A recap of mixed models in SAS and R

Søren Højsgaard

mailto:sorenh@agrsci.dk

Biometry Research Unit

Danish Institute of Agricultural Sciences

September 22, 2004

Printed: September 22, 2004 File: MixedModels-RandSAS.tex

Contents

1 Introduction and Goal

- Recap essentials from mixed models course given to the Department of Animal Nutrition and Physiology in 2002 (which was based on the SAS statistical system)
- Introduce the R statistical system
- In particular, show how to do the same analyses in R and SAS

1.1 Mixed models in a general setting

In the course we discussed mixed models of the form

$$y = X\beta + Zu + e$$

where $X\beta$ are the systematic effects, u are the random effects and e denote the residuals.

In SAS:

- Random effects are specified using the `random` statement.
- To account for 1) correlated residuals and/or 2) variance heterogeneity one can use the `repeated` statement.

Here we focus on specification of the random effects, because that is the most important issue in most practical applications.

The special structures on the error terms can be discussed later – provided that public demand exists!

1.2 What is R?

- R is: An open source (free!) statistical package, available from www.R-project.org
- R is continuously under development by dedicated people all over the world – including people from the Biometry Research Unit at DIAS.

1.3 R and SAS– peaceful co–existence

There are many good reasons for starting to use R – not necessarily as a substitute for SAS, but as a supplement.

- Claim: The learning curve for SAS is quite steep – it is difficult to get started.
- Claim: The proportion of scientists at DIAS who know SAS today is smaller than 10 years ago
- Claim: An increasing number of scientists at DIAS tend to do their statistical analyses using Excel – which provides only very limited possibilities.

At DIAS, there is a place for R, SAS and Excel as statistical tools – yet it is of paramount importance to choose the appropriate package for the task at hand.

1.4 Looking at data

Thiele, Thorvald Nicolai, 1838–1910 (Famous Danish statistician):

> "Man skal tegne før man må regne"

SHD: (Not famous Danish statistician);

> "Things you can not see in informative plots are (usually) not worth looking for in a more formal statistical analysis".
>
> "You can never spend too much time looking at the data"

Claim: SAS is NOT particularly well suited for this – extremely important – part of a statistical analysis.

Claim: Many misleading (or even disaterous) statistical analyses are made with SAS because the exploratory step is often skipped and one jumps straight on to fitting models.

1.5 GUI or language based packages

Today many people are attracted by GUI (Graphical User Interface) packages with "point-and-click" abilities.

- Graphics and (some simple) statistical analyses can be made that way in Excel – and that is fine when that solves the problem at hand.
- Likewise, `SAS insight` provides such GUI facilities.

However, one very soon runs into the limitations of GUI systems when wanting to do more advanced or non–standard analyses.

In that case, the only way to proceed is to do some programming.

- `SAS` provides a programming language (data steps and procedure steps).
- Claim: What `SAS` provides is not very flexible
- `R` provides a programming language.
- Claim: The `R` programming language is very flexible, intuitive and easy to get started with

2 Split plot experiment

2.1 Bacterial inoculation treatment

Aim: Evaluate effect of three bacterial inoculation treatments applied to two grass cultivars on dry weight yield.

Layout of experiment:

- Four `blocks`, each block divided in half and cultivar A or B randomly assigned to each half (=whole–plot unit).
- Each whole-plot subdivided into 3 areas (=split–plots) and 3 inoculation treatments were applied randomly to these.

The classical model is

$$y_{bci} = \mu + R_b + \alpha_c + \beta_i + (\alpha\beta)_{ci} + W_{bc} + e_{bci}$$

where $R_b \sim N(0, \sigma_R^2)$, $W_{bc} \sim N(0, \sigma_W^2)$ and $e_{bci} \sim N(0, \sigma_e^2)$ and all error terms are independent.

Write this model briefly as

$$[y] = 1 + [block] + cult + inoc + cult * inoc + [block * cult] + [error]$$

Convention in model formulae used here:

- $[...]$ denotes random effects. (So the response is a sum of systematic and random effects and therefore we write $[y]$)
- Covariates will be underlined. (So with a time covariate, we write $\underline{Time}$).

2.2 In SAS and R

```
proc mixed data=inoc;
   class block cult inoc;
   model drywt = block cult inoc cult*inoc;
   random int cult /subject=block;
run;
```

Note: The `random` statement can be written more briefly as `random block block*cult`.

Note: One may choose to regard `block` as a fixed effect instead of a random effect – but that is not so important for now.

The `R` "equivalent" of `proc mixed` is the function `lme` which is in the library `nlme`. To make that function available, the library must be loaded into `R`:

```
> library(nlme)
```

Data, which is stored as a comma–separated file (.csv–file) is read into a `data.frame` which is the `R` equivalent of a `SAS` data set:

```
> inoc <- read.csv("Inoculation.csv")
```

To see the first few lines of the data frame we can do

```
> inoc[1:4, ]
```

```
  block cult inoc drywt
1     1    a  con  27.4
2     1    a  dea  29.7
3     1    a  liv  34.5
4     1    b  con  29.4
```

R will regard block as a numeric variable (because it consists only of numbers and no text). To change this we must declare it to be a factor:

```
> inoc$block <- as.factor(inoc$block)
```

Note: In SAS it is specified in the procedure whether a variable is a factor or a covariate, in R such an attribute is made part of the data by changing the data frame.

Now we are ready to fit the model:

```
> inocFit1 <- lme(drywt ~ cult + inoc + cult:inoc,
+     data = inoc, random = ~1 | block/cult)
```

The specifications in R and SAS are quite similar:

Model formula: The model formula in R

```
drywt ~ block + cult + inoc + cult:inoc
```

corresponds to in SAS

```
model drywt = block cult inoc cult*inoc;
```

Random effects: The specification of the random effects in R

```
random=~1 | block/cult
```

corresponds to in SAS

```
random int cult /subject=block;
```

Note: Many SAS users prefer the shorter form: `random block block*cult;`

2.3 Output from SAS and R

SAS and R differ quite substantially with respect to the output given.

The philosophy in SAS seems to be to provide the user with all sorts of information which he/she may possibly want. It is then the task of the user to extract the relevant bits and pieces.

The philosophy in R is to provide the user with what the user asks for – which is comforting.

Above we created what in R is called a model object which contains all the information one can possibly want about the model.

