POLSCI 9592

Lecture 8: Distributional Regression and Smoothing Splines

Dave Armstrong

Goals for This Session

1. Distributional Regression Models
2. Splines
3. Penalized Splines
4. GAMLSS

Distributional Regression Models

Model not just $E(y_i|\mathbf{x}_i)$ (i.e., the first *moment* of the distribution), but higher moments, too.

- GLMs make assumptions about the variance (e.g., in the Poisson $\mu_i = E(y_i|\mathbf{x}_i) = V(y_i|\mathbf{x}_i)$.
- We can relax these assumptions by parameterizing higher moments.

Heteroskedastic Linear Model

$$y_i = \mathbf{x}_i\mathbf{b} + e_i$$
$$\log(\mathrm{var}(e_i)) = \mathbf{x}_i\mathbf{g} + w_i$$

Here, instead of assuming $\mathrm{var}(e_i) = \sigma_e$ for all observations, we can parameterize the variance and have it change as a function of $\mathbf{x}$.

Example

```
library(gamlss)
library(psre)
data(wvs)
wvs2 <- wvs %>%
  select(resemaval, pct_high_rel_imp, pct_univ_degree) %>% na.omit()
g1 <- gamlss(resemaval ~ pct_high_rel_imp + pct_univ_degree, data=wvs2)
```

```
## GAMLSS-RS iteration 1: Global Deviance = -326.1249
## GAMLSS-RS iteration 2: Global Deviance = -326.1249
```

```
g2 <- gamlss(resemaval ~ pct_high_rel_imp + pct_univ_degree,
             sigma.formula =  ~ pct_high_rel_imp + pct_univ_degree,
             data=wvs2)
```

```
## GAMLSS-RS iteration 1: Global Deviance = -340.5837
## GAMLSS-RS iteration 2: Global Deviance = -343.1948
## GAMLSS-RS iteration 3: Global Deviance = -343.2899
## GAMLSS-RS iteration 4: Global Deviance = -343.2929
## GAMLSS-RS iteration 5: Global Deviance = -343.293
```

```
VC.test(g1, g2)
```

```
##  Vuong's test: -0.715 it is not possible to discriminate between models: g1 and g2
## Clarke's test: 59 p-value= 0.0014 g2 is preferred over g1
```

Summaries

```
# Constant Variance
brief(g1)
```

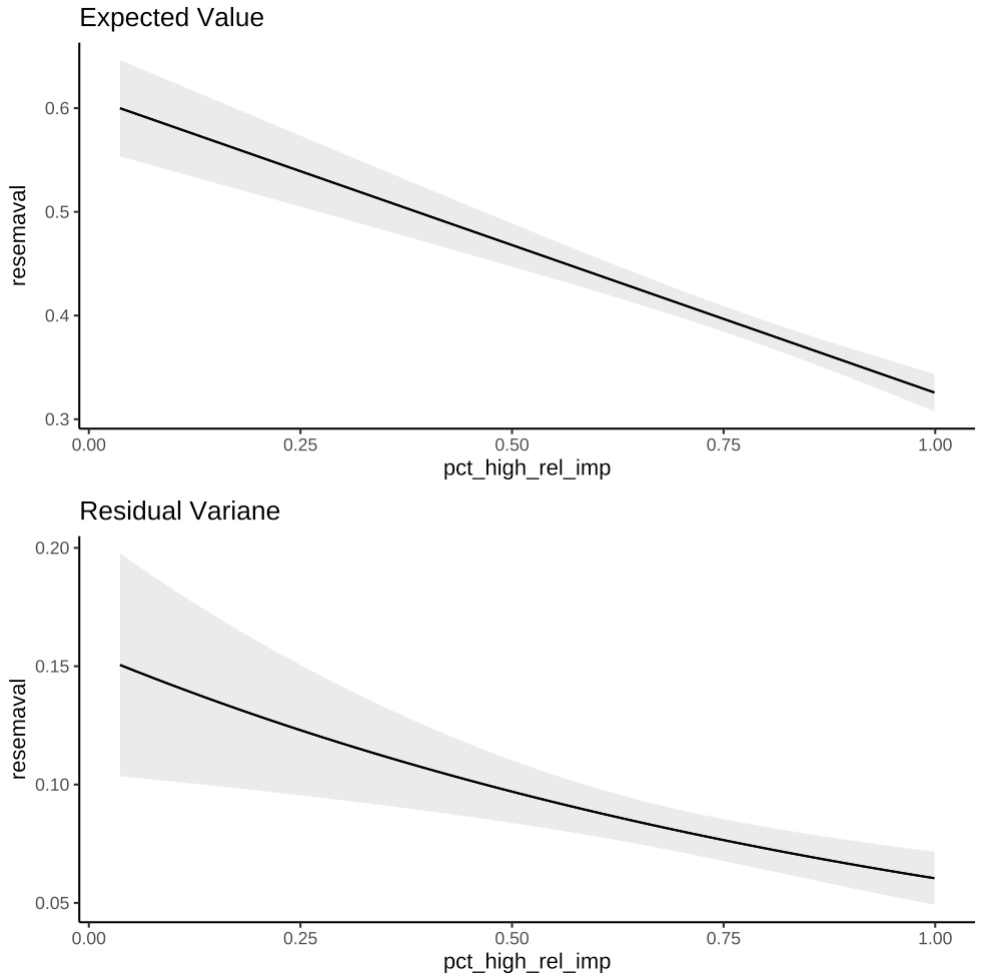
```
##                    Estimate Std. Error  t value  Pr(>|t|)
## (Intercept)        0.559886   0.022378  25.0194 < 2.2e-16 ***
## pct_high_rel_imp -0.240816   0.027268  -8.8313  2.09e-15 ***
## pct_univ_degree   0.083562   0.039645   2.1077   0.03666 *
## (Intercept)      -2.444488   0.056077 -43.5915 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Parameterized Variance
brief(g2)
```

```
##                    Estimate Std. Error t value  Pr(>|t|)
## (Intercept)        0.601693   0.028168 21.3606 < 2.2e-16 ***
## pct_high_rel_imp -0.284923   0.030756 -9.2639 < 2.2e-16 ***
## pct_univ_degree   0.039818   0.038172  1.0431    0.2985
## (Intercept)      -1.777345   0.199637 -8.9029 1.475e-15 ***
## pct_high_rel_imp -0.948363   0.233877 -4.0550 7.962e-05 ***
## pct_univ_degree  -0.371108   0.330964 -1.1213    0.2639
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Effects

```
library(patchwork)
p1 <- plot_predictions(g2,
                       condition="pct_high_rel_imp",
                       what="mu") +
  theme_classic() +
  ggtitle("Expected Value")
p2 <- plot_predictions(g2,
                       condition="pct_high_rel_imp",
                       what="sigma") +
  theme_classic() +
  ggtitle("Residual Variane")
p1 + p2 + plot_layout(ncol=1)
```



Splines

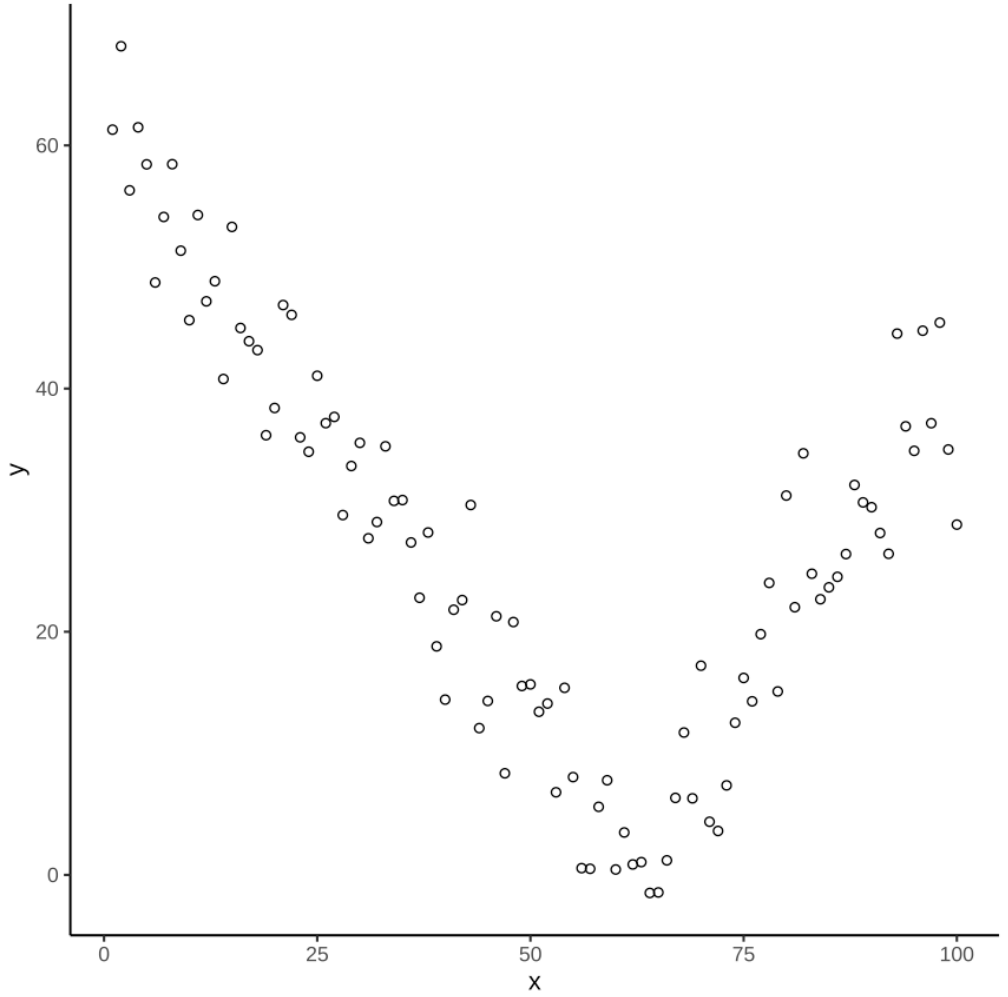
> "... piecewise regression functions we constrain to join at points called knots (Keele 2007, 70)"

- 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.

Simple Example

It is easy to figure out what sort of model we want. It appears that the relationship between x and y would be well-characterized by two lines.

- One with a negative slope in the range 0-60
- One with a positive slope in the range 60-100



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 $b_j(x_i) = x_i^j$

Truncated Power Basis Functions

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

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

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.

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.

Estimating the Model

