432 Class 07 Slides

thomaselove.github.io/432

2021-02-23

Setup

```
library(janitor); library(magrittr)
library(here); library(knitr)

library(skimr)
library(broom)
library(rms)
library(patchwork)
library(GGally)

library(tidyverse)

theme_set(theme_bw())
```

Today's Materials

- The `maleptsd` data
- Using `ols` to fit a linear model
  - Obtaining coefficients and basic summaries
  - Validating summary statistics like R^2
  - ANOVA in `ols`
  - Plot Effects with `summary` and `Predict`
  - Building and using a nomogram
  - Evaluating Calibration
  - Influential points and `dfbeta`
- Spending Degrees of Freedom on Non-Linearity
  - The Spearman ρ^2 (rho-squared) plot
- Building Non-Linear Predictors in `ols`
  - Polynomial Functions
  - Restricted Cubic Splines

The `maleptsd` data: Background and Exploration

The maleptsd data

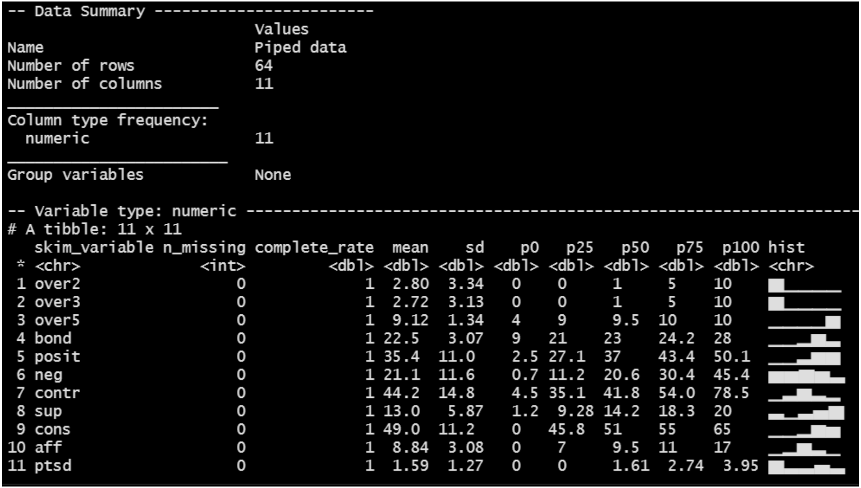
The maleptsd file on our web site contains information on PTSD (post traumatic stress disorder) symptoms following childbirth for 64 fathers[1]. There are ten predictors and the response is a measure of PTSD symptoms. The raw, untransformed values (ptsd_raw) are right skewed and contain zeros, so we will work with a transformation, specifically, ptsd = log(ptsd_raw + 1) as our outcome, which also contains a lot of zeros.

```
maleptsd <- read_csv(here("data/maleptsd.csv")) %>%
    clean_names() %>%
    mutate(ptsd = log(ptsd_raw + 1))
```

[1] Source: Ayers et al. 2007 *J Reproductive and Infant Psychology*. The data are described in more detail in Wright DB and London K (2009) *Modern Regression Techniques Using R* Sage Publications.

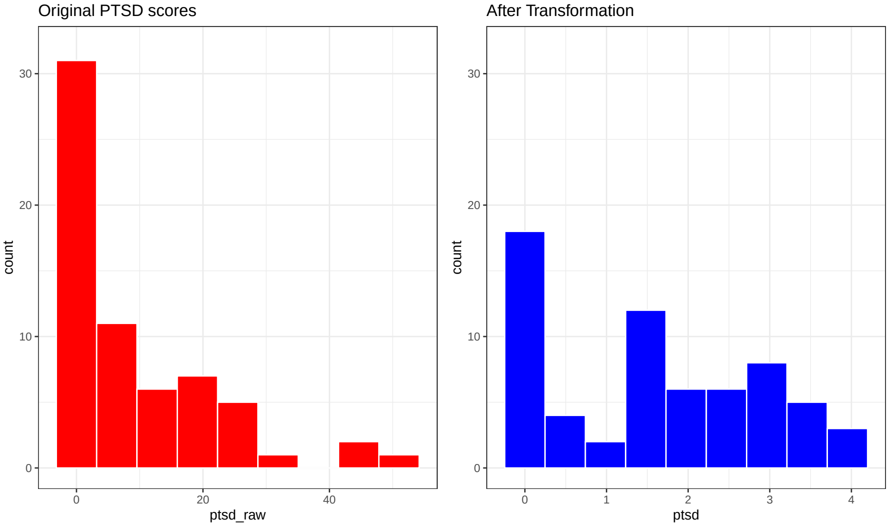
Skimming the maleptsd data

```
maleptsd %>% select(-id, -ptsd_raw) %>% skim()
```



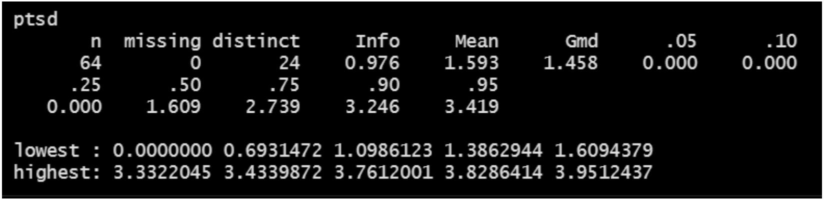
```
-- Data Summary ------------------------
                           Values
Name                       Piped data
Number of rows             64
Number of columns          11
_______________________
Column type frequency:
  numeric                  11
________________________
Group variables            None

-- Variable type: numeric ------------------------------------------------------------
# A tibble: 11 x 11
   skim_variable n_missing complete_rate  mean    sd    p0   p25   p50   p75  p100 hist
 * <chr>             <int>         <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <chr>
 1 over2                 0             1  2.80  3.34   0     0     1     5    10   ▇▁▁▁▁
 2 over3                 0             1  2.72  3.13   0     0     1     5    10   ▇▁▁▁▁
 3 over5                 0             1  9.12  1.34   4     9     9.5  10    10   ▁▁▁▁▇
 4 bond                  0             1 22.5   3.07   9    21    23    24.2  28   ▁▁▂▇▃
 5 posit                 0             1 35.4  11.0    2.5 27.1  37    43.4  50.1  ▁▁▃▇▇
 6 neg                   0             1 21.1  11.6    0.7 11.2  20.6  30.4  45.4  ▇▇▇▇▃
 7 contr                 0             1 44.2  14.8    4.5 35.1  41.8  54.0  78.5  ▁▃▇▃▂
 8 sup                   0             1 13.0   5.87   1.2  9.28 14.2  18.3  20    ▃▁▃▅▇
 9 cons                  0             1 49.0  11.2    0   45.8  51    55    65    ▁▁▂▇▆
10 aff                   0             1  8.84  3.08   0     7     9.5  11    17   ▁▂▇▃▁
11 ptsd                  0             1  1.59  1.27   0     0     1.61  2.74  3.95 ▇▃▃▅▃
```

Transformation of Outcome



Hmisc::describe() for this transformed outcome

```
maleptsd %$% describe(ptsd)
```



```
ptsd
       n  missing distinct     Info     Mean      Gmd      .05      .10
      64        0       24    0.976    1.593    1.458    0.000    0.000
     .25      .50      .75      .90      .95
   0.000    1.609    2.739    3.246    3.419

lowest : 0.0000000 0.6931472 1.0986123 1.3862944 1.6094379
highest: 3.3322045 3.4339872 3.7612001 3.8286414 3.9512437
```

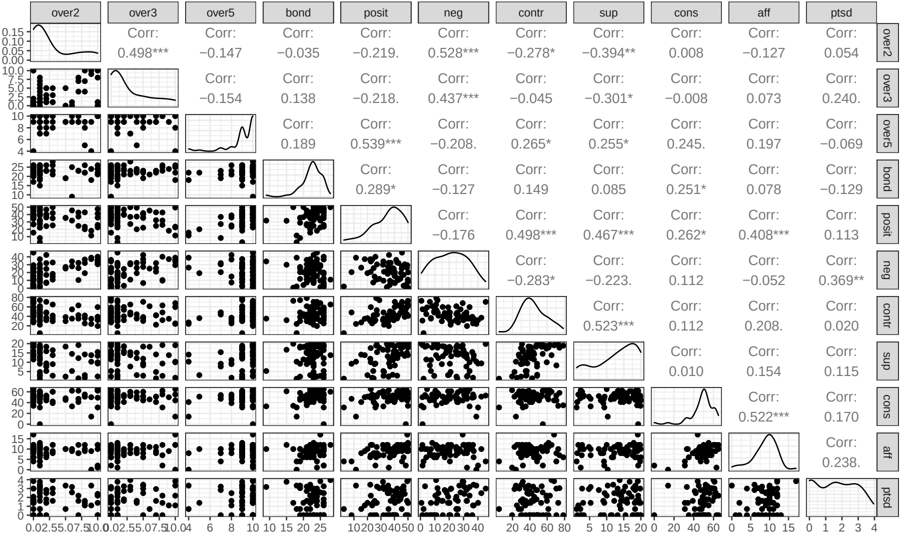
- `Gmd` is Gini's mean difference, a robust measure of variation.
- If you randomly selected two of the 64 subjects, the average difference in `ptsd` would be 1.458.

