Selecting Random Effects

Automatic Selection of Random Effects

- RE account for correlated errors due to repeated measures
  - Inclusion of only 2 RE provide covariance structure among repeated measures that appears to be appropriate for most empirical work
  - Rule of thumb: Include at least two RE in model unless there is very compelling evidence to the contrary
- Maximum number of RE is determined by number of time transformations ($k + 1$)
- One approach is to include RE for every time predictor
  - Can lead to including needless variance components
    - Relative statistical efficiency is decreased (SEs are larger)
    - Superfluous VC can lead to estimation problems
  - Less appealing when change curve is nonlinear

Random Effects and Variance Components

- Important to distinguish between REs and VC of the REs
  - REs are random variables
  - VCs are the variances and covariances of the random variables
- In LMER it is common to estimate the VCs (which index the aggregate effects) and not deal directly with the RE
- Consider the LMER model

$$y_{ij} = (\beta_0 + b_{0i}) + (\beta_1 + b_{1i})(\text{grade5}_{ij}) + \epsilon_{ij}$$

  - Includes 2 REs: b_{0i} and b_{1i}
  - REs are the individual deviations from the fixed effects

- The variance of a RE indexes individual variability
  - *Var*(b_{0i}) indexes variability in intercepts
  - *Var*(b_{1i}) indexes variability in slopes
- For inferences with LMER
  - Assumptions: REs and random errors are normally distributed with means = 0; and
  - REs may be correlated, but each is independent of the random errors
- Variance–covariance matrix among the random effects

$$\mathbf{G} = \begin{bmatrix} Var(b_{0i}) & Cov(b_{01}, b_{1i}) \\ Cov(b_{01}, b_{1i}) & Var(b_{1i}) \end{bmatrix}$$

Symmetric matrix

Number of unique elements

$$\frac{q^*(q^*+1)}{2}$$

q^* = Number of RE

LMER Equation	**G**
$y_{ij} = (\beta_0 + b_{0i}) + \beta_1(\text{grade5}_{ij}) + \epsilon_{ij}$	$\left[Var(b_{0i})\right]$
$y_{ij} = (\beta_0 + b_{0i}) + (\beta_1 + b_{1i})(\text{grade5}_{ij}) + \epsilon_{ij}$	$\begin{bmatrix} Var(b_{0i}) & 0 \\ 0 & Var(b_{1i}) \end{bmatrix}$
$y_{ij} = (\beta_0 + b_{0i}) + (\beta_1 + b_{1i})(\text{grade5}_{ij}) + \epsilon_{ij}$	$\begin{bmatrix} Var(b_{0i}) & Cov(b_{01}, b_{1i}) \\ Cov(b_{01}, b_{1i}) & Var(b_{1i}) \end{bmatrix}$

- All models have same fixed effects structure
- Why select different structure for **G**?
  - Theory or nature of data might suggest particular structure
  - Preliminary graphing of data or preliminary analysis might indicate one structure or another
    - Suppose initial estimation shows $Var(b_{1i}) \sim 0$ and $Cov(b_{0i}, b_{1i}) \sim -1$
    - High negative correlation and small estimated variance indicate **G** has too many parameters

Restricted Maximum Likelihood

- Decisions about inclusion of REs should be based on models estimated using REML rather than ML
  - ML yields VC that are biased downward (smaller than they should be); REML corrects for this
- To use REML estimation, use the argument `REML = TRUE` in the `lmer()` function (or omit the argument altogether)

```
## Estimate model using REML
> remlMod <- lmer(read ~ 1 + grade5 + (1 + grade5 | subid),
    data = mpls.l, REML = TRUE)

## Estimate model using ML
> mlMod <- lmer(read ~ 1 + grade5 + (1 + grade5 | subid),
    data = mpls.l, REML = FALSE)
```

```
## Print the REs

> VarCorr(remlMod)

$subid
            (Intercept)  grade5
(Intercept)      399.11 -19.586
grade5           -19.59   7.547
attr(,"stddev")
(Intercept)      grade5
     19.978       2.747
attr(,"correlation")
            (Intercept)  grade5
(Intercept)      1.0000 -0.3569
grade5           -0.3569  1.0000

attr(,"sc")
[1] 4.278
```

Estimated **G** matrix

Estimated standard deviations of REs

Estimated correlation matrix of REs

square root of estimated error variance

```
> VarCorr(remlMod)
$subid
            (Intercept)  grade5
(Intercept)      399.11 -19.586
grade5           -19.59   7.547
attr(,"stddev")
(Intercept)      grade5
     19.978       2.747
attr(,"correlation")
            (Intercept)  grade5
(Intercept)      1.0000 -0.3569
grade5          -0.3569  1.0000

attr(,"sc")
[1] 4.278
```

```
> VarCorr(remlMod)$subid
```

