Regression III

Splines

Dave Armstrong

Goals for Today

1. Develop the idea of splines as piecewise regression functions.
    - Motivate with piecewise linear model.
2. Consider truncated power basis functions for cubic regression splines.
    - How and where should knots be placed?
3. Discuss other more robust bases - B-splines.
4. Show an example of regression splines at work.

Definition of Splines

Splines are:

> ... piecewise regression functions we constrain to join at points called knots (Keele 2007, 70)

- In their simplest form, they are dummy regressors that we use to force the regression line to change direction at some value(s) of X.
- These are similar in spirit to LPR models where we use a subset of data to fit local regressions (but the window doesn't move here).
- These are also allowed to take any particular functional form, but they are a bit more constrained than the LPR model.

Notes

Type notes here...

Splines vs. LPR Models

Splines provide a better MSE fit to the data.

- Where $MSE\left(\hat{\theta}\right) = \mathrm{Var}\left(\hat{\theta}\right) + \mathrm{Bias}\left(\hat{\theta}, \theta\right)^2$
- Generally, LPR models will have smaller bias, but much greater variance.
- Splines can be designed to prevent over-fitting (smoothing splines)
- Splines are more easily incorporated in *semi*-parametric models.

Notes

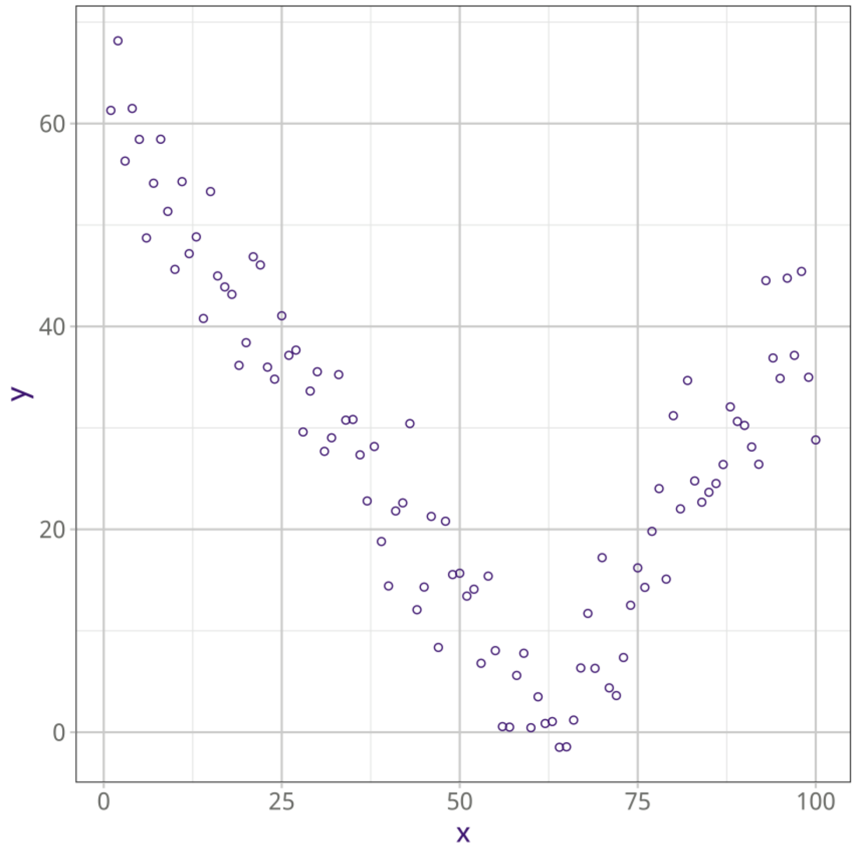
Type notes here...

Regression Splines

We start with the following familiar model:

$$y = f(x) + \varepsilon$$

Here, we would like to estimate this with one model rather than a series of local models.

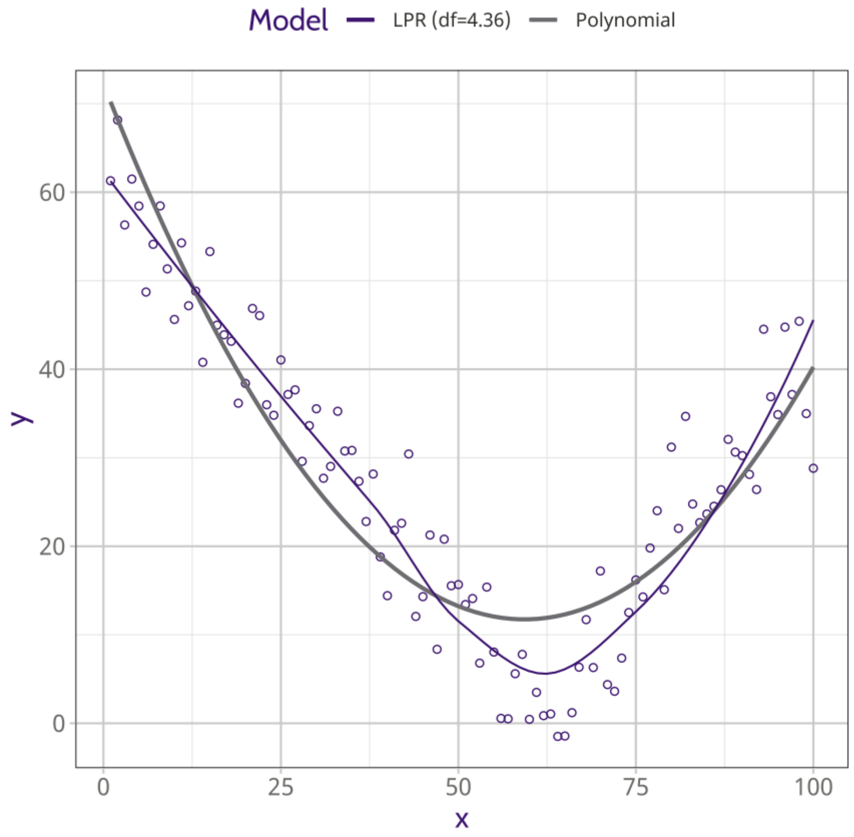


Notes

Type notes here...

Failure of Polynomials and LPR

Given what we already learned, we could fit a quadratic polynomial or a LPR:



Notes

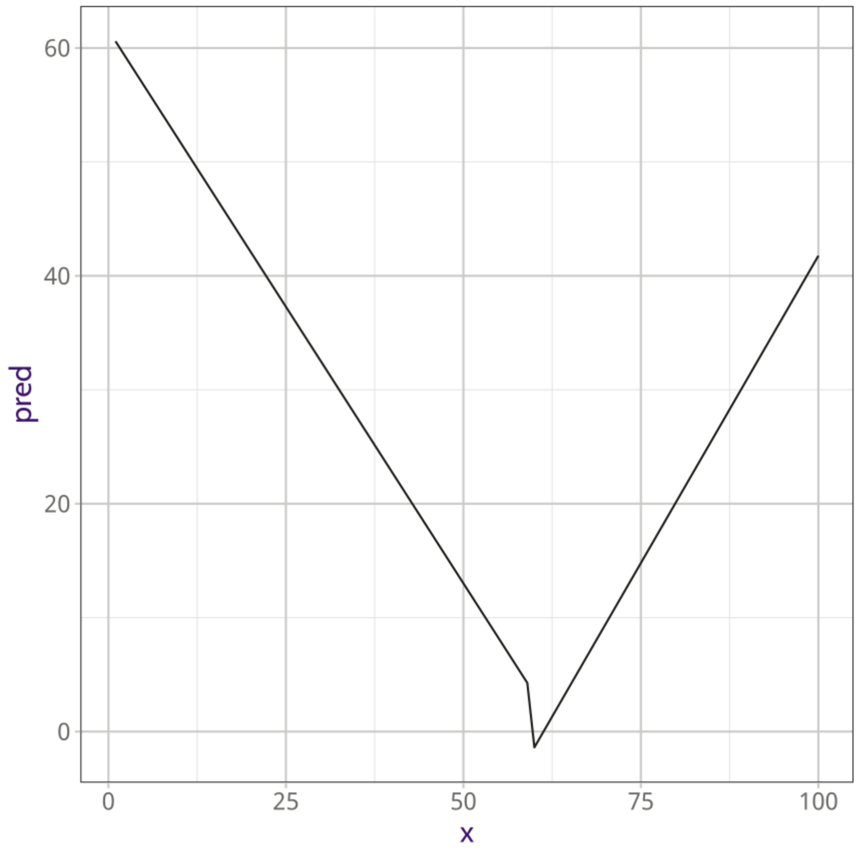
Type notes here...

Dummy Interaction

You might ask, couldn't we just use an interaction between x and a dummy variable coded 1 if $x > 60$ and zero otherwise.

$$y = b_0 + b_1 x_1 + b_2 d + b_3 x \times d + e$$

This seems like a perfectly reasonable thing to do. What can it give you though:



Notes

Type notes here...

Basis Functions

A basis function is really just a function that transforms the values of X. So, instead of estimating:

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$$

we estimate:

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \ldots + \beta_k b_k(x_i) + \varepsilon_i$$

The basis functions $b_k(\cdot)$ are known ahead of time (not estimated by the model).

- We can think of polynomials as basis functions where $b_j(x_i) = x_i^j$

Notes

Type notes here...

Piecewise Polynomials

One way that we can think about regression splines is as piecewise polynomial functions:

$$y_i = \begin{cases} \beta_{01} + \beta_{11}x_i + \beta_{21}x_i^2 + \beta_{31}x_i^3 + \varepsilon_i & x_i < c \\ \beta_{02} + \beta_{12}x_i + \beta_{22}x_i^2 + \beta_{32}x_i^3 + \varepsilon_i & x_i \geq c \end{cases}$$

Just as above though, these polynomials are unconstrained and can generate a discontinuity at the *knot* location c.

Notes

Type notes here...

Constraining the Model

To constrain the model, the splines are constructed:

- such that the first and second derivatives of the function continuous.
- Each constraint reduces the number of degrees of freedom we use by one.
- In general, the model uses: Polynomial Degree $+$ # Knots $+$ 1 (for the intercept) degrees of freedom

Notes

Type notes here...

Truncated Power Basis Functions

The easiest set of Spline functions to consider (for knot location k) are called truncated power functions, defined as:

$$h(x, k) = (x - k)^3_+ = \begin{cases} (x-k)^3 & \text{if } x > k \\ 0 & \text{otherwise} \end{cases}$$

When using these basis functions in, we put the full (i.e., global) parametric function in and a truncated power function of degree n for each knot.

Notes

Type notes here...

Linear Truncated Power Functions

To use the truncated power basis for our problem, we need:

- The global linear model
- One truncated power function for the x values greater than the knot location (60).

$$y = b_0 + b_1 x + b_2 (x - 60)^1_+ + e$$

This sets up essentially 2 equations:

$$\begin{aligned} x \leq 60 :& y = b_0 + b_1 x \\ x > 60 :& y = b_0 + b_1 x + b_2 (x - 60) = (b_0 - 60 b_2) + (b_1 + b_2) x \end{aligned}$$

Notice that here we are only estimating 3 parameters, where the interaction would estimate 4 parameters. Thus, this is a constrained version of the interaction.

Notes

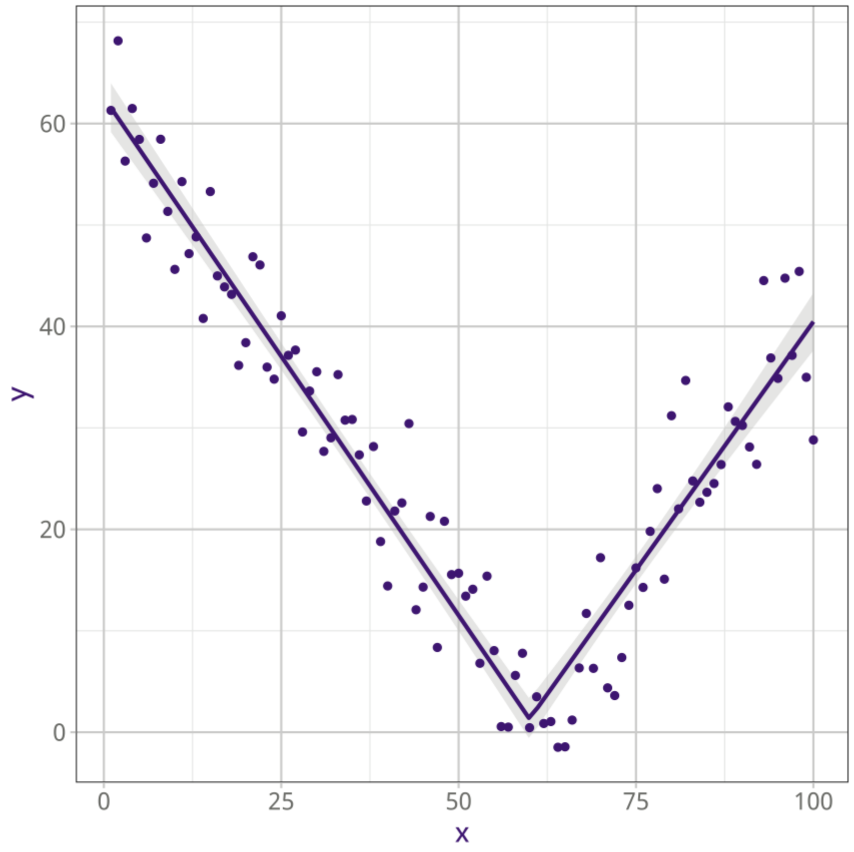
Type notes here...

Fixing the Discontinuity

Including x and $(x - 60)_+$ as regressors, which generates the following predictions:

```r
pwl <- function(x, k)ifelse(x >= k, x-k, 0)

ggplot(mapping=aes(x=x, y=y)) +
  geom_smooth(method="lm",
              formula=y ~ x + pwl(x, 60)) +
  geom_point() +
  theme_bw() +
  mytheme()
```



Notes

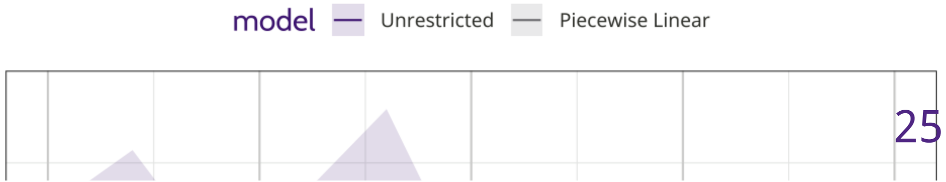
Type notes here...