Scatterplot Matrix (code)

```
temp <- maleptsd %>%
    select(over2, over3, over5, bond, posit, neg, contr,
           sup, cons, aff, ptsd)

ggpairs(temp)  ## ggpairs from the GGally package
```

- Note that I've placed the outcome (`ptsd`) last here.
- Result is on the next slide...

Scatterplot Matrix (result)



Using `ols` to fit a linear regression model

Fitting using ols

The ols function stands for ordinary least squares and comes from the rms package, by Frank Harrell and colleagues. Any model fit with lm can also be fit with ols.

To predict var_y using var_x from the my_tibble data, we would use the following syntax:

```
dd <- datadist(my_tibble)
options(datadist = "dd")

model_name <- ols(var_y ~ var_x, data = my_tibble,
                  x = TRUE, y = TRUE)
```

This leaves the following questions:

1. What's the datadist stuff doing?
2. Why use x = TRUE, y = TRUE in the fit?

What is datadist?

Before we fit any ols model to data from my_tibble, we'll use:

```
dd <- datadist(my_tibble)
options(datadist = "dd")
```

> *Run (the datadist code above) once before any models are fitted, storing the distribution summaries for all potential variables. Adjustment values are 0 for binary variables, the most frequent category (or optionally the first category level) for categorical (factor) variables, the middle level for ordered factor variables, and medians for continuous variables.*

- excerpted from the datadist documentation

Why use x = TRUE, y = TRUE in the fit?

Once we've set up the distribution summaries with the datadist code, we fit linear regression models using the same fitting routines as lm with ols:

```
model_name <- ols(var_y ~ var_x, data = my_tibble,
                  x = TRUE, y = TRUE)
```

- ols stores additional information beyond what lm does
- x = TRUE and y = TRUE save even more expanded information that we'll need in building plots and summaries of the fit.
- The defaults are x = FALSE, y = FALSE, but in this class, we'll always want to include these additional pieces.

Using ols to fit a Two-Predictor Model

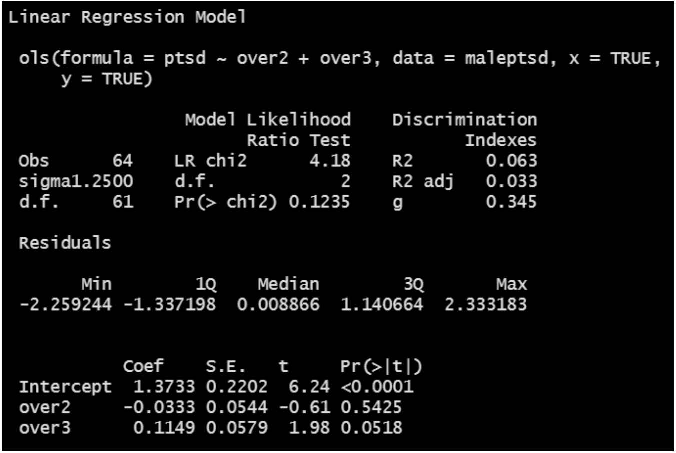
Now, we'll fit an ols model predicting our outcome (ptsd) using two predictors (over2 and over3) using the maleptsd tibble.

- Start with setting the datadist up
- Then fit the model, including x = TRUE, y = TRUE

```
dd <- datadist(maleptsd)
options(datadist = "dd")

mod_first <- ols(ptsd ~ over2 + over3, data = maleptsd,
                 x = TRUE, y = TRUE)
```

Contents of `mod_first`?

```
mod_first
```



```
Linear Regression Model

 ols(formula = ptsd ~ over2 + over3, data = maleptsd, x = TRUE,
     y = TRUE)

                Model Likelihood    Discrimination
                   Ratio Test          Indexes
 Obs       64    LR chi2      4.18    R2       0.063
 sigma1.2500    d.f.            2    R2 adj   0.033
 d.f.      61    Pr(> chi2) 0.1235    g        0.345

 Residuals

      Min        1Q    Median        3Q       Max
 -2.259244 -1.337198  0.008866  1.140664  2.333183


           Coef    S.E.   t     Pr(>|t|)
 Intercept  1.3733 0.2202  6.24 <0.0001
 over2     -0.0333 0.0544 -0.61 0.5425
 over3      0.1149 0.0579  1.98 0.0518
```

- Likelihood Ratio test?
- What is g?

New elements in ols

For our mod_first,

- Model Likelihood Ratio test output includes LR chi2 = 4.18, d.f. = 2, Pr(> chi2) = 0.1235

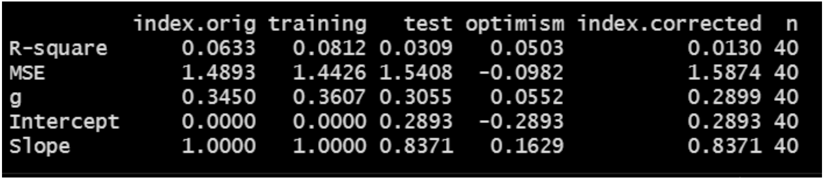
The log of the likelihood ratio, multiplied by -2, yields a test against a χ^2 distribution. Interpret this as a goodness-of-fit test that compares mod_first to a null model with only an intercept term. In ols this is similar to a global (ANOVA) F test.

- Under the R^2 values, we have g = 0.345.
- This is the *g*-index, based on Gini's mean difference. If you randomly selected two of the subjects in the model, the average difference in predicted ptsd will be 0.345.
- This can be compared to the Gini's mean difference for the original ptsd values, from Hmisc::describe, which was Gmd = 1.458.

Validate the summary statistics of an `ols` fit

- Can we validate summary statistics by resampling?

```
set.seed(432010)
validate(mod_first)
```



```
          index.orig training   test optimism index.corrected  n
R-square      0.0633   0.0812 0.0309   0.0503          0.0130 40
MSE           1.4893   1.4426 1.5408  -0.0982          1.5874 40
g             0.3450   0.3607 0.3055   0.0552          0.2899 40
Intercept     0.0000   0.0000 0.2893  -0.2893          0.2893 40
Slope         1.0000   1.0000 0.8371   0.1629          0.8371 40
```

- The data used to fit the model provide an over-optimistic view of the quality of fit.
- We're interested here in assessing how well the model might work in new data, and to do so, we can use a resampling approach.
- Consider R^2 here. . .

Interpreting the Resampling Validation Results

```
    index.orig training   test optimism index.corrected  n
R-sq     0.0633   0.0812 0.0309   0.0503          0.0130 40
```

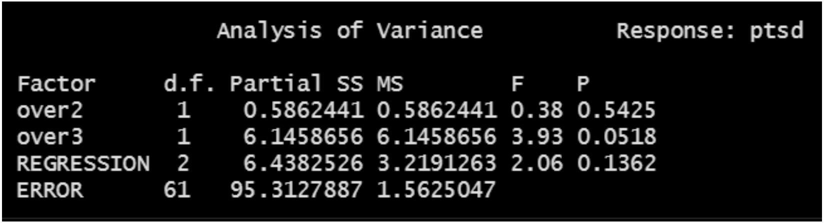
- `index.orig` for R^2 is 0.0633. That's what we get from the data we used to fit the model, and is what we see in our standard output.
- With `validate` we create 40 (by default) bootstrapped resamples of the data and then split each of those into training and test samples.
  - For each of the 40 splits, R refits the model (same predictors) in the `training` sample to obtain R^2: mean across 40 splits is 0.0812.
  - Check each model in its `test` sample: average R^2 was 0.0309.
- `optimism` = `training` result - `test` result = 0.0503
- `index.corrected` = `index.orig` - `optimism` = 0.0130

While our *nominal* R^2 is 0.0633 for this model, but correcting for optimism yields a *validated* R^2 of 0.0130.

- $R^2 = 0.0130$ better estimates how the model will perform in new data.