Use `attr()` function

```
> attr(VarCorr(remlMod)$subid, "stddev")
> attr(VarCorr(remlMod)$subid, "correlation")
> attr(VarCorr(remlMod)$subid, "sc")
```

```
## Print G matrix (REML)
> data.frame(VarCorr(remlMod)$subid)

            X.Intercept.  grade5
(Intercept)       399.11 -19.586
grade5            -19.59   7.547


## Print G matrix (ML)
> data.frame(VarCorr(mlMod)$subid)

            X.Intercept.  grade5
(Intercept)       380.59 -18.573
grade5            -18.57   6.966
```

VCs estimated using maximum likelihood are all smaller than those estimated with restricted maximum likelihood.

Extent of similarity depends on sample size

Random Effects and Correlated Data

- Approach with static predictors is not clear cut
- Possible to use multimodal approach
  - Results are vulnerable to misinterpretation
- To illustrate, consider the following models

$$y_{ij} = (\beta_0 + b_{0i}) + (\beta_1 + b_{1i})(\texttt{grade}_{ij}) + \beta_2(\texttt{dadv}_i) + \epsilon_{ij}$$

$$y_{ij} = (\beta_0 + b_{0i}) + (\beta_1 + b_{1i})(\texttt{grade}_{ij}) + \beta_2(\texttt{dadv}_i) + \beta_3(\texttt{ethW}_i) + \epsilon_{ij}$$

Suppose goal is to choose time transformations and there is suspicion that quadratic term is necessary

Models are

$$y_{ij} = (\beta_0 + b_{0i}) + (\beta_1 + b_{1i})(\texttt{grade}_{ij}) + \beta_2(\texttt{grade}^2_{ij}) + \beta_3(\texttt{dadv}_i) + \epsilon_{ij}$$

and

$$y_{ij} = (\beta_0 + b_{0i}) + (\beta_1 + b_{1i})(\texttt{grade}_{ij}) + \beta_2(\texttt{grade}^2_{ij}) + \beta_3(\texttt{dadv}_i) + \beta_4(\texttt{ethW}_i) + \epsilon_{ij}$$

The second model has one additional quadratic term.

	Modnames	K	AICc	Delta_AICc	AICcWt	eratio
1	1L	7	579.5	3.8907	0.07594	6.996
2	1Q	8	580.3	4.7414	0.04963	10.705
3	2L	8	575.6	0.0000	0.53131	1.000
4	2Q	9	576.4	0.8746	0.34311	1.549

- Model 2L is best fitting; model 2Q second best
- Because of this you might say that the best approximating model may include a quadratic change curve
  - 2Q is fitting well because it shares static predictors with 2L not because of the quadratic term
  - Comparing 1Q to 1L it is seen that the quadratic also fits worse
- With respect to the change curve, model 1L fits better then 2Q

- When change curves of static predictor subgroups are not parallel then the order of polynomial should be selected with the interactions in the model
- Suppose intention is to evaluate models that differ in ethnicity and risk effects
  - Three polynomial models are examined (I, L, and Q)
    - Two static predictors as single effects and all possible interactions

Intercept-only model

$$y_{ij} = (\beta_0 + b_{0i}) + (\beta_1 + b_{1i})(\texttt{grade5}_{ij}) + \beta_2(\texttt{dadv}_i) + \beta_3(\texttt{ethW}_i) + \epsilon_{ij}$$

Linear model

$$y_{ij} = (\beta_0 + b_{0i}) + (\beta_1 + b_{1i})(\texttt{grade5}_{ij}) + \beta_2(\texttt{dadv}_i) + \beta_3(\texttt{ethW}_i) + \beta_4(\texttt{grade5}_{ij} \cdot \texttt{dadv}_i) + \beta_5(\texttt{grade5}_{ij} \cdot \texttt{ethW}_i) + \epsilon_{ij}$$

