Predicting 2000 Presidential Election Results At the County Level Using Public Data

Mose Wintner

November 2016

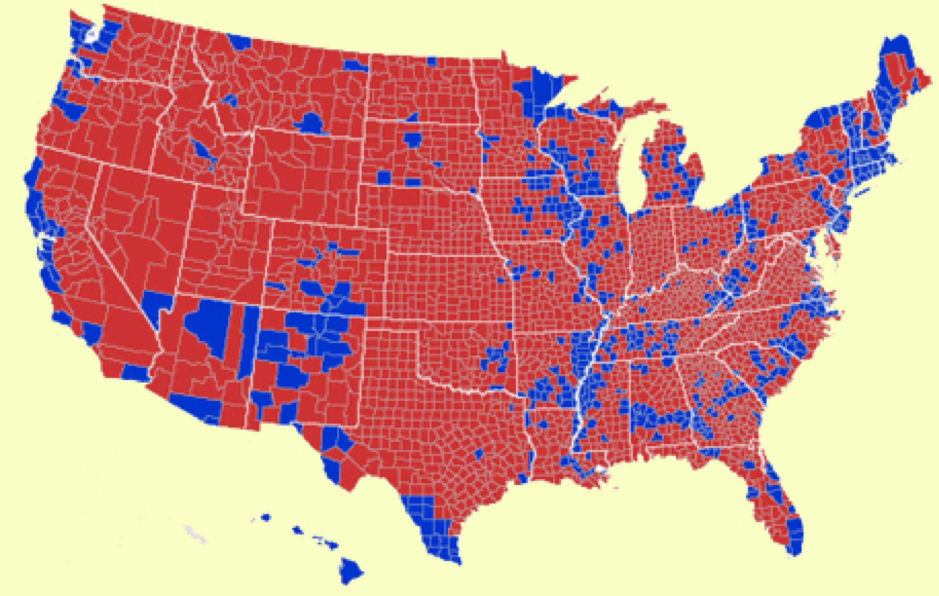
The data

All data came from the U.S. Census Bureau's Counties Database at `http://www.census.gov/support/USACdataDownloads.html`, which is no longer actively maintained. After cleaning data, $n = 3074$ counties. Used $p = 77$ total variables consisting of demographic data, housing data, and features engineered from those.

The data

All data came from the U.S. Census Bureau's Counties Database at `http://www.census.gov/support/USACdataDownloads.html`, which is no longer actively maintained. After cleaning data, $n = 3074$ counties. Used $p = 77$ total variables consisting of demographic data, housing data, and features engineered from those.

```
 [1] "emplgov"              "emplstatelocalgov"    "hospinsured"       "nohealthinsurance"  "vehperhouse"
 [6] "publicindus"          "flow10yr"             "birthsper1000"     "deathsper1000"      "publicwaterpercap"
[11] "domesticwaterpercap"  "irrigationwater"      "industrialwater"   "onfarms"            "age35_44"
[16] "black1"               "indian1"              "asian1"            "other1"             "multrace"
[21] "hisppop"              "white1nh"             "nevermarried"      "married"            "widowed"
[26] "englishonly"          "noenglish"            "foreignborn"       "naturalized"        "enteredlast10yrs"
[31] "samehouse"            "samecounty"           "inpvtschools"      "incollege"          "lessthan9th"
[36] "somehighschool"       "highschool"           "somecollege"       "bachelors"          "veteran"
[41] "civlabforce"          "unemployedclf"        "manprofoccs"       "serviceoccs"        "salesoffoccs"
[46] "farmfishoccs"         "consoccs"             "transoccs"         "pci"                "poor"
[51] "vacant"               "mobilehomes"          "ageunit5"          "avgageunit"         "mediangrossrent"
[56] "nocashrent"           "medianhvalue"         "latitude"          "longitude"          "lnpop"
[61] "lnland"               "lnpoppersqm"          "gwagegap"          "sepdiv"             "urban"
[66] "homeless"             "housingincomediscrep" "over55"            "age18to35"          "european"
[71] "asian"                "latinamerican"        "boomers"           "bankspers1000pop"   "vcper1000"
[76] "grad"                 "voterparticip"
```



I used the R package `caret` for my analyses. With it, one can

- easily fit any of 230 (at present) models, using any of several cross-validation methods, including
  - LOOCV
  - k-fold cross-validation
  - repeated k-fold cross-validation
  - bootstrap