ANOVA for `mod_first` fit by `ols`

```
anova(mod_first)
```

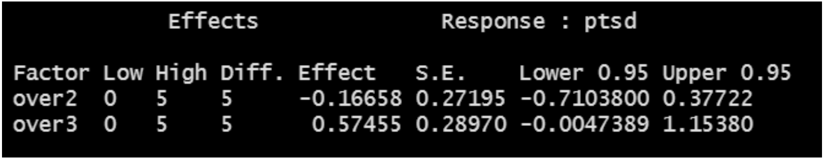


```
                Analysis of Variance          Response: ptsd

 Factor     d.f. Partial SS MS        F    P
 over2       1    0.5862441 0.5862441 0.38 0.5425
 over3       1    6.1458656 6.1458656 3.93 0.0518
 REGRESSION  2    6.4382526 3.2191263 2.06 0.1362
 ERROR      61   95.3127887 1.5625047
```

- This adds a line for the complete regression model (both terms) which can be helpful, but is otherwise the same as `anova` after `lm`.
- As with `lm`, this is a sequential ANOVA table, so if we had included over3 in the model first, we'd get a different SS, MS, F and p for `over2` and `over3`, but the same `REGRESSION` and `ERROR` results.

summary for `mod_first` fit by `ols`

```
summary(mod_first)
```



```
             Effects              Response : ptsd

 Factor Low High Diff. Effect   S.E.    Lower 0.95 Upper 0.95
 over2  0    5    5     -0.16658 0.27195 -0.7103800 0.37722
 over3  0    5    5      0.57455 0.28970 -0.0047389 1.15380
```

- For `over2` effect `-0.16658` is the estimated change in `ptsd` associated with a move from `over2` = 0 (see `Low` value) to `over2` = 5 (the `High` value) assuming no change in `over3`.
- `ols` chooses the `Low` and `High` values from the interquartile range.

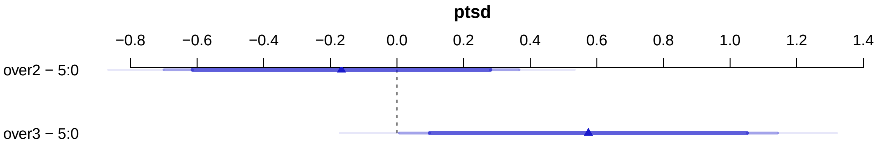
```
maleptsd %$% quantile(over2, c(0.25, 0.75))
```

```
25% 75% 
  0   5 
```

Plot the summary to see effect sizes

- Goal: plot effect sizes for similar moves within predictor distributions.

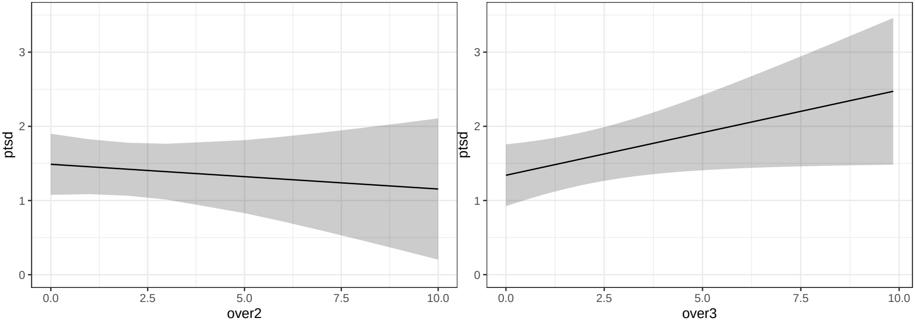
```
plot(summary(mod_first))
```



- The triangles indicate the point estimate, augmented with confidence interval bars.
  - The 90% confidence intervals are plotted with the thickest bars.
  - The 95% CIs are then shown with thinner, more transparent bars.
  - Finally, the 99% CIs are shown as the longest, thinnest bars.

What do the individual effects look like?

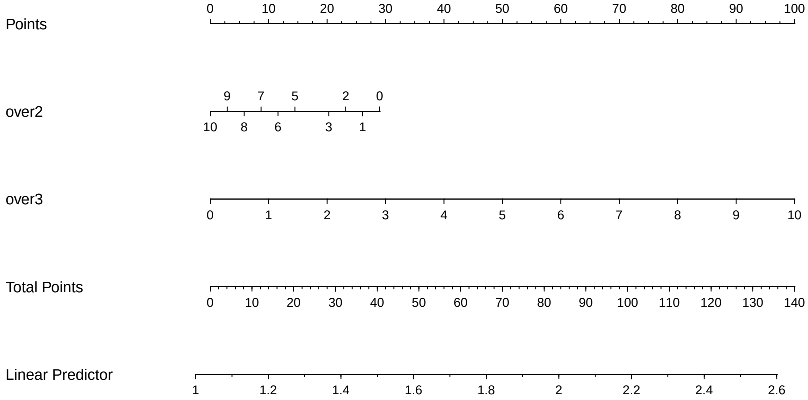
```
ggplot(Predict(mod_first, conf.int = 0.95), layout = c(1,2))
```



- The left plot shows the impact of changing `over2` on `ptsd` holding `over3` constant at the median `over3` value, which is 1.
- The right plot shows the impact of changing `over3` on `ptsd` holding `over2` constant at its median value, which is 1.
- Defaults: add 95% CI bands and layout tries for a square.

Build a nomogram for the `ols` fit

```
plot(nomogram(mod_first))
```



Nomograms

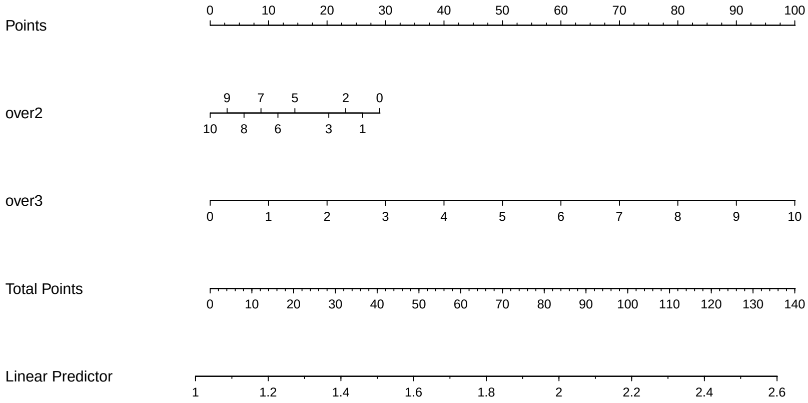
For complex models (this model isn't actually very complex) it can be helpful to have a tool that will help you see the effects of model in terms of their impact on the predicted outcome.

A *nomogram* is an established graphical tool for doing this.

- Find the value of each predictor on its provided line, and identify the "points" for that predictor by drawing a vertical line up to the "Points".
- Then sum up the points over all predictors to obtain "Total Points".
- Draw a vertical line down from the "Total Points" to the "Linear Predictor" to get the predicted `ptsd` for this subject.

Using the nomogram for the `ols` fit

Predicted `ptsd` for a subject with `over2` = 8 and `over3` = 7?



Actual Prediction for such a subject. . .

- The `predict` function for an `ols` fit can provide fitted values.

```
predict(mod_first,
           newdata = tibble(over2 = 8, over3 = 7))
```

```
       1 
1.911123 
```

- The `broom` package doesn't (really) support `rms` fits, and throws a warning (omitted here), but you could always refit the model with `lm`. . .

```
augment(mod_first,
           newdata = tibble(over2 = 8, over3 = 7))
```

```
# A tibble: 1 x 3
  over2 over3 .fitted
  <dbl> <dbl>   <dbl>
1     8     7    1.91
```

Assessing the Calibration of `mod_first`

We would like our model to be well-calibrated, in the following sense...

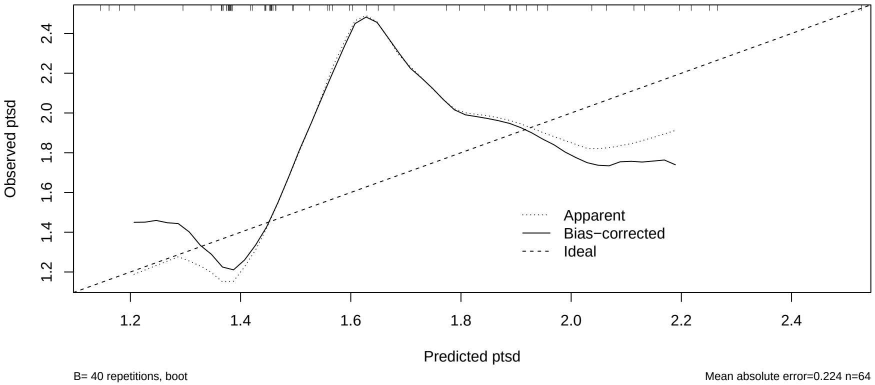
- Suppose our model assigns a predicted outcome of 6 to several subjects. If the model is well-calibrated, then we expect the mean of those subjects' actual outcomes to be very close to 6.

