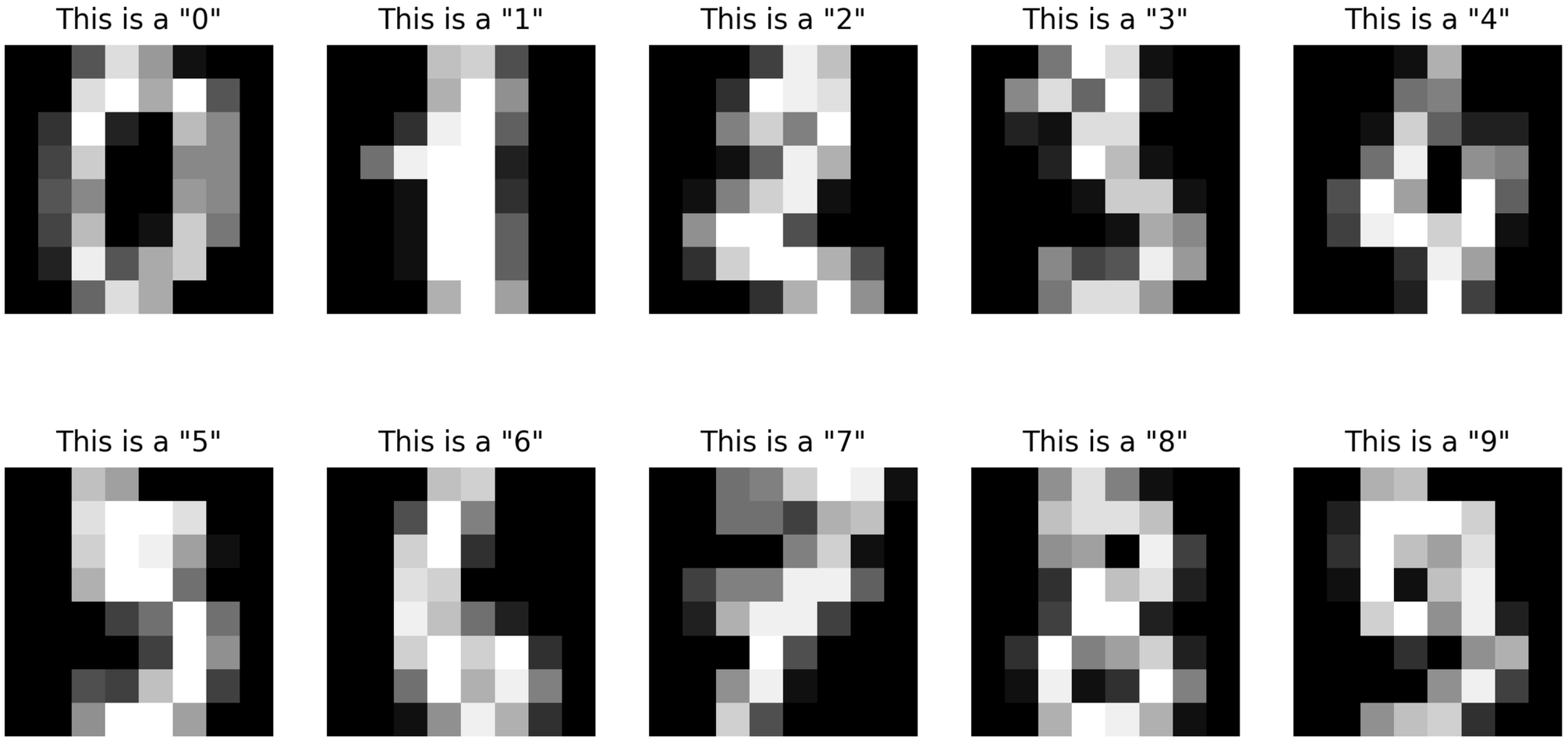
Reproducible Modelling

- practical tutorial -

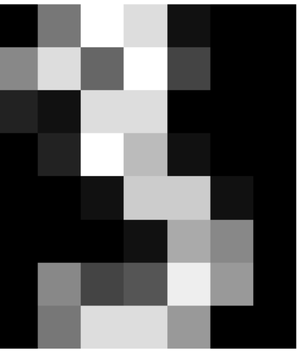
Armin Thomas @CORES 2021 fall lecture series

“Optical Recognition of Handwritten Digits Data Set”

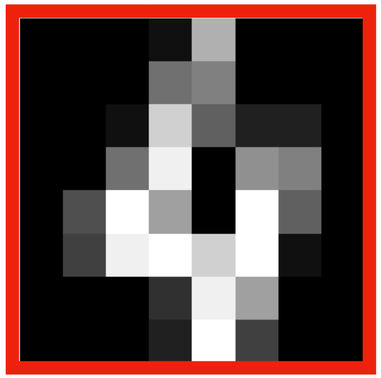


https://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits

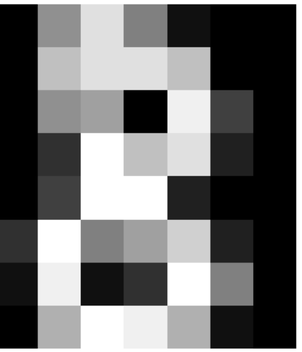
This is a "3"



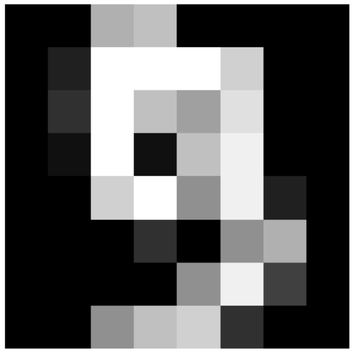
This is a "4"



This is a "8"



This is a "9"



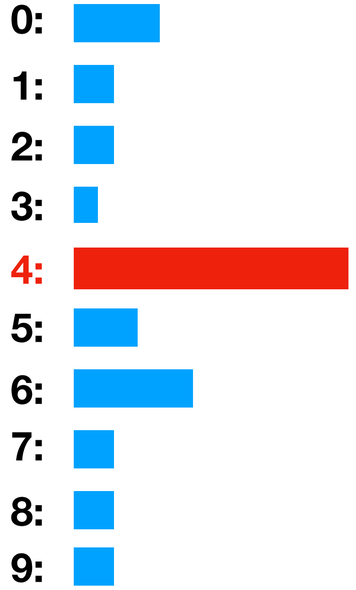
input

Predictive model

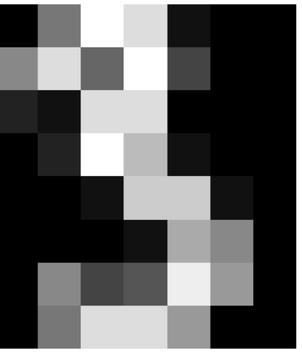
$f(\cdot)$

output

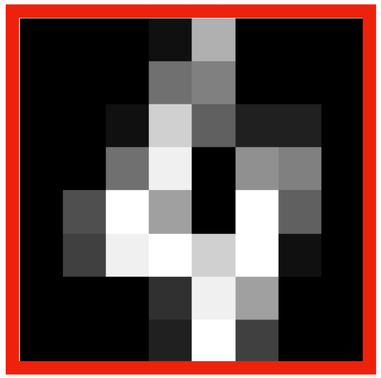
$f(x)$



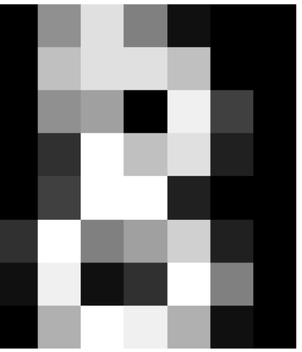
This is a "3"



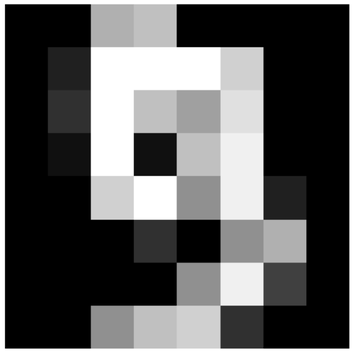
This is a "4"



This is a "8"



This is a "9"

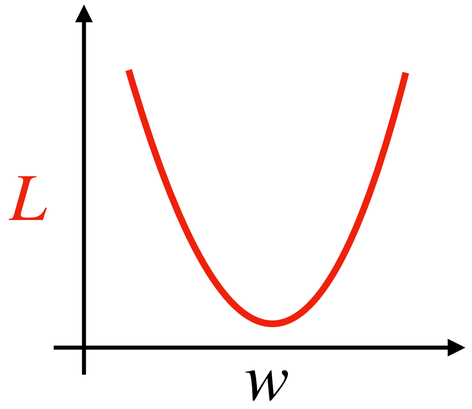


input

Predictive model

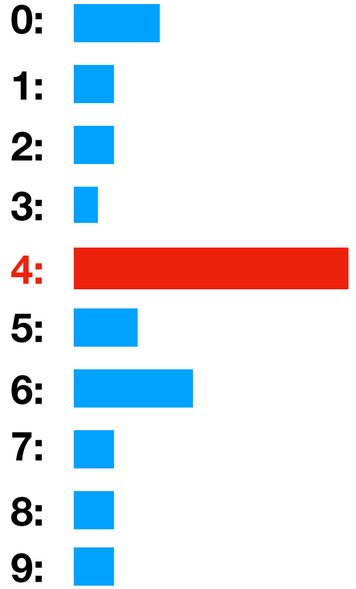
$f(\cdot)$

Model fitting



output

$f(x)$



Find model parameters "w", which minimise loss (L)

Two facets of reproducible modelling

Being able to independently…

…draw **qualitatively similar conclusions**
from new data or reanalysis → Inference reproducibility

…obtain **exact same results**
from original data and analysis code → Computational reproducibility

Goodman, S. N., Fanelli, D., & Ioannidis, J. P. (2016). What does research reproducibility mean?. Science translational medicine, 8(341), 341ps12-341ps12.

Inference reproducibility

1. **Transparent hyper-parameter evaluation**
2. **Reducing test error**
   A. Cross-validation

Inference reproducibility

1. **Transparent hyper-parameter evaluation**
2. **Reducing test error**
   A. Cross-validation

Beware of randomness

Inference reproducibility

1. **Transparent hyper-parameter evaluation**
2. **Reducing test error**
   A. Cross-validation

Beware of randomness

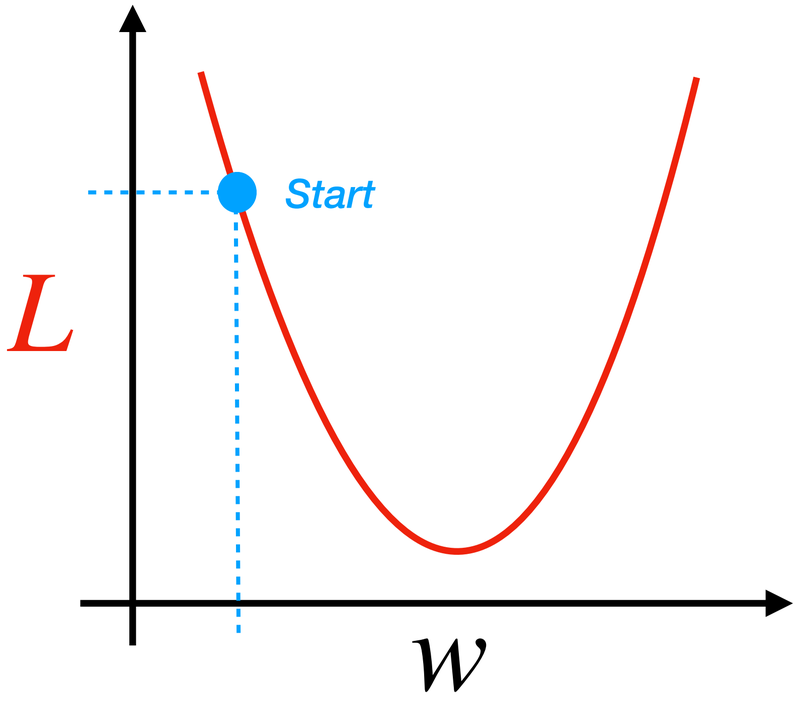
Computational reproducibility

1. **Organising your project**
2. **Coding understandably and portably**
3. **Automating your analysis**
4. **Git & GitHub**

Be transparent about your hyper-parameter choices

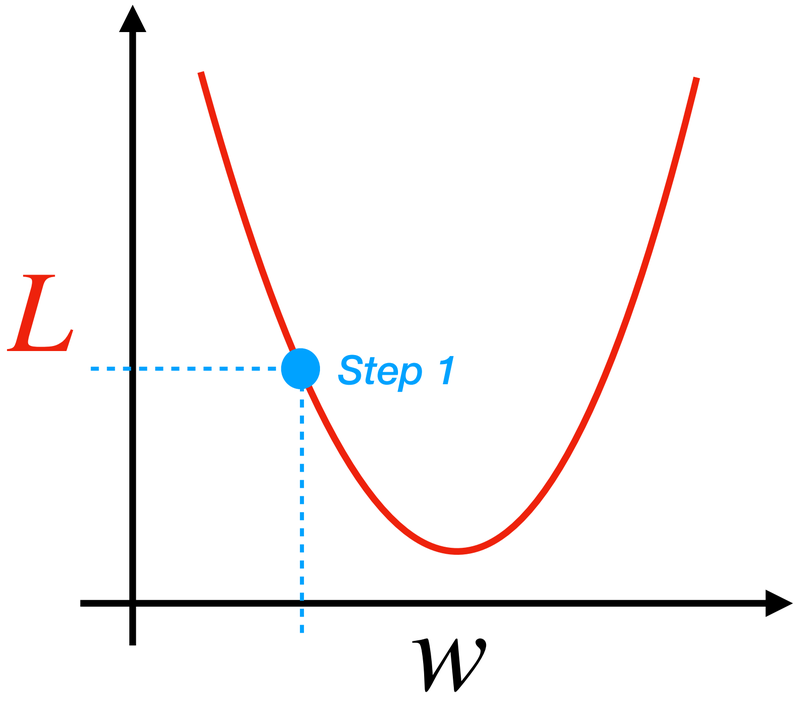
"Values that control model fitting"

Model fitting



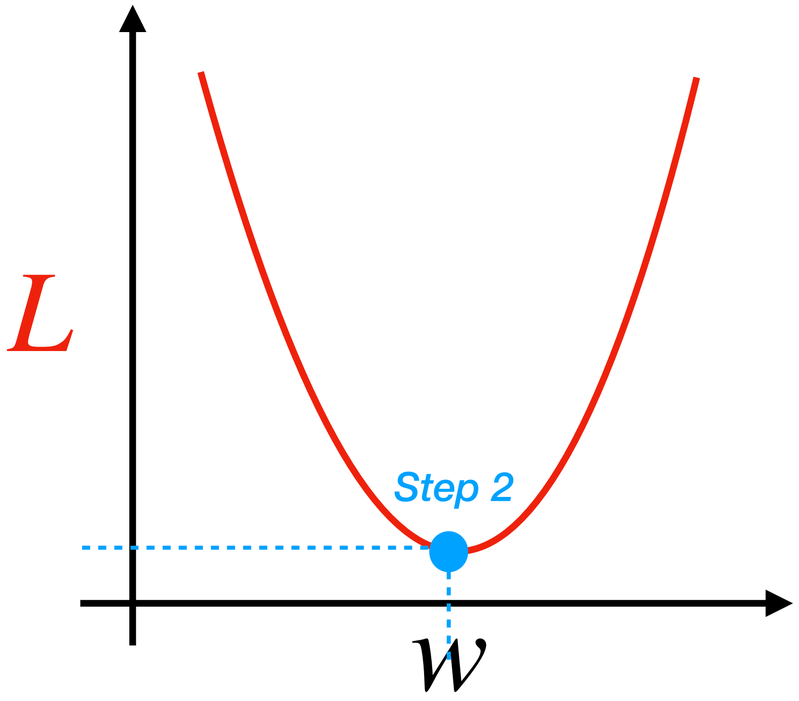
Stochastic gradient descent:
Iteratively find minimum of loss function

Model fitting



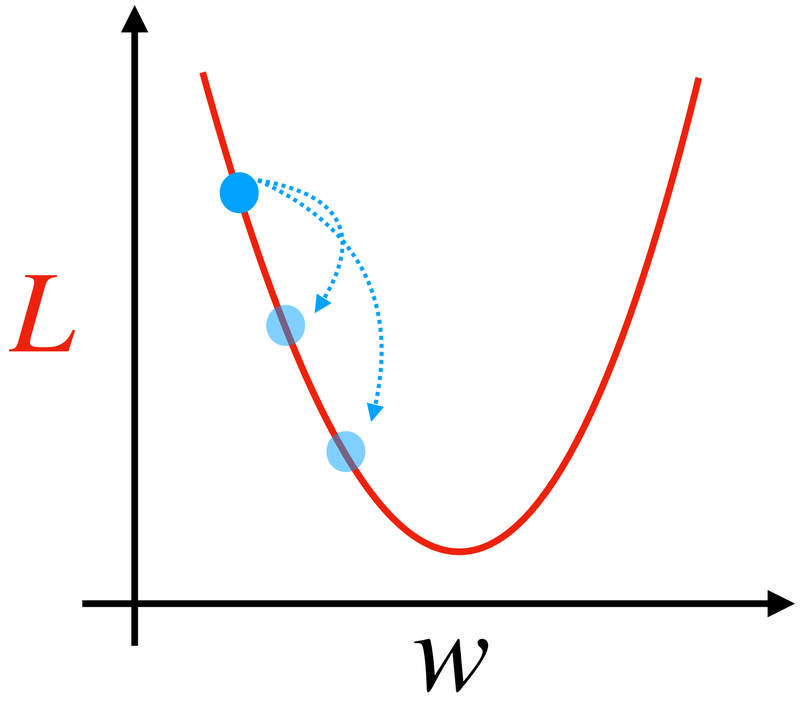
Stochastic gradient descent:
Iteratively find minimum of loss function

Model fitting



Stochastic gradient descent:
Iteratively find minimum of loss function

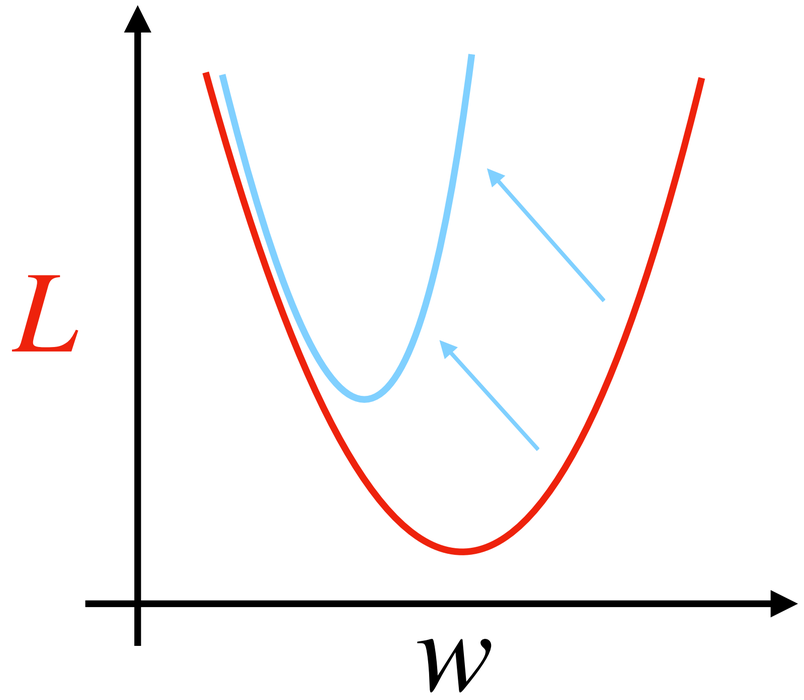
1. Learning Rate



Scales size of gradient descent step

$$w_{s+1} = w_s - \eta \times \frac{dw}{dL}$$

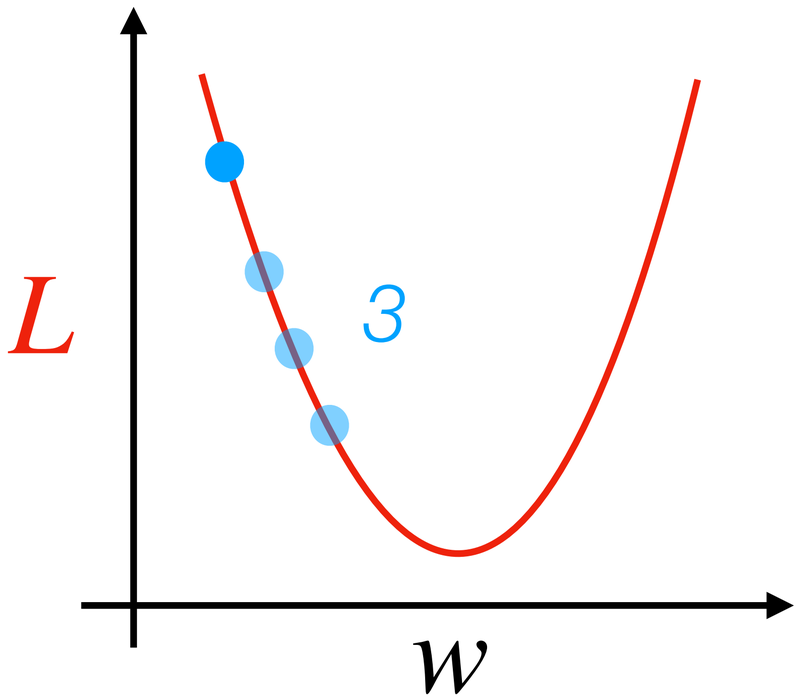
2. Regularisation



Penalises larger "w" to prevent overfitting

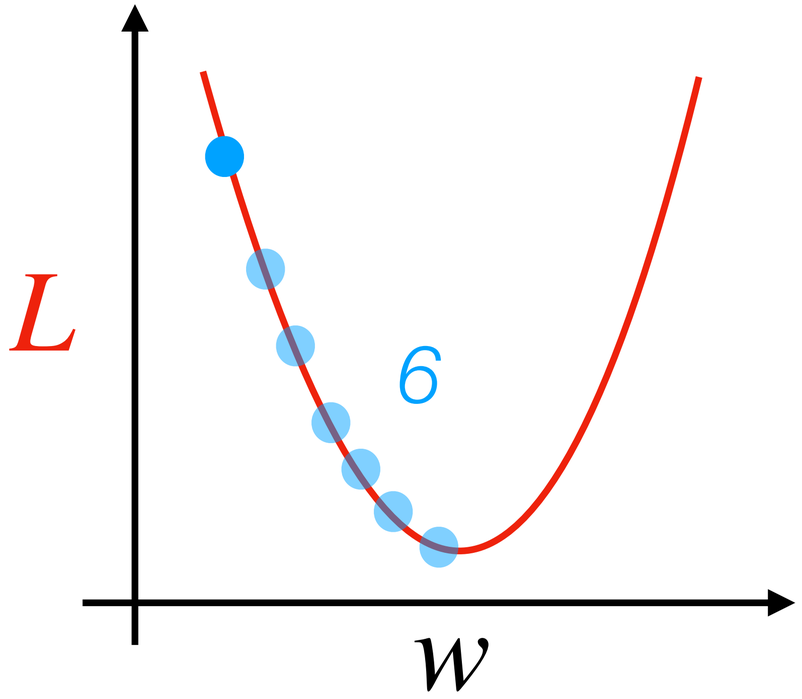
$$w_{s+1} = w_s - \eta \times \frac{dw}{dL} + \alpha \times \sum_i w_i^2$$

3. #Steps



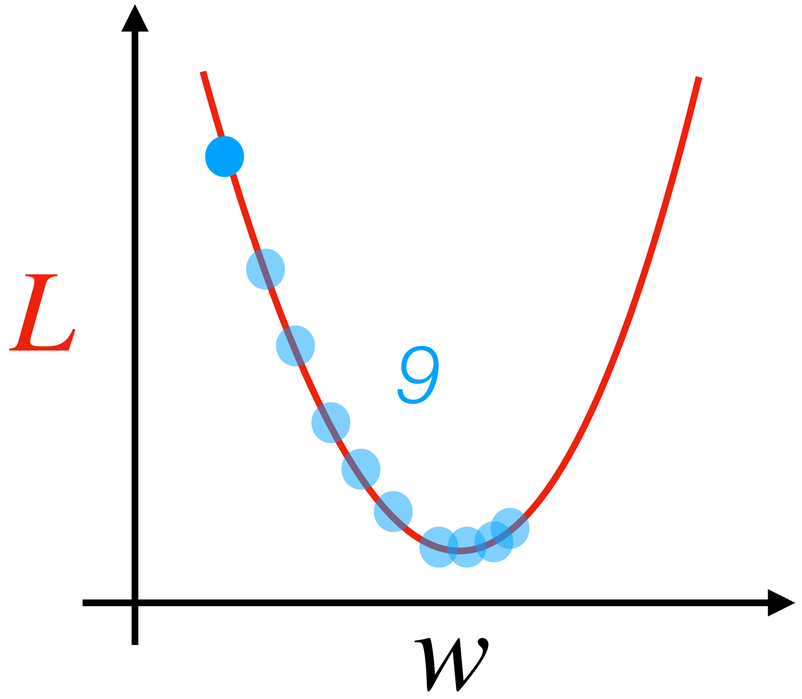
How many weight updates during fitting?

3. #Steps

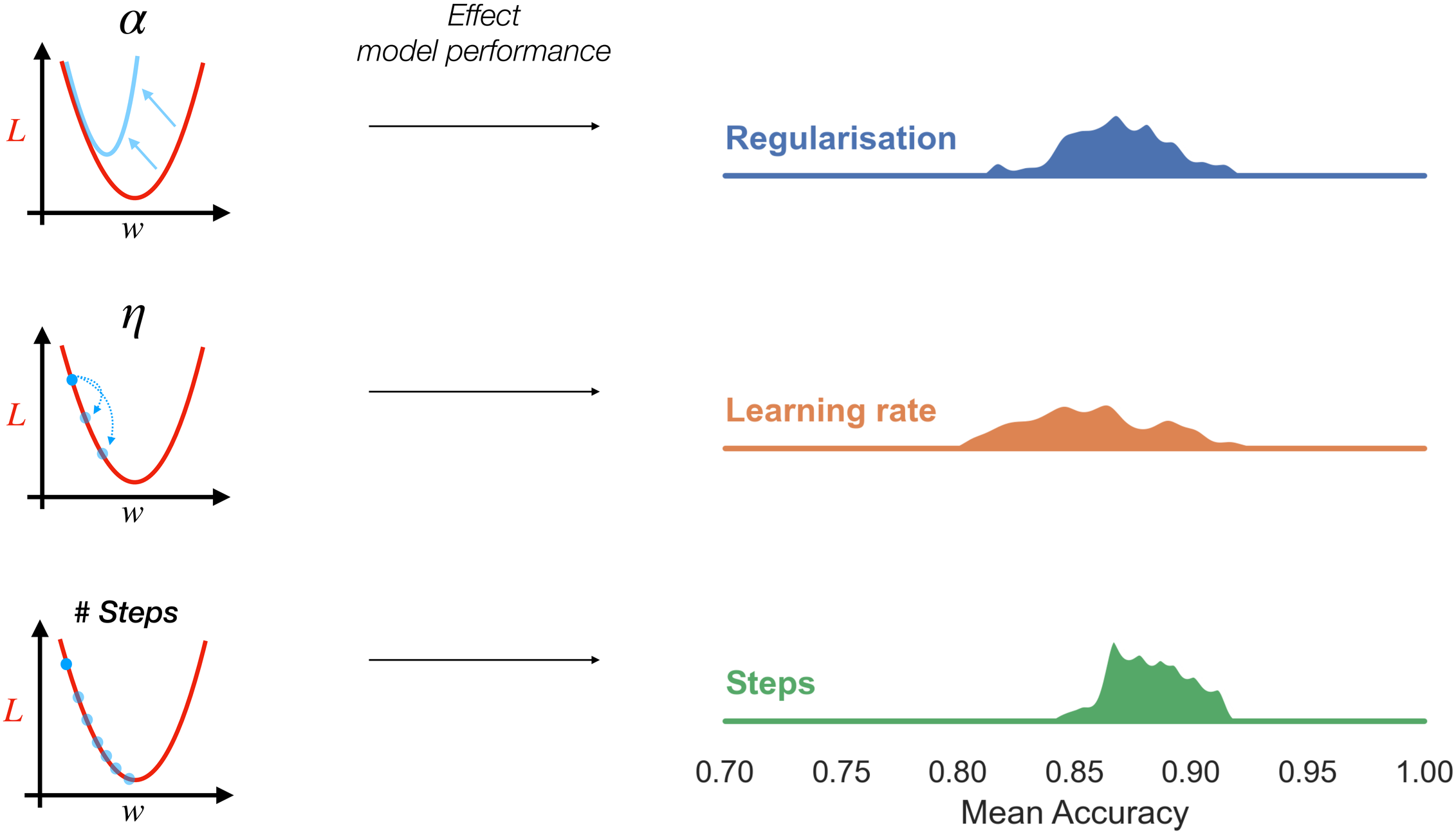


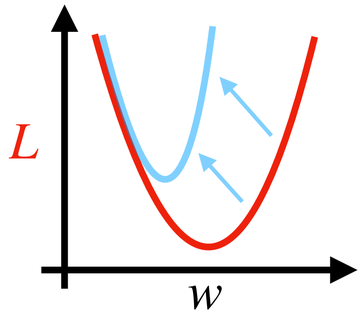
How many weight updates during fitting?

3. #Steps

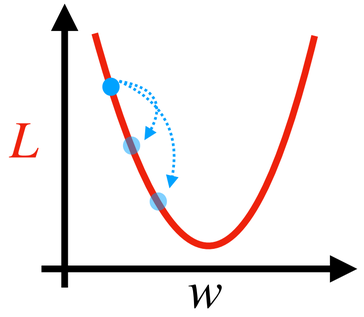


How many weight updates during fitting?

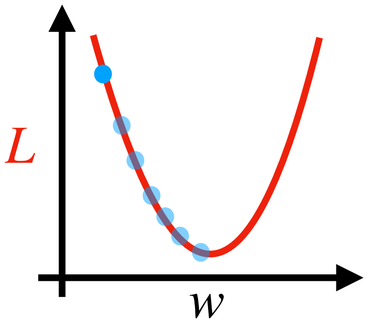

α
L
w
Effect
model performance
Regularisation
η
L
w
Learning rate
Steps
L
w
Steps
0.70
0.75
0.80
0.85
0.90
0.95
1.00
Mean Accuracy



α



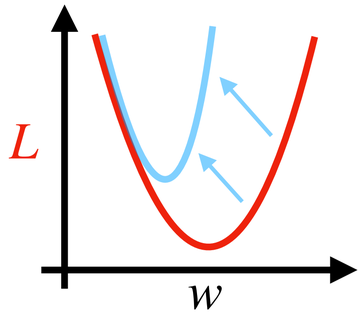
η



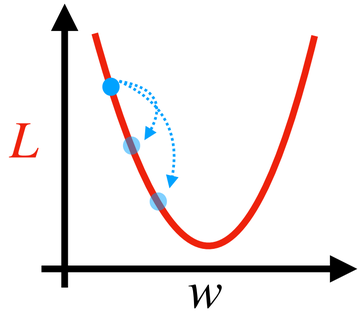
 # Steps

Hyper-parameter optimisation:

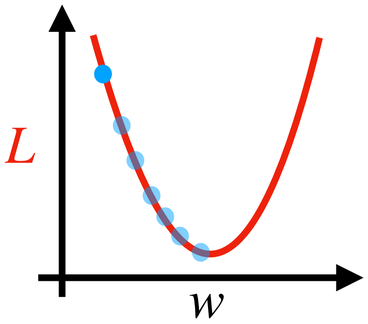
Fitting Run	Learning Rate	Regularisation	Steps	Accuracy
1	0.0001	0.0001	100	0.82
…				



α



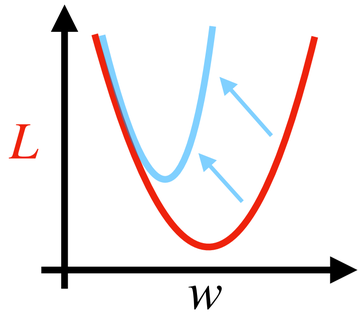
η



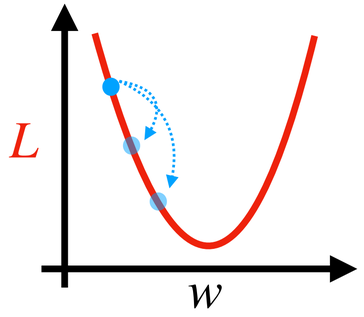
 # Steps

Hyper-parameter optimisation:

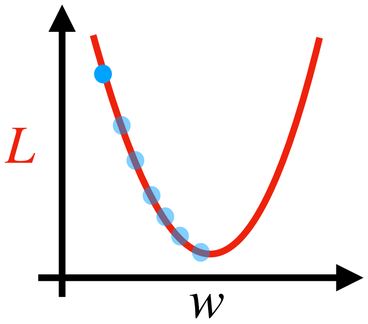
Fitting Run	Learning Rate	Regularisation	Steps	Accuracy
1	0.0001	0.0001	100	0.82
2	0.01	0.01	150	0.95
…				



α



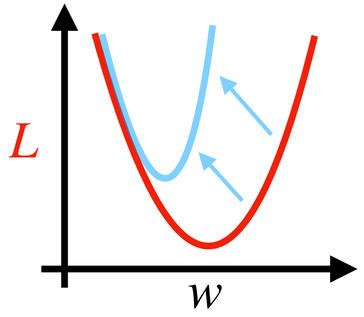
η



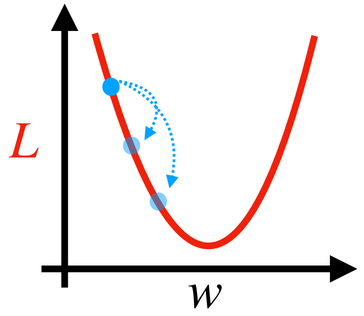
 # Steps

Hyper-parameter optimisation:

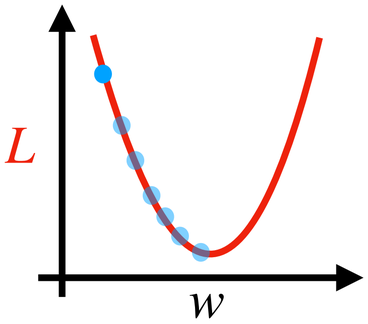
Fitting Run	Learning Rate	Regularisation	Steps	Accuracy
1	0.0001	0.0001	100	0.82
2	0.01	0.01	150	0.95
3	0.1	0.0001	100	0.92
…				



α



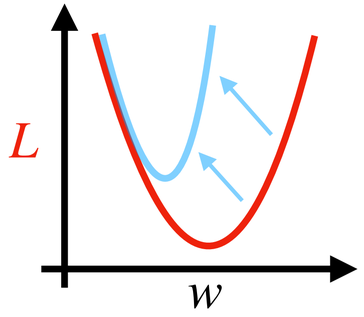
η



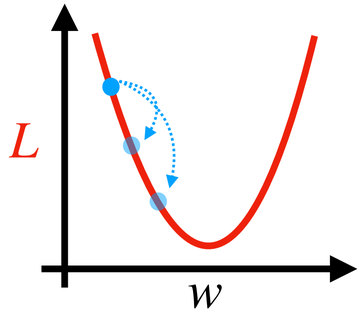
 # Steps

Hyper-parameter optimisation:

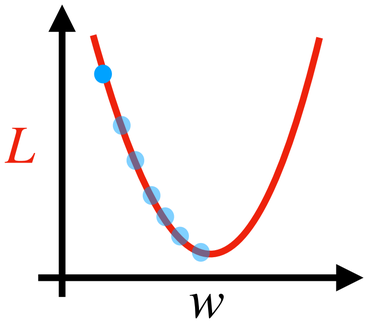
Fitting Run	Learning Rate	Regularisation	Steps	Accuracy
1	0.0001	0.0001	100	0.82
2	0.01	0.01	150	0.95
3	0.1	0.0001	100	0.92
4	0.2	0.0001	200	0.91
…				



α



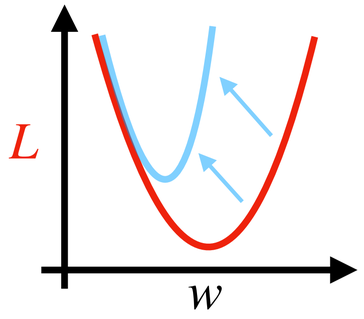
η



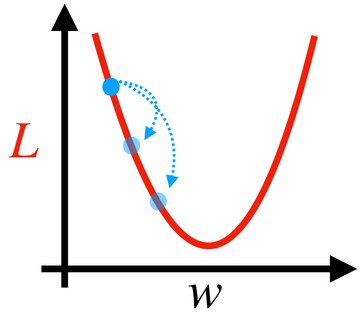
 # Steps

Hyper-parameter optimisation:

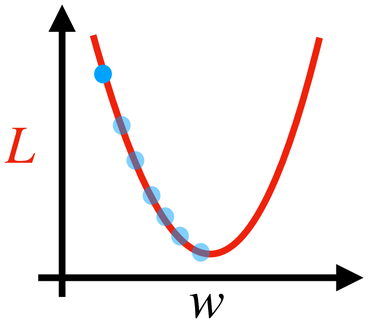
Fitting Run	Learning Rate	Regularisation	Steps	Accuracy
1	0.0001	0.0001	100	0.82
2	**0.01**	**0.01**	**150**	**0.95**
3	0.1	0.0001	100	0.92
4	0.2	0.0001	100	0.91
…				



α



η



 # Steps

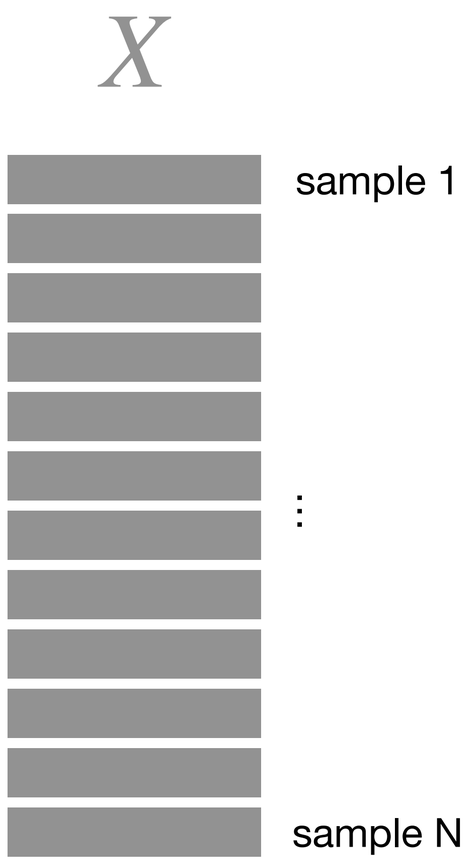
Hyper-parameter optimisation:

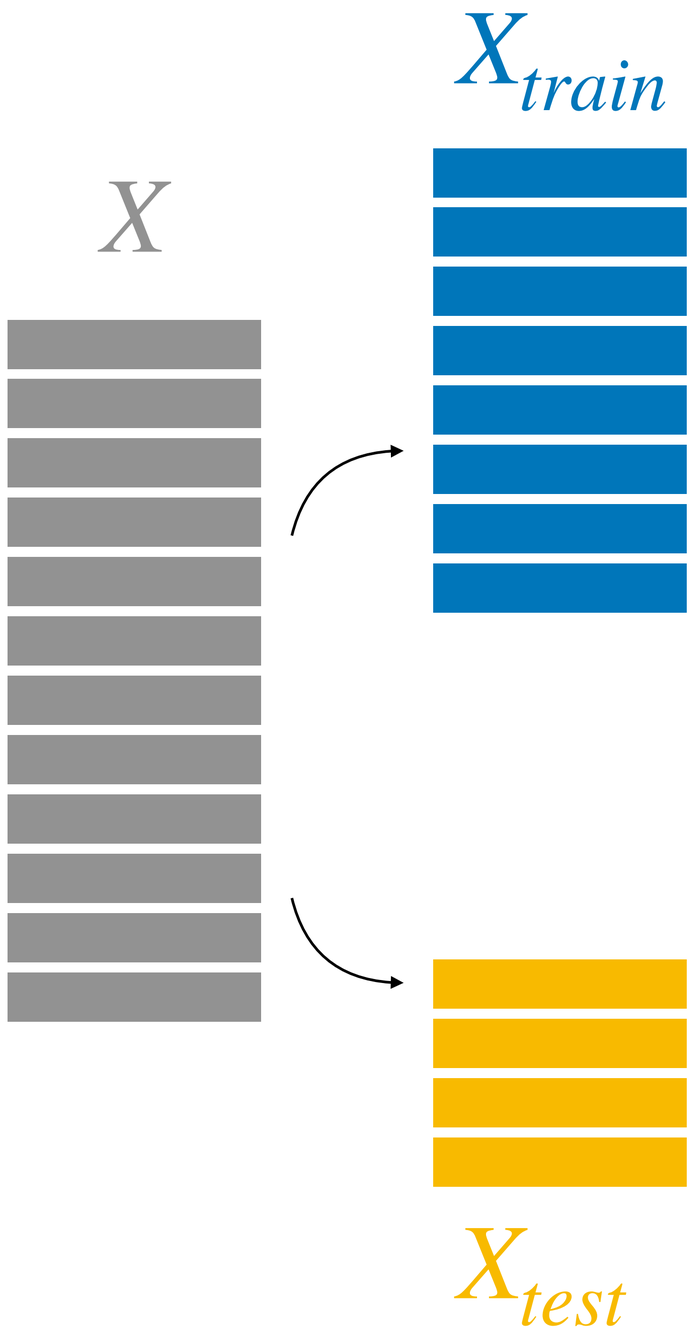
Fitting Run	Learning Rate	Regularisation	Steps	Accuracy
1	0.0001	0.0001	100	0.82
2	**0.01**	**0.01**	**150**	**0.95**
3	0.1	0.0001	100	0.92
4	0.2	0.0001	100	0.91
…				

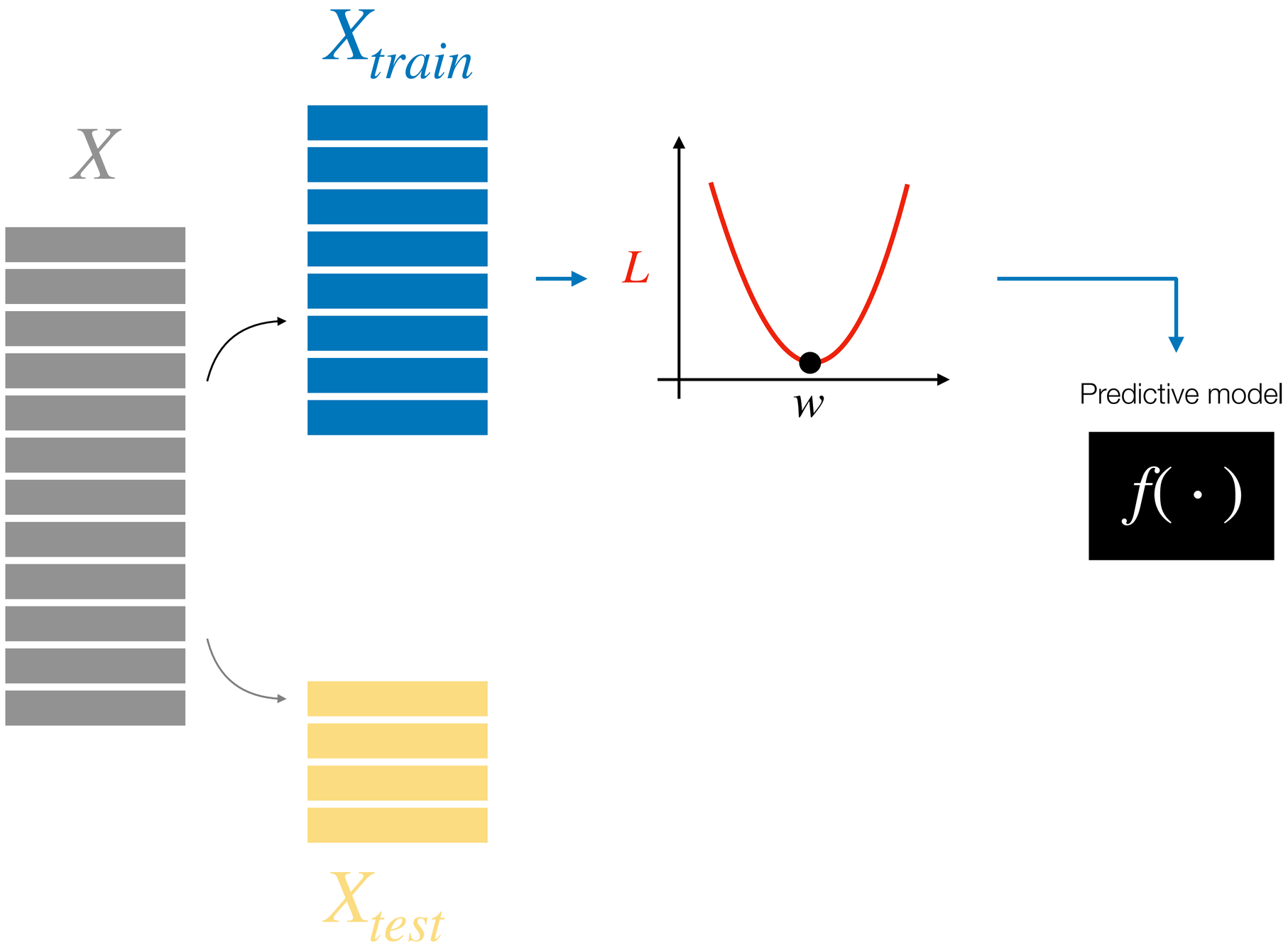
Transparency about hyper-parameter evaluation is essential to enable others to reproduce your analysis!

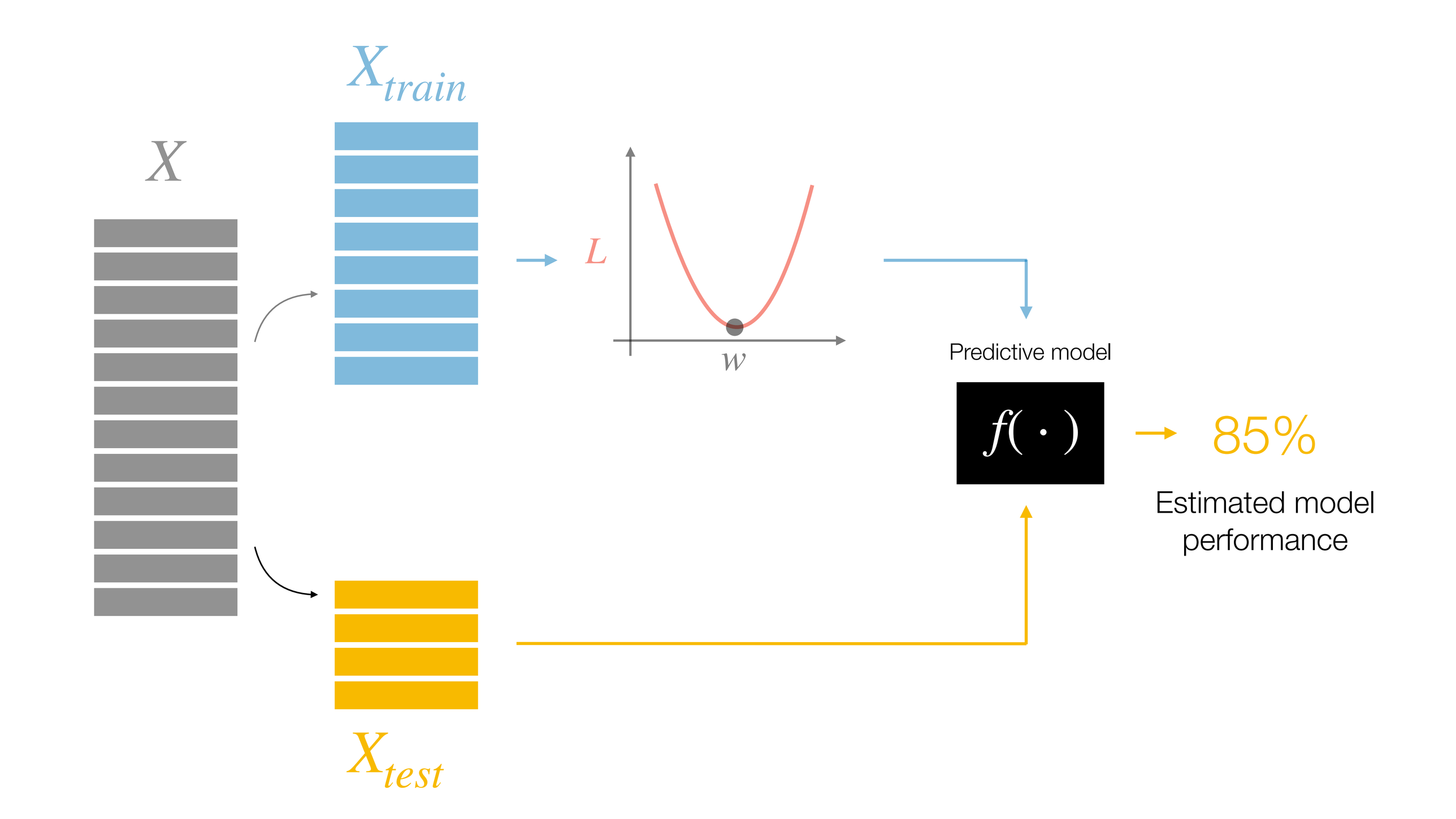
Reducing test error

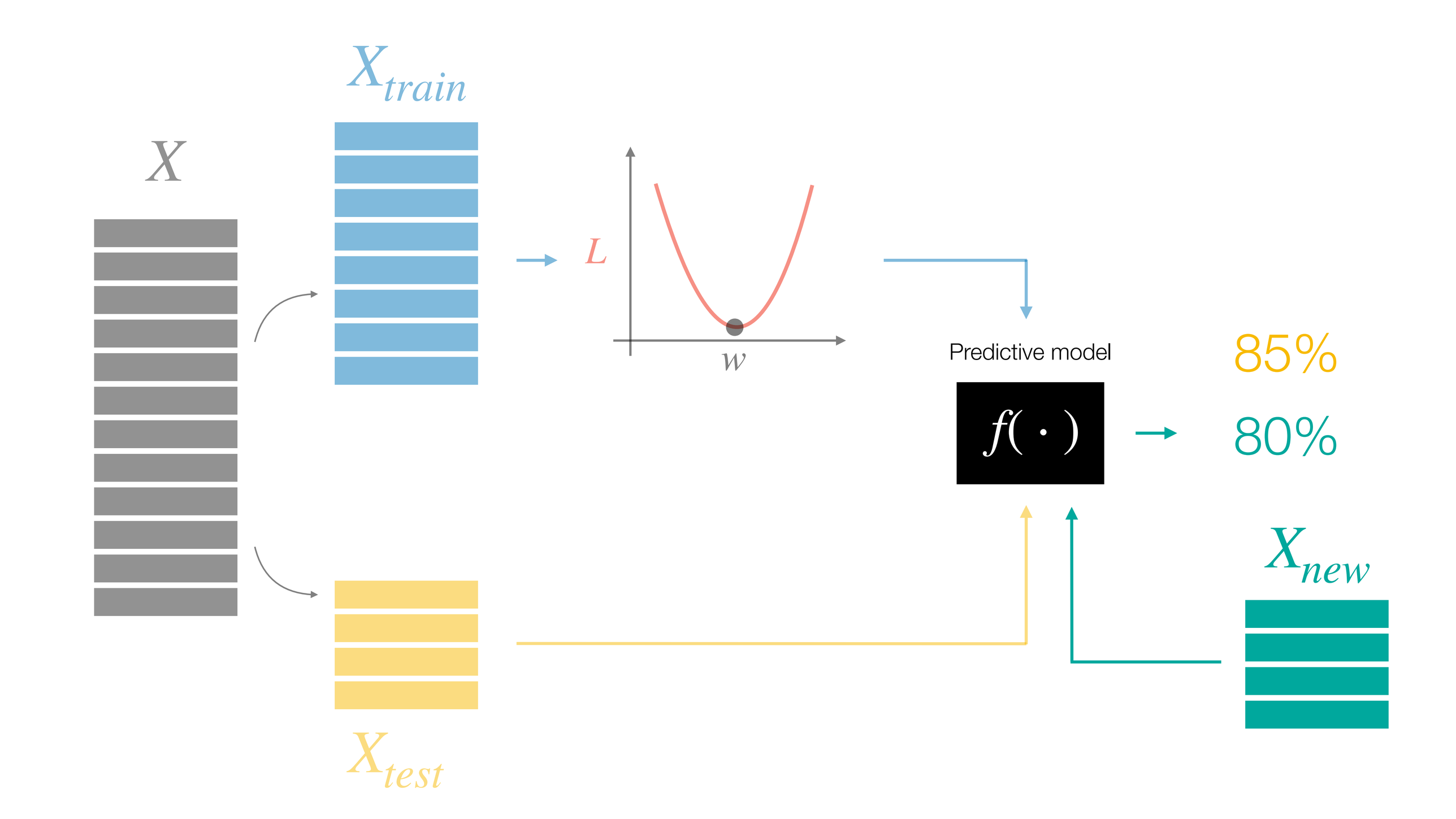
“Improving your estimate of how well model will perform in new data”

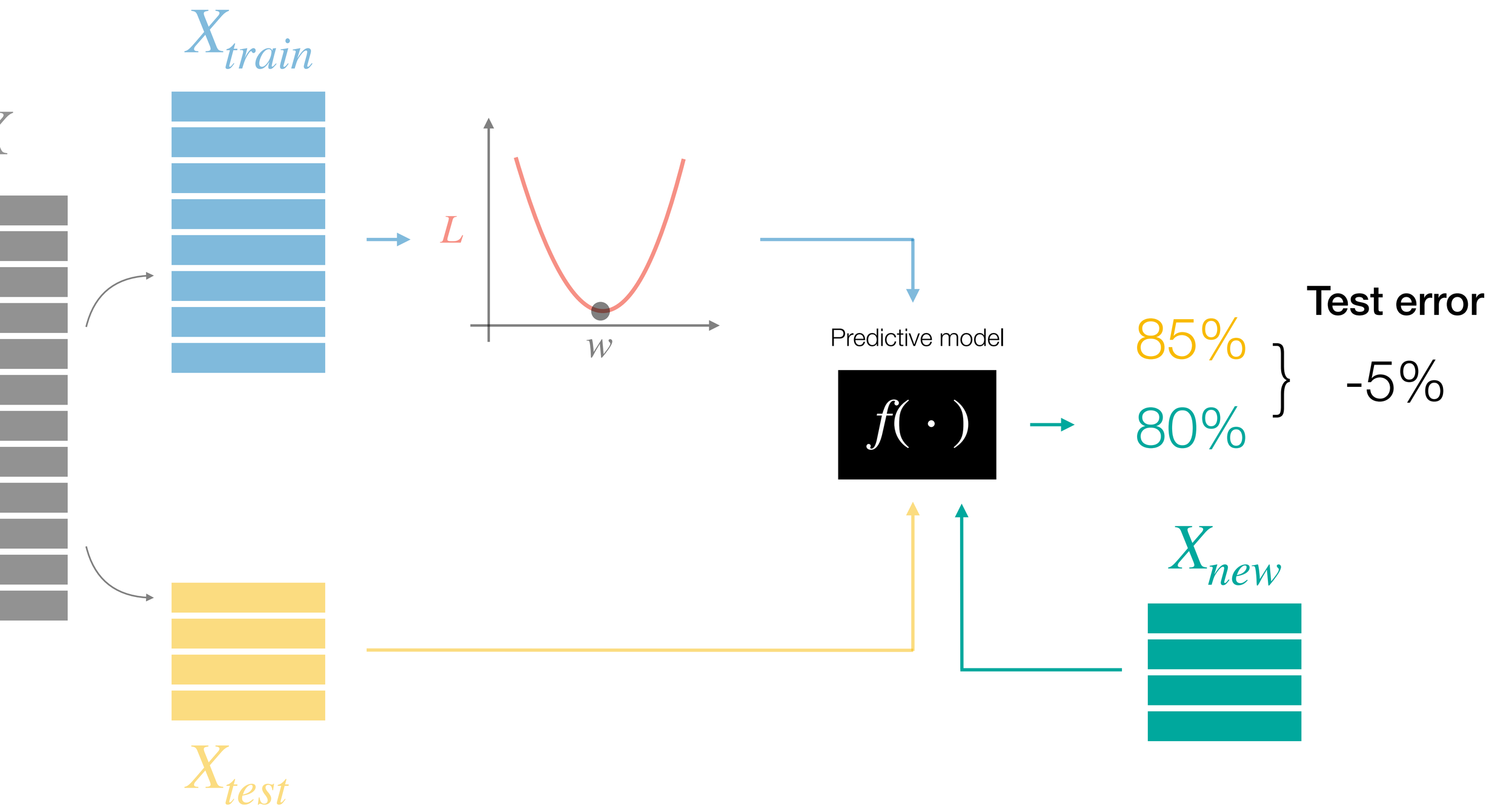

X
sample 1
sample N

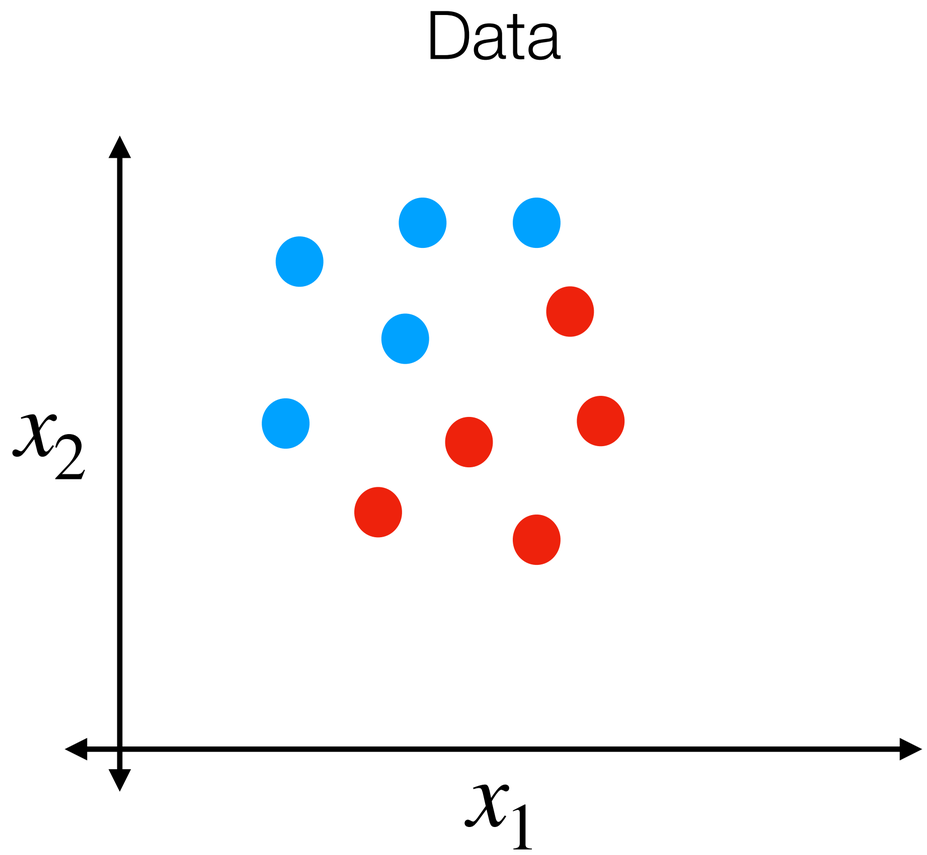

X_{train}
X
X_{test}


X_{train}
X
L
w
Predictive model
$f(\cdot)$
X_{test}

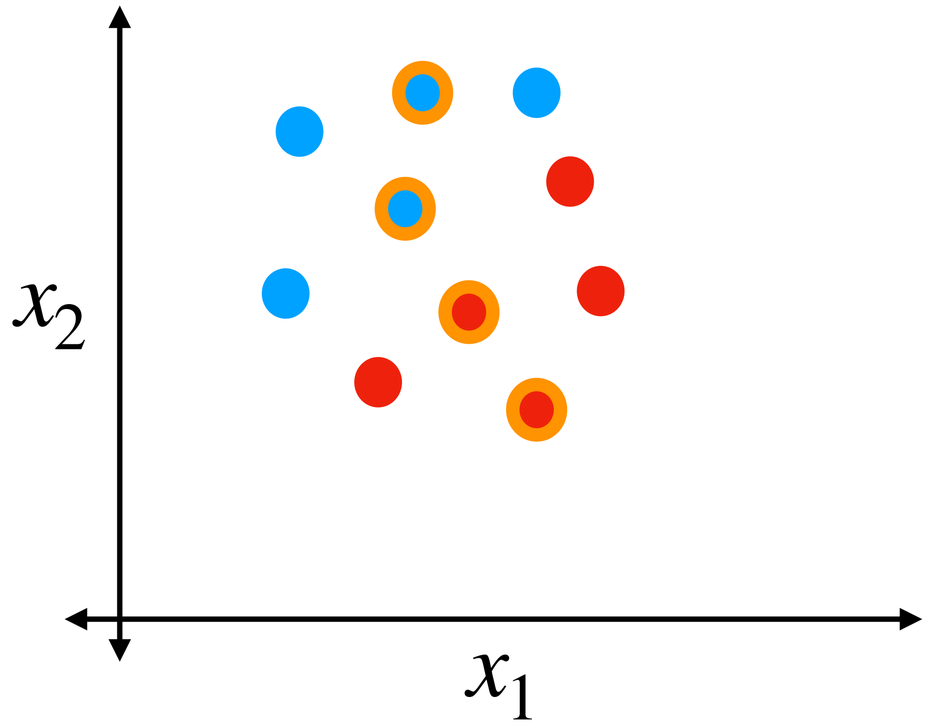

X_{train}
X
L
w
Predictive model
$f(\cdot)$
85%
Estimated model performance
X_{test}

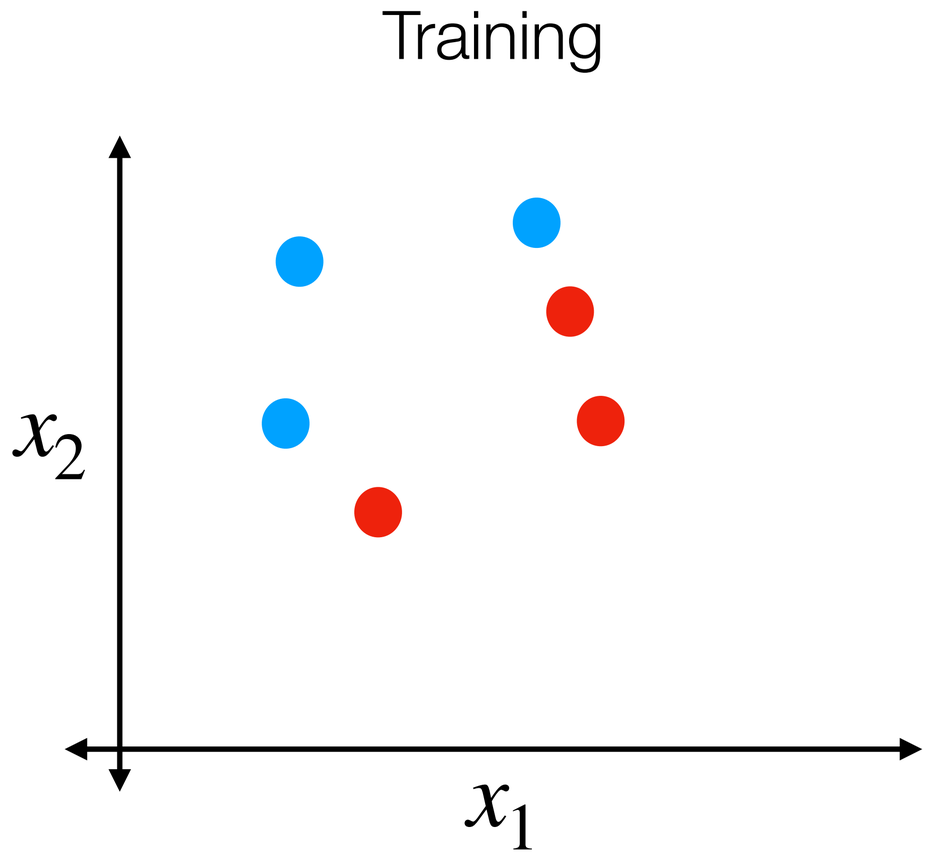

X_{train}
X
L
w
Predictive model
$f(\cdot)$
85%
80%
X_{new}
X_{test}


X_{train}
L
w
Predictive model
$f(\cdot)$
85%
80%
Test error
-5%
X_{new}
X_{test}

