[How to use these Notes?]
Here you'll find brief instructions for what will be done during the workshop. This is not a handbook or an academic script. Together with this file you'll have also a command file containing all codes used during the workshop. In general – we will try to work interactively, so that everybody could benefit as much as possible. It is important to realize that – although we'll learn how to use presented methods practically – we'll also use toy data and simulations to "feel" presented methods and become confident about what they really do.

In the text, I provide outputs from executing all commands, except graphics – plots are not included. So – every time a plotting function is invoked expect to see the picture on your default output (screen) You can find code for all exercises in the "codeblocks" files supplies with this text. You'll also get all necessary external files such as data files. All codes have been tested and should work without problems but ensure that all supplied files are inside your working directory. I assume all should have internet access during the classes but in case you didn't have – please find attached the list of all packages that will have to be installed to do the exercises. And finally – **please forgive me any typos and spelling mistakes**. Quite often I have problems writing correctly in my own language…

To avoid any permanent alterings of default graphical parameters do the following:

```
> par() -> opar
> par(opar) #to restore

# in R it means comment – I will frequently use this to introduce
comments inside the code
```

[Credits]
Parts of this workshop are based (inevitably) on the work of Jarrod Hadfield – author and maintainer of **MCMCglmm** package. I'm also using freely available datasets from several web-pages (see References), and two books on statistics: Crawley's "The R book" and Manly's "Multivariate statistical methods".

[Note for experienced R users]
I'm aware that every person works in its own personalized way. Most of what's going to be presented here can be done under every philosophy of working with R. I'm not used to working with R Editor or any graphical interface extensions – but if you are feel free to work the way you did so far. As most users use Windows – we'll work in this environment. If you're a Linux-mad person (like me;)) you shouldn't have any problems following the workshop –just remember about different formatting of file paths.

R's must-knows

I assume that most of you (all of you) had some previous experience with R. Here I'm going to provide some basics and useful information that should make some of our further code clearer. This whole part will be in the form of R code – so look for comments prefixed by #.

```
> ###### R's must-knows ######

> #always remember to set up your working directory before starting
> #any analyses

> setwd("C:/R")

> #R works on vectors so everything, even single number is a vector
> #vectors on higher-order structures such as matrices can be
> #added, multiplied, etc. in the same way we do this with simple
> #numbers

> vector <- 1:20
> vector
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
> matr <- matrix(vector,4,5)
> matr
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    5    9   13   17
[2,]    2    6   10   14   18
[3,]    3    7   11   15   19
[4,]    4    8   12   16   20
> matr*3.5
     [,1] [,2] [,3] [,4] [,5]
[1,]  3.5 17.5 31.5 45.5 59.5
[2,]  7.0 21.0 35.0 49.0 63.0
[3,] 10.5 24.5 38.5 52.5 66.5
[4,] 14.0 28.0 42.0 56.0 70.0
> sqrt(vector)
 [1]  1.000000  1.414214  1.732051  2.000000  2.236068  2.449490  2.645751
2.828427 3.000000 3.162278 3.316625 3.464102 3.605551 3.741657
[15] 3.872983 4.000000 4.123106 4.242641 4.358899 4.472136

> #several functions work on vectors in the way that they extract
> #some simple number from them

> mean(vector)
[1] 10.5
> sum(vector)
[1] 210
> colSums(matr)
[1] 10 26 42 58 74
> rowMeans(matr)
[1]  9 10 11 12
```

```
> #we can name elements of matrices and vectors
>
> colnames(matr) <- LETTERS[1:5]
> rownames(matr) <- LETTERS[15:18]
> matr
  A B  C  D  E
O 1 5  9 13 17
P 2 6 10 14 18
Q 3 7 11 15 19
R 4 8 12 16 20
>
> #other useful types of data structures are lists
> #they can contain data of different types (text,numbers,etc)
> #whereas vector convert data to one type or display error

> vector <- c(1,2,"word",c(22,33))
> list1 <- list(1,2,"word",c(22,33))
> vector
[1] "1"    "2"    "word" "22"   "33"
> list1
[[1]]
[1] 1

[[2]]
[1] 2

[[3]]
[1] "word"

[[4]]
[1] 22 33


> #mostly used data structures in R are data frames
> #you can load your own data to R like this:
> #mydata <- read.table("file",sep="\t",head=T) for .-delimited decimals
> #mydata <- read.delim2("file",sep="\t",head=T) for ,-delimited decimals
> #in both examples data are separated by Tabs and first row contains names
> #you can also use built-in data from R
> data(ChickWeight)
> summary(ChickWeight)
     weight           Time           Chick     Diet
 Min.   : 35.0   Min.   : 0.00   13     : 12   1:220
 1st Qu.: 63.0   1st Qu.: 4.00   9      : 12   2:120
 Median :103.0   Median :10.00   20     : 12   3:120
 Mean   :121.8   Mean   :10.72   10     : 12   4:118
 3rd Qu.:163.8   3rd Qu.:16.00   17     : 12
 Max.   :373.0   Max.   :21.00   19     : 12
                                 (Other):506
> names(ChickWeight)
[1] "weight" "Time"   "Chick"  "Diet"
```

```
> #remember that in R rows are indexed first, then columns
> #you can also refer to columns using their names
> #here we simultaneously select columns and then extract some subset
> #of their data using indexes on vectors
> ChickWeight[,1][1:10]
 [1]  42  51  59  64  76  93 106 125 149 171
> ChickWeight[2,]
Grouped Data: weight ~ Time | Chick
  weight Time Chick Diet
2     51    2     1    1
> ChickWeight$Diet[205:220]
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
Levels: 1 2 3 4

> matr[,2] #access 2nd column of the matrix but loosing its dimensions
O P Q R
5 6 7 8
> matr[,2,drop=F] #keep right dimensions
  B
O 5
P 6
Q 7
R 8
>
> #sometimes instead of numerical indexes logical subscripts can be used

> ChickWeight[ChickWeight$Diet=="A",][1:20,]
Grouped Data: weight ~ Time | Chick
   weight Time Chick Diet
1      42    0     1    A
2      51    2     1    A
3      59    4     1    A
4      64    6     1    A
5      76    8     1    A
6      93   10     1    A
7     106   12     1    A
8     125   14     1    A
9     149   16     1    A
10    171   18     1    A
11    199   20     1    A
12    205   21     1    A
13     40    0     2    A
14     49    2     2    A
15     58    4     2    A
16     72    6     2    A
17     84    8     2    A
18    103   10     2    A
19    122   12     2    A
20    138   14     2    A
> ChickWeight[ChickWeight$Diet=="A"&ChickWeight$Time==18,]
Grouped Data: weight ~ Time | Chick
```

```
    weight Time Chick Diet
10     171   18     1    A
22     187   18     2    A
34     187   18     3    A
46     154   18     4    A
58     199   18     5    A
70     160   18     6    A
82     250   18     7    A
94     134   18     8    A
105    100   18     9    A
117    112   18    10    A
129    184   18    11    A
141    185   18    12    A
153     81   18    13    A
165    248   18    14    A
192    123   18    17    A
206    120   18    19    A
218    107   18    20    A
> ix<-which(ChickWeight$Diet=="B"&
+ (ChickWeight$Time==10|ChickWeight$Time==12))
> ix
 [1] 226 227 238 239 250 251 262 263 274 275 286 287 298 299 310
[16] 311 322 323 334 335
> ChickWeight[ix,]
Grouped Data: weight ~ Time | Chick
    weight Time Chick Diet
226    163   10    21    B
227    217   12    21    B
238     95   10    22    B
239    108   12    22    B
250    103   10    23    B
251    127   12    23    B
262     68   10    24    B
263     70   12    24    B
274    124   10    25    B
275    146   12    25    B
286    114   10    26    B
287    136   12    26    B
298    100   10    27    B
299    115   12    27    B
310    114   10    28    B
311    145   12    28    B
322    106   10    29    B
323    134   12    29    B
334     98   10    30    B
335    115   12    30    B

> #as you probably suspect negative indexes result in deletion of
> #respective columns/rows

> #as you've noticed some data are in the form of categorical variables
> #called factors
```

```
> ChickWeight$Diet[1:10]
 [1] A A A A A A A A A A
Levels: A B C D
> levels(ChickWeight$Diet)
[1] "A" "B" "C" "D"
> levels(ChickWeight$Diet)<-LETTERS[1:4]
> summary(ChickWeight)
     weight           Time           Chick     Diet
 Min.   : 35.0   Min.   : 0.00   13     : 12   A:220
 1st Qu.: 63.0   1st Qu.: 4.00   9      : 12   B:120
 Median :103.0   Median :10.00   20     : 12   C:120
 Mean   :121.8   Mean   :10.72   10     : 12   D:118
 3rd Qu.:163.8   3rd Qu.:16.00   17     : 12
 Max.   :373.0   Max.   :21.00   19     : 12
                                 (Other):506

> #you can use a set of functions to change and query the types
> #of objects

> as.character(ChickWeight$Diet)[205:220]
 [1] "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A" "A"
[16] "A"
> as.numeric(ChickWeight$Chick)[205:220]
 [1] 10 10 10 10  6  6  6  6  6  6  6  6  6  6  6  6
> is.factor(ChickWeight$Diet)
[1] TRUE
> is.numeric(ChickWeight$Chick)
[1] FALSE

> #NA is a special type of data where there's no value
> #to test if sth is NA you should use is.na(x) instead of x==NA


> #manipulating large datasets is easy thanks to several automating
> #functions

> #1 combine two matrices/vectors/data frames

> cbind(matr,4*matr) #column-wise
  A B  C  D  E  A  B  C  D  E
O 1 5  9 13 17  4 20 36 52 68
P 2 6 10 14 18  8 24 40 56 72
Q 3 7 11 15 19 12 28 44 60 76
R 4 8 12 16 20 16 32 48 64 80
> rbind(matr,4*matr) #row-wise
   A  B  C  D  E
O  1  5  9 13 17
P  2  6 10 14 18
Q  3  7 11 15 19
R  4  8 12 16 20
O  4 20 36 52 68
```

```
P  8 24 40 56 72
Q 12 28 44 60 76
R 16 32 48 64 80

> #2 apply some function column- or row-wise in matrix or data frame

> apply(matr,1,var)
 O  P  Q  R
40 40 40 40
> apply(matr,2,mean)
   A    B    C    D    E
 2.5  6.5 10.5 14.5 18.5

> #3 apply a function group-wise

> tapply(ChickWeight$weight,ChickWeight$Diet,mean)
       A        B        C        D
102.6455 122.6167 142.9500 135.2627
> tapply(ChickWeight$weight,ChickWeight$Chick,var)
         18          16          15          13           9
    8.00000    28.90476   107.83929   266.51515   424.69697
         20          10           8          17          19
  704.62879   831.90152  1064.00000  1033.54545  1406.38636
          4           6          11           3           1
 1905.33333  2201.29545  3156.26515  3686.15152  3332.96970
         12           2           5          14           7
 3701.35606  3881.90152  5237.15152  7323.69697  9129.27273
         24          30          22          23          27
  107.29545  1774.81818  1752.20455  2256.44697  2758.62879
         28          26          25          29          21
 4777.17424  5263.09091  6520.26515  8348.51515 12209.00000
         33          37          36          31          39
 1862.02273  2293.72727  5117.35606  5310.99242  6090.02273
         38          32          40          34          35
 7722.42424  8924.44697  9426.26515 11726.15152 15234.15152
         44          45          43          41          47
 1532.10000  2990.26515  3818.36364  3379.35606  3544.62879
         49          46          50          42          48
 4750.93182  4803.53788  6527.18182  7139.53788  9810.42424

> aggregate(ChickWeight$weight,list(ChickWeight$Diet),mean)
  Group.1        x
1       A 102.6455
2       B 122.6167
3       C 142.9500
4       D 135.2627

> #4 build a contingency table

> table(ChickWeight$Diet,ChickWeight$Chick)

    18 16 15 13  9 20 10  8 17 19  4  6 11  3  1 12  2  5 14  7
```

```
  A  2  7  8 12 12 12 12 11 12 12 12 12 12 12 12 12 12 12 12 12
  B  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  C  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  D  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0

    24 30 22 23 27 28 26 25 29 21 33 37 36 31 39 38 32 40 34 35
  A  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  B 12 12 12 12 12 12 12 12 12 12  0  0  0  0  0  0  0  0  0  0
  C  0  0  0  0  0  0  0  0  0  0 12 12 12 12 12 12 12 12 12 12
  D  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0

    44 45 43 41 47 49 46 50 42 48
  A  0  0  0  0  0  0  0  0  0  0
  B  0  0  0  0  0  0  0  0  0  0
  C  0  0  0  0  0  0  0  0  0  0
  D 10 12 12 12 12 12 12 12 12 12
> #--------------------------------------
> #houskeeping
> rm(matr) #removes selected object
> attach(ChickWeight) #attaches objects so that variables can be called
> detach(ChickWeight) #detaches object
> #attach is useful but can be dangerous - you can use with() instead
> attach(ChickWeight)
> ix<-which(Diet=="B"&(Time==10|Time==12))
> detach(ChickWeight)
> #OR
> with(ChickWeight,ix<-which(Diet=="B"&(Time==10|Time==12)))
> #not very longer and much safer ;)
>
> library(lme4) #loads library

Attaching package: 'lme4'

The following object(s) are masked from 'package:coda':

    HPDinterval

The following object(s) are masked from 'package:stats':

    AIC

> detach(package:lme4) #unloads library

> save(file="filename") #saves current workspace
> savehistory(file="filename2") #saves history of commands

> ls() #lists all objects
 [1] "ChickWeight"   "cw"            "cw2"
 [4] "ix"            "list1"         "Loblolly"
 [7] "m1"            "m11"           "m2"
[10] "m3"            "m4"            "matr2"
[13] "Orange"        "OrchardSprays" "vector"
```

```
> #--------------------------------------
> #generating useful data
> seq(1,10,0.5)
 [1]  1.0  1.5  2.0  2.5  3.0  3.5  4.0  4.5  5.0  5.5  6.0  6.5
[13]  7.0  7.5  8.0  8.5  9.0  9.5 10.0
> seq(10,1,-1)
 [1] 10  9  8  7  6  5  4  3  2  1

> rep(c(1,2),times=3)
[1] 1 2 1 2 1 2
> rep(c(1,2),each=3)
[1] 1 1 1 2 2 2
> rep(c(1,2),each=3,times=3)
 [1] 1 1 1 2 2 2 1 1 1 2 2 2 1 1 1 2 2 2

> gl(4,5,labels=c("A","B","C","D"))
 [1] A A A A A B B B B B C C C C C D D D D D
Levels: A B C D
> sample(1:100,10)
 [1] 97 30 49 75  1 35 73 77 52 67
> sample(1:10,10,rep=T)
 [1] 2 1 2 4 5 9 3 2 9 5

> #operations on distributions
> rnorm(10,mean=2,sd=sqrt(2)) #10 numbers from normal distribution
 [1]  4.2036313  2.7054377  3.6804485  3.2729313  3.6927785
 [6]  0.8945778  1.0215094  2.3764460  3.5724806 -0.4478660
> pnorm(2,mean=2,sd=sqrt(2)) #proportion of distribution to to X
[1] 0.5
> qnorm(0.5,mean=2,sd=sqrt(2)) #quantile of distribution for P
[1] 2
> dnorm(0.5, mean=2, sd=sqrt(2)) #density of distribution at X
[1] 0.1607328

> #most popular distribution are:
> #norm - gaussian
> #pois - poisson
> #binom - binomial
> #gamma - gamma
> #beta - beta
> #chisq - Chi squared
> #unif - uniform
> #logis - logistic
> #lnrom - log-normal
> #--------------------------------------


> #basic analyses and interpretation
> #linear model - gaussian data
> m1<-lm(weight~Time*Diet,data=ChickWeight)
> #here we add higher-order terms to allow for curvilinear trends
> m2<-lm(weight~poly(Time,2,raw=T)*Diet,data=ChickWeight)
```

```
> summary(m1)

Call:
lm(formula = weight ~ Time * Diet, data = ChickWeight)

Residuals:
     Min       1Q   Median       3Q      Max
-135.425  -13.757   -1.311   11.069  130.391

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  30.9310     4.2468   7.283 1.09e-12 ***
Time          6.8418     0.3408  20.076  < 2e-16 ***
DietB        -2.2974     7.2672  -0.316  0.75202
DietC       -12.6807     7.2672  -1.745  0.08154 .
DietD        -0.1389     7.2865  -0.019  0.98480
Time:DietB    1.7673     0.5717   3.092  0.00209 **
Time:DietC    4.5811     0.5717   8.014 6.33e-15 ***
Time:DietD    2.8726     0.5781   4.969 8.92e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 34.07 on 570 degrees of freedom
Multiple R-squared: 0.773,      Adjusted R-squared: 0.7702
F-statistic: 277.3 on 7 and 570 DF,  p-value: < 2.2e-16

> summary.aov(m1)
             Df  Sum Sq Mean Sq  F value    Pr(>F)
Time          1 2042344 2042344 1759.757 < 2.2e-16 ***
Diet          3  129876   43292   37.302 < 2.2e-16 ***
Time:Diet     3   80804   26935   23.208 3.474e-14 ***
Residuals   570  661532    1161
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> summary(m2)

Call:
lm(formula = weight ~ poly(Time, 2, raw = T) * Diet, data = ChickWeight)

Residuals:
     Min       1Q   Median       3Q      Max
-143.152  -10.030   -0.732    8.232  123.298

Coefficients:
                               Estimate Std. Error t value
(Intercept)                    38.36142    5.65497   6.784
poly(Time, 2, raw = T)1         4.47324    1.25962   3.551
poly(Time, 2, raw = T)2         0.11202    0.05742   1.951
DietB                          -0.68130    9.74243  -0.070
DietC                           0.46254    9.74243   0.047
DietD                          -1.29627    9.75110  -0.133
poly(Time, 2, raw = T)1:DietB   1.33695    2.14254   0.624
```

```
poly(Time, 2, raw = T)2:DietB  0.01827    0.09677   0.189
poly(Time, 2, raw = T)1:DietC  0.58427    2.14254   0.273
poly(Time, 2, raw = T)2:DietC  0.18428    0.09677   1.904
poly(Time, 2, raw = T)1:DietD  3.28497    2.15223   1.526
poly(Time, 2, raw = T)2:DietD -0.02018    0.09770  -0.207
                             Pr(>|t|)
(Intercept)                  2.96e-11 ***
poly(Time, 2, raw = T)1      0.000415 ***
poly(Time, 2, raw = T)2      0.051574 .
DietB                        0.944274
DietC                        0.962150
DietD                        0.894292
poly(Time, 2, raw = T)1:DietB 0.532877
poly(Time, 2, raw = T)2:DietB 0.850357
poly(Time, 2, raw = T)1:DietC 0.785182
poly(Time, 2, raw = T)2:DietC 0.057375 .
poly(Time, 2, raw = T)1:DietD 0.127491
poly(Time, 2, raw = T)2:DietD 0.836412
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 33.53 on 566 degrees of freedom
Multiple R-squared: 0.7817,     Adjusted R-squared: 0.7774
F-statistic: 184.2 on 11 and 566 DF,  p-value: < 2.2e-16

> summary.aov(m2)
                            Df  Sum Sq Mean Sq F value
poly(Time, 2, raw = T)       2 2064290 1032145 918.082
Diet                         3  129682   43227  38.450
poly(Time, 2, raw = T):Diet  6   84264   14044  12.492
Residuals                  566  636320    1124
                              Pr(>F)
poly(Time, 2, raw = T)      < 2.2e-16 ***
Diet                        < 2.2e-16 ***
poly(Time, 2, raw = T):Diet 3.031e-13 ***
Residuals
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> #note that results in LM output are presented as intercept (global mean
> #containing alphabetically first levels from fixed effects, and for
> #continuous predictors = 0) and then slopes for
> #covariates and differences between intercept
> #and each level of categorical predictors

> #if you want - youcan set-up contrasts to make specific comparisons
> #let's assume we'd like to compare: diet 1 with 2,3,4 and diet 3 and 4
> contrasts(ChickWeight$Diet)<-cbind(c(-3,1,1,1),c(0,0,-1,1))
> contrasts(ChickWeight$Diet)
  [,1] [,2]          [,3]
A   -3    0  1.665335e-16
B    1    0 -8.164966e-01
C    1   -1  4.082483e-01
```

```
D    1    1  4.082483e-01
> #R generated additional contrast to ensure orthogonality of comparisons
> #we can force two contrasts but be careful using contrasts that are
> #non-orthogonal!
> contrasts(ChickWeight$Diet,how.many=2)<-cbind(c(-3,1,1,1),c(0,0,-1,1))
> contrasts(ChickWeight$Diet)
  [,1] [,2]
A   -3    0
B    1    0
C    1   -1
D    1    1
> m3<-lm(weight~Time*Diet,data=ChickWeight)
> summary(m3)

Call:
lm(formula = weight ~ Time * Diet, data = ChickWeight)

Residuals:
      Min        1Q    Median        3Q       Max
-159.9001  -13.8325   -0.5892   11.9680  130.3913

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  27.2188     2.8414   9.579  < 2e-16 ***
Time          9.1361     0.2230  40.968  < 2e-16 ***
Diet1        -1.2374     1.3973  -0.886  0.37622
Diet2         6.4049     4.2881   1.494  0.13582
Time:Diet1    0.7648     0.1110   6.890 1.48e-11 ***
Time:Diet2   -0.8761     0.3360  -2.608  0.00935 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 34.96 on 572 degrees of freedom
Multiple R-squared: 0.7601,     Adjusted R-squared: 0.758
F-statistic: 362.5 on 5 and 572 DF,  p-value: < 2.2e-16

> summary.aov(m3)
             Df  Sum Sq Mean Sq  F value    Pr(>F)
Time          1 2042344 2042344 1670.757 < 2.2e-16 ***
Diet          2  106275   53138   43.470 < 2.2e-16 ***
Time:Diet     2   66721   33360   27.291 4.783e-12 ***
Residuals   572  699216    1222
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
>
> #contrasts are compatible with most modelling functions
> #we can use them even with mixed models
> library(MCMCglmm)
> m4 <- MCMCglmm(weight~Diet, random=~Chick,data=ChickWeight,verbose=F)
> summary(m4) #2 contrasts instead of 3 levels for Diet 2,3,4

 Iterations = 12991
```

```
 Thinning interval  = 3001
 Sample size  = 1000

 DIC: 6534.775

 G-structure:  ~Chick

      post.mean l-95% CI u-95% CI eff.samp
Chick     317.3    22.11    622.5      556

 R-structure:  ~units

      post.mean l-95% CI u-95% CI eff.samp
units      4570     3998     5088    978.2

 Location effects: weight ~ Diet

            post.mean l-95% CI u-95% CI eff.samp  pMCMC
(Intercept)   125.283  117.381  132.716     1000 <0.001 ***
Diet1           7.840    4.243   12.009     1000 <0.001 ***
Diet2          -3.916  -15.894    7.223     1000   0.52
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

We will cover more sophisticated issues together with the rest of our workshop. Also – don't hesitate to ask if you don't know why I'm doing what I'm doing! ;)

Part 1

Overview

1. Presentation
    a. Essence of conventional and Bayesian statistics
    b. Likelihood-based methods
    c. MCMC approximation – how it works?
    d. Linear models – brief summary of underlying 'mechanics'; link functions; etc.
2. DIY – introduction to Bayes' way of thinking
    a. Brief summary of R – what do you need to know before entering MCMC world
    b. Brief summary of R – interpreting linear models, contrasts and why is it so confusing?
    c. Working with pure likelihood on simulated data
    d. Overview of **`MCMCglmm`** – arguments, calls, accessing results
    e. Priors – what is it, how it works, how to choose?
    f. Combining priors with likelihood – let's go Bayesian
    g. Unusual priors – improper priors
3. DIY – how to begin?
    a. Simple model with real data – Gaussian data in **`MCMCglmm`** and **`lmer`**
    b. When NOT to use **`lmer`**? Why one should use **`MCMCglmm`**?
    c. Diagnostics of **`MCMCglmm`**
    d. Non-Gaussian data: Poisson and overdispersion
    e. Non-Gaussian data: binomial and binary data, fixing (co)variance priors
    f. Random effects in Bayesian framework?
    g. Random interactions and confusions caused thereby

Likelihood – traditional approach

Maximum likelihood estimators are common in classical statistics. For instance, arithmetic mean, OLS estimates of regression coefficients – all are in fact estimators that maximize the likelihood of data given particular values of parameters, i.e. $max(P(\mathbf{y}|par))$. In general situations as considered here this likelihood is proportional to the product of probability densities of the data given particular values of parameters:

$L(par|\mathbf{y}) \sim \Pi_i P(\mathbf{y}|par_i)$

In this part we'll play with simple simulated Gaussian data and see how simple maximum likelihood estimation works. We'll learn how to use optimizing functions of R (which some of

you may find useful in other applications) and how to produce multivariate graphs of likelihood surfaces.

Maximum likelihood

First – we'll simulate simple normal data (10 observations) and see how do they look like on the distributions they were taken from. Likelihoods may be tricky and as you'll see – the likelihood of our data may be higher for different (!) parameters than those we'll use to simulate them.

```
> ###code block 1
>
> dataG <- data.frame(y = rnorm(10,mean=0,sd=sqrt(1)))
> dataG$y
 [1] -0.2079101 -1.1445615 -0.0656215 -0.6294617  0.5422668
 [6]  0.7025364  0.7627269  0.1905778  1.6687900  2.0852642
>
> yscale <- seq(-3,3,0.1) #possible values of y for the plot
> Prob<-dnorm(yscale,mean=0,sd=sqrt(1)) #pdf
> plot(Prob~yscale,type="l")
> Prob.y <- dnorm(dataG$y, mean=0, sd=sqrt(1))
> points(Prob.y~dataG$y)
>
> L <- prod(Prob.y) #likelihood
> L
[1] 5.94403e-07
>
> Lalt <- prod(dnorm(dataG$y,mean=0,sd=sqrt(0.5)))
> Lalt
[1] 1.107162e-07
>
>
> plot(dnorm(yscale,0,sqrt(0.5))~yscale,type="l")
> lines(Prob~yscale,col="red")
> points(Prob.y~dataG$y,col="red")
> points(dnorm(dataG$y,0,sqrt(0.5))~dataG$y)
```

As you can see – the likelihood of our data is higher under different set of parameters and it's apparent from the plot. In order to fully understand what's happening here we should evaluate the likelihood on the grid of possible parameters. Here we'll use simple loop to iterate through the space of our parameters (mean and variance) to calculate possible values of *L* and then we'll plot them as a flattened perspective plot (contours). Be aware that each of you has slightly different values in you simulated data (in **`rnorm`** 'r' means random!) and you'll probably have to rescale your plots so that they could contain whole likelihood surface.

```
> ###code block 2
>
> x=seq(-1,1,0.05)
> y=seq(0,2,0.05)
> z=matrix(numeric(length(x)*length(y)),c(length(x),length(y)))
>
> for (i in 1:length(x)) {
```

```
+
+ for (j in 1:length(y)){
+
+ z[i,j]=prod(dnorm(dataG$y,mean=x[i],sd=sqrt(y[j])))
+ }
+ }
>
> z<-z/max(z)
> contour(x,y,z,nlevels=10,xlab="mean",ylab="variance")
```

Importantly we don't have to rely on visual inspection looking for ML estimator. We can use R built-in features designed for searching for functions maxima and minima. In such case you should define your maximized/minimized function (in our case it will be the likelihood which is the product – or, on a log scale, the sum – of probability densities. In the optimizing routine you have to specify starting parameters (which may be our assumed parameters of the distribution; these will be coordinates of the space in which optimization will be done) – they have to be of the same number as parameters in the optimized function, you also have to provide all variables that are in the optimized function. We'll compare our optimum with the estimates of a linear model (which uses REML instead of ML).

```
> ###code block 3
>
> loglik <- function(pars,y) {
+ sum(dnorm(y,pars[1],sqrt(pars[2]),log=TRUE))
+ }
>
> ML <- optim(c(mean=0,v=1),fn=loglik,y=dataG$y,
+ control=list(fnscale=-1,reltol=1e-16))
> ML$par
     mean         v
0.3904607 0.8768050
>
> REML <- glm(y~1,data=dataG)
> summary(REML)

Call:
glm(formula = y ~ 1, data = dataG)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.53502  -0.56280  -0.02404   0.35722   1.69480

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.3905     0.3121   1.251    0.242

(Dispersion parameter for gaussian family taken to be 0.9742278)

    Null deviance: 8.768  on 9  degrees of freedom
Residual deviance: 8.768  on 9  degrees of freedom
AIC: 31.064

Number of Fisher Scoring iterations: 2

>
> #REML estimator is better (the bias is smaller by
```

```
> #factor of n/n-1)
> ML$par["v"]*(10/9)
        v
0.9742278
```

MCMCglmm – syntax

Before we "go Bayesian" it's important to know the monster, namely the **MCMCglmm** package. **MCMCglmm** works simply as any other modeling routine in R. It's most important argument is the formula, defined as usual:

```
y ~ A + B*C + D + E + D:E + I(A^2)
```

Importantly, this formula takes only fixed effects. All random effects are introduced by the **random** argument, which takes the same formula syntax, but without the LHS (left-hand-site, namely the response; remember to retain the tilde!). Below you'll find most important argument and keywords used by MCMCglmm together with their meaning.

Argument	What it does?
random	formula for random effects
prior	prior probabilities for random effects (fixed effects priors are default)
data	your data
rcov	optional, contains random effects structure, needed in multivariate models
verbose	if FALSE you won't get status updates while MCMCglmm works
pr	saves posterior distribution of random effects (similar to BLUPs in REML)
pl	saves posterior distribution of latent variables
nitt	number of iterations
thin	every **thin** sample will be saved to posterior
burnin	how many beginning iterations will be omitted?
idh	covariance structure fixing covariances to zero
us	covariance structure allowing for estimation of covariances
units	odd name for residuals
trait	dummy fixed factor indexing traits in multivariate models
animal	used with pedigree, estimates animal/individual specific random effects corrected for dependence
at.level	chooses levels of the fixed effect to fit the model to
sir	used to define recursive models
family	distribution of the response(s)

saveX	saves the design matrix of fixed effects
saveZ	saves the design matrix of random effects
mev	provides vector of measurements errors in meta-analysis
pedigree	provides pedigree for animal models or phylogeny for comparative analyses
singular.ok	in case of collinear/not estimable fixed effects use this argument as TRUE to estimate them irrespectively of these problems; be careful interpreting them
DIC	default TRUE, if TRUE – Deviance Information Criterion is calculated

After executing the program you'll get a model object from which you can extract several things. Below we'll define simple mixed model on the data from simple agricultural split-plot experiment with blocks and three crossed experimental treatments. You'll learn how to access information in the output object and how to manipulate this information.

