A simulation approach to calculating minimum sample sizes for prediction modelling

The pmsims package for R

Ewan Carr, Gordon Forbes, Diana Shamsutdinova, Daniel Stahl, and Felix Zimmer

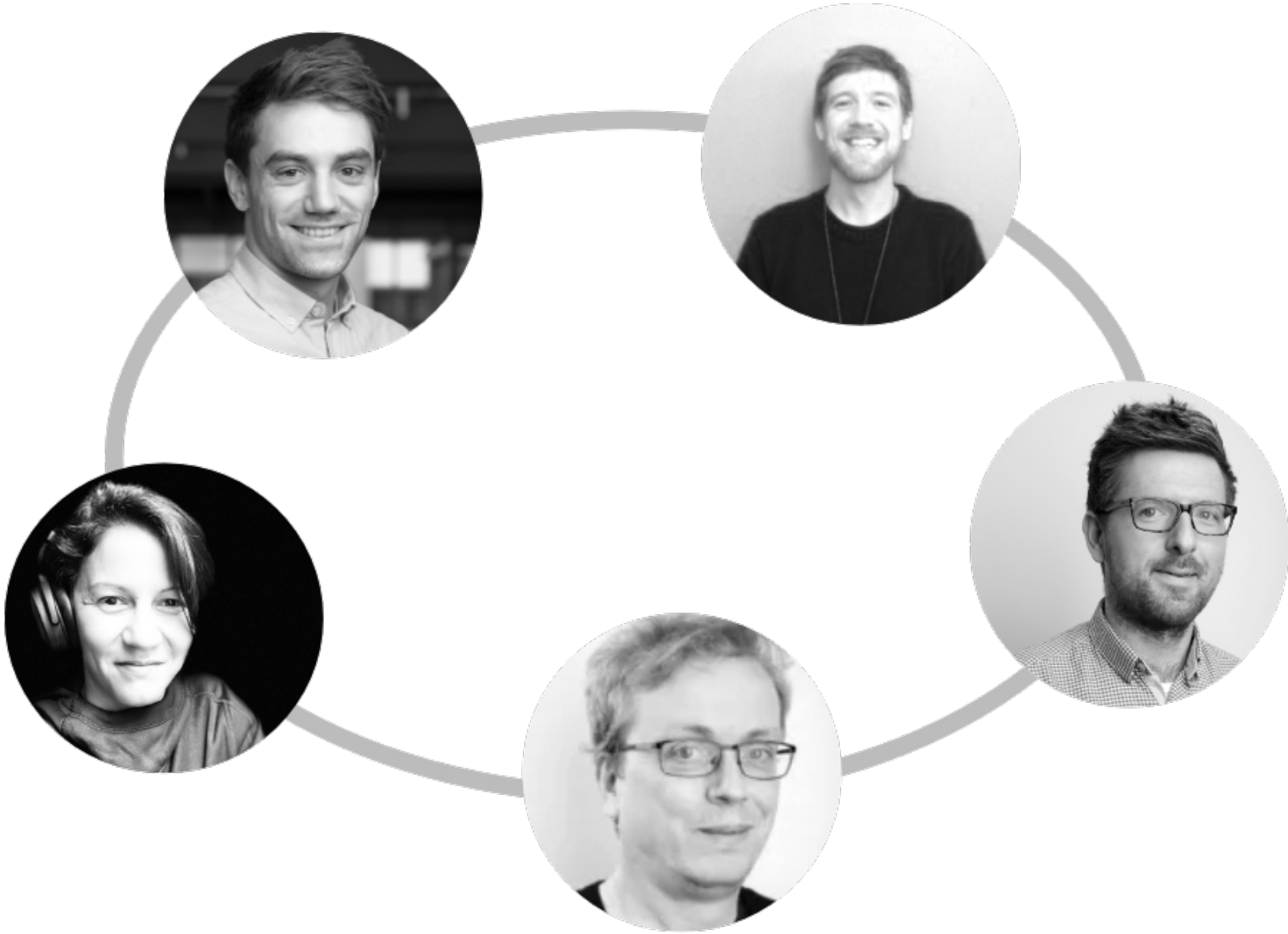
Department of Biostatistics & Health Informatics
King's College London

29th August 2023

30-second version

1. Most prediction models use small samples.
2. Small samples cause overfitting and imprecise estimates.
3. Existing tools can estimate minimum samples for continuous, binary, and survival outcomes.
4. Nothing exists for other models or data types.

We're developing a simulation-based approach that works with any outcome or method.



This talk

1. Background
   - What's the problem we're trying to solve?
   - What solutions currently exist?
2. Our simulation-based approach
   - Workflow and user interface
   - How it compares to other packages
3. Demonstration
4. Development status and next steps

This talk

1. Background
   - What's the problem we're trying to solve?
   - What solutions currently exist?
2. Our simulation-based approach
   - Workflow and user interface
   - How it compares to other packages
3. Demonstration
4. Development status and next steps

Under construction; feedback welcome.

Most models are developed with inadequate samples

- Inadequate sample sizes was the most common cause of bias in 731 models for COVID-19.[4]
- Inadequate samples have been found in:

67% models for COVID-19[4]

56% models using supervised machine learning[5]

73% models in psychiatry[7]

Inadequate samples = research waste

- Inadequate samples lead to overfitting and inaccurate estimates of model parameters.
- This may generate inappropriate decisions about patient care or lead to models not being implemented into clinical practice.
- Data collection can be invasive and inconvenient and diverts resources from other activities that benefit patients.

What tools exist?

Most studies ignore sample size.

Or use rules of thumb (e.g., 10 events per variable) that have no rationale in prediction modelling.[3]

RESEARCH ARTICLE

WILEY Statistics in Medicine

Minimum sample size for developing a multivariable prediction model: PART II - binary and time-to-event outcomes

Richard D Riley[1] | Kym IE Snell[1] | Joie Ensor[1] | Danielle L Burke[1] | Frank E Harrell Jr[2] | Karel GM Moons[3] | Gary S Collins[4]

[1]Centre for Prognosis Research, Research Institute for Primary Care and Health Sciences, Keele University, Staffordshire, UK

[2]Department of Biostatistics, Vanderbilt University School of Medicine, Nashville, Tennessee

[3]Julius Centre for Health Sciences and Primary Care, University Medical Centre Utrecht, Utrecht, The Netherlands

[4]Centre for Statistics in Medicine, Nuffield Department of Orthopaedics, Rheumatology and Musculoskeletal Sciences, University of Oxford, Oxford, UK

Correspondence
Richard D Riley, Centre for Prognosis Research, Research Institute for Primary Care and Health Sciences, Keele University, Staffordshire ST5 5BG, UK.
Email: r.riley@keele.ac.uk

When designing a study to develop a new prediction model with binary or time-to-event outcomes, researchers should ensure their sample size is adequate in terms of the number of participants (n) and outcome events (E) relative to the number of predictor parameters (p) considered for inclusion. We propose that the minimum values of n and E (and subsequently the minimum number of events per predictor parameter, EPP) should be calculated to meet the following three criteria: (i) small optimism in predictor effect estimates as defined by a global shrinkage factor of ≥ 0.9, (ii) small absolute difference of ≤ 0.05 in the model's apparent and adjusted Nagelkerke's R^2, and (iii) precise estimation of the overall risk in the population. Criteria (i) and (ii) aim to reduce overfitting conditional on a chosen p, and require prespecification of the model's anticipated Cox-Snell R^2, which we show can be obtained from previous studies. The values of n and E that meet all three criteria provides the minimum sample size required for model development. Upon application of our approach, a new diagnostic model for Chagas disease requires an EPP of at least 4.8 and a new prognostic model for recurrent venous thromboembolism requires an EPP of at least 23. This reinforces why rules of thumb (eg, 10 EPP) should be avoided. Researchers might additionally ensure the sample size gives precise estimates of

In 2018, Riley et al. released pmsampsize for R and Stata.

