Modeling Nonlinear Change

Data Set and Analysis Strategy

- Minneapolis data well characterized by linear growth
  - Limited time period (grade 5 – grade 8)
  - Empirical evidence suggests that change curves exhibit rapid growth followed by a plateau
- We will examine augmented set of Minneapolis data (grade 2 - grade 8)

```
> MPLSW <- read.table(file = "/.../MPLSW.txt", header = TRUE,
                      na.strings = "-9")
> MPLSW
```

	subid	read.2	read.3	read.4	read.5	read.6	read.7	read.8
1	1	137	186	192	172	185	179	194
2	2	154	163	186	200	210	209	NA
3	3	154	186	181	191	199	203	215
4	4	166	168	198	200	195	194	NA
5	5	179	185	183	207	213	212	213
6	6	156	171	175	191	189	206	195
7	7	163	177	184	199	208	213	218
8	8	175	159	185	191	194	194	NA
9	9	177	159	173	149	154	174	177
10	10	151	177	177	200	212	213	NA
11	11	145	168	171	178	191	193	199
12	12	157	170	172	188	192	208	206
13	13	181	195	203	228	236	228	239
14	14	177	185	199	199	210	225	235
15	15	183	202	204	218	223	236	NA
16	16	195	192	211	228	226	234	227
17	17	187	204	206	201	210	208	219
18	18	185	204	219	218	220	217	221
19	19	189	200	226	215	216	221	NA
20	20	203	195	208	204	215	219	214
21	21	182	194	177	237	241	243	NA
22	22	174	205	220	219	233	236	NA

Read in Minneapolis.csv data

```
> mpls <- read.csv(file = “/.../Minneapolis.csv”,
           na.strings = “-9”)
```

Create dadv predictor

```
> mpls$dadv <- ifelse(mpls$risk == "ADV", "ADV", "DADV")

> mpls$dadv <- factor(mpls$dadv)
```

Merge data frames

```
> mpls2 <- merge(MPLSW, subset(mpls, select = c(subid, dadv)),
              by = "subid")
```

```
> head(mpls2)
  subid read.2 read.3 read.4 read.5 read.6 read.7 read.8 dadv
1     1    137    186    192    172    185    179    194 DADV
2     2    154    163    186    200    210    209     NA DADV
3     3    154    186    181    191    199    203    215 DADV
4     4    166    168    198    200    195    194     NA DADV
5     5    179    185    183    207    213    212    213 DADV
6     6    156    171    175    191    189    206    195 DADV
```

Reshape to long format

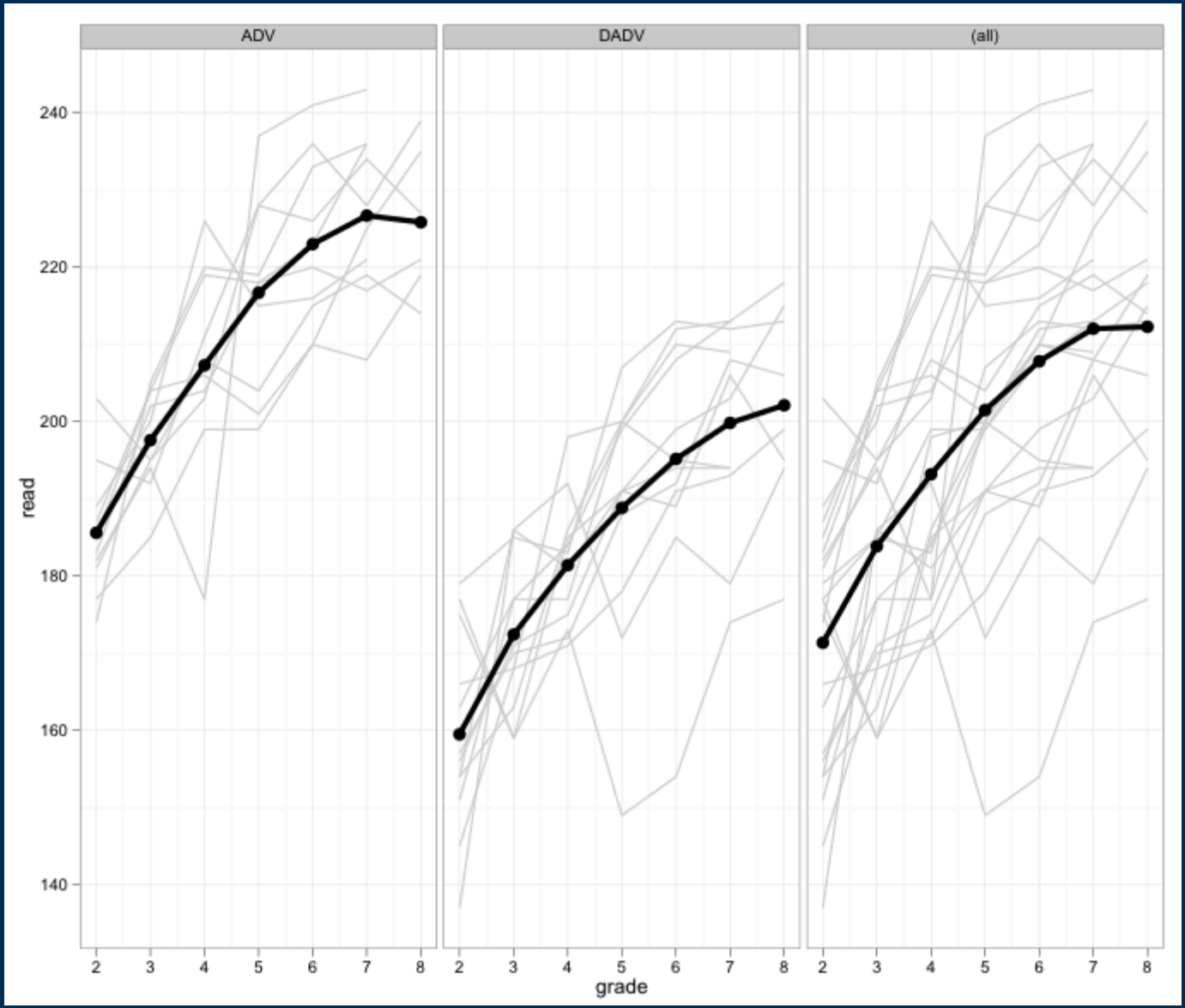
```
> mplsL <- reshape(mpls2, varying = 2:8, times = 2:8,
      idvar = "subid", timevar = "grade", direction = "long")
```

Arrange by subject ID number

```
> library(plyr)

> mplsL <- arrange(mplsL, subid)
```

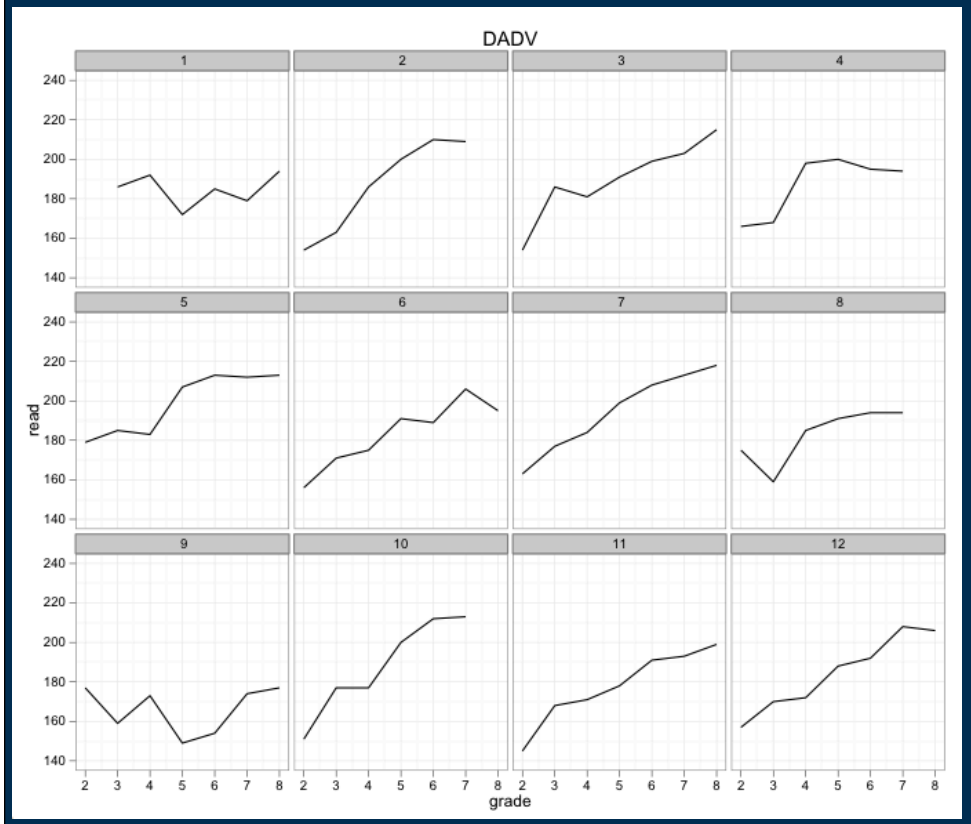
- Construct group-level and subject-level plots of the data

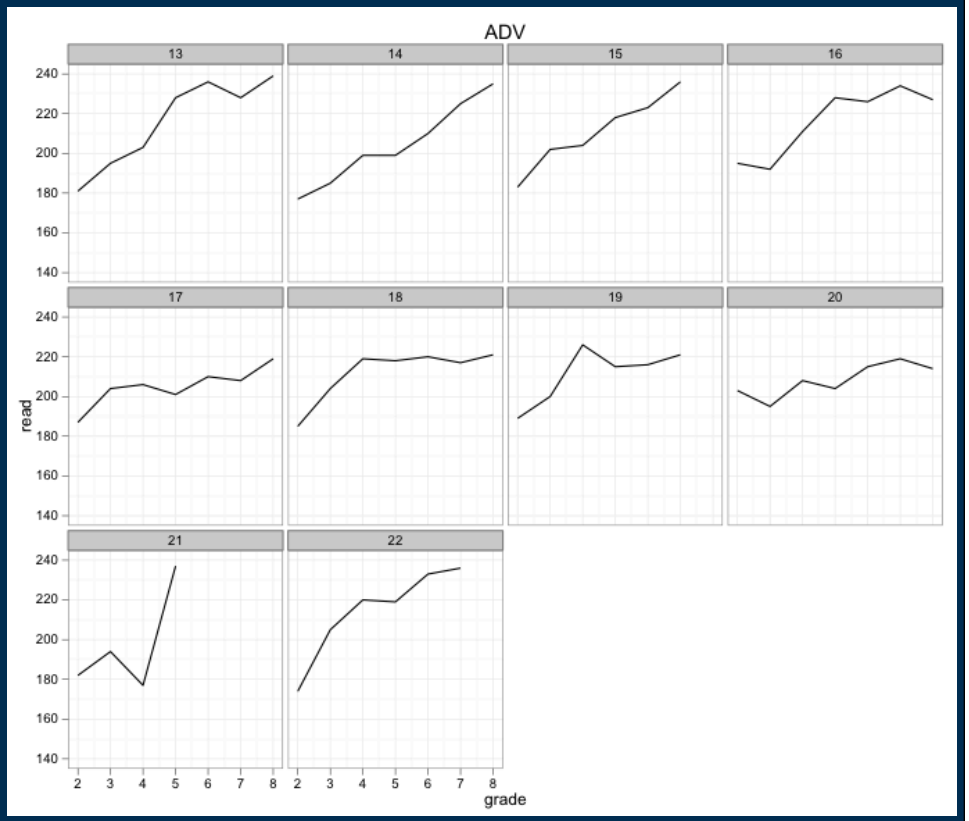
```
> ggplot(mplsL, aes(x = grade, y = read, group = subid)) +
    scale_x_continuous(breaks = 2:8) +
    theme_set(theme_bw()) +
    geom_line(colour = "grey80") +
    stat_summary(aes(group = 1), fun.y = mean, geom = "point",
       size = 3.5) +
    stat_summary(aes(group = 1), fun.y = mean, geom = "line",
       lwd=1.5) +
    facet_grid(. ~ dadv, margins = TRUE)
```


