Linear Mixed-Effects Models

(aka Statistics III)

Bernd Figner
b.figner@psych.ru.nl

Week 5: March 10, 2014

1

Today: Digging Deeper

- Questions: General, homework-related
- Some homework-related add-ons and leftovers from last class
- *P* values; multiple cores; post-hocs; writing up model and results; some plotting; linear and quadratic predictors; centering/scaling to reduce multicollinearity
- Advice from Barr et al and others
  - The R-sig-mixed-models debate
  - Testing parameters vs. effects: Bootstrap vs. LRT?
- Homework

2

Question 1

lab computers:
R or RStudio?

3

Question 2

YOUR data:
who's willing to present?

March 24 ideal, but other dates possible

4

Homework

Questions? Problems? Comments?

5

Some Comments

In the valuation ratings analysis:

Item as a random intercept or not? Why (not)?

More theoretical answer

- They are NOT random sample from all possible items
- They are carefully chosen and created by the experimenter ("repeatable")

More pragmatic answer

- We want to test the significance of money and time
- Adding the same predictor as fixed slope and random intercept is typically not a good idea

6

More Comments

Time and Money: What should they be?

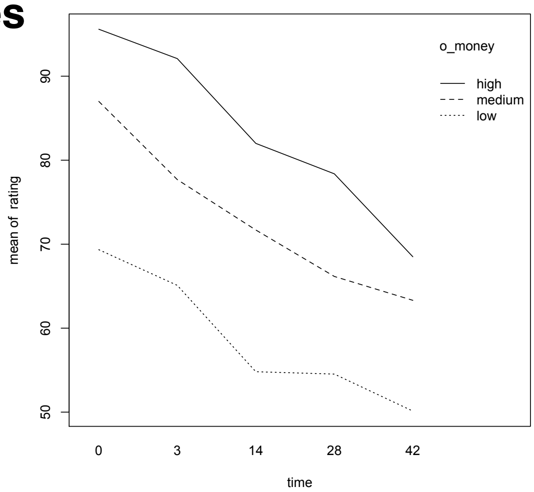
- Continuous?
- Unordered factors?
- Ordered factors?
- Linear?
- Linear and Quadratic?

7

Time

Occurring values

- 0 (= today)
- 3 days
- 14 days
- 28 days
- 42 days



8

As unordered Factor?

- Assumes that there is no inherent order among the levels

→ Seems odd, ignores a lot of information (Money: LOTS of levels, due to "jitter")

For most participants, there will be at least an ordinal relationship:

```
0 < 3 < 14 < 28 < 42
```

9

As ordered factor?

Recodes the actual values as numeric levels

- today = 1
- 3 days = 2
- 14 days = 3
- 28 days = 4
- 42 days = 5

→ polynomial contrasts then test for linear, quadratic (cubic, ...) effects

- More fine-grained information (days): thrown away
- Ordered levels treated as continuous, assuming **equal spacing** between levels (0 vs. 3 = 3 vs. 14??)
- Good: captures linear and higher-order effects: quadratic effects will probably capture unequal spacing

10

As continuous predictor?

Keeps the fine grained information and unequal spacing between occurring values:

- E.g., difference between 0 and 3 days is smaller than difference between 3 days and 14 days

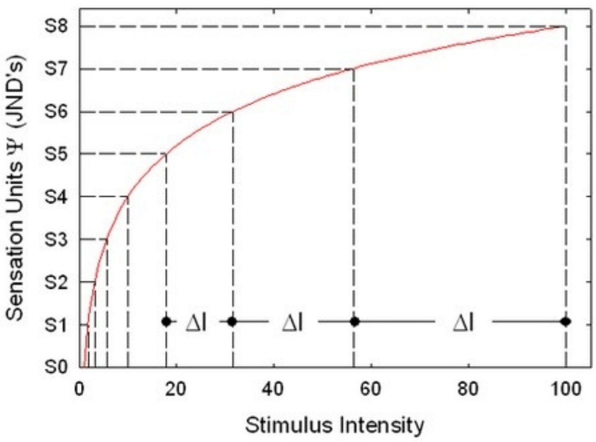
To capture non-linear effects

- You need to create a higher-order predictor yourself and add it to the model

11

Non-Linearity

- **Time: Sooner → more attractive**
- **More money → more attractive**
- **Psychophysics?** Weber-Fechner



12

Non-Linearity

- **Time: Sooner → more attractive**
- **More money → more attractive**
- **Psychophysics?** Weber-Fechner
  - Effects might not be strictly linear
    - Time: e.g., 0 days vs. 14 days larger than 14 days vs. 28 days?
    - Money: €20 vs €50 larger than €50 vs €80?

How to capture such possible quadratic effects with continuous predictors?