Simply typing the name gives

```
> inocFit1
```

```
Linear mixed-effects model fit by REML
  Data: inoc
  Log-restricted-likelihood: -32.53131
  Fixed: drywt ~ cult + inoc + cult:inoc
  (Intercept)         cultb       inocdea       inocliv
       27.900         0.125         1.350         5.250
cultb:inocdea cultb:inocliv
        1.275         0.250

Random effects:
 Formula: ~1 | block
        (Intercept)
StdDev:   0.9390806

 Formula: ~1 | cult %in% block
        (Intercept)  Residual
```

```
StdDev:   0.9045212 0.8397658

Number of Observations: 24
Number of Groups:
          block cult %in% block
              4               8
```

2.4 Model summary

A summary of the model is obtained by

```
> summary(inocFit1)
```

```
Linear mixed-effects model fit by REML
 Data: inoc
       AIC      BIC    logLik
  83.06262 91.07597 -32.53131

Random effects:
 Formula: ~1 | block
        (Intercept)
StdDev:   0.9390806

 Formula: ~1 | cult %in% block
        (Intercept)  Residual
StdDev:   0.9045212 0.8397658
```

```
Fixed effects: drywt ~ cult + inoc + cult:inoc
                Value Std.Error DF  t-value p-value
(Intercept)    27.900 0.7754414 12 35.97951  0.0000
cultb           0.125 0.8727443  3  0.14323  0.8952
inocdea         1.350 0.5938041 12  2.27348  0.0422
inocliv         5.250 0.5938041 12  8.84130  0.0000
cultb:inocdea   1.275 0.8397658 12  1.51828  0.1548
cultb:inocliv   0.250 0.8397658 12  0.29770  0.7710
 Correlation:
              (Intr) cultb  inocde inoclv cltb:ncd
cultb         -0.563
inocdea       -0.383  0.340
inocliv       -0.383  0.340  0.500
cultb:inocdea  0.271 -0.481 -0.707 -0.354
cultb:inocliv  0.271 -0.481 -0.354 -0.707  0.500

Standardized Within-Group Residuals:
        Min          Q1         Med          Q3         Max
-1.23352470 -0.47500060 -0.01607409  0.63490567  1.62294330
```

```
Number of Observations: 24
Number of Groups:
          block cult %in% block
              4                8
```

Note: It is easy to get only relevant parts out of the summary, but we will skip the details here.

2.5 Model diagnostics – plots

Model diagnostics is easily obtained, e.g. by plotting residuals etc (Figure 1 and Figure 2):

```
> plot(inocFit1, resid(.) ~ fitted(.) | block, abline = 0)
> plot(inocFit1, drywt ~ fitted(.) | cult, abline = c(0,
+     1))
```

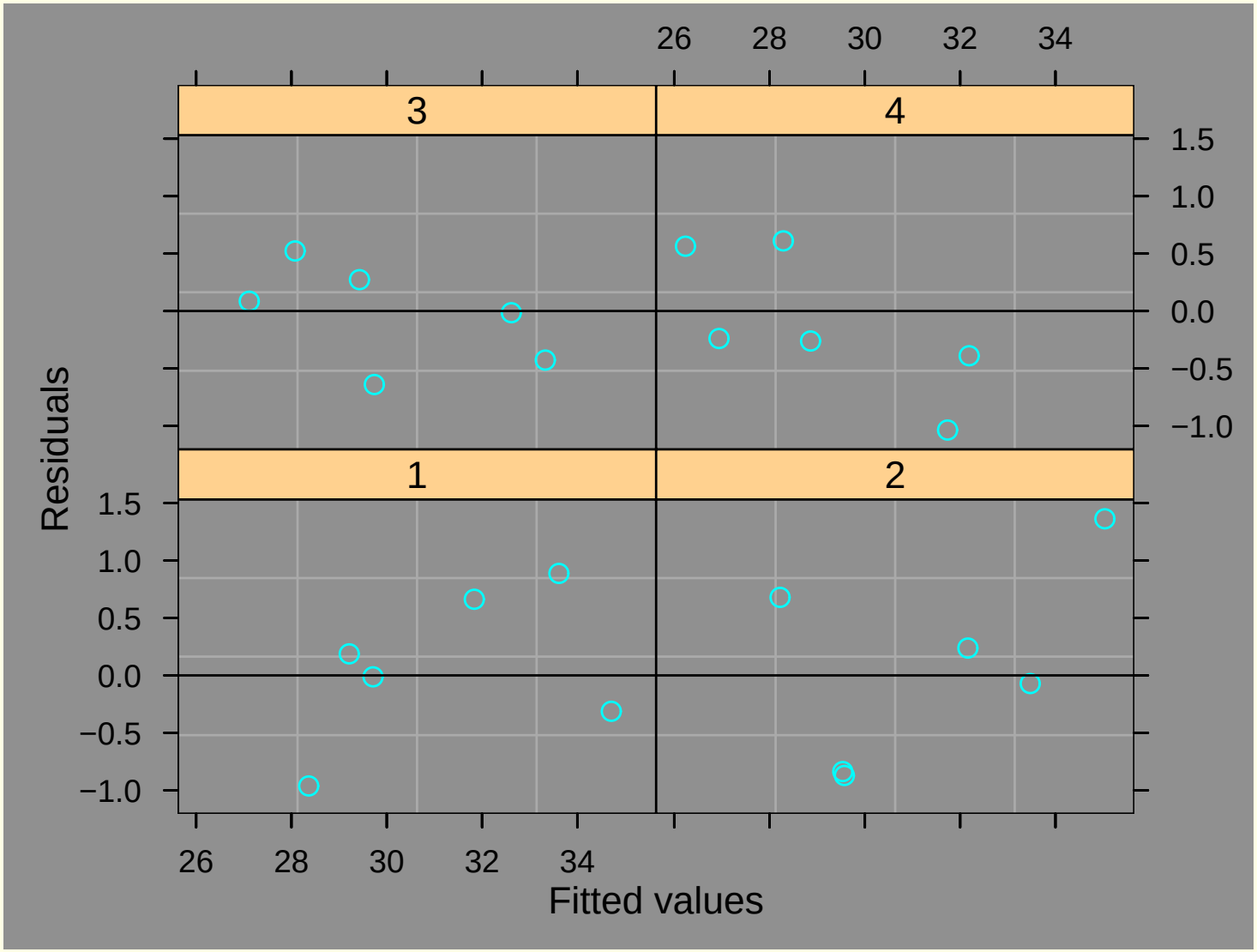

Figure 1:

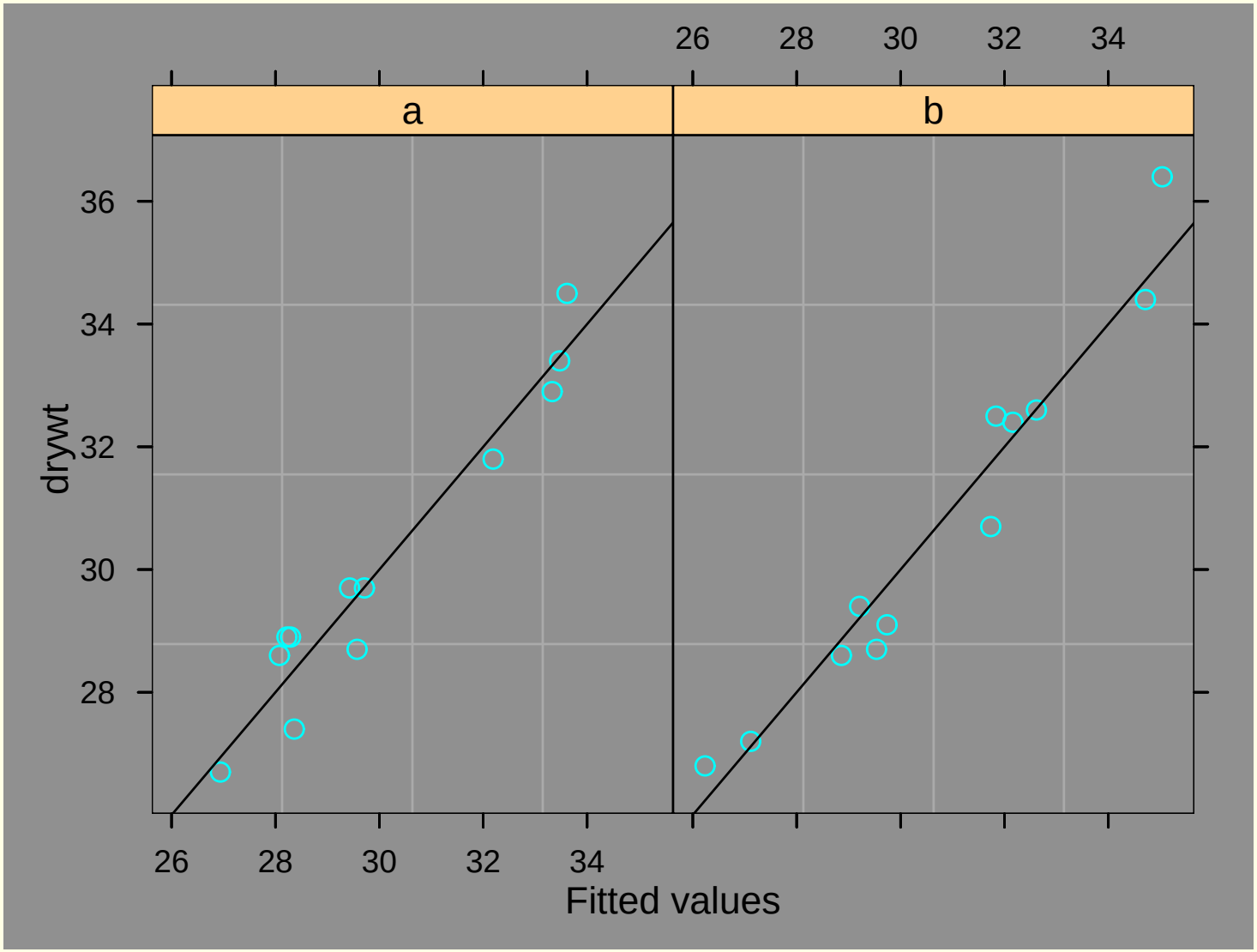

Figure 2:

3 Random Regression

3.1 Pig growth – `dietox`

- Does copper added to pig feed increase/decrease growth?
- Weight of slaughter pigs under treatment with Cu in their feed measured weekly over 12 week period.

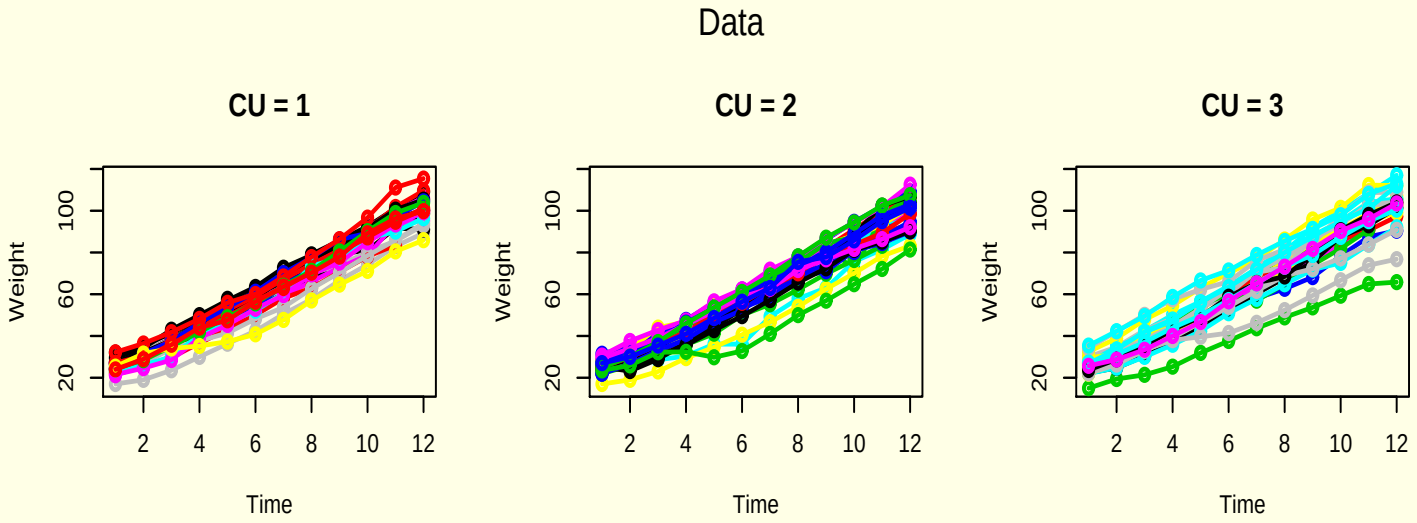

Figure 3:

3.2 Looking at data

Producing the plot in Figure 3 is difficult in SAS (if one wants the plots on the same piece of paper).

