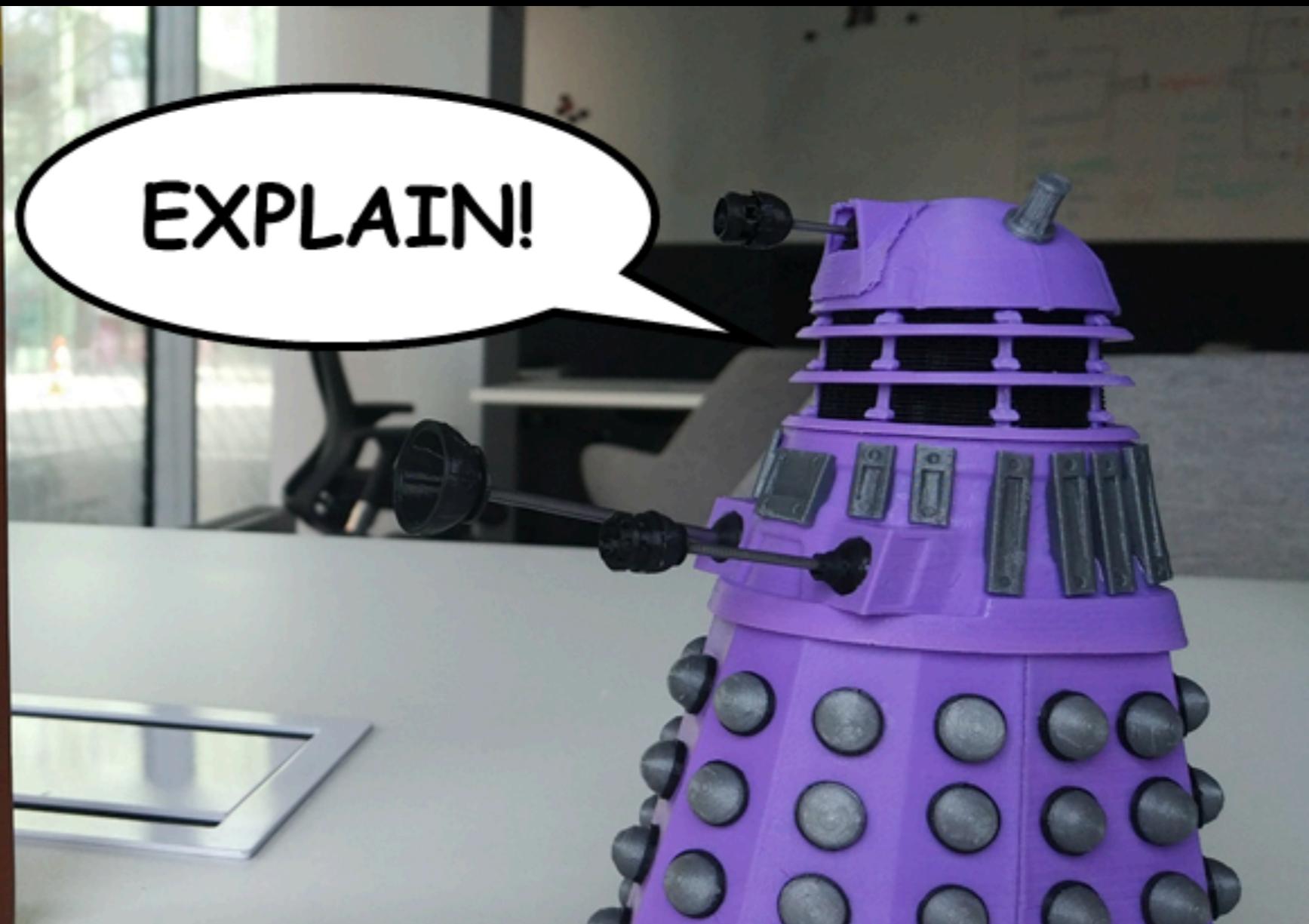
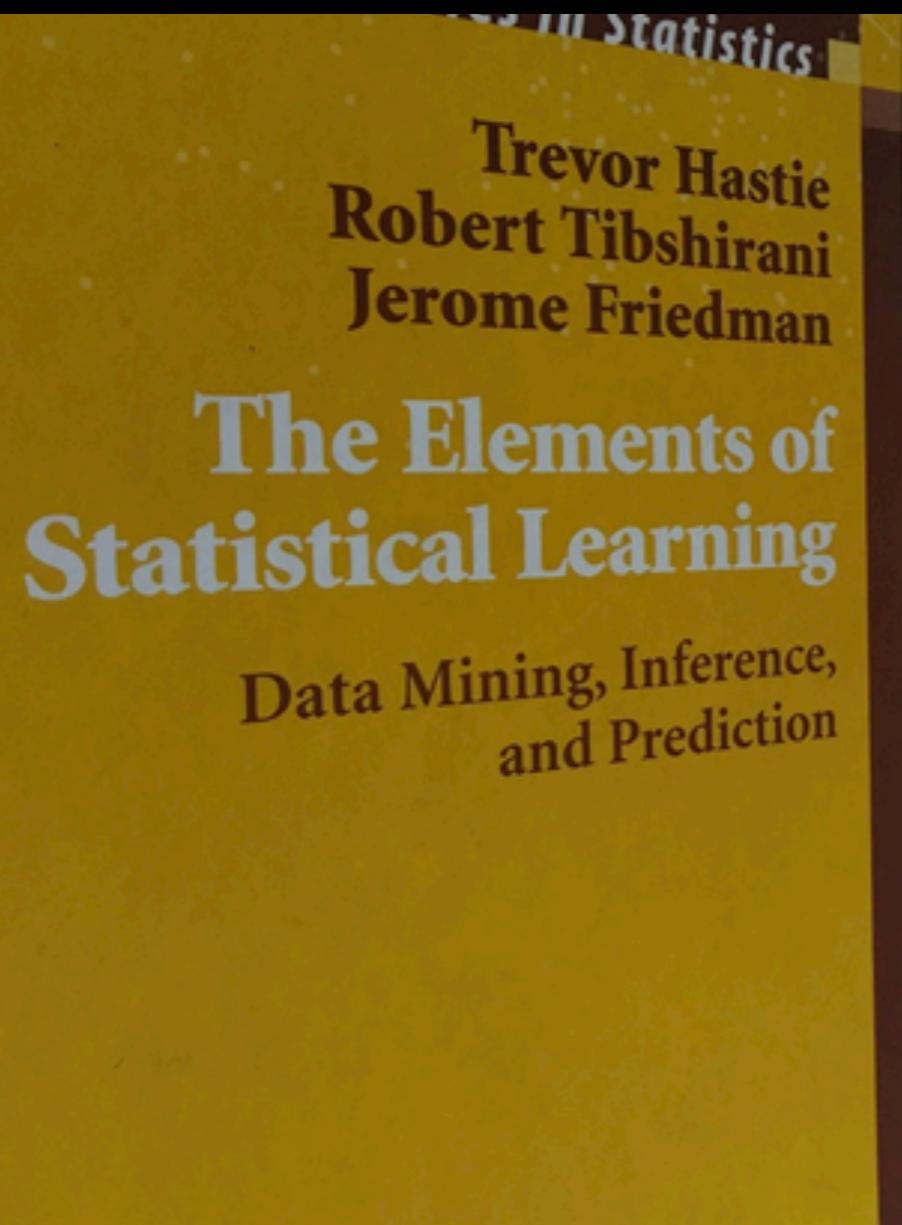


DALEX: Descriptive mAchine Learning EXplanations



Przemysław Biecek

Mateusz Staniak



Associate Professor in Machine
Learning @ Warsaw University of
Technology

15 years of experience in teaching,
using and programming in R

Interested in DataVis, Explanations of
black-boxes and cancer ML modelling

<http://biecek.pl/>

MSc student in Mathematics and
Statistics @ University of Wrocław

Member of MI2 DataLab.

Working on machine learning
interpretability

<http://www.mstaniak.pl/>

Agenda

Introduction

Model explainers - Continuous variable response

Hands-on

Model explainers - Discrete variable response

Hands-on

Model explainers - Model performance

Hands-on

Model explainers - Variable importance

Hands-on

Prediction explainers - Break-down

Hands-on

Prediction explainers - Live

Hands-on

Summary

What you need to install

```
# From CRAN  
# first part  
install.packages("DALEX")  
  
# second part  
install.packages("auditor")  
install.packages("live")
```

Introduction

Do we need ML explainers?

Prediction of cardiovascular risk factors from retinal fundus photographs via deep learning

Ryan Poplin^{1,4}, Avinash V. Varadarajan^{1,4}, Katy Blumer¹, Yun Liu¹, Michael V. McConnell^{2,3}, Greg S. Corrado¹, Lily Peng^{1,4*} and Dale R. Webster^{1,4}

Traditionally, medical discoveries are made by observing associations, making hypotheses from them and then designing and running experiments to test the hypotheses. However, with medical images, observing and quantifying associations can often be difficult because of the wide variety of features, patterns, colours, values and shapes that are present in real data. Here, we show that deep learning can extract new knowledge from retinal fundus images. Using deep-learning models trained on data from 284,335 patients and validated on two independent datasets of 12,026 and 999 patients, we predicted cardiovascular risk factors not previously thought to be present or quantifiable in retinal images, such as age (mean absolute error within 3.26 years), gender (area under the receiver operating characteristic curve (AUC) = 0.97), smoking status (AUC = 0.71), systolic blood pressure (mean absolute error within 11.23 mmHg) and major adverse cardiac events (AUC = 0.70). We also show that the trained deep-learning models used anatomical features, such as the optic disc or blood vessels, to generate each prediction.

Risk stratification is central to identifying and managing groups at risk for cardiovascular disease, which remains the leading cause of death globally¹. Although the availability of cardiovascular disease risk calculators, such as the Pooled Cohort equations², Framingham^{3–5} and Systematic Coronary Risk Evaluation (SCORE)^{6,7}, is widespread, there are many efforts to improve risk predictions. Phenotypic information, particularly of vascular health, may further refine or reclassify risk prediction on an

changes^{22,23} and temic health c clinical utility work, we dem cardiovascula Machine le ety of classifi of our disease





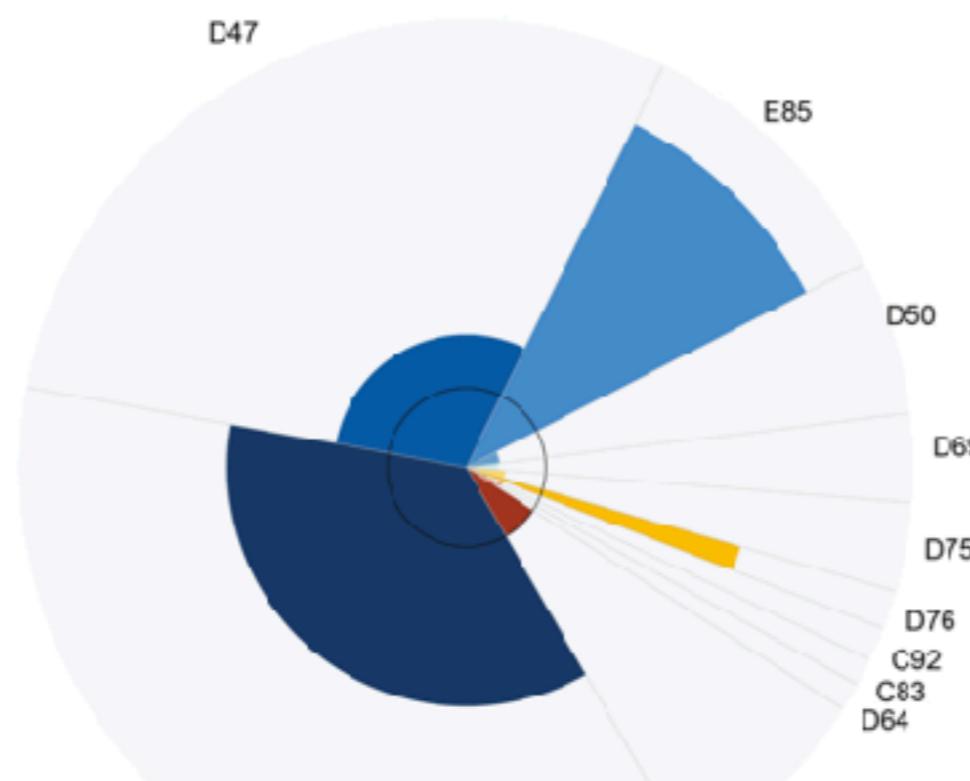
OPEN

An application of machine learning to haematological diagnosis

Received: 2 August 2017
Accepted: 14 December 2017
Published online: 11 January 2018

Gregor Gunčar¹, Matjaž Kukar¹, Mateja Notar¹, Miran Brvar², Peter Černelč³, Manca Notar¹ & Marko Notar¹

Quick and accurate medical diagnoses are crucial for the successful treatment of diseases. Using machine learning algorithms and based on laboratory blood test results, we have built two models to predict a haematologic disease. One predictive model used all the available blood test parameters and the other used only a reduced set that is usually measured upon patient admittance. Both models produced good results, obtaining prediction accuracies of 0.88 and 0.86 when considering the list of five most likely diseases and 0.59 and 0.57 when considering only the most likely disease. The models did not "finger and inc clinical special tests al unprec



ICD code	Prediction	Information score	Disease category
C90	36.20%	2.01	Multiple myeloma and malignant plasma cell neoplasms
D47	29.40%	0.67	Other neoplasms of uncertain or unknown behaviour of lymphoid, haematopoietic and related tissue
E85	10.20%	3.81	Amyloidosis
D50	5.60%	-1.37	Iron deficiency anaemia
D69	3.20%	-1.50	Purpura and other haemorrhagic conditions
D75	3.20%	-1.05	Other diseases of blood and blood-forming organs
D76	1.40%	2.60	Certain diseases involving lymphoreticular tissue and reticulohistiocytic system
C92	1.20%	-2.55	Myeloid leukaemia
C83	1.00%	-1.01	Diffuse non-Hodgkin lymphoma
D64	1.00%	-1.41	Other anaemias



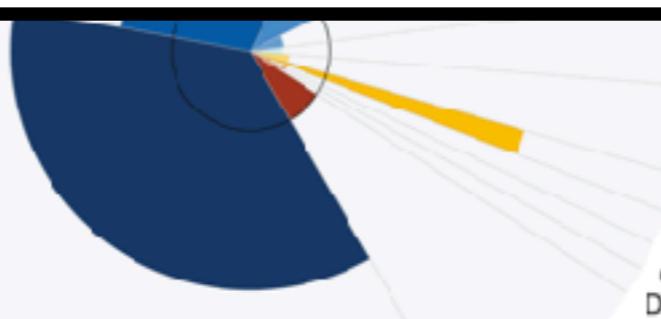
OPEN

An application of machine learning to haematological diagnosis

Received: 2 August
Accepted: 14 December
Published online: 1 January 2024

Imagine that these are predictions for
you.

Would you trust them?



D69	D69	3.20%	-1.50
D75	D75	3.20%	-1.05
D76	D76	1.40%	2.60
C92	C92	1.20%	-2.55
C83	C83	1.00%	-1.01
D64	D64	1.00%	-1.41

Black box models - what they are?

Trade-off between flexibility vs interpretability.

Complex structure

Ensembles like: Random Forests, Gradient Boosting Machines, Neural Networks have complex structure, highly non-linear and non-additive. It is hard to understand how input variables affect the final model outcome.

Wide input space

Even linear and additive models like Generalised Linear Regression Models, Generalised Additive Models, Rule Based Models suffer for lack of interpretability if there is a lot of non zero model components.

Non uniform/balanced variable distribution

Even additive models with sparse input space it may be hard to trace variable contributions for a single prediction if distribution of some variables is not balanced/uniform.

Why do we need explanations for complex models?

Domain validation

In ML the model overfitting is hard to control due to large number of tested classes of models, hyper-parameters optimisation, data-leaking in frequent validations. Explanations of Machine Learning models give the possibility of cross validate model behaviour against the domain knowledge.

New knowledge

Explanations for well working models may lead to new insights.

Trust

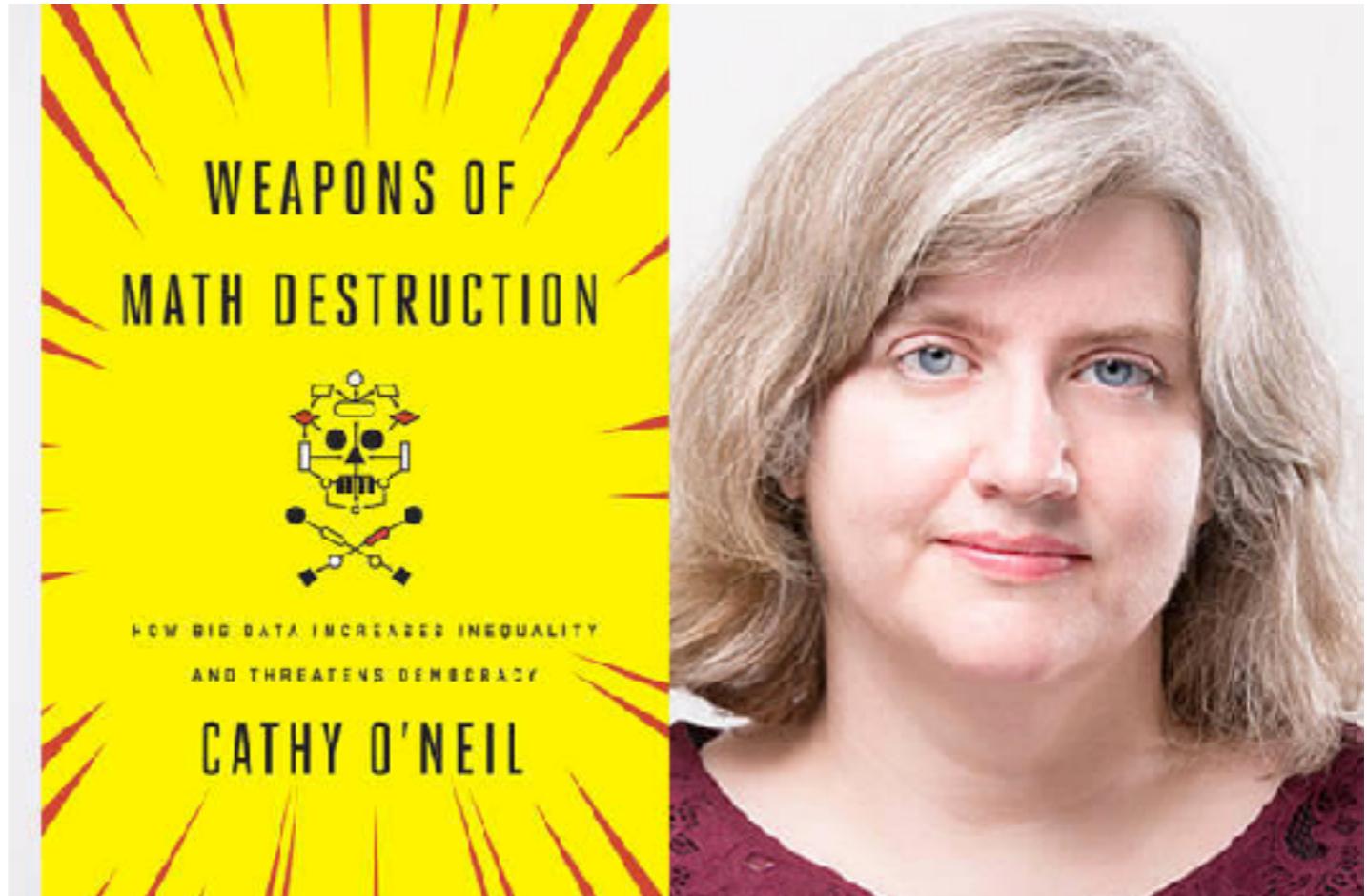
If model predictions are to be consumed they need to be trusted. For high-stake decisions, like medical treatment, this may be even a requirement.

