432 Class 7 Slides

github.com/THOMASELOVE/432-2018

2018-02-06

Setup

```r
library(skimr)
library(broom)
library(Hmisc)
library(rms)
library(tidyverse)
```

Today's Materials (Chapter 9)

- Spending Degrees of Freedom on Non-Linearity
  - The Spearman ρ^2 (rho-squared) plot
- Building Non-Linear Predictors with
  - Polynomial Functions
  - Splines, including 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.

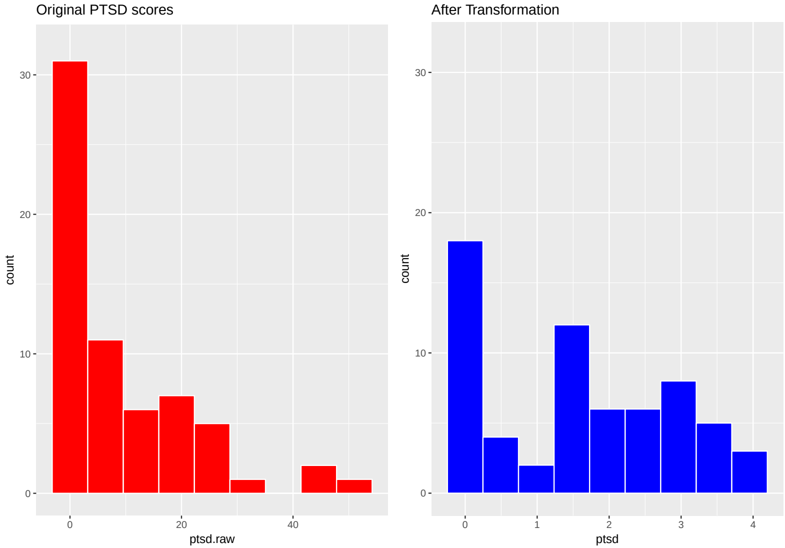
```
maleptsd <- read.csv("data/maleptsd.csv") %>% tbl_df %>%
    mutate(ptsd = log(ptsd.raw + 1))
```

[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.

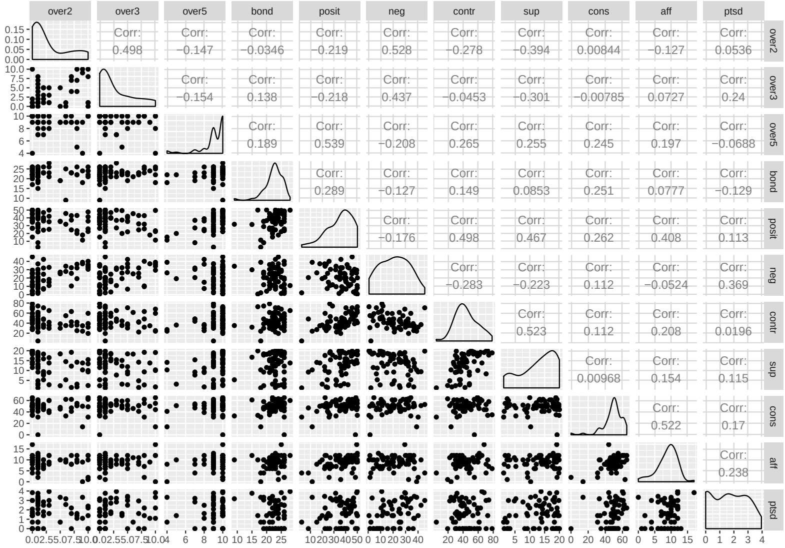
Skimming the `maleptsd` data

```
> maleptsd %>% select(-id, -ptsd.raw) %>% skim()
Skim summary statistics
 n obs: 64
 n variables: 11

Variable type: integer
 variable missing complete  n  mean    sd p0   p25 median   p75 p100     hist
      aff       0       64 64  8.84  3.08  0  7      9.5 11      17
     bond       0       64 64 22.52  3.07  9 21     23    24.25   28
     cons       0       64 64 48.98 11.15  0 45.75  51    55      65
    over2       0       64 64  2.8   3.34  0  0      1     5      10
    over3       0       64 64  2.72  3.13  0  0      1     5      10
    over5       0       64 64  9.12  1.34  4  9      9.5 10      10

Variable type: numeric
 variable missing complete  n  mean    sd  p0   p25 median   p75  p100     hist
    contr       0       64 64 44.2  14.84 4.5 35.1   41.75 53.98 78.5
      neg       0       64 64 21.09 11.6  0.7 11.17  20.65 30.42 45.4
    posit       0       64 64 35.44 11.02 2.5 27.05  37    43.35 50.1
     ptsd       0       64 64  1.59  1.27 0    0      1.61  2.74  3.95
      sup       0       64 64 13     5.87 1.2  9.28  14.25 18.3  20
```

Transformation of Outcome



Scatterplot Matrix



The Spearman ρ^2 Plot

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?

