Kitab at-Tabakh

Peter Geelan-Small

28/09/2020

Contents

Intro

This is a collection of miscellaneous statistics-related recipes and notes.

Chapter 1

Mawaqif

1.1 Confounders

https://en.wikipedia.org/wiki/Confounding

https://www.ucl.ac.uk/child-health/short-courses-events/about-statistical-courses/research-methods-and-statistics/chapter-1-content-0

1.2 Exponential models

$y = Ae^{bt}$

$t = 0, y = 5$

$A = 5$

Three states:

$t = 10, y_1 = 500, y_2 = 100, y_3 = 25$

$b = (\log y - \log 5)/t$

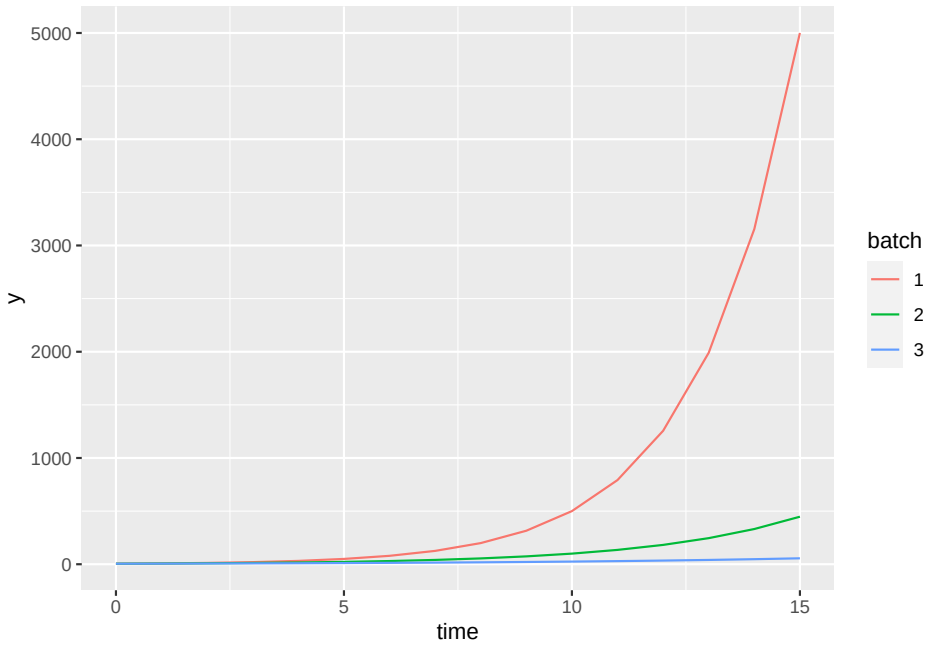
$b_1 = (\log 500 - \log 5)/10 = 0.461$

$b_2 = (\log 100 - \log 5)/10 = 0.300$

$b_3 = (\log 25 - \log 5)/10 = 0.161$

```
tt = 0:15

y1 <- A * exp(b_1 * tt)

y2 <- A * exp(b_2 * tt)
```

```
y3 <- A * exp(b_3 * tt)

dd1 <- data.frame(time = rep(tt, 3), y = c(y1, y2, y3),
                  batch = rep(1:3, each = length(tt)))

dd1$batch <- factor(dd1$batch)

ggplot(dd1, aes(x = time, y = y, colour = batch)) +
    geom_line()
```

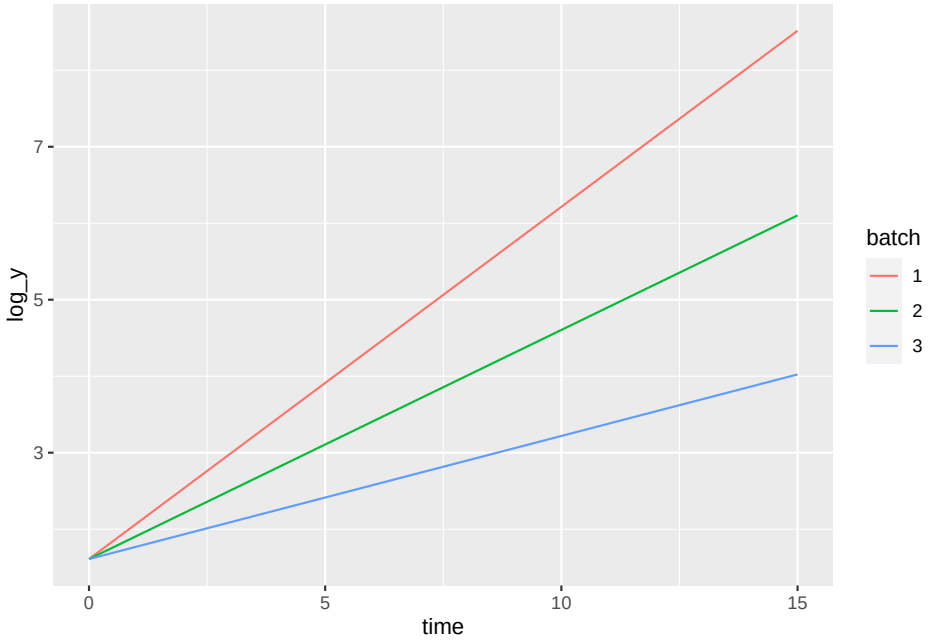


$y = Ae^{bt}$

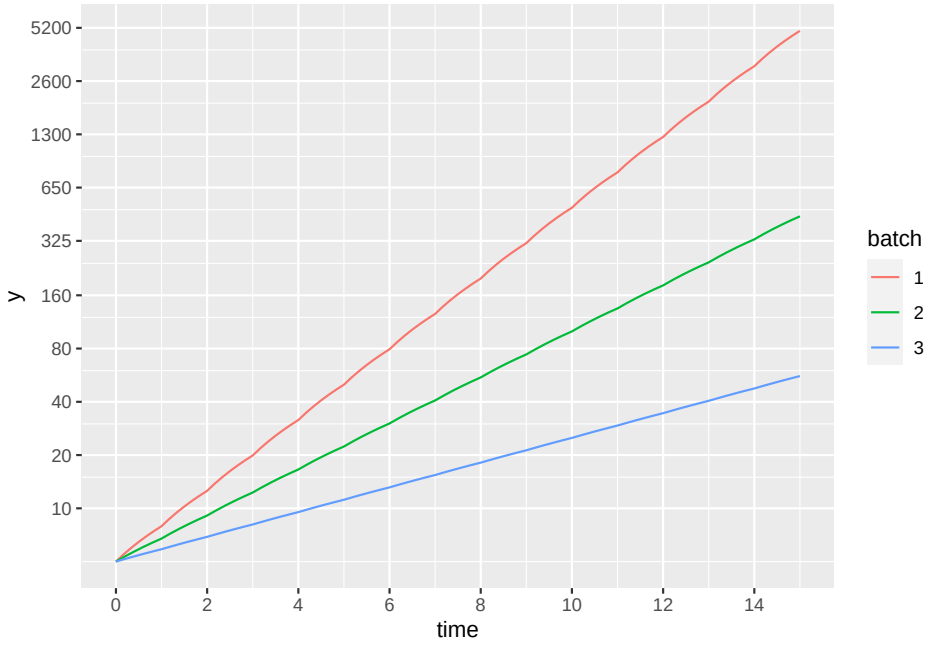
is the same as

$\log y = \log A + bt$

```
dd1$log_y <- log(dd1$y)

ggplot(dd1, aes(x = time, y = log_y, colour = batch)) +
    geom_line()
```




```
ggplot(dd1, aes(x = time, y = y, colour = batch)) +
    geom_line() +
    coord_trans(y = 'log') +
    scale_x_continuous(breaks = c(0:8*2)) +
    scale_y_continuous(breaks = c(10, 20, 40, 80, 160, 325,
                                  650, 1300, 2600, 5200))
```



For a one unit increase in time, the change in y is:

$\log y_j - \log y_i = b$

So, $y_j = e^b y_i$

For batch 1, $e^b = e^{0.461} = 1.59$, so for each one unit increase in time, y increases by a factor of 1.59 or by 59%.

Values of batch 1 at times from 0 to 15:

```
round(y1, digits = 3)
```

```
##  [1]    5.000    7.924   12.559   19.905   31.548   50.000   79.245
##  [8]  125.594  199.054  315.479  500.000  792.447 1255.943 1990.536
## [15] 3154.787 5000.000
```

Ratios of $y_{1,k} \,/\, y_{1,k-1}$ for some values are shown below, noting that $e^{b_1} = 1.585$.

For an increase of two units in time, $\log y_j - \log y_i = 2b$, so $y_j = e^{2b} y_i$.

$e^{2b_1} = 2.512$

```
rat_21 <- round(y1[2] / y1[1], digits = 3)

rat_32 <- round(y1[3] / y1[2], digits = 3)
```

```
rat_10_9 <- round(y1[10] / y1[9], digits = 3)

rat_15_13 <- round(y1[15] / y1[13], digits = 3)
```

Times	Ratio
1 & 2	1.585
2 & 3	1.585
9 & 10	1.585
13 & 15	2.512

Doubling time

When has y reached double its initial value - i.e. when is $y = 10$? Doubling time is t_{dbl}.

$e^{bt_{\text{dbl}}} = 2$

$t_{\text{dbl}} = \log 2/b$

For batch 1: $t_{\text{dbl}} = \log 2/0.461 = 1.51$

For batch 2: $t_{\text{dbl}} = \log 2/0.300 = 2.31$

For batch 1: $t_{\text{dbl}} = \log 2/0.161 = 4.31$

```
t10 <- 5

t12 <- t10 + t_dbl_1

t22 <- t10 + t_dbl_2

t32 <- t10 + t_dbl_3

y11 <- round(A * exp(b_1 * t10), digits = 2)

y21 <- round(A * exp(b_2 * t10), digits = 2)

y31 <- round(A * exp(b_3 * t10), digits = 2)

y12 <- round(A * exp(b_1 * t12), digits = 2)

y22 <- round(A * exp(b_2 * t22), digits = 2)

y32 <- round(A * exp(b_3 * t32), digits = 2)
```

Table below shows value of y at time = 5 and then at time = (5 + doubling time).

Batch	time 1 (= 5)	time 2 (= 5 + t_dbl)
1	50	100
2	22.36	44.72
3	11.18	22.36

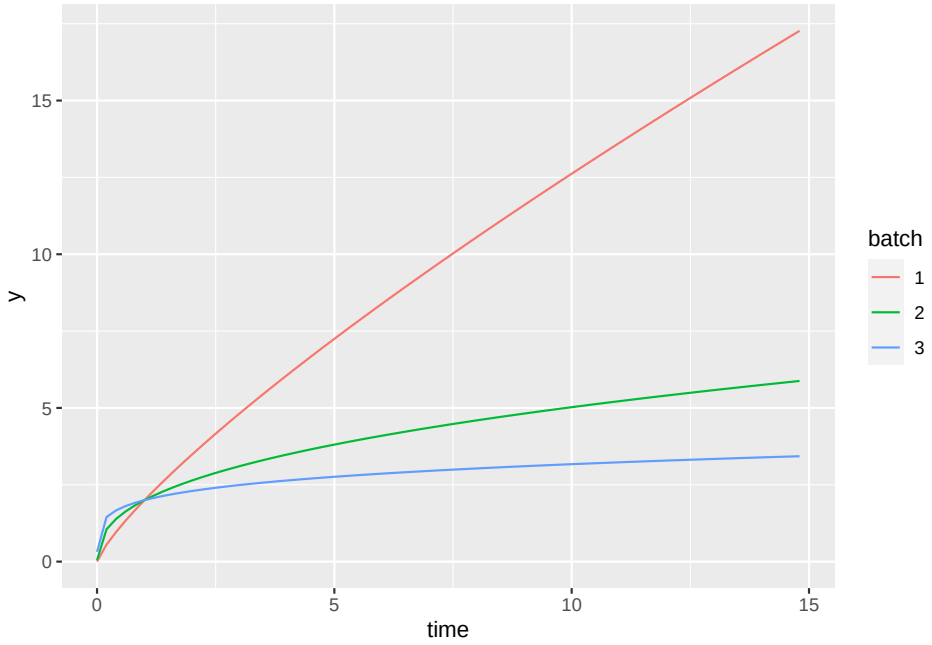
1.3 Power function

$y = Cx^b$

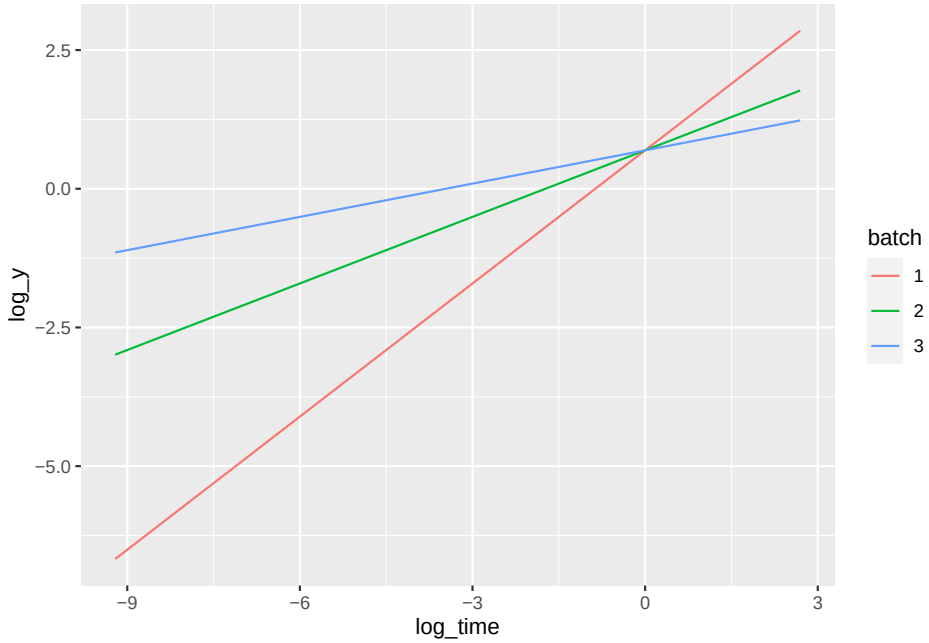
This becomes $\log y = \log C + \log(x^b)$ or $\log y = c + b \log x$.

Take $C = 2$ and $b_4 = 0.8, b_5 = 0.4, b_6 = 0.2$

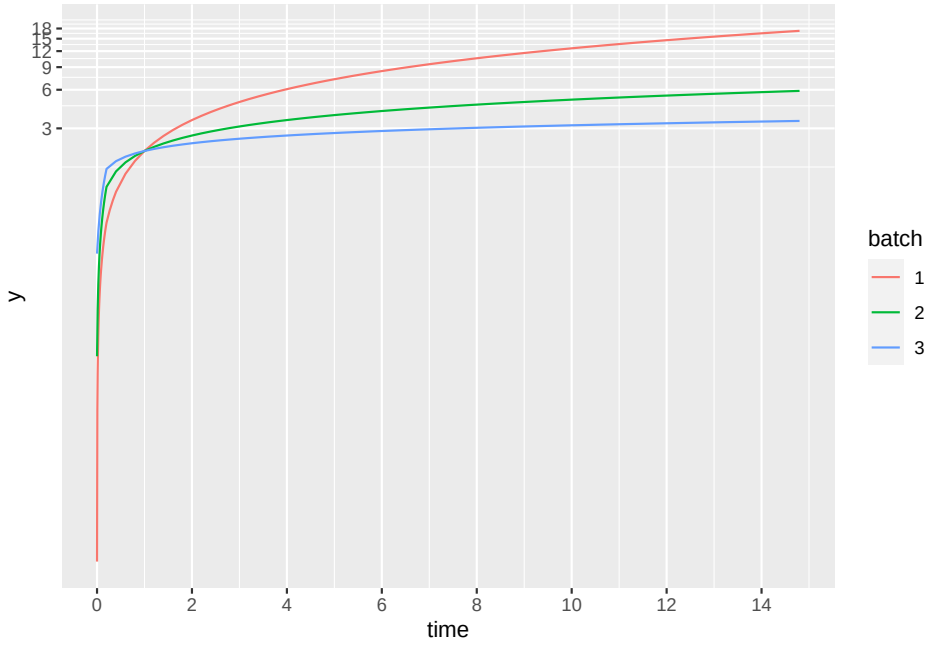
```
C <- 2

b_4 <- 0.8

b_5 <- 0.4

b_6 <- 0.2

tt = seq(0.0001, 15, by = 0.2)

y4 <- C * tt^b_4

y5 <- C * tt^b_5

y6 <- C * tt^b_6

dd2 <- data.frame(time = rep(tt, 3), y = c(y4, y5, y6),
                  batch = rep(1:3, each = length(tt)))

dd2$batch <- factor(dd2$batch)

ggplot(dd2, aes(x = time, y = y, colour = batch)) +
    geom_line()
```




```
dd2$log_y <- log(dd2$y)

dd2$log_time <- log(dd2$time)

ggplot(dd2, aes(x = log_time, y = log_y, colour = batch)) +
    geom_line()
```




```
ggplot(dd2, aes(x = time, y = y, colour = batch)) +
    geom_line() +
    coord_trans(x = 'log') +
    coord_trans(y = 'log') +
    scale_x_continuous(breaks = c(0:8*2)) +
    scale_y_continuous(breaks = c(0:6*3))
```



1.4 ROC curves

Example from Bland.

```
library(ggplot2)
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
roc_ami <- read.csv("data/roc bland.csv", header = T)

str(roc_ami)
```

```
## 'data.frame': 120 obs. of  2 variables:
##  $ CK     : int  23 33 36 37 37 41 41 41 42 42 ...
##  $ Disease: int  0 0 0 0 0 0 0 0 0 0 ...
```

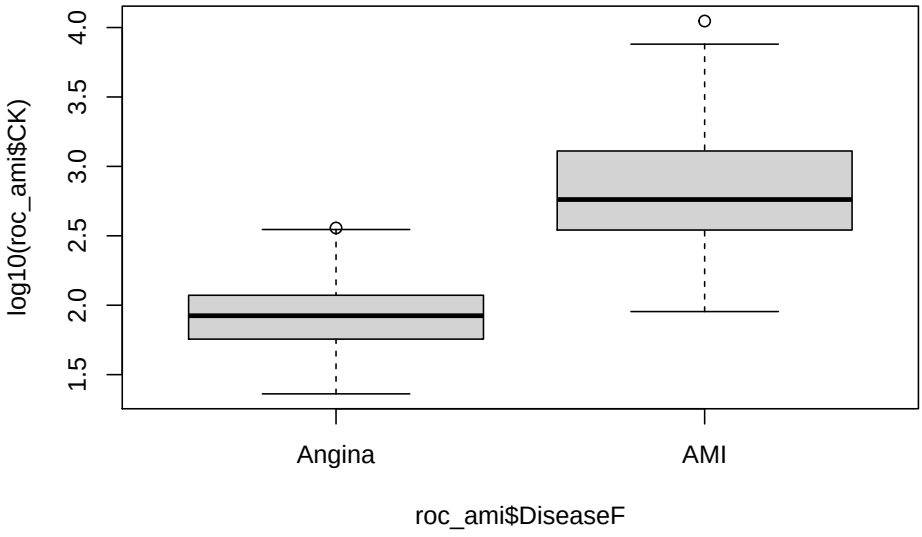
```r
summary(roc_ami)
```

```
##        CK              Disease
##  Min.   :   23.0   Min.   :0.000
##  1st Qu.:   61.5   1st Qu.:0.000
##  Median :  101.5   Median :0.000
##  Mean   :  410.0   Mean   :0.225
##  3rd Qu.:  257.0   3rd Qu.:0.000
##  Max.   :11138.0   Max.   :1.000
```