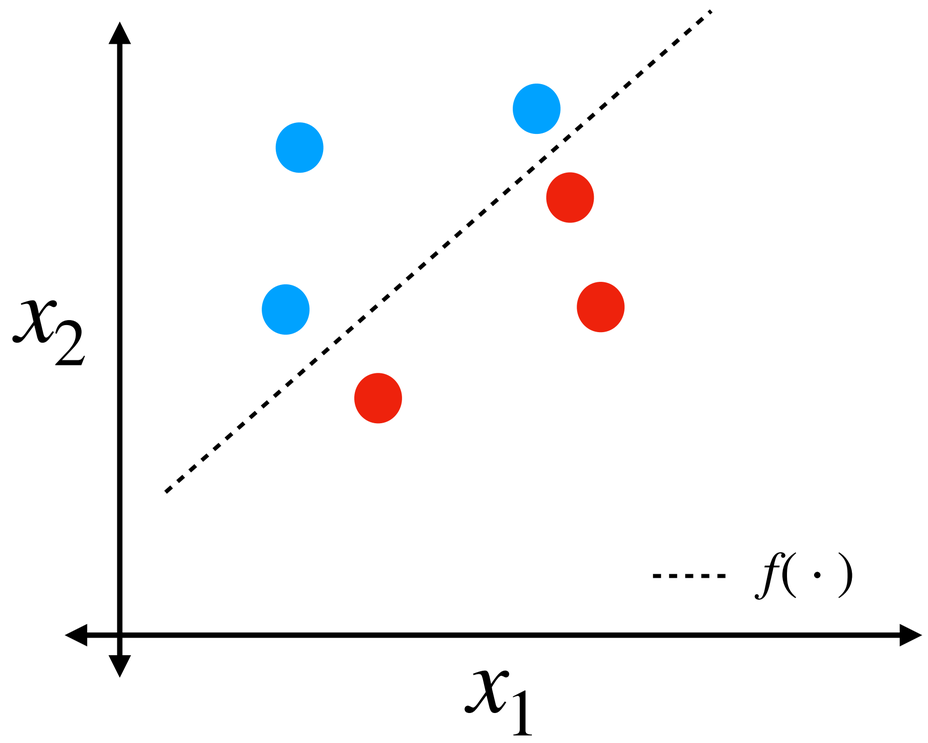

Data
x_2
x_1

Train / Test Split

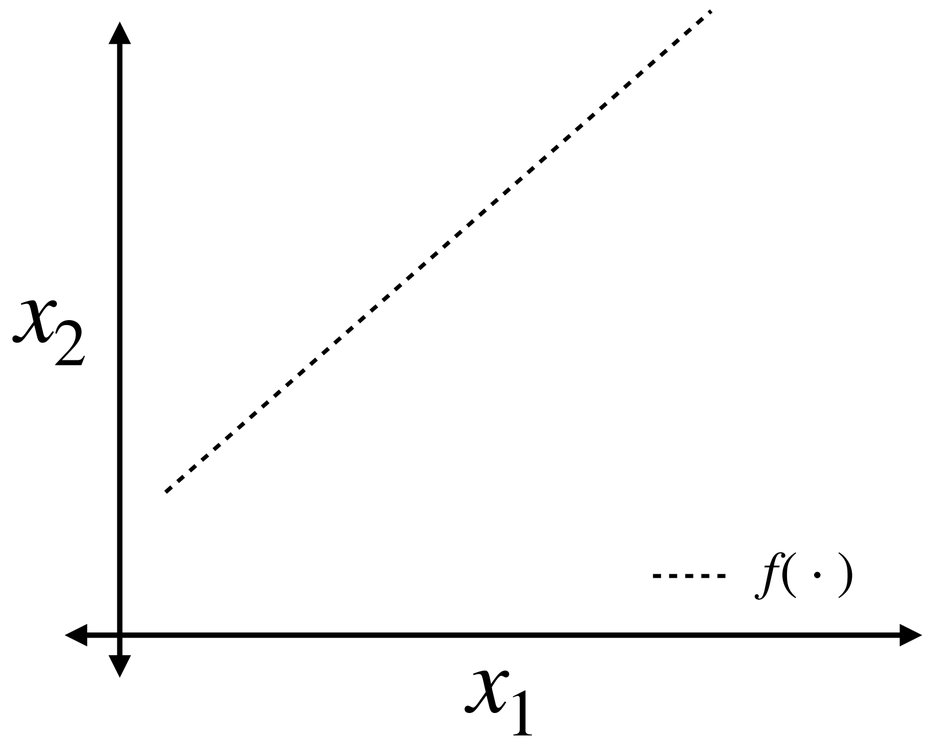


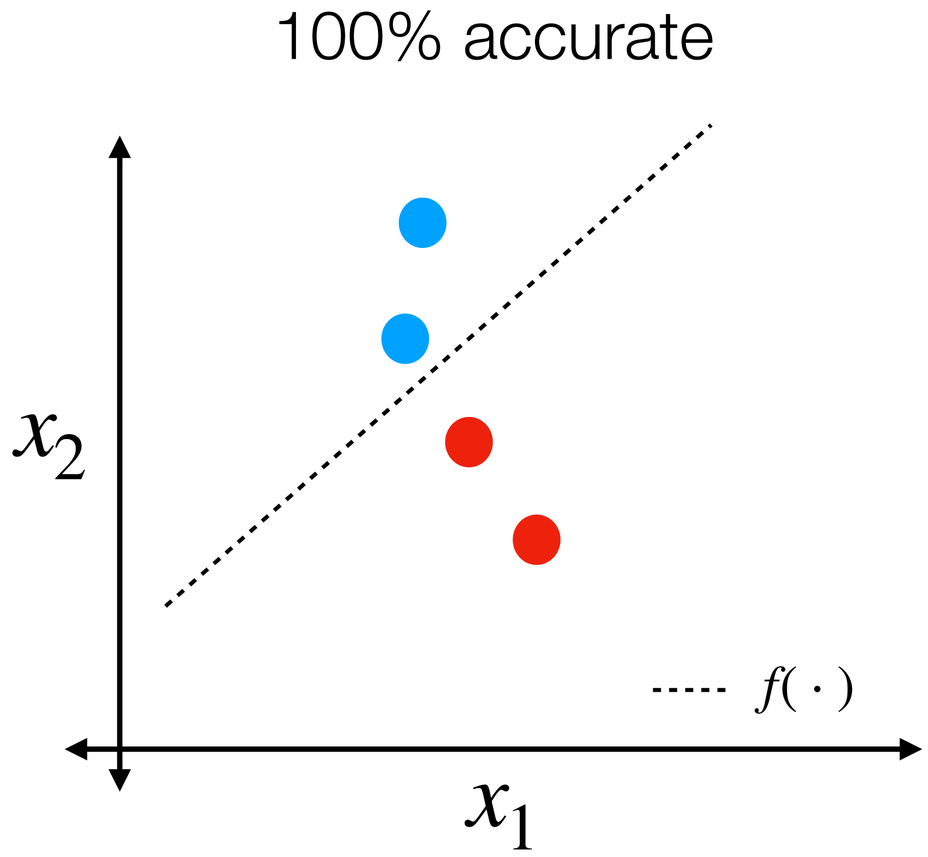

Training
x_2
x_1

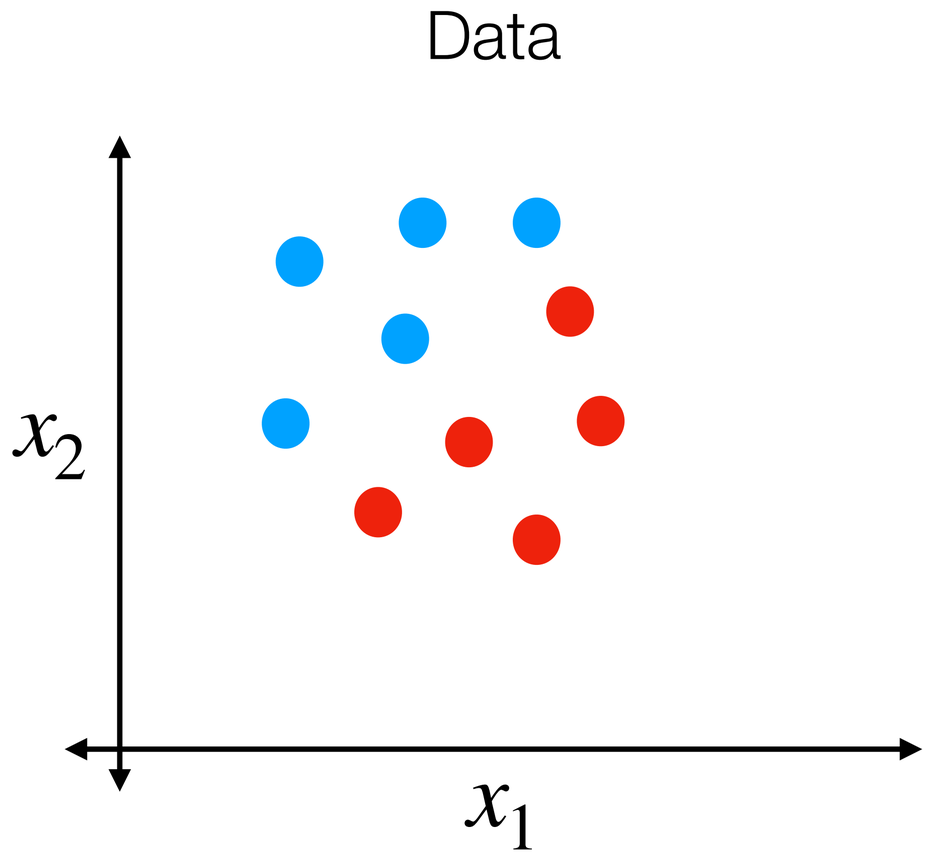
Fitted Model



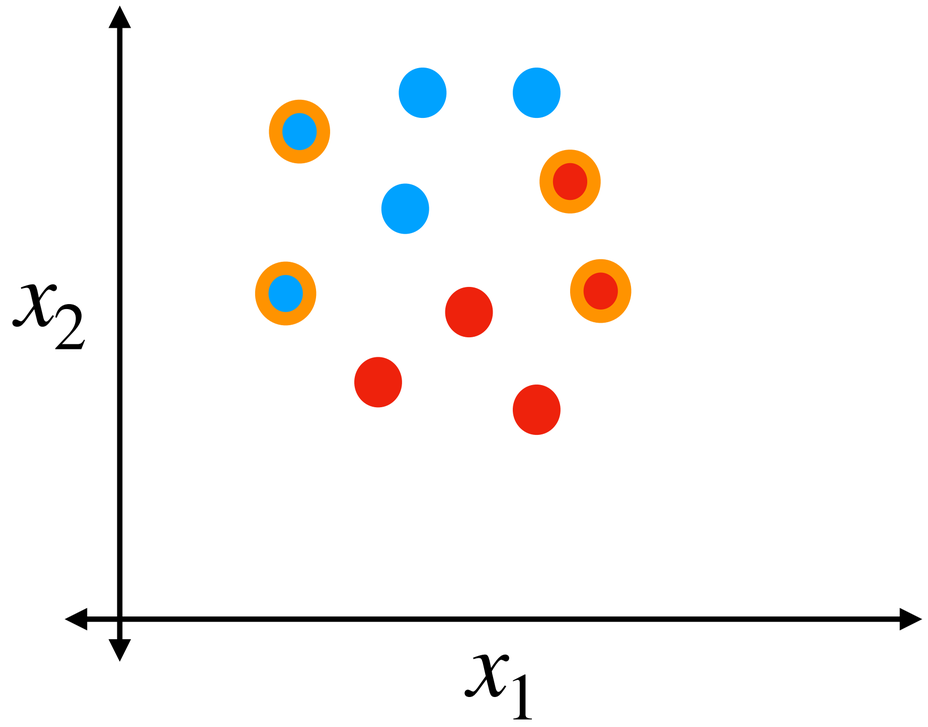
Evaluation

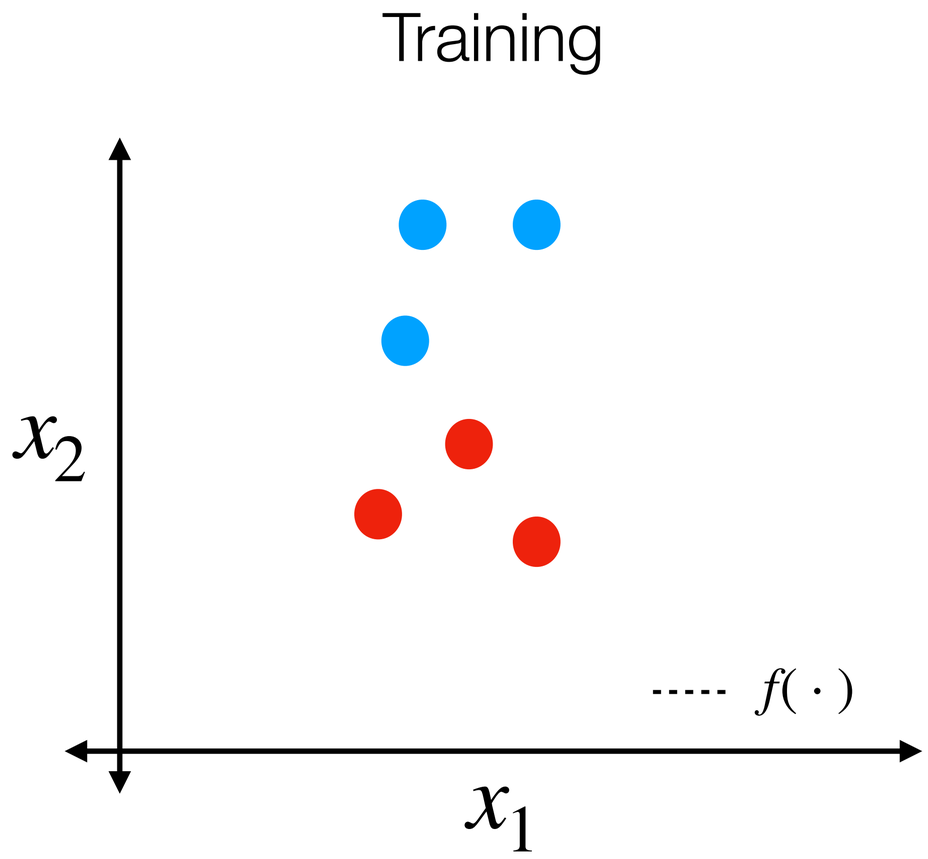



100% accurate
x_2
x_1
----- $f(\cdot)$

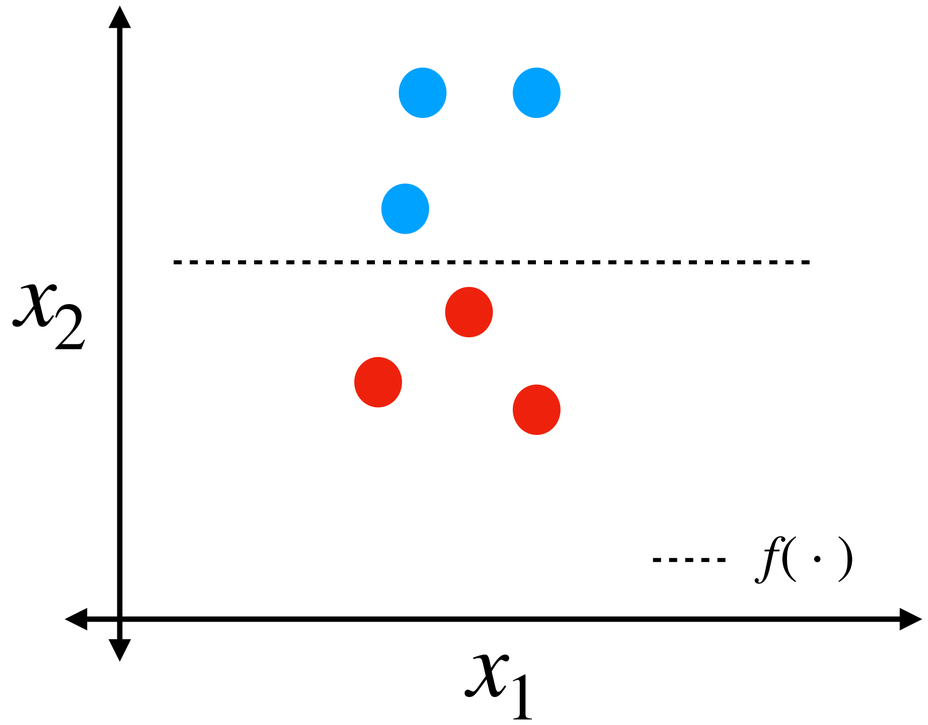

Data
x_2
x_1

Train / Test Split

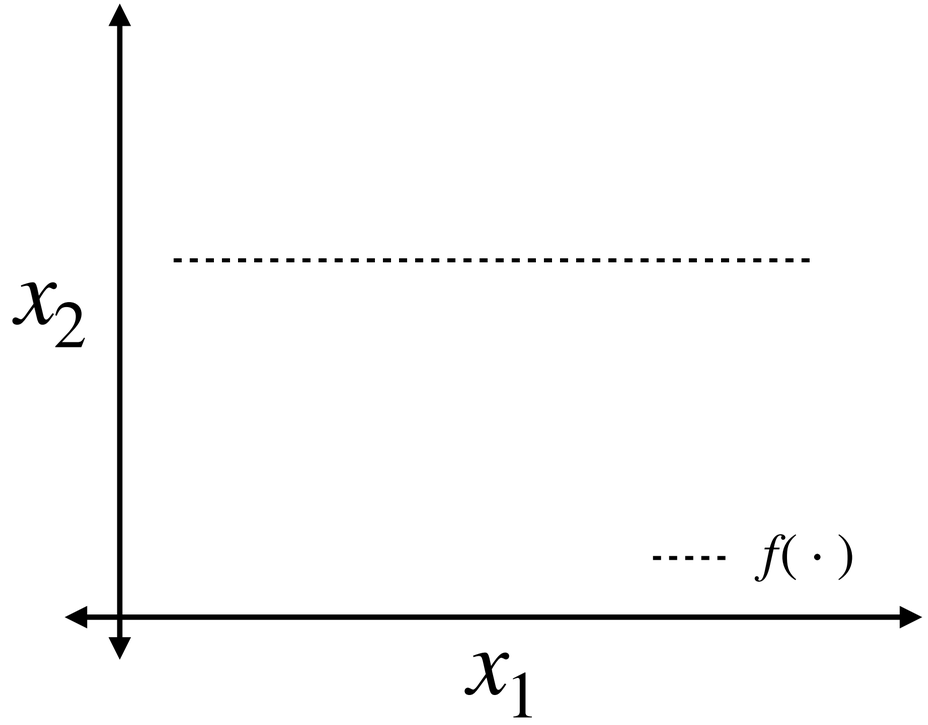



Training
x_2
x_1
----- $f(\cdot)$

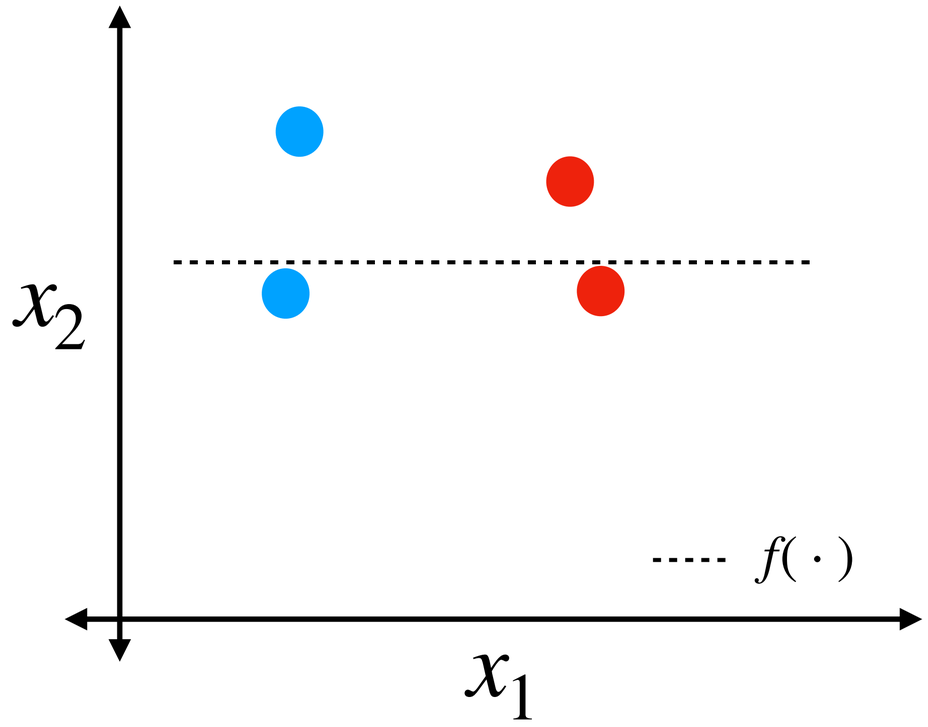
Fitted Model



Evaluation



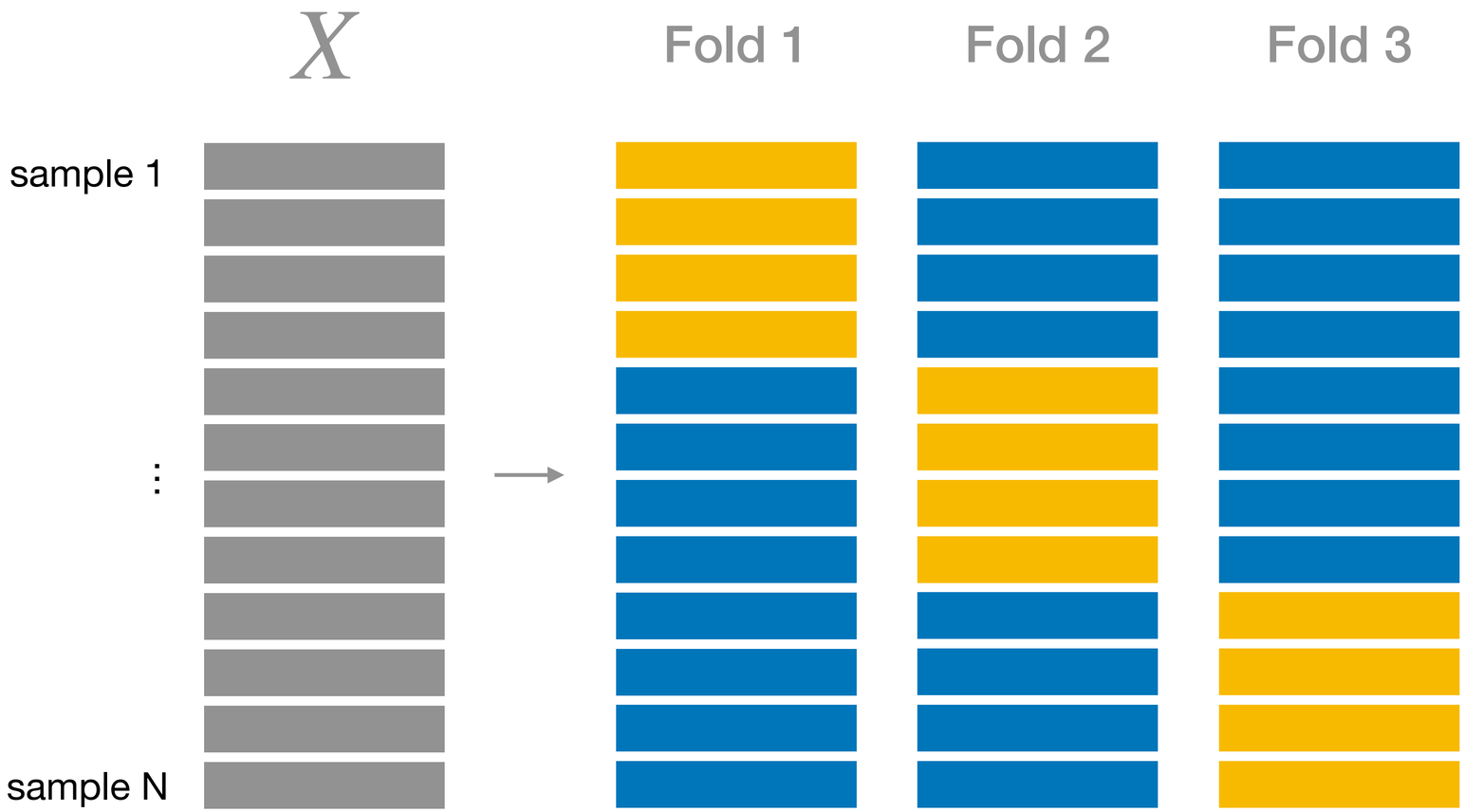
50% accurate



Cross-validation

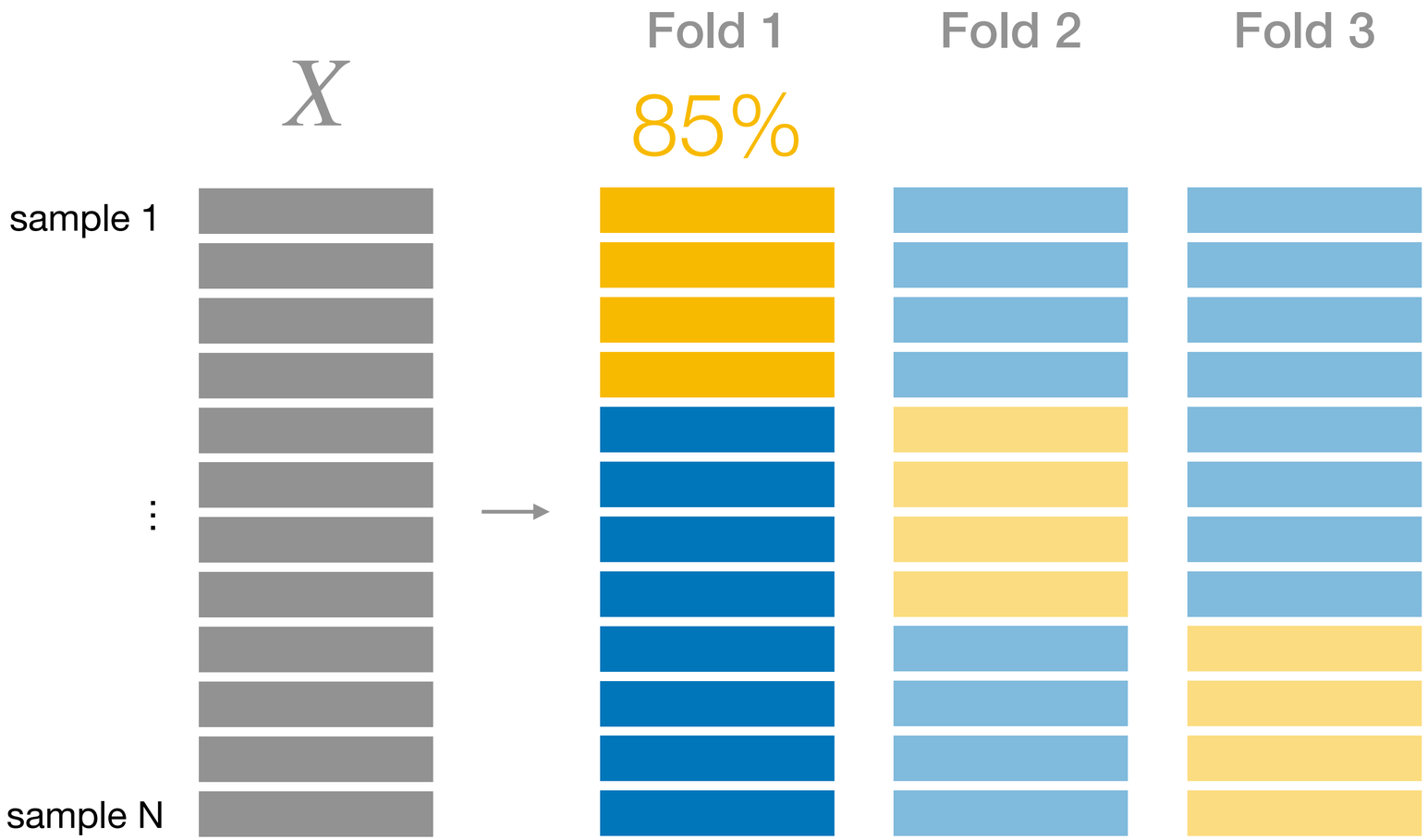
Account for effect of data split on model performance

Cross-validation



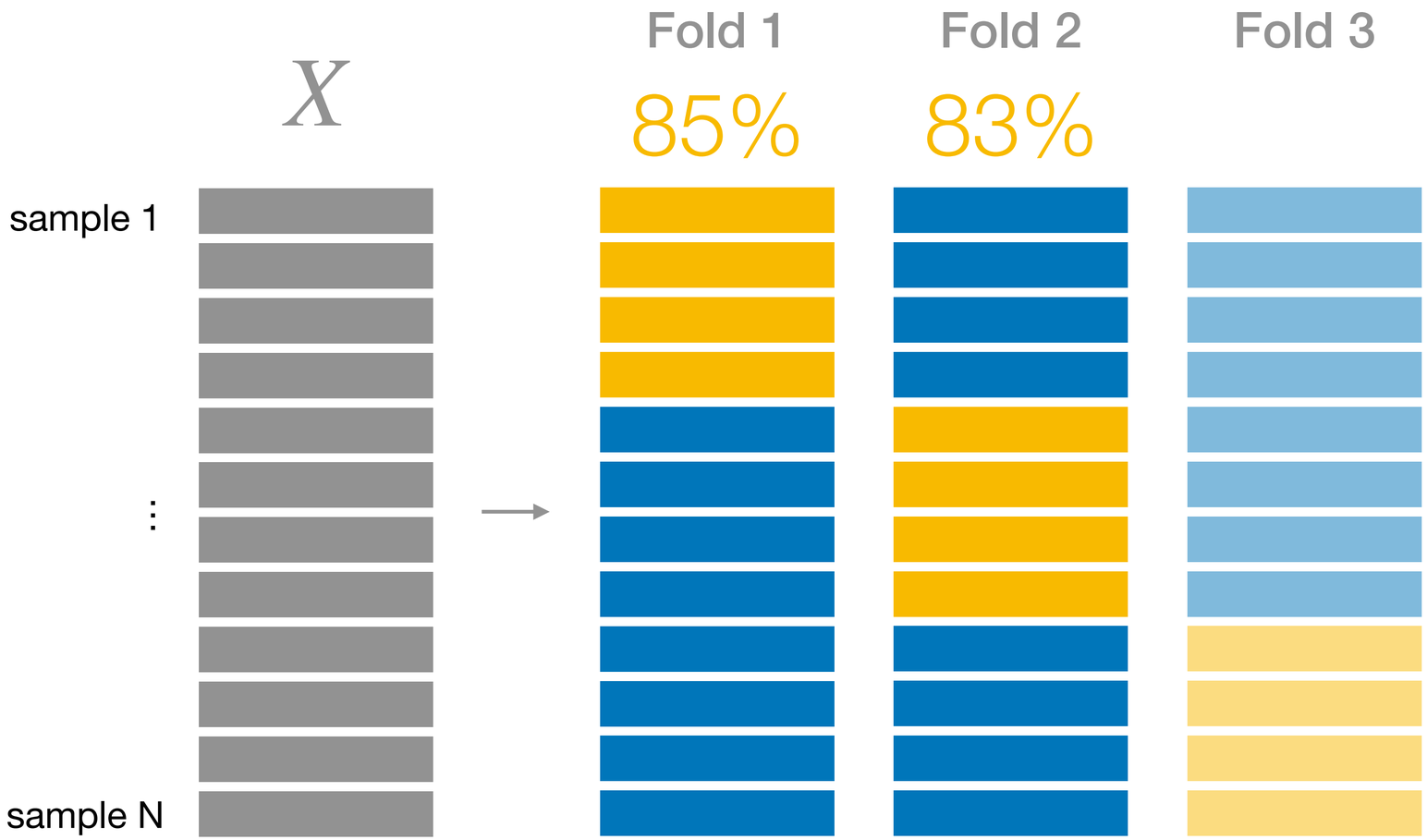
Repeatedly split data,
such that each sample is once in test set

Cross-validation



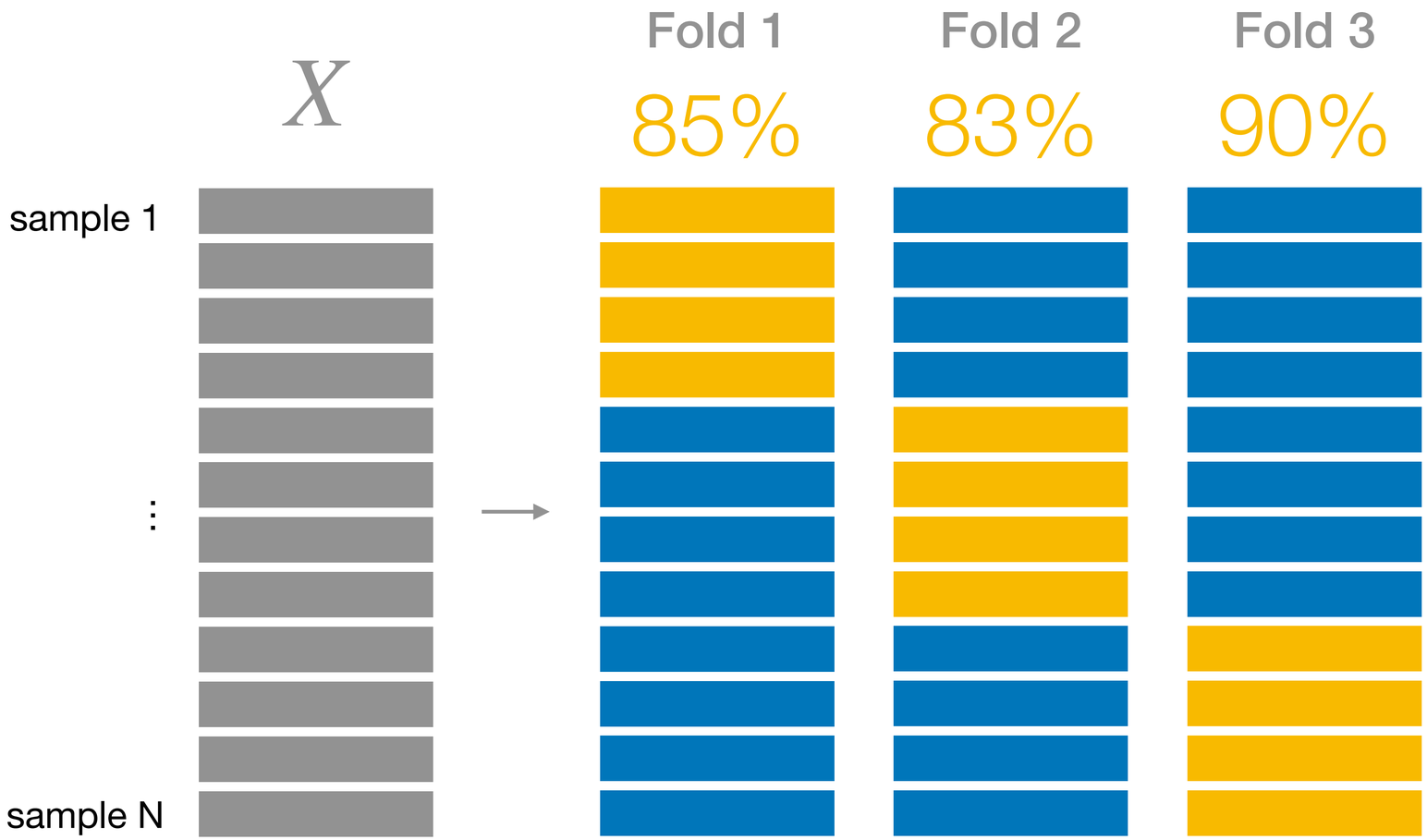
Fit & evaluate model for each fold (i.e., split)

Cross-validation



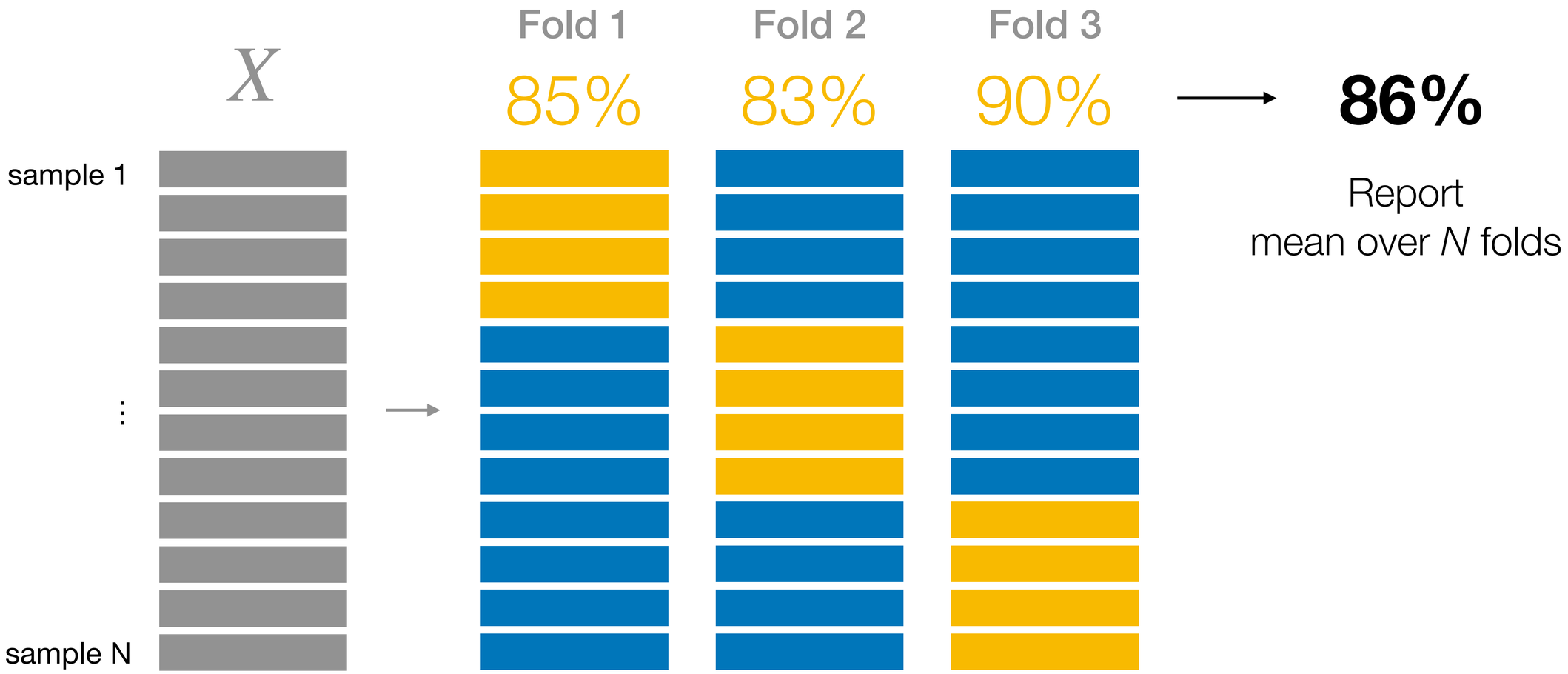
Fit & evaluate model for each fold (i.e., split)

Cross-validation



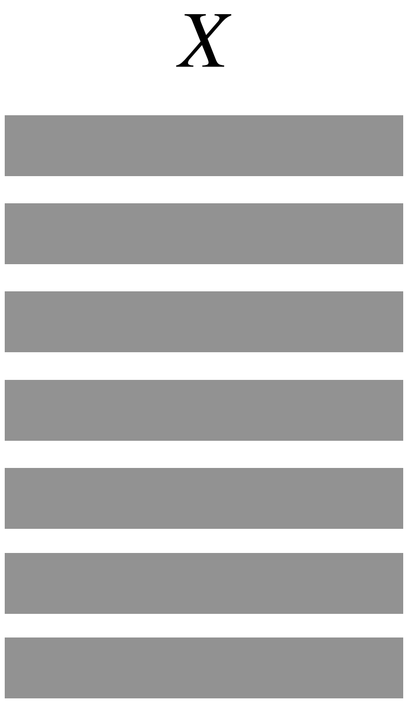
Fit & evaluate model for each fold (i.e., split)

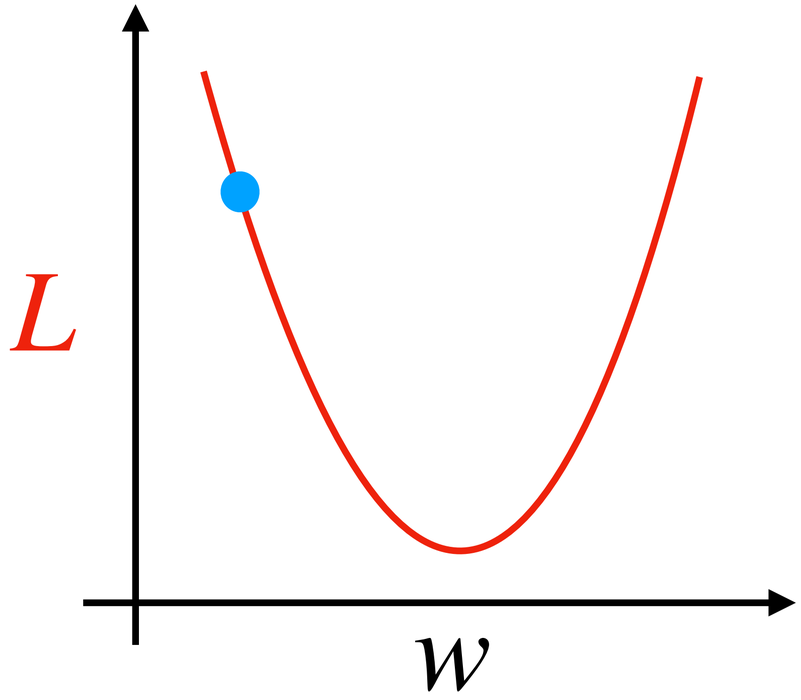
Cross-validation

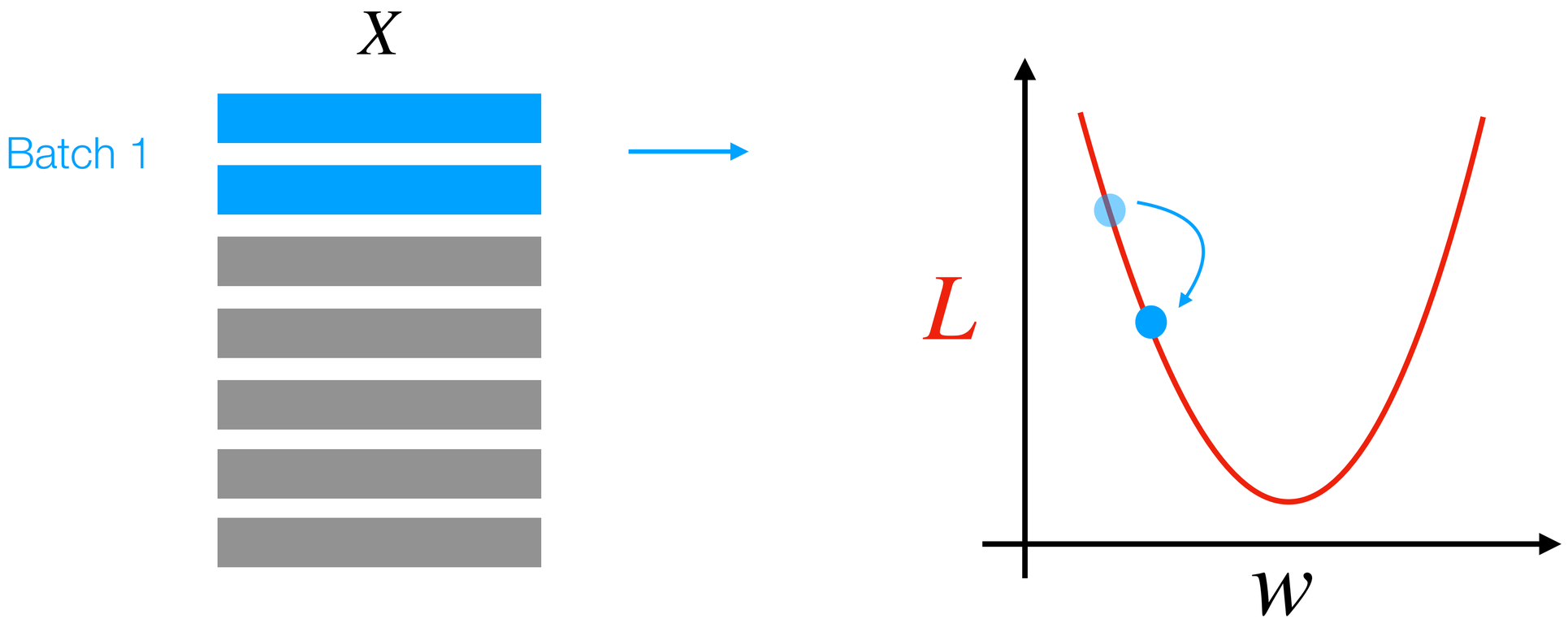


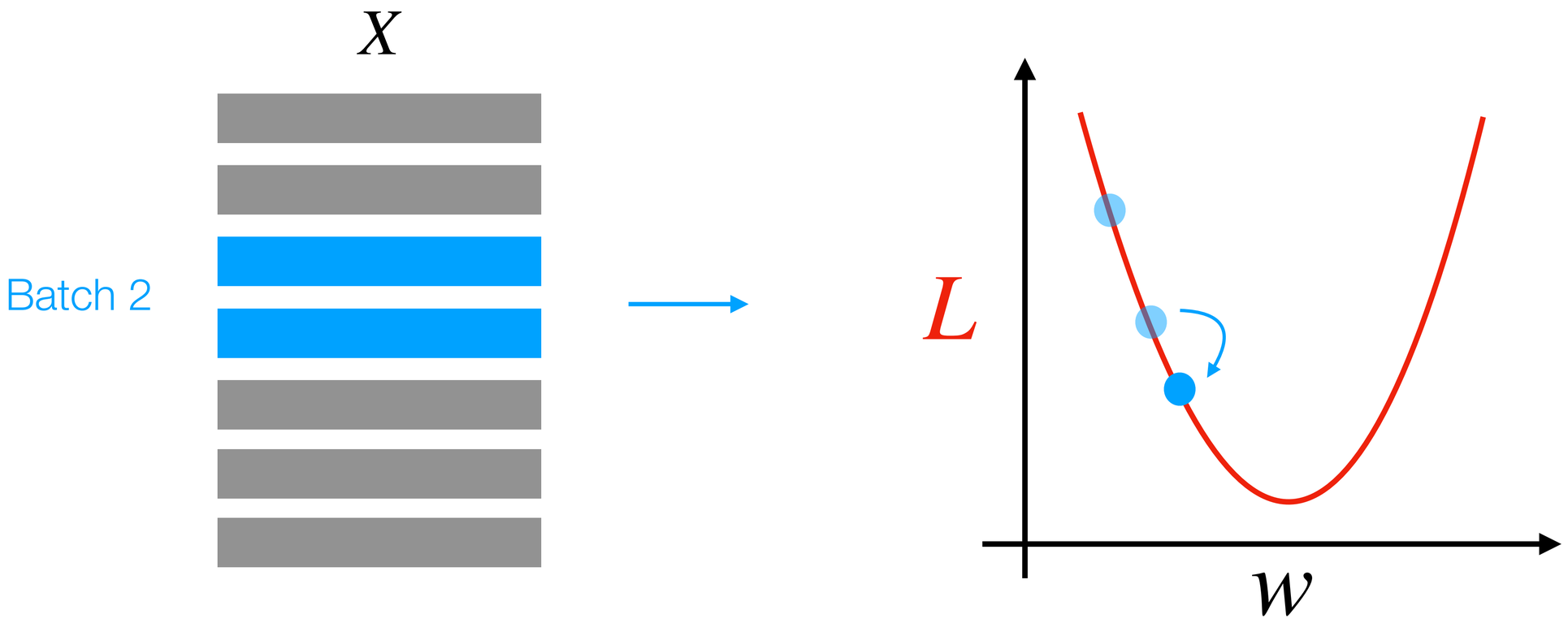
Report aggregate performance across folds

