Developing and Deploying Transparent and Reproducible Algorithms for Public Health

R: tidymodels & plumber

Doug Manuel
Juan Li
Wenshan Li
Stacey Fisher

Minicourse team

- **Doug Manuel** – Professor, OHRI and uOttawa
- **Juan Li** – Research Associate
- **Wenshan Li** – Post-Doc
- **Stacey Fisher**– Post-Doc

Course outline

1) Understand open science concepts for algorithms in public health applications.
2) Develop an understanding of programming libraries and platforms in R to develop and deploy predictive algorithms. – Demo
3) Develop and deploy transparent and reproducible machine learning or statistical algorithms.

Course outline

1) Understand open science concepts for algorithms in public health applications.

2) Develop an understanding of programming libraries and platforms in R to develop and deploy predictive algorithms. – Demo

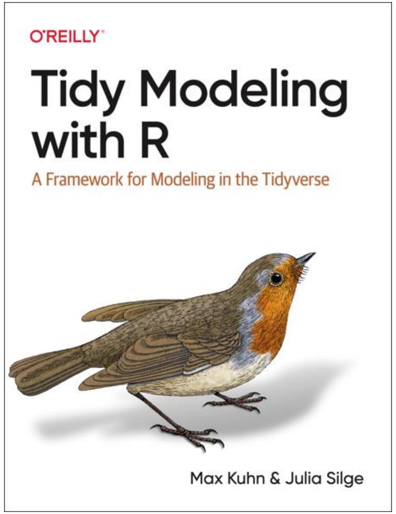
3) Develop and deploy transparent and reproducible machine learning or statistical algorithms.

TIDYMODELS

The tidymodels framework is a collection of packages for modeling and machine learning using tidyverse principles.

https://www.tidymodels.org/

https://www.tmwr.org/



O'REILLY

Tidy Modeling with R

A Framework for Modeling in the Tidyverse

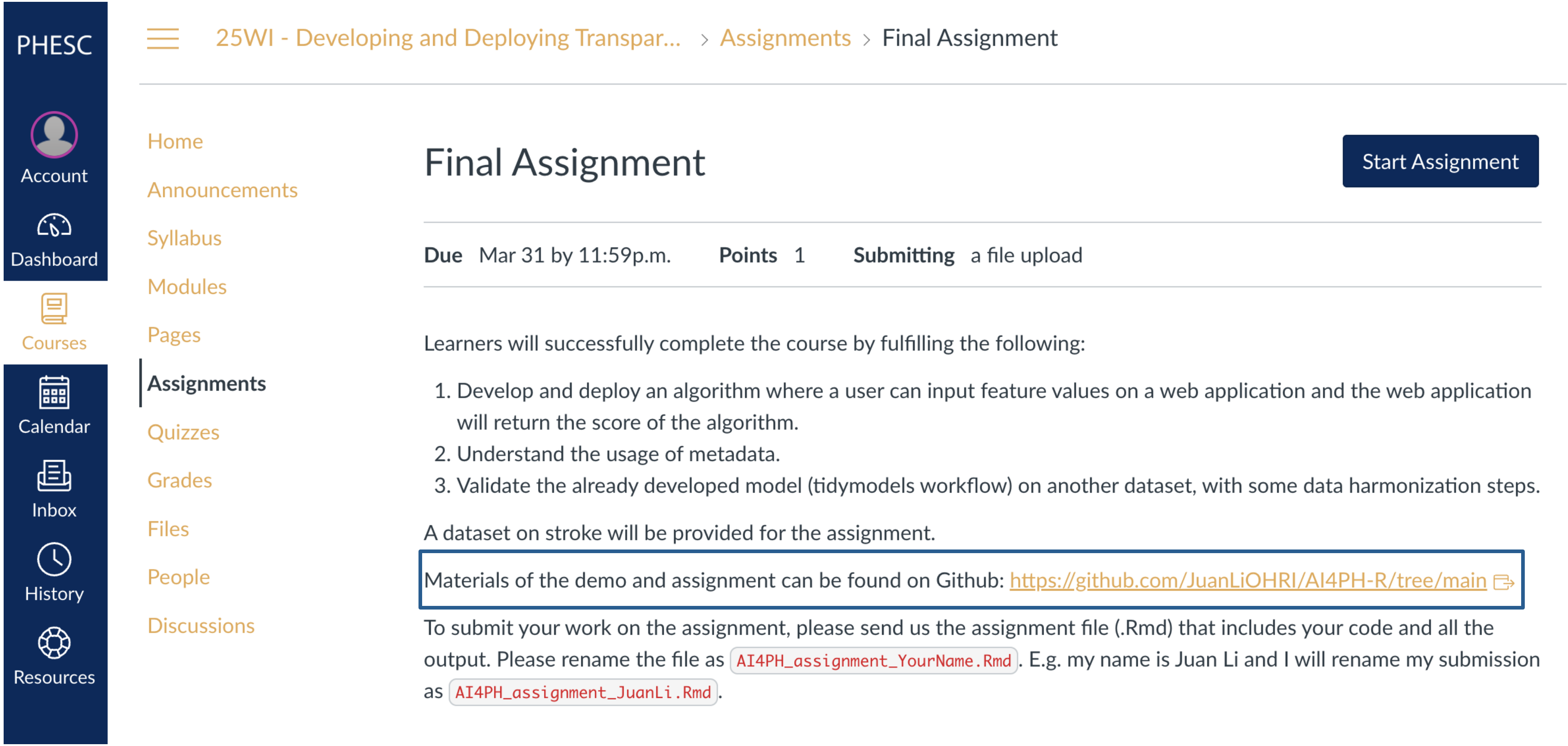
Max Kuhn & Julia Silge

PLUMBER

Plumber allows you to create a web API (Application Programming Interface) by merely decorating your existing R source code with roxygen2-like comments.

https://www.rplumber.io/

https://phesc.instructure.com/courses/398/assignments/1017



Final Assignment

Due Mar 31 by 11:59p.m. **Points** 1 **Submitting** a file upload

Learners will successfully complete the course by fulfilling the following:

1. Develop and deploy an algorithm where a user can input feature values on a web application and the web application will return the score of the algorithm.
2. Understand the usage of metadata.
3. Validate the already developed model (tidymodels workflow) on another dataset, with some data harmonization steps.

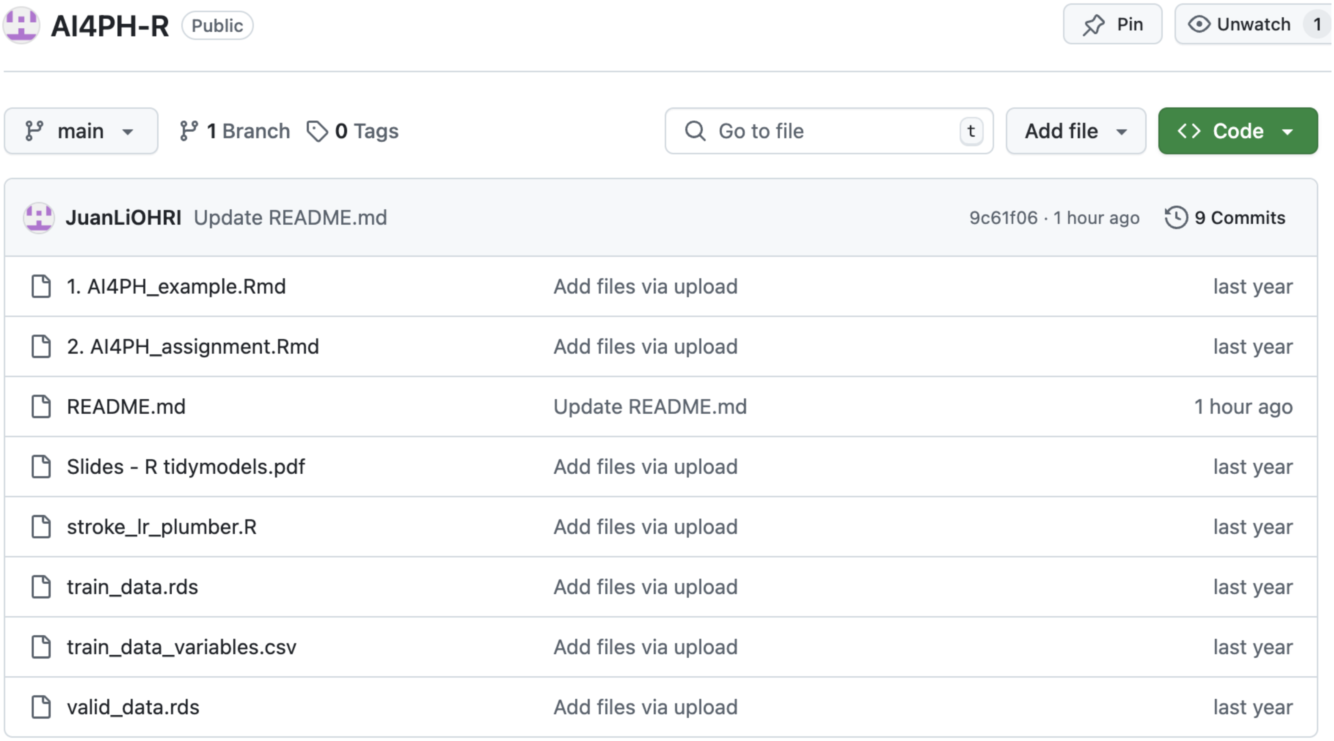
A dataset on stroke will be provided for the assignment.

Materials of the demo and assignment can be found on Github: https://github.com/JuanLiOHRI/AI4PH-R/tree/main

To submit your work on the assignment, please send us the assignment file (.Rmd) that includes your code and all the output. Please rename the file as `AI4PH_assignment_YourName.Rmd`. E.g. my name is Juan Li and I will rename my submission as `AI4PH_assignment_JuanLi.Rmd`.

