Regression III

Penalized Splines and GAMLSS

Dave Armstrong

Goals for Today

1. Discuss penalized splines.
2. Use GAMLSS to estimate semi-parametric models
3. Consider recent work on linear vs non-linear interactions
    - Use semi-parametric models to test for non-linearity in interactions.

Penalized Splines

Sometimes spline smooths can be a bit too variable (particularly with many knots).

- Regularization can help solve this problem, while still retaining the systematically important part of the fit.
- Rather than choosing a small number of knots, choose many and regularize.

If we're using a truncated power basis function of order p with k knots, where the first $p+1$ elements of the coefficient vector are the intercept and global polynomials, then we can define:

$$\mathbf{D} = \begin{bmatrix} \mathbf{0}_{p+1\times p+1} & \mathbf{0}_{p+1\times k} \\ \mathbf{0}_{k\times p+1} & \mathbf{I}_{k\times k} \end{bmatrix}$$

Notes

Type notes here...

Penalized Splines (2)

With $\mathbf{D}$ defined, we could then minimize:

$$|| \mathbf{y} - \mathbf{X}\beta ||^2 + \lambda^2 \beta' \mathbf{D} \beta$$

The **roughness penalty** really captures the sum of squared penalized coefficients - the ℓ_2 norm.

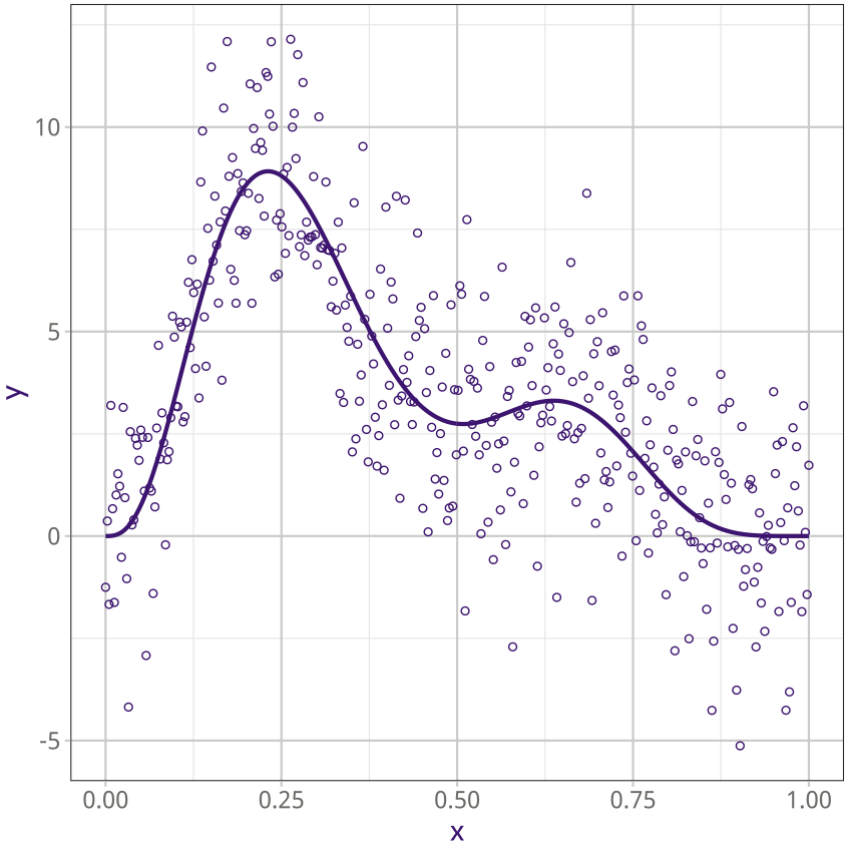
The penalized coefficients are, then:

$$\beta = (\mathbf{X}'\mathbf{X} + \lambda^2 \mathbf{D})^{-1} \mathbf{X}'\mathbf{y}$$

Notes

Type notes here...

