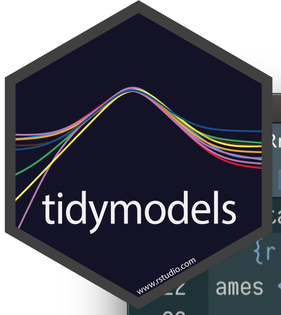
Practical plumber Patterns

James Blair
Solutions Engineer @ RStudio

Meet Susan



ppp - master - RStudio Source Editor

Rmd

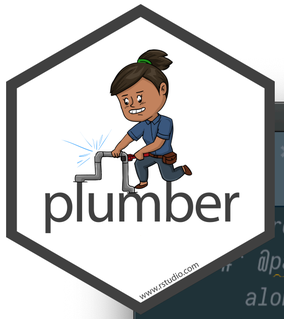
Preview | Insert | Run

```r
ta
```{r data}
ames <- make_ames()

data_split <- initial_split(ames, strata = "Sale_Price", p = 0.75)
ames_train <- training(data_split)
ames_test <- testing(data_split)
```

Pre-Processing
```{r}
ames_recipe <- recipe(
  Sale_Price ~ Longitude + Latitude + Lot_Area + Neighborhood + Year_Sold,
  data = ames_train
) %>%
  step_other(Neighborhood) %>%
  step_dummy(all_nominal()) %>%
  step_center(all_predictors()) %>%
  step_scale(all_predictors()) %>%
  step_log(Sale_Price, base = 10) %>%
  # estimate the means and standard deviations
  prep(training = ames_train, retain = TRUE)
```

Model
```{r}
ames_model <- linear_reg(penalty = 0.001, mixture = 0.5) %>%
  set_engine("glmnet") %>%
  fit(Sale_Price ~ ., data = juice(ames_recipe))
```
```

8:20 | # Packages | R Markdown

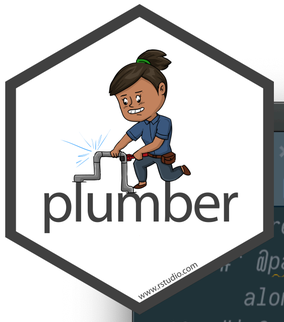


ppp - master - RStudio Source Editor

Run API

```r
redict sale price of new data
#* @param all Whether or not to return the pre-processed input data (TRUE)
  along with the predictions, or just the predictions (FALSE)
#* @param log Should the predicted values remain in log10 (TRUE) or be
  converted back to actual prices (FALSE)
#* @post /predict
function(req, res, all = FALSE, log = FALSE) {
  # Check to see if input data can be serialized from JSON
  new_data <- tryCatch(jsonlite::parse_json(req$postBody, simplifyVector =
    TRUE),
                       error = function(e) NULL)
  if (is.null(new_data)) {
    res$status <- 400
    return(list(error = "Invalid data submitted"))
  }

  # Convert character vectors to factors in new_data
  new_data <- dplyr::mutate_if(new_data, is.character, as.factor)

  # Apply recipe to new_data
  model_data <- recipes::bake(recipe, new_data)

  # Generate predictions using the model and model_data
  predictions <- predict(model, model_data)

  # If !log, transform predicted values back from log10
  if (!as.logical(log)) {
    predictions[[".pred"]] <- predictions[[".pred"]] ^ 10
  }
```

10:1 | # Load global objects | R Script

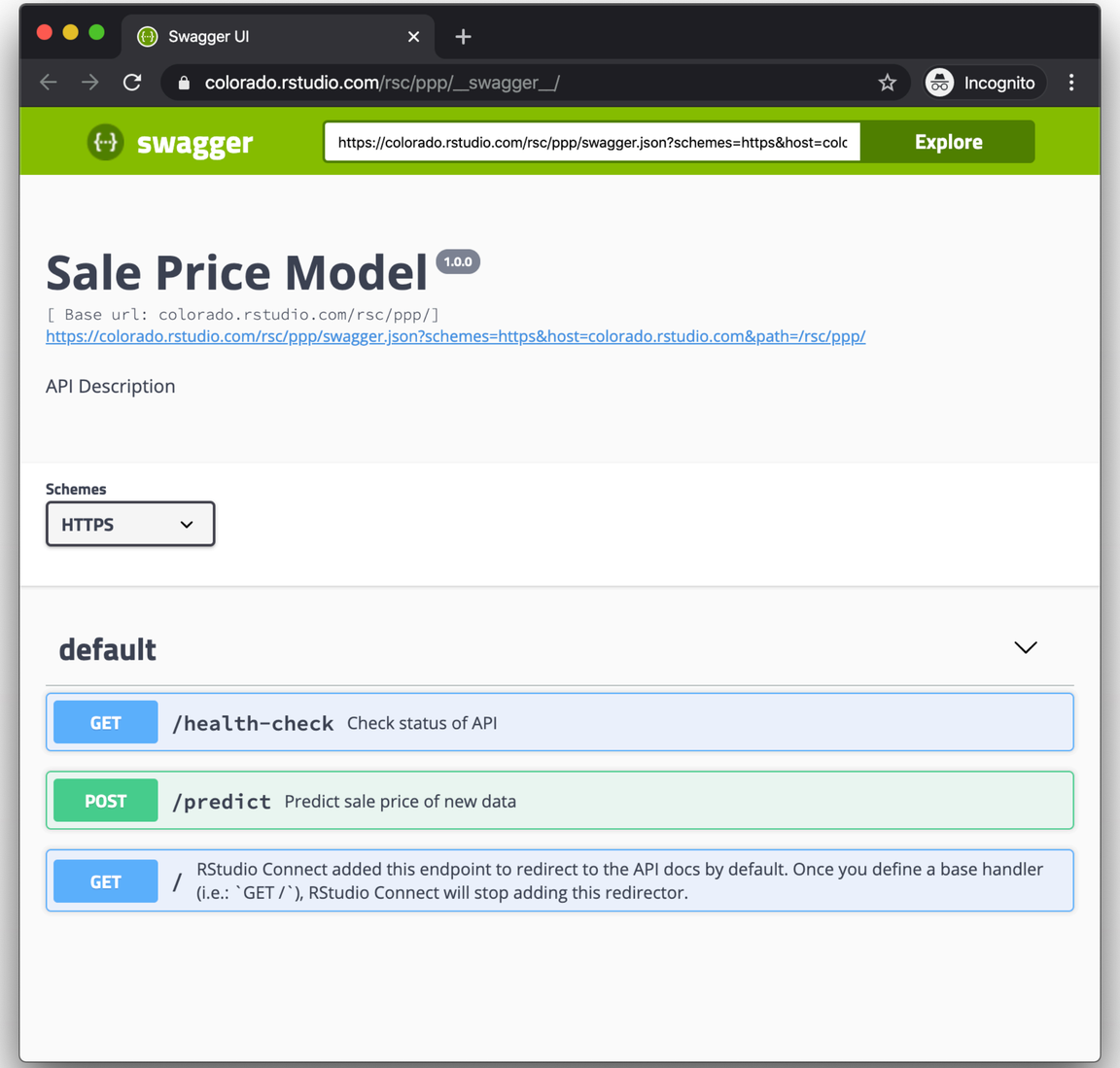


ppp - master - RStudio Source Editor

Run API

```
redict sale price of new data
* @param all Whether or not to return the pre-processed input data (TRUE)
  along with the predictions, or just the predictions (FALSE)
#* @param log Should the predicted values remain in log10 (TRUE) or be
  converted back to actual prices (FALSE)
#* @post /predict
function(req, res, all = FALSE, log = FALSE) {
  # Check to see if input data can be serialized from JSON
  new_data <- tryCatch(jsonlite::parse_json(req$postBody, simplifyVector =
    TRUE),
                       error = function(e) NULL)
  if (is.null(new_data)) {
    res$status <- 400
    return(list(error = "Invalid data submitted"))
  }

  # Convert character vectors to factors in new_data
  new_data <- dplyr::mutate_if(new_data, is.character, as.factor)

  # Apply recipe to new_data
  model_data <- recipes::bake(recipe, new_data)

  # Generate predictions using the model and model_data
  predictions <- predict(model, model_data)

  # If !log, transform predicted values back from log10
  if (!as.logical(log)) {
    predictions[[".pred"]] <- predictions[[".pred"]] ^ 10
  }
```

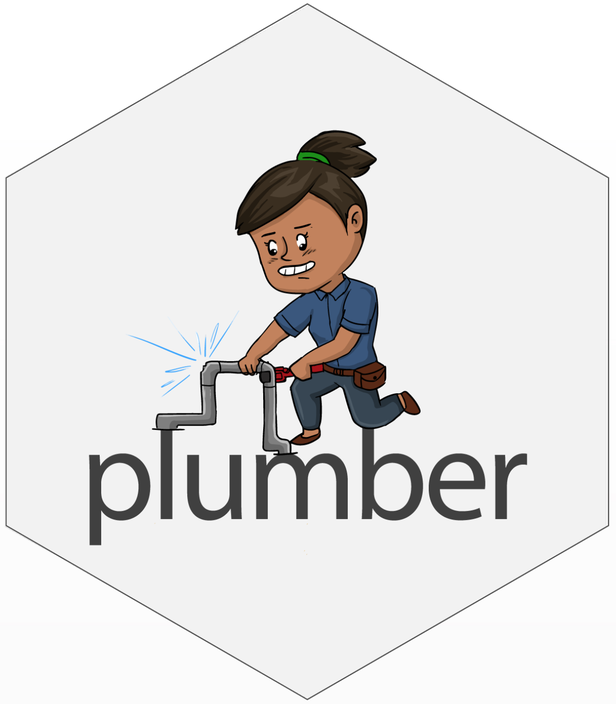
10:1 | Load global objects | R Script



Test

Packages are the fundamental units of reproducible R code.