Demo and assignment overview–
The Github repository (https://github.com/JuanLiOHRI/AI4PH-R/tree/main)

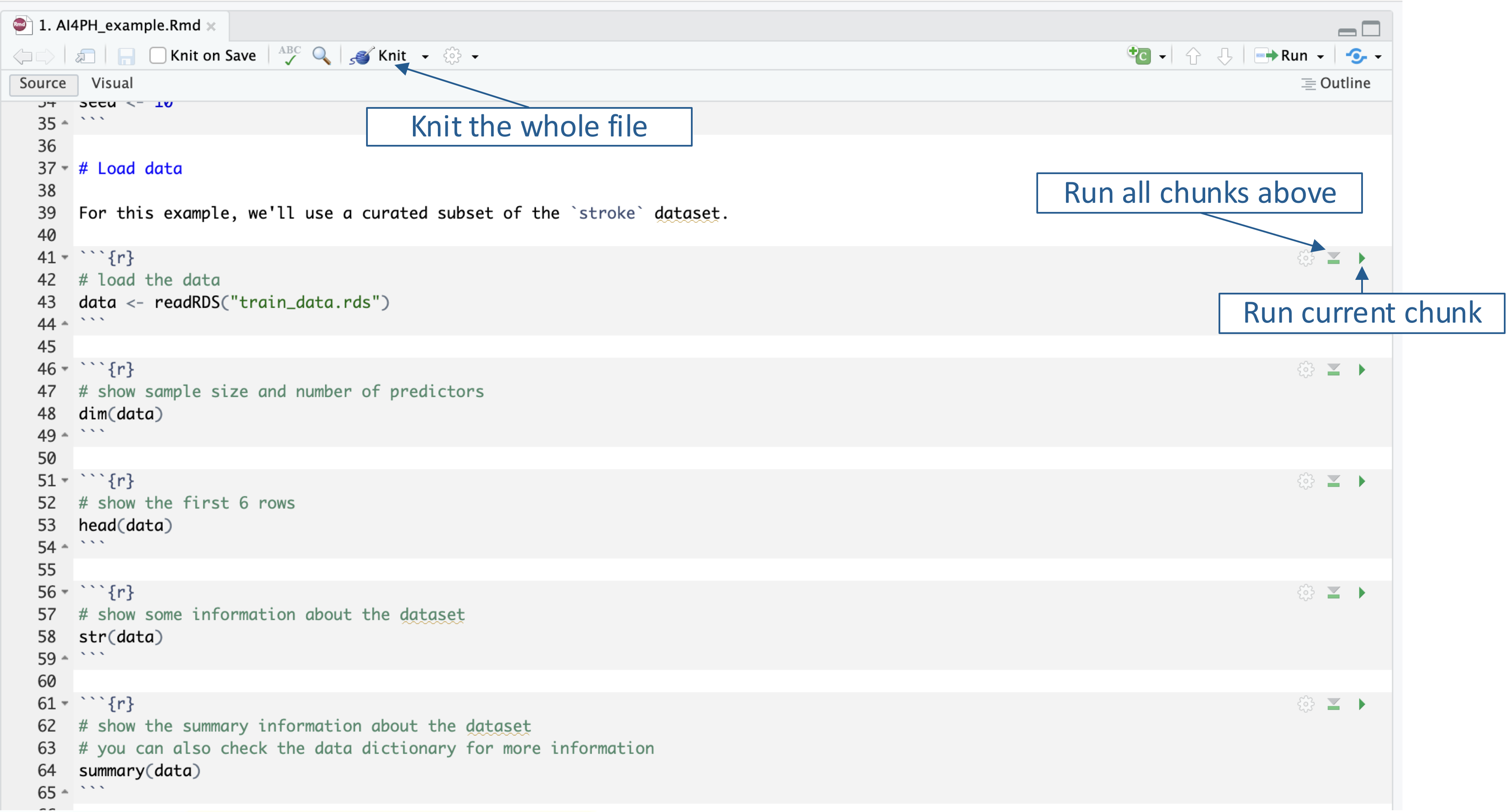
- The demo code
- The assignment code
- The Readme file
- The slide deck
- The plumber code
- Data for the demo
- The metadata
- Data for the assignment



JuanLiOHRI Update README.md 9c61f06 · 1 hour ago 9 Commits

1. AI4PH_example.Rmd	Add files via upload	last year
2. AI4PH_assignment.Rmd	Add files via upload	last year
README.md	Update README.md	1 hour ago
Slides - R tidymodels.pdf	Add files via upload	last year
stroke_lr_plumber.R	Add files via upload	last year
train_data.rds	Add files via upload	last year
train_data_variables.csv	Add files via upload	last year
valid_data.rds	Add files via upload	last year

Preparation - R markdown file (.Rmd)



Knit the whole file

Run all chunks above

Run current chunk

~~~
seed <- 10
```

# Load data

For this example, we'll use a curated subset of the `stroke` dataset.

```{r}
# load the data
data <- readRDS("train_data.rds")
```

```{r}
# show sample size and number of predictors
dim(data)
```

```{r}
# show the first 6 rows
head(data)
```

```{r}
# show some information about the dataset
str(data)
```

```{r}
# show the summary information about the dataset
# you can also check the data dictionary for more information
summary(data)
```
~~~

Preparation - Pipe operator: %>%

You can use the pipe operator (**%>%**) in R to “pipe” together a sequence of operations. This operator is most commonly used in R to perform a sequence of operations on a data frame.

The basic syntax for the pipe operator is:

```
df %>%
  operation1 %>%
  operation2 %>%
  operation3
  …
```

this is equivalent to

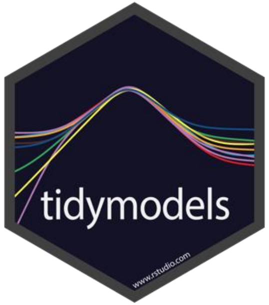
```
df_step1 <- operation1(df,…)
df_step2 <- operation2(df_step1, …)
df_step3 <- operation3(df_step2, …)
…
```

The pipe operator simply feeds the results of one operation into the next operation below it. The advantage of using the pipe operator is that it makes code extremely easy to read.

https://www.statology.org/pipe-in-r/

TIDYMODELS
Main packages

https://www.tidymodels.org/packages/



The meta-package

Data splitting
Train/test split, (nested)
cross validation (CV) etc.

Data pre-processing

Model interface

Model performance

Hyperparameter tuning
of your model and
pre-processing steps

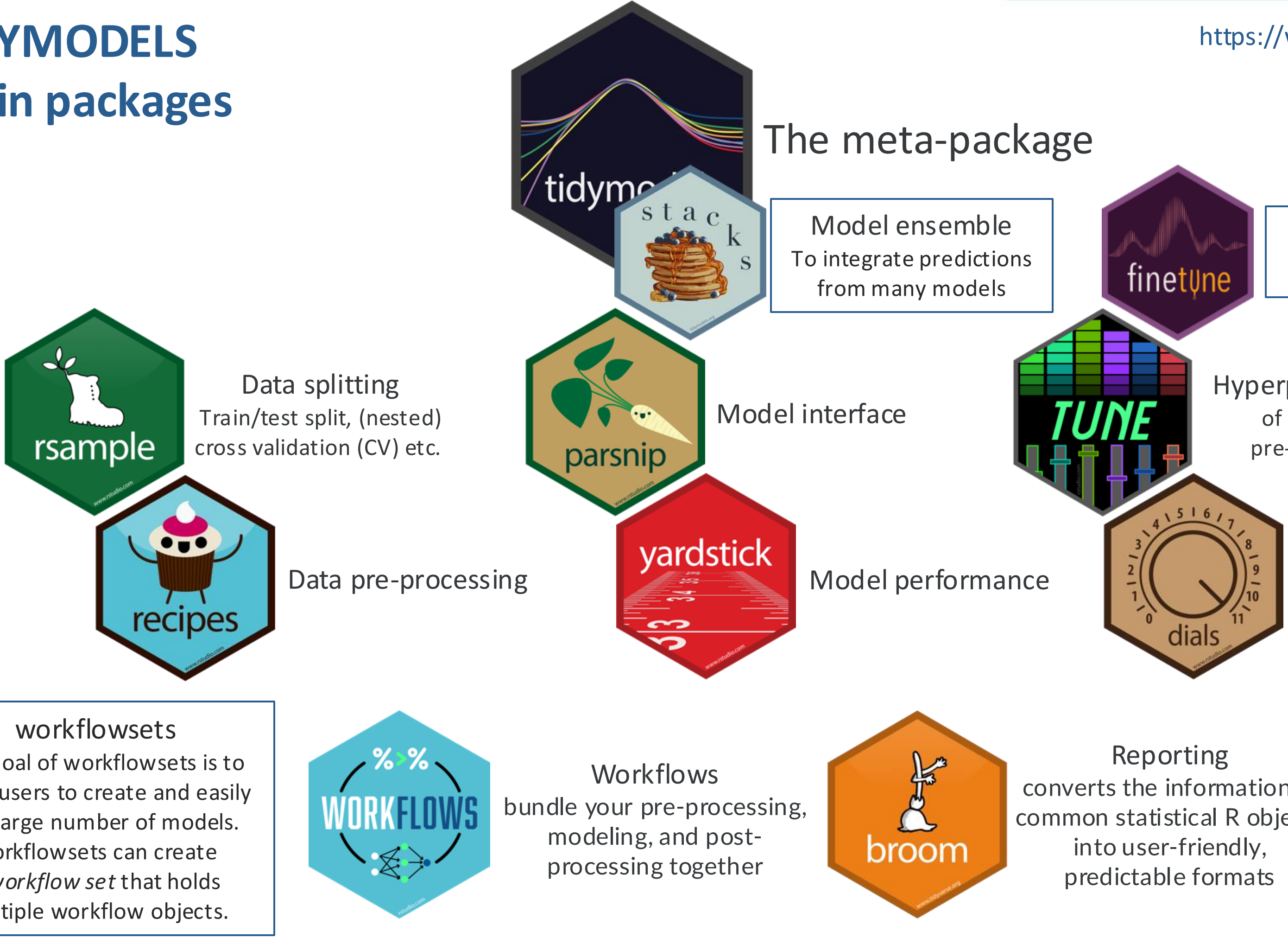
Creates and manages
tuning parameters
and parameter grids

Workflows
bundle your pre-processing,
modeling, and post-processing together

Reporting
converts the information in
common statistical R objects
into user-friendly,
predictable formats

TIDYMODELS
Main packages

https://www.tidymodels.org/packages/



The meta-package

Model ensemble
To integrate predictions from many models

Extra functions for tuning

Data splitting
Train/test split, (nested) cross validation (CV) etc.

Model interface

Hyperparameter tuning
of your model and pre-processing steps

Data pre-processing

Model performance

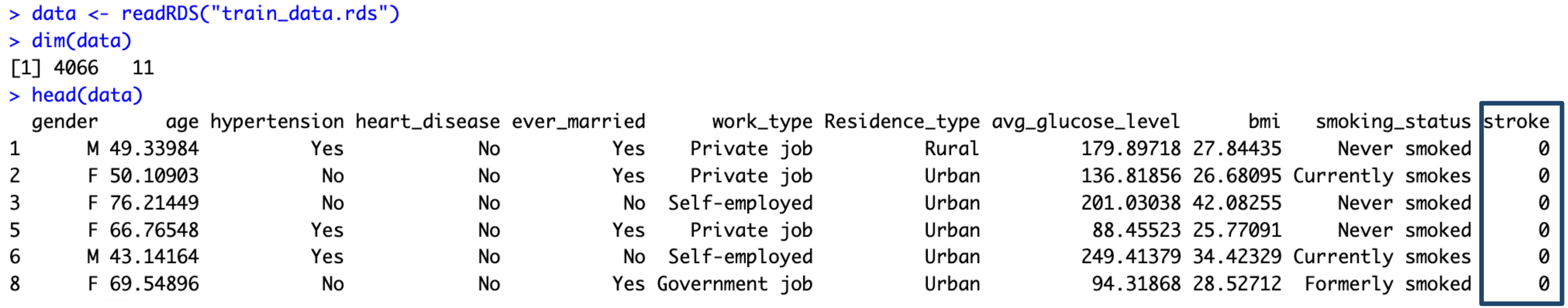
Creates and manages tuning parameters and parameter grids

workflowsets
The goal of workflowsets is to allow users to create and easily fit a large number of models. workflowsets can create a *workflow set* that holds multiple workflow objects.

Workflows
bundle your pre-processing, modeling, and post-processing together

Reporting
converts the information in common statistical R objects into user-friendly, predictable formats

TIDYMODELS (demo): a subset of the stroke (CCHS: Canadian Community Health Survey) data

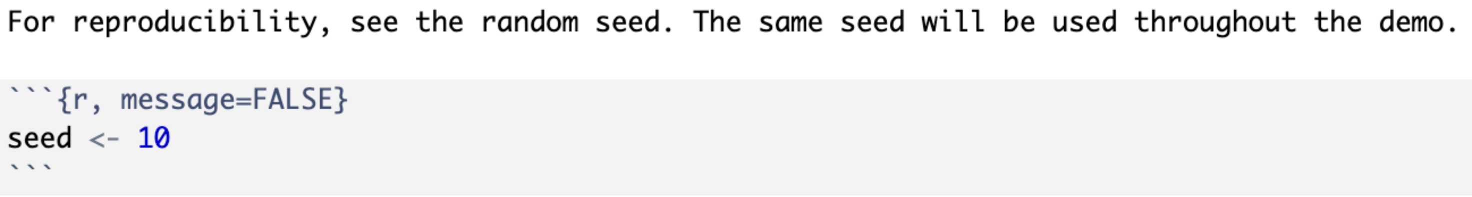