```
> ###code block 6a
>
> yield<-read.table("splityield.txt",head=T,sep="\t")
> summary(yield)
     yield        block      irrigation   density   fertilizer
 Min.   : 60.00   A:18   control  :36   high  :24   N :24
 1st Qu.: 86.00   B:18   irrigated:36   low   :24   NP:24
 Median : 95.00   C:18                  medium:24   P :24
 Mean   : 99.72   D:18
 3rd Qu.:114.00
 Max.   :136.00
>
> m1.yield <- MCMCglmm(yield~(irrigation+density+fertilizer)^2-
+ density:fertilizer,random=~block, data=yield,
+ verbose=F, prior=list(R=list(V=1,nu=0.002),
+ G=list(G1=list(V=1,nu=0.002))))
>
> plot(m1.yield$Sol)
Waiting to confirm page change...
Waiting to confirm page change...
Waiting to confirm page change...
> plot(m1.yield$VCV)
>
> HPDinterval(m1.yield$Sol)
                                        lower      upper
(Intercept)                          72.8982436  90.244904
irrigationirrigated                  16.0979333  39.673239
densitylow                           -4.6478603  13.467672
densitymedium                         1.2938722  20.161037
fertilizerNP                         -4.2757404  13.881927
fertilizerP                          -7.6785447  11.146146
irrigationirrigated:densitylow    -42.5905389 -16.324538
irrigationirrigated:densitymedium -32.3313328  -5.104103
```

```
irrigationirrigated:fertilizerNP    1.8248399  28.217214
irrigationirrigated:fertilizerP     0.7738581  27.669071
attr(,"Probability")
[1] 0.95
> HPDinterval(m1.yield$VCV)
             lower     upper
block 3.446255e-04  11.11558
units 9.147564e+01 189.55837
attr(,"Probability")
[1] 0.95
>
> posterior.mode(m1.yield$Sol)
                      (Intercept)
                        81.729989
               irrigationirrigated
                        28.476032
                       densitylow
                         4.655881
                    densitymedium
                        10.125696
                     fertilizerNP
                         5.427540
                      fertilizerP
                         1.466348
   irrigationirrigated:densitylow
                       -28.522910
irrigationirrigated:densitymedium
                       -16.715192
 irrigationirrigated:fertilizerNP
                        16.827155
  irrigationirrigated:fertilizerP
                        15.400264
> posterior.mode(m1.yield$VCV)
       block        units
 -0.05737882 123.32842262
```

It's important to realize that both **`Sol`** and **`VCV`** tables are not simply single numerical values, but they contain all posterior samples for the given parameter. Thus, distributions you see after using **`plot`** are not analytically derived but they're original, smoothed histograms based on these random posterior samples.

Combining prior knowledge – prior distributions

What's unique for Bayesian analysis is that we consider parameters as random rather than fixed and we use some knowledge about these parameters to estimate their values. In other words, the posterior probability of observing parameters of a given value depends both on the likelihood of the data of given these parameters and our prior knowledge about them:

$$P(par|\mathbf{y}) \sim L(par|\mathbf{y})P(par) \sim P(\mathbf{y}|par)P(par)$$

Diverse distributions could be used in the Bayesian framework to define priors but in our analyses we'll use two of them. Priors for fixed effects are defined using normal distribution with mean zero and very large (>1e+06) variance, making such prior essentially flat and uninformative. For (co)variances we use inverse Wishart distribution (IW) which is slightly problematic for multivariate (co)variance structures (and we'll come back to them later). For simple variances IW is defined by two parameters: variance – **V** and belief parameter – **nu**. When belief goes to infinity, the distribution tends to a mode equal to **V**. In general the mode of the distribution is **(V*nu)/nu+2**. In R we can model IW using inverse gamma distribution (e.g. function **dinvgamma**) with parameters: **shape=nu/2** and **scale=nu*V/2**). Care is needed to ensure that the prior is proper (integrates to one as an ordinary distribution) and this condition holds for single variance components when **V>0** and **nu>0**. When **nu≤0** we get improper prior which – although difficult – may be useful (as we'll see below).

Here we'll combine our likelihood function with prior densities to see how such estimates work compared to ML. First we'll define function for calculating prior probability for given values of parameters, then we'll combine these with likelihood and use to estimate values of the parameters. Since we're working on the log scale, it's summing and not multiplying that we'll employ.

```
> ###code block 4
>
> library(MCMCpack)
Loading required package: MASS
##
## Markov Chain Monte Carlo Package (MCMCpack)
##
## Support provided by the U.S. National Science Foundation
## (Grants SES-0350646 and SES-0350613)
##
>
> logprior <- function(pars,priorR,priorB) {
+ dnorm(pars[1],mean=priorB$mu,sd=sqrt(priorB$V),log=T)+
+ log(dinvgamma(pars[2],shape=priorR$nu/2,
+ scale=(priorR$nu*priorR$V)/2))
+ }
>
> prior <- list(R=list(V=1,nu=0.002),B=list(mu=0,V=1e+08))
>
> loglikprior <- function(pars,y,priorR,priorB) {
+ loglik(pars,y)+logprior(pars,priorR,priorB)
+ }
>
> Bayes <- optim(c(mean=0,v=1),fn=loglikprior,y=dataG$y,
+ priorR=prior$R,priorB=prior$B,
+ control=list(fnscale=-1,reltol=1e-16))
>
>
> x=seq(-1,1,0.05)
> y=seq(0,2,0.05)
> z1=matrix(numeric(length(x)*length(y)),c(length(x),length(y)))
```

```
>
> for (i in 1:length(x)) {
+
+ for (j in 1:length(y)){
+
+ z1[i,j]=exp(loglikprior(c(x[i],y[j]),
+ dataG$y,prior$R,prior$B))
+ }
+ }
>
> #z2<-z1/max(z1) sometimes does not work as NaNs are produced
> contour(x,y,z,nlevels=10,xlab="mean",ylab="variance")
> contour(x,y,z1,nlevels=10,xlab="mean",ylab="variance",
+ add=T,col="red")
```

As you can see – variance estimates using prior are even more downwardly biased – which is caused by the fact that simple optimization of the **`L*prior`** ignores uncertainty of the mean estimate. We can however integrate our bivariate distribution along the mean scale to get the posterior for variance, which would take uncertainty in mean into account:

$$P(\sigma^2|\boldsymbol{y}) \sim \int P(\sigma^2, \mu|\boldsymbol{y}) d\mu$$

Important advantage of MCMC-based methods is that analytically it's most often impossible to get the posterior marginal distribution of a parameter.

```
> ###code block 5
>
> contour(x,y,z1,nlevels=10,xlab="mean",ylab="variance",
+ col="red")
> library(MCMCglmm)
> m1 <- MCMCglmm(y~1,data=dataG,prior=prior,thin=1,nitt=30000,
+ verbose=F)
> points(cbind(m1$Sol,m1$VCV),pch=".")
```

Of course, one would ask how sure we can be that our sample space (visualized above) is appropriate and guaranties we're integrating true distribution (i.e. integrating to one using the boundaries of our space)? If we look at the proportion of samples from the posterior contained within considered sample space you'll see it's almost 1. Thus, we can construct the posterior distribution safely over this range (try using whole sample from the posterior – such histogram would be impossible to interpret).

```
> ###code block 6
>
> prop.table(table(m1$Sol > -1 & m1$Sol<1 & m1$VCV<2))

    FALSE      TRUE
0.1444815 0.8555185
> hist(m1$VCV[which(m1$VCV<2)],breaks=30)
> abline(v=Bayes$par["v"],col="red")
```

Improper priors – let's be nasty

As I mentioned, sometimes priors are not proper, i.e. they don't integrate to unity. The simplest example is a uniform prior defined over **R**. It's not proper since it integrates to +∞. Uniform prior would be proper only when defined over the range $A=[a,b]\subset\mathbf{R}$ so that $P(x\in A)(b-a)=1$.

In case of IW-distributed priors for single variance components they're improper when **nu≤0**. For nu=0 we get flat prior for variance. This reduces well known Bayesian equation to simple ML estimator: $P(par|\mathbf{y})\sim P(\mathbf{y}|par)$. In other words – the joint posterior distribution will be equal to ML estimator but remember – modes you're getting analyzing problems are from marginal distributions, not from the joint one.

```
> ###code block 7
>
> prior.fl <- list(R=list(V=1,nu=0))
> m1.fl <- MCMCglmm(y~1, data=dataG, thin=1,nitt=100000,
+ prior=prior.fl,verbose=F)
```

We may also define prior that will be non-informative for the variance and this could be achieved by setting V=0 and nu=-2. Such prior makes joint posterior to deviate from ML estimates but marginal estimates of variance are in agreement with REML.

In general – priors in MCMCglmm lead often to confusion. Several conventions exist for defining them. E.g. improper priors can be useful in a way that they allow for reducing our problem to simple ML estimator or REML estimator (for marginal distributions of parameters). However, improper priors must be used with caution – improper prior distribution may lead to improper posterior distribution, which would be meaningless from a statistical point of view. The question is – which strategy to adopt in defining priors? First of all – use weak priors unless you want to impose some (strong) constraints on the variance. In general – having good data, with appropriate levels of replication, and sampling populations of random effects accordingly should make priors less influential – in other words, when the data contain enough information to estimate the parameters, priors should not influence these estimates. In case of less informative data you might consider using improper priors, but be extremely cautious. From my point of view two approaches are recommended: use either priors with **V=1** and **nu=0.002** or calculate the variance from your data and use it (partitioned accordingly with respect to random effects) as values for **V**. You'll see these approaches in further parts of this workshop.

Fitting simple model in MCMCglmm and lmer

Here we'll use two packages – MCMCglmm and lme4 (more precisely lmer function) to fit linear mixed models. Loading both packages may lead to masking some useful functions, so remember to call **coda::HPDinterval** instead of **HPDinterval**, if the latter does not work. Alternatively, remember to clean your workspace and do **detach(package:lme4)** before proceeding further. The data we'll be using are centered data on blue tits: tarsus lengths and back colours, together with the information on genetic and social mothers (dams and fosternests; chicks were cross-fostered).

```
> ###code block 8
```

```
>
> library(lme4)

Attaching package: 'lme4'

The following object(s) are masked from 'package:coda':

    HPDinterval

The following object(s) are masked from 'package:stats':

    AIC

>
> data(BTdata)
> lm1.bt1 <- lmer(tarsus~sex+(1|fosternest)+(1|dam),data=BTdata)
> summary(lm1.bt1)
Linear mixed model fit by REML
Formula: tarsus ~ sex + (1 | fosternest) + (1 | dam)
   Data: BTdata
  AIC  BIC logLik deviance REMLdev
 2087 2115  -1038     2065    2075
Random effects:
 Groups     Name        Variance Std.Dev.
 dam        (Intercept) 0.220259 0.46932
 fosternest (Intercept) 0.069204 0.26307
 Residual               0.567919 0.75360
Number of obs: 828, groups: dam, 106; fosternest, 104

Fixed effects:
            Estimate Std. Error t value
(Intercept) -0.40566    0.06706  -6.049
sexMale      0.76879    0.05714  13.455
sexUNK       0.21043    0.12670   1.661

Correlation of Fixed Effects:
        (Intr) sexMal
sexMale -0.449
sexUNK  -0.210  0.224
> lm1.bt2 <- lmer(tarsus~sex+(1|dam),data=BTdata)
> anova(lm1.bt1,lm1.bt2)
Data: BTdata
Models:
lm1.bt2: tarsus ~ sex + (1 | dam)
lm1.bt1: tarsus ~ sex + (1 | fosternest) + (1 | dam)
        Df    AIC    BIC  logLik  Chisq Chi Df Pr(>Chisq)
lm1.bt2  5 2086.7 2110.2 -1038.3
lm1.bt1  6 2077.1 2105.4 -1032.6 11.518      1  0.0006893 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> lm1.bt3 <- lmer(tarsus~sex,data=BTdata)
Error in lmerFactorList(formula, fr, 0L, 0L) :
```

```
  No random effects terms specified in formula
>
> library(nlme)

Attaching package: 'nlme'

The following object(s) are masked from 'package:lme4':

    BIC, fixef, lmList, ranef, VarCorr

> lm1.bt3 <- lme(tarsus~sex,random=~1|dam,data=BTdata)
> lm1.bt3a <- lm(tarsus~sex,data=BTdata)
> anova(lm1.bt3,lm1.bt3a)
         Model df      AIC      BIC    logLik   Test  L.Ratio
lm1.bt3      1  5 2096.757 2120.334 -1043.379
lm1.bt3a     2  4 2237.927 2256.789 -1114.964 1 vs 2 143.1703
         p-value
lm1.bt3
lm1.bt3a  <.0001
> detach(package:lme4)
> detach(package:nlme)
```

As you can see, working with **lmer** is quite simple, just remember the way random effects are specified: **(1|effect)**. However, it has some drawbacks. You can do stepwise simplification of your model, using likelihood ratio tests with **anova()** but only until you have one random effect. Then simplification won't work as **lmer** cannot fit models without random effects. To proceed with the simplification you have to fit one model (with one last random effect) using **lme()** from library **nlme** and then compare it with model fitted using **lm()** and not containing any random effects. Also, remember that here you should use **@** rather than **$** to access elements of the model object, e.g. **model@ranef** and **model@fixef** (in **glm** it would be **model$coef**).

Working with lmer can be further extended to more complex models, where categorical interactions in random effects are defined. It simply requires replacing ones in random effect specifications with appropriate formulae defining effects that interact with random terms. Here we're fitting model that allows for different effects of dams in different sexes + allows for estimating covariances between sexes with respect to this random effects.

```
> summary(lm2.bt)
Linear mixed model fit by REML
Formula: tarsus ~ sex + (1 | fosternest) + (sex | dam)
   Data: BTdata
  AIC  BIC logLik deviance REMLdev
 2097 2149  -1037     2065    2075
Random effects:
 Groups     Name        Variance   Std.Dev. Corr
 dam        (Intercept) 0.22730365 0.476764
            sexMale     0.00035282 0.018784 -1.000
            sexUNK      0.00464764 0.068174  1.000 -1.000
 fosternest (Intercept) 0.06645438 0.257787
```

```
 Residual               0.56805496 0.753694
Number of obs: 828, groups: dam, 106; fosternest, 104

Fixed effects:
            Estimate Std. Error t value
(Intercept) -0.40616    0.06738  -6.028
sexMale      0.77018    0.05714  13.479
sexUNK       0.19938    0.12878   1.548

Correlation of Fixed Effects:
        (Intr) sexMal
sexMale -0.470
sexUNK  -0.171  0.218
> matrix(VarCorr(lm1.bt)$dam,3,3)
Error: object 'lm1.bt' not found
Error in VarCorr(lm1.bt) :
  error in evaluating the argument 'x' in selecting a method for function
'VarCorr'
> matrix(VarCorr(lm2.bt)$dam,3,3)
            [,1]          [,2]         [,3]
[1,]  0.22730365 -0.0089553503  0.032502689
[2,] -0.00895535  0.0003528245 -0.001280547
[3,]  0.03250269 -0.0012805468  0.004647637
```

Variance structure we've used here is extreme: we allow for both variance differences between levels of fixed effects and covariances>0. We could specify this variance structure in different way, putting different restrictions on (co)variances. Table below gives several examples (adapted from Hadfield 2010). As you can see – **`lmer`** can fit much less (co)variance structures than **`MCMCglmm`**.

lmer	**MCMCglmm**	**(Co)variance**	**Correlation**
`(1|dam)`	`dam`	$\begin{bmatrix} V & V & V \\ V & V & V \\ V & V & V \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$
`(sex-1|dam)`	`us(sex):dam`	$\begin{bmatrix} V_{1,1} & C_{1,2} & C_{1,3} \\ C_{2,1} & V_{2,2} & C_{2,3} \\ C_{3,1} & C_{3,2} & V_{3,3} \end{bmatrix}$	$\begin{bmatrix} 1 & r_{1,2} & r_{1,3} \\ r_{2,1} & 1 & r_{2,3} \\ r_{3,1} & r_{3,2} & 1 \end{bmatrix}$
`(1|sex:dam)`	`sex:dam`	$\begin{bmatrix} V & 0 & 0 \\ 0 & V & 0 \\ 0 & 0 & V \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

`(1|dam)+(1|sex:dam)`	`dam+sex:dam`	$\begin{bmatrix} V_1+V_2 & V_1 & V_1 \\ V_1 & V_1+V_2 & 0V_1 \\ V_1 & V_1 & V_1+V_2 \end{bmatrix}$	$\begin{bmatrix} 1 & r & r \\ r & 1 & r \\ r & r & 1 \end{bmatrix}$
-	`idh(sex):dam`	$\begin{bmatrix} V_{1,1} & 0 & 0 \\ 0 & V_{2,2} & 0 \\ 0 & 0 & V_{3,3} \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
-	`corh(sex):dam`	$\begin{bmatrix} V_{1,1} & rV_{1,1}V_{2,2} & rV_{1,1}V_{3,3} \\ rV_{1,1}V_{2,2} & V_{2,2} & rV_{3,3}V_{2,2} \\ rV_{1,1}V_{3,3} & rV_{3,3}V_{2,2} & V_{3,3} \end{bmatrix}$	$\begin{bmatrix} 1 & r & r \\ r & 1 & r \\ r & r & 1 \end{bmatrix}$
-	`cor(sex):dam`	$\begin{bmatrix} 1 & r_{1,2} & r_{1,3} \\ r_{2,1} & 1 & r_{2,3} \\ r_{3,1} & r_{3,2} & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & r_{1,2} & r_{1,3} \\ r_{2,1} & 1 & r_{2,3} \\ r_{3,1} & r_{3,2} & 1 \end{bmatrix}$

You can specify models with different types of (co)variance and compare them using likelihood-ratio tests.

One more disadvantage of **`lmer`** is the way you test the effects from the model. In the summary, random effects have just variance values (and testing random effects is done best by using likelihood-ratio tests; putting standard errors on variances from REML is dangerous and simple Wald-tests should be avoided at all costs). For fixed effects its worse: only t-values are provided with no df's. We can use them to test fixed effects suing some conservative values of df, e.g. number of records – sum(number of levels in i-th effect). See example below:

```
> ###code block 10
>
> tv<-summary(lm2.bt)@coefs[,3][2]
> #access to t-value for sexMale
> df<-dim(BTdata)[1]-nlevels(BTdata$dam)-
+ nlevels(BTdata$fosternest)
> 2*(1-pt(tv,df))
sexMale
      0
> #two-tailed test, beware of the sign of tv
>
> Fv<-anova(lm1.bt1)[,4][1]
> 1-pf(Fv,2,df)
[1] 0
```

How to do the same things in **MCMCglmm**? First we'll fit the simple model with both random effects and see if removal of the fosternest effect is still justified. As **MCMCglmm** doesn't generate likelihoods, we'll have to use DIC to check for significance of random effects. Putting confidence intervals on variance functions is easy and we'll see it for the proportion of variance explained by dam effect. Next we'll fit similar sex-specific model and see if conclusions hold.

```
> ###code block 11
>
> prior <- list(R=list(V=1,nu=0.002),
+ G=list(G1=list(V=1,nu=0.002),
+ G2=list(V=1,nu=0.002)))
> m2.bt1 <- MCMCglmm(tarsus~sex, random=~fosternest+dam,
+ prior=prior,verbose=F, data=BTdata)
> summary(m2.bt1)

 Iterations = 12991
 Thinning interval  = 3001
 Sample size  = 1000

 DIC: 1992.66

 G-structure:  ~fosternest

           post.mean l-95% CI u-95% CI eff.samp
fosternest   0.06633 0.007809   0.1265    277.7

              ~dam

    post.mean l-95% CI u-95% CI eff.samp
dam    0.2266   0.1368   0.3264    854.9

 R-structure:  ~units

      post.mean l-95% CI u-95% CI eff.samp
units    0.5727   0.5023   0.6309    768.7

 Location effects: tarsus ~ sex

            post.mean l-95% CI u-95% CI eff.samp  pMCMC
(Intercept)  -0.41128 -0.53400 -0.27243     1549 <0.001 ***
sexMale       0.76993  0.67216  0.88232     1000 <0.001 ***
sexUNK        0.20186 -0.04125  0.45252     1000  0.114
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
>
> m2.bt2 <- MCMCglmm(tarsus~sex, random=~dam, prior=prior,
+ verbose=F, data=BTdata)
> m2.bt1$DIC; m2.bt2$DIC #it seems we should keep fosternest
[1] 1992.66
[1] 2014.644
> plot(m2.bt1$VCV)
>
```

```
> prior <- list(R=list(V=1,nu=0.002),
+ G=list(G1=list(V=1,nu=0.002),
+ G2=list(V=diag(3),nu=1.002)))
> m2.bt3 <- MCMCglmm(tarsus~sex, random=~fosternest+us(sex):dam,
+ prior=prior,verbose=F, data=BTdata)
Warning message:
In MCMCglmm(tarsus ~ sex, random = ~fosternest + us(sex):dam, prior =
prior,  :
  some combinations in us(sex):dam do not exist and 75 missing records have
been generated
> summary(m2.bt3)

 Iterations = 12991
 Thinning interval  = 3001
 Sample size  = 1000

 DIC: 2010.066

 G-structure:  ~fosternest

           post.mean l-95% CI u-95% CI eff.samp
fosternest   0.06122 0.002835   0.1181    328.6

               ~us(sex):dam

              post.mean l-95% CI u-95% CI eff.samp
Fem:Fem.dam      0.2945  0.17344   0.4310   1000.0
Male:Fem.dam     0.1919  0.08804   0.2836    932.1
UNK:Fem.dam      0.2076  0.03547   0.4119    445.4
Fem:Male.dam     0.1919  0.08804   0.2836    932.1
Male:Male.dam    0.2726  0.15343   0.3905    910.0
UNK:Male.dam     0.2035  0.03732   0.4096    453.8
Fem:UNK.dam      0.2076  0.03547   0.4119    445.4
Male:UNK.dam     0.2035  0.03732   0.4096    453.8
UNK:UNK.dam      0.4720  0.11445   0.9037    330.8

 R-structure:  ~units

      post.mean l-95% CI u-95% CI eff.samp
units    0.5547   0.4944   0.6192     1000

 Location effects: tarsus ~ sex

            post.mean l-95% CI u-95% CI eff.samp  pMCMC
(Intercept)   -0.4054  -0.5391  -0.2557     1000 <0.001 ***
sexMale        0.7729   0.6407   0.9050     1000 <0.001 ***
sexUNK         0.1946  -0.1224   0.5021     1000  0.218
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> coda::HPDinterval(m2.bt3$VCV)
                    lower     upper
fosternest    0.002834977 0.1181360
```

```
Fem:Fem.dam   0.173436520 0.4309517
Male:Fem.dam  0.088043697 0.2836029
UNK:Fem.dam   0.035469593 0.4119485
Fem:Male.dam  0.088043697 0.2836029
Male:Male.dam 0.153425620 0.3904530
UNK:Male.dam  0.037318849 0.4096160
Fem:UNK.dam   0.035469593 0.4119485
Male:UNK.dam  0.037318849 0.4096160
UNK:UNK.dam   0.114453067 0.9037068
units         0.494441135 0.6191769
attr(,"Probability")
[1] 0.95
> r <- m2.bt3$VCV[,3]/sqrt(m2.bt3$VCV[,2]*m2.bt3$VCV[,6])
> coda::HPDinterval(r)
         lower     upper
var1 0.5093113 0.8522105
attr(,"Probability")
[1] 0.95
> m2.bt1$DIC; m2.bt3$DIC
[1] 1992.66
[1] 2010.066
>
> #correlations for different variance components are negative
> cor(m2.bt1$VCV)
           fosternest         dam       units
fosternest  1.0000000 -0.25064407 -0.22133074
dam        -0.2506441  1.00000000 -0.04342781
units      -0.2213307 -0.04342781  1.00000000
>
> #we could test dam effect in usual way but instead we'll see
> #how big proportion of variance it explains
>
> prop.v <- m2.bt1$VCV[,2]/rowSums(m2.bt1$VCV)
> coda::HPDinterval(prop.v)
         lower     upper
var1 0.1778132 0.3460457
attr(,"Probability")
[1] 0.95
> #if you remember to detach lmer after use, you can skip coda::
```

You should have already noticed great deal of advantages when using **`MCMCglmm`**. First of all – testing of random effects is not based on likelihood, which may be extremely biased for non-gaussian data; here you use DIC values. Secondly, you get straightforward tests of significance of fixed effects. Thirdly, calculating any variance-derived values (such as correlations) and putting confidence intervals on variance components couldn't be easier: you can simply manipulate whole distributions, stored in consecutive columns of model$VCV, add, substract, multiply, square and divide them. Derived numbers also have some posterior distributions, so you can easily put CIs on them as well. So the point is – when use **`lmer`** and when **`MCMCglmm`**.

Short guide – how to choose best method?

This is simple: if you have good, well replicated Gaussian data, with lots of information on large numbers of random effects' levels – use **lmer**. It performs well, but remember that significance tests may be a little cumbersome. However, if you want to fit categorical random interactions – avoid using **lmer**. It doesn't allow for residual variances to differ between levels of fixed effect and thus any differences here could possibly be confounded with the differences in variances associated with a particular random effect. For categorical interactions use **MCMCglmm**.

In case of non-Gaussian data use **MCMCglmm** – REML methods are not able to analytically derive likelihood in such data and work on approximations. If such approximations are then used in likelihood-ratio tests – results may not be reliable.

Finally – remember that Poisson and binomial data are almost always overdispersed. **lmer** has this famous "**quasi**" prefix for such distributions that should deal with it. However, it doesn't. **MCMCglmm** fits overdispersion by default – so it's much better choice. A good alternative is ASReml, which is faster than **MCMCglmm** – but it's not free which for many people is limiting. And it also works on REML estimates which may be problematic in case of "weird" distributions.

Diagnostics of MCMCglmm

MCMCglmm works using randomization so utmost care should be taken to ensure that this random sampling actually samples joint posterior distribution of parameters. Specifically, you have to check if consecutive samples from the posterior are independent from each other. At the beginning they may not be independent as the walk through the posterior starts from some values, but then the chain should converge and samples should be independent.

At first, let's generate some "artificial" problems by shortening the MCMC chain in the previous model (on blue tits). We achieve this by setting the number of iterations to some low value (**nitt=2000**). Default **burnin=3000**, so we should lower this value below **2000**. We'll sample every second iteration (**thin=2**).

```
> ###code block 12
>
> prior <- list(R=list(V=1,nu=0.002),
+ G=list(G1=list(V=1,nu=0.002),
+ G2=list(V=1,nu=0.002)))
> m3.bad <- MCMCglmm(tarsus~sex, random=~fosternest+dam,
+ prior=prior,verbose=F, data=BTdata, nitt=2000,
+ burnin=500, thin=2)
> plot(m3.bad$VCV)
> autocorr(m3.bad$VCV)
, , fosternest

          fosternest         dam       units
Lag 0    1.00000000 -0.24455001 -0.10430602
Lag 2    0.82815231 -0.23892563 -0.10144780
Lag 10   0.47711854 -0.09162047 -0.09937022
Lag 20   0.23792851 -0.08204370 -0.09793121
```

```
Lag 100 -0.02210706  0.04458598 -0.04048683

, , dam

           fosternest          dam       units
Lag 0   -0.2445500147  1.000000000 -0.10084968
Lag 2   -0.2245963939  0.359331075 -0.01494919
Lag 10  -0.1340634723 -0.019236027  0.01143639
Lag 20  -0.1250328753 -0.009754384  0.02827148
Lag 100  0.0009055187 -0.073661071  0.01977242

, , units

          fosternest         dam       units
Lag 0   -0.104306016 -0.10084968  1.00000000
Lag 2   -0.094951195 -0.09790104  0.07213249
Lag 10  -0.036437767 -0.04785507 -0.01063517
Lag 20  -0.031778578  0.01160051 -0.03949663
Lag 100 -0.009335712  0.03043995 -0.03858646

>
> m3.good <- MCMCglmm(tarsus~sex, random=~fosternest+dam,
+ prior=prior,verbose=F, data=BTdata,
+ nitt=50000, burnin=3000, thin=50)
> plot(m3.good$VCV)
> autocorr(m3.good$VCV)
, , fosternest

            fosternest          dam       units
Lag 0     1.0000000000 -0.225723765 -0.18379490
Lag 50    0.0007770752 -0.027699347  0.02414860
Lag 250   0.0223765238  0.019155326 -0.01651623
Lag 500  -0.0197449053  0.002413883 -0.02576015
Lag 2500  0.0043333577  0.019968891  0.04245713

, , dam

          fosternest         dam        units
Lag 0    -0.22572376  1.00000000 -0.015647709
Lag 50   -0.06615831  0.01547784  0.029970618
Lag 250  -0.06405542  0.03430238 -0.001347705
Lag 500   0.03822892 -0.02795335 -0.015139277
Lag 2500 -0.02030468  0.07640567  0.014159344

, , units

           fosternest          dam       units
Lag 0    -0.183794898 -0.015647709  1.00000000
Lag 50   -0.011400580  0.002101844 -0.02310039
Lag 250  -0.005789198 -0.060701762  0.04574427
Lag 500  -0.026688031 -0.023276145  0.02380457
Lag 2500 -0.009362384  0.012835732  0.02213512
```

```
> #try below if you don't want to have huge
> #complex matrix outputs
> diag(autocorr(m3.good$VCV)[2,,])
   fosternest           dam         units
 0.0007770752  0.0154778375 -0.0231003891
```

The first model mixes poorly, and clear trends in time series suggest non-independence of samples drawn from posterior. Additionally, **`autocorr`** indicates substantial autocorrelation in random effects of **`dam`** and **`fosternest`** (in **`units`** it's smaller). After extending the chain problems disappear. Chains are in the form of flat time series, and autocorrelations are well below 0.05.

Finally, there's one more aspect of MCMC diagnostics: we should not only ensure independence of consecutive samples but also make sure that all effects are sampled good enough, i.e. samples we based our estimation on are large enough.