13

Linear and Quadratic IVs

For the Time IV (same for Money)

(1) Center the predictor (or scale)

```
v5$c_time <- v5$time - mean(v5$time)
```

(2) Compute squared (quadratic) predictor

```
v5$q_time <- v5$c_time ^ 2
```

Do NOT center again

(3) Include both in your model

(don't forget the random slopes for both the linear and quadratic term)

14

Centering (scaling) reduces collinearity

- Between linear and quadratic variants of the same continuous predictor
- Between main effects and interactions

Time example

```
v5$q_time_NonCentered <- v5$time ^ 2
```

Correlation: .964!!!

```
with(v5, rcor.test(cbind(time, q_time_NonCentered)))
                    time    q_time_NonCentered
time                *****   0.964
q_time_NonCentered <0.001   *****
```

15

Same with centered predictors

Correlation: .487

```
with(v5, rcor.test(cbind(c_time, q_time)))

        c_time q_time
c_time  *****  0.487
q_time <0.001  *****
```

Still not perfect, but much better

This is the standard approach

16

`poly()` creates uncorrelated linear, quadratic, cubic, ... terms

`poly(v5$time, 2)` → **linear and quadratic terms**

```
v5$poly_lin_Time <- poly(v5$time, 2)[,1]
v5$poly_quadr_Time <- poly(v5$time, 2)[,2]
with(v5, rcor.test(cbind(poly_lin_Time, poly_quadr_Time)))
```

```
                 poly_lin_Time poly_quadr_Time
poly_lin_Time     *****          -0.000
poly_quadr_Time  >0.999           *****
```

Best approach, as collinearity increases probability of non-convergence (and inflates standard errors, makes interpretation difficult, ...)

17

- Model with linear and quadratic time and money generated with `poly()`
  - Gives warnings, but produces a model that looks reasonable
- Model with linear and quadratic time and money generated by squaring the centered predictors
  - Needs many more iterations to give reasonably looking model (~ 10,000,000 iterations)
  - More serious warnings even with that many iterations (but results quite similar to poly model)

18

Model with `poly()`

```
v_m1c_poly_lin_quad <- lmer(rating ~
poly_lin_Time + poly_quadr_Time +
poly_lin_Money + poly_quadr_Money +
(1 + poly_lin_Time + poly_quadr_Time +
poly_lin_Money + poly_quadr_Money | pp_code),
data = v5, control = lmerControl(optCtrl =
list(maxfun = 10000)))
```

Linear and quadratic predictors for time and money

- In fixed-effects part
- In random part (random slopes)

19

```
summary(v_m1c_poly_lin_quad)
Fixed effects:
                 Estimate Std. Error t value
(Intercept)        71.754      2.013   35.65
poly_lin_Time    -176.615     25.538   -6.92
poly_quadr_Time    40.375     10.945    3.69
poly_lin_Money    219.521     25.261    8.69
poly_quadr_Money  -24.326     10.717   -2.27
```

Quadratic term: opposite sign of linear term

- Consistent with Weber-Fechner idea, but better check in plots

20

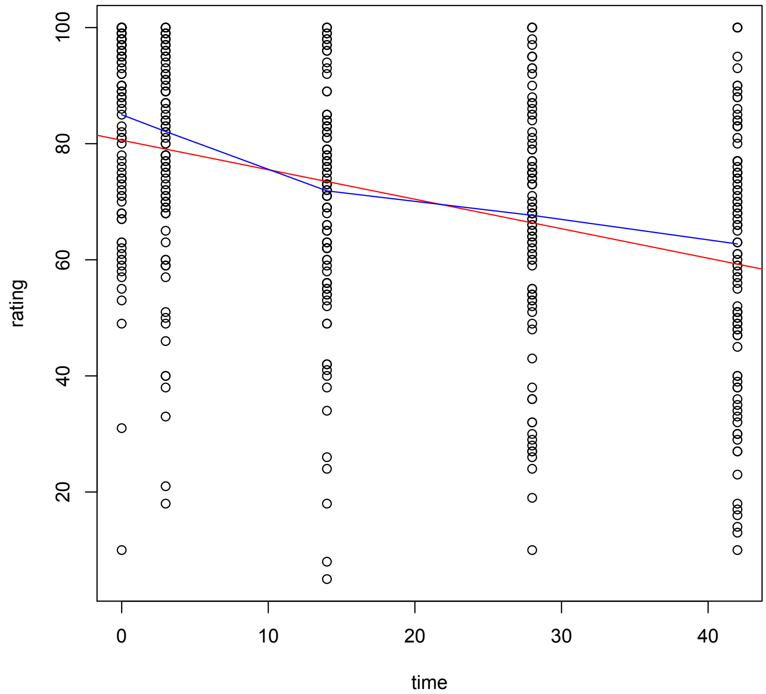
Idea: scatter plot with linear and smoothed (quadratic?) trend line

For time

```
with(v5, plot(time, rating))
with(v5, abline(lm(rating ~ time), col = 'red'))
with(v5, lines(lowess(time, rating), col =
'blue'))
```

- scatter plot with time on x axis and DV on y axis
- adds linear trend line in red
- adds smoothed trend line in blue