```
> data <- readRDS("train_data.rds")
> dim(data)
[1] 4066   11
> head(data)
  gender      age hypertension heart_disease ever_married      work_type Residence_type avg_glucose_level      bmi   smoking_status stroke
1      M 49.33984          Yes            No          Yes    Private job          Rural         179.89718 27.84435     Never smoked      0
2      F 50.10903           No            No          Yes    Private job          Urban         136.81856 26.68095 Currently smokes      0
3      F 76.21449           No            No           No  Self-employed          Urban         201.03038 42.08255     Never smoked      0
5      F 66.76548          Yes            No          Yes    Private job          Urban          88.45523 25.77091     Never smoked      0
6      M 43.14164          Yes            No           No  Self-employed          Urban         249.41379 34.42329 Currently smokes      0
8      F 69.54896           No            No          Yes Government job          Urban          94.31868 28.52712  Formerly smoked      0
```

variable	role	type	min	max	label
gender	predictor	Categorical			F; M
age	predictor	Continuous	40	100	
hypertension	predictor	Categorical			No; Yes
heart_disease	predictor	Categorical			No; Yes
ever_married	predictor	Categorical			No; Yes
work_type	predictor	Categorical			Self-employed; Private job; Government job
Residence_type	predictor	Categorical			Rural; Urban
avg_glucose_level	predictor	Continuous	0	310	
bmi	predictor	Continuous	9	73	
smoking_status	predictor	Categorical			Never smoked; Formerly smoked; Currently smokes
stroke	outcome	Categorical			0; 1

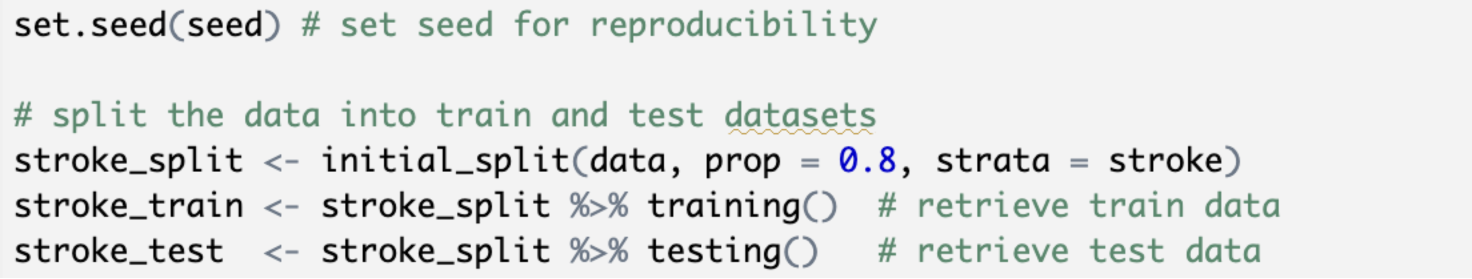
Actual or allowed range.

TIDYMODELS (demo): data splitting



For reproducibility, see the random seed. The same seed will be used throughout the demo.

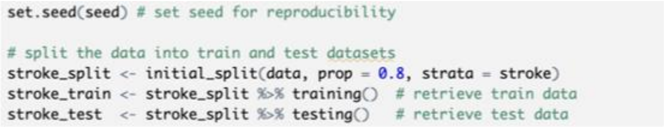
~~~
```{r, message=FALSE}
seed <- 10
```
~~~



```
set.seed(seed) # set seed for reproducibility

# split the data into train and test datasets
stroke_split <- initial_split(data, prop = 0.8, strata = stroke)
stroke_train <- stroke_split %>% training()  # retrieve train data
stroke_test  <- stroke_split %>% testing()   # retrieve test data
```

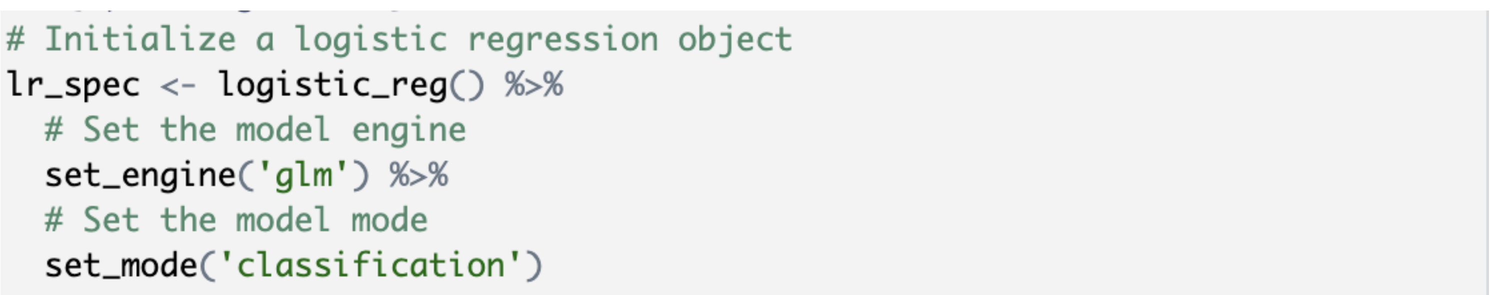
TIDYMODELS (demo): model initializing

rsample



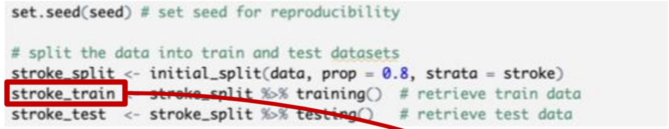
```r
set.seed(seed) # set seed for reproducibility

# split the data into train and test datasets
stroke_split <- initial_split(data, prop = 0.8, strata = stroke)
stroke_train <- stroke_split %>% training()  # retrieve train data
stroke_test  <- stroke_split %>% testing()   # retrieve test data
```

parsnip

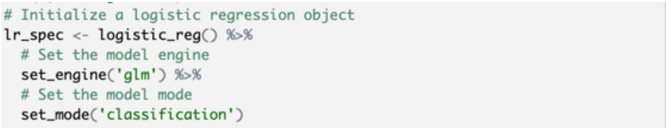


```r
# Initialize a logistic regression object
lr_spec <- logistic_reg() %>%
  # Set the model engine
  set_engine('glm') %>%
  # Set the model mode
  set_mode('classification')
```

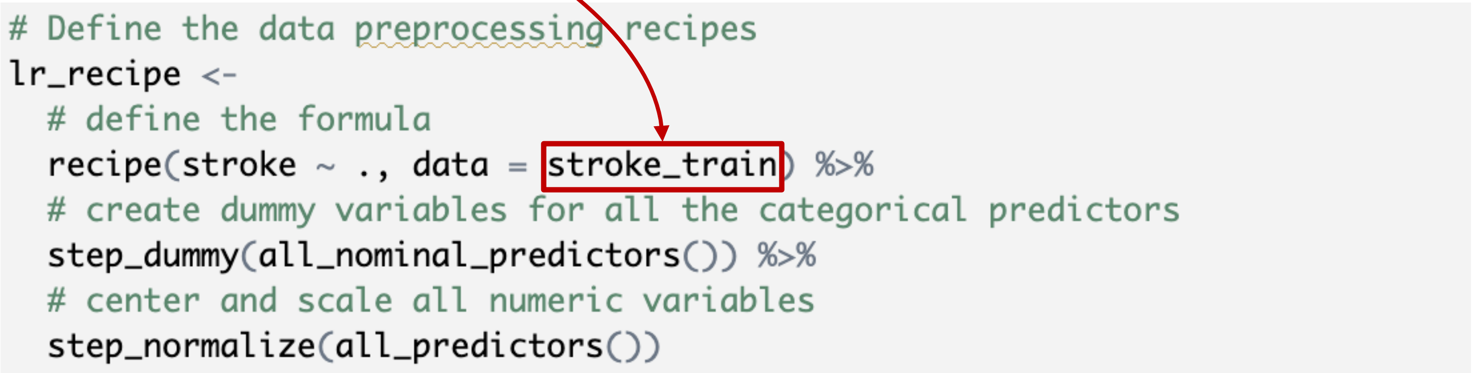
TIDYMODELS (demo): data preprocessing



```
set.seed(seed) # set seed for reproducibility

# split the data into train and test datasets
stroke_split <- initial_split(data, prop = 0.8, strata = stroke)
stroke_train <- stroke_split %>% training()  # retrieve train data
stroke_test  <- stroke_split %>% testing()   # retrieve test data
```



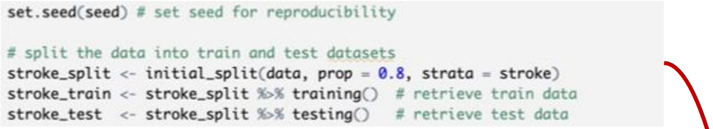
```
# Initialize a logistic regression object
lr_spec <- logistic_reg() %>%
  # Set the model engine
  set_engine('glm') %>%
  # Set the model mode
  set_mode('classification')
```



```
# Define the data preprocessing recipes
lr_recipe <-
  # define the formula
  recipe(stroke ~ ., data = stroke_train) %>%
  # create dummy variables for all the categorical predictors
  step_dummy(all_nominal_predictors()) %>%
  # center and scale all numeric variables
  step_normalize(all_predictors())
```

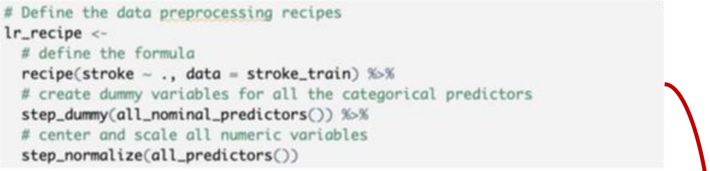
TIDYMODELS (demo): define and train the workflow (model with pre-processing steps)

rsample



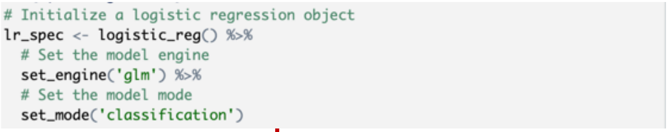
```
set.seed(seed) # set seed for reproducibility