Example

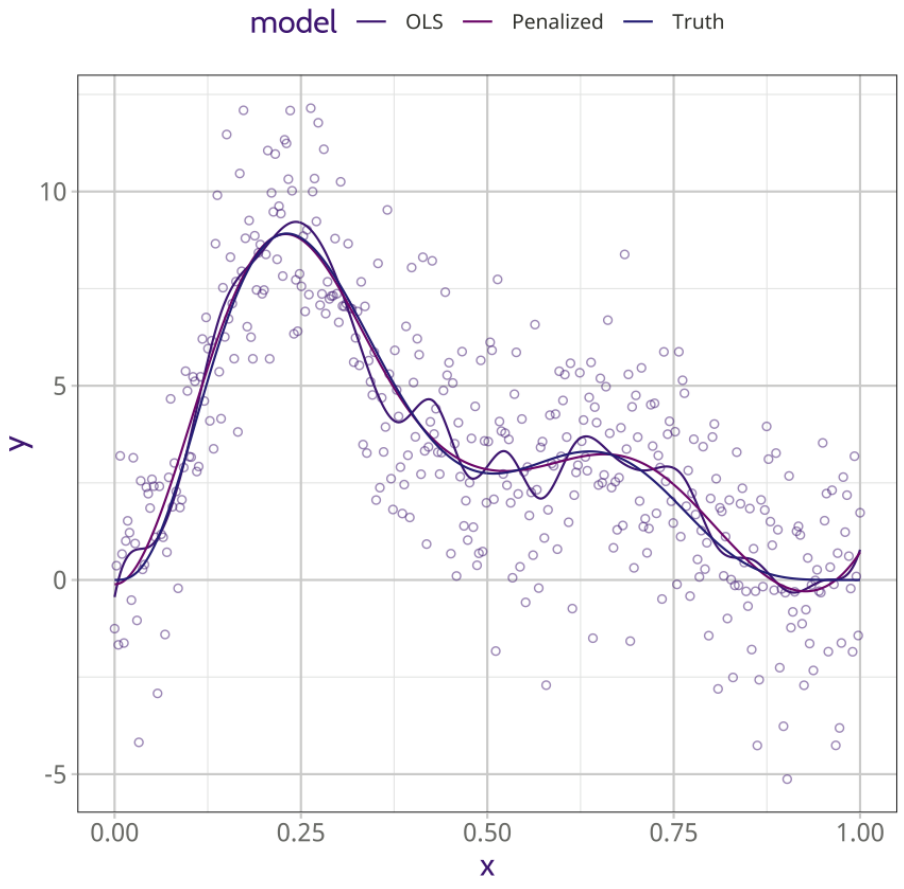


Notes

Type notes here...

Example

```
tpb <- function(x, degree=3, nknots=3){
out <- sapply(1:degree, function(d)x^d)
s <- seq(min(x, na.rm=TRUE), max(x, na.rm=TRUE), length=nknots+2)
s <- s[-c(1, length(s))]
for(i in 1:length(s)){
  out <- cbind(out, (x-s[i])^3*(x >= s[i]))
}
colnames(out) <- paste0("tpb", 1:ncol(out))
return(out)
}
df <- data.frame(x=x, y=y, f=f)
mod <- lm(y ~ tpb(x, 3, 20), data=df)
D <- diag(24)
D[1:4,1:4] <- 0
X <- cbind(1, tpb(x, 3, 20))
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
```



Notes

Type notes here...

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), data=dframe)
```

Notes

Type notes here...

Summary

```r
summary(mod)
```

```
## *******************************************************************
## Family:  c("NO", "Normal")
##
## Call:  gamlss(formula = y ~ pb(x), data = dframe)
##
## Fitting method: RS()
##
## ------------------------------------------------------------------
## Mu link function:  identity
## Mu Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   6.2785     0.1913   32.83   <2e-16 ***
## pb(x)        -5.6310     0.3311  -17.01   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ------------------------------------------------------------------
## Sigma link function:  log
## Sigma Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.65040    0.03536    18.4   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ------------------------------------------------------------------
## NOTE: Additive smoothing terms exist in the formulas:
##  i) Std. Error for smoothers are for the linear effect only.
```

Notes

Type notes here...

Smooth Term

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

```
## P-spline fit using the gamlss function pb()
## Degrees of Freedom for the fit : 11.25376
## Random effect parameter sigma_b: 1.29549
## Smoothing parameter lambda     : 0.613094
```

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

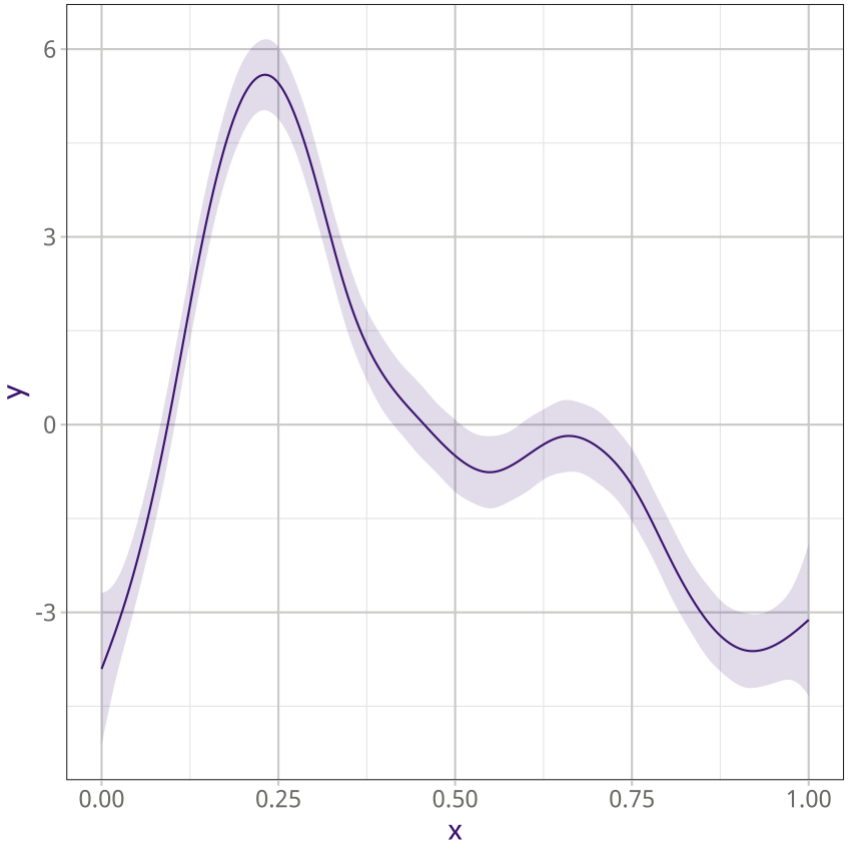
```
##  [1] -8.80532275 -7.07334035 -5.29320623 -2.54552539  1.14014680  3.66217710
##  [7]  4.41469105  3.17578616  1.02161400  0.12984069 -0.18952821 -0.54987832
## [13] -0.59184225  0.08041483  0.78006187  0.84320616  0.48724692 -0.51246890
## [19] -1.22287182 -1.40918552 -0.95398350 -0.14971796  0.69056121
```

Notes

Type notes here...

Plot