We'd like to look at the relationship between the observed `ptsd` outcome and our predicted `ptsd` from the model.

- The calibration plot we'll create provides two estimates (with and without bias-correction) of the predicted vs. observed values of our outcome, and compares these to the ideal scenario (predicted = observed).
- The plot uses resampling validation to produce bias-corrected estimates and uses lowess smooths to connect across predicted values.
- Calibration plots require `x = TRUE, y = TRUE` in the `ols` fit.

Calibration Plot for `mod_first`

```
set.seed(432); plot(calibrate(mod_first))
```



```
n=64   Mean absolute error=0.224   Mean squared error=0.1048
0.9 Quantile of absolute error=0.64
```

Influential Points for `mod_first`?

The `dfbeta` value for a particular subject and coefficient β is the change in the coefficient that happens when the subject is excluded from the model.

```
which.influence(mod_first, cutoff = 0.3)
```

```
$over2
[1] 32 33 37 40 43 57

$over3
[1] 32 43 57
```

- These are the subjects that have absolute values of `dfbetas` that exceed the specified cutoff (default is 0.2)

Show the influential points more directly?

```
w <- which.influence(mod_first, cutoff = 0.3)
d <- maleptsd %>% select(over2,over3,ptsd) %>% data.frame()
show.influence(w, d)
```

```
   Count over2 over3
32     2  * 0   *10
33     1  *10     1
37     1  *10     0
40     1  *10     1
43     2  * 6   * 0
57     2  * 1   * 8
```

- Count = number of coefficients where this row appears influential.
- Use `maleptsd %>% slice(32)` to see row 32 in its entirety.
- Use residual plots (with an `lm` fit) to check Cook's distances.

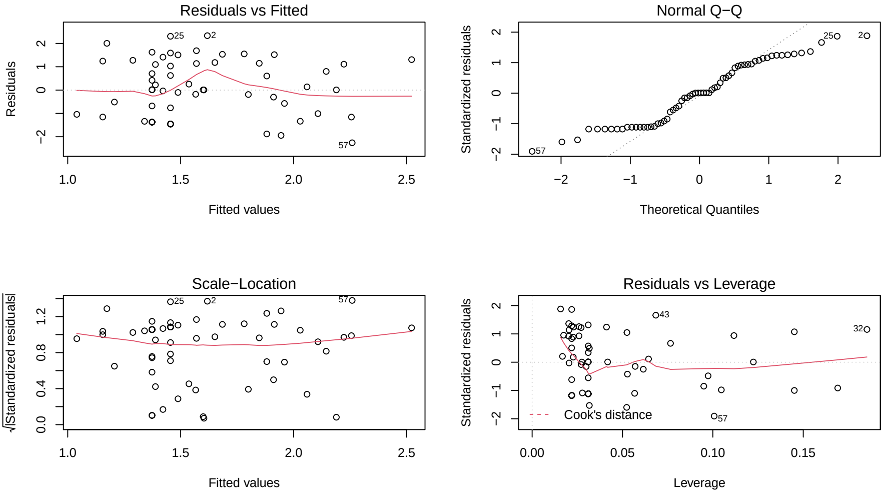
Fitting Residual Plots for this model

To fit residual plots (and sometimes to do other things) we will fit the `lm` version of this same model…

```r
mod_first_lm <- lm(ptsd ~ over2 + over3, data = maleptsd)

par(mfrow = c(2,2))
plot(mod_first_lm)
par(mfrow = c(1,1))
```

- Plots are shown on the next slide. While the subject in row 32 is more influential than most other points, it doesn't reach the standard of a problematic Cook's distance.

Residual Plots for `mod_first`



Thinking about Non-Linear Terms?

Non-Linear Terms

In building a linear regression model, we're most often going to be thinking about:

- for quantitative predictors, some curvature...
  - perhaps polynomial terms
  - but more often restricted cubic splines
- for any predictors, possible interactions
  - between categorical predictors
  - between categorical and quantitative predictors
  - between quantitative predictors

Polynomial Regression

A polynomial in the variable `x` of degree D is a linear combination of the powers of `x` up to D. Fitting such a model creates a **polynomial regression**.

- Linear: $y = \beta_0 + \beta_1 x$
- Quadratic: $y = \beta_0 + \beta_1 x + \beta_2 x^2$
- Cubic: $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$
- Quartic: $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4$
- Quintic: $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4 + \beta_5 x^5$

An **orthogonal polynomial** sets up a model design matrix and then scales those columns so that each column is uncorrelated with the previous ones.

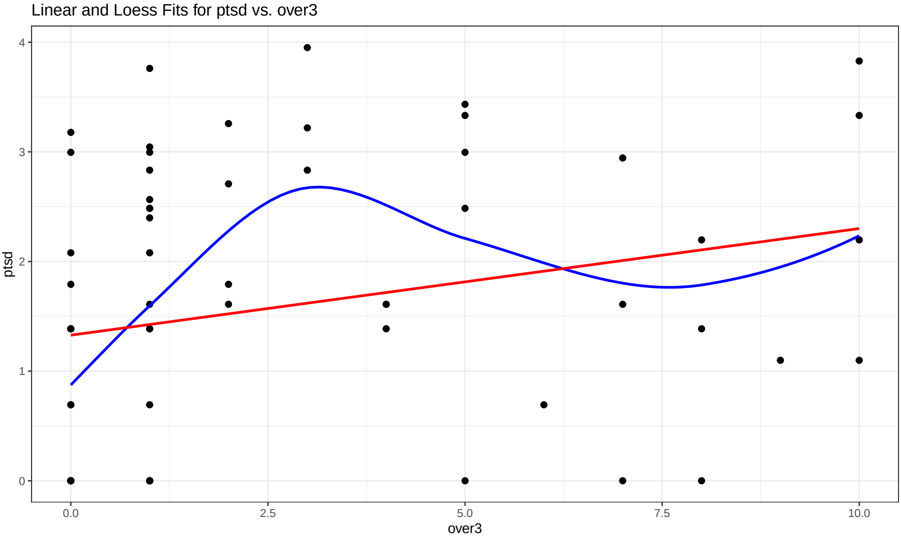
- This reduction in collinearity (correlation between predictors) lets us gauge whether the addition of any particular polynomial term improves model fit.

Using over3 to predict ptsd

- Let's look at both a linear fit and a loess smooth to see if they indicate meaningfully different things about the association.

```
ggplot(maleptsd, aes(x = over3, y = ptsd)) +
    geom_point(size = 2) +
    geom_smooth(method = "loess", formula = y ~ x,
                se = FALSE, col = "blue") +
    geom_smooth(method = "lm", formula = y ~ x,
                se = FALSE, col = "red") +
    labs(title = "Linear and Loess Fits for ptsd vs. over3")
```

Linear and Loess Fits for `ptsd` with `over3`



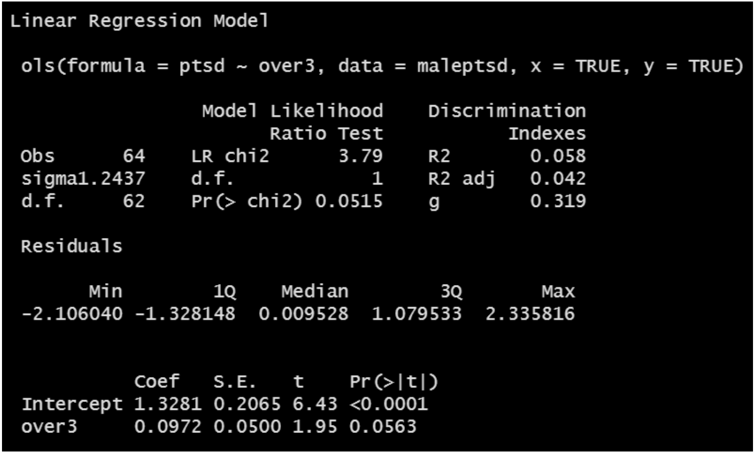
Fitting polynomial regressions with `ols`

```
dd <- datadist(maleptsd)
options(datadist = "dd")

mod_B1 <- ols(ptsd ~ over3,
              data = maleptsd, x = TRUE, y = TRUE)
mod_B2 <- ols(ptsd ~ pol(over3, 2),
              data = maleptsd, x = TRUE, y = TRUE)
mod_B3 <- ols(ptsd ~ pol(over3, 3),
              data = maleptsd, x = TRUE, y = TRUE)
```

- Note the use of `pol()` from the `rms` package here to fit orthogonal polynomials, rather than `poly()` which we used for an `lm` fit.