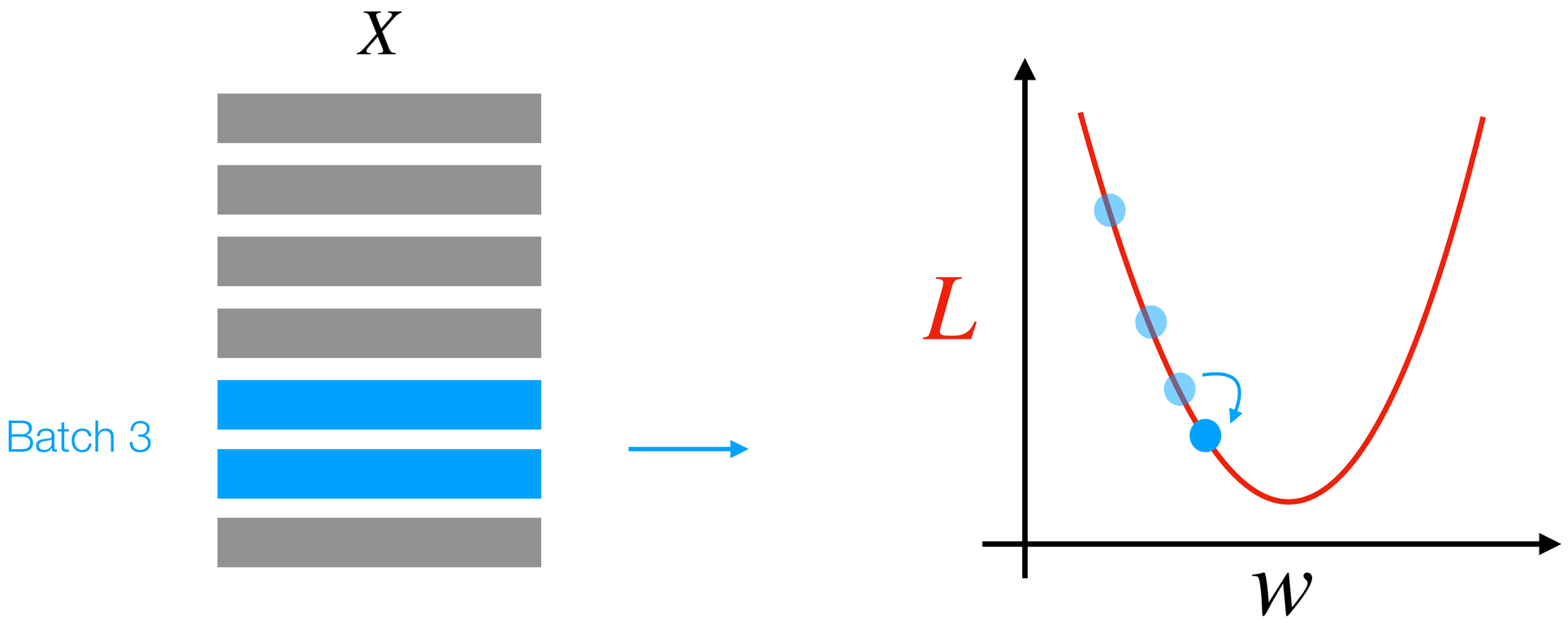
Beware of randomness


X


L
w

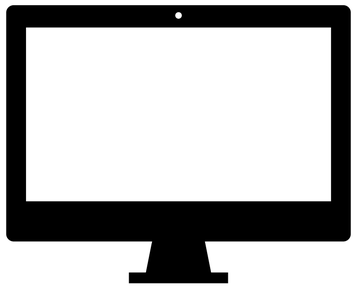

X
Batch 1
L
w

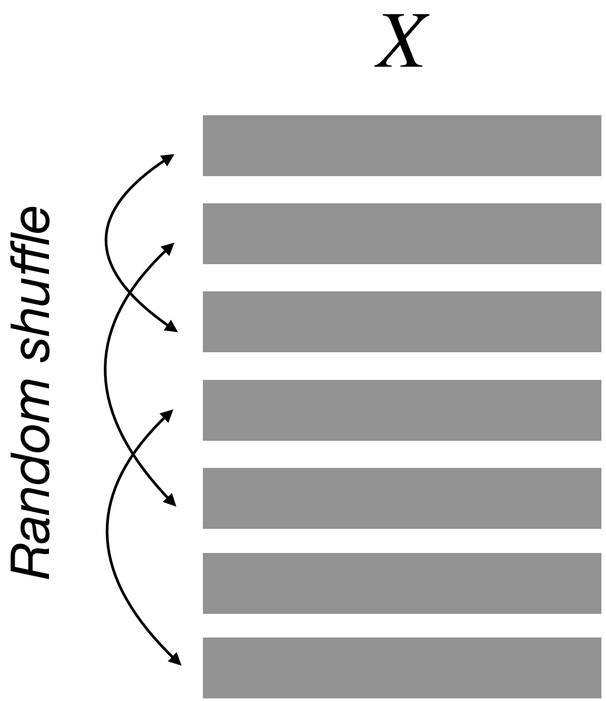

X
Batch 2
L
w


X
Batch 3
L
w


X
Random shuffle
Sequence in which data is
seen influences model performance:
Random state
0.70
0.75
0.80
0.85
0.90
0.95
1.00
Mean Accuracy

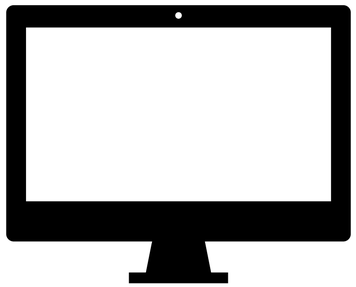
Pseudo**R**andom
Number **G**enerator





Sequence in which data is
seen by model during fitting

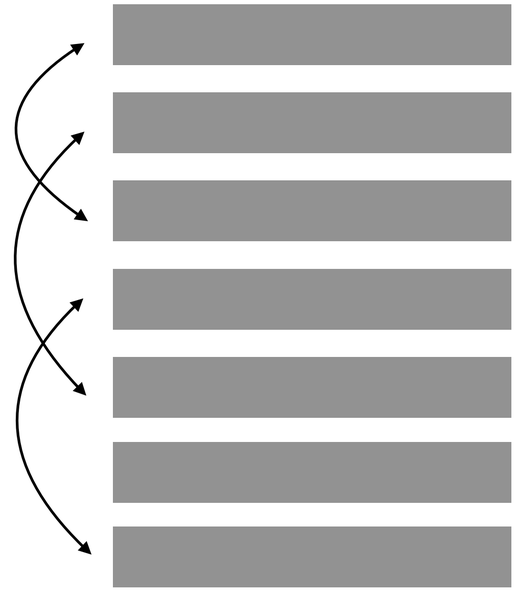
Pseudo**R**andom
Number **G**enerator



Random Seed
[start value for PNRG]

X

Random shuffle



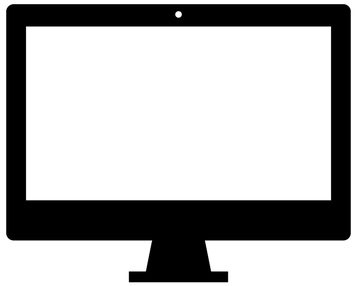
Sequence in which data is
seen by model during fitting

Random state

0.70 0.75 0.80

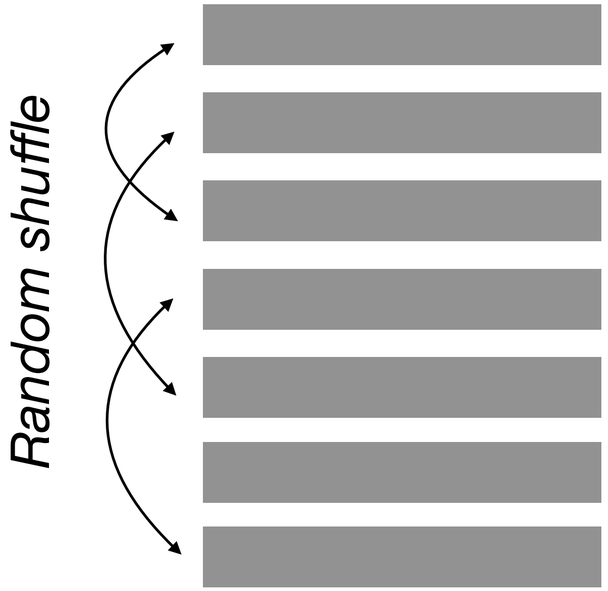
Mean

Pseudo**R**andom
Number **G**enerator



```
np.random.seed( 2368 )
```

X



Sequence in which data is
seen by model during fitting

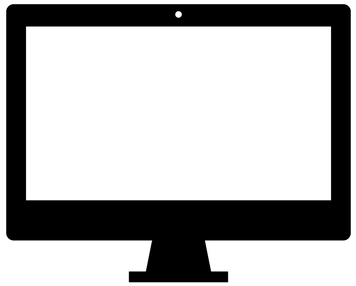
Random state

0.70 0.75 0.80

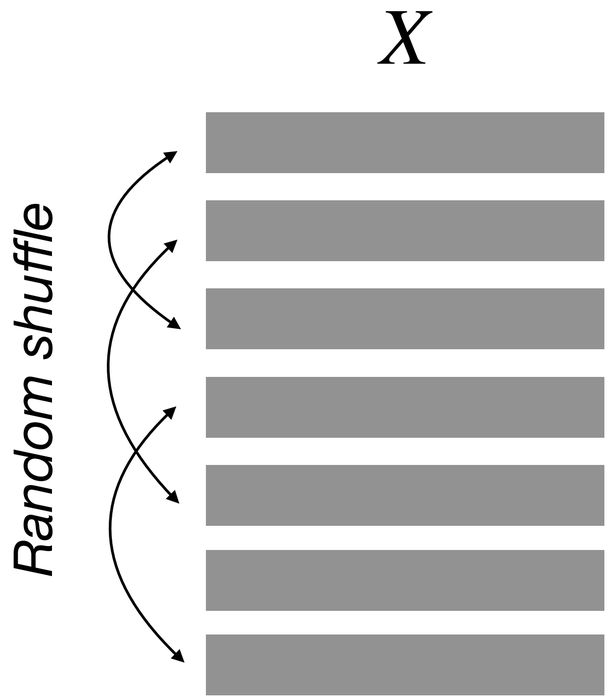
Mean

Set random seeds whenever your code uses PRNG!

Pseudo**R**andom
Number **G**enerator

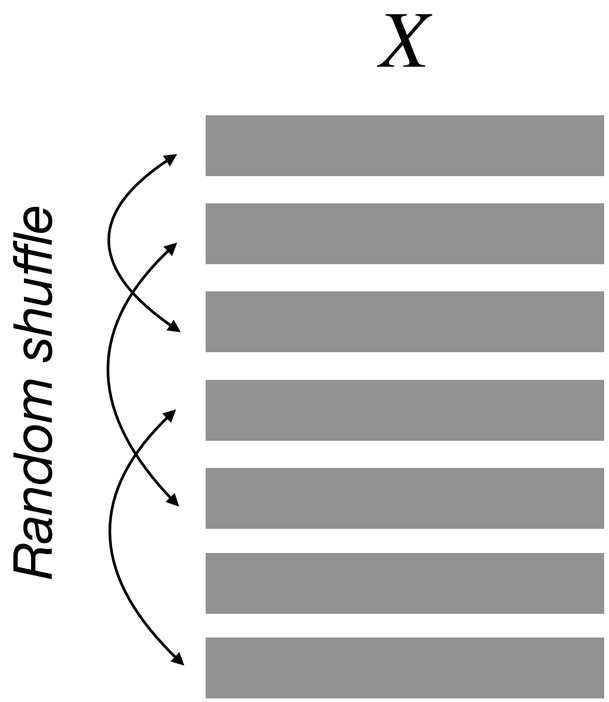


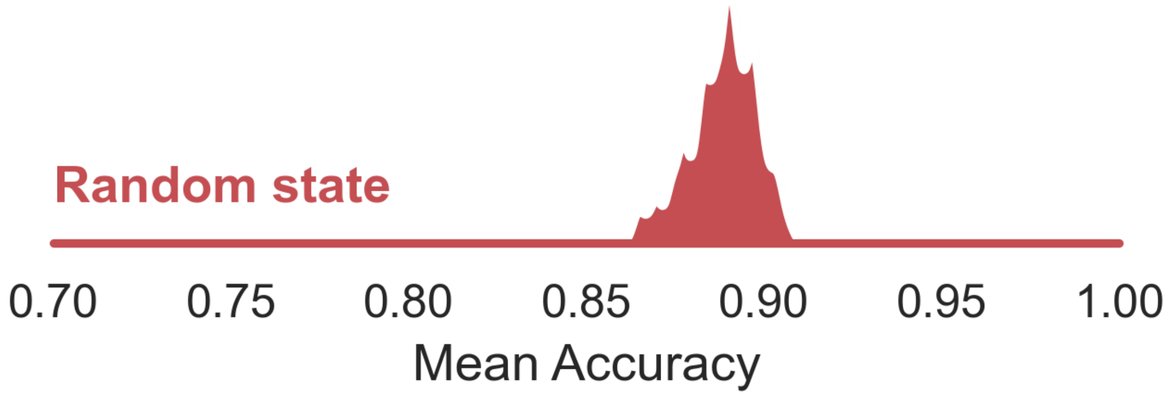
```
np.random.seed( 2368 )
```



Sequence in which data is seen by model during fitting

Account for influence of randomness by randomly changing seed across multiple fitting runs and aggregating performance





Sequence in which data is seen by model during fitting

Improve inference reproducibility by ..

… being transparent about hyper-parameter evaluations

… accounting for different data splits with cross-validation

… accounting for random factors of training

…

Improve inference reproducibility by ..

… being transparent about hyper-parameter evaluations

… accounting for different data splits with cross-validation

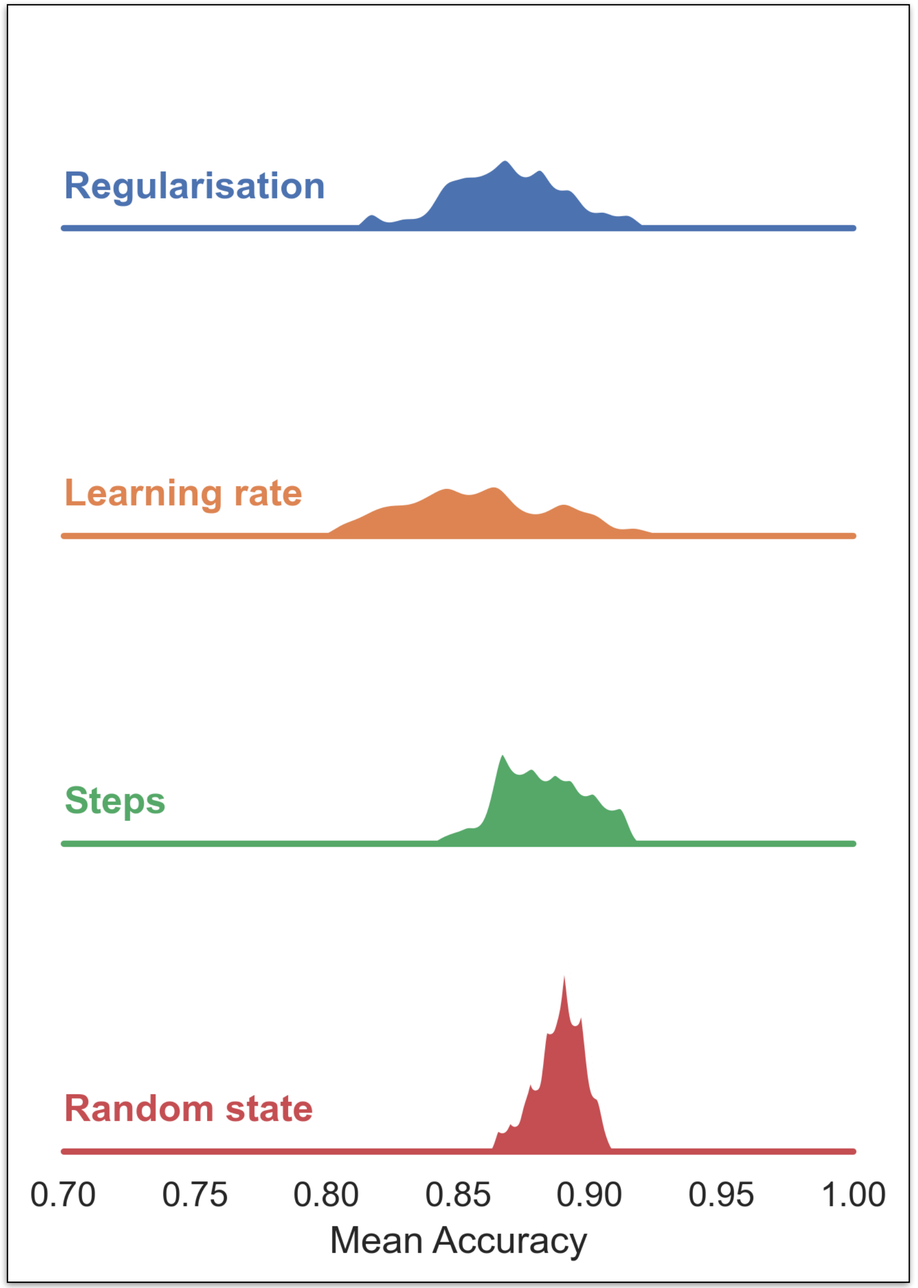
… accounting for random factors of training

Computational reproducibility

1. **Organising your project**
2. **Coding understandably and portably**
3. **Automating your analysis**
4. **Git & Github**

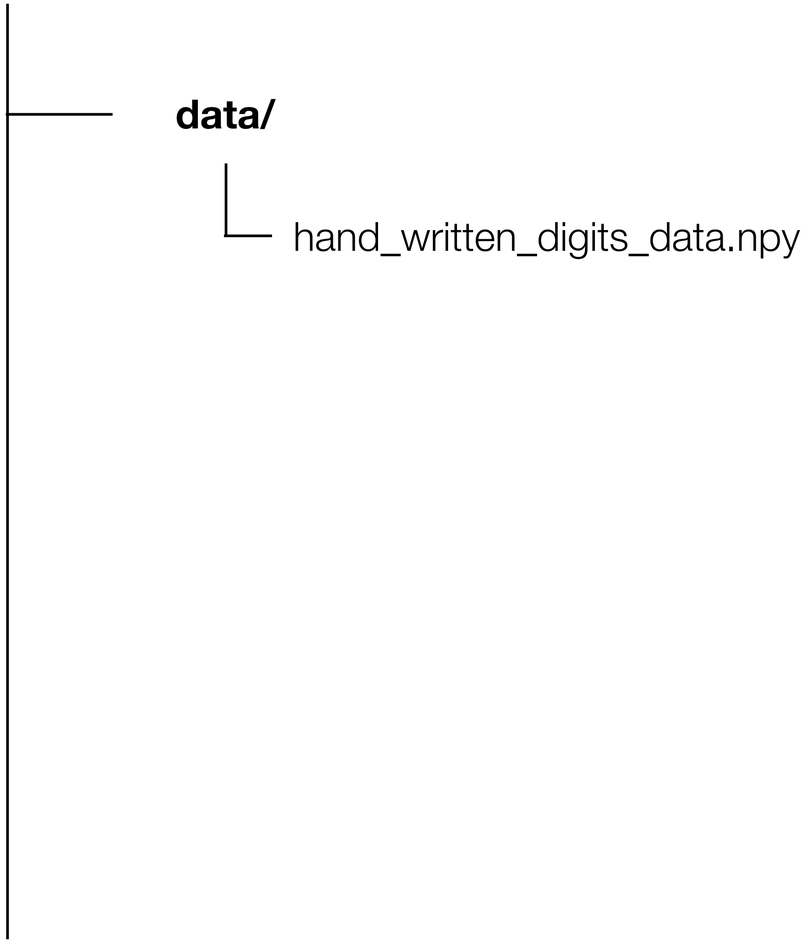
An example of a reproducible modelling project:

github.com/athms/reproducible-modelling



< reproducible-modelling >

< reproducible-modelling >



< reproducible-modelling >

- data/
- results/
  - estimates/
  - figures/

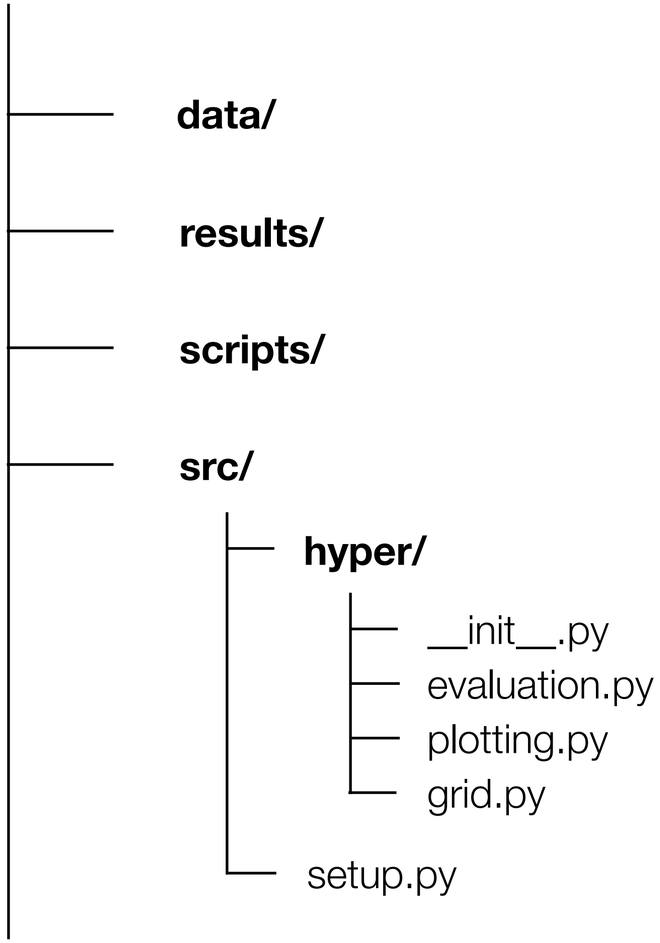
< reproducible-modelling >

- **data/**
- **results/**
  - **estimates/**
    - hyper_params_performance.csv
  - **figures/**
    - fig_hyper_params_effects.png

< reproducible-modelling >

- **data/**
- **results/**
- **scripts/**
  - load_data.py
  - evaluate_hyper_params_performance.py
  - plot_hyper_params_effect.py
  - run_analysis.sh

< reproducible-modelling >



< reproducible-modelling >

- **data/**
- **results/**
- **scripts/**
- **src/**
- README.md -> explain your code and repository

< reproducible-modelling >

- **data/**
- **results/**
- **scripts/**
- **src/**
- README.md -> explain your code and repository
- LICENSE -> License for your project

Code understandably

.. so that others can later reuse/understand your code

< reproducible-modelling >

- **data/**
- **results/**
- **scripts/**
- **src/**
- README.md -> explain your code and repository
- LICENSE -> License for your project

Code understandably

— so that others can later reuse/understand your code

< README.md >

Often written in markdown:

```
# An example of a reproducible modelling project


## What are we doing?

This example was created for the [2021 fall lecture series](https://datascience.stanford.edu/news/center-open-and-reproducible-science-cores-fall-lecture-series) of [Stanford's Center for Open and REproducible Science (CORES)](https://datascience.stanford.edu/cores).

The goal of this analysis is to study the effect of varying different hyper-parameters of the training of a simple classification model on its performance in [scikit-learn's handwritten digit dataset](https://scikit-learn.org/stable/datasets/toy_dataset.html#digits-dataset).

Specifically, we will study the effect of individually varying the learning rate, regularisation strength, number of gradient descent iterations, and random shuffling of the data on the 3-fold cross-validation performance of [scikit-learn's default linear one-vs-rest SVM classifier](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html).

Importantly, we vary each hyper-parameter separately while all other hyper-parameters are set to default values (see [scripts/evaluate_hyper_params_effect.py](scripts/evaluate_hyper_params_effect.py)).

```

Code understandably

— so that others can later reuse/understand your code

What is the purpose of this project?

< README.md >

README.md

An example of a reproducible modelling project

What are we doing?

This example was created for the 2021 fall lecture series of Stanford's Center for Open and REproducible Science (CORES).

The goal of this analysis is to study the effect of varying different hyper-parameters of the training of a simple classification model on its performance in scikit-learn's handwritten digit dataset.

Specifically, we will study the effect of individually varying the learning rate, regularisation strength, number of gradient descent iterations, and random shuffling of the data on the 3-fold cross-validation performance of scikit-learn's default linear one-vs-rest SVM classifier.

Importantly, we vary each hyper-parameter separately while all other hyper-parameters are set to default values (see scripts/evaluate_hyper_params_effect.py).

Project organization

```
├── LICENSE            <- MIT License
├── Makefile           <- Makefile with targets to 'load', 'evaluate', and 'plot' ('make all' runs a
├── poetry.lock        <- Details of used package versions
├── pyproject.toml     <- Lists all dependencies
├── README.md          <- This README file.
├── data/
│   └──                <- A copy of the handwritten digit dataset provided by scikit-learn
│
├── results/
│   ├── estimates/
│   │   └──            <- Generated estimates of classifier performance
│   └── figures/
```

Code understandably

— so that others can later reuse/understand your code

How is this project organised?

< README.md >

README.md

Project organization

```
├── LICENSE            <- MIT License
├── Makefile           <- Makefile with targets to 'load', 'evaluate', and 'plot' ('make all' runs a
├── poetry.lock        <- Details of used package versions
├── pyproject.toml     <- Lists all dependencies
├── README.md          <- This README file.
├── data/
│   └──                <- A copy of the handwritten digit dataset provided by scikit-learn
│
├── results/
│   ├── estimates/
│   │   └──            <- Generated estimates of classifier performance
│   └── figures/
│       └──            <- Generated figures
│
├── scrips/
│   ├── load_data.py                       <- Downloads the dataset to specified 'data-path'
│   ├── evaluate_hyper_params_effect.py    <- Runs cross-validated hyper-parameter evaluation
│   ├── plot_hyper_params_effect.py        <- Summarizes results of evaluation in a figure
│   └── run_analysis.sh                    <- Runs all analysis steps
│
└── src/
    ├── hyper/
    │   ├── __init__.py                    <- Makes 'hyper' a Python module
    │   ├── grid.py                        <- Functionality to sample hyper-parameter grid
    │   ├── evaluation.py                  <- Functionality to evaluate classifier performance, give
    │   └── plotting.py                    <- Functionality to visualize results
    └── setup.py                           <- Makes 'hyper' pip-installable (pip install -e .)
```

Data description

We use the handwritten digits dataset provided by scikit-learn. For details on this dataset, see scikit-learn's

Code understandably

— so that others can later reuse/understand your code

How are data formatted?

< README.md >

README.md

Data description

We use the handwritten digits dataset provided by scikit-learn. For details on this dataset, see scikit-learn's documentation:

https://scikit-learn.org/stable/datasets/toy_dataset.html#digits-dataset

Installation

This project is written for Python 3.9.5 (we recommend pyenv for Python version management).

All software dependencies of this project are managed with Python Poetry. All details about the used package versions are provided in pyproject.toml.

To clone this repository to your local machine, run:

```
git clone https://github.com/athms/reproducible-modelling
```

To install all dependencies with `poetry`, run:

```
cd reproducible-modelling/
poetry install
```

To reproduce our analyses, you additionally need to install our custom Python module (src/hyper) in your `poetry` environment:

```
cd src/
poetry run pip install -e .
```

Code understandably

— so that others can later reuse/understand your code

How to install all dependencies?

< README.md >

README.md

Data description

We use the handwritten digits dataset provided by scikit-learn. For details on this dataset, see scikit-learn's documentation:

https://scikit-learn.org/stable/datasets/toy_dataset.html#digits-dataset

Installation

This project is written for Python 3.9.5 (we recommend pyenv for Python version management).

All software dependencies of this project are managed with Python Poetry. All details about the used package versions are provided in pyproject.toml.

To clone this repository to your local machine, run:

```
git clone https://github.com/athms/reproducible-modelling
```

To install all dependencies with `poetry`, run:

```
cd reproducible-modelling/
poetry install
```

To reproduce our analyses, you additionally need to install our custom Python module (src/hyper) in your `poetry` environment:

```
cd src/
poetry run pip install -e .
```

Code understandably

— so that others can later reuse/understand your code

How to reproduce analyses?

< README.md >

README.md

Reproducing our analysis

Our analysis can be reproduced either by running scripts/run_analysis.sh:

```
cd scripts
poetry run bash run_analysis.sh
```

..or by the use of `make`:

```
poetry run make <ANALYSIS TARGET>
```

We provide the following targets for `make`:

Analysis target	Description
all	Runs the entire analysis pipeline
load	Downloads scikit-learn's handwritten digit dataset
evaluate	Runs our cross-validated hyper-parameter evaluation
plot	Creates our results figure

This README file is strongly inspired by the Cookiecutter Data Science Structure

< reproducible-modelling >

- **data/**
- **results/**
- **scripts/**
  - load_data.py
  - evaluate_hyper_params_performance.py
  - plot_hyper_params_effect.py
  - run_analysis.sh

Code understandably

— so that others can later reuse/understand your code

a. Begin script by briefly describing what it does

```
#!/usr/bin/env python3

"""
This script evaluates the effect of varying different factors of the
training of a classification model on its predictive performance in
sklearn's handwritten digit dataset.

Each factor is varied separately while all other factors are held
constant during training. Performance evaluation is performed by the
use of 3-fold cross-validation.

This script requires that sklearn's handwritten digit dataset is stored
in 'data-path'. This can be achieved by first running 'load_data.py'
or 'make load'.

The script outputs a dataframe to 'results-path', which contains the
classifier's predictive performance in each cross-validation fold.

Author: Armin W. Thomas; athms@stanford.edu
"""
```

Code understandably

— so that others can later reuse/understand your code

a. Begin script by briefly describing what it does

What does script compute?

```
#!/usr/bin/env python3

"""
This script evaluates the effect of varying different factors of the
training of a classification model on its predictive performance in
sklearn's handwritten digit dataset.

Each factor is varied separately while all other factors are held
constant during training. Performance evaluation is performed by the
use of 3-fold cross-validation.

This script requires that sklearn's handwritten digit dataset is stored
in 'data-path'. This can be achieved by first running 'load_data.py'
or 'make load'.

The script outputs a dataframe to 'results-path', which contains the
classifier's predictive performance in each cross-validation fold.

Author: Armin W. Thomas; athms@stanford.edu
"""
```

Code understandably

— so that others can later reuse/understand your code

a. Begin script by briefly describing what it does

What does script compute?

in / outputs

```
#!/usr/bin/env python3

"""
This script evaluates the effect of varying different factors of the
training of a classification model on its predictive performance in
sklearn's handwritten digit dataset.

Each factor is varied separately while all other factors are held
constant during training. Performance evaluation is performed by the
use of 3-fold cross-validation.

This script requires that sklearn's handwritten digit dataset is stored
in 'data-path'. This can be achieved by first running 'load_data.py'
or 'make load'.

The script outputs a dataframe to 'results-path', which contains the
classifier's predictive performance in each cross-validation fold.

Author: Armin W. Thomas; athms@stanford.edu
"""
```

Code understandably

— so that others can later reuse/understand your code

a. Begin script by briefly describing what it does

What does script compute?

in / outputs

author

```
#!/usr/bin/env python3

"""
This script evaluates the effect of varying different factors of the
training of a classification model on its predictive performance in
sklearn's handwritten digit dataset.

Each factor is varied separately while all other factors are held
constant during training. Performance evaluation is performed by the
use of 3-fold cross-validation.

This script requires that sklearn's handwritten digit dataset is stored
in 'data-path'. This can be achieved by first running 'load_data.py'
or 'make load'.

The script outputs a dataframe to 'results-path', which contains the
classifier's predictive performance in each cross-validation fold.

Author: Armin W. Thomas; athms@stanford.edu
"""
```

Code understandably

— so that others can later reuse/understand your code

b. Define operations in pseudocode

Pseudocode programming:

```
11    in ../data/. This can be achieved by running load_data.py first.
12
13    This script outputs a dataframe to ../results/data/, which stores the
14    classifier's predictive performance in each cross-validation fold
15
16    Author: Armin W. Thomas; athms@stanford.edu
17    """
18
19
20    # 1. load handwritten digits dataset
21    # from ../data/
22
23    # 2. define our classification model
24    # (we choose SGDClassifier from sklearn)
25
26    # 3. define a set of hyper-parameters
27    # with hyper module
28
29    # 4. evaluate classifier performance for
30    # the specified hyper-parameter values
31    # with hyper
32
33    # 5. save the results to ../results/data/
```

Code understandably

— so that others can later reuse/understand your code

b. Define operations in pseudocode

Pseudocode programming:

Individual computing steps in plain language

```
11  in ../data/. This can be achieved by running load_data.py first.
12
13  This script outputs a dataframe to ../results/data/, which stores the
14  classifier's predictive performance in each cross-validation fold
15
16  Author: Armin W. Thomas; athms@stanford.edu
17  """
18
19
20  # 1. load handwritten digits dataset
21  # from ../data/
22
23  # 2. define our classification model
24  # (we choose SGDClassifier from sklearn)
25
26  # 3. define a set of hyper-parameters
27  # with hyper module
28
29  # 4. evaluate classifier performance for
30  # the specified hyper-parameter values
31  # with hyper
32
33  # 5. save the results to ../results/data/
```

Code understandably

— so that others can later reuse/understand your code

c. Write code around pseudocode

Pseudocode programming:

```
27  def main():
28
29      # 1. load handwritten digits dataset
30      # from ../data/
31      data = np.load(
32          '../data/hand_written_digits_data.npy',
33          allow_pickle=True
34      )[()]
35
36      # 2. define our classification model
37      # (we choose SGDClassifier from sklearn)
38      classifier = sklearn.linear_model.SGDClassifier
39
40      # 3. define a set of hyper-parameters
41      # with hyper module
42      hyper_params = hyper.create_hyper_params(
43          n=100
44      )
45
46      # 4. evaluate classifier performance for
47      # the specified hyper-parameter values
48      # with hyper
49      results = hyper.evaluate_hyper_params_effects(
50          X=data["data"],
```