```
out <- termplot(mod, se=TRUE, plot=FALSE)
ggplot(out$x, aes(x=x, y=y)) +
  geom_ribbon(aes(ymin = y-1.96*se,
                  ymax=y+1.96*se),
              alpha=.15, fill=pal2[1],
              col="transparent") +
  geom_line(col=pal2[1]) +
  theme_bw() +
  mytheme() +
  labs(x="x", y="y")
```



Notes

Type notes here...

GAMLSS Algorithm

1. Outer iteration: loops over moments - μ, σ, ν and τ - calling inner iteration
2. Inner iteration - Calculate z_k (working response) and weights w_k and call modified backfitting algorithm.
3. Smooth residuals against X to calculate new parameter estimates.

Notes

Type notes here...

Estimating lambda

- $\varepsilon = \mathbf{Z}\gamma + \mathbf{e}$ (partial residual)
- $e \sim \mathcal{N}(0, \sigma_e^2 \mathbf{W}^{-1})$ (idiosyncratic residuals)
- $\gamma \sim \mathcal{N}(\mathbf{0}, \sigma_b^2 \mathbf{G}^{-1})$ (parameters)

1. $\hat{\gamma} = \left(\mathbf{Z}'\mathbf{W}\mathbf{Z} + \hat{\lambda}\mathbf{G}\right)^{-1} \mathbf{Z}'\mathbf{W}\varepsilon$ (weighted least squares)

2. $\hat{\lambda} = \frac{\hat{\sigma}_e^2}{\hat{\sigma}_b^2}$, where:

   - $\hat{\varepsilon} = \mathbf{Z}\hat{\gamma}$
   - $\hat{\sigma}_e^2 = (\varepsilon - \hat{\varepsilon})'(\varepsilon - \hat{\varepsilon})/(n - tr(\mathbf{S}))$
   - $\hat{\sigma}_b^2 = \hat{\gamma}'\hat{\gamma}/tr(\mathbf{S})$
   - $S = \mathbf{Z}\left(\mathbf{Z}'\mathbf{W}\mathbf{Z} + \hat{\lambda}\mathbf{G}\right)^{-1} \mathbf{Z}'\mathbf{W}$

Notes

Type notes here...

Replicating Ourselves

Let's do some of the same things we've done already in this framework:

1. Modeling non-linearity in the Jacobson Data.
2. Monotone smoothing of democracy and vote choice.
3. Ridge and LASSO models.

Notes

Type notes here...

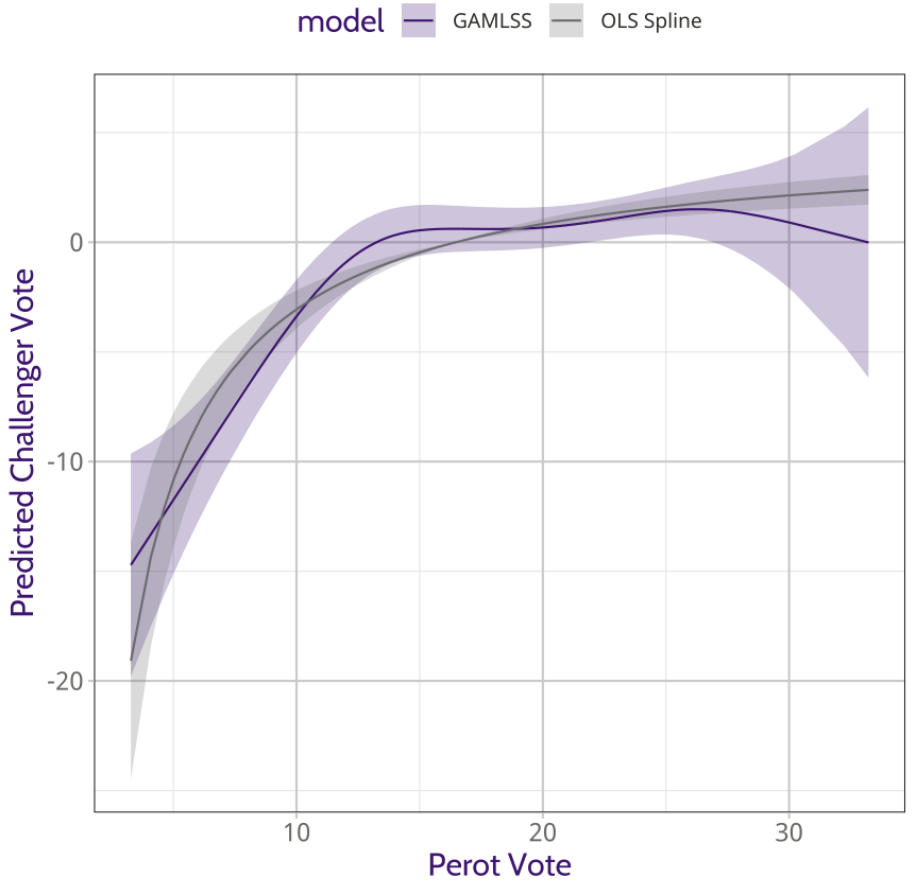
Jacobson Data

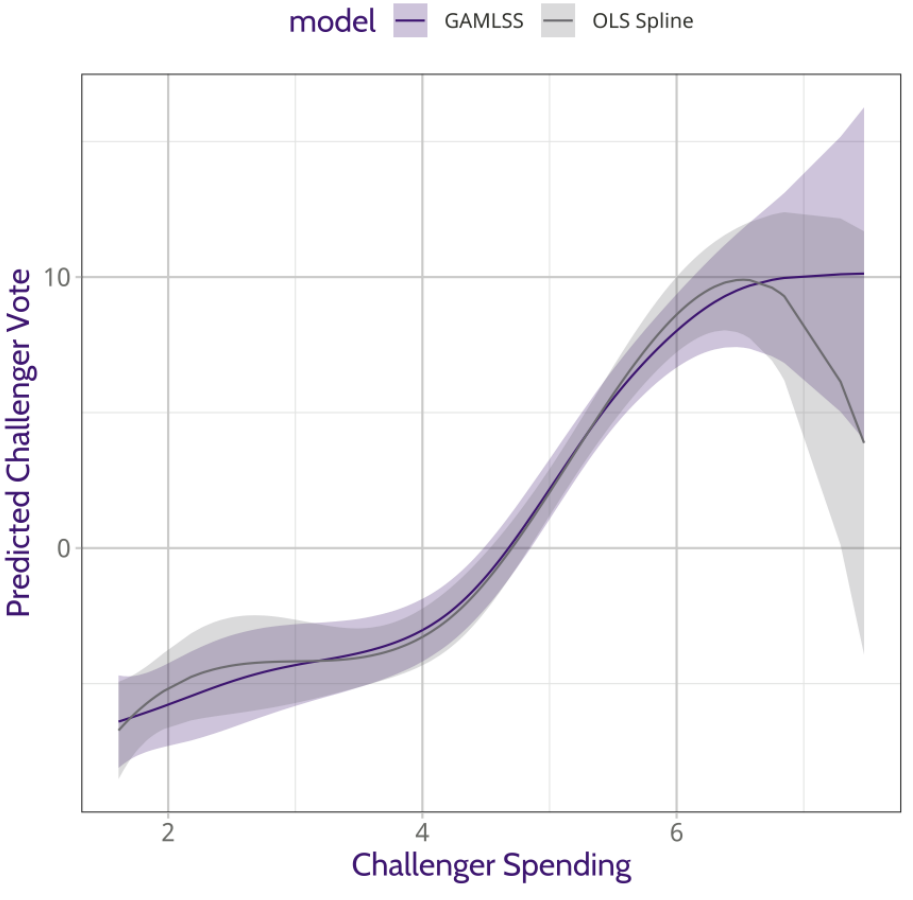
```
library(gamlss)
library(car)
library(foreign)
dat <- read.dta(
   "http://www.quantoid.net/files/reg3/jacob.dta")