Quadratic model

$$y_{ij} = (\beta_0 + b_{0i}) + (\beta_1 + b_{1i})(\texttt{grade5}_{ij}) + \beta_2(\texttt{grade5}_{ij}^2) + \beta_3(\texttt{dadv}_i) + \beta_4(\texttt{ethW}_i) + \beta_5(\texttt{grade5}_{ij} \cdot \texttt{dadv}_i) + \beta_6(\texttt{grade5}_{ij} \cdot \texttt{ethW}_i) + \beta_7(\texttt{grade5}_{ij}^2 \cdot \texttt{dadv}_i) + \beta_8(\texttt{grade5}_{ij}^2 \cdot \texttt{ethW}_i) + \epsilon_{ij}$$

```
## Estimate models
> model.i <- lmer(read ~ dadv + ethW + (1 | subid), mpls.l,
    REML = FALSE)
> model.l <- lmer(read ~ grade5 * dadv + grade5 * ethW + (1 | subid),
    data = mpls.l, REML = FALSE)
> model.q <- lmer(read ~ grade5 * dadv + grade5 * ethW +
    I(grade5 ^ 2) * dadv + I(grade5 ^ 2) * ethW + (1 | subid),
    data = mpls.l, REML = FALSE)

## Compute AIC
> myaicc <- as.data.frame(aictab(
    cand.set = list(model.i, model.l, model.q),
    modnames = c("I", "L", "Q"), sort = FALSE))[ , -c(5,7)]

## Compute evidence ratio
> myaicc$eratio <- max(myaicc$AICcWt) / myaicc$AICcWt

> myaicc
```

	Modnames	K	AICc	Delta_AICc	AICcWt	eratio
1	I	5	622.5	42.011	0.0000000007294	1325838559.08
2	L	8	580.5	0.000	0.9670361058595	1.00
3	Q	11	587.2	6.758	0.03296389341111	29.34

- Linear model has large weight of evidence

Likelihood Ratio Test: Analysis Without Static Predictors

- Analysis with polynomial terms can be performed in a step-up or top-down approach
  - Step-up approach compares initial (simpler) model to a more complex model that adds polynomial terms
  - Top-down approach compares initial (more complex) model to a simpler model that removes polynomial terms
- Recall that both approaches compare nested models (reduced model has fewer parameters)
- Emphasis when selecting polynomial terms is on accept/reject decision making
  - Testing typically terminates with the first non-significant result
  - Sometimes a rule of two consecutive non-significant results is used

- Again consider the 3 models with no static predictors fitted earlier (I, L, and Q)

```
## Fit models
> model.i <- lmer(read ~ 1 + (1 | subid), data = mpls.l,
                  REML = FALSE)

> model.l <- lmer(read ~ 1 + grade5 + (1 | subid),
                  data = mpls.l, REML = FALSE)

> model.q <- lmer(read ~ 1 + grade5 + I(grade5 ^ 2) +
                  (1 | subid), data = mpls.l, REML = FALSE)

## LRT
> anova(model.i, model.l, model.q)
```

```
Data: mpls.l
Models:
model.i: read ~ 1 + (1 | subid)
model.l: read ~ 1 + grade5 + (1 | subid)
model.q: read ~ 1 + grade5 + I(grade5^2) + (1 | subid)
        Df AIC BIC logLik Chisq Chi Df      Pr(>Chisq)
model.i  3 633 640   -313
model.l  4 587 597   -290 47.22      1 0.0000000000064 ***
model.q  5 589 601   -289  0.62      1            0.43
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

- Step-up approach
  - Begin with model.i compared to model.l (linear term is needed)
  - Model.l compared to model.q (quadratic term does not add anything...it is not needed)
- Top-down approach – begin with model.q compared to model.l (non-significance suggests more parsimonious model is accepted)

LRT with Static Predictors

- Use same modeling techniques as with AICc
- All polynomials should be fitted with interaction terms included in the model

Subject-Level Selection of Time Predictors

- Goal is to use individual change curves to suggest polynomial effects to be included in model
- Goodness-of-fit index computed for each subject to quantify adequacy of model
  - These indices are then summarized in plots or pooled to create a single descriptive measure
- When subgroups differ in curve shape, highest order polynomial should be fitted for all subjects
  - For subgroups with simpler curve, parameter estimates of higher order terms will be close to zero
- Disadvantage to subject-level selection is that it is only descriptive
  - Statistical theory cannot be used to judge sample effects
- Missing data may not allow the same model to be fitted to all subjects

Level 1 Polynomial Model

- Change curves are either connected observed values or connected fitted values
- Observed change curves are not model based
- Fitted change curves use the subject-specific Level 1 model
- Consider the following Level 1 model

$$y_{ij} = \beta^*_{0i} + \beta^*_{1i}(\texttt{grade5}_{ij}) + \beta^*_{2i}(\texttt{grade5}^2_{ij}) + \ldots + \beta_{ki}(\texttt{grade5}^k_{ij}) + \epsilon_{ij}$$

where each β^* is a subject-specific regression weight

- Since $i = 1, ..., N$, there are N Level 1 equations
- There are no static predictors (those are level 2 predictors)
- For subject-level analysis, the correlation among repeated measures is ignored and OLS estimation is used. This is justified since the concern is only with description.

Missing Data

- The requirement that the number of time points > k + 1 limits the polynomial for subjects with missing data
- Useful to create a new variable that indexes the number of missing points
- This can be used to exclude some subjects from the analysis

The `ddply()` function from the **plyr** library evaluates an expression/ function a given number of times and then stores the result in a data frame.