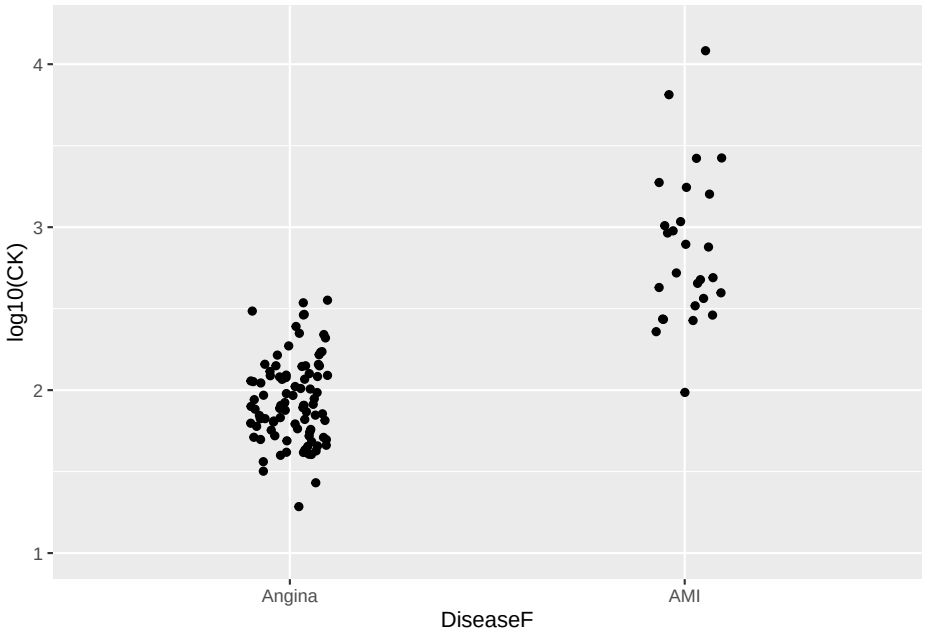
```r
##  Labels: Angina is "unstable angina";
##    AMI is "acute myocardial infarction"

roc_ami$DiseaseF <- factor(roc_ami$Disease, labels = c("Angina", "AMI"))

head(roc_ami)
```

```
##   CK Disease DiseaseF
## 1 23       0   Angina
## 2 33       0   Angina
## 3 36       0   Angina
## 4 37       0   Angina
## 5 37       0   Angina
## 6 41       0   Angina
```

```r
boxplot(log10(roc_ami$CK) ~ roc_ami$DiseaseF)
```



```
ggplot(roc_ami, aes(x = DiseaseF, y = log10(CK))) +
  ylim(1, 4.2) +
  geom_jitter(width = 0.1, height = 0.1)
```




```
ami_roc_curve <- roc(Disease ~ CK, data = roc_ami)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
ami_roc_curve
```

```
##
## Call:
## roc.formula(formula = Disease ~ CK, data = roc_ami)
##
## Data: CK in 93 controls (Disease 0) < 27 cases (Disease 1).
## Area under the curve: 0.9753
```

```
auc(Disease ~ CK, data = roc_ami)
```

```
## Setting levels: control = 0, case = 1
```

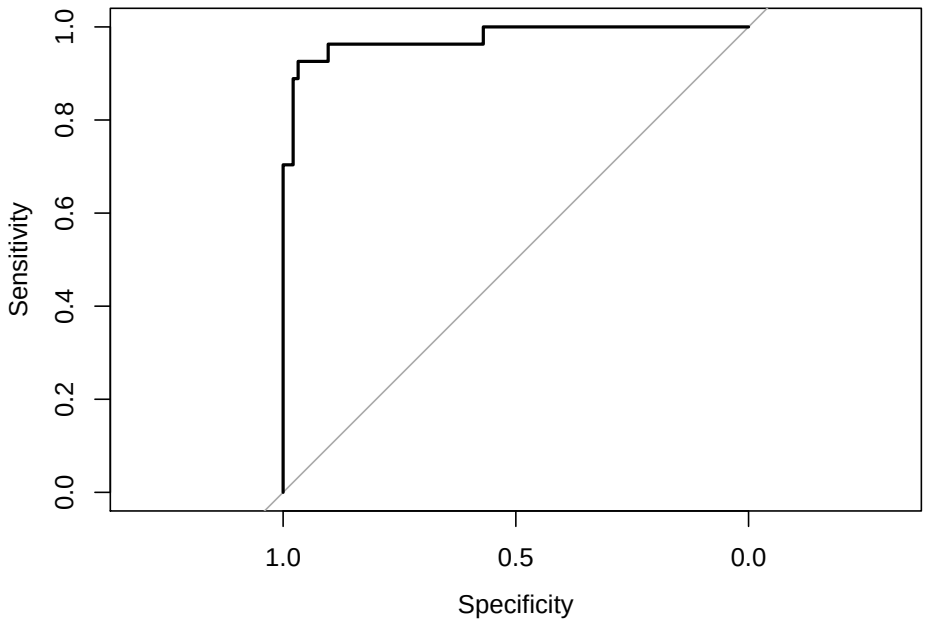
```
## Setting direction: controls < cases

## Area under the curve: 0.9753
```

```
ci(ami_roc_curve)
```

```
## 95% CI: 0.9424-1 (DeLong)
```

```
ami_roc_plot <- roc(Disease ~ CK, data = roc_ami, plot = T)
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```



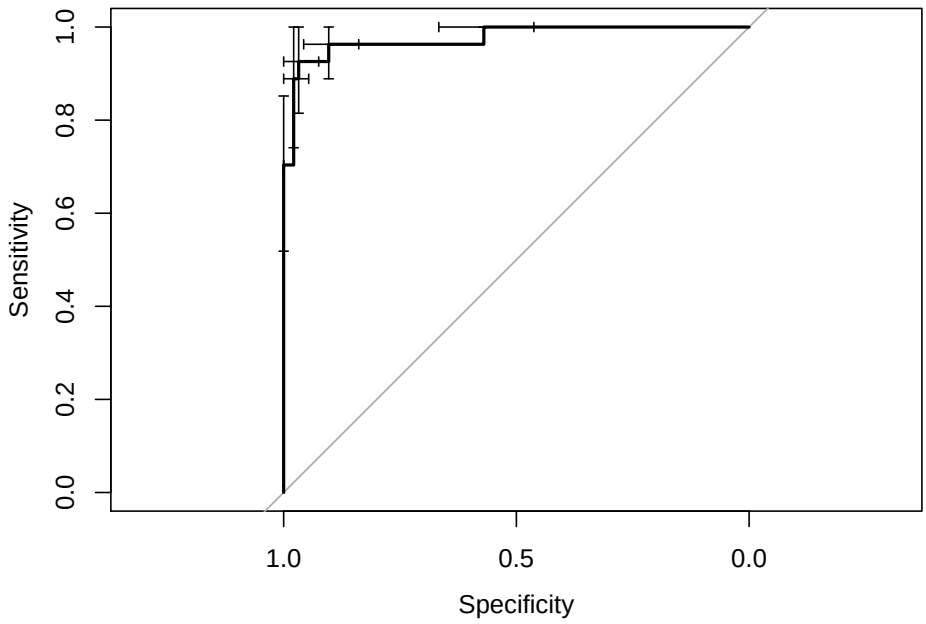

```
ami_roc_plot
```

```
##
## Call:
## roc.formula(formula = Disease ~ CK, data = roc_ami, plot = T)
##
## Data: CK in 93 controls (Disease 0) < 27 cases (Disease 1).
## Area under the curve: 0.9753
```

```
plot.roc(Disease ~ CK, data = roc_ami, ci = T, of = "thresholds")
```

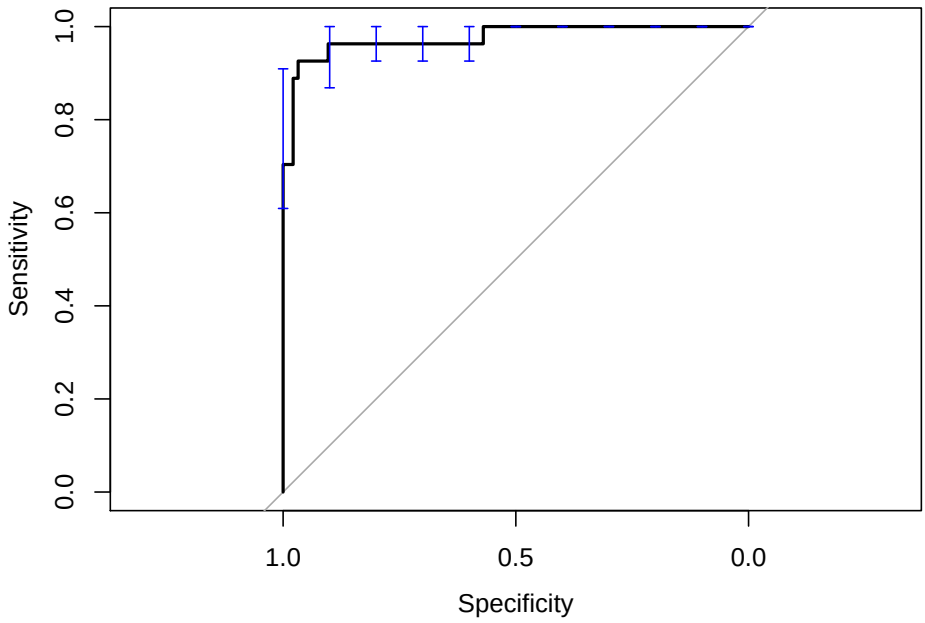
```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```



Compute the confidence interval (CI) of the sensitivity at the given specificity points

```
ci1 <- ci.se(ami_roc_curve, boot.n = 10)

plot(ami_roc_curve)

plot(ci1, col = "blue")
```




```
coords(ami_roc_plot, x = "all")
```

```
## Warning in coords.roc(ami_roc_plot, x = "all"): The 'transpose'
## argument to FALSE by default since pROC 1.16. Set transpose = TRUE
## explicitly to revert to the previous behavior, or transpose = TRUE
## to silence this warning. Type help(coords_transpose) for additional
## information.

##    threshold specificity sensitivity
## 1       -Inf  0.00000000  1.00000000
## 2       28.0  0.01075269  1.00000000
## 3       34.5  0.02150538  1.00000000
## 4       36.5  0.03225806  1.00000000
## 5       39.0  0.05376344  1.00000000
## 6       41.5  0.08602151  1.00000000
## 7       42.5  0.10752688  1.00000000
## 8       44.0  0.11827957  1.00000000
## 9       46.0  0.12903226  1.00000000
## 10      47.5  0.13978495  1.00000000
## 11      48.5  0.17204301  1.00000000
## 12      50.5  0.18279570  1.00000000
## 13      52.5  0.21505376  1.00000000
## 14      53.5  0.22580645  1.00000000
## 15      55.5  0.23655914  1.00000000
## 16      57.5  0.25806452  1.00000000
```

```
## 17      59.0  0.29032258  1.00000000
## 18      61.0  0.32258065  1.00000000
## 19      62.5  0.33333333  1.00000000
## 20      64.0  0.35483871  1.00000000
## 21      65.5  0.37634409  1.00000000
## 22      66.5  0.38709677  1.00000000
## 23      69.0  0.39784946  1.00000000
## 24      71.5  0.40860215  1.00000000
## 25      72.5  0.43010753  1.00000000
## 26      74.0  0.45161290  1.00000000
## 27      77.5  0.46236559  1.00000000
## 28      81.5  0.48387097  1.00000000
## 29      83.5  0.49462366  1.00000000
## 30      84.5  0.50537634  1.00000000
## 31      85.5  0.51612903  1.00000000
## 32      87.0  0.52688172  1.00000000
## 33      88.5  0.55913978  1.00000000
## 34      89.5  0.56989247  1.00000000
## 35      90.5  0.56989247  0.96296296
## 36      92.5  0.58064516  0.96296296
## 37      94.5  0.60215054  0.96296296
## 38      96.0  0.61290323  0.96296296
## 39      98.5  0.62365591  0.96296296
## 40     101.5  0.63440860  0.96296296
## 41     103.5  0.64516129  0.96296296
## 42     104.5  0.65591398  0.96296296
## 43     106.0  0.67741935  0.96296296
## 44     107.5  0.68817204  0.96296296
## 45     108.5  0.69892473  0.96296296
## 46     110.0  0.70967742  0.96296296
## 47     112.5  0.72043011  0.96296296
## 48     115.0  0.73118280  0.96296296
## 49     117.0  0.74193548  0.96296296
## 50     119.5  0.75268817  0.96296296
## 51     121.5  0.77419355  0.96296296
## 52     124.0  0.78494624  0.96296296
## 53     128.0  0.79569892  0.96296296
## 54     134.5  0.81720430  0.96296296
## 55     144.5  0.82795699  0.96296296
## 56     152.5  0.83870968  0.96296296
## 57     156.0  0.84946237  0.96296296
## 58     159.5  0.86021505  0.96296296
## 59     169.0  0.87096774  0.96296296
## 60     178.0  0.88172043  0.96296296
## 61     184.0  0.89247312  0.96296296
## 62     192.0  0.90322581  0.96296296
```

```
## 63     197.0  0.90322581  0.92592593
## 64     212.0  0.91397849  0.92592593
## 65     229.0  0.92473118  0.92592593
## 66     244.5  0.93548387  0.92592593
## 67     277.0  0.95698925  0.92592593
## 68     299.5  0.96774194  0.92592593
## 69     304.5  0.96774194  0.88888889
## 70     309.0  0.97849462  0.88888889
## 71     318.0  0.97849462  0.85185185
## 72     330.0  0.97849462  0.81481481
## 73     341.0  0.97849462  0.77777778
## 74     348.0  0.97849462  0.74074074
## 75     350.0  0.97849462  0.70370370
## 76     355.5  0.98924731  0.70370370
## 77     361.5  1.00000000  0.70370370
## 78     370.0  1.00000000  0.66666667
## 79     383.5  1.00000000  0.62962963
## 80     394.0  1.00000000  0.59259259
## 81     471.5  1.00000000  0.55555556
## 82     561.0  1.00000000  0.51851852
## 83     603.0  1.00000000  0.48148148
## 84     638.5  1.00000000  0.44444444
## 85     771.0  1.00000000  0.40740741
## 86     928.0  1.00000000  0.37037037
## 87     988.5  1.00000000  0.33333333
## 88    1079.0  1.00000000  0.29629630
## 89    1300.5  1.00000000  0.25925926
## 90    1706.5  1.00000000  0.22222222
## 91    2047.0  1.00000000  0.18518519
## 92    2169.5  1.00000000  0.14814815
## 93    2622.0  1.00000000  0.11111111
## 94    5317.0  1.00000000  0.07407407
## 95    9364.0  1.00000000  0.03703704
## 96       Inf  1.00000000  0.00000000
```

1.5 Variable selection

1.5.1 Davison 2003, Statistical Models

pp. 400-401

Automatic variable selection methods can fit complicated models to random data.

They are widely used but have no theoretical basis.

The rules used for deciding which variables to include vary and are arbitrary - various cut-offs for t or F values are used, for example; sometimes AIC is used.

These methods can be useful for screening.

Knowledge of the system being studied is essential in model building.

p. 402

Some measure of how well a model fits is a reasonable criterion to use.

Residual SS decreases the more terms are added, so this is not satisfctory.

A useful approach is to penalise a model according to its complexity and balance this against a measure of how well the model fits the data.

AIC is one approach of this type. It provides a balance between the fit of the model and its simplicity.

AICc, the corrected AIC, is best when the number of parameters is comparable to the number of rows of data (i.e. p is comparable to n).

p. 404

There may be a number of models with similar AICcs. If a single model is needed, use expert knowledge to choose it, if possible.

If more than one model is plausible, then accept that as the conclusion and discuss them.

p. 405

If automatic variable selection methods are used, making inferences after this model selection is restricted. The only covariates for which standard confidence intervals are reliable are the ones where the evidence that they be included in the model is very strong.

1.5.2 Faraway 2014, Linear Models with R

p. 153

Do not use testing-based variable selection methods.

With methods that are based on P values, so much multiple testing results in P values losing their usual meaning so that the overall process lacks validity.

The testing-based procedures are not directly related to the overall objectives, namely, to find the best prediction or explanatory model. Variable selection inflates the statistical significance of variables that are kept in the model. Variables that are dropped are not necessarily unrelated to the response variable; they just do not contribute any extra explanatory value over the ones already in the model. *The point of this is a bit unclear. What do other writers say about this?*

p. 159

Use information-criterion-based methods for model selection (e.g. AIC, BIC, adjusted R^2, Mallow's C_p)

If a number of models fit about the same as one another, accept it! Look at whether

- they each provide different but plausible explanations of the response
- they give similar predictions
- one or some have better model diagnostic features
- one or some include predictors that are cheaper or easier to measure

p. 161

Problems in having too many predictors in a regression model include:

- collinearity - this can impede how well a model explains the response
- the quality of predictions from the model can be reduced (Faraway probably means by this that the variance of predictions is increased)

The apparent contradiction is that more predictors should mean more information! There are methods that help to *shrink* this extra information into a useful form:

- PCA
- partial least squares (p. 172)
  - this constructs orthogonal linear combinations of predictors as new predictors
  - these combinations are explicitly constructed to predict Y as well as possible
- ridge regression (p. 174)
  - it produces coefficients which are not very large, which is reasonable if you have a lot of candidate predictors and you think many of them have an effect on the response
  - this shrinks coefficients, which are normalised (i.e. centred by their means and scaled by their SDs), towards zero
  - it's a form of penalised regression
- LASSO

1.5.3 Faraway 2016, Extending the Linear Model with R, 2nd ed.

p. 203

For linear mixed models, information-based criteria can be used. However, there are some problems with LMMs:

- because of dependence in the data, the effective sample size is less than the total number of cases
- it isn't clear how to count the number of fixed effect parameters and random effect parameters together
- most information-criteria measures are based on the likelihood, which does not behave well when parameters are estimated at the boundary of the parameter space as can happen with variance components

The AIC can be used when the models to be compared differ only in their fixed effects. If random effects also vary, how to count the number of parameters is a problem.

The BIC, which replaces the penalty of 2 $\times$ no. of parameters in the AIC with the term, $p\log(n)$, can also be used. It penalises larger models more heavily than the AIC and favours smaller models than the AIC.

1.5.4 Weisberg 2014, Applied Linear Regression, 4th ed.

pp. 234-235

Methods for selecting variables in regression modelling depend on the purpose of the analysis:

- *what is the effect of a focal predictor?* - examine the effect of a *focal* predictor, or a few predictors, on a response - including extra predictors apart from those of interest could help the interpretation or increase the precision of tests and estimates; including too many could decrease the precision
- *which of a number of potential predictors are active?* - determine the predictors that are associated with the response - i.e. distinguish the *active* predictors from the *inactive* ones; (estimating coefficients and testing whether coefficients are different from zero are not the main point of "variable discovery", p. 242)

- *predict values of a response using the predictors* - including too many predictors could produce inaccurate predictions because too much detail in the observed data is incorporated (i.e. overfitting); not including enough predictors can produce inaccurate predictions if important predictors are excluded (i.e. biased predictions, as mentioned by someone below)

Looking for active predictors

pp. 237ff

One approach is looking at groups of predictors and picking one that maximises some selection criterion.