We ❤️ pmsampsize, but. . .

We increasingly need to estimate minimum samples for:

- **Other models** (e.g., machine learning algorithms, random forests, gradient boosting);
- **Other data types** (e.g., longitudinal, clustered)

We're creating a **simulation-based framework** to estimate sample sizes for prediction.

The pmsims package for R

Flexible	Any model or data type
User-friendly	Defaults for common scenarios
Efficent	Estimation via surrogate modelling

Our approach

Setting

1. A **study population** represented by outcome-related individual characteristics (i.e., candidate predictors).
2. A chosen statistical or machine learning **model**.
3. Expected achievable **large-sample performance**, P^*, given population and model.
4. Minimum **acceptable test performance** of the model, P^{OK}.

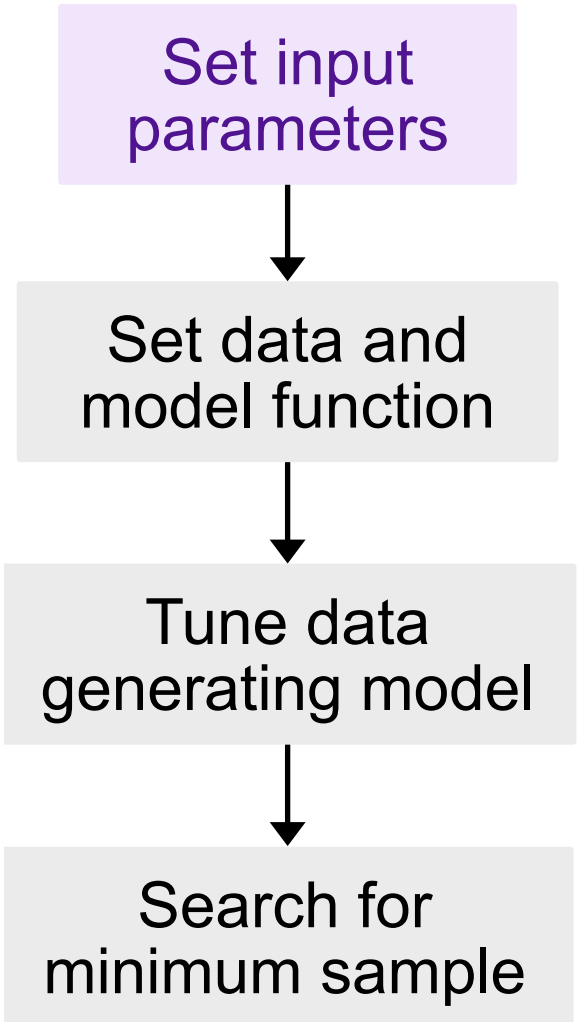
Our approach

Setting

1. A **study population** represented by outcome-related individual characteristics (i.e., candidate predictors).
2. A chosen statistical or machine learning **model**.
3. Expected achievable **large-sample performance**, P^*, given population and model.
4. Minimum **acceptable test performance** of the model, P^{OK}.

Find the minimum sample that ensures test performance of P^{OK} with probability of 80%, given the population, predictors, and P^*.

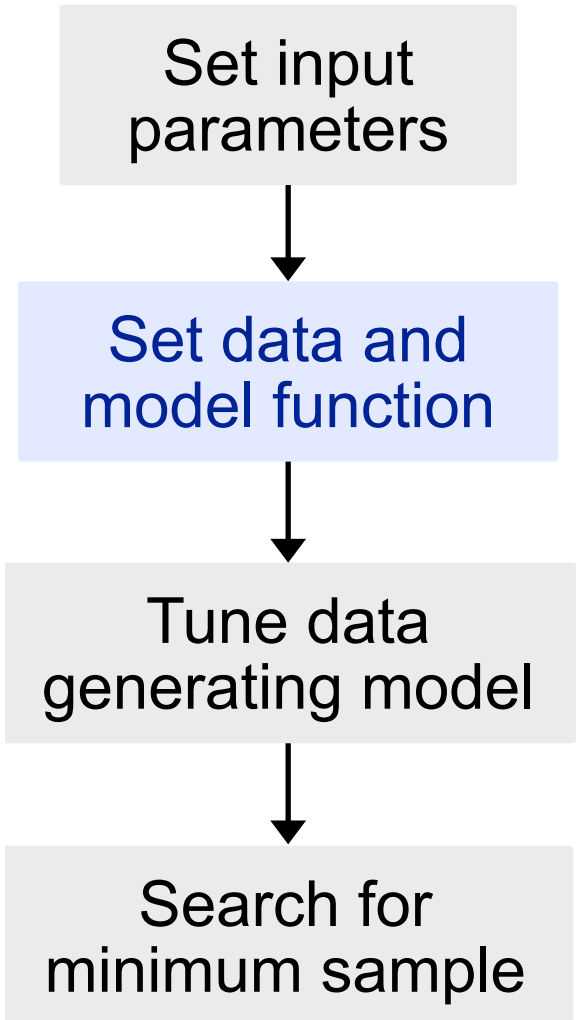
How does it work?



The user specifies:

1. The candidate predictors (number, type)
2. The chosen statistical model
3. The expected large sample performance (P^*)
4. The minimum acceptable performance (P^{OK})

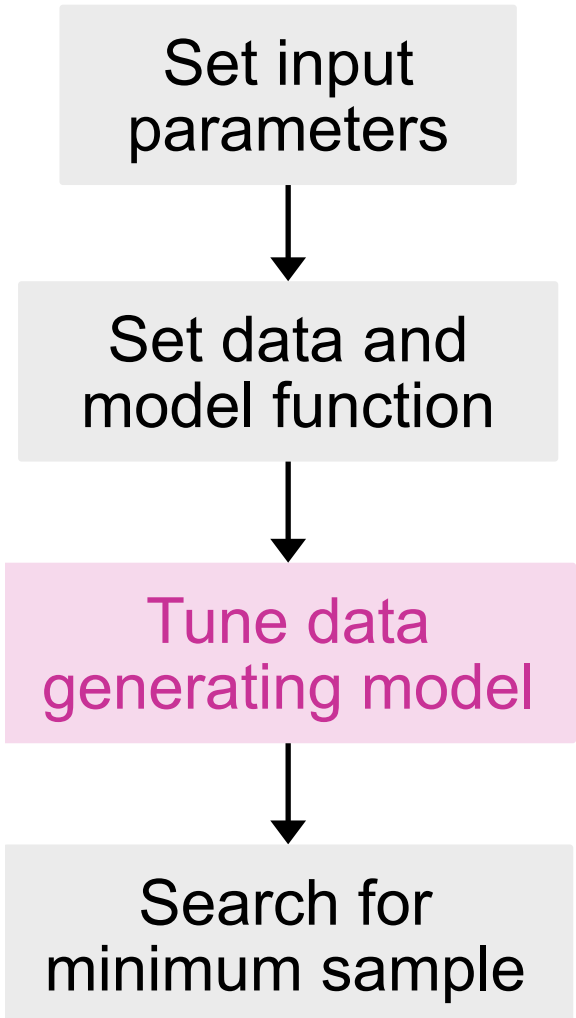
How does it work?



