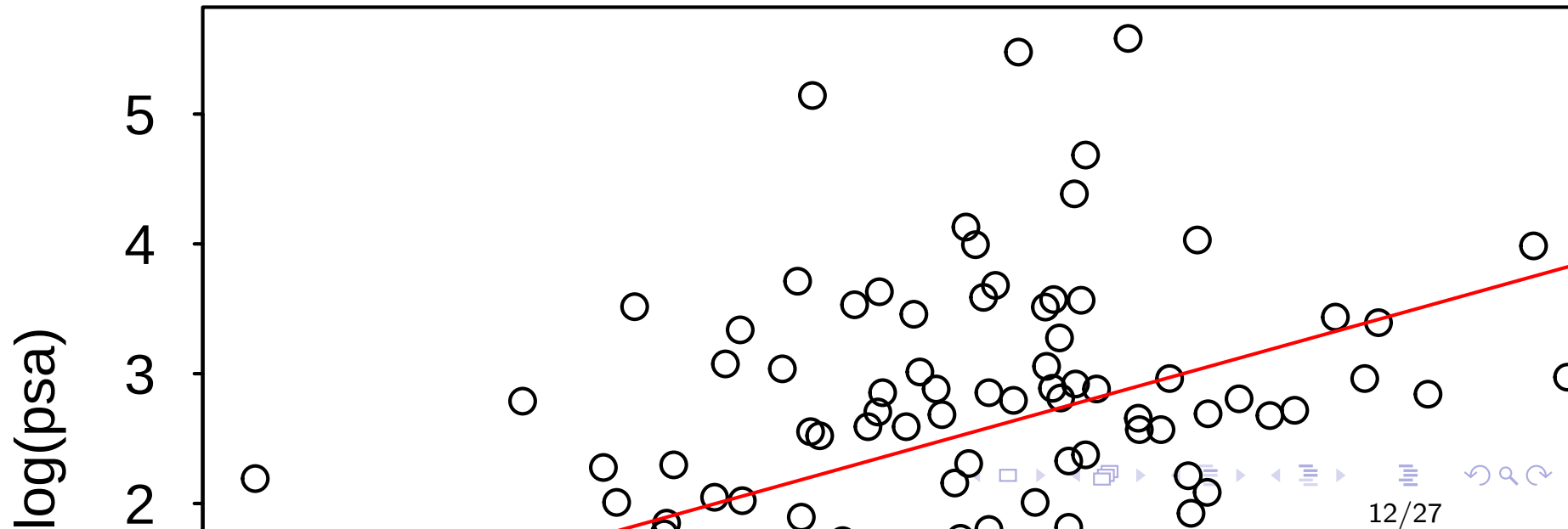
Background details

- The two values here are the y-intercept and the slope of the line
- R chooses these values by minimising the vertical distances between the black and the red points (called least squares)
- A key assumption in the model is that these vertical distances (known as *residuals*) are normally distributed
- R uses this assumption to run t-tests on the parameters, which you can see the results of in the `summary` output

Plotting the fit

- One way is to type `plot(model_1)` which gives residual diagnostics. A quick plot of the fitted line:

```
par(mar=c(3,3,2,1), mgp=c(2,.7,0), tck=-.01,las=1)
plot(prostate$lweight, prostate$lpsa, xlab = 'log(weight)',
abline(model_1, col='red')
```



Expanding the model with two explanatory variables

- Suppose we wanted to use two explanatory variables, `lweight` and age:

```
model_2 = lm(formula = lpsa ~ lweight + age, data = prostat
summary(model_2)
```

```
##
## Call:
## lm(formula = lpsa ~ lweight + age, data = prostate)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.2665 -0.6753 -0.0361  0.5317  2.9712
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.897709   1.119033  -1.696   0.0932 .
## lweight      1.147487   0.267094   4.296 4.23e-05 ***
```

Expanding the fit even more

```
model_3 = lm(formula = lpsa ~ . - train, data = prostate)
summary(model_3)
```

```
##
## Call:
## lm(formula = lpsa ~ . - train, data = prostate)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.76644 -0.35510 -0.00328  0.38087  1.55770
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.181561   1.320568   0.137  0.89096
## lcavol       0.564341   0.087833   6.425 6.55e-09 ***
## lweight      0.622020   0.200897   3.096  0.00263 **
## age         -0.021248   0.011084  -1.917  0.05848 .
## lbph         0.096713   0.057913   1.670  0.09848
```

Multiple regression

- When you have lots of explanatory variables this is known as *multiple regression*
- You can still use the values in the `Estimate` column to create predictions of `lpsa` by multiplying and adding up
- Beware the p-values as before: they might be highly dsignificant but still a very poor model
- R gives you two other useful statistics:
  - The R-squared which measures the proportion of variation in the repsonse variable explained by the explanatory variables
  - The residual standard error which measures how far away the data points are from the fitted line

Dealing with interactions

- Interactions are important; our explanatory variables will often interact with each other to affect the response variable
- The usual way to deal with interactions is to create *new explanatory variables* by multiplying them together

```
model_4 = lm(formula = lpsa ~ lweight + age + lweight:age,
summary(model_4)
```

```
##
## Call:
## lm(formula = lpsa ~ lweight + age + lweight:age, data =
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.23926 -0.62770 -0.00107  0.54302  3.00193
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
```

Regularisation and shrinkage

- When you have lots and lots of explanatory variables, the model can become very slow or might not fit at all
- Worse, we might have lots of spurious p-values
- It makes sense to remove or reduce some of the coefficients on the explanatory variables if we think their effect is over-stated
- One way of doing this is via *regularisation*, where we set some of the values to zero, another is via *shrinkage* where we reduce the values (shrink them towards zero)

Lasso and Ridge

- The R package `glmnet` will perform shrinkage and regularisation
- The *Lasso* model imposes a restricted sum on all of the coefficient values
- The *Ridge* model imposes an extra assumption that all of the coefficient values come from a normal distribution
- We will play with some of these models later

Even more advanced regression approaches

- There is lots of research on regression models of all different types
- The vast majority of them involve creating a set of coefficients to multiply the explanatory variables by and then adding everything up
- It's very important to check the model diagnostics

Intro to classification models

The logit transformation

Example: SA Heart rate

Extending the model

Understanding the output

Plotting the fitted model

Regularisation and shrinkage for classification

More advanced classification approaches