gmod <- gamlss(chal_vote ~ pb(perotvote) + pb(chal_spend) +
    exp_chal, data=dat)

invx <- function(x)(1/x)
spline.mod <- gamlss(chal_vote ~ invx(perotvote) +
                   bs(chal_spend, df=4) +
                   exp_chal, data=dat)

tp <- termplot(gmod, se=TRUE, plot=FALSE)
tp2 <- termplot(spline.mod, se=TRUE, plot=FALSE)
tp_pv <- bind_rows(tp[[1]], tp2[[1]])
tp_pv$model <- factor(rep(1:2, each=nrow(tp[[1]])),
                      labels=c("GAMLSS", "OLS Spline"))
tp_cs <- bind_rows(tp[[2]], tp2[[2]])
tp_cs$model <- factor(rep(1:2, each=nrow(tp[[2]])),
                      labels=c("GAMLSS", "OLS Spline"))
```

Notes

Type notes here...

plots


model
GAMLSS
OLS Spline
Predicted Challenger Vote
0
-10
-20
10
20
30
Perot Vote


model
GAMLSS
OLS Spline
Predicted Challenger Vote
10
0
2
4
6
Challenger Spending

Notes

Type notes here...

Testing the models

We can use the Clarke test to evaluate the difference between the two models.

```
VC.test(gmod, spline.mod)
```

```
##  Vuong's test: -2.475 model spline.mod is preferred over gmod
## Clarke's test: 95 p-value= 0 spline.mod is preferred over gmod
```

In this case, the spline model is better - generally similar parametric models will be preferred to non-parametric models.

Notes

Type notes here...

Monotonic Democracy

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

```
library(foreign)
dat <- read.dta("http://www.quantoid.net/files/reg3/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.mod1
```

```
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
```

Notes

Type notes here...

Monotone Party ID

```r
library(foreign)
anes <- read.dta("http://www.quantoid.net/files/reg3/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 = 768.0012
## GAMLSS-RS iteration 2: Global Deviance = 768.0012
```

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