# split the data into train and test datasets
stroke_split <- initial_split(data, prop = 0.8, strata = stroke)
stroke_train <- stroke_split %>% training()  # retrieve train data
stroke_test  <- stroke_split %>% testing()   # retrieve test data
```

recipes



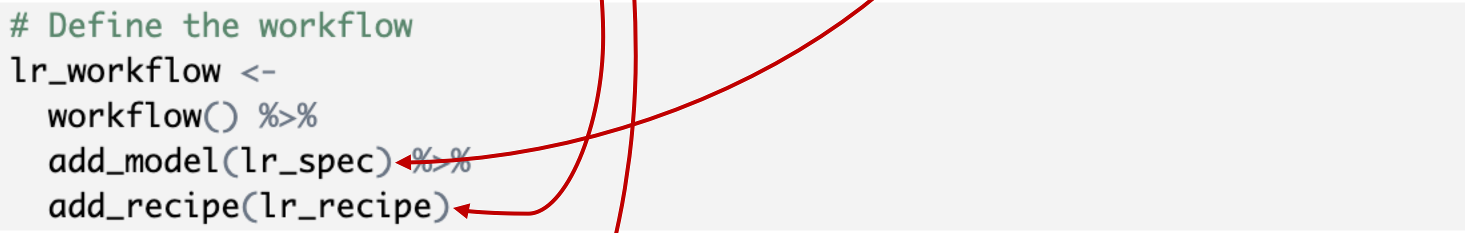
```
# Define the data preprocessing recipes
lr_recipe <-
  # define the formula
  recipe(stroke ~ ., data = stroke_train) %>%
  # create dummy variables for all the categorical predictors
  step_dummy(all_nominal_predictors()) %>%
  # center and scale all numeric variables
  step_normalize(all_predictors())
```

parsnip

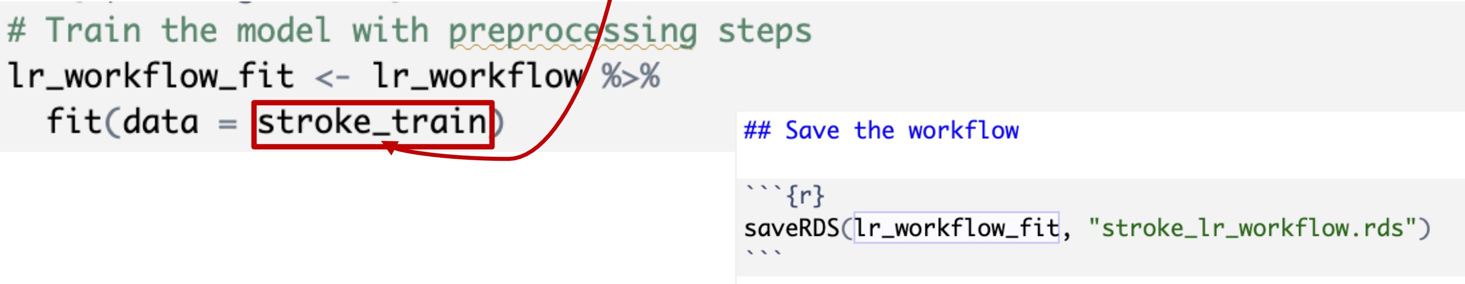


```
# Initialize a logistic regression object
lr_spec <- logistic_reg() %>%
  # Set the model engine
  set_engine('glm') %>%
  # Set the model mode
  set_mode('classification')
```

%>% WORKFLOWS



```
# Define the workflow
lr_workflow <-
  workflow() %>%
  add_model(lr_spec) %>%
  add_recipe(lr_recipe)
```

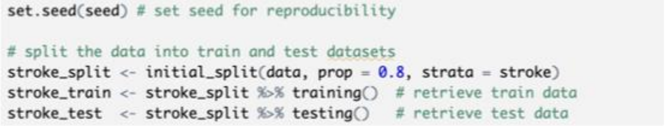


```
# Train the model with preprocessing steps
lr_workflow_fit <- lr_workflow %>%
  fit(data = stroke_train)
```

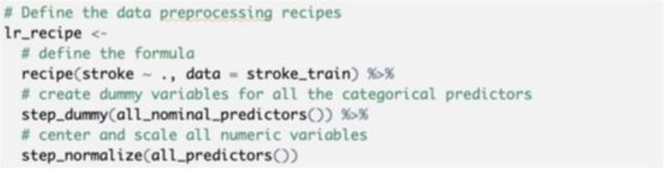
Save the workflow

~~~
```{r}
saveRDS(lr_workflow_fit, "stroke_lr_workflow.rds")
```
~~~

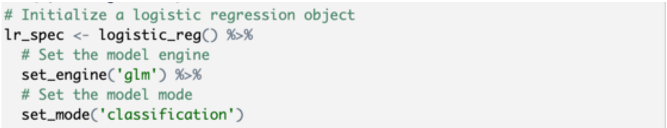
TIDYMODELS (demo): obtaining the estimated coefficients



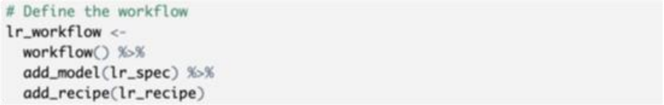
```r
set.seed(seed) # set seed for reproducibility

# split the data into train and test datasets
stroke_split <- initial_split(data, prop = 0.8, strata = stroke)
stroke_train <- stroke_split %>% training()  # retrieve train data
stroke_test  <- stroke_split %>% testing()   # retrieve test data
```



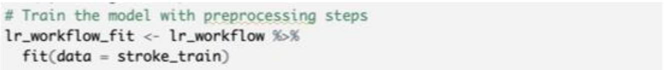
```r
# Define the data preprocessing recipes
lr_recipe <-
  # define the formula
  recipe(stroke ~ ., data = stroke_train) %>%
  # create dummy variables for all the categorical predictors
  step_dummy(all_nominal_predictors()) %>%
  # center and scale all numeric variables
  step_normalize(all_predictors())
```



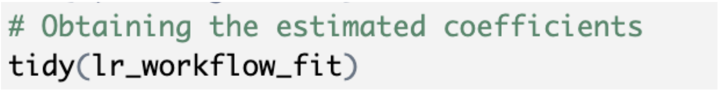
```r
# Initialize a logistic regression object
lr_spec <- logistic_reg() %>%
  # Set the model engine
  set_engine('glm') %>%
  # Set the model mode
  set_mode('classification')
```



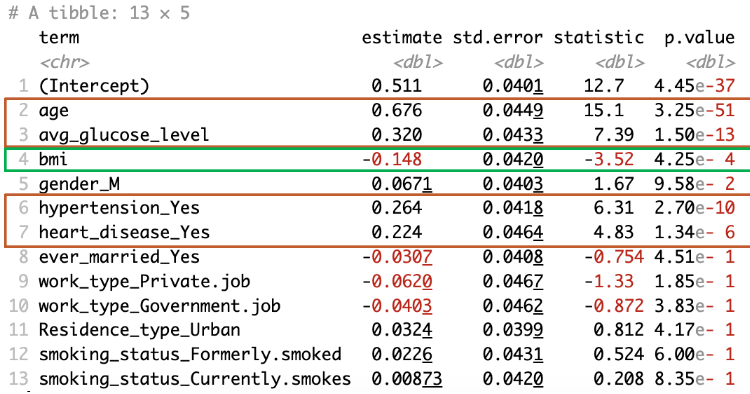
```r
# Define the workflow
lr_workflow <-
  workflow() %>%
  add_model(lr_spec) %>%
  add_recipe(lr_recipe)
```



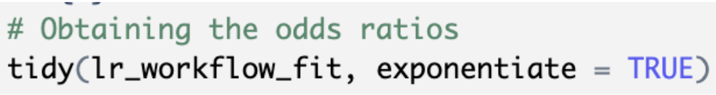
```r
# Train the model with preprocessing steps
lr_workflow_fit <- lr_workflow %>%
  fit(data = stroke_train)
```



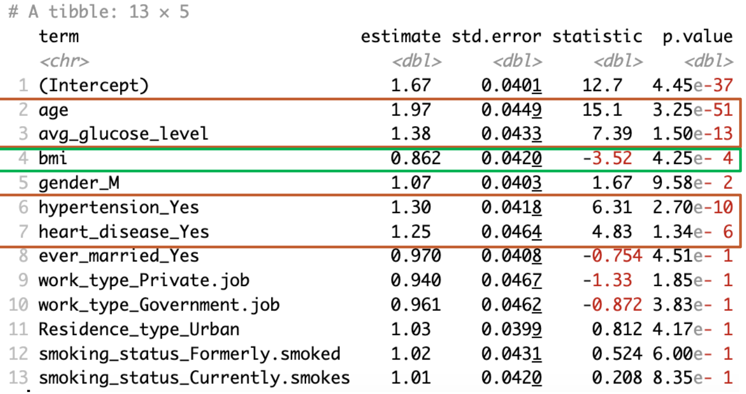
```r
# Obtaining the estimated coefficients
tidy(lr_workflow_fit)
```



```
# A tibble: 13 × 5
   term                           estimate std.error statistic  p.value
   <chr>                             <dbl>     <dbl>     <dbl>    <dbl>
 1 (Intercept)                     0.511      0.0401    12.7   4.45e-37
 2 age                             0.676      0.0449    15.1   3.25e-51
 3 avg_glucose_level               0.320      0.0433     7.39  1.50e-13
 4 bmi                            -0.148      0.0420    -3.52  4.25e- 4
 5 gender_M                        0.0671     0.0403     1.67  9.58e- 2
 6 hypertension_Yes                0.264      0.0418     6.31  2.70e-10
 7 heart_disease_Yes               0.224      0.0464     4.83  1.34e- 6
 8 ever_married_Yes               -0.0307     0.0408    -0.754 4.51e- 1
 9 work_type_Private.job          -0.0620     0.0467    -1.33  1.85e- 1
10 work_type_Government.job       -0.0403     0.0462    -0.872 3.83e- 1
11 Residence_type_Urban            0.0324     0.0399     0.812 4.17e- 1
12 smoking_status_Formerly.smoked  0.0226     0.0431     0.524 6.00e- 1
13 smoking_status_Currently.smokes 0.00873    0.0420     0.208 8.35e- 1
```

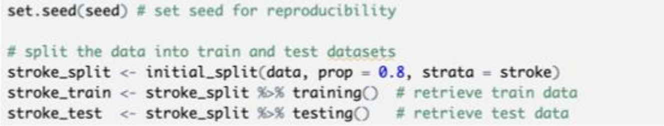