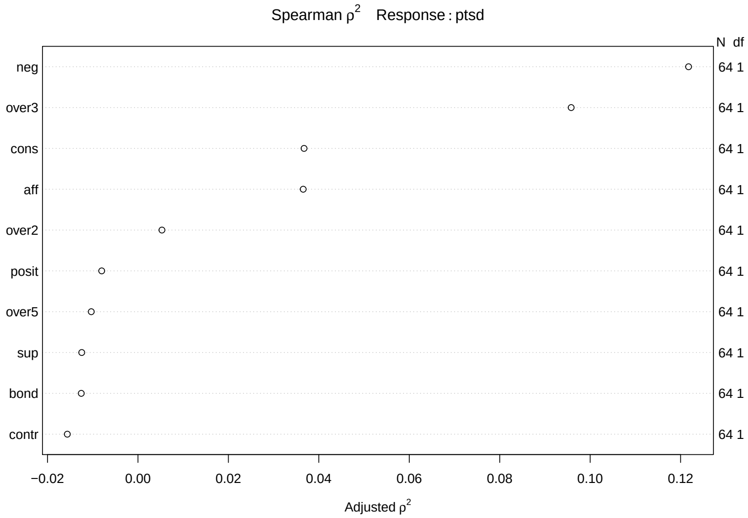
Spearman's ρ^2 is an indicator (not a perfect one) of potential predictive punch, but doesn't give away the game.

- 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

```r
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 no more than 63/15 regression inputs, including the intercept, even if all were linear.

- 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

in light of the small sample size.

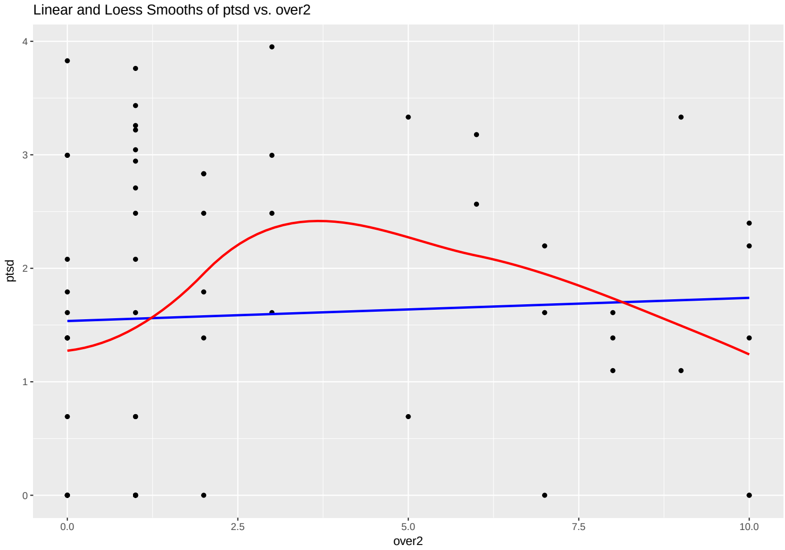
Contents of spear.ptsd