I used the R package `caret` for my analyses. With it, one can

- easily fit any of 230 (at present) models, using any of several cross-validation methods, including
  - LOOCV
  - *k*-fold cross-validation
  - repeated *k*-fold cross-validation
  - bootstrap
- can easily optimize model hyperparameters via grid search

I used the R package `caret` for my analyses. With it, one can

- easily fit any of 230 (at present) models, using any of several cross-validation methods, including
  - LOOCV
  - k-fold cross-validation
  - repeated k-fold cross-validation
  - bootstrap
- can easily optimize model hyperparameters via grid search
- train several different models on the same dataset

I used the R package `caret` for my analyses. With it, one can

- easily fit any of 230 (at present) models, using any of several cross-validation methods, including
  - LOOCV
  - k-fold cross-validation
  - repeated k-fold cross-validation
  - bootstrap
- can easily optimize model hyperparameters via grid search
- train several different models on the same dataset
- can keep track of cross-validation accuracy/fit easily

I used the R package `caret` for my analyses. With it, one can

- easily fit any of 230 (at present) models, using any of several cross-validation methods, including
  - LOOCV
  - *k*-fold cross-validation
  - repeated *k*-fold cross-validation
  - bootstrap
- can easily optimize model hyperparameters via grid search
- train several different models on the same dataset
- can keep track of cross-validation accuracy/fit easily

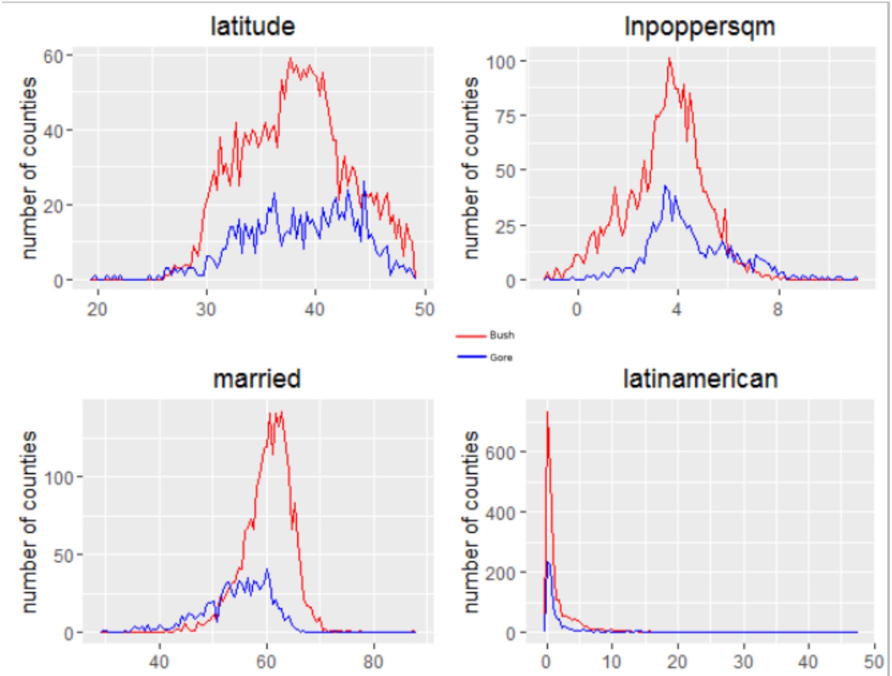
QDA

```
qda.control=trainControl(method="boot",number=10)
set.seed(323)
qda.fit=train(rpct~.,data=n.2,
              method="qda",
              trControl=qda.control)
```

```
> qda.fit$finalModel
Call:
qda(rpct ~ ., data = n.2)

Prior probabilities of groups:
     Bush      Gore 
0.7332466 0.2667534 

Group means:
      emplgov emplstatelocalgov hospinsured nohealthinsurance vehperhouse publicindus flow10yr birthsper1000 deathsper1000
Bush 8.356416          7.008239    16.69069          14.34338    1.897023    1196.015 12.09002      12.65018     10.366105
Gore 8.921362          7.391769    15.82640          14.49195    1.726756    4300.324  8.83378      13.21890      9.972439
     publicwaterpercap domesticwaterpercap irrigationwater industrialwater   onfarms age35_44    black1  indian1    asian1
Bush          170.4839            65.75942        48.54516        4.687516 5.683452 15.47831  6.286424 1.340861 0.5654836
Gore          157.5485            58.69385        33.27694       11.253049 2.712195 15.63927 14.967805 2.295122 1.4656098
       other1 multrace  hisppop white1nh nevermarried  married  widowed englishonly noenglish foreignborn naturalized
Bush 2.520186 1.408829 5.692902 85.09525     20.94099 60.32023 7.698935    92.49011 0.4133540    2.946894    1.061579
Gore 2.758415 1.763415 7.507195 72.49780     26.10451 54.37610 7.690610    89.04780 0.5904878    4.778171    1.989390
     enteredlast10yrs samehouse samecounty inpvtschools incollege lessthan9th somehighschool highschool somecollege bachelors
Bush         1.337844  58.91415   78.37582     6.742946  4.102573    9.028128       13.39703   35.38705    26.46664  10.62946
Gore         2.002439  59.37439   80.53146     9.030610  5.551341    9.332317       13.94817   33.05427    25.16829  11.80488
      veteran civlabforce unemployedclf manprofoccs serviceoccs salesoffoccs farmfishoccs consoccs transoccs      pci     poor
Bush 14.19942    60.79441      5.340728    28.21704    15.42875     22.91579     2.473203 12.02169  18.94401 17252.08 51.07910
Gore 13.06378    60.01354      6.828780    28.96902    16.38646     23.89098     1.626707 10.83500  18.28951 18146.66 55.97695
       vacant mobilehomes  ageunit5 avgageunit mediangrossrent nocashrent medianhvalue latitude longitude    lnpop   lnland
Bush 14.55568    15.73713 10.911713   34.96615        427.1690   14.45315     79073.91 38.15129 -84.71307 10.02019 6.554082
Gore 13.16671    13.24902  9.620122   35.33378        475.9927   11.38329     98283.42 38.56710 -82.29080 10.87630 6.415878
     lnpoppersqm gwagegap   sepdiv    urban homeless housingincomediscrep   over55 age18to35  european     asian latinamerican
Bush    3.466203 12736.07 11.03891 35.64139 3.189175            0.2437511 25.04037  20.31668 0.4685516 0.4507653      1.806624
Gore    4.460402 12815.53 11.82707 50.40220 3.635244            0.2155721 23.22780  22.21195 0.9508463 1.1188263      2.304384
      boomers bankspер1000pop vcper1000     grad voterparticip
Bush 13.58119       0.4948137  2.125813 5.093301      39.98576
Gore 13.63707       0.3823309  3.256220 6.691220      39.61901
```