```
mod <- lm(y ~ x + tpb(x, 1, 1, knot_loc=60))
summary(mod)
```

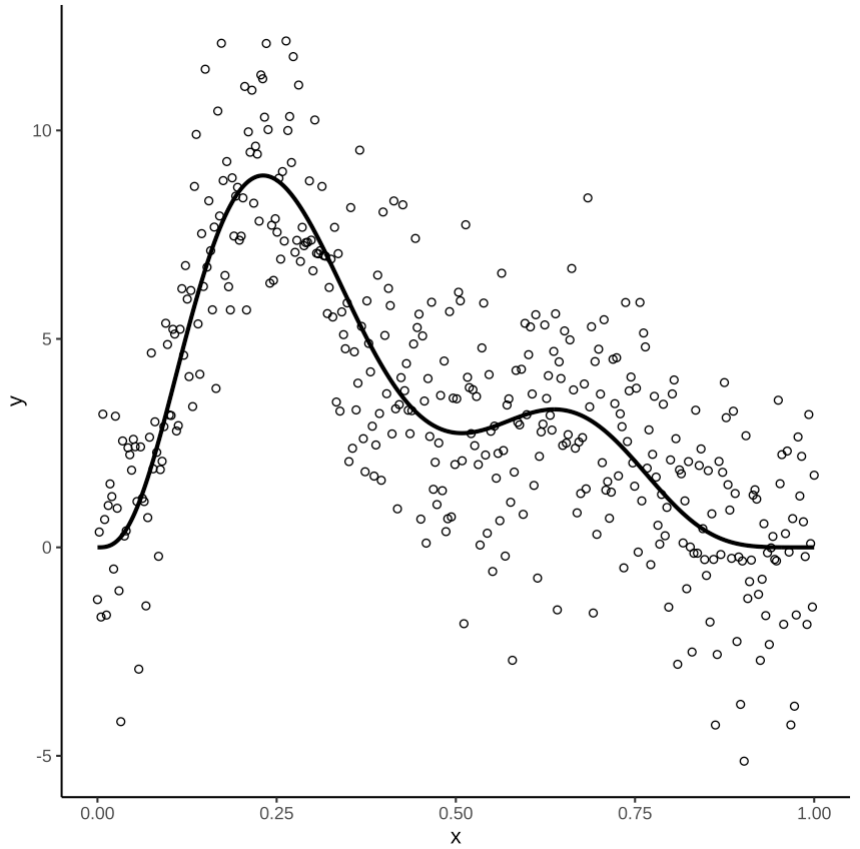
```
##
## Call:
## lm(formula = y ~ x + tpb(x, 1, 1, knot_loc = 60))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.6543  -3.2447  -0.4566   3.0872  11.8296
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 62.58109    1.23841   50.53   <2e-16 ***
## x                           -1.02121    0.03138  -32.55   <2e-16 ***
## tpb(x, 1, 1, knot_loc = 60)  2.00017    0.07295   27.42   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.982 on 97 degrees of freedom
## Multiple R-squared:  0.9161,    Adjusted R-squared:  0.9144
## F-statistic: 529.8 on 2 and 97 DF,  p-value: < 2.2e-16
```

```
car::linearHypothesis(mod,
                      "x + tpb(x, 1, 1, knot_loc = 60) = 0")
```

```
##
## Linear hypothesis test:
## x  + tpb(x,1, knot_loc = 60) = 0
##
## Model 1: restricted model
## Model 2: y ~ x + tpb(x, 1, 1, knot_loc = 60)
##
##   Res.Df     RSS Df Sum of Sq      F    Pr(>F)
## 1     98 11989.6
## 2     97  2407.9  1    9581.7 385.99 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Example: Cubic Spline

Consider the following relationship:



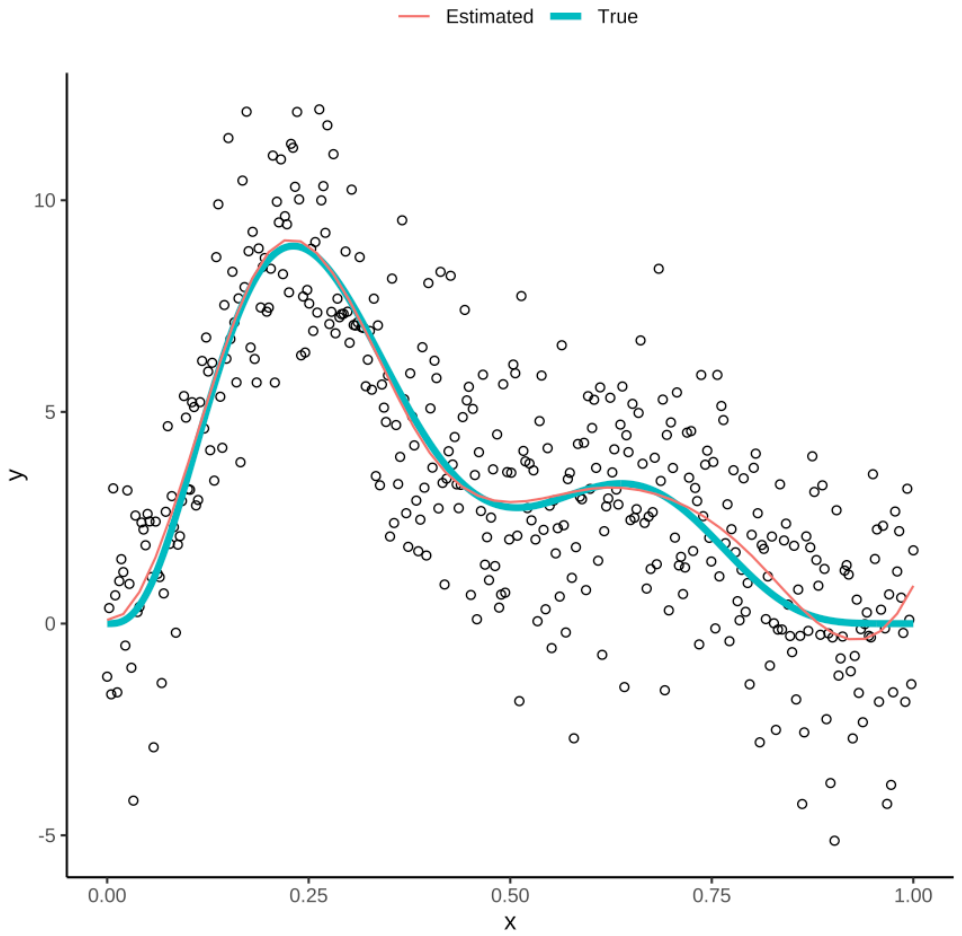
Cubic Spline

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

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 ***
## tpb(x, 3, 4, knot_loc = k)tpb1  2.771e+03  3.789e+02   7.314 1.48e-12 ***
## tpb(x, 3, 4, knot_loc = k)tpb2 -1.474e+03  1.821e+02  -8.094 7.36e-15 ***
## tpb(x, 3, 4, knot_loc = k)tpb3  3.866e+02  1.821e+02   2.123   0.0344 *
## tpb(x, 3, 4, knot_loc = k)tpb4  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
```

Predictions

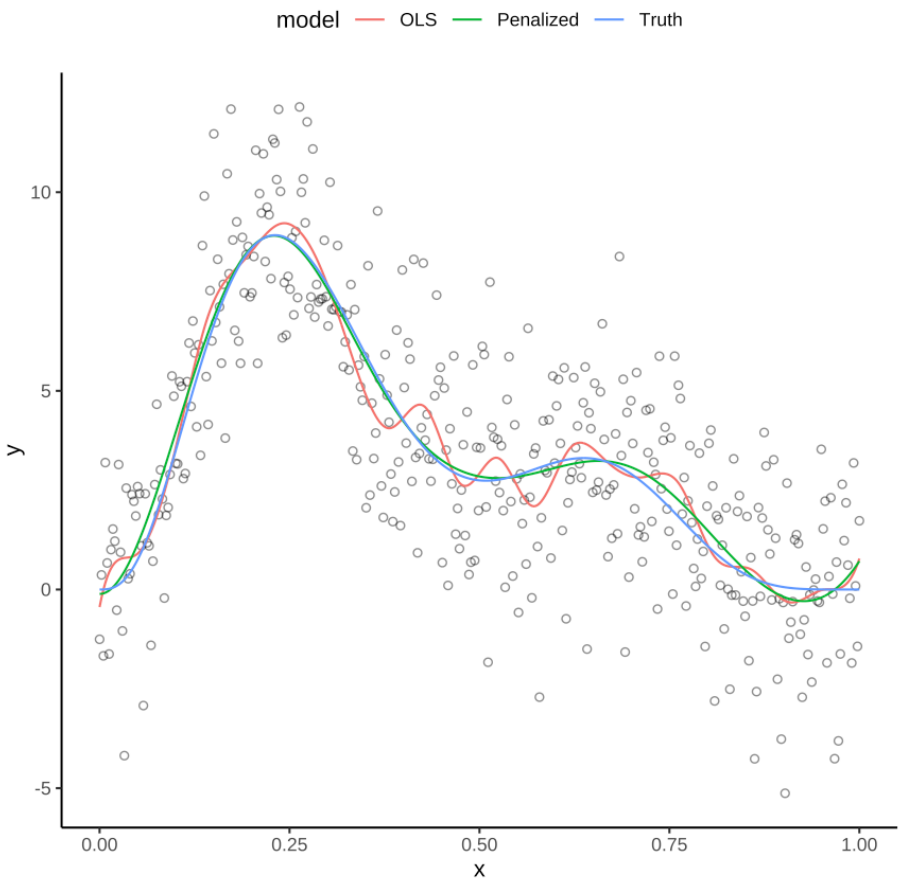


Example