Polity Example

Thinking back to the Polity example from Lecture 4. We suggested we maybe could fit a piecewise polynomial model:

```
library(foreign)
dat <- read.dta("http://www.quantoid.net/files/reg3/linear_ex.dta
dat$polity_dem_fac <- as.factor(dat$polity_dem)
unrestricted.mod <- lm(rep1 ~ polity_dem_fac + iwar +
    cwar + logpop + gdppc,data=dat)
pwlin.mod <- lm(rep1 ~ polity_dem + pwl(polity_dem, 9) +
                iwar + cwar + logpop + gdppc,data=dat)
anova(pwlin.mod, unrestricted.mod, test="F")
```

```
## Analysis of Variance Table
##
## Model 1: rep1 ~ polity_dem + pwl(polity_dem, 9) + iwar + cwar + logpop +
##     gdppc
## Model 2: rep1 ~ polity_dem_fac + iwar + cwar + logpop + gdppc
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1   2676 2172.9
## 2   2668 2163.3  8     9.651 1.4878 0.1562
```



Notes

Type notes here...

Unknown Knot Location

If you don't know the knot location, you could try a bunch of different options.

```
cvfun <- function(split, ...){
  mods <- lapply(1:9, function(i){
    lm(rep1 ~ polity_dem + pwl(polity_dem, i) +
         iwar + cwar + logpop + gdppc,
         data= analysis(split))})
yhat <- sapply(mods, function(x)predict(x, newdata=assessment(spl
y <- assessment(split)$rep1
e <- apply(yhat, 2, function(z)(y-z)^2)
sume2 <- colSums(e)
n <- length(y)
tibble(knot = 1:9, e2 = sume2, n = rep(n, 9))
}
out <- dat %>%
  vfold_cv(v=10, repeats=3) %>%
  mutate(err = map(splits, cvfun)) %>%
  unnest(err)  %>%
  group_by(id, knot) %>%
  summarise(mse = sum(e2)/sum(n)) %>%
  ungroup %>%
  group_by(knot) %>%
  summarise(mse = mean(mse))
```

```
aics <- sapply(1:9, function(i)
  AIC(lm(rep1 ~ polity_dem + pwl(polity_dem, i) +
         iwar + cwar + logpop + gdppc,
         data= dat)))
out <- out %>%
  mutate(aic = aics)

out
```

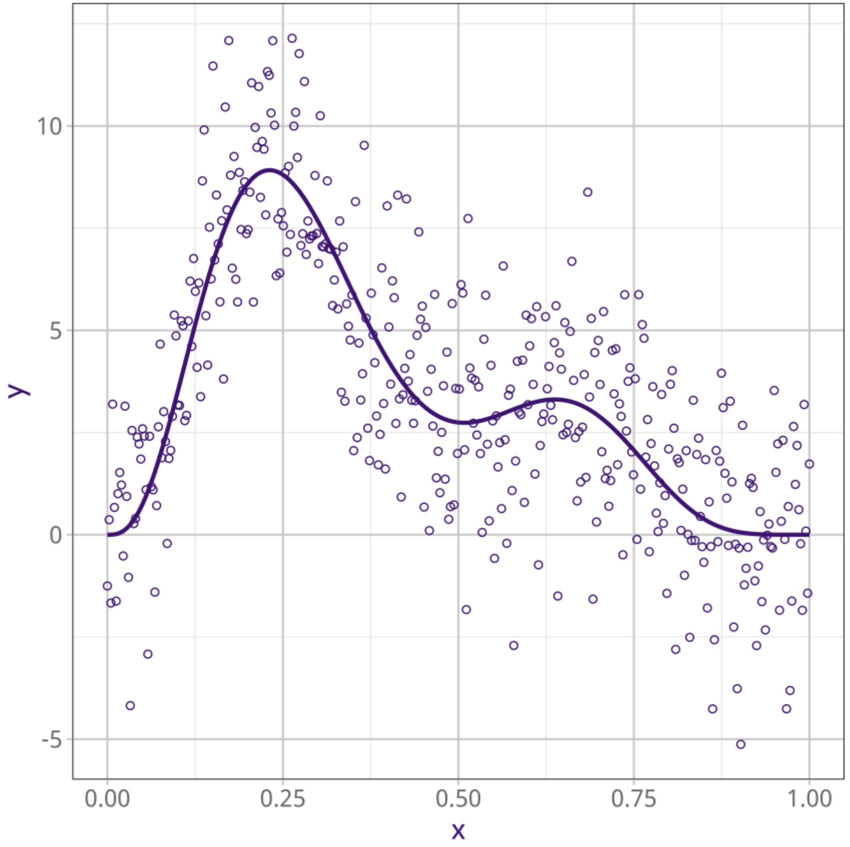
```
## # A tibble: 9 x 3
##    knot   mse   aic
##   <int> <dbl> <dbl>
## 1     1 0.934 7431.
## 2     2 0.929 7417.
## 3     3 0.919 7387.
## 4     4 0.907 7354.
## 5     5 0.900 7331.
## 6     6 0.887 7293.
## 7     7 0.868 7235.
## 8     8 0.840 7145.
## 9     9 0.815 7064.
```

Notes

Type notes here...

Example: Cubic Spline

Consider the following relationship:



Notes

Type notes here...

Cubic Spline

$$y = b_0 + b_1 x + b_2 x^2 + b_3 x^3 + \sum_{m-1}^{\#\text{ knots}} b_{k+3}(x - k_m)_+^3$$

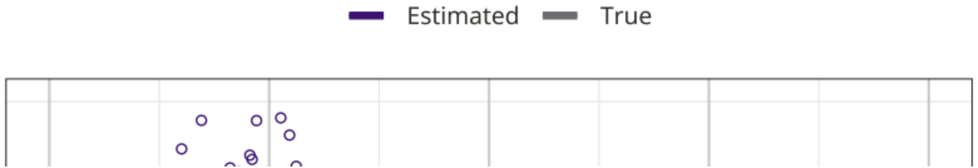
Let's consider our example with 3 knots $k = \{.2, .4, .6, .8\}$

```
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                8.602e-02  6.836e-01   0.126   0.8999
## x                         -3.885e+00  1.894e+01  -0.205   0.8376
## I(x^2)                     5.772e+02  1.386e+02   4.164 3.84e-05 ***
## I(x^3)                    -1.703e+03  2.877e+02  -5.921 6.99e-09 ***
## I((x - k[1])^3 * (x >= k[1]))  2.771e+03  3.789e+02   7.314 1.48e-12 ***
## I((x - k[2])^3 * (x >= k[2])) -1.474e+03  1.821e+02  -8.094 7.36e-15 ***
## I((x - k[3])^3 * (x >= k[3]))  3.866e+02  1.821e+02   2.123   0.0344 *
## I((x - k[4])^3 * (x >= k[4]))  7.080e+02  3.789e+02   1.869   0.0624 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard deviation: 1.951 on 392 degrees of freedom
## Multiple R-squared: 0.6665
## F-statistic: 111.9 on 7 and 392 DF,  p-value: < 2.2e-16
##     AIC     BIC
## 1679.71 1715.63
```

Notes

Type notes here...