latitude
lnpoppersqm
married
latinamerican
number of counties
Bush
Gore

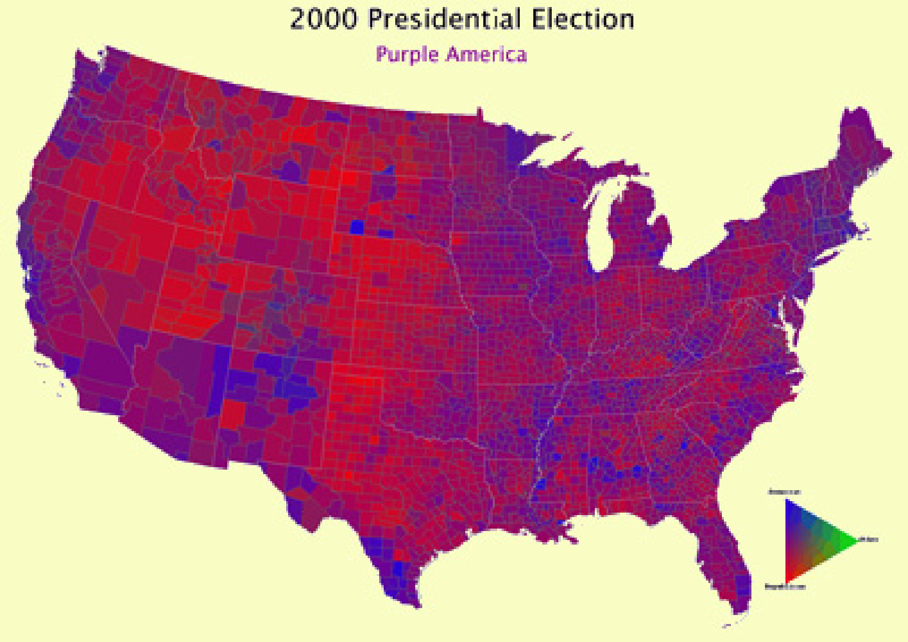
Classification models used

- SVM
- LDA
- QDA
- KNN (k=30)
- Adaboost

10-fold CV used everywhere.