GDPR - General Data Protection Regulation

Right to explanation of all decisions made by automated or artificially intelligent algorithmic systems (not in the GDPR text, yet often discussed).

Why do we need explanations for complex models?

Cathy O'Neil:
The era of blind faith
machine learning
~~in big data must end~~



- “You don’t see a lot of skepticism,” she says. “The algorithms are like shiny new toys that we can’t resist using. We trust them so much that we project meaning on to them.”
- Ultimately algorithms, according to O’Neil, reinforce discrimination and widen inequality, “using people’s fear and trust of mathematics to prevent them from asking questions”.

<https://www.theguardian.com/books/2016/oct/27/cathy-oneil-weapons-of-math-destruction-algorithms-big-data>

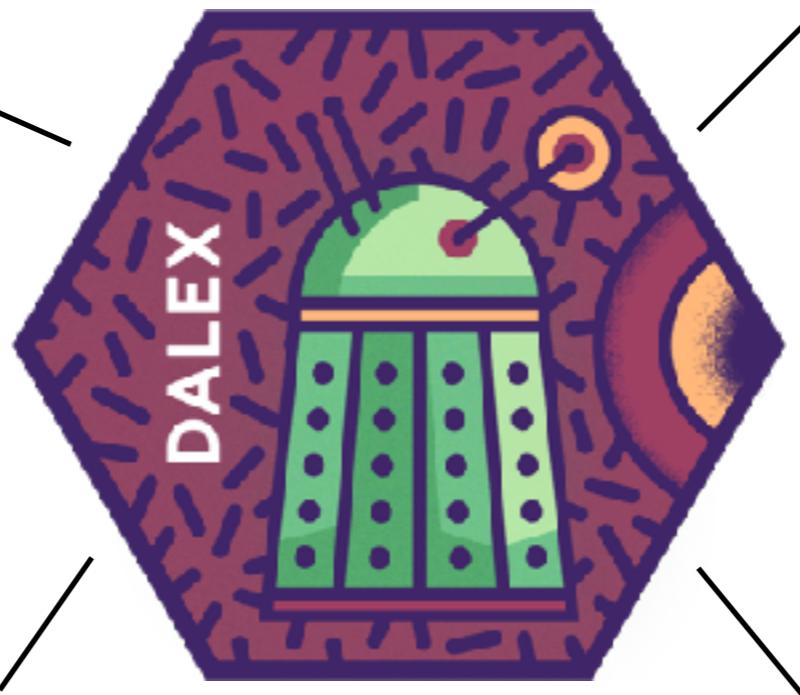
DALEX is a set of tools that helps to understand the way complex predictive models work

How good is the predictive model?

Which variables are the most important in general?

How good is the model fit?

Which variables influence the single prediction?



Variable Explainers
package: **pdp**, **ALEPlot**, **factorMerger**

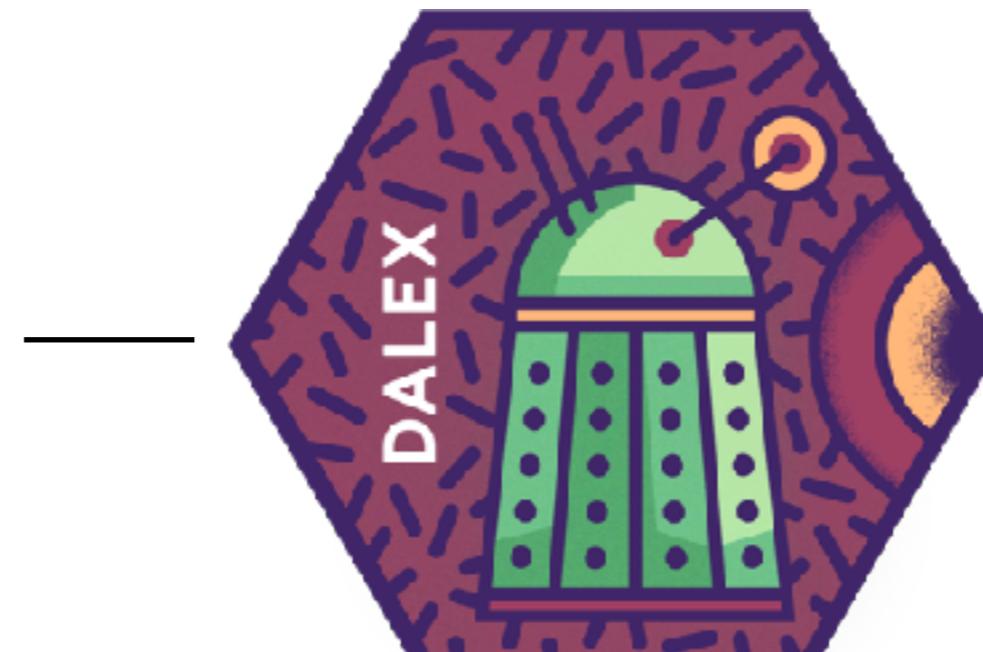
Structure Explainers
package: **randomForest**

Model Management
package: **archivist**

Model Diagnostic Tools
package: **auditor**, **ggfortify**

Model Performance Explainers
package: **auditor**, **ROCR**, **caret**, **mlr**

Model Predictions Explainers
package: **breakDown**, **live**, **shapleyr**, **lime**



How to...

<https://github.com/pbiecek/DALEX>

README.md

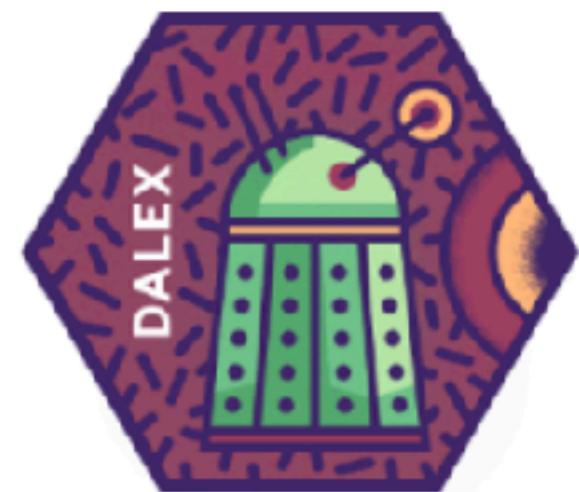
DALEX

[CRAN](#) 0.2.0 [downloads](#) 1377 [build](#) passing [coverage](#) 95%

Descriptive mAchine Learning EXplanations

DALEX Stories

- [A gentle introduction to DALEX with examples](#)
- [How to use DALEX with caret](#)
- [How to use DALEX with mlr](#)
- [How to use DALEX with xgboost package](#)



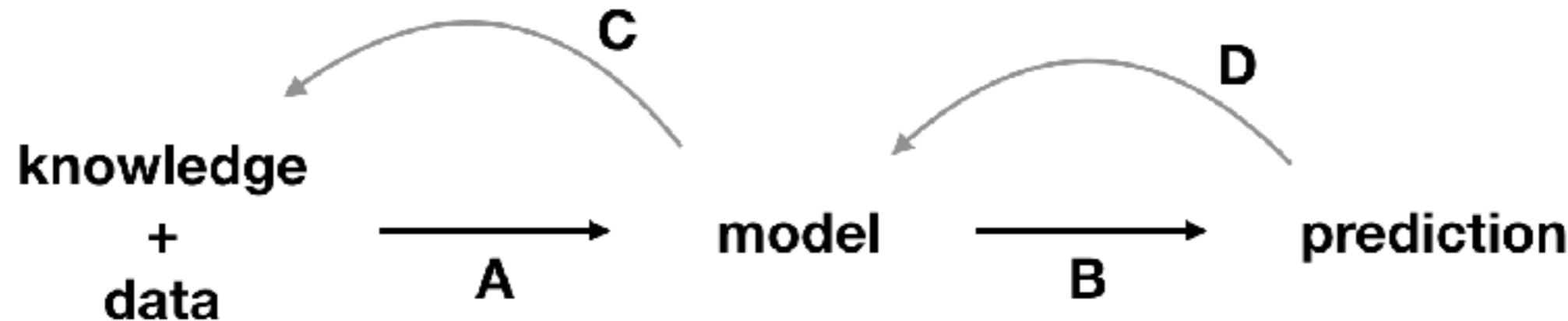
Use Case:
Predictive models for
apartment prices

Typical workflow in ML



- A. Modelling is a process in which domain knowledge and data are turned into models.
- B. Models are used to generate predictions.

Typical workflow in ML



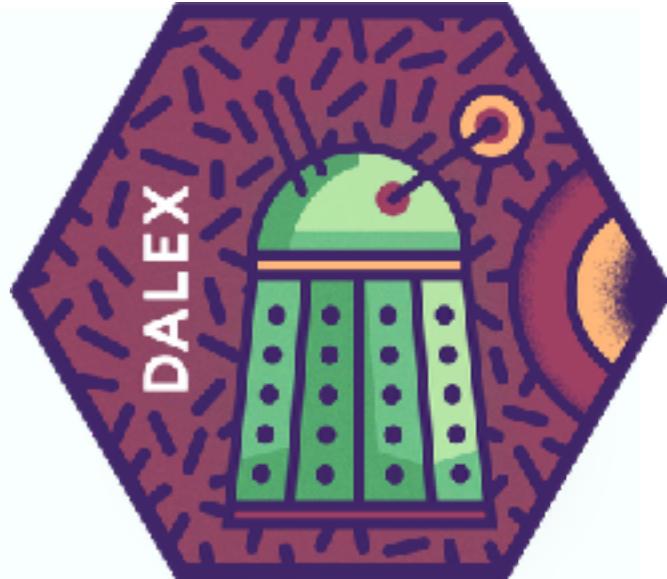
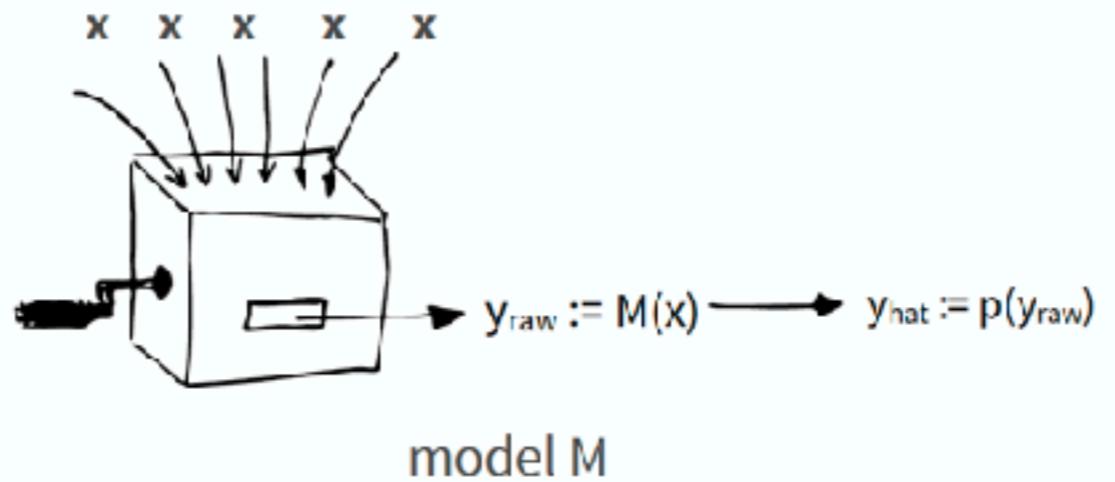
- A. Modelling is a process in which domain knowledge and data are turned into models.
- B. Models are used to generate predictions.
- C. Understanding of model structure may increase our knowledge and in consequence leads to a better model. *DALEX helps here.*
- D. Understanding of drivers behind particular model predictions may help to correct wrong decisions and in consequence leads to a better model.
DALEX helps here.





```
library("DALEX")
head(apartments)
```

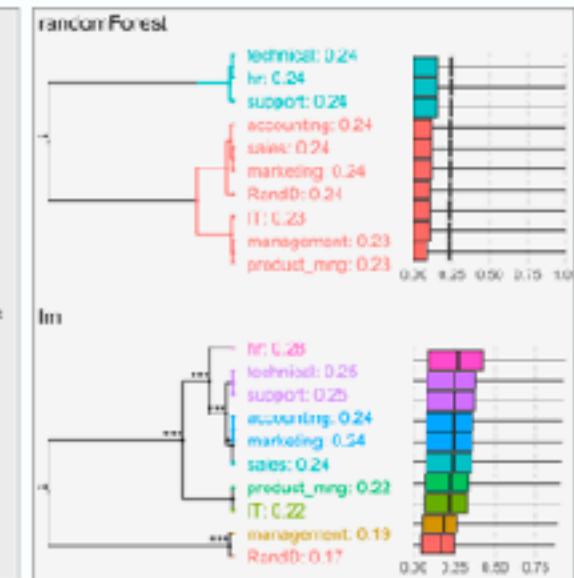
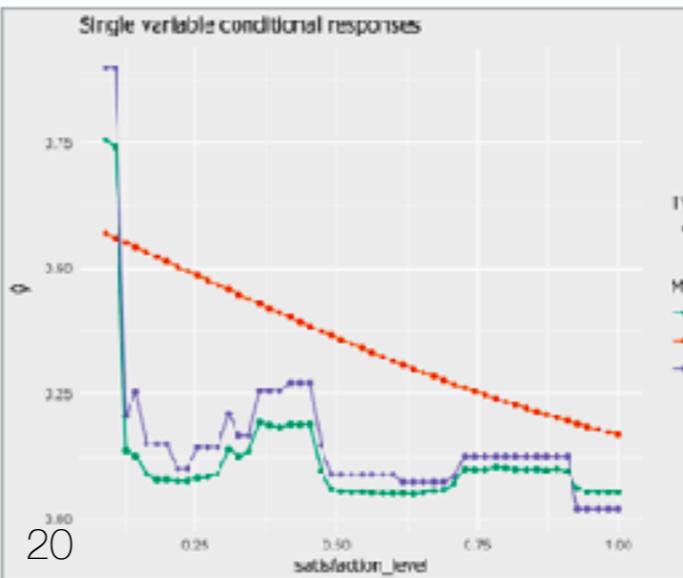
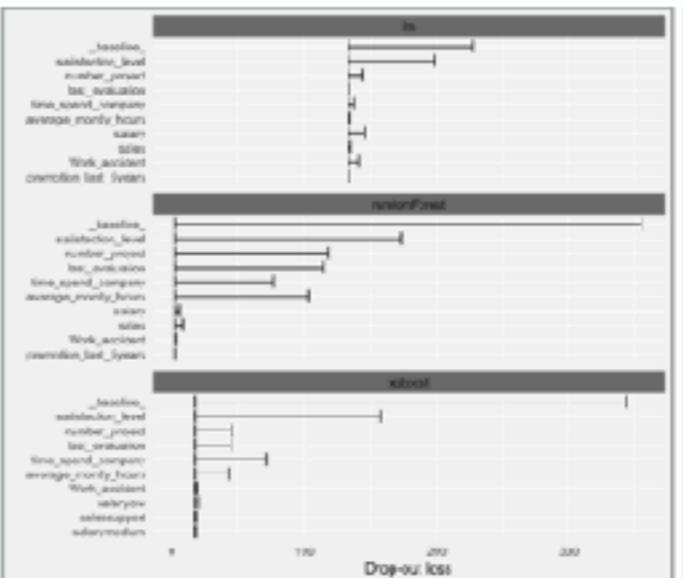
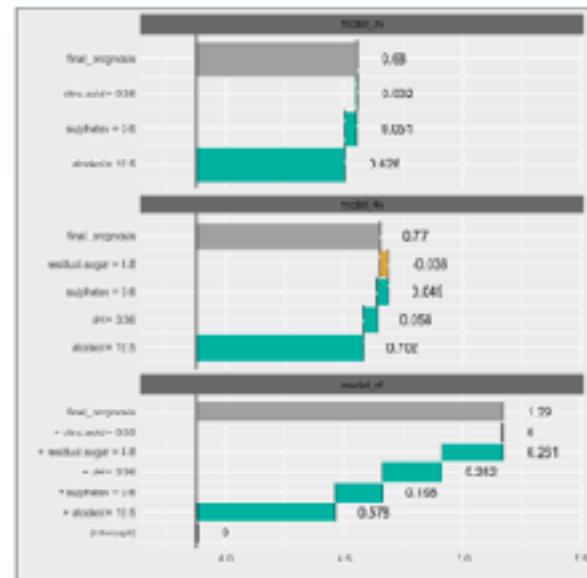
m2.price	construction.year	surface	floor	no.rooms	district
5897	1953	25	3		1 Śródmieście
1818	1992	143	9		5 Bielany
3643	1937	56	1		2 Praga
3517	1995	93	7		3 Ochota
3013	1992 ¹⁹	144	6		5 Mokotow