Predictions



Notes

Type notes here...

Problems with Truncated Power Basis Functions

- Highly collinear and can lead to instability and singularities (i.e., computationally bad stuff) at worst.
- Not as "local" as some other options, the support of the piecewise functions can be over the whole range of the data or nearly the whole range of the data.
- Can produce erratic tail behavior.

Other basis functions, like the B-spline basis functions solve all of these problems:

- Reduces collinearity (though doesn't eliminate it)
- Support of the function is more narrowly bounded.
- Uses knots at the boundaries of x and assumes linearity beyond the knots.

Notes

Type notes here...

Example: B-spline

```r
library(splines)
k <- c(.2,.4,.6,.8)
csmod2 <- lm(y ~ bs(x, knots=k))
S(csmod2, brief=TRUE)
```

```
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)         0.08602    0.68358   0.126   0.8999
## bs(x, knots = k)1 -0.25898    1.26296  -0.205   0.8376
## bs(x, knots = k)2 14.61385    0.80729  18.102  < 2e-16 ***
## bs(x, knots = k)3  1.33876    0.96742   1.384   0.1672
## bs(x, knots = k)4  3.73773    0.83755   4.463 1.06e-05 ***
## bs(x, knots = k)5  2.30614    1.01655   2.269   0.0238 *
## bs(x, knots = k)6 -1.83334    0.99321  -1.846   0.0657 .
## bs(x, knots = k)7  0.80657    0.97507   0.827   0.4086
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard deviation: 1.951 on 392 degrees of freedom
## Multiple R-squared: 0.6665
## F-statistic: 111.9 on 7 and 392 DF,  p-value: < 2.2e-16
##     AIC     BIC
## 1679.71 1715.63
```

Notice that the fit here is precisely the same as with the the truncated power basis functions

```r
p1 <- predict(csmod, se=TRUE)
p2 <- predict(csmod2, se=TRUE)
cor(p1$fit, p2$fit)
```

```
## [1] 1
```

```r
cor(p1$se.fit, p2$se.fit)
```

```
## [1] 1
```

Notes

Type notes here...

Interpreting Spline Coefficients

So, how do you interpret the spline coefficients?

- You don't.
- Remember that these are all functions of x, so we cannot change the values of one component of the basis function while holding the others constant, the others would have to change, too.

Notes

Type notes here...

Choices in Spline Models

- **Degree**: the analyst has to choose the degree of the polynomial fit to the subsets of the data.
- **Number of knots**: the analyst has to choose the number of knots
- **Location of knots**: Often, knots are spaced evenly over the support of the data (i.e., the range of x), but that needn't be the case.
  - Knot placement can be guided by theory if possible.
  - Otherwise, for the functions we generally need to estimate, a few knots should probably work just fine.

Notes

Type notes here...

How important is knot placement?

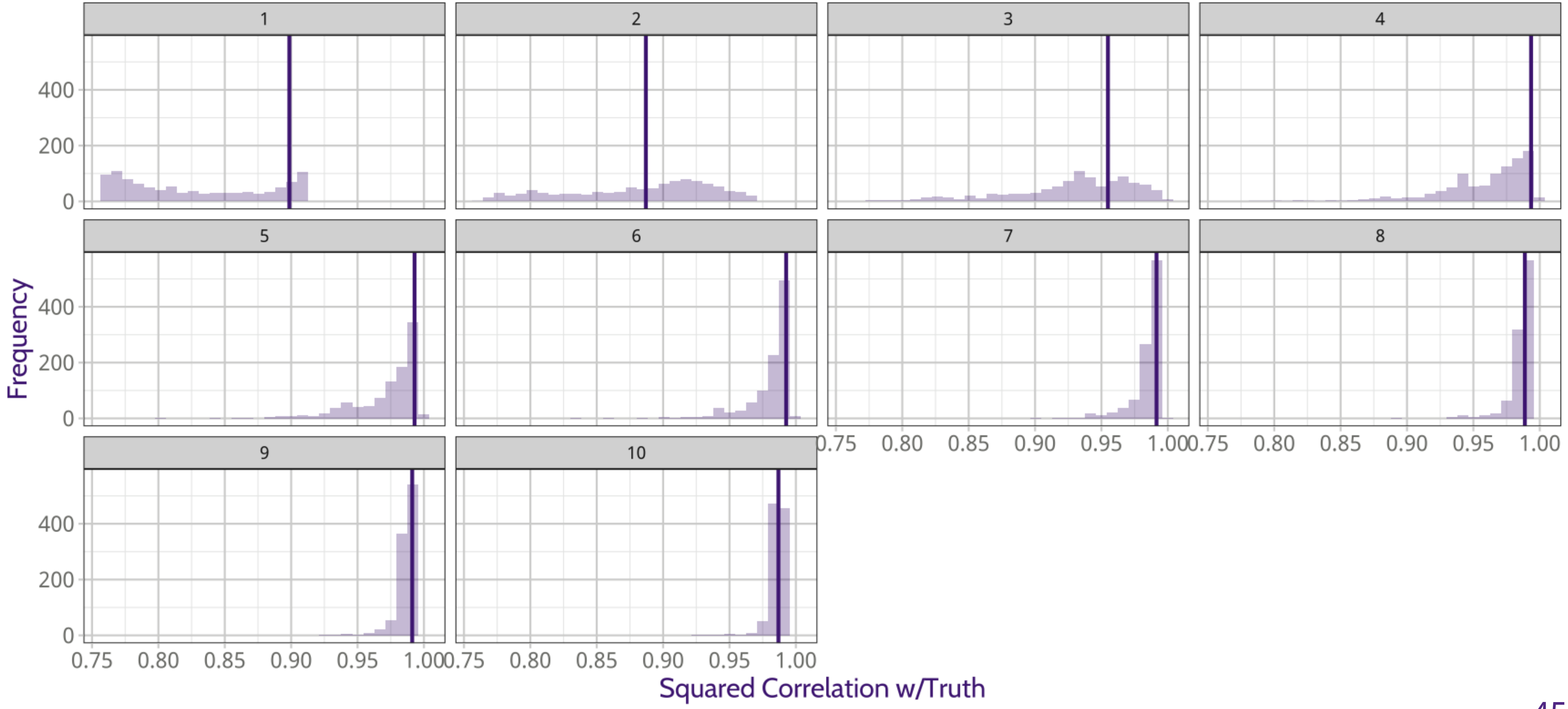
I did a simulation where I did the following:

- Using the function created above, I first estimated a B-spline with 1-10 knots.
    - Then, I calculated the R^2 with respect to the true point location.
- For each number of knots, randomly draw knots from a uniform distribution.
    - Estimate the model and calculate R^2 with respect to the truth.

Notes

Type notes here...

Results



1
2
3
4
5
6
7
8
9
10
400
200
0
Frequency
0.75
0.80
0.85
0.90
0.95
1.00
Squared Correlation w/Truth

Notes

Type notes here...

