Machine Learning Workshop

Iasmin Goes & Daniel Weitzel

Department of Political Science

Colorado State University

22-23 October 2024

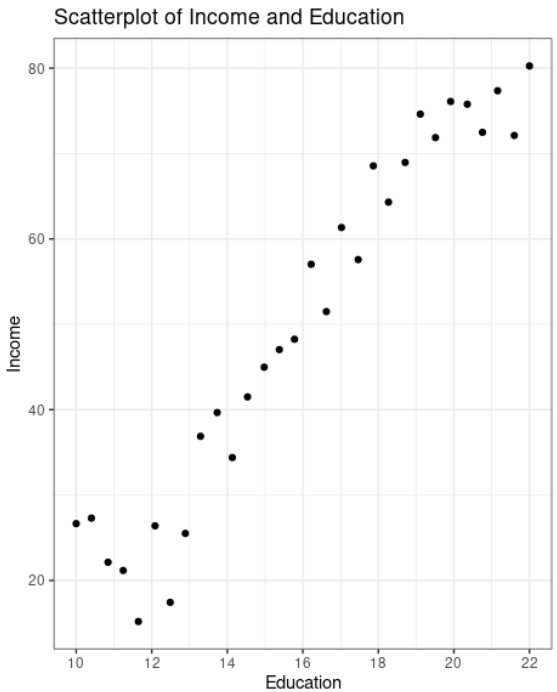
Slides

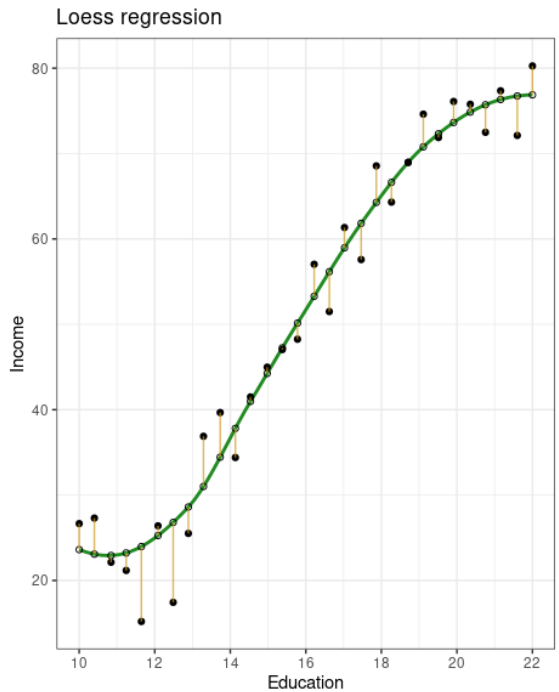
https://t.ly/x9-el

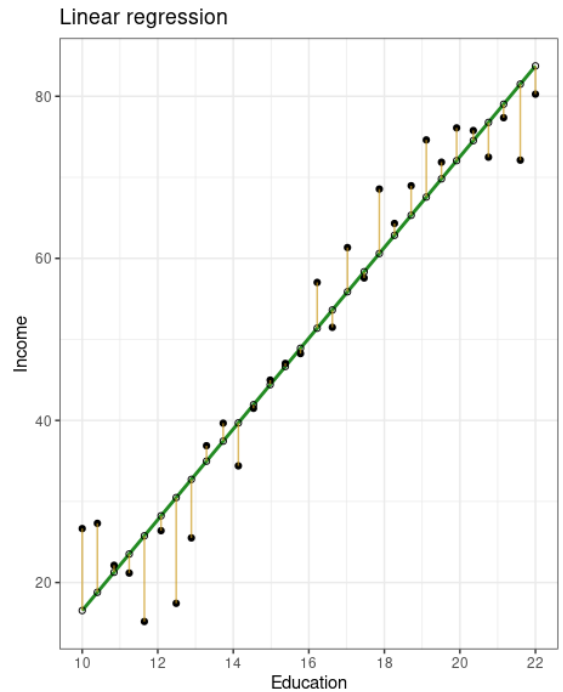
Why we analyze data

- **Goal:** understand the *systematic* relationship between an outcome (dependent variable) and predictors (independent variables)
- We want to know the function f in $Y = f(X) + \epsilon$
  - f is the unknown form with which X provides systematic information about Y

Why we analyze data







Difference between "traditional" statistics and ML

- Goal
  - **TS:** test hypotheses, make inferences about population parameters
  - **ML:** build predictive models for new data
- Approach
  - **TS:** pre-defined assumptions about the population distribution; these assumptions, in turn, inform model selection
  - **ML:** no assumptions about distribution or functional form; models learn patterns and relationships from the data
- Data
  - **TS:** good for smaller datasets that are a representative sample of a larger population
  - **ML:** good for larger and more complex data sets, also unstructured data (images, text, audio)

Difference between "traditional" statistics and ML

- Interpretability
  - **TS:** a set of pre-defined assumptions allows for higher interpretability; results are easier to understand
  - **ML:** can be "black boxes", models can be difficult to interpret
- Training
  - **TS:** usually no split between training and test data: the analysis is done on one full dataset
  - **ML:** the dataset is split into training, (cross-)validation, and test set to prevent overfitting of models to idiosyncratic features of the data

When ML makes sense

- Large data sets (number of observations)
- Large number of predictors and/or no theory about f
- Accurate predictions are more valuable than causal inference
- Complex non-linearity in the data
- Unstructured data
- Goal is feature generation (making new variables)

Types of ML

- **Supervised learning:** the computer is trained on a labeled dataset and learns to make predictions or classifications based on new data
- **Unsupervised learning:** the computer is given an unlabeled dataset and is tasked with finding patterns or relationships within the data. The model might also generate its own labels (self-supervised learning)
- **Semi-supervised learning:** the computer is given a dataset with labeled *and* unlabeled examples and uses the unlabeled data to improve its performance
- **Reinforcement learning:** the computer learns to make decisions based on a reward signal, which is provided when it performs an action that leads to a positive outcome

Example 1: supervised learning

This paper uses random forests to predict the "critical mass" of women legislators needed to affect health, education, and defense spending

Political Science Research and Methods (2021), page 1 of 19
doi:10.1017/psrm.2021.51

PS RM

ORIGINAL ARTICLE

Point break: using machine learning to uncover a critical mass in women's representation†

Kendall D. Funk[1], Hannah L. Paul[2]* and Andrew Q. Philips[2]

[1]School of Social and Behavioral Sciences, Arizona State University, Tempe, AZ, USA and [2]Department of Political Science, University of Colorado Boulder, Boulder, CO, USA
*Corresponding author. Email: hannah.paul@colorado.edu

(Received 6 February 2020; revised 9 June 2021; accepted 20 June 2021)

Abstract
Decades of research has debated whether women first need to reach a "critical mass" in the legislature before they can effectively influence legislative outcomes. This study contributes to the debate using supervised tree-based machine learning to study the relationship between increasing variation in women's legislative representation and the allocation of government expenditures in three policy areas: education, healthcare, and defense. We find that women's representation predicts spending in all three areas. We also find evidence of critical mass effects as the relationships between women's representation and government spending are nonlinear. However, beyond critical mass, our research points to a potential critical mass interval or critical limit point in women's representation. We offer guidance on how these results can inform future research using standard parametric models.