A)**B)**

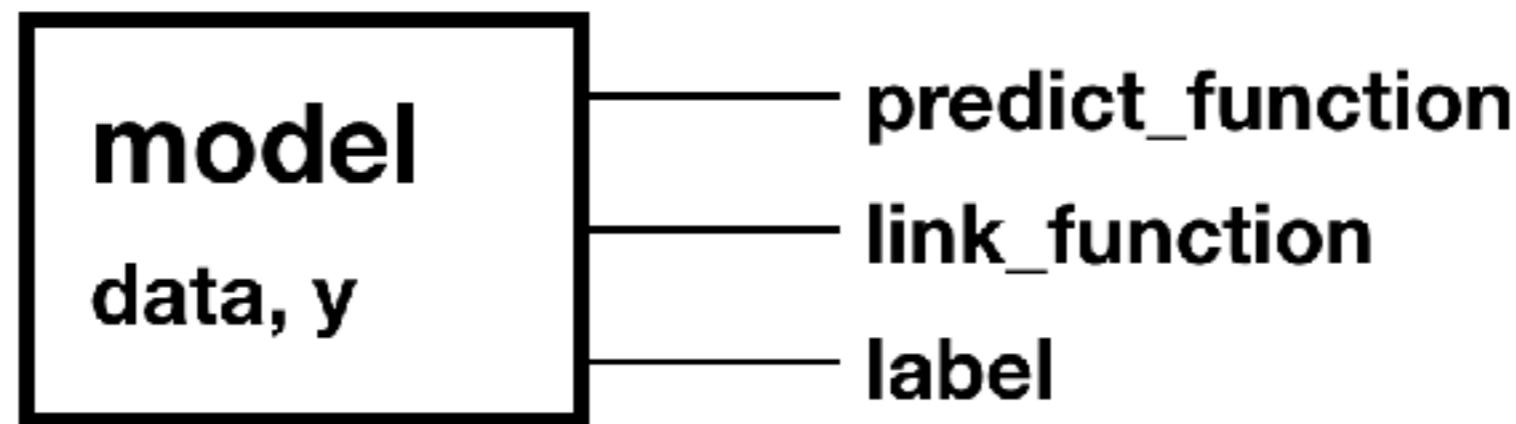
```
explain(model; data; y; predict_function; trans)
```

C)

```
single_prediction(explainer, x)    variable_dropout(explainer)    single_variable(explainer, variable)
```



Create an explainer

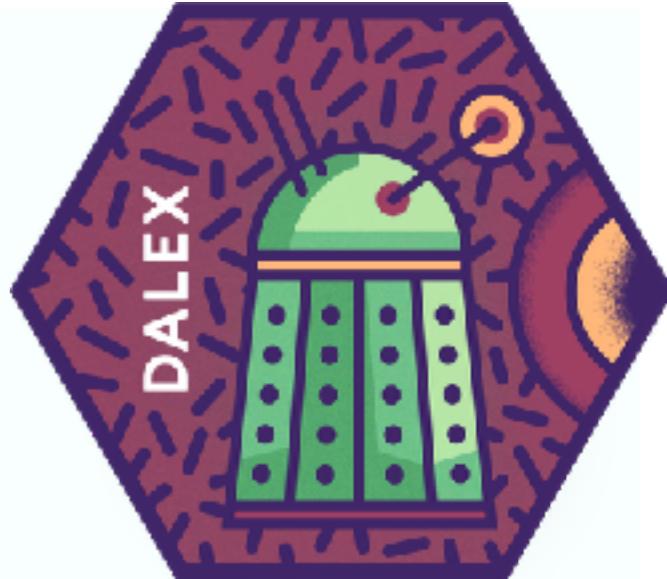
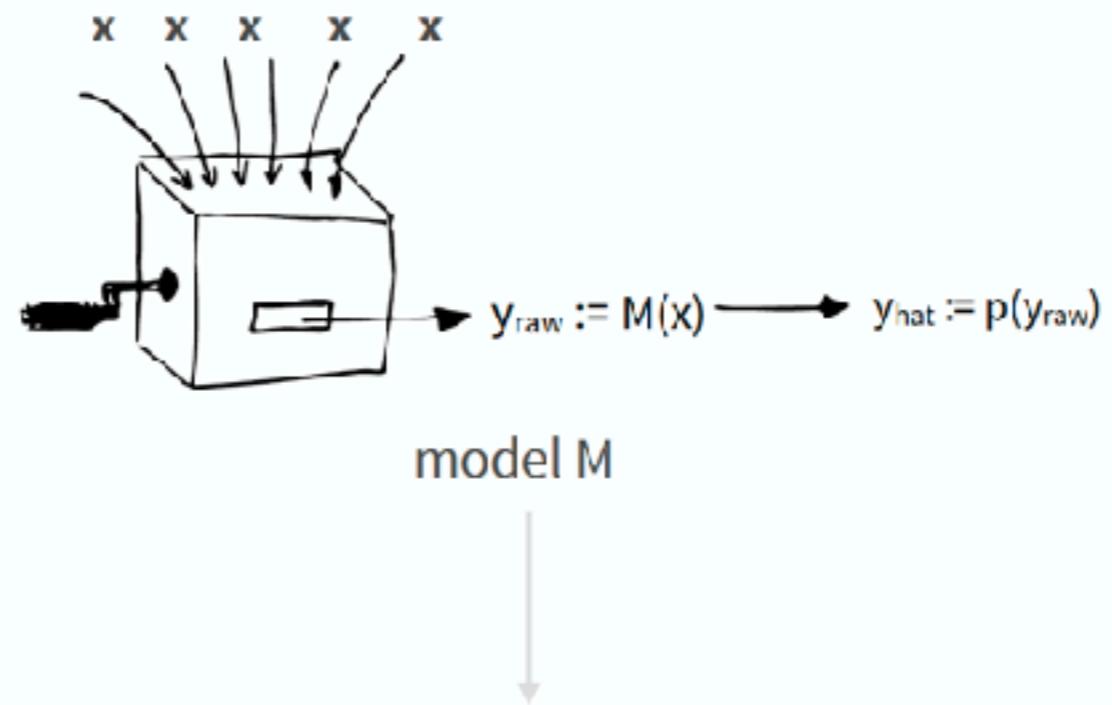


```
explain(model, data, y, predict_function,  
        link, ..., label)
```

Explainer encapsulates:

- model (any class)
- validation data and labels (y)
- predict interface that returns vector of scores
- other arguments that will be passed to predict_function
- model name

A)



B)

```
explain(model; data; y; predict_function; trans)
```

```
library("DALEX")
apartments_lm_model <- lm(m2.price ~ construction.year + surface + floor +
                           no.rooms + district, data = apartments)

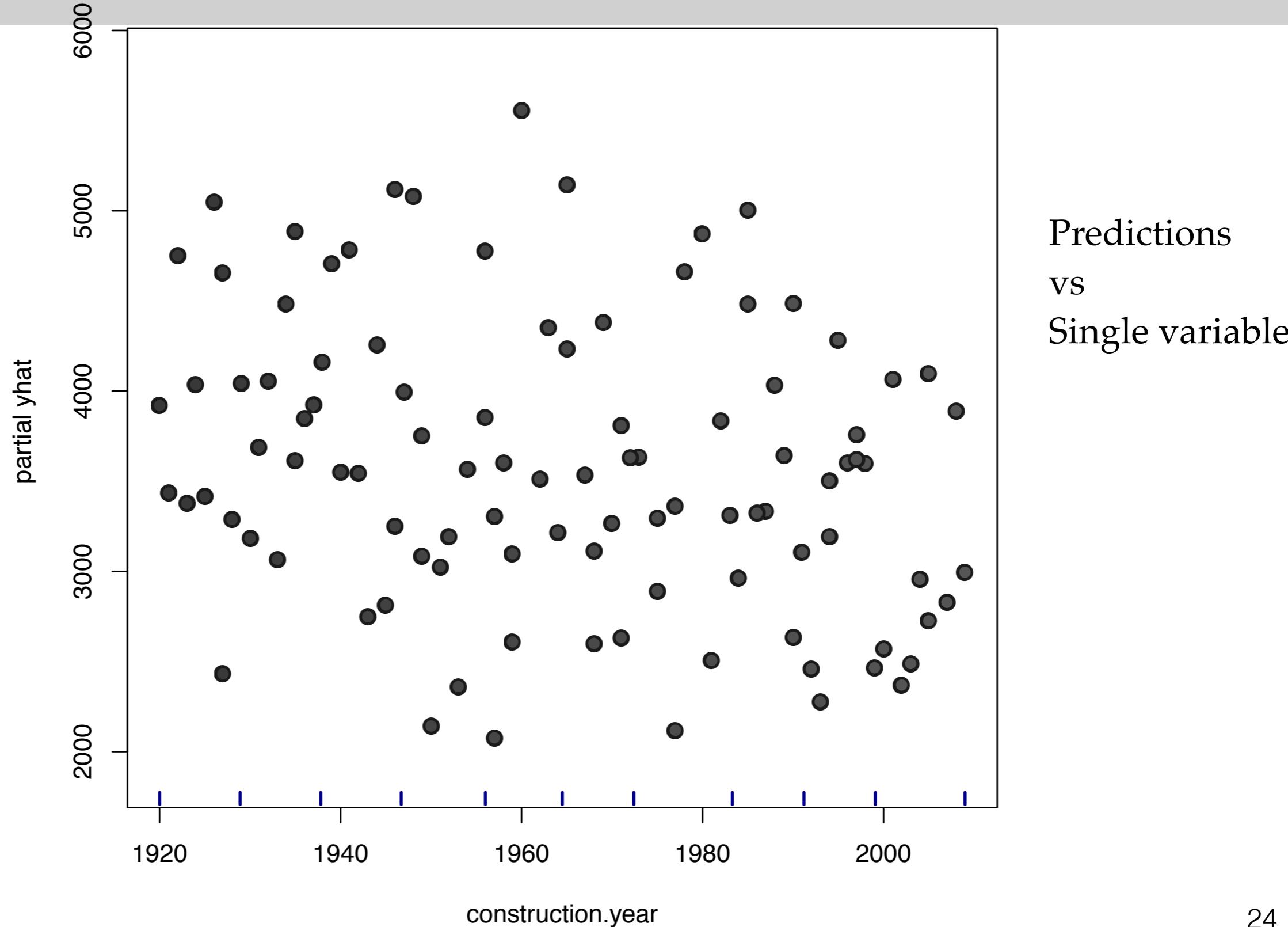
library("randomForest")
set.seed(59)
apartments_rf_model <- randomForest(m2.price ~ construction.year + surface + floor +
                                       no.rooms + district, data = apartments)

explainer_lm <- explain(apartments_lm_model,
                        data = apartmentsTest[,2:6], y = apartmentsTest$m2.price)
explainer_rf <- explain(apartments_rf_model,
                        data = apartmentsTest[,2:6], y = apartmentsTest$m2.price)
```

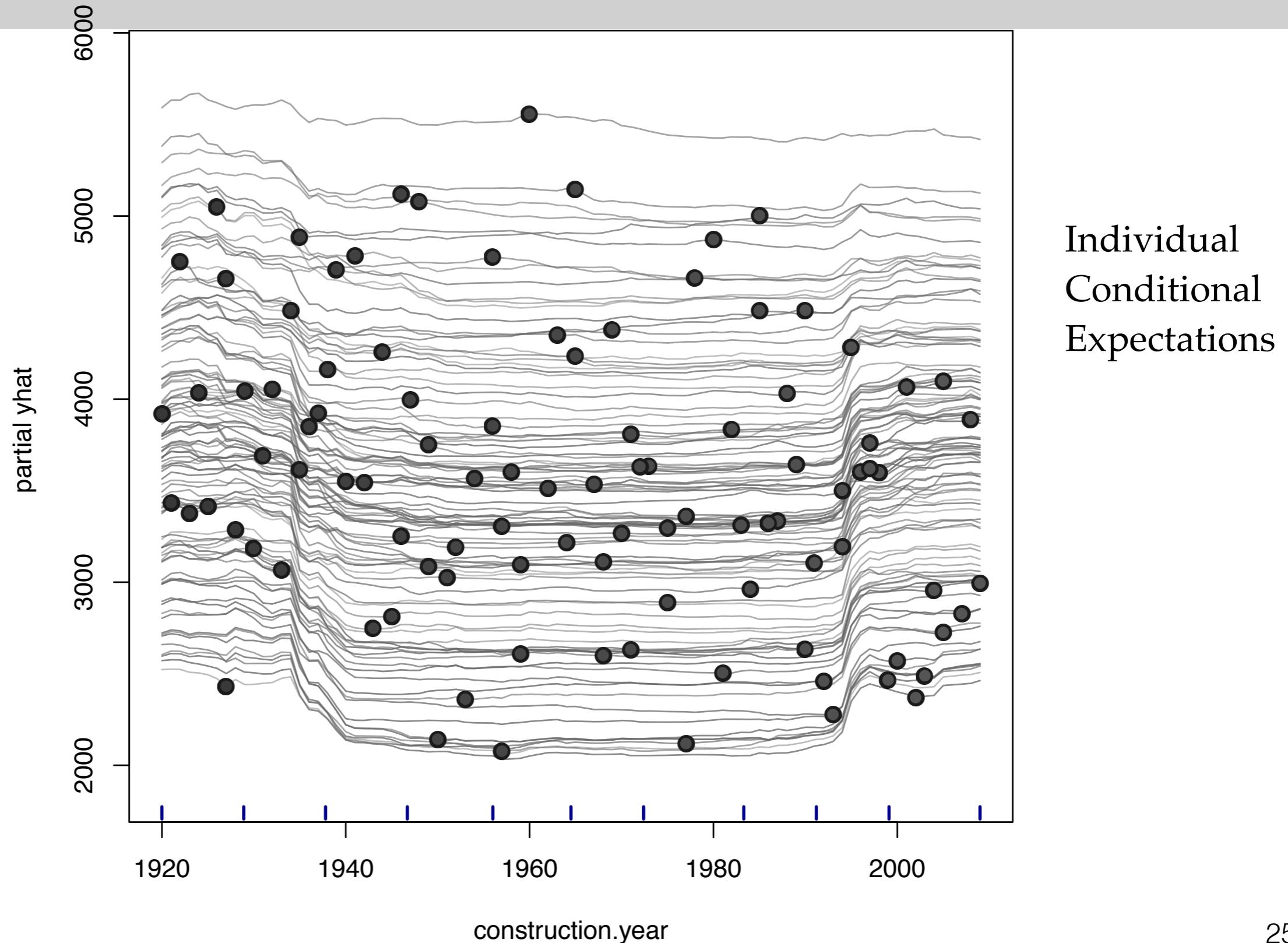
Model explainers - Continuous variable response

How a single feature
affects the model response?

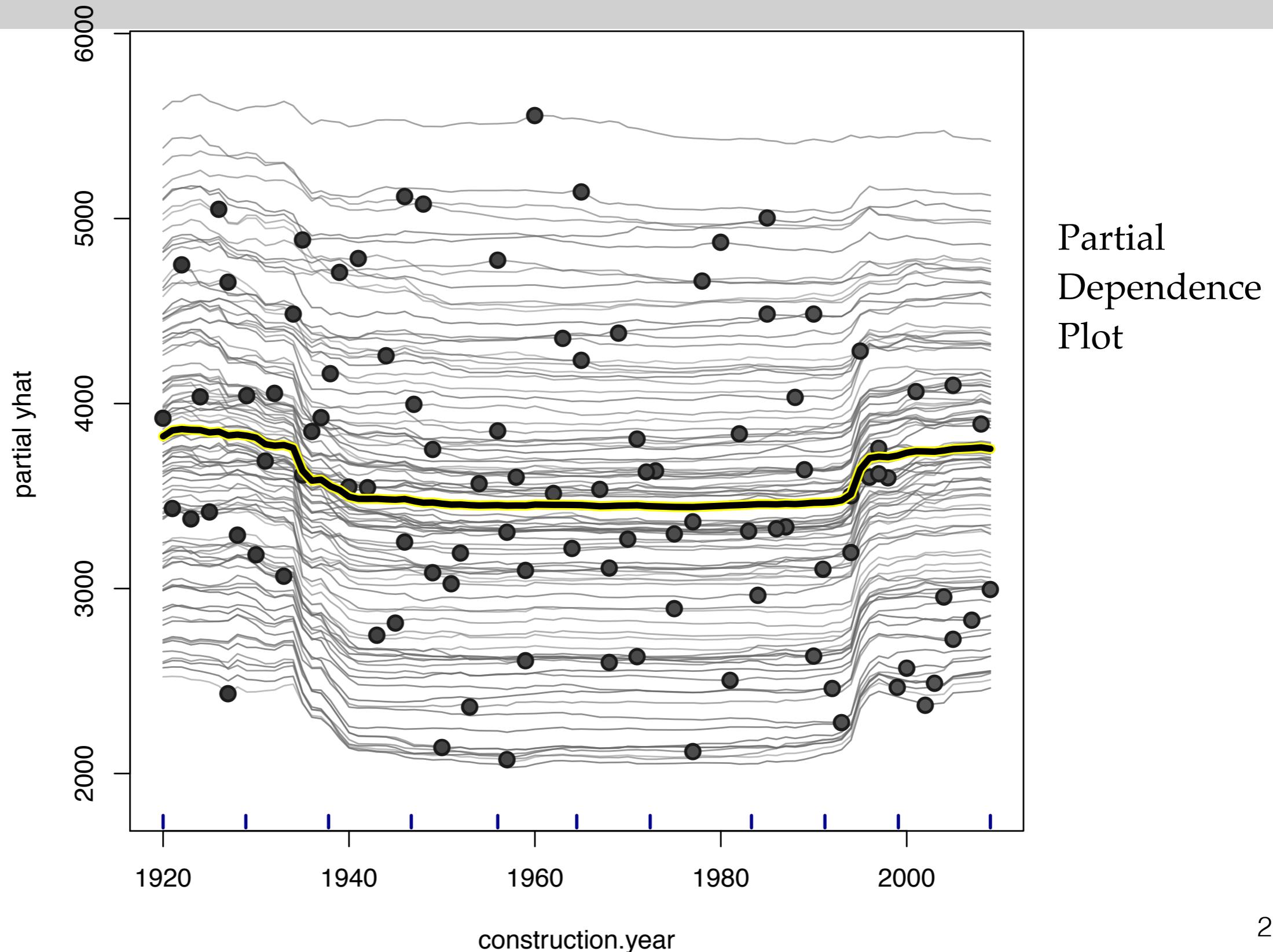
How a single variable affects the response?



How a single variable affects the response?



How a single variable affects the response?