Code understandably

— so that others can later reuse/understand your code

→ Pseudocode programming:

→ Stick to conventions:

PEP8

google.github.io/styleguide/

```
27  def main():
28
29      # 1. load handwritten digits dataset
30      # from ../data/
31      data = np.load(
32          '../data/hand_written_digits_data.npy',
33          allow_pickle=True
34      )[()]
35
36      # 2. define our classification model
37      # (we choose SGDClassifier from sklearn)
38      classifier = sklearn.linear_model.SGDClassifier
39
40      # 3. define a set of hyper-parameters
41      # with hyper module
42      hyper_params = hyper.create_hyper_params(
43          n=100
44      )
45
46      # 4. evaluate classifier performance for
47      # the specified hyper-parameter values
48      # with hyper
49      results = hyper.evaluate_hyper_params_effects(
50          X=data["data"],
```

Code understandably

— so that others can later reuse/understand your code

a. snake_case for variables and modules

→ Pseudocode programming:

→ Stick to conventions:

PEP8

google.github.io/styleguide/

```
27  def main():
28
29      # 1. load handwritten digits dataset
30      # from ../data/
31      data = np.load(
32          '../data/hand_written_digits_data.npy',
33          allow_pickle=True
34      )[()]
35
36      # 2. define our classification model
37      # (we choose SGDClassifier from sklearn)
38      classifier = sklearn.linear_model.SGDClassifier
39
40      # 3. define a set of hyper-parameters
41      # with hyper module
42      hyper_params = hyper.create_hyper_params(
43          n=100
44      )
45
46      # 4. evaluate classifier performance for
47      # the specified hyper-parameter values
48      # with hyper
49      results = hyper.evaluate_hyper_params_effects(
50          X=data["data"],
```

Code understandably

— so that others can later reuse/understand your code

b. CamelCase for classes

→ Pseudocode programming:

→ Stick to conventions:

PEP8

google.github.io/styleguide/

```
27  def main():
28
29      # 1. load handwritten digits dataset
30      # from ../data/
31      data = np.load(
32          '../data/hand_written_digits_data.npy',
33          allow_pickle=True
34      )[()]
35
36      # 2. define our classification model
37      # (we choose SGDClassifier from sklearn)
38      classifier = sklearn.linear_model.SGDClassifier
39
40      # 3. define a set of hyper-parameters
41      # with hyper module
42      hyper_params = hyper.create_hyper_params(
43          n=100
44      )
45
46      # 4. evaluate classifier performance for
47      # the specified hyper-parameter values
48      # with hyper
49      results = hyper.evaluate_hyper_params_effects(
50          X=data["data"],
```

Code understandably

— so that others can later reuse/understand your code

c. Indent with 4 spaces

→ Pseudocode programming:

→ Stick to conventions:

PEP8

google.github.io/styleguide/

```
27  def main():
28
29      # 1. load handwritten digits dataset
30      # from ../data/
31      data = np.load(
32          '../data/hand_written_digits_data.npy',
33          allow_pickle=True
34      )[()]
35
36      # 2. define our classification model
37      # (we choose SGDClassifier from sklearn)
38      classifier = sklearn.linear_model.SGDClassifier
39
40      # 3. define a set of hyper-parameters
41      # with hyper module
42      hyper_params = hyper.create_hyper_params(
43          n=100
44      )
45
46      # 4. evaluate classifier performance for
47      # the specified hyper-parameter values
48      # with hyper
49      results = hyper.evaluate_hyper_params_effects(
50          X=data["data"],
```

Code understandably

— so that others can later reuse/understand your code

a. Write docstrings

Pseudocode programming:

Stick to conventions:

Document well:

```
6   def sample_hyper_params(
7       n: int=10,
8       param_bounds: dict={
9           'alpha': (0.000001, 0.001),
10          'eta0': (0.001, 1),
11          'steps': (1, 100)
12          }
13      ) -> dict:
14      """Sample 'n' equally-spaced values for
15      {alpha, eta0, steps} between specified
16      parameter bounds, in addition to
17      'n' random seed values.
18
19      Args:
20          n (int, optional): How many paramater values to sample
21              for each hyper-parameter? Defaults to 10.
22          param_bounds (dict, optional): Parameter bounds.
23              Defaults to { 'alpha': (0.000001, 0.001),
24                            'eta0': (0.001, 1),
25                            'steps': (1, 100) }.
26
27      Returns:
28          dict: Individual parameter value grids.
29      """
```

Code understandably

— so that others can later reuse/understand your code

a. Write docstrings

Pseudocode programming:

Stick to conventions:

Document well:

What does function do?

```
6   def sample_hyper_params(
7       n: int=10,
8       param_bounds: dict={
9           'alpha': (0.000001, 0.001),
10          'eta0': (0.001, 1),
11          'steps': (1, 100)
12          }
13      ) -> dict:
14      """Sample 'n' equally-spaced values for
15      {alpha, eta0, steps} between specified
16      parameter bounds, in addition to
17      'n' random seed values.
18
19      Args:
20          n (int, optional): How many paramater values to sample
21              for each hyper-parameter? Defaults to 10.
22          param_bounds (dict, optional): Parameter bounds.
23              Defaults to { 'alpha': (0.000001, 0.001),
24                            'eta0': (0.001, 1),
25                            'steps': (1, 100) }.
26
27      Returns:
28          dict: Individual parameter value grids.
29      """
```

Code understandably

— so that others can later reuse/understand your code

a. Write docstrings

Pseudocode programming:

Stick to conventions:

Document well:

Specify inputs

```
6   def sample_hyper_params(
7       n: int=10,
8       param_bounds: dict={
9           'alpha': (0.000001, 0.001),
10          'eta0': (0.001, 1),
11          'steps': (1, 100)
12          }
13      ) -> dict:
14      """Sample 'n' equally-spaced values for
15      {alpha, eta0, steps} between specified
16      parameter bounds, in addition to
17      'n' random seed values.
18
19      Args:
20          n (int, optional): How many paramater values to sample
21              for each hyper-parameter? Defaults to 10.
22          param_bounds (dict, optional): Parameter bounds.
23              Defaults to { 'alpha': (0.000001, 0.001),
24                            'eta0': (0.001, 1),
25                            'steps': (1, 100) }.
26
27      Returns:
28          dict: Individual parameter value grids.
29      """
```

Code understandably

— so that others can later reuse/understand your code

a. Write docstrings

Pseudocode programming:

Stick to conventions:

Document well:

Specify outputs

```
6   def sample_hyper_params(
7       n: int=10,
8       param_bounds: dict={
9           'alpha': (0.000001, 0.001),
10          'eta0': (0.001, 1),
11          'steps': (1, 100)
12          }
13      ) -> dict:
14      """Sample 'n' equally-spaced values for
15      {alpha, eta0, steps} between specified
16      parameter bounds, in addition to
17      'n' random seed values.
18
19      Args:
20          n (int, optional): How many paramater values to sample
21              for each hyper-parameter? Defaults to 10.
22          param_bounds (dict, optional): Parameter bounds.
23              Defaults to { 'alpha': (0.000001, 0.001),
24                            'eta0': (0.001, 1),
25                            'steps': (1, 100) }.
26
27      Returns:
28          dict: Individual parameter value grids.
29      """
```

Code portably

..so that your code can run on other machines

Code portably

— so that your code can run on other machines

Always use relative paths!

```
3    data_path = "/home/Users/thomas/projects/reproducible-modelling/data"
```

Will only work on your machine

Code portably

— so that your code can run on other machines

Always use relative paths!

```
3   data_path = "/home/users/thomas/projects/reproducible-modelling/data"
```

```
3   data_path = "../data"
```

Relative to location of your script (e.g., scripts/)

Code portably

— so that your code can run on other machines

Always use relative paths!

```
3   data_path = "/home/users/thomas/projects/reproducible-modelling/data"
```

```
3   data_path = "../data"
```

Share your computing environment

Virtual environments: Isolated directory that contains a particular version of Python as well as a specific set of packages

Code portably

— so that your code can run on other machines

Always use relative paths!

```
3   data_path = "/home/users/thomas/projects/reproducible-modelling/data"
```

```
3   data_path = "../data"
```

Share your computing environment

Virtual environments: Isolated directory that contains a particular version of Python as well as a specific set of packages

Python:

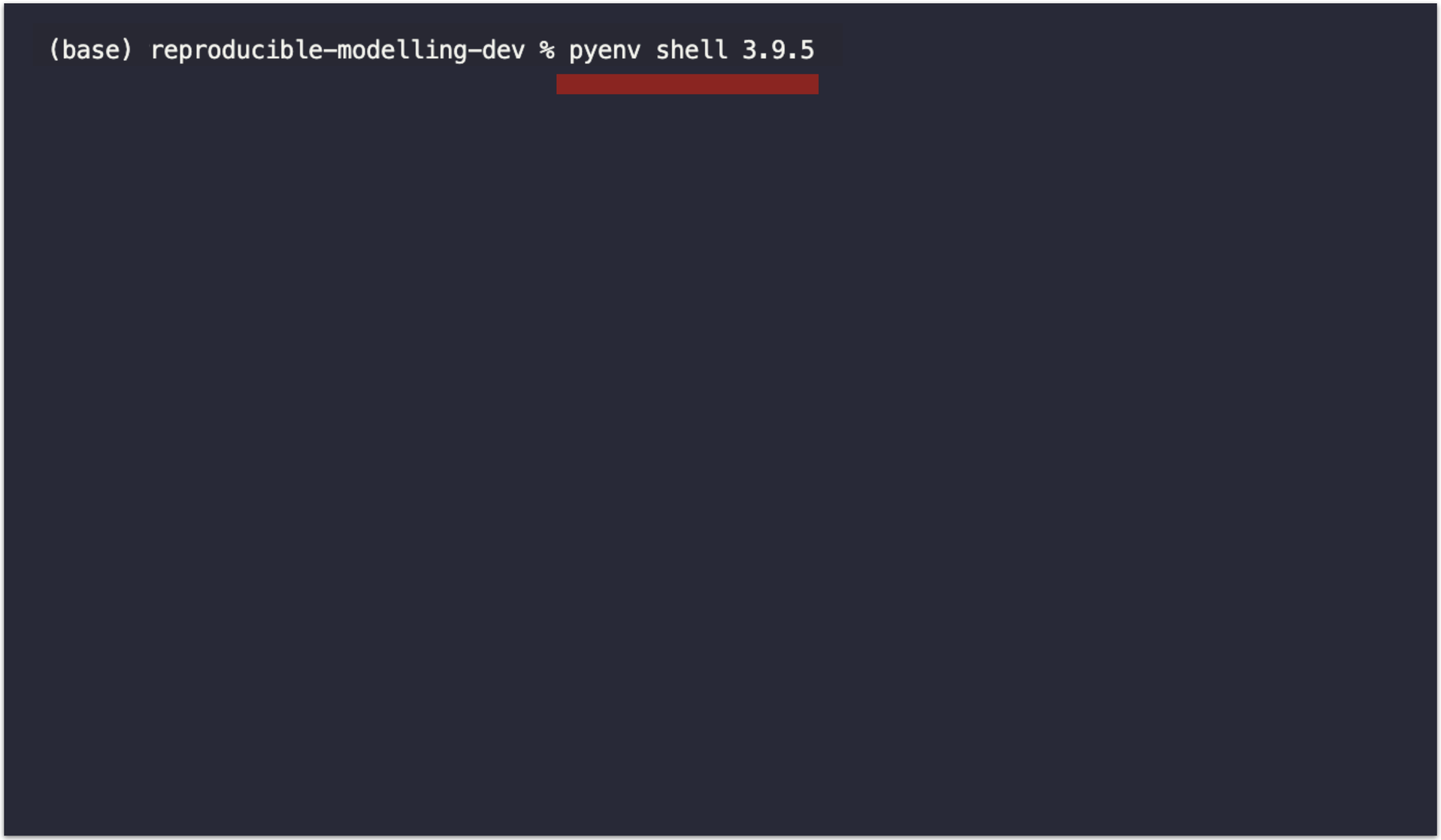
- pipenv
- Anaconda
- **Poetry**
- …

Code portably

— so that your code can run on other machines

a. specify environment

Activate Python version
(here with pyenv)

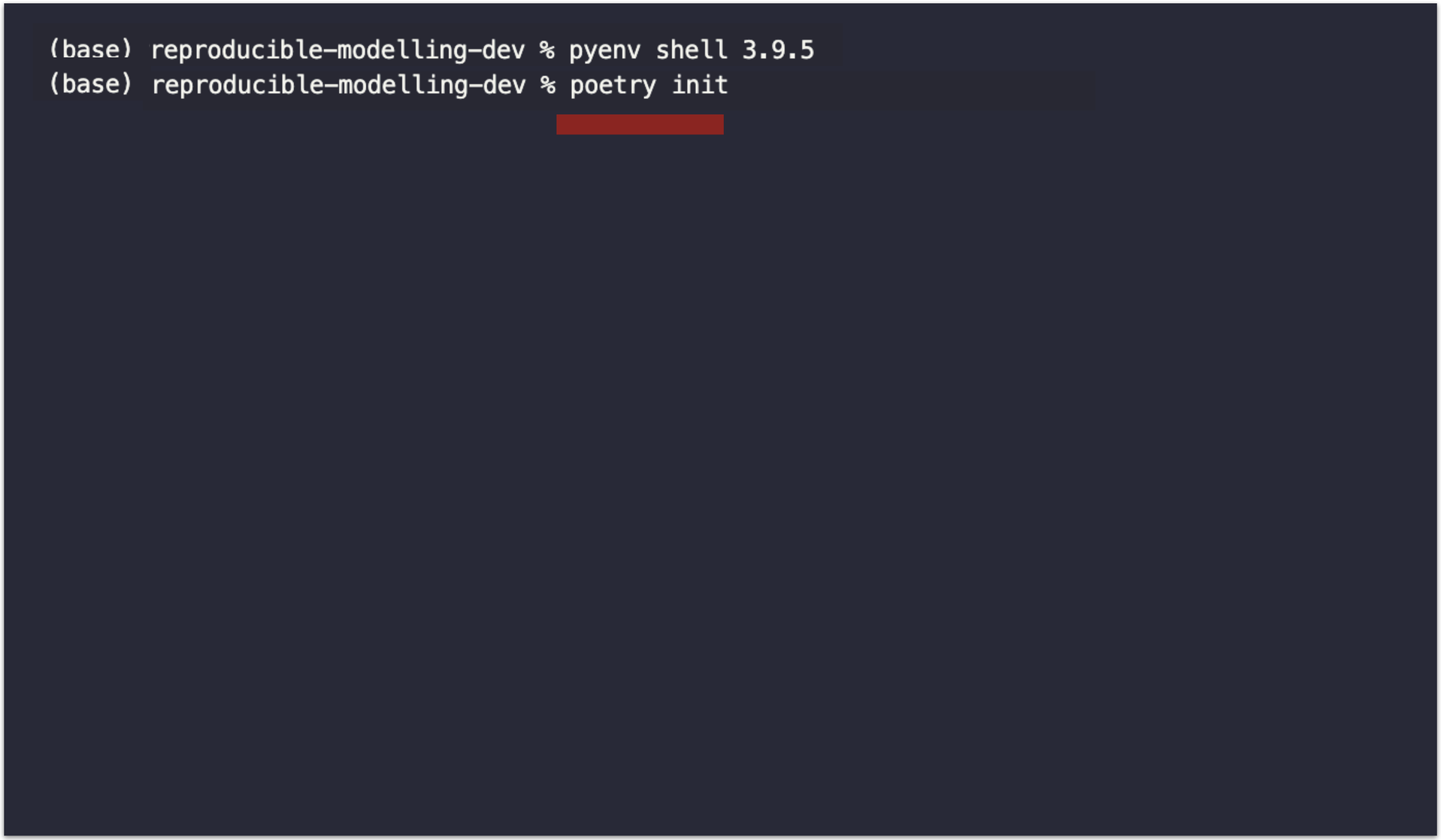


Code portably

— so that your code can run on other machines

b. init poetry

Initialise Python Poetry



Code portably

— so that your code can run on other machines

c. Initialise your project

Config file specifying our environment / project

```
(base) thomas@lip-osx-005002 reproducible-modelling % poetry init

This command will guide you through creating your pyproject.toml config.
```

Code portably

— so that your code can run on other machines

```
(base) thomas@lip-osx-005002 reproducible-modelling % poetry init

This command will guide you through creating your pyproject.toml config.

Package name [reproducible-modelling]:
```

Code portably

— so that your code can run on other machines

```
(base) thomas@lip-osx-005002 reproducible-modelling % poetry init

This command will guide you through creating your pyproject.toml config.

Package name [reproducible-modelling]:
Version [0.1.0]:
```

Code portably

— so that your code can run on other machines

```
(base) thomas@lip-osx-005002 reproducible-modelling % poetry init

This command will guide you through creating your pyproject.toml config.

Package name [reproducible-modelling]:
Version [0.1.0]:
Description []:  Exemplary reproducible modelling project
```

Code portably

— so that your code can run on other machines

```
(base) thomas@lip-osx-005002 reproducible-modelling % poetry init

This command will guide you through creating your pyproject.toml config.

Package name [reproducible-modelling]:
Version [0.1.0]:
Description []:  Exemplary reproducible modelling project
Author [                              n to skip]:  Armin W. Thomas <athms@stanford.edu>
```

Code portably

— so that your code can run on other machines

```
(base) thomas@lip-osx-005002 reproducible-modelling % poetry init

This command will guide you through creating your pyproject.toml config.

Package name [reproducible-modelling]:
Version [0.1.0]:
Description []:  Exemplary reproducible modelling project
Author [                              n to skip]:  Armin W. Thomas <athms@stanford.edu>
License []:  MIT
```

Code portably

— so that your code can run on other machines

```
(base) thomas@lip-osx-005002 reproducible-modelling % poetry init

This command will guide you through creating your pyproject.toml config.

Package name [reproducible-modelling]:
Version [0.1.0]:
Description []:  Exemplary reproducible modelling project
Author [                                n to skip]:  Armin W. Thomas <athms@stanford.edu>
License []:  MIT
Compatible Python versions [^3.9]:
```

Code portably

— so that your code can run on other machines

```
(base) thomas@lip-osx-005002 reproducible-modelling % poetry init

This command will guide you through creating your pyproject.toml config.

Package name [reproducible-modelling]:
Version [0.1.0]:
Description []:  Exemplary reproducible modelling project
Author [                                n to skip]:  Armin W. Thomas <athms@stanford.edu>
License []:  MIT
Compatible Python versions [^3.9]:

Would you like to define your main dependencies interactively? (yes/no) [yes] yes
You can specify a package in the following forms:
  - A single name (requests)
  - A name and a constraint (requests@^2.23.0)
  - A git url (git+https://github.com/python-poetry/poetry.git)
  - A git url with a revision (git+https://github.com/python-poetry/poetry.git#develop)
  - A file path (../my-package/my-package.whl)
  - A directory (../my-package/)
  - A url (https://example.com/packages/my-package-0.1.0.tar.gz)
```

Code portably

— so that your code can run on other machines

```
(base) thomas@lip-osx-005002 reproducible-modelling % poetry init

This command will guide you through creating your pyproject.toml config.

Package name [reproducible-modelling]:
Version [0.1.0]:
Description []:  Exemplary reproducible modelling project
Author [                                n to skip]:  Armin W. Thomas <athms@stanford.edu>
License []:  MIT
Compatible Python versions [^3.9]:

Would you like to define your main dependencies interactively? (yes/no) [yes] yes
You can specify a package in the following forms:
  - A single name (requests)
  - A name and a constraint (requests@^2.23.0)
  - A git url (git+https://github.com/python-poetry/poetry.git)
  - A git url with a revision (git+https://github.com/python-poetry/poetry.git#develop)
  - A file path (../my-package/my-package.whl)
  - A directory (../my-package/)
  - A url (https://example.com/packages/my-package-0.1.0.tar.gz)

Search for package to add (or leave blank to continue): numpy@1.21.1
Adding numpy@1.21.1
```

Code portably

— so that your code can run on other machines

```
Add a package: pandas@1.3.3
Adding pandas@1.3.3

Add a package: scikit-learn@1.0
Adding scikit-learn@1.0

Add a package: seaborn@0.11.2
Adding seaborn@0.11.2

Add a package: matplotlib@3.4.3
Adding matplotlib@3.4.3

Add a package:

Would you like to define your development dependencies interactively? (yes/no) [yes]
Search for package to add (or leave blank to continue):
```

Code portably

— so that your code can run on other machines

Project / author info

```
Add a package:

Would you like to define your development dependencies interactively? (yes/no) [yes]
Search for package to add (or leave blank to continue):

Generated file

[tool.poetry]
name = "reproducible-modelling"
version = "0.1.0"
description = "Exemplary reproducible modelling project"
authors = ["Armin W. Thomas <athms@stanford.edu>"]
license = "MIT"

[tool.poetry.dependencies]
python = "^3.9"
numpy = "1.21.1"
pandas = "1.3.3"
scikit-learn = "1.0"
seaborn = "0.11.2"
matplotlib = "3.4.3"

[tool.poetry.dev-dependencies]

[build-system]
requires = ["poetry-core>=1.0.0"]
build-backend = "poetry.core.masonry.api"


Do you confirm generation? (yes/no) [yes]
```

Code portably

— so that your code can run on other machines

Dependencies

```
Add a package:

Would you like to define your development dependencies interactively? (yes/no) [yes]
Search for package to add (or leave blank to continue):

Generated file

[tool.poetry]
name = "reproducible-modelling"
version = "0.1.0"
description = "Exemplary reproducible modelling project"
authors = ["Armin W. Thomas <athms@stanford.edu>"]
license = "MIT"

[tool.poetry.dependencies]
python = "^3.9"
numpy = "1.21.1"
pandas = "1.3.3"
scikit-learn = "1.0"
seaborn = "0.11.2"
matplotlib = "3.4.3"

[tool.poetry.dev-dependencies]

[build-system]
requires = ["poetry-core>=1.0.0"]
build-backend = "poetry.core.masonry.api"


Do you confirm generation? (yes/no) [yes]
```

Code portably

— so that your code can run on other machines

Poetry version

```
Add a package:

Would you like to define your development dependencies interactively? (yes/no) [yes]
Search for package to add (or leave blank to continue):

Generated file

[tool.poetry]
name = "reproducible-modelling"
version = "0.1.0"
description = "Exemplary reproducible modelling project"
authors = ["Armin W. Thomas <athms@stanford.edu>"]
license = "MIT"

[tool.poetry.dependencies]
python = "^3.9"
numpy = "1.21.1"
pandas = "1.3.3"
scikit-learn = "1.0"
seaborn = "0.11.2"
matplotlib = "3.4.3"

[tool.poetry.dev-dependencies]

[build-system]
requires = ["poetry-core>=1.0.0"]
build-backend = "poetry.core.masonry.api"


Do you confirm generation? (yes/no) [yes]
```

< reproducible-modelling >

- **data/**
- **results/**
- **scripts/**
- **src/**
- README.md
- pyproject.toml

Code portably

— so that your code can run on other machines

Installing dependencies

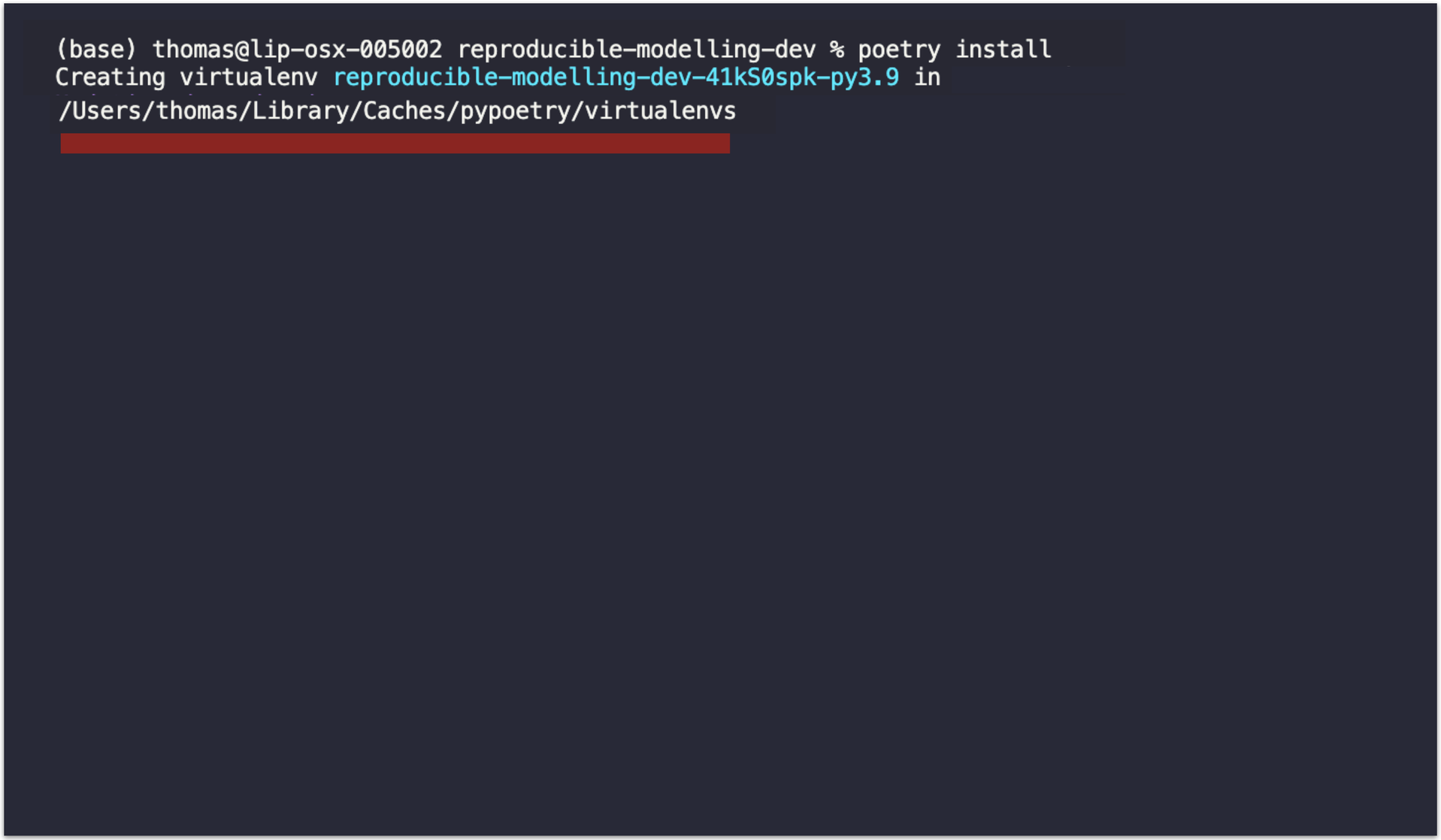
Code portably

— so that your code can run on other machines

Installing dependencies

Name of
our virtual environment

Directory of
our virtual environment



Code portably

— so that your code can run on other machines

Installing dependencies

```
(base) thomas@lip-osx-005002 reproducible-modelling-dev % poetry install
Creating virtualenv reproducible-modelling-dev-41kS0spk-py3.9 in
/Users/thomas/Library/Caches/pypoetry/virtualenvs

Updating dependencies
Resolving dependencies... (2.4s)
```

Code portably

— so that your code can run on other machines

Installing dependencies

```
(base) thomas@lip-osx-005002 reproducible-modelling-dev % poetry install
Creating virtualenv reproducible-modelling-dev-41kS0spk-py3.9 in
/Users/thomas/Library/Caches/pypoetry/virtualenvs

Updating dependencies
Resolving dependencies... (2.4s)

Writing lock file
```

Code portably

— so that your code can run on other machines

Installing dependencies

```
(base) thomas@lip-osx-005002 reproducible-modelling-dev % poetry install
Creating virtualenv reproducible-modelling-dev-41kS0spk-py3.9 in
/Users/thomas/Library/Caches/pypoetry/virtualenvs

Updating dependencies
Resolving dependencies... (2.4s)

Writing lock file

Package operations: 15 installs, 0 updates, 0 removals

  • Installing six (1.16.0)
  • Installing cycler (0.10.0)
  • Installing kiwisolver (1.3.2)
  • Installing numpy (1.21.1)
  • Installing pillow (8.4.0)
  • Installing pyparsing (2.4.7)
  • Installing python-dateutil (2.8.2)
  • Installing pytz (2021.3)
  • Installing joblib (1.1.0)
  • Installing matplotlib (3.4.3)
  • Installing pandas (1.3.3)
  • Installing scipy (1.6.1)
  • Installing threadpoolctl (3.0.0)
  • Installing scikit-learn (1.0)
  • Installing seaborn (0.11.2)
```

Code portably

— so that your code can run on other machines

Installing dependencies

```
(base) thomas@lip-osx-005002 reproducible-modelling-dev % poetry install
Creating virtualenv reproducible-modelling-dev-41kS0spk-py3.9 in
/Users/thomas/Library/Caches/pypoetry/virtualenvs

Updating dependencies
Resolving dependencies... (2.4s)

Writing lock file

Package operations: 15 installs, 0 updates, 0 removals

  • Installing six (1.16.0)
  • Installing cycler (0.10.0)
  • Installing kiwisolver (1.3.2)
  • Installing numpy (1.21.1)
  • Installing pillow (8.4.0)
  • Installing pyparsing (2.4.7)
  • Installing python-dateutil (2.8.2)
  • Installing pytz (2021.3)
  • Installing joblib (1.1.0)
  • Installing matplotlib (3.4.3)
  • Installing pandas (1.3.3)
  • Installing scipy (1.6.1)
  • Installing threadpoolctl (3.0.0)
  • Installing scikit-learn (1.0)
  • Installing seaborn (0.11.2)
(base) thomas@lip-osx-005002 reproducible-modelling-dev % ls
```

Code portably

— so that your code can run on other machines

Installing dependencies

```
/Users/thomas/Library/Caches/pypoetry/virtualenvs
Updating dependencies
Resolving dependencies... (2.4s)

Writing lock file

Package operations: 15 installs, 0 updates, 0 removals

  • Installing six (1.16.0)
  • Installing cycler (0.10.0)
  • Installing kiwisolver (1.3.2)
  • Installing numpy (1.21.1)
  • Installing pillow (8.4.0)
  • Installing pyparsing (2.4.7)
  • Installing python-dateutil (2.8.2)
  • Installing pytz (2021.3)
  • Installing joblib (1.1.0)
  • Installing matplotlib (3.4.3)
  • Installing pandas (1.3.3)
  • Installing scipy (1.6.1)
  • Installing threadpoolctl (3.0.0)
  • Installing scikit-learn (1.0)
  • Installing seaborn (0.11.2)
(base) thomas@lip-osx-005002 reproducible-modelling-dev % ls
   poetry.lock      results          setup.py         tests
   pyproject.toml   scripts          src
(base) thomas@lip-osx-005002 reproducible-modelling-dev % nano poetry.lock
```

Code portably

— so that your code can run on other machines

< poetry.lock >

Provides all details about installed packages

```
  GNU nano 2.0.6                          File: poetry.lock

name = "kiwisolver"
version = "1.3.2"
description = "A fast implementation of the Cassowary constraint solver"
category = "main"
optional = false
python-versions = ">=3.7"

[[package]]
name = "matplotlib"
version = "3.4.3"
description = "Python plotting package"
category = "main"
optional = false
python-versions = ">=3.7"

[package.dependencies]
cycler = ">=0.10"
kiwisolver = ">=1.0.1"
numpy = ">=1.16"
pillow = ">=6.2.0"
pyparsing = ">=2.2.1"
python-dateutil = ">=2.7"

[[package]]
name = "numpy"
version = "1.21.1"
description = "NumPy is the fundamental package for array computing with Python."
category = "main"
optional = false
python-versions = ">=3.7"
```

Code portably

— so that your code can run on other machines

poetry.lock

```
  GNU nano 2.0.6                          File: poetry.lock

    {file = "kiwisolver-1.3.2-cp39-cp39-macosx_10_9_x86_64.whl", hash = "sha256:80efd202108c3a4150e042b269f7c786434$
    {file = "kiwisolver-1.3.2-cp39-cp39-macosx_11_0_arm64.whl", hash = "sha256:f8eb7b6716f5b50e9c06207a14172cf2de20$
    {file = "kiwisolver-1.3.2-cp39-cp39-manylinux_2_12_i686.manylinux2010_i686.whl", hash = "sha256:f441422bb313ab2$
    {file = "kiwisolver-1.3.2-cp39-cp39-manylinux_2_12_x86_64.manylinux2010_x86_64.whl", hash = "sha256:30fa008c172$
    {file = "kiwisolver-1.3.2-cp39-cp39-manylinux_2_17_aarch64.manylinux2014_aarch64.whl", hash = "sha256:2f8f6c8f4$
    {file = "kiwisolver-1.3.2-cp39-cp39-manylinux_2_17_ppc64le.manylinux2014_ppc64le.whl", hash = "sha256:ba677bcaf$
    {file = "kiwisolver-1.3.2-cp39-cp39-manylinux_2_17_s390x.manylinux2014_s390x.whl", hash = "sha256:7843b1624d6cc$
    {file = "kiwisolver-1.3.2-cp39-cp39-win32.whl", hash = "sha256:e6f5eb2f53fac7d408a45fbcdeda7224b1cfff64919d0f95$
    {file = "kiwisolver-1.3.2-cp39-cp39-win_amd64.whl", hash = "sha256:eedd3b59190885d1ebdf6c5e0ca56828beb1949b4dfe$
    {file = "kiwisolver-1.3.2-pp37-pypy37_pp73-macosx_10_9_x86_64.whl", hash = "sha256:dedc71c8eb9c5096037766390172$
    {file = "kiwisolver-1.3.2-pp37-pypy37_pp73-manylinux_2_12_i686.manylinux2010_i686.whl", hash = "sha256:bf7eb45d$
    {file = "kiwisolver-1.3.2-pp37-pypy37_pp73-manylinux_2_12_x86_64.manylinux2010_x86_64.whl", hash = "sha256:2b65$
    {file = "kiwisolver-1.3.2-pp37-pypy37_pp73-manylinux_2_17_aarch64.manylinux2014_aarch64.whl", hash = "sha256:25$
    {file = "kiwisolver-1.3.2-pp37-pypy37_pp73-win_amd64.whl", hash = "sha256:bcadb05c3d4794eb9eee1dddf1c24215c92fb$
    {file = "kiwisolver-1.3.2.tar.gz", hash = "sha256:fc4453705b81d03568d5b808ad8f09c77c47534f6ac2e72e733f9ca4714aa$
]
matplotlib = [
    {file = "matplotlib-3.4.3-cp37-cp37m-macosx_10_9_x86_64.whl", hash = "sha256:5c988bb43414c7c2b0a31bd5187b4d27fd$
    {file = "matplotlib-3.4.3-cp37-cp37m-manylinux1_i686.whl", hash = "sha256:f1c5efc278d996af8a251b2ce0b07bbeccb82$
    {file = "matplotlib-3.4.3-cp37-cp37m-manylinux1_x86_64.whl", hash = "sha256:eeb1859efe7754b1460e1d4991bbd4a60a5$
    {file = "matplotlib-3.4.3-cp37-cp37m-manylinux2014_aarch64.whl", hash = "sha256:844a7b0233e4ff7fba57e90b8799eda$
    {file = "matplotlib-3.4.3-cp37-cp37m-win32.whl", hash = "sha256:85f0c9cf724715e75243a7b3087cf4a3de056b55e05d4d7$
    {file = "matplotlib-3.4.3-cp37-cp37m-win_amd64.whl", hash = "sha256:c70b6311dda3e27672f1bf48851a0de816d1ca6aaf3$
    {file = "matplotlib-3.4.3-cp38-cp38-macosx_10_9_x86_64.whl", hash = "sha256:b884715a59fec9ad3b6048ecf3860f3b2ce$
    {file = "matplotlib-3.4.3-cp38-cp38-manylinux1_i686.whl", hash = "sha256:a78a3b51f29448c7f4d4575e561f6b0dbb8d01$
    {file = "matplotlib-3.4.3-cp38-cp38-manylinux1_x86_64.whl", hash = "sha256:6a724e3a48a54b8b6e7c4ae38cd3d0708450$
    {file = "matplotlib-3.4.3-cp38-cp38-manylinux2014_aarch64.whl", hash = "sha256:48e1e0859b54d5f2e29bb78ca179fd59$
    {file = "matplotlib-3.4.3-cp38-cp38-win32.whl", hash = "sha256:01c9de93a2ca0d128c9064f23709362e7fefb34910c7c9e0$
    {file = "matplotlib-3.4.3-cp38-cp38-win_amd64.whl", hash = "sha256:ebfb01a65c3f5d53a8c2a8133fec2b5221281c053d94$
    {file = "matplotlib-3.4.3-cp39-cp39-macosx_10_9_x86_64.whl", hash = "sha256:b8b53f336a4688cfce615887505d7e41fd7$
    {file = "matplotlib-3.4.3-cp39-cp39-manylinux1_i686.whl", hash = "sha256:fcd6f1954943c0c192bfbebbac263f839d7055$
    {file = "matplotlib-3.4.3-cp39-cp39-manylinux1_x86_64.whl", hash = "sha256:6be8df61b1626e1a142c57e065405e869e94$
    {file = "matplotlib-3.4.3-cp39-cp39-manylinux2014_aarch64.whl", hash = "sha256:41b6e307458988891fcdea2d8ecf84a8$
```