Based on their input, we set:

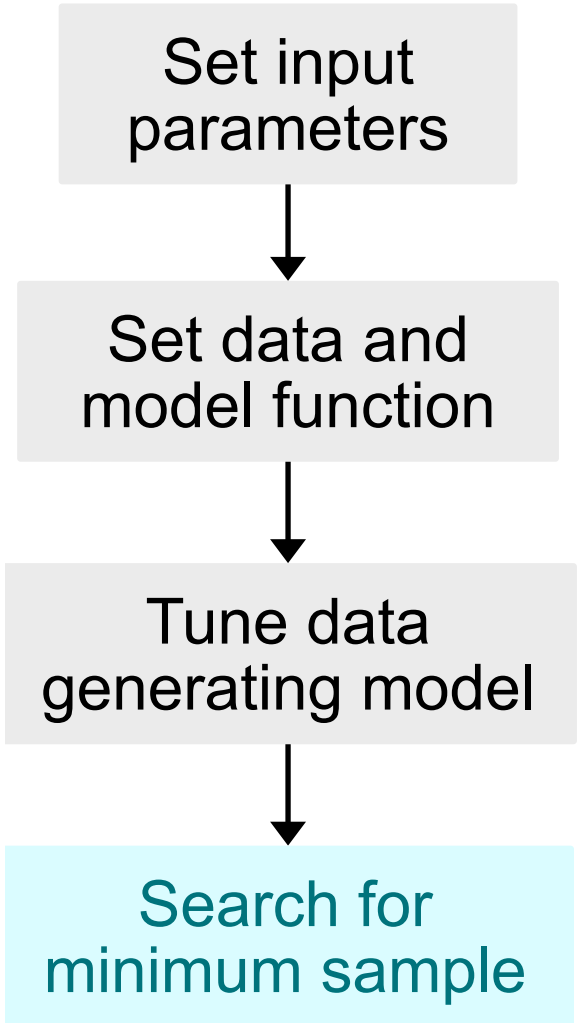
1. A data-generating function
2. A model function
3. A metric function

How does it work?



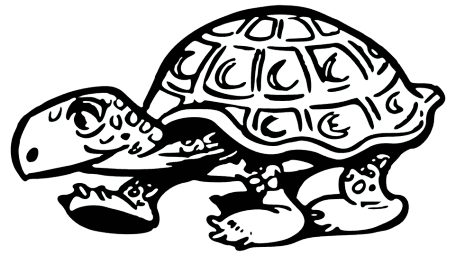
We tune the data generating model, so the large sample performance is P^*.

How does it work?



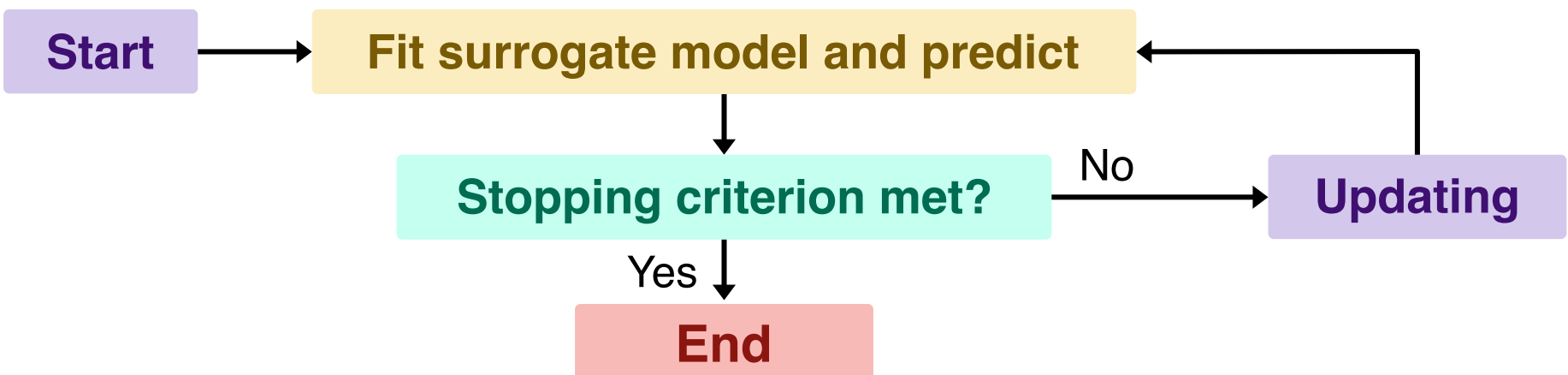
Performing the search

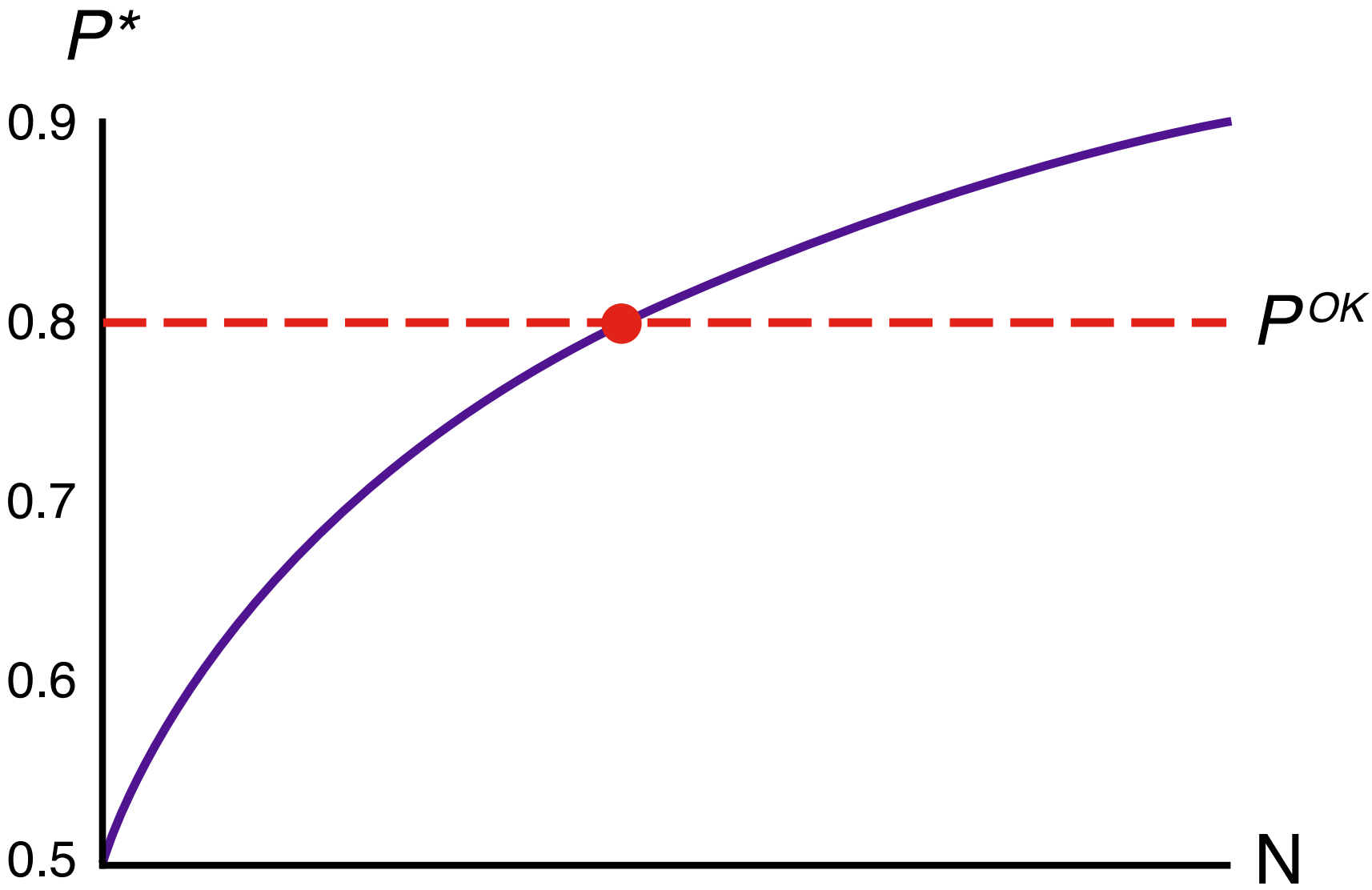
An exhaustive grid search would be too slow.