```
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
```

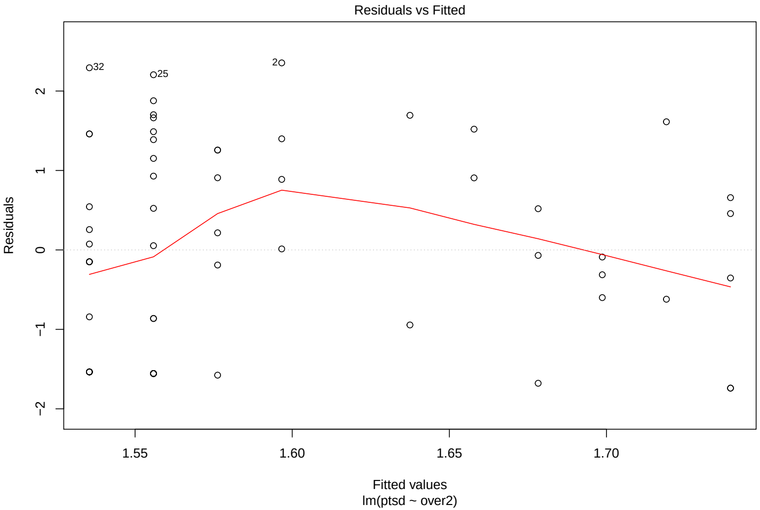
Actually Building Non-Linear Models

Predicting `ptsd` from `over2`



Linear Fit - does this work well?

```
plot(lm(ptsd ~ over2, data = maleptsd), which = 1)
```



Polynomial Regression

A polynomial in the variable `x` of degree D is a linear combination of the powers of `x` up to D.

For example:

- 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: $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4 + \beta_5 x^5$

Fitting such a model creates a **polynomial regression**.

Raw Quadratic Model for ptsd using over2

```r
modA <- lm(ptsd ~ over2 + I(over2^2), data = maleptsd)
modA
```

```

Call:
lm(formula = ptsd ~ over2 + I(over2^2), data = maleptsd)

Coefficients:
(Intercept)        over2   I(over2^2)
    1.23412      0.41128     -0.04213
```

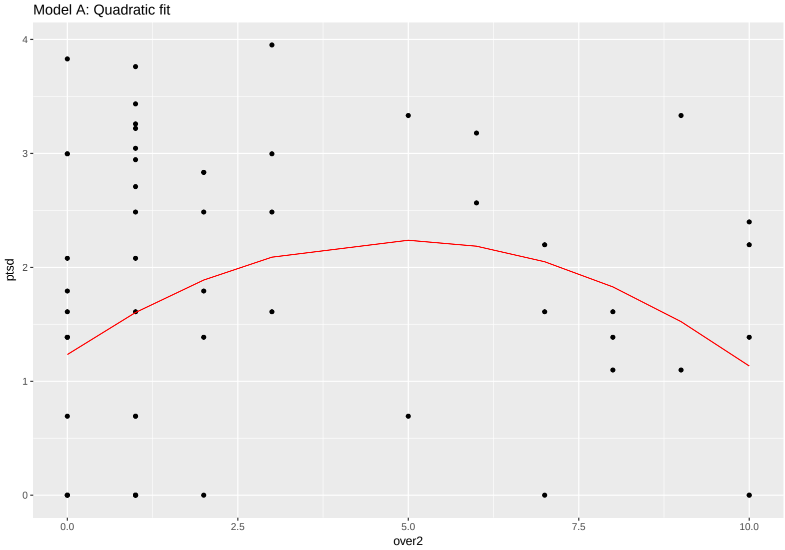
$$ptsd = 1.234 + 0.411(over2) - 0.042(over2)^2$$

Summary of Quadratic Fit

```
Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)    1.23412    0.24958   4.945 6.29e-06 ***
over2          0.41128    0.19271   2.134   0.0369 *  
I(over2^2)    -0.04213    0.02014  -2.092   0.0407 *  
```

```
Residual standard error: 1.246 on 61 degrees of freedom
Multiple R-squared:  0.0696,	Adjusted R-squared:  0.03909 
F-statistic: 2.281 on 2 and 61 DF,  p-value: 0.1108
```

Plot Fitted Values of Quadratic Fit



Code for Previous Slide

```
modA.aug <- augment(modA, maleptsd)

ggplot(modA.aug, aes(x = over2, y = ptsd)) +
    geom_point() +
    geom_line(aes(x = over2, y = .fitted),
              col = "red") +
    labs(title = "Model A: Quadratic fit")
```

Another Way to fit the Identical Model

```
modA2 <- lm(ptsd ~ pol(over2, degree = 2, raw = TRUE),
            data = maleptsd)
```

```
(Intercept)    pol(over2, degree = 2, raw = TRUE)over2  pol(over2, degree = 2, raw = TRUE)over2^2
    1.23412                                  0.41128                                   -0.04213
```

Do models give same fitted values?