In R it is easy, e.g. by using facilities in the doBy package:

Load the doBy package and read the dietox data:

```
> library(doBy)
```

```
Loading required package: ash
```

```
> dietox <- read.csv("dietox.csv")
> dietox$Cu <- as.factor(dietox$Cu)
```

Make space for 1 row and 3 columns of plots:

```
> par(mfrow = c(1, 3))
```

Now invoke the plotBy function:

```
> plotBy(Weight ~ Time, subject = Pig, group = Cu,
+     title = "Cu=", data = dietox, col = 1:100, lines = T)
```

Plot suggests

- Approximately linear growth curves
- Some tendency for variance to increase with mean

A next step could be to calculate means for each combination of Cu and Time:

```
> m.dietox <- summaryBy(Weight ~ Cu + Time, data = dietox,
+     FUN = mean)
> m.dietox[1:5, ]
```

```
  Cu Time mean.Weight
1  1    1    25.34782
2  2    1    25.50000
3  3    1    26.14999
4  1    2    29.48695
5  2    2    29.33599
```

– and to plot these (see Figure 4).

```
> par(mfrow = c(1, 1))
> plotBy(mean.Weight ~ Time, subject = Cu, data = m.dietox,
+     lines = T, col = c("black", "red", "green"),
+     silent = F)
```

```
  symbol colour group subject line
1      1  black  .by.       1    1
2      1    red  .by.       2    1
3      1  green  .by.       3    1
```

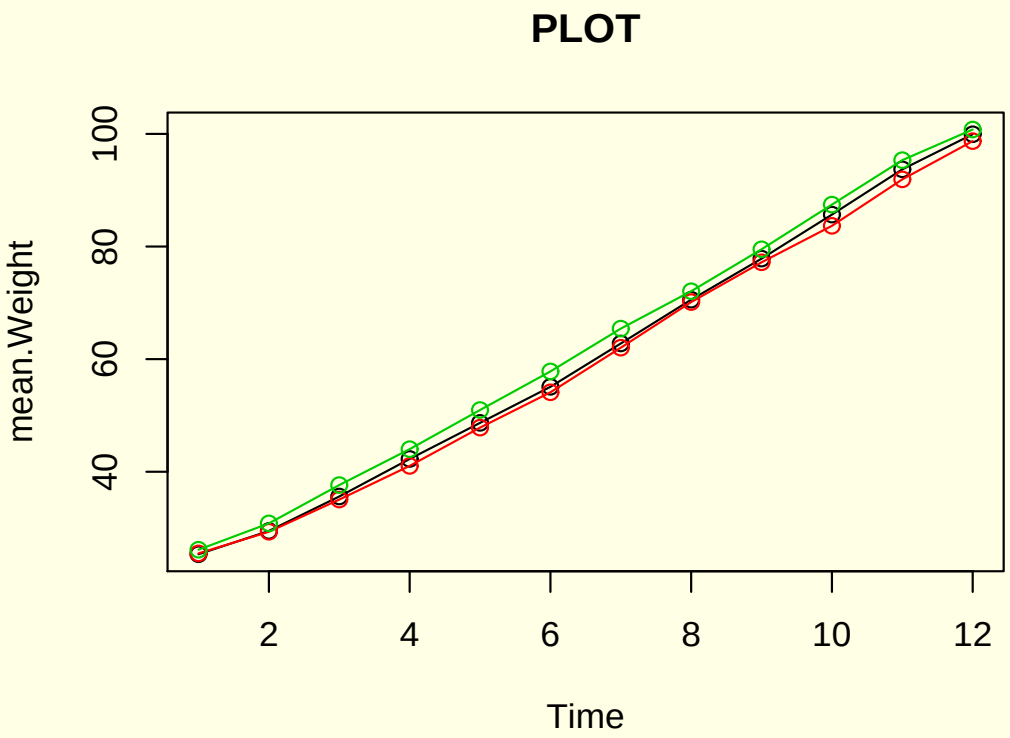

Figure 4:

Suggests that

- Growth is not quite linear. The curves are curved (slightly S–shaped)!
- If there is a treatment effect, then it is small!

3.3 Onto fitting some models

In the following we will

- Discuss various statistical models for these data
- Show (or rather recap) how to fit them in SAS
- Show how to fit them in R

3.3.1 Linear regression

Let c: Cu, p: pig (within treatment), t: time.

Simple regression model:

$$y_{cpt} = \alpha + \beta t + \alpha_c + \beta_c t + e_{cpt}; \quad e_{cpt} \sim N(0, \sigma^2)$$

Written shortly as:

$$[y] = 1 + \underline{time} + Cu + Cu * \underline{time} + [e]$$

3.3.2 In SAS and R

```
proc mixed data=dietox noinfo noclprint;
  class cu pig;
  model weight = time cu cu * time / solution htype=1;
run;
```

```
> fm0 <- lm(Weight ~ Time + Cu + Cu * Time, data = dietox)
```

To plot the residuals and the fitted values, one can do:

```
> par(mfrow = c(2, 3))
> plotBy(resid(fm0) ~ Time, subject = Pig, group = Cu,
+     data = dietox, lines = T, col = 1:100)
> plotBy(fitted(fm0) ~ Time, subject = Pig, group = Cu,
+     data = dietox, lines = T, col = 1:3)
```

