Presidential Forecasting with bigKRLS

Pete Mohanty, Stanford University
Robert B. Shaffer, University of Texas at Austin

BARUG October 2016

Overview

- **Part I**: Kernel Regularized Least Squares (KRLS)
  - What is KRLS? Why use it?
  - Speed/memory tradeoffs and bigKRLS
- **Part II**: Political Forecasting and bigKRLS
  - Exploratory Modeling in the Two-Party Case
  - (The Limits of) Pooling Publicly Available Polls
  - Forecasting Voting Choices

Kernel Regularized Least Squares

Motivations

- Maximize inference and out-of-sample prediction, minimize assumptions
- estimate not just "average" but "actual" marginal effects

Statistical Properties & Applications

- For details on statistical properties, see Hainmueller & Hazlett 2013
- "Kernel balancing" may help observational studies approximate experiments; see, e.g., Hazlett; Hastie et al 2008

The Challenge

Speed

- Luke Sonnet notes speed problems with *KRLS* starting at N $\approx$ 1,000 but has made substantial speed gains in Julia.

Size

- "Tikhonov regularization requires computation of weight matrices of dimension **N** x **N** which [...] may be unsuitable for large datasets." (Racine and Hayfield)
- Typical machines hit "cannot allocate vector" limits at N $\approx$ 3,000 with *KRLS*

Some details

We assume that the objective function $\mathbf{y} = \mathbf{f}(\mathbf{x})$ can be approximated by

$$\mathbf{y} = \mathbf{Kc}$$

With $\mathbf{K}$ the Gaussian kernel,

$$\mathbf{K} = e^{-||x_i - x_j||^2/\sigma^2}$$

and $\mathbf{c}$ a vector of weights chosen based on an L_2 penalty,

$$\hat{\mathbf{c}} = \underset{c \in \mathbb{R}^P}{\operatorname{argmin}} (\mathbf{y} - \mathbf{Kc})'(\mathbf{y} - \mathbf{Kc}) + \lambda \mathbf{c}'\mathbf{Kc}$$

Introducing bigKRLS

bigKRLS: a new version of the algorithm which minimizes memory constraints and boosts speed in **R**.

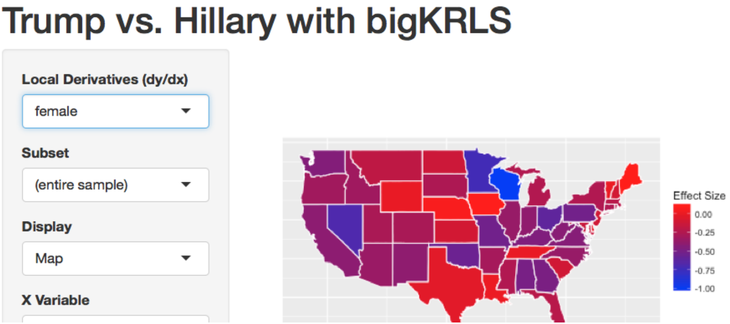
- Reduce peak memory requirements from $\approx 9PN^2$ to $\approx 5N^2$
- "big R" (bigmemory, bigalgebra & biganalytics)
- $R -> C++$ (Rcpp & RcppAramdillo)

bigKRLS Complexity

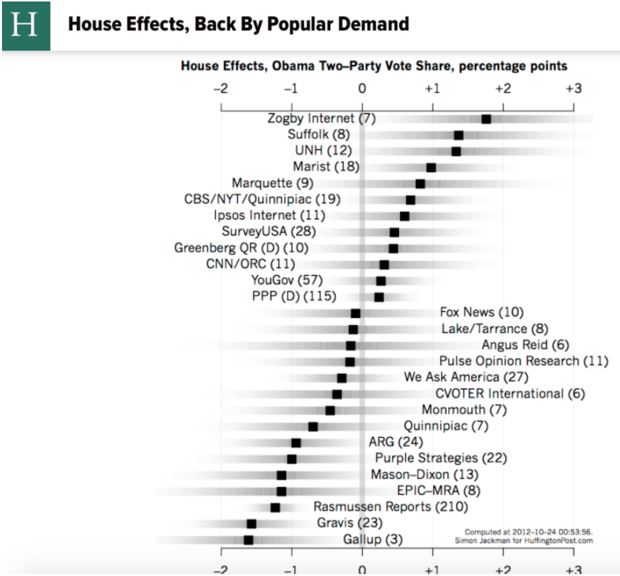
	Major Steps	Runtime	Memory
(1)	Standardize $\mathbf{X}_{\mathbf{N*P}}$, $\mathbf{y}$	—	—
(2)	Calculate kernel $\mathbf{K}_{N \times N}$	$O(N^2)$	$O(N^2)$
(3)	Eigendecompose $\mathbf{KE} = \mathbf{Ev}$	$O(N^3)$	$O(N^2)$
(4)	Regularization parameter λ	$O(N^3)$	—
(5)	Estimate weights $\hat{\mathbf{c}} = \mathbf{f}(\lambda, \mathbf{y}, \mathbf{E}, \mathbf{v})$	$O(N^3)$	—
(6)	Fit values $\hat{\mathbf{y}} = \mathbf{K}\hat{\mathbf{c}}$	—	—
(7)	Estimate local derivatives, $\hat{\mathbf{\Delta}}_{\mathbf{N*P}} = [\hat{\delta}_1 \quad \hat{\delta}_2 \ldots \hat{\delta}_{\mathbf{P}}]$	$O(PN^3)$	$O(N^2)$