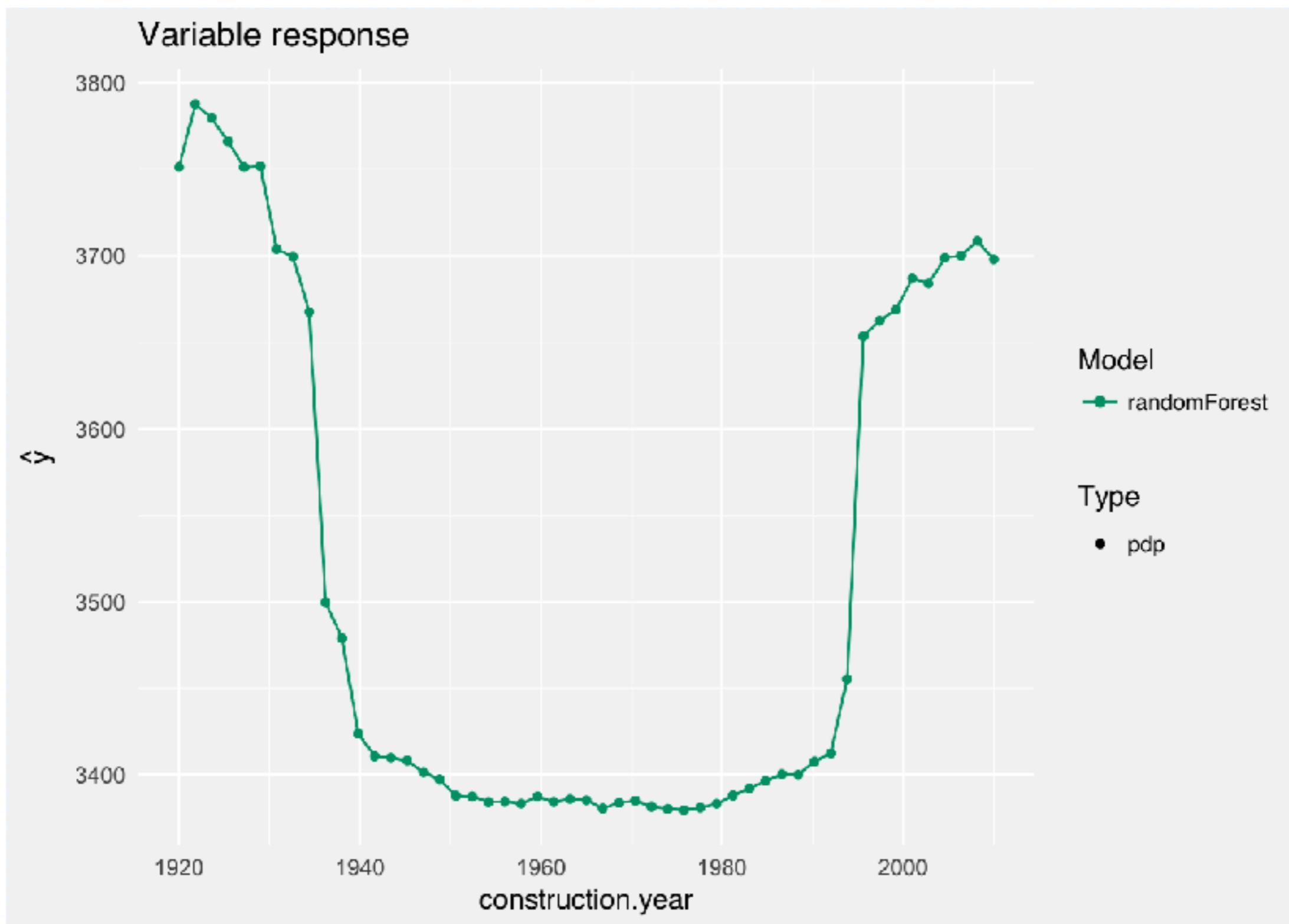
Formally, partial dependence may be defined as

$$p_i(x_i) = E_{x_{-i}}[f(x^i, x^{-i}; \theta)]$$

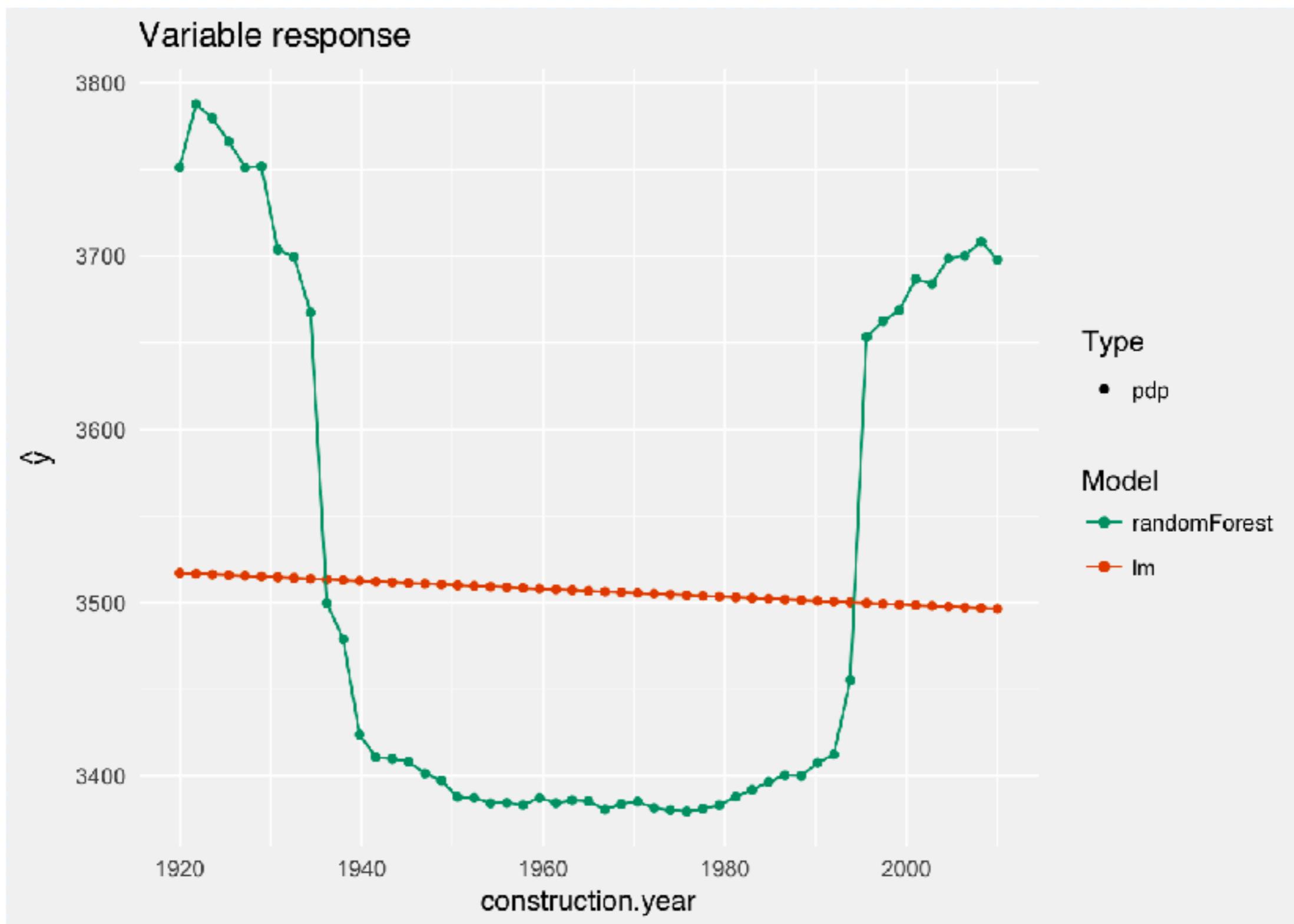
and estimated as

$$\hat{p}_i(x_i) = \frac{1}{n} \sum_{j=1}^n f(x_j^i, x_j^{-i}, \hat{\theta})$$

```
sv_rf <- single_variable(explainer_rf, variable = "construction.year", type = "pdp")
plot(sv_rf)
```



```
sv_lm <- single_variable(explainer_lm, variable = "construction.year", type = "pdp")  
  
plot(sv_rf, sv_lm)
```



The dark side of partial dependence

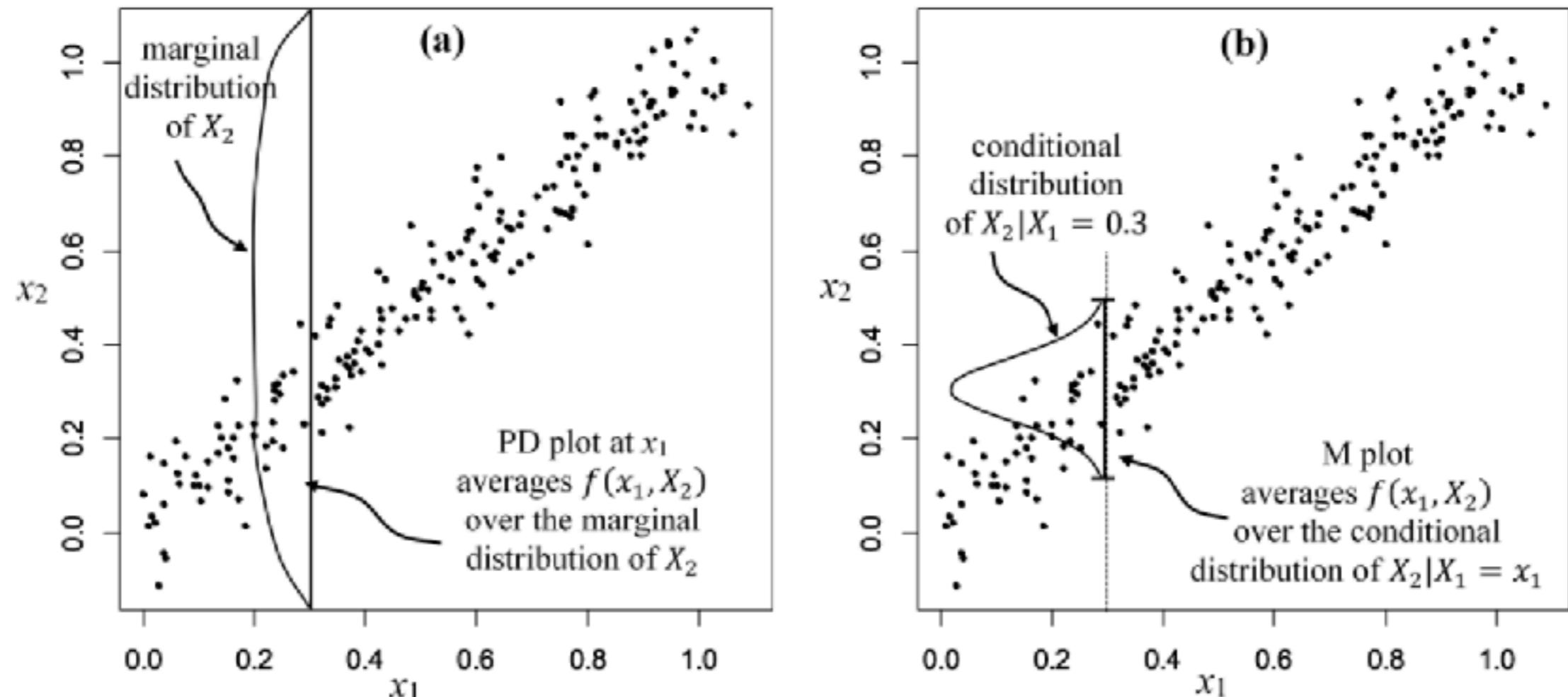
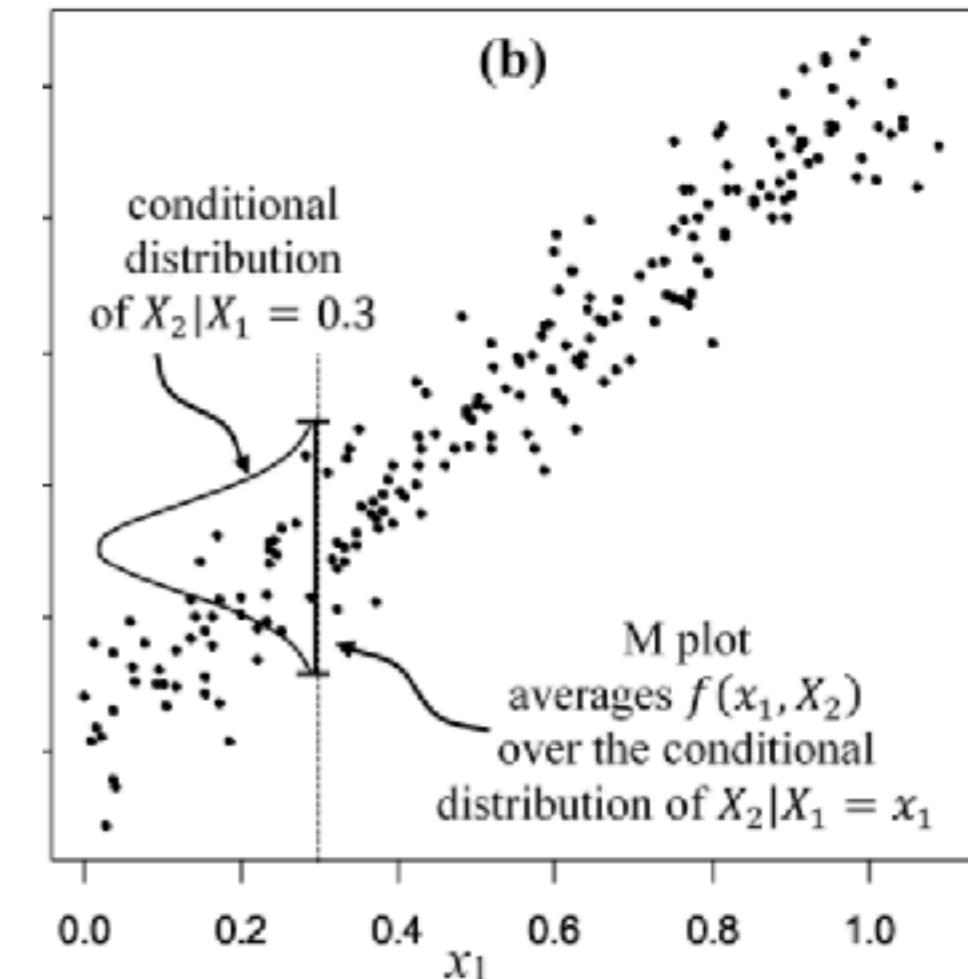
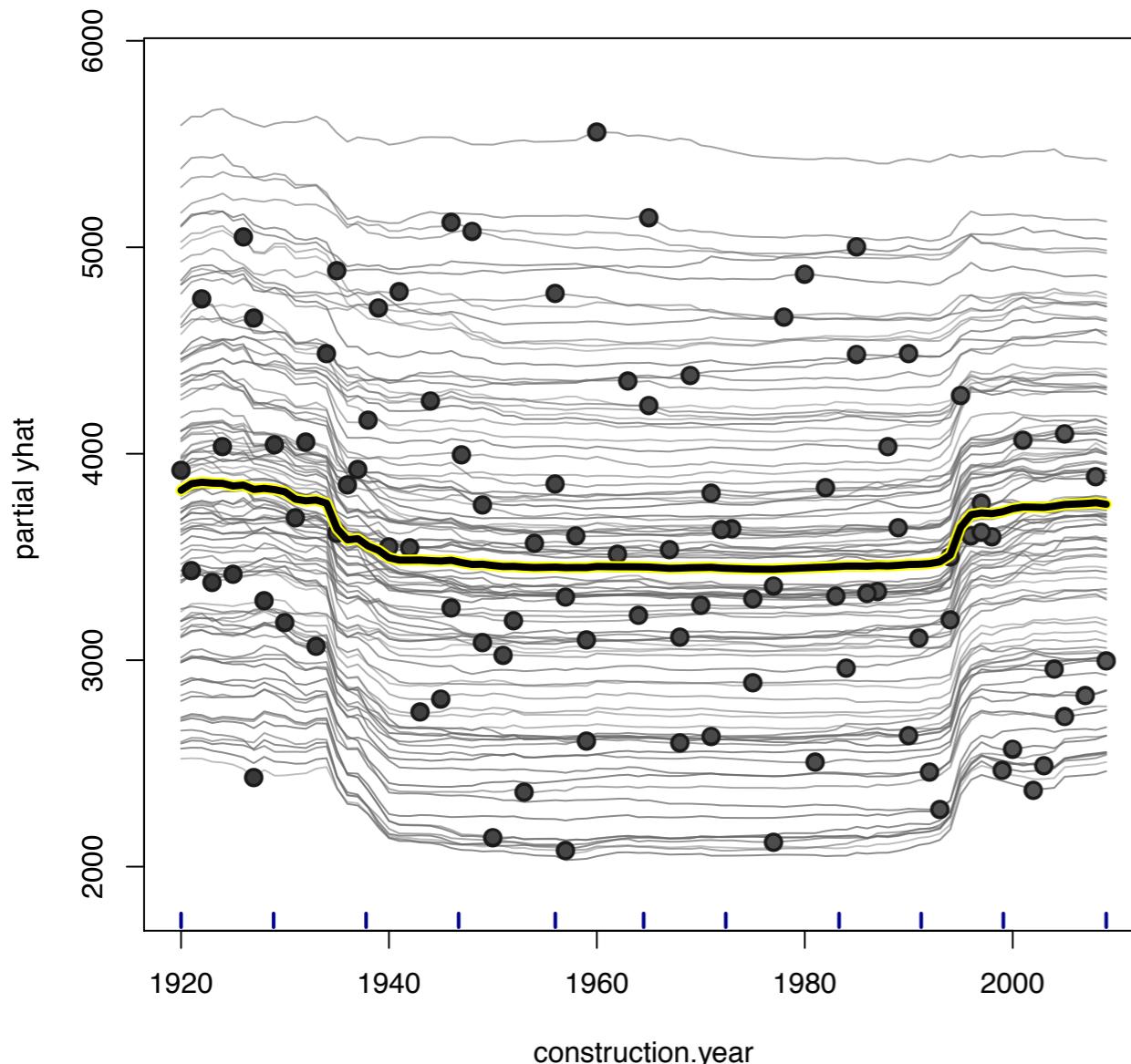


Figure 1: Illustration of the differences between the computation of (a) $f_{1,PD}(x_1)$ and (b) $f_{1,M}(x_1)$ at $x_1 = 0.3$.