- Hadley Wickham


plumber


BOX TEXT
Text placeholder. Replace this text. Dummy text. Default
Random Text: Default
Default Text:

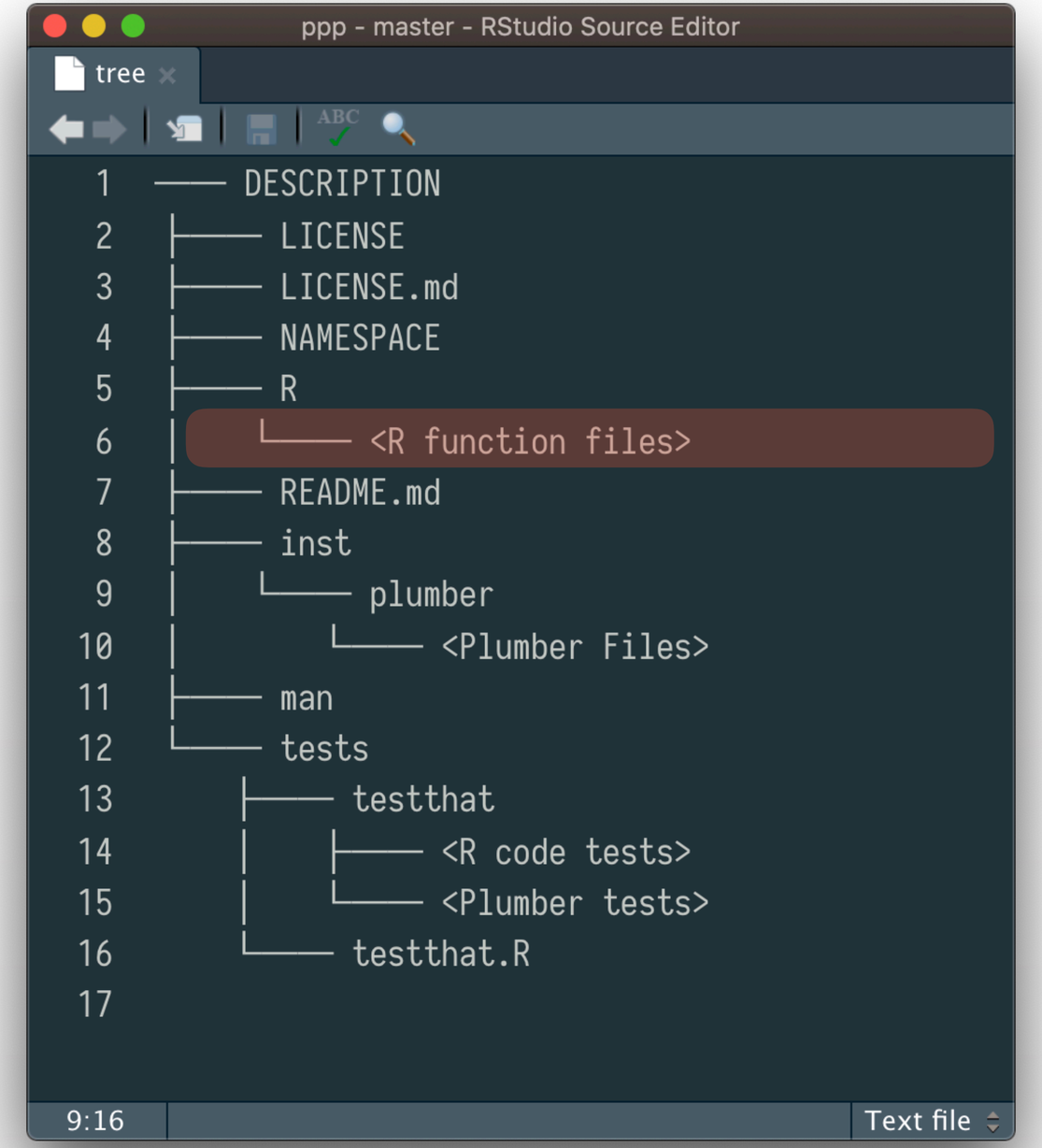

BOX TEXT
Text placeholder. Replace this text. Dummy text.
Random Text: Default
Default Text: Default

ppp - master - RStudio Source Editor

tree

```
1  ─── DESCRIPTION
2  ├─── LICENSE
3  ├─── LICENSE.md
4  ├─── NAMESPACE
5  ├─── R
6  │    └─── <R function files>
7  ├─── README.md
8  ├─── inst
9  │    └─── plumber
10 │         └─── <Plumber Files>
11 ├─── man
12 └─── tests
13      ├─── testthat
14      │    ├─── <R code tests>
15      │    └─── <Plumber tests>
16      └─── testthat.R
17
```

9:16 Text file


ppp - master - RStudio Source Editor
tree
1 ─── DESCRIPTION
2 ├─── LICENSE
3 ├─── LICENSE.md
4 ├─── NAMESPACE
5 ├─── R
6 │ └─── <R function files>
7 ├─── README.md
8 ├─── inst
9 │ └─── plumber
10 │ └─── <Plumber Files>
11 ├─── man
12 └─── tests
13 ├─── testthat
14 │ ├─── <R code tests>
15 │ └─── <Plumber tests>
16 └─── testthat.R
17
9:16
Text file

ppp - master - RStudio Source Editor

tree

```
1  ─── DESCRIPTION
2  ├─── LICENSE
3  ├─── LICENSE.md
4  ├─── NAMESPACE
5  ├─── R
6  │   └─── <R function files>
7  ├─── README.md
8  ├─── inst
9  │   └─── plumber
10 │       └─── <Plumber Files>
11 ├─── man
12 └─── tests
13     ├─── testthat
14     │   ├─── <R code tests>
15     │   └─── <Plumber tests>
16     └─── testthat.R
17
```