```
## load plyr library

> library( plyr )


## Create a vector of 0/1 that indicates missingness of read
## for each row

> mpls.l$miss <- as.numeric(is.na(mpls.l$read))


## Compute the number of missing values for each subject

> mysel <- ddply(.data = data.frame(mpls.l$miss),
    .variables = .(mpls.l$subid), .fun = sum)
```

```
  ## Change variable names

  > colnames(mysel) <- c("subid", "totmiss")


## Merge the number of missing values with the original data
## frame

> mpls.l2 <- merge(mpls.l, mysel, by = "subid")


> head(mpls.l2)

  subid risk gen eth ell sped  att grade read grade5 dadv ethW miss totmiss
1     1  HHM   F Afr   0    N 0.94     5  172      0 DADV   NW    0       0
2     1  HHM   F Afr   0    N 0.94     6  185      1 DADV   NW    0       0
3     1  HHM   F Afr   0    N 0.94     7  179      2 DADV   NW    0       0
4     1  HHM   F Afr   0    N 0.94     8  194      3 DADV   NW    0       0
5     2  HHM   F Afr   0    N 0.91     5  200      0 DADV   NW    0       1
6     2  HHM   F Afr   0    N 0.91     6  210      1 DADV   NW    0       1
```

Subject-Level Fits

- Examine the subject-specific fitted curves
  - For small *n*, graphing works well
  - For large *n* graphs are generally cumbersome and it works better to examine summary measure(s) from the fitted model (e.g., R^2)

1) Fit a linear model to each subject using the `lm()` function
2) Extract the summary measures of interest from each fitted model (e.g., coefficients, R^2, etc.)
3) Plot or summarize (e.g., through a mean) the summary measures that were extracted

```
## Function to fit lm

> fit.linear <- function(x) {
    lm(read ~ grade5, data = x)
    }


## Fit lm to each subject's data

> mylm.1 <- dlply(.data = mpls.l,
    .variables = .(mpls.l$subid), .fun = fit.linear)

> mylm.1

$`1`

Call:
lm(formula = read ~ grade5, data = x)

Coefficients:
(Intercept)       grade5
        174            6
          ⋮
```

The dlply() function fits the fit.linear() function to each subject's data and outputs the results in a list

```
## Function to obtain coefficients

> get.coef <- function(x) {
    x$coefficients
    }


## Obtain coefficients from each subject's model

> ldply(.data = mylm.1, .fun = get.coef)
```

The `ldply()` function fits the `get.coef()` function to each model stored in the list and then outputs the results as a data frame.

	mpls.l$subid	(Intercept)	grade5
1	1	173.5	6.0
2	2	201.8	4.5
3	3	190.6	7.6
4	4	199.3	-3.0
5	5	208.7	1.7
6	6	190.9	2.9
7	7	200.2	6.2
8	8	191.5	1.5
9	9	147.9	10.4
10	10	201.8	6.5
11	11	220.5	6.5
12	12	228.0	7.0
13	13	229.0	2.5
14	14	198.8	12.3
15	15	216.7	9.0
16	16	228.0	0.5
17	17	201.7	5.2
18	18	218.1	0.6
19	19	214.3	3.0
20	20	207.9	3.4
21	21	237.3	3.0
22	22	220.8	8.5

Subject 4 has a negative slope, but all the rest are positive

Repeat this Process for a Quadratic Model

```
## Function to fit quadratic

> fit.quad <- function(x) {
    lm(read ~ grade5 + I(grade5 ^ 2), data = x)
    }



## Fit lm to each subject's data

> mylm.1 <- dlply(.data = mpls.l,
    .variables = .(mpls.l$subid), .fun = fit.linear)



## Obtain coefficients from each subject's model

> ldply(.data = mylm.1, .fun = get.coef)
```

	mpls.l$subid	(Intercept)	grade5	I(grade5^2)
1	1	174.0	4.50	0.50
2	2	200.0	15.50	-5.50
3	3	191.6	4.60	1.00
4	4	200.0	-7.00	2.00
5	5	207.4	5.45	-1.25
6	6	188.7	9.65	-2.25
7	7	199.2	9.20	-1.00
8	8	191.0	4.50	-1.50
9	9	147.4	11.90	-0.50
10	10	200.0	17.50	-5.50
11	11	218.7	11.75	-1.75
12	12	226.5	11.50	-1.50
13	13	229.8	0.25	0.75
14	14	198.5	13.05	-0.25
15	15	218.0	1.00	4.00
16	16	226.8	4.25	-1.25
17	17	202.2	3.70	0.50
18	18	218.6	-0.90	0.50
19	19	215.0	-1.00	2.00
20	20	203.9	15.40	-4.00
21	21	237.0	5.00	-1.00
22	22	219.0	19.50	-5.50

Most subjects have a negative effect for the quadratic term.

The signs represent shape differences.

A negative sign is *concave* to the *x*-axis (∩).

A positive sign is *convex* to the *x*-axis (∪).