3.3.3 Figures

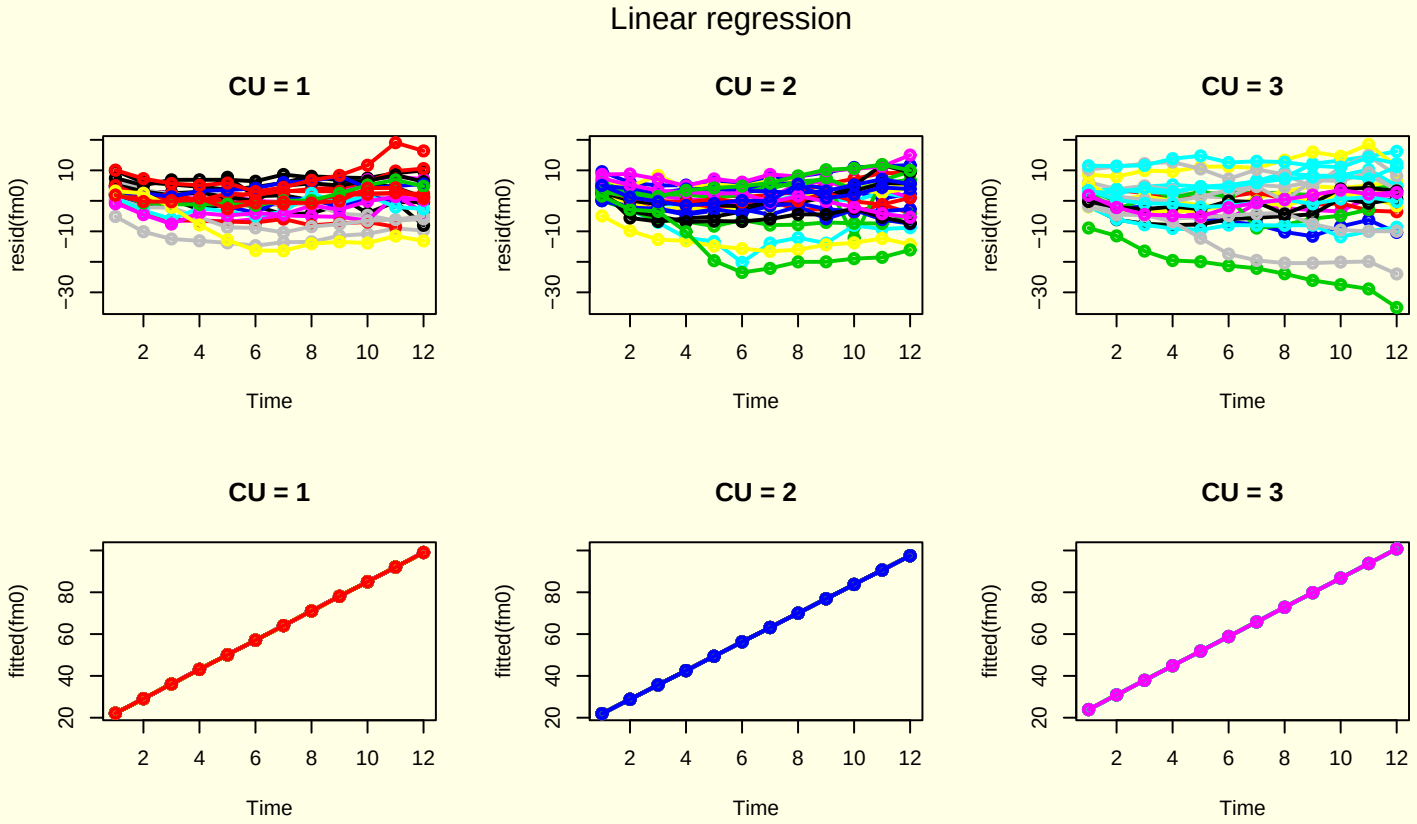

Figure 5:

If model is appropriate, residuals should fluctuate randomly around zero. That is clearly not so.

3.3.4 Adding a random intercept term

Pigs who start above (below) average tend to keep that position throughout the experiment.

This suggests to add a pig–specific (random) intercept term:

$$y_{cpt} = \alpha + \beta t + \alpha_c + \beta_c t + U_{cp} + e_{cpt};$$

Written shortly as:

$$[y] = 1 + \underline{time} + Cu + Cu * \underline{time} + [Cu * Pig] + [e]$$

3.3.5 In SAS and R

```
proc mixed data=dietox noinfo noclprint;
  class cu pig;
  model weight = time cu cu * time / solution htype=1;
  random int / subject=cu*pig;
run;
```

```
> fm1 <- lme(Weight ~ Time + Cu + Cu * Time, data = dietox,
+     random = ~1 | Pig)
```