How Important is Knot Placement? II

- So long as the polynomial degree is reasonably high (3 should be high enough for what we do, but 4 might be useful if you have a very complicated function), knot placement is not particularly important.
- Use theory, if it exists, to place knots.
- If theory doesn't exist, knots placed evenly across the range of $\mathbf{x}$ will, in general, minimize error.
- If you think about the knots as random variables (because we don't know their values) and further that they are distributed uniformly (i.e., neither middle or extreme values are more likely), then technically evenly spaced knots minimize distance to the true, but unknown knots.

Notes

Type notes here...

How Important is Polynomial Degree?

- Pretty important, particularly if we don't know or have a really good sense of where the knots should be.
- B-splines are more forgiving of knot placement errors the higher the polynomial degree.
- Generally no good reason to use something more restrictive than a cubic spline.
- We are generally not trying to model particularly complicated functions.
- More knots are more likely to be used than a higher polynomial degree to make the function more flexible.

Notes

Type notes here...

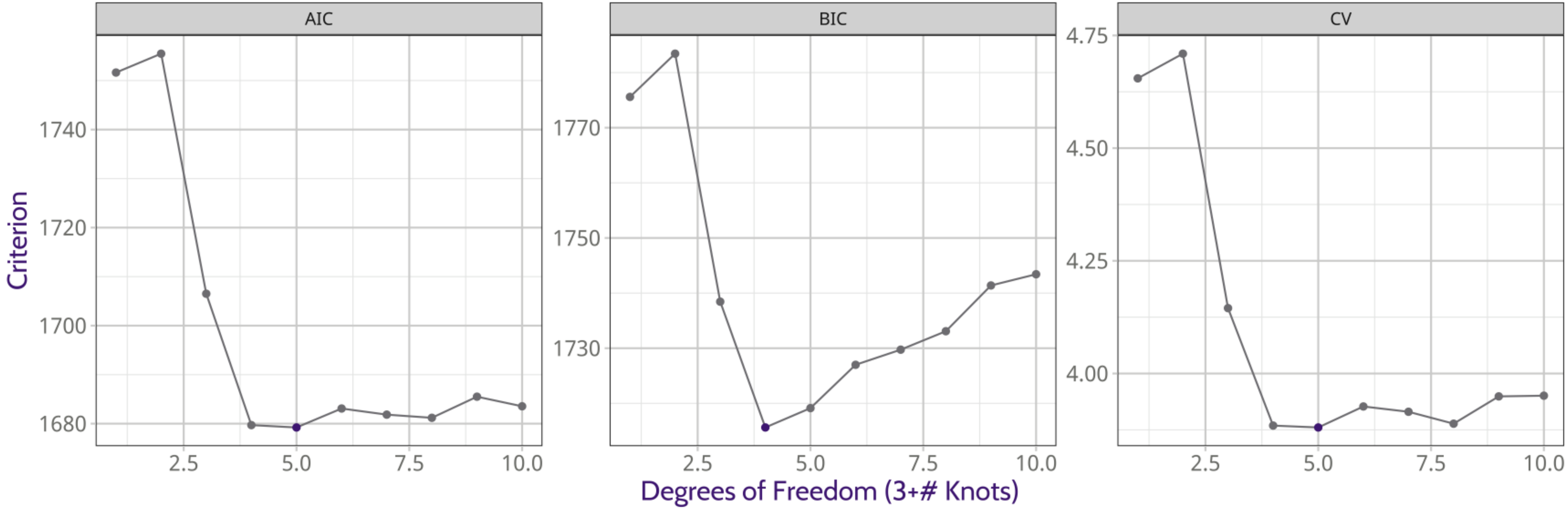
How Important is the Number of Knots

- Flexibility increases with number of knots and polynomial degree.
- Increasing number of knots can make the function more flexible.
- We can use AIC, BIC or Cross-Validation to choose number of knots.

Notes

Type notes here...

Choosing Number of Knots



Notes

Type notes here...

Worked Example

```r
library(car)
library(rio)
dat <- import(
   "http://www.quantoid.net/files/reg3/jacob.dta")

rawlm <- lm(chal_vote ~ perotvote + chal_spend +
    exp_chal, data=dat)
```

Notes

Type notes here...

Raw Model

```
S(rawlm, brief=TRUE)
```

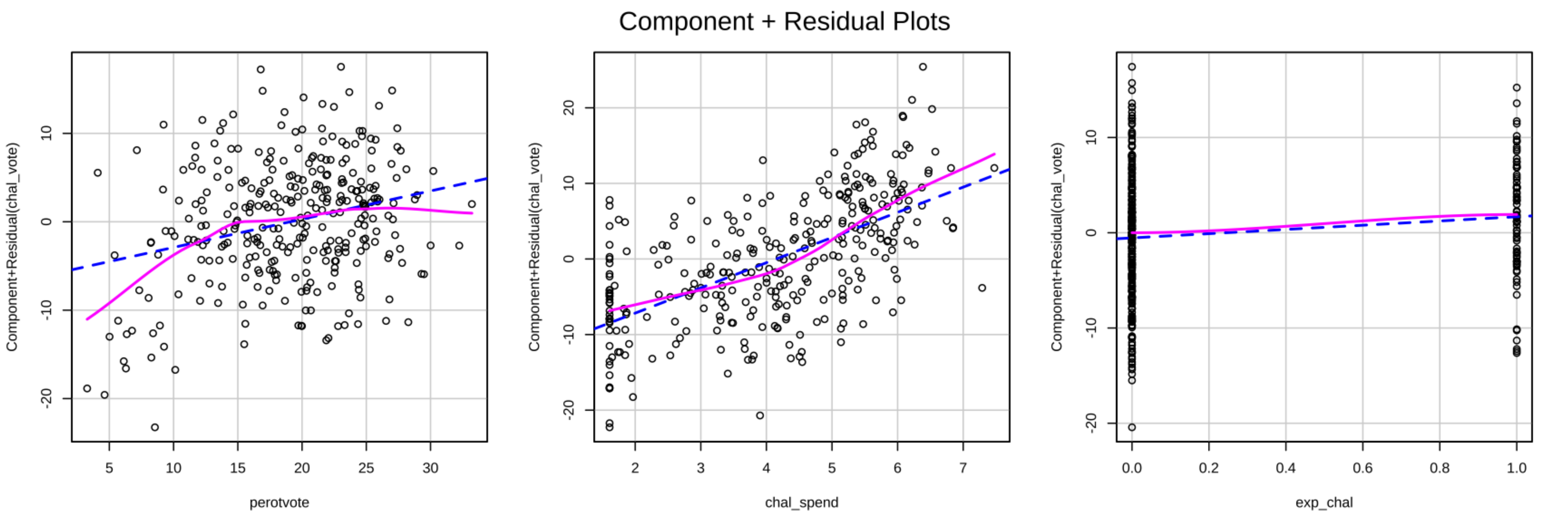
```
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 15.95365    1.59703   9.990  < 2e-16 ***
## perotvote    0.31943    0.06655   4.800 2.48e-06 ***
## chal_spend   3.33294    0.27869  11.959  < 2e-16 ***
## exp_chal     2.22053    0.98576   2.253    0.025 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard deviation: 6.74 on 308 degrees of freedom
## Multiple R-squared: 0.4552
## F-statistic:  85.8 on 3 and 308 DF,  p-value: < 2.2e-16
##     AIC     BIC
## 2082.02 2100.74
```

Notes

Type notes here...