Example 1: supervised learning

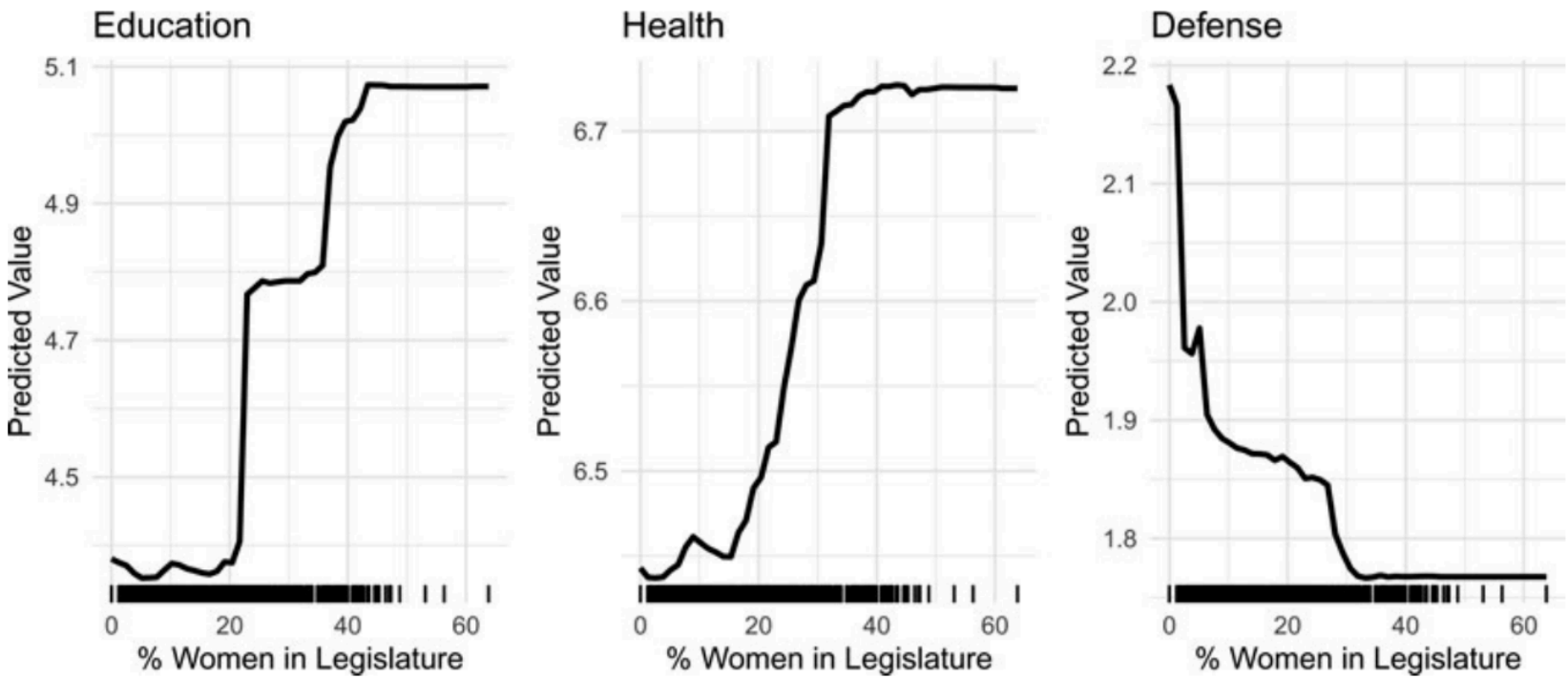
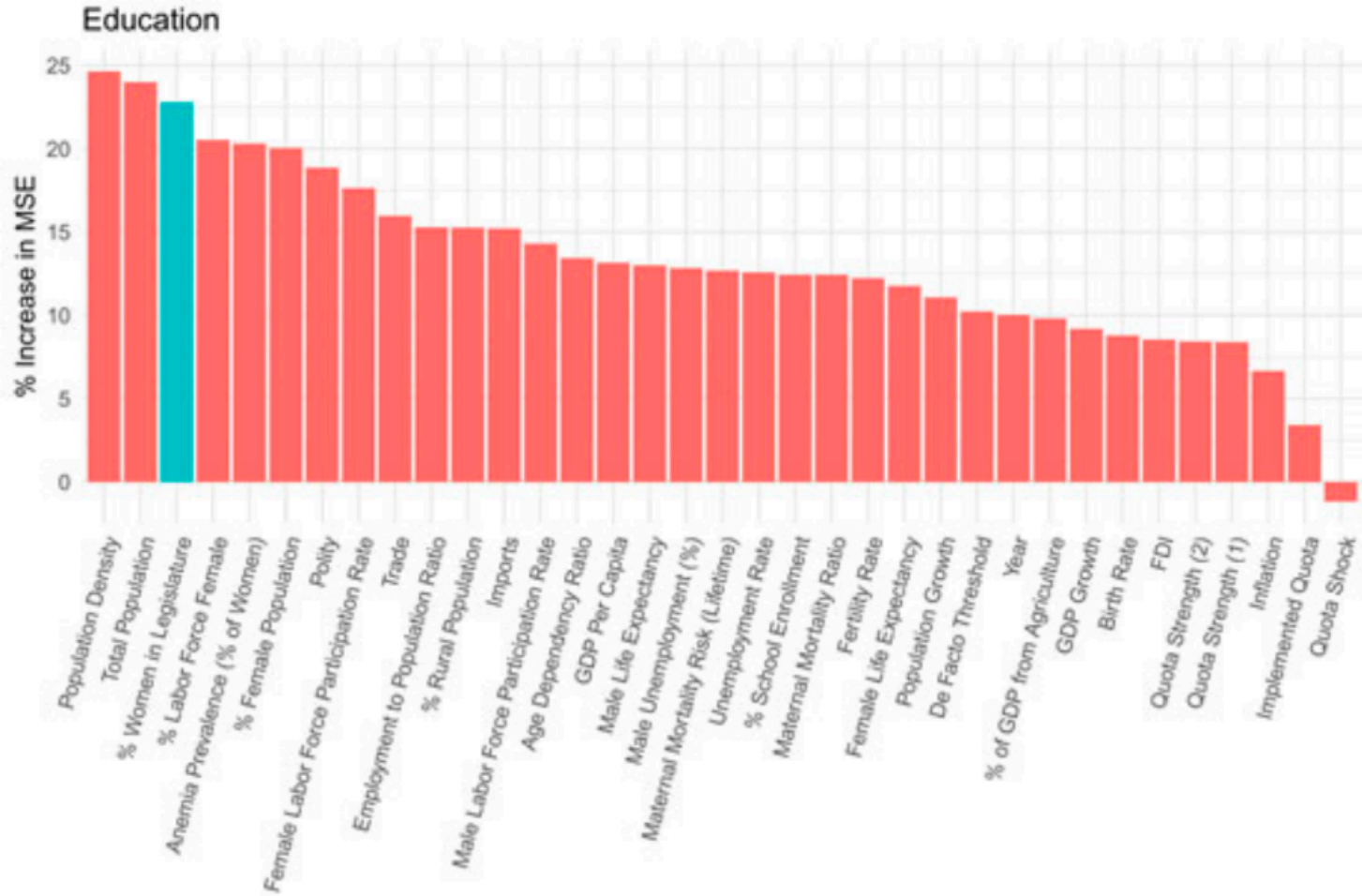


Fig. 4. PDPs of women's representation support critical mass.

Example 1: supervised learning



Example 2: unsupervised learning

This paper uses topic models to classify the content of reports:

Political Science Research and Methods Vol 6, No. 4, 661–677 October 2018

doi:10.1017/psrm.2016.44

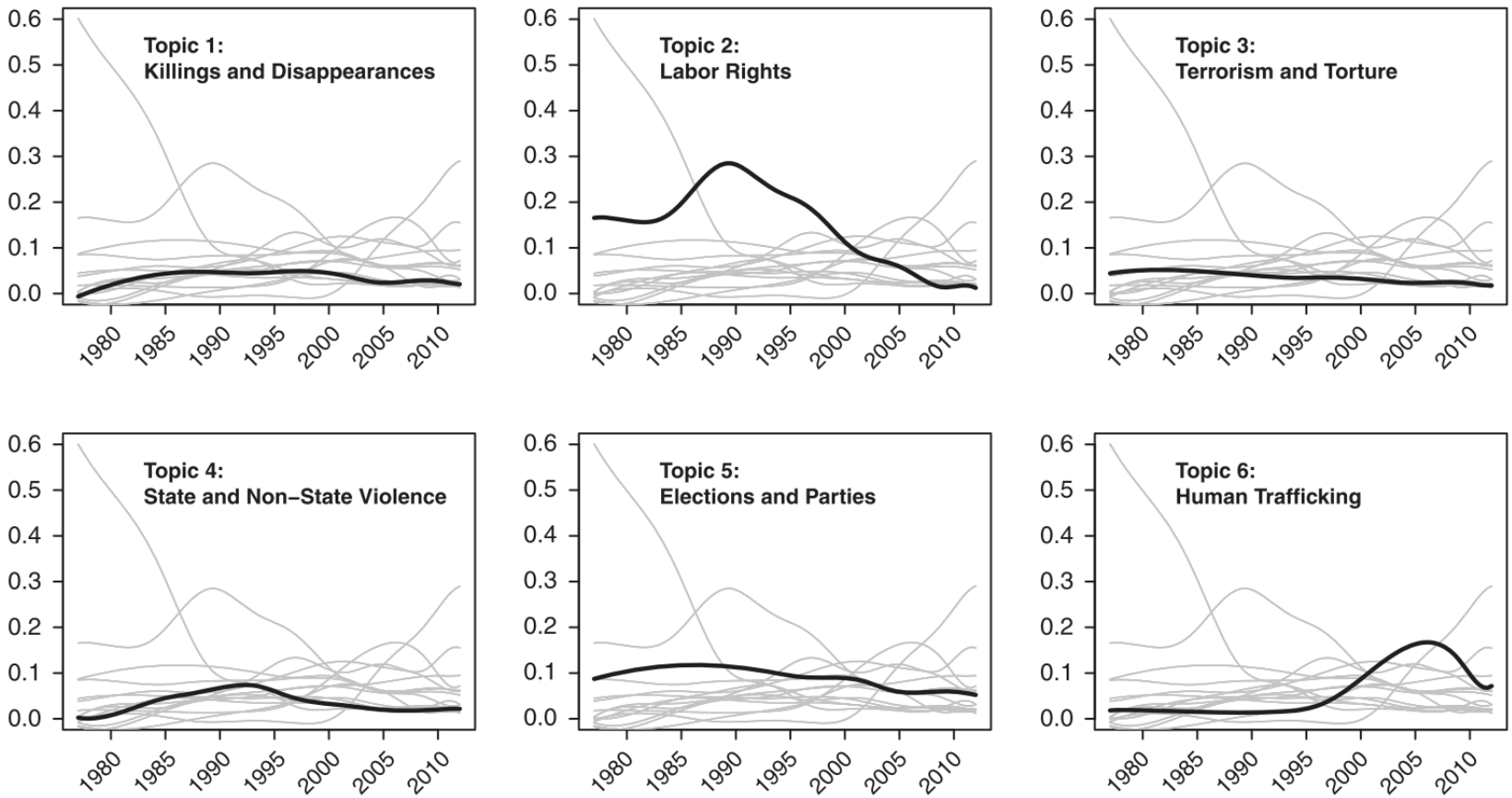
The Politics of Scrutiny in Human Rights Monitoring: Evidence from Structural Topic Models of US State Department Human Rights Reports*