3.3.6 Figures

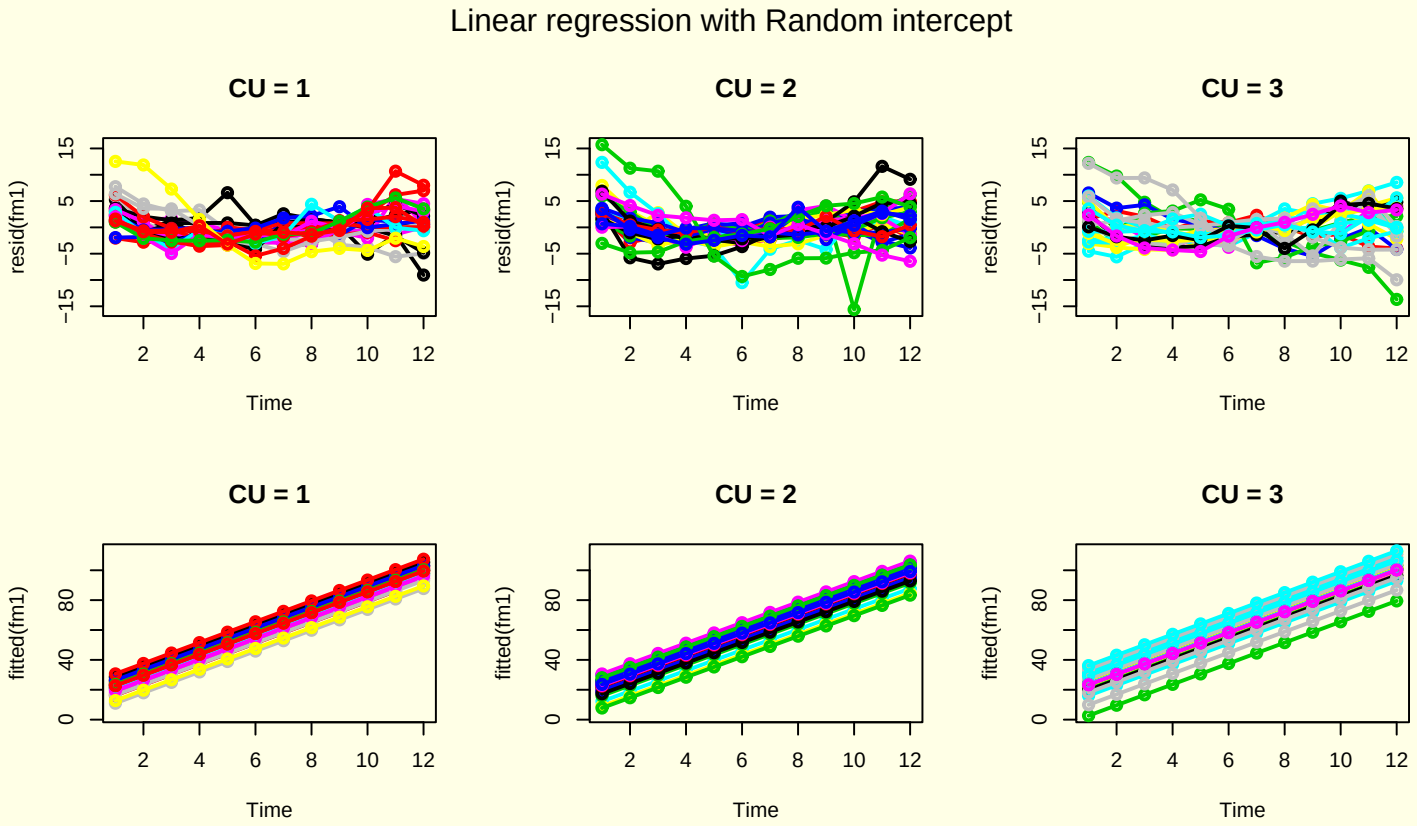

Figure 6:

Residuals still problematic: Residuals on some pigs steadily increasing, others steadily decreasing.

3.3.7 Adding a random slope term

To account for this phenomenon, one can add a pig–specific (random) slope term:

$$y_{cpt} = \alpha + \beta t + \alpha_c + \beta_c t + U_{cp} + W_{cp} t + e_{cpt};$$

Written shortly as:

$$[y] = 1 + \underline{time} + Cu + Cu * \underline{time} + [CU * Pig] + [CU * Pig] * \underline{time} + [e]$$

Such a model is called a random regression model.

3.3.8 In SAS and R

```
data dietox; set dietox;
  timec = time;

proc mixed data=dietox noinfo noclprint;
  class cu pig timec;
  model weight = time cu cu * time / solution htype=1;
  random int time/ subject=cu*pig;
run;
```

```
> fm2 <- lme(Weight ~ Cu * Time, data = dietox, random = ~1 +
+     Time | Pig)
```

3.3.9 Figures

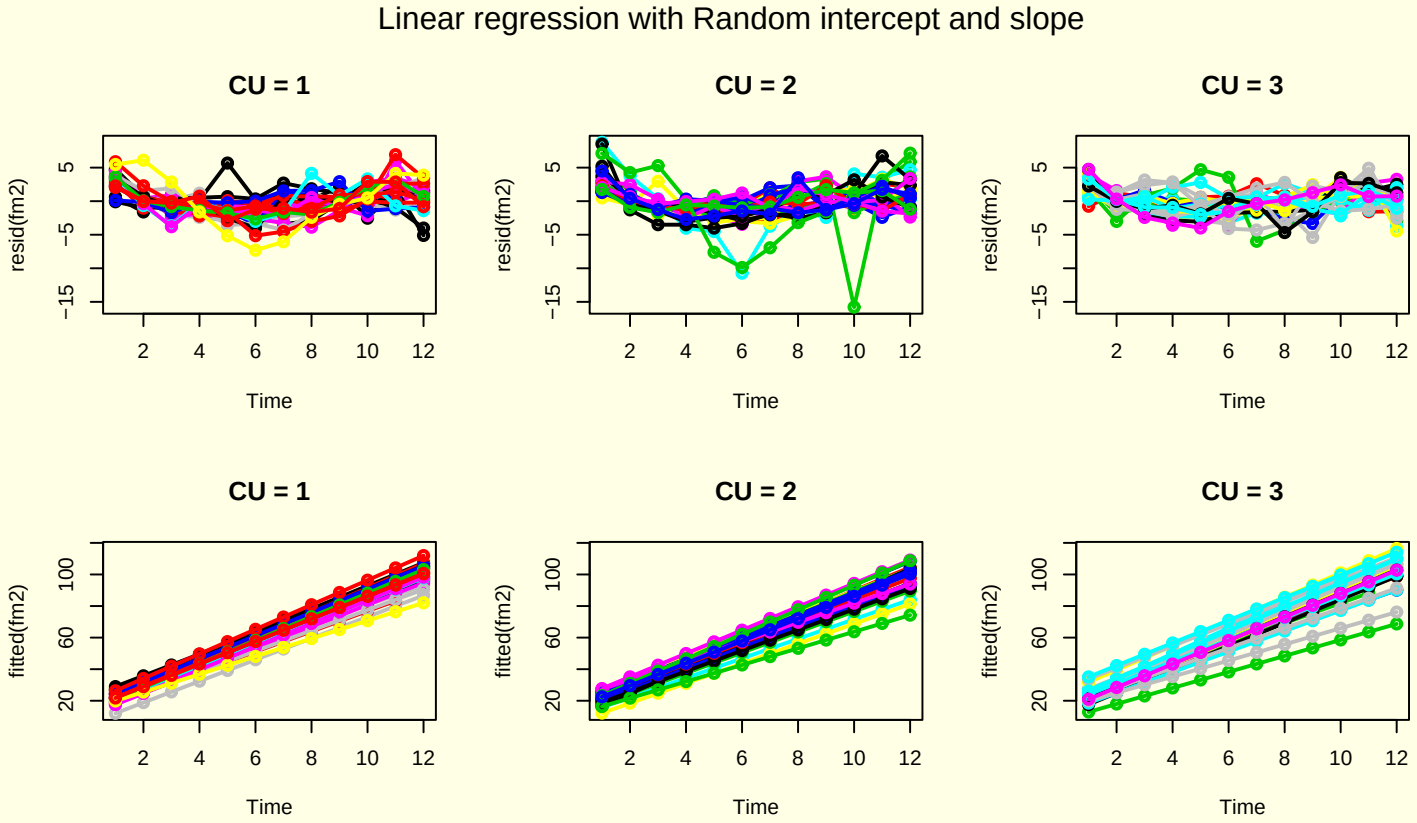

Figure 7:

3.3.10 Adding a quadratic and cubic term

There is still some curvature in the residuals. One solution to this is to add quadratic (and possibly cubic) terms to the model:

$$y_{cpt} = \alpha + \beta t + \gamma t^2 + \delta t^3 + \alpha_c + \beta_c t + \gamma_c t^2 + U_{cp} + W_{cp} t + e_{cpt};$$

Written shortly as:

$$\begin{aligned}[y] \quad = \quad & 1 + \underline{time} + \underline{time}^2 + \underline{time}^3 + Cu + Cu * \underline{time} + Cu * \underline{time}^2 + Cu * \underline{time}^3 \\ & + [Cu * Pig] + [Cu * Pig] * \underline{time} + [e]\end{aligned}$$

3.3.11 In SAS and R

```
proc mixed data=dietox noinfo noclprint;
  class cu pig timec;
  model weight = time time*time time*time*time cu
       cu*time cu*time*time cu*time*time*time /
       solution htype=1;
  random int time/ subject=pig;
run;
```

```
> fm3 <- lme(Weight ~ Cu * (Time + I(Time^2) + I(Time^3)),
+     data = dietox, random = ~1 + Time | Pig)
```

3.3.12 Figures

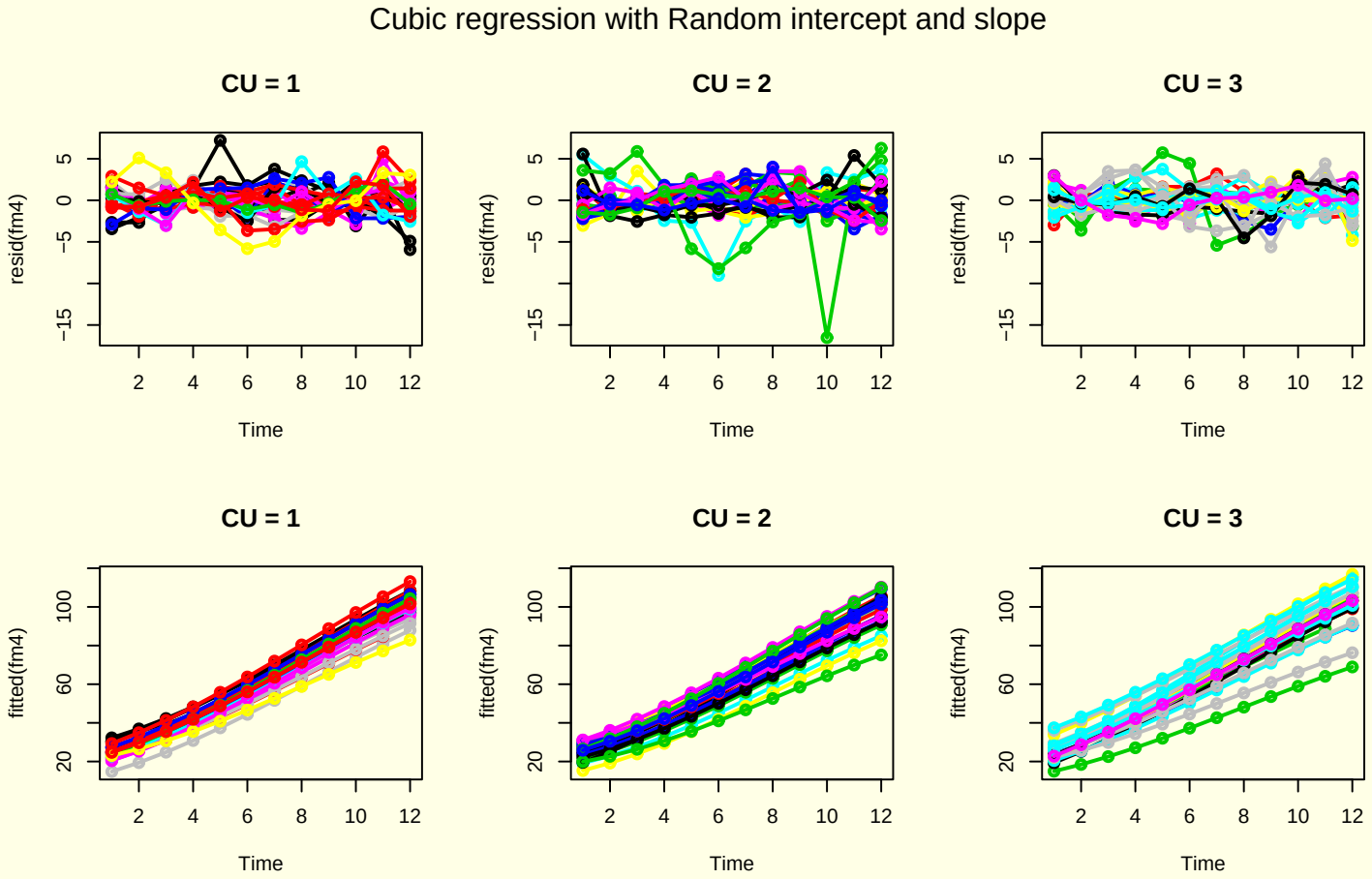

Figure 8:

These residuals look fairly reasonable!

3.3.13 The results

Stepwise addition of terms can be accomplished using the anova function:

```
> anova(fm3)
```

	numDF	denDF	F-value	p-value
(Intercept)	1	780	836.851	<.0001
Cu	2	69	1.236	0.2968
Time	1	780	7406.245	<.0001
I(Time^2)	1	780	228.253	<.0001
I(Time^3)	1	780	76.284	<.0001
Cu:Time	2	780	0.210	0.8107
Cu:I(Time^2)	2	780	6.547	0.0015
Cu:I(Time^3)	2	780	0.685	0.5044

There is an effect of Cu on the quadratic Time term, $Cu : I(Time^2)$.

3.4 Updating models

The term $Cu : I(Time^3)$ is not statistically significant, and can safely be eliminated from the model.