Classification results

Method	Accuracy	Kappa
SVM	0.8389568	0.5554758
LDA	0.8345383	0.5330159
QDA	0.8125396	0.4705189
KNN	0.7475475	0.1237399
Adaboost	0.8584849	0.6061885


2000 Presidential Election
Purple America

Regression models used

- Lasso
- Random forest
- Logit regression (scrapped)
- SVM
- Bagged decision trees
- Stochastic gradient boosted trees
- XGBoost linear
- XGBoost trees

Stochastic Gradient Boosting *(Friedman)*: Train sequence of trees, as with ordinary gradient boosting tree algorithm. However, only L terminal nodes per tree, and each tree is trained only on a randomly selected subsample of training data. Mitigates overfitting, generally performs better than random forest. Package `gbm` in R.

Stochastic Gradient Boosting *(Friedman)*: Train sequence of trees, as with ordinary gradient boosting tree algorithm. However, only L terminal nodes per tree, and each tree is trained only on a randomly selected subsample of training data. Mitigates overfitting, generally performs better than random forest. Package `gbm` in R.

XGBoost *(Chen & Guestrin)*: Regularized stochastic gradient boosting. Distributed, works very well on large datasets, sparse-data-aware, computationally sophisticated. Package `xgboost` in R.

Regression results

Method	Rsquared	RsquaredSD
Lasso	0.6415329	0.05642351
Random Forest	0.7334688	0.03450272
SVM	0.6481942	0.05012503
GBM	0.7721142	0.03053278
Bagged CART	0.5646940	0.02516780
XGBLinear	0.7405678	0.02932354
XGBTree	0.7478437	0.02707445

Variable Importance

From the caret documentation at

http://topepo.github.io/caret/variable-importance.html

Linear Models: the absolute value of the t-statistic for each model parameter is used.

Variable Importance

From the `caret` documentation at

`http://topepo.github.io/caret/variable-importance.html`

Linear Models: the absolute value of the t-statistic for each model parameter is used.

Random Forest: from the R package: For each tree, the prediction accuracy on the out-of-bag portion of the data is recorded. Then the same is done after permuting each predictor variable. The difference between the two accuracies are then averaged over all trees, and normalized by the standard error. For regression, the MSE is computed on the out-of-bag data for each tree, and then the same computed after permuting a variable. The differences are averaged and normalized by the standard error. If the standard error is equal to 0 for a variable, the division is not done.

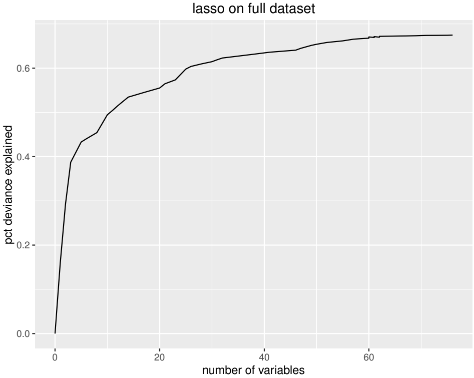
Recursive Partitioning: The reduction in the loss function (e.g. mean squared error) attributed to each variable at each split is tabulated and the sum is returned. Also, since there may be candidate variables that are important but are not used in a split, the top competing variables are also tabulated at each split. This can be turned off using the maxcompete argument in rpart.control. This method does not currently provide class-specific measures of importance when the response is a factor.

Recursive Partitioning: The reduction in the loss function (e.g. mean squared error) attributed to each variable at each split is tabulated and the sum is returned. Also, since there may be candidate variables that are important but are not used in a split, the top competing variables are also tabulated at each split. This can be turned off using the maxcompete argument in rpart.control. This method does not currently provide class-specific measures of importance when the response is a factor.
Bagged Trees: The same methodology as a single tree is applied to all bootstrapped trees and the total importance is returned.
Boosted Trees: This method uses the same approach as a single tree, but sums the importances over each boosting iteration (see the `gbm` package vignette).