```r
# Obtaining the odds ratios
tidy(lr_workflow_fit, exponentiate = TRUE)
```

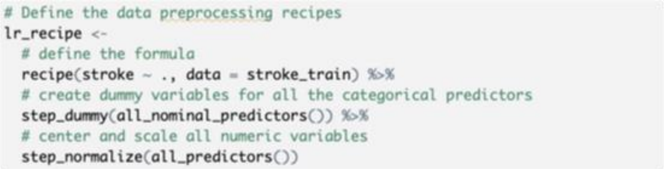


```
# A tibble: 13 × 5
   term                           estimate std.error statistic  p.value
   <chr>                             <dbl>     <dbl>     <dbl>    <dbl>
 1 (Intercept)                       1.67     0.0401    12.7   4.45e-37
 2 age                               1.97     0.0449    15.1   3.25e-51
 3 avg_glucose_level                 1.38     0.0433     7.39  1.50e-13
 4 bmi                               0.862    0.0420    -3.52  4.25e- 4
 5 gender_M                          1.07     0.0403     1.67  9.58e- 2
 6 hypertension_Yes                  1.30     0.0418     6.31  2.70e-10
 7 heart_disease_Yes                 1.25     0.0464     4.83  1.34e- 6
 8 ever_married_Yes                  0.970    0.0408    -0.754 4.51e- 1
 9 work_type_Private.job             0.940    0.0467    -1.33  1.85e- 1
10 work_type_Government.job          0.961    0.0462    -0.872 3.83e- 1
11 Residence_type_Urban              1.03     0.0399     0.812 4.17e- 1
12 smoking_status_Formerly.smoked    1.02     0.0431     0.524 6.00e- 1
13 smoking_status_Currently.smokes   1.01     0.0420     0.208 8.35e- 1
```

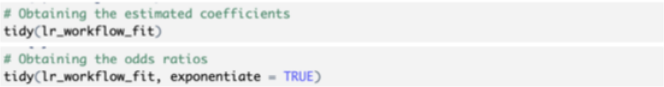
TIDYMODELS (demo): predicting on the test data



```r
set.seed(seed) # set seed for reproducibility

# split the data into train and test datasets
stroke_split <- initial_split(data, prop = 0.8, strata = stroke)
stroke_train <- stroke_split %>% training()  # retrieve train data
stroke_test  <- stroke_split %>% testing()   # retrieve test data
```

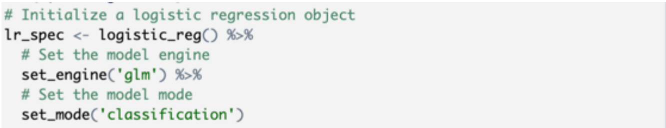


```r
# Define the data preprocessing recipes
lr_recipe <-
  # define the formula
  recipe(stroke ~ ., data = stroke_train) %>%
  # create dummy variables for all the categorical predictors
  step_dummy(all_nominal_predictors()) %>%
  # center and scale all numeric variables
  step_normalize(all_predictors())
```

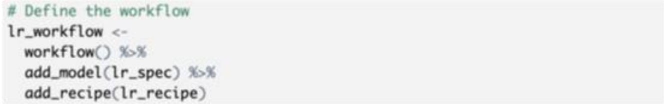


```r
# Obtaining the estimated coefficients
tidy(lr_workflow_fit)
```

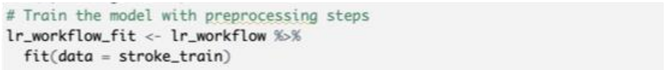
```r
# Obtaining the odds ratios
tidy(lr_workflow_fit, exponentiate = TRUE)
```



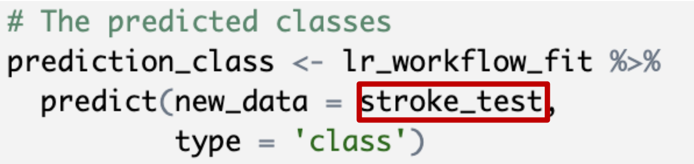
```r
# Initialize a logistic regression object
lr_spec <- logistic_reg() %>%
  # Set the model engine
  set_engine('glm') %>%
  # Set the model mode
  set_mode('classification')
```



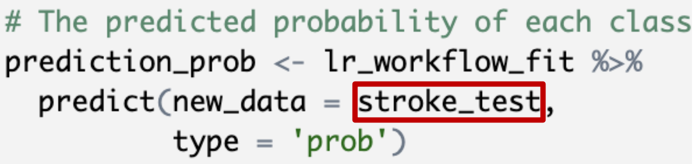
```r
# Define the workflow
lr_workflow <-
  workflow() %>%
  add_model(lr_spec) %>%
  add_recipe(lr_recipe)
```



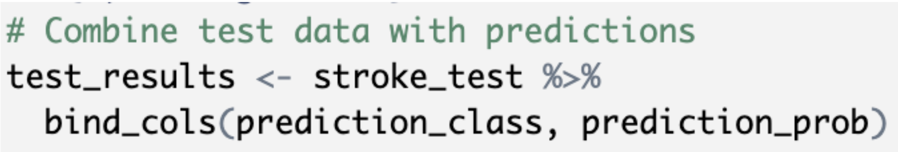
```r
# Train the model with preprocessing steps
lr_workflow_fit <- lr_workflow %>%
  fit(data = stroke_train)
```



```r
# The predicted classes
prediction_class <- lr_workflow_fit %>%
  predict(new_data = stroke_test,
          type = 'class')
```

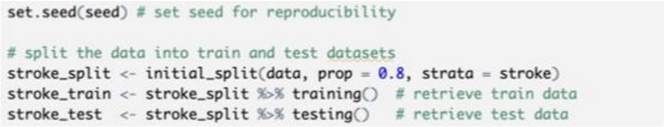


```r
# The predicted probability of each class
prediction_prob <- lr_workflow_fit %>%
  predict(new_data = stroke_test,
          type = 'prob')
```

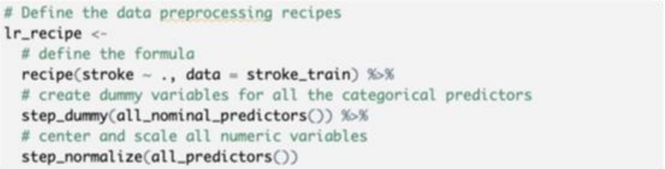


```r
# Combine test data with predictions
test_results <- stroke_test %>%
  bind_cols(prediction_class, prediction_prob)
```

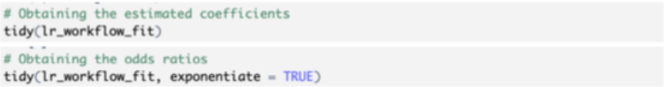
TIDYMODELS (demo): model evaluation on the test data



```
set.seed(seed) # set seed for reproducibility

# split the data into train and test datasets
stroke_split <- initial_split(data, prop = 0.8, strata = stroke)
stroke_train <- stroke_split %>% training()  # retrieve train data
stroke_test  <- stroke_split %>% testing()   # retrieve test data
```

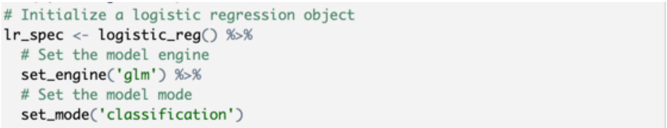


```
# Define the data preprocessing recipes
lr_recipe <-
  # define the formula
  recipe(stroke ~ ., data = stroke_train) %>%
  # create dummy variables for all the categorical predictors
  step_dummy(all_nominal_predictors()) %>%
  # center and scale all numeric variables
  step_normalize(all_predictors())
```

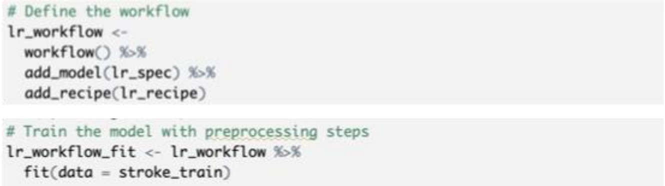


```
# Obtaining the estimated coefficients
tidy(lr_workflow_fit)
```

```
# Obtaining the odds ratios
tidy(lr_workflow_fit, exponentiate = TRUE)
```

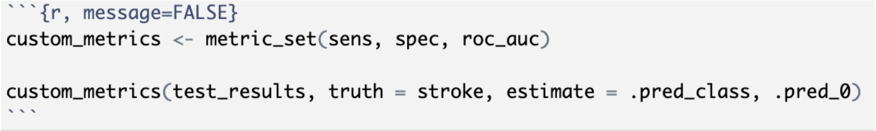


```
# Initialize a logistic regression object
lr_spec <- logistic_reg() %>%
  # Set the model engine
  set_engine('glm') %>%
  # Set the model mode
  set_mode('classification')
```

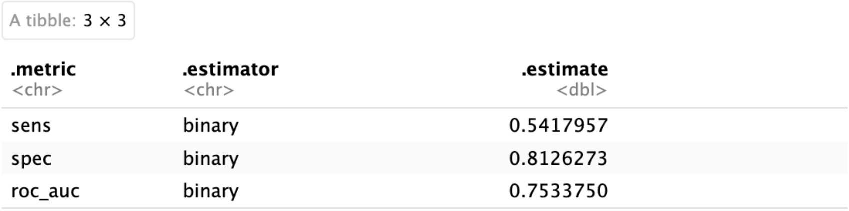


```
# Define the workflow
lr_workflow <-
  workflow() %>%
  add_model(lr_spec) %>%
  add_recipe(lr_recipe)
```

```
# Train the model with preprocessing steps
lr_workflow_fit <- lr_workflow %>%
  fit(data = stroke_train)
```

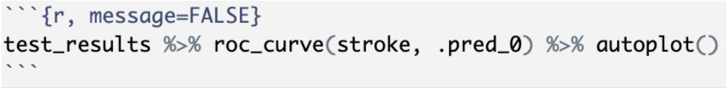