BENJAMIN E. BAGOZZI AND DANIEL BERLINER

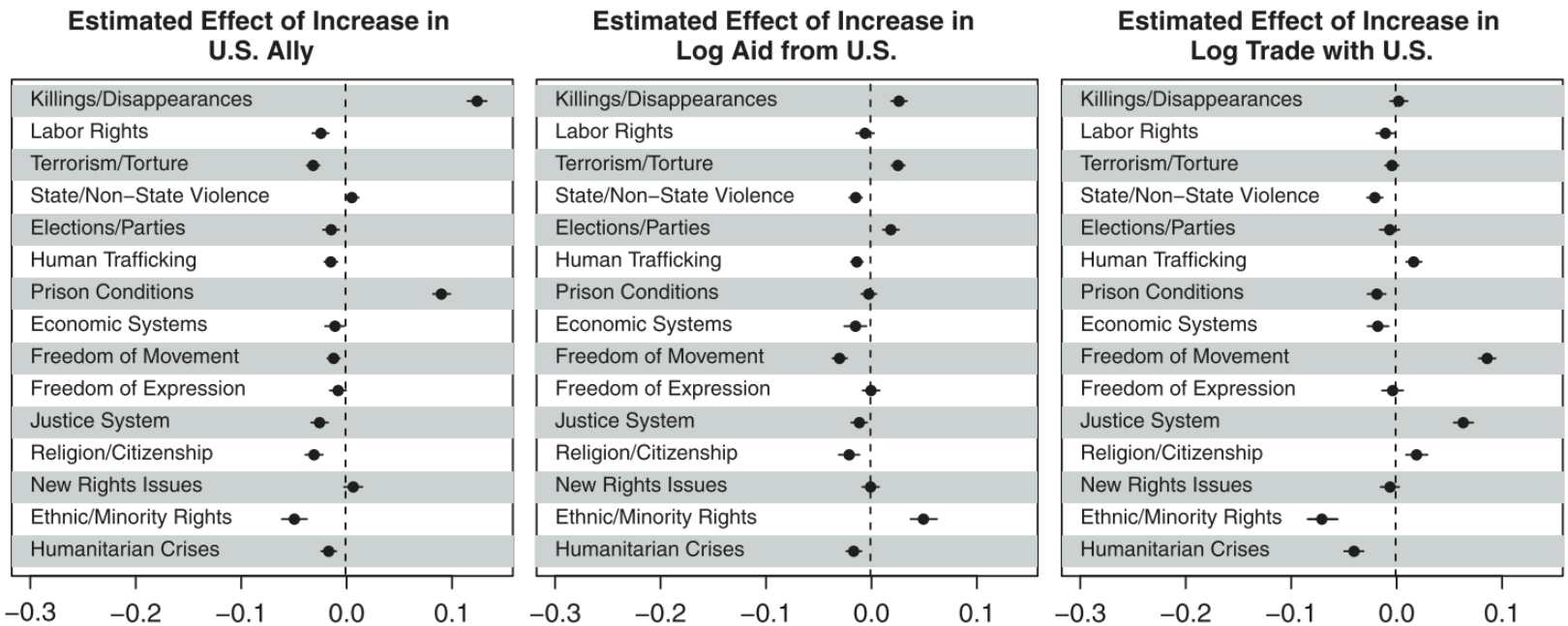
Human rights monitoring reports play important roles both in the international human rights regime and in productions of human rights data. However, human rights reports are produced by organizations subject to formal and informal pressures that may influence the topics considered salient for attention and scrutiny. We study this potential using structural topic models (STMs), a method used for identifying the latent topical dimensions of texts and assessing the effects of covariates on these dimensions. We apply STMs to a corpus of 6298 State Department Country Reports on Human Rights Practices (1977–2012), identifying a plausible set of topics including killings and disappearances, freedoms of expression and movement, and labor rights, among others. We find that these topics vary markedly both over time and space. We also find that while US domestic politics play no systematic role in shaping topic prevalence, US allies tend to receive more *attention to violations of physical integrity rights. These results challenge extant research, and illustrate the usefulness of STM methods for future study of foreign policy documents. Our findings also highlight the importance of* topical attention shifts *in documents that monitor and evaluate countries.*

Example 2: unsupervised learning

It is an unsupervised method because the authors use an unlabeled dataset. They let the model create labels of its own



Example 2: unsupervised learning



Example 3: semi-supervised learning

This paper uses neural networks to classify images and identify election fraud:

American Political Science Review (2019) 113, 3, 710–726
doi:10.1017/S0003055419000285

The Fingerprints of Fraud: Evidence from Mexico's 1988 Presidential Election

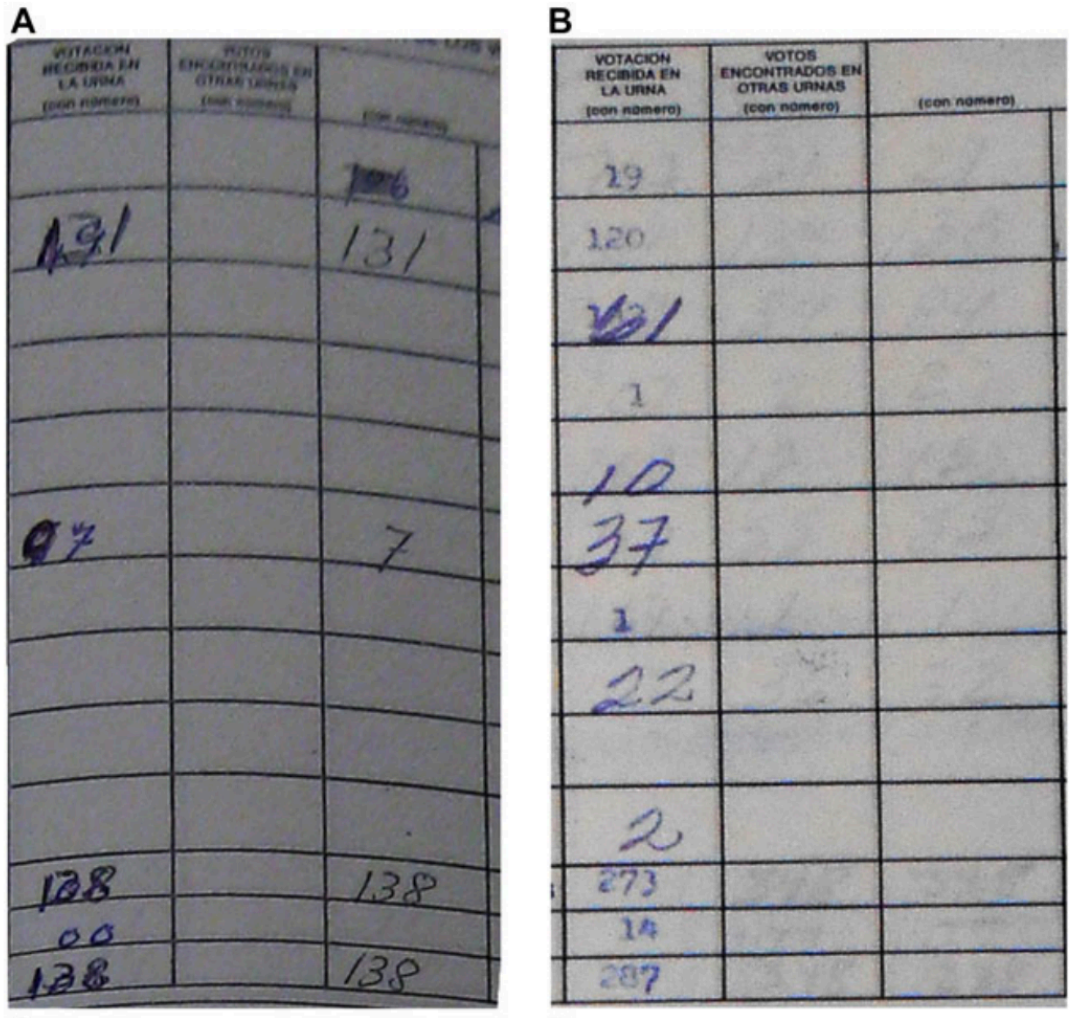
FRANCISCO CANTÚ *University of Houston*

This paper investigates the opportunities for non-democratic regimes to rely on fraud by documenting the alteration of vote tallies during the 1988 presidential election in Mexico. In particular, I study how the alteration of vote returns came after an electoral reform that centralized the vote-counting process. Using an original image database of the vote-tally sheets for that election and applying Convolutional Neural Networks (CNN) to analyze the sheets, I find evidence of blatant alterations in about a third of the tallies in the country. This empirical analysis shows that altered tallies were more prevalent in polling stations where the opposition was not present and in states controlled by governors with grassroots experience of managing the electoral operation. This research has implications for understanding the ways in which autocrats control elections as well as for introducing a new methodology to audit the integrity of vote tallies.