C+R Plots

```
crPlots(rawlm, layout=c(1,3))
```



Notes

Type notes here...

Transformation Model

```
boxTidwell(chal_vote ~ perotvote,
           ~ chal_spend + exp_chal, data=dat)
```

```
##  MLE of lambda Score Statistic (z)  Pr(>|z|)
##       -1.0634             -4.1129 3.908e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## iterations =  9
```

```
trans.mod <- lm(chal_vote ~ I(1/perotvote) +
                  chal_spend + exp_chal, data=dat)
S(trans.mod, brief=TRUE)
```

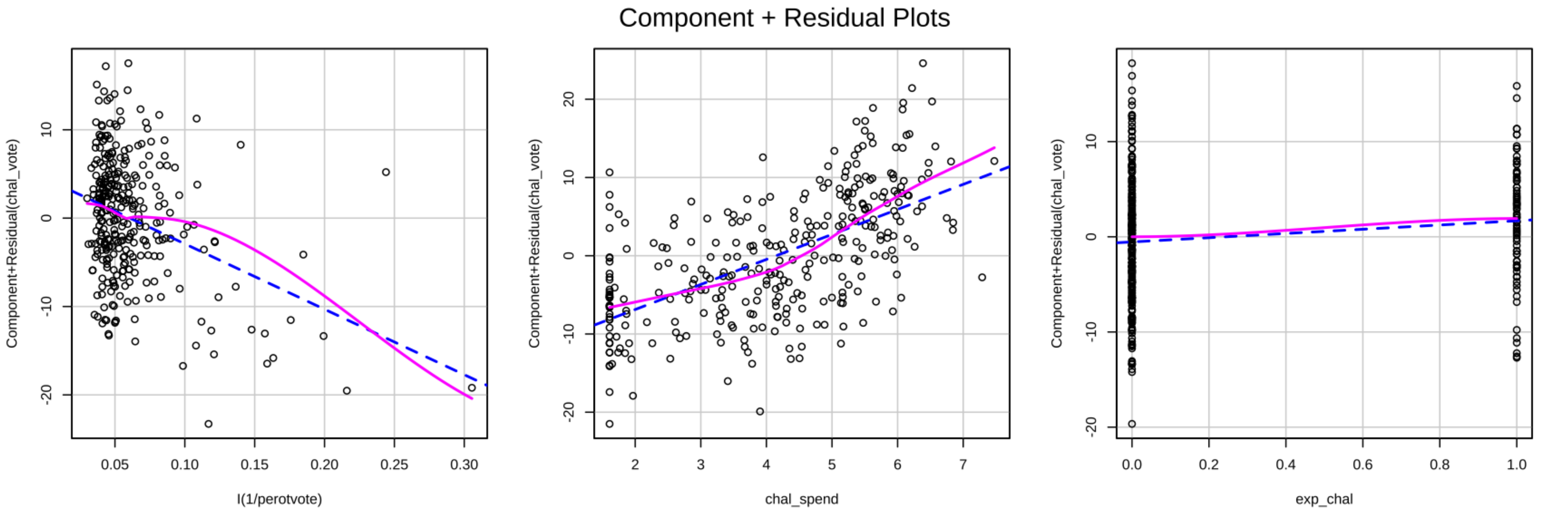
```
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     27.0997     1.4276  18.983  < 2e-16 ***
## I(1/perotvote) -74.0400    11.6678  -6.346  7.9e-10 ***
## chal_spend       3.1989     0.2737  11.687  < 2e-16 ***
## exp_chal         2.2218     0.9610   2.312   0.0214 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard deviation: 6.571 on 308 degrees of freedom
## Multiple R-squared: 0.4822
## F-statistic:  95.6 on 3 and 308 DF,  p-value: < 2.2e-16
##     AIC     BIC
```

Notes

Type notes here...

C+R Plots

```
crPlots(trans.mod, layout=c(1,3))
```

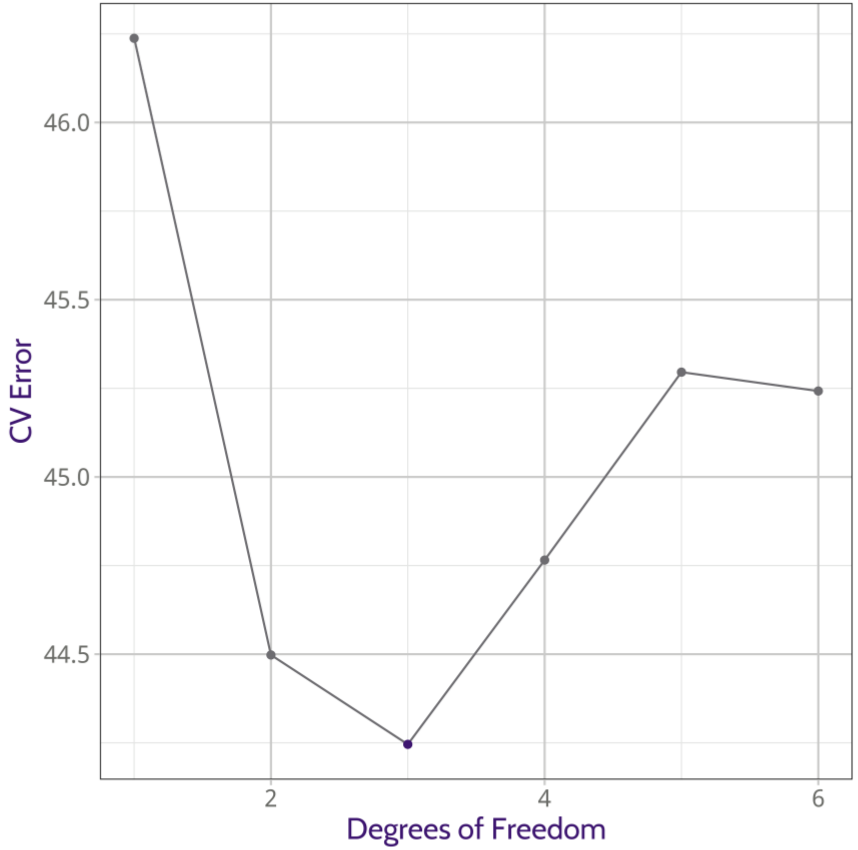


Notes

Type notes here...

Degrees of Freedom

```r
library(DAMisc)
nkp <- NKnots(chal_vote ~ chal_spend + exp_chal,
  "perotvote", max.knots=3, data=dat, includePol
  criterion="CV", plot=FALSE, cviter=10)
nkp$df <- 1:6
nkp$min <- factor(ifelse(nkp$stat == min(nkp$sta
                  levels=c(0,1),
                  labels=c("Other", "Minimum"))
```



Notes

Type notes here...

