Quantitative Applications for Data Analysis: linear models

Erik Spence

SciNet HPC Consortium

27 February 2024

SciNet compute • calcul CANADA

Today's slides

Today's slides can be found here. Go to the "Quantitative Applications for Data Analysis" page, under Lectures, "Linear models".

`https://scinet.courses/1346`

SciNet compute • calcul CANADA

Today's class

Today we will begin our adventures in actual data analysis.

- Initial data exploration.
- Linear models.
- Verification of models.
-
- Generalized linear models.

As always, ask questions.

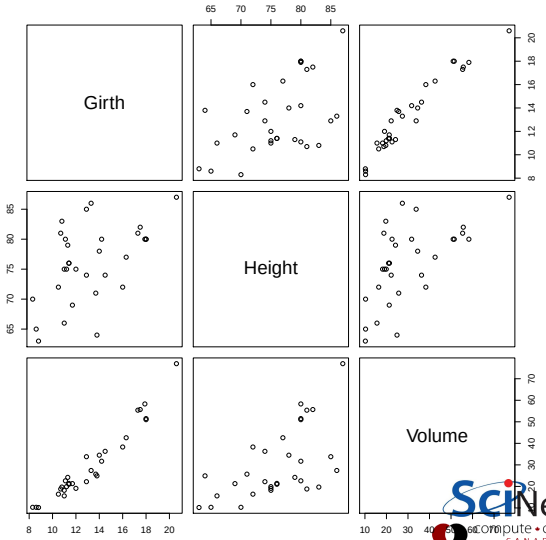
SciNet compute • calcul CANADA

Look at your data

There very first step to dealing with your data is to plot it. Always always always!

The 'pairs' function will do the same as demonstrated here.

```
>
> str(trees)
'data.frame':  31 obs.  of 3 variables:
$Girth :  num 8.3 8.6 8.8 10.5 10.7 ...
$Height:  num 70 65 63 72 81 ...
$Volume:  num 10.3 10.3 10.2 16.4 18.8 ...
>
> plot(trees)
>
```

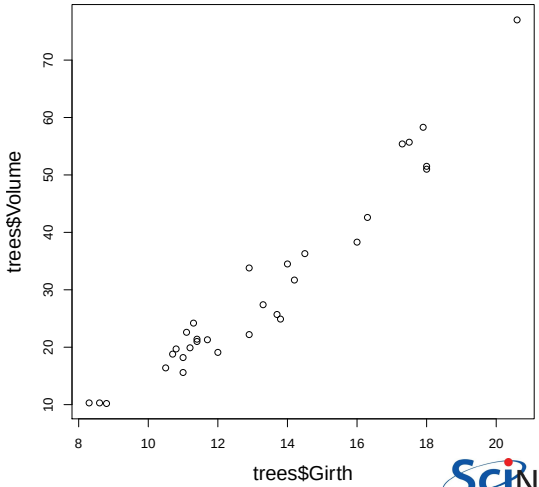


Look at your data, continued

Once you've had a first look, you might want to take a closer look at a particular pair of variables.

```
>
> plot(trees$Girth, trees$Volume)
>
```

Looks like they might be related.



Correlation and covariance

If we suspect that two variables might be related to one another, it's worth our time to look at the correlation and covariance of the variables.

Covariance:

$$\sigma_{XY} = E\left[\left(X - E\left(X\right)\right)\left(Y - E\left(Y\right)\right)\right]$$

Correlation (Pearson's correlation):

$$\rho_{XY} = \frac{E\left[\left(X - E\left(X\right)\right)\left(Y - E\left(Y\right)\right)\right]}{\sigma_X \sigma_Y}$$

Recall that the standard deviation:

$$\sigma_X = \sqrt{E\left[\left(x - E\left(x\right)\right)^2\right]}$$

Where E is the expectation value of the quantity in question.

Correlation and covariance, continued

The "cor" function will produce the correlation from the previous slide.

The "var" function returns the variance or the covariance, depending on the number of arguments.

Recall that the standard deviation is the square root of the variance.

```
>
> cor(trees$Girth, trees$Volume)
[1] 0.9671194
>
> var(trees$Girth, trees$Volume)
[1] 49.88812
>
> var(trees$Girth)
[1] 9.847914
>
> sd(trees$Girth)
[1] 3.138139
>
```

Model fitting

One important application of statistics is the fitting of models to empirical data. There are many ways to do this, but they are all based on the same principles:

- collect some data.
- propose a relationship between the 'features', and the 'target' in your data (if there is a 'target').
  - 'features' are the independent variables in your data ($\mathbf{x}$),
  - the 'target' is the dependent variable ($\boldsymbol{y}$). Not all data sets have dependent variables.
- Fit your model to the data,
- Test and evaluate the quality of the model.

Depending on the field, is this called: modelling, fitting, regression.