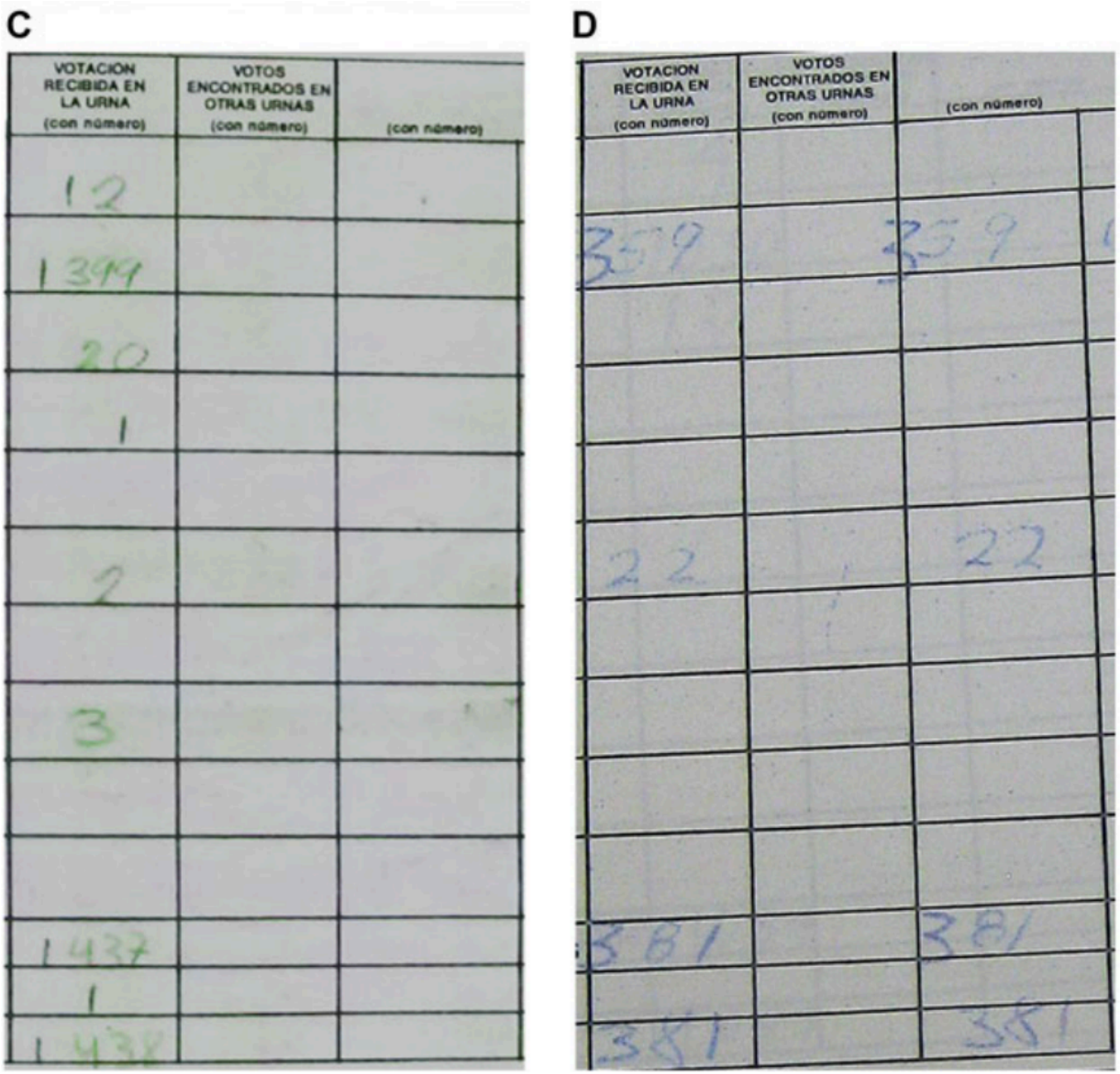
The author manually labeled 1,050 images as "with alterations" or "without alterations," then asked the model to make predictions for ~52,300 unlabeled images

Example 3: semi-supervised learning

FIGURE 1. Examples of Vote Tallies with Alteration in Their Numbers. Mexico, 1988

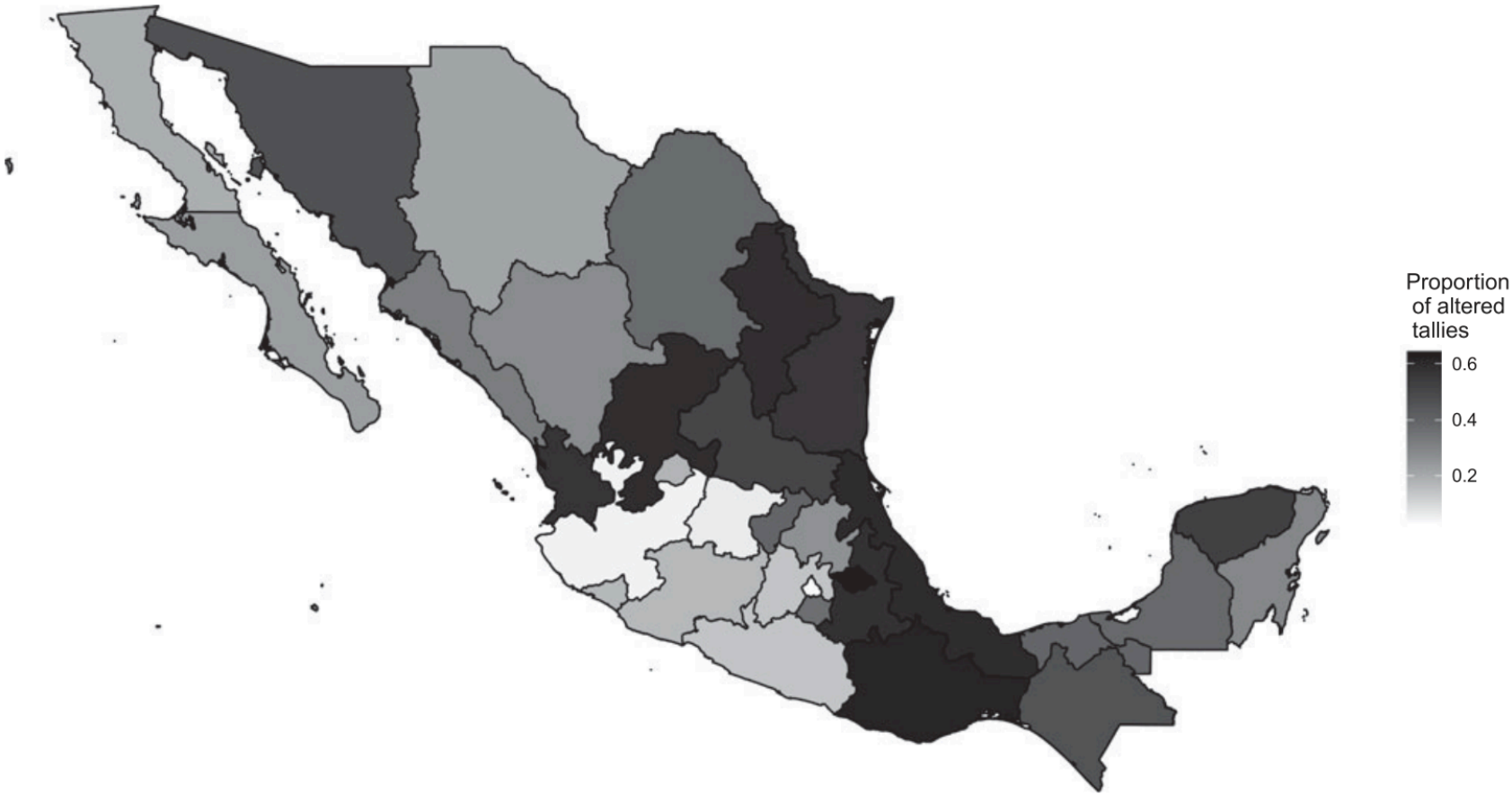


Example 3: semi-supervised learning



Example 3: semi-supervised learning

FIGURE 3. Rates of Tallies Classified as Altered by State



Notes: This figure shows the proportion of tallies in every state classified by the CNN as altered.

Example 4: semi-supervised learning

This paper uses keyword-assisted topic models to classify IMF conditionality

DOI: 10.1111/ecpo.12214

ORIGINAL ARTICLE

ECONOMICS & POLITICS WILEY

Examining the effect of IMF conditionality on natural resource policy

Iasmin Goes

The author provides some labels and keywords, but allows the model to find additional words and labels

Example 4: semi-supervised learning

Word
prices
percent
oil
petroleum
price
increase
products
gas
electricity
tariffs