~~~
```{r, message=FALSE}
custom_metrics <- metric_set(sens, spec, roc_auc)

custom_metrics(test_results, truth = stroke, estimate = .pred_class, .pred_0)
```
~~~

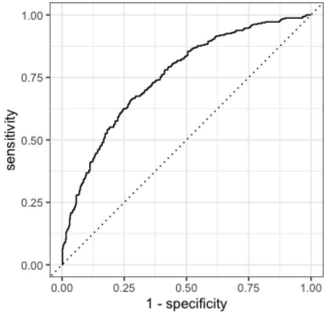


A tibble: 3 × 3

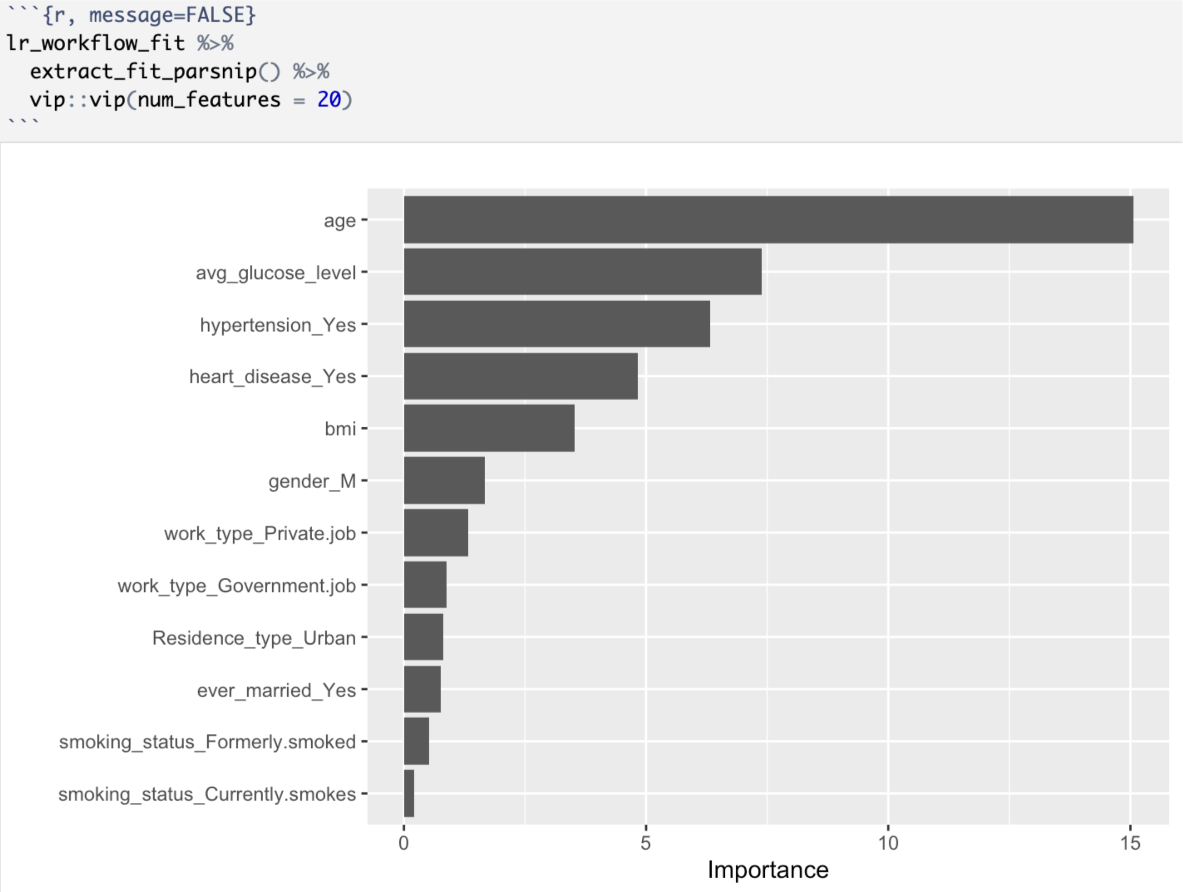
.metric <chr>	.estimator <chr>	.estimate <dbl>
sens	binary	0.5417957
spec	binary	0.8126273
roc_auc	binary	0.7533750



~~~
```{r, message=FALSE}
test_results %>% roc_curve(stroke, .pred_0) %>% autoplot()
```
~~~

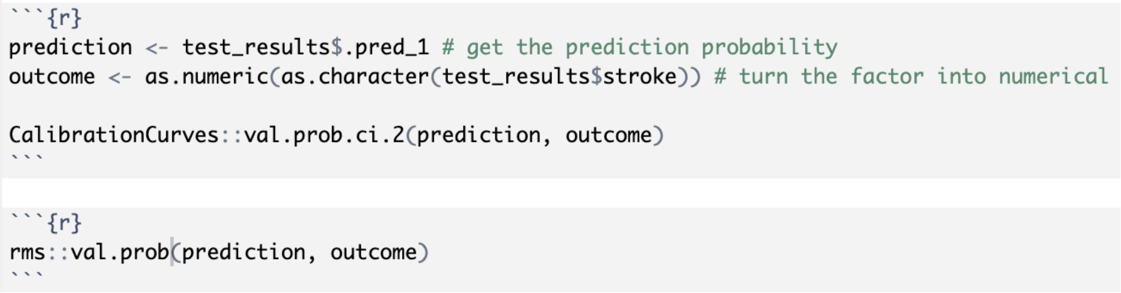


TIDYMODELS (demo): feature importance



~~~
```{r, message=FALSE}
lr_workflow_fit %>%
  extract_fit_parsnip() %>%
  vip::vip(num_features = 20)
```
~~~

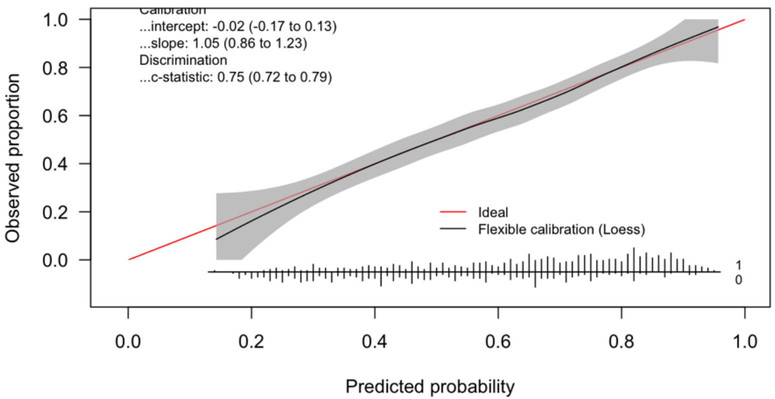
TIDYMODELS (demo): calibration



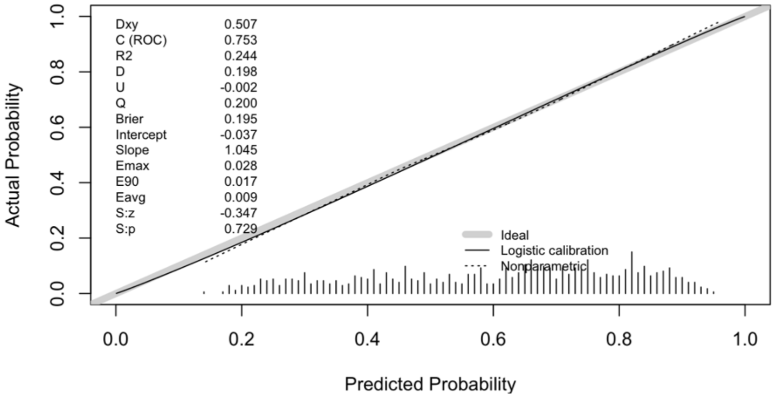
~~~
```{r}
prediction <- test_results$.pred_1 # get the prediction probability
outcome <- as.numeric(as.character(test_results$stroke)) # turn the factor into numerical

CalibrationCurves::val.prob.ci.2(prediction, outcome)
```
~~~

~~~
```{r}
rms::val.prob(prediction, outcome)
```
~~~

```
       Dxy     C (ROC)          R2           D     D:Chi-sq         D:p           U    U:Chi-sq         U:p           Q
0.50676259  0.75338130  0.24432443  0.19791253 162.10079850  0.00000000 -0.00207614  0.31002177  0.85640585  0.19998867
     Brier   Intercept       Slope        Emax Brier scaled        Eavg         ECI
0.19461773 -0.02020034  1.04540841  0.01588131   0.18689393  0.00899989  0.01437638
```