9:16 Text file

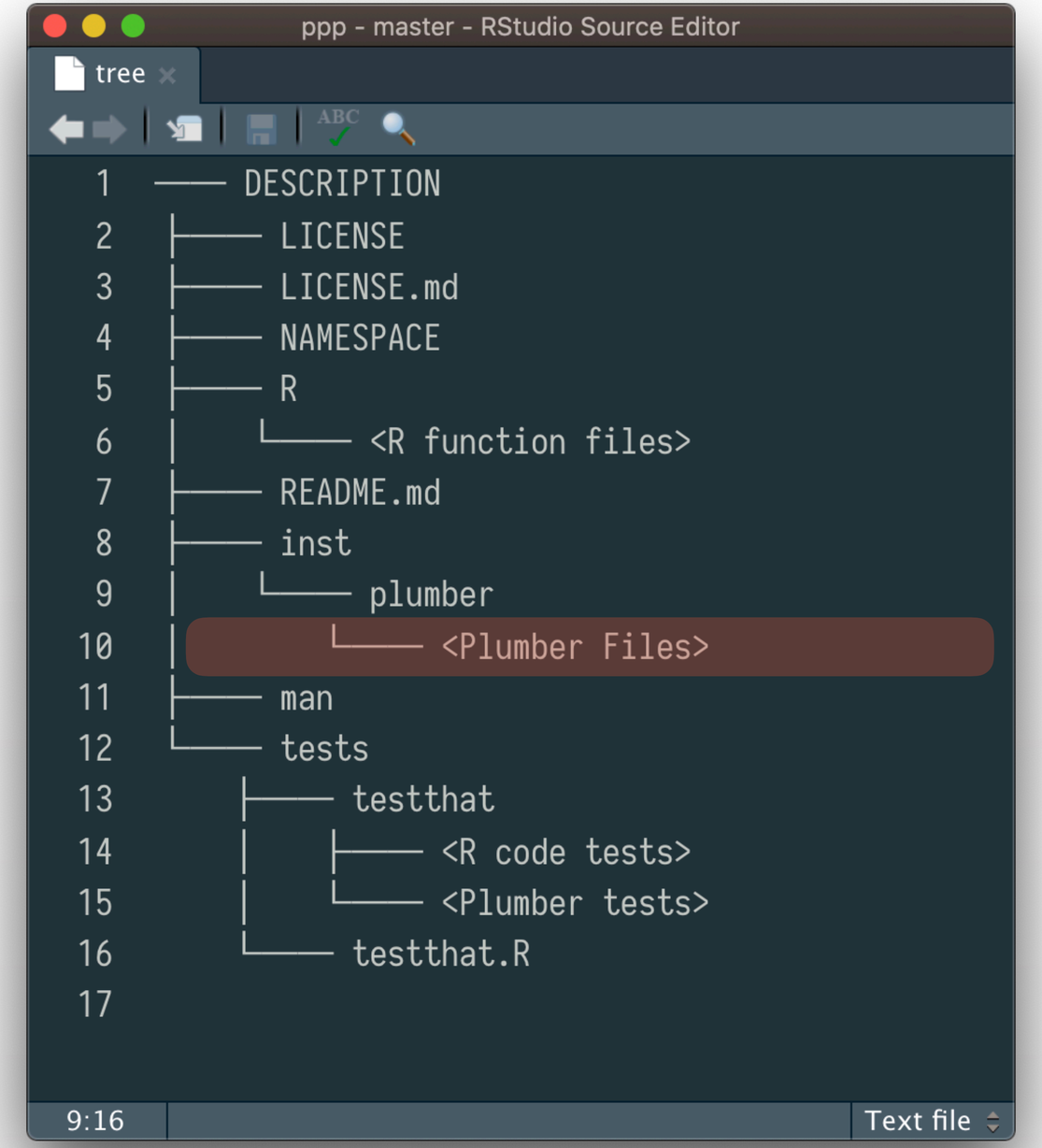
ppp - slides - RStudio Source Editor

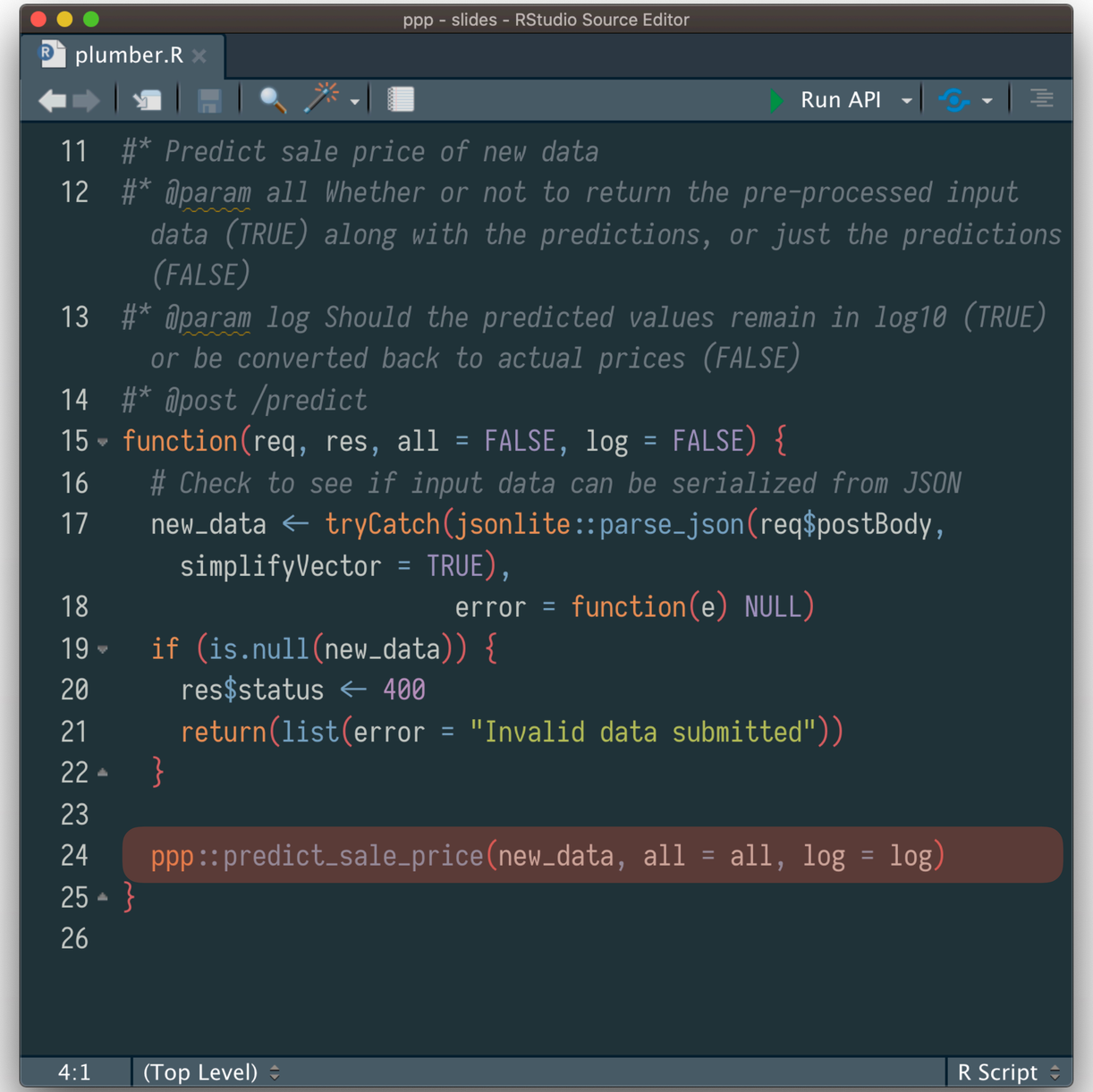
predict.R

Source on Save | Run | Source

```r
#' @export
predict_sale_price <- function(new_data, all = TRUE, log = FALSE)
  {
  # Load saved model and recipe
  model <- readRDS(system.file("plumber", "model", "ames-model
    .rds", package = "ppp"))
  recipe <- readRDS(system.file("plumber", "model", "ames-recipe
    .rds", package = "ppp"))

  # Convert character vectors to factors in new_data
  new_data <- dplyr::mutate_if(new_data, is.character, as.factor)

  # Apply recipe to new_data
  model_data <- recipes::bake(recipe, new_data)

  # Generate predictions using the model and model_data
  predictions <- parsnip::predict.model_fit(model, model_data)

  # If !log, transform predicted values back from log10
  if (!as.logical(log)) {
    predictions[[".pred"]] <- 10 ^ predictions[[".pred"]]
  }
```

64:17 (Top Level) R Script


ppp - master - RStudio Source Editor
tree
1 ─── DESCRIPTION
2 ├─── LICENSE
3 ├─── LICENSE.md
4 ├─── NAMESPACE
5 ├─── R
6 │ └─── <R function files>
7 ├─── README.md
8 ├─── inst
9 │ └─── plumber
10 │ └─── <Plumber Files>
11 ├─── man
12 └─── tests
13 ├─── testthat
14 │ ├─── <R code tests>
15 │ └─── <Plumber tests>
16 └─── testthat.R
17
9:16
Text file


ppp - slides - RStudio Source Editor
plumber.R
Run API
11 #* Predict sale price of new data
12 #* @param all Whether or not to return the pre-processed input data (TRUE) along with the predictions, or just the predictions (FALSE)
13 #* @param log Should the predicted values remain in log10 (TRUE) or be converted back to actual prices (FALSE)
14 #* @post /predict
15 function(req, res, all = FALSE, log = FALSE) {
16 # Check to see if input data can be serialized from JSON
17 new_data <- tryCatch(jsonlite::parse_json(req$postBody,
simplifyVector = TRUE),
18 error = function(e) NULL)
19 if (is.null(new_data)) {
20 res$status <- 400
21 return(list(error = "Invalid data submitted"))
22 }
23
24 ppp::predict_sale_price(new_data, all = all, log = log)
25 }
26
4:1
(Top Level)
R Script

ppp - master - RStudio Source Editor

tree

```
1  ─── DESCRIPTION
2  ├─── LICENSE
3  ├─── LICENSE.md
4  ├─── NAMESPACE
5  ├─── R
6  │    └─── <R function files>
7  ├─── README.md
8  ├─── inst
9  │    └─── plumber
10 │         └─── <Plumber Files>
11 ├─── man
12 └─── tests
13      ├─── testthat
14      │    ├─── <R code tests>
15      │    └─── <Plumber tests>
16      └─── testthat.R
17
```