```
> ###code block 12a
>
> effectiveSize(m3.good$VCV)
fosternest        dam      units
  940.0000   816.6073   940.0000
```

Non-Gaussian data and overdispersion

Roughly speaking, overdispersion happens in case of data from distributions like Poisson or binomial. In such distributions variance is a function of mean. If, for some reason, variation in our data exceeds this expected when using the mean estimate, it said that such data are overdispersed. Using some simple simulated data we'll show how it arises. We'll simulate Poisson-distributed data and analyse it twice. First – we'll use all predictors that are used in simulating the data (thus, we'll have complete information regarding variation in our data). Secondly, we'll intentionally omit one predictor, introducing to our data some extra variation that is not explained and that gets superimposed on the variation resulting from Poisson process.

```
> ###code block 13
>
> x <- runif(1000, 0, 1) #uniform random variable
> z <- rnorm(1000, 0, sqrt(1.5)) #normal random variable
> l <- 0.5 + 1*x + 2*z #desired linear predictor
> yp <- rpois(1000, exp(l)) #added Poisson residuals to response
> glmdata <- data.frame(y = yp, x = x, z = z)
> pois1<-glm(y~x+z,data=glmdata,family="poisson")
> summary(pois1) #coeficients are on the right scale

Call:
glm(formula = y ~ x + z, family = "poisson", data = glmdata)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
```

```
-3.0553  -0.7429  -0.2667   0.4608   2.8070

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) 0.478843   0.017342   27.61   <2e-16 ***
x           1.008212   0.017995   56.03   <2e-16 ***
z           2.003880   0.004654  430.58   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 224940.41  on 999  degrees of freedom
Residual deviance:    940.13  on 997  degrees of freedom
AIC: 3837.9

Number of Fisher Scoring iterations: 4

> #because we exp the linear pred
>
> pois2<-glm(y~x,data=glmdata,family="poisson")
> summary(pois2)

Call:
glm(formula = y ~ x, family = "poisson", data = glmdata)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-10.526   -8.860   -7.956   -5.327  268.267

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  3.48965    0.01041  335.07   <2e-16 ***
x            0.52518    0.01664   31.56   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 224940  on 999  degrees of freedom
Residual deviance: 223935  on 998  degrees of freedom
AIC: 226831

Number of Fisher Scoring iterations: 8

> pois2$deviance/pois2$df.residual
[1] 224.3836
>
> pois21<-glm(y~x,data=glmdata,family="quasipoisson")
> summary(pois21)

Call:
```

```
glm(formula = y ~ x, family = "quasipoisson", data = glmdata)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-10.526   -8.860   -7.956   -5.327  268.267

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   3.4897     0.4951   7.049 3.36e-12 ***
x             0.5252     0.7911   0.664    0.507
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for quasipoisson family taken to be 2259.535)

    Null deviance: 224940  on 999  degrees of freedom
Residual deviance: 223935  on 998  degrees of freedom
AIC: NA

Number of Fisher Scoring iterations: 8

>
> pois3<-MCMCglmm(y~x,data=glmdata,family="poisson",verbose=F)
> summary(pois3)

 Iterations = 12991
 Thinning interval  = 3001
 Sample size  = 1000

 DIC: 4650.252

 R-structure:  ~units

      post.mean l-95% CI u-95% CI eff.samp
units       5.4    4.735    6.049    550.9

 Location effects: y ~ x

            post.mean l-95% CI u-95% CI eff.samp  pMCMC
(Intercept)    0.4885   0.1709   0.8170     1000  0.008 **
x              1.0527   0.4874   1.5617     1000 <0.001 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Omitting one variable not only extremely biases estimates but also changes deviance to df ratio. In general, if the model fitted is correct, the asymptotic distribution of deviance should be proportional do a Chi-squared variable with n-p df (roughly speaking number of data minus number of predictors): $D \sim \chi^2(df=n-p)$. If $D > n-p > \mathrm{E}[\chi^2(df=n-p)]$ it may indicate overdispersion. To be more practical, in the presence of overdispersion the ratio of residual deviance to residual df will be greater than 1. It's value approximately tells us about the strength of overdispersion. In our case, this is extreme (the ration is >300). Even using quasipoisson distribution does not

change anything – estimates look the same. However, when fitting the same model in **MCMCglmm** – estimates are much better. They're still biased but much closer to their true values. It is because **MCMCglmm** uses additive model of overdispersion. What does it mean?

In its usual form linear model I defined like this: $\mathbf{y}=\mathbf{X}\boldsymbol{\beta}+\mathbf{e}$ where $\mathbf{e}$ is residual (unexplained variance in the response). Taking expectations gives: $E[\mathbf{y}]=\exp(\mathbf{X}\boldsymbol{\beta})$. Exponent indicates, that it's a Poisson process for which log is the link function. We may present this on the scale of the latent variable: $\mathbf{l}=\boldsymbol{\eta}$ which is equivalent to $\log(E[\mathbf{y}])=\mathbf{X}\boldsymbol{\beta}$. However, in the presence of overdispersion, there's additional variation on top of the predicted value and it gives: $E[\mathbf{y}]=\exp(\mathbf{X}\boldsymbol{\beta}+\mathbf{e}^*)$ or $\mathbf{l}=\boldsymbol{\eta}+\mathbf{e}^*$. Now it is not entirely true that $\mathbf{y}\sim Pois(\exp(\mathbf{l}))$ because there is this additional variation over the variability of Poisson process. We can actually see these additional "residuals" (quotation marks indicate that this residual shows deviation from the variance expected by the Poisson process for a given mean). We'll analyse data on traffic accidents in Sweden. Analysis was performed to see if speed limit has some effect on the number of accidents, and if there are any year-by-year and day-by-day trends.

```
> ###code block 14
>
> library(MASS)
> data(Traffic)
> Traffic$year<-as.factor(Traffic$year)
>
> m4.bad <- glm(y~limit+year+day,family="poisson",data=Traffic)
> summary(m4.bad)

Call:
glm(formula = y ~ limit + year + day, family = "poisson", data = Traffic)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-4.1774  -1.4067  -0.4040   0.9725   4.9920

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  3.0467406  0.0372985  81.685  < 2e-16 ***
limityes    -0.1749337  0.0355784  -4.917 8.79e-07 ***
year1962    -0.0605503  0.0334364  -1.811   0.0702 .
day          0.0024164  0.0005964   4.052 5.09e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 625.25  on 183  degrees of freedom
Residual deviance: 569.25  on 180  degrees of freedom
AIC: 1467.2

Number of Fisher Scoring iterations: 4

> m4.bad$deviance/m4.bad$df.residual
[1] 3.162493
```

```
> # >3 times greater variation than expected
```

We can extract information on these residuals – this would require recording the behaviour of the latent variable (logged expectation of the response in this case). We'll show how much this additional variation changes Poisson process.

```
> ###code block 15
>
> prior <- list(R=list(V=1,nu=0.002))
> m4.good <- MCMCglmm(y~limit+year+day,family="poisson",data=Traffic,
+ prior=prior,verbose=F,pl=T)
> lat92 <- m4.good$Liab[,92]
> eta92 <- m4.good$Sol[,"(Intercept)"]+m4.good$Sol[,"day"]*92
> resid92 <- lat92-eta92
> mean(resid92)
[1] -0.1240384
```

Additional code in this part makes it possible to actually visualize all possible realizations of this specific Poisson process with additional variation; each of these residuals however happens to be observed only on one specific day.

One last riddle about overdispersion: in the Poisson process variance cannot be uniquely estimated as it is equal to the mean. Why didn't we fix this variance in the prior specification to prevent it from being estimated?

More on random effects

The distinction between fixed and random effects is sometimes difficult and controversial (see the discussion about year effect in countless ecological papers). However, this controversy in Bayesian analysis largely vanishes since ALL effects are basically random, they just differ in the way we define their variances. For fixed effects variances are set as very large, yielding flat priors, whereas for variance components we shrink this variance to allow it's direct estimation (for in random effects it's variance we're interested in). Let's see how we can see this equivalence. First we'll fit simple fixed-effect model to our traffic data and obtain predictions for both years.

```
> ###code block 16
>
> X <- model.matrix(y~limit+year+day,data=Traffic)
> X[c(1,2,3,91,92,183,184),]
    (Intercept) limityes year1962 day
1             1        0        0   1
2             1        0        0   2
3             1        0        0   3
91            1        0        0  91
92            1        0        0  92
183           1        1        1  91
184           1        1        1  92
```

```
> m5.fix <-
MCMCglmm(y~limit+year+day,data=Traffic,verbose=F,family="poisson")
> # using default prior
> #prediction for 1961 and 1962 with no speed limit
> y61.m5.fix <- m5.fix$Sol[,"(Intercept)"]
> y62.m5.fix <- m5.fix$Sol[,"(Intercept)"]+m5.fix$Sol[,"year1962"]
> posterior.mode(y61.m5.fix)
    var1
2.974852
> posterior.mode(y62.m5.fix)
    var1
2.915276
```

Now we redefine model so that year is treated as random effect BUT is associated with large variance, so basically it's the same as fixed effect. Note different method for obtaining predictions as in random effects intercept is suppressed by default.

```
> ###code block 17
>
> Z <- model.matrix(~year-1,data=Traffic)
> Z[c(1,2,3,91,92,183,184),]
    year1961 year1962
1          1        0
2          1        0
3          1        0
91         1        0
92         1        0
183        0        1
184        0        1
> X2 <- model.matrix(y~limit+day,data=Traffic)
> X2[c(1,2,3,91,92,183,184),]
    (Intercept) limityes day
1             1        0   1
2             1        0   2
3             1        0   3
91            1        0  91
92            1        0  92
183           1        1  91
184           1        1  92
> W<-cbind(X2,Z)
> W[c(1,2,3,91,92,183,184),]
    (Intercept) limityes day year1961 year1962
1             1        0   1        1        0
2             1        0   2        1        0
3             1        0   3        1        0
91            1        0  91        1        0
92            1        0  92        1        0
183           1        1  91        0        1
184           1        1  92        0        1
>
> prior <- list(R=list(V=1,nu=0.002),
+ G=list(G1=list(V=1e+08,fix=1)))
```

```
> m5.ran<-MCMCglmm(y~limit+day,random=~year,family="poisson",
+ data=Traffic,verbose=F,prior=prior,pr=T)
> #pr save the posterior of random effects
> y61.m5.ran <- m5.ran$Sol[,"(Intercept)"]+
+ m5.ran$Sol[,"year.1961"]
> y62.m5.ran <- m5.ran$Sol[,"(Intercept)"]+
+ m5.ran$Sol[,"year.1962"]
>
> #comparing posteriors for year effects from fixed and random effects
>
> y.fix <- mcmc(cbind(y1961=y61.m5.fix,y1962=y62.m5.fix))
> y.ran <- mcmc(cbind(y1961=y61.m5.ran,y1962=y62.m5.ran))
> plot(mcmc.list(y.fix,y.ran)) #virtually the same!
>
> #unfortunately as we have just two levels of year
> #treating this as random confounds year effects with intercept
>
> plot(c(m5.ran$Sol[,"year.1962"]+
+ m5.ran$Sol[,"year.1961"])/2,m5.ran$Sol[,"(Intercept)"])
```

And what if we made a more sensible decision and treated day as random effect? We'll leave day as continuous predictor to capture any trends associated with day, but also we'll put categorical variable day as random effect, to account for between day variability. Recall that we've earlier observed this variability as overdispersed residuals in the Poisson process. Accounting for variability in days almost entirely removes overdispersion and shrinks residual variance to close to zero.

```
> ###code block 18
>
> Traffic$day<-as.factor(Traffic$day)
> prior <- list(R=list(V=1,nu=0.002),
+ G=list(G1=list(V=1,nu=0.002)))
> m6 <- MCMCglmm(y~limit+year+as.numeric(day),random=~day,
+ family="poisson",data=Traffic,prior=prior,verbose=F)
> summary(m6)

 Iterations = 12991
 Thinning interval  = 3001
 Sample size  = 1000

 DIC: 1166.191

 G-structure:  ~day

    post.mean l-95% CI u-95% CI eff.samp
day   0.09221  0.06065   0.1296    170.4

 R-structure:  ~units

      post.mean  l-95% CI u-95% CI eff.samp
units  0.006757 0.0002729  0.01838    49.26
```

```
 Location effects: y ~ limit + year + as.numeric(day)

                 post.mean   l-95% CI   u-95% CI eff.samp
(Intercept)      3.0116915  2.8451222  3.1551507    342.8
limityes        -0.2495462 -0.3345628 -0.1533956    145.4
year1962        -0.0377975 -0.1201447  0.0389475    193.7
as.numeric(day)  0.0024443 -0.0002819  0.0050961    268.4
                 pMCMC
(Intercept)     <0.001 ***
limityes        <0.001 ***
year1962         0.330
as.numeric(day)  0.096 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> plot(m6$VCV)
> autocorr(m6$VCV)
, , day

                day       units
Lag 0    1.00000000 -0.23114258
Lag 10   0.30761083 -0.20437420
Lag 50   0.12384555 -0.12673639
Lag 100  0.06815593 -0.11596727
Lag 500 -0.02644770  0.04207223

, , units

                day       units
Lag 0   -0.23114258  1.00000000
Lag 10  -0.23203677  0.84447766
Lag 50  -0.17455563  0.53681546
Lag 100 -0.13089769  0.36073419
Lag 500  0.02705169 -0.09263029

> #we'll run the model for longer to treat
> #autocorrelation in residuals
>
> m6 <- MCMCglmm(y~limit+year+as.numeric(day),random=~day,
+ family="poisson",data=Traffic,prior=prior,verbose=F,
+ nitt=100000,burnin=20000,thin=50)
> plot(m6$VCV)
> #traces look better butperhaps improper
> #or expanded priors would be better
```

Binary/categorical data

Often in biology our data can be expressed as categories, ordered or without any numerical value (e.g. colours, sexes, success/failure data). In such cases we should use **`categorical`** family (or **`ordinal`** if our categories are ordered in any way), associated with link-functions logit or probit, respectively. Such data can be troubling and difficult to analyse.

We're in the best positions if we have binomial data, i.e. we have some units and within every unit we count some successes and some failures. Having such data makes possible to see if there's any heterogeneity in those units with respect to underlying probabilities associated with the binomial process. Here we'll generate simple binomial data which show such heterogeneity. Note that if in such data only intercept is fitted, it indicates heterogeneity as this intercept would be different than probabilities in every unit.

```
> ###code block 19
>
> ones <- rbinom(20, size=5, prob=c(0.2,0.8))
> zeros <- 5-ones
> bdata <- rbind(ones,zeros)
> bdata<-rbind(bdata,unit=letters[1:20])
> bdata<-as.data.frame(t(bdata))
> prior <- list(R=list(V=1,nu=0.002))
> m7.bin <- MCMCglmm(cbind(ones,zeros)~1,
+ data=bdata,family="multinomial2",
+ prior=prior,verbose=F,nitt=100000,
+ burnin=20000,thin=50)
> summary(m7.bin)

 Iterations = 99951
 Thinning interval  = 20001
 Sample size  = 1600

 DIC: 193.2368

 R-structure:  ~units

      post.mean  l-95% CI u-95% CI eff.samp
units    0.4696 0.0003961    1.537    705.2

 Location effects: cbind(ones, zeros) ~ 1

            post.mean l-95% CI u-95% CI eff.samp pMCMC
(Intercept)   -0.1216  -0.6218   0.3347      983 0.596
> install.packages("boot");library(boot)
> inv.logit(summary(m7.bin)$solutions[1]) #intercept is 0.5
[1] 0.4696424
> plot(m7.bin$VCV)
>
> data(PlodiaR)
> m8.bin <- MCMCglmm(cbind(Pupated,Infected)~1,
+ family="multinomial2",
+ data=PlodiaR, verbose=F)
> plot(m8.bin$VCV)
>
> #are Family effects really so variable?
> mode.mu <- posterior.mode(m8.bin$Sol)
> mode.V <- posterior.mode(m8.bin$VCV)
> ondatascale <- inv.logit(rnorm(10000, mean=mode.mu,
+ sd=sqrt(mode.V)))
```

```
> hist(ondatascale) #yes, it is!
```

Things become more complicated if we don't have such unit-grouping and every binary observation is repeated only once. Then we are not able to distinguish between equal probabilities in every unit or extreme asymmetry in some groups compared to others. Such scenarios would be indistinguishable and importantly every numerical inference would be biased by the choice of underlying residual (units) variance as it would be meaningless. We'll reanalyse Plodia data, but rewritten in the form of binary variables. As in such process residual variance cannot be estimated we'll fix it at some value and see what happens for different fixing values.

```
> ###code block 20
>
> data(PlodiaRB)
> prior1 <- list(R=list(V=1,fix=1),G=list(G1=list(V=1,nu=0.002)))
> prior2 <- list(R=list(V=2,fix=1),G=list(G1=list(V=1,nu=0.002)))
>
> m9.bin1 <- MCMCglmm(Pupated~1,random=~FSfamily,
+ family="categorical",
+ data=PlodiaRB,prior=prior1,verbose=F)
> m9.bin2 <- MCMCglmm(Pupated~1,random=~FSfamily,
+ family="categorical",
+ data=PlodiaRB,prior=prior2,verbose=F)
>
> plot(mcmc.list(m9.bin1$Sol,m9.bin2$Sol))
> plot(mcmc.list(m9.bin1$VCV,m9.bin2$VCV))#both posteriors differ!
```

Both intercept and family variance posteriors differ with regard to the residualvariance we've chosen. However it should not worry us. First of all – what matter the most here is not the absolute variation among families, but the degree to which two states (Pupated/Infected) are correlated within the same family. This information is contained in the coefficient of intraclass correlation, calculated like this: **IC** = *Var*(**FSfamily**)/(*Var*(**FSfamily**)+*Var*(**units**)+**c**), where the constant **c=pi^2/3** for logit link, and **c=1** for probit link. Youcan check that both IC's have the same posterior distribution:

```
> ###code block 21
>
> IC1 <- m9.bin1$VCV[,1]/(rowSums(m9.bin1$VCV)+pi^2/3)
> IC2 <- m9.bin2$VCV[,1]/(rowSums(m9.bin2$VCV)+pi^2/3)
> plot(mcmc.list(IC1,IC2))
```

As for intercept, we can use Hadfield's results (2010), due to Diggle et al. (2004), and rescale estimates so that they assumed some particular value of residual variance (*Var*(**units**)=**v**). Location effects (intercept, regression coefficients) can be rescaled by factor **sqrt((1+c^2*v)/(1+c^2****Var*(**units**))) and variance estimates may be rescaled by factor **(1+c^2*v)/(1+c^2****Var*(**units**)). The constant is **1** for probit and **16*sqrt(3)/15*pi** for logit. Let's try this for assumed residual variance of zero (**v=0**). Posteriors of Intercept are the same, up to Monte Carlo error.

```
> ###code block 22
>
> c <- 16*sqrt(3)/(15*pi)
> Int1 <- m9.bin1$Sol/sqrt(1+c^2*m9.bin1$VCV[,2])
> Int2 <- m9.bin2$Sol/sqrt(1+c^2*m9.bin2$VCV[,2])
> plot(mcmc.list(Int1,Int2)) #the same
```

Importantly, binary data can cause problems when there's large (near complete) separation, i.e. when most successes happened in one unit and most failures in other. This is because although on the link (logit) scale prior for the mean is flat (large variance), it's not flat at all on the data scale:

```
> ###code block 23
>
> hist(inv.logit(rnorm(1000,0,sqrt(1e+08))))
> #alternatively
> #hist(plogis(rnorm(1000,0,sqrt(1e+08))))
```

Let's simulate toy data with such huge separation and see how we can analyse them using usual **`glm()`** and **`MCMCglmm()`**. It's apparent, that only after changing the prior (and removing intercept) we can get some sensible results (Hadfield 2010).

```
> ###code bloc 24
>
> exper <- gl(2,25)
> y <- rbinom(50,1,c(0.5, 0.001)[exper])
> bdata2 <- data.frame(exp=exper,y=y)
> table(bdata2)
   y
exp  0  1
  1 14 11
  2 25  0
>
> m10.glm <- glm(y~exp,data=bdata2,family="binomial")
> summary(m10.glm)

Call:
glm(formula = y ~ exp, family = "binomial", data = bdata2)

Deviance Residuals:
       Min          1Q      Median          3Q         Max
-1.077e+00  -1.077e+00  -7.976e-05  -7.976e-05   1.281e+00

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)   -0.2412     0.4029  -0.599    0.549
exp2         -19.3249  2150.8026  -0.009    0.993

(Dispersion parameter for binomial family taken to be 1)
```

```
    Null deviance: 52.691  on 49  degrees of freedom
Residual deviance: 34.296  on 48  degrees of freedom
AIC: 38.296

Number of Fisher Scoring iterations: 18

>
> prior.def<-list(R=list(V=1,fix=1))
> m10.mc <- MCMCglmm(y~exp,data=bdata2,family="categorical",
+ prior=prior.def,verbose=F)
> summary(m10.mc)

 Iterations = 12991
 Thinning interval  = 3001
 Sample size  = 1000

 DIC: 36.75788

 R-structure:  ~units

      post.mean l-95% CI u-95% CI eff.samp
units         1        1        1        0

 Location effects: y ~ exp

            post.mean l-95% CI u-95% CI eff.samp  pMCMC
(Intercept)   -0.2432  -1.1484   0.7405  321.737  0.632
exp2         -10.9369 -17.2524  -2.4858    6.142 <0.001 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> plot(m10.mc$Sol)
>
> prior.better <- list(R=list(V=1,fix=1),
+ B=list(mu=c(0,0),V=diag(2)*(1+pi^2/3)))
> m10.mc2 <- MCMCglmm(y~exp,data=bdata2,family="categorical",
+ prior=prior.better, verbose=F)
> plot(m10.mc2$Sol)
> #looks much better but may need running for longer
>
> #checking if the results conform to simpler test - exact binomial
> m10.test <- binom.test(table(bdata2)[2,2],25)
> m10.test

        Exact binomial test

data:  table(bdata2)[2, 2] and 25
number of successes = 0, number of trials = 25, p-value
= 5.96e-08
alternative hypothesis: true probability of success is not equal to 0.5
95 percent confidence interval:
 0.0000000 0.1371852
sample estimates:
```

```
probability of success
                    0

>
> predict(m10.mc2,interval="confidence")[26,]
        fit         lwr         upr
0.045980115 0.003148058 0.116423314
Warning message:
In predict.MCMCglmm(m10.mc2, interval = "confidence") :
  predict.MCMCglmm is still developmental - be careful
```

Categorical random interactions

We've heard something on random interactions in **`lmer`**. Here we'll extend this concept in **`MCMCglmm`** as it gives much greater control on (co)variance structures.

We could repeat our analysis when looking for the interaction between sex and dam in our system (BTdata). Previously we used **`us()`** function, allowing for non-zero covariances. Now we'll repeat this analysis but fix these covariances at zero. It's simpler, mainly from the point of view of prior structure.

```
> ###code block 25
>
>
> prior.a <- list(R=list(V=1,nu=0.002),
+ G=list(G1=list(V=1,nu=0.002),
+ G2=list(V=diag(3),nu=0.002)))
> m11.bta <- MCMCglmm(tarsus~sex, random=~fosternest+idh(sex):dam,
+ prior=prior.a,verbose=F, data=BTdata)
> plot(m11.bta$VCV)
Waiting to confirm page change...
> #UNK has low dam variance which may be problematic
```

We can see the actual matrix of correlations in the dam effects and its representation in the $\mathbf{R}^3$ space.

```
> ###code block 26
>
> Vdam.a <- diag(colMeans(m11.bta$VCV)[2:4])
> colnames(Vdam.a) <- colnames(m11.bta$VCV)[2:4]
> Vdam.a
       Fem.dam  Male.dam    UNK.dam
[1,] 0.1765957 0.0000000 0.00000000
[2,] 0.0000000 0.1715039 0.00000000
[3,] 0.0000000 0.0000000 0.05000367
>
> plotsubspace(Vdam.a,axes.lab=T)
Loading required package: rgl
> #elipsoid depicting this covariance structure
```

The same can be done with an alternative definition of covariance structure using us(). We've used this function already but now we'll specify better prior. In general, priors for complex (co)variance structures depend on the particular structure.

```
> ###code block 27
>
> prior.b <- list(R=list(V=1,nu=0.002),
+ G=list(G1=list(V=1,nu=0.002),
+ G2=list(V=diag(3)*0.02,nu=4)))
> m11.btb <- MCMCglmm(tarsus~sex, random=~fosternest+us(sex):dam,
+ prior=prior.b,
+ verbose=F, data=BTdata)
Warning message:
In MCMCglmm(tarsus ~ sex, random = ~fosternest + us(sex):dam, prior =
prior.b,  :
  some combinations in us(sex):dam do not exist and 75 missing records have
been generated
> plot(m11.btb$VCV)
Waiting to confirm page change...
Waiting to confirm page change...
Waiting to confirm page change...
>
> Vdam.b <- matrix(colMeans(m11.btb$VCV)[2:10],3,3)
> colnames(Vdam.b) <- colnames(m11.btb$VCV)[2:4]
> Vdam.b
     Fem:Fem.dam Male:Fem.dam UNK:Fem.dam
[1,]   0.2319357    0.1994277   0.2237089
[2,]   0.1994277    0.2117362   0.2120803
[3,]   0.2237089    0.2120803   0.2889578
>
> plotsubspace(Vdam.b,axes.lab=T)
> #elipsoid depicting this covariance structure
> plot(posterior.cor(m11.btb$VCV[,2:10])[,c(3,4,8)])
> #all r roughly equal to 1
>
> "simpler model";m2.bt1$DIC
[1] "simpler model"
[1] 1992.66
> "us() variance structure";m11.btb$DIC
[1] "us() variance structure"
[1] 1997.765
> "idh() variance structure";m11.bta$DIC
[1] "idh() variance structure"
[1] 2037.151
```

As you can see both correlations are strong (almost 1) and variances are equal. Model with zero covariances is the worst, based on DIC values. Remaining two are similar but simpler one (equal variances and unity correlations) is better. In general, be careful when comparing models with different prior structures (as it was done here). DIC differences smaller than 2 should be treated with caution in such cases.

Priors for complex covariance structures

Complex variance structures have to take into account possible dependence of variances (which arises in case of non-zero covariances). For **idh()** variance structures it's simple: each variance in the structure is distributed independently, so new prior (**nu_** and **V_**, notation adopted from Hadfield (2010)) relates to a single-variance prior (**nu** and **V**) like this:

$\sigma_i^2 \sim IW($**nu_=nu**, **V_=V[1,1]**$)$

Hence prior specification in the example: **V=diag(3), nu=0.002**.

For **us()** structures it's more complicated:

$\sigma_i^2 \sim IW($**nu_=nu-dim(V)+1, V_=V[1,1]*nu/nu_**$)$

Consequently, we used **V=diag(3)*0.02 and nu=4**. We did use **nu=4** instead of usual **nu=4.002** and lower variance value to make this prior proper but also uninformative for correlation. We could alternatively use an improper prior, by setting **V=diag(dim(V))*0** and **nu=dim(V)-3**, but remember dangers of using improper priors.

Using inverse gamma distribution, with **shape=nu/2** and **scale=(nu*V)/2** we can actually visualize this prior for one of its elements:

```
> ###code block 28
>
> nu.star <- prior.b$G$G2$nu - dim(prior.b$G$G2$V)[1]*1
> V.star <- prior.b$G$G2$V[1,1]*(prior.b$G$G2$nu/nu.star)
> xv <- seq(1e-16,1,length=100)
> library(MCMCpack)
> dv<-dinvgamma(xv,shape=nu.star/2,scale=(nu.star*V.star)/2)
> detach(package:MCMCpack)
> plot(dv~xv,type="l")
```

Part 2

Overview

1. Presentation (optional) – animal model, meta-analysis, comparative model – different names for basically the same.
2. DIY – animal model
    a. Defining pedigrees
    b. Running simple animal model and interpreting it
3. DIY – comparative model
    a. Working with phylogenies
    b. Package **`ape`**
    c. Incorporating phylogenetic data in a mixed model
    d. Interpreting comparative mixed models
4. DIY – random regression
    a. Random interactions – not only categorical: interpretation
    b. Building random regression models – logic and strategy
5. Multivariate models and estimation of genetic correlations
    a. The problem – many response variables
    b. Estimating genetic correlation

Animal model – what do we need?

Animal model as presented is just a very special case of mixed model, where for one particular fixed effect identity matrix is replaced by relationship matrix **A**, implying dependence of random effect between different data units. The degree of this dependence is described by the off-diagonal values in this matrix. In other words, considering random effect Z_1, it can be written that $\mathbf{u} \sim N(\mathbf{0}, \sigma^2\mathbf{A})$. Just to remind, in "usual" mixed models the effect of the random factor is assumed to be uncorrelated across units, and hence identity matrix **I** in place of **A**.

In order to successfully fit animal model we have to have two things: data file with the response assigned to data units (individuals), where all individuals are uniquely identified, and pedigree file containing information about the parents of each individual. The more genetic connections we have, the better our estimates will be. In general, data file should contain phenotypic/response information for all individuals, bot offspring and parents. Of course, usually there will be some source, first generation for which parents are not known. Remember also to use all available genealogy information; if for some reason we don't have measurements for some parents, but we KNOW that certain children come from these parents, we may assign dummy parent identifiers to indicate shared ancestry of these children. We will work on two examples – one is the data on blue tits from **`MCMCglmm`** package, and second is data on imaginary creatures, gryphones, from Wilson et al. (2010).