Model B1 (linear in `over3`)

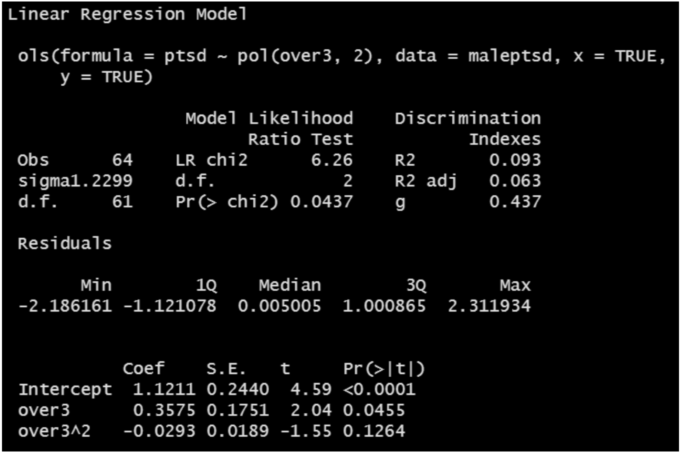
```
mod_B1
```



```
Linear Regression Model

 ols(formula = ptsd ~ over3, data = maleptsd, x = TRUE, y = TRUE)

                 Model Likelihood    Discrimination
                       Ratio Test           Indexes
 Obs       64    LR chi2      3.79    R2       0.058
 sigma1.2437    d.f.            1    R2 adj   0.042
 d.f.      62    Pr(> chi2) 0.0515    g        0.319

 Residuals

       Min         1Q    Median        3Q       Max
 -2.106040 -1.328148  0.009528  1.079533  2.335816


           Coef   S.E.   t    Pr(>|t|)
 Intercept 1.3281 0.2065 6.43 <0.0001
 over3     0.0972 0.0500 1.95 0.0563
```

Model B2 (quadratic polynomial in `over3`)

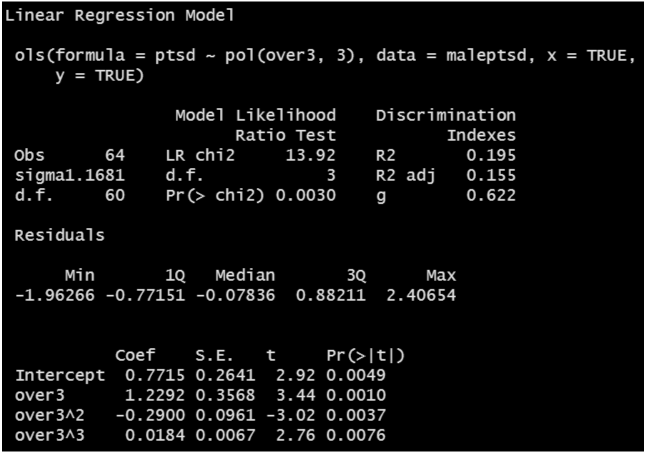
```
mod_B2
```



```
Linear Regression Model

 ols(formula = ptsd ~ pol(over3, 2), data = maleptsd, x = TRUE,
     y = TRUE)

                 Model Likelihood    Discrimination
                       Ratio Test           Indexes
 Obs       64    LR chi2      6.26    R2       0.093
 sigma1.2299    d.f.            2    R2 adj   0.063
 d.f.      61    Pr(> chi2) 0.0437    g        0.437

 Residuals

       Min         1Q    Median        3Q       Max
 -2.186161 -1.121078  0.005005  1.000865  2.311934


           Coef    S.E.   t     Pr(>|t|)
 Intercept  1.1211 0.2440  4.59 <0.0001
 over3      0.3575 0.1751  2.04 0.0455
 over3^2   -0.0293 0.0189 -1.55 0.1264
```

Model B3 (cubic polynomial in `over3`)

```
mod_B3
```



```
Linear Regression Model

 ols(formula = ptsd ~ pol(over3, 3), data = maleptsd, x = TRUE,
     y = TRUE)

                 Model Likelihood    Discrimination
                       Ratio Test           Indexes
 Obs       64    LR chi2      13.92    R2       0.195
 sigma1.1681     d.f.             3    R2 adj   0.155
 d.f.      60    Pr(> chi2) 0.0030    g        0.622

 Residuals

      Min       1Q    Median       3Q      Max
 -1.96266 -0.77151 -0.07836  0.88211  2.40654


           Coef    S.E.   t     Pr(>|t|)
 Intercept  0.7715 0.2641  2.92 0.0049
 over3      1.2292 0.3568  3.44 0.0010
 over3^2   -0.2900 0.0961 -3.02 0.0037
 over3^3    0.0184 0.0067  2.76 0.0076
```

Store the polynomial fits

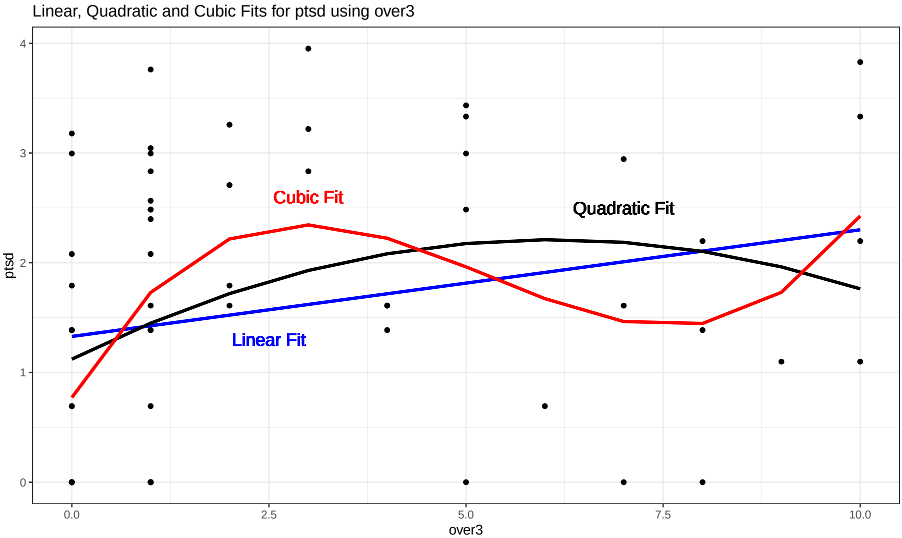
First, we need to store the values. Again `broom` doesn't play well with `ols` fits, so I'll just add the predictions as columns

```
ptsd_fits <- maleptsd %>%
    mutate(fitB1 = predict(mod_B1),
           fitB2 = predict(mod_B2),
           fitB3 = predict(mod_B3))
```

Code to plot polynomial fits

```r
ggplot(ptsd_fits, aes(x = over3, y = ptsd)) +
    geom_point() +
    geom_line(aes(x = over3, y = fitB1),
              col = "blue", size = 1.25) +
    geom_line(aes(x = over3, y = fitB2),
              col = "black", size = 1.25) +
    geom_line(aes(x = over3, y = fitB3),
              col = "red", size = 1.25) +
    geom_text(x = 2.5, y = 1.3, label = "Linear Fit",
              size = 5, col = "blue") +
    geom_text(x = 7, y = 2.5, label = "Quadratic Fit",
              size = 5, col = "black") +
    geom_text(x = 3, y = 2.6, label = "Cubic Fit",
              size = 5, col = "red") +
    labs(title = "Linear, Quadratic and Cubic Fits for ptsd us
```

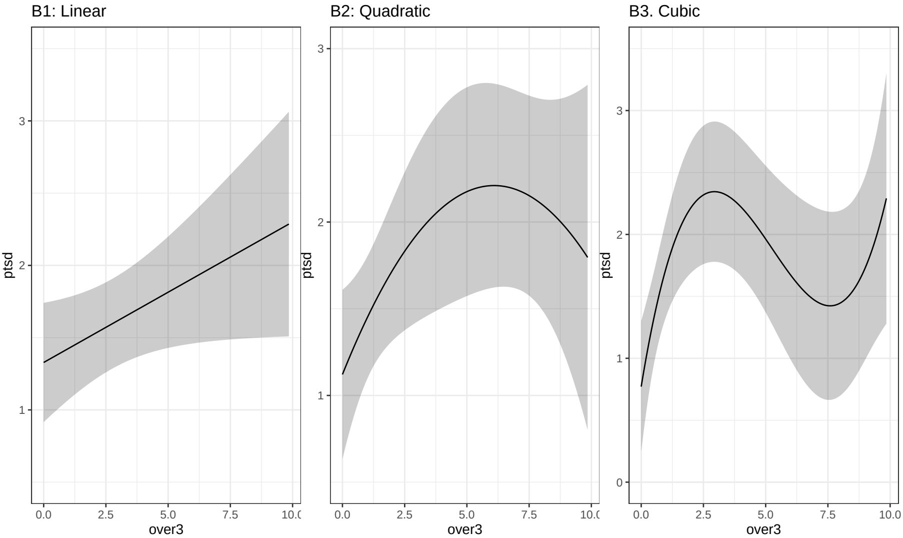
The Polynomial Fits, plotted



Code to plot polynomial fits with Predict

```
p1 <- ggplot(Predict(mod_B1)) + ggtitle("B1: Linear")
p2 <- ggplot(Predict(mod_B2)) + ggtitle("B2: Quadratic")
p3 <- ggplot(Predict(mod_B3)) + ggtitle("B3. Cubic")

p1 + p2 + p3
```

Visualizing the polynomial fits with Predict



Splines

- A **linear spline** is a continuous function formed by connecting points (called **knots** of the spline) by line segments.
- A **restricted cubic spline** is a way to build highly complicated curves into a regression equation in a fairly easily structured way.
- A restricted cubic spline is a series of polynomial functions joined together at the knots.
    - Such a spline gives us a way to flexibly account for non-linearity without over-fitting the model.
    - Restricted cubic splines can fit many different types of non-linearities.
    - Specifying the number of knots is all you need to do in R to get a reasonable result from a restricted cubic spline.