```
## GAMLSS-RS iteration 1: Global Deviance = 772.5639
## GAMLSS-RS iteration 2: Global Deviance = 772.5645
```

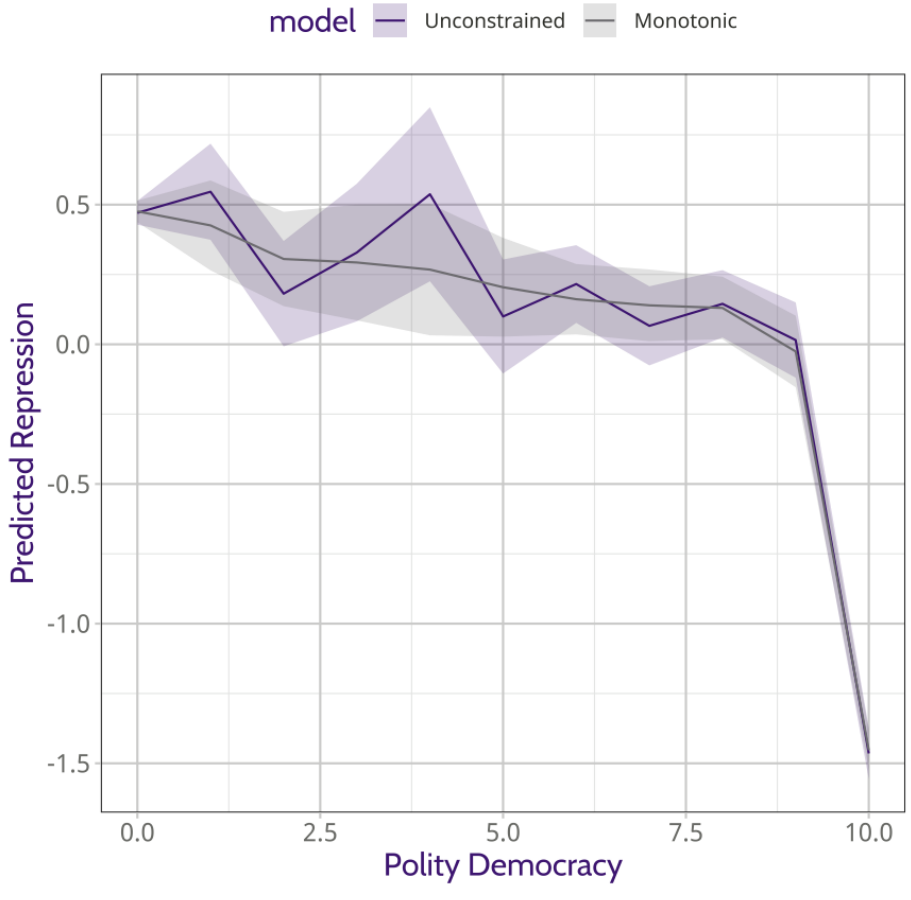
```r
source("https://quantoid.net/files/reg3/vctest2.r")
VC.test2(unrestricted.mod2, mono.mod2)
```

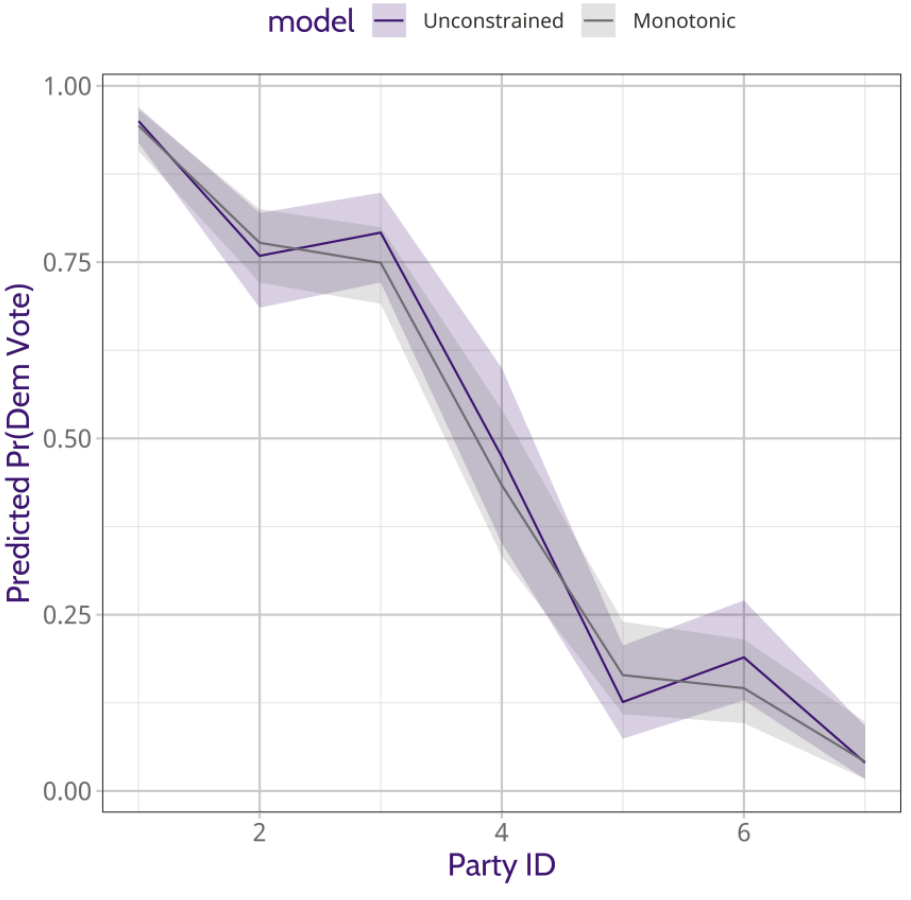
```
##  Vuong's test: -3.789 model mono.mod2 is preferred over unrestricted.mod2
## Clarke's test: 323 p-value= 0 mono.mod2 is preferred over unrestricted.mod2
```

Notes

Type notes here...

Plots


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

Notes

Type notes here...

Collinearity

```r
set.seed(1234)
Sig <- diag(5)
Sig[3:5,3:5] <- .99
diag(Sig) <- 1
X <- MASS::mvrnorm(500,rep(0,5), Sig)
b <- c(1,1,1,0,0)
ystar <- X %*% b
y <- ystar + rnorm(500, 0, 2)
```

```r
summary(m1 <- lm(y~ X))
```

```
##
## Call:
## lm(formula = y ~ X)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.4195 -1.3696 -0.0068  1.4012  5.3665
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.01686    0.08762   0.192   0.8475
## X1           1.17283    0.09359  12.532   <2e-16 ***
## X2           1.11657    0.09163  12.185   <2e-16 ***
## X3           1.21441    0.69342   1.751   0.0805 .
## X4           1.04118    0.68252   1.525   0.1278
## X5          -1.09301    0.70047  -1.560   0.1193
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.953 on 494 degrees of freedom
## Multiple R-squared:  0.469,    Adjusted R-squared:  0.4637
## F-statistic: 87.28 on 5 and 494 DF,  p-value: < 2.2e-16
```

Notes

Type notes here...

Partial Regularization