Polynomial

```
poly.mod <- lm(chal_vote ~ poly(perotvote, 3) +
                 chal_spend + exp_chal, data=dat)
S(poly.mod, brief=TRUE)
```

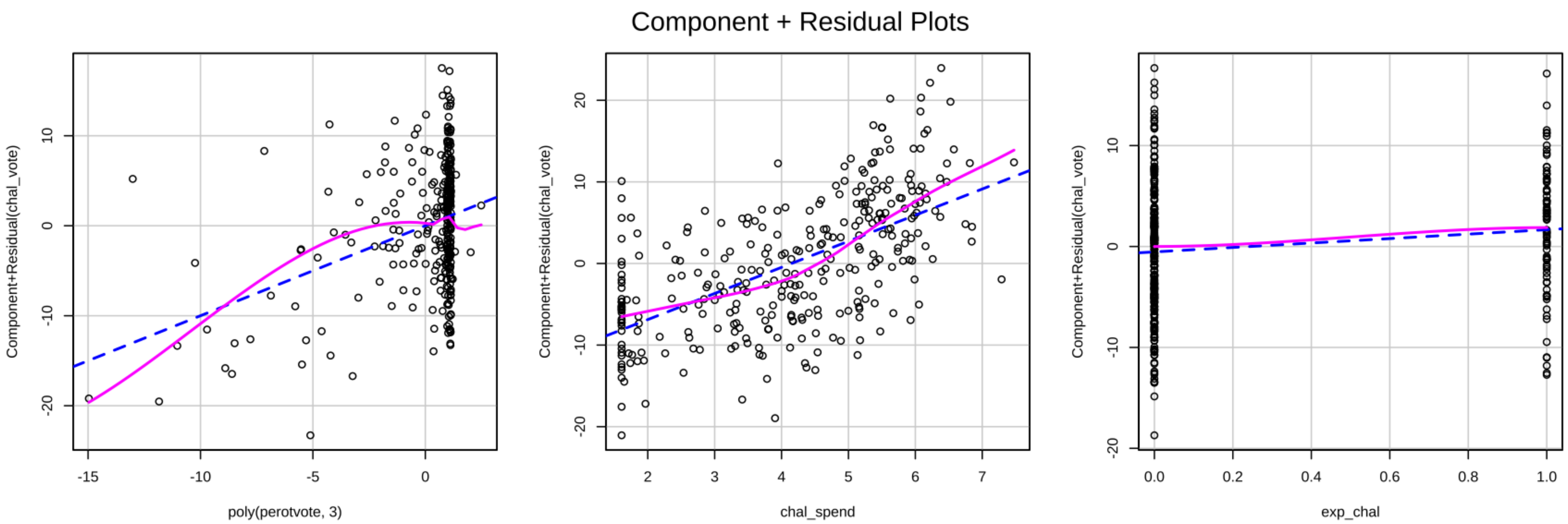
```
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)          22.6040     1.1222  20.143  < 2e-16 ***
## poly(perotvote, 3)1  33.2168     6.6421   5.001 9.63e-07 ***
## poly(perotvote, 3)2 -25.5074     6.5992  -3.865 0.000136 ***
## poly(perotvote, 3)3  11.8753     6.6112   1.796 0.073444 .
## chal_spend            3.2001     0.2738  11.688  < 2e-16 ***
## exp_chal              2.2016     0.9613   2.290 0.022683 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard deviation: 6.571 on 306 degrees of freedom
## Multiple R-squared: 0.4856
## F-statistic: 57.76 on 5 and 306 DF,  p-value: < 2.2e-16
##     AIC     BIC
## 2068.16 2094.36
```

Notes

Type notes here...

More CR Plots

```r
crPlots(poly.mod, layout=c(1,3))
```



Notes

Type notes here...

Which is better?

```r
library(clarkeTest)
clarke_test(trans.mod, poly.mod)
```

```
##
## Clarke test for non-nested models
##
## Model 1 log-likelihood: -1028
## Model 2 log-likelihood: -1027
## Observations: 312
## Test statistic: 207 (66%)
##
## Model 1 is preferred (p = 8e-09)
```

Notes

Type notes here...

Challenger Spending

```r
boxTidwell(chal_vote ~ chal_spend,
  ~ I(1/perotvote) + exp_chal,
  data=dat)
```

```
##  MLE of lambda Score Statistic (z)  Pr(>|z|)
##         2.2987              3.7127 0.0002051 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## iterations =  5
```

```r
boxTidwell(chal_vote ~ I(chal_spend^2),
  ~ I(1/perotvote) + exp_chal,
  data=dat)
```

```
##  MLE of lambda Score Statistic (z) Pr(>|z|)
##         1.1495              0.8603   0.3896
##
## iterations =  4
```

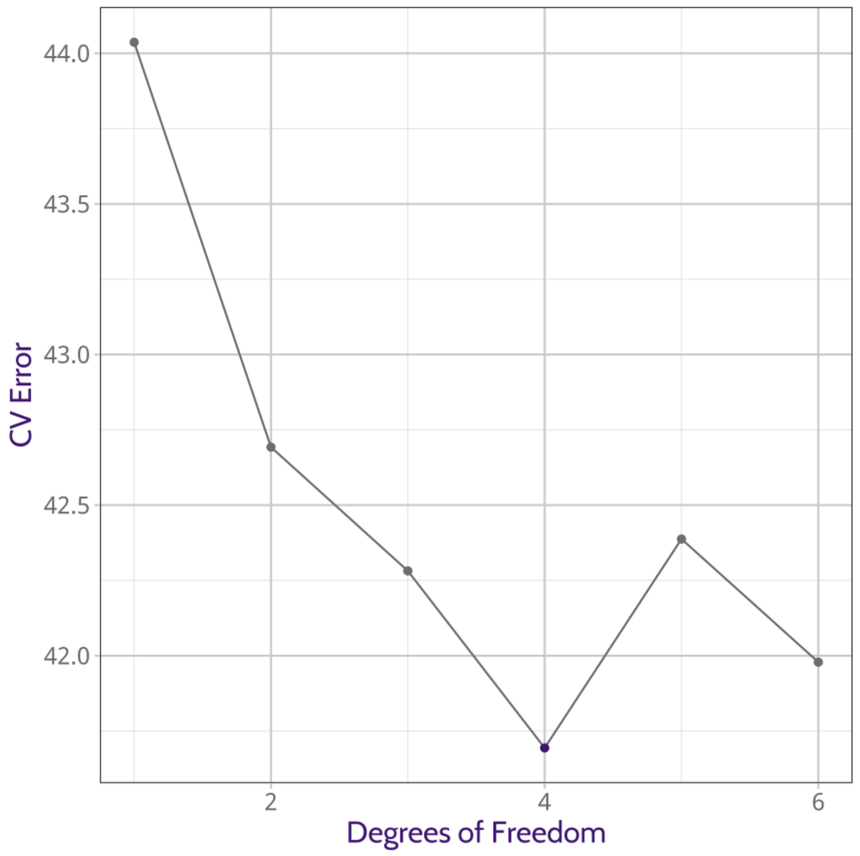
```r
trans.mod2 <- lm(chal_vote ~I(chal_spend^2) +
                I(1/perotvote) + exp_chal, data=dat)
```

Notes

Type notes here...

Degrees of Freedom

```
library(DAMisc)
nkp <- NKnots(chal_vote ~ I(1/perotvote) + exp_c
  "chal_spend", max.knots=3, data=dat, includePo
  criterion="CV", plot=FALSE, cviter=10)
nkp$df <- 1:6
nkp$min <- factor(ifelse(nkp$stat == min(nkp$sta
                levels=c(0,1),
                labels=c("Other", "Minimum"))
```



Notes

Type notes here...