Linear models

Let's consider the simplest possible case, that the relationship between the independent and dependent variables is linear.

$$y \simeq \beta_0 + \beta_1 x_1 + ... + \beta_n x_n + \delta$$

As usual:

- y is the dependent variable,
- $x_1, ..., x_n$ are the independent variables,
- β_0 is the intercept,
- $\beta_1, ..., \beta_n$ are the coefficients, and
- δ is noise.

For example, we might assume a relationship for plant growth:

$$\mathbf{growth} \simeq \mathbf{water} + \mathbf{temp} + \mathbf{fertilizer...}$$

The plant growth is linearly related to the temperature, amount of fertilizer and water, etc.

Fitting a linear model

We use the "lm" function to fit a linear model to our data.

The weird thing with the ~ ("tilde") is called a "formula".

Formulae are used, in R, to describe the functional relationship between variables when building models.

```
>
> model <- lm(trees$Volume ~ trees$Girth)
>
> model

Call:
lm(formula = trees$Volume ~ trees$Girth)

Coefficients:
  (Intercept)   trees$Girth
      -36.943         5.066
>
```

Volume $\simeq$ -36.943 + (5.066 x Girth).

Fitting a linear model, continued

The lm function returns an object of class 'lm'.

This is essentially just a very-deep named list.

If we so desire, we can now use the model to calculate the model's prediction for a new tree, with a girth of, say, 15.12.

- Use the 'predict' function.
- Use the coefficients directly.

```
>
> model <- lm(Volume ~ Girth, data = trees)
>
> predict(model, newdata = data.frame(Girth = 15.12))
       1
39.65229
>
> coef(model)
  (Intercept)   trees$Girth
   -36.943459      5.065856
>
> coef(model)[1] + coef(model)[2] * 15.12
(Intercept)
39.65229
>
```

Using formulae, an aside

Formulae show up all over the place in R. There are two ways of building a formula:

- Use vectors of data as the arguments to the formula.
- Specify the names of the columns of a data frame, and then pass the data frame as an argument to the function.
- The entry to the left of the tilde is the dependent variable.
- All the entries to the right of the tilde are the independent variables (there can be more than one).

```
>
> model <- lm(trees$Volume ~ trees$Girth)
>
> model <- lm(Volume ~ Girth, data = trees)
>
> model2 <- lm(Volume ~ Girth + Height,
+     data = trees)
>
```

The second option, specifying column names, is unfortunate due to its syntax being inconsistent with the rest of R, but it is the one more commonly used.

Using formulae, an aside, continued

There are a few other ways to specify a formula.

- A formula can be assigned to a variable, and used later.
- To specify "all columns which have not yet been mentioned" us the ".".
- To remove an already-specified feature, use the minus sign.
- You can also mix data frame columns and non-column vectors.

```
> f <- Volume ~ .
> class(f)
[1] "formula"
>
> trees.model <- lm(f, data = trees)
>
> library(boot)
> model <- lm(ulcer ~ .  - sex - year, data = melanoma)
>
> model
Call:
lm(formula = ulcer ~ .  - sex - year, data = melanoma)
Coefficients:
(Intercept)       time       status          age    thickness
6.146e-01   -5.595e-05  -1.417e-01   4.210e-04   6.045e-02
>
```

Calculating confidence intervals, an aside

Suppose that you've calculated the mean, $\bar{x}$, of some quantity. What is the uncertainty on that quantity?

To answer this question, we estimate the Standard Error

$$\mathbf{SE}\,(x)^2 = \frac{s^2}{n}$$

where $s^2 = \sum (x_i - \bar{x})^2 / (n-1)$. The 95% confidence interval, in which there is a 95% chance the true mean of the population lies, is given by

$$\mu \pm 1.96\ \mathbf{SE(x)}$$

This is because 1.96 standard deviations is approximately what contains 95% of the normal distribution.

```
>
> x <- trees$Girth
>
> my.mean <- mean(x)
>
> se2 <- var(x) / length(x)
>
> my.mean - 1.96 * sqrt(se2)
[1] 12.14368
>
> my.mean + 1.96 * sqrt(se2)
[1] 14.35309
>
```

Calculating confidence intervals, an aside, continued

We can also use the Standard Errors to perform hypothesis tests on the coefficients of our linear model.

- The Null Hypothesis is that the coefficients in our linear model have a value of zero, meaning that the dependent variable does not depend on the independent variable.
- To test this we need to determine whether our estimate of the coefficient is sufficiently far from zero to reject the Null Hypothesis. How far is far enough?
- We can check in the usual way, by calculating a t-statistic

$$t_i = \frac{\beta_i - 0}{\mathrm{SE}(\beta_i)}$$

- This estimates the number of standard deviations that β_i is away from zero.
- The question then becomes: what is the probability of getting a t statistic of value $|t|$, or larger? This is your p value.