```
df <- data.frame(x=x, y=y, f=f)
mod <- lm(y ~ poly(x, 3, raw=TRUE) + tpb(x, 3, 20), data=df)
D <- diag(24)
D[1:4,1:4] <- 0
X <- model.matrix(mod)
y <- model.response(model.frame(mod))

lambda <- .0025
b.constr <- solve(t(X) %*% X + lambda^2*D) %*% t(X) %*% y

fit0 <- X %*% mod$coef
fit1 <- X %*% b.constr
```



GAMLSS

We usually don't do the penalizing ourselves, we embed it in a regression model. GAMLSS is a framework for doing this (and many other things).

```
library(gamlss)
dframe <- data.frame(x=x, y=y, f=f)
mod <- gamlss(y ~ pb(x, control=pb.control(inter=50)), data=dframe)
```

```
## GAMLSS-RS iteration 1: Global Deviance = 1653.742
## GAMLSS-RS iteration 2: Global Deviance = 1653.742
```

```
brief(mod)
```

```
##                                      Estimate Std. Error t value  Pr(>|t|)
## (Intercept)                           6.278467   0.190859  32.896 < 2.2e-16
## pb(x, control = pb.control(inter = 50)) -5.630961   0.330371 -17.044 < 2.2e-16
## (Intercept)                           0.648239   0.035355  18.335 < 2.2e-16
##
## (Intercept)                          ***
## pb(x, control = pb.control(inter = 50)) ***
## (Intercept)                          ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Smooth Term

```
mod$mu.coefSmo[[1]]
```

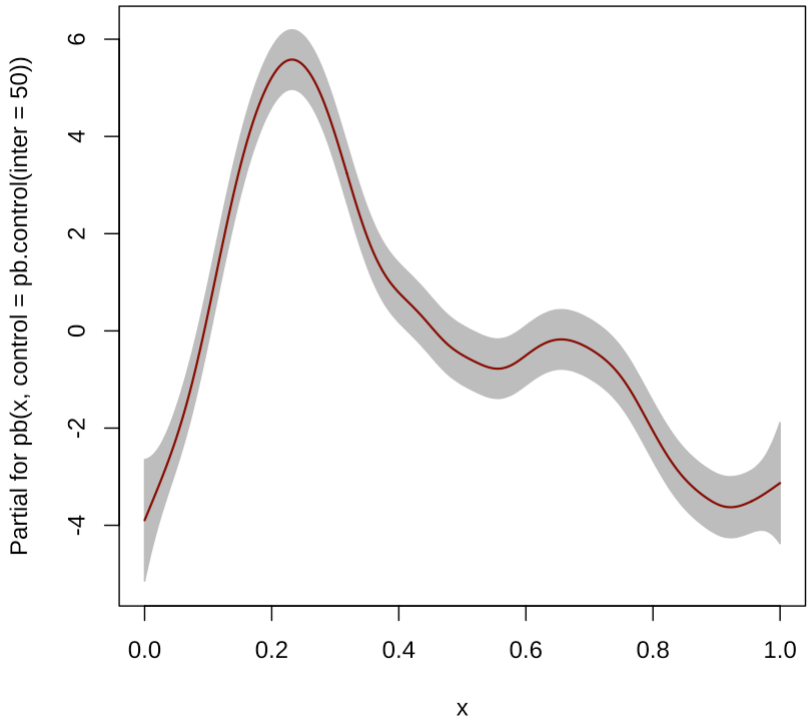
```
## P-spline fit using the gamlss function pb()
## Degrees of Freedom for the fit : 12.20499
## Random effect parameter sigma_b: 0.305637
## Smoothing parameter lambda     : 11.0419
```

```
# coefficients
c(mod$mu.coefSmo[[1]]$coef)
```

```
##  [1] -7.83987429 -7.08205107 -6.32457123 -5.55820839 -4.70740312 -3.68229997
##  [7] -2.43287518 -1.03781110  0.35715821  1.60436780  2.61615758  3.38244095
## [13]  3.91032869  4.13602021  4.05122654  3.69165811  3.07079075  2.29463666
## [19]  1.50317191  0.86189153  0.44471923  0.22884823  0.09157514 -0.07175075
## [25] -0.27758536 -0.44039906 -0.50226767 -0.52506893 -0.53208246 -0.45276313
## [31] -0.23082974  0.09126013  0.39642799  0.62672623  0.75195622  0.78664018
## [37]  0.75948287  0.68688824  0.53768191  0.26689779 -0.10429152 -0.48475629
## [43] -0.81851329 -1.05911040 -1.20412029 -1.29184188 -1.32162509 -1.22834632
## [49] -1.02434386 -0.76913754 -0.47207051 -0.16491794  0.14260257
```

Plot

```
term.plot(mod)
```



Democracy and Repression

We can use GAMLSS to estimate the relationship between democracy and repression - trying out different kinds of relationships.

Monotonic Democracy

We could use `gamlss()` to estimate a monotonic relationship.

```
library(foreign)
dat <- import("data/linear_ex.dta")
dat$polity_dem_fac <- as.factor(dat$polity_dem)
unrestricted.mod1 <- gamlss(rep1 ~ polity_dem_fac + iwar +
    cwar + logpop + gdppc,data=dat)
mono.mod1 <- gamlss(rep1 ~ pbm(polity_dem, mono="down") +
    iwar + cwar + logpop + gdppc,data=dat)
nonmono.mod1 <- gamlss(rep1 ~ pb(polity_dem) + iwar +
    cwar + logpop + gdppc,data=dat)
mod.2p <- gamlss(rep1 ~ polity_dem +
    I((polity_dem - 9)*(polity_dem >= 9)) + iwar +
    cwar + logpop + gdppc,data=dat)
```

```
VC.test(unrestricted.mod1, mono.mod1)
```

```
##  Vuong's test: -5.305 model mono.mod1 is preferred over unrestricted.mo
## Clarke's test: 679 p-value= 0 mono.mod1 is preferred over unrestricted
```

```
VC.test(mod.2p, nonmono.mod1)
```

```
##  Vuong's test: 8.167 model mod.2p is preferred over nonmono.mod1
## Clarke's test: 2192 p-value= 0 mod.2p is preferred over nonmono.mod1
```

```
VC.test(mono.mod1, nonmono.mod1)
```

```
##  Vuong's test: 5.324 model mono.mod1 is preferred over nonmono.mod1
## Clarke's test: 1998 p-value= 0 mono.mod1 is preferred over nonmono.mod
```

```
VC.test(mono.mod1, mod.2p)
```

```
##  Vuong's test: -7.701 model mod.2p is preferred over mono.mod1
## Clarke's test: 795 p-value= 0 mod.2p is preferred over mono.mod1
```

Monotone Party ID

```
anes <- import("data/anes1992.dta")
anes$pidfac <- as.factor(anes$pid)
unrestricted.mod2 <- gamlss(votedem ~ retnat + pidfac + age + male +
     educ + black + south, data=anes, family=BI)
```

```
## GAMLSS-RS iteration 1: Global Deviance = 769.7147
## GAMLSS-RS iteration 2: Global Deviance = 769.7147
```

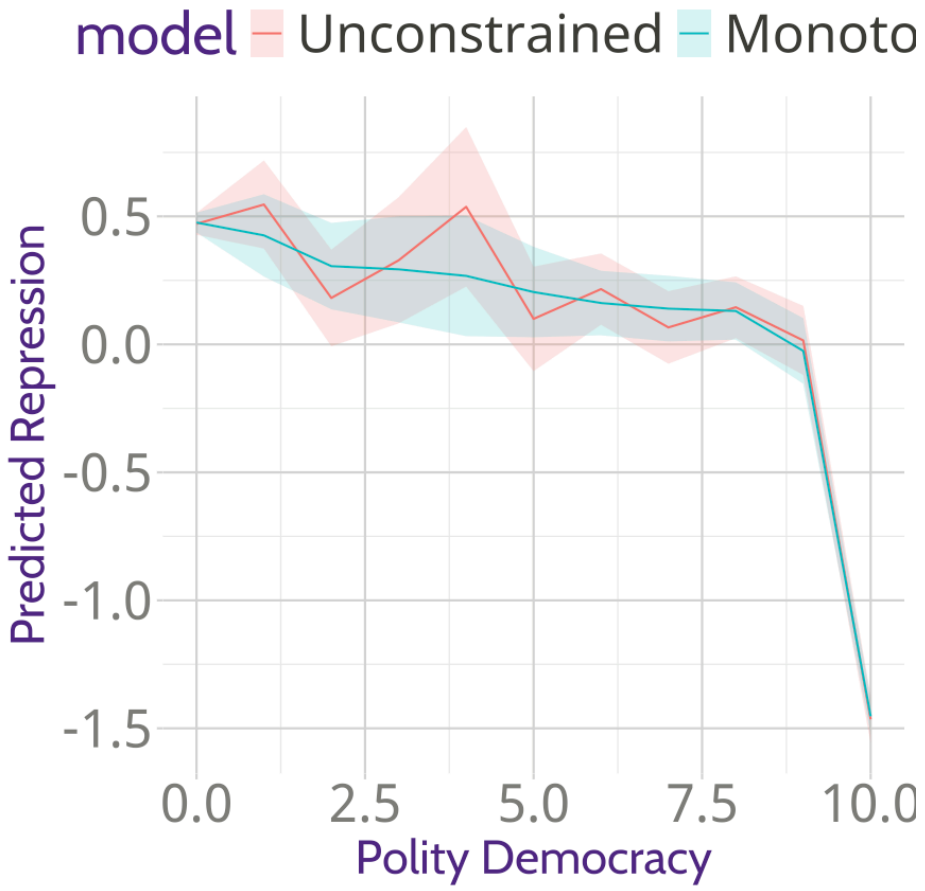
```
mono.mod2 <- gamlss(votedem ~ retnat + pbm(pid, mono="down") +
  age + male + educ + black + south, data=anes, family=BI)
```