```
## Function to extract R2

> get.R2 <- function(x) {
    summary(x)$r.squared
    }


## Extract R2 from each subject's model

> linear.r2 <- ldply(.data = mylm.1, .fun = get.R2)


## Change column names

> colnames(linear.r2) <- c("subid", "Rsq")


## Merge missingness and R2

> Rsq1 <- merge(mysel, linear.r2, by = "subid")
```

```
> Rsq1
   subid totmiss     Rsq
1      1       0 0.68966
2      2       1 0.66758
3      3       0 0.96267
4      4       1 0.87097
5      5       0 0.58384
6      6       0 0.24342
7      7       0 0.97563
8      8       1 0.75000
9      9       0 0.91197
10    10       1 0.80732
11    11       0 0.89989
12    12       0 0.81940
13    13       0 0.32982
14    14       0 0.99435
15    15       1 0.93822
16    16       0 0.03226
17    17       0 0.81939
18    18       0 0.18000
19    19       1 0.87097
20    20       0 0.47377
21    21       1 0.96429
22    22       1 0.87753
```

Repeat this Process for a Quadratic Model

```
> quad.r2 <- ldply(.data = mylm.2, .fun = get.R2)

> colnames(quad.r2) <- c("subid", "Rsq")

> Rsq2 <- merge(mysel, quad.r2, by = "subid")

> Rsq2
```

	subid	totmiss	Rsq
1	1	0	0.6935
2	2	1	1.0000
3	3	0	0.9760
4	4	1	1.0000
5	5	0	0.8364
6	6	0	0.3606
7	7	0	0.9959
8	8	1	1.0000
9	9	0	0.9137
10	10	1	1.0000
11	11	0	0.9521
12	12	0	0.8495
13	13	0	0.3536
14	14	0	0.9947
15	15	1	1.0000
16	16	0	0.1935
17	17	0	0.8255
18	18	0	0.2800
19	19	1	1.0000
20	20	0	0.9984
21	21	1	1.0000
22	22	1	1.0000

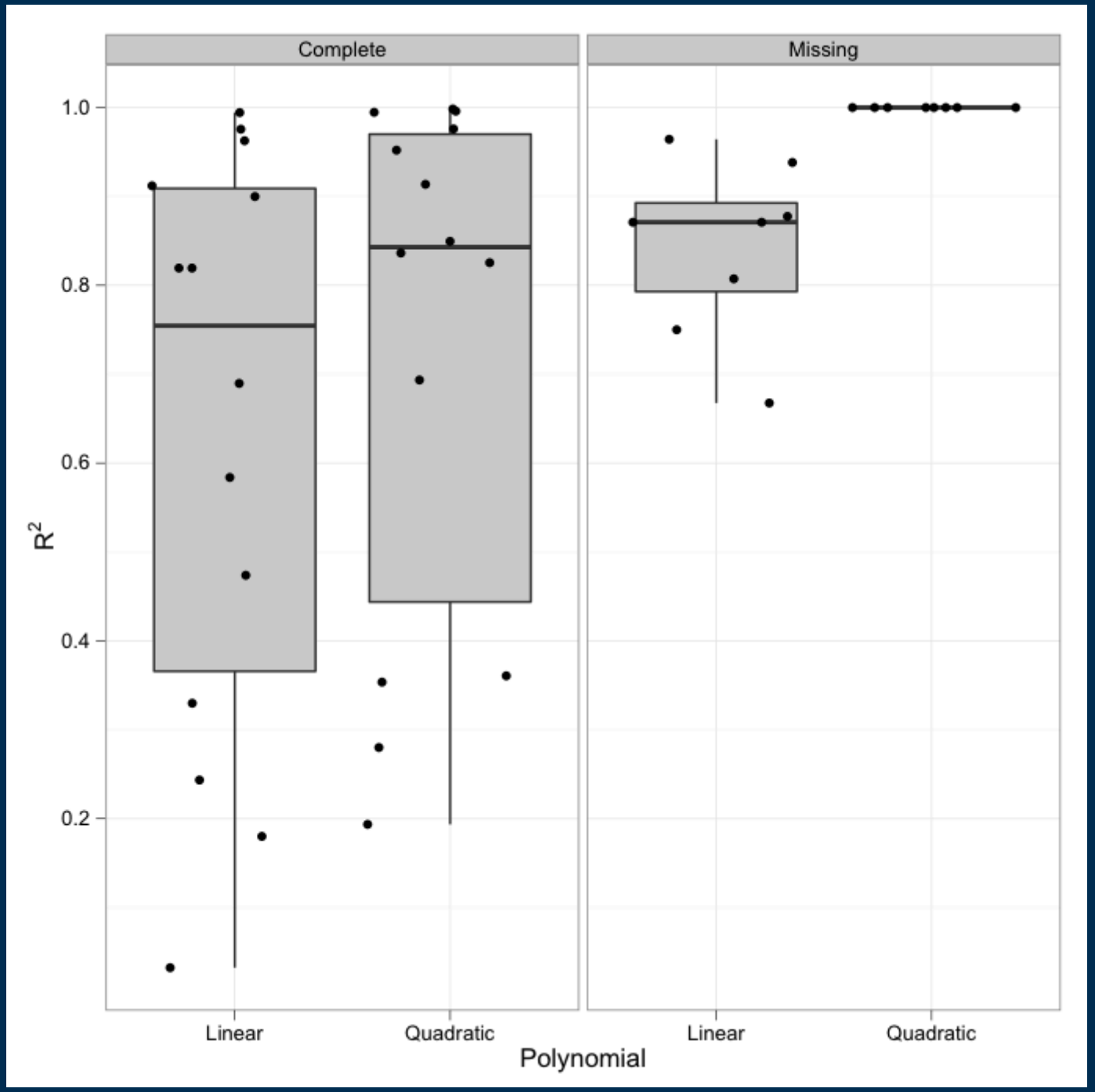
Quadratic polynomial fits perfectly for subjects with missing data (saturated model)

Plot the R^2 values Conditional on Missingness