- Common criterion is AIC (as well as BIC).

p. 239

- Stepwise methods
  - these methods are not guaranted to give you the model with the optimal value of the selection criterion but they can be useful (p. 240)
  - estimating coefficients and testing whether coefficients are different from zero are not the main point of "variable discovery" (p. 242)
  - after selecting a subset of variables, the t values and associated P values for coefficients *cannot* be trusted (as the t values may not follow a t distribution)
- methods that select subsets exaggerate significance of, for example, F statistics comparing models and t statistics for coefficients in models

p. 244

- Regularised methods - LASSO (incl. variant, elastic net)
  - work on the basis that as few predictors as possible are needed to model a response
  - work well on randomly sampled data and model with no factors or interaction

p. 245

Developing a model for prediction

Possible approaches

- model averaging (incl. Bayesian model averaging) - v. Hoeting tutorial

- cross-validation
- general question of getting predictions from training data has led to development of *machine learning;* methods include neural networks, random forests, ...

1.5.5 Harrell 2015, Regression Modeling Strategies, 2nd ed.

pp. 67-68

Stepwise methods of variable selection have no statistical justification.

They result in:

- R^2 values that are too high
- F and χ^2 statistics that are not distributed according to those distributions
- P values that are too small
- SEs that are too small
- model coefficients that are too large

p. 69-70

Do not use stepwise variable selection methods; use full-model fits or data reduction methods instead.

If you have to use stepwise variable selection:

- first of all, fit the full model and do a global test of no regression
- if this global test is not significant, it is not justifiable to pick out individually significant predictors
- if the global test is significant, use a stopping rule based on the AIC - i.e. smallest AIC (but there are no stopping rules for data-driven variable selection); if someone insists on using a stopping rule based on P values, use $\alpha = 0.5$
- use backwards elimination rather than forwards selection
  - it ensures you see the full model, which is the only one with accurate SEs, residual MS and P values
  - Lawless and Singhal's method (implemented with fastbw in rms package) is very efficent

p. 70

Bootstrapping can help you choose between the full and a reduced model.

pp. 71-72

Sometimes variables can be grouped (by subject matter or empirical correlations), tested as a group and kept or removed.

Possibly the most accurately measured variable in a group can be retained (v. p. 79f).

Lasso is a useful penalised estimation technique - it forces some coefficient estimates to be zero and in this way carries out variable selection, while also, however, shrinking the remaining coefficients to take account of the overfitting caused by using a data-based model selection method

Screening each variable singly and using only individually significant variables in a multivariable model can miss variables that become important only after adjusting for other variables.

Sample size, overfitting ...

"Overfitting" happens when the model fits some of the noise and not just the signal; it also happens when you find questionable associations between explanatory variables and the response variable.

A good general recommendation where there is a continuous response variable, for example, is to have no more than one candidate predictor variable per 15 observations (this includes interaction terms, if any). ??? df or terms?

If there are explanatory variables with a narrow range of values, you'll need a larger sample than the above recommendation.

p. 78 (In section on "shrinkage")

Penalised methods are very good ways to deal with the "too many variables, too little data" problem (more detail in s. 9.10).

Some useful shrinkage methods are:

- ridge regression
- penalised ML estimation

Cross-validation or AIC has to be used to choose the penalty factor.

p. 78 *Collinearity*

When one predictor can be predicted well from the others or expressed as a combination of the others, collinearity is present (i.e. if $\mathbf{Xa} = \mathbf{0}$, where $\mathbf{X}$ is the incidence matrix and $\mathbf{a}$ is a vector of constants).

This can inflate the SEs of the regression coefficients.

It can also make selecting "important" variables unreliable.

When groups of highly correlated variables are identified, they can be tested as a whole set with a multiple d.f. test, rather than using a one d.f. test on a single predictor.

To quantify collinearity, use the variance inflation factor.

p. 79 *Data reduction*

- Use expert knowledge and previous research to remove unimportant variables from the candidate variables
- Remove variables that have distributions that are too narrow
- Remove variables that have a large number of missing values

pp. 89-90

Recommended approaches

- Fit a full model without removing non-significant predictors
- Use data reduction methods to reduce dimensionality
- Use shrinkage (i.e. penalised estimation) to fit a large model and not worry about sample size

pp. 94ff *Harrell's summary*

Developing models to find active predictors

- Collect as much data as possible; ensure predictor variables have wide distributions
- Frame good hypotheses that specify predictor variables that are relevant, including interactions, if appropriate - do *not* use the response variable in this process of specifying predictors either graphically, using descriptive statistics or doing hypothesis tests of estimates (i.e. use expert knowledge to "screen" potential candidate predictors)
- Impute values as appropriate - see text for details
- Either use cross-validation or bootstrapping, or hold out some sample data as test data

The following steps need to be carried out *on each bootstrap or cross-validation sample.*

- When you can test the model for complexity in a structured way, you might be able to simplify it without using penalisation methods - e.g. test a group of predictors based on one P value (which is hopefully outside range of 0.05 to 0.2)
- Check additivity by testing all prespecified interaction terms; if the test is unequivocal (e.g. $P > 0.3$), drop all interaction terms.

Developing models for estimating effects

These are models for getting point or interval estimates for the response at various combinations of the predictors

- Parsimonious models are less important here, as including all predictors means there is a greater chance the CI for the effect of interest actually is the size stated; on the other hand, including unimportant predictors increases the variance of predicted values

Developing models for testing hypotheses

Very similar strategy as for models for estimating effects

- Parsimony has little importance as a full model fit, with non-significant variables, gives more accurate P values for variables of interest
- It is important to consider interactions; a well-defined test is whether an interaction should be included or not
- Model validation is not necessary

p. 118 *Simplifying the final model by approximation*

A model with all prespecified terms usually produces the most accurate predictions on new data

Predictions are conditional on all predictors

If a predictor is not significant, predictions can be obtained by using weights, based on the fraction of values of each level of that predictor - if a factor - in the data, to make the predictions *unconditional* on that predictor (e.g. average over categories of ethnicity).

If there are several non-significant predictors that need to be averaged over in the above way, the process is unwieldy.

What to do about this is a bit over the top!

1.5.6 David Warton's draft book, chapter 5

p. 121

Variable selection or model selection is a type of inference. Inference can take various forms, including:

- hypothesis testing - seeing whether data are consistent with a particular hypothesis

- parameter estimation - finding a plausible range of values (confidence interval) for a parameter of interest
- variable (model) selection - finding the set of predictor variables that characterise the true process by which the data are generated (inference by using a sample to draw general conclusions about how well a set of explanatory variables can predict)

pp. 118-119

Purpose of model selection:

- maximise predictive capability - i.e. find the model that would give best predictions of the response given new data

You need to choose the "right" number of predictors - i.e. the "right" level of model complexity.

A model that is too small gives biased predictions.

A model that is too big results in predictions with increased variance. This is because the model fits noise as well as signal.

Bias-variance trade-off is inevitable. How find the right balance? It depends on:

- how much data you've got
- how relatively complex the models being compared are
- how strong the signal is in each model

R-squared values cannot take account of how complex models are.

Using P values to remove non-significant predictors is not appropriate as there is no real a priori hypothesis being tested.

p. 121 (again)

Comparing a lot of models is of doubtful value as using the data alone to produce the best model is unreliable.

It's important to simplify potential models - use expert knowledge to remove unnecessary predictors.

Comparing predictive models

The easiest way to compare models is to see how well they predict from new data - this is "validation". Get new data or split data into a training set and a test set - the split needs to be random, the two sets must be independent.

- (K-fold) cross-validation can be used here (p. 125) - split the data into K groups and use each group once as the test data, fitting K models in all; in

each modelling run, hold out the particular test group, fit the model to the remaining $K-1$ groups combined and then compare the model predictions to the test observations; pool measures of predictive performance across all the runs.

- How do you choose the size of the training sample? Maybe along the lines of Shao's method - as sample size increases, use a *larger number* of observations but a *smaller proportion* of the total sample size. Some suggestions:
  - leave-one-out (i.e. N-fold c.v.) if n < 20
  - 10-fold c.v. for $20 < n < 100$
  - 5-fold c.v. for $n > 100$
- How do you measure predictive performance?
  - For linear regression (assuming constant variance) use mean squared error
  - Use the whole data set (no splits) and penalise larger models by means of an *information criterion* - e.g. AIC, BIC (p. 127)
  - what about the GIC ???
  - AIC overfits and is good if you want a prediction model; BIC does not overfit and is good for variable selection
- Information criteria - advantages and disadvantages
  - no random split in data so same answer and simpler to interpret
  - less intuitive than cross-validation as they judge predictive capability on new data indirectly (as there is no new data!)
  - they rely on the model being close to the correct model; cross-validation requires only that the training and test data are independent

p. 130 *Subset selection*

No clear winner among forwards, backwards or all subsets methods

- All subsets
  - more comprehensive but because so many models may be involved, there's no guarantee it will yield best model
  - if there are lots of explanatory variables, it's not practical
- Backwards elimination
  - not a good idea if there are a large number of explanatory variables, as full model could be quite unstable with some parameters that are not well estimated

- Forwards selection
  - with lots of explanatory variables, this might be a better place to start
  - not a good method if the optimal model ends up with a lot of terms, as there are lots of steps in getting to that model where things can go wrong
- Multi-collinearity

This can undermine how stepwise methods work as the chance of a particular term being added to the model drops substantially if it's correlated with a term already in the model

p. 134 *Penalised estimation*

A more modern and good way to do subset selection - e.g. LASSO

Parameter estimates are pushed towards zero - this introduces bias but reduces variance, so there's a bias-variance trade-off here

Penalised estimation is good when:

- you want to predict the response, as this method reduces variance of predictions
- you have lots of terms in your model or not a large sample relative to the number of terms

LASSO does variable selection by forcing some parameter estimates to be zero - if the parameter estimate for a term is zero, the terms drops out of the model

LASSO - advantages and disadvantages

- it does model selection as part of its routine and simplifies this model selection to estimating a single nuisance parameter that is part of the penalty term
- it produces good predictions by reducing variance
- parameter estimates are, unfortunately, biased
- standard errors are not easily available

1.5.7 James et al. Introduction to Statistical Learning, 2017 (corrected printing)

p. 30 *Assessing model accuracy*

For linear regression models, a common way of measuring how well the predictions a model makes match the observed data is to use the *residual mean square* or *mean squared error.*

This says how well the model works on the training data used to fit the model (i.e. the *training data*).

What's more interesting and important is how well the model would perform on new data (i.e. *test data*).

For example, suppose we have clinical measurements on a group of patients (e.g. age, BP, gender, height, weight) and whether each patient has diabetes. We can develop a model using this data to predict risk of diabetes. We are particularly interested whether this model accurately predicts the risk of diabetes for *future patients.*

How can you minimise the test MSE?

If test data is available, use this test data set and pick the method that gives smallest MSE.

If no test data is available, you could use the method that gives the smallest MSE for the training data .... But no! Lots of modelling methods specifically minimise the MSE in the fitting process; these methods can give quite a small MSE for the training set but give a much larger MSE on test data.

p. 32 - direct quote

> ... as the flexibility of the statistical learning method increases, we observe a monotone decrease in the training MSE and a U-shape in the test MSE. This is a fundamental property of statistical learning that holds regardless of the particular data set at hand and regardless of the statistical method being used. As model flexibility increases, training MSE will decrease, but the test MSE may not. When a given method yields a small training MSE but a large test MSE, we are said to be overfitting the data. This happens because our statistical learning procedure is working too hard to find patterns in the training data, and may be picking up some patterns that are just caused by random chance rather than by true properties of the unknown function f.

p. 34

Note explanation of *bias* and *variance* in context of modelling (or statistical learning).

Variance is the amount by which estimated response values would change using different training data. See example in test.

Bias is the error caused by approximating a possibly quite complex real-life system by a much simpler model.

As the flexibility of a model increases, the variance increases and the bias decreases.

p. 75

Important questions when doing multiple linear regression:

- is at least one predictor useful in predicting response?
- do all predictors, or only some of them, help to explain the response?
- how well does the model fit the data?
- given specific values for predictors, what is the predicted response value and how accurate is it?

pp. 79 and 208

If $p > n$, backward elimination cannot be used, as you cannot fit the full model; forward selection can always be used but if $p > n$, you cannot get a unique solution and can only fit submodels.

p. 92

Brief overview of potential problems with regression model:

- non-linearity of relationships between response and predictor
- correlation of errors (non-independence)
- non-constant variance of errors
- outliers
- points with high leverage
- collinearity - note very interesting plots showing effect of collinearity on estimated coefficients - fig. 3.15

Chapter 6 Linear model selection

pp. 205-210

All subsets, forward selection, backward elimination and hybrid approaches are outlined.

Chapter 2

ggplot

2.1 Vignettes

Extending ggplot2

Aesthetic specifications

Cheat sheet

2.2 Confidence bands

https://janhove.github.io/reporting/2017/05/12/visualising-models-2

2.3 Confidence bands: fitted models

Part of this comes from Gordana's code for calculating sample size by simulation

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:nlme':
##
##     collapse
```

```
## The following object is masked from 'package:MASS':
##
##     select

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(emmeans)
library(ggplot2)
library(ggpubr)


data(iris, package = "datasets")

str(iris)
```

```
## 'data.frame': 150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 .
```

```
table(iris$Species)
```

```
##
##     setosa versicolor  virginica
##         50         50         50
```

```
iris_versi <- filter(iris, Species == "versicolor")

summary(iris_versi)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length   Petal.Width
##  Min.   :4.900   Min.   :2.000   Min.   :3.00   Min.   :1.000
##  1st Qu.:5.600   1st Qu.:2.525   1st Qu.:4.00   1st Qu.:1.200
##  Median :5.900   Median :2.800   Median :4.35   Median :1.300
##  Mean   :5.936   Mean   :2.770   Mean   :4.26   Mean   :1.326
```

```
##  3rd Qu.:6.300   3rd Qu.:3.000   3rd Qu.:4.60   3rd Qu.:1.500
##  Max.   :7.000   Max.   :3.400   Max.   :5.10   Max.   :1.800
##        Species
##  setosa    : 0
##  versicolor:50
##  virginica : 0
##
##
##
```

```
iris_versi$Species <- factor(iris_versi$Species)

nsamp <- 12

iris_versi_samp <- sample(1:50, size = nsamp, replace = F)

pilot <- iris_versi[iris_versi_samp, ]


##  Gordana's simulation code

pilot_mod <- lm(Sepal.Length ~ Petal.Length, data = pilot)

summary(pilot_mod)
```

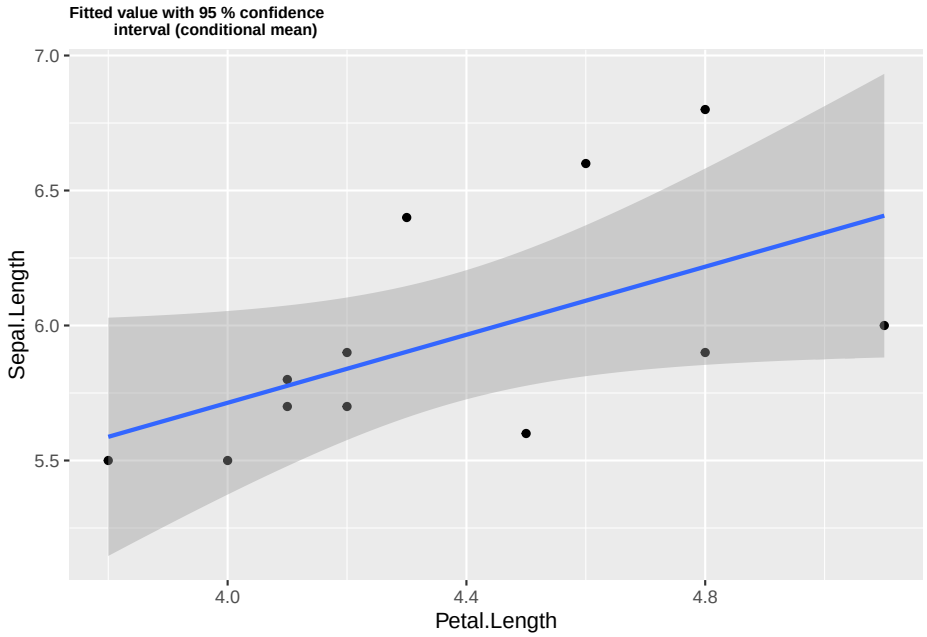
```
##
## Call:
## lm(formula = Sepal.Length ~ Petal.Length, data = pilot)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.42877 -0.23973 -0.08219  0.16952  0.58219
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    3.1932     1.2716   2.511   0.0309 *
## Petal.Length   0.6301     0.2896   2.176   0.0546 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3712 on 10 degrees of freedom
## Multiple R-squared:  0.3213,Adjusted R-squared:  0.2534
## F-statistic: 4.734 on 1 and 10 DF,  p-value: 0.05464
```

```
confint(pilot_mod)
```

```
##                    2.5 %  97.5 %
## (Intercept)   0.35974146 6.02656
## Petal.Length -0.01519612 1.27547
```

```
fit_gg <- ggplot(pilot, aes(x = Petal.Length,
                            y = Sepal.Length)) +
  geom_point() +
  geom_smooth(method = "lm") +
  ggtitle("Fitted value with 95 % confidence
          interval (conditional mean)") +
  theme(plot.title = element_text(size = 8, face = "bold"))

print(fit_gg)
```

```
## `geom_smooth()` using formula 'y ~ x'
```



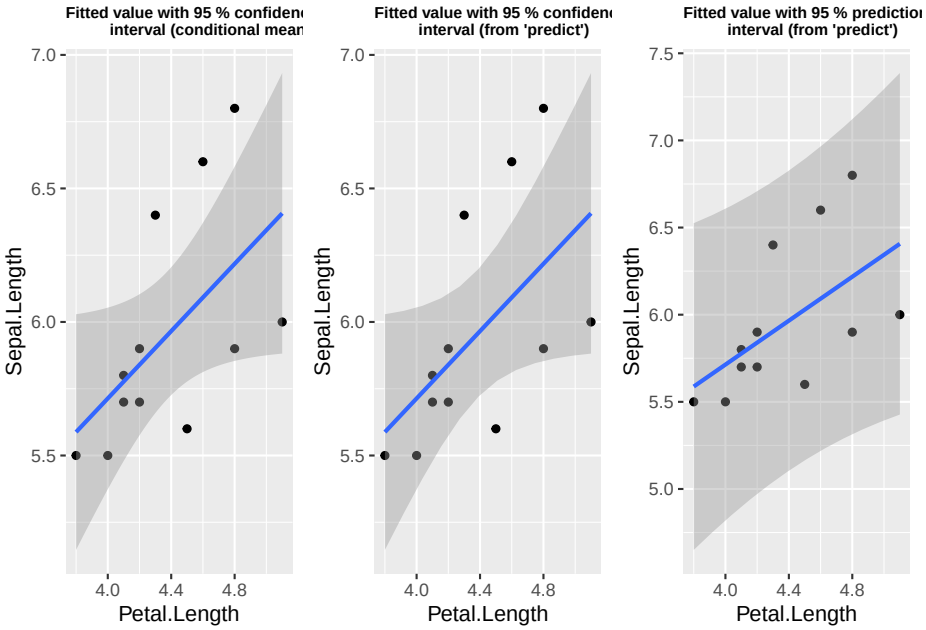
Check type of band (confidence or prediction) on above ggplot