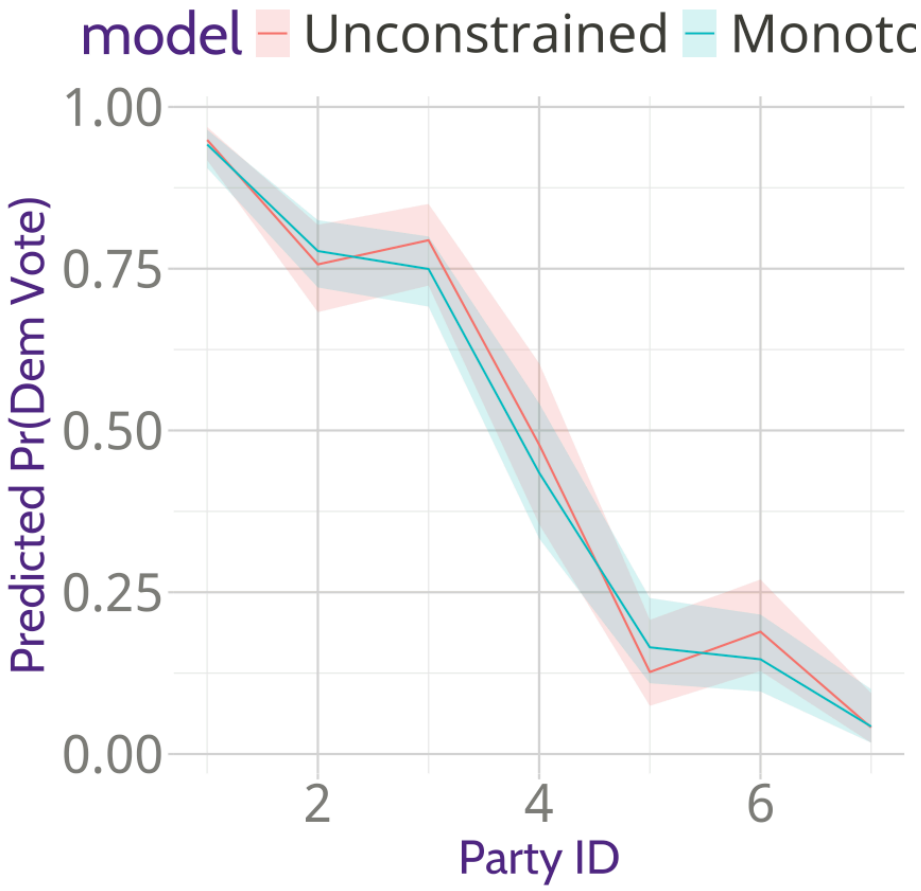
```
## GAMLSS-RS iteration 1: Global Deviance = 774.5032
## GAMLSS-RS iteration 2: Global Deviance = 774.5039
```

```
VC.test(unrestricted.mod2, mono.mod2)
```

```
##  Vuong's test: -3.648 model mono.mod2 is preferred over unrestricted.mod2
## Clarke's test: 328 p-value= 0 mono.mod2 is preferred over unrestricted.mod2
```

Plots


model
Unconstrained
Monoto
Predicted Repression
0.5
0.0
-0.5
-1.0
-1.5
0.0
2.5
5.0
7.5
10.0
Polity Democracy


model
Unconstrained
Monotc
Predicted Pr(Dem Vote)
1.00
0.75
0.50
0.25
0.00
2
4
6
Party ID

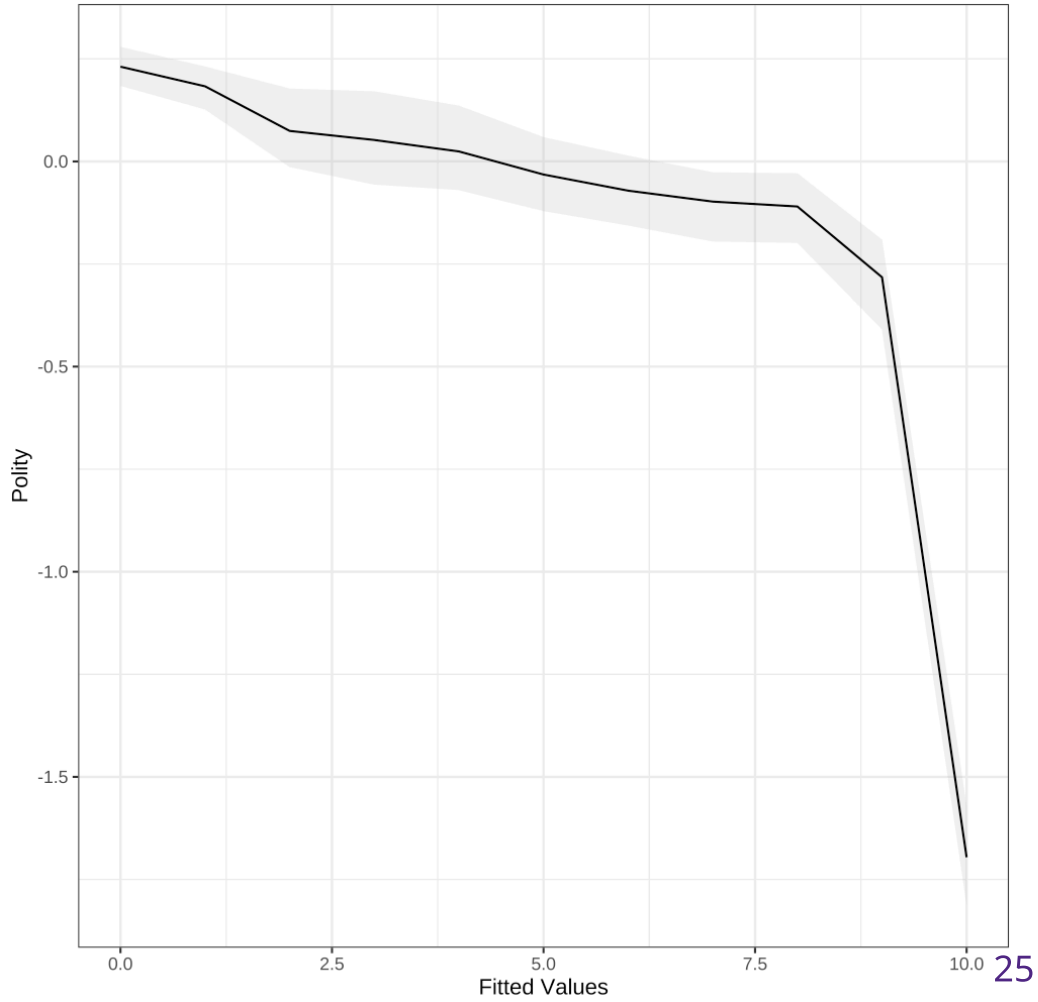
Effect Plot Data

```
library(gamlss.ggplots)
moddat <- get_all_vars(mono.mod1, dat)
cen_moddat <- lapply(moddat, \(x)DAMisc::central(x))
cen_moddat$polity_dem <- 0:10
cen_moddat <- do.call(data.frame, cen_moddat)

registerDoParallel(cores = 10)
B1 <- BayesianBoot(mono.mod1, B=100, newdata=cen_moddat)
stopImplicitCluster()
post_sum <- t(apply(B1$par$mu, 1, \(x)c(mean(x), sd(x), quantile(
colnames(post_sum) <- c("fit", "sd", "lwr", "upr")

cen_moddat <- cbind(cen_moddat, post_sum)
```

```
ggplot(cen_moddat,
       aes(x=polity_dem, y=fit,
           ymin=lwr, ymax=upr)) +
  geom_ribbon(fill="gray75", alpha=.25) +
  geom_line(color="black") +
  theme_bw() +
  labs(x="Fitted Values", y="Polity")
```



First Difference

```
moddat <- get_all_vars(mono.mod1, dat)
cen_moddat <- lapply(moddat, \(x)DAMisc::central(x))
cen_moddat$polity_dem <- c(0,10)

cen_moddat <- do.call(data.frame, cen_moddat)

registerDoParallel(cores = 10)
B1 <- BayesianBoot(mono.mod1, B=100, newdata=cen_moddat)
stopImplicitCluster()
post <- B1$par$mu

psum <- t(apply(post, 1, \(x)c(mean(x), quantile(x, c(.025, .975)))))
fd <- apply(post, 2, diff)
fd <- c(mean(fd), quantile(fd, c(.025, .975)))
out <- rbind(psum, fd) %>%
  as_tibble() %>%
  setNames(c("fit", "lwr", "upr")) %>%
  mutate(condition = c("Polity = 0", "Polity = 10", "Difference"), .before="fit")
out
```

```
## # A tibble: 3 × 4
##   condition      fit    lwr    upr
##   <chr>        <dbl>  <dbl>  <dbl>
## 1 Polity = 0   0.235  0.188  0.278
## 2 Polity = 10 -1.68  -1.80  -1.58
## 3 Difference  -1.92  -2.02  -1.81
```

Models Supported

Here are the models we've talked about that are supported by GAMLSS.