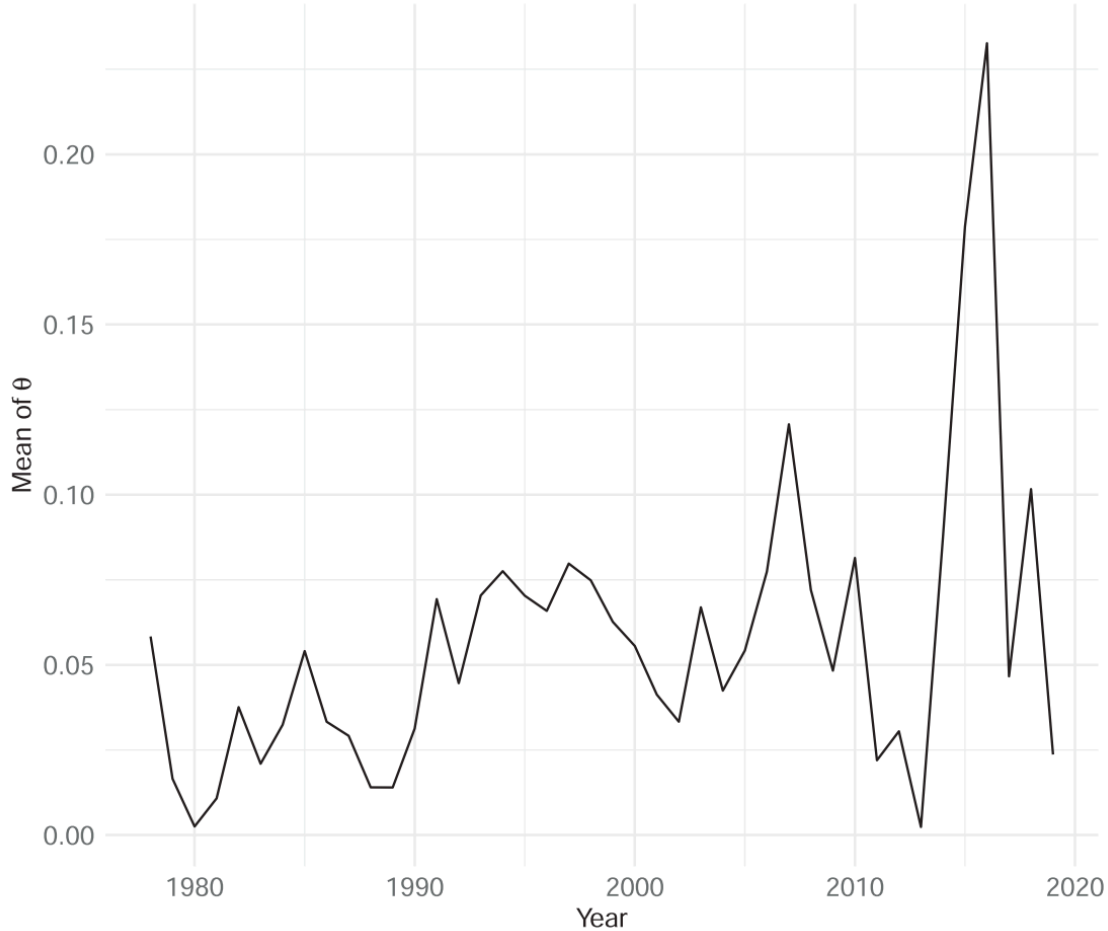
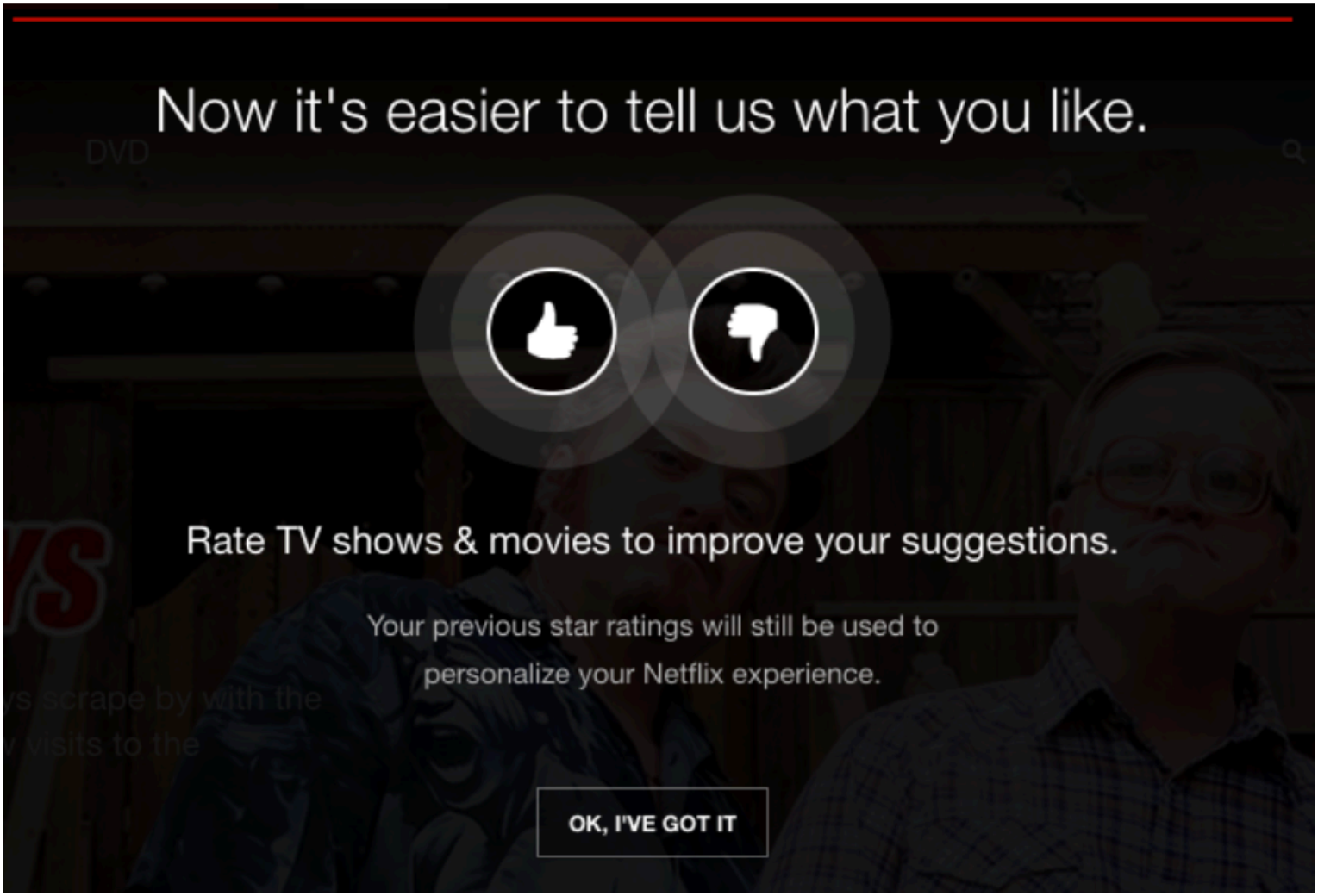


FIGURE 4 Topic prevalence over time, 1980–2019. This plot displays the prevalence of the natural resource topic over time, among all binding conditions, based on the year of program initiation (as indicated by the *x*-axis). The *y*-axis represents the relative proportion θ of this topic in each condition, averaged for all conditions over a year

Example 5: reinforcement learning

We couldn't think of any academic example. But Netflix does this



Supervised learning

Supervised tasks

- **Classification (categorical outcome)**
  - Naive Bayes
  - Clustering algorithms (kNN)
  - Logistic regression
  - Random forest
  - Gradient boosting machine
  - Support vector machine
  - Neural networks
- **Regression (continuous outcome)**
  - Lasso, ridge, and linear regression
  - Random forest
  - Gradient boosting machine
  - Support vector machine
  - Neural networks

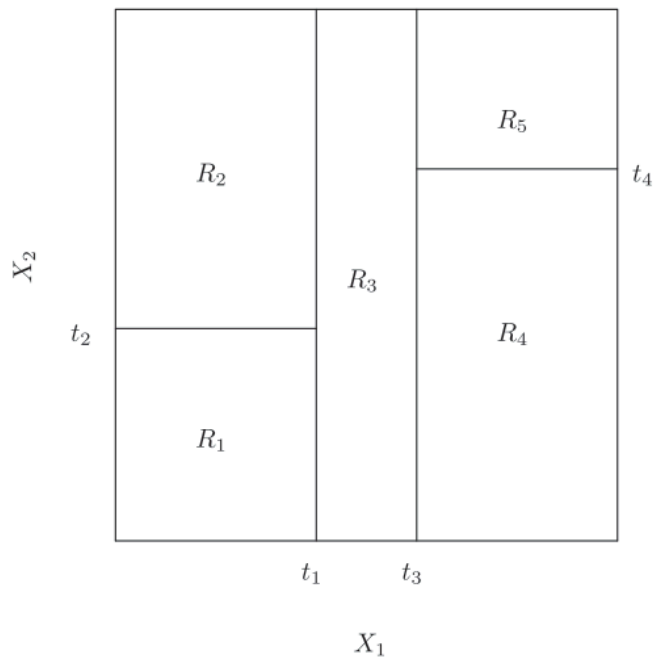
Supervised learning

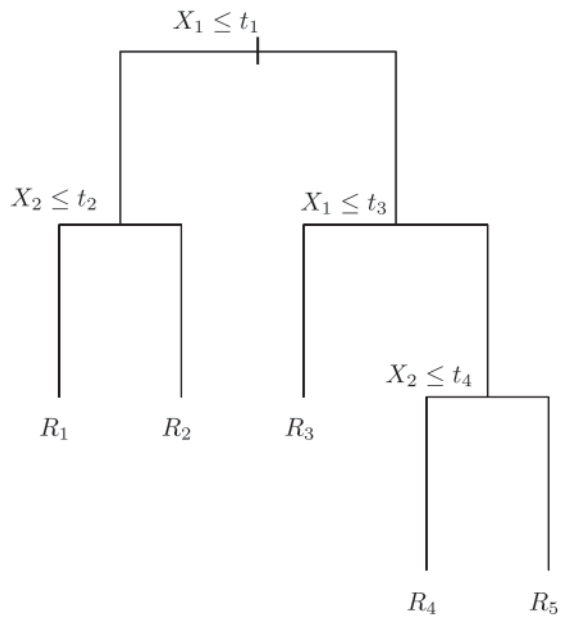
- Need to have a **labeled dataset**
- Split the data into training/test set (and ideally also validation set)
  - **Training set** (60-80% of the data): fit the model, let it learn patterns from the data
  - **Validation set** (10-20%, if you have enough observations): tune hyperparameters and evaluate model performance
  - **Test set** (10-20%): once the model is trained and tuned, make out-of-sample predictions for independent observations