```
## Get number of rows
> N <- nrow(Rsq1)

## Combine the two data frames into a new data frame
> plotdata <- data.frame(rbind(Rsq1, Rsq2), c(rep(1, N),
    rep(2, N)))

## Change 4th column name to poly
> colnames(plotdata)[4] <- "poly"

## Add variable to indicate polynomial term
> plotdata$poly.f <- factor(plotdata$poly,
    labels = c("Linear", "Quadratic"))

## Add variable to indicate degree of missingness
> plotdata$missing.f <- factor(plotdata$totmiss,
    labels = c("Complete", "Missing"))
```

	subid	totmiss	Rsq	poly	poly.f	missing.f
1	1	0	0.68966	1	Linear	Complete
2	2	1	0.66758	1	Linear	Missing
3	3	0	0.96267	1	Linear	Complete
4	4	1	0.87097	1	Linear	Missing
5	5	0	0.58384	1	Linear	Complete
6	6	0	0.24342	1	Linear	Complete
7	7	0	0.97563	1	Linear	Complete
8	8	1	0.75000	1	Linear	Missing
9	9	0	0.91197	1	Linear	Complete
10	10	1	0.80732	1	Linear	Missing
11	11	0	0.89989	1	Linear	Complete
12	12	0	0.81940	1	Linear	Complete
13	13	0	0.32982	1	Linear	Complete
14	14	0	0.99435	1	Linear	Complete
15	15	1	0.93822	1	Linear	Missing
16	16	0	0.03226	1	Linear	Complete
17	17	0	0.81939	1	Linear	Complete
18	18	0	0.18000	1	Linear	Complete
19	19	1	0.87097	1	Linear	Missing
20	20	0	0.47377	1	Linear	Complete
21	21	1	0.96429	1	Linear	Missing
22	22	1	0.87753	1	Linear	Missing

```
## Plot the R2 values

> ggplot(data = plotdata, aes(x = poly.f, y = Rsq)) +
    geom_boxplot(fill = "grey80") +
    geom_point(position = "jitter") +
    facet_grid(. ~ missing.f) +
    theme_bw() +
    xlab("Polynomial") +
    ylab(expression(R ^ 2))



## Obtain median values

> tapply(plotdata$Rsq, list(plotdata$missing.f, plotdata$poly.f),
    median)