- Normal (family `NO`) - mean and variance
- Binomial (family `BI`) - mean
- Ordinal (family `ocat`) - mean (only using `gam()` from the `mgcv` package).
- Multinomial (family `MN3`, `MN4` or `MN5`) - mean and variance (3,4 or 5 categories only)
- Poisson (family `PO`) - mean
- Negative Binomial (family `NBI` or `NBII`) - mean and variance
- Exponential (family `EXP`) - mean
- Weibull (family `WEI`) - mean and variance
- Cox PH (family `cox.ph`) - mean (only using `gam()` from the `mgcv` package)

Problems With Linear Interactions

Consider the following model:

$$y = a + b_1 X + b_2 D + b_3 X \times D + e$$

Hainmueller, Mummolo and Xu argue that our linear interaction models like above suffer from two potential problems:

- The conditional partial effect of each variable is a linear function of the other: $b_{D|X} = b_2 + b_3 X$. Thus for every additional unit of X the effect of D changes by the same amount.
- Support - most political scientists do not pay attention to the joint density of the interacting variables and what the implies about our ability to make inferences.

Linearity

Again, using the same model

$$y = a + b_1 X + b_2 D + b_3 X \times D + e$$

Consider two different values of $D : \{d_1, d_2\}$, the effect of this contrast is:

$$\begin{aligned}\hat{Y}(D = d_1|X) - \hat{Y}(D = d_2|X) =& (a + b_1 X + b_2 d_1 + b_3 X d_1) \\ & - (a + b_1 X + b_2 d_2 + b_3 X d_2) \\ =& b_2(d_1 - d_2) + b_3 X(d_1 - d_2)\end{aligned}$$

The effect will only be appropriately estimated if *both* functions of X (for d_1 and d_2 are linear).

Support

$$b_2(d_1 - d_2) + b_3 X(d_1 - d_2)$$

The equation above makes it clear that to reliably estimate the treatment:

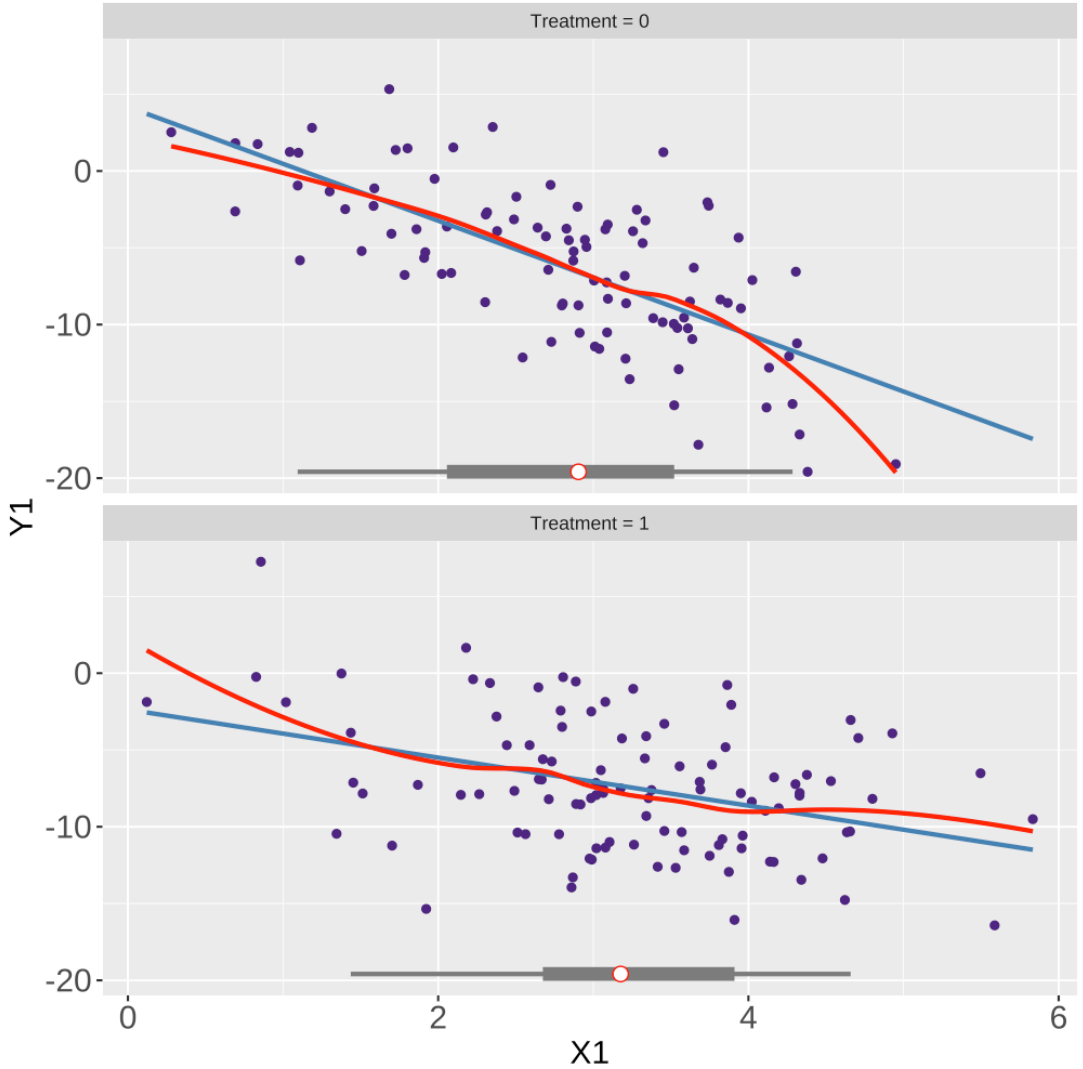
- We must have reasonable data density at both d_1 and d_2 for each value of X.
- Failure of this condition results in interaction effects that are a function of interpolation and extrapolation.

Diagnostics I

```
## Baseline group not specified; choose treat = 0 as the baseline group.
```

```
library(interflex)
set.seed(43901)
X1 <- rnorm(200, 3, 1)
X2 <- runif(200, -3, 3)
e <- rnorm(200, 0, 4)
D1 <- rbinom(200, 1, .5)
Y1 <- 5-4*X1 -9*D1 + 3*D1*X1 + e
Y2 <- 2.5- X2^2 -5*D1 + 2*D1*X2^2  + e
dat <- as.data.frame(cbind(Y1,Y2,D1,X1, X2))
dat$D10 <- 1-dat$D1
i1 <- interflex(estimator="raw",
                dat,
                "Y1",
                "D1",
                "X1",
                treat.type="discrete")

plot(i1)
```

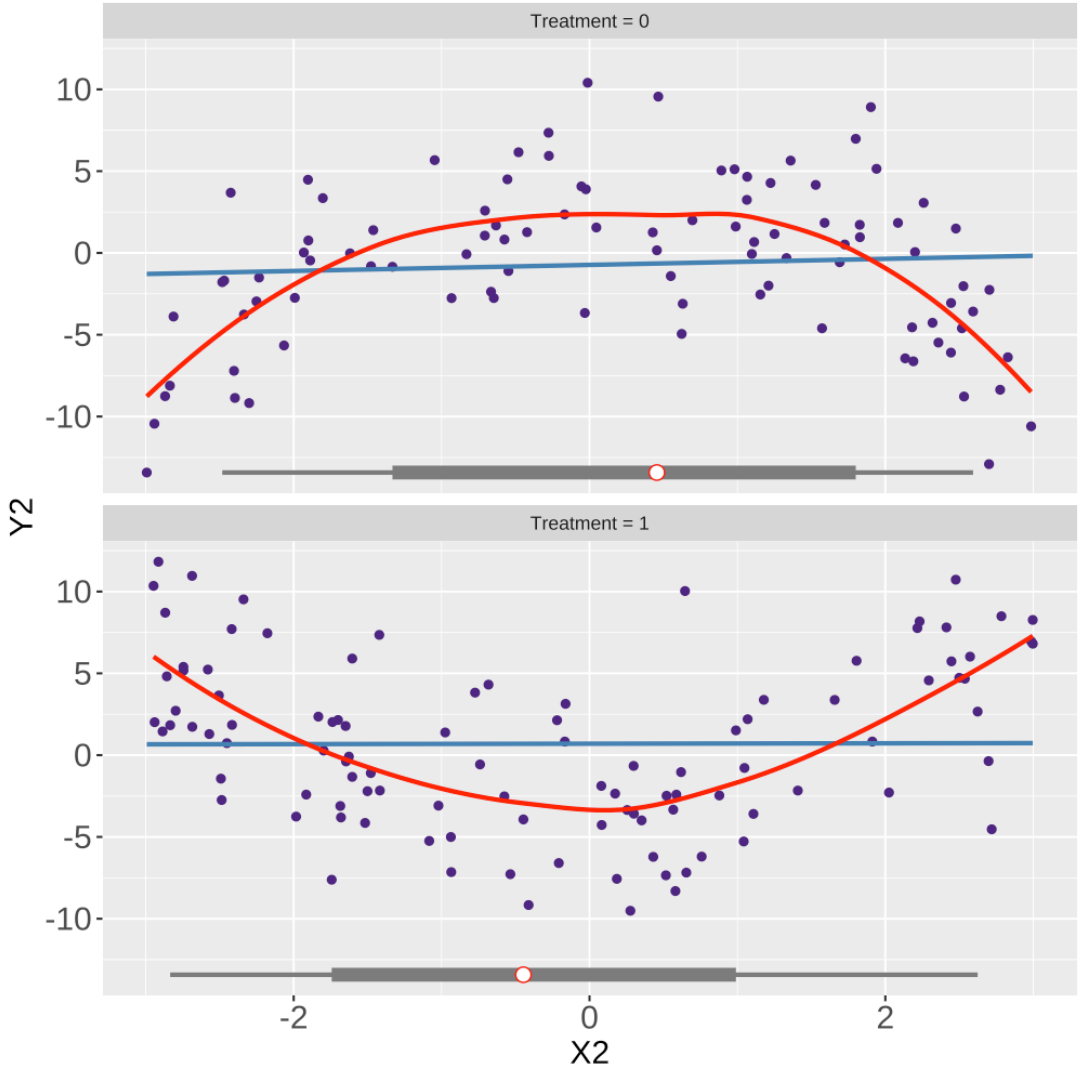


Diagnostics II

```
## Baseline group not specified; choose treat = 0 as the baseline group.
```

```
library(interflex)
i2 <- interflex(estimator="raw",
                dat,
                "Y2",
                "D1",
                "X2",
                treat.type="discrete")

plot(i2)
```