```
# range of petal length
my_range <- range(pilot$Petal.Length)
```

```r
X_pred <- seq(my_range[1], my_range[2], length = 12)

yhat_conf <- data.frame(
  predict(pilot_mod, interval = "confidence",
          newdata = data.frame(Petal.Length = X_pred)))

names(yhat_conf) <- c("yhat_conf_fit", "yhat_conf_lwr", "yhat_conf_upr")


yhat_pred <- data.frame(
  predict(pilot_mod, interval = "prediction",
          newdata = data.frame(Petal.Length = X_pred)))

names(yhat_pred) <- c("yhat_pred_fit", "yhat_pred_lwr", "yhat_pred_upr")


yhat_all <- data.frame(Petal.Length = X_pred, yhat_conf, yhat_pred[ , -1])

names(yhat_all)[names(yhat_all) %in% "yhat_conf_fit"] <- "Sepal.Length"

##  Add to sample data

pilot$X_pred <- X_pred

pilot <- data.frame(pilot, yhat_conf)

pilot <- data.frame(pilot,
                    yhat_pred_lwr = yhat_pred$yhat_pred_lwr,
                    yhat_pred_upr = yhat_pred$yhat_pred_upr)

##  Plot fitted values and intervals from "predict" function

fit_conf <- ggplot(pilot, aes(x = Petal.Length, y = Sepal.Length)) +
  geom_point() +
  geom_smooth(
    data = yhat_all, aes(ymin = yhat_conf_lwr,
                         ymax = yhat_conf_upr),
    stat = "identity") +
  ggtitle("Fitted value with 95 % confidence
          interval (from 'predict')")+
  theme(plot.title = element_text(size = 8, face = "bold"))


fit_pred <- ggplot(pilot, aes(x = Petal.Length, y = Sepal.Length)) +
  geom_point() +
```

```
  geom_smooth(data = yhat_all, aes(ymin = yhat_pred_lwr,
                                   ymax = yhat_pred_upr),
              stat = "identity") +
  ggtitle("Fitted value with 95 % prediction
          interval (from 'predict')")+
  theme(plot.title = element_text(size = 8, face = "bold"))

ggarrange(fit_gg, fit_conf, fit_pred, ncol = 3)
```

```
## `geom_smooth()` using formula 'y ~ x'
```



2.4 Histogram

2.4.1 Histogram: one vector of values

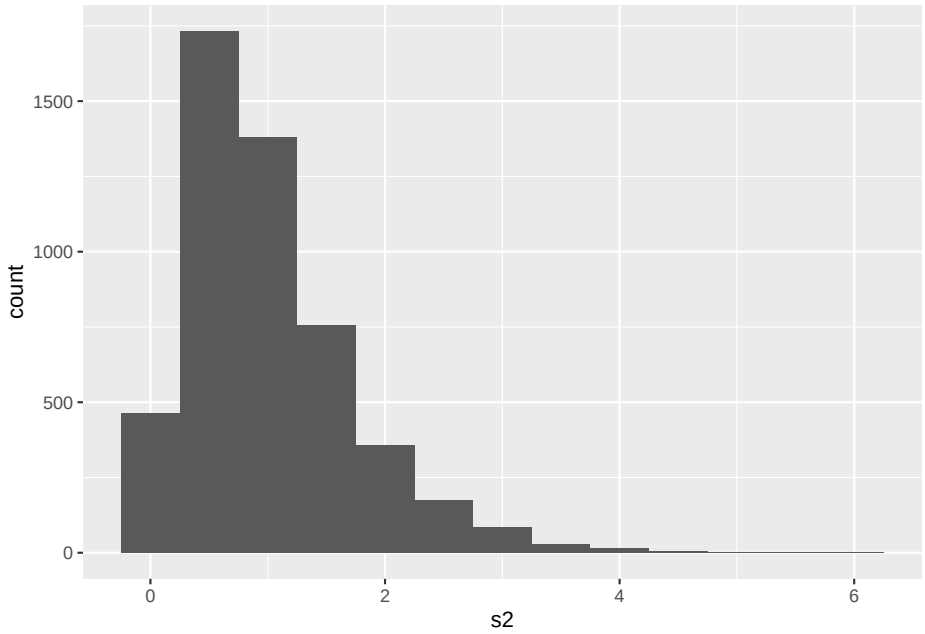
```
##  Histogram of one vector of values in ggplot2

##  Generate a vector of values

s2 <- numeric(5000)
for(i in 1:5000) {
```

```
  dd <- rnorm(5, 10, 1)
  s2[i] <- var(dd)
  rm(dd)
}

summary(s2)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.01141 0.47596 0.84707 1.00066 1.33430 6.07112
```

Data in numeric vector

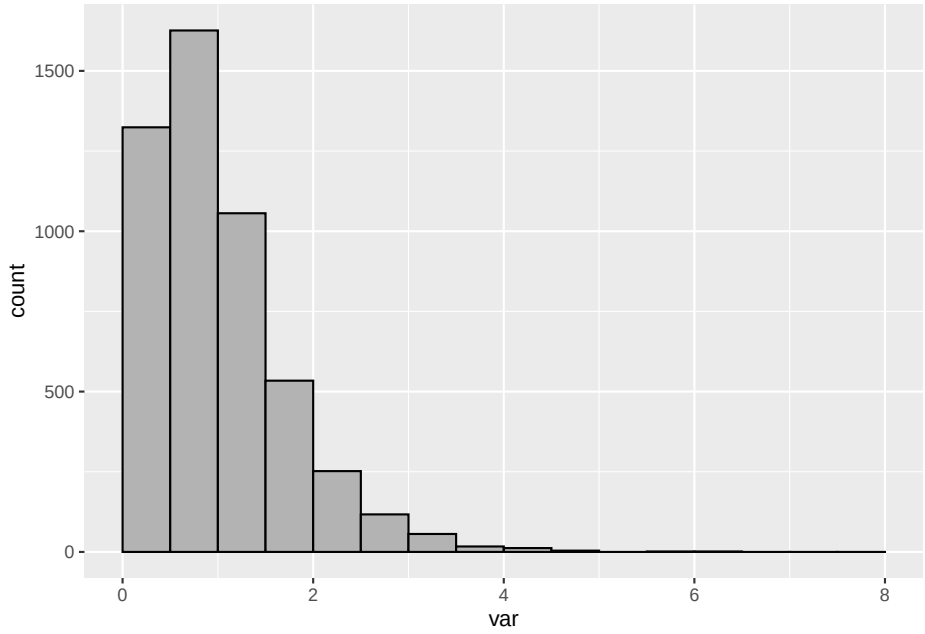
```
##  Data in numeric vector

ggplot() +
  aes(s2) +
  geom_histogram(binwidth = 0.5)
```



Data in single-column data frame - Y axis is frequency

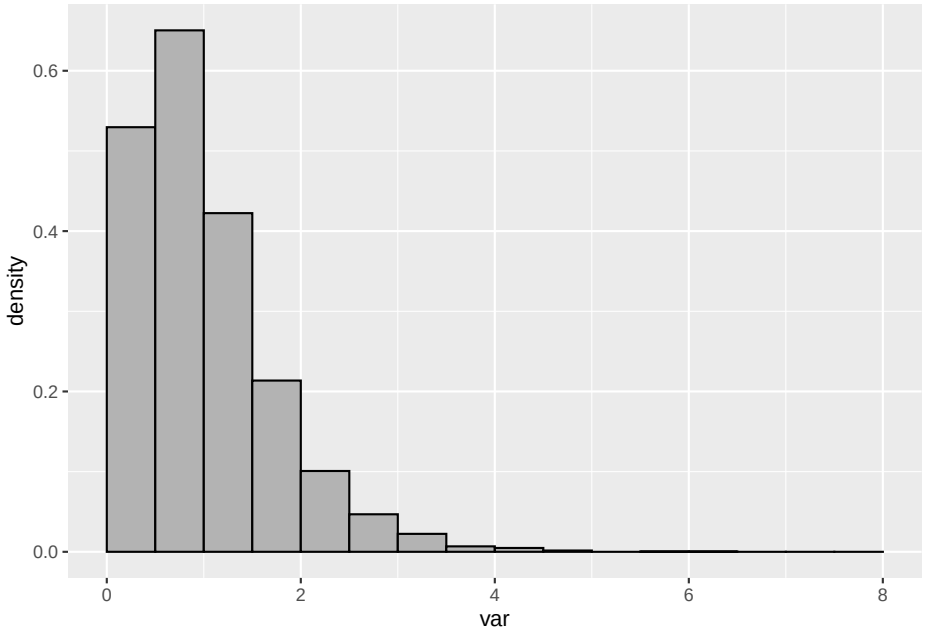
```
##  Data in single-column data frame

s2.df <- data.frame(var = s2)
```

```
##  Y axis is frequency

ggplot(s2.df, aes(x = var)) +
  geom_histogram(breaks = 0:16*0.5, col = "black",
                 fill = "grey70")
```



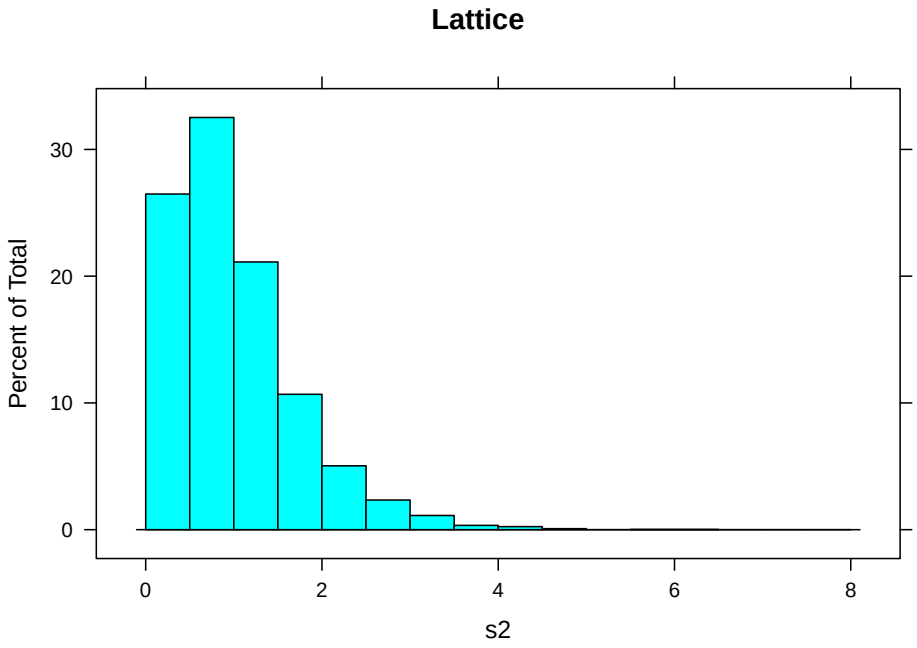
Data in single-column data frame - Y axis is density

```
##  Y axis is density

ggplot(s2.df, aes(x = var, y = ..density..)) +
  geom_histogram(breaks = 0:16*0.5, col = "black",
                 fill = "grey70")
```



For comparison, histogram in lattice

```
##  Lattice

histogram(s2, breaks = 0:16*0.5, right = F, main = "Lattice")
```

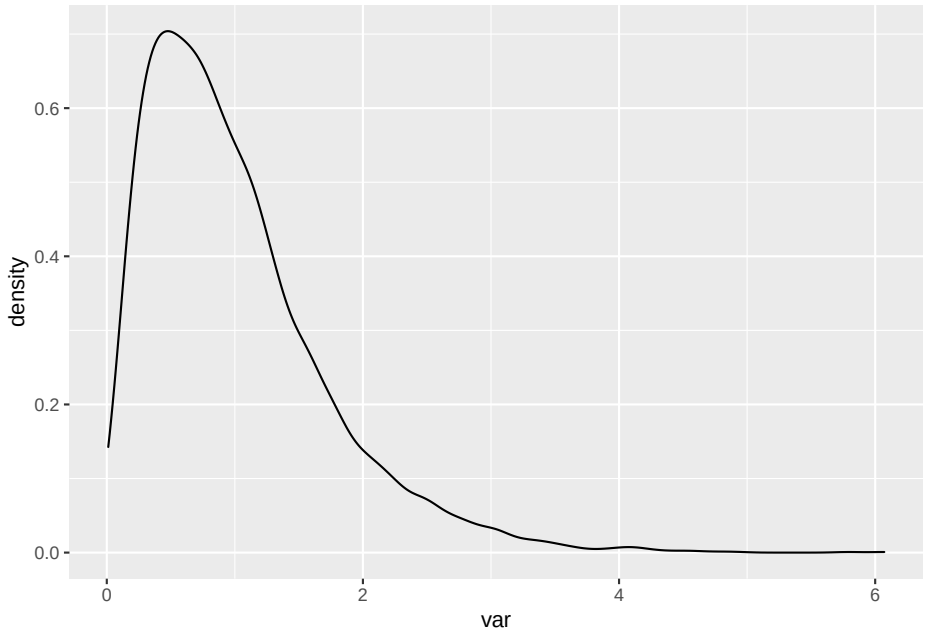


```
range(s2)
```

```
## [1] 0.01141184 6.07112259
```

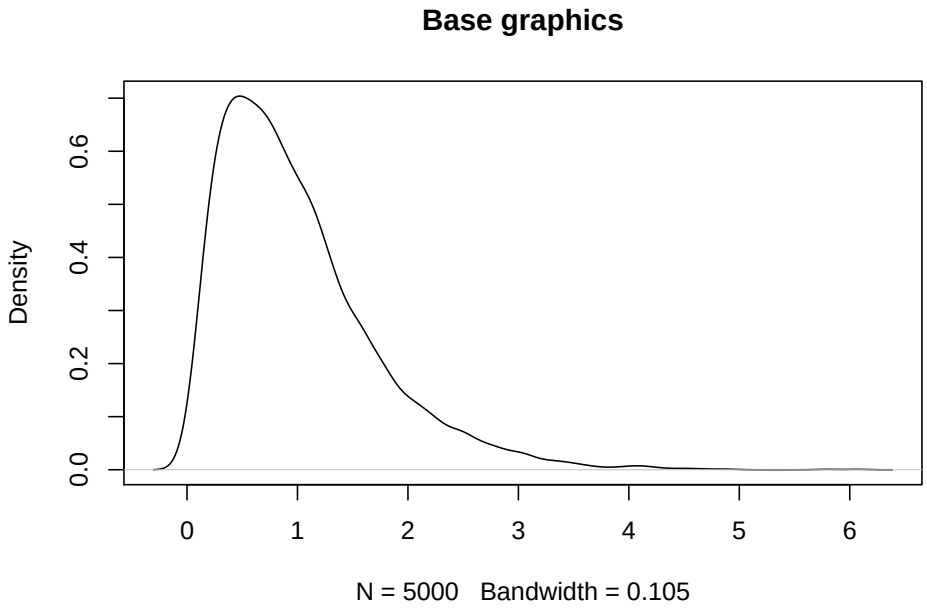
2.5 Density plot

```
##  Density plot

ggplot(s2.df, aes(x = var)) +
  geom_density()
```

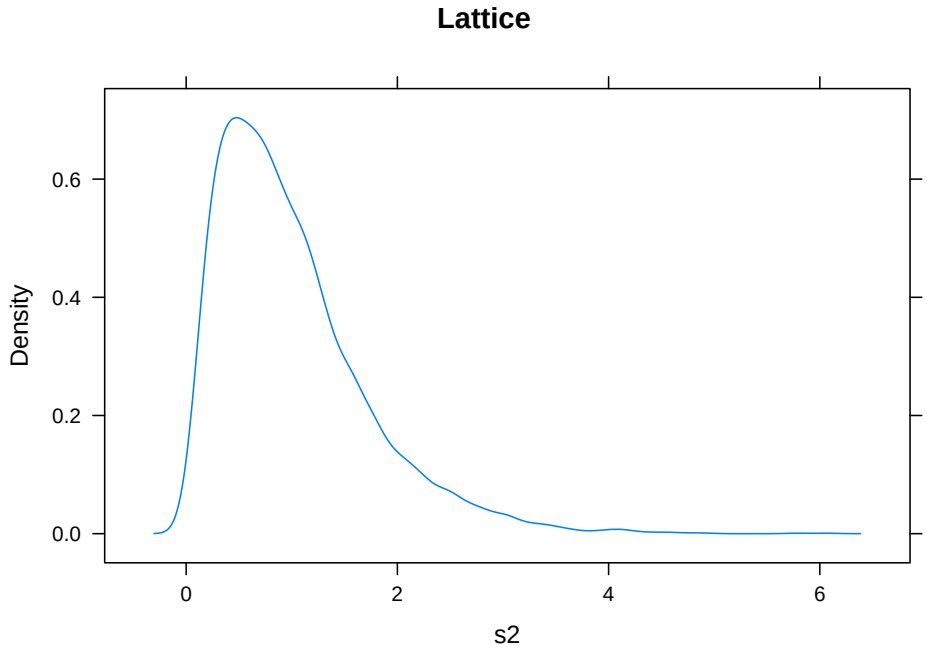


For comparison, density plot in base graphics and in lattice

```
##  Density plot - base graphics

plot(density(s2), main = "Base graphics")
```




```
##  Density plot - lattice

densityplot(s2, plot.points = F, main = "Lattice")
```



2.6 Plot structure

```
ggplot_build(p_obj)$data
```

2.7 Label legend

```
guides(fill = guide_legend(title = "Gender"))
```

2.8 Plot title

```
theme(plot.title = element_text(size = 11))
```

2.9 Three-category colours

```
scale_fill_manual(values =  c("#f8766d", "#00bfc4", "#b79f00")
```

2.10 Set breaks for scale

```
scale_y_continuous(breaks = seq(0, 12, by = 2))
```

2.11 Display table to 2 d.p.

```
mutate_if(is.numeric, ~ round(., 1)) %>%
kable()
```

2.12 Redefine factor levels for an individual plot

```
nt_youth_all_indiv %>%
  mutate(Race.nat = factor(
    Race.nat,
    levels = c("CALD", "Indigenous", "Caucasian", "Not recorded"))) %>%
  group_by(Care_status, Race.nat) %>%
  summarise(n_indiv = n_distinct(MP_id)) %>%
  ggplot(  etc.  )
```

Chapter 3

R Markdown

3.1 ioslides: Footnotes

Add this after yaml header

```
<style>
div.footnotes {
  position: absolute;
  bottom: 0;
  margin-bottom: 10px;
  width: 80%;
  font-size: 0.6em;
}
</style>


<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>

<script>
  $(document).ready(function() {
    $('slide:not(.backdrop):not(.title-slide)').append('<div class=\"footnotes\">');

    $('footnote').each(function(index) {
      var text  = $(this).html();
      var fnNum = (index+1).toString().sup();
      $(this).html(text + fnNum);

      var footnote   = fnNum + ': ' + $(this).attr('content') + '<br/>';
      var oldContent = $(this).parents('slide').children('div.footnotes').html();
      var newContent = oldContent + footnote;
```

```
      $(this).parents('slide').children('div.footnotes').html(newContent);
    });
  });
</script>
```

Example footnote:

```
<footnote content = "Dalgaard 2008, Introductory Statistics with R, Springer; O'Neill e
```

3.2 ioslides: Speaker's notes

Put notes in an html block:

```
<div class="notes">
This is my *note*.