Calculating confidence intervals, an aside, more

Given our t statistic (for a Null Hypothesis of zero):

$$t_i = \frac{\beta_i - 0}{\mathbf{SE}(\beta_i)}$$

we need only determine the probability of getting $|t|$ or greater. To determine this we use a 't' distribution, which is very close to the Gaussian CDF for $n > 30$. The second argument is the number of degrees of freedom. The factor of 2 is because it could be left or right tailed.

```
>
> coefs <- coef(summary(trees.model))
>
> est <- coefs["Height", "Estimate"]
> std.err <- coefs["Height", "Std. Error"]
>
> my.t <- (est - 0) / std.err
>
> my.t
[1] 2.606594
>
> 2 * (1 - pt(my.t, length(x) - 3))
[1] 0.01449097
>
```

The p value is small. The probability of committing a Type I error is quite low.

Fitting a linear model, continued more

Important details about the model can be found in the summary:

- The "t value" is the estimate divided by the standard error.
- The p-value is the probability of achieving a value of t, or larger, under the null hypothesis (estimate = 0).

```
> model <- lm(Volume ~ Girth + Height, data = trees)
> summary(model)
Call:
lm(formula = Volume   Girth + Height, data = trees)

Residuals:
     Min       1Q   Median       3Q      Max
 -6.4065  -2.6493  -0.2876   2.2003   8.4847
Coefficients:
             Estimate   Std. Error   t value   Pr(>|t|)
 (Intercept)  -57.9877       8.6382    -6.713   2.75e-07   ***
 Girth          4.7082       0.2643    17.816    < 2e-16   ***
 Height         0.3393       0.1302     2.607     0.0145   *
---
Signif.  codes:  0 *** 0.001 ** 0.01 * 0.05 .  0.1   1

Residual standard error:  3.882 on 28 degrees of freedom
Multiple R-squared:  0.948, Adjusted R-squared:  0.9442
F-statistic:  255 on 2 and 28 DF, p-value:  < 2.2e-16
```

Fitting a linear model, continued even more

What about that F-statistic at the bottom?

- The fit also gives an analysis of the null hypothesis that $\beta_1 = \beta_2 = ... = \beta_n = 0$.
- This means that there's no dependence on the independent variables at all.
- This null hypothesis can be rejected in this case.

```
> model <- lm(Volume ~ Girth + Height, data = trees)
> summary(model)
Call:
lm(formula = Volume   Girth + Height, data = trees)

Residuals:
     Min       1Q   Median       3Q      Max
 -6.4065  -2.6493  -0.2876   2.2003   8.4847
Coefficients:
             Estimate   Std. Error   t value   Pr(>|t|)
 (Intercept)  -57.9877       8.6382    -6.713   2.75e-07   ***
 Girth          4.7082       0.2643    17.816    < 2e-16   ***
 Height         0.3393       0.1302     2.607     0.0145   *
---
Signif.  codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.'  0.1 ' ' 1

Residual standard error:  3.882 on 28 degrees of freedom
Multiple R-squared:  0.948, Adjusted R-squared:  0.9442
F-statistic:  255 on 2 and 28 DF, p-value:  < 2.2e-16
```

Fitting a linear model, continued some more

There are some assumptions built into "lm". You need to know the fine print:

- The noise in the data, δ, is normally distributed about the true value.
- Homoscedasticity: the variance in the noise is constant throughout the data.

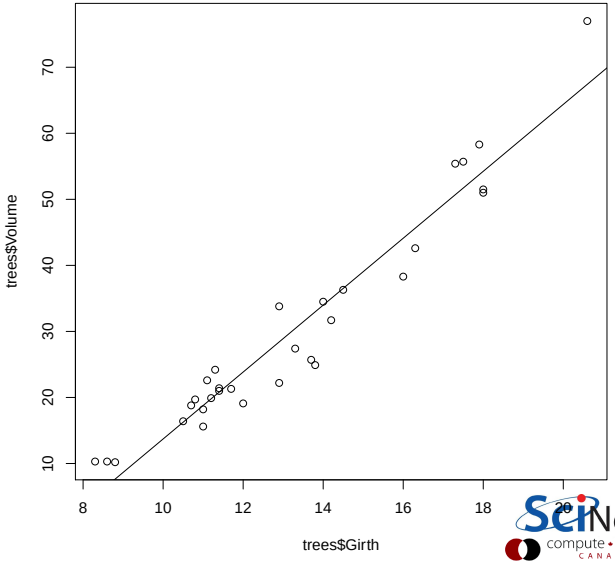
If these conditions are not met your model is not on a good statistical foundation.