ADV
DADV
(all)
240
220
200
180
160
140
read
2
3
4
5
6
7
8
grade

- Individual curves faceted by subject ID

```
## Create separate data frames
> DADV <- subset(mplsL, dadv == "DADV")
> ADV <- subset(mplsL, dadv == "ADV")

## DADV group
> ggplot(DADV, aes(x = grade, y = read)) +
    geom_line() +
    facet_wrap( ~ subid) +
    opts(title = "DADV") +
    ylim(140, 240)

## ADV group
> ggplot(ADV, aes(x = grade, y = read)) +
    geom_line() +
    facet_wrap( ~ subid) +
    opts(title = "ADV") +
    ylim(140, 240)
```


DADV
1
2
3
4
5
6
7
8
9
10
11
12
read
grade


ADV
13
14
15
16
17
18
19
20
21
22
read
grade

Global versus Local Models

- Differentiation made between global and local models in the modeling of nonlinear change
  - With global models, the change equations apply to the entire range of the data
    - Fixed effects define change over entire time span
    - Most common in social/behavioral sciences (typically polynomials)
  - With local models, the observed time period is chunked into a series of smaller time spans
    - Different change equations for different spans
    - Fixed effects define change over specific time span

Global Model: Polynomial

- Most common global models use polynomial
  - Discussed in Unit 9
  - Here we will look at different interpretations, and
  - Alternate forms of polynomials
- Why are they so common?
  - Flexible
  - Easy to construct and specify in a model
  - Familiar to most applied researchers

Raw Polynomials

$$E(y_{ij}) = \beta_0 + \beta_1(\text{grade5}_{ij}) + \beta_2(\text{grade5}^2_{ij}) + \beta_3(\text{grade5}^3_{ij}) + \ldots + beta_k(\text{grade5}^k_{ij})$$

Consider quadratic polynomial (simplest case)

$$E(y_{ij}) = \beta_0 + \beta_1(\text{grade5}_{ij}) + \beta_2(\text{grade5}^2_{ij})$$

Rewrite by factoring out grade

$$E(y_{ij}) = \beta_0 + \left[\beta_1 + \beta_2(\text{grade5}_{ij})\right](\text{grade5}_{ij})$$

The effect of grade on the response variable depends on grade itself! The slope term indicates there is a linear change in the slope for grade, as grade itself increases. **The linear relationship between grade and read changes as grade increases.**

Changing linear relationship quantified by derivative

- Derivative is slope of linear curve tangent to the polynomial curve at a given value of grade
- Derivative found using `expression()` and `D()` functions

```
> quad <- expression(B0 + B1 * grade + B2 * grade ^ 2)
> my.deriv <- D(quad, "grade")
> my.deriv

B1 + B2 * (2 * grade)
```

- Values for grade can be substituted in to find the particular slope

Consider quadratic model with two random effects

```
## Fit model
> lmer.1 <- lmer(read ~ grade + I(grade ^ 2) + (grade | subid),
                 data = mplsL, REML = FALSE)

## Fit model
> my.fe <- fixef(lmer.1)

## Assign values to B0, B1, B2
> for(i in 1:3){
    assign(paste("B", i - 1, sep = ""), my.fe[i])
    }

## Show values for B0, B1, B2
> print(c(B0, B1, B2))

(Intercept)       grade  I(grade^2)
 143.622847   15.745161   -0.839052
```

```
## Compute predicted values and derivatives
> grade <- 2:8
> myd <- data.frame(grade, fitted = eval(quad),
    slp = eval(my.deriv))
> myd

  grade   fitted       slp
1     2 171.7570 12.388953
2     3 183.3069 10.710849
3     4 193.1787  9.032745
4     5 201.3724  7.354642
5     6 207.8879  5.676538
6     7 212.7254  3.998434
7     8 215.8848  2.320330
```

Compute intercept for tangent line

$$\hat{y}_{ij} = \hat{\beta}_0 + \hat{\beta}_1(\text{grade5}_{ij}) + \hat{\beta}_2(\text{grade5}^2_{ij})$$

Derivative w.r.t. `grade5` is

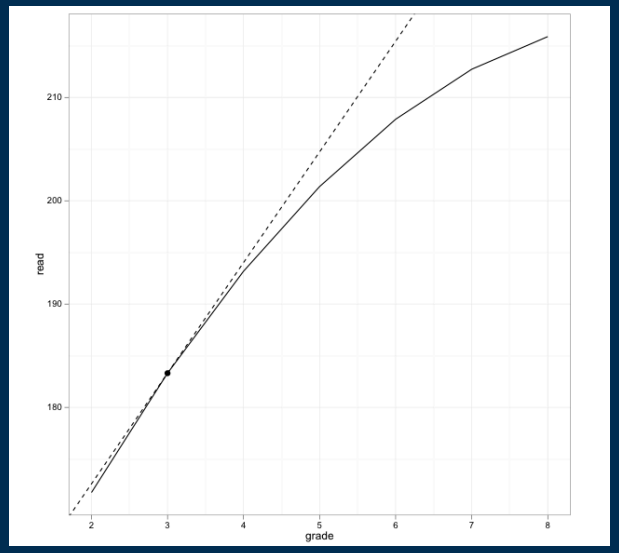
$$\hat{y}'_{ij} = \hat{\beta}_1 + 2\hat{\beta}_2(\text{grade5}_{ij})$$

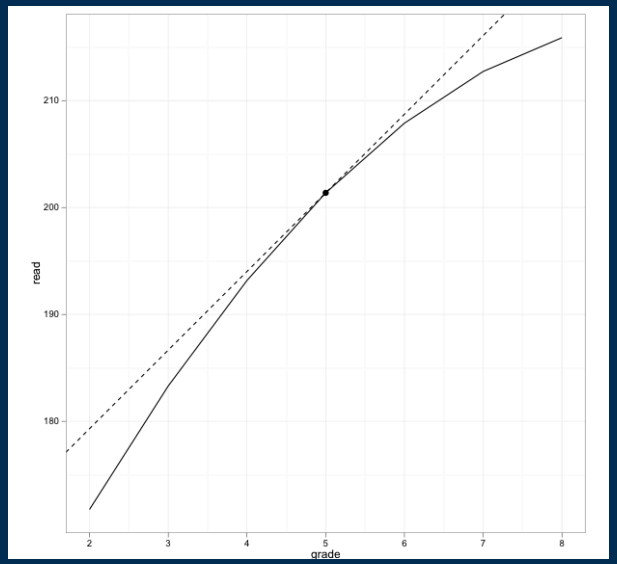
Solving for intercept

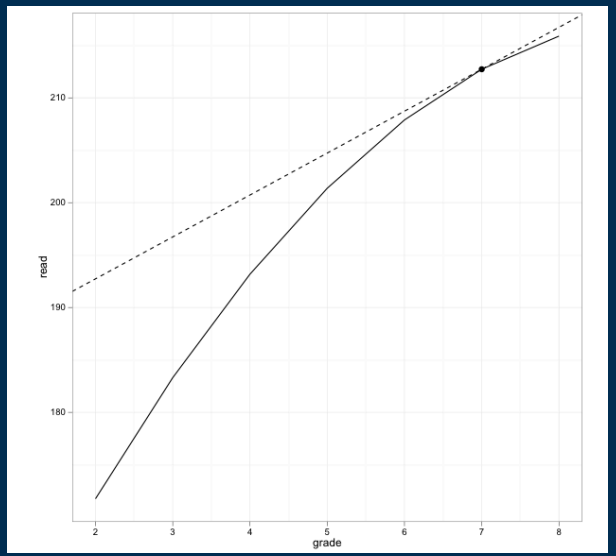
$$\hat{\beta}_1 = \hat{y}'_{ij} - 2\hat{\beta}_2(\text{grade5}_{ij})$$

slp values in myd

```
## Compute intercepts
> myd$int <- with(myd, fitted - slp * grade)

## Plot for grade 3
>  ggplot(data = myd[grade == 3, ], aes(x = grade, y = fitted)) +
    geom_line(data = myd, aes(x = grade, y = fitted)) +
    geom_abline(aes(intercept = int, slope = slp), linetype = 2) +
    geom_point(aes(x = grade, y = fitted), size = 3) +
    ylab("read") +
    opts(aspect.ratio = 1)
```







The solid line is the fitted curve, and the dashed line is the tangent. The slope of the tangent line decreases as grade increases.

Similar plots can be constructed for higher order polynomials

- Consider the cubic polynomial

$$E(y_{ij}) = \beta_0 + \beta_1(\text{grade5}_{ij}) + \beta_2(\text{grade5}_{ij}^2) + \beta_3(\text{grade5}_{ij}^3)$$

Rewrite by factoring out grade

$$E(y_{ij}) = \beta_0 + \left[\beta_1 + \beta_2(\text{grade5}_{ij}) + \beta_2(\text{grade5}_{ij}^2)\right](\text{grade5}_{ij})$$

Relationship between grade and the response is a nonlinear function of grade.

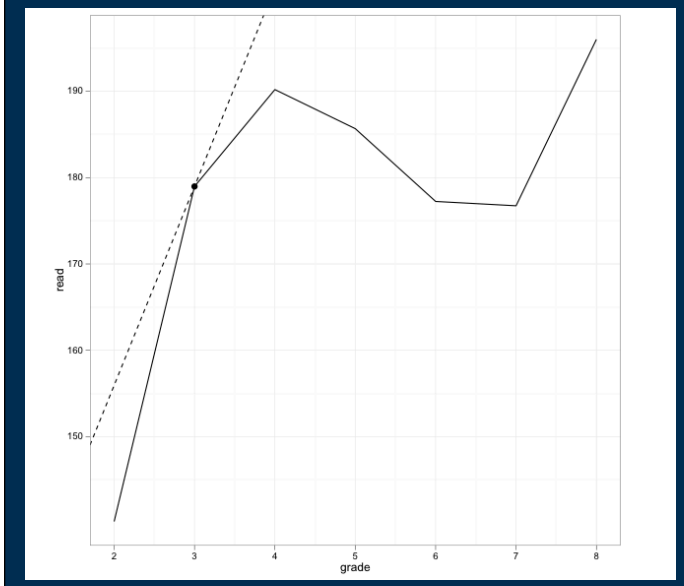
```
> cubic <- expression(B0 + B1 * grade + B2 * grade ^ 2 +
    B3 * grade ^ 3)
> my.deriv <- D(cubic, "grade")
> my.deriv

B1 + B2 * (2 * grade) + B3 * (3 * grade^2)
```

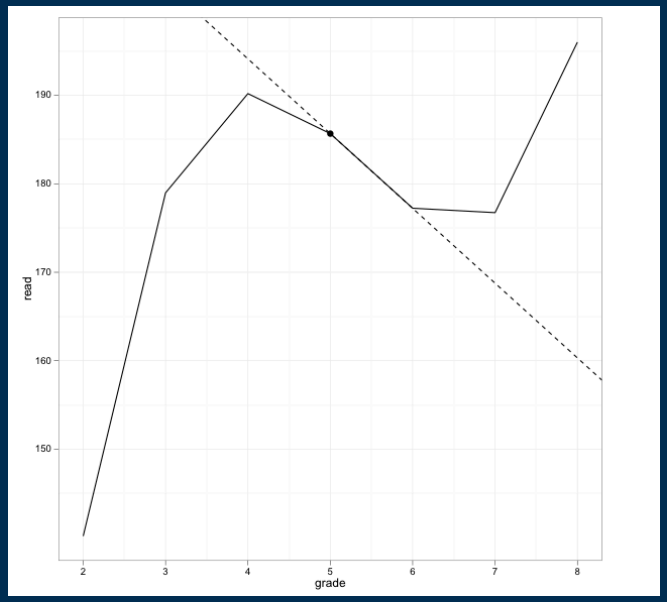
Result shows that the slope of the tangent line is a nonlinear function of grade. **Therefore, the slope can increase and decrease as grade increases.**