- It can contain markdown
- like this list
</div>
```

Add ?presentme=true to the link for the slides file

Open the file with the augmented link (Chrome works, Safari disables pop-ups).

Press "p" in the top window. Run the slides from the top window, with the other window as the display window.

Chapter 4

Tidyverse

4.1 Stacking and unstacking data

Stack a two-column data frame with `pivot_longer` and then unstack it with `pivot_wider`.

Firstly with `stack` and `unstack`

```
dd1 <- as.data.frame(matrix(rpois(10, 5), nrow = 5))
dd1
```

```
##   V1 V2
## 1  6 10
## 2  5  5
## 3  3  6
## 4  2  5
## 5  7  4
```

```
dd2s <- stack(dd1)
dd2s
```

```
##    values ind
## 1       6  V1
## 2       5  V1
## 3       3  V1
## 4       2  V1
## 5       7  V1
## 6      10  V2
## 7       5  V2
```

```
## 8       6  V2
## 9       5  V2
## 10      4  V2
```

```
unstack(dd2s)
```

```
##   V1 V2
## 1  6 10
## 2  5  5
## 3  3  6
## 4  2  5
## 5  7  4
```

Now with `pivot_longer` and `pivot_wider`.

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v tibble  3.0.3     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
## v purrr   0.3.4
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::collapse() masks nlme::collapse()
## x dplyr::filter()   masks stats::filter()
## x dplyr::lag()      masks stats::lag()
## x dplyr::select()   masks MASS::select()
```

```
dd2pl <- dd1 %>%
  pivot_longer(cols = c("V1", "V2"))
dd2pl
```

```
## # A tibble: 10 x 2
##    name  value
##    <chr> <int>
##  1 V1        6
##  2 V2       10
##  3 V1        5
##  4 V2        5
##  5 V1        3
##  6 V2        6
```

```
##  7 V1        2
##  8 V2        5
##  9 V1        7
## 10 V2        4
```

```
dd3pl <-
  dd2pl %>%
  group_by(name) %>%
  mutate(row = row_number() ) %>%
  tidyr::pivot_wider(names_from = name, values_from = value) %>%
  select(-row)

dd3pl
```

```
## # A tibble: 5 x 2
##      V1    V2
##   <int> <int>
## 1     6    10
## 2     5     5
## 3     3     6
## 4     2     5
## 5     7     4
```

Chapter 5

Linear Models

5.1 ANOVA: Orthogonal factors

Example from Meier, ANOVA (web document)

```
## Create data (skip if not interested) ####
acids <- c(1.697, 1.601, 1.830,
           2.032, 2.017, 2.409,
           2.211, 1.673, 1.973,
           2.091, 2.255, 2.987)
R50 <- rep(c("no", "yes", "no", "yes"), each = 3)
R21 <- rep(c("no", "no", "yes", "yes"), each = 3)
cheddar <- data.frame(R50, R21, acids)
```

```
str(cheddar)
```

```
## 'data.frame': 12 obs. of  3 variables:
##  $ R50  : chr  "no" "no" "no" "yes" ...
##  $ R21  : chr  "no" "no" "no" "no" ...
##  $ acids: num  1.7 1.6 1.83 2.03 2.02 ...
```

```
cheddar
```

```
##    R50 R21 acids
## 1   no  no 1.697
## 2   no  no 1.601
## 3   no  no 1.830
## 4  yes  no 2.032
```

```
## 5  yes  no 2.017
## 6  yes  no 2.409
## 7   no yes 2.211
## 8   no yes 1.673
## 9   no yes 1.973
## 10 yes yes 2.091
## 11 yes yes 2.255
## 12 yes yes 2.987
```

```
table(cheddar$R21, cheddar$R50)
```

```
##
##       no yes
##   no   3   3
##   yes  3   3
```

Get model matrix for two-factor model

```
cheddar.lm1 <- lm(acids ~ R21 + R50)
anova(cheddar.lm1)
```

```
## Analysis of Variance Table
##
## Response: acids
##           Df  Sum Sq Mean Sq F value  Pr(>F)
## R21        1 0.21440 0.21440  2.6526 0.13782
## R50        1 0.65614 0.65614  8.1178 0.01911 *
## Residuals  9 0.72744 0.08083
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
cheddar.lm2 <- lm(acids ~ R50 + R21)
anova(cheddar.lm2)
```

```
## Analysis of Variance Table
##
## Response: acids
##           Df  Sum Sq Mean Sq F value  Pr(>F)
## R50        1 0.65614 0.65614  8.1178 0.01911 *
## R21        1 0.21440 0.21440  2.6526 0.13782
## Residuals  9 0.72744 0.08083
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Sums of squares are the same regardless of the order of R21 and R50 in the model.

Get model matrix.

```
X <- model.matrix(cheddar.lm1)

X
```

```
##    (Intercept) R21yes R50yes
## 1            1      0      0
## 2            1      0      0
## 3            1      0      0
## 4            1      0      1
## 5            1      0      1
## 6            1      0      1
## 7            1      1      0
## 8            1      1      0
## 9            1      1      0
## 10           1      1      1
## 11           1      1      1
## 12           1      1      1
## attr(,"assign")
## [1] 0 1 2
## attr(,"contrasts")
## attr(,"contrasts")$R21
## [1] "contr.treatment"
##
## attr(,"contrasts")$R50
## [1] "contr.treatment"
```

Calculate $\mathbf{X}'\mathbf{X}$

```
t(X) %*% X
```

```
##             (Intercept) R21yes R50yes
## (Intercept)          12      6      6
## R21yes                6      6      3
## R50yes                6      3      6
```

```
solve(t(X) %*% X)
```

```
##             (Intercept)     R21yes     R50yes
## (Intercept)   0.2500000 -0.1666667 -0.1666667
## R21yes       -0.1666667  0.3333333  0.0000000
## R50yes       -0.1666667  0.0000000  0.3333333
```

5.2 Multiple linear regression - conditional and marginal effects

Note: See David's book, ch. 3.

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v tibble  3.0.3     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
## v purrr   0.3.4

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::collapse() masks nlme::collapse()
## x dplyr::filter()   masks stats::filter()
## x dplyr::lag()      masks stats::lag()
## x dplyr::select()   masks MASS::select()
```

```
data_height <- read_csv("data/plantHeightSingleSpp.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   Genus_species = col_character(),
##   Family = col_character(),
##   growthform = col_character(),
##   Country = col_character(),
##   Site = col_character(),
##   entered.by = col_character()
## )

## See spec(...) for full column specifications.
```

```
mod_height1 <- lm(height ~ lat, data = data_height)

summary(mod_height1)
```

```
##
## Call:
## lm(formula = height ~ lat, data = data_height)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -13.558  -6.955  -3.978   3.654  54.333
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 14.42516    1.72433   8.366 1.78e-14 ***
## lat         -0.17631    0.04847  -3.637 0.000362 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.95 on 176 degrees of freedom
## Multiple R-squared:  0.06991,Adjusted R-squared:  0.06463
## F-statistic: 13.23 on 1 and 176 DF,  p-value: 0.0003617
```

```
mod_height2 <- lm(height ~ rain + lat, data = data_height)

summary(mod_height2)
```

```
##
## Call:
## lm(formula = height ~ rain + lat, data = data_height)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -14.738  -5.932  -3.418   2.833  52.188
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.521724   3.077683   1.469 0.143575
## rain         0.004058   0.001062   3.823 0.000183 ***
## lat         -0.034109   0.059707  -0.571 0.568547
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.55 on 175 degrees of freedom
## Multiple R-squared:  0.1416,Adjusted R-squared:  0.1318
## F-statistic: 14.43 on 2 and 175 DF,  p-value: 1.579e-06
```

Reverse order of model terms

```
mod_height2a <- lm(height ~ lat + rain, data = data_height)

summary(mod_height2a)
```

```
##
## Call:
## lm(formula = height ~ lat + rain, data = data_height)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -14.738  -5.932  -3.418   2.833  52.188
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.521724   3.077683   1.469 0.143575
## lat         -0.034109   0.059707  -0.571 0.568547
## rain         0.004058   0.001062   3.823 0.000183 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.55 on 175 degrees of freedom
## Multiple R-squared:  0.1416,Adjusted R-squared:  0.1318
## F-statistic: 14.43 on 2 and 175 DF,  p-value: 1.579e-06
```

```
drop1(mod_height2, test = "F")
```

```
## Single term deletions
##
## Model:
## height ~ rain + lat
##        Df Sum of Sq   RSS    AIC F value    Pr(>F)
## <none>              19469 841.67
## rain    1   1625.55 21094 853.95 14.6117 0.0001834 ***
## lat     1     36.31 19505 840.00  0.3264 0.5685469
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

5.2.1 Marginal and Conditional interpretations

The **marginal effect** of a predictor variable in a regression model is the effect of that variable alone on the outcome, estimated without including any other predictor variable in the model.

The model above with `lat` as the only predictor variable, mod_height1, gives the **marginal effect** of `lat` as -0.176.

The **conditional effect** of a predictor variable in a regression model is the effect of that variable on the outcome when other predictors are included in the model and all those other predictors are held constant.

The model above with both `lat` and `rain` as predictor variables, mod_height2, gives the **conditional effect** of `lat`, after controlling for the effect of `rain`, as -0.034.

Note that in R, the "summary" output for "lm" gives the conditional effect of each predictor variable - i.e. the effect conditional on all the other predictor variables, or in other words, after adjusting for or controlling for, all the other predictor variables. This is shown by comparing the three blocks of output immediately above given by summary(mod_height2), summary(mod_height2a) and drop1(mod_height2).

5.3 Linear model with subsampling

5.3.1 Load necessary R packages

```
library(gamair)  ##  Data
library(ggplot2)
library(nlme)
library(emmeans)
library(dplyr)
```

Example

Note: This example is from the book, Wood, S. 2017. *Generalized Additive Models: An Introduction with R.* CRC Press, 2nd ed., pp. 61-65. This book is available as an e-book in UNSW library. (The example is also in the 1st edition of this book on pp. 277-281. Another example of this idea with a different data set, but without the mixed model analysis, is in Steel, R., Torrie, J. and Dickey, D. 1997. *Principles and Procedures of Statistics: A Biometrical Approach.* McGraw Hill, 3rd ed., pp. 157-165.)

Tree seedlings are grown under two levels of carbon dioxide concentration, with three trees assigned to each treatment. At six months of growth, stomatal area is measured at each of four random locations on each plant.

Notes

- "Stomates" are the tiny holes in plant leaves through which carbon dioxide and water vapour are exchanged with the atmosphere during photosynthesis and respiration.
- The sample sizes here are artificially small simply to illustrate the ideas.

5.3.2 Data

The data, "stomata", is included in the R package, "gamair".

```
data(stomata)
```

```
str(stomata)
```

```
## 'data.frame': 24 obs. of  3 variables:
##  $ area: num  1.61 1.63 1.54 1.72 1.39 ...
##  $ CO2 : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
##  $ tree: Factor w/ 6 levels "1","2","3","4",..: 1 1 1 1 2 2 2 2 3 3 ...
```

```
head(stomata)
```

```
##       area CO2 tree
## 1 1.605574   1    1
## 2 1.630071   1    1
## 3 1.539119   1    1
## 4 1.718732   1    1
## 5 1.389616   1    2
## 6 1.585880   1    2
```

Convert CO2 and tree to factors.

```
stomata$CO2 <- factor(stomata$CO2)

##  For CO2, "1" is "low" and "2" is "high".
##  Add these labels.

stomata$CO2 <- factor(stomata$CO2, labels = c("low", "high"))

stomata$tree <- factor(stomata$tree)
```

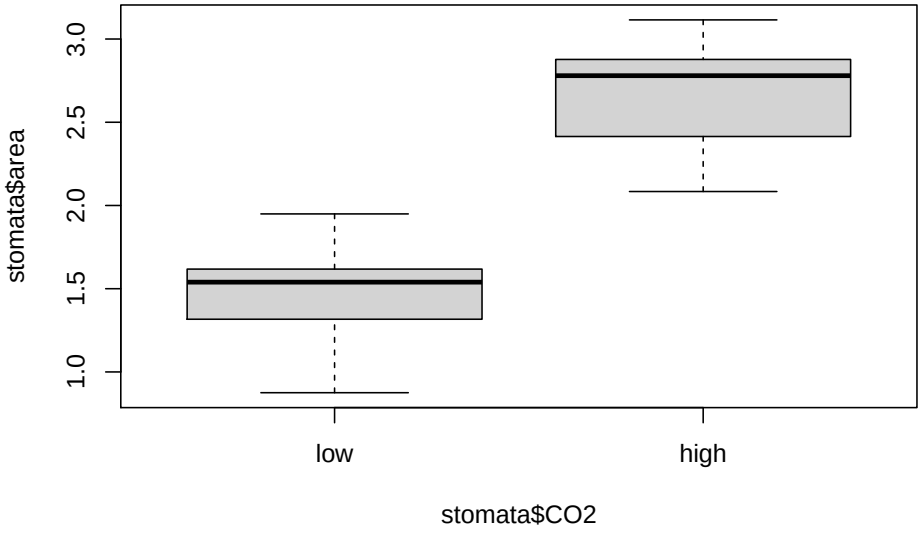
5.3.3 Summary statistics and plots

```
summary(stomata)
```

```
##       area          CO2     tree
##  Min.   :0.8753   low :12   1:4
##  1st Qu.:1.5396   high:12   2:4
```

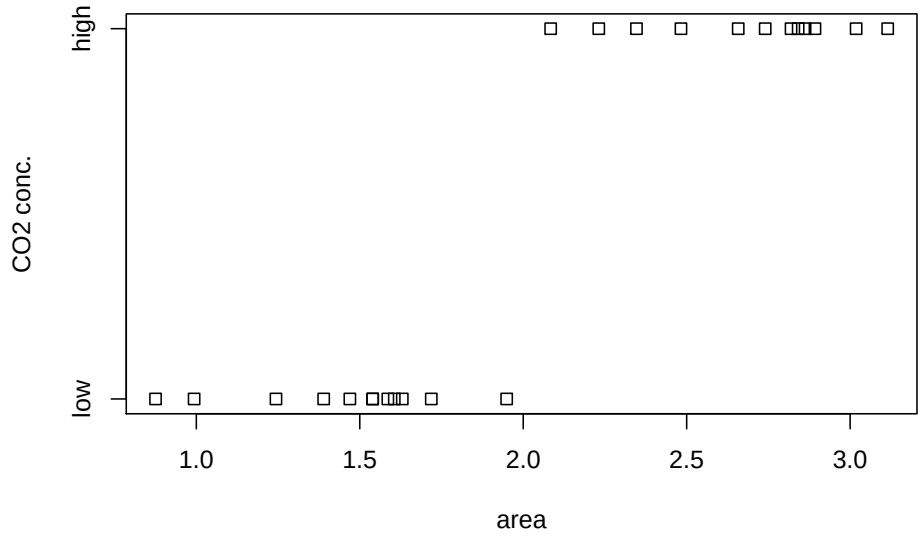
```
##  Median :2.0166            3:4
##  Mean   :2.0679            4:4
##  3rd Qu.:2.7600            5:4
##  Max.   :3.1149            6:4
```

```
boxplot(stomata$area ~ stomata$CO2)
```

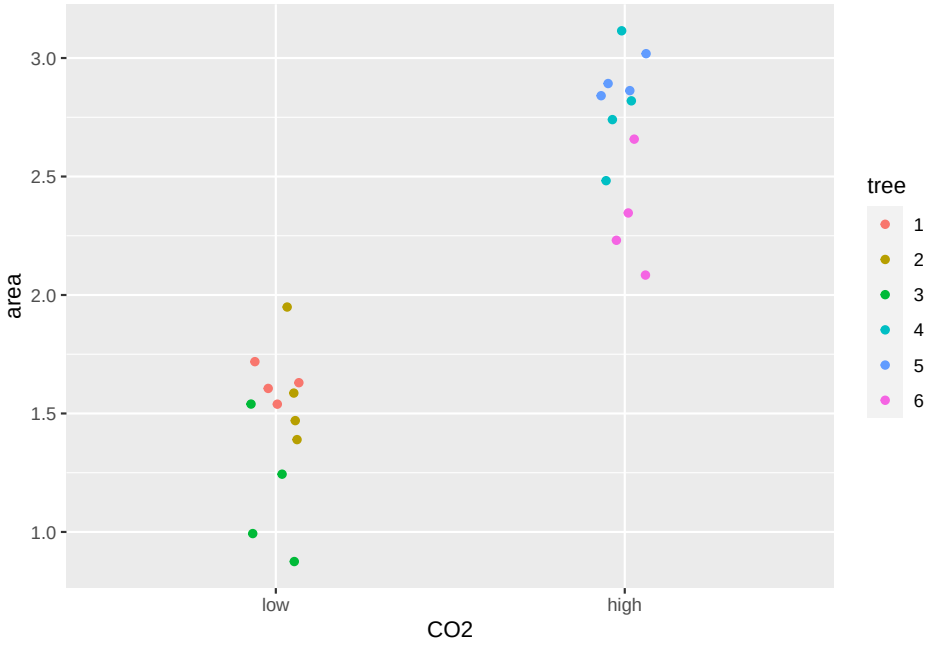


Plot with a strip plot (strip chart) as sample sizes are small.

```
stripchart(area ~ CO2, data = stomata, ylab = "CO2 conc.")
```



```
ggplot(stomata, aes(x = CO2, y = area, colour = tree)) +
  geom_jitter(position = position_jitter(0.075))
```



The variance appears fairly constant and the distribution looks fairly symmetric. (This suggests the linear model assumptions will be met, but they'll be checked later. The purpose of this example is to illustrate the proper way to analyse data from this type of experimental design and not to worry about assumptions, though.)

Note: Where you've got data with a quantitative, continuous response variable, the usual function to use for a linear model is "lm". In the short course, *Introductory Statistics for Researchers*, we used a function, "aov", for the one-way ANOVA model we fitted. That function is fairly limited in when you can use it and so "lm" is the function most commonly used. However, there are situations where lm is not appropriate.

The following analyses are taken from the book by Simon Wood cited above. Explanations of what's going on are more detailed in the book. It's useful to read the material in the book about this example but not to worry about the technical details there.

5.3.4 Wrong analysis: fixed effects linear model

To account for stomatal area here, we'd expect some contribution from CO2 concentration (there are two levels, low and high) and some contribution from

individual trees. The following analysis looks reasonable at face value.