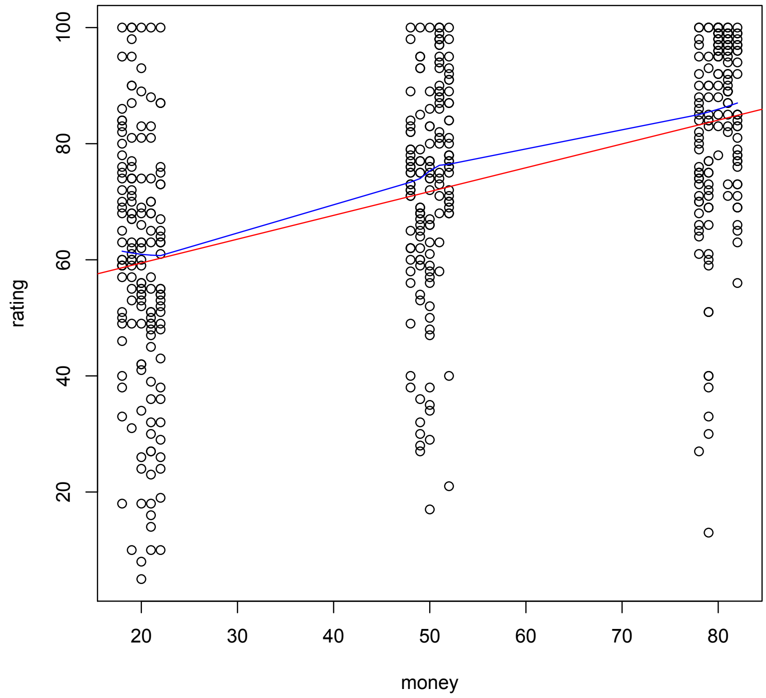
21



22

Same for money

```
with(v5, plot(money, rating))
with(v5, abline(lm(rating ~ money), col = 'red'))
with(v5, lines(lowess(money, rating), col =
'blue'))
```

Significance?

```
Anova(v_m1c_poly_lin_quad, type = 3, test = 'F')

                         F Df Df.res           Pr(>F)
(Intercept)      1270.9801  1     31        < 2.2e-16 ***
poly_lin_Time      47.8298  1     31 0.0000000934307 ***
poly_quadr_Time    13.6077  1     31       0.0008602 ***
poly_lin_Money     75.5162  1     31 0.0000000008191 ***
poly_quadr_Money    5.1521  1     31       0.0303224 *
```

25

Recap and Leftovers from last class

Significance tests

- **Typically only for fixed effects of interest**
- **Several different options**
- Tests of **coefficents** vs. test of **effects**
- **Coefficients** → like regression: coefficient significantly different from 0?
- **Effects** → like ANOVA: Is whole predictor significant?
  - factor with more than 2 levels
  - whole interaction term
- Post-hoc tests

27

Tests of Coefficients and „Effects“

- **Example: 3 experimental conditions**
  - neutral
  - positive mood induction
  - negative mood induction
- **Effects**: familiar from ANOVA framework
  - Is the **whole factor** significant? → „Effect of mood induction“
  - Similarly: interaction between categorical factors: Is **whole interaction** significant?

28

- **Coefficients**: familiar from regression framework
  - Is the coefficient significantly different from 0?
  - Continuous predictors (→ 1 df)
  - Interactions between continuous predictors (→ 1 df)
  - Factors with 2 levels (→ 1df)
  - **Factors with more levels** (mood induction example)
    - Neutral condition vs average of positive and negative mood significant?
    - Positive vs negative mood condition significant?
    - ...

    → **How you set up contrasts matters!**

29

Getting p values

→ Different methods (usually similar results)

(Douglas Bates: https://stat.ethz.ch/pipermail/r-help/2006-May/094765.html)

List ordered according to recommendations

http://glmm.wikidot.com/faq

30

Three "most recommended" ones

(1) (non)Parametric bootstrap

- coefficients: `bootMer()` → `boot.ci()`
- effects: `PBmodcomp()` (package pbkrtest)

(2) Conditional F-tests with df correction

- `Anova(..., test="F")` (package car)
- `KRmodcomp()` (package pbkrtest)

(3) Likelihood Ratio Tests

- `anova()` or `drop1()`

31

Conditional F test with df correction (Kenward-Roger)

```
library(car)
Anova(mymodel, type = 3, test = "F")
```

To get type 2 test: `type = 2`

`Anova()` calls `KRmodcomp` from `pbkrtest`

```
KRmodcomp(LargeModel, SmallModel)
```

`SmallModel`: same as `LargeModel`, but without the fixed effect of interest

Notes

- Typically fast; widely accepted
- NOT available for generalized mixed models

32

Valuation Rating Example

```
Anova(v_m1c, type = 3, test = 'F')

Analysis of Deviance Table (Type III Wald F tests
with Kenward-Roger df)

Response: rating
                    F Df Df.res           Pr(>F)
(Intercept) 1270.924  1     31        < 2.2e-16 ***
s_time        47.587  1     31 0.0000000980867 ***
s_money       75.760  1     31 0.0000000007902 ***
```

VERY detailed example R script

BlackBoard → Course Documents → Week 4 → Homework Week 4 Example Solution

33

Bootstrap methods: bootMer()

```
library(lme4)
```

(1) Run your lmer model (mymodel)

(2) Define a function (only once per R session)

```
FUN_bootMer <- function(fit) {
      return(fixef(fit))
}
```

(3) Run bootstrap (can take a while!)

```
boot_mymodel <- bootMer(mymodel, FUN_bootMer,
nsim = 1000, type = "parametric", .progress =
"txt", PBargs = list(style = 3))
```

→ **Look up ?bootMer**

→ **Possible to use several cores → much faster!**

34

Bootstrapping can take a long time!