```
temp <- fitted(modA2) - fitted(modA)
sum(temp != 0)
```

```
[1] 0
```

Orthogonal Polynomials

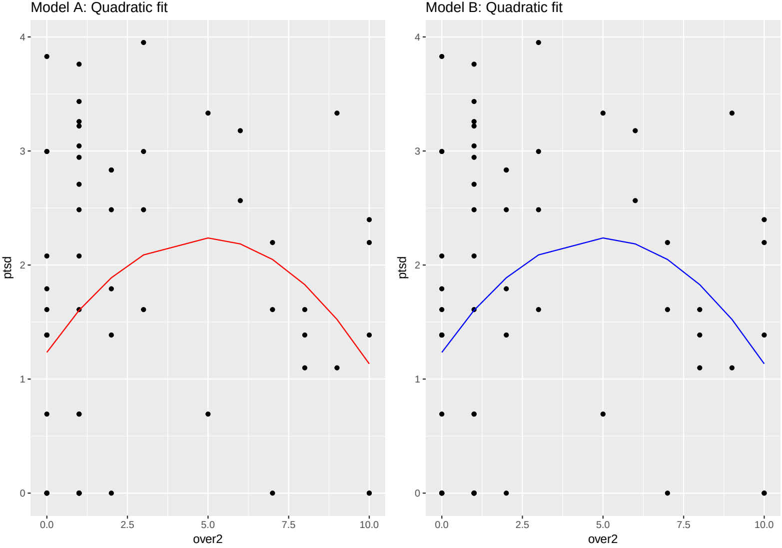
Now, let's fit an orthogonal polynomial of degree 2 to predict `ptsd` using `over2`.

```r
modB <- lm(ptsd ~ poly(over2, 2), data = maleptsd)
```

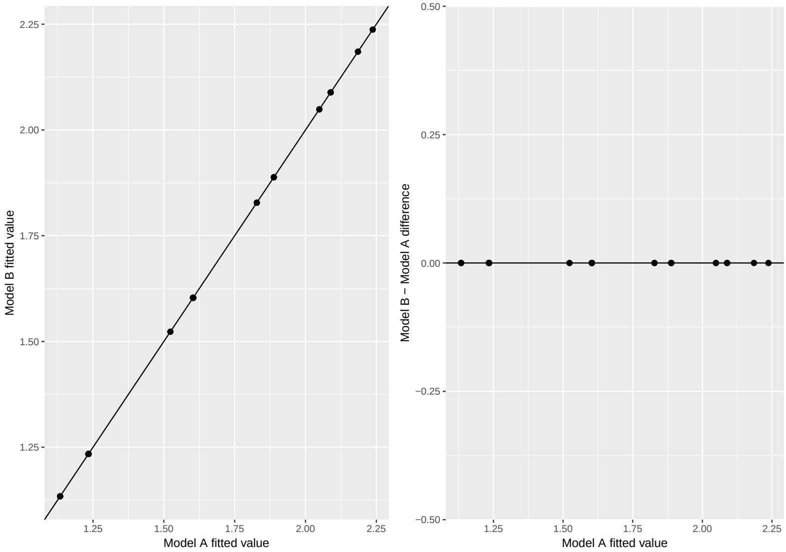
Looks very different . . .

```
> modB

Call:
lm(formula = ptsd ~ poly(over2, 2), data = maleptsd)