```
m1 <- lm(area ~ CO2 + tree, data = stomata)

summary(m1)
```

```
##
## Call:
## lm(formula = area ~ CO2 + tree, data = stomata)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.30672 -0.10625 -0.01528  0.08436  0.37674
##
## Coefficients: (1 not defined because of singularities)
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.62337    0.10932  14.850 1.52e-11 ***
## CO2high      0.70639    0.15460   4.569 0.000238 ***
## tree2       -0.02473    0.15460  -0.160 0.874685
## tree3       -0.46041    0.15460  -2.978 0.008059 **
## tree4        0.45948    0.15460   2.972 0.008166 **
## tree5        0.57378    0.15460   3.711 0.001597 **
## tree6             NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2186 on 18 degrees of freedom
## Multiple R-squared:  0.9215,Adjusted R-squared:  0.8997
## F-statistic: 42.24 on 5 and 18 DF,  p-value: 2.511e-09
```

From above, $P < 0.0001$. The "NAs" in the output look odd.

The above model is wrong, however, because CO2 and tree are confounded. Trees are nested in CO2 treatments - we cannot, for example, separate the effect of low CO2 from the effect of tree 1 on the predicted value of area for that tree.

Compare the above model with a model including only CO2. The "anova" function below compares the two models - the P value in the output is testing whether the two models are no better than each other.

```
m0 <- lm(area ~ CO2, data = stomata)

anova(m0, m1)
```

```
## Analysis of Variance Table
```

```
##
## Model 1: area ~ CO2
## Model 2: area ~ CO2 + tree
##   Res.Df    RSS Df Sum of Sq      F   Pr(>F)
## 1     22 2.1348
## 2     18 0.8604  4    1.2744 6.6654 0.001788 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The above output says that the models are not equally good ($P = 0.002$, the P value is very small); consequently, the larger model is better - with more explanatory variables, it has more explanatory value. As model m0 contains only CO2 and m1 contains both CO2 and tree, then tree differences are important. However, from those models, we have no way of testing whether CO2 differences are important (as the only difference between the models was whether they included tree or not).

Test for CO2 effects by comparing models with tree but with and without CO2.

```
m2 <- lm(area ~ tree, data = stomata)

anova(m2, m1)
```

```
## Analysis of Variance Table
##
## Model 1: area ~ tree
## Model 2: area ~ CO2 + tree
##   Res.Df    RSS Df  Sum of Sq F Pr(>F)
## 1     18 0.8604
## 2     18 0.8604  0 2.2204e-16
```

The above output shows the two models are the same (the RSS, residual sum of squares, for each is the same) and so we cannot use this analysis to test for CO2 effects. This is caused by the confounding problem.

One way of getting around this problem is to average the response variable values for each tree and use the averaged values as our response variable.

Note you can only do this if there are the same number of data values in each group you average and there are no missing values!

```
stomata.avg <- stomata %>% group_by(tree) %>%
  summarise(area = mean(area))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
CO2_by_tree <- stomata %>% distinct(tree, CO2)

stomata.avg <- data.frame(stomata.avg, CO2 = CO2_by_tree$CO2)

str(stomata.avg)
```

```
## 'data.frame': 6 obs. of  3 variables:
##  $ tree: Factor w/ 6 levels "1","2","3","4",..: 1 2 3 4 5 6
##  $ area: num  1.62 1.6 1.16 2.79 2.9 ...
##  $ CO2 : Factor w/ 2 levels "low","high": 1 1 1 2 2 2
```

```
m3 <- lm(area ~ CO2, data = stomata.avg)

summary(m3)
```

```
##
## Call:
## lm(formula = area ~ CO2, data = stomata.avg)
##
## Residuals:
##       1       2       3       4       5       6
##  0.1617  0.1370 -0.2987  0.1151  0.2294 -0.3444
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.4617     0.1629   8.970 0.000855 ***
## CO2high       1.2125     0.2304   5.262 0.006247 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2822 on 4 degrees of freedom
## Multiple R-squared:  0.8738,Adjusted R-squared:  0.8422
## F-statistic: 27.69 on 1 and 4 DF,  p-value: 0.006247
```

```
anova(m3)
```

```
## Analysis of Variance Table
##
## Response: area
##           Df  Sum Sq Mean Sq F value   Pr(>F)
## CO2        1 2.20531 2.20531  27.687 0.006247 **
## Residuals  4 0.31861 0.07965
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The above analysis is valid, although it does involve a two-step process where you average the data for each tree first.

Another way of getting around this problem is to use the "aov" function here. *Note the same conditions in bold above apply to using the aov function.* We use the original, non-averaged data with aov.

```
m4 <- aov(area ~ CO2 + Error(tree), data = stomata)

summary(m4)
```

```
##
## Error: tree
##           Df Sum Sq Mean Sq F value  Pr(>F)
## CO2        1  8.821   8.821   27.69 0.00625 **
## Residuals  4  1.274   0.319
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Error: Within
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals 18 0.8604  0.0478
```

From the above two analyses, the correct P value for the effect of CO2 is 0.00625.

5.3.5 Right analysis: mixed effects linear model

A better (and correct) approach is to use a mixed effects model. *This works even if treatment groups do not have the same number of replicates (or if there are missing values).*

In this approach, tree is included as a grouping variable to identify clusters of data values (i.e. to identify which observed values come from which tree). Observations from the same tree will be correlated to some extent and this needs to be incorporated in the model via a grouping variable. If these groups were not specified in the model, it would appear as if all observations were independent and this would lead to smaller uncertainty in the estimates of treatment effects (because you'd be overstaing the number of independent observations, and hence, the amount of information you really had, or "information value" of your data). The trees in the study are considered as being randomly sampled from the (large) population of trees we are studying. The effects of individual trees are not considered important but they need to be included in the model to identify groups of data values.

So, let's fit a mixed model. You can use the "nlme" package or the "lme4" package. The nlme package is used here because it gives P values, while the lme4 package does not (another story!).

```
m5 <- lme(area ~ CO2, random = ~ 1 | tree, data = stomata)

#summary(m5)

summary(m5)$tTable
```

```
##                Value Std.Error DF  t-value      p-value
## (Intercept) 1.461659 0.1629435 18 8.970341 4.625996e-08
## CO2high     1.212522 0.2304370  4 5.261837 6.246688e-03
```

From the above output, the P value for the effect of CO2 concentration is 0.00625. This is the same as the P values above using the averaged data and using the aov function.

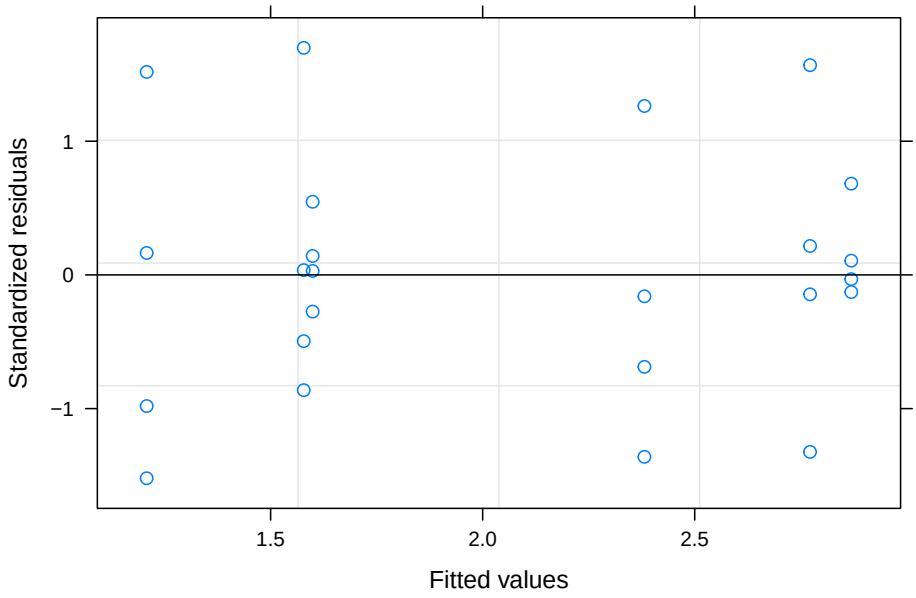
5.3.6 An aside - checking model assumptions

Let's check the assumptions for the mixed model because we can. This shows how to check the two assumptions of constant variance and Normal distribution for a linear mixed model. We won't worry here about whether the assumptions are satisfied, as the emphasis is on appropriate methods to use.

Equal variance assumption

Use the plot function with the model name. This produces a residuals vs fitted values plot, which is the standard plot for checking the constant variance assumption.

```
plot(m5)
```

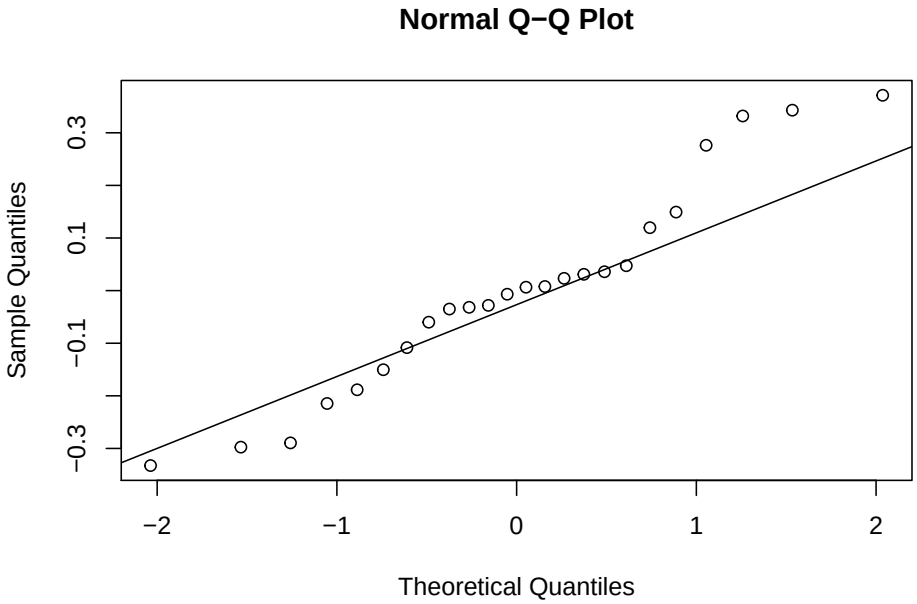


The variances appear to be somewhat unequal.

Normal distribution assumption

Use a Normal quantile-quantile (Q-Q) plot.

```
qqnorm(residuals(m5))
qqline(residuals(m5))
```



The Normal distribution assumption does not appear to be particularly well met.

Back to the main story ...

With balanced data, you'll get the same results from an analysis using aov or lme. However, it's preferable to use a mixed effects model, as recommended by Simon Wood and described here.

5.3.7 Fitted values and confidence intervals

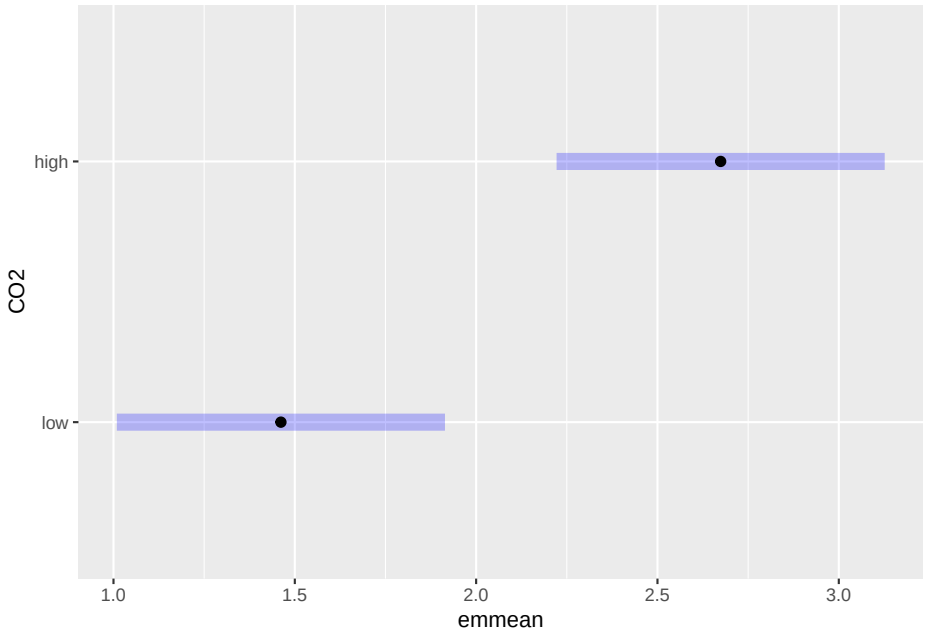
The R package, "emmeans", is very useful for getting means and their confidence intervals and plots of these. This package will work with models from lm, aov , lme, ...

Estimates and plots from the fixed effects model using averaged data and the mixed model are below.

```
m3.emm <- emmeans(m3, ~ CO2)

summary(m3.emm)
```

```
##  CO2  emmean    SE df lower.CL upper.CL
##  low    1.46 0.163  4     1.01     1.91
##  high   2.67 0.163  4     2.22     3.13
##
## Confidence level used: 0.95
```

```
plot(m3.emm)
```



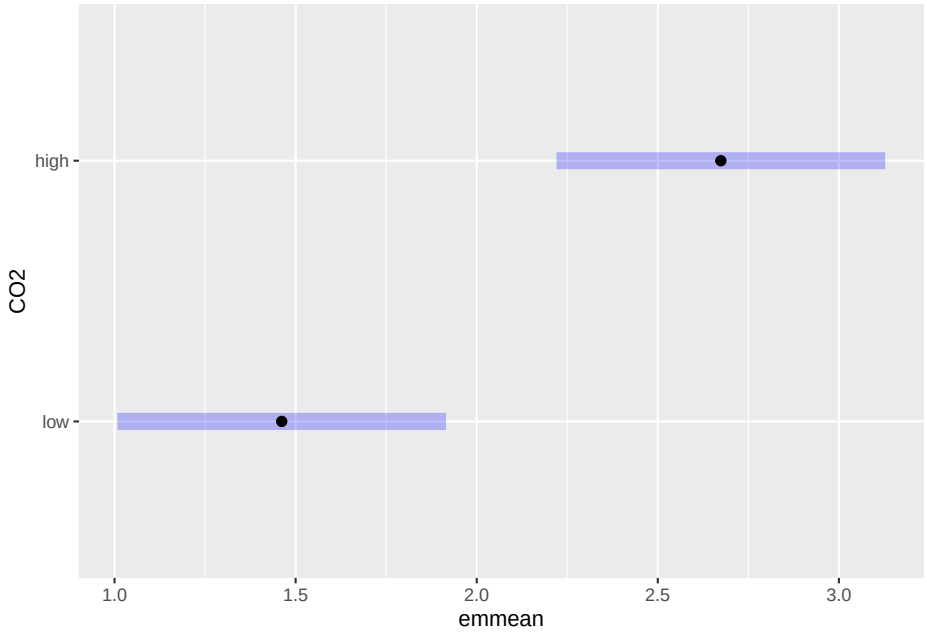

```
m5.emm <- emmeans(m5, ~CO2, mode = "satterthwaite")
```

```
## Warning in sweep(X, 1, sqrt(weights), "*"): STATS is longer than the
## extent of 'dim(x)[MARGIN]'
```

```
summary(m5.emm)
```

```
##  CO2  emmean    SE   df lower.CL upper.CL
##  low    1.46 0.163 3.97     1.01     1.92
##  high   2.67 0.163 3.97     2.22     3.13
##
## Degrees-of-freedom method: satterthwaite
## Confidence level used: 0.95
```

```
plot(m5.emm)
```



5.3.8 Another aside - code the nested structure - not recommended

The analysis can be done with lm if the model is coded so that the nesting of trees within CO2 treatments is incorporated. It can only be done this way because, again, the data is balanced and there are no missing values. The full data set is used.

```
m6 <- lm(area ~ CO2 / tree, data = stomata)

anova(m6)
```

```
## Analysis of Variance Table
##
## Response: area
##           Df Sum Sq Mean Sq  F value    Pr(>F)
## CO2        1 8.8213  8.8213 184.5452 6.686e-11 ***
## CO2:tree   4 1.2744  0.3186   6.6654  0.001788 **
## Residuals 18 0.8604  0.0478
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

In the output above, the effect of CO2 needs to be compared against the correct residual term. The P value against CO2 shown in the output is obtained

by comparing the variation of CO2 treatments against the variation *among observations within trees*. This is not the correct comparison. As tree is the experimental unit (leaf is the observational unit), the correct term for the comparison is the term corresponding to the variation *between trees within CO2 treatments*. In the output, that term is CO2:tree. Consequently, the correct P value for the effect of CO2 is obtained from the F statistic calculated as 8.8213 / 0.3186 (i.e. (Mean Sq for CO2) / (Mean Sq for appropriate residual term)). The calculation for obtaining the P value is below.

```
f_obs <- 8.8213 / 0.3186

f_obs
```

```
## [1] 27.6877
```

This is the correct F value, as seen previously.

```
p_val <- 1 - pf(f_obs, 1, 4)

p_val
```

```
## [1] 0.00624638
```

The P value is also clearly the same as seen previously.

It could be awkward to get correct confidence intervals from this model.

5.3.9 Summary

When you have subsampling (i.e. multiple measurements made on individual experimental units), use a mixed effects model.

If you have unequal replication across treatments or missing values, definitely use a mixed effects model.

5.4 Marginal means - emmeans example

```
library(emmeans)
library(tidyr)
library(dplyr)
library(RcmdrMisc)
```

```
## Loading required package: car

## Loading required package: carData

##
## Attaching package: 'car'

## The following object is masked from 'package:dplyr':
##
##     recode

## The following object is masked from 'package:purrr':
##
##     some

## The following object is masked from 'package:boot':
##
##     logit

## Loading required package: sandwich
```

From the document, *Basics of estimated marginal means.*

https://cran.r-project.org/web/packages/emmeans/vignettes/basics.html

pigs {emmeans} R Documentation

Effects of dietary protein on free plasma leucine concentration in pigs

Description

A two-factor experiment with some observations lost

A data frame with 29 observations and 3 variables:

- source: Source of protein in the diet (factor with 3 levels: fish meal, soybean meal, dried skim milk)
- percent: Protein percentage in the diet (numeric with 4 values: 9, 12, 15, and 18)
- conc: Concentration of free plasma leucine, in mcg/ml

Source: Windels HF (1964) PhD thesis, Univ. of Minnesota. (Reported as Problem 10.8 in Oehlert G (2000) A First Course in Design and Analysis of Experiments, licensed under Creative Commons, http://users.stat.umn.edu/~gary/Book.html.) Observations 7, 22, 23, 31, 33, and 35 have been omitted, creating a more notable imbalance.