→ Try it first out doing only a few iterations

→ use `Sys.time()` as a stop watch

```
t1 <- Sys.time()
boot_v_m1c <- bootMer(v_m1c, FUN_bootMer, nsim
= 3, type = "parametric")
t2 <- Sys.time()
t2 - t1
```

→ Tells you how long the 3 iterations took
→ Gives you a feel how long 1000 will take...
→ Then run the 1000 iterations (e.g., before going to bed)

35

Progress bar

```
boot_mymodel <- bootMer(mymodel, FUN_bootMer,
nsim = 1000, type = "parametric", .progress =
"txt", PBargs = list(style = 3))
```

- **Shows a simple progress bar plus percentage how many of the simulations R has already finished**
- **Not possible if you use more than 1 core**

36

Multiple Cores with bootMer()

- `detectCores()` from library(pbkrtest) → tells you the number of CPUs (cores) of your computer
- `bootMer()` can use more than 1 → MUCH faster often!

NOTES

- **Don't use ALL your cores for R!!!**
- **Leave at least 1 for other tasks**

```
boot_mymodel <- bootMer(mymodel, FUN_bootMer,
nsim = 1000, parallel = "multicore", ncpus =
3)
```

```
boot_mymodel <- bootMer(mymodel, FUN_bootMer,
nsim = 1000, parallel = "snow", ncpus = 3)
```

37

Get Confidence Intervals (for *p* values)

Have a look at `boot_mymodel`

```
head(as.data.frame(boot_mymodel))
```

Get CIs

Intercept

```
boot.ci(boot_mymodel, index = 1, conf =
0.95, type=c("norm", "basic", "perc"))
```

99% CI

```
boot.ci(boot_mymodel, index = 1, conf =
0.99, type=c("norm", "basic", "perc")
```

38

For second column **(= first coefficient after intercept)**

```
boot.ci(boot_mymodel, index = 2, conf = 0.95, type=c("norm", "basic", "perc"))
```

etc for 3rd column, 4th column, ...

When a CI does NOT include 0 → significant!

- 95% CI does NOT include 0 → $p < .05$
- 99% CI does NOT include 0 → $p < .01$
- 95% CI DOES include 0 → *ns*, $p > .05$
- 90% CI DOES include 0 → *ns*, $p > .10$

39

Bootstrap method to test EFFECTS

- `PBmodcomp(LargeModel, SmallModel)` from `pbkrtest` → test of **effects**

Same idea as KRmodcomp

- First, fit model with and without effect of interest
- Then, compare the fit of the 2 models via `PBmodcomp()`

```
PBmodcomp(LargeModel, SmallModel)
```

Notes

- Can take quite some time
- → possible to distribute across several clusters; much faster (check ?PBmodcomp)

40

Multiple Cores with PBmodcomp()

```
n_cores <- detectCores()
```

NOTE: Don't use all cores; leave at least 1 (i.e., n_cores - 1)

Create the clusters

```
clusters <- makeCluster(rep("localhost",
n_cores - 1))
```

Run your model comparison

```
PB_1 <- PBmodcomp(mylarge_m, mysmall_m, nsim = 1000, cl
= clusters)
```

Look at the result

```
PB_1
```

Stop the cluster

```
StopCluster(clusters)
```

41

LRTs

Again model comparison approach

- First, fit model with and without effect of interest
- Then, compare the fit of the 2 models via anova()

```
anova(SmallModel, LargeModel)
```

Notes

- **Models need to be nested!!**
- LRTs often NOT recommended (for smaller data sets)
- BUT: Barr et al. (2013) → worked very well!
- Recent discussion on R-sig-mixed-models:
- LRTs does not rely on a precise estimate of the coefficients, thus may be better than other methods when model is "overparametrized"

42

- LRTs: models must be fit with ML, not REML

→ Fit lmer models with `REML = FALSE`

BUT: R likes you!

→ R refits the models in the `anova()` command using ML

→ Then does model comparison with the ML models

43

drop1()

→ If you need to compare many models

```
drop1_mymodel_Chisq <- drop1(mymodel,
~., test = "Chisq")
```

→ Does all the relevant `anova()` model comparisons (i.e., LRTs) for you.

44

Post-hoc tests

- **If you have a factor with more than 2 levels**
- **Age Group**
  - Children
  - Adolescents
  - Adults
- **You know how to test whether Age Group as a whole is significant**
- **How to do pairwise post-hoc comparisons?**
  - Children vs. Adolescents
  - Children vs. Adults
  - Adolescents vs. Adults

45

```
myAgeModel <- lmer(DV ~ f_Agegroup + IV + (1 +
IV | f_pp), data = mydata)
```

Simple way to do post-hoc tests for f_Agegroup

```
library(lsmeans)
posthoc_Agegroup <- lsmeans(myAgeModel,
list(pairwise ~ f_Agegroup))
posthoc_Agegroup
```

Same, but `glht()` from `multcomp`

```
library(multcomp)
posthoc_Agegroup <- glht(myAgeModel, linfct =
mcp(f_Agegroup = "Tukey"))
summary(posthoc_Agegroup)
```