Fitting a linear model, continued even more

It's always good to visualize your model once it's been made.

```
>
> model <- lm(Volume ~ Girth,
+       data = trees)
>
> plot(trees$Girth, trees$Volume)
> abline(model)
>
```

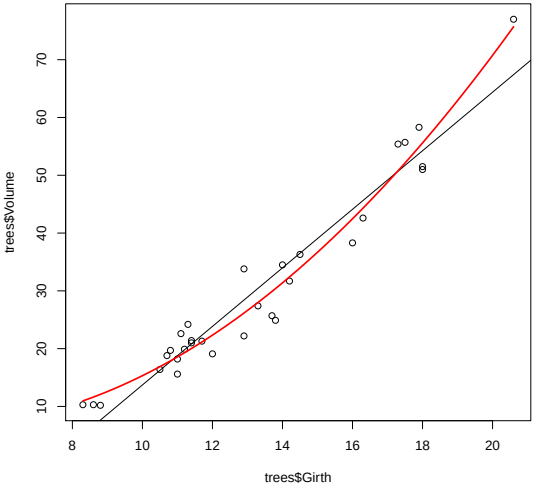
Not bad, but could be better.



Fitting a quadratic model

We can increase the order of the polynomial which we are fitting to the data.

```
> Girth2 <- trees$Girth**2
>
> model2 <- lm(Volume ~ Girth + Girth2,
+       data = trees)
>
> plot(trees$Girth, trees$Volume)
> abline(model)
>
> xx <- seq(min(trees$Girth), max(trees$Girth),
+       len = 100)
> yy <- predict(model2, data.frame(Girth = xx,
+                                  Girth2 =
xx**2))
>
> lines(xx, yy, lwd = 2, col = "red")
```

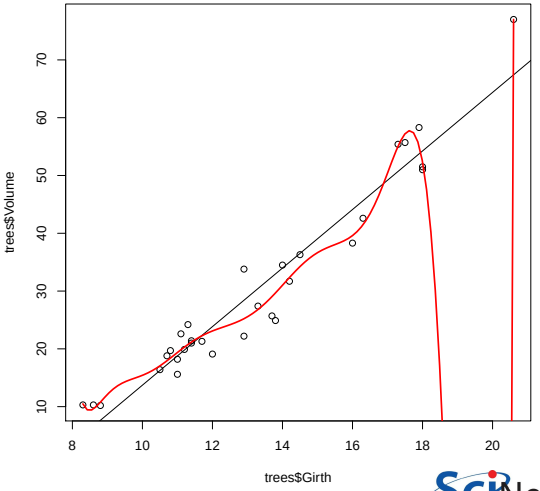


The "abline" command only works with linear fits.

Fitting a higher-order model

There are several ways to have a higher-order fit. The best one is to use 'poly'.

```
>
> model10 <- lm(Volume ~ poly(Girth, 10),
+               data = trees)
>
> plot(trees$Girth, trees$Volume)
> abline(model)
>
> p.model10 <- predict(model10,
+                      data.frame(Girth = xx))
>
> lines(xx, p.model10, lwd = 2, col = "red")
>
```



What's up with 'poly'?

We've seen that one way to do a non-linear fit is to use the commands

```
> xsq <- x * x
> model3 <- lm(y ~ x + xsq)
```

or it can be written as

```
> model3 <- lm(y ~ x + I(x**2) + I(x**3))
```

Unfortunately, when used this way, x, x^2 and x^3 will be correlated with each other. This can lead to resolution problems, especially at higher orders.

The 'poly' function fixes this by generating at set of orthogonal polynomials evaluated at 'x'.

That's great, but we're not done yet

It's always a good idea to do some further analysis of your model before declaring success. There are a few things in particular that should always be done.

- plot the residuals of the model, in various ways,
- examine the statistics of the residuals,
- examine the statistics of the model.

What are residuals? Residuals are the distance between the actual value, and the value predicted by the model, for each data point:

$$R_i = f(x_i) - y_i$$

where f is the model, evaluated at data point x_i, and y_i is the actual value of the dependent variable.

Step 1: plot the residuals

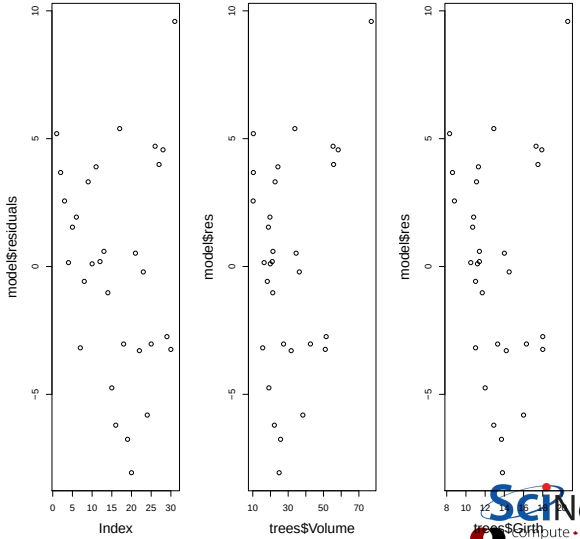
Always plot your residuals. Always.

```
> par(mfrow = c(1, 3))
>
> plot(model$residuals)
> plot(trees$Volume, model$residuals)
> plot(trees$Girth, model$residuals)
>
```

Plot your residuals against everything:

- index,
- against the dependent variables,
- against the independent variables.