In more detail: tree-based models

A decision tree

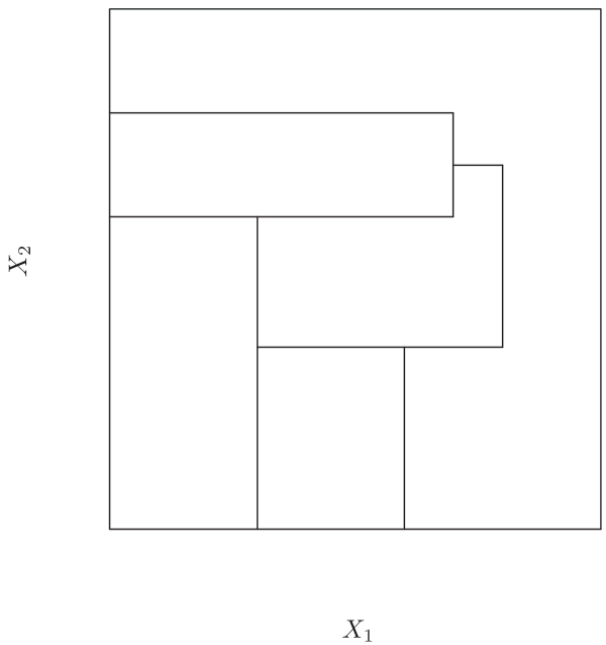
- **Goal:** split covariate space into regions, with each region corresponding to a unique covariate combination
  - The model then makes one prediction for all observations within this region

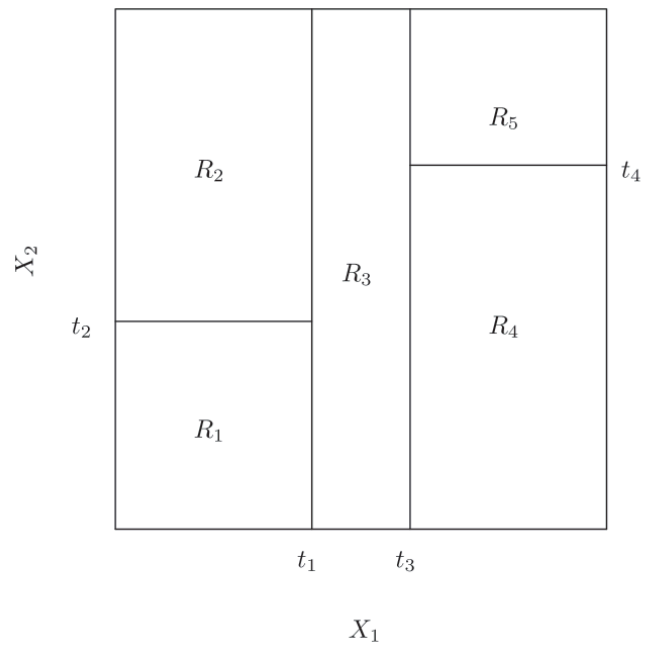




In theory, regions could have any shape

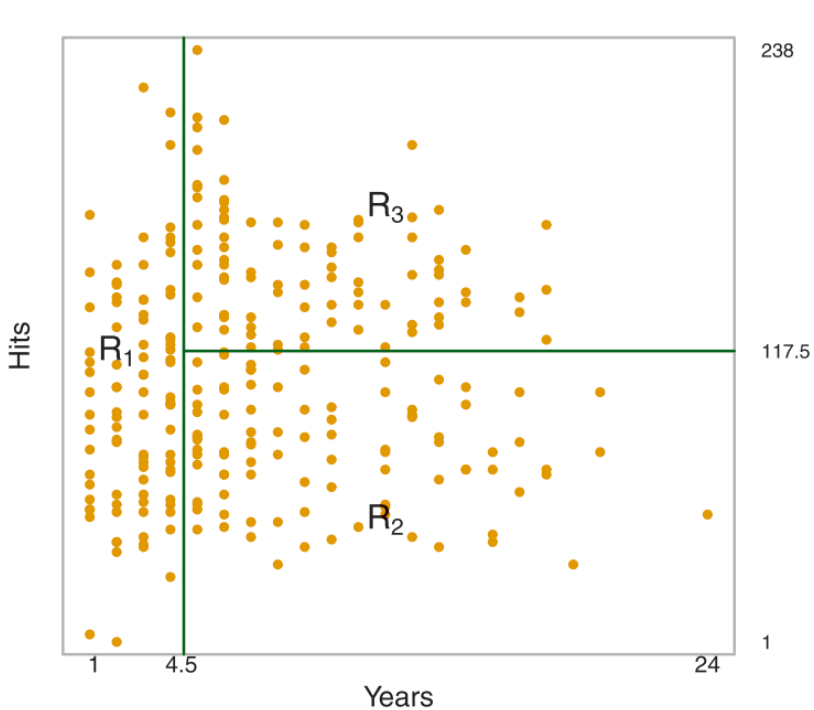
But it is computationally infeasible to consider every possible partition (left), which is why the model works with high-dimensional rectangles (right). This is called *recursive binary splitting*

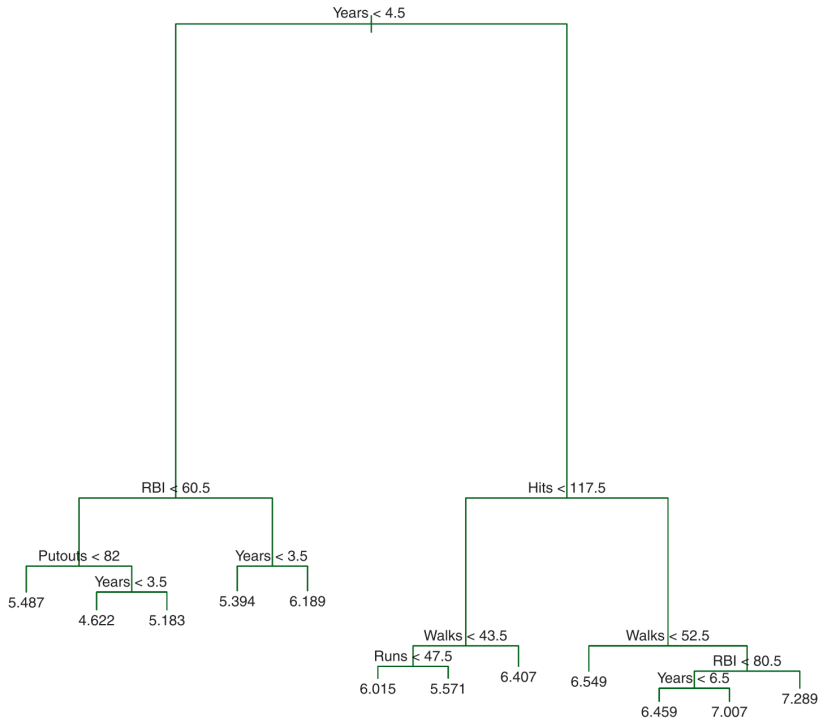




For example, the salary of baseball players

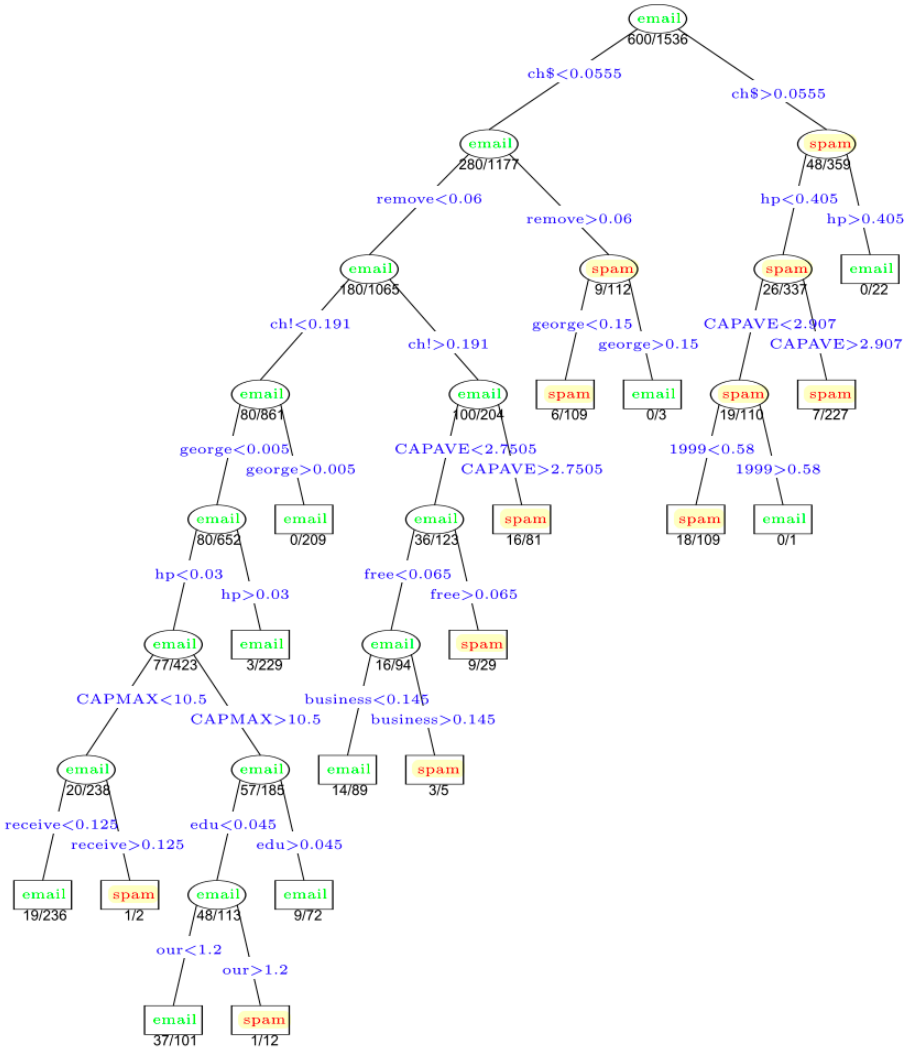
This is a *regression* tree: the outcome is quantitative