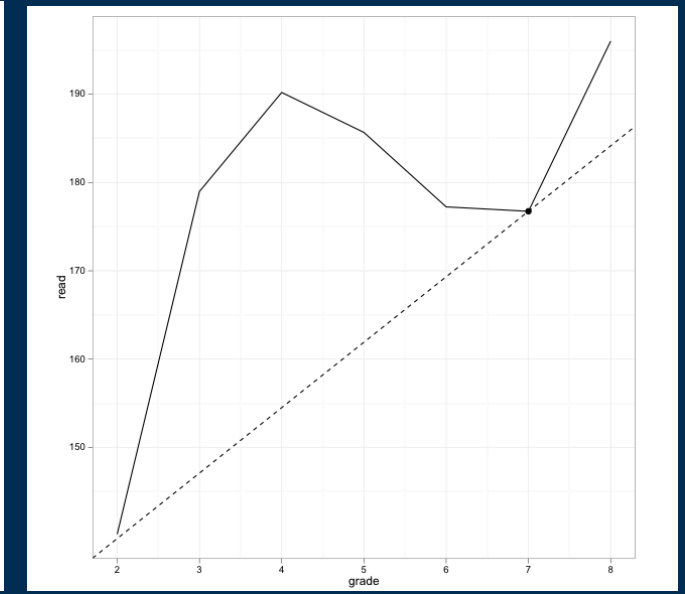
- Consider first subject's change curve
  - Estimation with `lm()` since it is single subject
  - Goal = description with focus on regression weights
  - Argument `subset=` included in `lm()`

```
## Estimate fitted model
> lm.1 <- lm(read ~ grade + I(grade ^ 2) + I(grade ^ 3),
    data = mplsL, subset = (subid == 1))

> myfe <- coef(lm.1)

> for(i in 1:4){
     assign(paste("B", i-1, sep = ""), myfe[i])
     }

> print(c(B0, B1, B2, B3))

(Intercept)       grade  I(grade^2)  I(grade^3)
 -67.428571  158.992063  -31.535714    1.972222

## Compute fitted values, derivatives, and intercepts
> grade <- 2:8
> myd <- data.frame(grade, fitted = eval(cubic),
    slp = eval(my.deriv))
> myd$int <- with(myd, fitted - slp * grade)
```

```
> myd

  grade   fitted       slp       int
1     2 140.1905 56.515873  27.15873
2     3 178.9762 23.027778 109.89286
3     4 190.1905  1.373016 184.69841
4     5 185.6667 -8.448413 227.90873
5     6 177.2381 -6.436508 215.85714
6     7 176.7381  7.408730 124.87698
7     8 196.0000 33.087302 -68.69841

## Plot grade 3
> ggplot(data = myd[grade == 3, ], aes(x = grade, y = fitted)) +
    geom_line(data = myd, aes(x = grade, y = fitted)) +
    geom_abline(aes(intercept = int, slope = slp), linetype = 2) +
    geom_point(aes(x = grade, y = fitted), size = 3) +
    ylab("read") +
    opts(aspect.ratio = 1
```







The solid line is the fitted curve, and the dashed line is the tangent. The slope of the tangent changes as grade increases, alternating between a positive slope (grades 3 and 7) and a negative slope (grade 5).

Mean Corrected Polynomials

- Raw polynomial terms tend to be highly correlated
- As order of polynomial increases, variance associated with the polynomial term tends to decrease
  - Variance associated with b_{2i} tends to be less than that associated with b_{1i} and b_{0i}
- Random effects of polynomial terms also tend to be highly correlated
- Small variance components alonf with highly correlated RE lead to estimation issues
- One solution is to subtract out mean from the time predictor before creating polynomial terms (*mean correcting*)
- Mean correcting anchors intercept at mean of time predictor

Orthogonal Polynomials

- Mean correcting lowers correlation between some terms, but not others
- Alternative to mean correcting is to create orthogonal polynomials
  - Represents same effects as raw polynomials, but correlation between terms is 0
  - Created by applying a transformation to the set of raw polynomials
- In R we use the `poly()` function

- Consider first subject

```
## Get subject 1
> p <- subset(mplsL, subid == 1, select = -c(read, dadv))

## Create raw polynomial terms
> p$rawp <- poly(p$grade, 3, raw = TRUE)

## Create mean corrected polynomial terms
> p$mcor <- poly(p$grade - mean(p$grade), 3, raw = TRUE)

## Create orthogonal polynomial terms
> p$orth <- poly(p$grade, 3)
```

```
> round(p[ ,-1], 4)

  grade rawp.1 rawp.2 rawp.3 mcor.1 mcor.2 mcor.3  orth.1  orth.2
1     2      2      4      8     -3      9    -27 -0.5669  0.5455
2     3      3      9     27     -2      4     -8 -0.3780  0.0000
3     4      4     16     64     -1      1     -1 -0.1890 -0.3273
4     5      5     25    125      0      0      0  0.0000 -0.4364
5     6      6     36    216      1      1      1  0.1890 -0.3273
6     7      7     49    343      2      4      8  0.3780  0.0000
7     8      8     64    512      3      9     27  0.5669  0.5455
   orth.3
1 -0.4082
2  0.4082
3  0.4082
4  0.0000
5 -0.4082
6 -0.4082
7  0.4082
```

```
## Raw
> round(cor(p$rawp), 2)
     1    2    3
1 1.00 0.99 0.95
2 0.99 1.00 0.99
3 0.95 0.99 1.00

## Mean corrected
> round(cor(p$mcor), 2)
     1 2    3
1 1.00 0 0.93
2 0.00 1 0.00
3 0.93 0 1.00

## Orthogonal
> round(cor(p$orth), 2)
  1 2 3
1 1 0 0
2 0 1 0
3 0 0 1
```

- `poly()` function can be used in `lm()` and `lmer()` functions
- Labeling in the output will consist of the syntax used to create the polynomials
  - Can be cryptic
- Save polynomials to data frame
  - This way single label of data frame name specifies the group of polynomials
  - Individual polynomials are accessed using indexing

```
## Create polynomials
> mplsL$rp <- poly(mplsL$grade - 2, degree = 2, raw = TRUE)
> mplsL$mc <- poly(mplsL$grade - mean(mplsL$grade),
    degree = 2, raw = TRUE)
> mplsL$op <- poly(mplsL$grade, degree = 2)
```

- Estimate models
  - Each has two RE
  - Same scaling used in RE as in FE

```
## Estimate models
> lmer.raw <- lmer(read ~ rp + (rp[ , 1] | subid),
    data = mplsL, REML = FALSE)
> lmer.mcor <- lmer(read ~ mc + (mc[ , 1] | subid),
    data = mplsL, REML = FALSE)
> lmer.orth <- lmer(read ~ op + (op[ , 1] | subid),
    data = mplsL, REML = FALSE)
```

```
## Raw
> round(summary(lmer.raw)@coefs, 4)
             Estimate Std. Error t value
(Intercept) 171.7570     3.3086 51.9130
rp1          12.3890     1.4021  8.8361
rp2          -0.8391     0.2193 -3.8260

## Mean corrected
> round(summary(lmer.mcor)@coefs, 4)
             Estimate Std. Error t value
(Intercept) 201.3724     3.3018 60.9891
mc1           7.3546     0.6501 11.3125
mc2          -0.8391     0.2193 -3.8260

## Orthogonal
> round(summary(lmer.orth)@coefs, 4)
             Estimate Std. Error t value
(Intercept) 198.0161     3.2093 61.7007
op1         182.5371    16.1360 11.3124
op2         -36.0697     9.4275 -3.8260
```

```
## Raw
> print(summary(lmer.raw)@REmat, quote = FALSE)
 Groups   Name        Variance Std.Dev. Corr
 subid    (Intercept) 184.6118 13.5872
          rp[, 1]       6.0792  2.4656  -0.121
 Residual              72.7715  8.5306

## Mean corrected
> print(summary(lmer.mcor)@REmat, quote = FALSE)
 Groups   Name        Variance Std.Dev. Corr
 subid    (Intercept) 215.0901 14.6660
          mc[, 1]       6.0791  2.4656  0.393
 Residual              72.7715  8.5306

## Orthogonal
> print(summary(lmer.orth)@REmat, quote = FALSE)
 Groups   Name        Variance Std.Dev. Corr
 subid    (Intercept)  215.091 14.6660
          op[, 1]     3744.761 61.1945  0.393
 Residual               72.772  8.5306
```

```
## Raw
> print(summary(lmer.raw)@AICtab, quote = FALSE)
      AIC      BIC    logLik deviance  REMLdev
 1142.437 1163.322 -564.2183 1128.437 1124.581

## Mean corrected
> print(summary(lmer.mcor)@AICtab, quote = FALSE)
      AIC      BIC    logLik deviance  REMLdev
 1142.437 1163.322 -564.2183 1128.437 1124.581

## Orthogonal
> print(summary(lmer.orth)@AICtab, quote = FALSE)
      AIC      BIC    logLik deviance  REMLdev
 1142.437 1163.322 -564.2183 1128.437 1110.636
```

Step Up Analysis with Orthogonal Polynomials

```
## Construct orthogonal polynomials
> op1 <- poly(mplsL$grade, 1)
> op2 <- poly(mplsL$grade, 2)
> op3 <- poly(mplsL$grade, 3)

## Estimate the models
> l.out <- lmer(read ~ op1 + (op1 | subid), data = mplsL,
      REML = FALSE)
> q.out <- lmer(read ~ op2 + (op2[ ,1] | subid),
      data = mplsL, REML = FALSE)
> c.out <- lmer(read ~ op3 + (op3[ ,1] | subid),
      data = mplsL, REML = FALSE)
```