The most common choices are 3, 4, or 5 knots.

- 3 Knots, 2 degrees of freedom, allows the curve to “bend” once.
- 4 Knots, 3 degrees of freedom, lets the curve “bend” twice.
- 5 Knots, 4 degrees of freedom, lets the curve “bend” three times.

Fitting Restricted Cubic Splines with ols

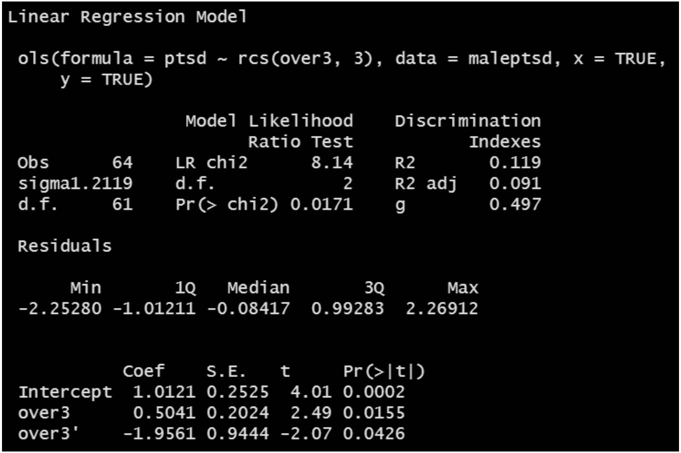
Let's consider a restricted cubic spline model for ptsd based on over3 with:

- 3 knots in modC3, 4 knots in modC4, and 5 knots in modC5

```
dd <- datadist(maleptsd)
options(datadist = "dd")

mod_C3 <- ols(ptsd ~ rcs(over3, 3),
              data = maleptsd, x = TRUE, y = TRUE)
mod_C4 <- ols(ptsd ~ rcs(over3, 4),
              data = maleptsd, x = TRUE, y = TRUE)
mod_C5 <- ols(ptsd ~ rcs(over3, 5),
              data = maleptsd, x = TRUE, y = TRUE)
```

Model C3 (3-knot spline in over3)

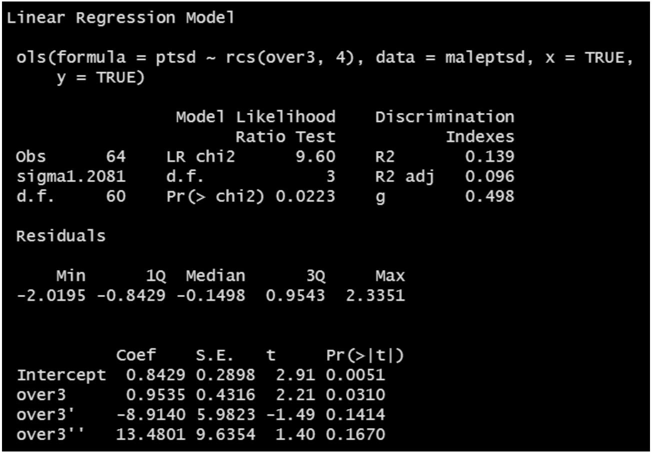
```
mod_C3
```



```
Linear Regression Model

 ols(formula = ptsd ~ rcs(over3, 3), data = maleptsd, x = TRUE,
     y = TRUE)

                  Model Likelihood    Discrimination
                        Ratio Test           Indexes
 Obs       64     LR chi2      8.14    R2       0.119
 sigma1.2119      d.f.            2    R2 adj   0.091
 d.f.      61     Pr(> chi2) 0.0171    g        0.497

 Residuals

      Min       1Q    Median       3Q      Max
 -2.25280 -1.01211 -0.08417  0.99283  2.26912


           Coef    S.E.   t     Pr(>|t|)
 Intercept  1.0121 0.2525  4.01 0.0002
 over3      0.5041 0.2024  2.49 0.0155
 over3'    -1.9561 0.9444 -2.07 0.0426
```

Model C4 (4-knot spline in over3)

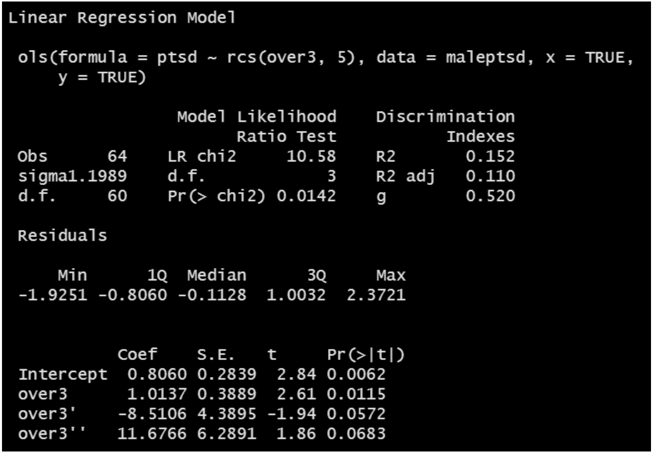
```
mod_C4
```



```
Linear Regression Model

 ols(formula = ptsd ~ rcs(over3, 4), data = maleptsd, x = TRUE,
     y = TRUE)

                 Model Likelihood    Discrimination
                       Ratio Test           Indexes
 Obs       64    LR chi2      9.60    R2       0.139
 sigma1.2081     d.f.            3    R2 adj   0.096
 d.f.      60    Pr(> chi2) 0.0223    g        0.498

 Residuals

     Min      1Q  Median      3Q     Max
 -2.0195 -0.8429 -0.1498  0.9543  2.3351


           Coef    S.E.   t     Pr(>|t|)
 Intercept  0.8429 0.2898  2.91 0.0051
 over3      0.9535 0.4316  2.21 0.0310
 over3'    -8.9140 5.9823 -1.49 0.1414
 over3''   13.4801 9.6354  1.40 0.1670
```

Model C5 (5-knot spline in over3)

```
mod_C5
```

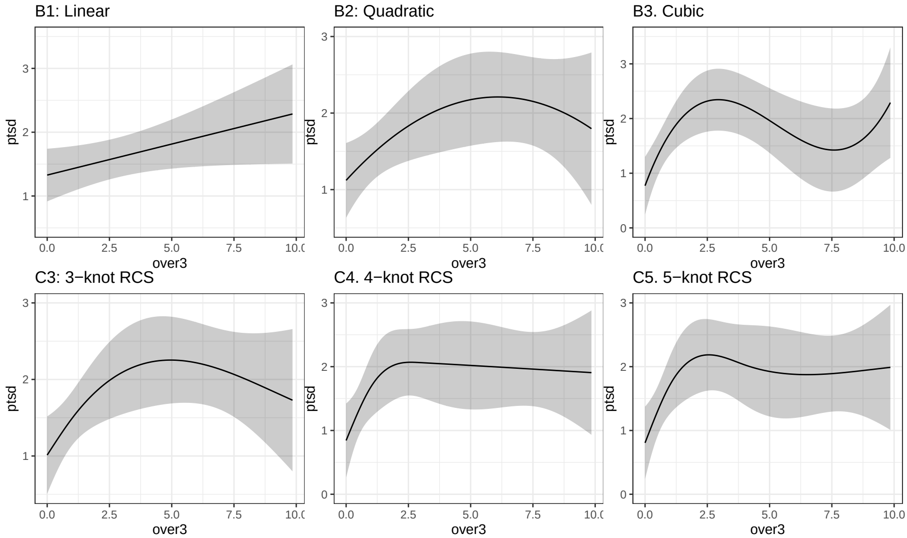


```
Linear Regression Model

 ols(formula = ptsd ~ rcs(over3, 5), data = maleptsd, x = TRUE,
     y = TRUE)

                 Model Likelihood    Discrimination
                       Ratio Test           Indexes
 Obs       64    LR chi2     10.58    R2       0.152
 sigma1.1989    d.f.            3    R2 adj   0.110
 d.f.      60    Pr(> chi2) 0.0142    g        0.520

 Residuals

     Min      1Q  Median      3Q     Max
 -1.9251 -0.8060 -0.1128  1.0032  2.3721


           Coef    S.E.   t     Pr(>|t|)
 Intercept  0.8060 0.2839  2.84 0.0062
 over3      1.0137 0.3889  2.61 0.0115
 over3'    -8.5106 4.3895 -1.94 0.0572
 over3''   11.6766 6.2891  1.86 0.0683
```