```
dframe <- as.data.frame(cbind(y, X))
names(dframe) <- c("y", "x1", "x2", "x3", "x4", "x5")
rmod <- gamlss(y ~ x1 + x2 + ri(x.vars=c("x3", "x4", "x5"), Lp = 2), data=dframe)
lmod <- gamlss(y ~ x1 + x2 + ri(x.vars=c("x3", "x4", "x5"), Lp = 1), data=dframe)
noreg <- gamlss(y ~ x1 + x2 + x3 + x4 + x5, data=dframe)
```

```
VC.test(rmod, lmod)
```

```
##  Vuong's test: -2.807 model lmod is preferred over rmod
## Clarke's test: 186 p-value= 0 lmod is preferred over rmod
```

```
VC.test(rmod, noreg)
```

```
##  Vuong's test: 5.159 model rmod is preferred over noreg
## Clarke's test: 339 p-value= 0 rmod is preferred over noreg
```

```
VC.test(lmod, noreg)
```

```
##  Vuong's test: 4.627 model lmod is preferred over noreg
## Clarke's test: 329 p-value= 0 lmod is preferred over noreg
```

Notes

Type notes here...

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.

Notes

Type notes here...

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).

Notes

Type notes here...

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.

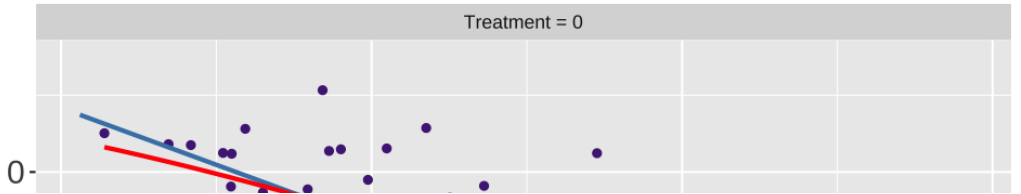
Notes

Type notes here...

Diagnostics I

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

```r
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)
```



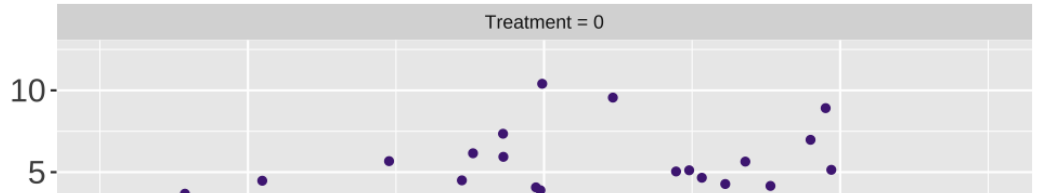
Notes

Type notes here...

Diagnostics II

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

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

plot(i2)
```



Notes

Type notes here...

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:

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

Notes

Type notes here...

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.

Notes

Type notes here...

Binning Estimator

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

Notes

Type notes here...

Binning Estimator II

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

Notes

Type notes here...

Kernel Estimator

A kernel estimator is more flexible allowing the conditional partial effect to change across the values of X.

$$Y = f(X) + g(X)D + \gamma(X)Z + \varepsilon$$

- This is the standard linear interaction effect model when $f(x) = \mu + \nu X$ and $g(x) = \alpha + \beta X$ and $\gamma(X) = \gamma$.

Notes

Type notes here...

Figure

```r
k1 <- interflex(estimator="kernel",
                dat,
                "Y1",
                "D1",
                "X1",
                treat.type="discrete",
                parallel=FALSE)
plot(k1)
```

```
## Baseline group not specified; choose treat = 0 as the baseline group.
## Cross-validating bandwidth ...
## ............................Optimal bw=0.1712.
## Number of evaluation points:50
## ..................................................50.........................................
```

Notes

Type notes here...

Figure II

```
k2 <- interflex(estimator="kernel",
                dat,
                "Y2",
                "D1",
                "X2",
                treat.type="discrete",
                parallel=FALSE)
plot(k2)
```

```
## Baseline group not specified; choose treat = 0 as the baseline group.
## Cross-validating bandwidth ...
## ............................Optimal bw=0.6046.
## Number of evaluation points:50
## ..................................................50.........................................
```

Notes

Type notes here...

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

Notes

Type notes here...

Wald Test

```r
it1 <- inter.test(k1, diff.values=c(2.4,3,3.6), percentile=FALSE)
it1
```

```
## $`1`
##            diff.estimate    sd z-value p-value lower CI(95%) upper CI(95%)
## 3 vs 2.4           0.593 1.520   0.390   0.697        -2.386         3.571
## 3.6 vs 3           0.598 1.313   0.456   0.649        -1.975         3.172
## 3.6 vs 2.4         1.191 1.719   0.693   0.488        -2.178         4.560
```

```r
it2 <- inter.test(k2, diff.values=c(-1.7, 0, 1.7), percentile=FALSE)
it2
```