46

An output example (from one of my studies)

```
$`f_Agegroup lsmeans`
f_Agegroup   lsmean       SE       df lower.CL upper.CL
     child 59.42722 1.611361 193.7102 56.24916 62.60529
adolescent 65.61498 1.413526 198.4173 62.82752 68.40245
     adult 66.37734 1.613881 194.8105 63.19442 69.56027

$`f_Agegroup pairwise differences`
                     estimate       SE       df  t.ratio p.value
child - adolescent -6.1877614 2.143488 195.7366 -2.88677 0.01200
child - adult      -6.9501220 2.280591 194.2601 -3.04751 0.00738
adolescent - adult -0.7623606 2.145383 196.3642 -0.35535 0.93279
    p values are adjusted using the tukey method for 3 means
```

47

Post-hocs for factor interactions

```
mymodel <- lmer(DV ~ f_1 * f_2 + ....)
```

(1) Create one single new factor that combines the two predictors IV1 and IV2

```
f_1 <- c('a', 'a', 'b', 'b')
f_2 <- c('d', 'e', 'e', 'd')
int_f_12 <- interaction(f_1, f_2)
[1] a.d a.e b.e b.d
Levels: a.d b.d a.e b.e
```

same: `paste(f_1, f_2, sep = '_')`

48

(2) Run the your model again, but use the newly created factor instead of `f_1 * f_2`

```
mymodel_2 <- lmer(DV ~ int_f_12 + ....)
```

(3) Use `lsmeans()` or `glht()` to do all pairwise post-hoc comparisons

```
lsmeans(mymodel_2, list(pairwise ~
int_f_12))
```

```
glht(mymodel_2, linfct = mcp(int_f_12 =
"Tukey"))
```

49

How to write up a model and the results

Mixed-models are relatively new methods

(e.g., „mixed-model“ vs „multilevel“)

- No established standards how to write it up (no APA guidelines for mixed-models)
- Some examples: BlackBoard → Course Documents → Examples of papers using lme4
- Example how I tend to write up my results (made-up) → It‘s also on BlackBoard

NOTE: cite R and and packages you use!

`citation()` → Citation info for R
`citation("lme4")` → Citation info for lme4
`citation("car")` → Citation info for car
...

51

Model Setup

The valuation rating data were analyzed with a linear mixed-effects model approach, using the lmer function of the lme4 package (version 1.1.-4; Bates, Maechler, Bolker, & Walker, 2014) in R (R Core Team, 2013). The model included a fixed intercept and each a fixed slope for the continuous predictors Time and Money. These continuous predictors were scaled before we entered them into the model (i.e., they had a mean of 0 and a standard deviation of 1).

52

cont.

We followed Barr, Levy, Scheepers, and Tily's (2013) advice to use a maximal random-effects structure: The repeated-measures nature of the data was accordingly modeled by including a per-participant random adjustment to the fixed intercept ("random intercept"), as well as per-participant random adjustments to the Time and Money slopes ("random slopes"); in addition, we included all possible random correlation terms among the random effects.

53

P Values

P values were determined using conditional F tests with Kenward-Roger correction of degrees-of-freedom, as implemented in the Anova function (with Type III F tests) from the package car (version 2.0.19; Fox & Sanford, 2011; this function calls the KRmodcomp function from the package pbkrtest: Halekoh & Højsgaard, 2013).

54

OR: *P* values were determined using Likelihood Ratio Tests as implemented in the anova function of the R base package.

OR: *P* values were determined using parametric bootstrapping as implemented in lme4's bootMer function, with 1000 simulations and deriving confidence intervals using the function boot.ci of the package boot (version 3.1.9.; Canty & Ripley, 2013; Davison & Hinkley, 1997.

OR: ... whatever you used!

55

Results

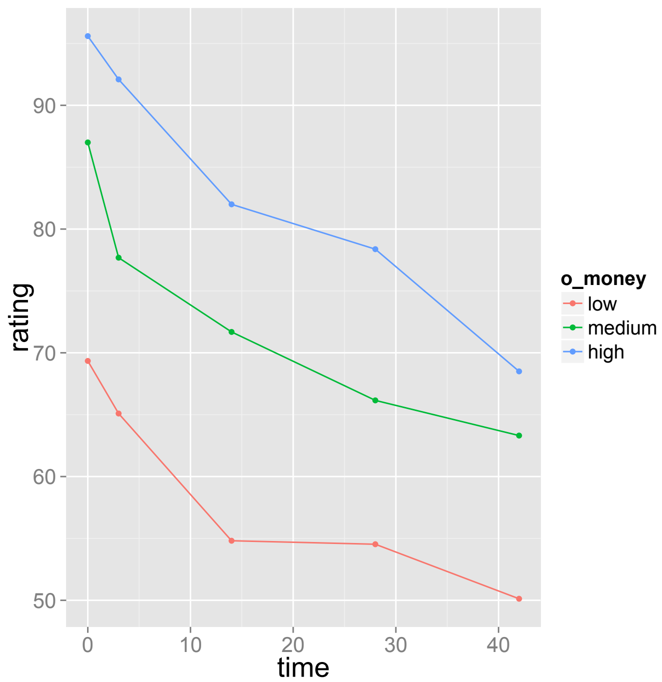
As expected, we found significant effects of Time (coef = -7.965; $p < .001$) and Money (coef = 10.038; $p < .001$): As time-of-delivery increased, the valuation ratings decreased and as Money increased, the valuation ratings increased. Figure 1 shows the effects of Time and Money on the valuation ratings. ETC ETC...