As well as
specifics of install files

< reproducible-modelling >

- **data/**
- **results/**
- **scripts/**
- **src/**
- README.md
- pyproject.toml
- poetry.lock

Activating environment

Run new shell in virtual environment

Activating environment

Run scripts in virtual environment

```
% poetry run python scripts/evaluate_hyper_params_effect.py
```

Automate your analysis

..so that it can be run with a single command

Automate your analysis

— so that it can be run with a single command

< scripts/run_analysis.sh >

Run script 1

Run script 2

Run script 3

```
#!/usr/bin/env bash

CMD1="python load_data.py --data-path ../data/"
echo "Now running: $CMD1"
$CMD1
echo "..done."

CMD2="python evaluate_hyper_params_effect.py --data-path ../data/ --results-path ../results/"
echo "Now running: $CMD2"
$CMD2
echo "..done."

CMD3="python plot_hyper_params_effect.py --results-path ../results/"
echo "Now running: $CMD3"
$CMD3
echo "..done"
```

Automate your analysis

— so that it can be run with a single command

< scripts/run_analysis.sh >

```
(base) thomas@lip-osx-005002 reproducible-modelling-dev % cd scripts
(base) thomas@lip-osx-005002 scripts % poetry run bash run_analysis.sh
```

< reproducible-modelling >

- **data/**
- **results/**
- **scripts/**
  - load_data.py
  - evaluate_hyper_params_performance.py
  - plot_hyper_params_effect.py
  - **run_analysis.sh**

Automate your analysis

— so that it can be run with a single command

<Makefile>

Automate your analysis

— so that it can be run with a single command

<Makefile>

Target for running all analysis steps

```
10  # define a target for sequentially running each analysis step
11  # (can be called with: 'make all')
12  all : load evaluate plot
13
```

Automate your analysis

— so that it can be run with a single command

<Makefile>

Details of each analysis step

```
10  # define a target for sequentially running each analysis step
11  # (can be called with: 'make all')
12  all : load evaluate plot
13
14  # 1. load the handwritten digits dataset
15  # this analysis step can be called with 'make load'
16  load : data/hand_written_digits_data.npy
17  data/hand_written_digits_data.npy : scripts/load_data.py
18      mkdir -p data/;
19      python scripts/load_data.py --data-path 'data/'
20
```

Automate your analysis

— so that it can be run with a single command

<Makefile>

```
10  # define a target for sequentially running each analysis step
11  # (can be called with: 'make all')
12  all : load evaluate plot
13
14  # 1. load the handwritten digits dataset
15  # this analysis step can be called with 'make load'
16  load : data/hand_written_digits_data.npy
17  data/hand_written_digits_data.npy : scripts/load_data.py
18      mkdir -p data/;
19      python scripts/load_data.py --data-path 'data/'
20
```

Automate your analysis

— so that it can be run with a single command

<Makefile>

```
10  # define a target for sequentially running each analysis step
11  # (can be called with: 'make all')
12  all : load evaluate plot
13
14  # 1. load the handwritten digits dataset
15  # this analysis step can be called with 'make load'
16  load : data/hand_written_digits_data.npy
17  data/hand_written_digits_data.npy : scripts/load_data.py
18      mkdir -p data/;
19      python scripts/load_data.py --data-path 'data/'
20
```

Automate your analysis

— so that it can be run with a single command

<Makefile>

scripts/run_analysis.sh

```
10  # define a target for sequentially running each analysis step
11  # (can be called with: 'make all')
12  all : load evaluate plot
13
14  # 1. load the handwritten digits dataset
15  # this analysis step can be called with 'make load'
16  load : data/hand_written_digits_data.npy
17  data/hand_written_digits_data.npy : scripts/load_data.py
18      mkdir -p data/;
19      python scripts/load_data.py --data-path 'data/'
20
```

Automate your analysis

— so that it can be run with a single command

<Makefile>

```
# define a target for sequentially running each analysis step
# (can be called with: 'make all')
all : load evaluate plot

# 1. load the handwritten digits dataset
# this analysis step can be called with 'make load'
load : data/hand_written_digits_data.npy
data/hand_written_digits_data.npy : scripts/load_data.py
    mkdir -p data/;
    python scripts/load_data.py --data-path 'data/'

# 2. evaluate effect of hyper-parameter on predictive model performance
# this analysis step can be called with 'make evaluate'
evaluate : results/data/hyper_params_performance.csv
results/data/hyper_params_performance.csv : scripts/evaluate_hyper_params_effect.py
    mkdir -p results/;
    python scripts/evaluate_hyper_params_effect.py --n 100 --data-path 'data/' --results-path

# 3. plot effect of hyper-parameters on predictive model performance
# this analysis step can be called with 'make plot'
plot : results/figures/fig_hyper_params_effects.png
results/figures/fig_hyper_params_effects.png : scripts/plot_hyper_params_effect.py
    python scripts/plot_hyper_params_effect.py --results-path 'results/'
```

Automate your analysis

— so that it can be run with a single command

<Makefile>

```
(base) thomas@lip-osx-005002 reproducible-modelling-dev % poetry run make load
```

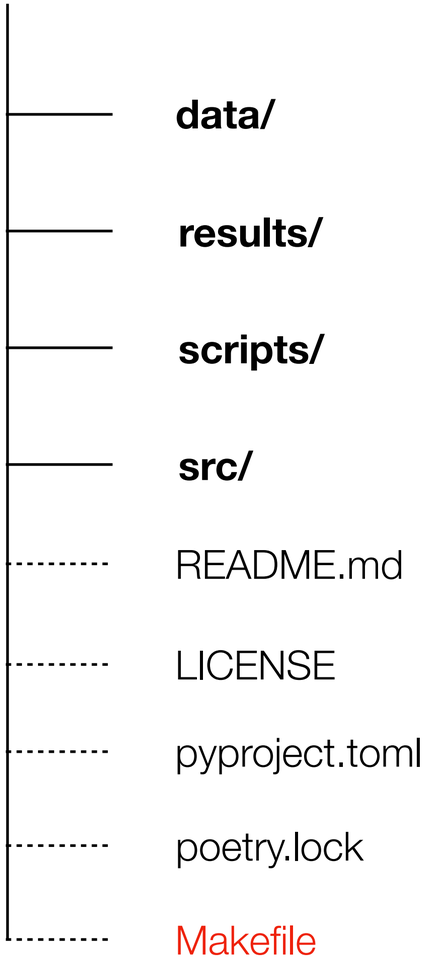
Automate your analysis

— so that it can be run with a single command

<Makefile>

```
(base) thomas@lip-osx-005002 reproducible-modelling-dev % poetry run make load
mkdir -p data/;
python scripts/load_data.py --data-path 'data/'
(base) thomas@lip-osx-005002 reproducible-modelling-dev %
```

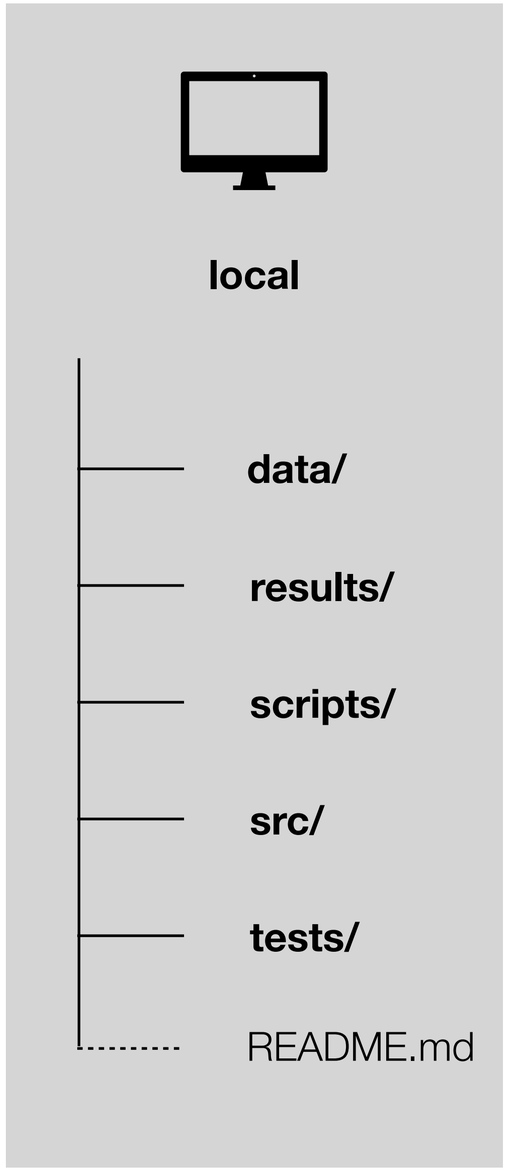
< reproducible-modelling >

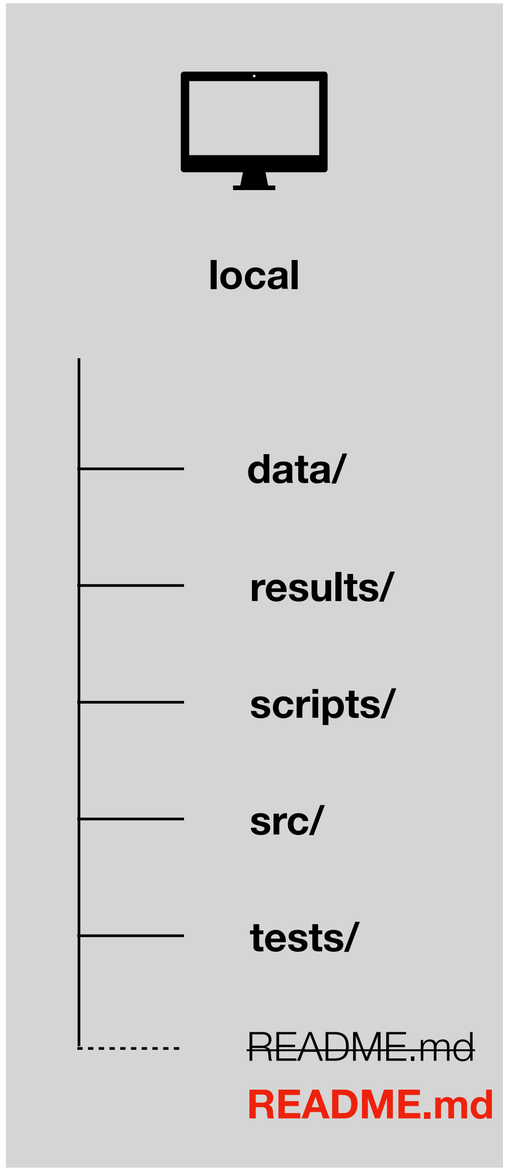


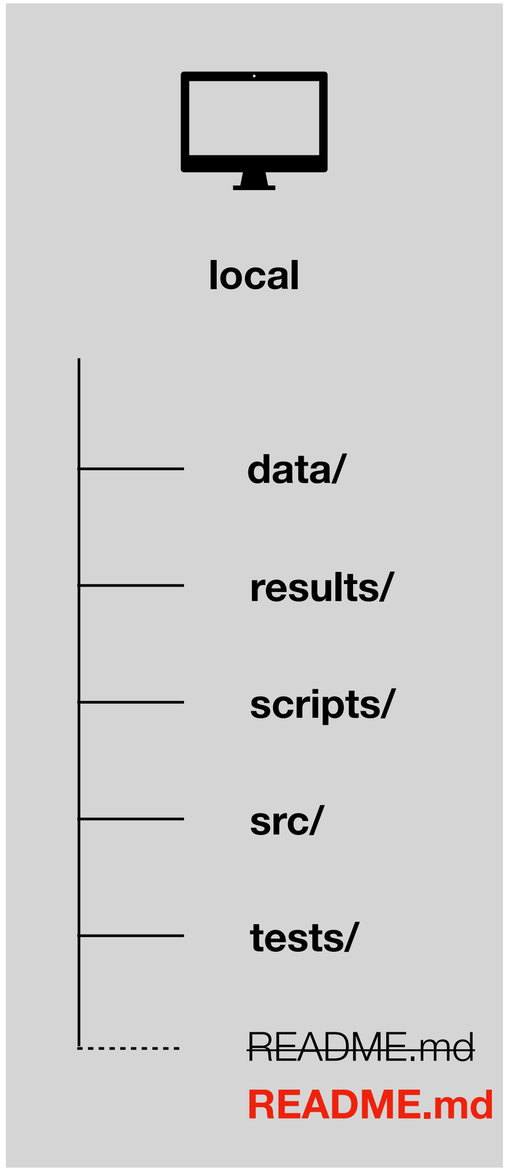
Track and share your code with Git and GitHub

A version control system

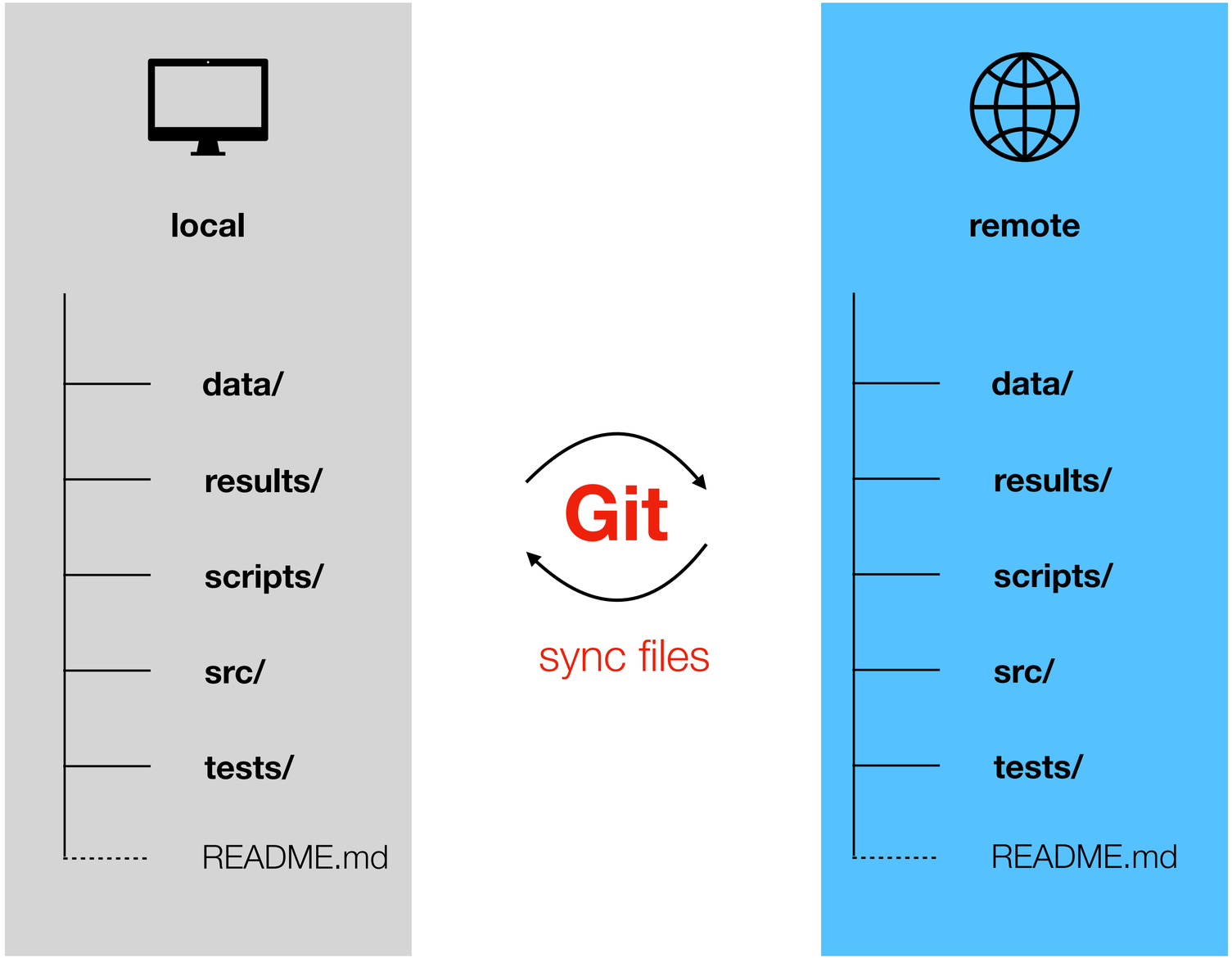
provider of internet hosting

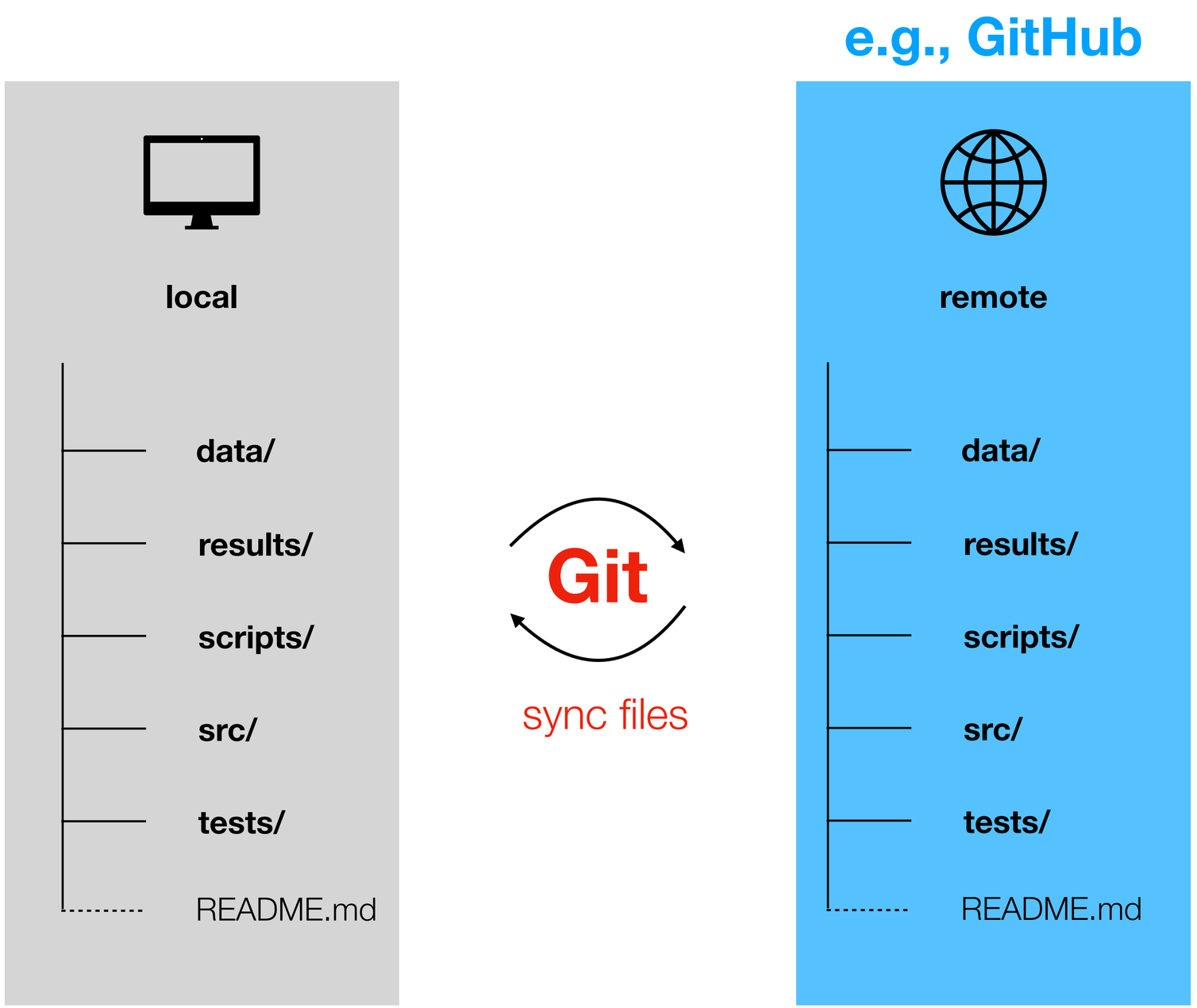

local
data/
results/
scripts/
src/
tests/
README.md

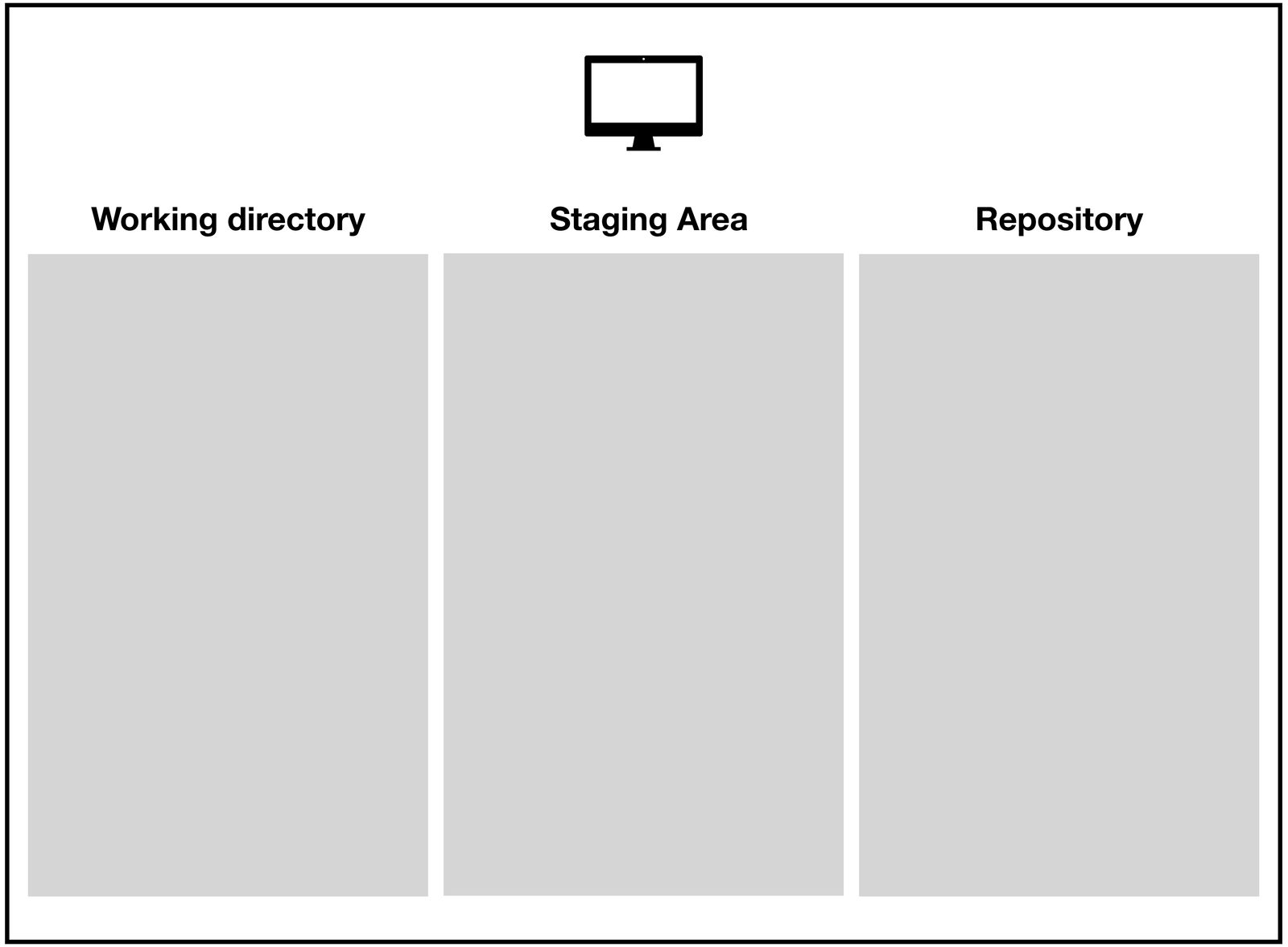

local
data/
results/
scripts/
src/
tests/
README.md
README.md

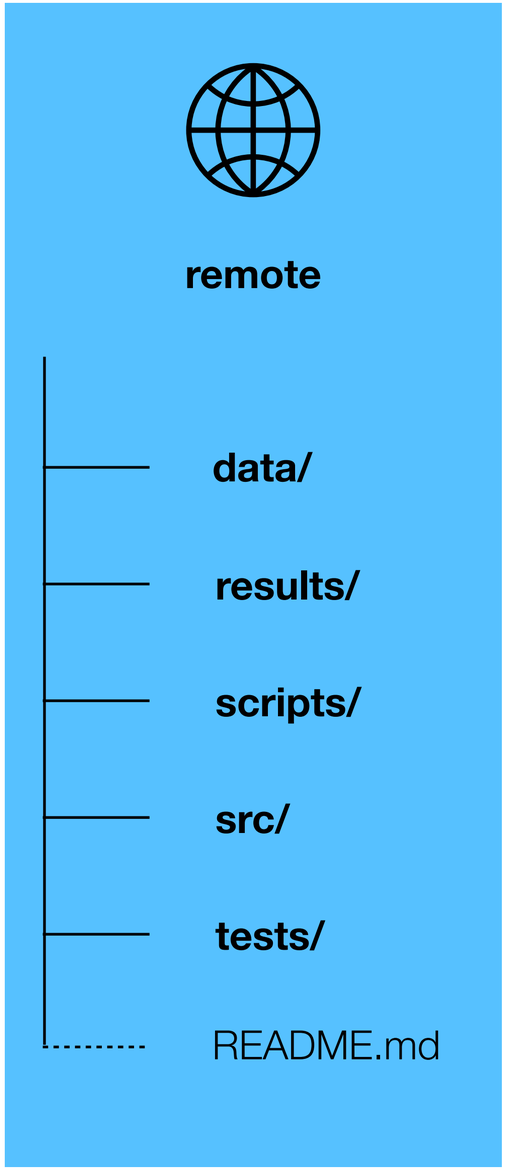


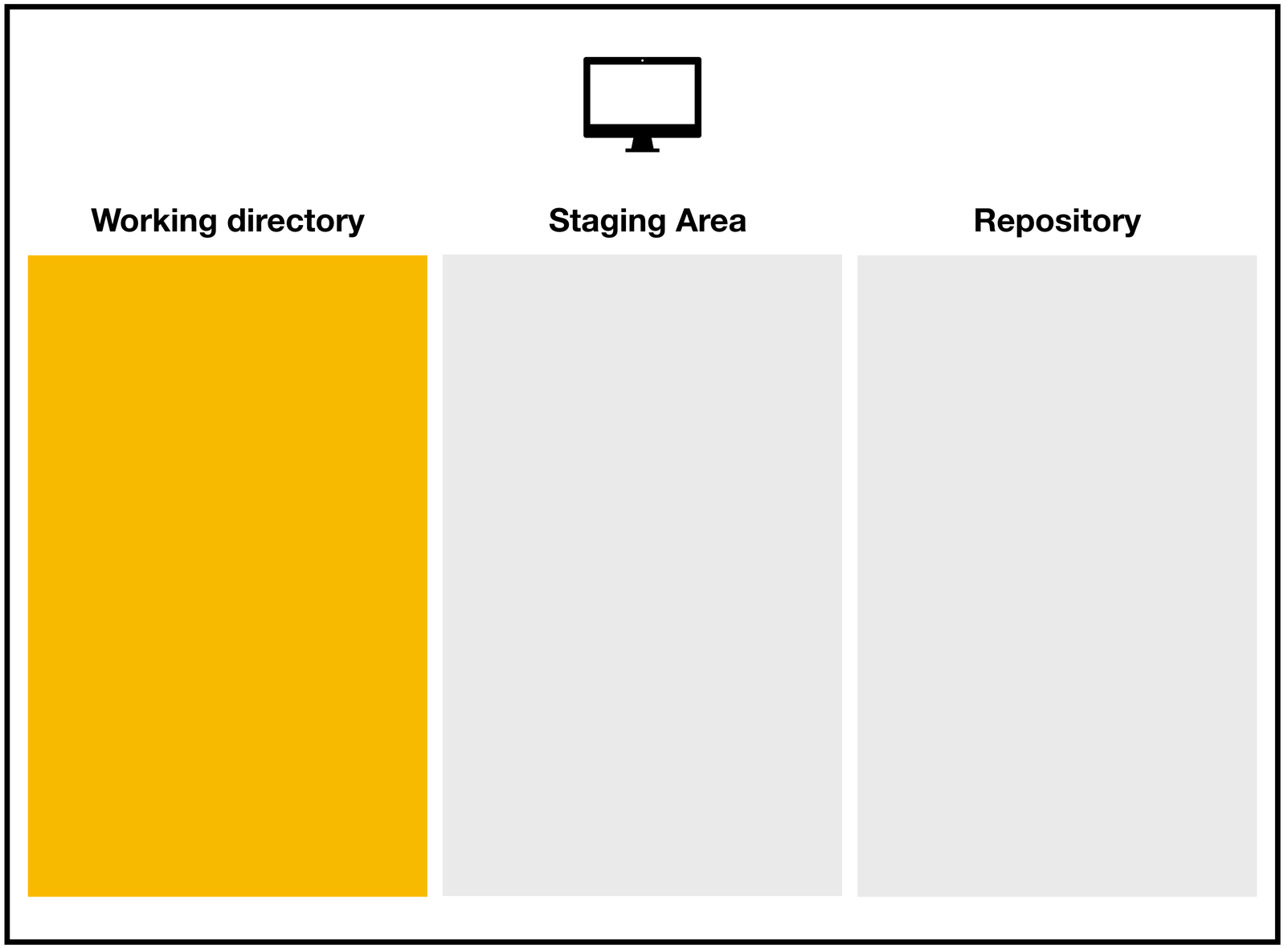
Git allows to
track changes

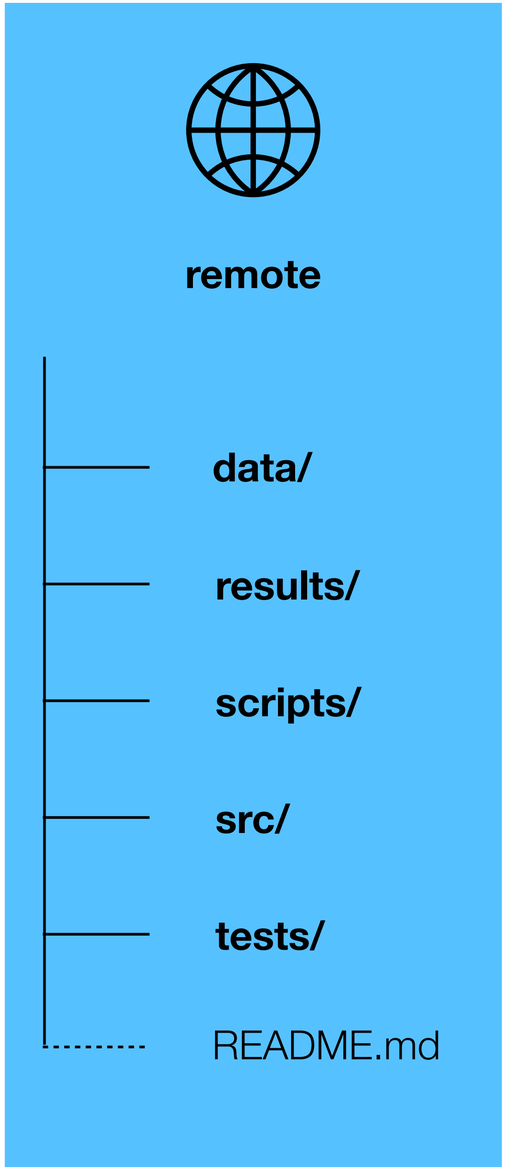

local
data/
results/
scripts/
src/
tests/
README.md
Git
sync files
remote
data/
results/
scripts/
src/
tests/
README.md

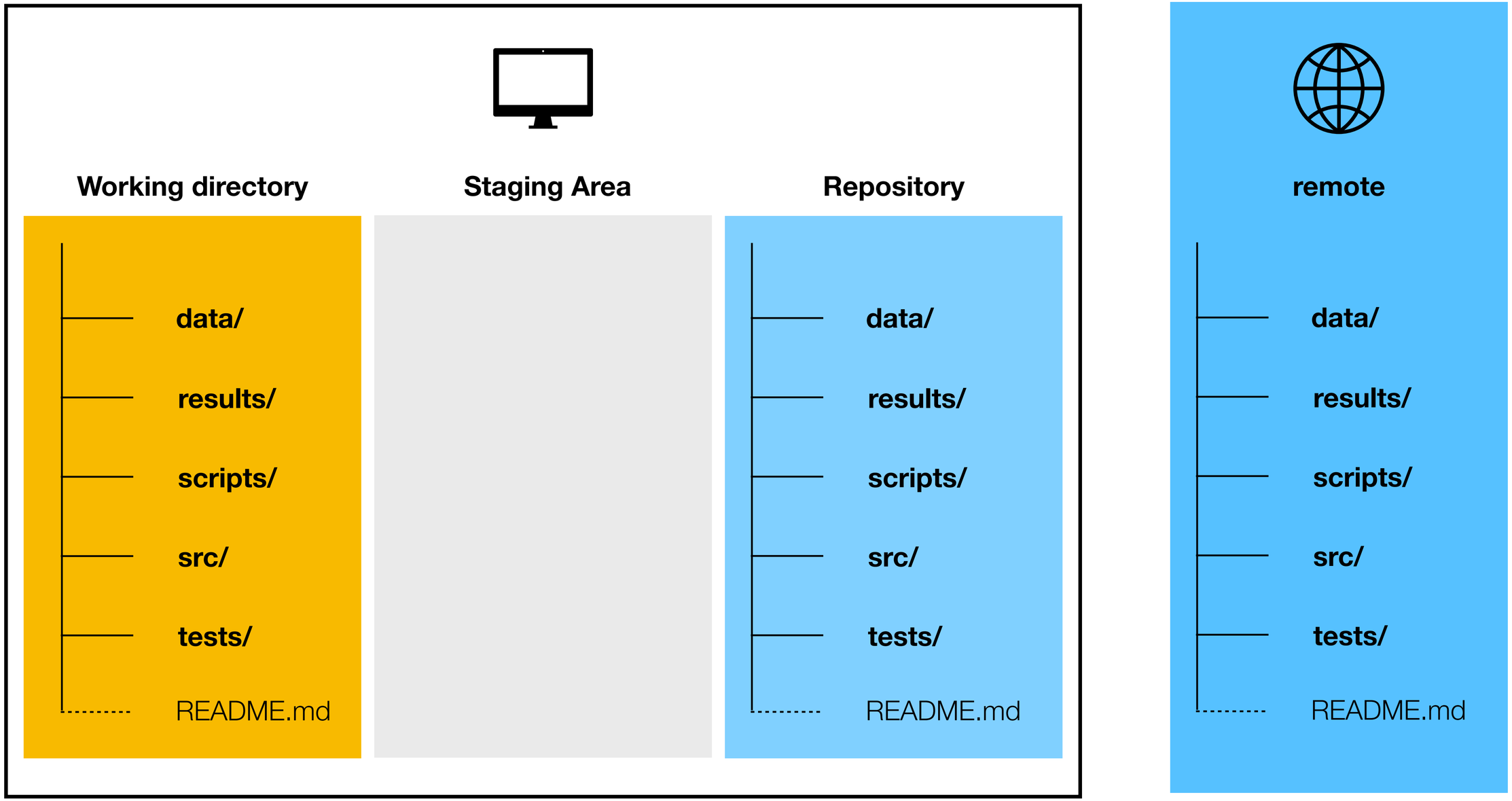

e.g., GitHub
local
data/
results/
scripts/
src/
tests/
README.md
Git
sync files
remote
data/
results/
scripts/
src/
tests/
README.md