First of all – make yourself familiar with the pedigree file.

```
> ###code block B1
>
```

```
> library(MCMCglmm)
> data(BTdata)
> data(BTped)
>
> #lets look at some chosen family
>
> parents <- function(x) {
+ any(x=="R187920"|x=="R187921")
+ }
>
> where<-apply(BTped,1,parents)
> fam <- BTped[which(where),]
> fam
      animal     dam    sire
66   R187920    <NA>    <NA>
172  R187921    <NA>    <NA>
325  R187726 R187920 R187921
411  R187724 R187920 R187921
503  R187723 R187920 R187921
838  R187613 R187920 R187921
932  R187612 R187920 R187921
1030 R187609 R187920 R187921
>
> #to see the monster - just calculate the relationships
> install.packages("kinship"); library(kinship)
>
> #multiply by 2 since relatedness is twice P
> #that an allele is ibd in 2 individuals
> A <- 2*kinship(fam[,1],fam[,2],fam[,3])
> A
        R187920 R187921 R187726 R187724 R187723 R187613 R187612
R187920     1.0     0.0     0.5     0.5     0.5     0.5     0.5
R187921     0.0     1.0     0.5     0.5     0.5     0.5     0.5
R187726     0.5     0.5     1.0     0.5     0.5     0.5     0.5
R187724     0.5     0.5     0.5     1.0     0.5     0.5     0.5
R187723     0.5     0.5     0.5     0.5     1.0     0.5     0.5
R187613     0.5     0.5     0.5     0.5     0.5     1.0     0.5
R187612     0.5     0.5     0.5     0.5     0.5     0.5     1.0
R187609     0.5     0.5     0.5     0.5     0.5     0.5     0.5
        R187609
R187920     0.5
R187921     0.5
R187726     0.5
R187724     0.5
R187723     0.5
R187613     0.5
R187612     0.5
R187609     1.0
```

Fitting simple animal model in MCMCglmm is simple. Having data and pedigree we can just proceed. Remember to use restricted variable animal to account for additive genetic effect. Here will fit simple animal model to estimate additive genetic variance and heritability of tarsus length in blue tits. We'll see if model works fine (diagnostics) and correct it if necessary.

```
> ###code block B2
>
> prior.b1 <- list(R=list(V=1,nu=0.002),G=list(G1=list(V=1,nu=0.002),
+ G2=list(V=1,nu=0.002),G3=list(V=1,nu=0.002)))
>
> m.b1 <- MCMCglmm(tarsus~sex,random=~animal+dam+fosternest,
+ data=BTdata,pedigree=BTped,verbose=F,prior=prior.b1)
Warning message:
In MCMCglmm(tarsus ~ sex, random = ~animal + dam + fosternest, data =
BTdata,  :
  some combinations in animal do not exist and 212 missing records have
been generated
> plot(m.b1$VCV)#needs attention - poor mixing
> # may take up to 5 min!
> m.b1 <- MCMCglmm(tarsus~sex,random=~animal+dam+fosternest,
+ data=BTdata,pedigree=BTped,verbose=F,prior=prior.b1,
+ nitt=100000,burnin=30000,thin=50)
Warning message:
In MCMCglmm(tarsus ~ sex, random = ~animal + dam + fosternest, data =
BTdata,  :
  some combinations in animal do not exist and 212 missing records have
been generated
> plot(m.b1$VCV)
>
> #perhaps simplification is a better choice
> #since we have only full-sibs in our data
> #genetic effects may be confounded with
> #the dam effect and hence this poor mixin
>
> m.b2 <- MCMCglmm(tarsus~sex,random=~animal+fosternest,
+ data=BTdata,pedigree=BTped,verbose=F,prior=prior.b1,
+ nitt=30000,burnin=10000,thin=30)
Warning message:
In MCMCglmm(tarsus ~ sex, random = ~animal + fosternest, data = BTdata,  :
  some combinations in animal do not exist and 212 missing records have
been generated
> plot(m.b2$VCV) #much better!
> plot(m.b2$Sol)
>
> diag(autocorr(m.b2$VCV)[2,,])
    animal fosternest      units
 0.5009898  0.0999186  0.4380569
> # correlations are a bit high so we might run model for longer
> # i'll leavy explaration to you to save our class time
> # i'd recommend running this model for at least 100000 iterations
>
> HPDinterval(m.b2$VCV[,1]) #CI for VA
```

```
         lower     upper
var1 0.2868201 0.6573943
attr(,"Probability")
[1] 0.9504505
> h2 <- m.b2$VCV[,"animal"]/rowSums(m.b2$VCV)
> posterior.mode(h2)
     var1
0.5364527
> HPDinterval(h2)
         lower     upper
var1 0.3523153 0.6978943
attr(,"Probability")
[1] 0.9504505
```

As you can see calculating heritabilities couldn't be simpler. We can verify also another results – in the model with just dam and fosternest effects we could estimate broad-sense heritability taking advantage of the fact that the variance associated with dams should approximate half of additive genetic variance (plus some maternal, dominance and epistasis effects if present). In fact, they are in close agreement – but not entirely. Why?

```
> ###code block B3
>
> # for broad sense heritability
> m.b2a <- MCMCglmm(tarsus~sex,random=~dam+fosternest,
+ data=BTdata,verbose=F,prior=prior.b1)
> plot(m.b2a)
Waiting to confirm page change...
> H2 <- 2*(m.b2a$VCV[,"dam"]/rowSums(m.b2a$VCV))
> posterior.mode(H2)
     var1
0.5050896
> HPDinterval(H2)
         lower     upper
var1 0.3569514 0.6906795
attr(,"Probability")
[1] 0.95
```

Defining proper models is essential for correct interpretation of estimated effects. Here we'll use simulated data on gryphons to follow patterns of variation between individuals. As you will see, results will depend on the inclusion of specific effects and on the way we partition variance. At the beginning try fitting simple animal model, starting with appropriate priors. Assume no fixed effects except the intercept. I'm suggesting using stronger priors that would improve mixing of our chain – we can afford it as our data are well structured and contain a lot of information on desired effects. I'm suggesting calculating overall variance in the trait (TARSUS) and partitioning it equally between all random terms, with belief parameter equal to one.

(no peeking ;))

```
> ###code block B4
>
> gryph<-read.table("Gryphon.txt",head=T,sep="\t")
```

```
> gryped<-read.table("gryphonped.txt",head=T,sep="\t")
>
>
> names(gryph)[1]<-"animal"
> gryph$animal <- as.factor(gryph$animal)
> gryph$MOTHER <- as.factor(gryph$MOTHER)
> gryph$BYEAR <- as.factor(gryph$BYEAR)
> gryph$SEX <- as.factor(gryph$SEX)
> gryph$BWT <- as.numeric(gryph$BWT)
> gryph$TARSUS <- as.numeric(gryph$TARSUS)
>
> #factorise all ids in pedigree
> for (i in 1:3) {gryped[,i]<-as.factor(gryped[,i])}
>
> m.b31 <- MCMCglmm(TARSUS~1, random=~animal, data=gryph,
+ pedigree=gryped, verbose=F)
Warning message:
In MCMCglmm(TARSUS ~ 1, random = ~animal, data = gryph, pedigree = gryped,
:
  some combinations in animal do not exist and 225 missing records have
been generated
> plot(m.b31$VCV)
>
> VTAR <- var(gryph$TARSUS,na.rm=T)
> prior.b31<- list(R=list(V=VTAR/2,nu=1),G=list(G1=list(V=VTAR/2,nu=1)))
> m.b31 <- MCMCglmm(TARSUS~1, random=~animal, data=gryph,
+ pedigree=gryped, verbose=F, nitt=50000, burnin=10000,
+ thin=50, prior=prior.b31)
Warning message:
In MCMCglmm(TARSUS ~ 1, random = ~animal, data = gryph, pedigree = gryped,
:
  some combinations in animal do not exist and 225 missing records have
been generated
> plot(m.b31$VCV) #much better although even longer=better
>
> #heritability
> h21 <- m.b31$VCV[,"animal"]/rowSums(m.b31$VCV)
> posterior.mode(h21)
     var1
0.4418477
> HPDinterval(h21)
         lower     upper
var1 0.2288822 0.6083062
attr(,"Probability")
[1] 0.95
```

Now let's see what's happening when we include additional effects?

```
> m.b32 <- MCMCglmm(TARSUS~SEX, random=~animal, data=gryph,
+ pedigree=gryped, verbose=F, nitt=50000, burnin=10000,
+ thin=50, prior=prior.b31)
Warning message:
```

```
In MCMCglmm(TARSUS ~ SEX, random = ~animal, data = gryph, pedigree =
gryped,  :
  some combinations in animal do not exist and 225 missing records have
been generated
> plot(m.b32$VCV) #much better although even longer=better
>
> #heritability
> h22 <- m.b32$VCV[,"animal"]/rowSums(m.b32$VCV)
> posterior.mode(h22)
     var1
0.3659256
> HPDinterval(h22)
         lower     upper
var1 0.2312707 0.5787381
attr(,"Probability")
[1] 0.95
> #sex took some VR and thus increased h2
>
> prior.b33<- list(R=list(V=VTAR/3,nu=1),G=list(G1=list(V=VTAR/3,nu=1),
+ G2=list(V=VTAR/3,nu=1)))
> m.b33 <- MCMCglmm(TARSUS~SEX, random=~animal+BYEAR, data=gryph,
+ pedigree=gryped, verbose=F, nitt=50000, burnin=10000,
+ thin=50, prior=prior.b33)
Warning message:
In MCMCglmm(TARSUS ~ SEX, random = ~animal + BYEAR, data = gryph,  :
  some combinations in animal do not exist and 225 missing records have
been generated
> plot(m.b33$VCV) #much better although even longer=better
>
> #heritability
> h23 <- m.b33$VCV[,"animal"]/rowSums(m.b33$VCV)
> posterior.mode(h23)
     var1
0.3444932
> HPDinterval(h23)
         lower     upper
var1 0.1714646 0.5258129
attr(,"Probability")
[1] 0.95
> posterior.mode(m.b32$VCV[,"units"])
    var1
18.20563
> posterior.mode(m.b33$VCV[,"units"]+m.b33$VCV[,"BYEAR"])
    var1
19.91133
>
>
> prior.b34<- list(R=list(V=VTAR/4,nu=1),G=list(G1=list(V=VTAR/4,nu=1),
+ G2=list(V=VTAR/4,nu=1),G3=list(V=VTAR/4,nu=1)))
> m.b34 <- MCMCglmm(TARSUS~SEX, random=~animal+BYEAR+MOTHER, data=gryph,
+ pedigree=gryped, verbose=F, nitt=50000, burnin=10000,
+ thin=50, prior=prior.b34)
```

```
Warning message:
In MCMCglmm(TARSUS ~ SEX, random = ~animal + BYEAR + MOTHER, data = gryph,
:
  some combinations in animal do not exist and 225 missing records have
been generated
> plot(m.b34$VCV)
>
> #heritability
> h24 <- m.b34$VCV[,"animal"]/rowSums(m.b34$VCV)
> posterior.mode(h24)
     var1
0.3241248
> HPDinterval(h24)
         lower    upper
var1 0.1142319 0.454118
attr(,"Probability")
[1] 0.95
> #here heritability of TARSUS dropped dramatically as we used
> #another random effect that accounts for similarity between sibs
```

As expected, playing with fixed and random effects changes our conclusions. Is this bad? Not really. We just have to realize how we interpret results with respect to fixed effects (we conditions our estimates on them) and random effects (they help partition the variance more adequately). Another important use of mixed models is estimation of repeatability. Here we'll show how to achieve this and how to account for this extra variability to correct additive genetic effects for the presence of permanent environmental effects.

```
> ###code block B5
>
> gryrm<-read.table("gryphonRM.txt",head=T,sep="\t")
> names(gryrm)[1]<-"animal"
> gryrm$animal<-as.factor(gryrm$animal)
> gryrm$BYEAR<-as.factor(gryrm$BYEAR)
> gryrm$AGE<-as.factor(gryrm$AGE)
> gryrm$YEAR<-as.factor(gryrm$YEAR)
> gryrm$LAYDATE<-as.numeric(gryrm$LAYDATE)
> gryrm$ID<-gryrm$animal
>
> VLAY <- var(gryrm$LAYDATE, na.rm=T)
> prior.b41 <- list(R=list(V=VLAY/2,nu=1),G=list(G1=list(V=VLAY/2,nu=1)))
> m.b41 <- MCMCglmm(LAYDATE~1,random=~ID,verbose=F,data=gryrm,
+ prior=prior.b41)
> plot(m.b41$VCV)
> #repeatability
> rep1 <- m.b41$VCV[,"ID"]/rowSums(m.b41$VCV)
> posterior.mode(rep1)
     var1
0.3446529
> HPDinterval(rep1)
         lower     upper
var1 0.2879485 0.3939303
attr(,"Probability")
```

```
[1] 0.95
>
> #lets account from age differences
> m.b42 <- MCMCglmm(LAYDATE~AGE,random=~ID,verbose=F,data=gryrm,
+ prior=prior.b41)
> plot(m.b42$VCV)
> #repeatability
> rep2 <- m.b42$VCV[,"ID"]/rowSums(m.b42$VCV)
> posterior.mode(rep2)
     var1
0.4259473
> HPDinterval(rep2)
         lower     upper
var1 0.3728316 0.4801913
attr(,"Probability")
[1] 0.95
>
> #naive heritability
> m.b43 <- MCMCglmm(LAYDATE~AGE,random=~animal,verbose=F,data=gryrm,
+ prior=prior.b41,pedigree=gryped)
Warning message:
In MCMCglmm(LAYDATE ~ AGE, random = ~animal, verbose = F, data = gryrm,  :
  some combinations in animal do not exist and 840 missing records have
been generated
> plot(m.b43$VCV)
> #heritability
> h25 <- m.b43$VCV[,"animal"]/rowSums(m.b43$VCV)
> posterior.mode(h25)
     var1
0.4313042
> HPDinterval(h25)
        lower     upper
var1 0.385356 0.5117057
attr(,"Probability")
[1] 0.95
>
> #heritability + permanet env effects
> prior.b41 <- list(R=list(V=VLAY/3,nu=1),G=list(G1=list(V=VLAY/3,nu=1),
+ G2=list(V=VLAY/3,nu=1)))
> m.b44 <- MCMCglmm(LAYDATE~AGE,random=~animal+ID,verbose=F,data=gryrm,
+ prior=prior.b41,pedigree=gryped)
Warning message:
In MCMCglmm(LAYDATE ~ AGE, random = ~animal + ID, verbose = F, data =
gryrm,  :
  some combinations in animal do not exist and 840 missing records have
been generated
> plot(m.b44$VCV)#be careful - running for longer required
> #heritability and repeatability
> h26 <- m.b44$VCV[,"animal"]/rowSums(m.b44$VCV)
> rep2 <- m.b44$VCV[,"ID"]/rowSums(m.b44$VCV)
> posterior.mode(h26)
     var1
```

```
0.1853370
> HPDinterval(h26)
          lower     upper
var1 0.08038286 0.2952880
attr(,"Probability")
[1] 0.95
> posterior.mode(rep2)
     var1
0.2580697
> HPDinterval(rep2)
        lower     upper
var1 0.148573 0.3530787
attr(,"Probability")
[1] 0.95
```

Phylogenies – a short guide

Phylogenies in some way are similar to pedigrees. They also represent relationships, however not between individuals, but between species or higher taxa. We may use them in **MCMCglmm** in exactly the same way as we did with pedigrees and thus build phylogenetic comparative models, accounting for variability that might have arisen from evolutionary history rather than genuine ecological/individual-based processes. First we'll learn how to build and handle phylogenies in R. We'll use the package **ape** and as its output objects can be directly handled by **MCMCglmm**.

We'll work with the mammals species phylogeny based on mammals super-tree and provided in Adams (2007).

```
> ###code block B6
>
> mammals <- read.nexus("mammals.nex")
> mammals

Phylogenetic tree with 40 tips and 35 internal nodes.

Tip labels:
        Rattus_rattus,        Sigmodon_hispidus,        Peromyscus_eremicus,
Peromyscus_maniculatus, Neotoma_cinerea, Microtus_pennsylvanicus, ...

Rooted; includes branch lengths.
> summary(mammals)

Phylogenetic tree: mammals

  Number of tips: 40
  Number of nodes: 35
  Branch lengths:
    mean: 19.89730
    variance: 641.1923
    distribution summary:
   Min. 1st Qu.  Median 3rd Qu.    Max.
   0.10    3.10    9.10   26.52   94.50
```

```
  No root edge.
  First ten tip labels: Rattus_rattus
                        Sigmodon_hispidus
                        Peromyscus_eremicus
                        Peromyscus_maniculatus
                        Neotoma_cinerea
                        Microtus_pennsylvanicus
                        Microtus_montebelli
                        Chaetodipus_penicillatus
                        Dipodomys_ordii
                        Dipodomys_compactus
  No node labels.
> mammals.plot<-plot(mammals,font=1,cex=0.75)
> nodelabels()
> mammals$edge
      [,1] [,2]
 [1,]   41   42
 [2,]   42   43
 [3,]   43   44
 [4,]   44   45
 [5,]   45   46
 [6,]   46    1
 [7,]   46   47
 [8,]   47    2
 [9,]   47   48
[10,]   48   49
[11,]   49    3
[12,]   49    4
[13,]   48    5
[14,]   46   50
[15,]   50    6
[16,]   50    7
[17,]   45   51
[18,]   51   52
[19,]   52    8
[20,]   52   53
[21,]   53   54
[22,]   54   55
[23,]   55    9
[24,]   55   10
[25,]   54   56
[26,]   56   57
[27,]   57   58
[28,]   58   11
[29,]   58   12
[30,]   57   59
[31,]   59   60
[32,]   60   13
[33,]   60   14
[34,]   59   15
[35,]   57   16
[36,]   57   17
```

```
[37,]   56   61
[38,]   61   18
[39,]   61   19
[40,]   53   62
[41,]   62   20
[42,]   62   63
[43,]   63   21
[44,]   63   22
[45,]   51   64
[46,]   64   23
[47,]   64   24
[48,]   44   65
[49,]   65   25
[50,]   65   26
[51,]   43   27
[52,]   42   28
[53,]   41   66
[54,]   66   67
[55,]   67   68
[56,]   68   29
[57,]   68   69
[58,]   69   70
[59,]   70   71
[60,]   71   72
[61,]   72   30
[62,]   72   31
[63,]   72   32
[64,]   71   33
[65,]   70   34
[66,]   69   35
[67,]   67   73
[68,]   73   36
[69,]   73   74
[70,]   74   37
[71,]   74   38
[72,]   66   75
[73,]   75   39
[74,]   75   40
> #if you want you can write the tree in newick format
> write.tree(mammals,file="mammals.nck")
```

If you don't have the tree and just have information to build one (e.g. DNA sequences) you can do this in **ape**. You can choose among different methods of clustering and different models of evolution.

```
> ###code block B7
>
> data(woodmouse)
> woodmouse
15 DNA sequences in binary format stored in a matrix.

All sequences of same length: 965
```

```
Labels: No305 No304 No306 No0906S No0908S No0909S ...

Base composition:
    a     c     g     t
0.307 0.261 0.126 0.306
> base.freq(woodmouse)
        a         c         g         t
0.3065414 0.2613083 0.1260264 0.3061239
> write.dna(woodmouse,"woodmouse.fas",format="fasta")
> rodents <- read.dna("woodmouse.fas",format="fasta")
>
> rodents[1,] #first sequence
1 DNA sequences in binary format stored in a matrix.

All sequences of same length: 965

Labels: No305

Base composition:
    a     c     g     t
0.304 0.262 0.129 0.306
> as.character(rodents[1,1:50])
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
[,14] [,15] [,16] [,17]
No305 "n"  "t"  "t"  "c"  "g"  "a"  "a"  "a"  "a"  "a"   "c"   "a"   "c"
"a"   "c"   "c"   "c"
      [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25] [,26] [,27] [,28]
[,29] [,30] [,31] [,32] [,33]
No305 "a"   "c"   "t"   "a"   "c"   "t"   "a"   "a"   "a"   "a"   "n"   "t"
"t"   "a"   "t"   "c"
      [,34] [,35] [,36] [,37] [,38] [,39] [,40] [,41] [,42] [,43] [,44]
[,45] [,46] [,47] [,48] [,49]
No305 "a"   "g"   "t"   "c"   "a"   "c"   "t"   "c"   "c"   "t"   "t"   "c"
"a"   "t"   "c"   "g"
      [,50]
No305 "a"
> paste(as.character(rodents[1,1:50]),collapse="")
[1] "nttcgaaaaacacacccactactaaaanttatcagtcactccttcatcga"
>
> #calculate phylogeny based on these sequences
> dist.dna(rodents[1:5,])
              No305       No304       No306     No0906S
No304   0.015975800
No306   0.013815969 0.004210551
No0906S 0.019213434 0.013802125 0.009514854
No0908S 0.017059224 0.011665428 0.007391898 0.012726856
> rodents.dist<-dist.dna(rodents)
> as.matrix(dist.dna(rodents[1:5,])) # looks much better
             No305       No304       No306     No0906S     No0908S
No305   0.00000000 0.015975800 0.013815969 0.019213434 0.017059224
No304   0.01597580 0.000000000 0.004210551 0.013802125 0.011665428
```

```
No306   0.01381597 0.004210551 0.000000000 0.009514854 0.007391898
No0906S 0.01921343 0.013802125 0.009514854 0.000000000 0.012726856
No0908S 0.01705922 0.011665428 0.007391898 0.012726856 0.000000000
>
> #build a tree using UPGMA
> cluster<-hclust(rodents.dist)
> rodents.upgma<-as.phylo(cluster)
> plot(rodents.upgma,cex=0.75,font=1,no.margin=T)
>
> #we can use neighbour joining instead
> cluster.nj<-nj(rodents.dist)
> rodents.nj<-as.phylo(cluster.nj)
> plot(rodents.nj,cex=0.75,font=1,no.margin=T)
```

Having several trees it's good to be able to compare them.

```
> ###codeblock B8
>
> #compare trees
> all.equal(rodents.nj,rodents.bionj)
[1] FALSE
> #and ignoring branch lengths - i.e. comparring only topologies
> all.equal(rodents.nj,rodents.bionj,use.edge.length=F)
[1] TRUE
>
> #having a lot of trees you can calculate distances between them
> dist.topo(rtree(30),rtree(30))
[1] 54
```

However, real comparison of trees employs testing, either using bootstrapping or likelihood methods. Here we'll bootstrap our NJ tree. Fro those who are interested – see package **phangorn** which offers much more advanced functions for bootstrapping and ML-ing trees.

```
> ###code block B9
>
>                                rodents.boot                                <-
boot.phylo(rodents.nj,rodents,function(x){nj(dist.dna(x))},
+ B=200,block=1)
> rodents.boot/2
 [1] 100.0  22.0  53.5  51.5  56.0  42.0  67.5  65.5  87.5  90.0  87.0
99.5  59.0
> plot(rodents.nj)
> nodelabels(rodents.boot/2)
```

Some of you may be used to using more traditional approaches in fitting phylogenetic models. Falsenstein (1985) proposed method based on phylogenetically independent contrasts, i.e. leading to such transformation of the original data that measurements become statistically independent (removes any relationships resulting from shared ancestry) and identically distributed. In **ape** there's on function for doing this – and we'll try this with some simulated toy data on rodents; we'll use additional data set on rodents since contrasts require the tree to be rooted. On the other hand, if you'd like to use packages other than **MCMCglmm** for comparative

analyses, you might want to obtain for a given phylogeny distance matrix to pass it to another modelling function.

```
> ###code block B10
>
> rodents2 <- read.tree("rodents.tre")
> summary(rodents2)

Phylogenetic tree: rodents2

  Number of tips: 14
  Number of nodes: 13
  Branch lengths:
    mean: 0.02021977
    variance: 0.0002354209
    distribution summary:
    Min.  1st Qu.   Median  3rd Qu.     Max.
0.002857 0.008928 0.015000 0.030710 0.071430
  No root edge.
  First ten tip labels: Apodemus_alpicola
                        Apodemus_uralensis
                        Apodemus_flavicollis
                        Apodemus_sylvaticus
                        Apodemus_hermonensis
                        Apodemus_mystacinus
                        Apodemus_peninsulae
                        Apodemus_semotus
                        Apodemus_agrarius
                        Tokudaia_minutus
  No node labels.
> mouseData <- rnorm(14,mean=10,sd=sqrt(5))
> mouseCont <- pic(mouseData,rodents2)
> mouseCont #remember to supress intercept using PICs
         15          16          17          18          19          20
21          22
  2.4808244   8.7577759  10.8454153  -0.2523556   6.7139111 -34.9264549  -
0.6190993   9.9784157
         23          24          25          26          27
-36.0750266   8.6993401   0.7376448  19.3082225  23.8880437
>
> vcv.phylo(rodents2)[1:3,1:3]
                  Apodemus_alpicola Apodemus_uralensis Apodemus_flavicollis
Apodemus_alpicola           0.089999         0.078570             0.075713
Apodemus_uralensis          0.078570         0.092856             0.075713
Apodemus_flavicollis        0.075713         0.075713             0.088570
```

What else can be done in R with phylogenies? You can extract relevant information from phylogenies. Also, for fans of art, different plotting modes are available.

```
> ###code block B11
>
> #sample data on bird families
```

```
> data(bird.families)
> bird.families$tip.label[1:20]
 [1] "Struthionidae"  "Rheidae"        "Casuariidae"    "Apterygidae"
"Tinamidae"
 [6] "Cracidae"       "Megapodiidae"   "Phasianidae"    "Numididae"
"Odontophoridae"
[11] "Anhimidae"      "Anseranatidae"  "Dendrocygnidae" "Anatidae"
"Turnicidae"
[16] "Indicatoridae"  "Picidae"        "Megalaimidae"   "Lybiidae"
"Ramphastidae"
> bird.families<-makeNodeLabel(bird.families)#naming all ancestral nodes
> #let's choose 4 families and extract frm the phylogeny
> bird.fam.4 <- drop.tip(bird.families, setdiff(bird.families$tip.label,
+ some.families))
Error in as.vector(y) : object 'some.families' not found
> vcv.phylo(bird.fam.4,cor=T)
Error in vcv.phylo(bird.fam.4, cor = T) : object 'bird.fam.4' not found
> plot(bird.fam.4)
Error in plot(bird.fam.4) : object 'bird.fam.4' not found
> plot(bird.families,font=1,cex=0.4)
> some.families<-c("Certhiidae","Paridae","Gruidae","Struthionidae")
>
> bird.fam.30 <- drop.tip(bird.families, setdiff(bird.families$tip.label,
+ bird.families$tip.label[1:20]))
> plot(bird.fam.30)
> plot(bird.fam.30,type="cladogram")
> plot(bird.fam.30,type="fan")
> plot(bird.fam.30,type="unrooted")
> plot(bird.fam.30,type="radial")
```

Comparative analysis – simple case

We'll use sample data on carnivorous mammals from ape. As there's no phylogeny there, we have to build one by ourselves. I found phylogeny of carnivore families in the Internet and transformed it to Newick format. You can load it directly to ape. Note that there's no information on branch lengths, just the topology. After loading we'll replace the names with the correct family names. They must correspond to the names of taxa in the dataset.

```
> ###code block B12
>
> data(carnivora)
> carphyl <- read.tree("carnphyl.tre")
> plot(carphyl)
> carphyl$tip.label
[1] "Fel"    "Vive"   "Hyen"   "Cani"   "Must"   "Procyo" "Ursi"   "Ailur"
> levels(carnivora$Family)
Error in levels(carnivora$Family) : object 'carnivora' not found
> carphyl$tip.label<-c("Felidae","Viverridae","Hyaenidae",
+ "Canidae","Mustelidae","Procyonidae",
+ "Ursidae","Ailuridae")
> plot(carphyl)
```