Code to plot all six fits

```r
p1 <- ggplot(Predict(mod_B1)) + ggtitle("B1: Linear")
p2 <- ggplot(Predict(mod_B2)) + ggtitle("B2: Quadratic")
p3 <- ggplot(Predict(mod_B3)) + ggtitle("B3. Cubic")
p4 <- ggplot(Predict(mod_C3)) + ggtitle("C3: 3-knot RCS")
p5 <- ggplot(Predict(mod_C4)) + ggtitle("C4. 4-knot RCS")
p6 <- ggplot(Predict(mod_C5)) + ggtitle("C5. 5-knot RCS")

(p1 + p2 + p3) / (p4 + p5 + p6)
```

Visualizing the fits better?



Data Spending: Non-Linearity Prior to Fits

Spending degrees of freedom wisely

- Suppose we have a data set with many possible predictors, and minimal theory or subject matter knowledge to guide us.
- We might want our final inferences to be as unbiased as possible. To accomplish this, we have to pay a penalty (in terms of degrees of freedom) for any "peeks" we make at the data in advance of fitting a model.
- So that rules out a lot of decision-making about non-linearity based on looking at the data, if our sample size isn't much larger than 15 times the number of predictors we're considering including in our model.
- In our case, we have n = 64 observations on 10 predictors.
- In addition, adding non-linearity to our model costs additional degrees of freedom.
- What can we do?

Spearman's ρ^2 plot: A smart first step?

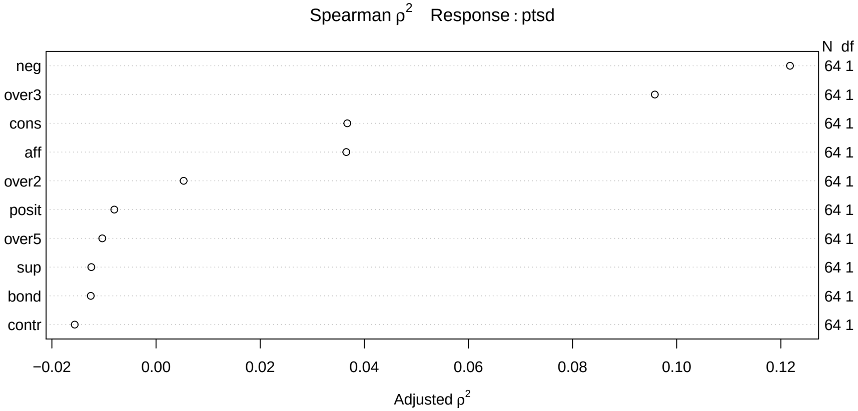
Spearman's ρ^2 is an indicator (not a perfect one) of potential predictive punch, but doesn't give away the game.

- Idea: Perhaps we should focus our efforts re: non-linearity on predictors that score better on this measure.

```
spear_ptsd <- spearman2(ptsd ~ over2 + over3 + over5 + bond +
                   posit + neg + contr + sup + cons + aff,
               data = maleptsd)
```

Spearman's ρ^2 Plot

```
plot(spear_ptsd)
```



Conclusions from Spearman ρ^2 Plot

- `neg` is the most attractive candidate for a non-linear term, as it packs the most potential predictive punch, so if it does turn out to need non-linear terms, our degrees of freedom will be well spent.
    - By no means is this suggesting that `neg` actually needs a non-linear term, or will show significant non-linearity. We'd have to fit a model with and without non-linearity in `neg` to know that.
    - Non-linearity will often take the form of a product term, a polynomial term, or a restricted cubic spline.
    - Since all of these predictors are quantitative, we'll think about polynomial or spline terms, soon.
- `over3`, also quantitative, has the next most potential predictive punch
- these are followed by `cons` and `aff`

Grim Reality

With 64 observations (63 df) we should be thinking about models with relatively tiny numbers of regression inputs.

- Non-linear terms (polynomials, splines) just add to the problem, as they need additional df to be estimated.

In this case, we might choose between

- including non-linearity in one (or maybe 2) variables (and that's it),
- or a linear model including maybe 3-4 predictors, tops

and even that would be tough to justify with this small sample size.

Contents of spear_ptsd

```
spear_ptsd
```

```
Spearman rho^2    Response variable:ptsd

       rho2    F df1 df2      P Adjusted rho2  n
over2 0.021 1.34   1  62 0.2522         0.005 64
over3 0.110 7.67   1  62 0.0074         0.096 64
over5 0.006 0.36   1  62 0.5527        -0.010 64
bond  0.004 0.22   1  62 0.6405        -0.013 64
posit 0.008 0.50   1  62 0.4825        -0.008 64
neg   0.136 9.73   1  62 0.0027         0.122 64
contr 0.001 0.03   1  62 0.8602        -0.016 64
sup   0.004 0.23   1  62 0.6357        -0.012 64
cons  0.052 3.40   1  62 0.0699         0.037 64
aff   0.052 3.39   1  62 0.0704         0.037 64
```

Proposed New Model

Fit a model to predict `ptsd` using:

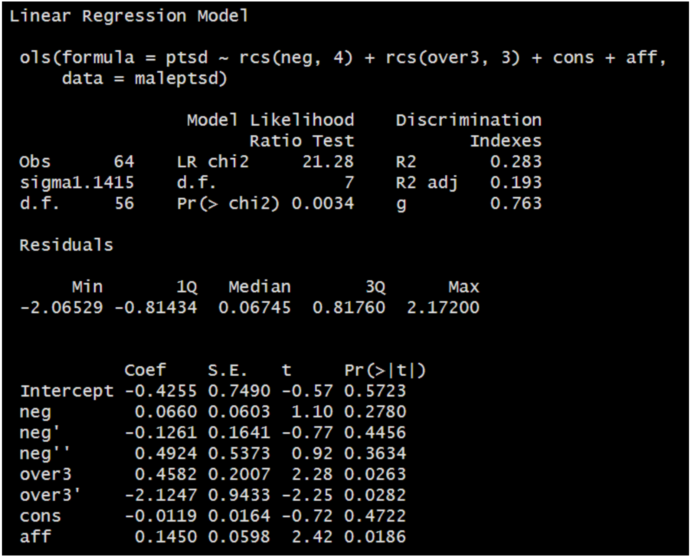
- a 4-knot spline on `neg`
- a 3-knot spline on `over3`
- a linear term on `cons`
- a linear term on `aff`

Perhaps more than we can reasonably do with 64 observations, but let's see how it looks.

```
dd <- datadist(maleptsd)
options(datadist = "dd")

mod_second <- ols(ptsd ~ rcs(neg, 4) + rcs(over3, 3) +
                     cons + aff, data = maleptsd,
                     x = TRUE, y = TRUE)
```

Our second model

```
mod_second
```



```
Linear Regression Model

 ols(formula = ptsd ~ rcs(neg, 4) + rcs(over3, 3) + cons + aff,
     data = maleptsd)

                 Model Likelihood    Discrimination
                       Ratio Test           Indexes
 Obs       64    LR chi2     21.28    R2       0.283
 sigma1.1415     d.f.            7    R2 adj   0.193
 d.f.      56    Pr(> chi2) 0.0034    g        0.763

 Residuals

      Min       1Q   Median       3Q      Max
 -2.06529 -0.81434  0.06745  0.81760  2.17200


           Coef    S.E.   t     Pr(>|t|)
 Intercept -0.4255 0.7490 -0.57 0.5723
 neg        0.0660 0.0603  1.10 0.2780
 neg'      -0.1261 0.1641 -0.77 0.4456
 neg''      0.4924 0.5373  0.92 0.3634
 over3      0.4582 0.2007  2.28 0.0263
 over3'    -2.1247 0.9433 -2.25 0.0282
 cons      -0.0119 0.0164 -0.72 0.4722
 aff        0.1450 0.0598  2.42 0.0186
```

ANOVA for this model

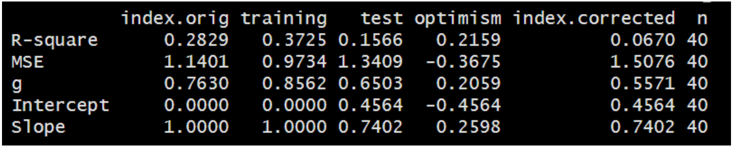
```
anova(mod_second)
```

```
                Analysis of Variance          Response: ptsd

 Factor          d.f. Partial SS MS        F    P
 neg              3   11.4062336 3.8020779 2.92 0.0420
  Nonlinear       2    1.6536591 0.8268295 0.63 0.5339
 over3            2    6.8378486 3.4189243 2.62 0.0814
  Nonlinear       1    6.6106843 6.6106843 5.07 0.0282
 cons             1    0.6826901 0.6826901 0.52 0.4722
 aff              1    7.6565797 7.6565797 5.88 0.0186
 TOTAL NONLINEAR  3    7.8079300 2.6026433 2.00 0.1248
 REGRESSION       7   28.7821644 4.1117378 3.16 0.0070
 ERROR           56   72.9688769 1.3030157
```

- Remember that this ANOVA testing is sequential.