Coefficients:
    (Intercept)  poly(over2, 2)1  poly(over2, 2)2
         1.5925           0.5407          -2.6056
```

But it fits the same model, exactly!



Or, if you don't believe me...



Orthogonal Polynomial

An orthogonal polynomial sets up a model design matrix using the coding we've seen previously: `over2` and `over2^2` in our case, and then scales those columns so that each column is **orthogonal** to the previous ones. - Two columns are orthogonal if their correlation is zero.

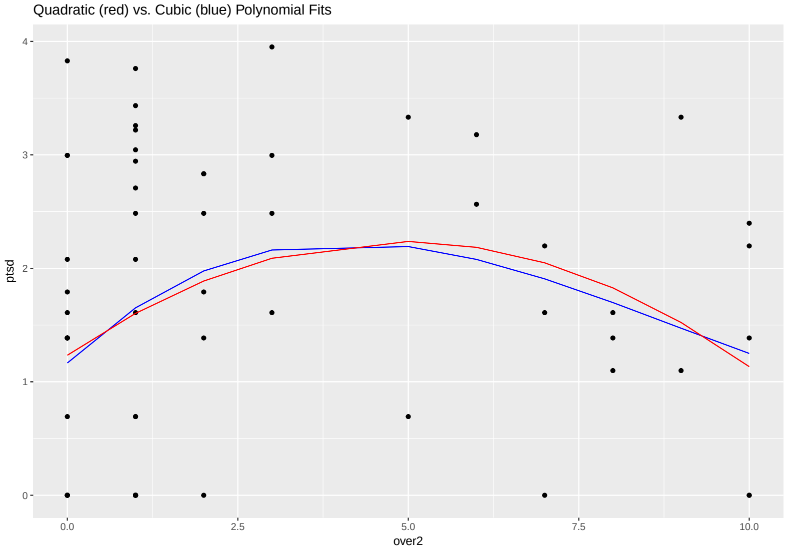
This eliminates the collinearity (correlation between predictors) and lets our t tests tell us whether the addition of any particular polynomial term improves the fit of the model over the lower orders.

Would adding a cubic term help predict `ptsd`?

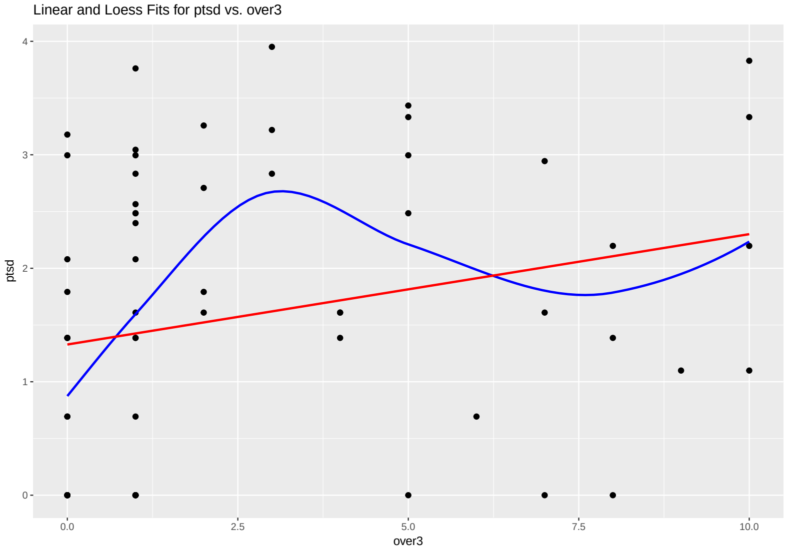
```r
modC <- lm(ptsd ~ poly(over2, 3), data = maleptsd)
```

```
> round(summary(modC)$coef, 3)
                Estimate Std. Error t value Pr(>|t|)
(Intercept)        1.593      0.157  10.164    0.000
poly(over2, 3)1    0.541      1.253   0.431    0.668
poly(over2, 3)2   -2.606      1.253  -2.079    0.042
poly(over2, 3)3    0.636      1.253   0.508    0.614
>
```

Comparing Quadratic (red) and Cubic (blue) Models



What if we look instead at `over3` as a predictor?



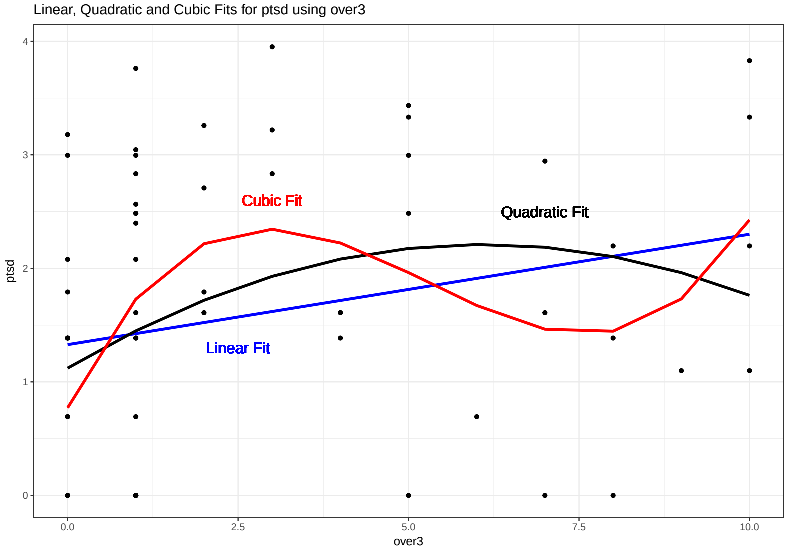
What if we predict using `over3`?