Spline Model

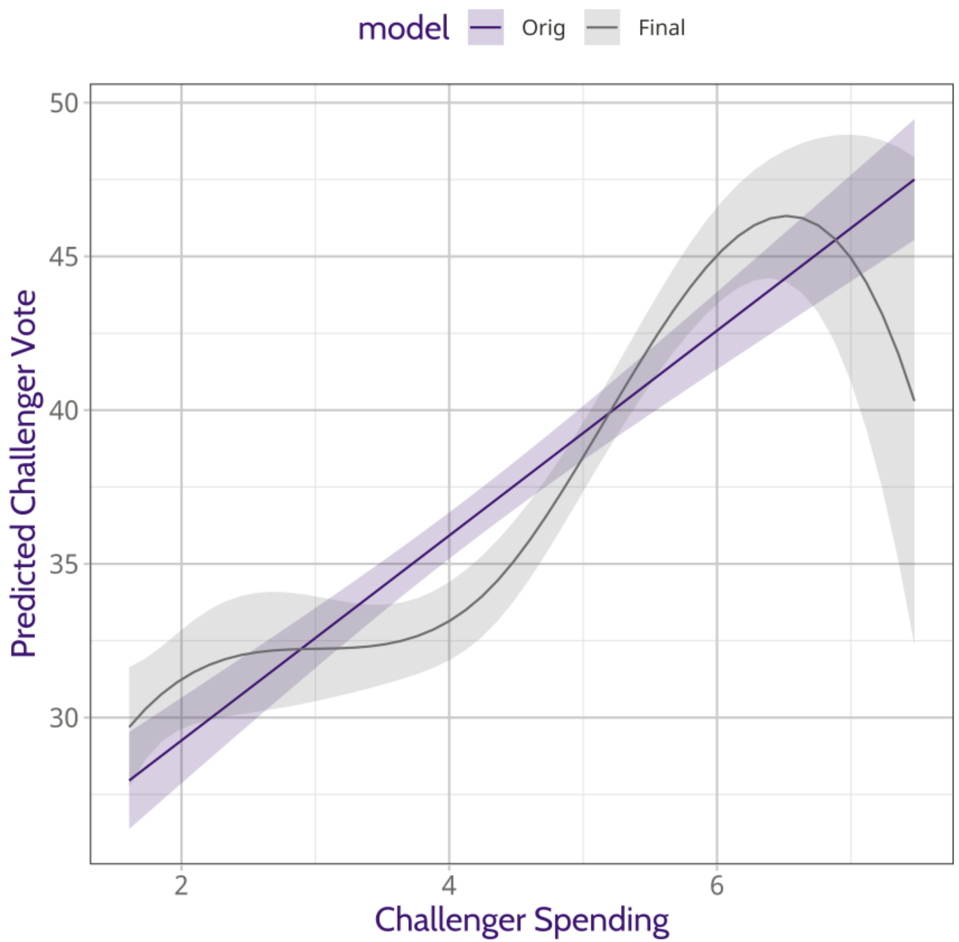
```
spline.mod <- dat %>%
  mutate(inv_perotvote = 1/perotvote) %>%
  lm(chal_vote ~ inv_perotvote +
                   bs(chal_spend, df=4) +
                   exp_chal, data=.)
anova(trans.mod2, spline.mod, test="F")
```

```
## Analysis of Variance Table
##
## Model 1: chal_vote ~ I(chal_spend^2) + I(1/perotvote) + exp_chal
## Model 2: chal_vote ~ inv_perotvote + bs(chal_spend, df = 4) + exp_chal
##   Res.Df   RSS Df Sum of Sq      F   Pr(>F)
## 1    308 12816
## 2    305 12208  3    608.53 5.0679 0.001939 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

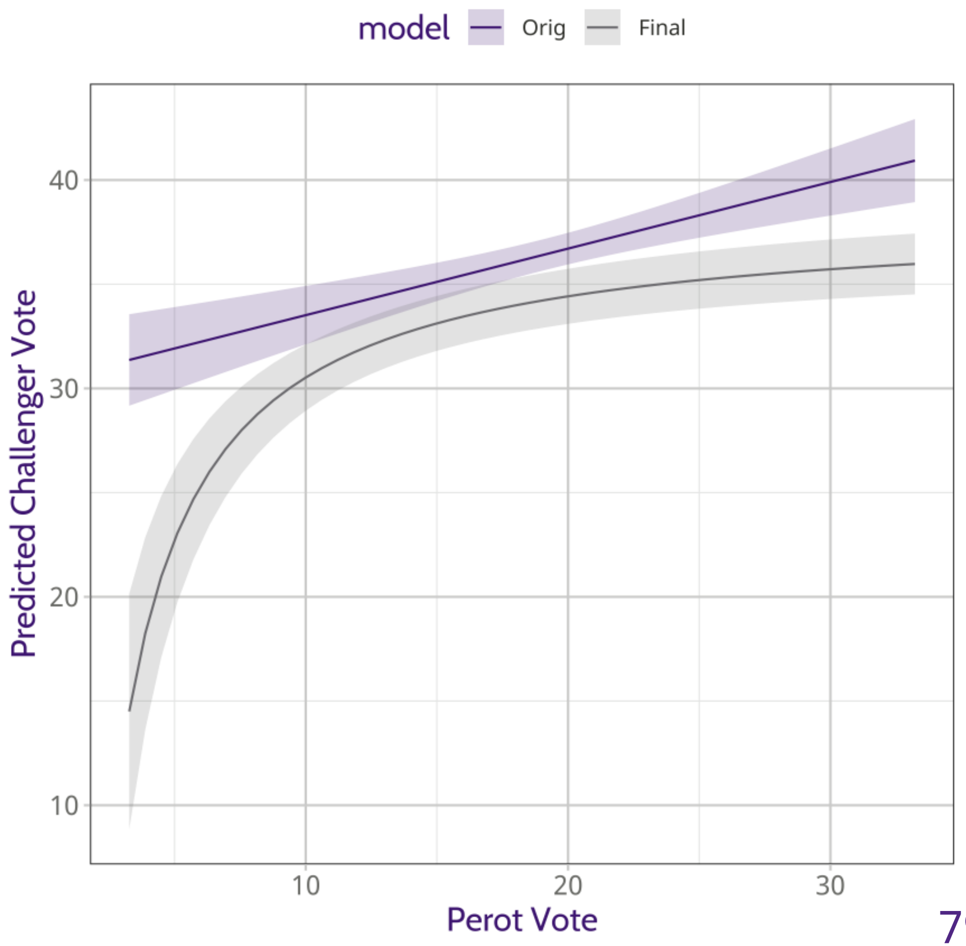
Notes

Type notes here...

Effects



model
Orig
Final
50
45
40
35
30
Predicted Challenger Vote
2
4
6
Challenger Spending



model
Orig
Final
40
30
20
10
Predicted Challenger Vote
10
20
30
Perot Vote

Notes

Type notes here...

Redux

Splines can be a good way to ...

- Fit models that cannot be easily fit by other simpler parametric forms still within the OLS/GLM framework.
- Test a wider array of possible alternative functional forms.

Notes

Type notes here...