```
        Dxy      C (ROC)           R2            D     D:Chi-sq          D:p            U     U:Chi-sq          U:p            Q
0.506800426  0.753400213  0.243902052  0.197531667 161.790776727           NA -0.002076140  0.310021772  0.856405855  0.199607807
      Brier    Intercept        Slope         Emax          E90         Eavg          S:z          S:p
0.194617731 -0.036699797  1.045408415  0.028080302  0.016670572  0.008667897 -0.346720345  0.728801424
```



PLUMBER (demo)

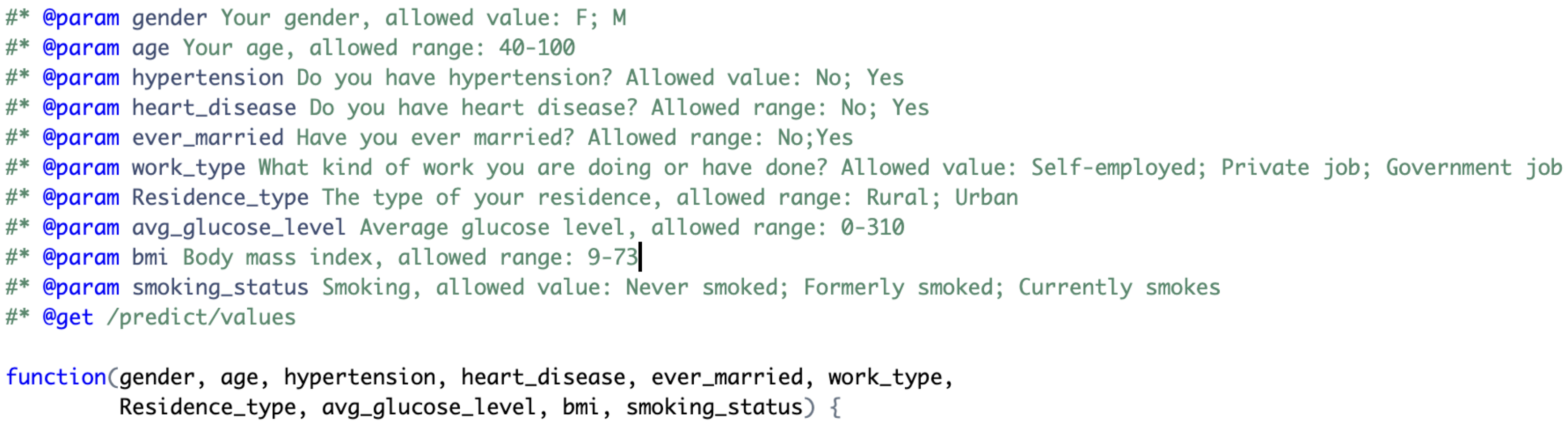
stroke_lr_workflow.rds

The saved workflow (model with data preprocessing steps) that has been trained on the train data (and hopefully evaluated on the test data).

stroke_lr_plumber.R

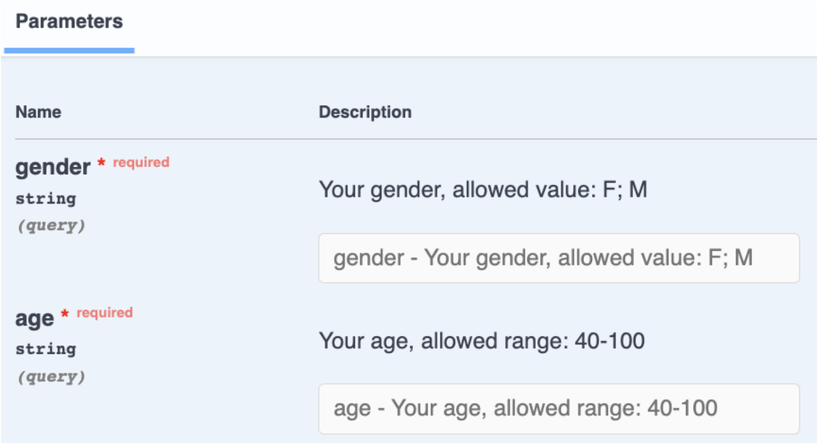
The plumber script.

PLUMBER (demo)



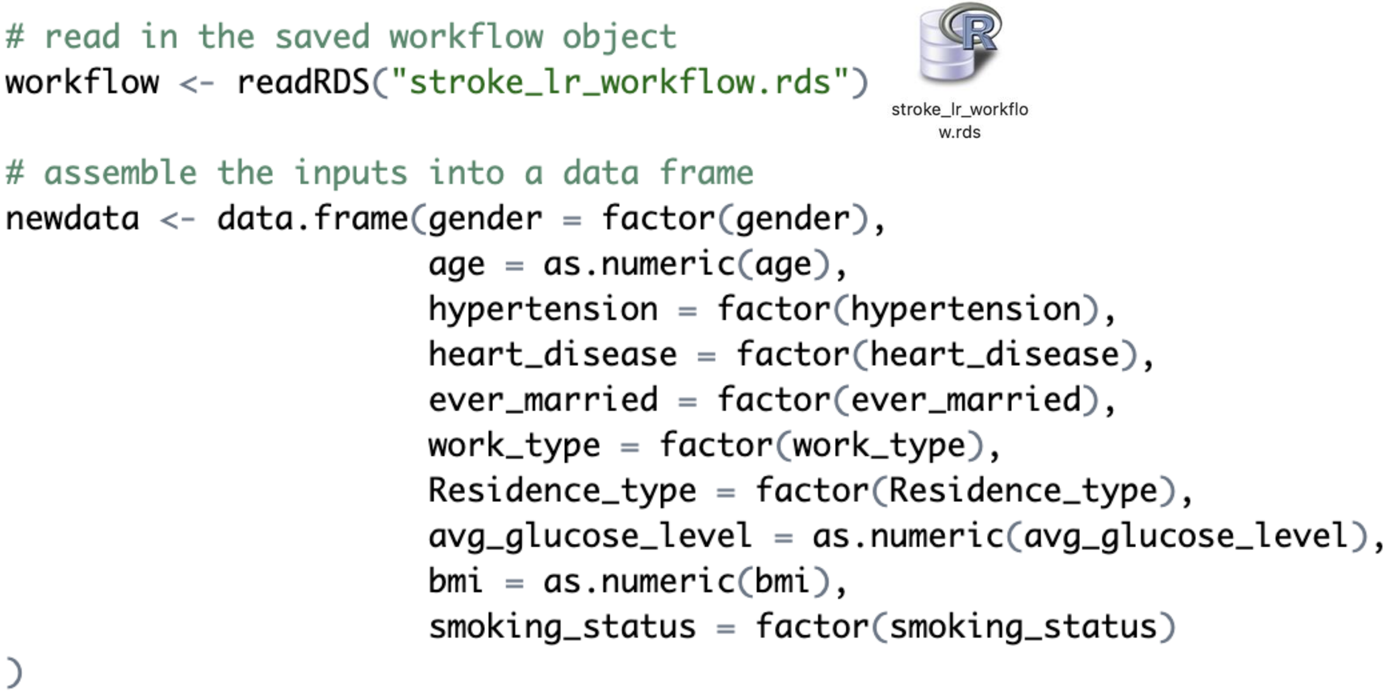
```r
#* @param gender Your gender, allowed value: F; M
#* @param age Your age, allowed range: 40-100
#* @param hypertension Do you have hypertension? Allowed value: No; Yes
#* @param heart_disease Do you have heart disease? Allowed range: No; Yes
#* @param ever_married Have you ever married? Allowed range: No;Yes
#* @param work_type What kind of work you are doing or have done? Allowed value: Self-employed; Private job; Government job
#* @param Residence_type The type of your residence, allowed range: Rural; Urban
#* @param avg_glucose_level Average glucose level, allowed range: 0-310
#* @param bmi Body mass index, allowed range: 9-73
#* @param smoking_status Smoking, allowed value: Never smoked; Formerly smoked; Currently smokes
#* @get /predict/values

function(gender, age, hypertension, heart_disease, ever_married, work_type,
         Residence_type, avg_glucose_level, bmi, smoking_status) {
```

PLUMBER
Plumber allows you to create a web API (Application Programming Interface) by merely decorating your existing R source code with roxygen2-like comments.
https://www.rplumber.io/



Name	Description
gender * required string *(query)*	Your gender, allowed value: F; M gender - Your gender, allowed value: F; M
age * required string *(query)*	Your age, allowed range: 40-100 age - Your age, allowed range: 40-100

PLUMBER (demo)

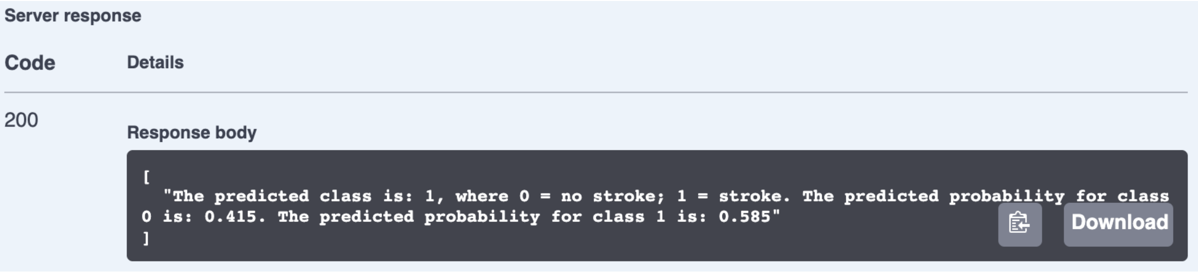


```
# read in the saved workflow object
workflow <- readRDS("stroke_lr_workflow.rds")

# assemble the inputs into a data frame
newdata <- data.frame(gender = factor(gender),
                      age = as.numeric(age),
                      hypertension = factor(hypertension),
                      heart_disease = factor(heart_disease),
                      ever_married = factor(ever_married),
                      work_type = factor(work_type),
                      Residence_type = factor(Residence_type),
                      avg_glucose_level = as.numeric(avg_glucose_level),
                      bmi = as.numeric(bmi),
                      smoking_status = factor(smoking_status)
)
```

stroke_lr_workflow.rds

PLUMBER (demo)

```
# predict on the new data - class
prediction_class <- workflow %>%
  predict(new_data = newdata,
          type = 'class')

# predict on the new data - probability
prediction_prob <- workflow %>%
  predict(new_data = newdata,
          type = 'prob')

# report result
print(paste("The predicted class is: ", prediction_class$.pred_class, ", where 0 = no stroke; 1 = stroke",
            ". The predicted probability for class 0 is: ", round(prediction_prob[1],3),
            ". The predicted probability for class 1 is: ", round(prediction_prob[2],3), sep = ""))
```

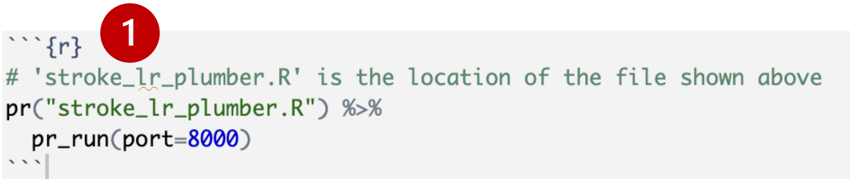


Server response

Code	Details
200	Response body

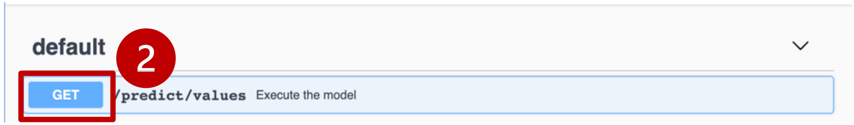
```
[
  "The predicted class is: 1, where 0 = no stroke; 1 = stroke. The predicted probability for class
0 is: 0.415. The predicted probability for class 1 is: 0.585"
]
```

PLUMBER (demo)

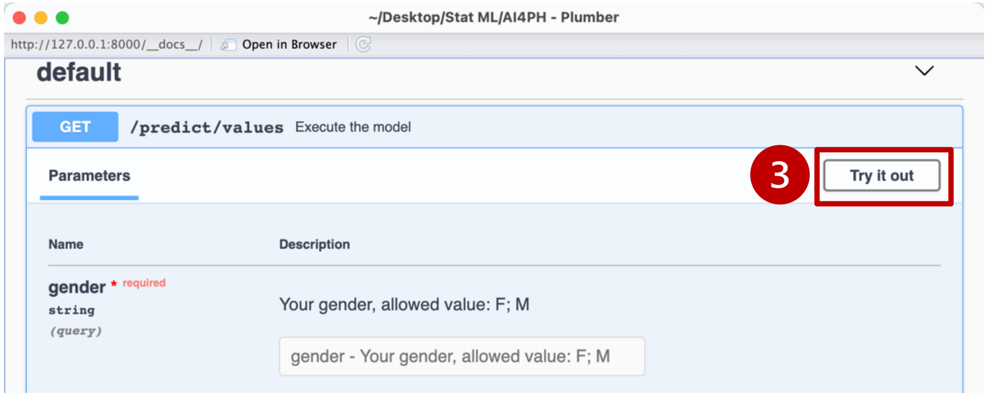


1