9:16 Text file

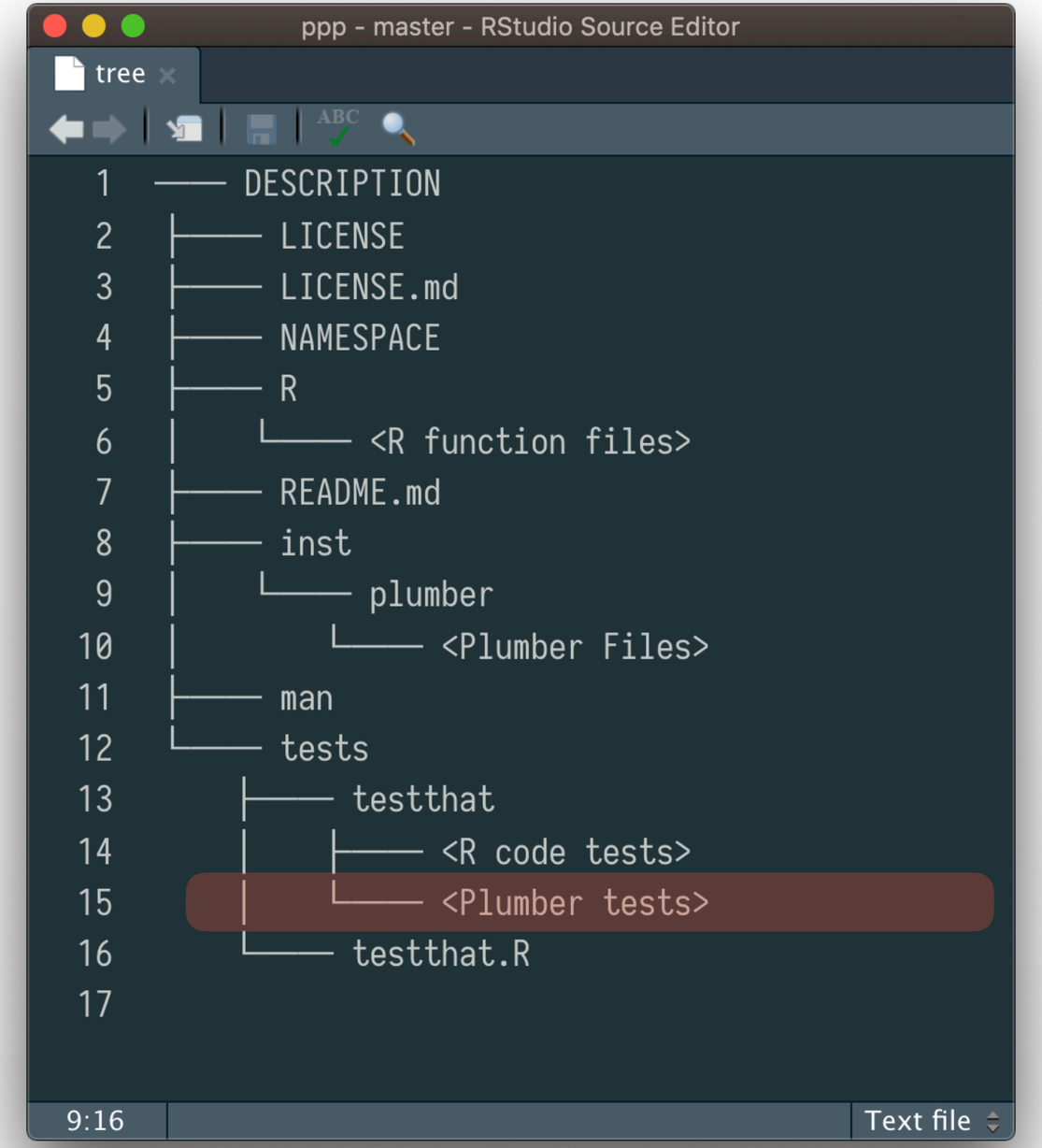
ppp - slides - RStudio Source Editor

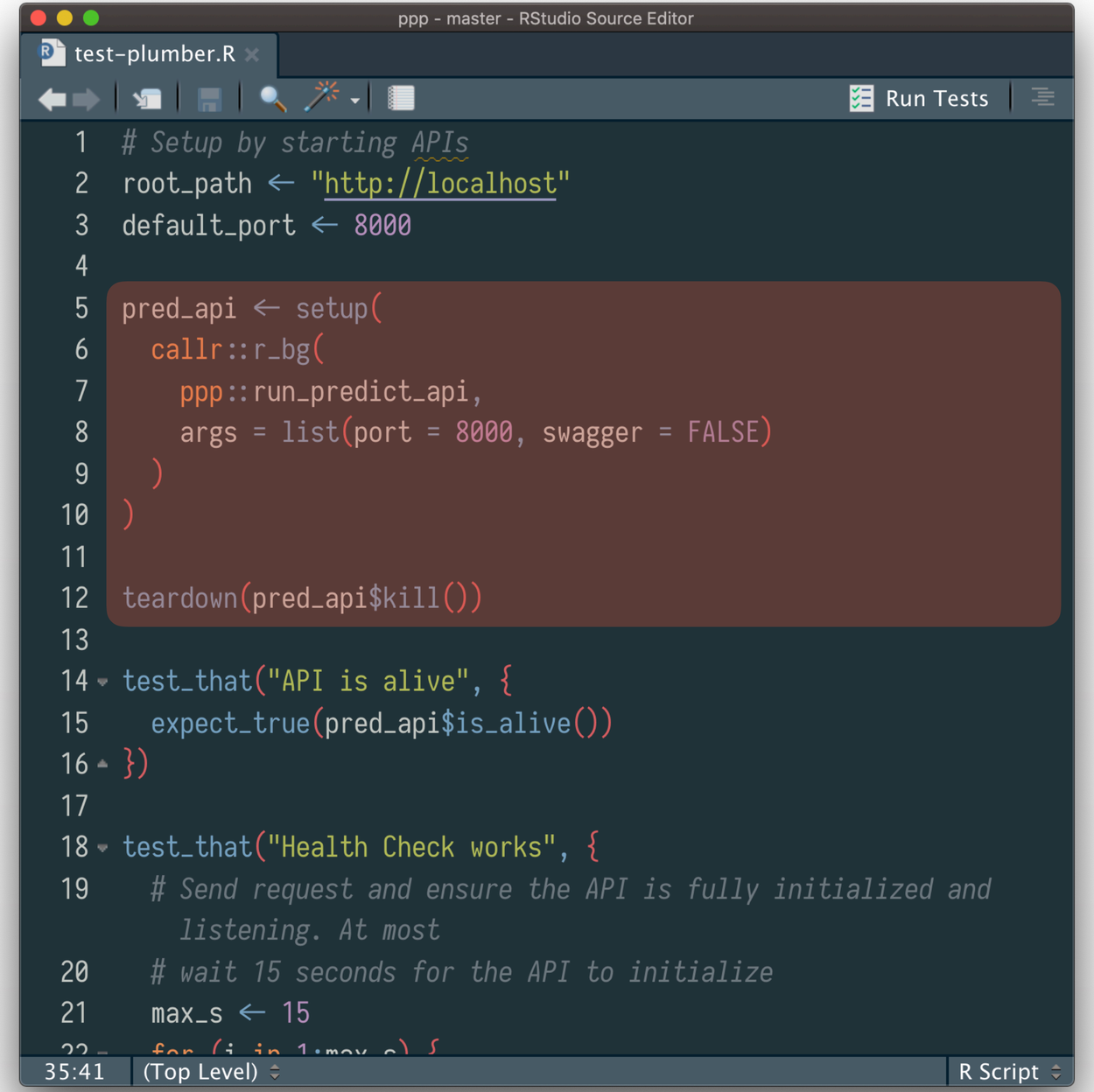
test-predict.R

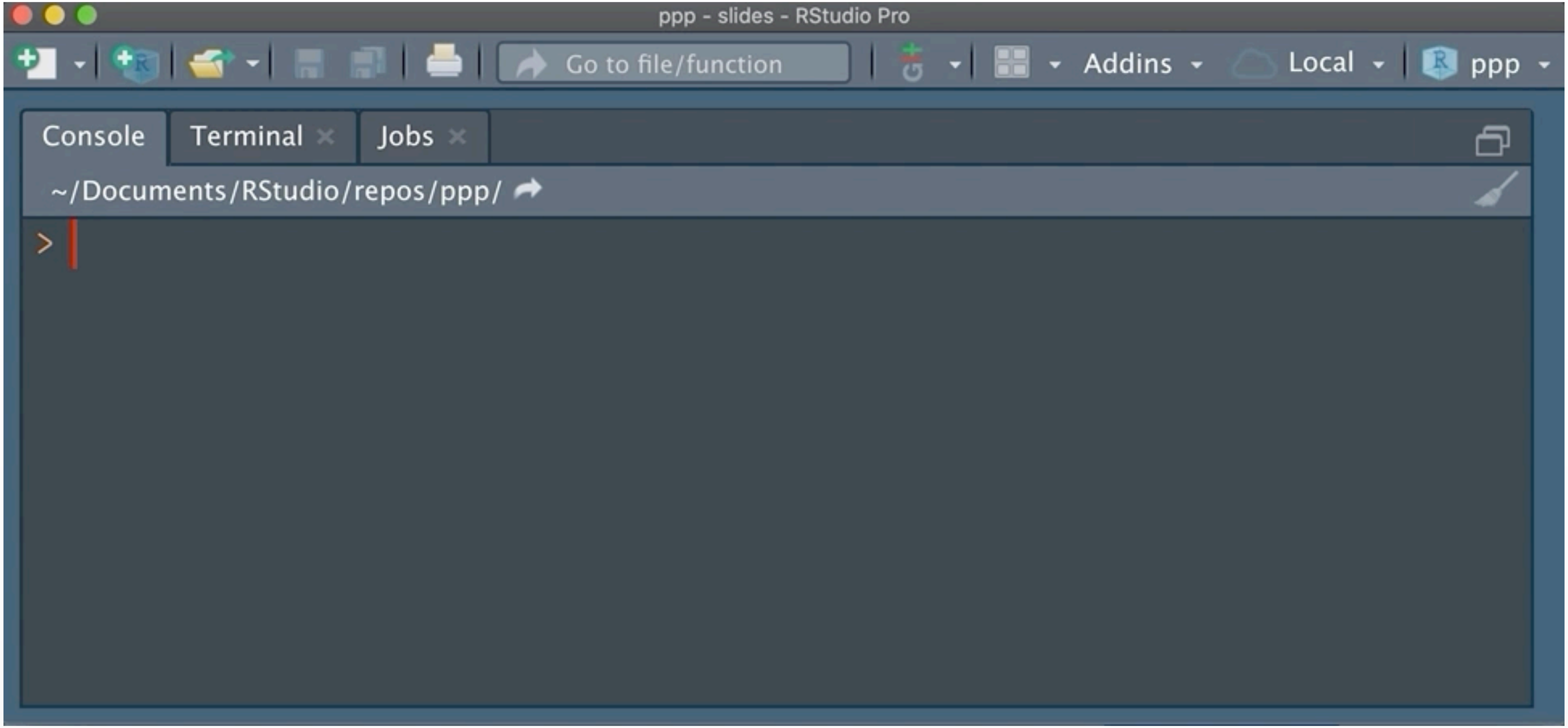
Run Tests

```
8  test_that("predict works", {
9    v_out <- predict_sale_price(test_data, all = FALSE, log = FALSE)
10   vl_out <- predict_sale_price(test_data, all = FALSE, log = TRUE)
11   d_out <- predict_sale_price(test_data, all = TRUE, log = FALSE)
12   dl_out <- predict_sale_price(test_data, all = TRUE, log = TRUE)
13
14   # Check return types
15   expect_type(v_out, "double")
16   expect_type(d_out, "list")
17
18   # Check return values
19   expect_gte(min(v_out), 0)
20   expect_gte(min(vl_out), 0)
21   expect_lt(max(vl_out), 10)
22   expect_equal(log10(v_out), vl_out)
23   expect_equal(10 ^ vl_out, v_out)
24
25   # Bad data
26   expect_error(predict_sale_price())
27   expect_error(predict_sale_price(mtcars))
28   expect_warning(predict_sale_price(bad_data))
29 })
```