Binning Estimator for Interaction Effects

- Discretize X using its three terciles.

$$G_1 = \begin{cases} 1, & \text{if } X < \delta_{\frac{1}{3}} \\ 0, & \text{Otherwise} \end{cases} \quad G_2 = \begin{cases} 1, & \text{if } \delta_{\frac{1}{3}} \geq X < \delta_{\frac{2}{3}} \\ 0, & \text{Otherwise} \end{cases} \quad G_3 = \begin{cases} 1, & \text{if } \delta_{\frac{2}{3}} \geq X \\ 0, & \text{Otherwise} \end{cases}$$

- Pick an evaluation point, x_j, in each of the J bins (usually the median of the x values within the bin)

- Estimate the model:

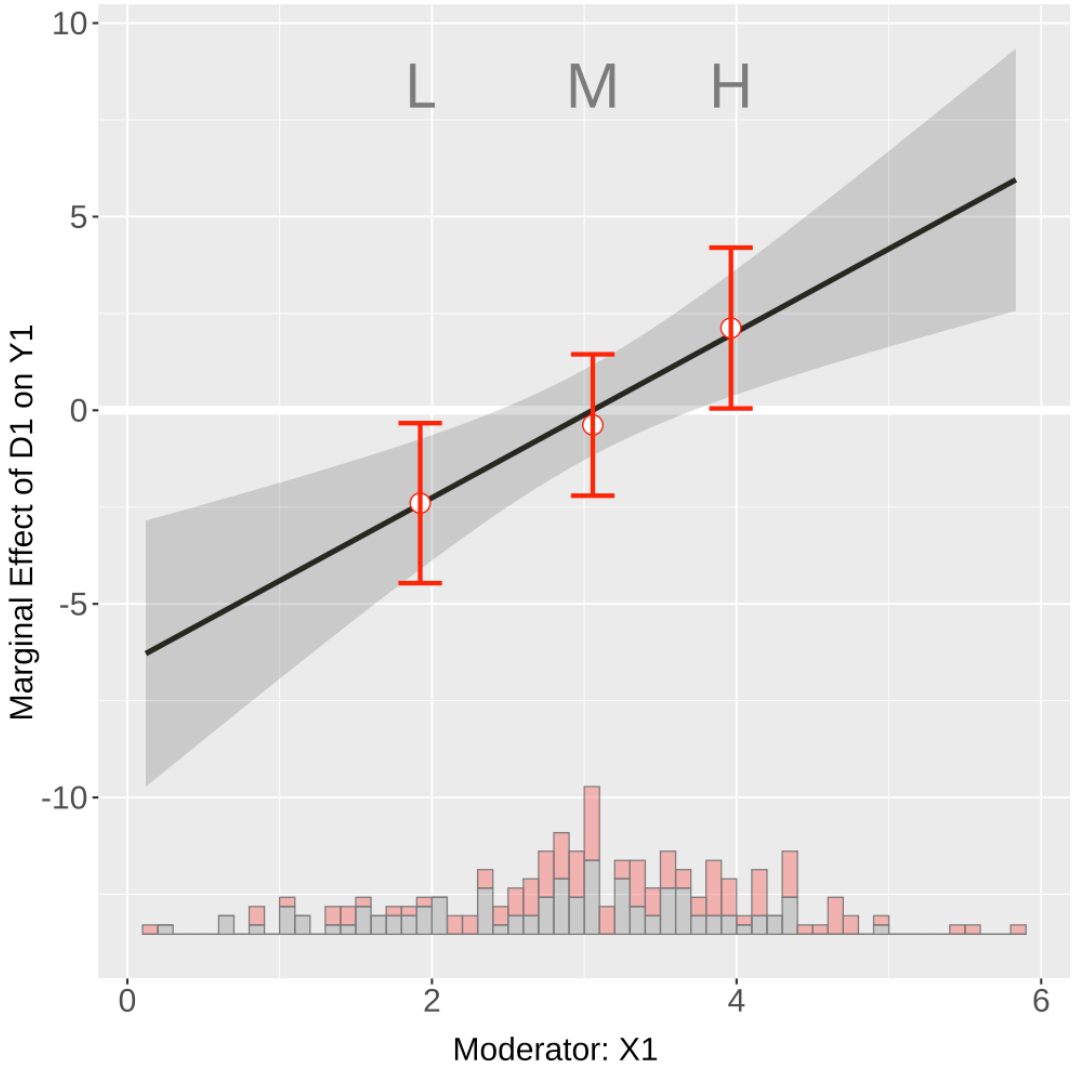
$$Y = \sum_{j=1}^{J} \left\{ u_j + \alpha_j D + \eta_j (X - x_j) + \beta_j (X - x_j) D \right\} G_j + \ldots + \varepsilon$$

Advantages over Linear Interaction Effect

- More flexible for non-linear functional forms. It fits separate interactions to each bin.
- Since $(X - x_j) = 0$ at the evaluation point, the partial conditional effect within each bin is just α_j.
- Binning ensures that there is sufficient joint support on X and D since they are constructed from X.
- This model nests the linear interaction model, so it can serve as a test of the linear interaction effect model.

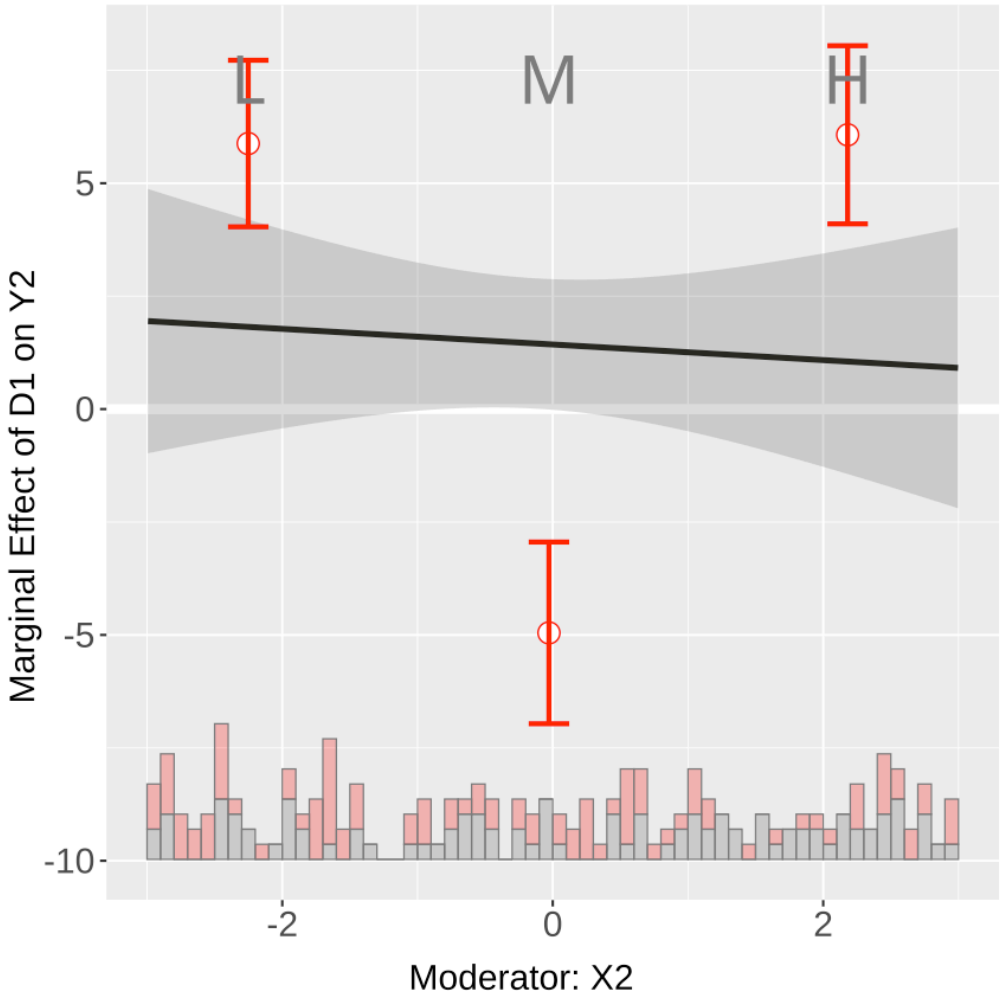
Binning Estimator

```
b1 <- interflex(estimator="binning",
                dat,
                "Y1",
                "D1",
                "X1",
                treat.type="discrete")
plot(b1)
```