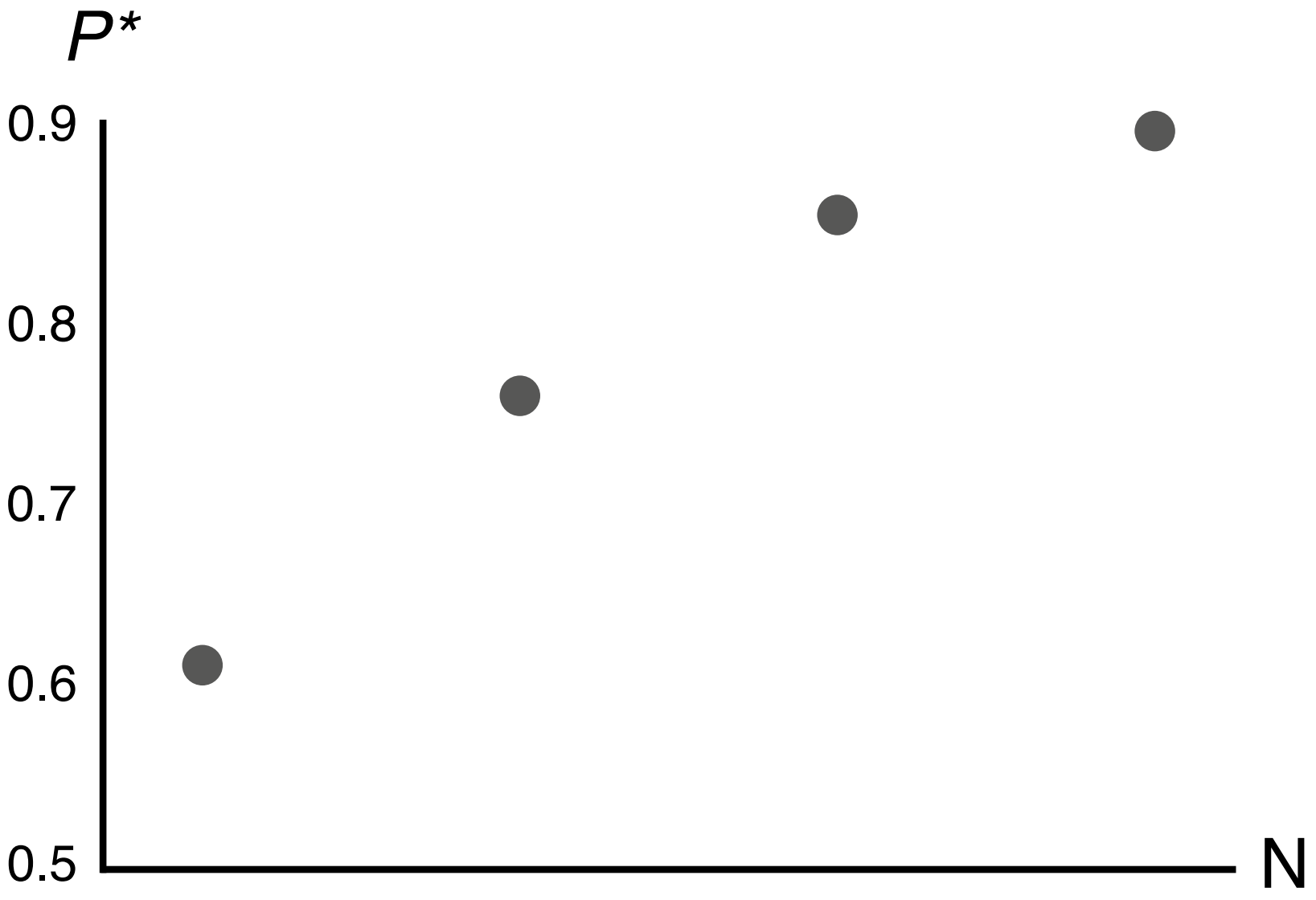


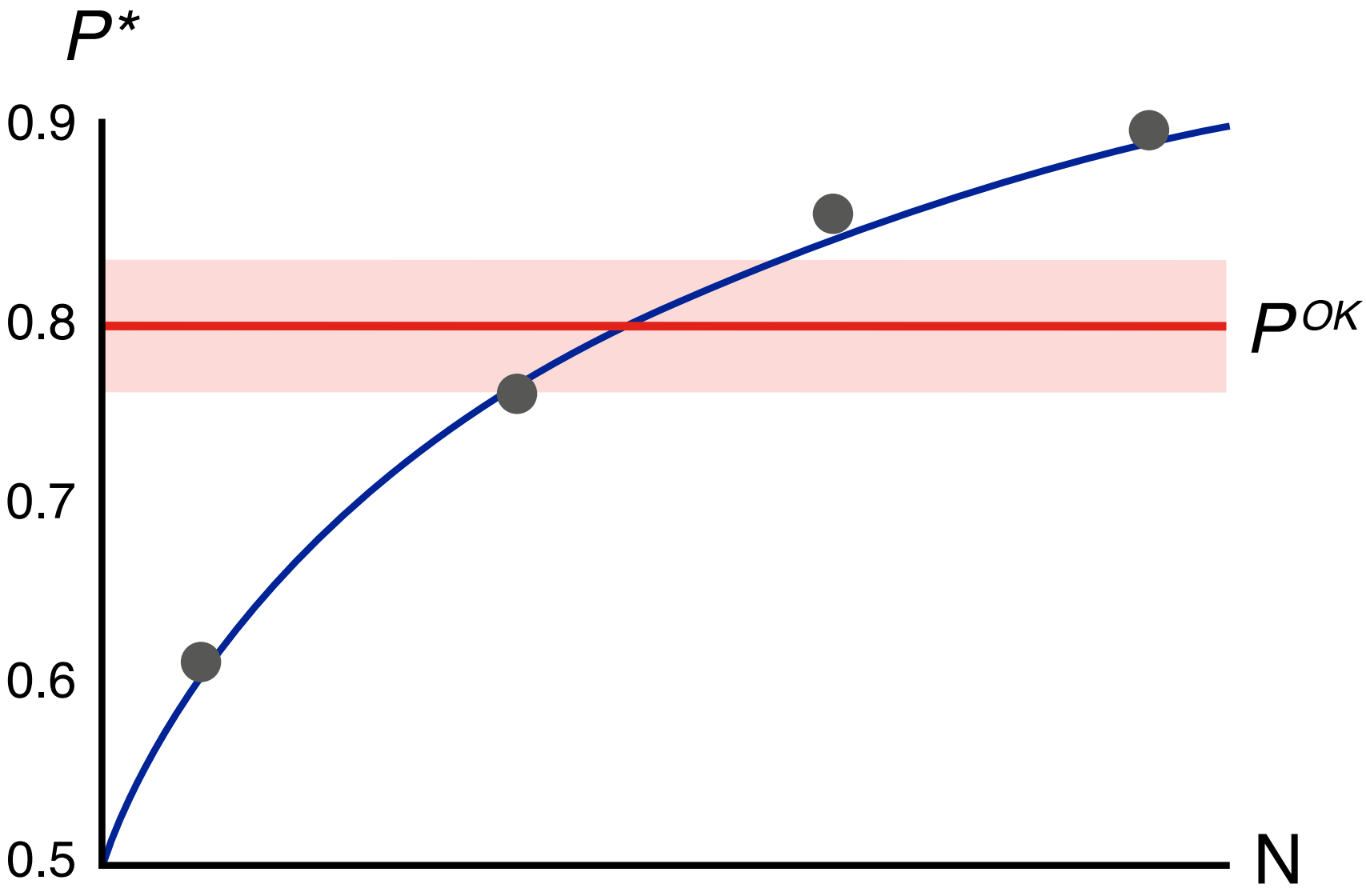
Surrogate modelling with mlpwr

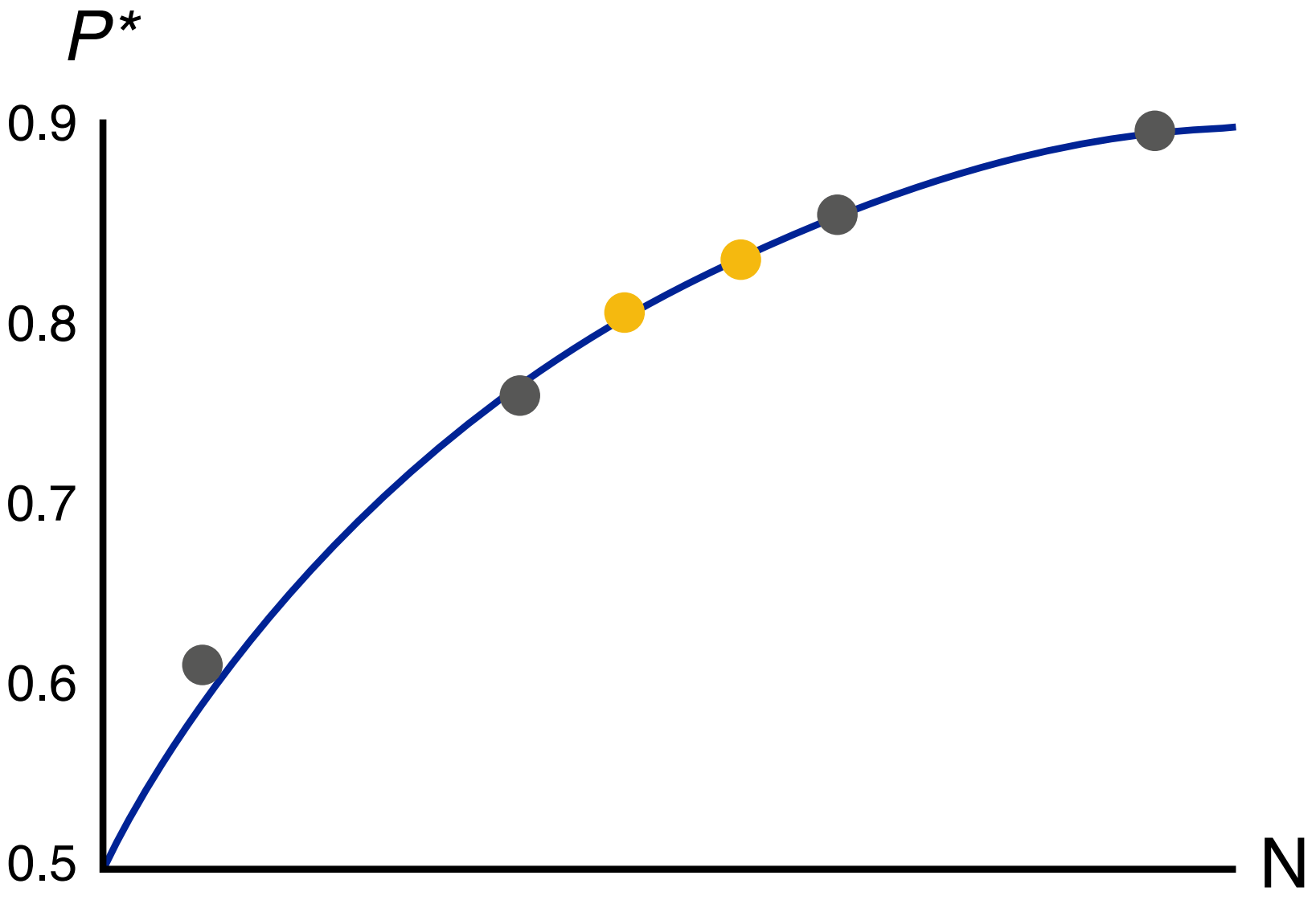
- Approximates the relationship between sample size and P^{OK} using Gaussian process regression.
- Also referred to as 'learning curve fitting'.[2,8]
- Uses the mlpwr R package by Zimmer and Debelak.[9]