Variable Importance

Lasso, $\lambda = 0.110$

variable	scaled $\|t\|$	coefficient
married	100.00000	3.982
white1nh	90.73494	3.597
latitude	79.23087	-3.136
vehperhouse	58.08542	2.311
voterparticip	42.87387	-1.705
longitude	42.33608	1.668
nohealthinsurance	38.40696	1.515
lnpoppersqm	35.62778	-1.465
mediangrossrent	35.12867	-1.369
hisppop	30.43560	-1.151
samehouse	29.85393	-1.190
age45_54	27.37211	-1.091
unemployedclf	26.20114	-1.045
gwagegap	25.28031	0.995
other1	24.25983	0.931
manprofoccs	23.78744	0.917
multrace	22.90927	-0.908
european	22.22989	-0.888
grad	22.13447	-0.823
medianhvalue	21.67482	-0.861



Variable Importance

Random Forest

married	100.00000
latitude	89.89273
white1nh	74.55183
lnpoppersqm	67.49755
longitude	65.79855
voterparticip	64.61488
european	46.66031
irrigationwater	45.50158
vehperhouse	44.30524
black1	43.87877

Bagged CART

married	100.00000
vehperhouse	89.18799
white1nh	75.76427
lnpoppersqm	67.69912
mobilehomes	50.83137
irrigationwater	46.27078
farmfishoccs	39.91635
nevermarried	38.41166
european	33.18302
voterparticip	27.54177

XGBoost Linear

married	100.000000
lnpoppersqm	36.208238
white1nh	25.779966
vehperhouse	20.182694
european	15.525901
longitude	15.002006
latitude	14.585078
domesticwaterpercap	9.377486
black1	5.702451
voterparticip	5.146734

SVM

married	100.00000
vehperhouse	79.80770
nevermarried	69.66694
lnpoppersqm	63.34632
lnpop	50.25822
white1nh	45.73674
publicindus	40.53753
onfarms	37.97240
unemployedclf	35.03867
black1	31.25504

Stochastic Gradient Boosting

married	100.000000
white1nh	30.195746
lnpoppersqm	29.143112
latitude	21.147230
longitude	20.295298
vehperhouse	16.882951
european	12.616580
voterparticip	8.606195
irrigationwater	8.403941
mobilehomes	7.442040

XGBoost Trees

married	100.000000
vehperhouse	28.415682
lnpoppersqm	19.172165
white1nh	16.947449
longitude	14.699382
european	11.990038
latitude	9.642352
Inland	8.979136
nevermarried	7.947897
black1	7.568393

Importance Score, R^2 scaled to [0,1]

Sum of scores weighted by R^2

married	431.79529
white1nh	194.74779
lnpoppersqm	193.42342
vehperhouse	171.51529
latitude	148.61230
longitude	129.50990
european	105.01412
nevermarried	98.29973
voterparticip	92.84896
black1	77.29452
domesticwaterpercap	70.24794
unemployedclf	69.44665
farmfishoccs	64.97608
irrigationwater	59.62696
nohealthinsurance	58.69887
lnpop	55.16021
medianhvalue	54.22515
onfarms	54.10720
pci	52.30743
publicindus	50.06372

Regression on vote count

Obviously, `lnpop` is the most significant predictor. More subtle issue: linear regressions are not usually appropriate for count variables. Three of the not-very-successful approaches:

1. Use Poisson regression ($R^2 = 0.5367002$)
2. Use previous tree-based regression methods on the quantity `rvotes` (huge test error rates; near-meaningless R^2 due to extremely high correlation with `lnpop`).
3. Use previous regression methods on the quantity `log(rvotes)` (lower test error, but still unreliable R^2, especially now since the response is the log of what we're interested in, and still highly correlated with `lnpop`).

Variable importance scores for regressions on `log(rvotes)`

lnpop	516.09110
lnpoppersqm	237.87708
urban	138.20016
nocashrent	107.73767
farmfishoccs	83.98845
salesoffoccs	73.03108
medianhvalue	69.76324
mediangrossrent	65.07785
onfarms	61.91551
asian	51.81594
pci	51.24471
white1nh	45.01021
european	42.83201
housingincomediscrep	42.46306
gwagegap	39.86286
poor	39.44514
banksper1000pop	38.55845
grad	36.50980
latitude	34.96502
voterparticip	30.96742

Questions?