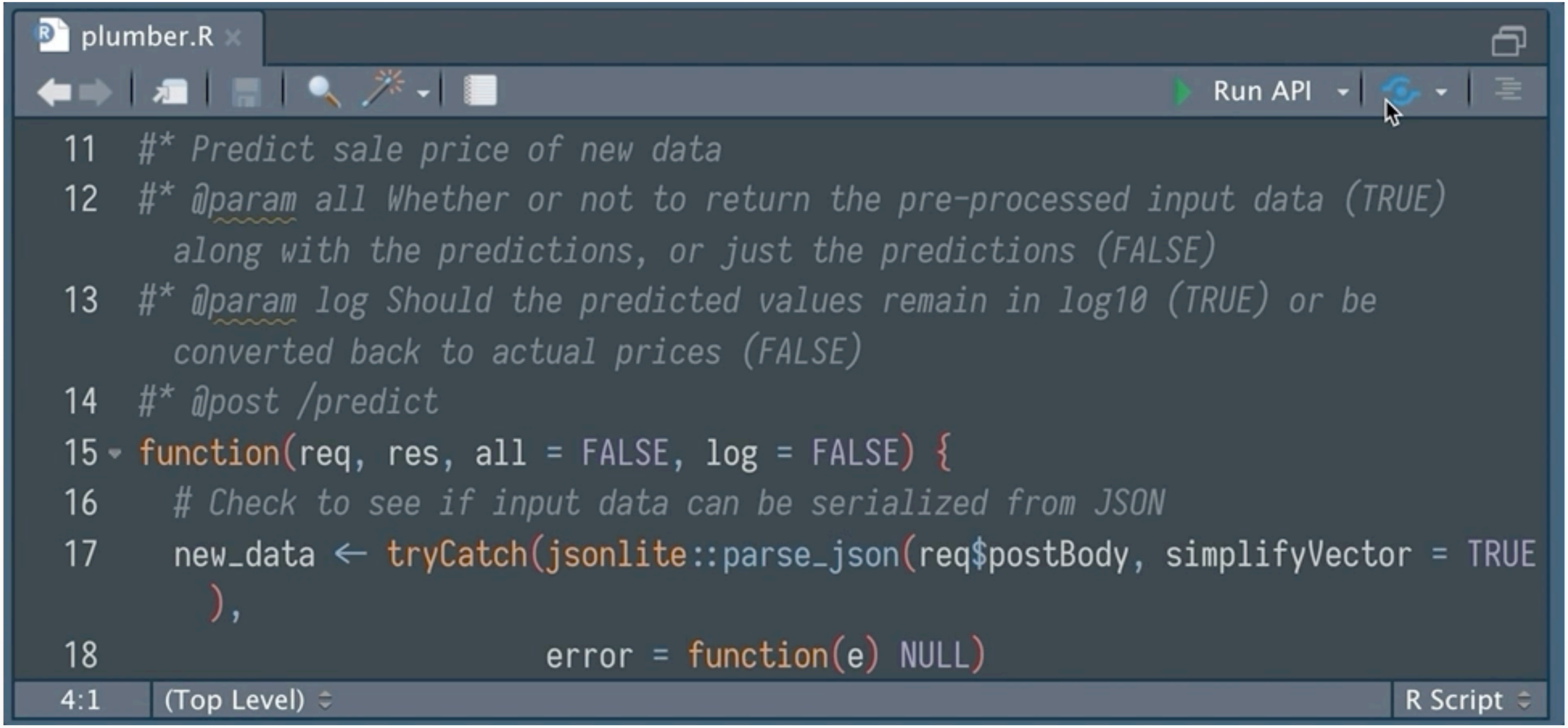
3:26 rand_nas(v, p) R Script

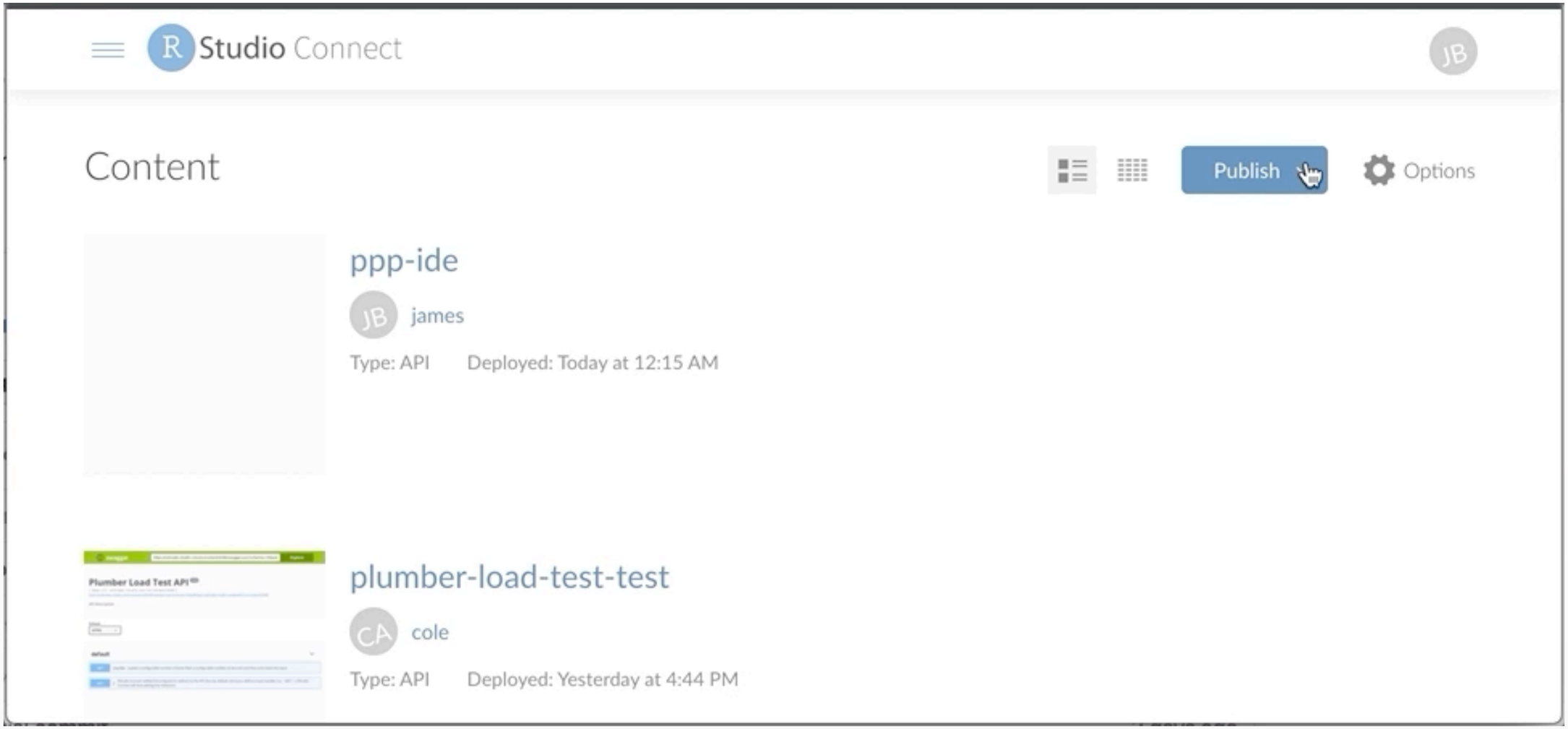

ppp - master - RStudio Source Editor
tree
1 ─── DESCRIPTION
2 ├─── LICENSE
3 ├─── LICENSE.md
4 ├─── NAMESPACE
5 ├─── R
6 │ └─── <R function files>
7 ├─── README.md
8 ├─── inst
9 │ └─── plumber
10 │ └─── <Plumber Files>
11 ├─── man
12 └─── tests
13 ├─── testthat
14 │ ├─── <R code tests>
15 │ └─── <Plumber tests>
16 └─── testthat.R
17
9:16
Text file

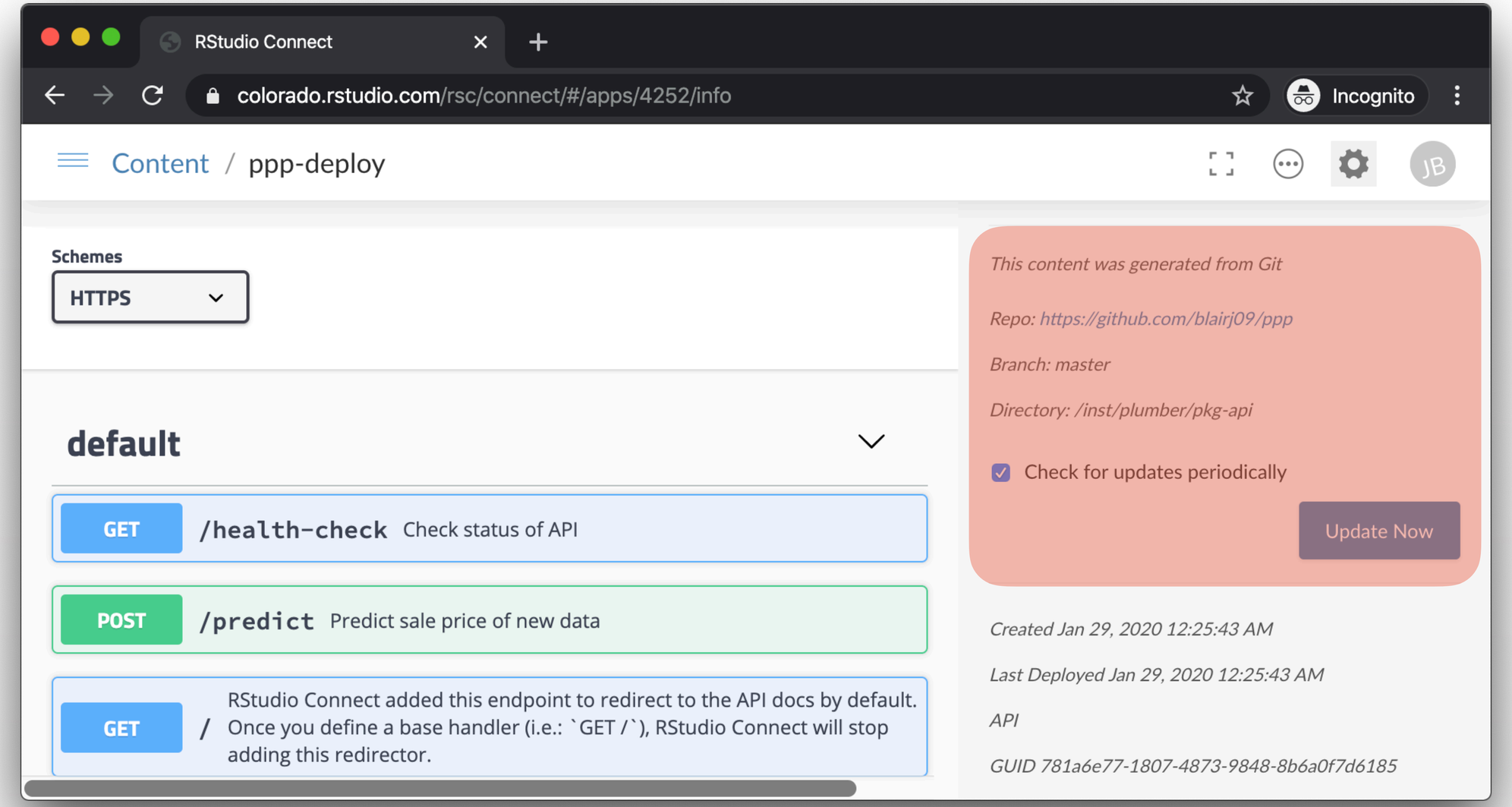

ppp - master - RStudio Source Editor
test-plumber.R
Run Tests
1 # Setup by starting APIs
2 root_path <- "http://localhost"
3 default_port <- 8000
4
5 pred_api <- setup(
6 callr::r_bg(
7 ppp::run_predict_api,
8 args = list(port = 8000, swagger = FALSE)
9)
10)
11
12 teardown(pred_api$kill())
13
14 test_that("API is alive", {
15 expect_true(pred_api$is_alive())
16 })
17
18 test_that("Health Check works", {
19 # Send request and ensure the API is fully initialized and
listening. At most
20 # wait 15 seconds for the API to initialize
21 max_s <- 15
35:41
(Top Level)
R Script


ppp - slides - RStudio Pro
Go to file/function
Addins
Local
ppp
Console
Terminal
Jobs
~/Documents/RStudio/repos/ppp/
>