         Linear Quadratic
Complete 0.7545    0.8429
Missing  0.8710    1.0000
```


Complete
Missing
1.0
0.8
0.6
0.4
0.2
R^2
Linear
Quadratic
Linear
Quadratic
Polynomial

Adjusted R^2

- Drawback of R^2 is that it increases as more predictors are add into the model (even if they are worthless)
- Alternative is to use the penalized measure, *adj.* R^2

$$\bar{R}_i^2 = 1 - (1 - R_i^2)\left(\frac{n_i - 1}{n_i - k - 1}\right)$$

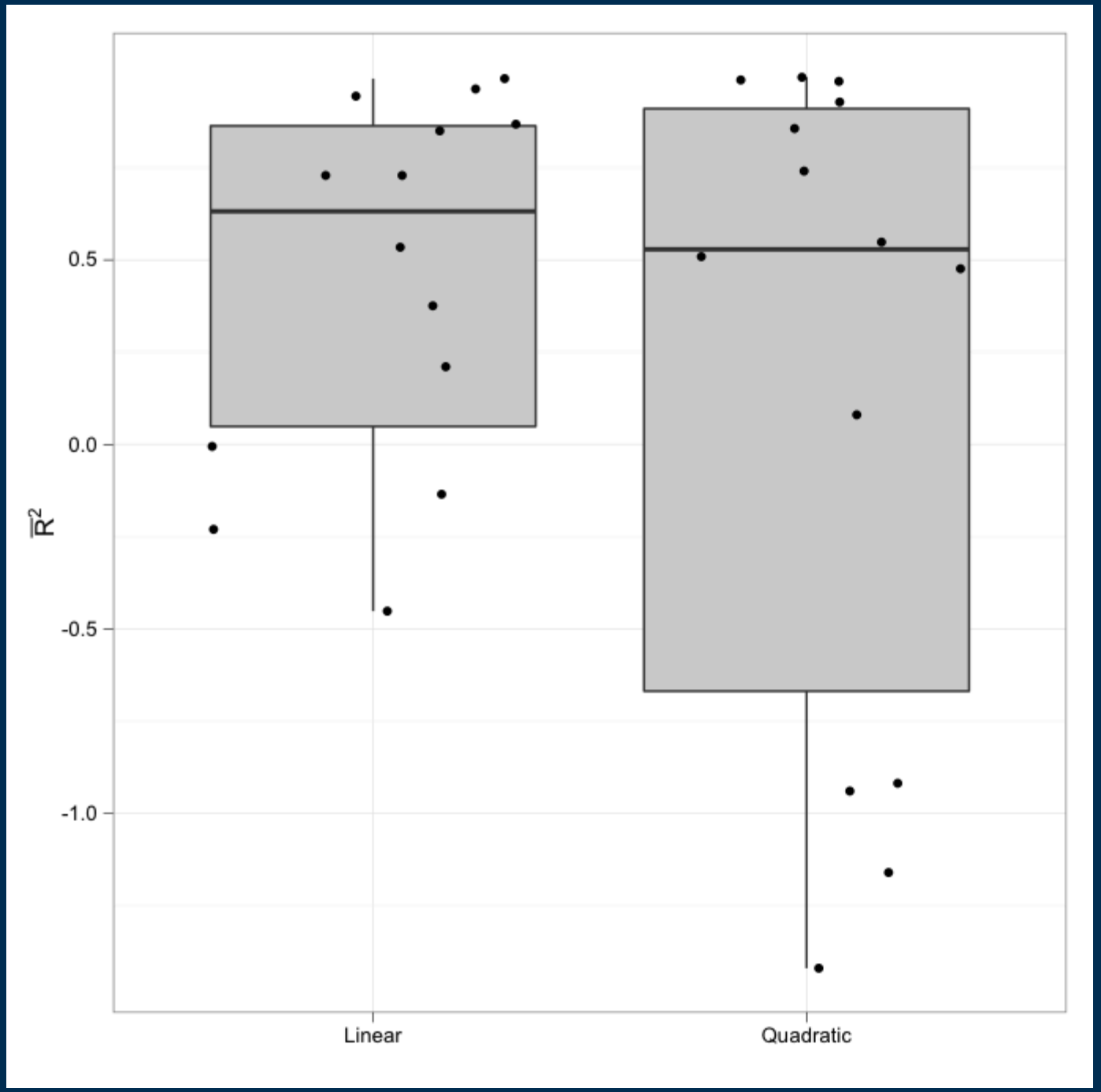
where k is the number of polynomial terms bigger than degree 0 (number of predictors)

$\bar{R}_i^2$ is not defined for saturated models

$\bar{R}_i^2$ can be extracted from the `summary()` object using `$adj.r.squared`

```
> mysub <- subset(mpls.l2, totmiss == 0)