~~~
```{r}
# 'stroke_lr_plumber.R' is the location of the file shown above
pr("stroke_lr_plumber.R") %>%
  pr_run(port=8000)
```
~~~

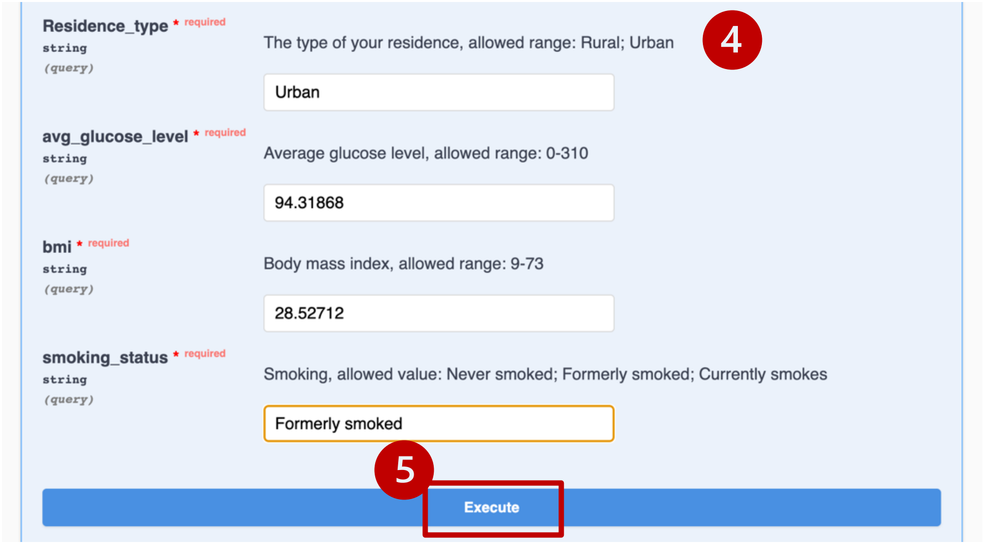


2



3

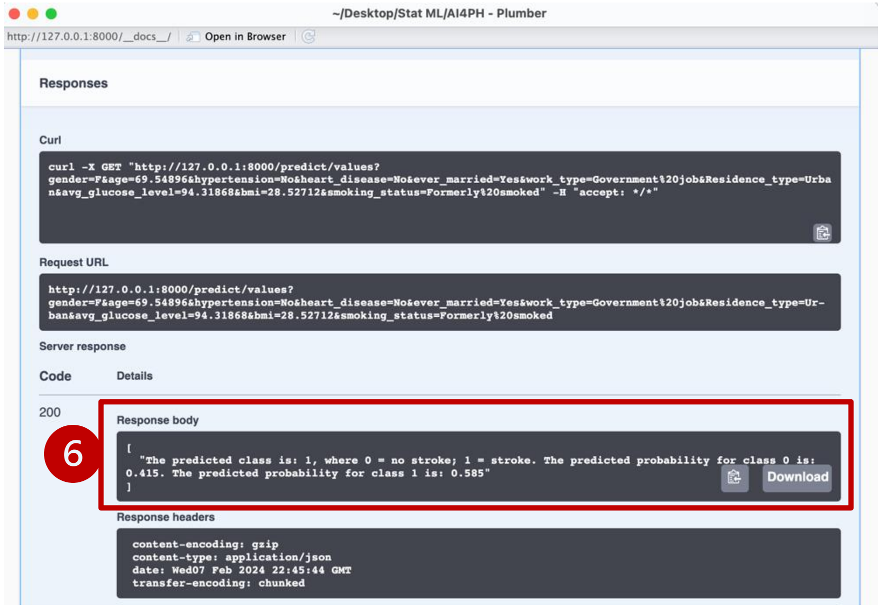
Name	Description
gender * required string *(query)*	Your gender, allowed value: F; M gender - Your gender, allowed value: F; M



4

Name	Description
Residence_type * required string *(query)*	The type of your residence, allowed range: Rural; Urban Urban
avg_glucose_level * required string *(query)*	Average glucose level, allowed range: 0-310 94.31868
bmi * required string *(query)*	Body mass index, allowed range: 9-73 28.52712
smoking_status * required string *(query)*	Smoking, allowed value: Never smoked; Formerly smoked; Currently smokes Formerly smoked

5



Responses

Curl

```
curl -X GET "http://127.0.0.1:8000/predict/values?
gender=F&age=69.54896&hypertension=No&heart_disease=No&ever_married=Yes&work_type=Government%20job&Residence_type=Urba
n&avg_glucose_level=94.31868&bmi=28.52712&smoking_status=Formerly%20smoked" -H "accept: */*"
```

Request URL

```
http://127.0.0.1:8000/predict/values?
gender=F&age=69.54896&hypertension=No&heart_disease=No&ever_married=Yes&work_type=Government%20job&Residence_type=Ur-
ban&avg_glucose_level=94.31868&bmi=28.52712&smoking_status=Formerly%20smoked
```

Server response

Code	Details
200	Response body

6

```
[
  "The predicted class is: 1, where 0 = no stroke; 1 = stroke. The predicted probability for class 0 is:
0.415. The predicted probability for class 1 is: 0.585"
]
```

Response headers

```
content-encoding: gzip
content-type: application/json
date: Wed07 Feb 2024 22:45:44 GMT
transfer-encoding: chunked
```

Demo

Please copy and paste the below examples and see if the returned results are as expected:

Example 1

```
gender              F
age                 69.54896
hypertension        No
heart_disease       No
ever_married        Yes
work_type           Government job
Residence_type      Urban
avg_glucose_level   94.31868
bmi                 28.52712
smoking_status      Formerly smoked
==============
The observed class: 1
==============
The expected predicted class: 1
The expected probability for class 0: 0.415
The expected probability for class 1: 0.585
```

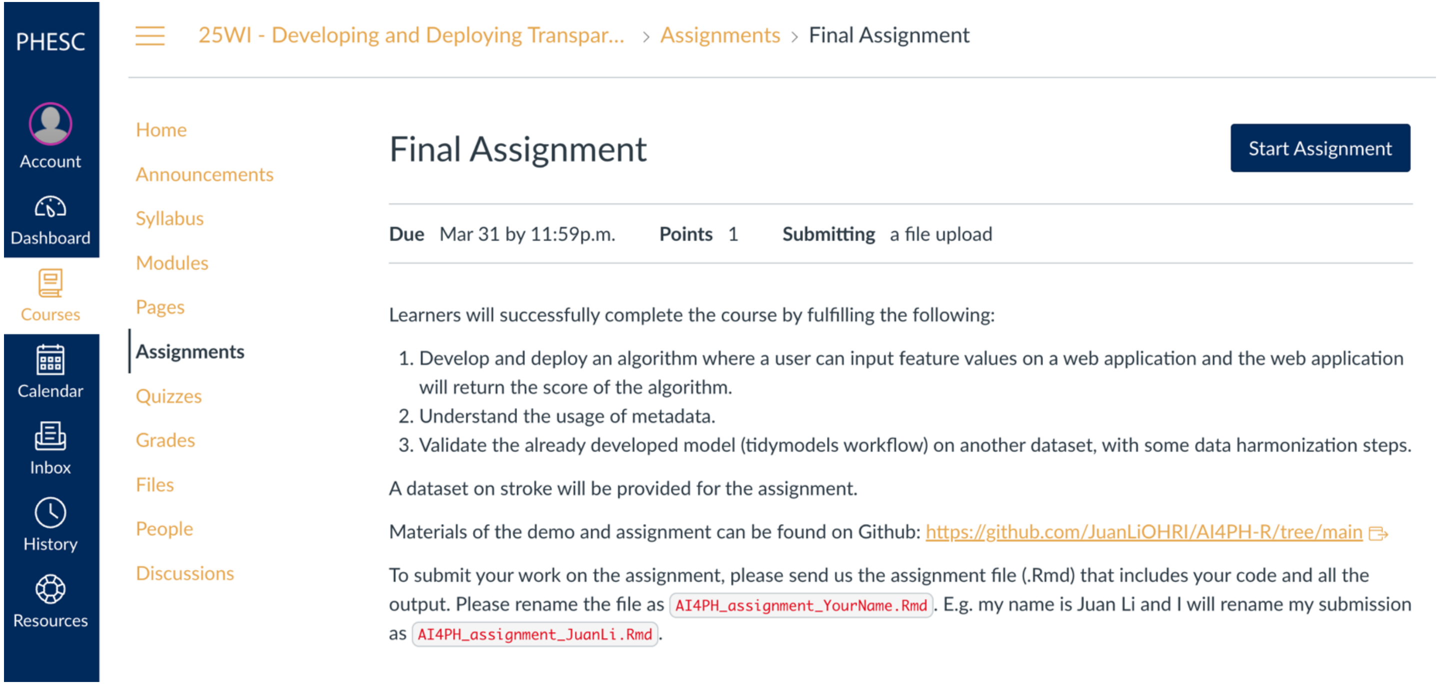
Example 2

```
gender              F
age                 48.50831
hypertension        No
heart_disease       No
ever_married        Yes
work_type           Private job
Residence_type      Rural
avg_glucose_level   61.57483
bmi                 27.60176
smoking_status      Never smoked
==============
The observed class: 0
==============
The expected predicted class: 0
The expected probability for class 0: 0.742
The expected probability for class 1: 0.258
```

Example 3

```
gender              M
age                 68.96269
hypertension        No
heart_disease       No
ever_married        Yes
work_type           Self-employed
Residence_type      Urban
avg_glucose_level   77.88883
bmi                 25.83863
smoking_status      Formerly smoked
==============
The observed class: 1
==============
The expected predicted class: 1
The expected probability for class 0: 0.371
The expected probability for class 1: 0.629
```

The final assessment – How to submit



Final Assignment

Due Mar 31 by 11:59p.m. **Points** 1 **Submitting** a file upload

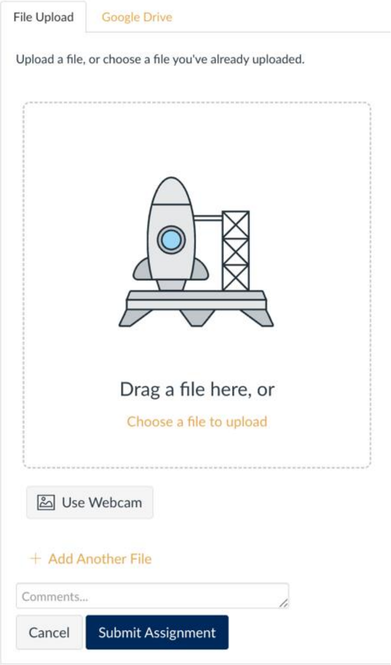
Learners will successfully complete the course by fulfilling the following:

1. Develop and deploy an algorithm where a user can input feature values on a web application and the web application will return the score of the algorithm.
2. Understand the usage of metadata.
3. Validate the already developed model (tidymodels workflow) on another dataset, with some data harmonization steps.

A dataset on stroke will be provided for the assignment.

Materials of the demo and assignment can be found on Github: https://github.com/JuanLiOHRI/AI4PH-R/tree/main

To submit your work on the assignment, please send us the assignment file (.Rmd) that includes your code and all the output. Please rename the file as `AI4PH_assignment_YourName.Rmd`. E.g. my name is Juan Li and I will rename my submission as `AI4PH_assignment_JuanLi.Rmd`.



Upload a file, or choose a file you've already uploaded.

The final assessment

In this assignment, you will validate the model we developed in class using a different dataset: `valid_data.rds`. You will run into issues using this dataset as it is because this is a raw dataset without data harmonization, which means that some variables in this dataset are different from the harmonized dataset we used to train and evaluate the model. Your job here is to harmonize the validation data so that it's in the same format as the example data we used in class. You can refer to `train_data_variables.csv` to see the format in the harmonized train data.

All materials including this slide deck and the demo code can be found on Github: https://github.com/JuanLiOHRI/AI4PH-R/tree/main

THANK YOU

TIDYMODELS – other packages that might be useful

Some R objects become inconveniently large when saved to disk. The butcher package can reduce the size of those objects by removing the sub-components.

e.g. logistic regression model object that contains the training data. Especially in public health, where the sample size can be huge.

The tidyposterior package enables users to make formal statistical comparisons between models using resampling and Bayesian methods.

infer is a high-level API for tidyverse-friendly statistical inference.

The corrr package has tidy interfaces for working with correlation matrices.

https://www.tidymodels.org/packages/