```
modD1 <- lm(ptsd ~ over3, data = maleptsd)
modD2 <- lm(ptsd ~ poly(over3, degree = 2), data = maleptsd)
modD3 <- lm(ptsd ~ poly(over3, degree = 3), data = maleptsd)
```

```
> round(summary(modD1)$coef, 3)
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    1.328      0.206   6.432    0.000
over3          0.097      0.050   1.945    0.056
```

```
> round(summary(modD3)$coef, 3)
                        Estimate Std. Error t value Pr(>|t|)
(Intercept)                1.593      0.146  10.907    0.000
poly(over3, degree = 3)1   2.419      1.168   2.071    0.043
poly(over3, degree = 3)2  -1.906      1.168  -1.631    0.108
poly(over3, degree = 3)3   3.225      1.168   2.761    0.008
```

Plotting the Fitted Models



Using Restricted Cubic Splines to Capture Non-Linearity

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 `lm` and `rcs`

For most applications, three to five knots strike a nice balance between complicating the model needlessly and fitting data pleasingly. Let's consider a restricted cubic spline model for `ptsd` based on `over3` again, but now with:

- in `modE3`, 3 knots, and
- in `modE4`, 4 knots,

```
modE3 <- lm(ptsd ~ rcs(over3, 3), data = maleptsd)
modE4 <- lm(ptsd ~ rcs(over3, 4), data = maleptsd)
```

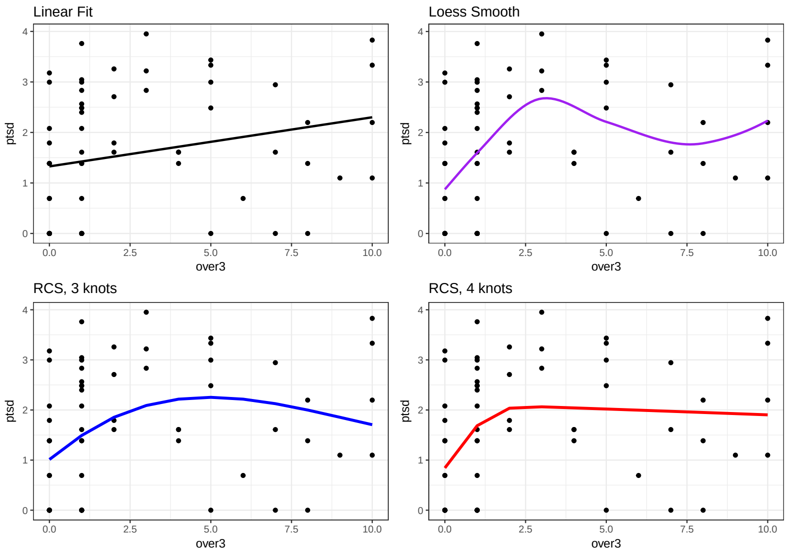
Summarizing the 4-knot model coefficients

```
> round(summary(modE4)$coef, 3)
                        Estimate Std. Error t value Pr(>|t|)
(Intercept)                0.843      0.290   2.908    0.005
rcs(over3, 4)over3         0.953      0.432   2.209    0.031
rcs(over3, 4)over3'       -8.914      5.982  -1.490    0.141
rcs(over3, 4)over3''      13.480      9.635   1.399    0.167
```

and where are the knots located?

```
> attributes(rcs(maleptsd$over3, 4))$parms
[1] 0.00 1.00 2.95 9.00
```

Plotting the spline models



Does the fit improve markedly from 3 to 4 knots?

In-sample comparison via ANOVA

```
anova(modE3, modE4)
```

```
Analysis of Variance Table

Model 1: ptsd ~ rcs(over3, 3)
Model 2: ptsd ~ rcs(over3, 4)
  Res.Df    RSS Df Sum of Sq      F Pr(>F)
1     61 89.598
2     60 87.573  1    2.0246 1.3871 0.2435
```

Does the fit improve markedly from 3 to 4 knots?

In-Sample comparisons of information criteria, etc.