Validation of Summary Statistics

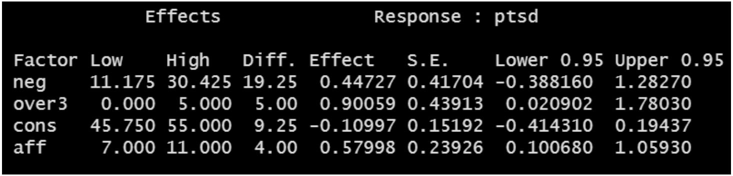
```
set.seed(432); validate(mod_second)
```



```
          index.orig training   test optimism index.corrected  n
R-square      0.2829   0.3725 0.1566   0.2159          0.0670 40
MSE           1.1401   0.9734 1.3409  -0.3675          1.5076 40
g             0.7630   0.8562 0.6503   0.2059          0.5571 40
Intercept     0.0000   0.0000 0.4564  -0.4564          0.4564 40
Slope         1.0000   1.0000 0.7402   0.2598          0.7402 40
```

summary results for mod_second

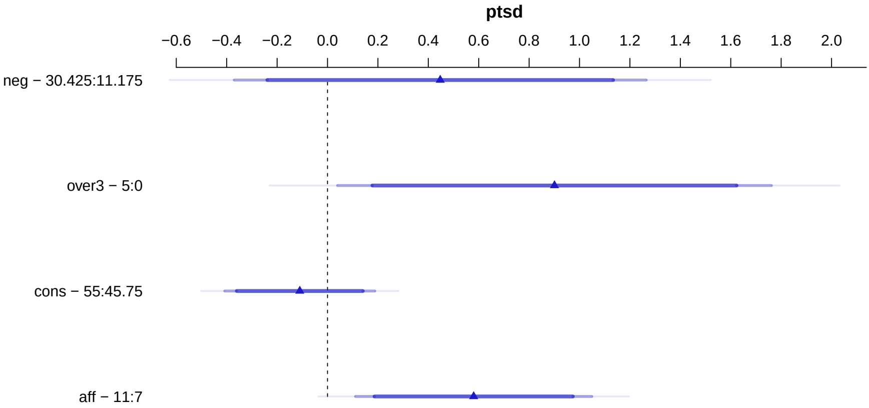
```
summary(mod_second)
```



```
             Effects              Response : ptsd

 Factor Low    High   Diff. Effect   S.E.    Lower 0.95 Upper 0.95
 neg    11.175 30.425 19.25  0.44727 0.41704 -0.388160  1.28270
 over3   0.000  5.000  5.00  0.90059 0.43913  0.020902  1.78030
 cons   45.750 55.000  9.25 -0.10997 0.15192 -0.414310  0.19437
 aff     7.000 11.000  4.00  0.57998 0.23926  0.100680  1.05930
```

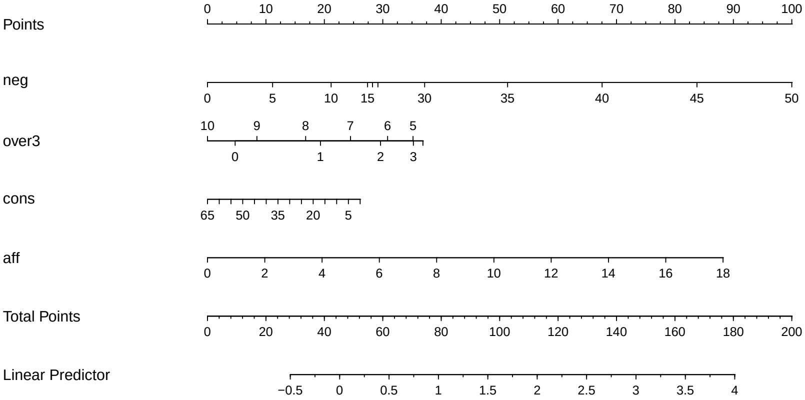
Plot of `summary` results for `mod_second`

```
plot(summary(mod_second))
```



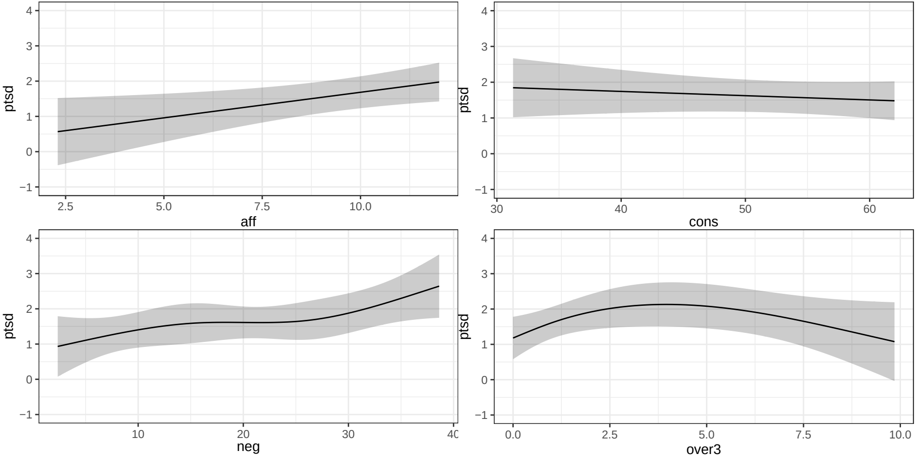
Nomogram for `mod_second`

```
plot(nomogram(mod_second))
```



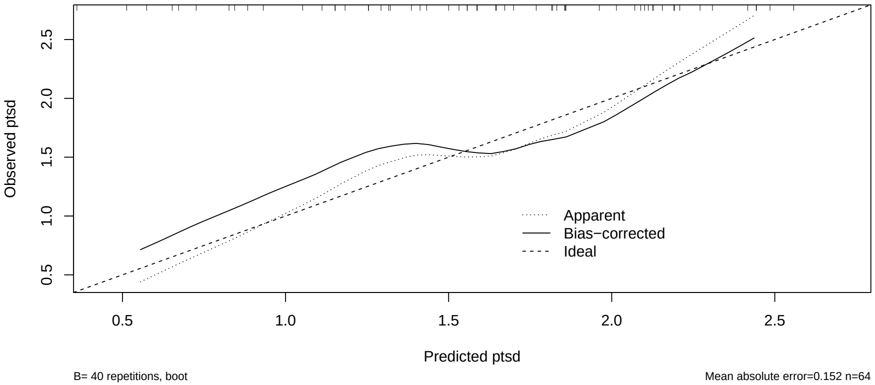
Seeing the impact of the modeling another way

```
ggplot(Predict(mod_second))
```



Checking the model's calibration

```
set.seed(432); plot(calibrate(mod_second))
```



```
n=64   Mean absolute error=0.152   Mean squared error=0.03106
0.9 Quantile of absolute error=0.279
```

Limitations of lm for fitting complex linear models

We can certainly assess this big, complex model using lm, too:

- with in-sample summary statistics like adjusted R^2, AIC and BIC,
- we can assess its assumptions with residual plots, and
- we can also compare out-of-sample predictive quality through cross-validation,

But to really delve into the details of how well this complex model works, and to help plot what is actually being fit, we'll probably want to fit the model using ols.

- In Project 1, we expect some results that are most easily obtained using lm and others that are most easily obtained using ols.

Next Time

- The HERS data
- Fitting a more complex linear regression model
  - Dealing with categorical predictors
  - Dealing with interactions (another form of non-linearity)
  - Adding missing data into all of this, and running multiple imputation

Don't forget to complete the Minute Paper after Class 7 by tomorrow at Noon!