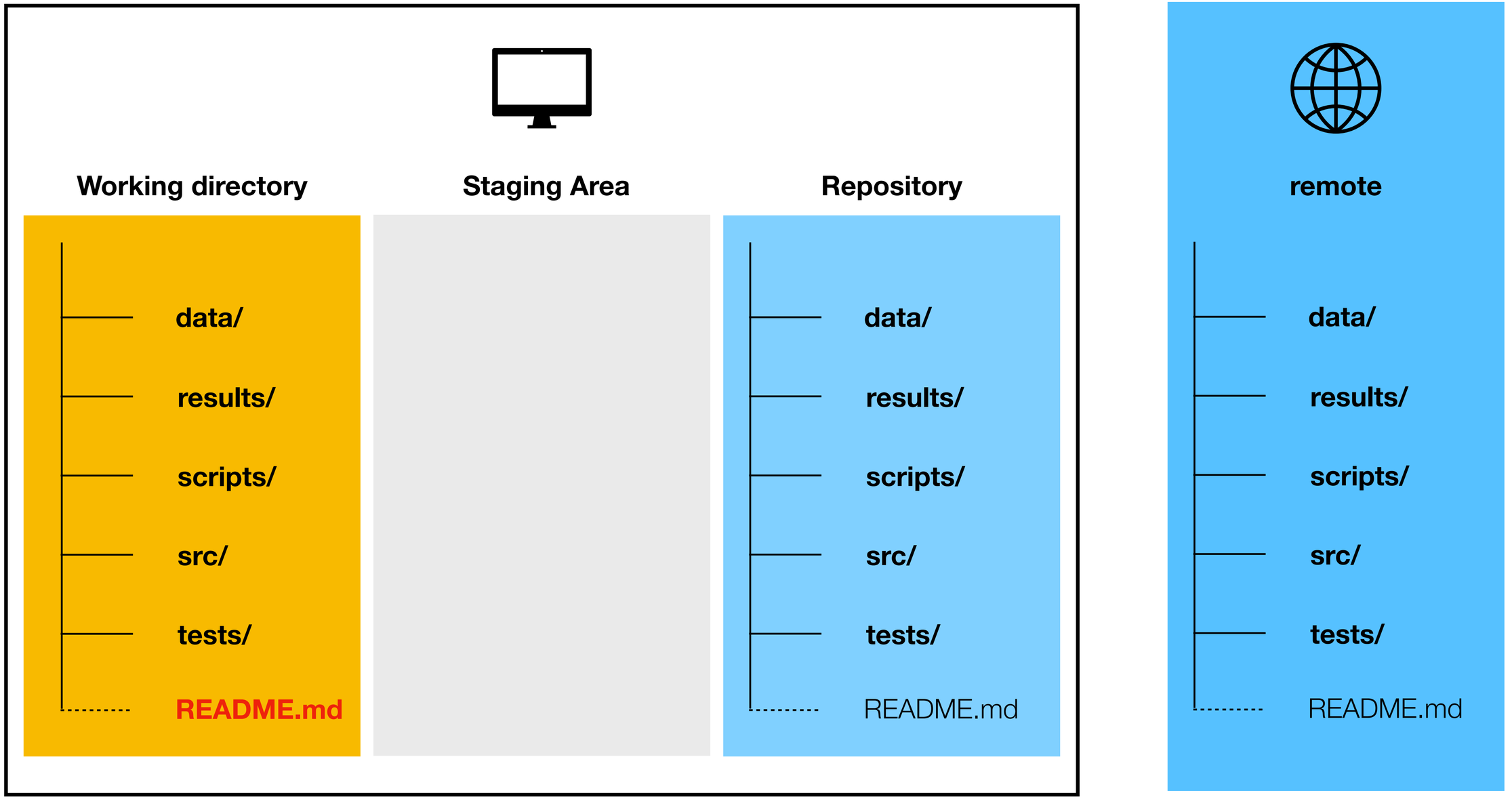

Working directory
Staging Area
Repository

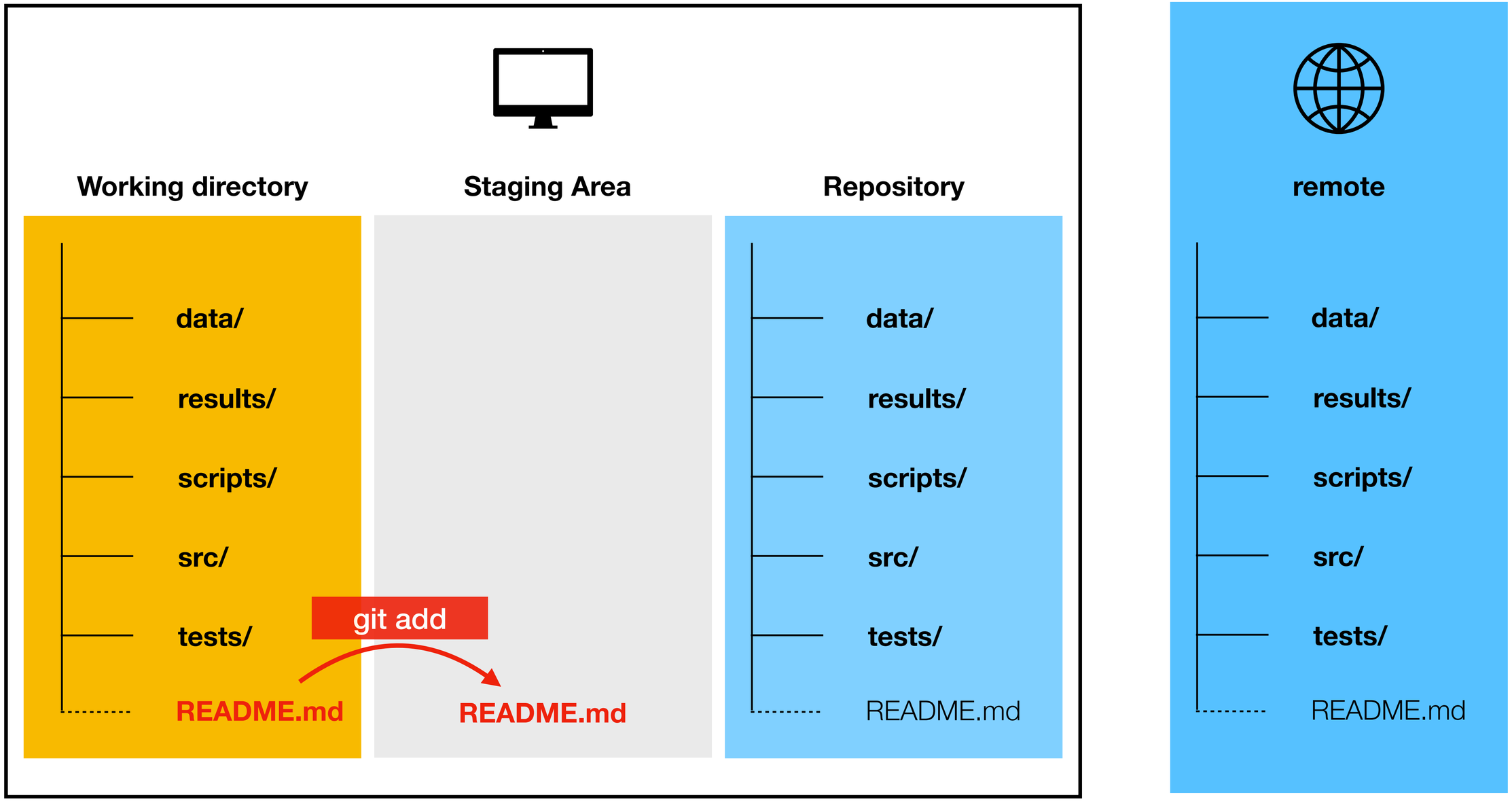

remote
data/
results/
scripts/
src/
tests/
README.md

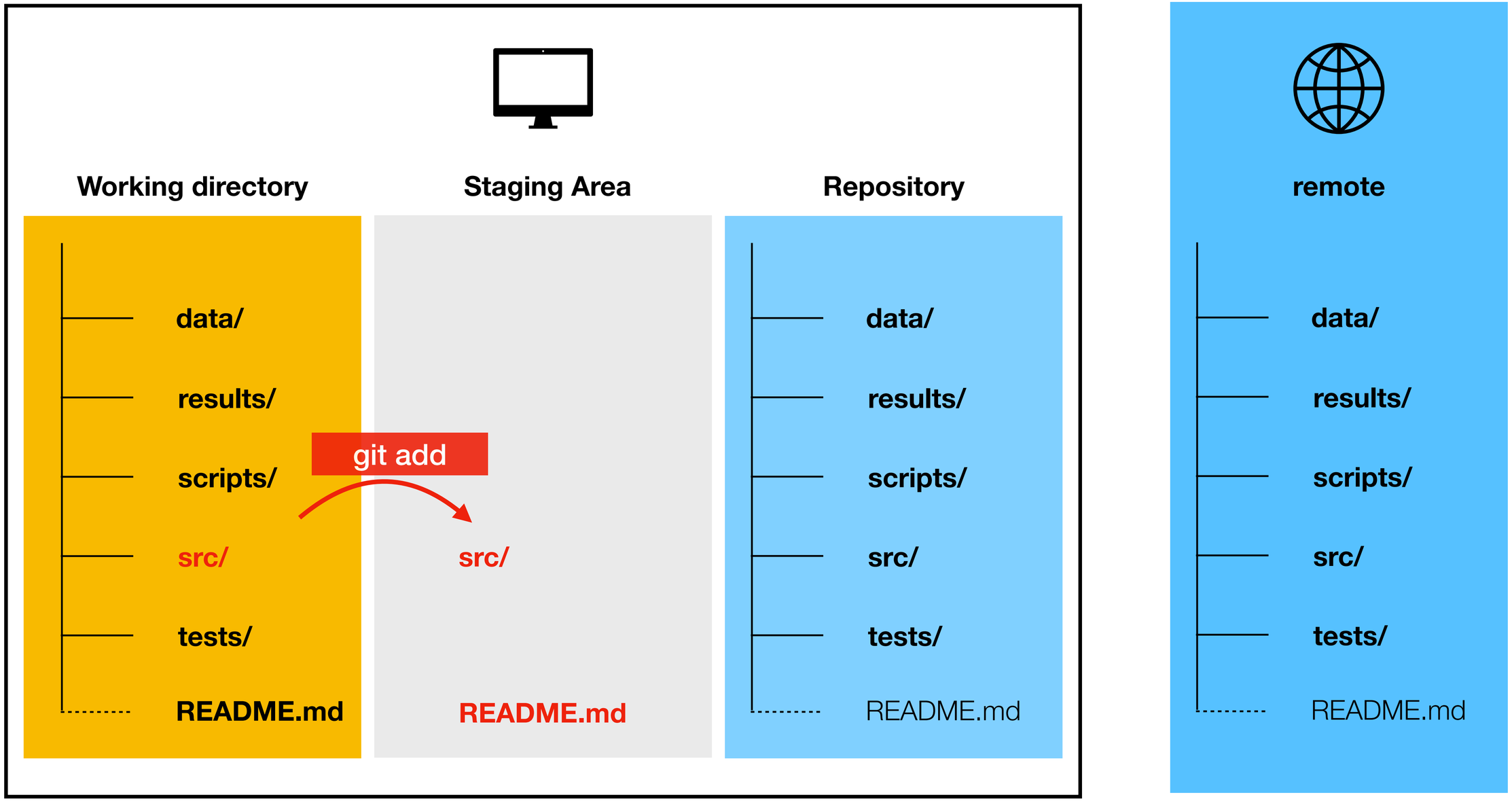

Working directory
Staging Area
Repository

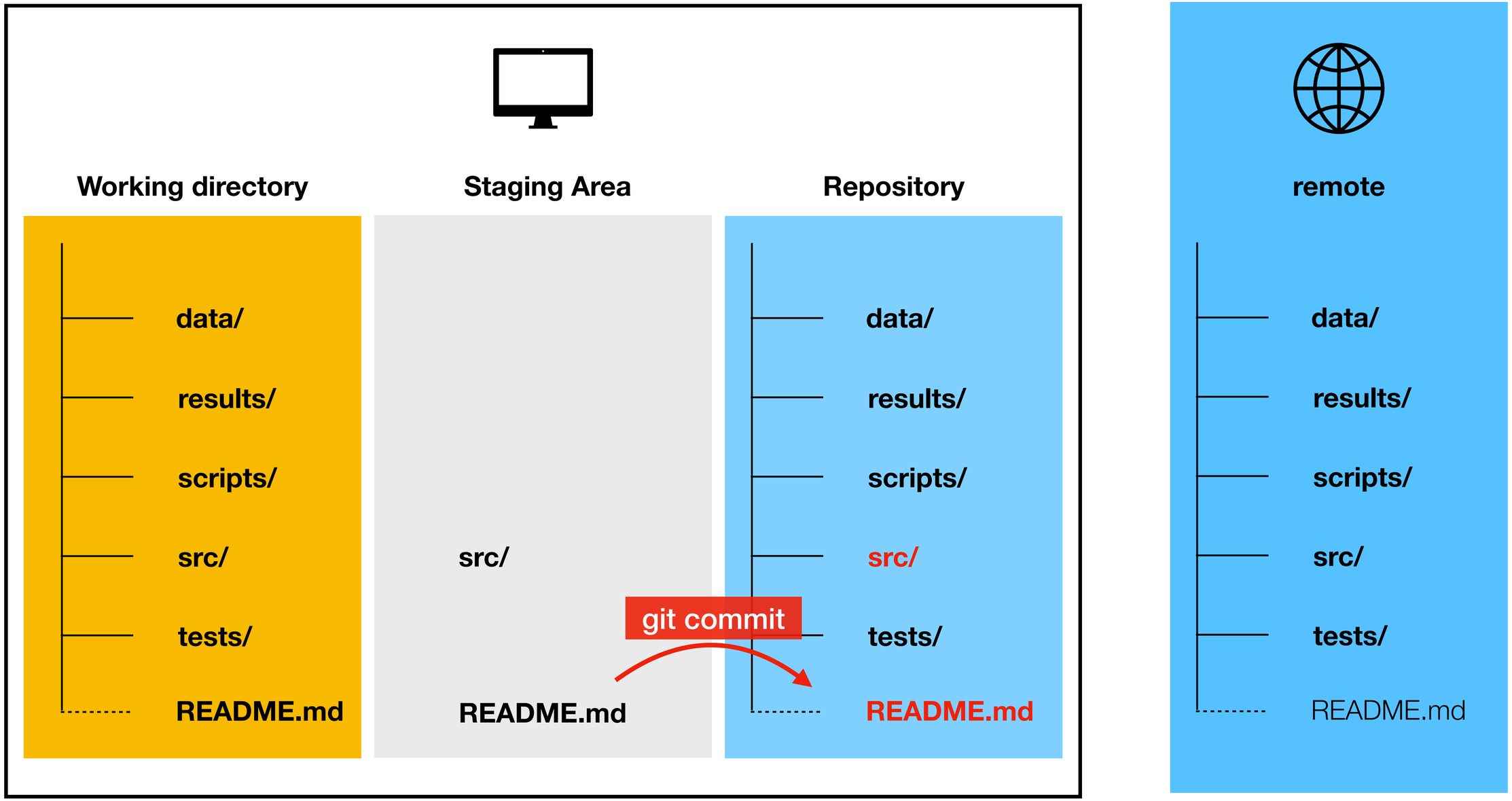

remote
data/
results/
scripts/
src/
tests/
README.md

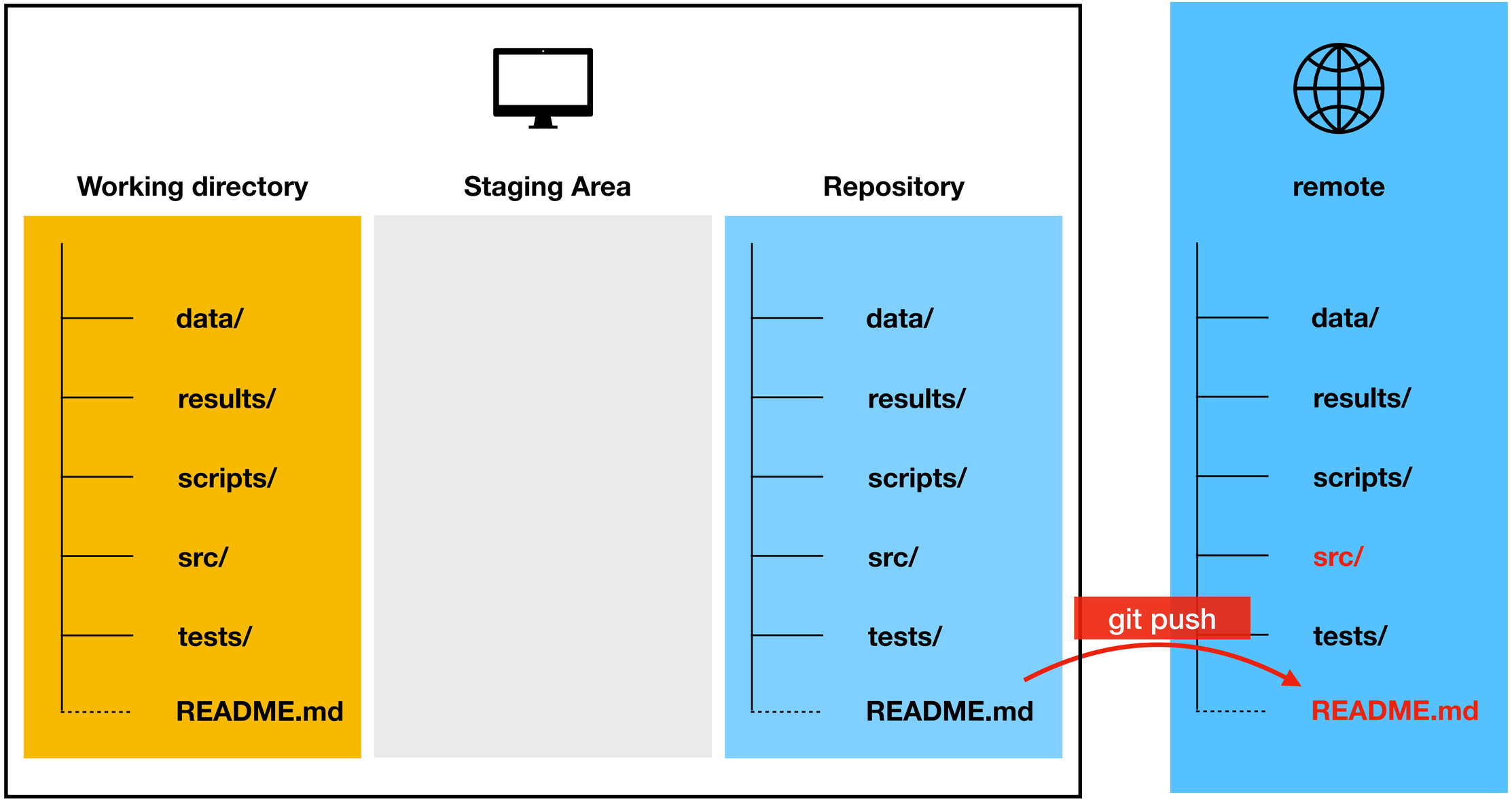

Working directory
Staging Area
Repository
remote
data/
results/
scripts/
src/
tests/
README.md
data/
results/
scripts/
src/
tests/
README.md
data/
results/
scripts/
src/
tests/
README.md

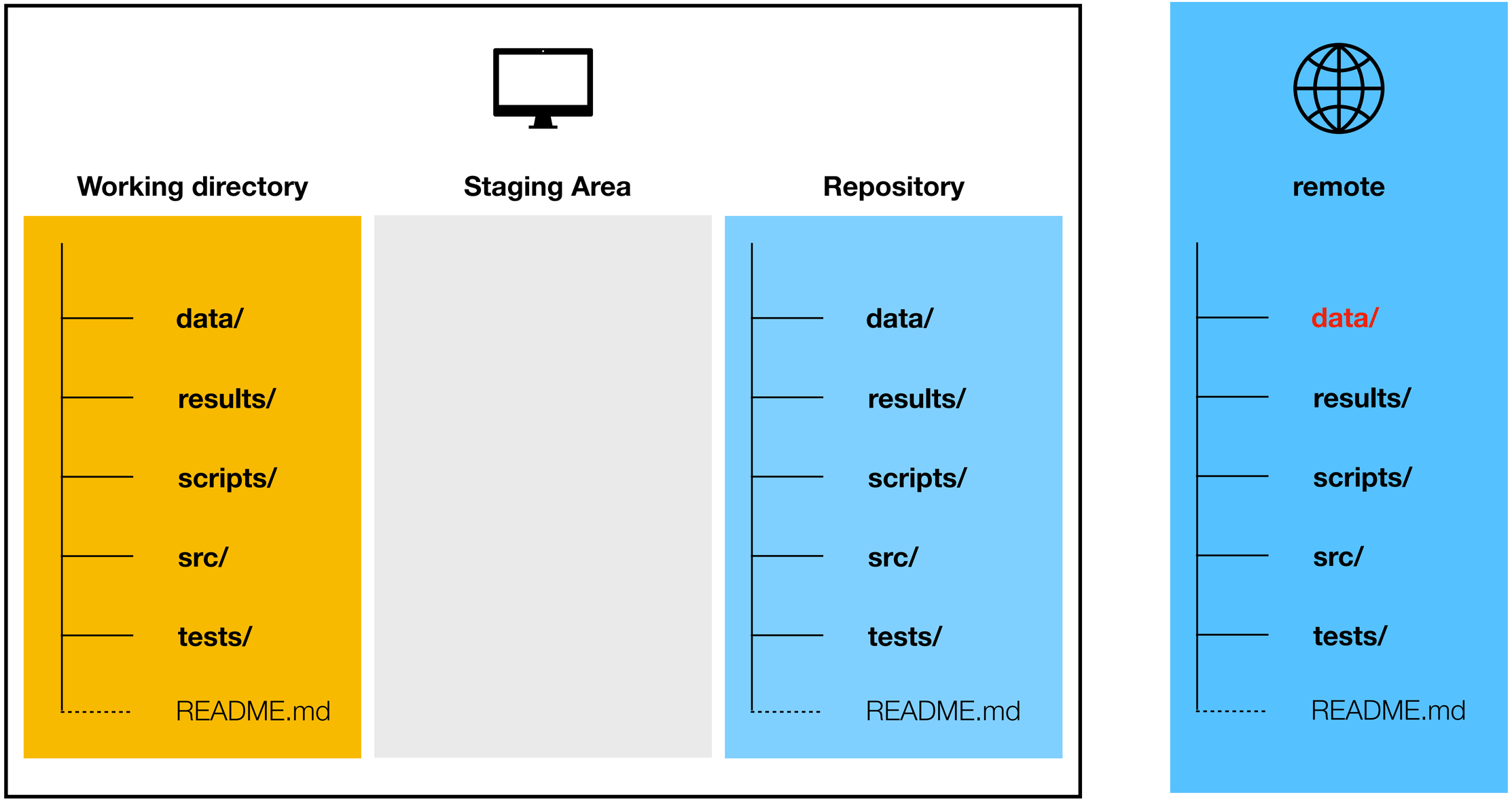

Working directory
Staging Area
Repository
remote
data/
results/
scripts/
src/
tests/
README.md
data/
results/
scripts/
src/
tests/
README.md
data/
results/
scripts/
src/
tests/
README.md

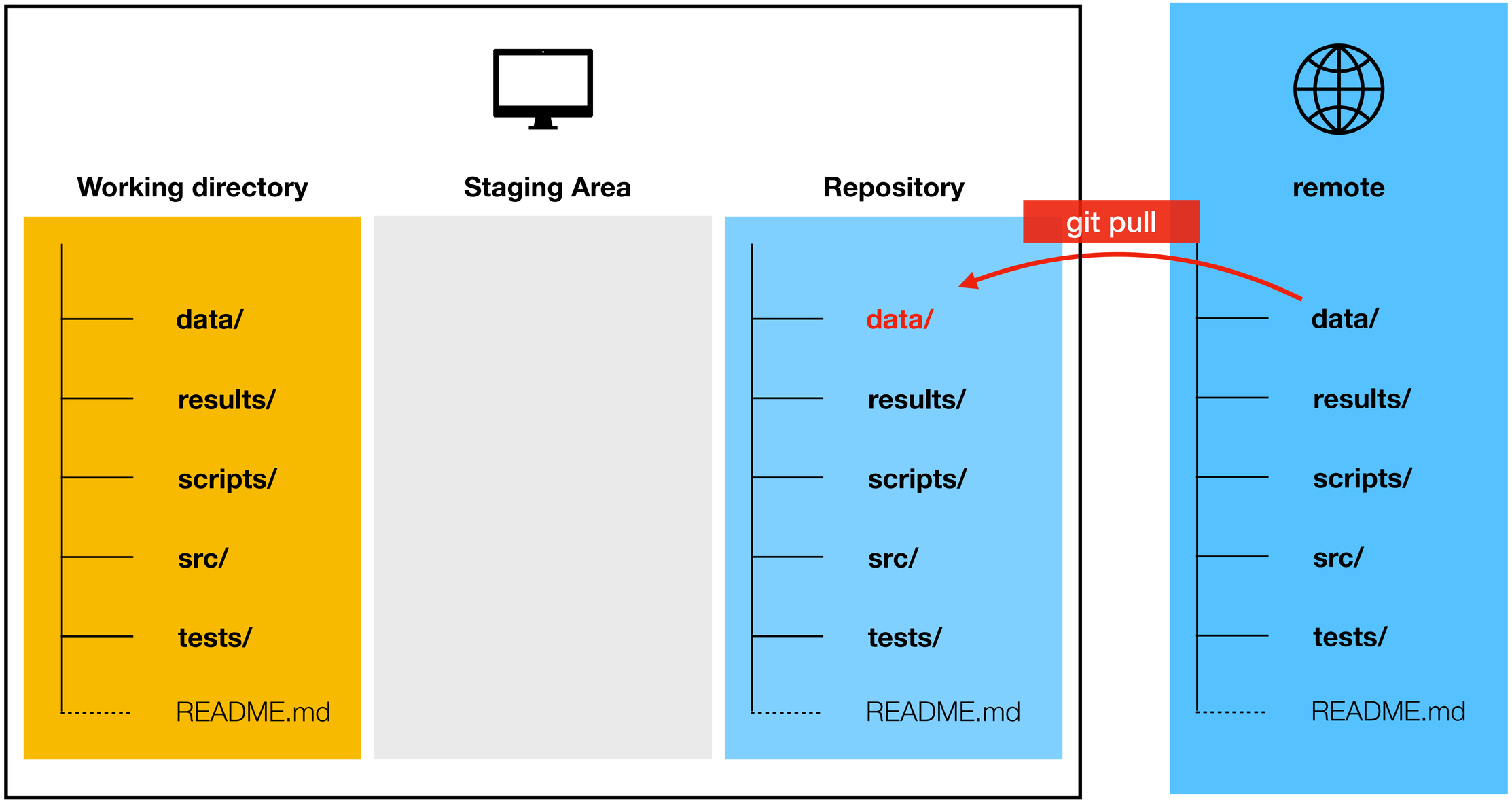

Working directory
Staging Area
Repository
remote
data/
results/
scripts/
src/
tests/
README.md
git add
README.md
data/
results/
scripts/
src/
tests/
README.md
data/
results/
scripts/
src/
tests/
README.md


Working directory
Staging Area
Repository
remote
data/
results/
scripts/
src/
tests/
README.md
git add
src/
README.md
data/
results/
scripts/
src/
tests/
README.md
data/
results/
scripts/
src/
tests/
README.md


Working directory
Staging Area
Repository
remote
data/
results/
scripts/
src/
tests/
README.md
src/
README.md
git commit
data/
results/
scripts/
src/
tests/
README.md
data/
results/
scripts/
src/
tests/
README.md

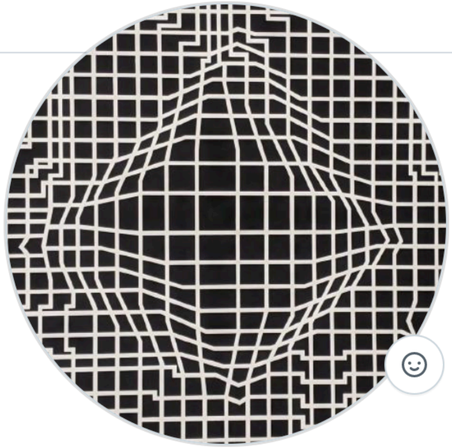

Working directory
Staging Area
Repository
remote
data/
results/
scripts/
src/
tests/
README.md
data/
results/
scripts/
src/
tests/
README.md
git push
data/
results/
scripts/
src/
tests/
README.md


Working directory
Staging Area
Repository
remote
data/
results/
scripts/
src/
tests/
README.md
data/
results/
scripts/
src/
tests/
README.md
data/
results/
scripts/
src/
tests/
README.md


Working directory
Staging Area
Repository
remote
git pull
data/
results/
scripts/
src/
tests/
README.md
data/
results/
scripts/
src/
tests/
README.md
data/
results/
scripts/
src/
tests/
README.md

Search or jump to... / Pull requests Issues Marketplace Explore

Overview Repositories 8 Projects Packages



Armin Thomas

athms

Ram and Vijay Shriram Data Science Fellow at Stanford Data Science

Edit profile

9 followers · 0 following · 2

Stanford University
athms.research@gmail.com
athms.me
@rmin_thomas

Achievements

Find a repository... Type Language Sort New

8 results for **public** repositories sorted by **stars** Clear filter

deep-learning-basics Public

A reproducible introductory course on the basics of deep learning.

deep-learning course tutorial jupyter-notebook binder

Jupyter Notebook 7 3 Updated 5 days ago

Star

many-item-choice Public

Jupyter Notebook 3 1 MIT License Updated on 12 Apr

Star

interprettensor Public

Forked from VigneshSrinivasan10/interprettensor

Python 35 Other Updated on 29 Mar 2018

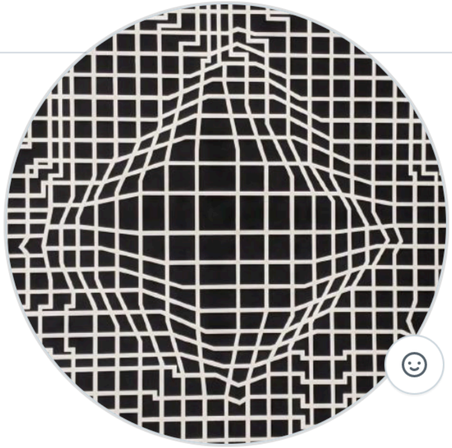
Star

evaluating-deeplight-transfer Public

Evaluating deep transfer learning for whole-brain cognitive decoding

Star

Search or jump to... / Pull requests Issues Marketplace Explore



Armin Thomas

athms

Ram and Vijay Shriram Data Science Fellow at Stanford Data Science

Edit profile

9 followers · 0 following · 2

Stanford University
athms.research@gmail.com
athms.me
@rmin_thomas

Achievements

Overview | Repositories 8 | Projects | Packages

Find a repository... | Type | Language | Sort | New

8 results for **public** repositories sorted by **stars** | Clear filter

deep-learning-basics Public

A reproducible introductory course on the basics of deep learning.