Let's try simple model, relating female brain size of mammals to litter size and age of independence.

```
> ###code block B13
>
> summary(carnivora)
       Order         SuperFamily         animal        Genus
 Carnivora:112   Caniformia:57   Viverridae :32   Mustela : 9
                 Feliformia:55   Mustelidae :30   Herpetes: 8
                                 Felidae    :19   Panthera: 5
                                 Canidae    :18   Canis   : 4
                                 Hyaenidae  : 4   Martes  : 4
                                 Procyonidae: 4   Felis   : 3
                                 (Other)    : 5   (Other) :79
                    Species          FW                SW
 Acinonyx jubatus       :  1   Min.   :  0.050   Min.   :  0.050
 Ailuropoda melanoleuca :  1   1st Qu.:  1.245   1st Qu.:  1.400
 Alopex lagopus         :  1   Median :  3.400   Median :  3.895
 Aonyx capensis         :  1   Mean   : 18.099   Mean   : 20.084
 Arctictis binturong    :  1   3rd Qu.: 10.363   3rd Qu.: 11.592
 Arctogalidia trivirgata:  1   Max.   :320.000   Max.   :365.000
 (Other)                :106
       FB               SB               LS
 Min.   :  1.00   Min.   :  1.00   Min.   :1.000
 1st Qu.: 15.25   1st Qu.: 15.68   1st Qu.:2.500
 Median : 33.00   Median : 33.75   Median :3.000
 Mean   : 53.40   Mean   : 56.43   Mean   :3.232
 3rd Qu.: 57.38   3rd Qu.: 57.17   3rd Qu.:3.800
 Max.   :365.00   Max.   :459.50   Max.   :8.800
                                   NA's   :2.000
       GL               BW                WA
 Min.   : 23.50   Min.   :   0.01   Min.   : 21.0
 1st Qu.: 53.80   1st Qu.:  41.88   1st Qu.: 54.5
 Median : 63.00   Median : 116.25   Median : 70.0
 Mean   : 65.79   Mean   : 249.31   Mean   :104.0
 3rd Qu.: 73.50   3rd Qu.: 286.88   3rd Qu.:117.0
 Max.   :168.00   Max.   :1650.00   Max.   :730.0
 NA's   : 21.00   NA's   :  50.00   NA's   : 49.0
       AI               LY             AM           IB
 Min.   :  56.0   Min.   : 96          :57          :55
 1st Qu.: 183.8   1st Qu.:141   365    : 8   12     :27
 Median : 365.0   Median :162   730    : 5   6      :13
 Mean   : 407.8   Mean   :182   913    : 4   24     : 2
 3rd Qu.: 592.5   3rd Qu.:207   450    : 2   27     : 2
 Max.   :1080.0   Max.   :408   1095   : 1   4      : 2
 NA's   :  82.0   NA's   : 63   (Other):35   (Other):11
>
> m.b51 <-
MCMCglmm(FW~LS+GL+as.numeric(AM),data=na.omit(carnivora),verbose=F)
> plot(m.b51$Sol)
> plot(m.b51$VCV)
```

```
> summary(m.b51)

 Iterations = 12991
 Thinning interval  = 3001
 Sample size  = 1000

 DIC: 166.9515

 R-structure:  ~units

      post.mean l-95% CI u-95% CI eff.samp
units      7340     2524    15855      810

 Location effects: FW ~ LS + GL + as.numeric(AM)

              post.mean  l-95% CI  u-95% CI eff.samp pMCMC
(Intercept)     108.5660 -112.0664  329.0216     1000 0.292
LS              -12.7827  -34.1070   12.9590     1402 0.264
GL                0.5361   -1.4051    2.7961     1000 0.568
as.numeric(AM)   -2.0536   -5.8273    1.1087     1000 0.188
>
> names(carnivora)[3]<-"animal"
> prior.b52 <- list(R=list(V=1,nu=0.002),G=list(G1=list(V=1,nu=0.002)))
> m.b52 <- MCMCglmm(FW~LS+GL+as.numeric(AM),data=na.omit(carnivora),
+ verbose=F,random=~animal,prior=prior.b52,pedigree=carphyl)
Warning messages:
1: In inverseA(pedigree, nodes = nodes, scale = scale) :
  no branch lengths: compute.brlen from ape has been used
2: In MCMCglmm(FW ~ LS + GL + as.numeric(AM), data = na.omit(carnivora),  :
  some combinations in animal do not exist and 9 missing records have been
generated
> plot(m.b52$Sol)
> plot(m.b52$VCV)
> summary(m.b52)

 Iterations = 12991
 Thinning interval  = 3001
 Sample size  = 1000

 DIC: 166.5641

 G-structure:  ~animal

       post.mean  l-95% CI u-95% CI eff.samp
animal      3420 0.0003233    13340    215.9

 R-structure:  ~units

      post.mean l-95% CI u-95% CI eff.samp
units      6940     1888    14724     1000

 Location effects: FW ~ LS + GL + as.numeric(AM)
```

```
               post.mean  l-95% CI  u-95% CI eff.samp pMCMC
(Intercept)      96.2407 -129.8394  314.2464     1000 0.354
LS              -12.1530  -38.5514   13.1992     1000 0.284
GL                0.6461   -1.3016    2.9291     1000 0.508
as.numeric(AM)   -1.8809   -5.2790    2.0228     1000 0.272
>
> #let's try for simulated data ob bird families
>
> bf.sim <-
rTraitCont(bird.families,sigma=runif(Nedge(bird.families),0.1,0.7))
> bf.sim <- data.frame(y=bf.sim,animal=names(bf.sim))
> bf.sim[1:20,]
                          y         animal
Struthionidae  -0.60573476  Struthionidae
Rheidae        -2.68284592        Rheidae
Casuariidae     2.46897329    Casuariidae
Apterygidae     1.09753708    Apterygidae
Tinamidae       0.24638486      Tinamidae
Cracidae       -0.67455764       Cracidae
Megapodiidae    1.74500202   Megapodiidae
Phasianidae     0.61040649    Phasianidae
Numididae       1.65685699      Numididae
Odontophoridae  0.77349921 Odontophoridae
Anhimidae      -2.55968724      Anhimidae
Anseranatidae   1.37302713  Anseranatidae
Dendrocygnidae  0.05206279 Dendrocygnidae
Anatidae       -2.81114176       Anatidae
Turnicidae      2.64692783     Turnicidae
Indicatoridae  -0.16953127  Indicatoridae
Picidae        -0.91794680        Picidae
Megalaimidae    0.82388019   Megalaimidae
Lybiidae        1.38448623       Lybiidae
Ramphastidae    0.19388505   Ramphastidae
> #we'll add some residuals on top
> err <- rnorm(137,sd=sqrt(3))
> bf.sim[,1]<-bf.sim[,1]+err
> #and replication
> bf.sim2 <- as.data.frame(bf.sim[sample(1:137,50),])
> err2 <- rnorm(50,runif(20,1,2),sqrt(3))
> bf.sim2[,1]<-bf.sim2[,1]+err2
> bf.sim3 <- as.data.frame(bf.sim[sample(1:137,50,replace=T),])
> err3 <- rnorm(50,runif(20,1.5,2.5),sqrt(3))
> bf.sim3[,1]<-bf.sim3[,1]+err3
> #and combine the three
> bf.sim<-rbind(bf.sim,bf.sim2,bf.sim3)
> summary(bf.sim)
       y                       animal
 Min.   :-6.6159   Casuariidae    :  4
 1st Qu.:-1.9046   Laridae        :  4
 Median : 0.2256   Pedionomidae   :  4
 Mean   : 0.2603   Acanthisittidae:  3
```

```
 3rd Qu.: 2.1467   Climacteridae  :  3
 Max.   : 9.9202   Cracidae       :  3
                   (Other)        :216
>
> #prior is the same
>
> m.b53 <- MCMCglmm(y~1,random=~animal,pedigree=bird.families,
+ data=bf.sim,verbose=F,prior=prior.b52)
Warning message:
In MCMCglmm(y ~ 1, random = ~animal, pedigree = bird.families, data =
bf.sim,  :
  some combinations in animal do not exist and 134 missing records have
been generated
> plot(m.b53$VCV)
> posterior.mode(m.b53$Sol)
(Intercept)
  0.7599772
>
> #not accounting for phylogenetic depndence yields false picture
> m.b53a <- MCMCglmm(y~1,data=bf.sim,verbose=F)
> posterior.mode(m.b53a$Sol)
(Intercept)
  0.2946834
>
> #is in ordinary animal model we can estimate so called phylogenetic
heritability
> #the proportion of total variance explained by phylogenetic effects of
shared ancestry
>
> hp2 <- m.b53$VCV[,"animal"]/rowSums(m.b53$VCV)
> posterior.mode(hp2)
     var1
0.7400869
> HPDinterval(hp2)
         lower     upper
var1 0.6475381 0.8305757
attr(,"Probability")
[1] 0.95
```

As you can see, not accounting for phylogeny may yield false picture of the reality underlying measured traits. We could expand this model and analyse more than one trait and see if they evolve in a correlated fashion across the phylogeny – which would be equivalent of calculating ordinary genetic correlation in an animal model framework.

Random regression – expanding random interactions

In the previous chapter we learned how to fit categorical random interactions. It's equal to allowing for differences in the intercept of our model across the levels of random term, crossed with the chose fixed effect. However, sometimes it is sensible to add also differences in slopes among individuals/units. Such models are called random regression models. We will use

strategy from Hadfield (2010) for the longitudinal data on chicken growth. First, let's analyse it with a simple model. Fitting random effect of chick id means that we want to have separate intercepts for each chick. As the data are not linear will stick to some polynomial approximations of curvilinearity.

```
> ###code block B14
>
> data(ChickWeight)
> xyplot(weight~Time|Chick,data=ChickWeight)
>
> prior.b61 <- list(R=list(V=1e-16,nu=-2),G=list(G1=list(V=1,nu=1)))
> m.b61 <- MCMCglmm(weight~Diet+poly(Time,2,raw=T),random=~Chick,
+ data=ChickWeight, verbose=F, pr=T, prior=prior.b61,
+ saveX=T, saveZ=T)
> pop.int <- posterior.mode(m.b61$Sol[,1])
> pop.slope <- posterior.mode(m.b61$Sol[,5])
> pop.quad <- posterior.mode(m.b61$Sol[,6])
> chick.int <- posterior.mode(m.b61$Sol[,c(7:56)])
>
> time <- ChickWeight$Time[1:12]
> plot(pop.int+pop.slope*I(time^1)+pop.quad*I(time^2)~time,
+ type="l",lwd=2,ylim=c(-50,400))
> for(i in 1:50) {
+ lines(pop.int+chick.int[i]+pop.slope*I(time^1)+
+ pop.quad*I(time^2)~time,lty=3,col="red")
+ }
>
> #we can print predictions from our model for each chick by combining
> #multiplying desing matrix W=[X,Z] for effects with
> #parameter vector theta=[beta,u]
> W1 <- cBind(m.b61$X,m.b61$Z)
> theta <- posterior.mode(m.b61$Sol)
> prediction1 <- W1 %*% theta
> xyplot(weight+prediction1[,1]~Time|Chick,data=ChickWeight)
```

As expected, model fits well, predictions look reasonable. However, slight differences are visible between predicted and real curves for some chicks. Thus, we might as well allow for differences in slopes. In the simpler model in random effects we fitted just single variance, i.e. σ^2**(Intercept).** Interacting random term with both intercept and slope yields 2x2 covariance structure:

$$\mathbf{V}_{\text{Chick}} = \begin{bmatrix} \sigma^2_{\textbf{(Intercept)}} & \sigma_{\textbf{(Intercept),Time}} \\ \sigma_{\textbf{(Intercept),Time}} & \sigma^2_{\textbf{Time}} \end{bmatrix}$$

```
> ###code block B15
>
> prior.b62 <- list(R=list(V=1e-16,nu=-2),G=list(G1=list(V=diag(2),nu=2)))
> m.b62 <-
MCMCglmm(weight~Diet+poly(Time,2,raw=T),random=~us(1+Time):Chick,
+ data=ChickWeight, verbose=F, pr=T,prior=prior.b62,saveX=T,saveZ=T)
> plot(m.b62$VCV)
```

```
Waiting to confirm page change...
> diag(autocorr(m.b62$VCV)[2,,])
(Intercept):(Intercept).Chick        Time:(Intercept).Chick
                0.0919710564                  0.0309195190
       (Intercept):Time.Chick               Time:Time.Chick
                0.0309195190                -0.0003973471
                       units
               -0.0392904703
> r.int.slope <- m.b62$VCV[,2]/sqrt(m.b62$VCV[,1]*m.b62$VCV[,4])
> posterior.mode(r.int.slope)
     var1
-0.9778924
> #cor close to space boundary - should be run for longer
>
> #could do predictions by hand or like here by using predict()
> xyplot(weight+predict(m.b62,marginal=NULL)~Time|Chick,data=ChickWeight)
Warning message:
In predict.MCMCglmm(m.b62, marginal = NULL) :
  predict.MCMCglmm is still developmental - be careful
> #looks MUCH better - may could be better
> #adding second random slope for quadratic term?
> prior.b63 <- list(R=list(V=1e-16,nu=-2),G=list(G1=list(V=diag(3),nu=3)))
> m.b63 <- MCMCglmm(weight~Diet+poly(Time,2,raw=T),
+ random=~us(1+poly(Time,2,raw=T)):Chick,
+ data=ChickWeight, verbose=F, pr=T,
+ prior=prior.b63,saveX=T,saveZ=T)
> xyplot(weight+predict(m.b63,marginal=NULL)~Time|Chick,data=ChickWeight)
Warning message:
In predict.MCMCglmm(m.b63, marginal = NULL) :
  predict.MCMCglmm is still developmental - be careful
>
> #DICs confirm its the best model - hence chicks differ both in
> #intercepts and (quadratic)slopes
> m.b61$DIC;m.b62$DIC;m.b63$DIC
[1] 5525.291
[1] 4544.456
[1] 3932.947
>
> #to confirm we could see if REML estimators corroborate these conclusions
> library(lme4)

Attaching package: 'lme4'

The following object(s) are masked from 'package:coda':

    HPDinterval

The following object(s) are masked from 'package:stats':

    AIC
```

```
> m.b61reml <-
lmer(weight~Diet+poly(Time,2,raw=T)+(1|Chick),data=ChickWeight)
> summary(m.b61reml)@AICtab[1]
      AIC
 5578.963
>
> m.b62reml <- lmer(weight~Diet+poly(Time,2,raw=T)+(1+Time|Chick),
+ data=ChickWeight)
> summary(m.b62reml)@AICtab[1]
      AIC
 4732.387
>
> m.b63reml <-
lmer(weight~Diet+poly(Time,2,raw=T)+(1+poly(Time,2,raw=T)|Chick),
+ data=ChickWeight)
> summary(m.b63reml)@AICtab[1]
      AIC
 4267.013
> detach(package:lme4)
```

Unfortunately, in pursue for the best model we forgot about one thing. In case of random slope models we should check not only if model is the best-fitting one, but also how well it's variance structure describes variance in the real data. Particularly, having intercept + *n* slopes fitted as random we expect that variance should change as the function of *n*-th degree with the continuous predictor. We'll how it works for toy data and then inspect our models. In general, from linear modelling theory, variance in the response should follow something like this: *Var*[*y*]=diag(**ZVZ'**) where **Z** is the design matrix for random effects and **V** is estimated covariance matrix. We can calculate this directly, having saved design matrices in our models (**`saveZ=T`**). However, here we'll create our own **Z** to avoid problems caused by duplication of records (we had several Diets and several Time points for every Chicken). We create hypothetical design matrix as if there was one chicken measured over 100 time points.

```
> ###code block B16
>
> toyslope <- rnorm(30)#30 random slopes ~N(0,1)
> #prepare space for the plots
> plot(0,type="n",xlim=c(-1,1),ylim=c(-3,3),ylab="y",xlab="time")
> for (i in 1:30) { #for each of 30 slopes
+ abline(a=0,b=toyslope[i])
+ }
>
> time<-seq(0,21,length=100)
> polynomial<-leg(time,2,normalized=F)
Loading required package: polynom
> #better than poly because generates first column of ones giving
> #appropriate design matrix for fixed and random slope effects
>
> #coeficient for fixed effects
> beta1 <- c(posterior.mode(m.b61$Sol[,1]),posterior.mode(m.b61$Sol[,5]),
+ posterior.mode(m.b61$Sol[,6]))
```

```
> beta2 <- c(posterior.mode(m.b62$Sol[,1]),posterior.mode(m.b62$Sol[,5]),
+ posterior.mode(m.b62$Sol[,6]))
> beta3 <- c(posterior.mode(m.b63$Sol[,1]),posterior.mode(m.b63$Sol[,5]),
+ posterior.mode(m.b63$Sol[,6]))
>
> #covariance matrices and residuals
>
> VCV1 <- matrix(posterior.mode(m.b61$VCV)[1],1,1)#single variance
> VCV2 <- matrix(posterior.mode(m.b62$VCV)[1:(2^2)],2,2)#4 parameters
> VCV3 <- matrix(posterior.mode(m.b63$VCV)[1:(3^2)],3,3)#9 parameters
> units1 <- posterior.mode(m.b61$VCV)[2]
> units2 <- posterior.mode(m.b62$VCV)[5]#5th parameter cause 4 for
(co)variances
> units3 <- posterior.mode(m.b63$VCV)[10]#10th cause 9 pars for
(co)variances
>
> plot(weight~Time,data=ChickWeight,cex.lab=1.5)
> mu1 <- polynomial %*% beta1 #population line across time
> sd1 <- sqrt(units1+diag(polynomial[,1,drop=F]%*%
+ VCV1%*%t(polynomial[,1,drop=F])))
> #%*% multiplies matrices; drop lets matrix be a matrix after extracting
one
> #dimension, otherwise it would be a vector and would cause problems when
> #trying to multiply to get ZVZ'; by using first column of polynomial we
> #create 'new' Z matrix appropriate for the time sequence we have, of
length
> #100 rather than 12
> lines(mu1~time,lwd=2)
> lines(I(mu1+1.96*sd1)~time,lty=2,lwd=1,col="red")
> lines(I(mu1-1.96*sd1)~time,lty=2,lwd=1,col="red")
>
> plot(weight~Time,data=ChickWeight,cex.lab=1.5)
> mu2 <- polynomial %*% beta2 #population line across time
> sd2 <- sqrt(units2+diag(polynomial[,1:2,drop=F]%*%VCV2%*%
+ t(polynomial[,1:2,drop=F])))
> lines(mu2~time,lwd=2)
> lines(I(mu2+1.96*sd2)~time,lty=2,lwd=1,col="red")
> lines(I(mu2-1.96*sd2)~time,lty=2,lwd=1,col="red")
> #very good fit of variance change to data
>
> plot(weight~Time,data=ChickWeight,cex.lab=1.5,ylim=c(-150,600))
> mu3 <- polynomial %*% beta3 #population line across time
> sd3 <- sqrt(units2+diag(polynomial[,1:3,drop=F]%*%VCV3%*%
+ t(polynomial[,1:3,drop=F])))
> lines(mu3~time,lwd=2)
> lines(I(mu3+1.96*sd3)~time,lty=2,lwd=1,col="red")
> lines(I(mu3-1.96*sd3)~time,lty=2,lwd=1,col="red")
> #very poor  fit of variance change to data, 2nd model seems the best!
```

Multiple responses – lessons from money spending

I'm many time you've asked yourself if an univariate model is appropriate. Nature by definition is multivariate, and not only that. Complex patterns of correlations exist between these variables. Many techniques exist to deal with such complex data, of which MANOVA should be one of better known. Problem with multivariate statistics is that they're often misused and abused. It always looks fancier and more trendy when you include complex multivariate models but beware: they're not always applicable. Here we'll learn several applications of multivariate models where it's essential to simultaneously model several variables.

First example shows that plain methods such as simple regression may not discover patterns in our data. It comes from Hadfield's notes on **MCMCglmm** and serves as good introduction to multivariate models. We'll use simulated data on spending money on car and holidays and ask if simple regression of one against the other is enough in describing underlying relationships. After that we'll develop simple bivariate model and learn its characteristic parameters. In general, **MCMCglmm** gives us control over almost every aspect of such model.

```
> ###code block 17
>
> id <- gl(200,4) #200 people, 4 records for each
> id[1:50]
 [1] 1  1  1  1  2  2  2  2  3  3  3  3  4  4  4  4  5  5  5  5  6
[22] 6  6  6  7  7  7  7  8  8  8  8  9  9  9  9  10 10 10 10 11 11
[43] 11 11 12 12 12 12 13 13
200 Levels: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 ... 200
> av.wealth<-rlnorm(200,0,1)
> ac.wealth<-av.wealth[id]+rlnorm(800,0,1)
> av.ratio<-rbeta(200,10,10)#beta is good distribution for ratios
> ac.ratio<-rbeta(800,2*(av.ratio[id]),2*(av.ratio[id]))
> y.car<-(ac.wealth*ac.ratio)^0.25
> y.hol<-(ac.wealth*(1-ac.ratio))^0.25
> spend<-data.frame(y.hol=y.hol,y.car=y.car,id=id)
>
> #simple regression - are these two related?
> summary(lm(y.car~y.hol,data=spend))

Call:
lm(formula = y.car ~ y.hol, data = spend)

Residuals:
     Min       1Q   Median       3Q      Max
-0.89103 -0.18154 -0.01299  0.18072  1.16946

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.00768    0.03726   27.05   <2e-16 ***
y.hol        0.01957    0.03492    0.56    0.575
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2922 on 798 degrees of freedom
Multiple R-squared: 0.0003934,  Adjusted R-squared: -0.0008593
F-statistic: 0.314 on 1 and 798 DF,  p-value: 0.5754
```

```
>
> #accounting for random variation doesn't change the point
> summary(m.b7a <- MCMCglmm(y.car~y.hol,random=~id,data=spend,verbose=F))

 Iterations = 12991
 Thinning interval  = 3001
 Sample size  = 1000

 DIC: 218.2472

 G-structure:  ~id

   post.mean l-95% CI u-95% CI eff.samp
id   0.02024  0.01252   0.0289      852

 R-structure:  ~units

      post.mean l-95% CI u-95% CI eff.samp
units   0.06681  0.05946  0.07436     1000

 Location effects: y.car ~ y.hol

            post.mean l-95% CI u-95% CI eff.samp  pMCMC
(Intercept)   1.13230  1.05227  1.21743     1000 <0.001 ***
y.hol        -0.10219 -0.18636 -0.03304     1000  0.012 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
>
> #is this model valid? univariate model assumes that there is causal link
> #between spending money on holiday and spending it on car
> #however we've model the data so that not only spending on car affects
> #spending on holiday - or vice versa - but also there's some year-to-year
> #variation in the income, affecting the spendings
> #adopting regression ignores variation in x and hence in
> #(unmeasured) income; we'll model these variabilities fitting
> #both spendings as two separate traits
>
> prior.b7<-list(R=list(V=diag(2),nu=2),G=list(G1=list(V=diag(2),nu=2)))
> m.b7 <- MCMCglmm(cbind(y.hol,y.car)~trait-1, random=~us(trait):id,
+ rcov=~us(trait):units,data=spend,family=c("gaussian","gaussian"),
+ verbose=F,prior=prior.b7)
> cor(matrix(colMeans(m.b7$VCV),2,2))
     [,1] [,2]
[1,]    1   -1
[2,]   -1    1
> #let's look at regression coefficients describing different parts of
variability
> #in this data: a = Cov/Var
>
> id.r <- m.b7$VCV[,2]/m.b7$VCV[,1]
> res.r <- m.b7$VCV[,6]/m.b7$VCV[,5]
```

```
>
plot(mcmc.list(m.b7a$Sol[,"y.hol"],id.r,res.r),density=F,col=c("red","green
","blue"))
> #trends are opposite depending on the part of variation we look on
> #note that if the data were generated so that relationship between
spending on holidays
> #and car was purely causal
> spend$y.hol2<-rnorm(200,0,sqrt(2))[spend$id]+rnorm(800,0,sqrt(1))
> spend$y.car2<-spend$y.hol2*-
0.3+rnorm(200,0,sqrt(1))[spend$id]+rnorm(800,0,sqrt(2))
>
> m.b7ab <- MCMCglmm(y.car2~y.hol2,random=~id,data=spend,verbose=F)
> m.b7b <- MCMCglmm(cbind(y.hol2,y.car2)~trait-1, random=~us(trait):id,
+ rcov=~us(trait):units,data=spend,family=c("gaussian","gaussian"),
+ verbose=F,prior=prior.b7)
> id.r2 <- m.b7b$VCV[,2]/m.b7b$VCV[,1]
> res.r2 <- m.b7b$VCV[,6]/m.b7b$VCV[,5]
>
plot(mcmc.list(m.b7ab$Sol[,"y.hol2"],id.r2,res.r2),density=F,col=c("red","g
reen","blue"))
> #slopes are wirtually the same up to Monte Carlo error
```

OK, but what's the point? When we should adopt such multiple-response modelling? In general every time when causality is not obvious. Moreover, every time we want to model covariance of several (2 or more) variables with respect to some sources of variation – we should always employ multivariate model. Below we will build on already known animal model and try to look for correlations arising between two phenotypic traits at the genetic level. It's important to realize that such models can be generalized to every random effect in action. Here we'll work with additive genetic effect, which differs only in the use of additional data, the pedigree. However, we could model correlations arising in every random effect – we just have to put these effects into appropriate variance functions.

```
> ###code block B18
>
> data(BTdata)
> data(BTped)
> #2x2 prior since we model 2 traits - tarsus and back color
> prior.b81 <- list(R=list(V=diag(2),nu=1.002),
+ G=list(G1=list(V=diag(2),nu=1.002),
+ G2=list(V=1,nu=0.002)))
> m.b81 <- MCMCglmm(cbind(tarsus,back)~trait+sex:trait-1,
+ random=~us(trait):animal+fosternest,
+ rcov=~us(trait):units,verbose=F,data=BTdata,
+ pedigree=BTped,nitt=70000,burnin=15000,thin=70,
+ family=c("gaussian","gaussian"),prior=prior.b81)
Warning message:
In MCMCglmm(cbind(tarsus, back) ~ trait + sex:trait - 1, random =
~us(trait):animal +  :
  some combinations in us(trait):animal do not exist and 212 missing
records have been generated
```

```
> plot(m.b81$VCV)#traces show autocorrelation and probably nitt around
200000 would be required
Waiting to confirm page change...
Waiting to confirm page change...
> #we'll go on with these results as running longer model would require >15
min!
> #both back colour and tarsus seem to be genetically controlled
> #but what about genetic correlation?
> #rG=Cov/sqrt(Var1*Var2)
> rG<-m.b81$VCV[,2]/sqrt(m.b81$VCV[,1]*m.b81$VCV[,4])
> posterior.mode(rG)
      var1
-0.4546824
> HPDinterval(rG)#rG is not zero truncated so can be tested using CIs
          lower       upper
var1 -0.6567884 -0.03121054
attr(,"Probability")
[1] 0.9503185
```

Model clearly states that there's no correlation on genetic level in our system. However, we could treat as two traits not only different characters. Recently it's very popular to look at intersexual genetic correlations. It's very straightforward to fit them but be careful. As each individual is measured only for one trait (you can be either male or female, never both) residual covariance cannot be estimated and hence we use **`idh()`** variance function.

```
> ###code block B19
>
> prior.b82 <-
list(R=list(V=diag(3),nu=2.002),G=list(G1=list(V=diag(3),nu=2.002),
+ G2=list(V=1,nu=0.002)))
> m.b82 <- MCMCglmm(tarsus~sex,
+ random=~us(sex):animal+fosternest,
+ rcov=~idh(sex):units,verbose=F,data=BTdata,
+ nitt=50000,burnin=12000,thin=50,prior=prior.b82,pedigree=BTped)
Warning message:
In MCMCglmm(tarsus ~ sex, random = ~us(sex):animal + fosternest,  :
  some combinations in us(sex):animal do not exist and 2292 missing records
have been generated
> posterior.mode(m.b82$VCV)
  Fem:Fem.animal  Male:Fem.animal   UNK:Fem.animal
       0.47857343       0.31716790       0.27689074
 Fem:Male.animal Male:Male.animal  UNK:Male.animal
       0.31716790       0.47316007       0.32215996
  Fem:UNK.animal  Male:UNK.animal   UNK:UNK.animal
       0.27689074       0.32215996       0.46095586
       fosternest        Fem.units       Male.units
       0.06331766       0.27364180       0.36449503
        UNK.units
       0.22388796
> rMF<-m.b82$VCV[,2]/sqrt(m.b82$VCV[,1]*m.b82$VCV[,5])
> HPDinterval(rMF)
```

```
         lower     upper
var1 0.4642555 0.8228062
attr(,"Probability")
[1] 0.95
> posterior.mode(rMF)
     var1
0.6943073
> #rG is not zero truncated so can be tested using CIs
```

As an exercise see if there's any evidence for genetic covariance between tarsus length and birth weight in gryphons.

Part 3

Overview

1. DIY - Meta-analysis
    a. How to define?
    b. Interpretation
    c. Comparative phylogenetic meta-analysis
2. DIY – advanced issues in **`MCMCglmm`**
    a. Parameter expanded priors and their usefulness
    b. Zero-inflated and Hurdle models
    c. Zero-altered models
    d. (Time permitting) Recursive models – using non-standard (co)variance structures
3. DIY – how to analyse survival data in R?

Meta-analysis

Today we'll extend what we've learned yesterday and fit meta-analysis. Meta-analytical approach became very popular recently as it allows for answering very general questions. In its essence meta-analysis is very simple – instead of analysis raw data you take already calculated trends/statistics and look at their variability. In general, meta-analysis asks if predicted values of statistics holds after accounting for many studies, or if predicted relationship exists at the level of many studies. In such a case you assume that any error (residual variation) in our data is due to error in estimating statistics. In other words we can insert this error as some *a priori* known "residuals". Note, that sometimes meta-analyst is able to get accurate "raw" data from publications. In this case we use ordinary GLMM with response and estimated residual variance (one of the best examples is Cornwallis et al. (2010)).

Here we'll use example from Adams (2007). He examined if there are any body size clines in mammals, i.e. if mammals are larger in larger latitudes, where the climate is cooler. He gathered data on different mammal taxa from many papers, and for each paper he calculated effect size as the correlation between mammal body size and latitude. Following his paper and general strategy of meta-analysis we'll estimate measurement error (sampling variance of the statistic) based on the number of geographic locations from which data were available in each study. At first we'll try simple meta-analysis, ignoring any phylogenetic dependence of examined taxa.

```
> ###code block C1
>
> library(MCMCglmm)
Loading required package: tensorA

Attaching package: 'tensorA'

The following object(s) are masked from 'package:base':
```

```
    norm

Loading required package: Matrix
Loading required package: lattice

Attaching package: 'Matrix'

The following object(s) are masked from 'package:base':

    det

Loading required package: coda
Loading required package: ape
Loading required package: corpcor
Warning message:
package 'tensorA' was built under R version 2.12.0
> clines <- read.csv("mamm_clines.csv",head=T)
> clines<-clines[,-4]
> clines$FisherZ<-0.5*log((1+clines$corr)/(1-clines$corr))
> #effect size
> clines$mev<-1/(clines$N-3)#measurement error
> clines$weight<-1/clines$mev
> #weights if analysing in ordinary lm/lmer/glm/lme
>
> prior.c1 <- list(R=list(V=1,nu=0.002))
>
> m.c11 <- MCMCglmm(FisherZ~1, verbose=F, prior=prior.c1,
+ data=clines)
> summary(m.c11)

 Iterations = 12991
 Thinning interval  = 3001
 Sample size  = 1000

 DIC: 97.02173

 R-structure:  ~units

      post.mean l-95% CI u-95% CI eff.samp
units     0.649   0.3998   0.9898     1000

 Location effects: FisherZ ~ 1

            post.mean l-95% CI u-95% CI eff.samp pMCMC
(Intercept)   0.29285  0.06895  0.56036     1000 0.016 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
>
> #we add sampling error of statistics mev
> m.c12 <- MCMCglmm(FisherZ~1, verbose=F, prior=prior.c1,
+ data=clines,
```

```
+ mev=clines$mev)
Loading required package: polynom
> summary(m.c12)

 Iterations = 12991
 Thinning interval  = 3001
 Sample size  = 1000

 DIC: 88.56011

 R-structure:  ~units

                                                post.mean
leg(mev, -1, FALSE):leg(mev, -1, FALSE).meta         1
                                                l-95% CI
leg(mev, -1, FALSE):leg(mev, -1, FALSE).meta         1
                                                u-95% CI
leg(mev, -1, FALSE):leg(mev, -1, FALSE).meta         1
                                                eff.samp
leg(mev, -1, FALSE):leg(mev, -1, FALSE).meta         0

 Location effects: FisherZ ~ 1

            post.mean  l-95% CI  u-95% CI eff.samp pMCMC
(Intercept)  0.231696 -0.002097  0.488288     1000  0.06 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> plot(m.c12)
Waiting to confirm page change...
>
> plot(corr~N,data=clines,type="p",pch=20,ylab="Correlation",
+ xlab="N locations")
> abline(h=mean(clines$corr),lwd=1,lty=3)
>
> #both ordinary metanalysis and funell plot
> #indicate that net effects
> #exists and correlation is positive
> #what if we take phylogeny into account?
```

It seems that there's an overall tendency in mammals to be bigger as they live further from the equator. If this phenomenon was due to ecological processes it might indicate that, as endotherms, mammals tend to be larger in cooler climate to conserve heat. Such pattern would thus indicate that during evolution mammals evolved this mechanism of saving body heat. However, such correlation of body size and latitudinal distribution could also arise simply during evolutionary history as a result of non-random migration patterns etc. If so we would expect that closely related species would show similar relationship of body size vs. latitude; in other words, in such a scenario phylogenetic variation would explain large proportion of variance in our effect size measures. To test this we perform comparative meta-analysis, taking into account phylogeny of mammals. As it turns out – the overall effect disappears clearly showing that any observed relationships are only due to shared evolutionary history.

```
> ###code block C2
>
> library(ape)
> mammals <- read.nexus("mammals.nex")
> plot(mammals,cex=0.75)
>
> names(clines)[1]<-"animal"
> prior.c2 <- list(R=list(V=1,nu=0.002),
+ G=list(G1=list(V=1,nu=0.002)))
> m.c13 <- MCMCglmm(FisherZ~1, verbose=F, prior=prior.c2,
+ data=clines,
+ mev=clines$mev,random=~animal,
+ pedigree=mammals,
+ nitt=150000,burnin=30000,thin=150)
Warning message:
In MCMCglmm(FisherZ ~ 1, verbose = F, prior = prior.c2, data = clines,  :
  some combinations in animal do not exist and 34 missing records have been
generated
> plot(m.c13)
Waiting to confirm page change...
> summary(m.c13)

 Iterations = 149851
 Thinning interval  = 30001
 Sample size  = 800

 DIC: 58.62508

 G-structure:  ~animal

       post.mean  l-95% CI u-95% CI eff.samp
animal    0.3707 0.0003644   0.9974    577.9

 R-structure:  ~units

      post.mean l-95% CI u-95% CI eff.samp
units    0.2259 0.001698   0.4743    512.4

 Location effects: FisherZ ~ 1

            post.mean l-95% CI u-95% CI eff.samp pMCMC
(Intercept)   0.34912 -0.09985  0.91288    699.7  0.13
> diag(autocorr(m.c13$VCV)[2,,])
                                      animal
                                   0.1605732
leg(mev, -1, FALSE):leg(mev, -1, FALSE).meta
                                         NaN
                                       units
                                   0.2185705
> #should be ran for longer - >300000
> #BUT effect we looked for disappeared...
> #conclusions?
```

Advanced issues – parameter expansion

One drawback of using MCMC is it's randomness and sensitivity to the *i*-1th values of the chain. In practise it means that if in our model some variance components yield low values, close to zero, the chain may be trapped at some low value close to zero causing mixing-problems and in general poor convergence. It may also happen when some parameters, such as correlations, are close their space boundaries (-1 and 1). Such problems arise especially when residual variance *per se* cannot be estimated, as it is in binomial or Poisson models.