From: [https://cran.r-project.org/web/packages/ALEPlot/vignettes/](https://cran.r-project.org/web/packages/ALEPlot/vignettes/AccumulatedLocalEffectPlot.pdf)
[AccumulatedLocalEffectPlot.pdf](https://cran.r-project.org/web/packages/ALEPlot/vignettes/AccumulatedLocalEffectPlot.pdf)

The dark side of partial dependence

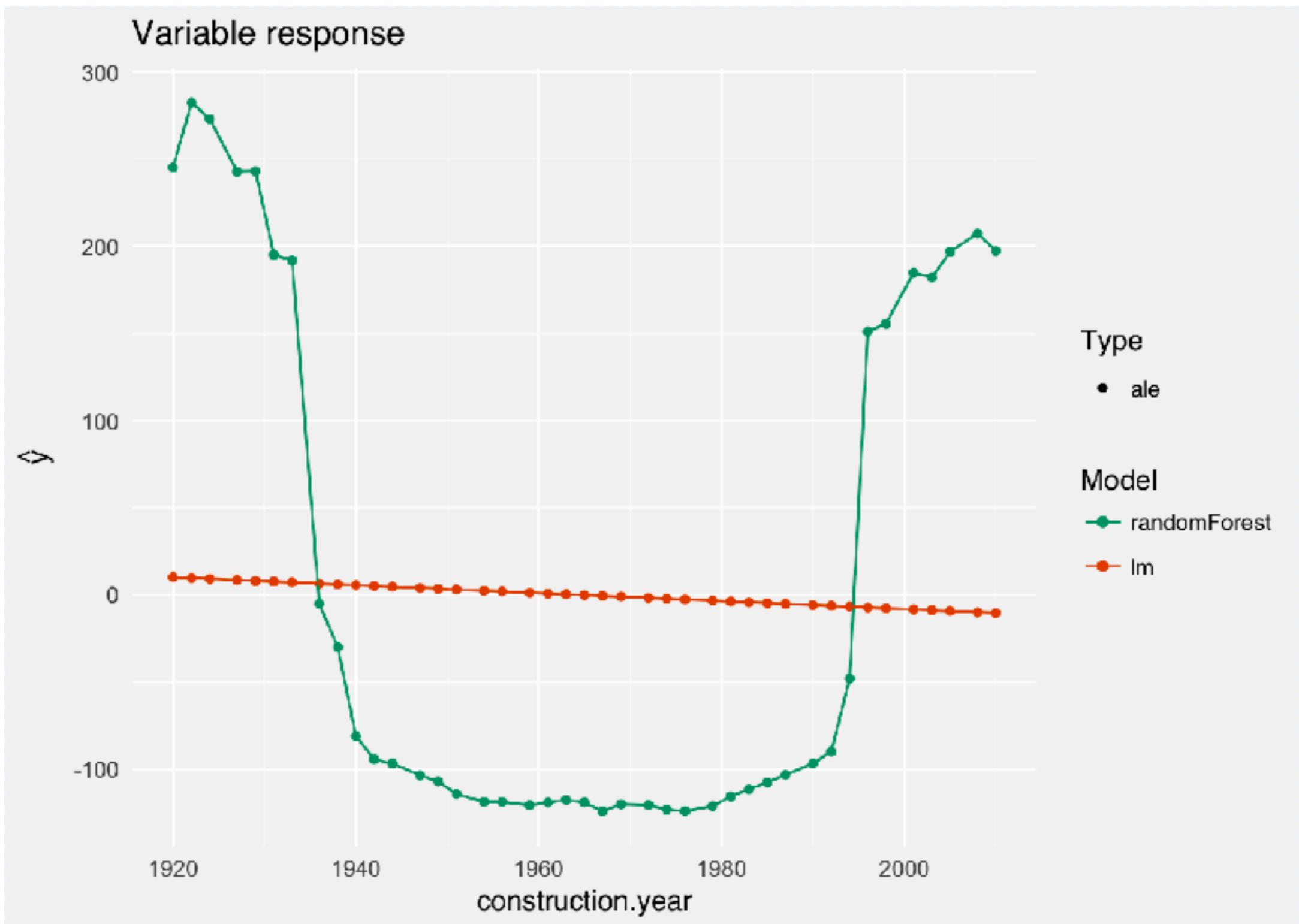


computation of (a) $f_{1,PD}(x_1)$ and (b)

From: [https://cran.r-project.org/web/packages/ALEPlot/vignettes/](https://cran.r-project.org/web/packages/ALEPlot/vignettes/AccumulatedLocalEffectPlot.pdf)
[AccumulatedLocalEffectPlot.pdf](https://cran.r-project.org/web/packages/ALEPlot/vignettes/AccumulatedLocalEffectPlot.pdf)

```
sva_rf <- single_variable(explainer_rf, variable = "construction.year", type = "ale")
sva_lm <- single_variable(explainer_lm, variable = "construction.year", type = "ale")

plot(sva_rf, sva_lm)
```



Find more in:

Package pdf

Partial Dependence Plots, Greenwell 2017

Package ALEPlot

Accumulated Local Effects Plots, Apley 2017

Package ICEbox

Individual Conditional Expectations, Goldstein ,
Kapelner, Bleich, Pitkin 2015

Your turn!

1. Plot single variable explainer for the surface variable.
2. Fit gbm (Generalized Boosted Regression Models), knn (k-Nearest Neighbour) and svm (Support Vector Machines) models.
3. Plot single variable explainer for the construction.year variable.

Find some help in the Workshop_eRum_2018_part1.R

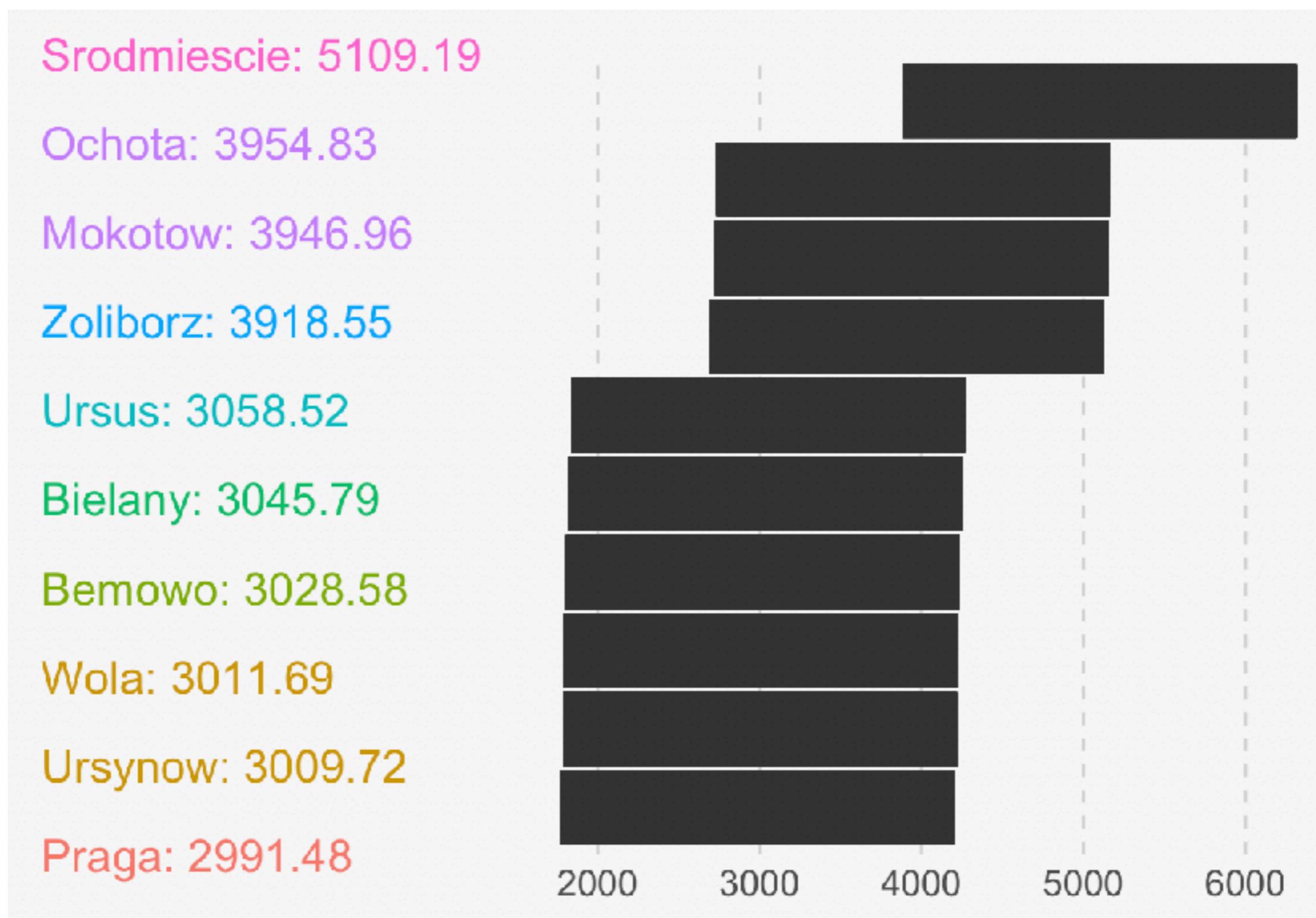
Model explainers - Discrete variable response

How a single feature
affects the model response?

What to do with factors?

- Partial Dependency Plots are designed for continues variables.
- For categorical variables we would like to see how the model response is affected by a particular level of categorical variable.
- To show this we are using The Merging Path Plots, as described in Sitko, Biecek (2017) <https://arxiv.org/pdf/1709.04412.pdf>
- Similarity of factors are shown with dendrogram, which height corresponds to log likelihood.

How a single variable affects the response?



How a single variable affects the response?

Algorithm 1 The outline of the Merging Path Plot algorithm implemented in **factorMerger**

```
function MERGEFACTORS(responseVariable, groupingVariable, adjacent)
2:   currentModel := createModel(responseVariable, groupingVariable)
      mergingPath := list(currentModel)
4:   while |levels(groupingVariable)| ≥ 1 do
      pairsSet := generatePairs(groupingVariable, responseVariable, adjacent)
6:   selectedPair := argmaxpair ∈ pairsSet objectiveFunction(pair, responseVariable,
      groupingVariable)
      groupingVariable := mergeLevels(groupingVariable, selectedPair)
8:   currentModel := createModel(responseVariable, groupingVariable)
      mergingPath := add(mergingPath, currentModel)
10:  end while
       return(mergingPath)
12: end function
```

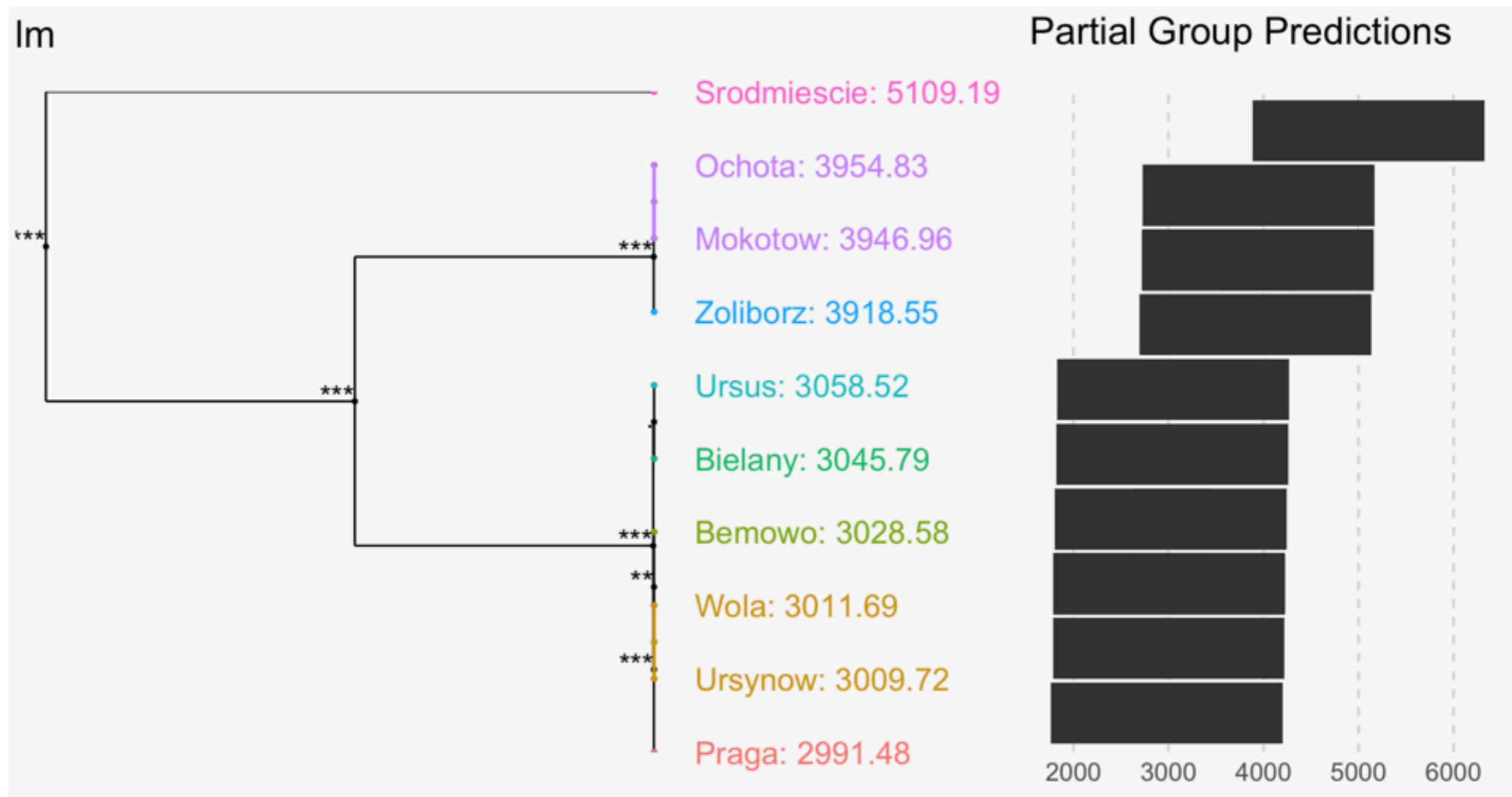
The Merging Path Plot: adaptive fusing of k-groups with likelihood-based model selection

Agnieszka Sitko, Przemysław Biecek (2017)

<https://arxiv.org/abs/1709.04412>

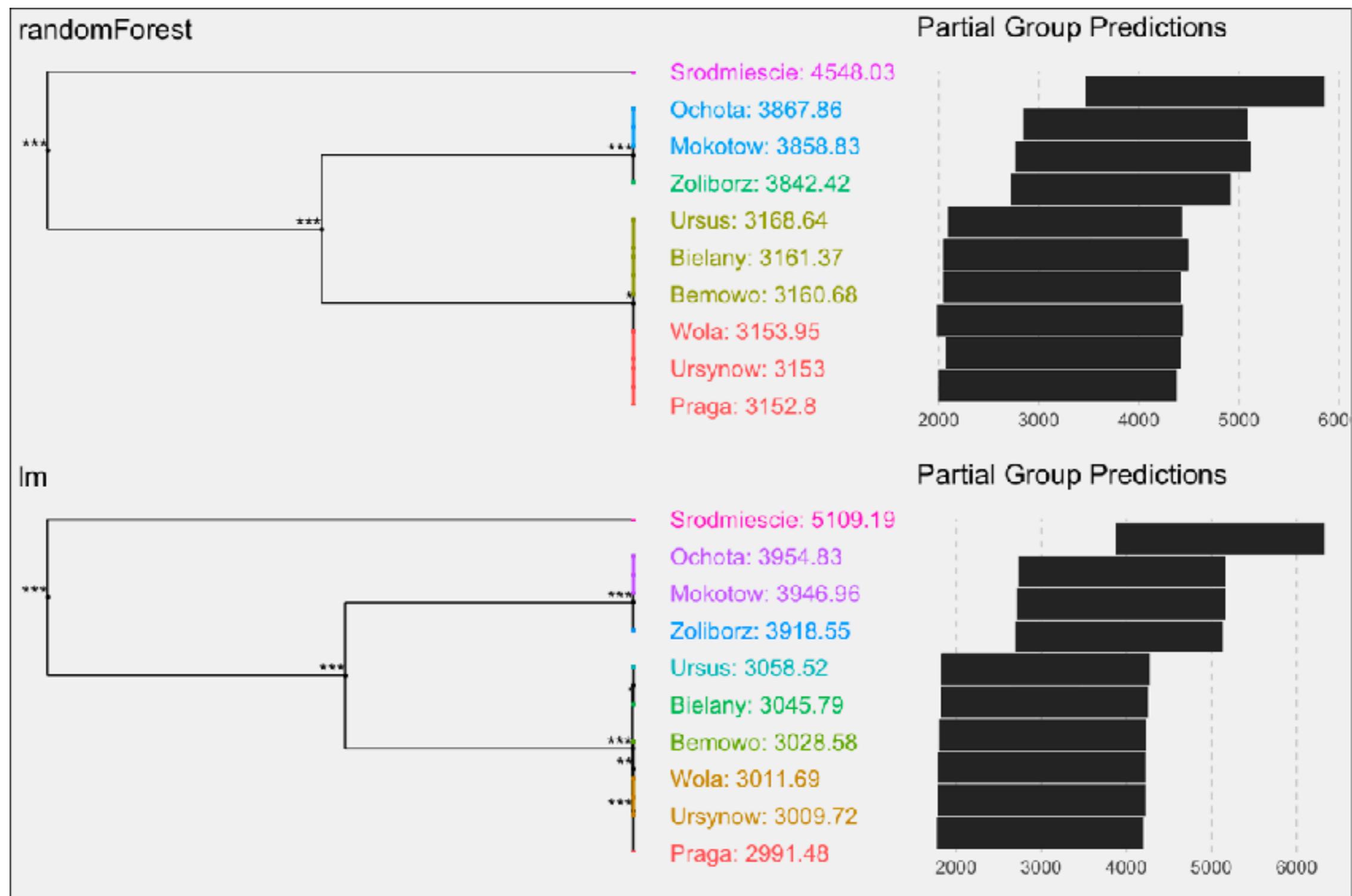
<https://github.com/geneticsMiNIng/FactorMerger>

```
svd_lm <- single_variable(explainer_lm, variable = "district", type = "factor")  
  
plot(svd_lm)
```