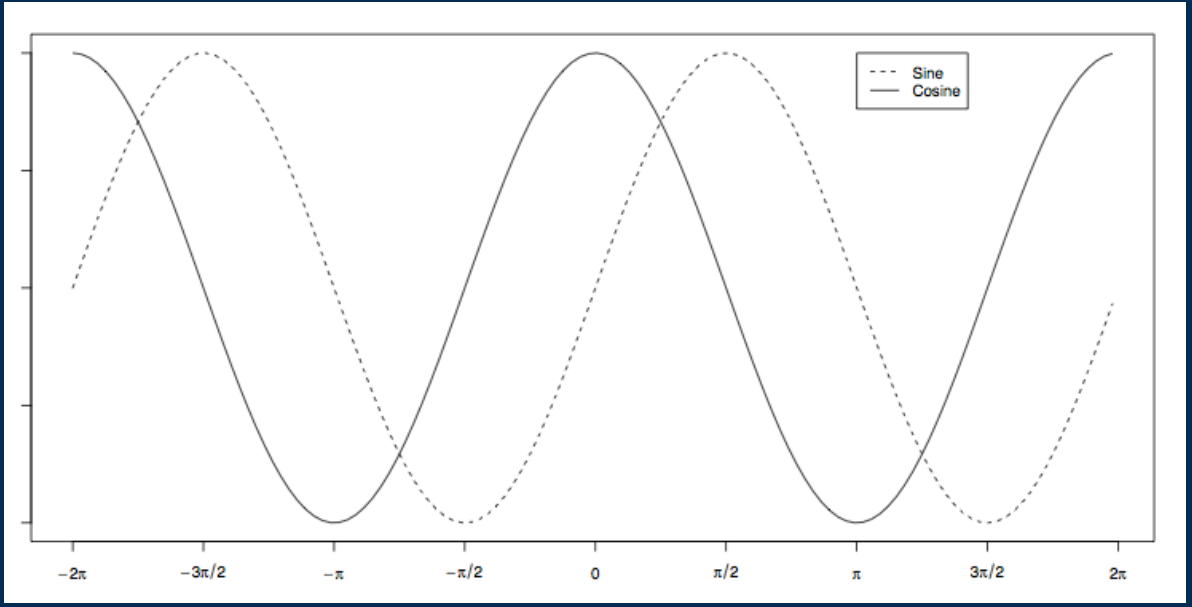
```
## Model fit
> print(aictab(list(l.out, q.out, c.out),
    c("linear", "quadratic", "cubic")), LL = FALSE)

Model selection based on AICc :

          K    AICc Delta_AICc AICcWt Cum.Wt
quadratic 7 1143.25       0.00   0.73   0.73
cubic     8 1145.22       1.97   0.27   1.00
linear    6 1154.87      11.62   0.00   1.00

## LRT
> anova(l.out, q.out, c.out)
Data: mplsL
Models:
l.out: read ~ op1 + (op1 | subid)
q.out: read ~ op2 + (op2[, 1] | subid)
c.out: read ~ op3 + (op3[, 1] | subid)
      Df    AIC    BIC  logLik   Chisq Chi Df Pr(>Chisq)
l.out  6 1154.3 1172.2 -571.13
q.out  7 1142.4 1163.3 -564.22 13.8268      1  0.0002005 ***
c.out  8 1144.2 1168.0 -564.08  0.2707      1  0.6028941
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Three Alternative Nonlinear Models

- Trigonometric Functions
  - Useful when there are period effects
  - Seasonal variation
- Fractional Polynomials
  - Similar curve shapes as polynomials, but with fewer parameters
  - Helpful when parsimony is a high priority
- Spline Transformations
  - Useful when the study involves landmark time points that are associated with changes in the direction of the trajectory

Trigonometric Functions

- Change curves are "wavy" or show oscillation



- Sine and cosine functions are periodic
- Can be used with lower order polynomials to account for waves in the trajectories

- Consider Level 1 model

$$y_{ij} = \beta^*_{0i} + \beta^*_{1i}(\text{c.grade}_{ij}) + \beta^*_{2i}\left[cos(\text{c.grade}_{ij})\right] + \epsilon_{ij}$$

- where *cos*(.) is the cosine transformation
- *c.grade* is a transformed grade predictor (explained momentarily)
- Note that the sine transformation, *sin*(.) can be used in place of *cos*(.)

It is useful to have the time predictor between certain values, namely 0 and 360 (or 0 and 2π radians).

$$\text{c.grade}_{ij} = 2\pi \left(\frac{\text{grade}_{ij} - min(\text{grade}_{ij})}{max\left[\text{grade}_{ij} - min(\text{grade}_{ij})\right]} \right)$$

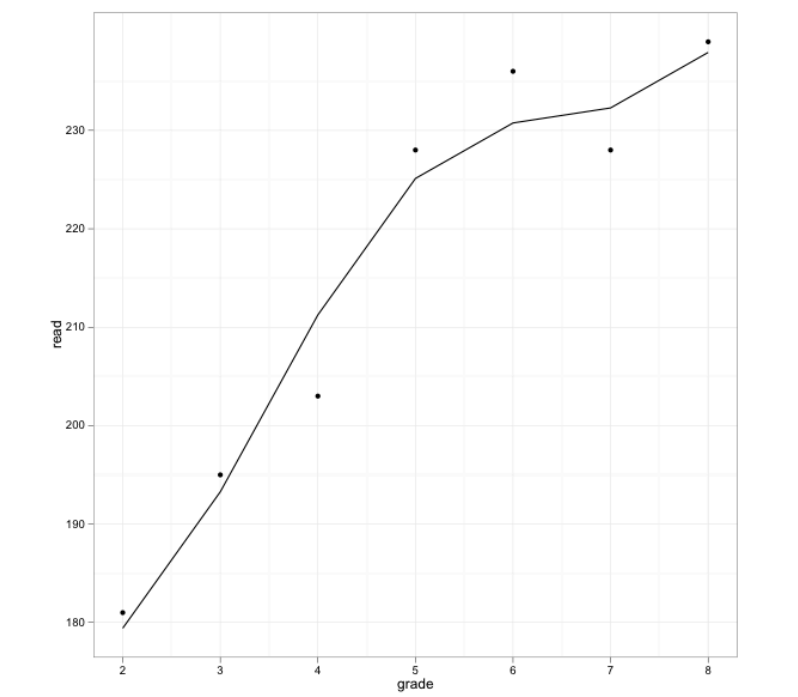
```
> mplsL$c.grade <- {2 * pi * (mplsL$grade - min(mplsL$grade)) /
    max(mplsL$grade - min(mplsL$grade))}

> head(data.frame(subid = mplsL$subid, grade = mplsL$grade,
    c.grade = mplsL$c.grade), n = 7)
  subid grade  c.grade
1     1     2 0.000000
2     1     3 1.047198
3     1     4 2.094395
4     1     5 3.141593
5     1     6 4.188790
6     1     7 5.235988
7     1     8 6.283185
```

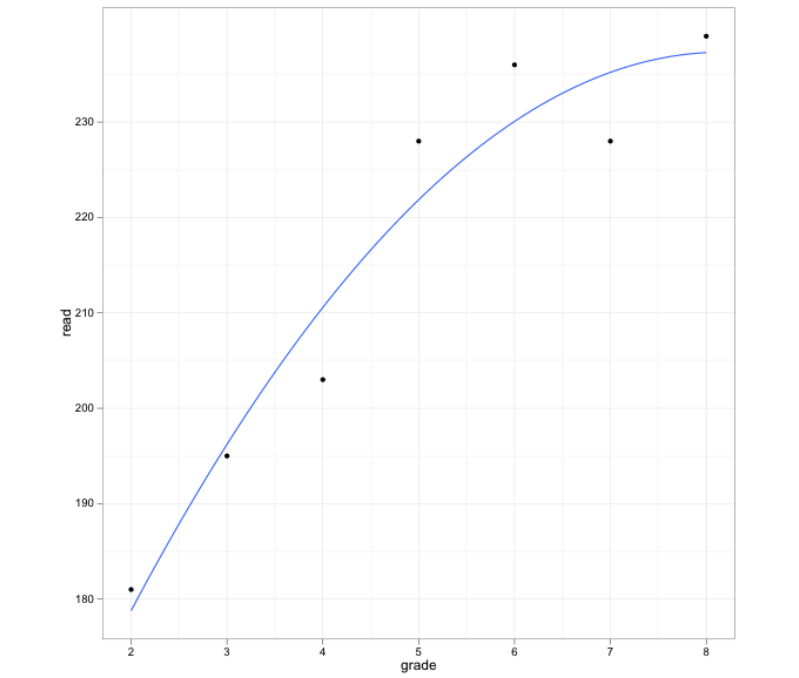
- To gain insight into linear-cosine model we examine derivative

```
## Derivative of linear-cosine model
> cos <- expression(B0 + B1 * c.grade + B2 * cos(c.grade))
> my.deriv <- D(cos, "c.grade")
> my.deriv

B1 - B2 * sin(c.grade)
```

The slope of the tangent line is a nonlinear function of `c.grade`. This means the slope can vary in sign and value as grade increases.

```
## Get subject 13's data
> subid13 <- with(mplsL[mplsL$subid == 13, ],
    data.frame(read, grade, c.grade))

## Estimate linear-cosine model.
> lm.cos <- lm(read ~ c.grade + I(cos(c.grade)), subid13)

## Create data frame for graphing.
> plotdata <- data.frame(read = subid13$read,
    grade = subid13$grade, fitted = fitted(lm.cos))

## Linear-cosine graph.
> ggplot(data = plotdata, aes(x = grade, y = read)) +
    geom_point() +
    geom_line(aes(y = fitted)) +
    opts(aspect.ratio = 1)

## Quadratic polynomial graph
> ggplot(data = plotdata, aes(x = grade, y = read)) +
    geom_point() +
    opts(aspect.ratio = 1) +
    stat_smooth(method = "lm", formula = y ~ poly(x, 2), se = FALSE)
```





- Linear-cosine fitted curve is monotonically increasing, but wavy
- Similar convexity to that of the quadratic polynomial up to grade 7
- unlike the quadratic polynomial, the linear-cosine curve continues to increase after grade 7

- Preferable to choose between the two models based on theoretical considerations
  - If there is a substantive reason to believe that the reading scores will continue to increase after grade 7, then the linear-cosine model is appropriate.
  - Otherwise, the quadratic polynomial is favorable.

```
## AICc to empirically select model
> quad <- lm(read ~ poly(grade, 2), data = subid13)
> cosine <- lm(read ~ c.grade + I(cos(c.grade)), data = subid13)
> sine <- lm(read ~ c.grade + I(sin(c.grade)), data = subid13)
> print(aictab(list(quad, cosine, sine),
    c("quad", "cosine", "sine")), LL = FALSE)

Model selection based on AICc :

       K  AICc Delta_AICc AICcWt Cum.Wt
cosine 4 68.26       0.00   0.78   0.78
quad   4 71.03       2.77   0.20   0.98
sine   4 75.62       7.36   0.02   1.00
```

Consider LMER model

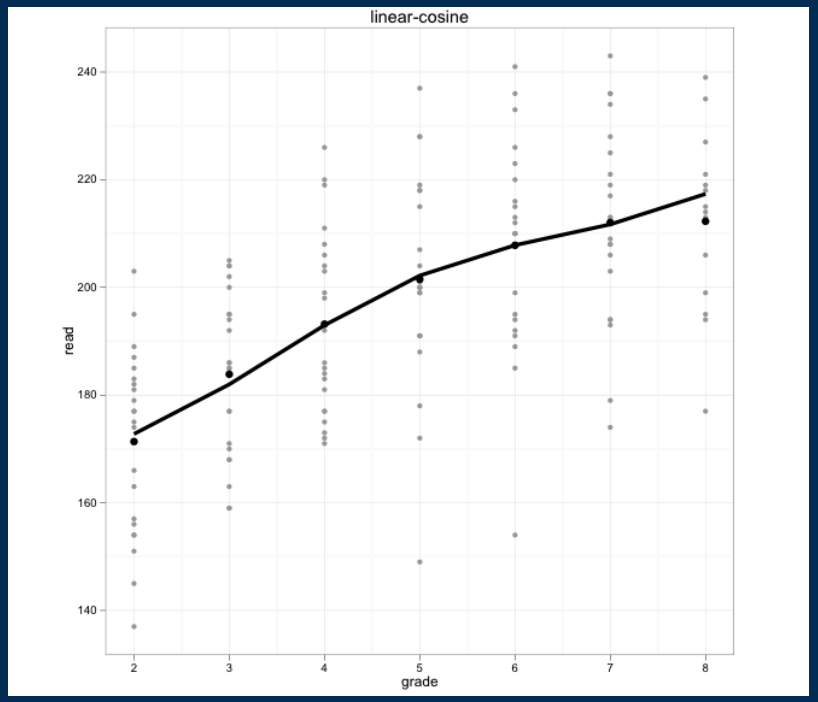
$$y_{ij} = (\beta_0 + b_{0i}) + (\beta_1 + b_{1i})(\texttt{c.grade}_{ij}) + \beta_2 \left[cos(\texttt{c.grade}_{ij})\right] + \epsilon_{ij}$$

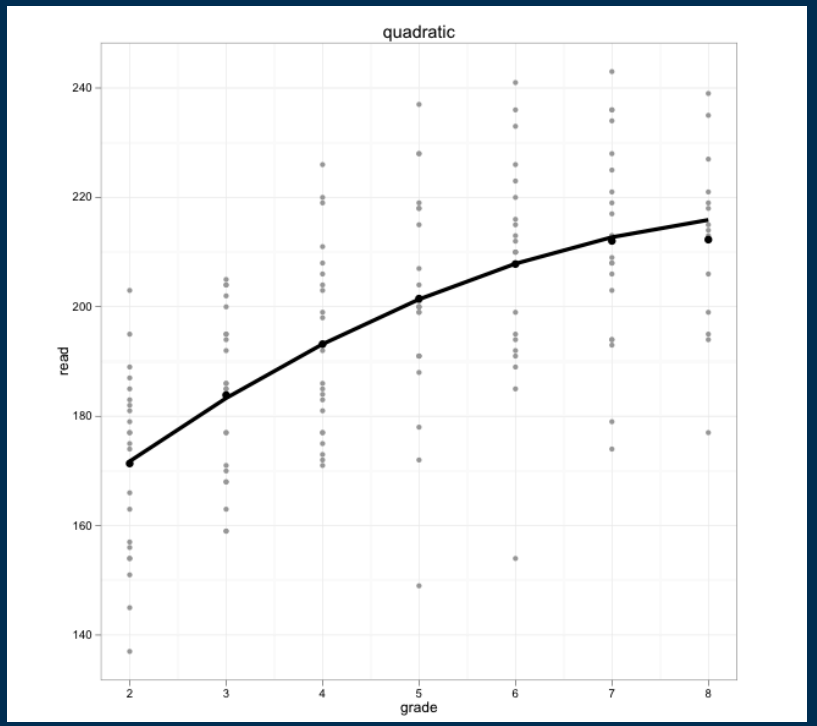
Random effects only for polynomial part of the model

```
## Fit LMER models
> quad <- lmer(read ~ poly(grade, 2) +
    (poly(grade, 1) | subid), data = mplsL, REML = FALSE)
> lincos <- lmer(read ~ c.grade + I(cos(c.grade)) +
    (c.grade | subid), data = mplsL, REML = FALSE)