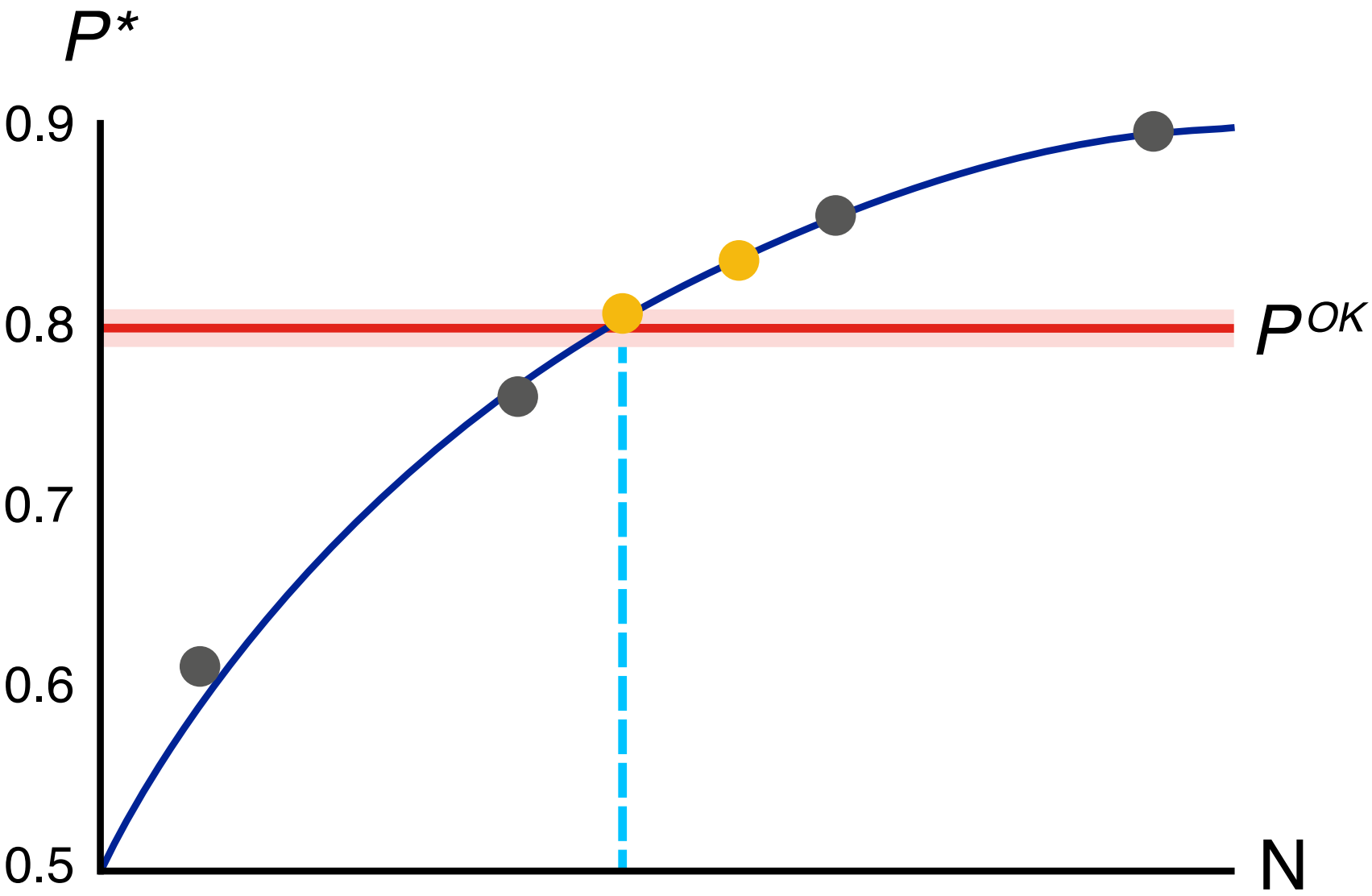



P*
0.9
0.8
0.7
0.6
0.5
P^OK
N

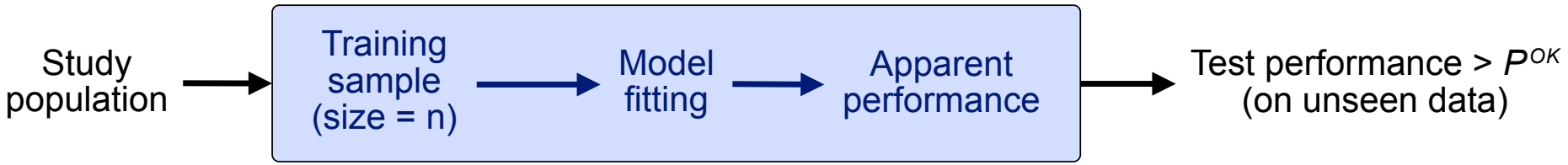

P*
0.9
0.8
0.7
0.6
0.5
N


P*
0.9
0.8
0.7
0.6
0.5
P OK
N


P*
0.9
0.8
0.7
0.6
0.5
N

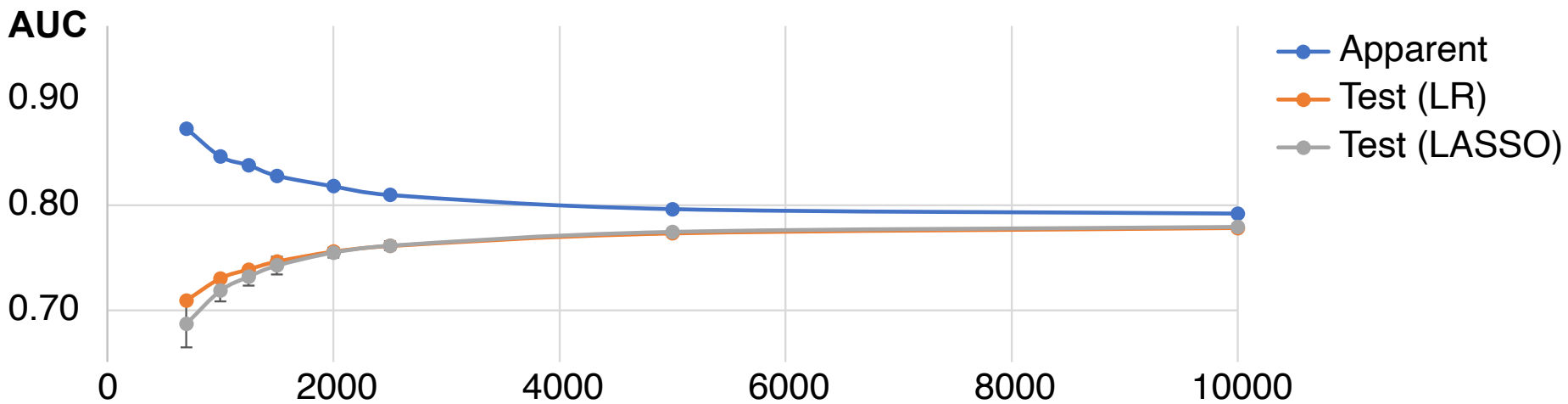

P*
0.9
0.8
0.7
0.6
0.5
P OK
N

What is the performance of a prediction model?



Apparent vs. test performance (or "actual" performance)

- Train/test performances are random variables of the drawn sample.
- Test performance is expected to be worse than apparent; but difference reduced with higher n.
- A prediction model is as good as its test performance.



How do we assess performance?

We identify the minimum sample that meets three criteria:

1. Overall fit

Within 0.1 of the achievable large sample fit (e.g., R^2, Brier).

2. Discrimination

Within 0.1 of the achievable large sample discrimination (e.g., C-statistic, AUC).

3. Calibration slope

A calibration slope of 0.9 to 1.1.

The threshold and metric are user-configurable.

Two approaches to estimating minimum samples

	pmsims	pmsampsize
Target?	Minimum development sample that ensures expected apparent and test performances are sufficiently close.	Minimum development sample that ensures test performance of P^{OK} with 80% probability.
How?	Simulate absolute test performance	Analytical closeness of test vs. train performance

- Tune data generator to an expected achievable performance.
- For training data of different sizes, use mlpwr to find n at which 80% of test performances achieves P^{OK}.
- Calibration slope criterion is similar to uniform shrinkage criterion (slope is defined as minimizing the error between y^{test} and $\alpha + slope \times \hat{y}^{test}$).

- Targets small train-test difference in R^2; or uniform shrinkage above given threshold (e.g., 0.9).
- Uniform shrinkage: considers GLM models, where estimates depend on a linear predictor, $x^T\hat{\beta}$, with $\hat{\beta}-$ OLS/ML estimates from the training sample.
- Using $s \cdot x^T\hat{\beta}$, instead of $x^T\hat{\beta}$ may prevent overfitting and perform better on unseen cases.

What are the distinctive features of these approaches?

pmsims

- Targets absolute performance
- Flexible
- Does not aim to prevent overfitting per se
- Targets test performance*
- Adjusts recommendations for test performance variance**

However:

- Computationally demanding
- User must specify data/model for complex designs
- Simulation variability

pmsampsize

- Fast, closed-form solutions
- Ensures sufficient training sample to prevent overfitting

However:

- Closed-form only for some models.[6]
- Does not adjust predictions to the variance of the test performance.[1]
- As only one training sample will be available to the model developers, actual performance once deployed may be much lower**.