```
glance(modE3)
```

```
  r.squared adj.r.squared    sigma statistic    p.value df
1 0.1194384    0.09056753 1.211949  4.136986 0.02066199  3
     logLik     AIC      BIC deviance df.residual
1 -101.5785 211.157 219.7925 89.59806          61
```

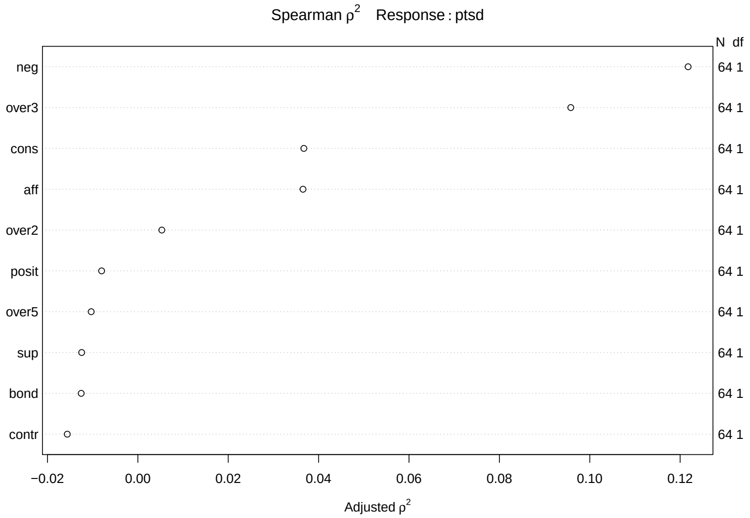
```
glance(modE4)
```

```
  r.squared adj.r.squared    sigma statistic    p.value df
1 0.1393362      0.096303 1.208121  3.237877 0.02828895  4
     logLik      AIC      BIC deviance df.residual
1 -100.8471 211.6942 222.4886 87.57344          60
```

Back to the Spearman's ρ^2 Plot

Spearman's ρ^2 Plot

```r
plot(spear.ptsd)
```



Proposed New Model F

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`

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

Fit model F

```r
modelF <- lm(ptsd ~ rcs(neg, 4) + rcs(over3, 3) +
                cons + aff, data = maleptsd)
```

```
> round(summary(modelF)$coeff, 3)
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)           -0.425      0.749  -0.568    0.572
rcs(neg, 4)neg         0.066      0.060   1.095    0.278
rcs(neg, 4)neg'       -0.126      0.164  -0.768    0.446
rcs(neg, 4)neg''       0.492      0.537   0.916    0.363
rcs(over3, 3)over3     0.458      0.201   2.283    0.026
rcs(over3, 3)over3'   -2.125      0.943  -2.252    0.028
cons                  -0.012      0.016  -0.724    0.472
aff                    0.145      0.060   2.424    0.019
```

ANOVA for Model F

```
anova(modelF)
```

```
Analysis of Variance Table

Response: ptsd
               Df Sum Sq Mean Sq F value  Pr(>F)
rcs(neg, 4)     3 14.597  4.8657  3.7342 0.01617 *
rcs(over3, 3)  2  5.892  2.9460  2.2609 0.11369
cons            1  0.636  0.6365  0.4885 0.48751
aff             1  7.657  7.6566  5.8760 0.01860 *
Residuals      56 72.969  1.3030
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Remember that this ANOVA testing is sequential.

Is Model F better than Model E3?

```
anova(modelF, modE3)
```

```
Analysis of Variance Table

Model 1: ptsd ~ rcs(neg, 4) + rcs(over3, 3) + cons + aff
Model 2: ptsd ~ rcs(over3, 3)
  Res.Df    RSS Df Sum of Sq      F  Pr(>F)
1     56 72.969
2     61 89.598 -5   -16.629 2.5524 0.03769 *
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Limitations of `lm` for fitting complex linear models

We can certainly assess this big, complex model using `lm` in comparison to other models:

- 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 an alternative method for fitting linear models, called `ols`, from the `rms` package developed by Frank Harrell and colleagues. That's Chapter 10.

Next Time

- Logistic Regression (Chapters 12 and some of 13)
- We'll return to Chapter 10 next week.
- We'll get to Chapter 11 (lasso and ridge regression) eventually.