56

Figures: make them pretty...

```
ggplot(data = v5, aes(x = time, y = rating,
colour = o_money, group = o_money)) +
stat_summary(fun.y = mean, geom = "point") +
stat_summary(fun.y = mean, geom = "line") +
theme(axis.text.x = element_text(size = 15),
axis.title.x = element_text(size = 20),
axis.text.y = element_text(size = 15),
axis.title.y = element_text(size = 20),
legend.title = element_text(size = 14),
legend.text = element_text(size = 14))
```

57



58

Make them prettier...

- While this is better than my usual ugly ones, there's room for further improvement:
- Categories on the x axis
- Title of the legend (o_money) and its position
- ...

Error bars?

- Long and complicated story (mixed designs!)
- I don't add error bars usually
- Recent advice → if you want to show uncertainty, plot the coefficients and **their** error bars

59

xyplot() for individual participant patterns

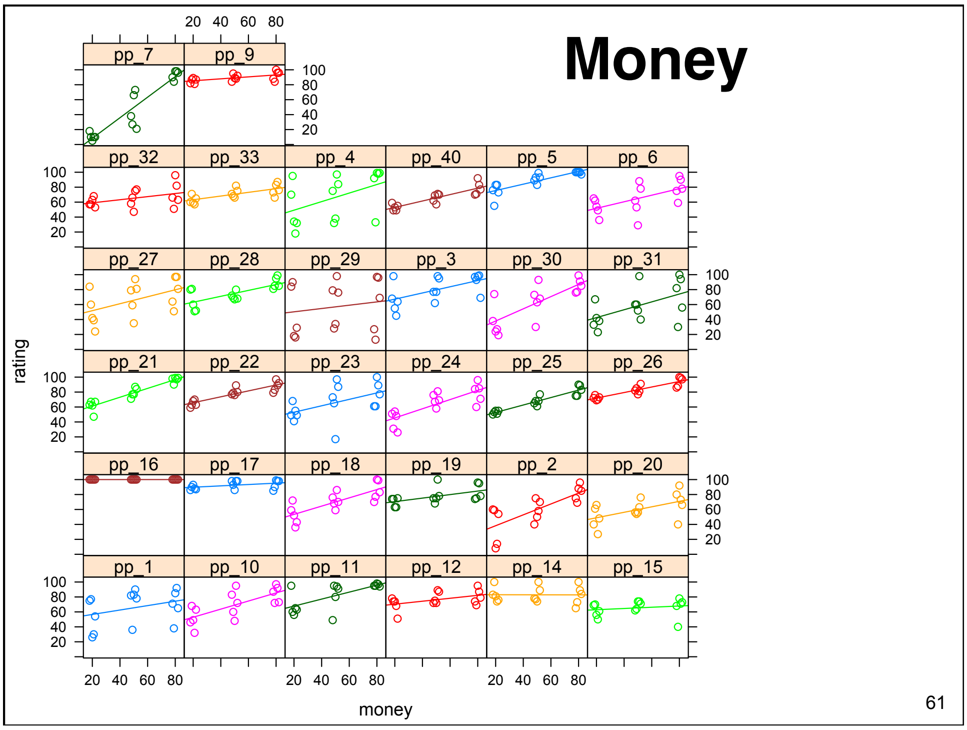
For Money

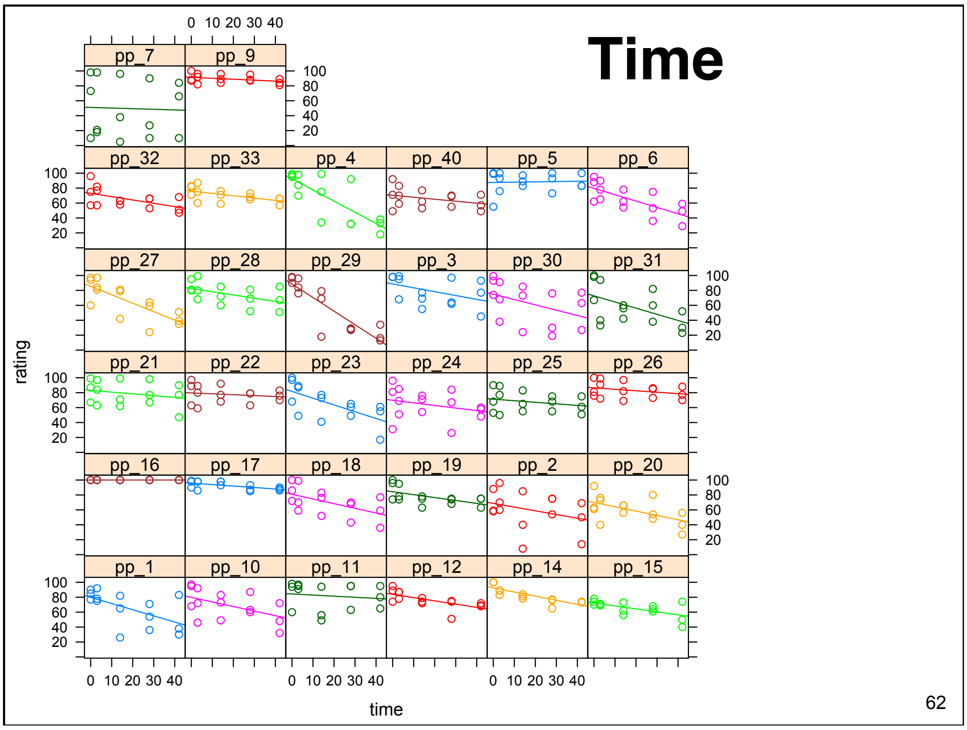
```
xyplot(rating ~ money | pp_code, groups = pp_code, data = v5, type = c('p', 'r'))
```