You should see a snowstorm. There should be no clumps.

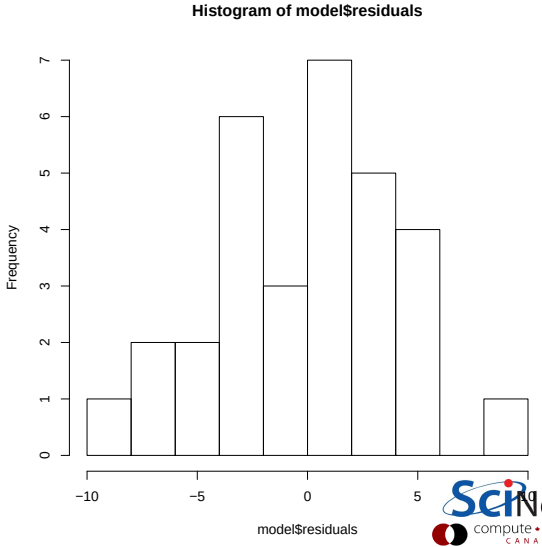


Step 2: plot the residuals via histogram

Always plot a histogram of your residuals.
Things to look for:

- The mean should be zero. If your residuals are not centered on zero your model is missing something.
- The distribution should be symmetric. If it's not, it's biased (there 'structure' in the data which has not been captured by the model).
- Distribution should be a Gaussian (an assumption made as part of the fit).

```
> par(mfrow = c(1, 1))
> hist(model$residuals, breaks = 11)
```

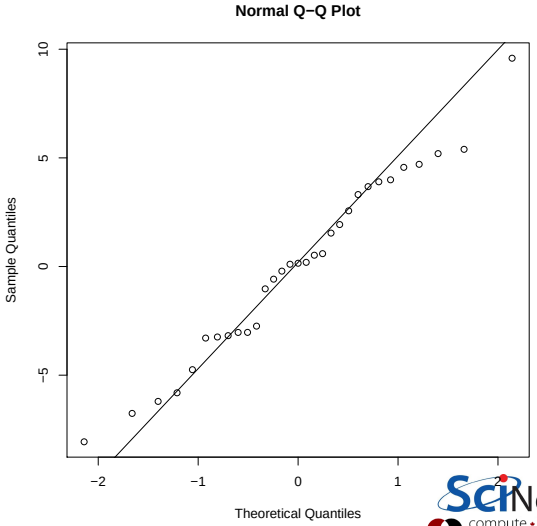


Step 3: plot the residuals via Q-Q plot

Plot your residuals on a Q-Q plot.

- A Q-Q plot graphically demonstrates how normally-distributed the residuals are.
- Ideally the residuals should be normally distributed.

```
>
> qqnorm(model$residuals)
> qqline(model$residuals)
>
```



Using R^2

R^2 = (explained variation) / (total variation).

- Explains how much of the variance in the data can be explained by the model.
- All other variation is caused by shortcomings in the model, or noise.
- A high R^2 value is necessary, but not sufficient, for the model to be satisfactory.

```
> summary(model)
Call:
lm(formula = Volume   Girth + Height, data = trees)

Residuals:
     Min       1Q   Median       3Q      Max
 -6.4065  -2.6493  -0.2876   2.2003   8.4847
Coefficients:
             Estimate   Std. Error   t value   Pr(>|t|)
 (Intercept)  -57.9877       8.6382    -6.713   2.75e-07   ***
 Girth          4.7082       0.2643    17.816    < 2e-16   ***
 Height         0.3393       0.1302     2.607     0.0145   *
---
Signif.  codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’  0.1 ‘ ’
1

Residual standard error:  3.882 on 28 degrees of freedom
Multiple R-squared:  0.948, Adjusted R-squared:  0.9442
F-statistic:  255 on 2 and 28 DF, p-value:  < 2.2e-16
```

Adjusted R^2

Because more-complex models will always have a higher R^2, since it does not punish models for complexity, it's better to use Adjusted R^2 as a measure of how well the model is doing.

Adjusted R^2 punishes the model's R^2 score for the number of free parameters that are used.