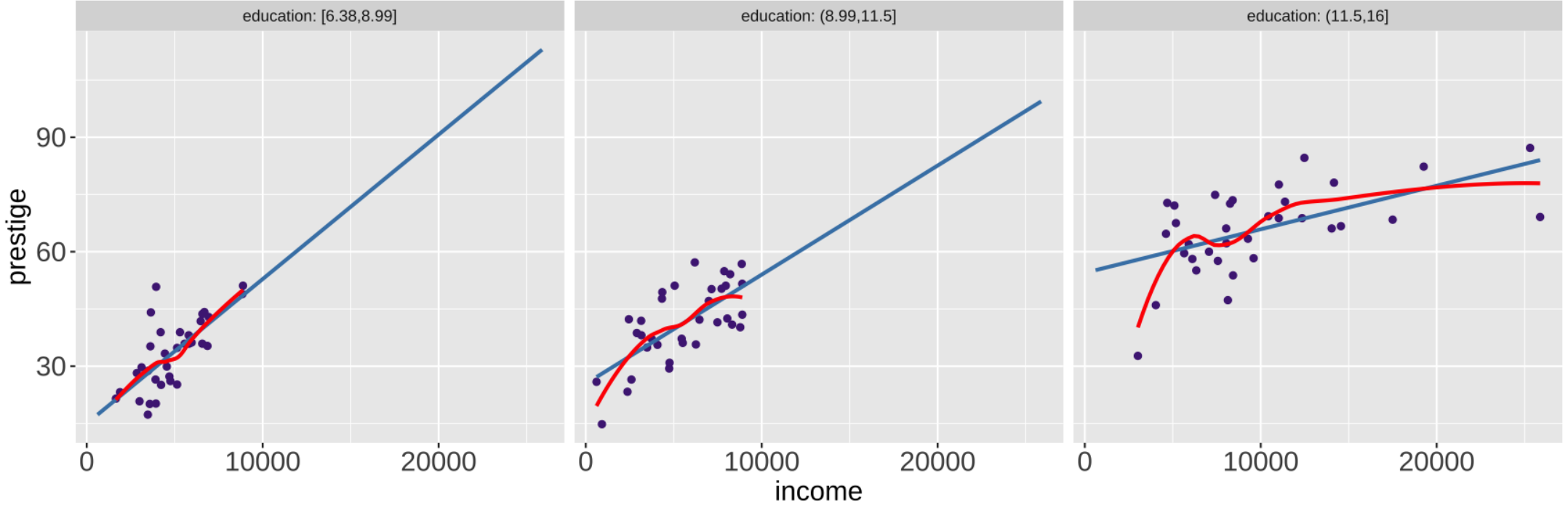
```
## $`1`
##             diff.estimate    sd z-value p-value lower CI(95%) upper CI(95%)
## 0 vs -1.7          -6.008 1.202  -4.999   0.000        -8.364        -3.652
## 1.7 vs 0            6.750 1.192   5.662   0.000         4.413         9.087
## 1.7 vs -1.7         0.742 1.305   0.569   0.569        -1.815         3.300
```

Notes

Type notes here...

On Prestige Data

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



Notes

Type notes here...

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
```

Notes

Type notes here...

Kernel Estimator: Prestige Data

```
pres.k <- interflex("kernel",
                    Prestige,
                    "prestige",
                    "income",
                    "education",
                    Z=c("type",
                        "women"),
                    na.rm=T,
                    figure=FALSE,
                    treat.type="continuous",
                    parallel=FALSE)
```

Notes

Type notes here...

Test: Prestige Data

```r
inter.test(pres.k, diff.values=c(9,11,13), percentile=FALSE)
```

```
## $`income=6035.5 (50%)`
##          diff.estimate    sd z-value p-value lower CI(95%) upper CI(95%)
## 11 vs 9         -0.003 0.001  -4.316   0.000        -0.004        -0.001
## 13 vs 11        -0.001 0.000  -2.808   0.005        -0.002        -0.000
## 13 vs 9         -0.004 0.001  -7.253   0.000        -0.005        -0.003
```

Notes

Type notes here...

GAMs for the Fake Data I

```r
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
```

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

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

```r
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
```

Notes

Type notes here...

Bootstrapping the GAMLSS Model

Assuming Fixed λ.

```
tot_rep <- 250
# p <- progress_estimated(tot_rep+1)  # init progress bar
newdat <- tibble(X1 = rep(seq(0.1, 5.8, length=50), 2),
                 D1 = rep(c(0,1), each=50),
                 D10 = 1-D1)
res <- NULL
for(i in 1:tot_rep){
  inds <- sample(1:nrow(dat), nrow(dat), replace=TRUE)
  m <- gamlss(Y1 ~ D1 +
               pb(I(X1*(D1 == 1)), lambda = mod1$mu.coefSmo[[1]]$lambda) +
               pb(I(X1*(D1 == 0)), lambda = mod1$mu.coefSmo[[2]]$lambda),
              data=dat[inds, ], trace=FALSE)
  sink(tempfile())
  fit <- predict(m, newdata=newdat)
  sink()
  # p$tick()$print()
  res <- rbind(res, fit)
}

boot.cis <- t(apply(res, 2, quantile, c(.025,.975)))
```

Notes

Type notes here...

Bootstrapping the GAMLSS Model

Including uncertainty in λ.

```
tot_rep <- 250
p <- progress_estimated(tot_rep+1)  # init progress bar
newdat <- tibble(X1 = rep(seq(0.1, 5.8, length=50), 2),
                 D1 = rep(c(0,1), each=50),
                 D10 = 1-D1)
res <- NULL
for(i in 1:tot_rep){
  inds <- sample(1:nrow(dat), nrow(dat), replace=TRUE)
  m <- gamlss(Y1 ~ D1 +
               pb(I(X1*(D1 == 1))) +
               pb(I(X1*(D1 == 0))),
              data=dat[inds, ], trace=FALSE)
  sink(tempfile())
  fit <- predict(m, newdata=newdat)
  sink()
  p$tick()$print()
  res <- rbind(res, fit)
}
```

```
boot.cis2 <- t(apply(res, 2, quantile, c(.025,.975)))
```

Notes

Type notes here...

Bootstrap CI