We can try alleviate these problems by using stronger priors – or improper priors. However, there's one much better solution caused parameter expansion. Assume we have the design matrix $\mathbf{W}$ of the form $[\mathbf{X}\ \mathbf{Z_1}\ \mathbf{Z_2}\ \ldots\ \mathbf{Z_k}]$. We can rescale this matrix (and thus – whole MC-sampled parameter space) by some parameters $\boldsymbol{\alpha} = [1, \alpha_1, \alpha_2, \ldots \alpha_k]$. This would yield $\mathbf{W}_{\boldsymbol{\alpha}} = [\mathbf{X}\ \mathbf{Z_1}\ \alpha_1\ \mathbf{Z_2}\ \alpha_2\ \ldots\ \mathbf{Z_k}\ \alpha_k]$. With these alphas we would actually sample new location effects that could be rescaled to original values: $\boldsymbol{\theta} = (\mathbf{I}_{\beta} \bigoplus^{k}_{i=1} \mathbf{I}_{u\{i\}} \cdot \alpha_i)\boldsymbol{\theta}_{\alpha}$. Likewise, rescaling could also be applied to (co)variance matrices: $\mathbf{V} = Diag(\boldsymbol{\alpha}_V)\ \mathbf{V}_{\alpha}\ Diag(\boldsymbol{\alpha}_V)'$.

Here, we'll analyse data on sex-ration in blue tits (you already know this dataset) using both parameter-expanded and standard priors. We'll compare mixing properties of these runs.

```
> ###code block C3
>
> data(BTdata)
> #we'll remove unkown sex
> BTdata$sex[which(BTdata$sex=="UNK")]<-NA
> BTdata$sex<-gdata::drop.levels(BTdata$sex)
> #we remove UNK level from the variable
>
> prior.c31 <-list(R=list(V=1,fix=1),
+ G=list(G1=list(V=1,nu=0.002,
+ alpha.mu=0,alpha.V=1000)))
> prior.c32 <- list(R=list(V=1,fix=1),
+ G=list(G1=list(V=1,nu=0.002)))
> m.c4a <- MCMCglmm(sex~1,random=~dam,data=BTdata,
+ family="categorical",prior=prior.c31,verbose=F,
+ nitt=25000,burnin=5000,thin=25)
> m.c4b <- MCMCglmm(sex~1,random=~dam,data=BTdata,
+ family="categorical",prior=prior.c32,verbose=F,
+ nitt=25000,burnin=5000,thin=25)
> plot(mcmc.list(m.c4a$VCV[,"dam"],m.c4b$VCV[,"dam"]),
+ col=c("red","green"))
> effectiveSize(m.c4a$VCV[,"dam"])
   var1
378.695
> effectiveSize(m.c4b$VCV[,"dam"])
    var1
```

```
157.4276
>
>
> #we might try equlize prior densities
> #for both models and use alternative priors
> #proper for PE model and improper
> #flat for sd for the second model
> prior.c31 <-list(R=list(V=1,fix=1),
+ G=list(G1=list(V=1,nu=1,
+ alpha.mu=0,alpha.V=1000)))
> prior.c32 <- list(R=list(V=1,fix=1),
+ G=list(G1=list(V=1e-16,nu=-1)))
> m.c4a <- MCMCglmm(sex~1,random=~dam,data=BTdata,
+ family="categorical",prior=prior.c31,verbose=F,
+ nitt=25000,burnin=5000,thin=25)
> m.c4b <- MCMCglmm(sex~1,random=~dam,data=BTdata,
+ family="categorical",prior=prior.c32,verbose=F,
+ nitt=25000,burnin=5000,thin=25)
> plot(mcmc.list(m.c4a$VCV[,"dam"],m.c4b$VCV[,"dam"]),
+ col=c("red","green"))
> effectiveSize(m.c4a$VCV[,"dam"])
    var1
410.4393
> effectiveSize(m.c4b$VCV[,"dam"])
    var1
111.8734
```

The reason for which second set of priors yields similar densities is that in expanded priors distribution of variance is no longer Inverse Wishart. It follows non-central F distribution of the form **`df(v/alpha.V,df1=1,df2=nu,ncp=(alpha.mu^2)/alpha.V)`**, which for standard deviations yields density: **`2*dt(sqrt(v)/sqrt(alpha.V), df=nu,ncp=alpha.mu/sqrt(alpha.V))`**. Second distribution is equal to the proper Cauchy prior for standard deviation. As you can see parameter expansion is useful when inverse gamma priors are to strong and flat priors, apart from being improper, yield distorted SD posteriors.

We'll see how much prior specification influences posterior distributions using the example provided by Gelman (2006), original proposer of parameter expansion.

```
> ###code block C4
>
> library(rbugs)
> data(schools)
>
> prior.c41 <- list(R=list(V=diag(schools$sd^2),fix=1),
+ G=list(G1=list(V=1e-16,nu=-1)))
> m.c41 <- MCMCglmm(estimate~1,random=~school,rcov=~idh(school):units,
+ data=schools,prior=prior.c41,verbose=F)
> sd1<-sqrt(m.c41$VCV[,1])
> ####OPTIONAL CODE####
```

```
> hist(sd1[which(sd1<35)],breaks=40)
> abline(h=50)
> ####END OF OPTIONAL CODE####
>
> prior.c42 <- list(R=list(V=diag(schools$sd^2),fix=1),
+ G=list(G1=list(V=1,nu=0.002)))
> m.c42 <- MCMCglmm(estimate~1,random=~school,rcov=~idh(school):units,
+ data=schools,prior=prior.c42,verbose=F)
> sd2<-sqrt(m.c42$VCV[,1])
> ####OPTIONAL CODE####
> hist(sd2[which(sd2<35)],breaks=40)
> xv<-seq(1e-16,35,length=200)
> dv<-MCMCpack::dinvgamma(xv,shape=0.001,scale=0.001)
> lines(1000*sqrt(dv)~xv)
> ####END OF OPTIONAL CODE####
>
> prior.c43 <- list(R=list(V=diag(schools$sd^2),fix=1),
+ G=list(G1=list(V=1,nu=1,alpha.mu=0,alpha.V=25^2)))
> m.c43 <- MCMCglmm(estimate~1,random=~school,rcov=~idh(school):units,
+ data=schools,prior=prior.c43,verbose=F)
> sd3<-sqrt(m.c43$VCV[,1])
> ####OPTIONAL CODE####
> hist(sd3[which(sd2<35)],breaks=40)
>
> dv<-(1/pi)*(25/((xv)^2+25^2))
> lines(4000*dv~xv)
> ####END OF OPTIONAL CODE####
```

You can see that prior distribution largely affects the outcome of the analysis, mainly in the form of effective sample size and mixing properties of the chain. As general indications, I would use parameter expansion every time chain experiences mixing problems and variances got stuck in values near to zero. Also, in binary data models, expansion improves chain maixing (which equivalently could be obtained by rescaling problematic parameters – such as residual variation – and then back-scaling resulting estimates), as shown below:

```
> ###code block C5
>
> prior.c5a <- list(R=list(V=10,fix=1),G=list(G1=list(V=1,nu=1,
+ alpha.mu=0, alpha.v=1000)))
> m.c5a <- MCMCglmm(sex~1,random=~dam,data=BTdata,family="categorical",
+ prior=prior.c5a,verbose=F,nitt=25000,burnin=5000,thin=25)
>
> c2<-((16*sqrt(3))/(15*pi))^2 #rescaling to Vres=0
> plot(mcmc.list(m.c4a$VCV[,1],m.c5a$VCV[,1]))
> plot(mcmc.list(m.c4a$VCV[,1]/(1+c2*m.c4a$VCV[,"units"]),
+ m.c5a$VCV[,1]/(1+c2*m.c5a$VCV[,"units"])))
>
>
> effectiveSize(m.c4a$VCV[,1])
    var1
410.4393
```

```
> effectiveSize(m.c5a$VCV[,1])
    var1
488.7346
> #we might even increase effective size using different sampling method
> m.c6a <- MCMCglmm(sex~1,random=~dam,data=BTdata,family="categorical",
+ prior=prior.c5a,verbose=F,nitt=25000,burnin=5000,thin=25,slice=T)
> effectiveSize(m.c6a$VCV[,1])
    var1
597.7756
```

Zero-inflated models

In biology often we end up with data where our treatments had no effect on the subject. It's especially apparent for count data, generated by Poisson processes. In such data, zeros are often – and sometimes too often. In MCMCglmm there's one special class of distributions – zero-inflated distributions, which in fact model two variables. E.g. in zero-inflated Poisson (ZIP), first variable models probability from a Poisson process, and second models probability (binomial) that zero comes from a zero-inflated process (yes or no). We have to account for this structure of effects in our (co)variance structure, remembering that covariance between these two processes cannot be estimated since they never occur together in one data point. To illustrate we will fit a ZIP model to data on PhD. Students publishing rates, compared to different features of their supervisors.

```
> ###code block C6
>
> library(pscl)
Loading required package: MASS

Attaching package: 'MASS'

The following object(s) are masked _by_ '.GlobalEnv':

    mammals

Loading required package: mvtnorm
Loading required package: gam
Loading required package: splines
Loading required package: akima
Classes and Methods for R developed in the
Political Science Computational Laboratory
Department of Political Science
Stanford University
Simon Jackman
hurdle and zeroinfl functions by Achim Zeileis
> data(bioChemists)
> head(bioChemists)
  art   fem     mar kid5  phd ment
1   0   Men Married    0 2.52    7
2   0 Women  Single    0 2.05    6
3   0 Women  Single    0 3.75    6
```

```
4   0   Men Married    1 1.18    3
5   0 Women  Single    0 3.75   26
6   0 Women Married    2 3.59    2
> #it seems there are lots of zeros in art
> sum(bioChemists$art==0)/length(bioChemists$art)#more than 30% are zeros
[1] 0.3005464
> #end we'd expect only 18% under Poisson process
> ppois(0,mean(bioChemists$art))
[1] 0.1839859
>
> prior.c71 <- list(R=list(V=diag(2),nu=0.002,fix=2))
> m.c71 <- MCMCglmm(art~trait-
1+at.level(trait,1):fem+at.level(trait,1):mar+
+ at.level(trait,1):kid5+at.level(trait,1):phd+at.level(trait,1):ment,
+ rcov=~idh(trait):units,data=bioChemists,prior=prior.c71,
+ family="zipoisson",verbose=F)
Warning message:
In MCMCglmm(art ~ trait - 1 + at.level(trait, 1):fem + at.level(trait,  :
  some fixed effects are not estimable and have been removed. Use
singular.ok=TRUE to sample these effects, but use an informative prior!
> plot(m.c71$Sol)
Waiting to confirm page change...
> quantile(boot::inv.logit(m.c71$Sol[,2]/sqrt(1+c2)))
         0%         25%         50%         75%        100%
0.004040984 0.008367092 0.011399349 0.019700200 0.036401536
> #or
> quantile(plogis(m.c71$Sol[,2]/sqrt(1+c2)))
         0%         25%         50%         75%        100%
0.004040984 0.008367092 0.011399349 0.019700200 0.036401536
>
> prior.c72<-list(R=list(V=1,nu=0.002))
> m.c72 <- MCMCglmm(art~fem+mar+kid5+phd+ment,data=bioChemists,
+ prior=prior.c72,family="poisson",verbose=F,saveX=T)
> #posterior check of the need of ZIP model
> ob.zer <- sum(bioChemists$art==0)
> nr.zer <- 1:1000
> for(i in 1:1000) {
+ pred1 <- rnorm(915,(m.c72$X%*%m.c72$Sol[i,])@x,sqrt(m.c72$VCV[i]))
+ nr.zer[i]<-sum(rpois(915,exp(pred1))==0)
+ }
> hist(nr.zer,breaks=20)
> abline(v=ob.zer,lwd=2)
```

As you can see – fitting ZIP model is simple. However, even when we think we need ZIP, it may be not really necessary – as seen here, based on naïve quantiles or post-fitting check based on predicted values.

Alternative for ZIP models can be found and it's called Hurdle models. They're very similar to ZIP models in that they also model two variables. However, the first one models the probability from zero-truncated Poisson distribution (Poisson process without zeros; in ZIP it was just Poisson process distribution) and the second one models binary process (yeas or not)

that the response is zero (in ZIP that was probability that zero comes from zero-inflation). We'll fit Hurdle model analogical to the ZIP model (previous section) and see how to interpret it. As it will be seen – Hurdle models show much better mixing properties.

```
> ###code block C7
>
> m.c73 <- MCMCglmm(art~trait-
1+at.level(trait,1):fem+at.level(trait,1):mar+
+ at.level(trait,1):kid5+at.level(trait,1):phd+at.level(trait,1):ment,
+ rcov=~idh(trait):units,data=bioChemists,prior=prior.c71,
+ family="hupoisson",verbose=F)
Warning message:
In MCMCglmm(art ~ trait - 1 + at.level(trait, 1):fem + at.level(trait,  :
  some fixed effects are not estimable and have been removed. Use
singular.ok=TRUE to sample these effects, but use an informative prior!
> plot(m.c73$Sol)
Waiting to confirm page change...
> c2 <- (16*sqrt(3)/(15*pi))^2
> #constant proportions of zeros across fixed effects
> #since only intercept fitted in hurdle process
> HPDinterval(boot::inv.logit(m.c73$Sol[,2]/sqrt(1+c2)))
         lower     upper
var1 0.2689855 0.3265462
attr(,"Probability")
[1] 0.95
> #proportion of zeros in non-zero truncated Poisson distribution
> HPDinterval(ppois(0,exp(m.c73$Sol[,1]+0.5*m.c73$VCV[,1])))
         lower     upper
var1 0.1503793 0.3547181
attr(,"Probability")
[1] 0.95
> #CIs overlap
```

OK, so there seems to be little support for Hurdle model as well – proportions of zeros generated in Hurdle model are similar to those generated by ordinary Poisson process. However, remember that in our model Hurdle process was crossed only with intercept, which means our conclusions should only for single women with no young children who obtained PhD from department without prestige and whose mentors published nothing in the past 3 years. Let's see if the same holds for e.g. men.

```
> ###code block C8
>
> HPDinterval(ppois(0,exp(m.c73$Sol[,1]+m.c73$Sol[,3]+
+ 0.5*m.c73$VCV[,1])))
          lower     upper
var1 0.09651753 0.2769605
attr(,"Probability")
[1] 0.95
```

This yields lower CI and suggests in case of men there may be zero-inflation. Let's verify this by relaxing our model and fitting separate Hurdle processes for males and women.

```
> ###code block C9
>
> m.c74 <- MCMCglmm(art~trait-
1+at.level(trait,1:2):fem+at.level(trait,1):mar+
+ at.level(trait,1):kid5+at.level(trait,1):phd+at.level(trait,1):ment,
+ rcov=~idh(trait):units,data=bioChemists,prior=prior.c71,
+ family="hupoisson",verbose=F)
Warning message:
In MCMCglmm(art ~ trait - 1 + at.level(trait, 1:2):fem + at.level(trait,  :
  some fixed effects are not estimable and have been removed. Use
singular.ok=TRUE to sample these effects, but use an informative prior!
> HPDinterval(boot::inv.logit((m.c74$Sol[,2]+m.c74$Sol[,4])/sqrt(1+c2)))
         lower     upper
var1 0.2327803 0.3107261
attr(,"Probability")
[1] 0.95
> #expected zeros from Hurdle proces shrunk
> HPDinterval(ppois(0,exp(m.c74$Sol[,1]+m.c74$Sol[,3]+0.5*m.c74$VCV[,1])))
         lower     upper
var1 0.0768166 0.2500347
attr(,"Probability")
[1] 0.95
> #but in males still there seem to be zero-deflation
```

Still, zero-inflation in men is detectable. This highlights one serious drawback of Hurdle models – if we want to model zero-inflation adequately, we have to build complex models with as many Hurdle-related probabilities as we have fixed effects. In ZIP models it's simpler, but in the absence of zero-inflation we risk computational problems.

Zero-altered models

Do we really need another zero-something? Zero-altered models are similar to Hurdle models, but instead of using logit link we use log-log link. Such formulation allows for fitting both zero-inflation and zero-deflation in the Poisson process. Formulation of such model should follow two simple rules: we interact residual variance with the trait effect, equalizing overdispersion in both Poisson and zero-altering processes, and we contrast (interact) two processes with fixed effects yielding two sets of parameters: original coefficients for Poisson process and second set of coefficients for zero-altering process (if they're zero – there's no zero-altering; if they're negative – we detect zero-inflation, when they're positive – there's zero-deflation).

```
> ###code block C10
>
> m.c8 <- MCMCglmm(art~trait*(fem+mar+kid5+phd+ment),
+ rcov=~trait:units, data=bioChemists,
+ family="zapoisson", verbose=F)
> summary(m.c8)
```

```
 Iterations = 12991
 Thinning interval  = 3001
 Sample size  = 1000

 DIC: 3040.3

 R-structure:  ~trait:units

           post.mean l-95% CI u-95% CI eff.samp
trait:units    0.3619   0.2652   0.4803     49.6

 Location effects: art ~ trait * (fem + mar + kid5 + phd + ment)

                      post.mean l-95% CI u-95% CI eff.samp
(Intercept)             0.35604  0.03343  0.73650    201.8
traitza_art            -0.55981 -1.17703 -0.07475    184.7
femWomen               -0.20474 -0.36742 -0.04315    309.4
marMarried              0.08079 -0.10376  0.25805    299.2
kid5                   -0.12991 -0.24227 -0.01249    287.2
phd                     0.01300 -0.06669  0.10893    265.2
ment                    0.01938  0.01295  0.02710    360.5
traitza_art:femWomen    0.02373 -0.28562  0.27828    193.3
traitza_art:marMarried  0.16840 -0.12911  0.48070    241.9
traitza_art:kid5       -0.08838 -0.31326  0.09718    169.4
traitza_art:phd         0.01530 -0.12286  0.15102    207.2
traitza_art:ment        0.02851  0.01372  0.04408    112.4
                       pMCMC
(Intercept)            0.058 .
traitza_art            0.044 *
femWomen               0.022 *
marMarried             0.392
kid5                   0.022 *
phd                    0.802
ment                  <0.001 ***
traitza_art:femWomen   0.870
traitza_art:marMarried 0.272
traitza_art:kid5       0.386
traitza_art:phd        0.804
traitza_art:ment      <0.001 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Interpretation is simple – the more papers our mentor produces, the greater zero-deflation (listen to that! ;).

Analysing survival data in R

Survival data are difficult – both to fit and interpret. Main difficulty is that variance in such data increases in a non-linear way with the mean, following exponential function (errors are gamma-distributed). Data of such form are most often measurements of time to death or time to some

kind of failure. They pose a special problem and thus we treat them separately. Efficient tools exist in R for dealing with such data and thus we'll leave MCMCglmm for a while.

We'll use sample data on lung cancer survival. Before fitting data has to be prepared. They should be in general of form "start time", "stop time", "status" or "time to event", "status", where status is 0 when the event (e.g. death) didn't occur and 1 when it did occur. Data subjects with 0 are called right-censored because we don't have information on the occurrence of the event, we only know it will probably happen in the future (e.g. subject will eventually die).

```
> ###code block C11
>
> install.packages("survival");library(survival)
> data(lung)
> head(lung)
  inst time status age sex ph.ecog ph.karno pat.karno
1    3  306      2  74   1       1       90       100
2    3  455      2  68   1       0       90        90
3    3 1010      1  56   1       0       90        90
4    5  210      2  57   1       1       90        60
5    1  883      2  60   1       0      100        90
6   12 1022      1  74   1       1       50        80
  meal.cal wt.loss
1     1175      NA
2     1225      15
3       NA      15
4     1150      11
5       NA       0
6      513       0
> ?lung
```

Analysing the data may include plotting overall survival patterns and comparing specific groups with respect to survival.

```
> ###code block C12
>
> surlun <- with(na.omit(lung), Surv(time,status))
> m.c91 <- survfit(surlun~1,data=na.omit(lung))
> plot(m.c91,yscale=100) #general distribution of survival
>
> m.c92 <- survfit(surlun~sex,data=na.omit(lung))
> plot(m.c92,yscale=100,col=c("red","blue")) #sex-specific survival
distribution
> #and test therefor
> survdiff(surlun~sex,data=na.omit(lung))
Call:
survdiff(formula = surlun ~ sex, data = na.omit(lung))

        N Observed Expected (O-E)^2/E (O-E)^2/V
sex=1 103       82     68.7      2.57      6.05
sex=2  64       38     51.3      3.44      6.05
```

```
 Chisq= 6  on 1 degrees of freedom, p= 0.0139
```

Finally, if our data are associated with some continuous predictors, we can see if any of these predictors predict the probability of survival. In such case, **`coxph()`** function is used.

```
> ###code block C13
>
>
> m.c93 <-
coxph(surlun~age+ph.ecog+ph.karno+pat.karno,data=na.omit(lung),x=T)
> m.c93
Call:
coxph(formula = surlun ~ age + ph.ecog + ph.karno + pat.karno,
    data = na.omit(lung), x = T)


              coef exp(coef) se(coef)     z      p
age        0.01269      1.01  0.01146  1.11 0.2700
ph.ecog    0.59875      1.82  0.22163  2.70 0.0069
ph.karno   0.02077      1.02  0.01168  1.78 0.0750
pat.karno -0.00965      0.99  0.00766 -1.26 0.2100

Likelihood ratio test=17.6  on 4 df, p=0.00146  n= 167
> cox.zph(m.c93)#testing for assumptions of survival analysis
             rho  chisq     p
age       0.0721 0.7102 0.399
ph.ecog   0.0110 0.0143 0.905
ph.karno  0.1537 2.2380 0.135
pat.karno 0.0634 0.5280 0.467
GLOBAL        NA 7.1518 0.128
```

Try analysing data on rats that have been injected with carcinogen and then subjected to experimental treatment (administered with either placebo or drug).

```
> ###code block C14
>
> data(rats)
> summary(rats)
     litter           rx              time
 Min.   : 1.0   Min.   :0.0000   Min.   : 34.00
 1st Qu.:13.0   1st Qu.:0.0000   1st Qu.: 78.25
 Median :25.5   Median :0.0000   Median : 94.50
 Mean   :25.5   Mean   :0.3333   Mean   : 89.43
 3rd Qu.:38.0   3rd Qu.:1.0000   3rd Qu.:104.00
 Max.   :50.0   Max.   :1.0000   Max.   :104.00
     status
 Min.   :0.0000
 1st Qu.:0.0000
 Median :0.0000
 Mean   :0.2667
```

```
 3rd Qu.:1.0000
 Max.   :1.0000
> ratss <- with(rats,Surv(time,status))
> m.c94 <- survfit(ratss~rx,data=rats)
> plot(m.c94,yscale=100,col=c("green","red"))
```

Part 4

Overview

1. Presentation – multivariate methods, their logic, dangers and indications of use.
2. DIY – simple ordination
    a. Principal component analysis in R
    b. Factor analysis in R
    c. Multidimensional scaling
3. DIY – clustering methods
    a. Clustering methods in R – agglomerating methods
    b. Hierarchical methods
4. DIY – multiple dependent and independent variables
    a. Canonical correlation in R
5. Discriminant function analysis
6. Visualizing multivariate data
7. DIY (if time permits) – introduction to tree-regression and its use in multivariate problems.

Principal component analysis

Briefly, when you have to much (presumably correlated) predictors you will probably start with PCA. It allows transforming your data into a series of linear combinations based on eigenvalue calculations. These combinations called components are formed in such a way that the first one explains the majority of variance, and the next ones explain decreasing proportions of variance from original data. Of course, when predictors are entirely uncorrelated decomposition will be meaningless and PCA doesn't make any sense.

Here and throughout this part we will use datasets provided by Crawley (2010) and Manly (2005). Our first example is from Bumpus (1898) and describes survival rates of sparrows that were injured during storm. Bumpus took these birds, observed their survival and recorded several body measurements to see if they relate to survival in any way. In this he saw opportunity to test Darwin's theory of natural selection.

```
> ###code block D1
>
> sparr<-read.table("sparr.txt",sep="\t",head=T)
> sparr<-sparr[,-c(8:10)]#check if last columnsare improper
> sparrpca<-sparr[,2:6]
> sparrpca<-na.omit(sparrpca)
> for(i in 1:5){
+ sparrpca[,i]<-(sparrpca[,i]-mean(sparrpca[,i]))/sqrt(var(sparrpca[,i]))
+ }
>
> m.d1 <- prcomp(sparrpca)
```

```
> summary(m.d1)#check variance partitioning
Importance of components:
                          PC1   PC2    PC3    PC4    PC5
Standard deviation     1.902 0.729 0.6216 0.5491 0.4056
Proportion of Variance 0.723 0.106 0.0773 0.0603 0.0329
Cumulative Proportion  0.723 0.830 0.9068 0.9671 1.0000
> m.d1$rotation#see eigenvectors
                PC1         PC2         PC3         PC4
totLen    0.4517989 -0.05072137  0.6904702 -0.42041399
alarExt   0.4616809  0.29956355  0.3405484  0.54786307
beakHead  0.4505416  0.32457242 -0.4544927 -0.60629605
lenHume   0.4707389  0.18468403 -0.4109350  0.38827811
keelStern 0.3976754 -0.87648935 -0.1784558  0.06887199
                PC5
totLen     0.3739091
alarExt   -0.5300805
beakHead  -0.3427923
lenHume    0.6516665
keelStern -0.1924341
> #it seems that pc1 describes the size of a bird and
> #pc2 describes its shape (is contrasts 2 groups of variables)
> plot(m.d1)#or do it here
> biplot(m.d1)
> abline(h=0,lty=2)
> abline(v=0,lty=2)
>
> as.matrix(sparrpca)%*%as.matrix(m.d1$rotation[,1,drop=F])
           PC1
1   0.06428901
2  -2.18031283
3  -1.14556567
4  -2.31106565
5  -0.29504203
6   1.91626198
7  -1.05036763
8   0.43854156
9   2.69147373
10  0.18568959
11  0.37111481
12  0.26770575
13  2.35924685
14  0.71464741
15 -1.39425236
16 -1.55867849
17  0.54832983
18 -1.65771758
19 -1.77666826
20  2.17605614
21 -0.45737249
22 -0.96511115
23 -0.65805539
24  1.58405400
```

```
25 -3.71680462
26  2.12356038
27 -1.32885487
28  1.72330431
29  3.99433726
30 -3.71422633
31  0.14837847
32  1.19507659
33  1.02993746
34 -0.71482109
35 -0.31746568
36  2.79633717
37 -4.24025643
38 -0.54188759
39 -1.90570270
40  4.07138353
41  0.06283901
42 -0.93831834
43 -0.42284580
44  1.58678784
45 -2.50895504
46  1.61879121
47 -1.55900792
48  1.55698964
49  2.13422241
> pcas <- data.frame("id"=1:length(sparrpca[,1]))
> for (i in 1:5) {
+ a<-as.matrix(sparrpca) %*% as.matrix(m.d1$rotation[,i,drop=F])
+ pcas<-cbind(pcas,a)
+ }
> pcas<-cbind(pcas,sparr$surv[1:49])
> names(pcas)[7]<-"surv"
> plot(PC2~PC1,data=subset(pcas,surv==2),pch=20,col="red")
> points(PC2~PC1,data=subset(pcas,surv==1))
> abline(v=0,lty=2)
> abline(h=0,lty=2)
> t.test(pcas$PC1~pcas$surv)

        Welch Two Sample t-test

data:  pcas$PC1 by pcas$surv
t = -0.3315, df = 46.754, p-value = 0.7418
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -1.2335820  0.8846343
sample estimates:
mean in group 1 mean in group 2
    -0.09969935      0.07477451

> var.test(pcas$PC1~pcas$surv)

        F test to compare two variances
```

```
data:  pcas$PC1 by pcas$surv
F = 0.4788, num df = 20, denom df = 27, p-value =
0.09353
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.2124834 1.1371463
sample estimates:
ratio of variances
         0.4787834

> if(!require("Rcmdr")) install.packages("Rcmdr");require("Rcmdr")
Loading required package: Rcmdr
Loading required package: tcltk
Loading Tcl/Tk interface ... done
Loading required package: car
Loading required package: nnet

Rcmdr Version 1.6-0


Attaching package: 'Rcmdr'

The following object(s) are masked from 'package:tcltk':

    tclvalue

> leveneTest(pcas$PC1~as.factor(pcas$surv),data=pcas,center=median)
Levene's Test for Homogeneity of Variance (center = median)
      Df F value Pr(>F)
group  1  2.8692 0.0969 .
      47
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
> #it seems that survivors are more average
> #nonsurvivors are more variable nad extreme
```

What we get is apparent simplification of our predictors into just two: size and shape related. It makes the whole analysis easier and more intuitive. This is especially important in case of sociological and cultural data where often datasets are huge and comprise dozens of variables. Let's analyse such complex example. These data show percentages of employment in different branches of economy for 30 countries. The question is – could we describe these countries using fewer variables and thus reducing dimensionality of the problem?