[Package emmeans version 1.4.6]

```
data(pigs)

pigs
```

```
##    source percent conc
## 1    fish       9 27.8
## 2    fish       9 23.7
## 3    fish      12 31.5
## 4    fish      12 28.5
## 5    fish      12 32.8
## 6    fish      15 34.0
## 7    fish      15 28.3
## 8    fish      18 30.6
## 9    fish      18 32.7
## 10   fish      18 33.7
## 11    soy       9 39.3
## 12    soy       9 34.8
## 13    soy       9 29.8
## 14    soy      12 39.8
## 15    soy      12 40.0
## 16    soy      12 39.1
## 17    soy      15 38.5
## 18    soy      15 39.2
## 19    soy      15 40.0
## 20    soy      18 42.9
## 21   skim       9 40.6
## 22   skim       9 31.0
## 23   skim       9 34.6
## 24   skim      12 42.9
## 25   skim      12 50.1
## 26   skim      12 37.4
## 27   skim      15 59.5
## 28   skim      15 41.4
## 29   skim      18 59.8
```

```
pigs$percent <- factor(pigs$percent)
```

```
numSummary(pigs[ , 3], statistics = "mean",
           groups = interaction(pigs$source, pigs$percent))
```

```
##             mean n
## fish.9  25.75000 2
## soy.9   34.63333 3
## skim.9  35.40000 3
```

```
## fish.12 30.93333 3
## soy.12  39.63333 3
## skim.12 43.46667 3
## fish.15 31.15000 2
## soy.15  39.23333 3
## skim.15 50.45000 2
## fish.18 32.33333 3
## soy.18  42.90000 1
## skim.18 59.80000 1
```

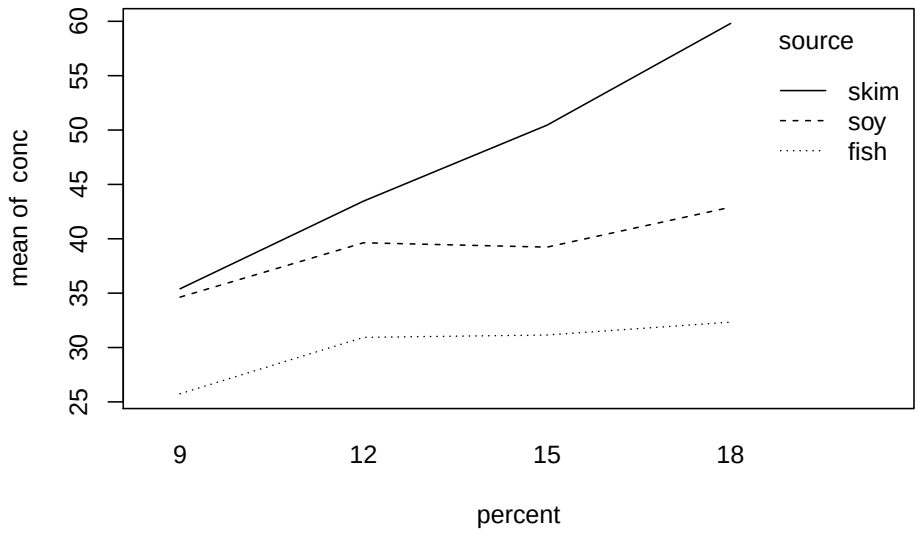
Cell means (see also below)

These are arithmetic means of values in each two-way combination

```
with(pigs, tapply(conc, INDEX = list(source, percent), mean))
```

```
##              9       12       15       18
## fish 25.75000 30.93333 31.15000 32.33333
## soy  34.63333 39.63333 39.23333 42.90000
## skim 35.40000 43.46667 50.45000 59.80000
```

```
with(pigs, interaction.plot(percent, source, conc))
```



Marginal means

Arithmetic mean of values in each “percent” group

```
with(pigs, tapply(conc, percent, mean))
```

```
##        9       12       15       18
## 32.70000 38.01111 40.12857 39.94000
```

Find average for percent = 15

```
( (34.0 + 28.3) + (38.5 + 39.2 + 40.0) + (59.5 + 41.4) ) / 7
```

```
## [1] 40.12857
```

Cell means

Arithmetic means of values in each combination - same as above with two-way tapply call

```
cell.means <- matrix(with(pigs,
    tapply(conc, interaction(source, percent), mean)),
    nrow = 3)
cell.means
```

```
##          [,1]     [,2]     [,3]     [,4]
## [1,] 25.75000 30.93333 31.15000 32.33333
## [2,] 34.63333 39.63333 39.23333 42.90000
## [3,] 35.40000 43.46667 50.45000 59.80000
```

```
apply(cell.means, 2, mean)
```

```
## [1] 31.92778 38.01111 40.27778 45.01111
```

Compare with arithmetic mean of values in each "percent" group

```
with(pigs, tapply(conc, percent, mean))
```

```
##        9       12       15       18
## 32.70000 38.01111 40.12857 39.94000
```

The two sets are different because the different cells do not all have the same number of observations (some observations were lost).

```
with(pigs, table(source, percent))
```

```
##        percent
## source 9 12 15 18
##   fish 2  3  2  3
##   soy  3  3  3  1
##   skim 3  3  2  1
```

The marginal mean for percent = 12 is the same as the average of the cell means because no observations were lost - i.e. there are equal weights on all the cell means.

We can reproduce the *marginal means* by weighting the cell means with these frequencies. For example, in the last column:

```
sum(c(3, 1, 1) * cell.means[ , 4]) / 5
```

```
## [1] 39.94
```

5.5 Multiple regression example

```
library(dplyr)
library(ggplot2)
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
##
## Attaching package: 'GGally'
```

```
## The following object is masked _by_ '.GlobalEnv':
##
##     pigs
```

```
## The following object is masked from 'package:emmeans':
##
##     pigs
```

```
library(sjPlot)
```

```
## Registered S3 methods overwritten by 'lme4':
##   method                          from
##   cooks.distance.influence.merMod car
##   influence.merMod                car
##   dfbeta.influence.merMod         car
##   dfbetas.influence.merMod        car
```

```
## Install package "strengejacke" from GitHub (`devtools::install_github("strengejacke/
```

```
library(ggpubr)
library(emmeans)
```

Example from StatSci.org - OzDASL, Australian Institute of Sport

http://www.statsci.org/data/oz/ais.html

Variable Description

Sport Sport Sex male or female Ht Height in cm Wt Weight in kg LBM Lean body mass RCC Red cell count WCC White cell count Hc Hematocrit Hg Hemoglobin Ferr Plasma ferritin concentration BMI Body mass index = weight/height^2 SSF Sum of skin folds %Bfat % body fat

```
athletes <- read.table("http://www.statsci.org/data/oz/ais.txt", sep = "", header = T)

str(athletes)

save(athletes, file = "athletes_OzDASL.RData")
```

```
load("data/athletes_OzDASL.Rdata")

str(athletes)
```

```
## 'data.frame': 202 obs. of  13 variables:
##  $ Sex   : chr  "female" "female" "female" "female" ...
##  $ Sport : chr  "BBall" "BBall" "BBall" "BBall" ...
##  $ RCC   : num  3.96 4.41 4.14 4.11 4.45 4.1 4.31 4.42 4.3 4.51 ...
##  $ WCC   : num  7.5 8.3 5 5.3 6.8 4.4 5.3 5.7 8.9 4.4 ...
##  $ Hc    : num  37.5 38.2 36.4 37.3 41.5 37.4 39.6 39.9 41.1 41.6 ...
##  $ Hg    : num  12.3 12.7 11.6 12.6 14 12.5 12.8 13.2 13.5 12.7 ...
##  $ Ferr  : int  60 68 21 69 29 42 73 44 41 44 ...
##  $ BMI   : num  20.6 20.7 21.9 21.9 19 ...
```

```
##  $ SSF   : num  109.1 102.8 104.6 126.4 80.3 ...
##  $ X.Bfat: num  19.8 21.3 19.9 23.7 17.6 ...
##  $ LBM   : num  63.3 58.5 55.4 57.2 53.2 ...
##  $ Ht    : num  196 190 178 185 185 ...
##  $ Wt    : num  78.9 74.4 69.1 74.9 64.6 63.7 75.2 62.3 66.5 62.9 ...
```

```
athletes$Sex <- factor(athletes$Sex)

athletes$Sport <- factor(athletes$Sport)

#names( athletes )[names(athletes) == "X.Bfat"] <- "Bfat_pc"

##  or

athletes <- athletes %>% rename( Bfat_pc = X.Bfat)
```

5.5.1 Regression model

Response: LBM

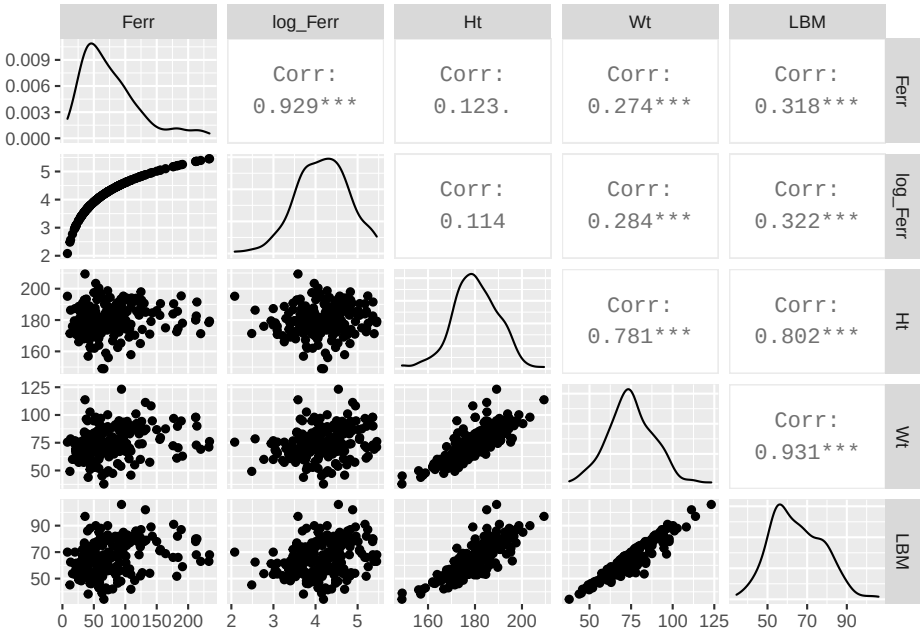
Predictors:

- height
- weight
- gender
- Ferr

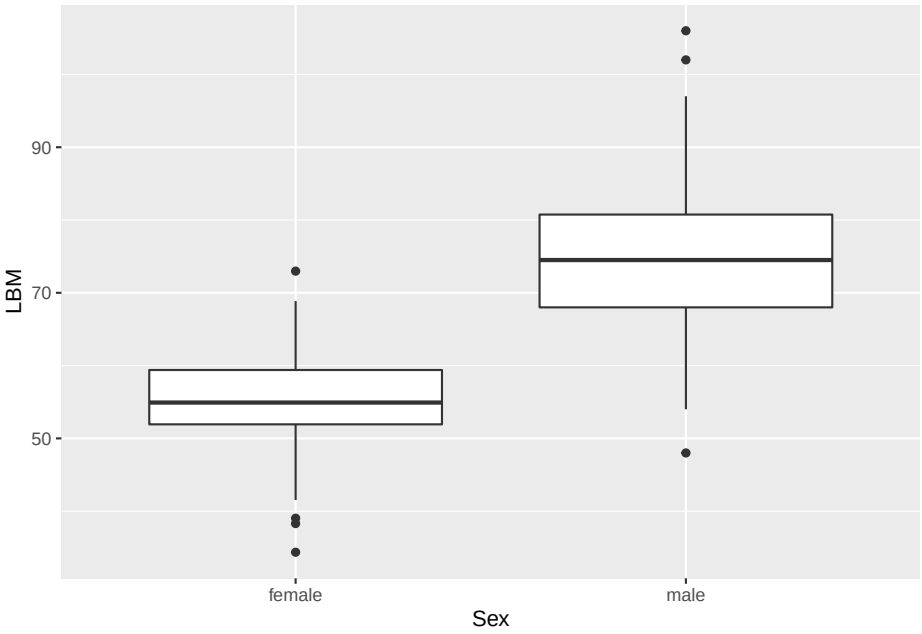
Include log(Ferr) as suggested on OzDASL web page.

```
athletes$log_Ferr <- log(athletes$Ferr)
```

```
ggpairs(athletes, columns = c(7, 14, 12, 13, 11))
```




```
ggplot(athletes, aes(x = Sex, y = LBM)) +
  geom_boxplot()
```



```
ath.lm1 <- lm(LBM ~ Ht + Wt +log_Ferr + Sex, data = athletes)

summary(ath.lm1)
```

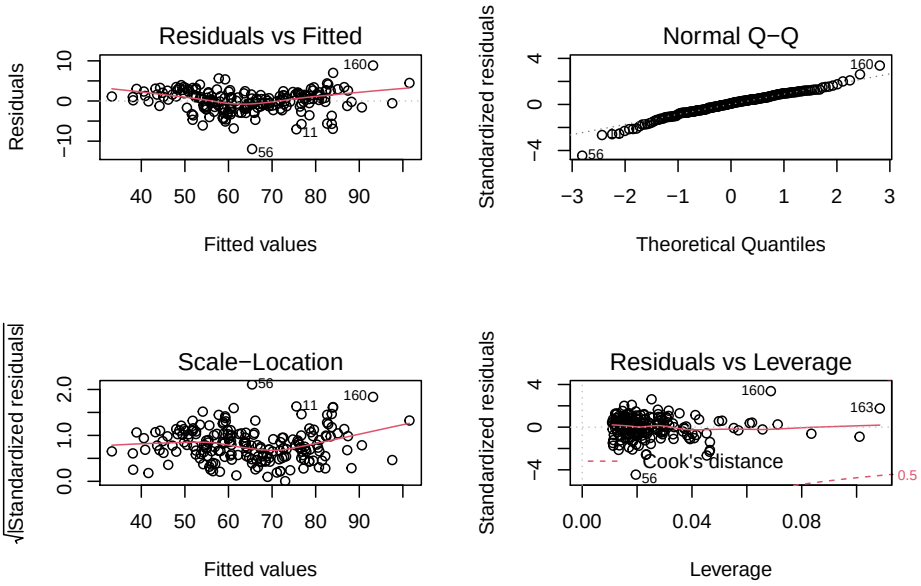
```
##
## Call:
## lm(formula = LBM ~ Ht + Wt + log_Ferr + Sex, data = athletes)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.9543  -1.5237   0.2123   1.6439   8.8396
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.30242    5.49116  -0.237   0.8128
## Ht           0.08320    0.03400   2.447   0.0153 *
## Wt           0.65455    0.02326  28.141   <2e-16 ***
## log_Ferr    -0.61689    0.35494  -1.738   0.0838 .
## Sexmale      9.23148    0.51135  18.053   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.713 on 197 degrees of freedom
## Multiple R-squared:  0.9578,Adjusted R-squared:  0.9569
## F-statistic:  1117 on 4 and 197 DF,  p-value: < 2.2e-16
```

```
ath.lm2 <- lm(LBM ~ Ht , data = athletes)

summary(ath.lm2)
```

```
##
## Call:
## lm(formula = LBM ~ Ht, data = athletes)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -18.566  -4.913  -0.556   4.171  31.853
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -129.0947    10.2259  -12.62   <2e-16 ***
## Ht             1.0770     0.0567   19.00   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

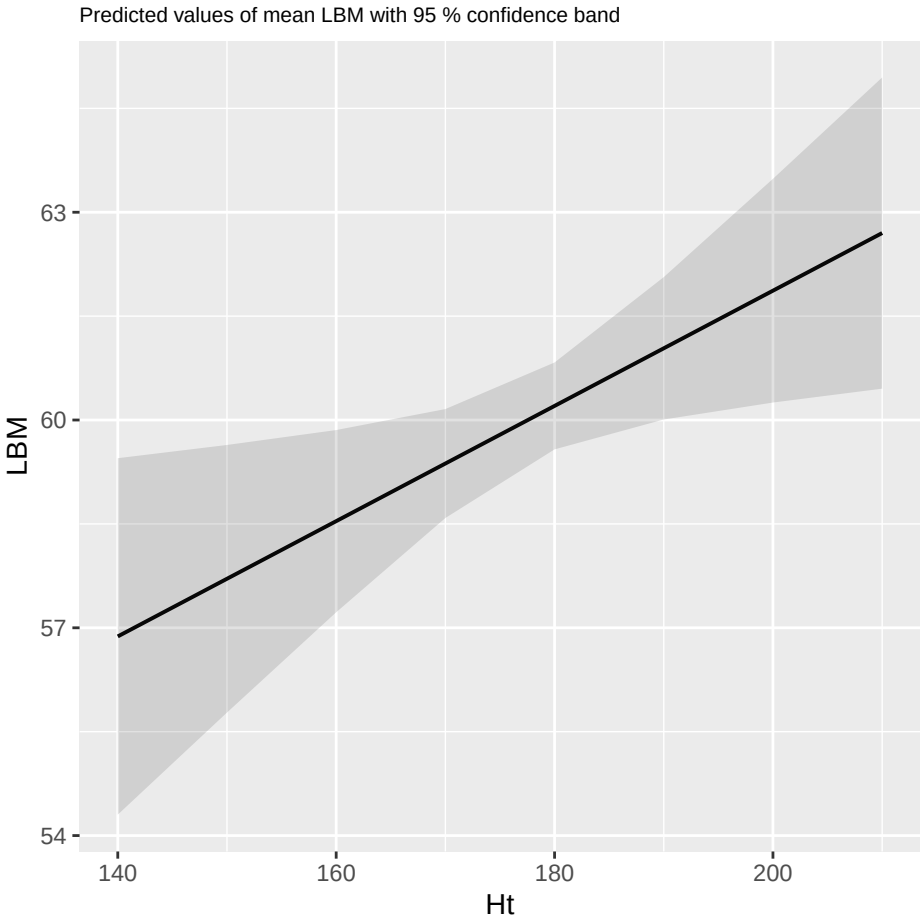
```
##
## Residual standard error: 7.825 on 200 degrees of freedom
## Multiple R-squared:  0.6434,Adjusted R-squared:  0.6416
## F-statistic: 360.8 on 1 and 200 DF,  p-value: < 2.2e-16
```

```
par(mfrow = c(2, 2))
plot(ath.lm1)
```




```
par(mfrow = c(1, 1))
```

```
plot1 <- plot_model(ath.lm1, type = "pred", terms = "Ht") +
  ggtitle("Predicted values of mean LBM with 95 % confidence band") +
  theme(plot.title = element_text(size = 8))

print(plot1)
```



The plot above shows a confidence band for predicted mean LBM, as seen by comparing the intervals below with the output of the "predict" function further below.