```
plot.dat <- rbind(newdat, newdat)
plot.dat$method <- factor(
  rep(1:2, each=nrow(newdat)),
  labels=c("Fixed Lambda", "Variable Lambda"))
plot.dat$lower <- c(boot.cis[,1], boot.cis2[,1])
plot.dat$upper <- c(boot.cis[,2], boot.cis2[,2])
plot.dat$fit <- rep(predict(mod1, newdata=newdat), 2)
```

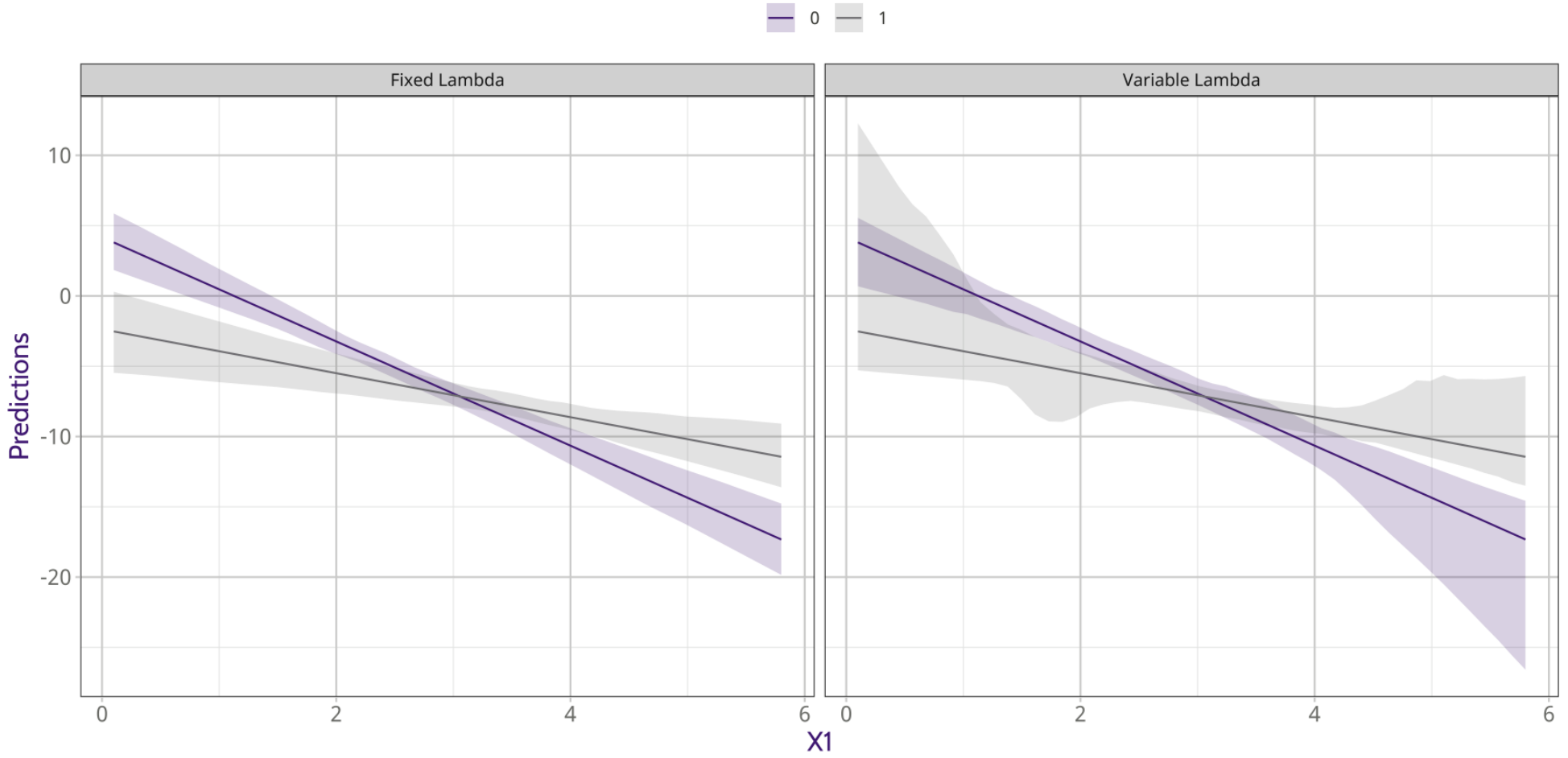
```
## new prediction
## New way of prediction in pb()  (starting from GAMLSS version 5.0-3)
## New way of prediction in pb()  (starting from GAMLSS version 5.0-3)
```

```
g <- ggplot(plot.dat, aes(x=X1, y=fit,
                   colour=as.factor(D1),
                   fill=as.factor(D1))) +
  geom_ribbon(aes(ymin=lower, ymax=upper),
              alpha=.2,
              col="transparent") +
  geom_line() +
  facet_wrap(~method) +
  scale_colour_manual(values=pal2) +
  scale_fill_manual(values=pal2) +
  theme_bw() +
  mytheme() +
  theme(legend.position = "top") +
  labs(x = "X1", y="Predictions", colour="", fill="")
```

Notes

Type notes here...

Figure


0
1
Fixed Lambda
Variable Lambda
Predictions
10
0
-10
-20
0
2
4
6
X1

Notes

Type notes here...

GAMs for the Fake Data II

```r
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
```

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

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

```r
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
```

Notes

Type notes here...

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), ]
```

Notes

Type notes here...

Treatment Effect Plot

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

Notes

Type notes here...

GAMs for the Prestige Data

```r
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
```

```r
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
```

```r
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
```

Notes

Type notes here...

Interactions Redux

The GAMLSS framework provides us the ability to:

- model general non-linear interactions in both continuous and discrete variables
- do most everything suggested by Hainmueller et al.
- use non-linear interactions in variance modeling, too.

Notes

Type notes here...