A very easy way to accomplish this is to update the model object as follows:

```
> fm32 <- update(fm3, . ~ . - Cu:I(Time^3))
> anova(fm32)
```

```
               numDF denDF  F-value p-value
(Intercept)        1   782  837.846  <.0001
Cu                 2    69    1.236  0.2968
Time               1   782 7403.378  <.0001
I(Time^2)          1   782  228.465  <.0001
I(Time^3)          1   782   76.356  <.0001
Cu:Time            2   782    0.210  0.8108
Cu:I(Time^2)       2   782    6.553  0.0015
```

3.5 Parameter estimates

```
> round(summary(fm32)$tTable, 3)
```

	Value	Std.Error	DF	t-value	p-value
(Intercept)	21.261	1.078	782	19.723	0.000
Cu2	0.178	1.432	69	0.125	0.901
Cu3	0.395	1.447	69	0.273	0.786
Time	3.447	0.316	782	10.916	0.000
I(Time^2)	0.496	0.045	782	10.929	0.000
I(Time^3)	-0.020	0.002	782	-8.743	0.000
Cu2:Time	-0.238	0.290	782	-0.823	0.411
Cu3:Time	0.587	0.293	782	2.003	0.046
Cu2:I(Time^2)	0.010	0.016	782	0.605	0.545
Cu3:I(Time^2)	-0.045	0.017	782	-2.730	0.006

3.6 Estimating contrasts

The estimated difference (gain) in going from Cu=1 (no copper) to Cu=3 (high level) at time t is then

$$Cu3 + Cu3 : Time * t + Cu3 : I(Time^2) * t^2$$

This is achieved by multiplying the regression coefficients by the vector

```
> time <- 2
> L <- c(0, 0, 1, 0, 0, 0, 0, time, 0, time^2)
> L
```

```
 [1] 0 0 1 0 0 0 0 2 0 4
```

and summing the results afterwards

```
> sum(fm32$coef$fixed * L)
```

```
[1] 1.387609
```

However, such an estimate is not of much interest unless one can obtain the standard error, which can be achieved using the `esticon` function:

```
> esticon(fm32, L)

                     beta0 Estimate Std.Error   t.value DF
(0 0 1 0 0 0 0 2 0 4)     0 1.387609  1.421196 0.9763673 69
                      Pr(>|t|)
(0 0 1 0 0 0 0 2 0 4) 0.3322932
```

It may be of interest to calculate this estimate for a variety of different t values and to plot these together with a confidence interval, Figure 9:

```
> e <- NULL
> for (time in 1:12) {
+     L <- c(0, 0, 1, 0, 0, 0, 0, time, 0, time^2)
+     e <- rbind(e, esticon(fm32, L))
+ }
> plot(1:12, e$Estimate, type = "l", ylim = c(-5, 7),
+     col = "red", lwd = 2)
> abline(h = 0)
> lines(1:12, e$Estimate + 2 * e$Std.Error, type = "l",
+     col = "green", lwd = 2)
> lines(1:12, e$Estimate - 2 * e$Std.Error, type = "l",
+     col = "green", lwd = 2)
```

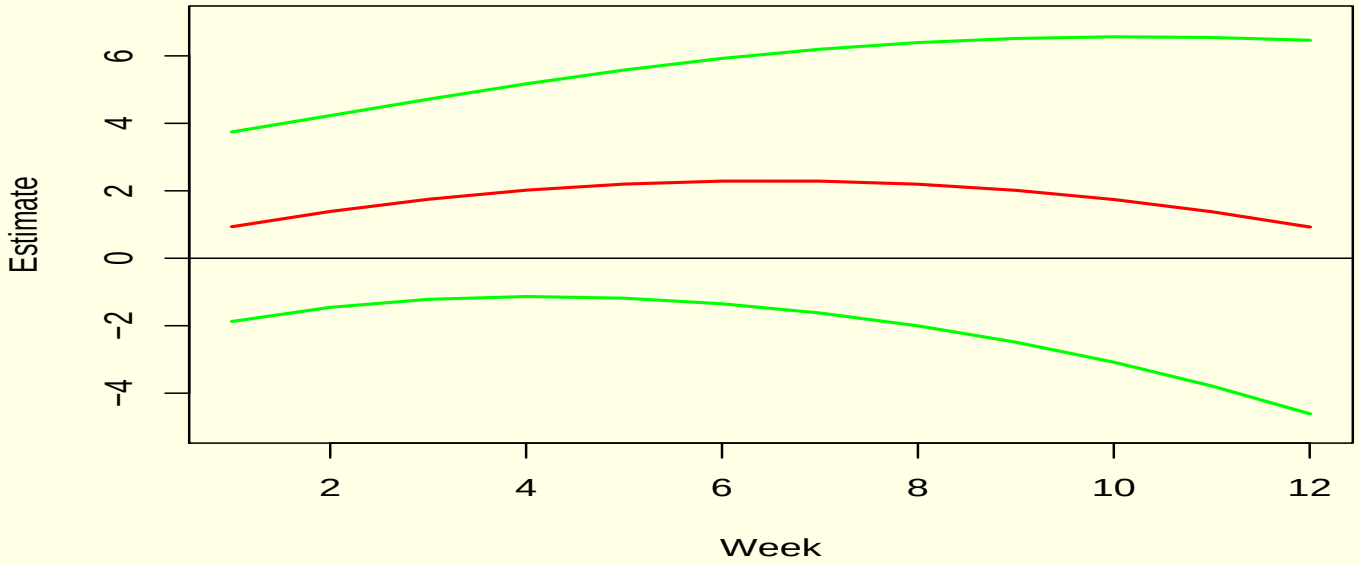

Figure 9:

Statistical significance and practical significance is two different things.

Whether this effect is of any practical relevance is a good question!

4 Final remarks on SAS and R

- R is not as good as SAS for manipulating large data sets (like 200 MB)
- In non–linear models (including generalized linear models) one sometimes encounters situations where R functions can not converge – and where the corresponding SAS function can.
- In many cases fitting mixed models in R is substantially faster than in SAS.
- It is difficult to obtain the celebrated (and much misused) LSMEANS in R.

5 Names of commonly used statistcal functions

R–function	SAS–procedure
`lm`	`proc glm`
`glm`	`proc genmod`
`lme`	`proc mixed`
`nlme`	`proc nlmixed`
`nls`	`proc nlin`

6 When I do statistics...

– I tend to use both SAS and R:

- I use SAS for manipulating data
- I use R for plotting and analyzing data