```
> ###code block D2
>
> employ <- read.table("employ.txt",head=T,sep="\t")
> summary(employ)
              Country       Group         AGR
 Albania          : 1   Eastern: 8   Min.   : 0.00
 Austria          : 1   EFTA   : 6   1st Qu.: 4.40
```

```
 Belgium           : 1   EU     :12   Median : 8.45
 Bulgaria          : 1   Other  : 4   Mean   :12.19
 Cyprus            : 1                3rd Qu.:14.93
 Czech/Slovak Reps: 1                Max.   :55.50
 (Other)           :24
      MIN              MAN              PS
 Min.   : 0.000   Min.   : 0.00   Min.   :0.000
 1st Qu.: 0.125   1st Qu.:19.00   1st Qu.:0.275
 Median : 0.500   Median :20.30   Median :0.800
 Mean   : 3.447   Mean   :20.29   Mean   :0.800
 3rd Qu.: 1.050   3rd Qu.:24.55   3rd Qu.:1.175
 Max.   :37.300   Max.   :38.70   Max.   :2.200

      CON              SER              FIN
 Min.   : 0.60   Min.   : 3.30   Min.   : 0.000
 1st Qu.: 6.40   1st Qu.:12.62   1st Qu.: 3.300
 Median : 7.05   Median :16.80   Median : 7.150
 Mean   : 7.53   Mean   :15.64   Mean   : 6.650
 3rd Qu.: 9.10   3rd Qu.:19.62   3rd Qu.: 9.325
 Max.   :16.90   Max.   :24.50   Max.   :15.300

      SPS              TC               X
 Min.   : 0.00   Min.   :3.000   Min.   : 99.80
 1st Qu.:22.95   1st Qu.:5.800   1st Qu.: 99.90
 Median :27.00   Median :6.750   Median :100.00
 Mean   :26.99   Mean   :6.453   Mean   : 99.98
 3rd Qu.:33.17   3rd Qu.:7.150   3rd Qu.:100.00
 Max.   :41.60   Max.   :8.800   Max.   :100.10

   X.1             X.2              X.3
 Mode:logical   Mode:logical   Min.   :1989
 NA's:30        NA's:30        1st Qu.:1990
                               Median :1991
                               Mean   :1991
                               3rd Qu.:1992
                               Max.   :1995
```

```
> employ<-employ[,-c(13:15)]
> emppca <- employ[,3:11]
> for(i in 1:length(names(emppca))){
+ emppca[,i]<-(emppca[,i]-mean(emppca[,i]))/sqrt(var(emppca[,i]))
+ }
>
> rownames(emppca)<-employ[,1]
> m.d2<-prcomp(emppca)
> plot(m.d2)
> biplot(m.d2,cex=0.5)
> abline(v=0,lty=2)
> abline(h=0,lty=2)
>
> summary(m.d2)
Importance of components:
```

```
                          PC1   PC2   PC3   PC4    PC5
Standard deviation     1.764 1.345 1.223 1.031 0.8428
Proportion of Variance 0.346 0.201 0.166 0.118 0.0789
Cumulative Proportion  0.346 0.547 0.713 0.831 0.9102
                           PC6    PC7    PC8     PC9
Standard deviation     0.5580 0.5417 0.4515 0.00266
Proportion of Variance 0.0346 0.0326 0.0226 0.00000
Cumulative Proportion  0.9447 0.9774 1.0000 1.00000
> m.d2$rotation
           PC1          PC2         PC3         PC4
AGR  0.5114918  0.023474999  0.27859140 -0.01649218
MIN  0.3749833 -0.000490734 -0.51505210 -0.11360623
MAN -0.2461613 -0.431752051  0.50205622 -0.05827010
PS  -0.3161203 -0.109144430  0.29369499 -0.02324549
CON -0.2215986  0.242470912 -0.07153072 -0.78266601
SER -0.3815359  0.408255893 -0.06514938 -0.16903778
FIN -0.1310884  0.552938958  0.09565440  0.48921763
SPS -0.4281618 -0.054705874 -0.36015928  0.31724250
TC  -0.2050706 -0.516649883 -0.41299565  0.04206329
            PC5         PC6         PC7         PC8
AGR  0.02403794 -0.04239691 -0.16357428 -0.54040909
MIN -0.34631272  0.19857439  0.21259036  0.44859201
MAN  0.23362179 -0.03091715  0.23601541  0.43175735
PS  -0.85444839  0.20647051 -0.06056504 -0.15512240
CON -0.06215096 -0.50263565 -0.02028469 -0.03082345
SER  0.26667324  0.67269361  0.17483893 -0.20175280
FIN -0.13128795 -0.40593492  0.45764510  0.02726352
SPS  0.04571821 -0.15845276 -0.62133030  0.04147562
TC   0.02290077 -0.14189804  0.49214521 -0.50212355
           PC9
AGR 0.58203611
MIN 0.41881803
MAN 0.44708636
PS  0.03025124
CON 0.12865575
SER 0.24502068
FIN 0.19075812
SPS 0.41031481
TC  0.06074315
>
> #it seems that here more pcs are meaningfull
> #pc1 basically contrasts agriculture and mining with
> #the remining branches whereas pc2 is positively
> #influenced by finances and services and negatively
> #by manufacturing and transportation
>
> if(!require("FactoMineR"))
install.packages("FactoMineR");require("FactoMineR")
Loading required package: FactoMineR
Loading required package: ellipse

Attaching package: 'ellipse'
```

```
The following object(s) are masked from 'package:car':

    ellipse

Loading required package: cluster
Loading required package: scatterplot3d
> #tryFactoMineR
> #FactoMineR(emppca)
> m.d3<-PCA(emppca)
```

This is but the foretaste of more sophisticated multivariate statistics, but as seen here, simple PCA analysis can cluster countries based on some sociological and economic measures. To practise – try analysing data called "protein" depicting the patterns of protein acquisition in different countries with regard to different food types.

Factor analysis

Factor analysis is complementary to PCA. However, whereas PCA returns as many variables as there were predictors and puts most of data variance into several first PCs, factor analysis estimates some (most often small) number of factors that (as we hope) summarise our data with respect to some abstract, unmeasured variables. E.g. based on several geographical and geological measures, plus some plant-distribution variables, we could develop several factors allowing for easy quantification of the character of the community (e.g. its "grassiness", "it's tendency to being wet", "it's xerothermic-like character" etc.). There's one difficulty in factor analysis – we have to arbitrarily choose the number of factors to estimate. Thus, often people decide to begin factor building with few first PCs from simple PCA. Alternatively we may just decide and see what happens. Factor analysis is more exploratory and subjective, but carefully done and interpreted may be really useful. Either PCA-based or non-PCA-based methods are often defaults in statistical software.

Here we'll build on data both from biology and sociology. First analysis should explain everything.

```
> ###code block D3
>
>
> m.d4<-factanal(employ[,3:11],4,rotation="varimax")
> print(m.d4,digits=2,sort=T)

Call:
factanal(x = employ[, 3:11], factors = 4, rotation = "varimax")

Uniquenesses:
 AGR  MIN  MAN   PS  CON  SER  FIN  SPS   TC
0.00 0.00 0.00 0.74 0.60 0.17 0.51 0.00 0.47

Loadings:
    Factor1 Factor2 Factor3 Factor4
```

```
AGR -0.81   -0.25   -0.53
SPS  0.95                    0.29
TC   0.61           -0.10   -0.37
MIN         -0.87           -0.47
MAN          0.94           -0.31
CON                  0.63
SER  0.22    0.15    0.78    0.38
FIN                  0.43    0.55
PS   0.20    0.41    0.21

               Factor1 Factor2 Factor3 Factor4
SS loadings       2.03    1.92    1.55    0.99
Proportion Var    0.23    0.21    0.17    0.11
Cumulative Var    0.23    0.44    0.61    0.72

Test of the hypothesis that 4 factors are sufficient.
The chi square statistic is 181.1 on 6 degrees of freedom.
The p-value is 1.98e-36
> #loadings help us to assign meaning to factors
> #factor 1: agriculture and other rural rather than services
> #factor 2: lack of finance
> #factor 3: mining rather than manufacturing
> #factor 4: construction and service
>
> load<-m.d4$loadings
> XG<-as.matrix(employ[,3:11])%*%as.matrix(load)
> #to get factor scores for data ponits
> #we have to solve: F*=X%*%G%*%solve(t(G)%*%G)
> GG<-solve(t(as.matrix(load))%*%as.matrix(load))
> XG%*%GG
         Factor1     Factor2     Factor3     Factor4
 [1,]  18.527338   7.1476858  -1.5117551  11.8167832
 [2,]  17.485692   7.0058911  -3.5327028  12.6937944
 [3,]  15.235347   7.0760577  -0.2809193  11.4530867
 [4,]  11.694210   9.8441156   4.3784289   5.5575214
 [5,]   2.533935   8.1776177  -0.8339880   7.9391785
 [6,]   7.893162   7.7079987   0.2500979   9.6644631
 [7,]  10.173907   8.3226035   4.0960946   5.9972304
 [8,]  13.177124   7.1642395   5.5754666   7.7943653
 [9,]  19.905801   6.1136986  -5.3318075  17.2510259
[10,]   6.971428   9.9356712   3.0908998   6.1196875
[11,]   9.158649   8.3201096   3.5580757   6.6667216
[12,]  12.903817   7.8124382   4.8032089   8.6478574
[13,]   7.506946  11.9125545   5.0777618   2.3297695
[14,]  15.003910   6.7854931  -3.0130551  11.7617878
[15,]  12.150673   6.7632404  -0.8861064  10.4053888
[16,]  19.179363   3.4120072  -2.5791257  13.9305401
[17,]  20.402234   5.8200999  -4.1117375  14.0347627
[18,]   7.774464  10.3733198   6.9319765   4.8052685
[19,] -16.599403  -9.8522857  -8.8928128  11.5968394
[20,]   2.970290  16.8623595  -4.5397876  -0.8424827
[21,]  11.242079 -20.3652527   7.5463535 -11.3615473
```

```
[22,]  13.770822 -16.6189733   2.4668140  -3.7616371
[23,]   4.535917   9.0542604  -6.6319136   5.1902652
[24,]  -1.511447  18.2113839  -3.9804436  -5.8138159
[25,]   6.576236  13.0223493  -5.2595669   1.7702921
[26,]   5.218100  18.1353546   5.6429925  -9.0575058
[27,]   4.924561   7.7653555   5.9706257   6.6221596
[28,]  16.349561  -0.1049708  10.8686056  13.3402483
[29,]  21.466433  10.8095610  -7.7651874   8.7648192
[30,]  -7.070818   6.7196373 -11.0696070  13.7138604
> #now it seems ovious that e.g. Turkey and Albania
> #are dominated by rural branches of economy
> #and Bulgaria, Hungary and former USSR have few
> #employed in finance
```

As you can see factor analysis is quite straightforward. The only glitch we might want to see is this arbitrary choice of the number of factors we want to have. However, based on the correlation matrix for analysed predictors and its eigenvalues, it's possible to get some vague indication of how many factors to develop.

```
> ###code block D4
>
> if(!require("nFactors")) install.packages("nFactors");
>
> ev <-eigen(cor(employ[,3:11]))
> ap <- parallel(subject=nrow(employ[,3:11]),
+ var=ncol(employ[,3:11]),
+ rep=100, cent=.05)
> nS <- nScree(ev$values, ap$eigen$qevpea)
> plotnScree(nS)
>
> #classical interpretation is to use as many factors
> #as there are eigenvalues greater than one
```

We can analyse much more complex data from Crawley (2005), provided in Crawley (2010). These are data on 54 plant species abundance, plus some environmental variables, gathered on nearly 100 plots. Factor analysis generating 8 factors yields interesting insights. Two-letter shortcuts are not helpful, maybe even plant names would not be. However, going through the list: factor 1 has large +loadings AE, AP, AS, and large – for AC, AO, FR: thus it represents continuum between tall neutral grassland and short acidic grassland. Factor 2 has positive loadings from high pH low plants, whereas negative from acidic low-growing plants. Factor 3 captures high positive correlations with legume (N-fixing) plants.

```
> ###code block D5
>
> plspec <- read.table("pgfull.txt", head=T, sep="\t")
> spec <- plspec[,1:54]
> m.d5 <- factanal(spec,8,rotation="varimax")
> m.d5
```

```
Call:
factanal(x = spec, factors = 8, rotation = "varimax")

Uniquenesses:
   AC    AE    AM    AO    AP    AR    AS    AU    BH    BM    CC    CF
0.638 0.086 0.641 0.796 0.197 0.938 0.374 0.005 0.852 0.266 0.056 0.574
   CM    CN    CX    CY    DC    DG    ER    FM    FP    FR    GV    HI
0.786 0.579 0.549 0.733 0.837 0.408 0.072 0.956 0.371 0.815 0.971 0.827
   HL    HP    HS    HR    KA    LA    LC    LH    LM    LO    LP    OR
0.921 0.218 0.332 0.915 0.319 0.305 0.349 0.333 0.927 0.121 0.403 0.005
   PL    PP    PS    PT    QR    RA    RB    RC    SG    SM    SO    TF
0.286 0.606 0.336 0.401 0.913 0.491 0.005 0.754 0.341 0.212 0.825 0.428
   TG    TO    TP    TR    VC    VK
0.476 0.469 0.309 0.611 0.651 0.170

Loadings:
   Factor1 Factor2 Factor3 Factor4 Factor5 Factor6 Factor7 Factor8
AC -0.512  -0.268                           0.121
AE  0.925  -0.107          -0.146          -0.118
AM -0.206   0.413   0.213           0.163   0.115   0.153   0.186
AO -0.312  -0.196  -0.151  -0.105          -0.148  -0.102
AP  0.827  -0.173  -0.195  -0.167          -0.123
AR          0.150           0.111                  0.127
AS  0.778
AU                                                          0.996
BH  0.380
BM -0.116   0.292           0.695                  0.380
CC -0.152                   0.159           0.943
CF          0.539                   0.342
CM                  0.434  -0.110
CN -0.276   0.143                           0.541   0.147
CX                          0.628           0.169   0.146
CY -0.211          -0.162   0.340                  0.270
DC         -0.125                           0.372
DG  0.738                  -0.127           0.145
ER                                  0.960
FM -0.108                                   0.133
FP  0.245   0.226           0.478   0.493          -0.176
FR -0.386          -0.144
GV -0.134
HI -0.202  -0.129  -0.163   0.182                  0.216
HL         -0.157          -0.127          -0.139
HP -0.155   0.832                                  0.240
HS  0.746  -0.102   0.257  -0.152
HR -0.155  -0.107  -0.122   0.101                  0.150
KA -0.167   0.774  -0.169   0.139
LA                                          0.829
LC -0.306   0.378  -0.125   0.529                          0.328
LH -0.256   0.556  -0.132   0.421           0.223   0.195
LM                                  0.112   0.221
LO -0.129   0.432           0.781                  0.251
LP  0.115           0.745
```

```
OR                                                              0.996
PL          0.369   0.675           0.337
PP  0.527           0.226  -0.167          -0.175
PS -0.212   0.301  -0.130   0.681           0.150   0.158
PT  0.741                  -0.100   0.150  -0.105
QR -0.194  -0.135
RA  0.195   0.227   0.578           0.205  -0.166  -0.107
RB -0.122   0.158           0.272                   0.934
RC  0.361                  -0.198          -0.176  -0.152
SG                                 0.806
SM          0.388                                   0.787
SO                 -0.100   0.386
TF          0.702   0.260
TG  0.141           0.583  -0.110           0.367   0.107
TO  0.418           0.567  -0.158
TP                  0.818
TR          0.141   0.306   0.238                   0.458
VC          0.403   0.246   0.309          -0.169
VK                                 0.909

                Factor1 Factor2 Factor3 Factor4 Factor5 Factor6 Factor7
SS loadings       5.840   3.991   3.577   3.540   3.028   2.644   2.427
Proportion Var    0.108   0.074   0.066   0.066   0.056   0.049   0.045
Cumulative Var    0.108   0.182   0.248   0.314   0.370   0.419   0.464
                Factor8
SS loadings       2.198
Proportion Var    0.041
Cumulative Var    0.505

Test of the hypothesis that 8 factors are sufficient.
The chi square statistic is 1675.57 on 1027 degrees of freedom.
The p-value is 5.92e-34
```

Multidimensional scaling

This one of my favourites. Quite often we end up with data on several subjects in the form of distance matrix, i.e. each cell of such matrix describes the distance between these two subjects/units on some scale. It may be geographical distance, but also e.g. difference in thinking measured by disagreements in questionnaire responses, of resemblance of plant-species composition. The key is that based on this distances and special recursive optimising algorithm we may estimate relative relationships of these 3 variables in some n-dimensional space (best if it's 2-D or, at worst, 3-D space – which both are easy to interpret and imagine).

As our first example we'll consider road distances on New Zealand's South Island. We'll see if scaling is able to disentangle actual geometric pattern of distribution in two-dimensional geographical space.

```
> ###code block D6
>
> nzsi <- read.table("nzsi.txt",head=T,sep="\t")
```

```
> nzsi<-nzsi[,-c(14:19)]
> nzsi<-nzsi[-c(14:19),]
> for(i in 1:length(names(nzsi))){
+ nzsi[,i]<-(nzsi[,i]-mean(nzsi[,i]))/sqrt(var(nzsi[,i]))
+ }
> m.d6 <- cmdscale(nzsi,eig=T,k=2)
> xv<-m.d6$points[,1]
> yv<-m.d6$points[,2]
> plot(xv,yv,type="n")
> text(xv,yv,labels=colnames(nzsi),cex=.7)
> #those who visited new zealand should see that
> #similarity of this pattern to actual map of SI
> #is striking
```

Another interesting example regards voting behaviour of American congressman in a public debate about environmental change. Here distances are numbers of questions two congressmen differed in their answers. Can multidimensional scaling find any pattern here?

```
> ###code block D7
>
> congress <- read.table("congress.txt",head=T,sep="\t")
> m.d7<-cmdscale(congress,eig=T,k=2)
> xv<-m.d7$points[,1]
> yv<-m.d7$points[,2]
> plot(xv,yv,type="n")
> text(xv,yv,labels=colnames(congress),cex=.7)
> abline(h=0,v=0,lty=2)
> #its apparent that our analysis manged to separate two political
> #fractions almost perfectly
> #interpreting second domension is tougher but it relates
> #to the number of absentions from voting - none for Maraziti
> #and almost half of votings for Sandman
```

In general multidimensional scaling should be used in situations when we want to depict some abstract distances geometrically. The more realistic and interpretable these geometric distances are the better. Ensuring this we avoid stepping into abstract interpretations.

Cluster analysis

Cluster analysis is mostly thought of as the mean to obtaining phylogenetic trees, which basically is true. Phylogenetic trees are one of possible outcomes of cluster analysis. But cluster analysis has much more general applications. As we've seen before, grouping of similar objects can be obtained using PCA – on the plane of first two PCs often we can see conspicuous grouping of objects. Here we'll use grouping methods more explicitly. We'll exploit one earlier data-set: on employment in European countries, and additional one, on phylogeny of canine species based on mandible measurements.

```
> ###code block D8
>
```

```
> #before clusteirng remove any NAs and standardize your data
>
> employ <- read.table("employ.txt",head=T,sep="\t")
> summary(employ)
              Country       Group         AGR              MIN
 Albania           : 1   Eastern: 8   Min.   : 0.00   Min.   : 0.000
 Austria           : 1   EFTA   : 6   1st Qu.: 4.40   1st Qu.: 0.125
 Belgium           : 1   EU     :12   Median : 8.45   Median : 0.500
 Bulgaria          : 1   Other  : 4   Mean   :12.19   Mean   : 3.447
 Cyprus            : 1                3rd Qu.:14.93   3rd Qu.: 1.050
 Czech/Slovak Reps: 1                 Max.   :55.50   Max.   :37.300
 (Other)           :24
      MAN              PS              CON             SER
 Min.   : 0.00   Min.   :0.000   Min.   : 0.60   Min.   : 3.30
 1st Qu.:19.00   1st Qu.:0.275   1st Qu.: 6.40   1st Qu.:12.62
 Median :20.30   Median :0.800   Median : 7.05   Median :16.80
 Mean   :20.29   Mean   :0.800   Mean   : 7.53   Mean   :15.64
 3rd Qu.:24.55   3rd Qu.:1.175   3rd Qu.: 9.10   3rd Qu.:19.62
 Max.   :38.70   Max.   :2.200   Max.   :16.90   Max.   :24.50

      FIN              SPS              TC              X
 Min.   : 0.000   Min.   : 0.00   Min.   :3.000   Min.   : 99.80
 1st Qu.: 3.300   1st Qu.:22.95   1st Qu.:5.800   1st Qu.: 99.90
 Median : 7.150   Median :27.00   Median :6.750   Median :100.00
 Mean   : 6.650   Mean   :26.99   Mean   :6.453   Mean   : 99.98
 3rd Qu.: 9.325   3rd Qu.:33.17   3rd Qu.:7.150   3rd Qu.:100.00
 Max.   :15.300   Max.   :41.60   Max.   :8.800   Max.   :100.10

   X.1             X.2              X.3
 Mode:logical   Mode:logical   Min.   :1989
 NA's:30        NA's:30        1st Qu.:1990
                               Median :1991
                               Mean   :1991
                               3rd Qu.:1992
                               Max.   :1995

> employ<-employ[,-c(12:15)]
> summary(employ)#no NAs
              Country       Group         AGR              MIN
 Albania           : 1   Eastern: 8   Min.   : 0.00   Min.   : 0.000
 Austria           : 1   EFTA   : 6   1st Qu.: 4.40   1st Qu.: 0.125
 Belgium           : 1   EU     :12   Median : 8.45   Median : 0.500
 Bulgaria          : 1   Other  : 4   Mean   :12.19   Mean   : 3.447
 Cyprus            : 1                3rd Qu.:14.93   3rd Qu.: 1.050
 Czech/Slovak Reps: 1                 Max.   :55.50   Max.   :37.300
 (Other)           :24
      MAN              PS              CON             SER
 Min.   : 0.00   Min.   :0.000   Min.   : 0.60   Min.   : 3.30
 1st Qu.:19.00   1st Qu.:0.275   1st Qu.: 6.40   1st Qu.:12.62
 Median :20.30   Median :0.800   Median : 7.05   Median :16.80
 Mean   :20.29   Mean   :0.800   Mean   : 7.53   Mean   :15.64
 3rd Qu.:24.55   3rd Qu.:1.175   3rd Qu.: 9.10   3rd Qu.:19.62
```

```
 Max.   :38.70   Max.   :2.200   Max.   :16.90   Max.   :24.50

      FIN              SPS              TC
 Min.   : 0.000   Min.   : 0.00   Min.   :3.000
 1st Qu.: 3.300   1st Qu.:22.95   1st Qu.:5.800
 Median : 7.150   Median :27.00   Median :6.750
 Mean   : 6.650   Mean   :26.99   Mean   :6.453
 3rd Qu.: 9.325   3rd Qu.:33.17   3rd Qu.:7.150
 Max.   :15.300   Max.   :41.60   Max.   :8.800

> employ2<-scale(employ[,-c(1:2)])
>
> #simplest clustering methods takes arbitrary nr of clusters
> #and divides data into groups
>
> m.d81<-kmeans(employ2,5)
> #if you want to see means for each predictor in each cluster
> aggregate(employ2,by=list(m.d81$cluster),FUN=mean)
  Group.1        AGR        MIN         MAN          PS         CON
1       1  1.7886983 -0.1180576 -0.06203656 -0.40264110 -0.65127847
2       2  3.5194346  1.7994379 -2.14519617 -1.28845153 -1.51111239
3       3  0.1514056  3.3447137 -2.14519617 -1.28845153 -0.04756528
4       4 -0.3950358 -0.3459011 -0.02555483  0.09663386  0.18660226
5       5  0.1002148 -0.2669454  1.41415166  0.54759190 -0.16464905
         SER        FIN        SPS          TC
1 -0.8307239 -1.2040092 -0.8581401 -1.34050340
2 -2.3907537  2.1697250 -3.0912894 -2.79992242
3 -0.7532069 -1.4673862 -0.2168254  1.13240106
4  0.5684581  0.4540118  0.3334455 -0.05945781
5 -1.1621091 -1.1814341 -0.2855377  0.88105667
> employ2<-data.frame(employ2,m.d81$cluster)
> names(employ2)[10]<-"group"
> head(employ2)
         AGR        MIN          MAN         PS         CON         SER
1 -0.7789667 -0.3662040  0.054281993  0.0000000 -0.4500407  0.2448245
2 -0.5352011 -0.3774834  0.011984336 -0.1610564 -0.4134521 -0.2202775
3 -0.5758287 -0.3549247 -0.009164492  0.1610564 -0.1573313  0.2060660
4 -0.7302136 -0.3098071  0.477258563  0.3221129  0.6842083  0.3029623
5  0.8136356 -0.3323659 -0.114908635  0.3221129 -0.2670973  0.4967548
6  0.1310918 -0.3210865 -0.051462149  0.6442258 -0.1573313  0.4192378
         FIN        SPS          TC group
1  0.5142123  1.1345162  0.28107329     4
2  0.6145464  1.0658039  0.44323096     4
3  0.8904652  0.6993384 -0.04324205     4
4  0.7399640  0.1610921 -0.69187272     4
5 -0.3386276 -0.8237840  0.36215213     4
6  0.4389617 -0.1710172 -0.52971505     4
> plot(employ2$AGR,employ2$MAN,type="n")
> text(employ2$AGR,employ2$MAN,labels=employ[,1],
+ cex=0.5,col=employ2$group)
> #try to play with interpretation of these results
>
```

```
>
> dogs<-read.delim2("dogs.txt",head=T,sep="\t")
> summary(dogs)
           Group        MBrth             MHgt            Mol1L
              :18   Min.   : 0.000   Min.   : 0.000   Min.   : 0.000
 Chinese wolf : 1   1st Qu.: 0.250   1st Qu.: 1.000   1st Qu.: 0.250
 Cuon         : 1   Median : 1.000   Median : 1.000   Median : 1.000
 Dingo        : 1   Mean   : 4.064   Mean   : 7.977   Mean   : 7.573
 Golden jackal: 1   3rd Qu.: 9.225   3rd Qu.:19.925   3rd Qu.:18.900
 Indian wolf  : 1   Max.   :13.500   Max.   :27.300   Max.   :26.800
 (Other)      : 2   NA's   : 3.000   NA's   : 3.000   NA's   : 3.000
    Mol1B            Mol13L          Mol14L            X
 Min.   : 0.000   Min.   : 0.00   Min.   : 0.00   Min.   : 0.0
 1st Qu.: 0.000   1st Qu.: 0.25   1st Qu.: 0.25   1st Qu.: 0.0
 Median : 1.000   Median : 1.00   Median : 1.00   Median : 1.0
 Mean   : 3.250   Mean   :11.62   Mean   :13.35   Mean   : 1.2
 3rd Qu.: 7.525   3rd Qu.:29.93   3rd Qu.:34.48   3rd Qu.: 1.0
 Max.   :10.600   Max.   :41.90   Max.   :48.10   Max.   : 9.0
 NA's   : 3.000   NA's   : 3.00   NA's   : 3.00   NA's   :10.0
     X.1              X.2             X.3           X.4
 Min.   : 0.0000   Min.   : 0.000   Min.   : 0.000   Mode:logical
 1st Qu.: 0.0000   1st Qu.: 1.000   1st Qu.: 0.000   NA's:25
 Median : 1.0000   Median : 1.000   Median : 1.000
 Mean   : 0.9333   Mean   : 1.467   Mean   : 1.067
 3rd Qu.: 1.0000   3rd Qu.: 1.000   3rd Qu.: 1.000
 Max.   : 7.0000   Max.   :11.000   Max.   : 8.000
 NA's   :10.0000   NA's   :10.000   NA's   :10.000
     X.5
 Min.   : 1
 1st Qu.: 1
 Median : 1
 Mean   : 1
 3rd Qu.: 1
 Max.   : 1
 NA's   :13
> dogs<-dogs[-(8:25),-(8:13)]
> m.d82<-kmeans(scale(dogs[,-c(1,8)]),3)
> dogs<-data.frame(dogs,group=m.d82$cluster)
> plot(dogs$MBrth,dogs$MHgt,type="n")
> text(dogs$MBrth,dogs$MHgt,labels=dogs$Group,
+ cex=0.7,col=dogs$group)
```

To illustrate that unfortunately clustering based on centroids is ambiguous we we'll analyse simple data on two variables that are a priori divided into groups. We'll see if we're able to reproduce this pattern of grouping.