Another example: legit emails vs. spam

This is a *classification* tree: the outcome is categorical/qualitative



Multiple models are better than one model

- **Ensemble methods** combine the predictions of multiple models to create a stronger model
  - One model can make errors. Multiple models will make different errors that balance each other out
- Two common types of ensemble methods: bagging ad boosting
  - **Bagging:** train each model on a different random subset (with replacement), then aggregate them
  - **Boosting:** train models sequentially, with one model correcting the errors of the previous one

An example of bagging: random forest

- **Why forest?** Because one tree can be sensitive to data changes
- **Why random?** Because each binary split only considers a random sample of predictors
  - If there is a very strong predictor in the dataset, we don't want *all* trees to use this predictor in the first split
- The model then aggregates the results based on the predictions of most trees

Things you can explain with tree-based models

- Civil war onset (Muchlinski et al 2016)
- Supreme Court rulings (Kaufman, Kraft and Sen 2019)
- Women's legislative representation and the allocation of government expenditures (Funk, Paul and Philips 2022)
- Negative campaigning and voting (Montgomery and Olivella 2016)
- Democracy (Weitzel et al 2024a) and democratic backsliding (Weitzel et al 2024b)
- Variation in GDP data reported across different sources (Goes 2024)

Advantages of tree-based models

- No need to develop theoretical expectations
- No need to make assumptions about...
  - predictor variables
  - functional form of predictors (linear, log, squared)
- More honesty about the lack of causality
  - Regressions are not causal either, but people often interpret them causally

Disadvantages of tree-based models (and ML more broadly)

- Increased computational cost
- Need to have a large dataset
- No causal inference — ML is mostly concerned with making **predictions**
- Generally no confidence intervals or p-values to quantify uncertainty
  - **Exception:** conformal prediction
- Complexity: ML models are black boxes and difficult to interpret
  - Most social scientists likely won't know what you're doing

ML steps

- Data cleaning and preparation
  - Missing values, feature engineering (one-hot encoding, ranging, standardizing)
- Choose a model and train it
  - Select appropriate ML algorithm
  - Split the data into training, (cross-)validation, and testing sets to evaluate the model's performance
  - Train the algorithm on the prepared training set
  - Be aware of *data leakage*
- Evaluate the model
  - Evaluate the model's performance on the (cross-)validation set
  - Fine-tune it as necessary to improve accuracy

An applied example

Predicting democracy scores

- Are democracy scores subjective or objective?
- Idea: Use a democracy score (liberal democracy) as the outcome variable and train it on a model with purely objective indicators
- Predict democracy scores based on this model and compare to observed democracy scores

The code

- Entire code available on Github
- We use the V-Dem package to load V-Dem 13
- Pre-processing steps involve cleaning the data, transforming variables, visualizations of the data

```
## Libraries
devtools::install_github("vdeminstitute/vdemdata",
                         force = TRUE)

library("tidyverse") # For data processing
library("vdemdata")  # The data set we will use
library("h2o")       # the machine learning package
library("randomizr") # for grouped fold assignment
library("naniar")    # Missing data visualization
```

Train set

- We reduce the data set to the id, outcome, and predictor variables
- We also remove all rows in which the outcome (Liberal Democracy) is missing
- We reduce the training set to years before 2012

```
df_vdem_train <-
  df_vdem |>
  ungroup() |>
  dplyr::select(all_of(ids),
                all_of(preds),
                all_of(outcome)) |>
  drop_na(all_of(outcome)) |>
  dplyr::filter(year <= 2011)
```

Cross-validation set

- We add cross-validation folds to the training data
- There are six equally sized folds
- Stratification is based on country
- This allows us to train the model and validate it on data that it has never learned about

```
df_vdem_train$folds <- cluster_ra(
  clusters = df_vdem_train$country_id,
  conditions = c("Fold_1", "Fold_2",
                 "Fold_3", "Fold_4",
                 "Fold_5", "Fold_6"))
```

Test set

- We generate a test set that we use to assess the quality of our model
- This is data that never was part of the training process. The model does not know it
- Our test set are all years after 2011

```
df_vdem_predict <-
  df_vdem |>
  ungroup() |>
  filter(year > 2011) |>
  dplyr::select(all_of(ids),
                all_of(preds),
                all_of(outcome))
```

H2O Config

- We use H2O for the random forest
- Configuration uses n-1 cores and 20GB of RAM
- H20 works in R, Python, and Flow
- Other packages also great!

```
# Initialization
h2o.no_progress()
h2o.init(nthreads=-1, max_mem_size = "20g")
h2o.removeAll()

# H2O data objects
train_h2o  <- as.h2o(df_vdem_train)
test_h2o   <- as.h2o(df_vdem_predict)
```

Estimating the model

```
model_rf <-
  h2o.randomForest(
    model_id = "ld_1",
    x = predictors,
    y = outcome,
    fold_column = "folds",
    training_frame = train_h2o,
    ntrees = 400,
    mtries = 4,
    col_sample_rate_per_tree = 0.8,
    seed = 1904,
    keep_cross_validation_predictions = TRUE)
```

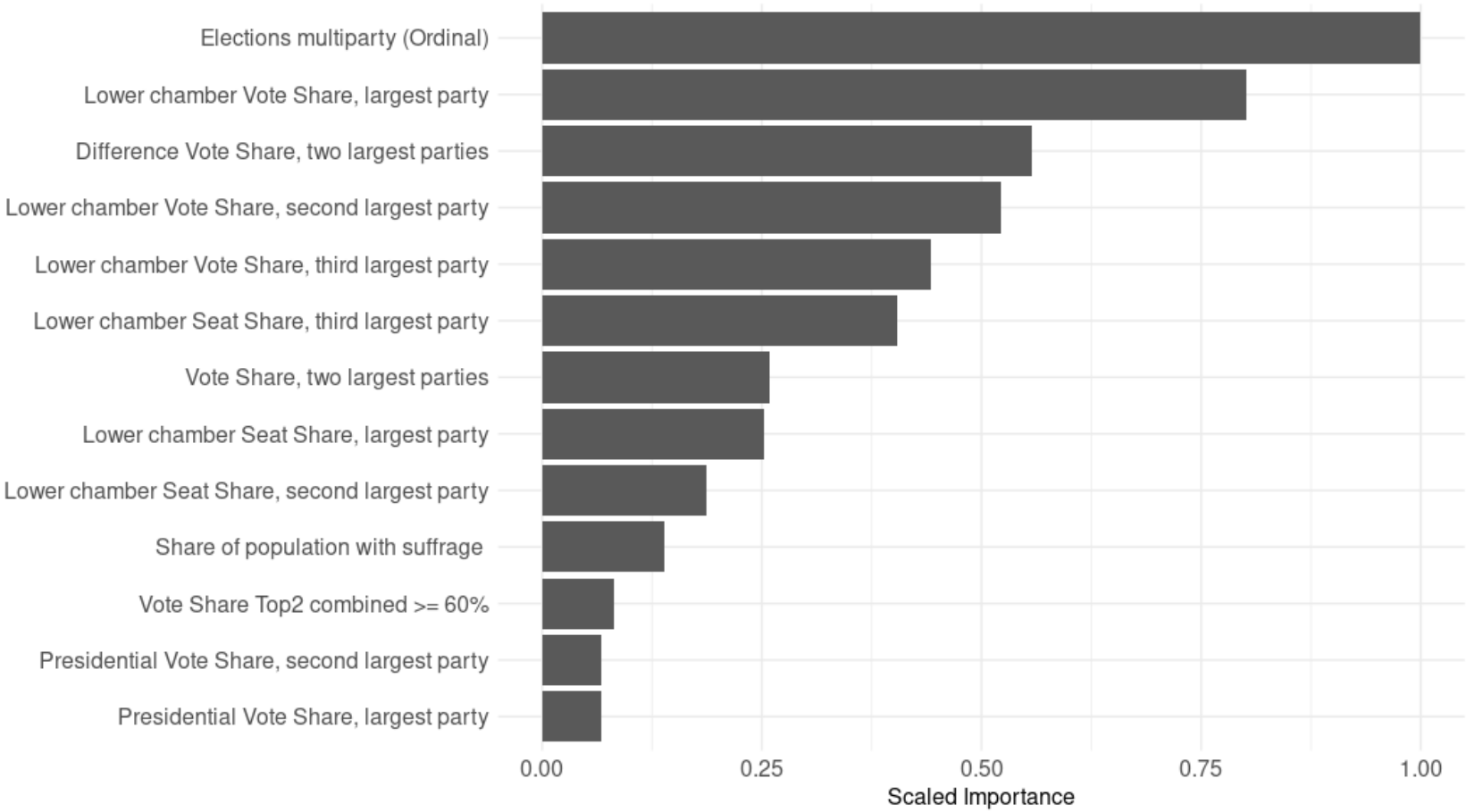
Goodness of fit

- After estimation we can look at the performance metrics in the training and cross-validation data set
- Common are R2 and Mean Squared Error