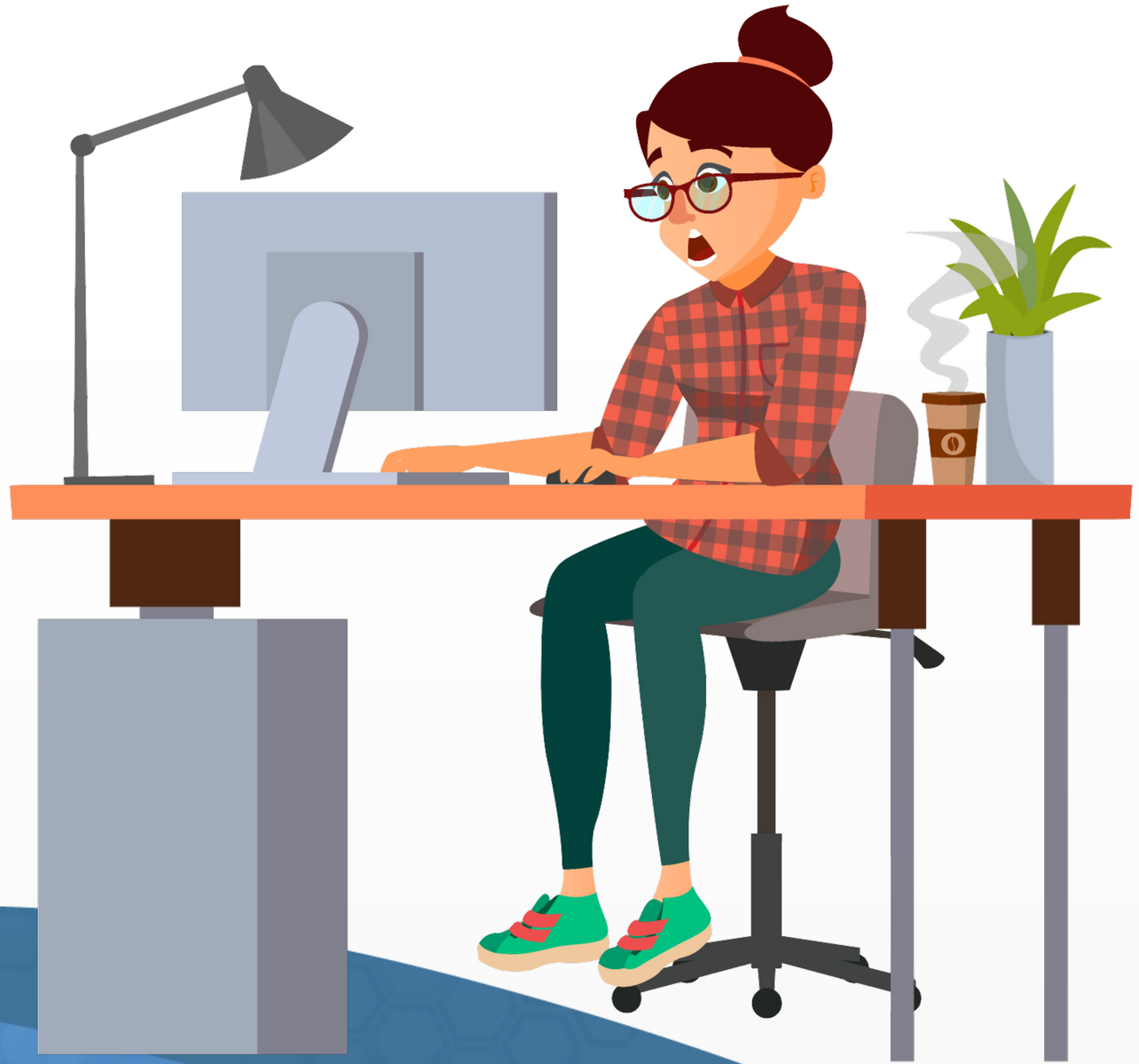
Deploy

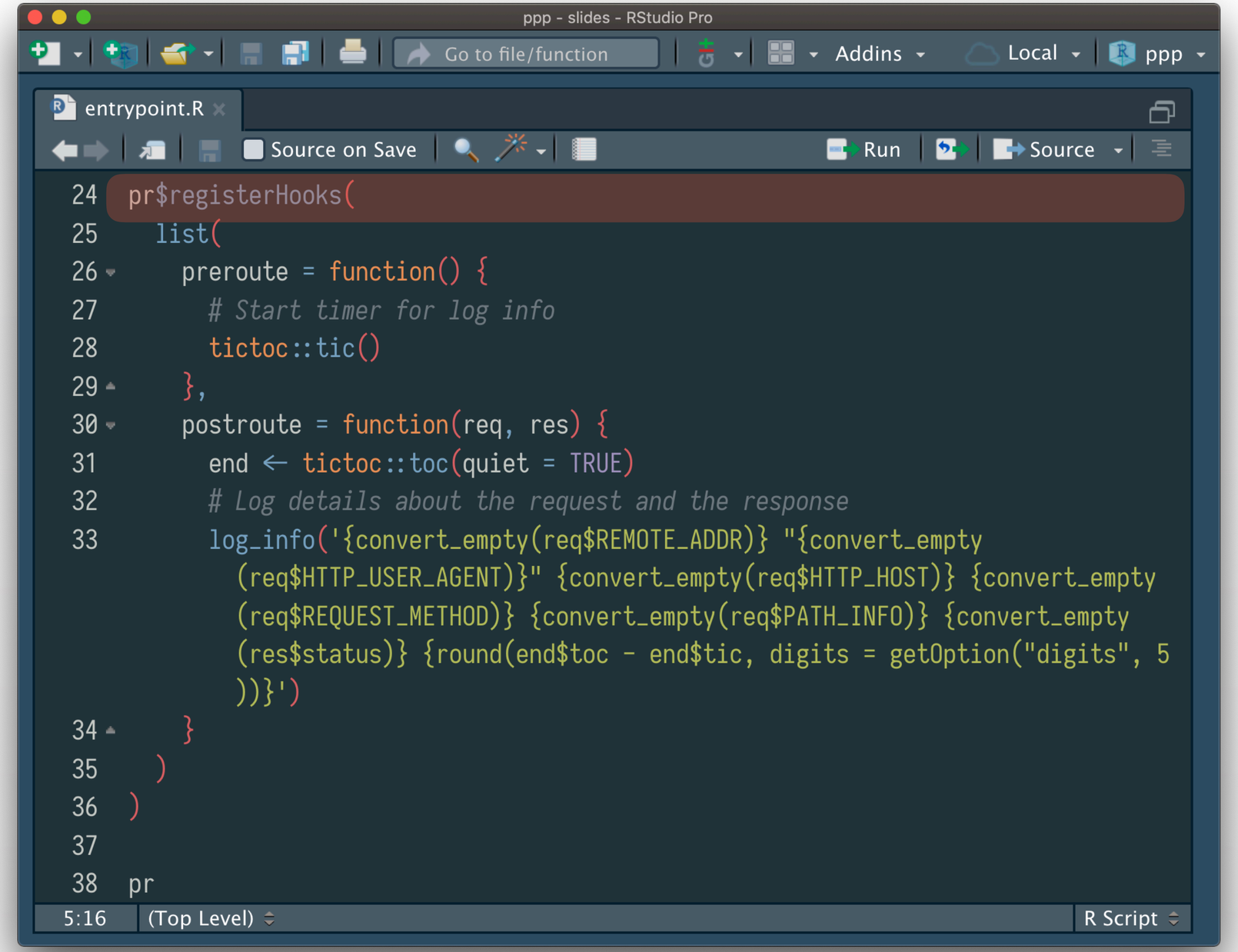

plumber.R
Run API
#* Predict sale price of new data
#* @param all Whether or not to return the pre-processed input data (TRUE) along with the predictions, or just the predictions (FALSE)
#* @param log Should the predicted values remain in log10 (TRUE) or be converted back to actual prices (FALSE)
#* @post /predict
function(req, res, all = FALSE, log = FALSE) {
# Check to see if input data can be serialized from JSON
new_data ← tryCatch(jsonlite::parse_json(req$postBody, simplifyVector = TRUE),
error = function(e) NULL)
4:1
(Top Level)
R Script


RStudio Connect
JB
Content
Publish
Options
ppp-ide
JB
james
Type: API
Deployed: Today at 12:15 AM
Plumber Load Test API
plumber-load-test-test
CA
cole
Type: API
Deployed: Yesterday at 4:44 PM

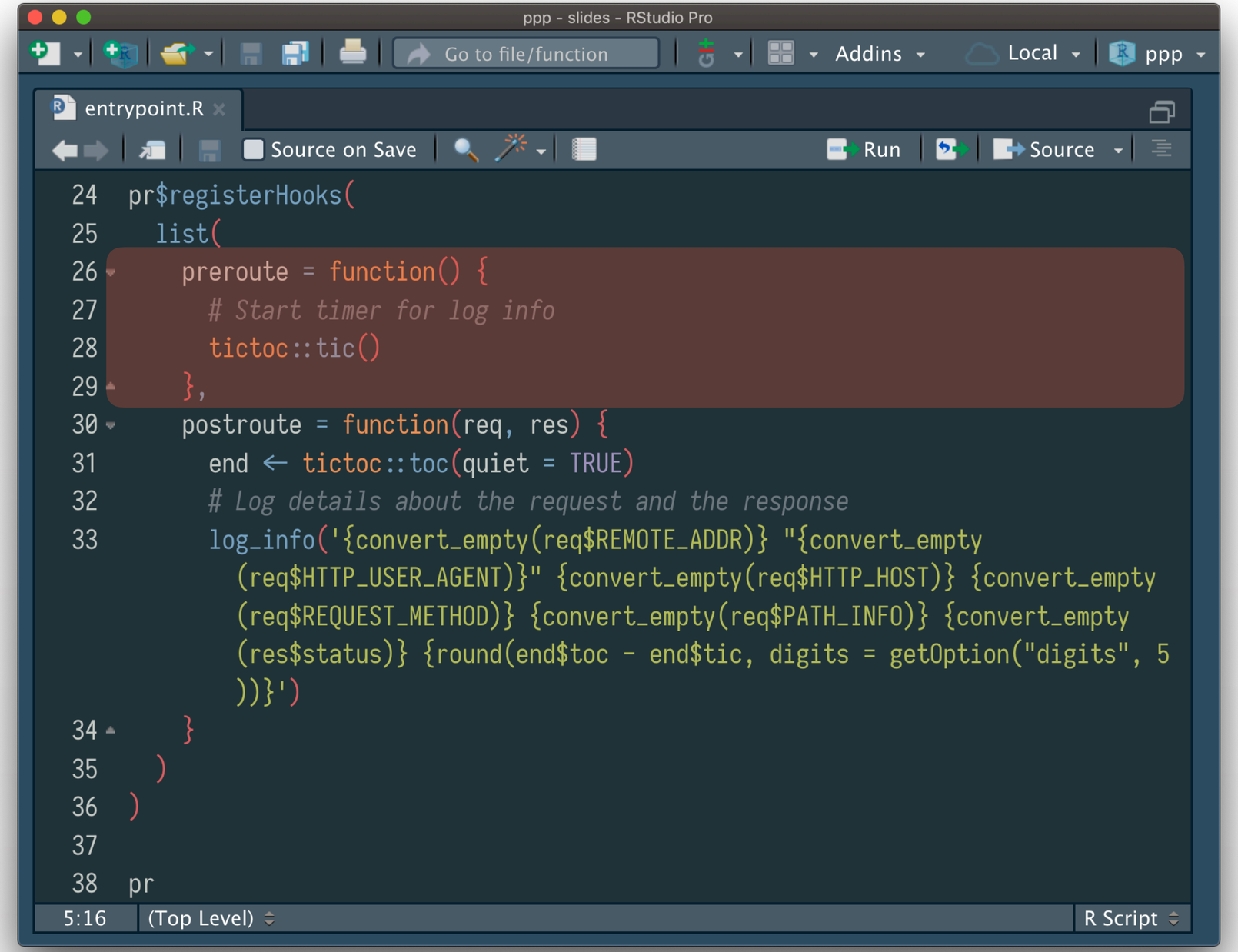

RStudio Connect
colorado.rstudio.com/rsc/connect/#/apps/4252/info
Incognito
Content / ppp-deploy
JB
Schemes
HTTPS
default
GET /health-check Check status of API
POST /predict Predict sale price of new data
GET / RStudio Connect added this endpoint to redirect to the API docs by default. Once you define a base handler (i.e.: `GET /`), RStudio Connect will stop adding this redirector.
This content was generated from Git
Repo: https://github.com/blairj09/ppp
Branch: master
Directory: /inst/plumber/pkg-api
Check for updates periodically
Update Now
Created Jan 29, 2020 12:25:43 AM
Last Deployed Jan 29, 2020 12:25:43 AM
API
GUID 781a6e77-1807-4873-9848-8b6a0f7d6185