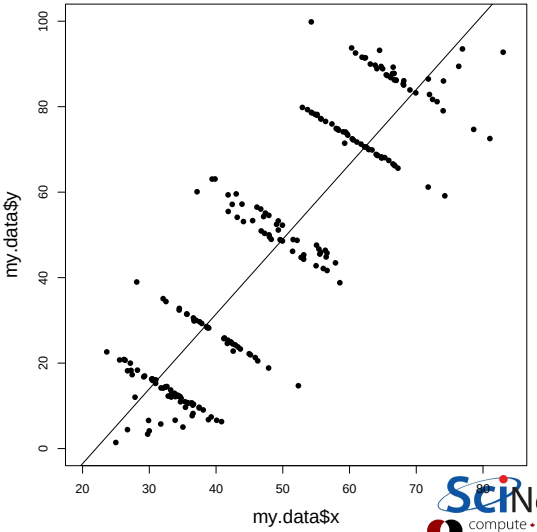
```
> summary(model)
Call:
lm(formula = Volume ~ Girth, data = trees)
Residuals:
     Min       1Q   Median       3Q      Max
 -8.065   -3.107    0.152   3.495   9.587
Coefficients:
             Estimate   Std. Error   t value   Pr(>|t|)
 (Intercept)  -36.9435       3.3651    -10.98   7.62e-12   ***
 Girth          5.0659       0.2474     20.48    < 2e-16   ***
---
Signif.  codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’  0.1 ‘ ’ 1

Residual standard error:  4.252 on 29 degrees of freedom
Multiple R-squared:  0.9353, Adjusted R-squared:  0.9331
F-statistic:  419.4 on 1 and 29 DF, p-value:  < 2.2e-16
>
```

Simpson's paradox

Simpson's paradox (also called the Yule-Simpson effect) is a phenomenon in statistics in which a trend appears in several different groups of data, but disappears or reverses when these groups are combined.

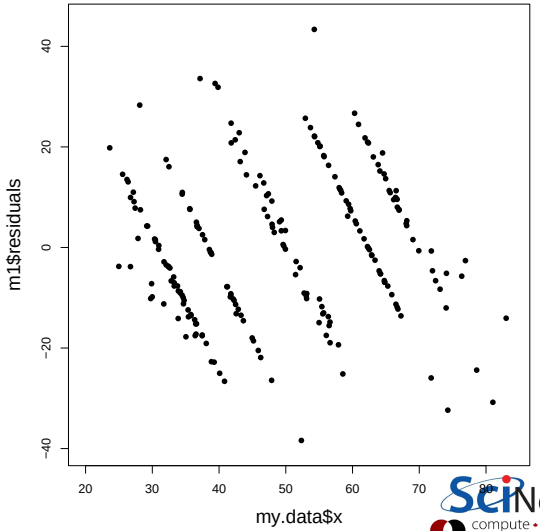
```
> library(datasauRus)
>
> sp <- simpsons_paradox
> my.data <- sp[sp$dataset == 'simpson_2',]
>
> plot(my.data$x, my.data$y)
> m1 <- lm(y ~ x, data = my.data)
> abline(m1)
```



Simpson's paradox, continued

Plotting the residuals in this case will demonstrate plenty of structure, indicating that there's something wrong with the model, or that, as in this case, the model is incomplete.

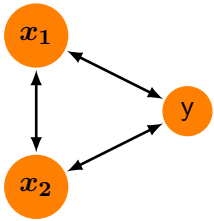
```
>
> plot(my.data$x, m1$residuals)
>
```



Confounding variables

Confounding variables are independent variables that influence both the dependent variable and at least one other independent variable.

- Failing to include, or deal with, these variables can result in an exaggeration or masking of the relationship between the independent and dependent variables.
- Omitting confounding variables forces the model to attribute their affects to other variables, resulting in a "confounding" of the actual relationship between variables.
- This results in "omitted variable bias".



On the right, two independent variables, x_1 and x_2, are correlated with both the dependent variable, y, and each other.

Preventing confounding-variable problems

If you have the opportunity, you can deal with potential problems with confounding variables at the design stage of your study. These are several ways to approach this:

- Matching: create two groups, where each member of one group has an identical counterpart, with respect to confounding variables, in the other group.
- Restriction: creating similar groups of subjects that are all the same with respect to the confounding variable, and thus the confounding variable has no effect on the dependent variable.
- Stratification: estimate the relationship between your dependent and independent variables within groups with different values of confounding variables, then average the estimates.
- Randomization: subjects are randomly, with respect to confounding variables, assigned to different groups. As such the average values of confounding variables should be the same in each group.

SciNet compute • calcul CANADA

Confounding variables, continued

Failing to include your confounding variables in your model may cause problems.

- The missing confounding variable can cause a bias in the estimated coefficients in your model, especially if the two variables are negatively correlated, as the model attempts to compensate for the missing variable.
- The amount of bias depends on the strength of the correlation. The stronger the correlation the stronger the bias.
- Conversely, independent variables that have no correlation to any other independent variables will not be biased at all. Thus the coefficient for this feature will not change as different features are added/removed from the model.
- If you include different combinations of independent variables in your model, and the coefficients change significantly, you are observing "omitted variable bias".