## Compare models
> print(aictab(list(quad, lincos),
    c("Quad", "Linear-Cosine")), LL = FALSE)

Model selection based on AICc :

              K    AICc Delta_AICc AICcWt Cum.Wt
Quad          7 1143.25        0.0   0.68   0.68
Linear-Cosine 7 1144.75        1.5   0.32   1.00
```

```
## Estimate "dummy" model
> lmer.d <- lmer(read ~ 1 + grade + (1 + grade | subid),
      data = mplsL, REML = FALSE)

## Create plotting data set
> plotdata <- data.frame(
    grade = lmer.d@frame$grade,
    read = lmer.d@frame$read,
    q.pred = model.matrix(quad) %*% fixef(quad),
    c.pred = model.matrix(lincos) %*% fixef(lincos)
    )

## Plot linear-cosine model
> ggplot(data = plotdata, aes(x = grade, y = read)) +
    geom_point(colour = "grey60") +
    stat_summary(fun.y = mean, geom = "point", size = 3) +
    opts(aspect.ratio = 1) +
    stat_summary(fun.y = mean, geom = "line", lwd = 1.5,
       aes(y = c.pred)) +
    opts(title = "linear-cosine")
```

```
## Plot quadratic model
> ggplot(data = plotdata, aes(x = grade, y = read)) +
    geom_point(colour = "grey60") +
    stat_summary(fun.y = mean, geom = "point", size = 3) +
    opts(aspect.ratio = 1) +
    stat_summary(fun.y = mean, geom = "line", lwd = 1.5,
       aes(y = q.pred)) +
    opts(title = "quadratic")
```





Finally, when the amount of short-term oscillation is relatively large, the **sine and cosine terms can be used together**. The model is useful when there is a global tendency to increase or decrease, and the oscillation is particularly strong.

$$y_{ij} = (\beta_0 + b_{0i}) + (\beta_1 + b_{1i})(\texttt{c.grade}_{ij}) + \beta_2 \left[cos(\texttt{c.grade}_{ij})\right] + \beta_3 \left[sin(\texttt{c.grade}_{ij})\right] + \epsilon_{ij}$$

Fractional Polynomials

- Similar to the raw polynomials in that transformations of the time metric are used to account for nonlinear change
- Power terms need not be the counting numbers, 1, 2, 3, ...
    - Counting numbers are among the possible power transformations, so raw polynomials are special cases of FPs
- FPs advantageous when the number of time points is limited, but theory suggests nonlinear trends
    - Can help avoid saturated models
        - Similar curve shapes as polynomials, but with fewer parameters
- FPs are also useful when a curve appears to plateau
- FPs can also have less extreme behavior at the edges of the observed time span making them more attractive for extrapolation
- Discussion limited to first order and second order FPs

First Order Fractional Polynomials

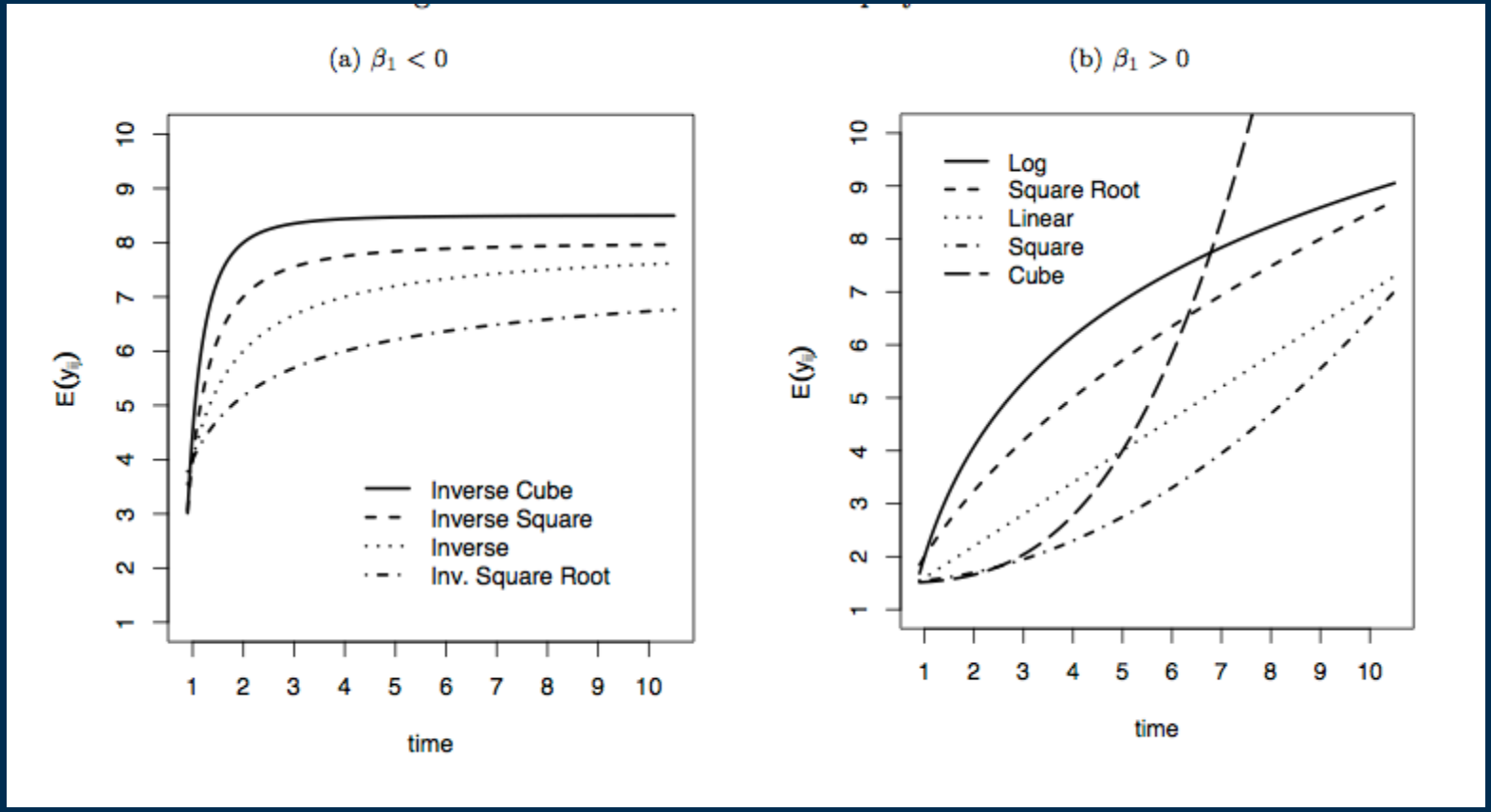
- Consists of monotonic change curves
  - Monotonic refers to curves that are constantly increasing or decreasing, but not necessarily at a constant rate
  - Linear curve being a special case
- First order FPs models have a single time transformation
  - Power transformation or the natural log transformation
- Suppose the power value or exponent is denoted by *a*. First order FP for LMER is defined as

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

- Same transformation is used in the fixed and random effects portions of the model

- Power value, *a*, is limited to one of the following elements

Element	Transformation	Description
-3	$\frac{1}{\texttt{grade}_{ij}^3}$	Inverse cube
-2	$\frac{1}{\texttt{grade}_{ij}^2}$	Inverse square
-1	$\frac{1}{\texttt{grade}_{ij}}$	Inverse
-0.5	$\frac{1}{\sqrt{\texttt{grade}_{ij}}}$	Inverse square root
0	$log(\texttt{grade}_{ij})$	Natural log
0.5	$\sqrt{\texttt{grade}_{ij}}$	Square root
1	$\texttt{grade}_{ij}$	Linear (no transformation)
2	$\texttt{grade}_{ij}^2$	Square
3	$\texttt{grade}_{ij}^3$	Cube


(a) $\beta_1 < 0$
(b) $\beta_1 > 0$
$E(y_{ij})$
time
Inverse Cube
Inverse Square
Inverse
Inv. Square Root
Log
Square Root
Linear
Square
Cube

Fixed Effects Interpretation

- Intercept, β_0, has the same interpretation as in the polynomial model
  - Predicted mean value when the time score is 0
- Different transformations (controlled by *a*) result in the anchoring of the intercept at different time points
- Suppose a subject's grade values are 1, 2, 3, and 4 and $a = 0.5$

$$\sqrt{\texttt{grade}_{i1}} = \sqrt{1} = 1$$

β_0 is anchored at the first time point

- Now suppose $a = 0$

$$log(\texttt{grade}_{i1}) = log(1) = 0$$

β_0 is anchored at the unobserved time point previous to the first

- Main issue is that values of β_0 and $Var(b_{0i})$ are not consistent

- Slope, β_1, is **not** the linear slope, except when $a = 1$
- Consider the symbolic derivative for the models with a = 1, −1, 0

```
> D(expression(B0 + B1 * grade), "grade")
B1
```

For the linear model, the tangent line is identical to the regression line. The derivative for the $grade_{ij}$ linear model is β_1, and does not include grade. Thus, the slope of the tangent line is constant for all grades.

```
> D(expression(B0 + B1 * grade ^ -1), "grade")
-(B1 * grade^-(1 + 1))

> D(expression(B0 + B1 * log(grade)), "grade")
B1 * (1/grade)
```

For the other two transformations, the derivative is a function of grade. This means the slope of the tangent is not constant over time, and the trend lines for the original functions are nonlinear.

Rather than attempt to interpret β_1 directly when $a \neq 1$, it is perhaps more useful to discuss the slope of the tangent. As an illustration, consider the derivative for the log model.

```
> D(expression(B0 + B1 * log(grade)), "grade")
B1 * (1/grade)
```

$$\beta_1 \left(\frac{1}{\texttt{grade}_{ij}} \right)$$

When $\beta_1 > 0$, the curve is monotonically increasing, but the slope of the tangent decreases as grade increases.

When $\beta_1 < 0$, the curve is monotonically decreasing, but the slope of the tangent increases as grade increases.

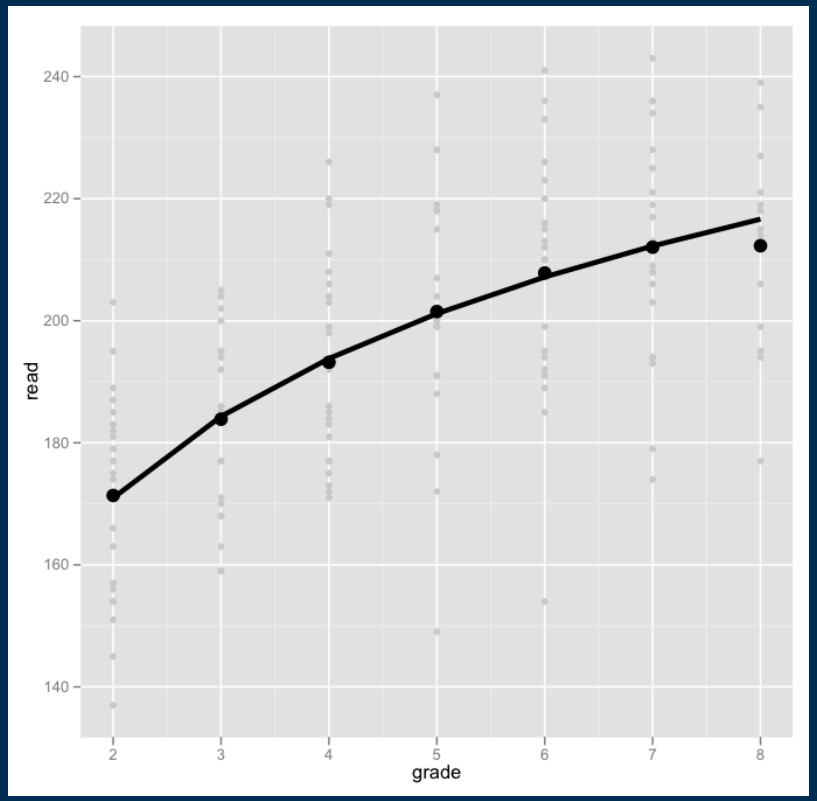
Selection of FP

- Which of the nine FPs should be used?
  - Chosen based on past research
  - Curve appears appropriate for the data at hand
  - Data-driven
    - Fit all nine transformations
    - Determine model fit
    - Compare to linear model
    - Might also be compared to quadratic polynomial since quadratic model can account for monotonic change over the observed time period, much like a first order FP

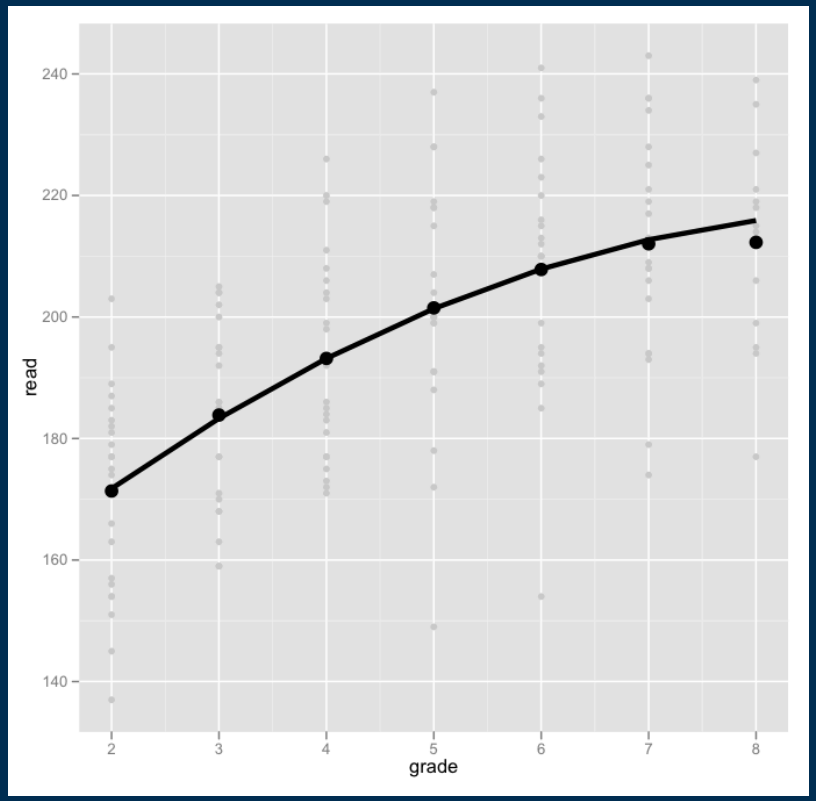
```
## Quadratic Model
> quad <-  lmer(read ~ poly(grade, 2, raw = TRUE) + (grade | subid),
    data = mplsL, REML = FALSE)