Letting i, j index observations and $p = 1, 2, \ldots P$ index x variables. Steps 4-6 are followed by uncertainty estimates.

Effect of Gender on Two-Party Preference



Trust the Polls? Or a Poll of Polls?



Simon Jackman in HuffPost, 10/12

pollstR: the Huffington Post R API

```
library(dplyr); library(tidyr);
library(pollstR); library(bigKRLS)
```

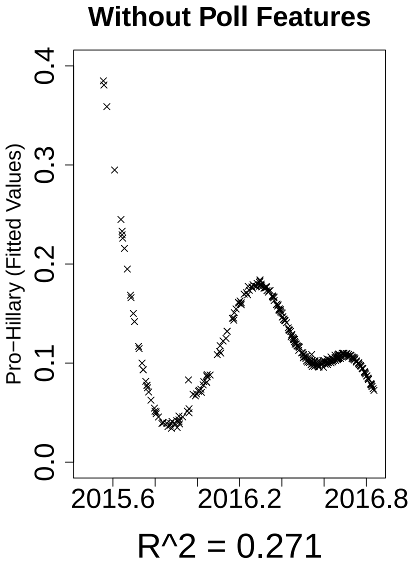
```
slug <- "2016-general-election-trump-vs-clinton"
pollstR_data <- read.csv(pollstR::chart_data_url(slug))
glimpse(pollstR_data)
```

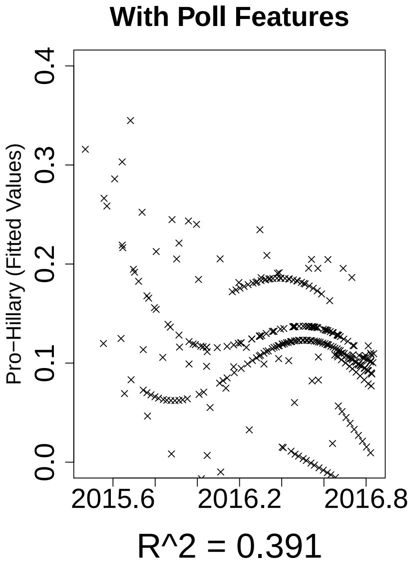
```
## Observations: 310
## Variables: 13
## $ Trump                <dbl> 43, 39, 43, 45, 44, 49, 44,
## $ Clinton              <dbl> 48, 46, 49, 51, 50, 47, 49,
## $ Other                <dbl> 8, NA, 4, 4, NA, 4, 2, NA,
## $ Undecided            <dbl> 1, 16, 5, 0, 6, NA, 6, 6, 1
## $ poll_id              <int> 25893, 25827, 25894, 25876,
## $ pollster             <fctr> YouGov/Economist, Morning
## $ start_date           <fctr> 2016-10-01, 2016-09-30, 20
## $ end_date             <fctr> 2016-10-03, 2016-10-02, 20
## $ sample_subpopulation <fctr> Registered Voters, Likely
```

Testing for House Effects with bigKRLS

```r
y <- log(pollstR_data$Clinton/pollstR_data$Trump)

t <- grep("start", colnames(X))
time_only <- bigKRLS(y , X[,t])

P <- ncol(X)
type <- grep("asked_third_party", colnames(X))
with_features <- bigKRLS(y, X[,c(type, t:P)])
```

Not So Random Errors...