Binning Estimator II

```
b2 <- interflex(estimator="binning",
                dat,
                "Y2",
                "D1",
                "X2",
                treat.type="discrete")
plot(b2)
```



Testing for Linear Interaction Effect

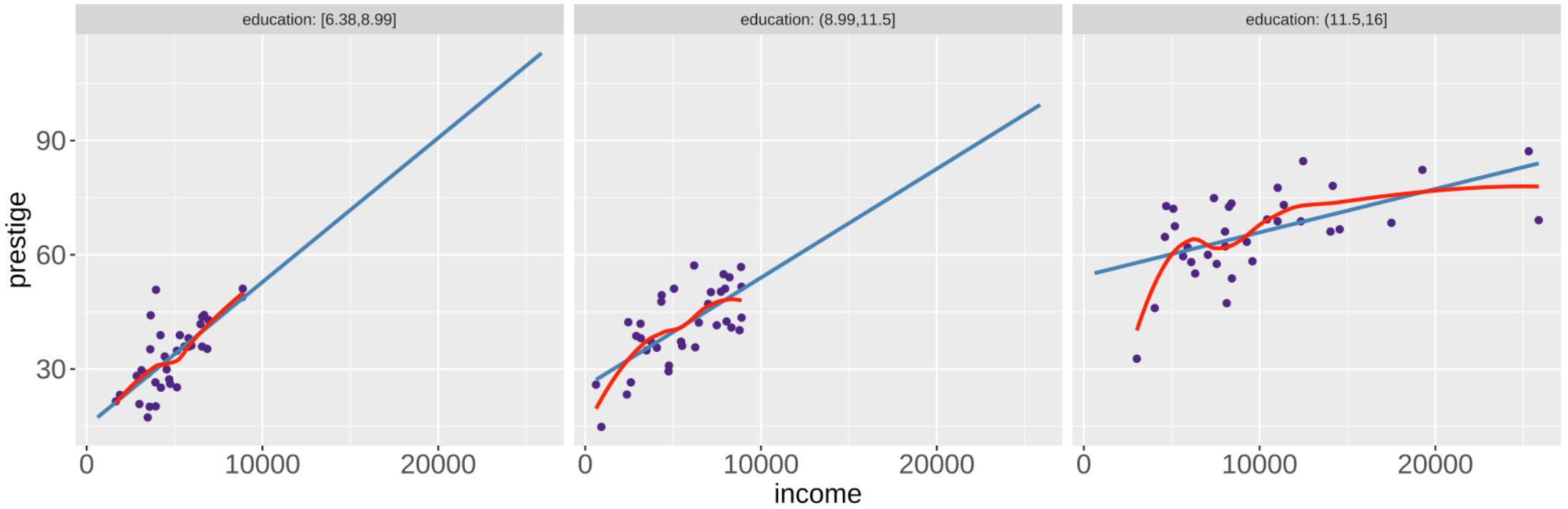
We could recast the model above (the binning estimator one) to be:

$$\begin{aligned} Y =& \mu + \alpha D + \eta X + \beta DX + G_2(\mu_{2'} + \alpha_{2'} D + \eta_{2'} X + \beta_{2'} DX) \\ &+ G_3(\mu_{3'} + \alpha_{3'} D + \eta_{3'} X + \beta_{3'} DX) \end{aligned}$$

We can then use a Wald test (i.e., F-test) to test whether all of the $\theta_{2'}$ and $\theta_{3'}$ terms are simultaneously zero

On Prestige Data

```
data(Prestige, package="carData")
interflex("raw", Prestige, "prestige", "income", "education", ncols=3)
```

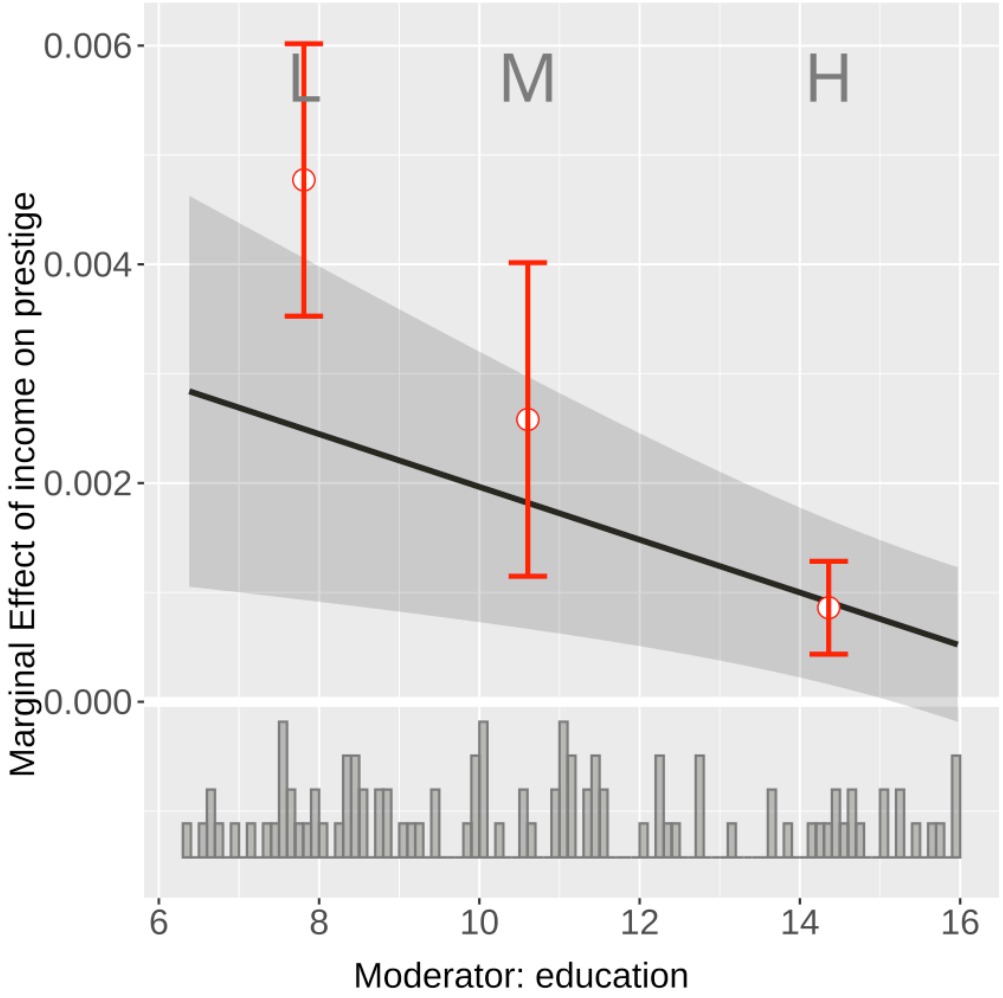


Binning Estimator: Prestige Data

```
pres.b <- interflex("binning",
                    Prestige,
                    "prestige",
                    "income",
                    "education",
                    Z = c("type",
                          "women"),
                    na.rm=T,
                    figure=FALSE)
```

```
pres.b$tests
```

```
## $treat.type
## [1] "continuous"
##
## $X.Lkurtosis
## [1] "0.023"
##
## $p.wald
## [1] "0.000"
##
## $p.lr
## [1] "0.024"
##
## $formula.restrict
## prestige ~ education + income + DX + women + Dummy.Covariate.1 +
##     Dummy.Covariate.2
## <environment: 0x115498368>
```



GAMs for the Fake Data I

```
library(gamlss.add)
mod1 <- gamlss(Y1 ~ D1 +
                 pb(X1) +
                 pb(I(X1*(D1 == 1))),
               data=dat)
```

```
## GAMLSS-RS iteration 1: Global Deviance = 1105.683
## GAMLSS-RS iteration 2: Global Deviance = 1105.683
```

```
mod2 <- gamlss(Y1 ~ X1*D1, data=dat)
```

```
## GAMLSS-RS iteration 1: Global Deviance = 1105.683
## GAMLSS-RS iteration 2: Global Deviance = 1105.683
```

```
VC.test(mod1, mod2)
```

```
##  Vuong's test: -5.8 model mod2 is preferred over mod1
## Clarke's test: 29 p-value= 0 mod2 is preferred over mod1
```

GAMs for the Fake Data II

```
library(gamlss.add)
mod1 <- gamlss(Y2 ~ ga(~ D1 +
                  s(X2, by=D1, bs="ts")+
                  s(X2, by=D10, bs="ts")),
                data=dat)
```

```
## GAMLSS-RS iteration 1: Global Deviance = 1094.086
## GAMLSS-RS iteration 2: Global Deviance = 1094.086
```

```
mod2 <- gamlss(Y2 ~ X2*D1, data=dat)
```

```
## GAMLSS-RS iteration 1: Global Deviance = 1214.533
## GAMLSS-RS iteration 2: Global Deviance = 1214.533
```

```
VC.test(mod1, mod2)
```