## 1st Order FP models
> n3 <-  lmer(read ~ I(grade ^ -3) + (I(grade ^ -3) | subid),
    data = mplsL, REML = FALSE)
> n2 <-  lmer(read ~ I(grade ^ -2) + (I(grade ^ -2) | subid),
    data = mplsL, REML = FALSE)
> n1 <-  lmer(read ~ I(grade ^ -1) + (I(grade ^ -1) | subid),
    data = mplsL, REML = FALSE)
> n05 <- lmer(read ~ I(grade ^ -0.5) + (I(grade ^ -0.5) | subid),
    data = mplsL, REML = FALSE)
> ze <-  lmer(read ~ I(log(grade))  + (I(log(grade)) | subid),
    data = mplsL, REML = FALSE)
> p05 <- lmer(read ~ I(grade ^ 0.5) + (I(grade ^ 0.5) | subid),
    data = mplsL, REML = FALSE)
> p1 <- lmer(read ~ grade + (grade | subid), data = mplsL,
    REML = FALSE)
> p2 <- lmer(read ~ I(grade ^ 2) + (I(grade ^ 2) | subid),
    data = mplsL, REML = FALSE)
> p3 <-  lmer(read ~ I(grade ^ 3) + (I(grade ^ 3) | subid),
    data = mplsL, REML = FALSE)
```

```
## Fit
> mymods <- list(n3, n2, n1, n05, ze, p05, p1, p2, p3, quad)
> mynames <- c("-3", "-2", "-1", "-.5", "0", ".5", "1", "2", "3",
    "Quad")
> print(aictab(mymods, mynames), LL = FALSE)

Model selection based on AICc :

     K    AICc Delta_AICc AICcWt Cum.Wt
0    6 1138.88       0.00   0.76   0.76
.5   6 1143.04       4.16   0.09   0.85
Quad 7 1143.25       4.37   0.09   0.94
-.5  6 1143.85       4.97   0.06   1.00
1    6 1154.87      15.99   0.00   1.00
-1   6 1155.69      16.81   0.00   1.00
-2   6 1184.87      45.99   0.00   1.00
2    6 1186.93      48.06   0.00   1.00
-3   6 1208.05      69.17   0.00   1.00
3    6 1215.00      76.12   0.00   1.00
```

```
## Graph data
> plotdata <- data.frame(
    read = p1@frame$read,
    grade = p1@frame$grade,
    ze.pred = model.matrix(ze) %*% fixef(ze),
    quad.pred = model.matrix(quad) %*% fixef(quad)
    )

## Plot of log transformation
> ggplot(data = plotdata, aes(x = grade, y = read)) +
    geom_point(colour = "grey80") +
    geom_line(aes(y = ze.pred), size = 1.5) +
    stat_summary(aes(y = read), fun.y = mean, geom = "point",
       size = 4) +
    opts(aspect.ratio = 1)

## Plot of quadratic
> ggplot(data = plotdata, aes(x = grade, y = read)) +
    geom_point(colour="grey80") +
    geom_line(aes(y = quad.pred), size = 1.5) +
    stat_summary(aes(y = read), fun.y = mean, geom = "point",
       size = 4) +
    opts(aspect.ratio = 1)
```



Fitted curve for the log transformation



Fitted curve for the quadratic polynomial

Second Order Fractional Polynomials

- For more complex patterns than monotonic curves, higher order FPs are used
- Second order FPs are similar to the quadratic polynomial in that, their curves all have a single extreme point
- They are either concave or convex with respect to the time axis
- Unlike the quadratic polynomial, second order FPs are asymmetric about their extreme point
- Second order FPs have two power terms, a and b
  - Constraint is imposed that $a \leq b$
  - Form of the model depends on whether $a < b$ or $a = b$

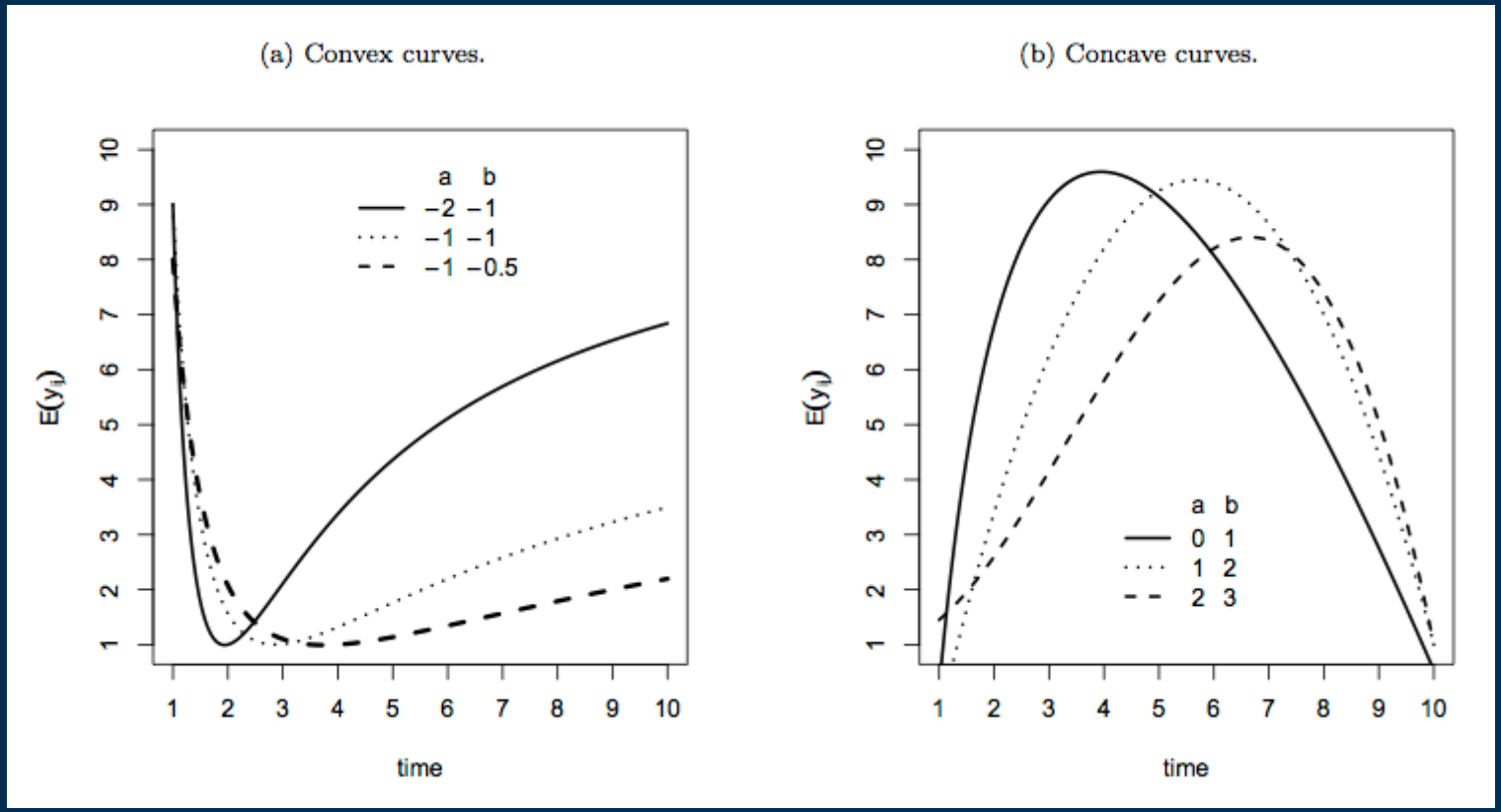
- When *a* < *b*, the second order model is

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

- The values of *a* and *b* are chosen from same set of values as before.
- When *a* = *b*, the second order model is

$$y_{ij} = (\beta_0 + b_{0i}) + (\beta_1 + b_{1i})(\texttt{grade}_{ij}^{a}) + \beta_2(\texttt{grade}_{ij}^{a})\left[log(\texttt{grade}_{ij})\right] + \epsilon_{ij}$$

- Based on the set of values for *a* and *b*, 36 models are possible.
- Quadratic polynomial is the special case of *a* = 1, *b* = 2



- Quadratic polynomial is symmetric about its extreme value whereas the other second order FPs are not
- Extent of asymmetry varies based on the combination of transformations and the values of the fixed effects

- Three FP models are considered
- Models having power values of
  - $a = -2, b = -1$ (inverse quadratic model)
  - $a = 0, b = 0$ (log–log square model)
  - $a = 1, b = 2$ (quadratic polynomial)
- For comparison purposes, the first order log model from the last section is also included

```
## Fit models
> n2n1 <- lmer(read ~ I(grade ^ -2) + I(grade ^ -1) +
     (I(grade ^ -2) | subid), data = mplsL, REML = FALSE)
> zeze <- lmer(read ~ I(log(grade)) + I(log(grade) ^ 2) +
     (I(log(grade)) | subid), data = mplsL, REML = FALSE)
```

```
## Model fit measures
> mymods <- list(ze, n2n1, zeze, quad)
> mynames <- c("( 0,   )", "(-2, -1)", "( 0,  0)", "( 1,  2)")
> print(aictab(mymods, mynames), LL = FALSE)

Model selection based on AICc :