One of the assumptions of our linear model is that the noise is uncorrelated with the data. Any correlations in the data will appear as some sort of structure in your residuals.

Multicollinearity in your data

Suppose you've included all confounding variables in your model in the hopes that the model can deal with them. This introduces what is called "multicollinearity" into your model.

- This can be an issue in itself.
- Why? Well, the independent variables should be independent. If they're correlated to each other they're obviously not independent.
- If they're not independent, it becomes harder for the model to estimate the coefficient associated with the independent variables, but they don't vary independently of each other.
- As a result,
  - coefficients become unstable depending on which variables are included in the model.
  - the precision of the coefficients is reduced

If the two variables are positively correlated you may be able to remove one of them from the model. If they're negatively correlated you probably cannot.

SciNet compute • calcul CANADA

Multicollinearity in your data, continued

So how much is multicollinearity a problem?

If the correlations between variables isn't too strong, you can probably safely ignore it.

If it is strong you should probably remove a variable, or possibly centring and scaling the data may help.

Not surprisingly, there's a measure we can use to determine the multicollinearity of the independent variables in our model: "variance inflation factor" (VIF).

- VIF gives the strength of the correlation between independent variables.
- It starts at 1, and has no upper limit.
  - A value near 1 indicates weak correlation.
  - A value from 1 - 5 indicates moderate correlation, but is probably harmless.
  - A value greater than 5 indicates strong multicollinearity.

As you might expect, code to calculate VIF in R already exists.

SciNet compute • calcul CANADA

Measuring multicollinearity

The 'car' library contains the 'vif' function, which calculates the multicollinearity of the independent variables in the model.

The 'ape' library contains many phylogenetic data sets, including data concerning species in the order 'carnivora'.

This model is looking at the relationship between weaning age (WA), gestation length (GL), litter size (LS), female weight (FW), female brain weight (FB) and birth weight (BW).

Examining the coefficients indicates that FB can be dropped from the model.

```
>
> library(ape)
> library(car)
> data(carnivora)
>
> model <- lm(WA ~ GL + LS + FW + FB + BW,
+             data = carnivora)
> vif(model)
      GL       LS       FW       FB       BW
2.267679 1.295047 8.339708 10.930205 3.182454
> >
> model2 <- lm(WA ~ GL + LS + FW + BW,
+              data = carnivora)
> vif(model2)
      GL       LS       FW       BW
1.752621 1.158614 2.197752 2.940201
>
```

Generalized linear models

The linear model built by "lm" has some built-in assumptions:

- Normally distributed noise.
- Uncorrelated noise.
- Constant variance of the noise.
- Data is not correlated to the noise.
- Independent variables are independent.

There are situations where these assumptions are dramatically violated. To deal with this, let us examine "Generalized Linear Models". These allow

- Non-normally distributed noise.
- Non-constant variance.

If you find that you have structure in your residuals, it's possible that you need to use a generalized linear model.

Generalized linear models, continued

When should you use a generalized linear model?

- You know that your data should come from a non-linear, non-polynomial distribution (exponential, Poisson, etc).
- You don't know what your distribution should be, and you've got structure in your residuals.

How do generalized linear models work? Let's start with a regular linear model. Assuming the vectors of data are $(\boldsymbol{X}, \boldsymbol{Y})$, the problem is to find the vector of coefficients $\boldsymbol{\beta}$ such that

- $\boldsymbol{E}(\boldsymbol{Y}) = \boldsymbol{X}\boldsymbol{\beta}$
- assuming that $\boldsymbol{Y} \sim \boldsymbol{N}(\boldsymbol{X}\boldsymbol{\beta}, \boldsymbol{\sigma^2})$,

where $\boldsymbol{E}$ is the expectation value, $\boldsymbol{N}(\boldsymbol{\mu}, \boldsymbol{\sigma^2})$ is the symbol for a normal distribution centred on $\boldsymbol{\mu}$ with a standard deviation of $\boldsymbol{\sigma}$.

Generalized linear models, continued

As an example, for a log-linked Gaussian GLM, we have

- $\log\left(\boldsymbol{E}(\boldsymbol{Y})\right) = \boldsymbol{X}\boldsymbol{\beta}$,
- which means that $\boldsymbol{E}(\boldsymbol{Y}) = e^{\boldsymbol{X}\boldsymbol{\beta}}$,
- $\boldsymbol{Y} \sim \boldsymbol{N}(e^{\boldsymbol{X}\boldsymbol{\beta}}, \boldsymbol{\sigma^2})$.

where $\boldsymbol{E}$ is the expectation value, $\boldsymbol{N}(\boldsymbol{\mu}, \boldsymbol{\sigma^2})$ is the symbol for a normal distribution centred on $\boldsymbol{\mu}$ with a standard deviation of $\boldsymbol{\sigma}$.