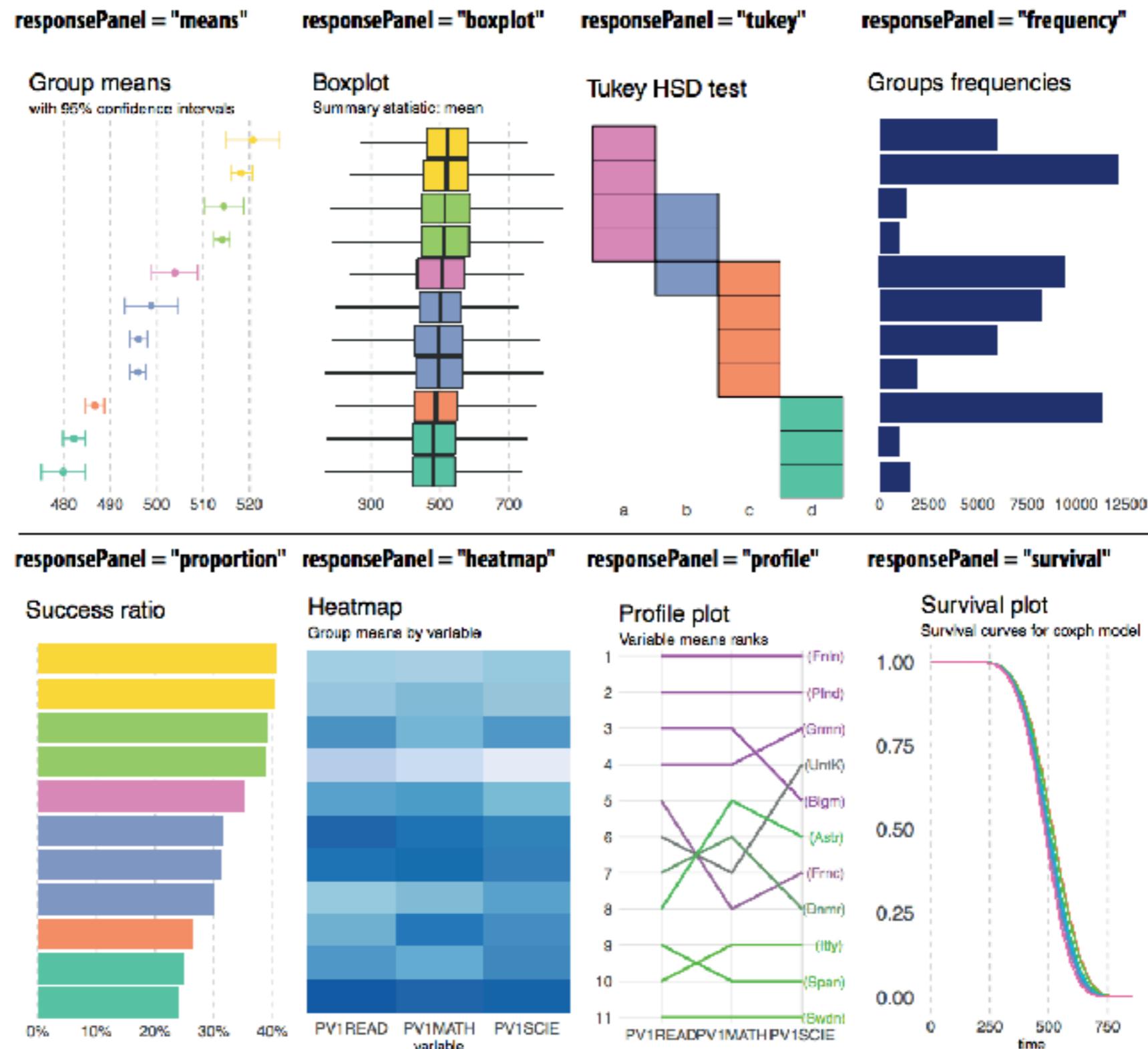


```
svd_rf <- single_variable(explainer_rf, variable = "district", type = "factor")
svd_lm <- single_variable(explainer_lm, variable = "district", type = "factor")

plot(svd_rf, svd_lm)
```



Other panels supported by factorMerger



Your turn!

1. Fit gbm (Generalized Boosted Regression Models) and knn (k-Nearest Neighbour) model.
2. Plot single variable explainer for the district variable.

Find some help in the Workshop_eRum_2018_part1.R

Model explainers - Performance

How accurate is the model?

Accuracy as a single number is not enough!

It is common to select a model based on a single criteria, like an accuracy calculated on test set (or with CV-like technique).

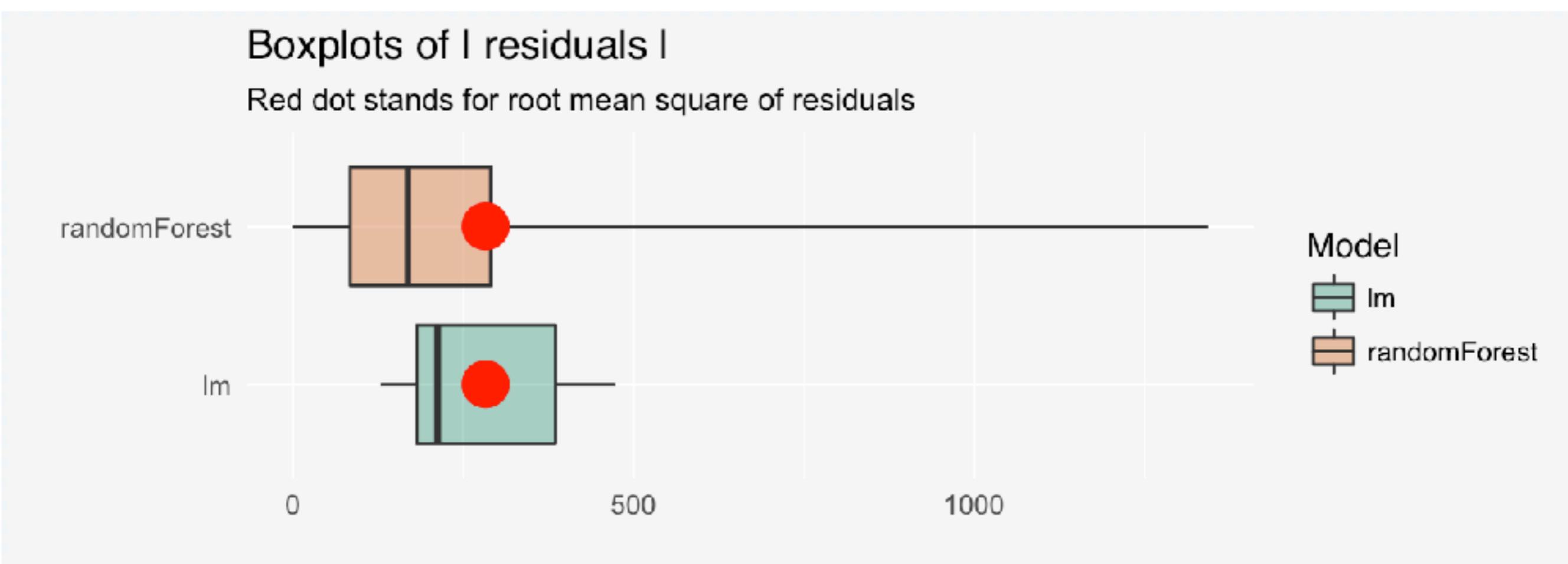
But what to do in such situation:

```
# root mean square
predicted_mi2_lm <- predict(apartments_lm_model, apartmentsTest)
sqrt(mean((predicted_mi2_lm - apartmentsTest$m2.price)^2))
## [1] 283.0865
```

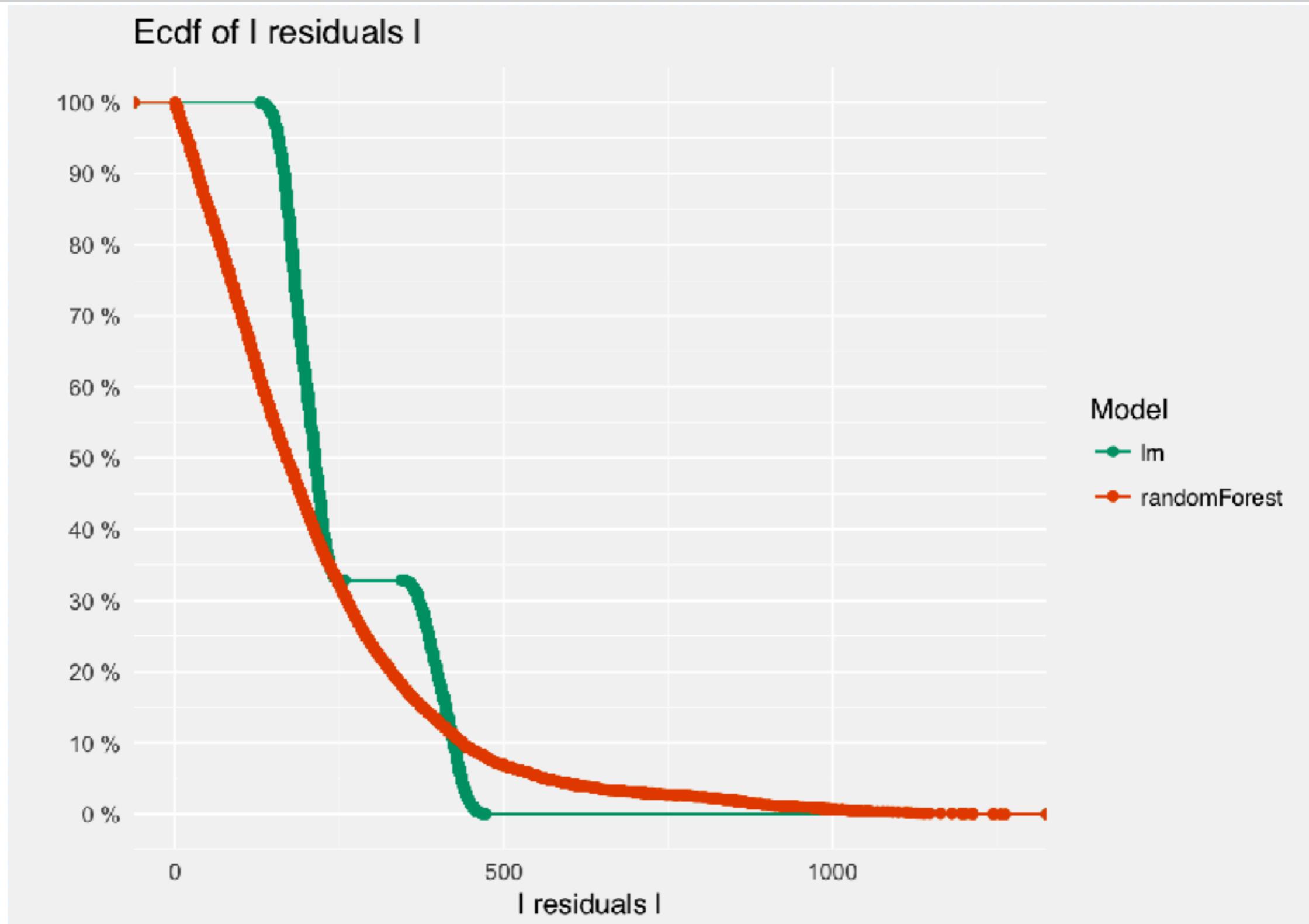
```
# root mean square
predicted_mi2_rf <- predict(apartments_rf_model, apartmentsTest)
sqrt(mean((predicted_mi2_rf - apartmentsTest$m2.price)^2))
## [1] 283.3479
```

Accuracy as a single number is not enough!

```
plot(mp_lm, mp_rf, geom = "boxplot")
```



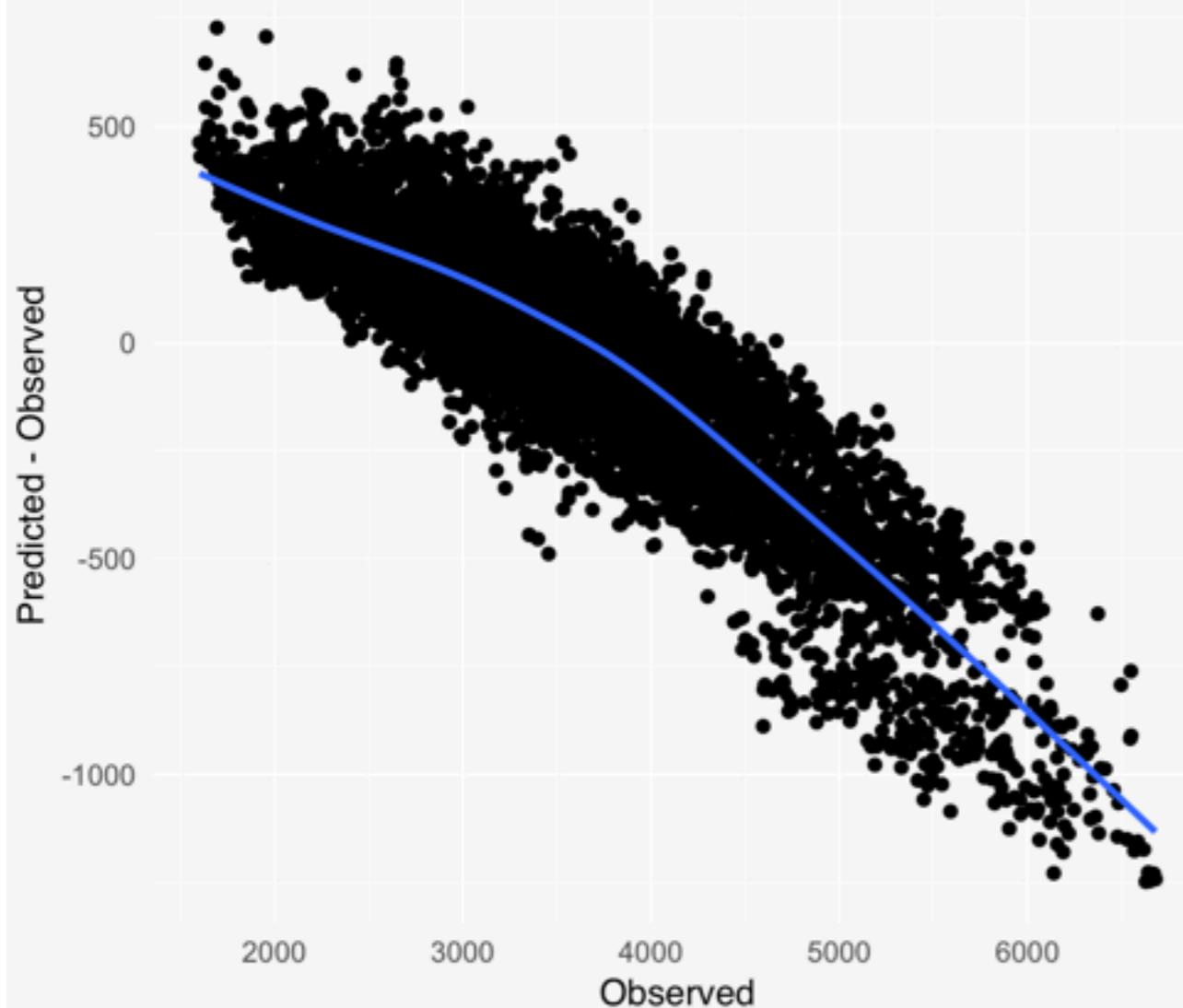
Accuracy as a single number is not enough!



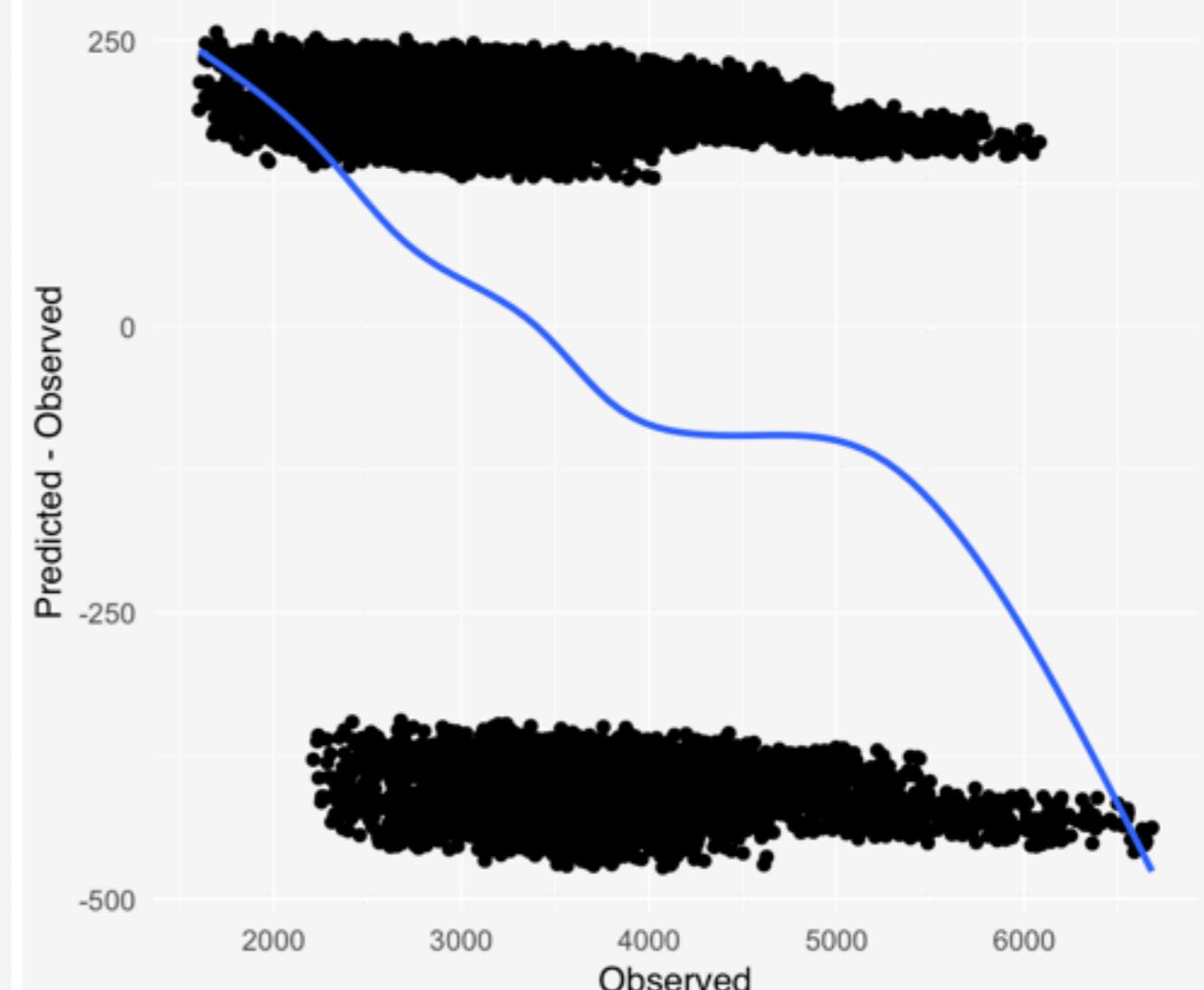
Accuracy as a single number is not enough!

Always validate your model!