         K    AICc Delta_AICc AICcWt Cum.Wt
( 0,   ) 6 1138.88       0.00   0.65   0.65
( 0,  0) 7 1141.05       2.17   0.22   0.87
( 1,  2) 7 1143.25       4.37   0.07   0.95
(-2, -1) 7 1143.90       5.02   0.05   1.00
```

Static Predictors

- Static predictors can be included in models with FPs
- Method of inclusion is the same as with conventional polynomials
- Consider the following multilevel model depiction, with a first order FP and the risk static predictor

$$
\begin{aligned}
y_{ij} &= \beta_{0i} + \beta_{2i}\left[log(\texttt{grade}_{ij})\right] + \epsilon_{ij} \\
\beta_{0i} &= \beta_0 + \beta_2(\texttt{dadv}_i) + b_{0i} \\
\beta_{2i} &= \beta_0 + \beta_3(\texttt{dadv}_i) + b_{1i}
\end{aligned}
$$

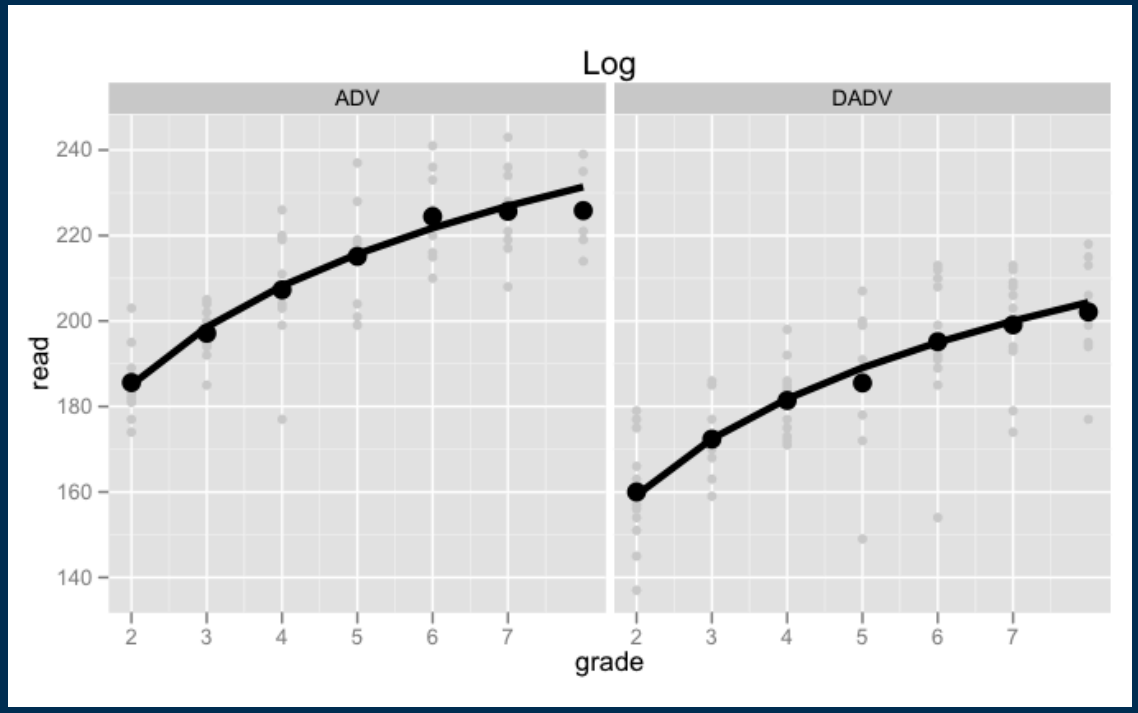
- By substitution the LMER is

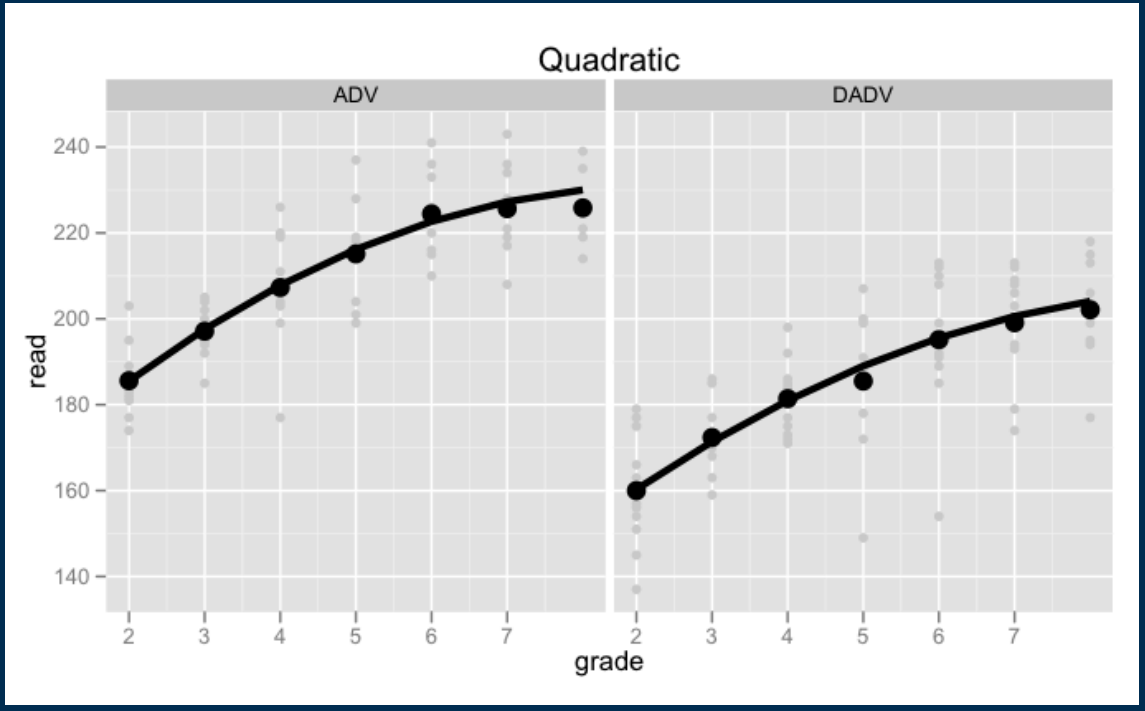
$$
\begin{aligned}
y_{ij} = (\beta_0 + b_{0i}) + (\beta_1 + b_{1i})\left[log(\texttt{grade}_{ij})\right] + \beta_2(\texttt{dadv}_i) \\
+ \beta_3\left[log(\texttt{grade}_{ij})\right](\texttt{dadv}_i) + \epsilon_{ij}
\end{aligned}
$$

- Interpretations of the parameters are similar to polynomial models
  - β_2 represents intercept differences
  - β_3 representing differences in trajectory
- Caveat is that when $a = 1$, the change curve is not linear
- Furthermore, when $\beta_3 \neq 0$, the change curves of the groups are not parallel

```
> quadrisk <- lmer(read ~ poly(grade, 2) * dadv +
    (poly(grade, 1) | subid), data = mplsL, REML = FALSE)
> logrisk <- lmer(read ~ I(log(grade)) * dadv +
    (I(log(grade)) | subid), data = mplsL, REML = FALSE)
> print(aictab(list(quadrisk, logrisk), c("Quad", "Log")),
    LL = FALSE)

Model selection based on AICc :

      K    AICc Delta_AICc AICcWt Cum.Wt
Log   8 1101.45       0.00   0.98   0.98
Quad 10 1108.91       7.45   0.02   1.00
```

```
## Plot of the fitted models
> plotdata <- data.frame(
      read = logrisk@frame$read,
      grade = exp(logrisk@frame[ ,2]),
      dadv = logrisk@frame$dadv,
      q.pred = model.matrix(quadrisk) %*% fixef(quadrisk),
      l.pred = model.matrix(logrisk) %*% fixef(logrisk)
      )

## logrisk model
> ggplot(data = plotdata, aes(x = grade, y = read)) +
      geom_point(color = "grey80") +
      stat_summary(fun.y = mean, geom = "point", size = 4) +
      geom_line(aes(y = l.pred), lwd = 1.5) +
      facet_grid(. ~ dadv) +
      opts(aspect.ratio = 1, title = "Log")


## quadrisk model
> ggplot(data = plotdata, aes(x = grade, y = read)) +
      geom_point(color = "grey80") +
      stat_summary(fun.y = mean, geom = "point", size = 4) +
      geom_line(aes(y = q.pred), lwd = 1.5) +
      facet_grid(. ~ dadv) +
      opts(aspect.ratio = 1, title = "Quadratic")
```


Log
ADV
DADV
240
220
200
180
160
140
read
2
3
4
5
6
7
grade


Quadratic
ADV
DADV
240
220
200
180
160
140
read
2
3
4
5
6
7
grade

Spline Models

- Local models are useful for situations in which the researcher wants to have different curve equations for different spans of time
    - For example, one particular model for the span of time prior to administering a treatment, and a different model after the treatment commences
- The local behavior of the curve varies based on the span of time
- Spline transformations considered are piecewise polynomials also known as *regression splines*
- Idea is to break up the observed time span into two or more pieces and fit a different polynomial model to each piece
- Most easily accomplished by constructing spline transformations in the data frame, and including them as additional predictors in `lmer()`
    - Transformations chosen so that the fitted curve is smooth and continuous

Linear Spline Models

- Fit a different straight line for each time span defined by the researcher
- Simplest linear spline breaks the time span into two pieces, defined by the split point of a single knot
  - Knot, denoted *k*, is constant among the subjects

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

where the spline transformation is

$$(\texttt{grade}_{ij} - k)_+ = \begin{cases} 0, & \text{if } \texttt{grade}_{ij} \leq k \\ \texttt{grade}_{ij} - k, & \text{if } \texttt{grade}_{ij} > k \end{cases}$$

- Suppose *k* = 6

$$y_{ij} = (\beta_0 + b_{0i}) + (\beta_1 + b_{1i})(\texttt{grade}_{ij}) + \beta_2(\texttt{grade}_{ij} - 6)_+ + \epsilon_{ij}$$

which is

$$y_{ij} = \begin{cases} (\beta_0 + b_{0i}) + (\beta_1 + b_{1i})(\texttt{grade}_{ij}) + \epsilon_{ij}, & \text{if } \texttt{grade}_{ij} \leq 6 \\ (\beta_0 + 6\beta_2 + b_{0i} - 6b_{2i}) + (\beta_1 + b_{1i} + \beta_2)(\texttt{grade}_{ij}) + \epsilon_{ij} & \text{if } \texttt{grade}_{ij} > 6 \end{cases}$$

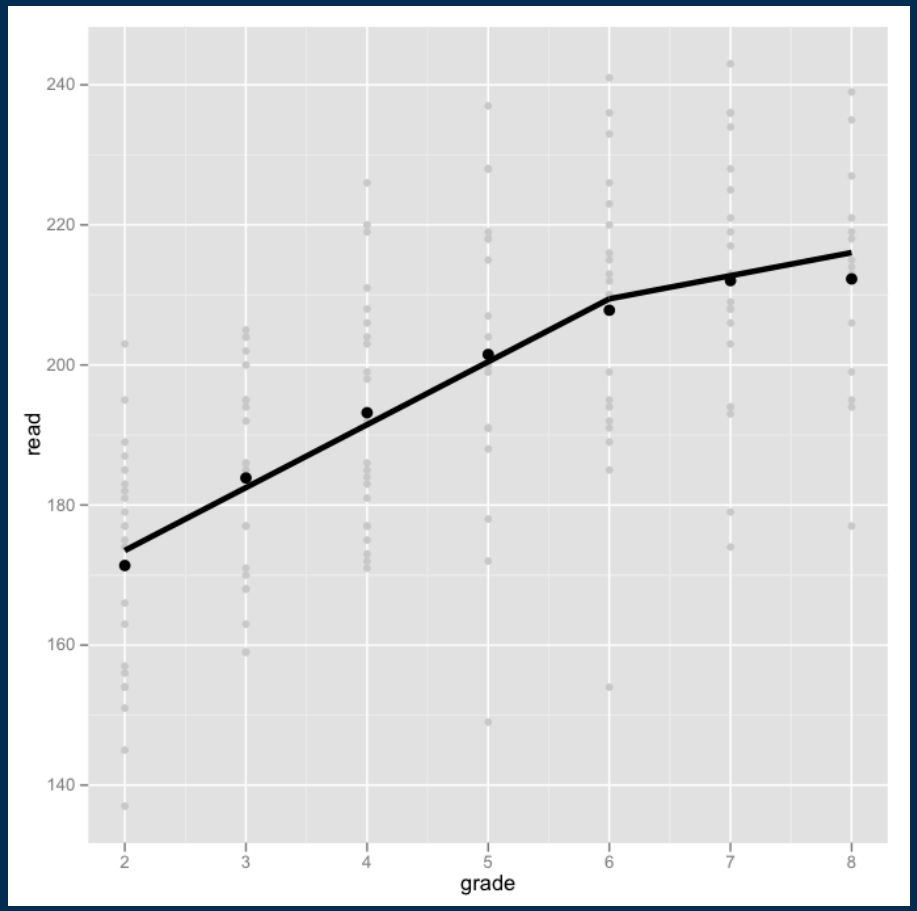
- This indicates that when $\beta_2 \neq 0$, the slope over the interval [2, 6] will be different than the slope over the interval (6, 8]
- If $\beta_2 = 0$, then the lower-order β-terms are sufficient, and the spline is not needed
- Thus, an evaluation of the estimate of β_2 is usually of prime interest