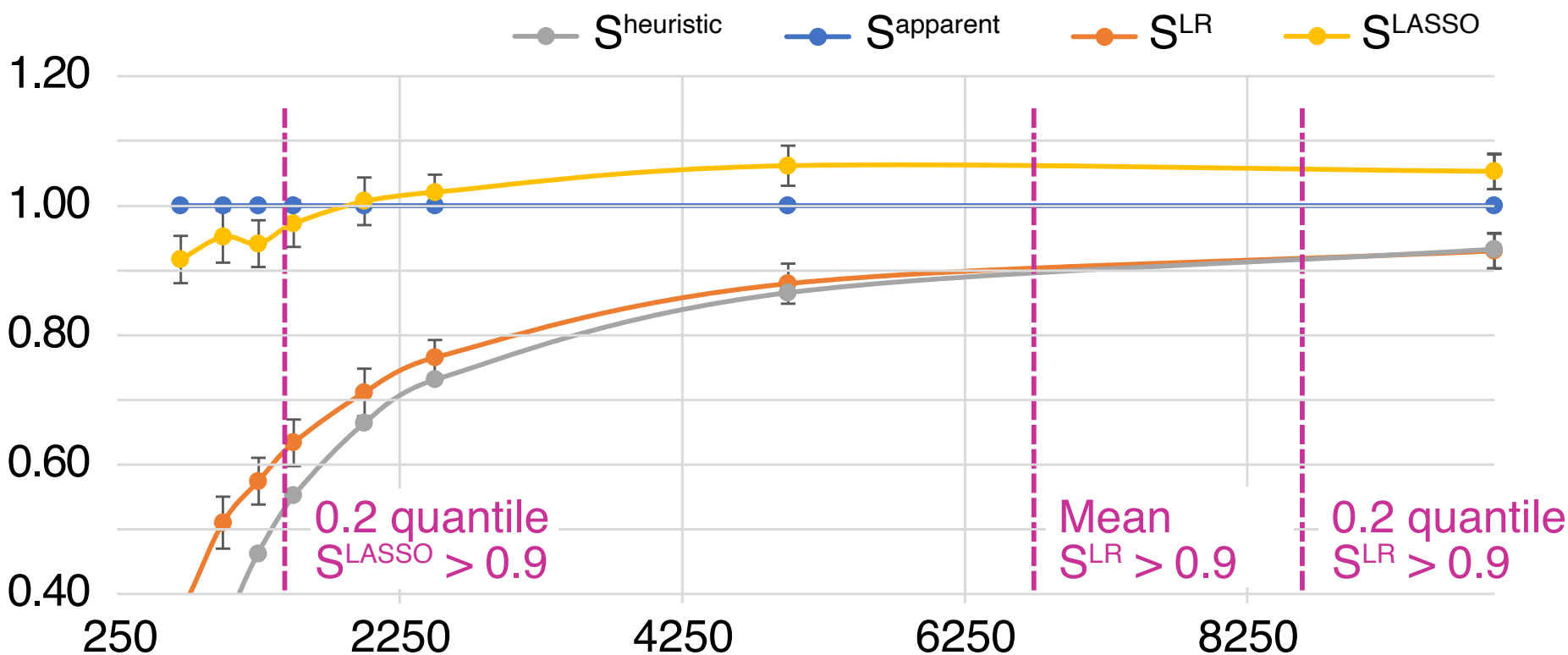
Compared to pmsampsize, our approach will require:

Smaller N for machine learning models:

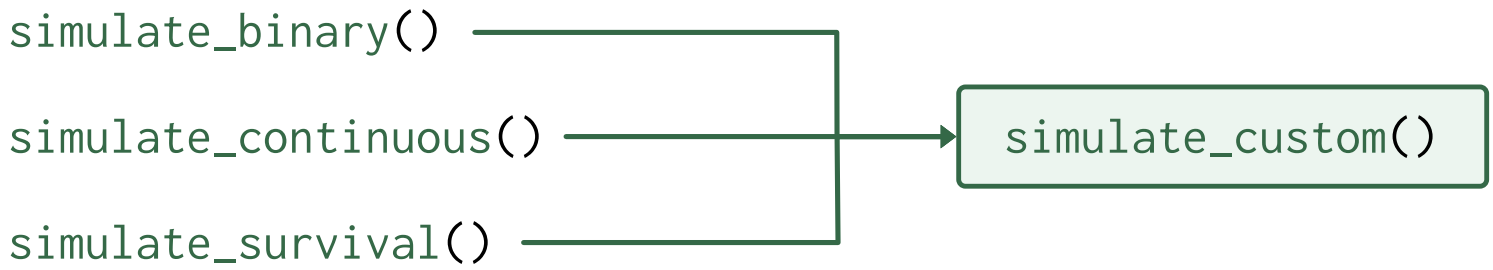
- Tend to overfit but may still achieve high test performance.
- Targeting shrinkage requires ↑N than performance.

Larger N for noisy data and models with high variance:

- 0.2 quantile test performance < mean performance.



The user interface

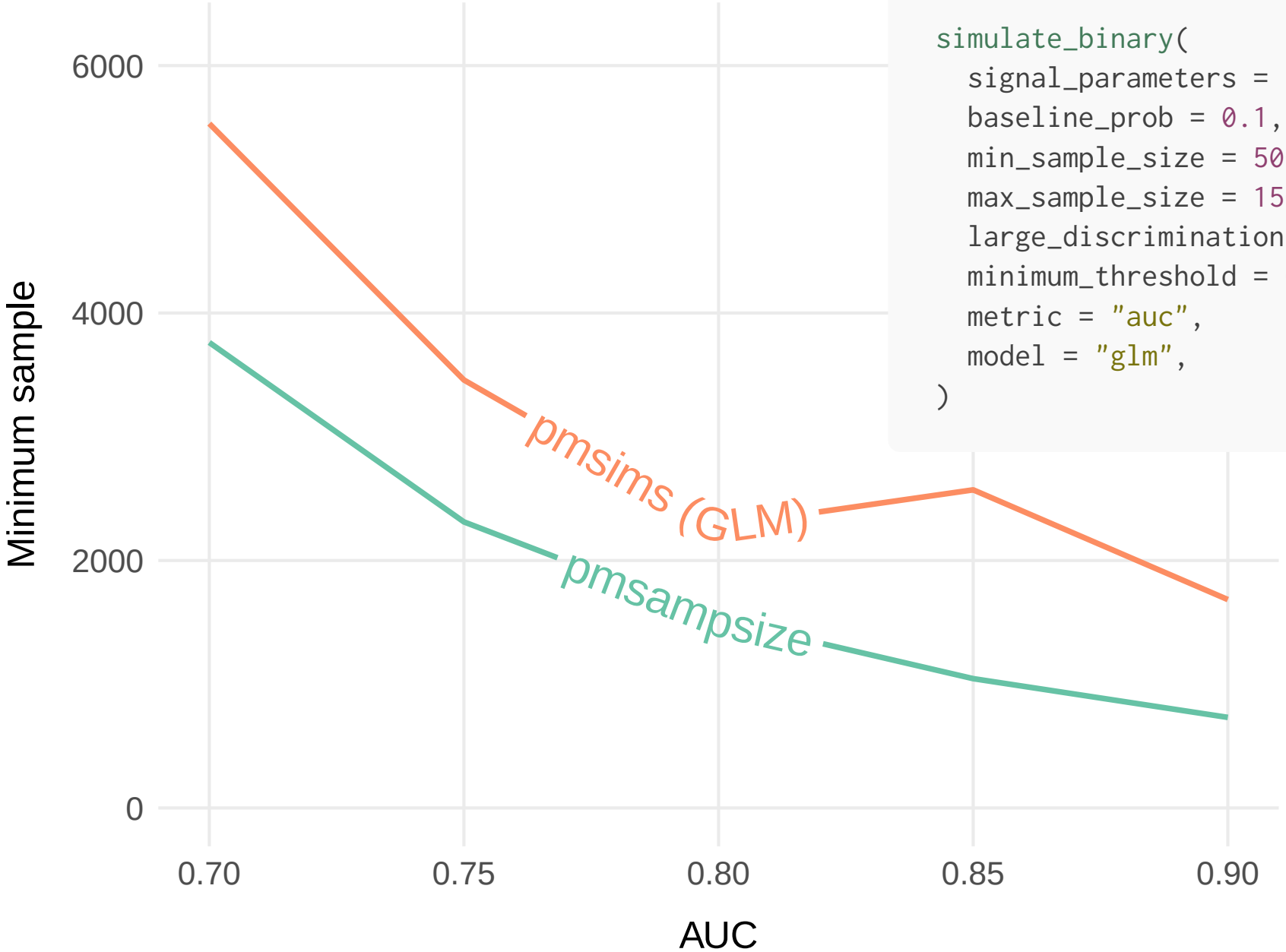



```
simulate_continuous <-
  function(
    signal_parameters = 30,
    noise_parameters = 0,
    min_sample_size = 300,
    max_sample_size = 10000,
    large_discrimination = 0.7,
    minimum_threshold = 0.1,
    model = "lm",
    metric = "r2",
    ...
  )
```

```
simulate_binary <-
  function(
    signal_parameters = 30,
    noise_parameters = 0,
    baseline_prob = 0.1,
    min_sample_size = 300,
    max_sample_size = 10000,
    large_discrimination = 0.8,
    minimum_threshold = 0.1,
    metric = "auc",
    model = "glm",
    ...
  )
```