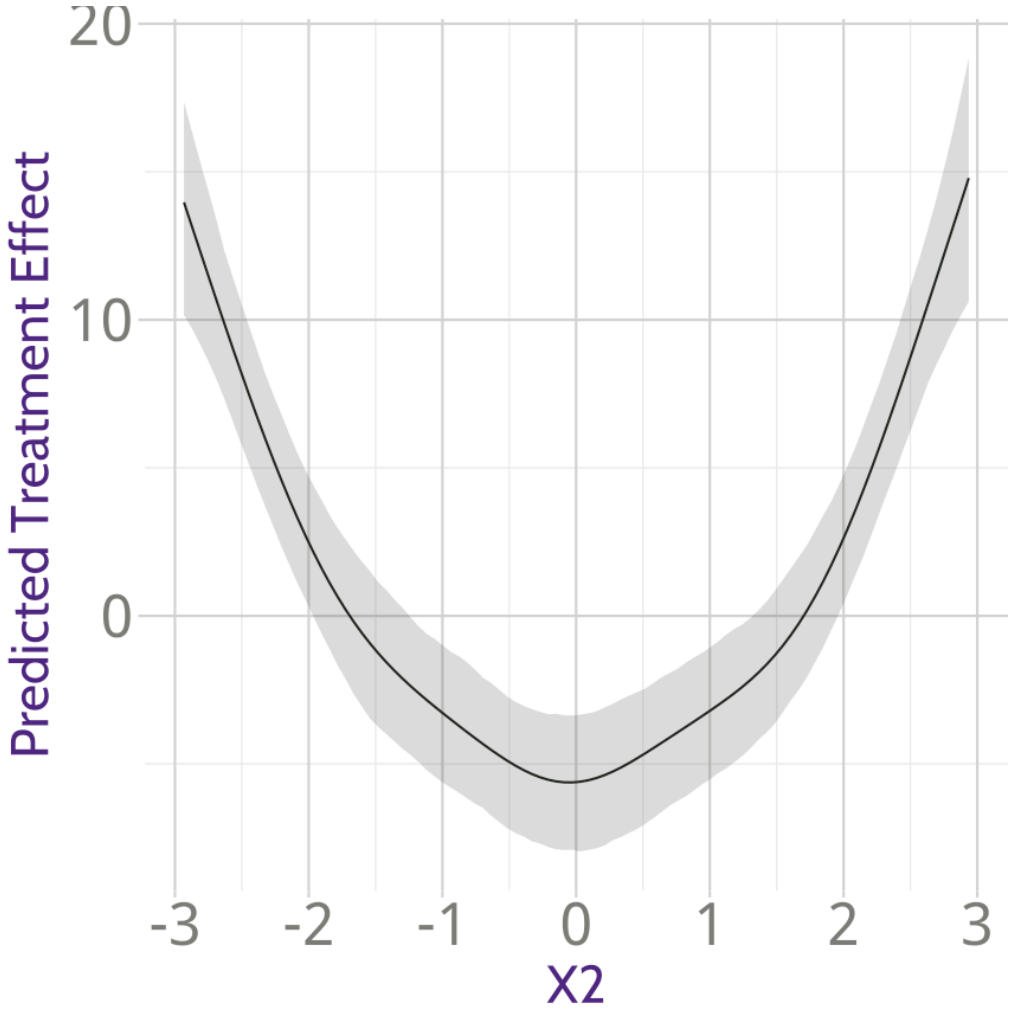
```
##  Vuong's test: 4.793 model mod1 is preferred over mod2
## Clarke's test: 155 p-value= 0 mod1 is preferred over mod2
```

Treatment Effect Code

```
fake.dat <- expand.grid(
  X2 = seq(min(dat$X2), max(dat$X2), length=100),
  D1 = c(0,1)
)
fake.dat$D10 <- 1-fake.dat$D1
lbx2 <- max(min(dat$X2[which(dat$D1 == 0)]),
            min(dat$X2[which(dat$D1 == 1)]))
ubx2 <- min(max(dat$X2[which(dat$D1 == 0)]),
            max(dat$X2[which(dat$D1 == 1)]))
fake.dat <- fake.dat %>%
  filter(X2 > lbx2 & X2 < ubx2)
X <- model.matrix(mod1$mu.coefSmo[[1]], newdata=fake.dat)
fit <- X %*% mod1$mu.coefSmo[[1]]$coefficients
b <- MASS::mvrnorm(1500,
                   coef(mod1$mu.coefSmo[[1]]),
                   vcov(mod1$mu.coefSmo[[1]]))
Xb <- X %*% t(b)
diff <- Xb[99:196, ] - Xb[1:98, ]
mean.diff <- rowMeans(diff)
diff.ci <- apply(diff, 1, quantile, c(.025,.975))
fake.dat$diff <- fit[99:196]-fit[1:98]
fake.dat$lower <- diff.ci[1,]
fake.dat$upper <- diff.ci[2,]
fake.dat <- fake.dat[which(fake.dat$D1 == 0), ]
```

Treatment Effect Plot

```
ggplot(fake.dat, aes(x=X2, y=diff,
                     ymin=lower, ymax=upper)) +
  geom_ribbon(alpha=.2, col="transparent") +
  geom_line() +
  theme_xaringan() +
  labs(x="X2", y="Predicted Treatment Effect")
```



GAMs for the Prestige Data

```
library(mgcv)
Prestige <- na.omit(Prestige)
mod1 <- gamlss(prestige ~ log(income)*education +
    women + type, data=Prestige)
```

```
## GAMLSS-RS iteration 1: Global Deviance = 632.6185
## GAMLSS-RS iteration 2: Global Deviance = 632.6185
```

```
mod2 <- gamlss(prestige ~
    ga(~ ti(income) + ti(education) + ti(income, education)) +
    women + type, data=Prestige)
```

```
## GAMLSS-RS iteration 1: Global Deviance = 627.3067
## GAMLSS-RS iteration 2: Global Deviance = 627.3019
## GAMLSS-RS iteration 3: Global Deviance = 627.3011
```

```
VC.test(mod1, mod2)
```

```
##  Vuong's test: 2.75 model mod1 is preferred over mod2
## Clarke's test: 72 p-value= 0 mod1 is preferred over mod2
```

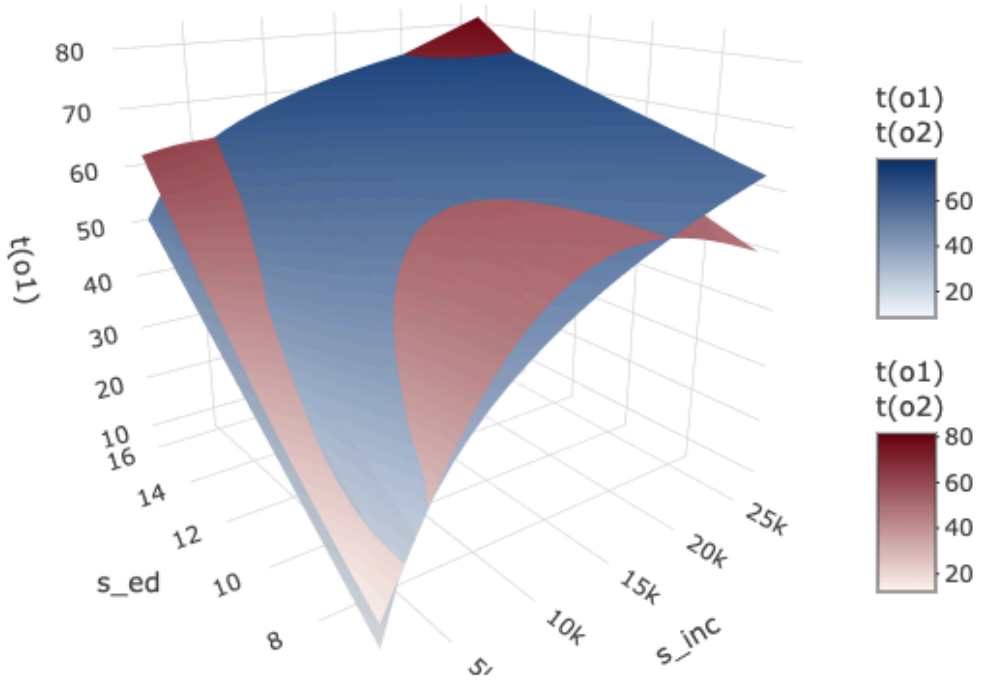
Two 3-D Surfaces for the Two Models

```
s_inc <- with(Prestige, seq(min(income), max(income), length=25))
s_ed <- with(Prestige, seq(min(education), max(education), length
s_linc <- log(s_inc)

p1 <- Vectorize(function(x, y, ...){
  d <- data.frame(type = factor(1, levels=1:3, labels=c("bc", "pr
                  women = 29,
                  income = x,
                  education = y)
  predict(mod1, newdata=d)
})
p2 <- Vectorize(function(x, y, ...){
  d <- data.frame(type = factor(1, levels=1:3,
                    labels=c("bc", "prof", "wc")),
                  women = 29,
                  income = x,
                  education = y)
  predict(mod2, newdata=d)
})

o1 <- outer(s_inc, s_ed, p1)
o2 <- outer(s_inc, s_ed, p2)

plot_ly() %>%
  add_trace(x=~s_inc, y=~s_ed, z=~t(o1), type="surface",
    colorscale=list(c(0,1),
      RColorBrewer::brewer.pal(9, "Blues")[c(1,9)])) %>%
  add_trace(x=~s_inc, y=~s_ed, z=~t(o2), type="surface",
    colorscale=list(c(0,1),
      RColorBrewer::brewer.pal(9, "Reds")[c(1,9)]))
```



Review

1. Distributional Regression Models
2. Splines
3. Penalized Splines
4. GAMLSS

Exercise

Using the Fearon and Laitin data, estimate the baseline model below and then estimate the model in the GAMLSS framework with penalized splines on the continuous variables. How do the relationships change relative to the parametric model?

```
fldat <- rio::import("fl_repdata.dta")
fldat$onset <- ifelse(fldat$onset > 1, 1, fldat$onset)
bmod <- glm(onset ~ warl + gdpenl + lpopl1 +
      lmtnest + ncontig + Oil + nwstate + instab +
      polity2l + ethfrac + relfrac,
    data=fldat, family=binomial)
```