deep-learning course tutorial jupyter-notebook binder

Jupyter Notebook 7 3 Updated 5 days ago

Star

many-item-choice Public

Jupyter Notebook 3 1 MIT License Updated on 12 Apr

Star

interprettensor Public

Forked from VigneshSrinivasan10/interprettensor

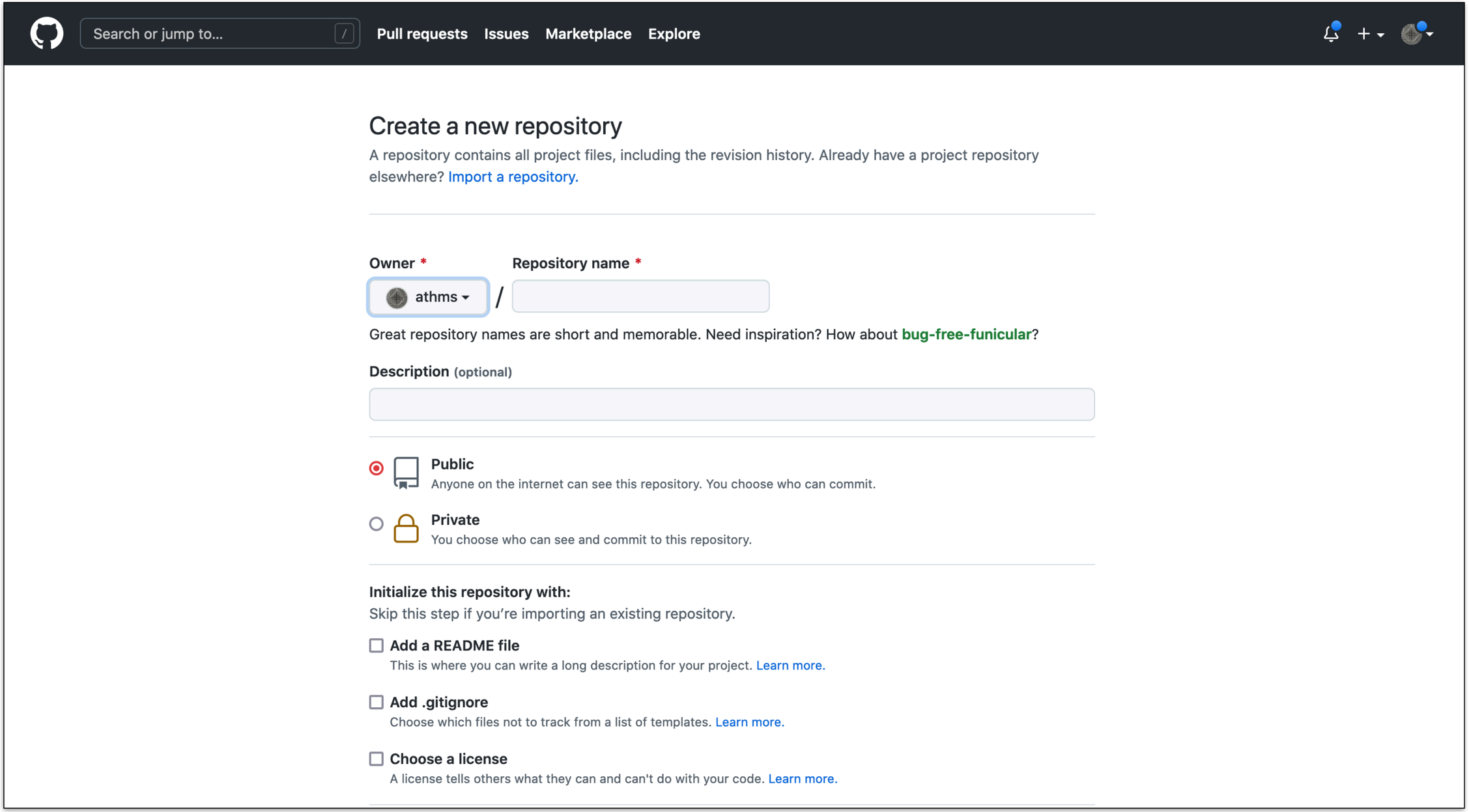
Python 35 Other Updated on 29 Mar 2018

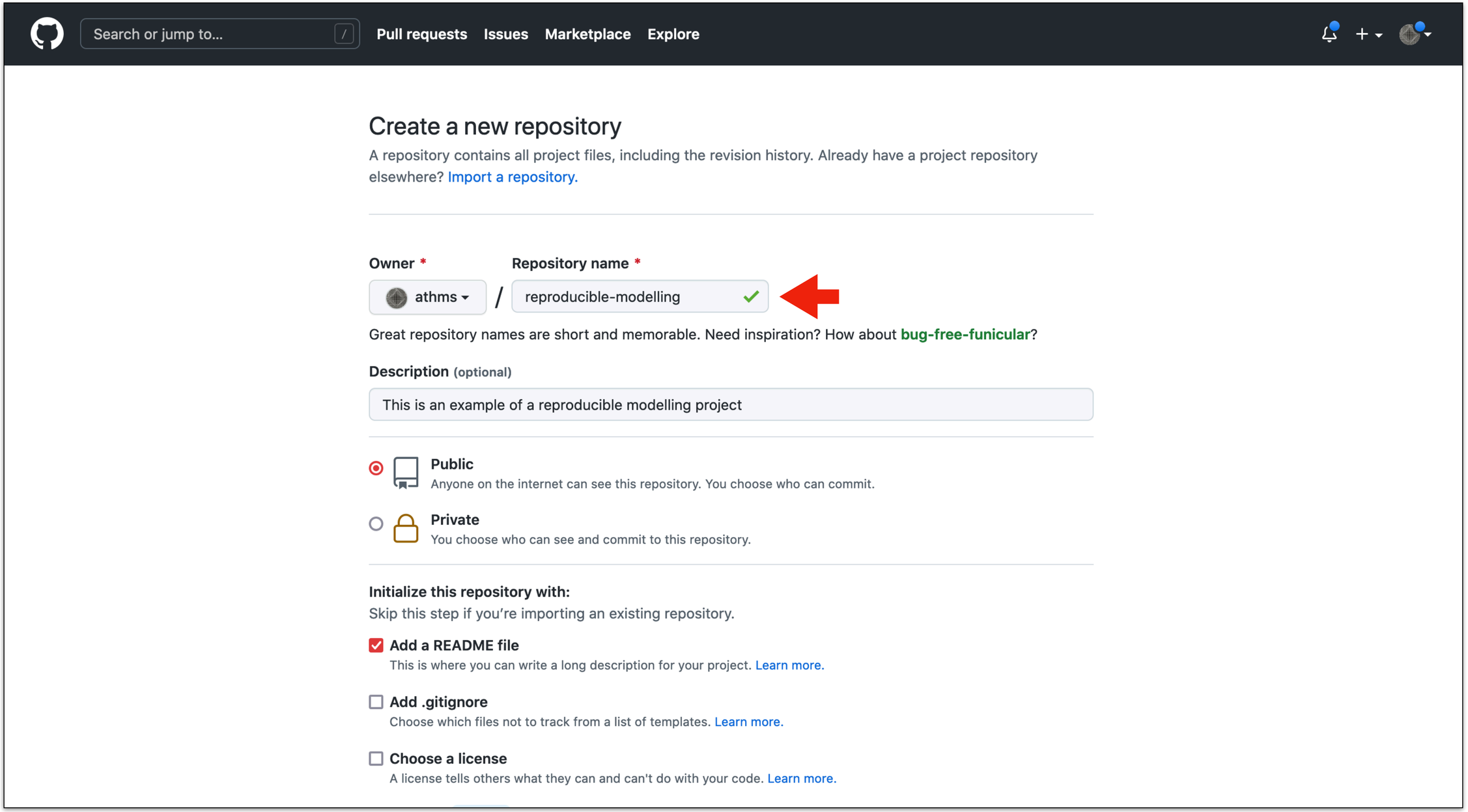
Star

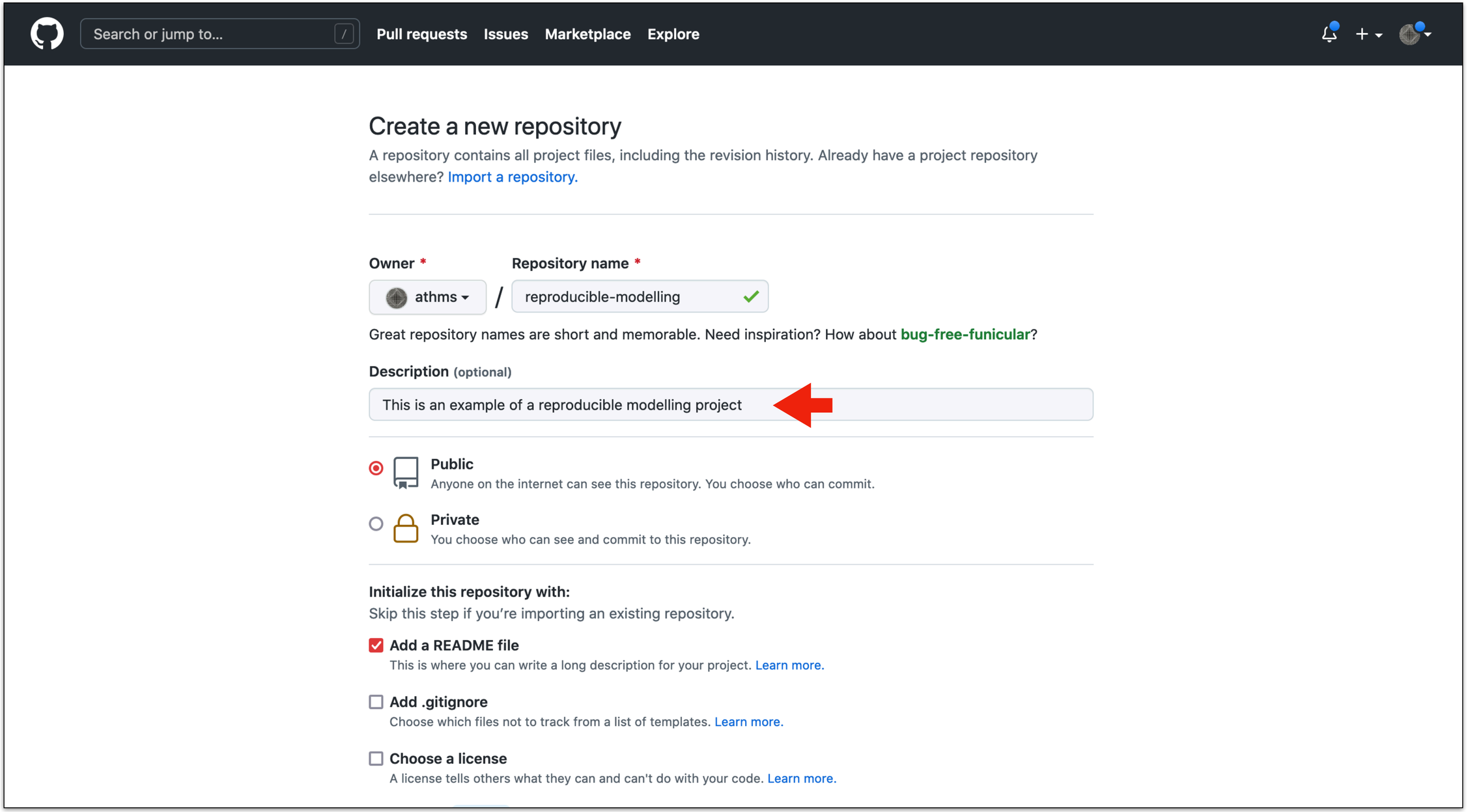
evaluating-deeplight-transfer Public

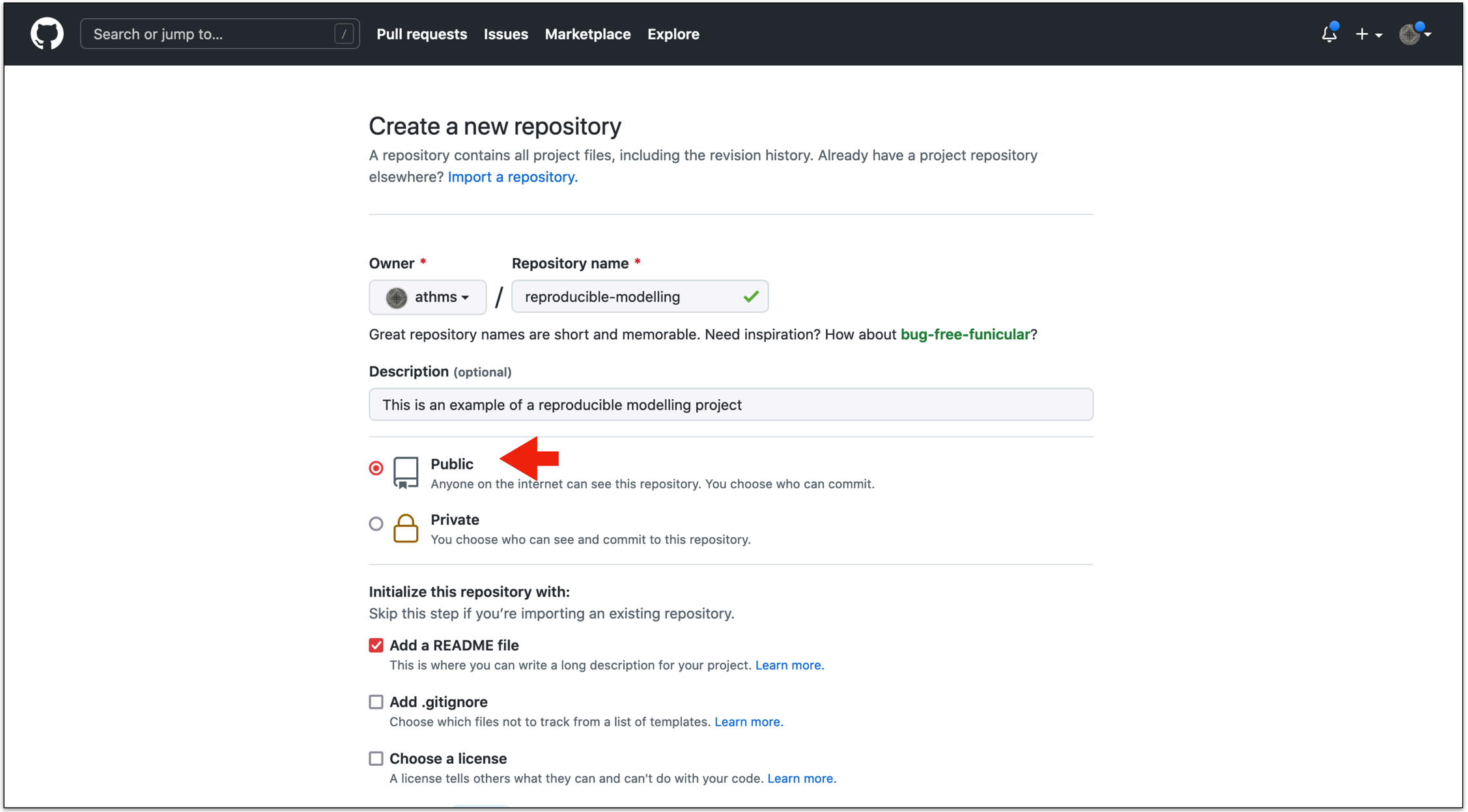
Evaluating deep transfer learning for whole-brain cognitive decoding

Star


Search or jump to...
/
Pull requests
Issues
Marketplace
Explore
Create a new repository
A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.
Owner *
Repository name *
athms
/
Great repository names are short and memorable. Need inspiration? How about bug-free-funicular?
Description (optional)
Public
Anyone on the internet can see this repository. You choose who can commit.
Private
You choose who can see and commit to this repository.
Initialize this repository with:
Skip this step if you're importing an existing repository.
Add a README file
This is where you can write a long description for your project. Learn more.
Add .gitignore
Choose which files not to track from a list of templates. Learn more.
Choose a license
A license tells others what they can and can't do with your code. Learn more.


Search or jump to...
Pull requests
Issues
Marketplace
Explore
Create a new repository
A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.
Owner *
Repository name *
athms
/
reproducible-modelling
Great repository names are short and memorable. Need inspiration? How about bug-free-funicular?
Description (optional)
This is an example of a reproducible modelling project
Public
Anyone on the internet can see this repository. You choose who can commit.
Private
You choose who can see and commit to this repository.
Initialize this repository with:
Skip this step if you're importing an existing repository.
Add a README file
This is where you can write a long description for your project. Learn more.
Add .gitignore
Choose which files not to track from a list of templates. Learn more.
Choose a license
A license tells others what they can and can't do with your code. Learn more.


Search or jump to...
Pull requests
Issues
Marketplace
Explore
Create a new repository
A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.
Owner *
Repository name *
athms
/
reproducible-modelling
Great repository names are short and memorable. Need inspiration? How about bug-free-funicular?
Description (optional)
This is an example of a reproducible modelling project
Public
Anyone on the internet can see this repository. You choose who can commit.
Private
You choose who can see and commit to this repository.
Initialize this repository with:
Skip this step if you're importing an existing repository.
Add a README file
This is where you can write a long description for your project. Learn more.
Add .gitignore
Choose which files not to track from a list of templates. Learn more.
Choose a license
A license tells others what they can and can't do with your code. Learn more.


Search or jump to...
Pull requests
Issues
Marketplace
Explore
Create a new repository
A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.
Owner *
Repository name *
athms
/
reproducible-modelling
Great repository names are short and memorable. Need inspiration? How about bug-free-funicular?
Description (optional)
This is an example of a reproducible modelling project
Public
Anyone on the internet can see this repository. You choose who can commit.
Private
You choose who can see and commit to this repository.
Initialize this repository with:
Skip this step if you're importing an existing repository.
Add a README file
This is where you can write a long description for your project. Learn more.
Add .gitignore
Choose which files not to track from a list of templates. Learn more.
Choose a license
A license tells others what they can and can't do with your code. Learn more.

athms / reproducible-modelling

Great repository names are short and memorable. Need inspiration? How about **bug-free-funicular**?

Description (optional)

This is an example of a reproducible modelling project

Public
Anyone on the internet can see this repository. You choose who can commit.

Private
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

Add a README file
This is where you can write a long description for your project. Learn more.

Add .gitignore
Choose which files not to track from a list of templates. Learn more.

Choose a license
A license tells others what they can and can't do with your code. Learn more.

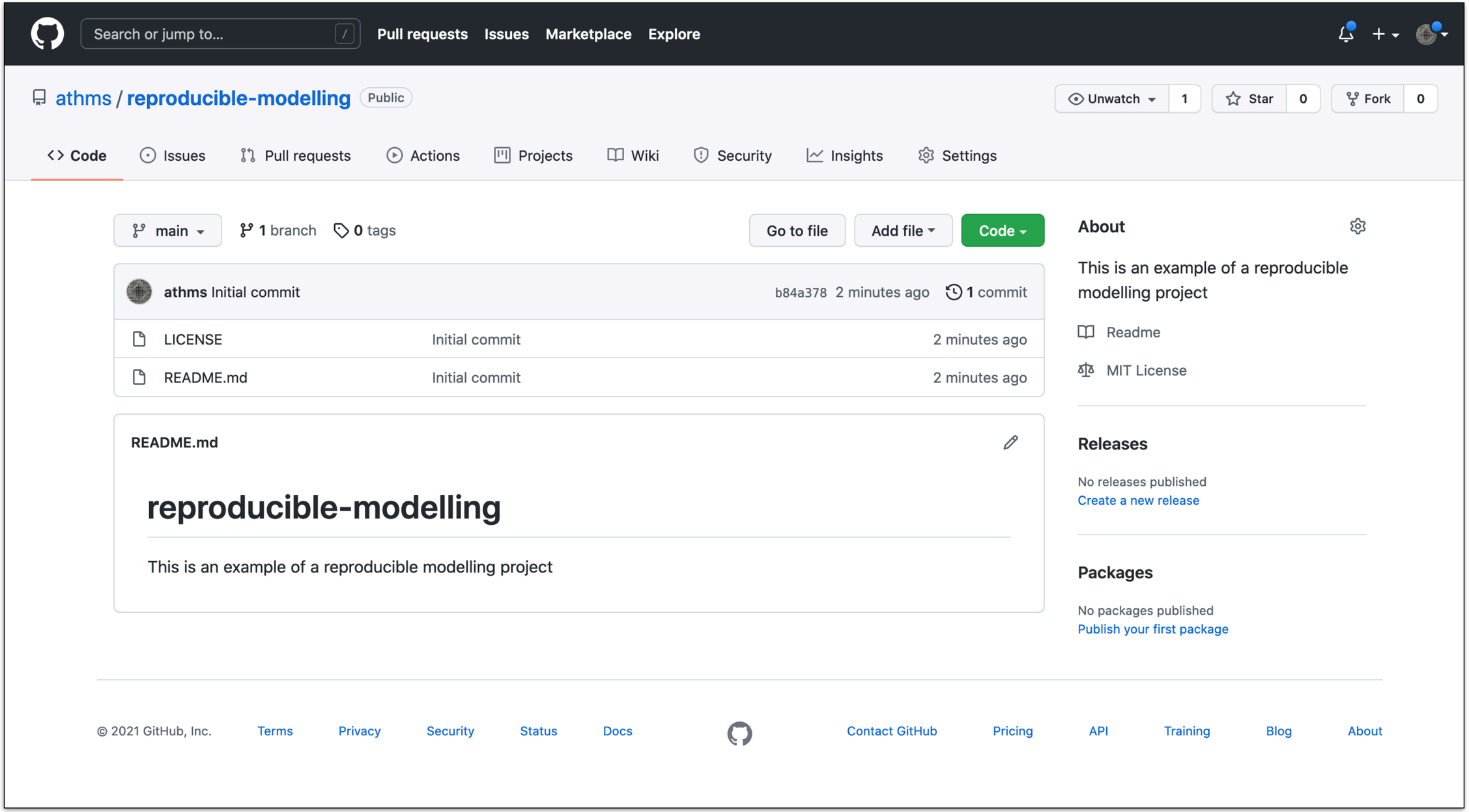
License: None

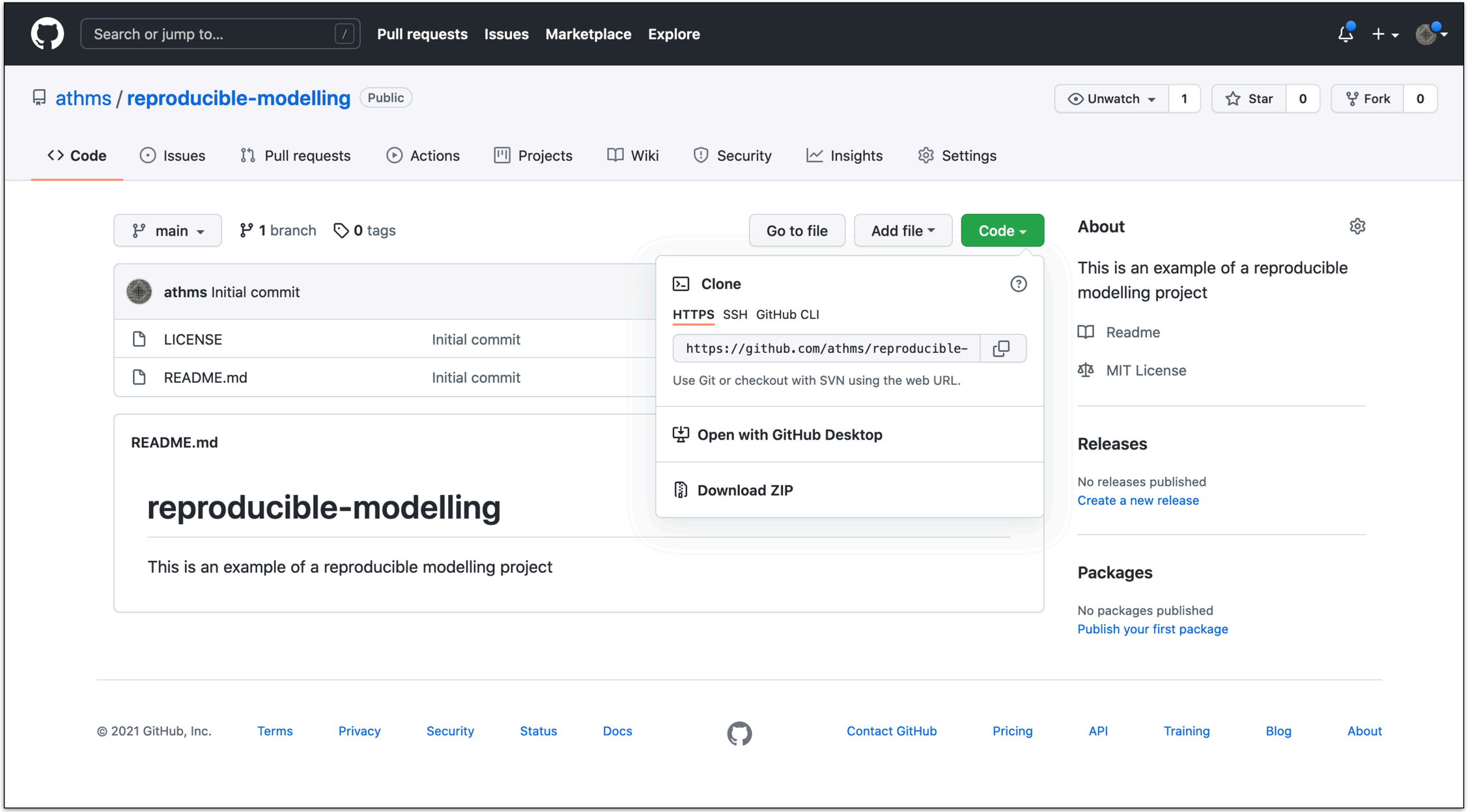
Thi ge the default name in your settings.

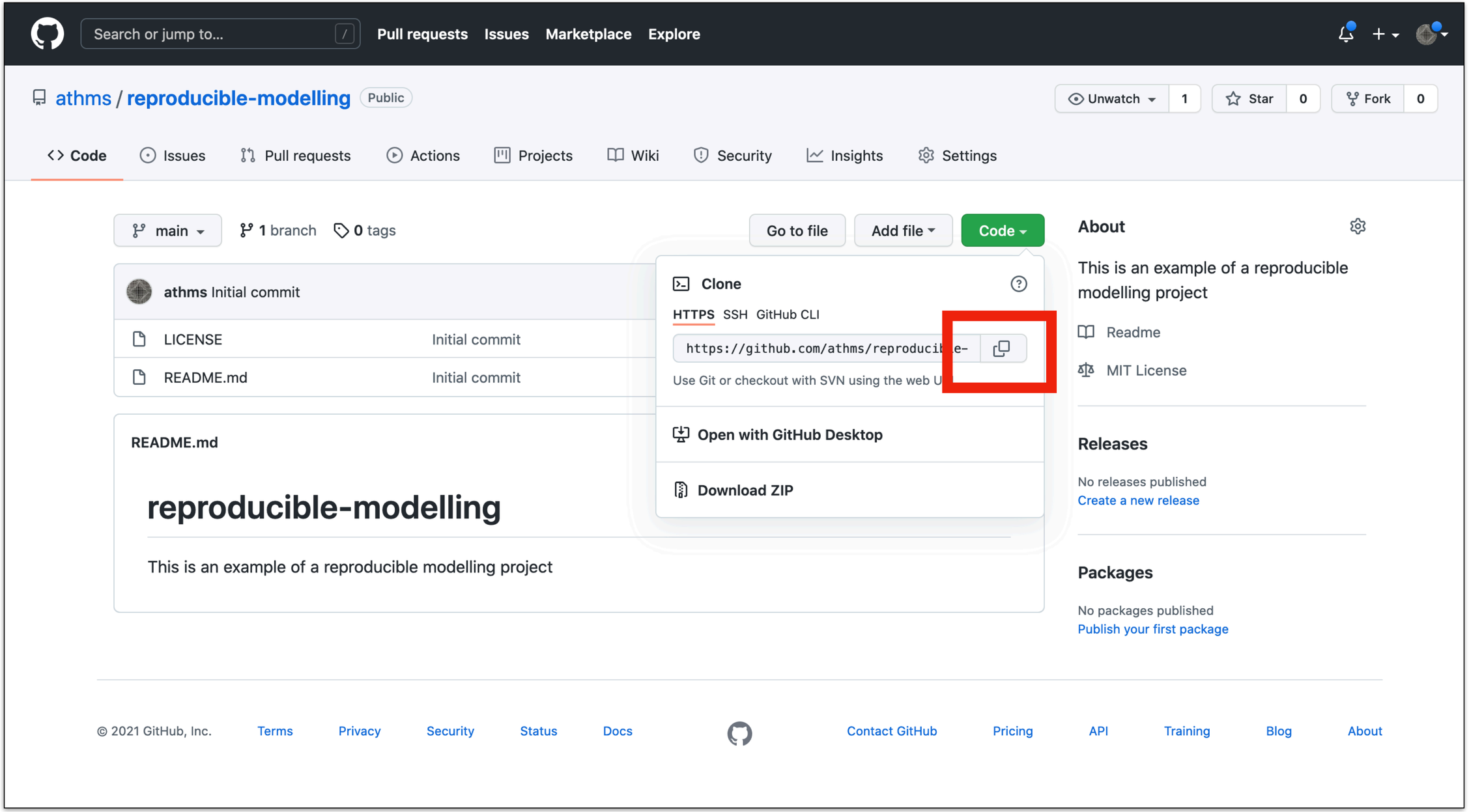
License

MIT

MIT License


Search or jump to...
Pull requests
Issues
Marketplace
Explore
athms / reproducible-modelling
Public
Unwatch 1
Star 0
Fork 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
main
1 branch
0 tags
Go to file
Add file
Code
athms Initial commit
b84a378 2 minutes ago
1 commit
LICENSE
Initial commit
2 minutes ago
README.md
Initial commit
2 minutes ago
README.md
reproducible-modelling
This is an example of a reproducible modelling project
About
This is an example of a reproducible modelling project
Readme
MIT License
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
© 2021 GitHub, Inc.
Terms
Privacy
Security
Status
Docs
Contact GitHub
Pricing
API
Training
Blog
About


Search or jump to...
Pull requests
Issues
Marketplace
Explore
athms / reproducible-modelling
Public
Unwatch 1
Star 0
Fork 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
main
1 branch
0 tags
Go to file
Add file
Code
athms Initial commit
LICENSE
Initial commit
README.md
Initial commit
Clone
HTTPS SSH GitHub CLI
https://github.com/athms/reproducible-
Use Git or checkout with SVN using the web URL.
Open with GitHub Desktop
Download ZIP
README.md
reproducible-modelling
This is an example of a reproducible modelling project
About
This is an example of a reproducible modelling project
Readme
MIT License
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
© 2021 GitHub, Inc.
Terms
Privacy
Security
Status
Docs
Contact GitHub
Pricing
API
Training
Blog
About


Search or jump to...
Pull requests
Issues
Marketplace
Explore
athms / reproducible-modelling
Public
Unwatch 1
Star 0
Fork 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
main
1 branch
0 tags
Go to file
Add file
Code
athms Initial commit
LICENSE
Initial commit
README.md
Initial commit
Clone
HTTPS SSH GitHub CLI
https://github.com/athms/reproducible-
Use Git or checkout with SVN using the web U
Open with GitHub Desktop
Download ZIP
README.md
reproducible-modelling
This is an example of a reproducible modelling project
About
This is an example of a reproducible modelling project
Readme
MIT License
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
© 2021 GitHub, Inc.
Terms
Privacy
Security
Status
Docs
Contact GitHub
Pricing
API
Training
Blog
About

Sharing your code

— so that others can see and use it

a. clone the repository to your local machine

```
(base) thomas@lip-osx-005002 projects % git clone https://github.com/athms/reproducible-modelling.git
```

Sharing your code

— so that others can see and use it

a. clone the repository to your local machine

Remote repository is downloaded

```
(base) thomas@lip-osx-005002 projects % git clone https://github.com/athms/reproducible-modelling.git
Cloning into 'reproducible-modelling'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (4/4), done.
```

Sharing your code

— so that others can see and use it

a. enter repository

```
(base) thomas@lip-osx-005002 projects % git clone https://github.com/athms/reproducible-modelling.git
Cloning into 'reproducible-modelling'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (4/4), done.
(base) thomas@lip-osx-005002 projects % cd reproducible-modelling/
```

Sharing your code

— so that others can see and use it

a. check status

‘git status’ displays state of working directory and staging area

```
(base) thomas@lip-osx-005002 projects % git clone https://github.com/athms/reproducible-modelling.git
Cloning into 'reproducible-modelling'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (4/4), done.
(base) thomas@lip-osx-005002 projects % cd reproducible-modelling/
(base) thomas@lip-osx-005002 reproducible-modelling % git status
```

Sharing your code

— so that others can see and use it

a. check status

```
(base) thomas@lip-osx-005002 projects % git clone https://github.com/athms/reproducible-modelling.git
Cloning into 'reproducible-modelling'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (4/4), done.
(base) thomas@lip-osx-005002 projects % cd reproducible-modelling/
(base) thomas@lip-osx-005002 reproducible-modelling % git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

We are in sync with remote

Sharing your code

— so that others can see and use it

a. copy your files to the repository

When adding files to the directory, these are at first ‘untracked’

```
(base) thomas@lip-osx-005002 reproducible-modelling % git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .DS_Store
        Makefile
        data/
        poetry.lock
        pyproject.toml
        results/
        scripts/
        setup.py
        src/

nothing added to commit but untracked files present (use "git add" to track)
```

Sharing your code

— so that others can see and use it

a. adding files

Add files to staging area

```
(base) thomas@lip-osx-005002 reproducible-modelling % git add -A
```

Sharing your code

— so that others can see and use it

a. committing files

Briefly describe your change

```
(base) thomas@lip-osx-005002 reproducible-modelling % git add -A
(base) thomas@lip-osx-005002 reproducible-modelling % git commit -m 'initial commit'
```

Sharing your code

— so that others can see and use it

a. committing files

```
(base) thomas@lip-osx-005002 reproducible-modelling % git add -A
(base) thomas@lip-osx-005002 reproducible-modelling % git commit -m 'initial commit'
[main fc0816b] initial commit
 17 files changed, 2259 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 Makefile
 create mode 100644 data/hand_written_digits_data.npy
 create mode 100644 poetry.lock
 create mode 100644 pyproject.toml
 create mode 100644 results/estimates/hyper_params_performance.csv
 create mode 100644 results/figures/fig_hyper_params_effects.png
 create mode 100644 scripts/evaluate_hyper_params_effect.py
 create mode 100644 scripts/load_data.py
 create mode 100644 scripts/plot_hyper_params_effect.py
 create mode 100644 scripts/run_analysis.sh
 create mode 100644 setup.py
 create mode 100644 src/__init__.py
 create mode 100644 src/hyper/__init__.py
 create mode 100644 src/hyper/evaluation.py
 create mode 100644 src/hyper/grid.py
 create mode 100644 src/hyper/plotting.py
```

Sharing your code

— so that others can see and use it

a. push

```
(base) thomas@lip-osx-005002 reproducible-modelling % git add -A
(base) thomas@lip-osx-005002 reproducible-modelling % git commit -m 'initial commit'
[main fc0816b] initial commit
 17 files changed, 2259 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 Makefile
 create mode 100644 data/hand_written_digits_data.npy
 create mode 100644 poetry.lock
 create mode 100644 pyproject.toml
 create mode 100644 results/estimates/hyper_params_performance.csv
 create mode 100644 results/figures/fig_hyper_params_effects.png
 create mode 100644 scripts/evaluate_hyper_params_effect.py
 create mode 100644 scripts/load_data.py
 create mode 100644 scripts/plot_hyper_params_effect.py
 create mode 100644 scripts/run_analysis.sh
 create mode 100644 setup.py
 create mode 100644 src/__init__.py
 create mode 100644 src/hyper/__init__.py
 create mode 100644 src/hyper/evaluation.py
 create mode 100644 src/hyper/grid.py
 create mode 100644 src/hyper/plotting.py
(base) thomas@lip-osx-005002 reproducible-modelling % git push
Enumerating objects: 27, done.
Counting objects: 100% (27/27), done.
Delta compression using up to 8 threads
Compressing objects: 100% (24/24), done.
Writing objects: 100% (26/26), 243.29 KiB | 3.74 MiB/s, done.
Total 26 (delta 0), reused 0 (delta 0)
To https://github.com/athms/reproducible-modelling.git
   b84a378..fc0816b  main -> main
```

Push your changes to remote

Sharing your code

— so that others can see and use it

a. you are in sync again :)

```
 create mode 100644 data/hand_written_digits_data.npy
 create mode 100644 poetry.lock
 create mode 100644 pyproject.toml
 create mode 100644 results/estimates/hyper_params_performance.csv
 create mode 100644 results/figures/fig_hyper_params_effects.png
 create mode 100644 scripts/evaluate_hyper_params_effect.py
 create mode 100644 scripts/load_data.py
 create mode 100644 scripts/plot_hyper_params_effect.py
 create mode 100644 scripts/run_analysis.sh
 create mode 100644 setup.py
 create mode 100644 src/__init__.py
 create mode 100644 src/hyper/__init__.py
 create mode 100644 src/hyper/evaluation.py
 create mode 100644 src/hyper/grid.py
 create mode 100644 src/hyper/plotting.py
(base) thomas@lip-osx-005002 reproducible-modelling % git push
Enumerating objects: 27, done.
Counting objects: 100% (27/27), done.
Delta compression using up to 8 threads
Compressing objects: 100% (24/24), done.
Writing objects: 100% (26/26), 243.29 KiB | 3.74 MiB/s, done.
Total 26 (delta 0), reused 0 (delta 0)
To https://github.com/athms/reproducible-modelling.git
   b84a378..fc0816b  main -> main
(base) thomas@lip-osx-005002 reproducible-modelling % git status
```

Sharing your code

— so that others can see and use it

a. you are in sync again :)

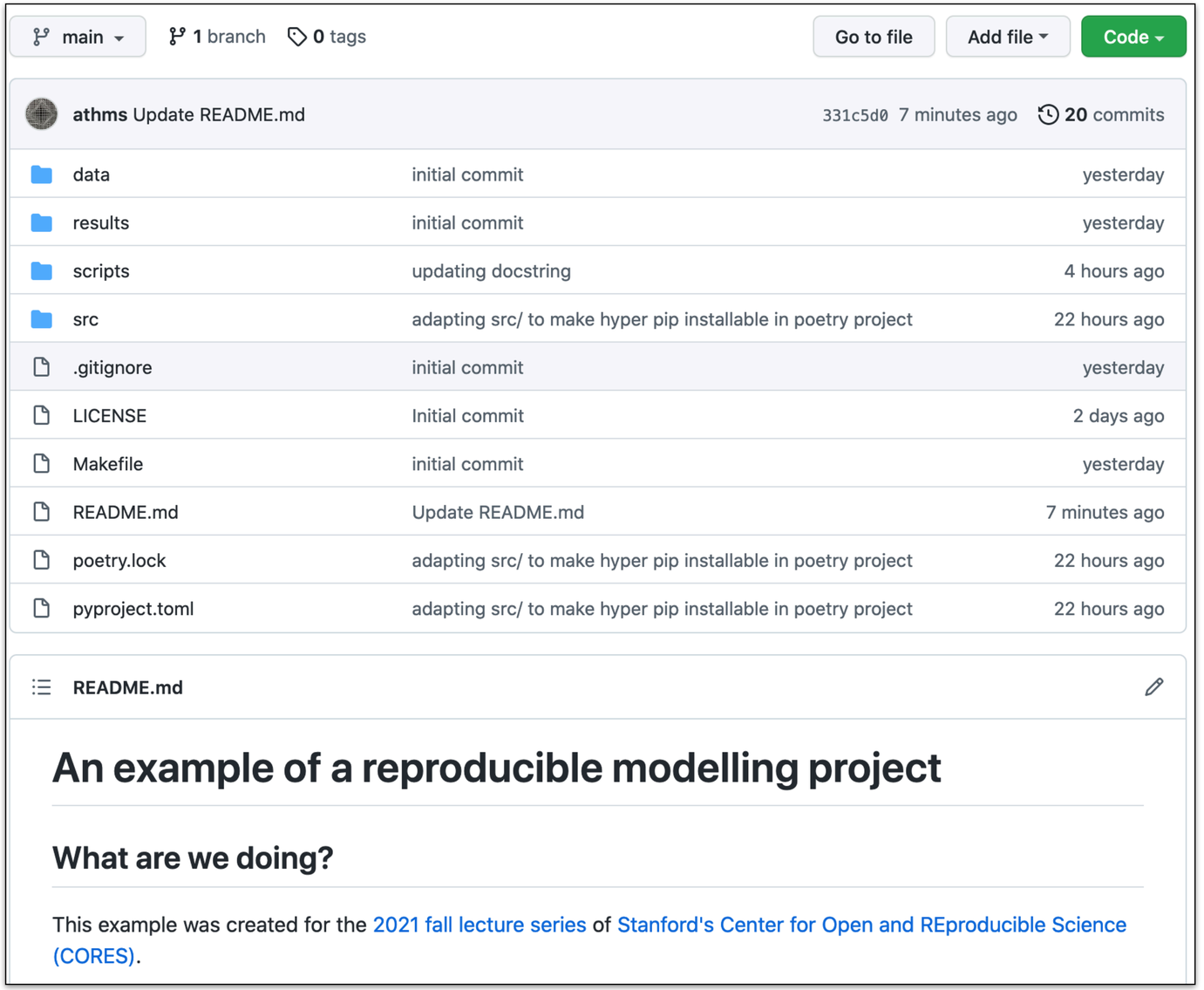
```
 create mode 100644 data/hand_written_digits_data.npy
 create mode 100644 poetry.lock
 create mode 100644 pyproject.toml
 create mode 100644 results/estimates/hyper_params_performance.csv
 create mode 100644 results/figures/fig_hyper_params_effects.png
 create mode 100644 scripts/evaluate_hyper_params_effect.py
 create mode 100644 scripts/load_data.py
 create mode 100644 scripts/plot_hyper_params_effect.py
 create mode 100644 scripts/run_analysis.sh
 create mode 100644 setup.py
 create mode 100644 src/__init__.py
 create mode 100644 src/hyper/__init__.py
 create mode 100644 src/hyper/evaluation.py
 create mode 100644 src/hyper/grid.py
 create mode 100644 src/hyper/plotting.py
(base) thomas@lip-osx-005002 reproducible-modelling % git push
Enumerating objects: 27, done.
Counting objects: 100% (27/27), done.
Delta compression using up to 8 threads
Compressing objects: 100% (24/24), done.
Writing objects: 100% (26/26), 243.29 KiB | 3.74 MiB/s, done.
Total 26 (delta 0), reused 0 (delta 0)
To https://github.com/athms/reproducible-modelling.git
   b84a378..fc0816b  main -> main
(base) thomas@lip-osx-005002 reproducible-modelling % git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
(base) thomas@lip-osx-005002 reproducible-modelling %
```

We are up to date!

Sharing your code

— so that others can see and use it

Your code is publicly available!



Improve inference reproducibility by ..

… being transparent about hyper-parameter evaluations

… accounting for different data splits with cross-validation

… accounting for random factors of training

…

Improve computational reproducibility by ..

… organising and documenting your project well

… sharing your virtual environment

… automating your analysis

… tracking and sharing your code with Git & GitHub

Thank you!