For Time

```
xyplot(rating ~ time | pp_code, groups = pp_code, data = v5, type = c('p', 'r'))
```

60


Money
pp_7
pp_9
pp_32
pp_33
pp_4
pp_40
pp_5
pp_6
pp_27
pp_28
pp_29
pp_3
pp_30
pp_31
pp_21
pp_22
pp_23
pp_24
pp_25
pp_26
pp_16
pp_17
pp_18
pp_19
pp_2
pp_20
pp_1
pp_10
pp_11
pp_12
pp_14
pp_15
rating
money
20 40 60 80
100 80 60 40 20
61


Time
pp_7
pp_9
pp_32
pp_33
pp_4
pp_40
pp_5
pp_6
pp_27
pp_28
pp_29
pp_3
pp_30
pp_31
pp_21
pp_22
pp_23
pp_24
pp_25
pp_26
pp_16
pp_17
pp_18
pp_19
pp_2
pp_20
pp_1
pp_10
pp_11
pp_12
pp_14
pp_15
rating
time
0 10 20 30 40
100 80 60 40 20
62

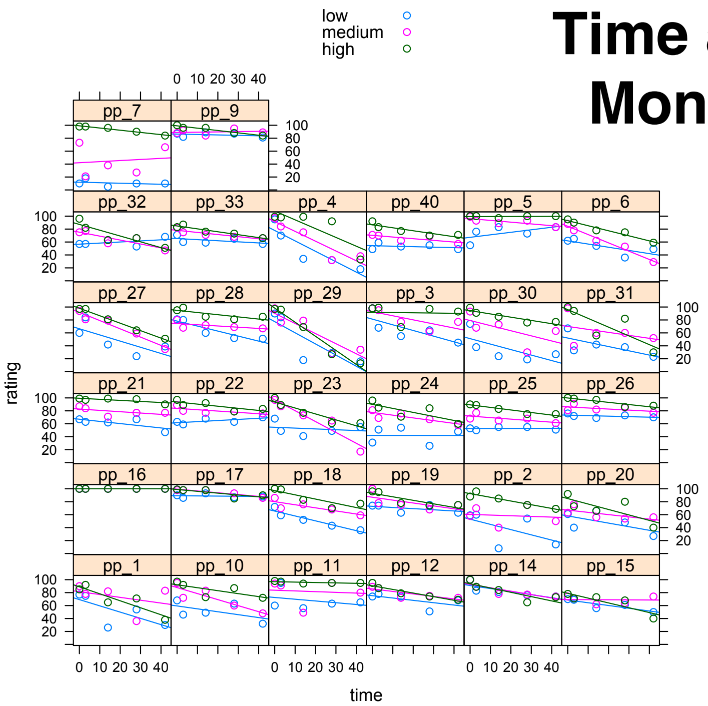
Time and Money

Time on x axis; o_money as separate lines

```
xyplot(rating ~ time | pp_code, groups =
o_money, data = v5, type = c('p', 'r'),
auto.key = TRUE)
```

`auto.key = TRUE` → adds figure legend

63



Time and Money

64

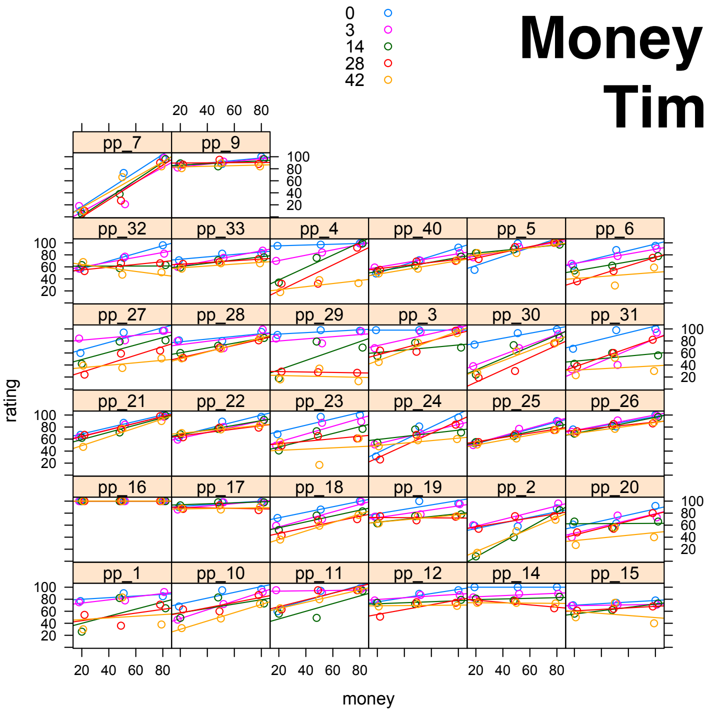
Money and Time

Money on x axis; Time as separate lines

```
xyplot(rating ~ money | pp_code, groups
= time, data = v5, type = c('p', 'r'),
auto.key = TRUE)
```