```
> ###code block D9
>
> kmd <- read.table("kmeansdata.txt",head=T,sep="\t")
> head(kmd)
          x         y group
```

```
1  2.918896  8.587122     2
2 10.724510  8.194907     1
3  5.588091 10.382890     2
4  6.619314  5.399704     5
5  8.725792  4.253471     5
6  9.923255  3.216071     4
> m.d91 <- kmeans(kmd[,1:2],6)
> m.d92 <- kmeans(kmd[,1:2],4)
> par(mfrow=c(2,2))
> with(kmd,plot(x,y,pch=20))
> with(kmd,plot(x,y,pch=20,col=group))
> with(kmd,plot(x,y,pch=20,col=m.d91[[1]]))
> with(kmd,plot(x,y,pch=20,col=m.d92[[1]]))
> par(mfrow=c(1,1))
> #lets check accuracy of assignment
> table(m.d91[[1]],kmd$group)

     1  2  3  4  5  6
  1  0 24  0  0  3  0
  2  0  0 25  0  0  0
  3  0  1  0  1 26  0
  4  0  0  0 15  0  0
  5  0  0  0  0  0 25
  6 20  0  0  4  1  0
>
> #to conlcude - not so accurate!
```

Apart from these simple depictions we might also have some more sophisticated illustrations.

```
> ###code block D9a
>
> install.packages("cluster");install.packages("fpc");library(cluster)
> library(fpc)
> clusplot(kmd[,1:2],m.d92$cluster,color=T,shade=T,labels=2,lines=1)
> plotcluster(kmd[,1:2],m.d91$cluster)
```

Much more accurate are hierarchical methods. They not only group objects but also create whole hierarchy of grouping allowing to track increasing/decreasing similarity of objects and their groups. These methods are most similar to what is thought of as phylogenetic trees. We'll see how they perform using the same data sets.

```
> ###code block D10
>
> demploy <- dist(employ[,-c(1:2)],method="euclidean")
> ddogs <- dist(dogs[,-c(1,8)],method="euclidea")
> m.10a <- hclust(demploy,method="ward")
> plot(m.10a,labels=paste(employ[,1],employ[,2]),cex=0.7)
> #grouped <- cutree(m.10a,3)
> rect.hclust(m.10a, k=3, border="red")
>
```

```
> m.10b <- hclust(ddogs,method="ward")
> plot(m.10b,labels=dogs[,1],cex=0.7)
>
> #to test relationships within a tree use the following
> #remeber to transpose data as pvclust clusterscolumns not rows
> temploy<-as.data.frame(t(employ))
> temploy<-temploy[-c(1:2),]
> names(temploy)<- paste(employ[,1],employ[,2])
> install.packages("pvclust");library(pvclust)
> m.10aa <- pvclust(temploy,method.hclust="ward",method.dist="euclidean")
Bootstrap (r = 0.44)... Done.
Bootstrap (r = 0.56)... Done.
Bootstrap (r = 0.67)... Done.
Bootstrap (r = 0.78)... Done.
Bootstrap (r = 0.89)... Done.
Bootstrap (r = 1.0)... Done.
Bootstrap (r = 1.0)... Done.
Bootstrap (r = 1.11)... Done.
Bootstrap (r = 1.22)... Done.
Bootstrap (r = 1.33)... Done.
Warning message:
In a$p[] <- c(1, bp[r == 1]) :
  number of items to replace is not a multiple of replacement length
> plot(m.10aa,cex=0.7)
> pvrect(m.10aa,alpha=0.95)
>
> #finally you might want to use ML methods to choose most optimal solution
>
> install.packages("mclust");library(mclust)
> m.10c <- Mclust(employ[,-c(1:2)])
> plot(m.10c,employ[,-c(1:2)])#several very informative plots
Waiting to confirm page change...
Waiting to confirm page change...
Waiting to confirm page change...
Waiting to confirm page change...
```

Canonical correlation

OK. Let's move to something bigger. These two words were always associated with power and multidimensionality. Canonical correlation can be regarded as more advanced version of PCA. The latter looks at variance and maximizes it whereas the former looks at correlations and maximizes them. Where in PCA we get PCs, in CC we gat canonical correlations, in order of decreasing relationship strength. CCA can also be regarded as generalisation of multiple regression for the cases with many response variables.

We'll be analysing two datasets. The first one contains data on 16 colonies of the butterfly *Euphydrys editha*, comprising four environmental variables and six genetic parameters. We'll look for relationship between environmental variables and genetics. Before we start we'll have to change the data a little. Since genetic parameters are percentages of presence of six

phosphoglucose-isomerase genes we have to omit one gene – 1.30 - (to prevent these frequencies from summing to 100). Further, we'll combine percentages for 0.40 and 0.60 genes as their frequencies are low.

Second data-set contains information on soil-related and vegetation related variables in some Maya-related archaeological locations in Belize. The relationships we're looking for are between soil and vegetation.

```
> ###code block D11
>
> install.packages("CCA")
> library(CCA)
> install.packages("yacca")
> library(yacca)
>
> belize<-read.table("belize.txt",head=T,sep="\t")
> butter <- read.table("butter.txt",head=T,sep="\t")
>
> #first lets look at correlations
> summary(butter)
     Colony       altid          prcipit           maxT
 AF     : 1   Min.   :  380   Min.   :10.00   Min.   : 81.00
 CR     : 1   1st Qu.:  565   1st Qu.:19.75   1st Qu.: 98.00
 DP     : 1   Median :  869   Median :25.00   Median : 99.00
 GH     : 1   Mean   : 2102   Mean   :28.06   Mean   : 97.25
 GL     : 1   3rd Qu.: 2000   3rd Qu.:36.00   3rd Qu.:101.00
 IF     : 1   Max.   :10500   Max.   :58.00   Max.   :105.00
 (Other):10
      minT             g040            g060             g080
 Min.   :-12.00   Min.   : 0.00   Min.   : 0.000   Min.   : 1.00
 1st Qu.: 17.75   1st Qu.: 0.00   1st Qu.: 2.750   1st Qu.:12.50
 Median : 26.00   Median : 0.00   Median : 4.500   Median :18.00
 Mean   : 20.88   Mean   : 1.75   Mean   : 7.188   Mean   :18.56
 3rd Qu.: 27.00   3rd Qu.: 0.25   3rd Qu.: 7.500   3rd Qu.:23.50
 Max.   : 32.00   Max.   :14.00   Max.   :26.000   Max.   :40.00

      g100            g116            g130
 Min.   :25.00   Min.   : 0.00   Min.   : 0.000
 1st Qu.:36.75   1st Qu.:10.00   1st Qu.: 0.000
 Median :47.00   Median :17.00   Median : 3.000
 Mean   :51.19   Mean   :17.19   Mean   : 4.125
 3rd Qu.:59.25   3rd Qu.:27.00   3rd Qu.: 6.500
 Max.   :92.00   Max.   :35.00   Max.   :14.000

> envir<-as.data.frame(scale(butter[,2:5]))
> genet<-as.data.frame(scale(butter[,6:10]))
> genet[,1]<-genet[,1]+genet[,2]
> genet<-genet[,-2]
> names(genet)[1]<-"g040060"
>
> matcor(envir,genet)
$Xcor
```

```
             altid    prcipit       maxT       minT
altid    1.0000000  0.5674063 -0.8279572 -0.9359200
prcipit  0.5674063  1.0000000 -0.4786956 -0.7046199
maxT    -0.8279572 -0.4786956  1.0000000  0.7191035
minT    -0.9359200 -0.7046199  0.7191035  1.0000000

$Ycor
           g040060       g080       g100       g116
g040060  1.0000000  0.6401128 -0.5604365 -0.5725101
g080     0.6401128  1.0000000 -0.8234638 -0.1266726
g100    -0.5604365 -0.8234638  1.0000000 -0.2637612
g116    -0.5725101 -0.1266726 -0.2637612  1.0000000

$XYcor
             altid    prcipit       maxT       minT    g040060
altid    1.0000000  0.5674063 -0.8279572 -0.9359200 -0.2037111
prcipit  0.5674063  1.0000000 -0.4786956 -0.7046199 -0.4879830
maxT    -0.8279572 -0.4786956  1.0000000  0.7191035  0.2432135
minT    -0.9359200 -0.7046199  0.7191035  1.0000000  0.2426795
g040060 -0.2037111 -0.4879830  0.2432135  0.2426795  1.0000000
g080    -0.5728800 -0.5497990  0.5357866  0.5933225  0.6401128
g100     0.7268903  0.6990375 -0.7172780 -0.7590314 -0.5604365
g116    -0.4578018 -0.1380033  0.4383080  0.4122114 -0.5725101
              g080       g100       g116
altid   -0.5728800  0.7268903 -0.4578018
prcipit -0.5497990  0.6990375 -0.1380033
maxT     0.5357866 -0.7172780  0.4383080
minT     0.5933225 -0.7590314  0.4122114
g040060  0.6401128 -0.5604365 -0.5725101
g080     1.0000000 -0.8234638 -0.1266726
g100    -0.8234638  1.0000000 -0.2637612
g116    -0.1266726 -0.2637612  1.0000000

> #note high correlations among variables - this should  be avoided
> #and in general such data should not be qualified for CCA
> #we'll do CCA as an instructuive example
>
>
> m.d11a <- cca(envir,genet)
> m.d11a$xcoef
              CV 1        CV 2        CV 3       CV 4
altid   -0.1480997 -1.25319075  3.27195606 -2.0508141
prcipit -0.3432402 -1.46021241 -0.01621605 -0.4409097
maxT     0.4872846 -0.07857902  0.43274794 -1.8015295
minT     0.1744802 -2.44463710  2.84326319 -0.6517615
> m.d11a$ycoef
               CV 1       CV 2      CV 3       CV 4
g040060  0.30016891 1.17079158 -1.239480 -0.1927490
g080     0.39215037 0.04270905 -2.221736  1.5223304
g100    -0.09604674 1.55674499 -4.238341  0.8202052
g116     0.81846126 1.13726187 -3.100224 -0.3943251
> m.d11a$corr
```

```
      CV 1       CV 2       CV 3       CV 4
0.86513953 0.46847806 0.40093924 0.09911816
> 1-pchisq(m.d11a$chisq,m.d11a$df)#non significant variables
     CV 1      CV 2      CV 3      CV 4
0.2667287 0.8719932 0.7460977 0.7474734
> #U1 - contrast of temperature and precipitation
> #V1 - positive loadings for 0.40, 0.60, 0.80, 1.16 genes
> m.d11a$xstructcorr#U1 describes low precipit/altitude and high
temperatures
              CV 1        CV 2       CV 3       CV 4
altid   -0.9096067  0.27132042  0.2433912 -0.1994030
prcipit -0.7834761 -0.41112540 -0.3702618 -0.2829263
maxT     0.8996817 -0.09994043 -0.2239288 -0.3611656
minT     0.9053516 -0.29936248  0.1035906  0.2828242
> m.d11a$ystructcorr#V1 describes best high freqs of 40,60,80 and 116 genes
              CV 1        CV 2       CV 3       CV 4
g040060  0.4144094  0.75878260  0.3407777  0.3693026
g080     0.7376384  0.06017695  0.1329623  0.6592340
g100    -0.9588563 -0.04217106 -0.2531783 -0.1213130
g116     0.4631300 -0.56976425 -0.3341328 -0.5909607
> plot(m.d11a$canvarx[,1],m.d11a$canvary[,1],type="n",xlab="U1",ylab="V1")
> text(m.d11a$canvarx[,1],m.d11a$canvary[,1],labels=butter[,1],cex=.7)
> abline(h=0,v=0,lty=2)
>
> summary(belize)
      Case          limeEnr          meadowCa       coralLime
 Min.   :  1.0   Min.   :  0.00   Min.   : 0.00   Min.   :  0.000
 1st Qu.: 38.5   1st Qu.: 20.00   1st Qu.: 0.00   1st Qu.:  0.000
 Median : 76.0   Median : 50.00   Median : 0.00   Median :  0.000
 Mean   : 76.0   Mean   : 47.71   Mean   :11.32   Mean   :  9.854
 3rd Qu.:113.5   3rd Qu.: 75.00   3rd Qu.:20.00   3rd Qu.:  0.000
 Max.   :151.0   Max.   :100.00   Max.   :80.00   Max.   :100.000
 alluvialSaline      decid           marsh            palm
 Min.   : 0.00   Min.   :  0.00   Min.   :  0.00   Min.   : 0.000
 1st Qu.: 0.00   1st Qu.:  7.50   1st Qu.: 10.00   1st Qu.: 0.000
 Median :10.00   Median : 50.00   Median : 40.00   Median : 0.000
 Mean   :20.15   Mean   : 43.31   Mean   : 43.05   Mean   : 1.026
 3rd Qu.:40.00   3rd Qu.: 75.00   3rd Qu.: 70.00   3rd Qu.: 0.000
 Max.   :90.00   Max.   :100.00   Max.   :100.00   Max.   :50.000
     mixed
 Min.   : 0.000
 1st Qu.: 0.000
 Median : 0.000
 Mean   : 2.351
 3rd Qu.: 0.000
 Max.   :90.000
> soil<-as.data.frame(scale(belize[,2:5]))
> vege<-as.data.frame(scale(belize[,6:9]))
> #percentages do not add up to 100 so we dont have to remove any variables
> matcor(soil,vege)
$Xcor
                  limeEnr    meadowCa   coralLime alluvialSaline
```

```
limeEnr         1.0000000 -0.14326685 -0.40885658    -0.46922525
meadowCa       -0.1432668  1.00000000 -0.09588203    -0.09483207
coralLime      -0.4088566 -0.09588203  1.00000000    -0.23873341
alluvialSaline -0.4692253 -0.09483207 -0.23873341     1.00000000

$Ycor
            decid       marsh        palm       mixed
decid  1.00000000 -0.78605573 -0.05950747 -0.15400206
marsh -0.78605573  1.00000000 -0.06817932 -0.13657853
palm  -0.05950747 -0.06817932  1.00000000 -0.02351585
mixed -0.15400206 -0.13657853 -0.02351585  1.00000000

$XYcor
                   limeEnr    meadowCa   coralLime alluvialSaline
limeEnr         1.00000000 -0.14326685 -0.40885658    -0.46922525
meadowCa       -0.14326685  1.00000000 -0.09588203    -0.09483207
coralLime      -0.40885658 -0.09588203  1.00000000    -0.23873341
alluvialSaline -0.46922525 -0.09483207 -0.23873341     1.00000000
decid           0.37826377 -0.22909283  0.34875268    -0.39405458
marsh          -0.26932914  0.38308412 -0.22379544     0.34748408
palm           -0.02920585 -0.10448319 -0.01721720     0.20699455
mixed           0.14137377 -0.04942136 -0.07477096    -0.01281809
                     decid       marsh        palm       mixed
limeEnr         0.37826377 -0.26932914 -0.02920585  0.14137377
meadowCa       -0.22909283  0.38308412 -0.10448319 -0.04942136
coralLime       0.34875268 -0.22379544 -0.01721720 -0.07477096
alluvialSaline -0.39405458  0.34748408  0.20699455 -0.01281809
decid           1.00000000 -0.78605573 -0.05950747 -0.15400206
marsh          -0.78605573  1.00000000 -0.06817932 -0.13657853
palm           -0.05950747 -0.06817932  1.00000000 -0.02351585
mixed          -0.15400206 -0.13657853 -0.02351585  1.00000000

> m.d11b<-cca(soil,vege)
> m.d11b$xcoef
                     CV 1       CV 2       CV 3        CV 4
limeEnr        1.3255395 -0.4989341  0.3836752  0.43805739
meadowCa       0.2870778 -0.8799652 -0.5703078  0.01739138
coralLime      1.1212504 -0.2916985  0.1427833 -0.71860445
alluvialSaline 0.5561916 -0.9665043  0.8666097 -0.15160040
> m.d11b$ycoef
           CV 1       CV 2       CV 3        CV 4
decid 1.6768275 -0.6967539 -0.2208609 -0.12032174
marsh 1.0018424 -1.5000802 -0.3046702 -0.01098072
palm  0.2156676 -0.3172089  0.9165872 -0.26264203
mixed 0.5081984 -0.3062062  0.2004938  0.93469975
> m.d11b$corr
     CV 1      CV 2      CV 3      CV 4
0.7610692 0.5324149 0.2382218 0.1214940
> 1-pchisq(m.d11b$chisq,m.d11b$df)#significant variables
        CV 1         CV 2         CV 3         CV 4
0.000000e+00 1.942401e-09 3.060714e-02 1.413036e-01
> m.d11b$xstructcorr
```

```
                      CV 1       CV 2          CV 3        CV 4
limeEnr          0.56500103  0.1999069  0.0003682975  0.80050667
meadowCa        -0.06308064 -0.6888603 -0.7211485117  0.03791011
coralLime        0.41898770  0.2274037 -0.1662912582 -0.86318254
alluvialSaline -0.36068909 -0.5793048  0.7065760097 -0.18724236
> m.d11b$ystructcorr
             CV 1        CV 2        CV 3        CV 4
decid  0.79822609  0.54842546 -0.06679338 -0.24000680
marsh -0.40035051 -0.88894455 -0.22093666 -0.02615429
palm   0.03562821 -0.16627171  0.94578747 -0.27671359
mixed  0.10806174  0.01343353  0.25456389  0.96090552
> #U1 - presence of soils 1 and 3
> #V1 - deciduous forest
> #U2 - soils 2 and 4
> #V2 - marshes and absence of deciduous forests
> #U3 - soil 4 and absence of soil 2
> #V3 - presence of palms
> #U4 - presence of soil 3 and absence of soil 1
> #V4 - absence of mixed forest
> ccvars <- as.data.frame(cbind(m.d11b$canvarx,m.d11b$canvary))
> names(ccvars)[1:4]<-c("u1","u2","u3","u4")
> names(ccvars)[5:8]<-c("v1","v2","v3","v4")
> xyplot(v4+v3+v2+v1~u1+u2+u3+u4,data=ccvars,outer=T)
```

Short on discriminant function analysis

DFA is the last one of many clustering routines in R. It works slightly different than methods presented so far. Briefly, it attempts to form (as a linear or quadratic combination) so called discriminant functions of decreasing predictive power that should be able to assign objects to groups based on their measurements. In the first step we form these discriminant functions based on our data and known assignments of objects to groups. Then we hope that we will be able to use this DFs as tools to assign new objects to groups based on the same measurements.

In R most useful and simple DF routines are in the package MASS. We'll briefly consider their application, using example data on European countries.

```
> ###code block D12
>
> library(MASS)
>
> summary(employ)#we'll try to assign countries to the econimic groups
using employment data
              Country       Group          AGR              MIN
 Albania          : 1   Eastern: 8   Min.   : 0.00   Min.   : 0.000
 Austria          : 1   EFTA   : 6   1st Qu.: 4.40   1st Qu.: 0.125
 Belgium          : 1   EU     :12   Median : 8.45   Median : 0.500
 Bulgaria         : 1   Other  : 4   Mean   :12.19   Mean   : 3.447
 Cyprus           : 1                3rd Qu.:14.93   3rd Qu.: 1.050
 Czech/Slovak Reps: 1                Max.   :55.50   Max.   :37.300
 (Other)          :24
```

```
      MAN              PS              CON             SER        
 Min.   : 0.00   Min.   :0.000   Min.   : 0.60   Min.   : 3.30  
 1st Qu.:19.00   1st Qu.:0.275   1st Qu.: 6.40   1st Qu.:12.62  
 Median :20.30   Median :0.800   Median : 7.05   Median :16.80  
 Mean   :20.29   Mean   :0.800   Mean   : 7.53   Mean   :15.64  
 3rd Qu.:24.55   3rd Qu.:1.175   3rd Qu.: 9.10   3rd Qu.:19.62  
 Max.   :38.70   Max.   :2.200   Max.   :16.90   Max.   :24.50  

      FIN              SPS              TC        
 Min.   : 0.000   Min.   : 0.00   Min.   :3.000  
 1st Qu.: 3.300   1st Qu.:22.95   1st Qu.:5.800  
 Median : 7.150   Median :27.00   Median :6.750  
 Mean   : 6.650   Mean   :26.99   Mean   :6.453  
 3rd Qu.: 9.325   3rd Qu.:33.17   3rd Qu.:7.150  
 Max.   :15.300   Max.   :41.60   Max.   :8.800  

> m.d12a <- lda(Group~AGR+MIN+MAN+PS+CON+SER+FIN+SPS+TC,data=employ,CV=T)
> ct <- table(employ$Group,m.d12a$class)
> diag(prop.table(ct))#proportions of correct assignments for eachcategory
of Group
   Eastern       EFTA         EU      Other 
0.20000000 0.06666667 0.26666667 0.00000000 
> sum(diag(prop.table(ct)))#and total % of correct assignemnts - not so
good
[1] 0.5333333
>
> #advanced illustration of DF
> install.packages("klaR");library(klaR)
> dogsa <- read.delim2("dogsall.txt",sep="\t",head=T)
> dogsa<-dogsa[-c(78:88),]
> dogsa<-gdata::drop.levels(dogsa)
> partimat(Group~X1+X2+X3+X4,data=dogsa,method="lda")
> #alternatively we can use quadratic discriminant function
> partimat(Group~X1+X2+X3+X4,data=dogsa,method="qda")
```

Visualizing multivariate data

Our last journey through multivariate data will have some of artistic taste. We often encounter problem of showing our multivariate data. We have several options and we'll go through them briefly in this section.

```
> ###code block D13
>
> #pairwise plot
> #good but captures no multivariate influence of other traits
> pairs(dogs[,-c(1,8)])
> pairs(employ[,-c(1,2)][,1:5])
>
> #perspective plot
> install.packages("scatterplot3d")
> install.packages("Rcmdr")
```

```
> library(scatterplot3d);library(rgl);library(Rcmdr)
>
> scatterplot3d(dogs$MBrth,dogs$MHgt,dogs$Mol1L,type="h")
> plot3d(dogs$MBrth,dogs$MHgt,dogs$Mol1L,col="red",size=3)
> scatter3d(dogs$MBrth,dogs$MHgt,dogs$Mol1L)
>
> #Chernoff faces
> install.packages("TeachingDemos");library(TeachingDemos)

> faces2(scale(dogs[,-c(1,8)]),labels=dogs[,1])
> faces2(scale(employ[,-c(1,2)]),labels=employ[,1])
>
> #star plots
> stars(scale(dogs[,-c(1,8)]),labels=dogs[,1])
```

Regression trees

Among all modern multivariate techniques – regression trees are the least known, despite their great usability. What are regression trees? They can be seen as tree representations of multivariate (multiple) regressions. They cannot be used in estimation but are extremely helpful when developing multiple regression models. They can be especially useful when deciding whether to include – or not – interactions. In multivariate problems, where interactions can be generated in huge numbers, it seems essential to be able to arbitrarily decide whether to include – or not – interaction terms.

We will analyse here simple data-set describing the taste of different kinds of cheese based on some biochemical features. We will see how these features contribute to explaining variability in cheese taste and can we *a priori* see if there's potential for interactions.

```
> ###code block D14
>
> cheese <- read.table(file="cheese.txt", head=T, sep="\t")
> cheese <- cheese[,-1]
> pairs(cheese, panel=panel.smooth)
> #strong collinearity can be spotted
> #we'll ignore it but beware - it should be dealt with
>
> #at the beginning it'salways best to check for possible curvilinear
trends
> #we can then include them as polynomial terms in our models
> install.packages("mgcv");library(mgcv)
> par(mfrow=c(2,2))
> m.d14a <- gam(taste~s(Acetic)+s(H2S)+s(Lactic), data=cheese)
> plot(m.d14a)
> par(mfrow=c(2,2))
> m.d14b <- gam(AGR~s(SER)+s(FIN)+s(MIN),data=employ)
> plot(m.d14b)
> par(mfrow=c(1,1))
>
> #now we'll employ tree regression methods
```

```
> install.packages("tree");library(tree)
> m.d14c <- tree(taste~.,data=cheese)
> plot(m.d14c);text(m.d14c)
> #it seems that the only significant terms are H2S and Lactic Acid
> #equal lengths of all branches and forks indicates lack of any
interaction
>
> #lets verify this by fitting overparametrised model and simplifying it
using AIC
> m.d14d <- lm(taste ~ Acetic+Lactic+H2S+Acetic*Lactic*H2S-
Acetic:Lactic:H2S,
+ data = cheese)
> summary(m.d14d)

Call:
lm(formula = taste ~ Acetic + Lactic + H2S + Acetic * Lactic *
    H2S - Acetic:Lactic:H2S, data = cheese)

Residuals:
    Min      1Q  Median      3Q     Max
-19.675  -5.273  -1.011   5.785  25.072

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   104.5479   102.2068   1.023    0.317
Acetic        -30.7974    23.9574  -1.286    0.211
Lactic        -60.1029   116.5393  -0.516    0.611
H2S             6.1573    20.9537   0.294    0.772
Acetic:Lactic  19.0531    22.6367   0.842    0.409
Acetic:H2S      0.5734     3.5246   0.163    0.872
Lactic:H2S     -3.6493     4.4944  -0.812    0.425

Residual standard error: 10.33 on 23 degrees of freedom
Multiple R-squared: 0.6795,     Adjusted R-squared: 0.5959
F-statistic: 8.127 on 6 and 23 DF,  p-value: 8.718e-05

> m.d14e<-step(m.d14d)
Start:  AIC=146.15
taste ~ Acetic + Lactic + H2S + Acetic * Lactic * H2S - Acetic:Lactic:H2S

               Df Sum of Sq    RSS    AIC
- Acetic:H2S     1     2.826 2458.9 144.19
- Lactic:H2S     1    70.402 2526.4 145.00
- Acetic:Lactic  1    75.651 2531.7 145.06
<none>                       2456.0 146.15

Step:  AIC=144.19
taste ~ Acetic + Lactic + H2S + Acetic:Lactic + Lactic:H2S

               Df Sum of Sq    RSS    AIC
- Lactic:H2S     1    69.427 2528.3 143.02
<none>                       2458.9 144.19
```

```
- Acetic:Lactic  1   204.750 2663.6 144.59

Step:  AIC=143.02
taste ~ Acetic + Lactic + H2S + Acetic:Lactic

                Df Sum of Sq    RSS    AIC
- Acetic:Lactic  1    140.13 2668.4 142.64
<none>                       2528.3 143.02
- H2S            1    992.67 3521.0 150.96

Step:  AIC=142.64
taste ~ Acetic + Lactic + H2S

         Df Sum of Sq    RSS    AIC
- Acetic  1      0.55 2669.0 140.65
<none>                2668.4 142.64
- Lactic  1    533.32 3201.7 146.11
- H2S     1   1007.66 3676.1 150.25

Step:  AIC=140.65
taste ~ Lactic + H2S

         Df Sum of Sq    RSS    AIC
<none>                2669.0 140.65
- Lactic  1    617.18 3286.1 144.89
- H2S     1   1193.52 3862.5 149.74
> summary(m.d14e)

Call:
lm(formula = taste ~ Lactic + H2S, data = cheese)

Residuals:
    Min      1Q  Median      3Q     Max
-17.343  -6.530  -1.164   4.844  25.618

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -27.592      8.982  -3.072  0.00481 **
Lactic        19.887      7.959   2.499  0.01885 *
H2S            3.946      1.136   3.475  0.00174 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 9.942 on 27 degrees of freedom
Multiple R-squared: 0.6517,     Adjusted R-squared: 0.6259
F-statistic: 25.26 on 2 and 27 DF,  p-value: 6.551e-07
```

Great thing about regression trees is that they can be used to discover patterns allowing for successful taxonomical classification of objects. Here we will consider simple case of the genus *Epilobium* and see if – based on several morphological measurements, we are able to create simple classification criteria for these plants.

```
> ###code block D15
>
> epilo<- read.table("taxonomy.txt",head=T,sep="\t")
> pairs(epilo[,-1],col=epilo$Taxon)
> #several discontinuities are visible in several morphology traits
> #tree model finds unambigous separation easily
> m.d15a <- tree(Taxon~., epilo)
> plot(m.d15a);text(m.d15a)
> print(m.d15a)#raw key-like output from tree()
node), split, n, deviance, yval, (yprob)
      * denotes terminal node

1) root 120 332.70 I ( 0.2500 0.2500 0.2500 0.2500 )
  2) Sepal < 3.53232 90 197.80 I ( 0.3333 0.3333 0.3333 0.0000 )
    4) Leaf < 2.00426 60  83.18 I ( 0.5000 0.5000 0.0000 0.0000 )
      8) Petiole < 9.91246 30   0.00 II ( 0.0000 1.0000 0.0000 0.0000 ) *
      9) Petiole > 9.91246 30   0.00 I ( 1.0000 0.0000 0.0000 0.0000 ) *
    5) Leaf > 2.00426 30   0.00 III ( 0.0000 0.0000 1.0000 0.0000 ) *
  3) Sepal > 3.53232 30   0.00 IV ( 0.0000 0.0000 0.0000 1.0000 ) *
>
> #as sepal and leaf traits are the easiest separators we'll
> #depict our individuals onthe phase-plane of these two traits
> m.d15b<-tree(Taxon~Sepal+Leaf,epilo)
> partition.tree(m.d15b)
> attach(epilo)
> labels <-
ifelse(Taxon=="I","a",ifelse(Taxon=="II","b",ifelse(Taxon=="III","c","d")))
> text(Sepal,Leaf,labels)
> detach(epilo)
```

References

Hadfield J. 2010. Course Notes.
http://cran.r-project.org/web/packages/MCMCglmm/vignettes/CourseNotes.pdf

Hadfield J. MCMC methods for multi-response GLMM: The MCMCglmm R package. Journal of Statistical Software. 33: 1-22.

Gelman A. 2006. Prior distributions for variance parameters in hierarchical models. Bayesian Analysis. 1: 515-533.

Gianola D. Sorensen D. 2004. Likelihood, Bayesian and MCMC methods in quantitative genetics. Springer.

Gianola D., Sorensen D. 2004. Quantitative genetic models for describing simultaneous and recursive relationships between phenotypes. Genetics. 167: 1407-1424.

Crawley M.J. 2010. The R Book. Wiley-Blackwell.

Manly B.J.F. 2000. Multivariate statistical methods. A primer. Chapman & Hall.

Adams D.C. 2008. Phylogenetic meta-analysis. 62: 567-572.

Gelman A. et al. 2008. A weakly informative default prior distribution for logistic and other regression models. Annals of Applied Statistics. 2: 1360-1383.

Dongen Van S. 2006. Prior specification in Bayesian statistics: three cautionary tales. Journal of Theoretical Biology. 242:90-100.

See ?PACKAGE for libraries we've used to obtain exact citation information.