Scale

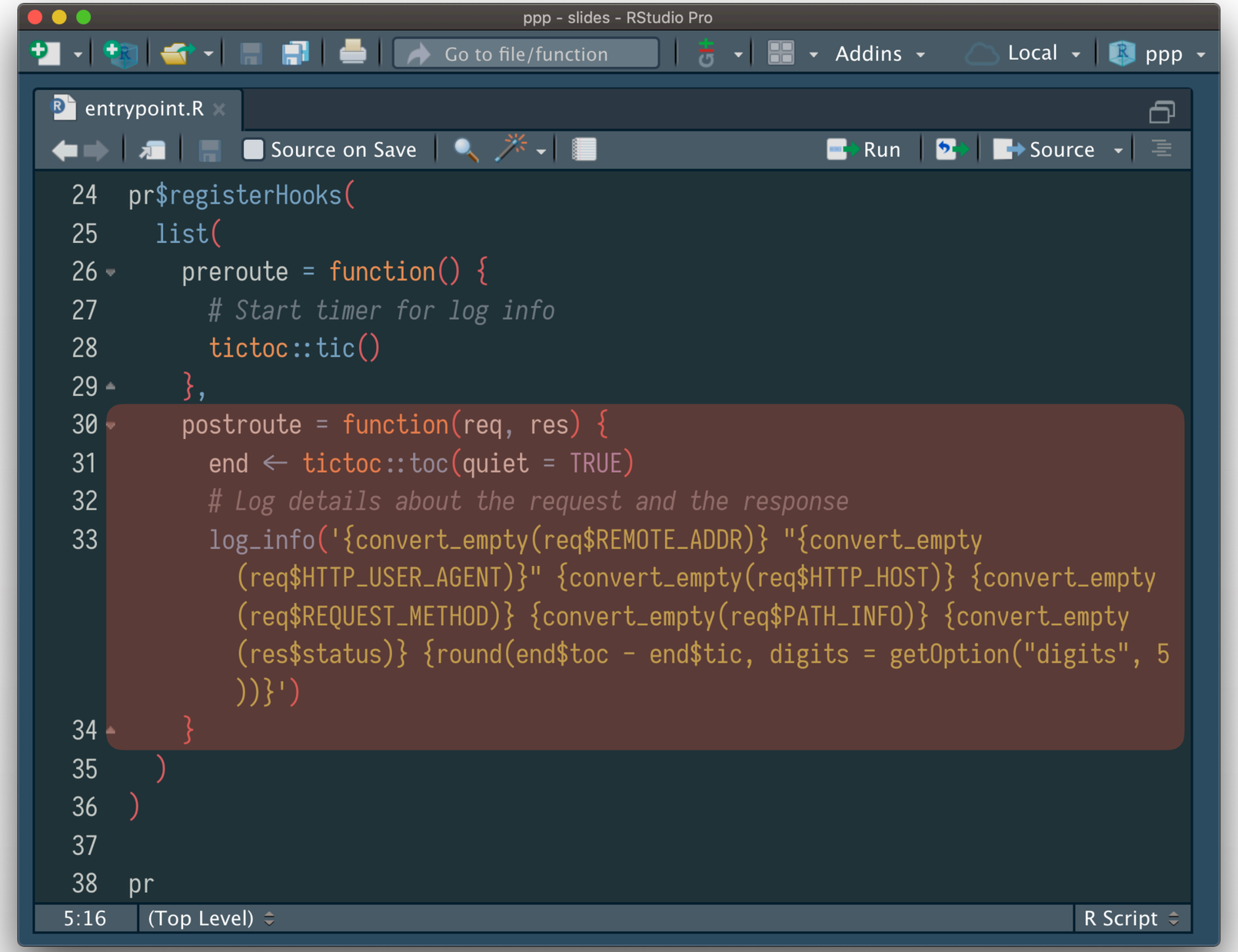




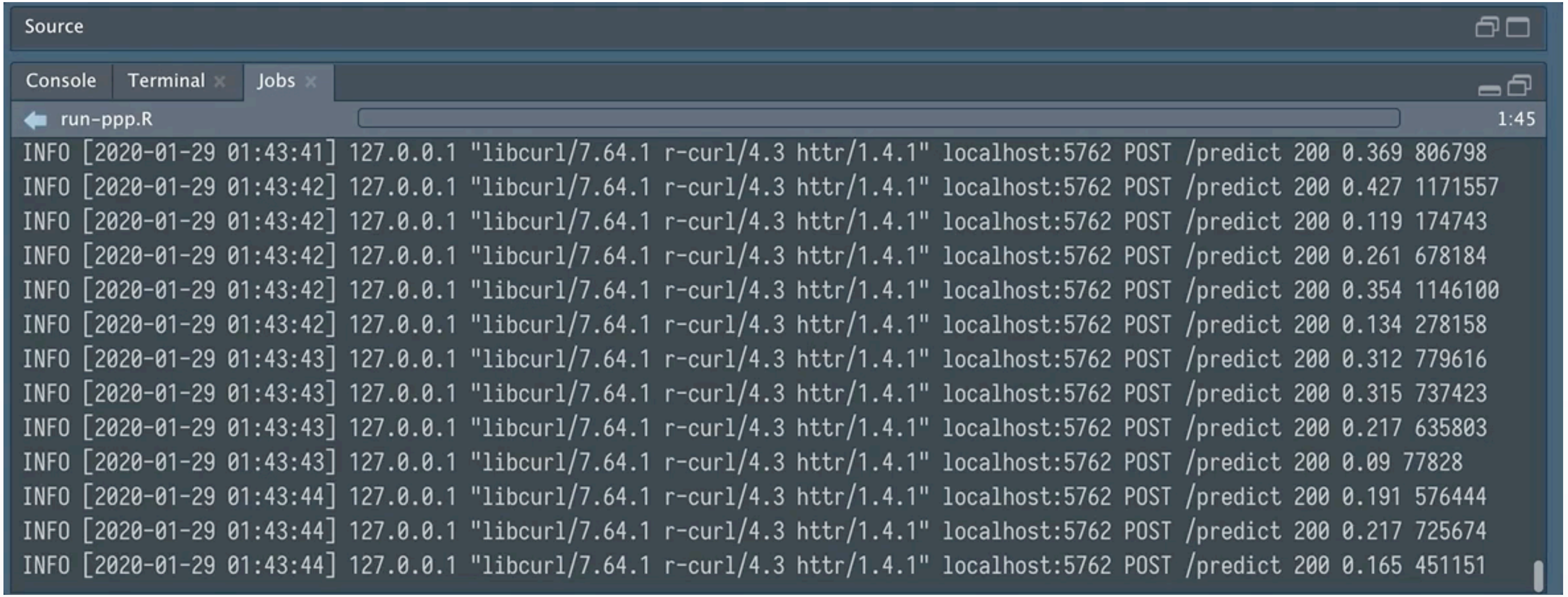
Define actions that take place on the Plumber router