Diagnostic plot for the random forest model



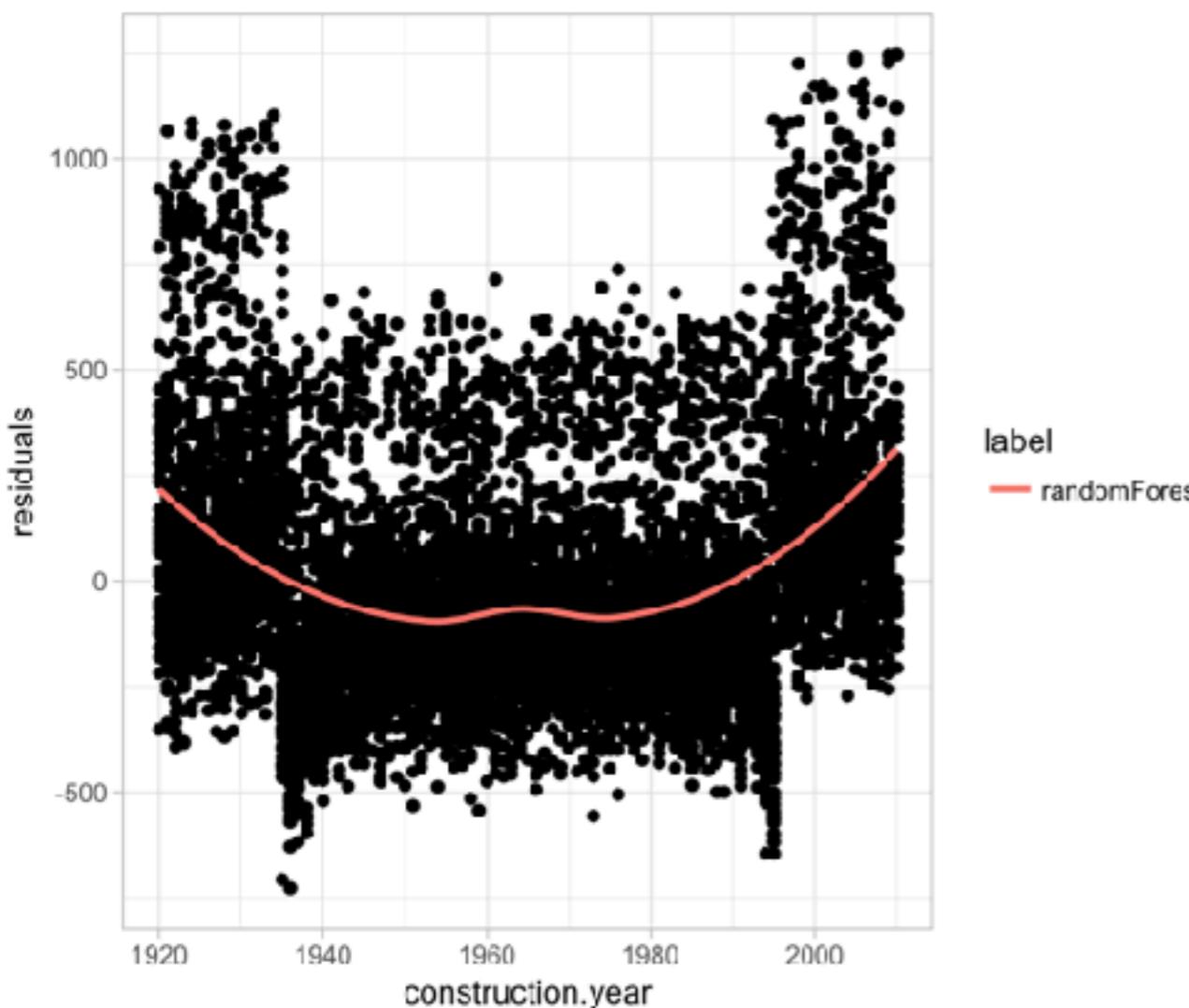
Diagnostic plot for the linear model



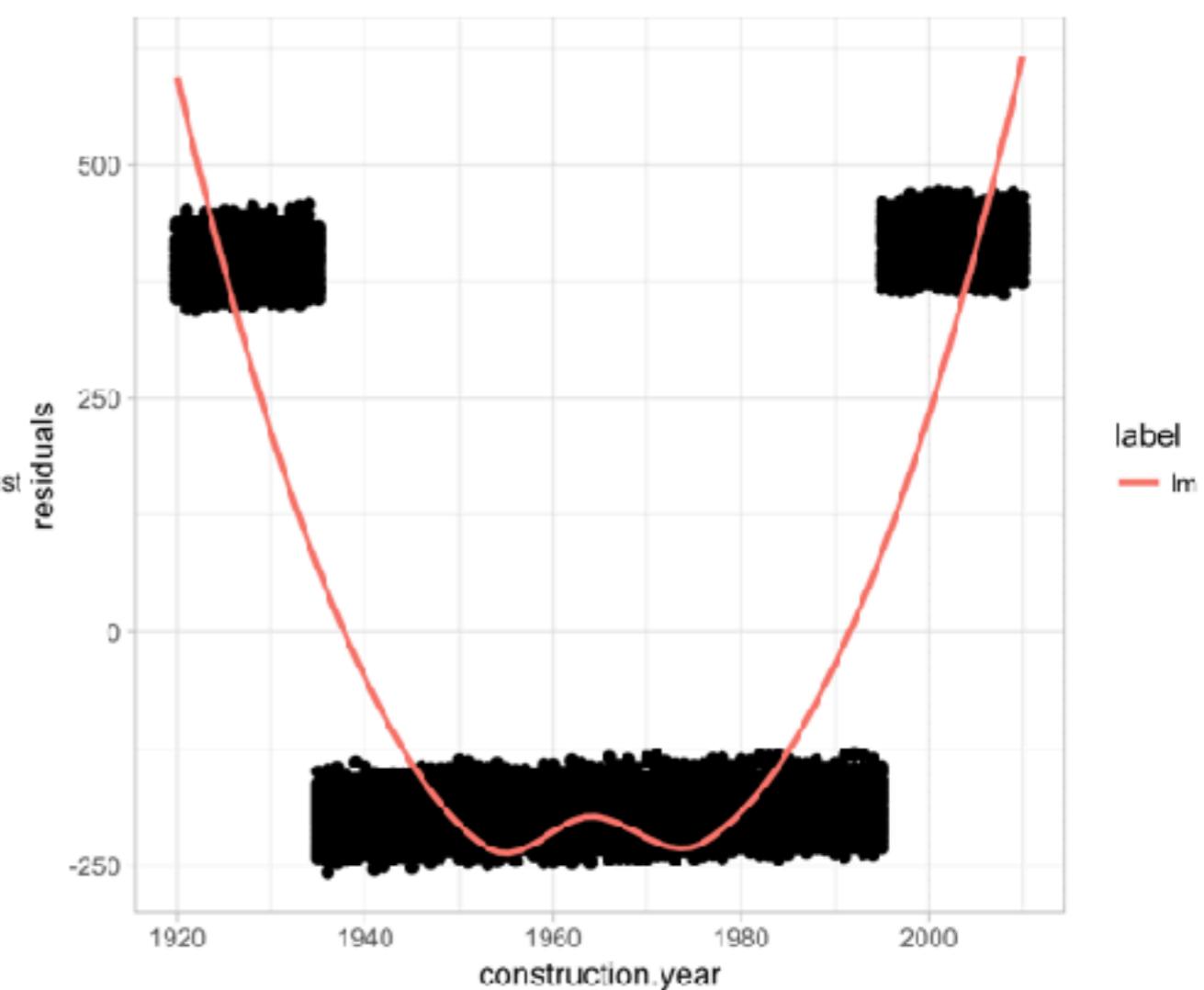
```
library(auditor)
audit_rf <- audit(explainer_rf)
plotResidual(audit_rf, variable = "construction.year")
```

```
audit_lm <- audit(explainer_lm)
plotResidual(audit_lm, variable = "construction.year")
```

Residuals vs construction.year



Residuals vs construction.year



Error analysis with auditor :: CHEAT SHEET

Basics

Package **auditor** provides several methods for model verification and validation by error analysis. This includes both, graphical methods and scores, which can be useful for comparison of models performance.

Residual analysis is widely used for linear and generalized linear models, so there are many statistical tools to evaluate the goodness-of-fit. However, these methods are not suitable for each machine learning model. In particular, it is difficult to apply them to black box models such as random forest, due to the lack of information about the error distribution.

The tools included in **auditor** can be used to assess the fit of many types of models, including black boxes.

MODEL PREPARATION

We will show the use of a package for a logistic regression model.

The example uses the *Pima Indian Diabetes* data set.

```
library(milbench)
data("PimaIndiansDiabetes")
mod_glm <- glm(diabetes~.,
  family=binomial,
  data=PimaIndiansDiabetes)
```

In order to analyse the model residuals, we need to convert model into a uniform structure readable by the **auditor** package.

```
library(auditor)
audit_glm <- audit(mod_glm)
```

An object created with the **audit()** function can be used to draw different diagnostic plots. We will show some of them in this cheatsheet.

More detailed description and additional functionalities are presented in the package vignettes which can be found on the **auditor** website: <https://ml2-warsaw.github.io/auditor>

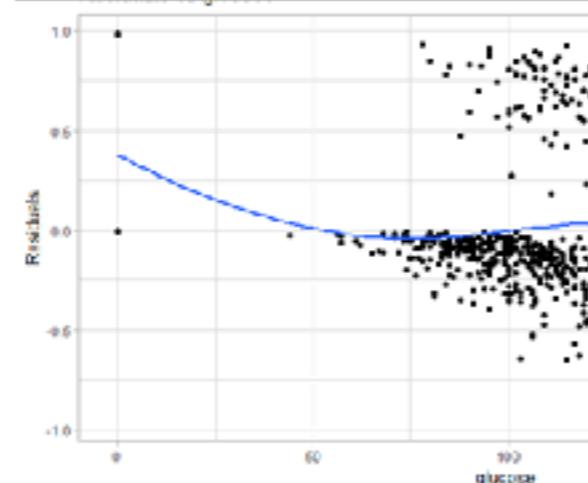
REFERENCES

- Moral, R., Linde, J., & Demétrio, C. (2017). Half-Normal Plots and Overdispersed Models in R: The *hnp* Package. *Journal of Statistical Software*, 81(10), 1–23.

Basic plots

The **plot()** function can be used to draw several di By default it is drawn Residuals vs variable plot.

```
plot(audit_glm, variable = "glucose")
```



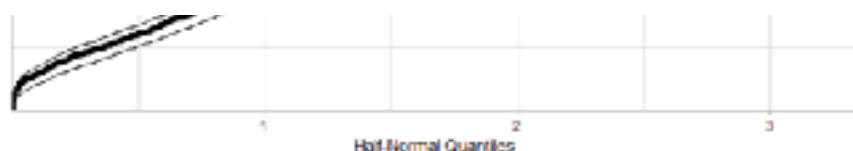
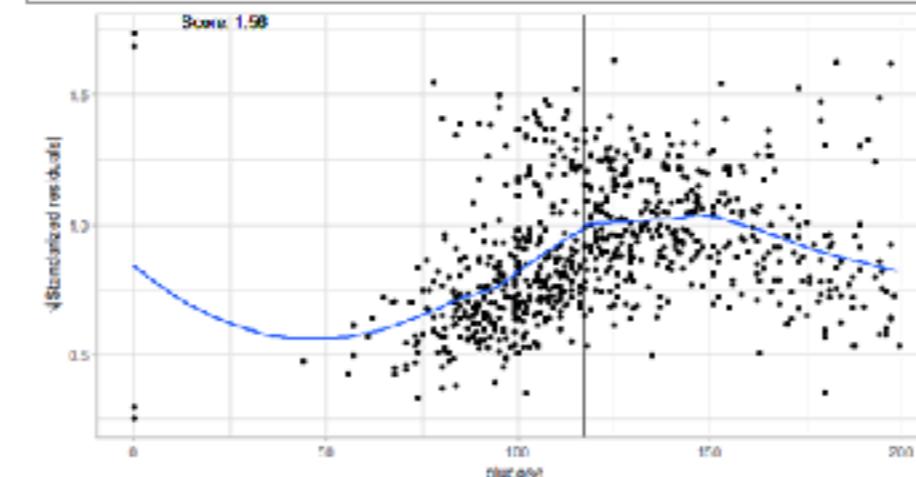
In the example above there were used response to provide any predict and residual function.

To use non-defaulted functions to calculate the residuals and make prediction, you can use arguments **residual.function** and **predict.function** while using an **audit** function.

```
p.fun <- function(model, data) {
  predict(model, data, type = "link")
}
r.fun <- function(model, y){residuals(model)}
audit_glm.dev <- audit(mod_glm,
  predict.function = p.fun,
  residual.function = r.fun)
```

Use **type=plot.name** to draw different types of diagnostic plots. The available parameter values are 'ACF', 'Autocorrelation', 'Cook', 'HalfNormal', 'Residuals' and 'ScaleLocation'.

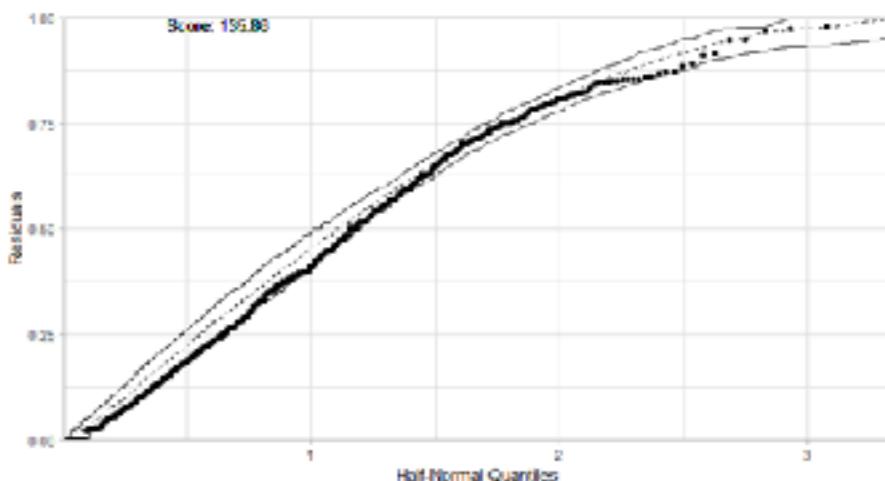
```
plot(audit_glm.dev, type="ScaleLocation",
  variable = "glucose")
```



Half-Normal plots allow to check the model fit but comparing two models may be difficult.

A useful tool to compare goodness-of-fit of two models is score displayed on the plot. Score is a sum of logarithms of estimated PDF at each point. It is calculated on the basis of simulated data, so it may differ between function calls.

```
library(randomForest)
mod_rf <- randomForest(diabetes~.,
  data=PimaIndiansDiabetes, ntree=100)
audit_rf <- audit(mod_rf)
plotHalfNormal(audit_rf)
```



Your turn!

1. Fit gbm (Generalized Boosted Regression Models) and knn (k-Nearest Neighbour) model.
2. Plot model performance explainer for all four models.

Find some help in the Workshop_eRum_2018_part1.R

Model explainers - Variable importance

Which variables are
influential?

Which variables are influential?

- Some models (additive models, random forest, etc) have built-in techniques to evaluate variable importance.
- Some of these concepts may be applied in a model agnostic fashion (for example see Fisher, Rudin, Dominici (2018)).
- It's a global view on variable importance calculated on test dataset.

Intuition

Calculate loss function for original data vs loss function for data with a single variable permuted/blinded.

$$\hat{e}_{\text{orig}}(f) := \frac{1}{n} \sum_{i=1}^n L(f, \mathbf{Z}_{[i,\cdot]}) = \frac{1}{n} \sum_{i=1}^n L\{f, (\mathbf{y}_{[i]}, \mathbf{X}_{1[i,\cdot]}, \mathbf{X}_{2[i,\cdot]})\} = \hat{\mathbb{E}}L(f, Z),$$

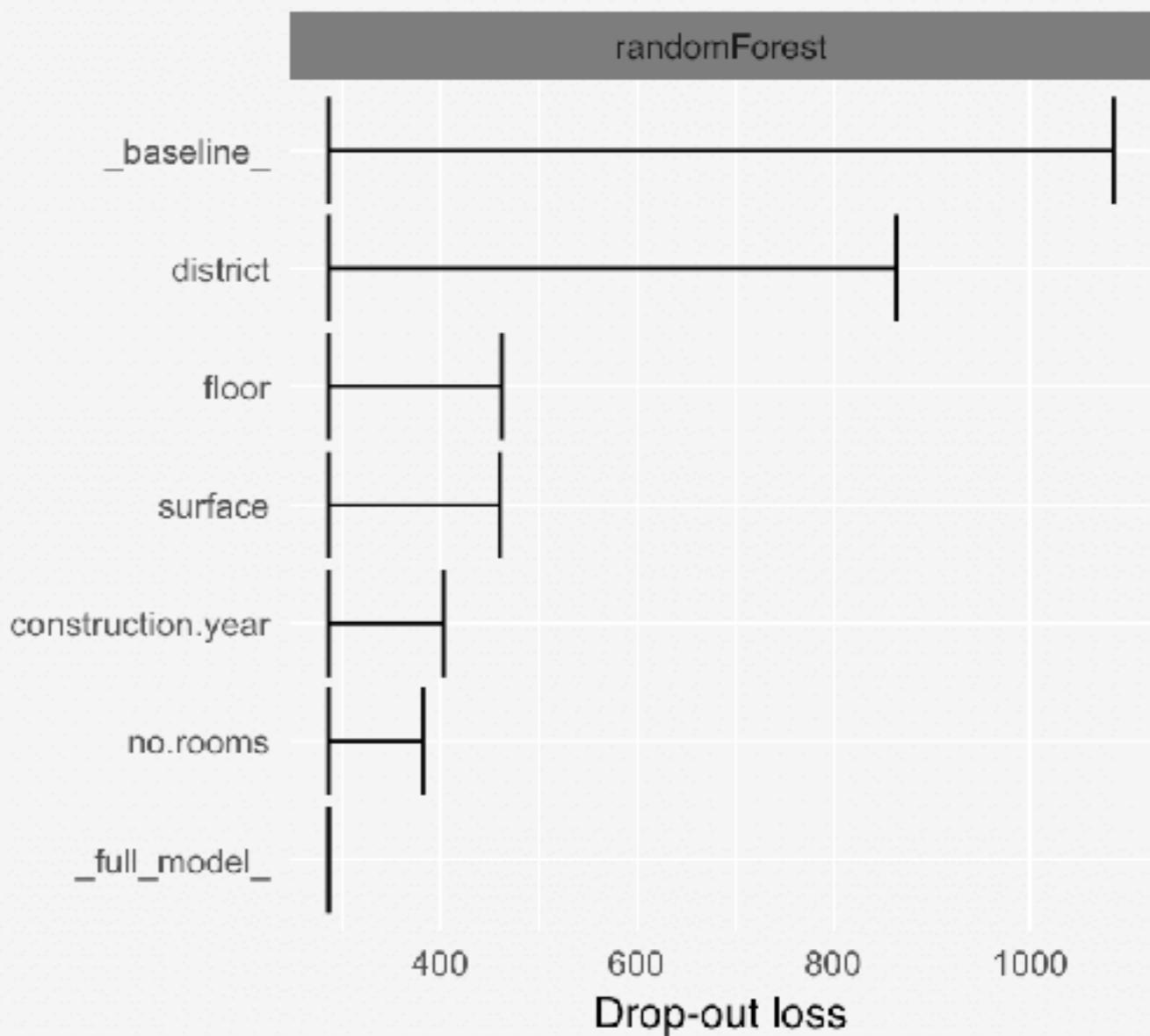
$$\begin{aligned}\hat{e}_{\text{switch}}(f) &:= \frac{1}{n(n-1)} \sum_{i=1}^n \sum_{j \neq i} h_f(\mathbf{Z}_{[i,\cdot]}, \mathbf{Z}_{[j,\cdot]}) \\ &= \frac{1}{n(n-1)} \sum_{i=1}^n \sum_{j \neq i} L\{f, (\mathbf{y}_{[j]}, \mathbf{X}_{1[i,\cdot]}, \mathbf{X}_{2[j,\cdot]})\}.\end{aligned}$$

$$\text{Model Class Reliance} = \frac{\hat{e}_{\text{switch}}(f)}{\hat{e}_{\text{orig}}(f)},$$