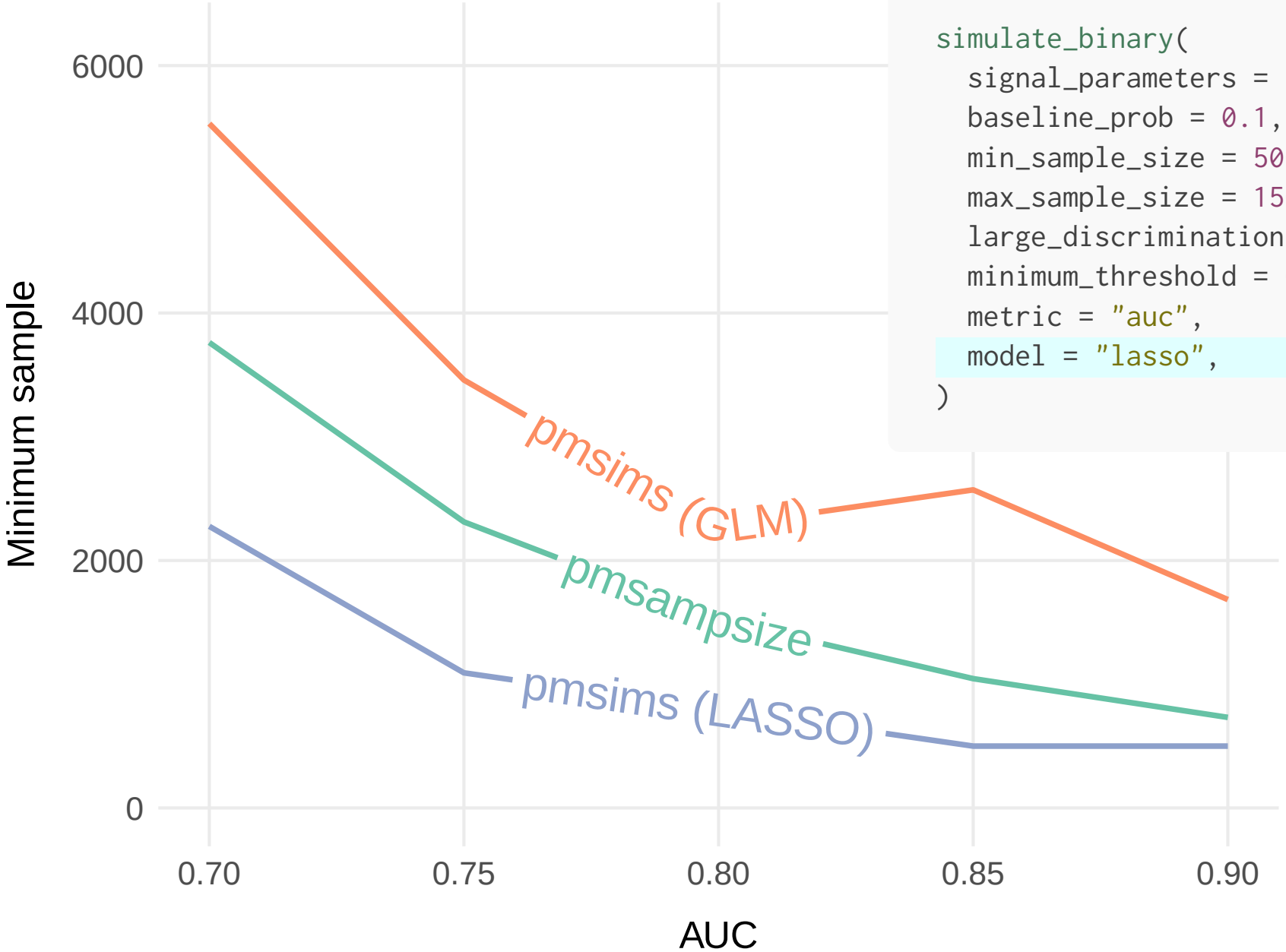
Example 1: Binary outcome, logistic regression

```
simulate_binary(
  signal_parameters = 20,
  baseline_prob = 0.1,
  min_sample_size = 500,
  max_sample_size = 15000,
  large_discrimination = ...,
  minimum_threshold = 0.1,
  metric = "auc",
  model = "glm",
)
```



Example 2: Binary outcome, LASSO regression

```
simulate_binary(
  signal_parameters = 20,
  baseline_prob = 0.1,
  min_sample_size = 500,
  max_sample_size = 15000,
  large_discrimination = ...,
  minimum_threshold = 0.1,
  metric = "auc",
  model = "lasso",
)
```



Example 3: Custom model function

What if a model hasn't been implemented?

```
model_function <- function(d) {
  dmat <- xgboost::xgb.DMatrix(
    as.matrix(d[, -1]),
    label = d[, 1]
  )
  param <- list(
    objective = "binary:logistic",
    booster = "gblinear",
    alpha = 0.0001,
    lambda = 1
  )
  xgboost::xgb.train(
    param,
    dmat,
    nrounds = 2
  )
}
```

```
metric_function <- function(data,
                            fit,
                            model) {
  dmat <- xgboost::xgb.DMatrix(
    as.matrix(data[, -1]),
    label = data[, 1]
  )
  y_hat <- predict(fit, dmat)
  pROC::auc(data[, 1], y_hat)[1]
}
```

```
simulate_custom(
  data_function = data_function,
  model_function = model_function,
  metric_function = metric_function,
  ...
)
```

Development status

We're still developing the package.

fediscience.org/@ewan for updates

Enter email at tinyurl.com/is-pmsims-ready-yet to get one email when a public release is available.

Come and talk to us.

Next steps:

Next steps

1. Machine learning
2. Longitudinal data
3. Common data types
4. Performance

What's next?

1. Machine learning

2. Longitudinal data

3. Common data types

e.g., clinical, NLP, genetic.

4. Performance