> mylm.1 <- dlply(.data = mysub,
    .variables = .(mysub$subid), .fun = fit.linear)

> mylm.2 <- dlply(.data = mysub,
    .variables = .(mysub$subid), .fun = fit.quad)


## Get adjusted R2

> get.adj.R2 <- function(x){
   summary(x)$adj.r.squared
   }


> adjRsq1 <- ldply(.data = mylm.1, .fun = get.adj.R2)

> colnames(adjRsq1) <- c("subid", "adjRsq")

> adjRsq2 <- ldply(.data = mylm.2, .fun = get.adj.R2)

> colnames(adjRsq2) <- c("subid", "adjRsq")
```

```
## Create data frame

> N <- nrow(adjRsq1)

> plotdata <- data.frame(rbind(adjRsq1, adjRsq2),
    c(rep(1, N), rep(2, N)))

> colnames(plotdata)[3] <- "poly"

> plotdata$poly.f <- factor(plotdata$poly,
    labels = c("Linear", "Quadratic"))


## Plot adjusted R2

> ggplot(data = plotdata, aes(x = poly.f, y = adjRsq)) +
   geom_boxplot(fill = "grey80") +
   geom_point(position = "jitter") +
   theme_bw() +
   xlab("") +
   ylab(expression(bar(R)^2))
```


0.5
0.0
-0.5
-1.0
$\overline{R}^2$
Linear
Quadratic

Pooled Measures of Fit

- Another alternative is to pool information from all subjects into a single statistic
- We will combine the subject-level sums of squares (foundation for R^2)

$$R_i^2 = 1 - \left(\frac{SSR_i}{SST_i} \right)$$

where *SSR* is the sum of squares residuals and *SST* is the sum of squares total. A pooled version can be created by replacing each of the sum of squares by the sum among all subjects

$$R^2_{meta} = 1 - \left(\frac{\sum_{i=1}^{N} SSR_i}{\sum_{i=1}^{N} SST_i} \right)$$

- The meta-R^2 will always increase as predictors are added (just like R^2)
- Better to use penalized fit index
  - Meta-*RSE* (residual standard error)
  - Meta-R^2 *adjusted*

Subject-level RSE

$$RSE_i = \sqrt{\frac{SSR_i}{n_i - k_i - 1}}$$

Pooled meta-RSE

$$RSE_{meta} = \sqrt{\frac{\sum_{i=1}^{N} SSR_i}{\sum_{i=1}^{N} (n_i - k_i - 1)}}$$

Compute Meta-RSE

```
## Estimate coefficients

> my1 <- dlply(.data = mpls.l,
    .variables = .(mpls.l$subid), .fun = fit.linear)

> my2 <- dlply(.data = mpls.l,
    .variables = .(mpls.l$subid), .fun = fit.quad)


## Sum of squares residuals

> ssResid <- function(x){
   sum(resid(x) ^ 2)
   }

> sse1 <- sum(ldply(.data = my1, .fun = ssResid)[ ,2])

> sse2 <- sum(ldply(.data = my2, .fun = ssResid)[ ,2])
```

```
## Residual df

> dfResid <- function(x){
    x$df.residual
    }

> df1 <- sum(ldply(.data = my1, .fun = dfResid)[, 2])

> df2 <- sum(ldply(.data = my2, .fun = dfResid)[, 2])


## Compute RSE meta

> RSEmeta.1 <- sqrt(sse1/df1)

> RSEmeta.2 <- sqrt(sse2/df2)
```

```
> RSEmeta.1

[1] 4.262


> RSEmeta.2

[1] 5.623
```

Since the meta-RSE value is smaller for the linear model, it is evidence that the linear model is sufficient

Meta-R^2 *Adjusted*

- Obtained by dividing meta-R^2 numerator and denominator by *df*
- We will combine the subject-level sums of squares (foundation for R^2)

$$R^2_{meta} = 1 - \left(\frac{\sum_{i=1}^{N} SSR_i}{\sum_{i=1}^{N} (n_i - k_i - 1)} \Big/ \frac{\sum_{i=1}^{N} SST_i}{\sum_{i=1}^{N} (n_i - k_i - 1)} \right)$$

```
## Use function in script file
## Only change the first 3 lines (if need be)

> Rsq.meta

  lm.objects Rsqmeta
1     mylm.1  0.8298
2     mylm.2  0.8848


> adjRsq.meta

  lm.objects adjRsqmeta
1     mylm.1     0.6432
2     mylm.2     0.2962
```

These values are closer together than the medians computed before.

Although the quadratic model has a higher meta-R^2, the increase is not amazing.

After penalization, the linear model has a meta-R^2 that is more than twice that of the quadratic model.