Generalized linear models consist of 3 parts:

- A "link" function. A function which transforms the data such that it becomes linear.
- A linear predictor ($\boldsymbol{X}\boldsymbol{\beta}$).
- A probability distribution, which describes the type of noise to be expected in the dependent variable.

Generalized linear models, continued more

There are many possible link functions available. The most common ones are

- Identity: $\boldsymbol{E(Y) = X\beta}$,
- Log: $\boldsymbol{\log\left(E(Y)\right) = X\beta \quad \rightarrow \quad E(Y) = e^{X\beta}}$.
- Logit: $\boldsymbol{\log\left(\frac{E(Y)}{1-E(Y)}\right) = X\beta \quad \rightarrow \quad E(Y) = \frac{1}{1+e^{-X\beta}}}$
- Inverse: $\boldsymbol{1/E(Y) = X\beta \quad \rightarrow \quad E(Y) = 1/\left(X\beta\right)}$

The identity link function results in a standard linear regression. By performing a generalized linear model using this link function, with Gaussian noise, you will get the same result as using the "lm" function.

Generalized linear models, continued even more

Once a link function has been chosen, the type of error in the data must be chosen. The different error families have different default link functions.

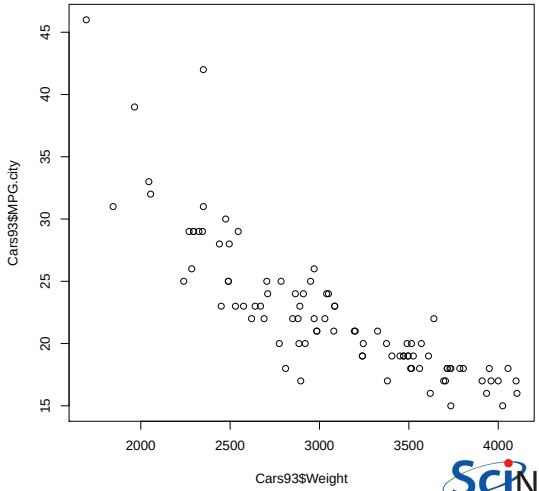
Error family	Default link	Link inverse	Use for:
gaussian	identity	1	Normally distributed error
poisson	log	exp	Counts
binomial	logit	$1/(1+e^{-x})$	Proportions or binary data
Gamma	inverse	$1/x$	Continuous data with non-constant error

```
> glm(formula, family = binomial(link = log))
```

GLM example

Consider the Cars93 data set.
Plotting the MPG in the city, versus Weight, suggests a non-linear relationship.

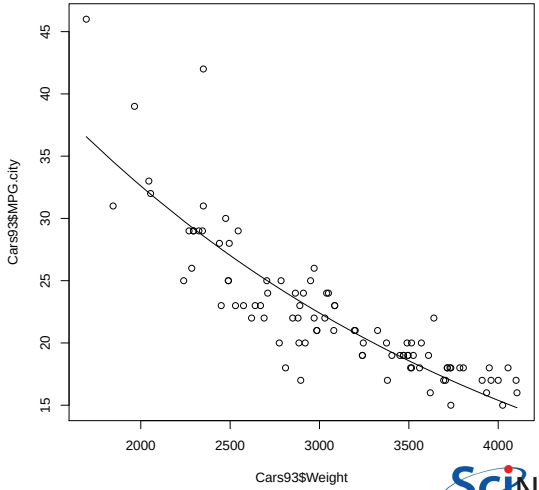
```
>
> library(MASS)
>
> plot(Cars93$Weight, Cars93$MPG.city)
>
```



GLM example, continued

Let's perform a GLM, using Gaussian noise and the log link function.

```
> sorted.weights <- sort(Cars93$Weight)
>
> glm1 <- glm(MPG.city ~ Weight,
+     data = Cars93,
+     family = gaussian(link = "log"))
>
> plot(Cars93$Weight, Cars93$MPG.city)
> lines(sorted.weights,
+     predict(glm1,
+     data.frame(Weight = sorted.weights),
+     type = "response"))
>
```



Summary

We've started looking at data, and fitting it. Things to remember:

- Plot your data!
- Start with lm, both for linear and other polynomial fits.
- Plot the residuals! There is important information in there!
- If the data are not polynomial, or the residuals are not normally distributed, you may need to use a Generalized Linear Model.
- You will likely need to play around with the different noise families and link functions to find one that best works with your data.
- Other types of regression include logistic regression, for fitting categories, and multinomial regression, for multiple dependent variables.

SciNet compute • calcul CANADA