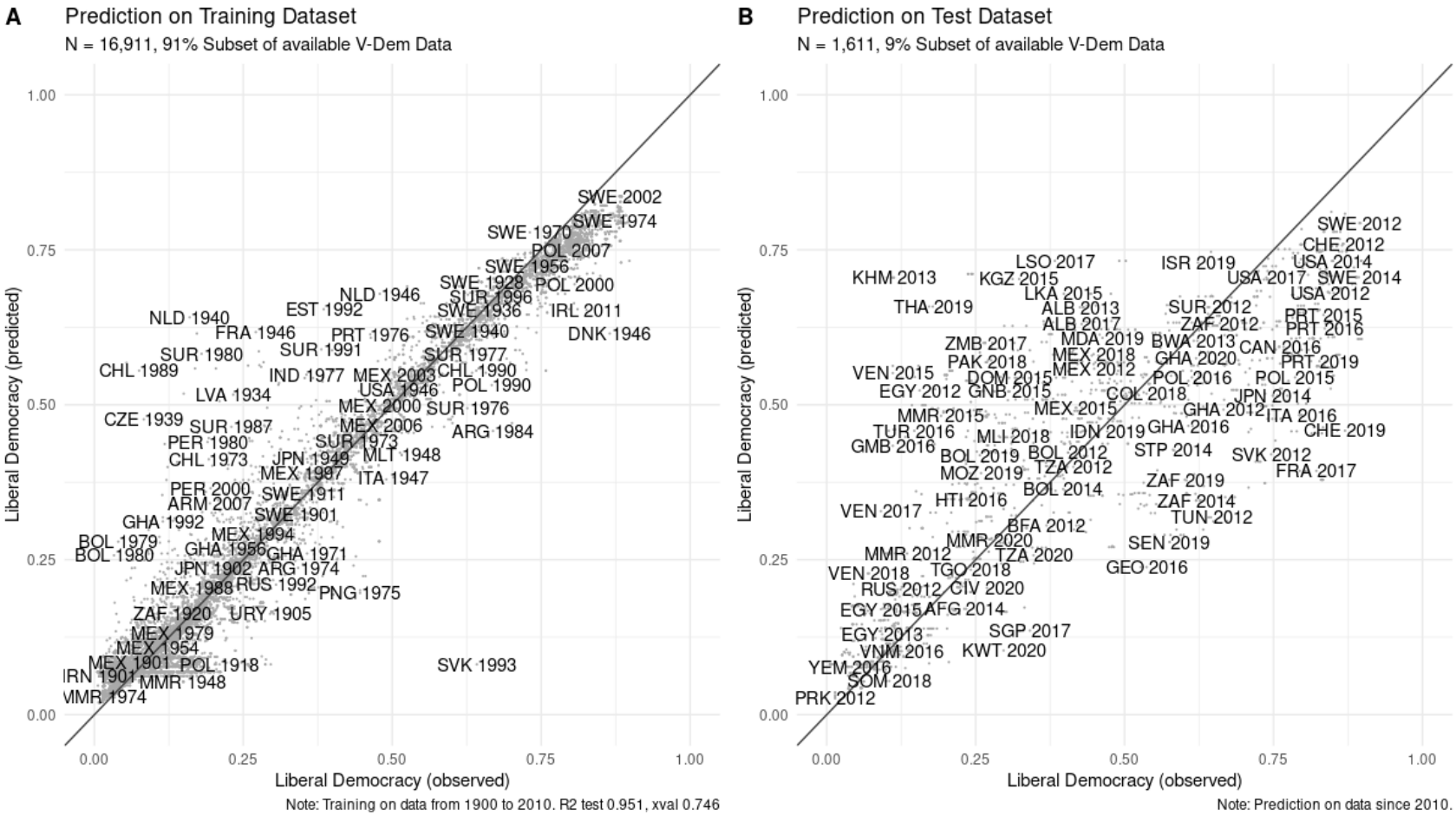
```
> h2o.r2(model_rf_libdem, train = TRUE, xval = TRUE)
    train      xval
0.9539659 0.7451879
> h2o.performance(model_rf_libdem, train = TRUE)
H2ORegressionMetrics: drf
** Reported on training data. **
** Metrics reported on Out-Of-Bag training samples **

MSE:  0.002712985
RMSE:  0.05208633
MAE:  0.03132654
RMSLE:  0.04023765
Mean Residual Deviance :  0.002712985
```

What matters

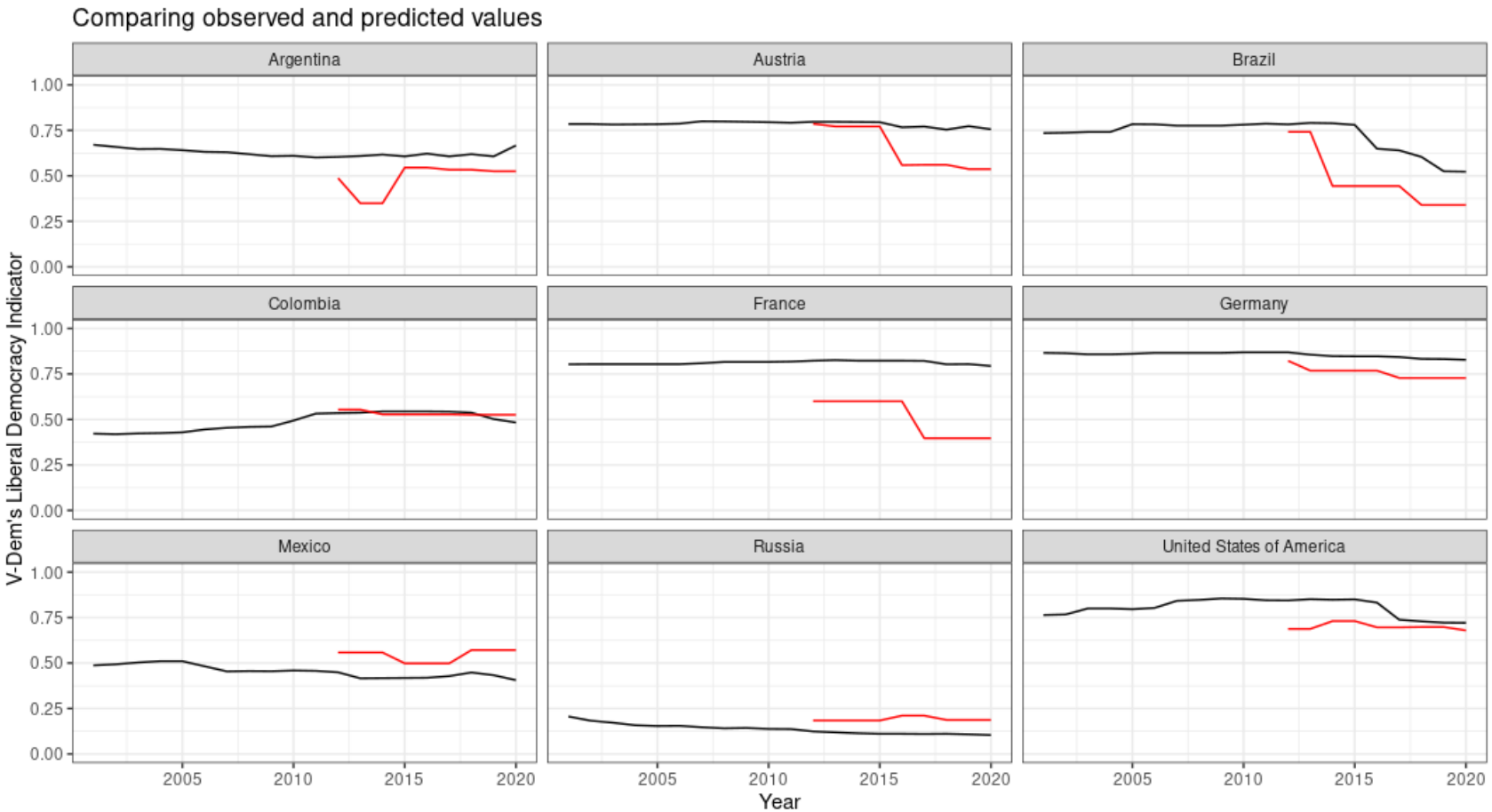
- Variable Importance Plots show us the scaled importance of individual predictors for the random forest



The predictions



The predictions



Why is the performance so poor?

- Statistical reasons:
  - More feature engineering necessary!
  - More model tuning is necessary, we have not adjusted any of the hyperparameters of the random forest
- Theoretical reasons:
  - Our outcome is liberal democracy and all our predictors are indicators of an electoral democracy

Additional Resources

Books, packages, and articles

- Introduction to Statistical Learning, great introduction
- Elements of Statistical Learning, more advanced
- List of R packages for Machine Learning, CRAN Task View List
- Machine Learning Methods That Economists Should Know About, academic article with an overview