```
## Create spline terms within the data frame
> mplsL$spline1 <- ifelse(mplsL$grade > 6, mplsL$grade - 6, 0)
> head(data.frame(grade = mplsL$grade, spline1 = mplsL$spline1),
     n = 7)
  grade spline1
1     2       0
2     3       0
3     4       0
4     5       0
5     6       0
6     7       1
7     8       2

## Run lmer() with spline term as second predictor
> sp1 <- lmer(read ~ 1 + grade + spline1 + (1 + grade | subid),
     data = mplsL, REML = FALSE)
```

```
Linear mixed model fit by maximum likelihood
Formula: read ~ 1 + grade + spline1 + (1 + grade | subid)
   Data: mplsL
  AIC  BIC logLik deviance REMLdev
 1146 1167 -566.1     1132    1124
Random effects:
 Groups   Name        Variance Std.Dev. Corr
 subid    (Intercept) 222.1986 14.9063
          grade         5.9856  2.4465  -0.431
 Residual              75.4386  8.6855
Number of obs: 146, groups: subid, 22

Fixed effects:
            Estimate Std. Error t value
(Intercept) 155.5373     4.0103   38.78
grade         8.9845     0.7706   11.66
spline1      -5.6834     1.7505   -3.25

Correlation of Fixed Effects:
        (Intr) grade
grade   -0.654
spline1  0.362 -0.544
```

```
## Reduced model
> sp0 <- lmer(read ~ 1 + grade + (1 + grade | subid),
      data = mplsL, REML = FALSE)

## LRT
> anova(sp0, sp1)
Data: mplsL
Models:
sp0: read ~ 1 + grade + (1 + grade | subid)
sp1: read ~ 1 + grade + spline1 + (1 + grade | subid)
    Df    AIC    BIC  logLik  Chisq Chi Df Pr(>Chisq)
sp0  6 1154.3 1172.2 -571.13
sp1  7 1146.2 1167.0 -566.08 10.108      1   0.001476 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

## AICc
> print(aictab(list(sp1, sp0), c("Spline", "Linear")),
      LL = FALSE)

Model selection based on AICc :

       K    AICc Delta_AICc AICcWt Cum.Wt
Spline 7 1146.97        0.0   0.98   0.98
Linear 6 1154.87        7.9   0.02   1.00
```

```
## Create data frame for plot
> plotdata <- data.frame(
      read = sp1@frame$read,
      grade = sp1@frame$grade,
      pred = model.matrix(sp1) %*% fixef(sp1)
      )

## Plot the model
> ggplot(data = plotdata, aes(x = grade, y = read)) +
      geom_point(colour = "grey80") +
      stat_summary(fun.y = mean, geom = "point", size = 3) +
      geom_line(aes(y = pred), size = 1.5)
```



The fitted spline curve has a linear increase for the first piece (up to grade 6), and a smaller rate of increase for the second piece. The overall pattern is similar to the convexity of the quadratic model however, the splines are constrained to be straight lines.

Linear Spline Model with Multiple Knots

- Knots can occur at more than one time point
  - Knots at two time points, denoted as k_1 and k_2, with $k_1 < k_2$

$$y_{ij} = (\beta_0 + b_{0i}) + (\beta_1 + b_{1i})(\texttt{grade}_{ij}) + \beta_2(\texttt{grade}_{ij} - k_1)_+ + \beta_3(\texttt{grade}_{ij} - k_2)_+ + \epsilon_{ij}$$

where the spline transformation is

$$(\texttt{grade}_{ij} - k_1)_+ = \begin{cases} 0, & \text{if } \texttt{grade}_{ij} \leq k_1 \\ \texttt{grade}_{ij} - k_1, & \text{if } \texttt{grade}_{ij} > k_1 \end{cases}$$

$$(\texttt{grade}_{ij} - k_2)_+ = \begin{cases} 0, & \text{if } \texttt{grade}_{ij} \leq k_2 \\ \texttt{grade}_{ij} - k_2, & \text{if } \texttt{grade}_{ij} > k_2 \end{cases}$$

- Suppose knots are at $k_1 = 4$ and $k_2 = 6$, then the fitted change curve is

$$E(y_{ij}) = \beta_0 + \beta_1(\texttt{grade}_{ij}) + \beta_2(\texttt{grade}_{ij} - 4)_+ + \beta_3(\texttt{grade}_{ij} - 6)_+$$

The change curves are then

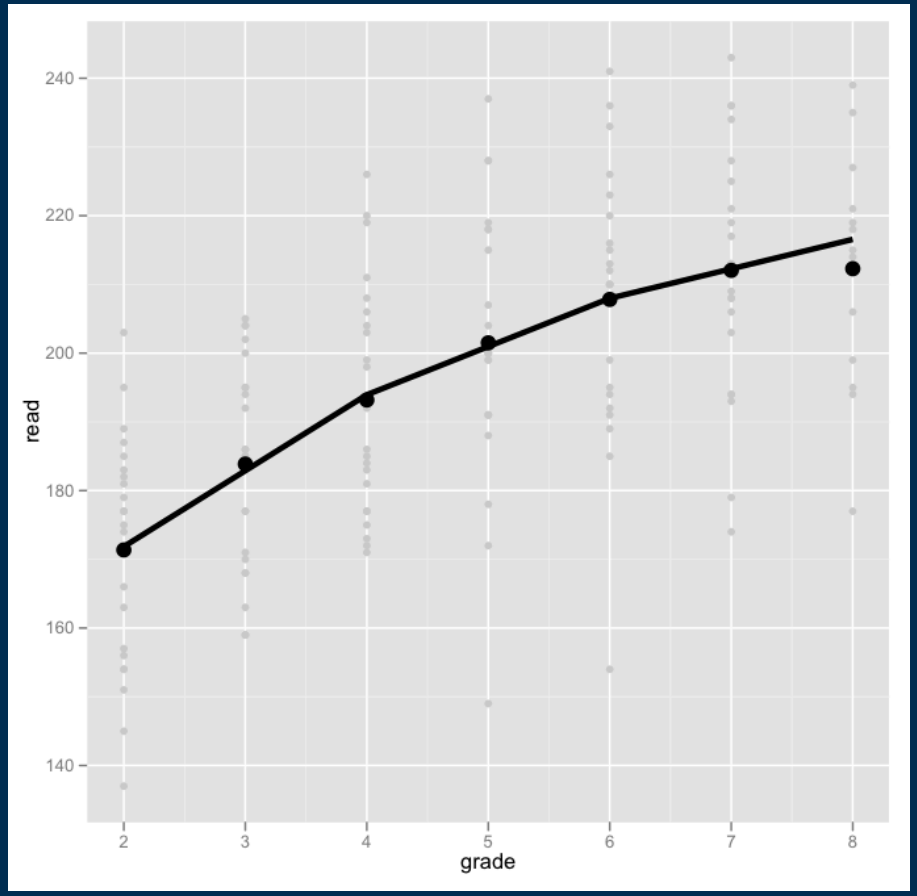
$$E(y_{ij}) = \begin{cases} \beta_0 + \beta_1(\texttt{grade}_{ij}), & \text{if } \texttt{grade}_{ij} \leq 4 \\ (\beta_0 - 4\beta_2) + (\beta_1 + \beta_2)(\texttt{grade}_{ij}), & \text{if } 4 < \texttt{grade}_{ij} \leq 6 \\ (\beta_0 - 4\beta_2 - 6\beta_3) + (\beta_1 + \beta_2 + \beta_3)(\texttt{grade}_{ij}), & \text{if } \texttt{grade}_{ij} > 6 \end{cases}$$

- This indicates that when $\beta_2 = \beta_3 = 0$, there is no difference in the change curve for the different pieces
- Thus, β_2 and β_3 are usually of prime interest

```
## Create spline terms
> mplsL$spline1 <- ifelse(mplsL$grade > 4, mplsL$grade - 4, 0)
> mplsL$spline2 <- ifelse(mplsL$grade > 6, mplsL$grade - 6, 0)

## Use spline terms in lmer()
> sp2 <- lmer(read ~ 1 + grade + spline1 + spline2 +
     (1 + grade | subid), data = mplsL, REML = FALSE)
```

```
Linear mixed model fit by maximum likelihood
Formula: read ~ grade + spline1 + spline2 + (grade | subid)
   Data: mplsL
  AIC  BIC logLik deviance REMLdev
 1145 1169 -564.3     1129    1117
Random effects:
 Groups   Name        Variance Std.Dev. Corr
 subid    (Intercept) 225.7414 15.0247
          grade         6.1857  2.4871  -0.441
 Residual              72.7652  8.5303
Number of obs: 146, groups: subid, 22

Fixed effects:
            Estimate Std. Error t value
(Intercept)  149.760      5.045  29.683
grade         11.045      1.338   8.256
spline1       -4.024      2.136  -1.884
spline2       -2.732      2.319  -1.178

Correlation of Fixed Effects:
        (Intr) grade  splin1
grade   -0.796
spline1  0.608 -0.818
spline2 -0.198  0.321 -0.671
```

```
> plotdata <- data.frame(
      read = sp2@frame$read,
      grade = sp2@frame$grade,
      sp2.pred = model.matrix(sp2) %*% fixef(sp2)
      )

> ggplot(data = plotdata, aes(x = grade, y = read)) +
      geom_point(colour = "grey80") +
      stat_summary(fun.y = mean, geom = "point", size = 4) +
      opts(aspect.ratio = 1) +
      geom_line(aes( y = sp2.pred), size = 1.5)
```



The graph indicates the segments [2, 4], (4, 6], and (6, 8] have straight lines with different slopes. The first two segments correspond rather closely with the observed means, but the last segment over-estimates the final mean in a similar manner as the quadratic model

- Static predictors can also be added to spline models
- In the case where one wants to account for additional variation within a span of time, the spline model can be based on higher order polynomials
- Similar to the linear spline model, higher order spline models consist of a polynomial portion and spline transformations, with the latter being based on the number of knots
- Unlike the polynomial portion of the model, there is a single spline transformation whose power coefficient is the same as the highest order polynomial
- Consider the fitted part of the quadratic spline with a single knot

$$E(y_{ij}) = \beta_0 + \beta_1(\texttt{grade}_{ij}) + \beta_2(\texttt{grade}_{ij}^2) + \beta_3(\texttt{grade}_{ij} - k)_+^2$$

- Single-knot model for the two spans of time, assume the knot is located at grade 3

For $\texttt{grade}_{ij} \leq 3$

$$E(y_{ij}) = \beta_0 + \beta_1(\texttt{grade}_{ij}) + \beta_2(\texttt{grade}_{ij}^2)$$

For $\texttt{grade}_{ij} > 3$

$$\begin{aligned} E(y_{ij}) &= \beta_0 + \beta_1(\texttt{grade}_{ij}) + \beta_2(\texttt{grade}_{ij}^2) + \beta_3(\texttt{grade}_{ij} - 3)^2 \\ &= \beta_0 + \beta_1(\texttt{grade}_{ij}) + \beta_2(\texttt{grade}_{ij}^2) + \beta_3(\texttt{grade}_{ij} - 3)(\texttt{grade}_{ij} - 3) \\ &= \beta_0 + \beta_1(\texttt{grade}_{ij}) + \beta_2(\texttt{grade}_{ij}^2) + \beta_3\left(\texttt{grade}_{ij}^2 - 6\texttt{grade}_{ij} + 9\right) \\ &= (\beta_0 + 9\beta_3) + (\beta_1 - 6\beta_3)(\texttt{grade}_{ij}) + (\beta_2 + \beta_3)(\texttt{grade}_{ij}^2) \end{aligned}$$