```
plot1$data
```

```
##
## # Predicted values of LBM
## # x = Ht
##
##   x | Predicted |   SE | group_col |         95% CI
## ------------------------------------------------------
## 140 |     56.88 | 1.31 |         1 | [54.30, 59.45]
## 150 |     57.71 | 0.99 |         1 | [55.78, 59.64]
## 160 |     58.54 | 0.67 |         1 | [57.22, 59.86]
## 170 |     59.37 | 0.40 |         1 | [58.59, 60.16]
```

```
## 180 |     60.20 | 0.32 |         1 | [59.58, 60.83]
## 190 |     61.04 | 0.52 |         1 | [60.01, 62.06]
## 200 |     61.87 | 0.82 |         1 | [60.25, 63.48]
## 210 |     62.70 | 1.15 |         1 | [60.45, 64.95]
##
## Adjusted for:
## *       Wt =  75.01
## * log_Ferr =   4.16
## *      Sex = female
```

Confidence band - band for fitted mean

```
X_new <- expand.grid(Ht = 14:21*10, Wt = 75.01, log_Ferr = 4.16,
                     Sex = "female")


ath.lm1.pred.CI <- predict(ath.lm1, newdata = X_new, se.fit = T,
                           type = "response",
                           interval = "confidence")

data.frame(Ht = 14:21*10, se = ath.lm1.pred.CI$se.fit,
           ath.lm1.pred.CI$fit)
```

```
##    Ht        se      fit      lwr      upr
## 1 140 1.3130285 56.87714 54.28775 59.46654
## 2 150 0.9858112 57.70914 55.76505 59.65324
## 3 160 0.6714313 58.54115 57.21703 59.86526
## 4 170 0.4012888 59.37315 58.58177 60.16452
## 5 180 0.3200962 60.20515 59.57389 60.83640
## 6 190 0.5245063 61.03715 60.00278 62.07152
## 7 200 0.8239954 61.86915 60.24417 63.49414
## 8 210 1.1463186 62.70115 60.44052 64.96178
```

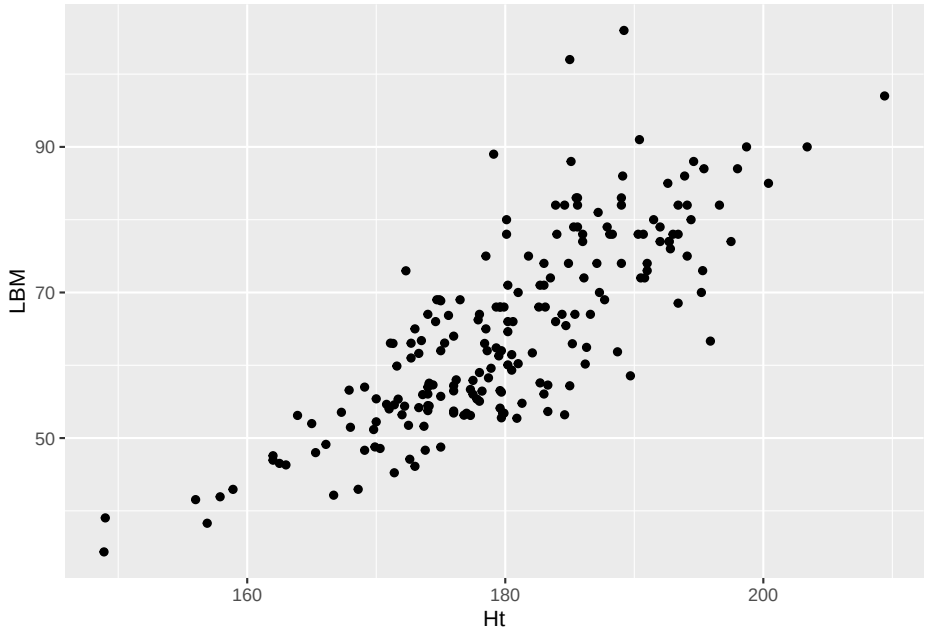
Prediction band - band for single prediction

```
ath.lm1.pred.PI <- predict(ath.lm1, newdata = X_new, se.fit = T,
                           type = "response",
                           interval = "prediction")

data.frame(Ht = 14:21*10, se = ath.lm1.pred.PI$se.fit,
           ath.lm1.pred.PI$fit)
```

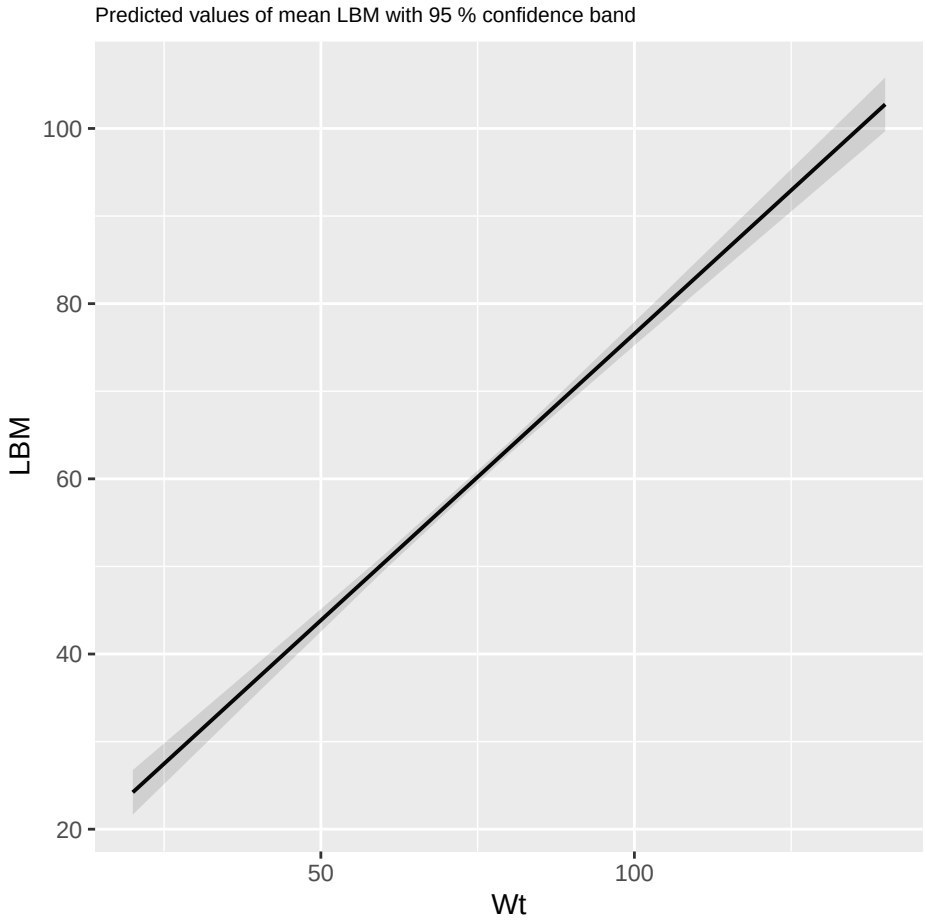
```
##    Ht        se      fit      lwr      upr
## 1 140 1.3130285 56.87714 50.93353 62.82075
```

```
## 2 150 0.9858112 57.70914 52.01695 63.40134
## 3 160 0.6714313 58.54115 53.02981 64.05248
## 4 170 0.4012888 59.37315 53.96502 64.78127
## 5 180 0.3200962 60.20515 54.81812 65.59217
## 6 190 0.5245063 61.03715 55.58816 66.48614
## 7 200 0.8239954 61.86915 56.27790 67.46040
## 8 210 1.1463186 62.70115 56.89323 68.50908
```

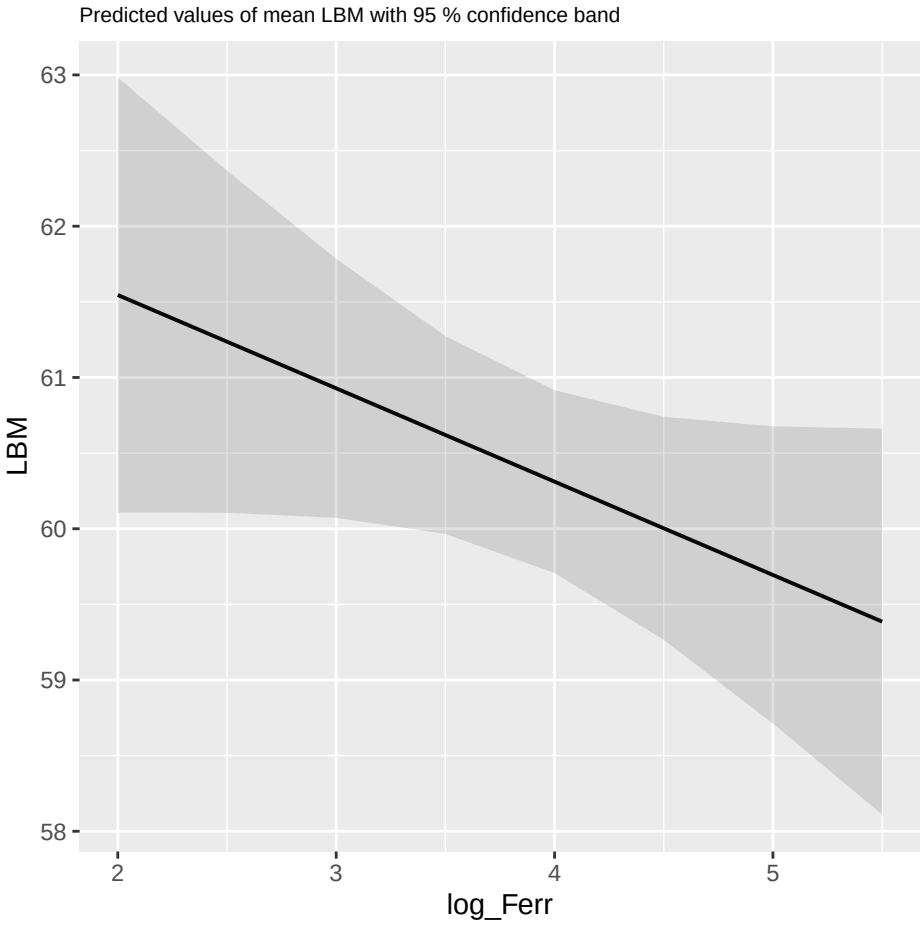
```
ggplot(data = athletes, aes(x = Ht, y = LBM)) +
  geom_point()
```



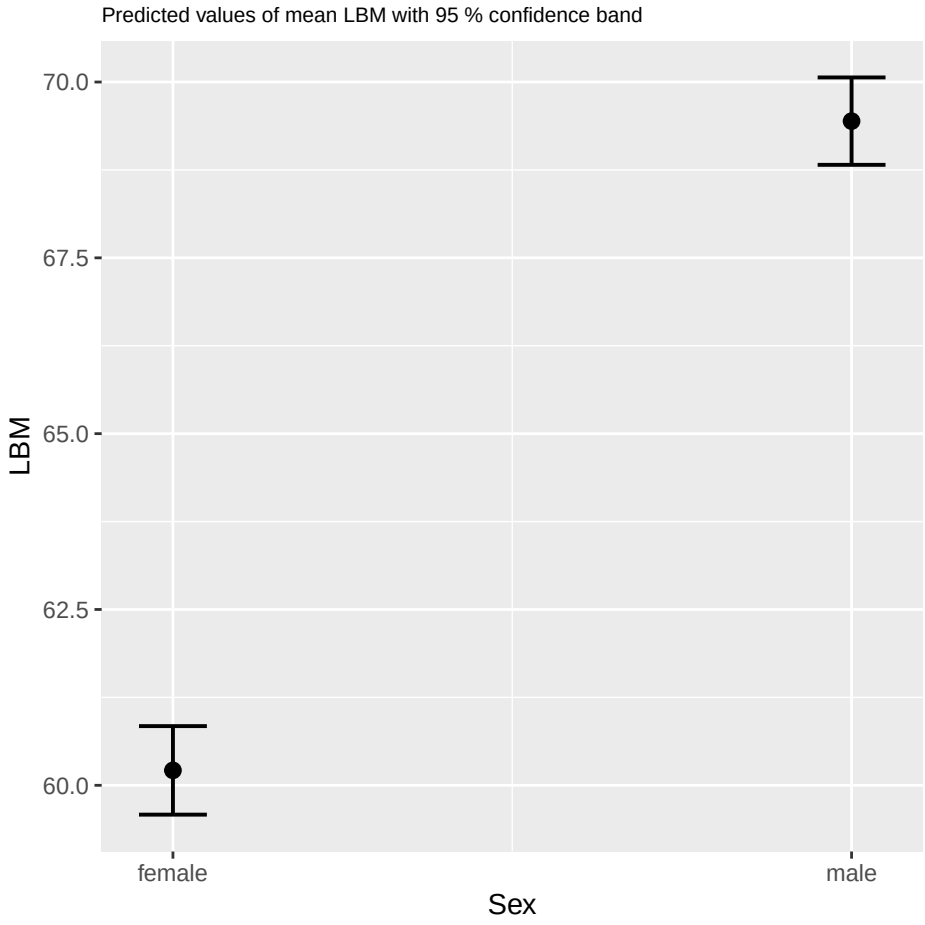

```
plot2 <- plot_model(ath.lm1, type = "pred", terms = "Wt") +
  ggtitle("Predicted values of mean LBM with 95 % confidence band") +
  theme(plot.title = element_text(size = 8))

print(plot2)
```




```
plot3 <- plot_model(ath.lm1, type = "pred",
                    terms = "log_Ferr") +
  ggtitle("Predicted values of mean LBM with 95 % confidence band") +
  theme(plot.title = element_text(size = 8))

print(plot3)
```



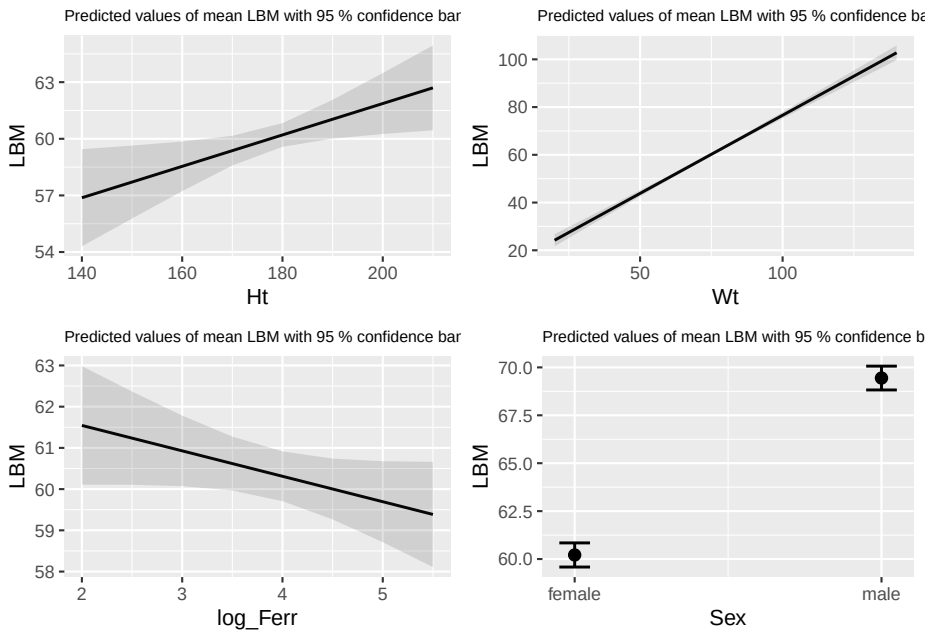

```
plot4 <- plot_model(ath.lm1, type = "pred", terms = "Sex") +
  ggtitle("Predicted values of mean LBM with 95 % confidence band") +
  theme(plot.title = element_text(size = 8))

print(plot4)
```




```
plot4$data
```

```
##
## # Predicted values of LBM
## # x = Sex
##
## x | Predicted |   SE | group_col |         95% CI
## -------------------------------------------------
## 1 |     60.21 | 0.32 |         1 | [59.58, 60.84]
## 2 |     69.44 | 0.32 |         1 | [68.82, 70.07]
##
## Adjusted for:
## *       Ht = 180.10
## *       Wt =  75.01
## * log_Ferr =   4.16
```

```
plot_all <- ggarrange(plot1, plot2, plot3, plot4, nrow = 2,
                      ncol = 2)

print(plot_all)
```




```
pdf(file = "athletes_reg_plot_panel.pdf", width = 7,
    height = 5)

plot_all

dev.off()
```

```
## pdf
##   2
```

5.6 Residual plots - redres

`redres` package

https://goodekat.github.io/redres/index.html

Chapter 6

Linear mixed models

6.1 Finding variance components from lme models

```
estu <- read.csv("data/Estuaries.csv")

str(estu)
```

```
## 'data.frame': 54 obs. of  7 variables:
##  $ X                  : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Modification       : chr  "Modified" "Modified" "Modified" "Modified" ...
##  $ Estuary            : chr  "JAK" "JAK" "JAK" "JAK" ...
##  $ Site               : int  1 1 2 2 3 3 4 4 1 1 ...
##  $ Hydroid            : int  0 0 0 0 1 1 0 0 7 5 ...
##  $ Total              : int  44 42 32 44 42 48 45 34 29 51 ...
##  $ Schizoporella.errata: int  15 8 9 14 6 12 28 1 0 0 ...
```

```
estu.lme1 <- lme(Total ~ Modification, random = ~ 1 | Estuary,
                 data = estu,
                 correlation = corCompSymm(form = ~ 1 | Estuary))

summary(estu.lme1)
```

```
## Linear mixed-effects model fit by REML
##  Data: estu
##        AIC      BIC    logLik
##   404.6881 414.4444 -197.3441
```

```
##
## Random effects:
##  Formula: ~1 | Estuary
##         (Intercept) Residual
## StdDev:    7.424348 9.277184
##
## Correlation Structure: Compound symmetry
##  Formula: ~1 | Estuary
##  Parameter estimate(s):
## Rho
##   0
## Fixed effects: Total ~ Modification
##                            Value Std.Error DF   t-value p-value
## (Intercept)            40.97295  4.726969 47  8.667912  0.0000
## ModificationPristine -14.47295  6.230091  5 -2.323072  0.0678
##  Correlation:
##                       (Intr)
## ModificationPristine -0.759
##
## Standardized Within-Group Residuals:
##        Min         Q1        Med         Q3        Max
## -2.3859461 -0.7141963  0.2765803  0.5239747  2.0455630
##
## Number of Observations: 54
## Number of Groups: 7
```

```
VarCorr(estu.lme1)
```

```
## Estuary = pdLogChol(1)
##             Variance StdDev
## (Intercept) 55.12094 7.424348
## Residual    86.06614 9.277184
```

Chapter 7

Experimental design

7.1 Sample size by simulation

From: Study Design, 2hr - Gordana

Using pilot data, find values of linear model parameters. In this experiment, I would like to see if there is an effect of Petal length on Sepal length. I have pilot data with 15 observations.

```
data(iris)

str(iris)
```

```
## 'data.frame': 150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
table(iris$Species)
```

```
##
##     setosa versicolor  virginica
##         50         50         50
```

Get some pilot data (or get different data from Gordana's)

```r
# rand_setosa <- sample(1:50, size = 15, replace = F)

# pilot <- iris[rand_setosa, ]

##  Gordana's data

pilot <- iris[1:15, c(1, 3)]
```

```r
pilot_mod <- lm(Sepal.Length ~ Petal.Length, data = pilot)

summary(pilot_mod)
```

```
##
## Call:
## lm(formula = Sepal.Length ~ Petal.Length, data = pilot)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.50175 -0.25965 -0.05965  0.14825  1.01404
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    4.0912     1.0441   3.918  0.00176 **
## Petal.Length   0.5789     0.7316   0.791  0.44296
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4034 on 13 degrees of freedom
## Multiple R-squared:  0.04595,Adjusted R-squared:  -0.02743
## F-statistic: 0.6262 on 1 and 13 DF,  p-value: 0.443
```

```r
beta_0 <- coef(pilot_mod)[1] # estimated intercept (from pilot)

sd <- summary(pilot_mod)$sigma # estimated variability

my_range <- range(pilot$Petal.Length) # range of petal length

nsim <- 500 # number of simulated datasets
```

Then specify a meaningful effect size (here it's the slope of Petal length)

```r
# ecologically meaningful slope (same scale as data)

beta_1 <- 1
```

```r
N <- 20 # desired sample size

sim_dat = data.frame(Sepal.Length = NA,
                     Petal.Length = seq(my_range[1],
                                        my_range[2],
                                        length = N) )

pval = rep(NA, nsim)

for (i in 1:nsim) {
  mean_y <- beta_0 + beta_1 * sim_dat$Petal.Length
  sim_dat$Sepal.Length <- rnorm(N, mean = mean_y, sd = sd)
  m <- lm(Sepal.Length ~ Petal.Length, data = sim_dat)
  pval[i] <- coef(summary(m))["Petal.Length", "Pr(>|t|)"]
  } # cycle through all N values

sum(pval < 0.05) / nsim
```

```
## [1] 0.46
```