Fisher, Aaron, Cynthia Rudin, and Francesca Dominici. 2018. “Model Class Reliance: Variable Importance Measures for Any Machine Learning Model Class, from the ‘Rashomon’ Perspective.” Journal of Computational and Graphical Statistics <http://arxiv.org/abs/1801.01489>

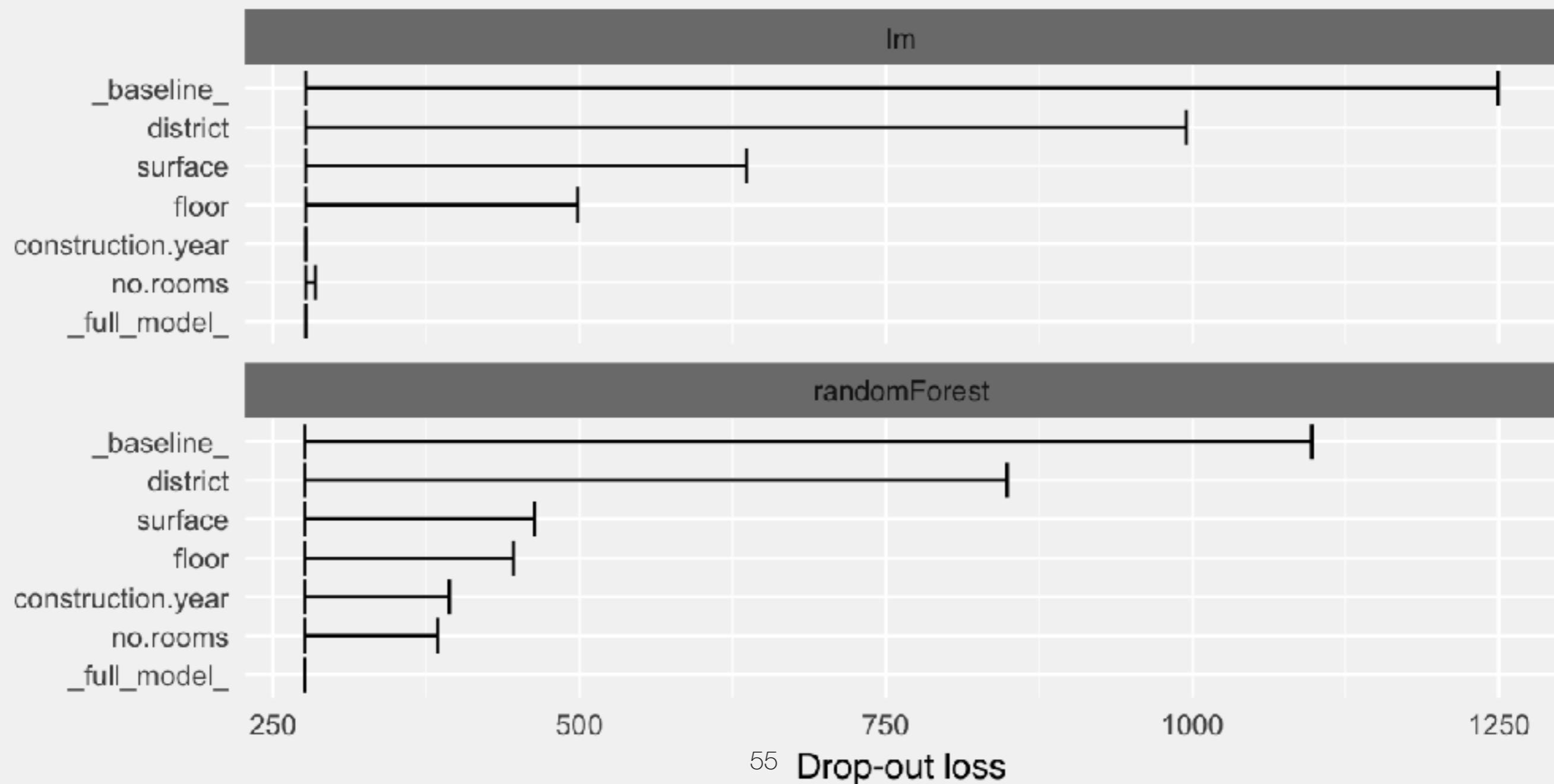
Which variables are influential for RF?

```
> vi_rf <- variable_importance(explainer_rf, loss_function = loss_root_mean_square)
> vi_rf
  variable dropout_loss      label
1 _full_model_    286.2676 randomForest
2      no.rooms   381.5975 randomForest
3 construction.year 403.4376 randomForest
4        surface  461.0018 randomForest
5         floor   462.3999 randomForest
6     district  864.4315 randomForest
7 _baseline_    1084.9218 randomForest
>
> plot(vi_rf)
```

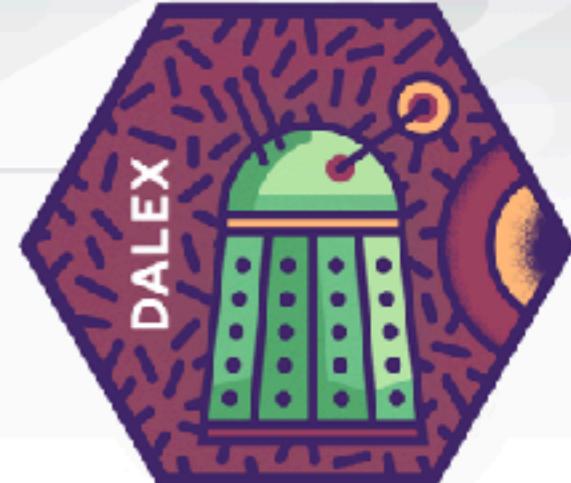


Find 5 differences!

Such statistics may be directly compared across variables and across models!



DALEX::variable_importance() - Explanations for Variable Importance



Basics

Black-box models, like random forest or extreme gradient boosting machines, are commonly used due to their high performance. They are very flexible, what often results in high accuracy.

The problem is, that due to their complicated structure it is hard to understand which variables were the most influential for a particular model prediction.

Variable Importance Explainers are designed to assess the influence of a single variable on the final model accuracy.

How does it work?

The concept is straightforward, calculate how much we will loose on accuracy if a selected variable is perturbed.

For a selected loss function the model loss is calculated for the trained model applied to the original dataset against the original target. Then the drop in model loss is calculated for each variable in the dataset. It is calculated as the loss for a model applied to a dataset with selected column being permuted.

The variable importance plots shows the initial loss of the trained model, size of the additional losses that come from permutations of selected variables and the 'baseline' which is the loss for permuted model responses.

It is useful when different model are compared, since on a single plot we see the initial model performance and also drops that come from permutations of a single variable in the selected model.

References

- Molnar, Christoph. 2018. *Interpretable Machine Learning*. <https://christophm.github.io/>
- Breiman, Leo. 2001. Random Forests. *Machine Learning* 45 (1). Springer: 5–32.
- Fisher, Aaron, Cynthia Rudin, and Francesca Dominici. 2018. *Model Class Reliance: Variable Importance Measures for any Machine Learning Model Class, from the ‘Rashomon’ Perspective*. <http://arxiv.org/abs/1801.01489>.

Use-Case

Why are our best and most experienced employees leaving prematurely? Let's see with a dataset from Kaggle Human Resources competition <https://www.kaggle.com/ludobenistant/hr-analytics/data>.

Here we are building a random forest model. The nice thing about this model is that it embeds some variable importance measure.

```
library("randomForest")
HR_rf_model <- randomForest(left~., data =
breakDown::HR_data, ntree = 100)
importance(HR_rf_model)
```

	IncNodePurity
satisfaction_level	896.584411
last_evaluation	294.195269
number_project	496.397115
average_montly_hours	398.495343

So, now let's build an explainer and use it to calculate our model agnostic variable importance measure with `variable_importance()` function.

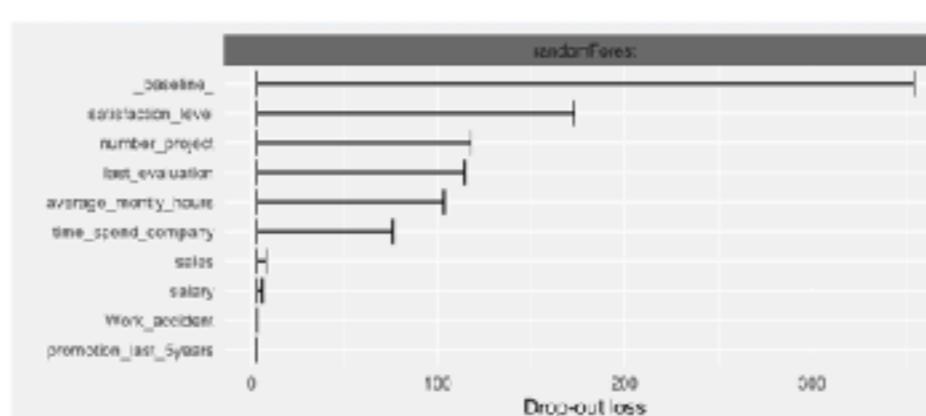
```
explainer_rf <- explain(HR_rf_model, data = HR_data,
y = HR_data$left)
vd_rf <- variable_importance(explainer_rf, type = "raw")
vd_rf
```

variable	dropout_loss	label
_baseline	335.556063	randomForest
satisfaction_level	168.001733	randomForest
last_evaluation	100.714534	randomForest
number_project	100.568214	randomForest
time_spend_company	90.349207	randomForest

Values are different because different measures of importance are being calculated, but the order is more or less the same. Now we are ready to plot the variable importance with the generic `plot()` function.

Note that in addition to `type="raw"` one can also use "`difference`" or "`ratio`" as suggested in the Fisher et al 2018 article.

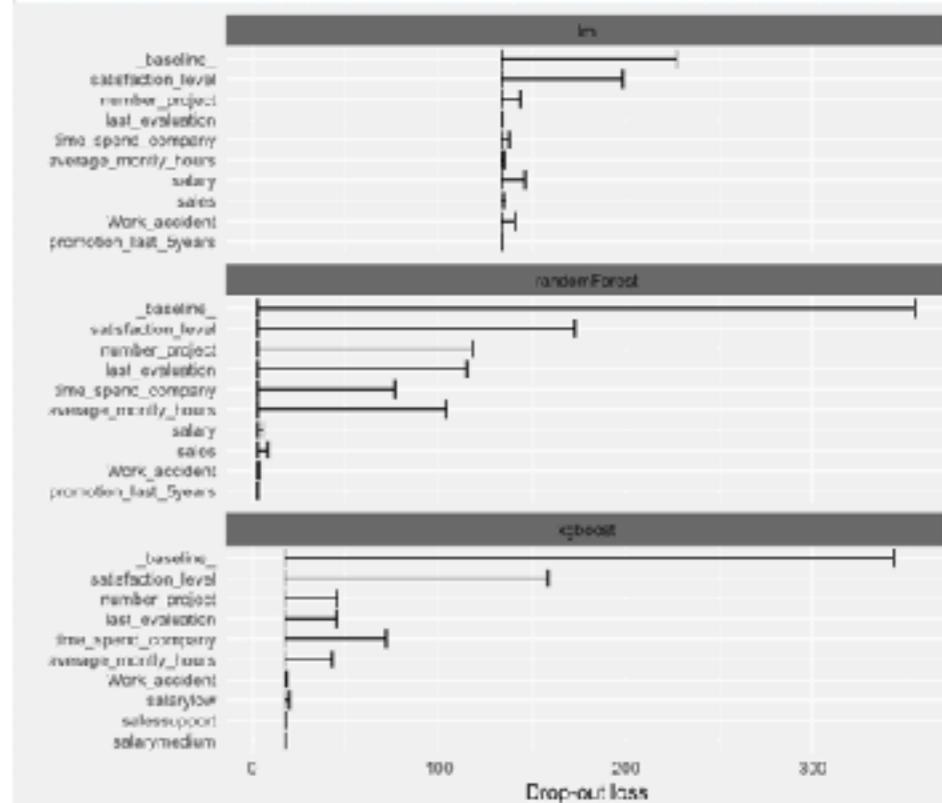
```
plot(vd_rf)
```



A very useful feature of the DALEX package is the capability to overlay responses from different models in a single plot. Below we present results for a Random Forest, GLM and XGBoost.

```
library("xgboost")
mm_train <- model.matrix(left~-1, HR_data)
data_train <- xgb.DMatrix(model_matrix_train,
label = HR_data$left)
param <- list(max_depth = 2,
objective = "binary:logistic", eval_metric = "auc")
HR_xgb <- xgb.train(param, data_train, nrounds = 50)
ex_xgb <- explain(HR_xgb, data = mm_train,
y = HR_data$left, label = "xgboost")
vd_xgb <- variable_importance(ex_xgb, type = "raw")

HR_glm <- glm(left~., data=HR_data, family = "binomial")
ex_glm <- explain(HR_glm, data=HR_data, y = HR_data$left)
logit <- function(x) exp(x)/(1+exp(x))
vd_glm <- variable_importance(ex_glm, type = "raw")
plot(vd_rf, vd_glm, vd_xgb)
```



Your turn!

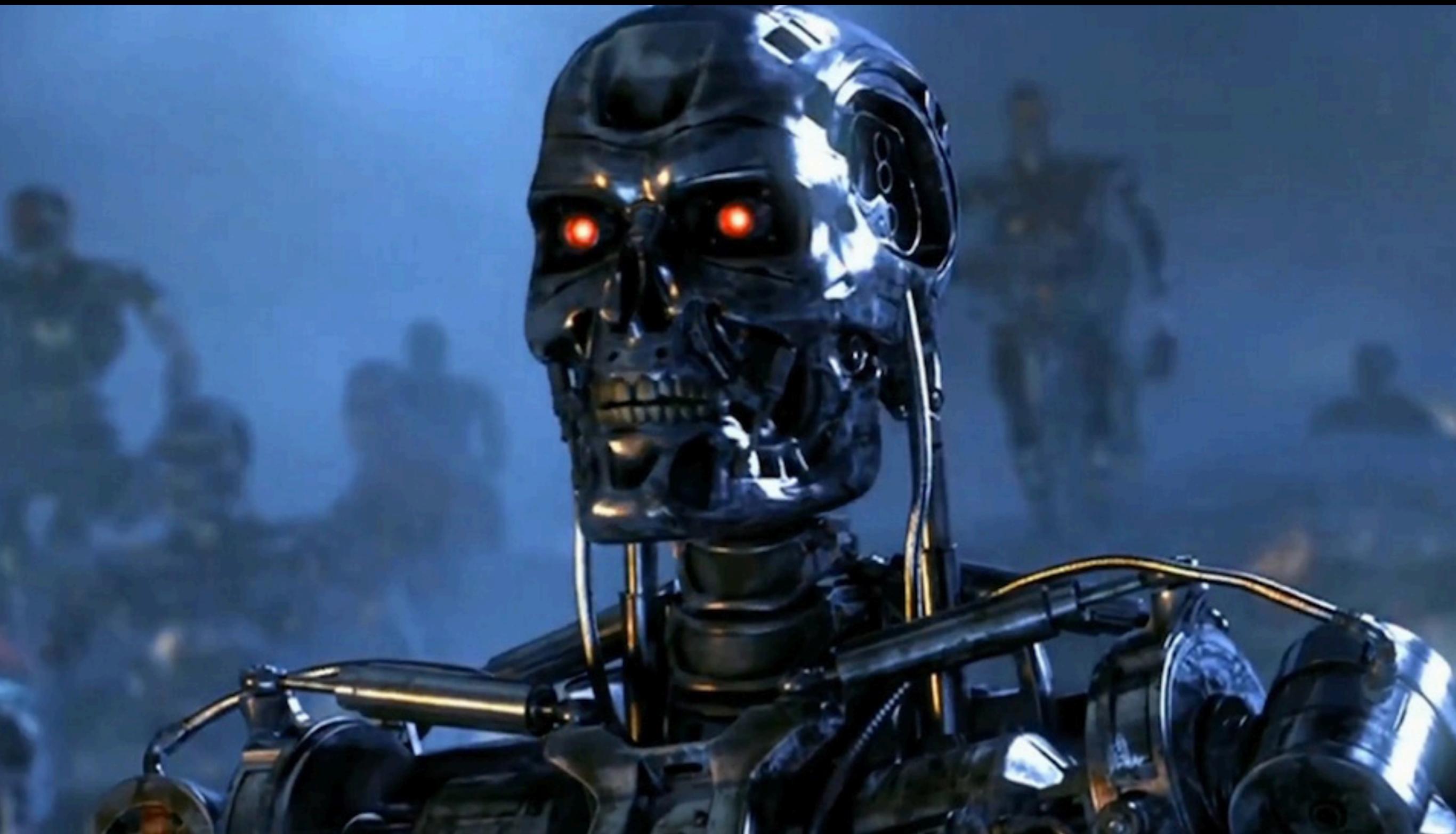
1. Fit gbm (Generalized Boosted Regression Models) and knn (k-Nearest Neighbour) model.
2. Plot variable importance explainer for all four models.

Find some help in the Workshop_eRum_2018_part1.R

DALEX:

Model - Human interface

Machine Learning Models will replace humans



Machine Learning Models will empower humans



Find more at

<https://github.com/pbiecek/DALEX>

README.md

DALEX

CRAN 0.2.0 downloads 1377 build passing coverage 95%

Descriptive mAchine Learning EXplanations

DALEX Stories

- [A gentle introduction to DALEX with examples](#)
- [How to use DALEX with caret](#)
- [How to use DALEX with mlr](#)
- [How to use DALEX with xgboost package](#)