65

Money and Time



66

Individuals differ a lot in effects of Time and Money

It would be nice to get the numbers how the participants differ from each other (i.e., the random intercepts and random slopes)

`fixef()`

→fixed effect estimates of your model

```
> fixef(v_m1c)
(Intercept)      s_time     s_money
  71.754167   -7.964761   10.037750
```


`ranef()`

→ per-participant adjustments to fixed effects

```
> ranef(v_m1c)
$pp_code
        (Intercept)      s_time    s_money
pp_1   -5.616019409  -4.7786399 -1.6513253
pp_10  -2.931644297  -1.0844274  2.3557412
pp_11   9.114218849   5.5266371  0.2360635
pp_12   4.772589075   0.6948460 -4.0505320
pp_14  10.897122095  -0.4130446 -8.6088317
...
```

coef()

→ random and fixed effects together
i.e., best estimate for each pp's coefficients

```
> coef(v_m1c)
$pp_code
      (Intercept)      s_time    s_money
pp_1     66.13815 -12.74340079  8.3864244
pp_10    68.82252  -9.04918823 12.3934909
pp_11    80.86839  -2.43812376 10.2738132
pp_12    76.52676  -7.26991482  5.9872177
pp_14    82.65129  -8.37780546  1.4289180
pp_15    67.73749  -8.60869805  5.7816330
...
```

69

If you have more than 1 grouping variable

```
coef(mymodel)$pp_code
coef(mymodel)$f_item
```

To save the coefs in a csv file

```
v_m1c_coefs <- coef(v_m1c)$pp_code
write.csv(v_m1c_coefs, row.names = TRUE, file = 'v_m1c_coefs.csv')
```

`row.names = TRUE` → pp_code is included as 1st column

Why useful to save them?

- Descriptives, plots, ...
- Use in some other analyses (e.g., fMRI)

70

	A	B	C	D
1		(Intercept)	s_time	s_money
2	pp_1	66.1381472572	-12.7434007914	8.3864244114
3	pp_10	68.82252237	-9.0491882322	12.3934908987
4	pp_11	80.8683855158	-2.438123758	10.2738132174
5	pp_12	76.5267557415	-7.2699148168	5.9872177209
6	pp_14	82.6512887612	-8.3778054582	1.4289179751
7	pp_15	67.7374862816	-8.6086980507	5.7816329675
8	pp_16	98.4146427397	0.0584053472	0.1251181152
9	pp_17	91.2446027856	-2.8344483949	2.3280414432
10	pp_18	69.4885297045	-9.0795906928	12.4211233642
11	pp_19	77.5661384166	-6.4604392642	6.533258789
12	pp_2	60.646298243	-8.2973887227	17.0735410335
13	pp_20	61.60410219	-10.4891250255	10.9044332116
14	pp_21	77.8488171908	-3.3717236554	12.5630517301
15	pp_22	77.598557432	-2.9070871406	9.754576843
16	pp_23	65.8868752597	-13.1229341827	10.3932670512
17	pp_24	64.5482595516	-6.5455323517	14.940972429
18	pp_25	68.5583559368	-4.8510239655	12.7445090569
19	pp_26	82.2097265985	-3.1033641951	8.3688932814
20	pp_27	65.1912963837	-15.4904477586	10.5387662904
21	pp_28	74.7331485863	-6.7622635605	9.4681826219
22	pp_29	57.07762275	-25.2000636475	6.3157032038

71

Today: Digging Deeper

- Questions: General, homework-related
- Some homework-related add-ons
- Leftovers from last class: p values; multiple cores
- Post-hocs; writing up model and results; some plotting; linear and quadratic predictors; centering/ scaling to reduce multicollinearity
- **Advice from Barr et al and others**
  - **The R-sig-mixed-models debate**
  - **Testing parameters vs. effects: Bootstrap vs. LRT?**
- Homework

72

R-sig-mixed models

Who has read these emails recently?

- If not, do it as homework (available online as archives)
  → debate started on Feb 28, 2014

Short summary

- **E. Ellsiepen:** Followed Barr et al advice: maximal random-effects structure + LRTs for p values; odd results
- **D. Bates:** model is overparametrized (too complex for data); "Barr et al advice is dangerous"
- **R. Levy** (from Barr et al): perhaps no overparametrization; Bates misunderstood advice in paper; LRTs are fine even when overparametrized (testing significance of coefficient more problematic!)
- **D. Bates:** I should have re-read the paper

73

- **R. Levy:** More detailed advice about strategies; suggestions: try different optimizer; use LRTs, not tests that rely on precise estimate of coefficient
- Maechler, Bolker, ... thread is still alive

Important thoughts

If model overparametrized/with convergence problems:

- Try different optimizer(s): glmer: nlminb from package optimx
  `optimizer="optimx", optCtrl=list(method="nlminb")`
- **Use LRTs for p values** → For model comparison approach, not important whether the really-really best value for a coefficient is found!

→ Different coefficient values give virtually equally good fit

→ Actual value of coeff irrelevant for comparison of models with/without the predictor → more trustworthy p values!

74