Election Forecasting and bigKRLS

- We model Presidential voting as a three-step process:
  - Turnout?
  - If turnout, third party or two-party?
  - If two-party, Democrat or Republican?
- Assume that third-party voters need to be modeled differently than two-party voters (fits with Perot, Gary Johnson, etc. . .)
- For our forecast, we need to:
  - Generate probability for each step
  - Model each individual-level probability
  - Split sample and predict probability of each outcome for each individuals
  - Simulate Many Elections

Variable Selection and Preprocessing

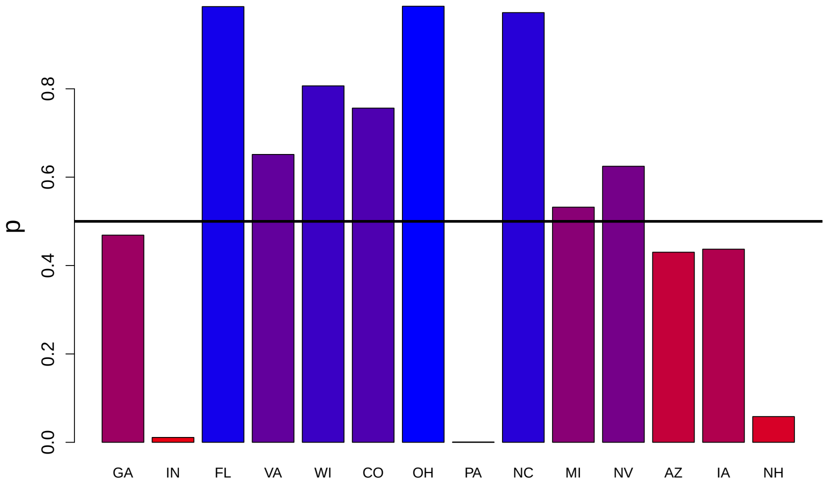
- Generate dependent variables using Bayesian measurement model
  - MCMCpack's method for mixed factor analysis
  - ANES (American National Elections Study) Jan 2016 Data (Individual Level)
  - State-Level Data on Recent Elections
- Then, model probabilities as Function of...
  - Individual Level Data: Political Preferences, Demographics...
  - State Level Data: State Demographchcis (Kaggle), Geolocation
  - Survey Weights

prediction with bigKRLS

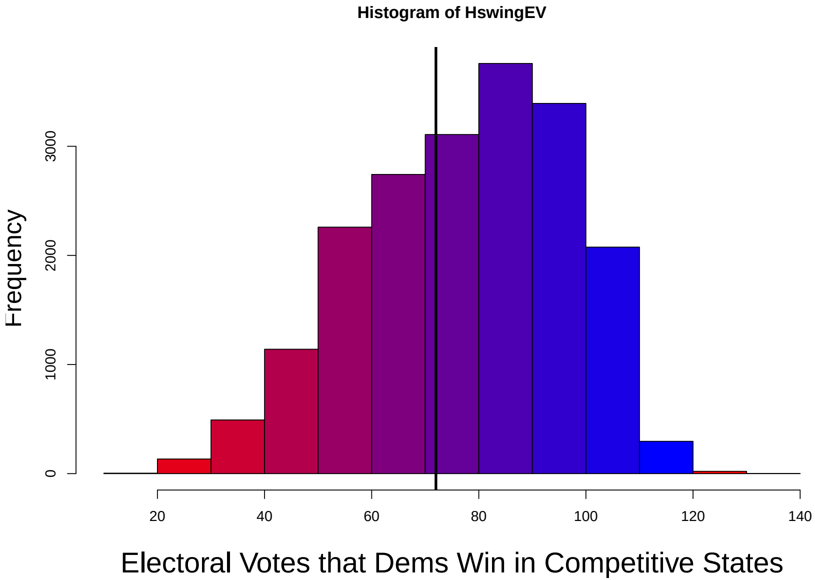
```r
set.seed(1234)
train <- sample(N, 800)
test <- !(1:N %in% train)

turnout_out <- bigKRLS(p_turnout[train, ],
                       Xturnout[train, ])

predict_turnout <- predict(turnout_out,
                           Xturnout[test, ])

cor(predict_turnout$fit, p_turnout[test,])^2
```

Out-of-Sample Pseudo R^2 $>$ 0.945 for all models

Swing State Simulations



Forecast: Hillary Wins in 62.9% of Sims



Take Aways

- *bigKRLS* is flexible and interpretable but very computationally intensive
- Not a panacea but able to replicate leading forecasts with small ammounts of limited data

Next Steps for the Algorithm...

- Parallel Processing via RcppParallel or snow (in development)
- Develop practical benchmarks for assessing asymptotics
- "big" Eigentruncation (*partial_eigen* in irlba or the power method...)]
- more "Shiny" features

Thank You!!

bigKRLS on GitHub:

`https://github.com/rdrr1990/bigKRLS.git`

Keep in Touch!

Robert: @rbshaffer (Me): @petemohanty