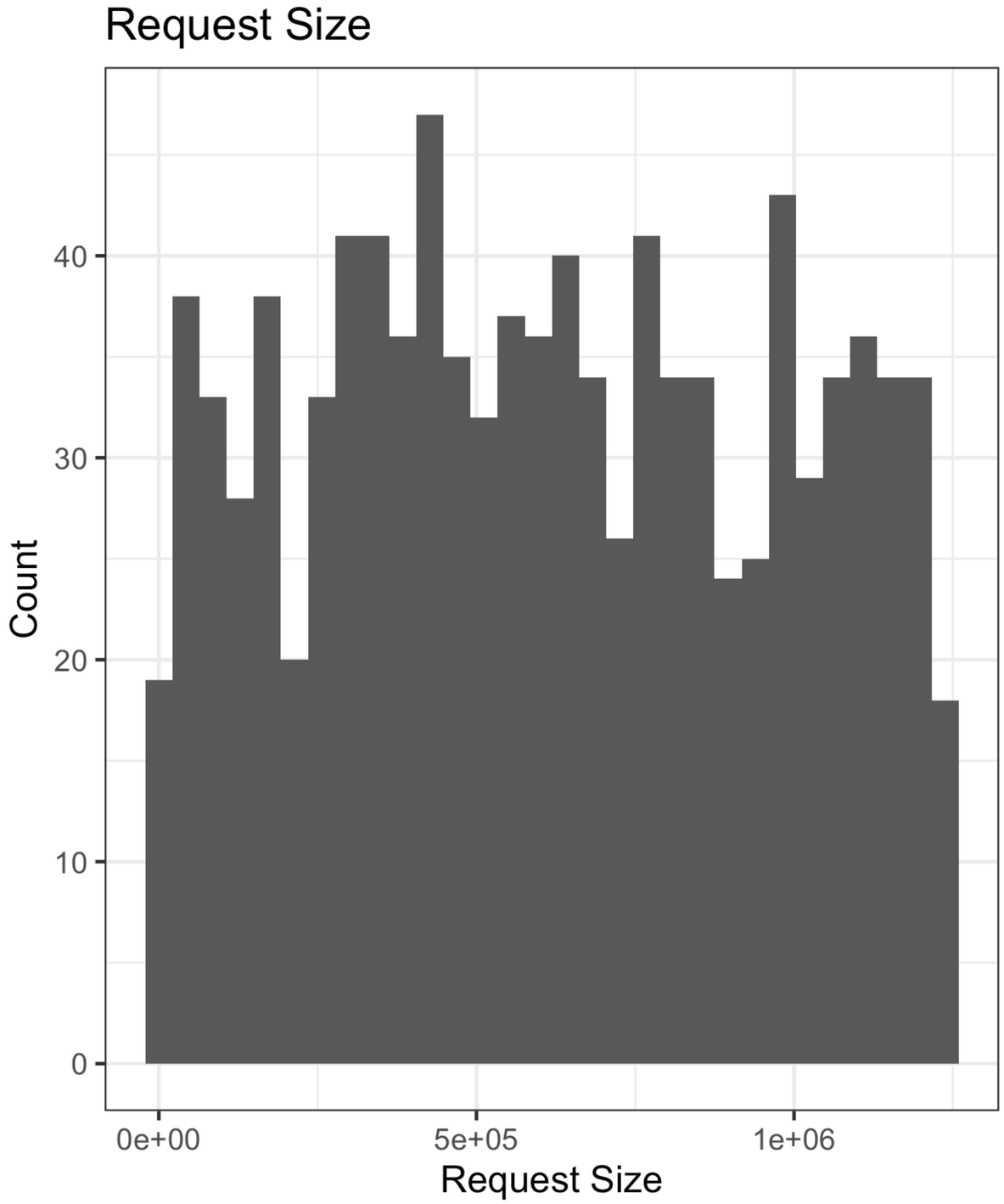


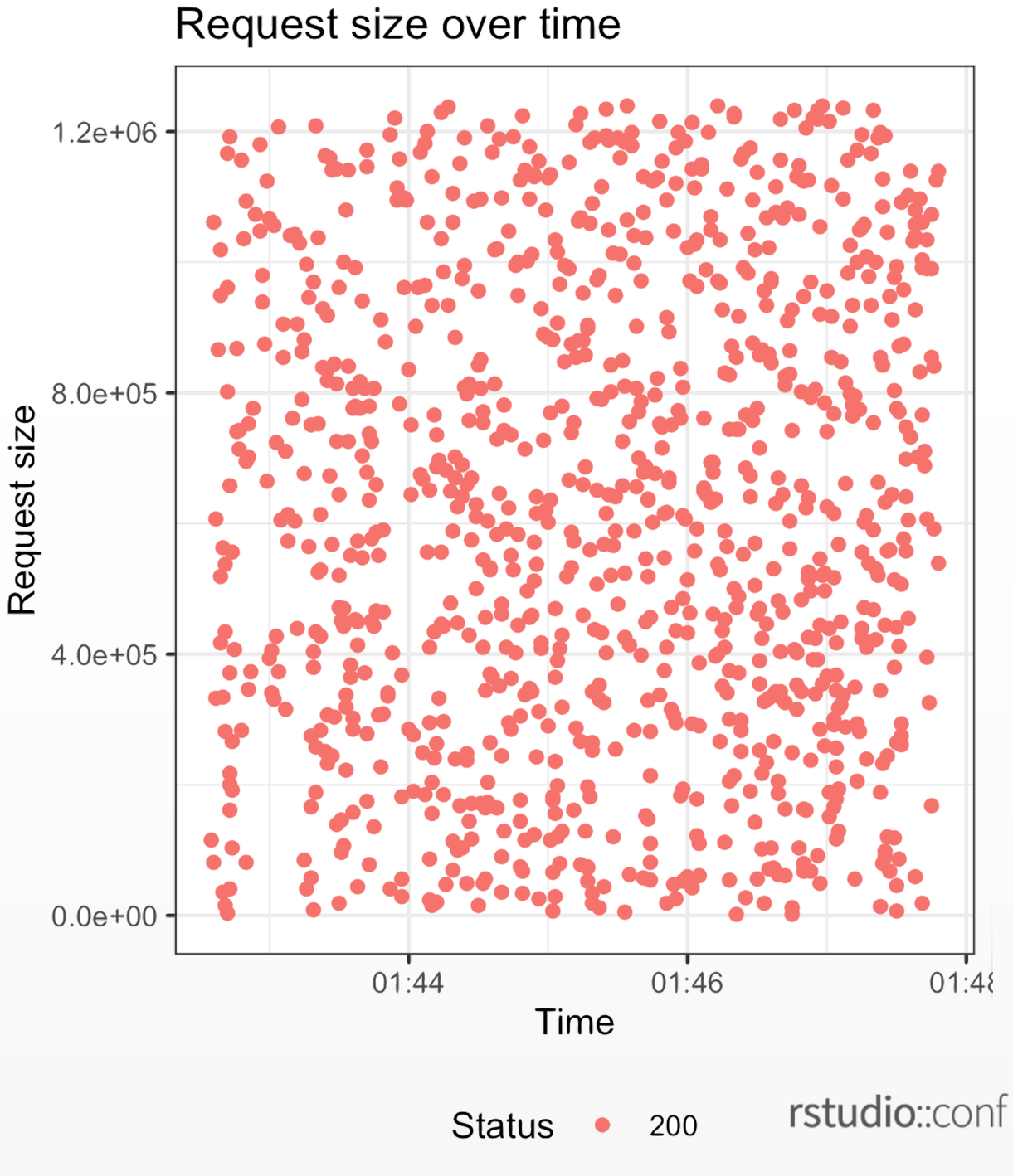
Pre-route actions



Post-route actions


Source
Console
Terminal
Jobs
run-ppp.R
1:45
INFO [2020-01-29 01:43:41] 127.0.0.1 "libcurl/7.64.1 r-curl/4.3 httr/1.4.1" localhost:5762 POST /predict 200 0.369 806798
INFO [2020-01-29 01:43:42] 127.0.0.1 "libcurl/7.64.1 r-curl/4.3 httr/1.4.1" localhost:5762 POST /predict 200 0.427 1171557
INFO [2020-01-29 01:43:42] 127.0.0.1 "libcurl/7.64.1 r-curl/4.3 httr/1.4.1" localhost:5762 POST /predict 200 0.119 174743
INFO [2020-01-29 01:43:42] 127.0.0.1 "libcurl/7.64.1 r-curl/4.3 httr/1.4.1" localhost:5762 POST /predict 200 0.261 678184
INFO [2020-01-29 01:43:42] 127.0.0.1 "libcurl/7.64.1 r-curl/4.3 httr/1.4.1" localhost:5762 POST /predict 200 0.354 1146100
INFO [2020-01-29 01:43:42] 127.0.0.1 "libcurl/7.64.1 r-curl/4.3 httr/1.4.1" localhost:5762 POST /predict 200 0.134 278158
INFO [2020-01-29 01:43:43] 127.0.0.1 "libcurl/7.64.1 r-curl/4.3 httr/1.4.1" localhost:5762 POST /predict 200 0.312 779616
INFO [2020-01-29 01:43:43] 127.0.0.1 "libcurl/7.64.1 r-curl/4.3 httr/1.4.1" localhost:5762 POST /predict 200 0.315 737423
INFO [2020-01-29 01:43:43] 127.0.0.1 "libcurl/7.64.1 r-curl/4.3 httr/1.4.1" localhost:5762 POST /predict 200 0.217 635803
INFO [2020-01-29 01:43:43] 127.0.0.1 "libcurl/7.64.1 r-curl/4.3 httr/1.4.1" localhost:5762 POST /predict 200 0.09 77828
INFO [2020-01-29 01:43:44] 127.0.0.1 "libcurl/7.64.1 r-curl/4.3 httr/1.4.1" localhost:5762 POST /predict 200 0.191 576444
INFO [2020-01-29 01:43:44] 127.0.0.1 "libcurl/7.64.1 r-curl/4.3 httr/1.4.1" localhost:5762 POST /predict 200 0.217 725674
INFO [2020-01-29 01:43:44] 127.0.0.1 "libcurl/7.64.1 r-curl/4.3 httr/1.4.1" localhost:5762 POST /predict 200 0.165 451151


Request Size
Count
40
30
20
10
0
0e+00
5e+05
1e+06
Request Size


Request size over time
Request size
1.2e+06
8.0e+05
4.0e+05
0.0e+00
01:44
01:46
Time
Status
200
rstudio::conf

API Usage

615
Total Requests

21.96
Requests per Second

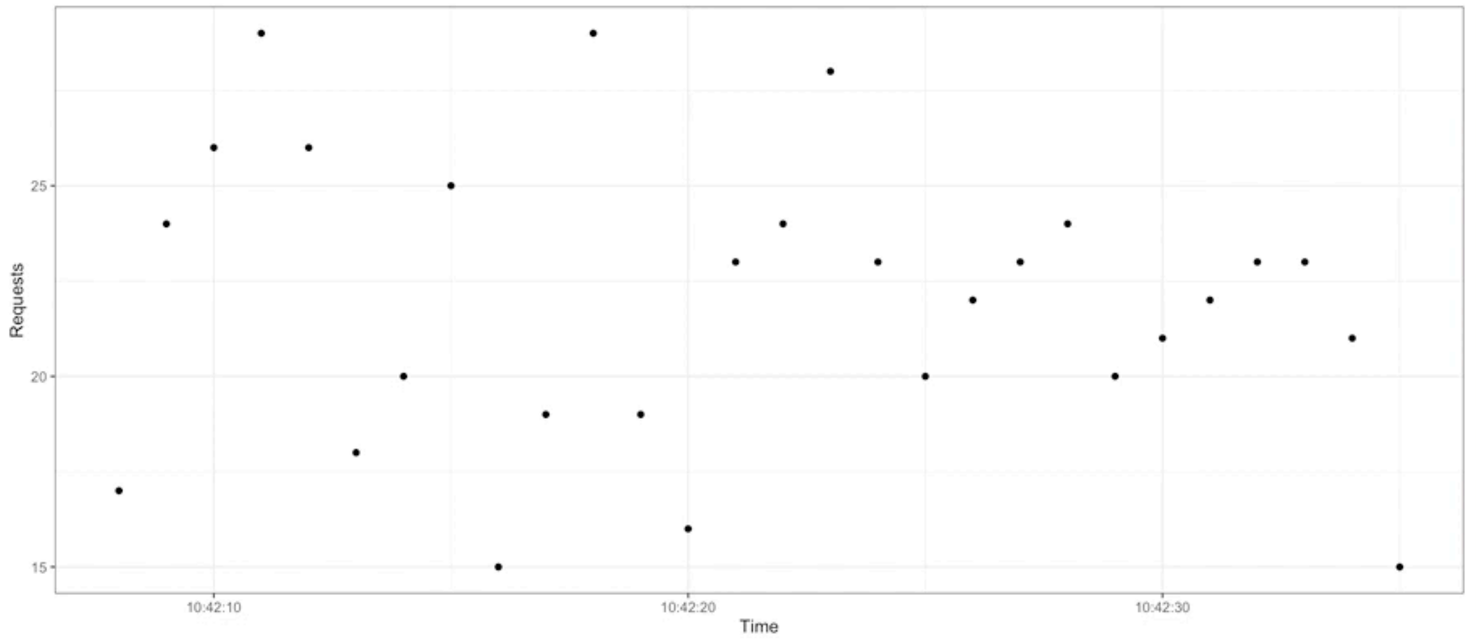
100%
Success

0.0413
Average Execution Time (S)

615
POST localhost:5762/predict

615
Mozilla/5.0 (apple-x86_64-darwin18.6.0) Siege/4.0.4@127.0.0.1

Requests Overview




plumber

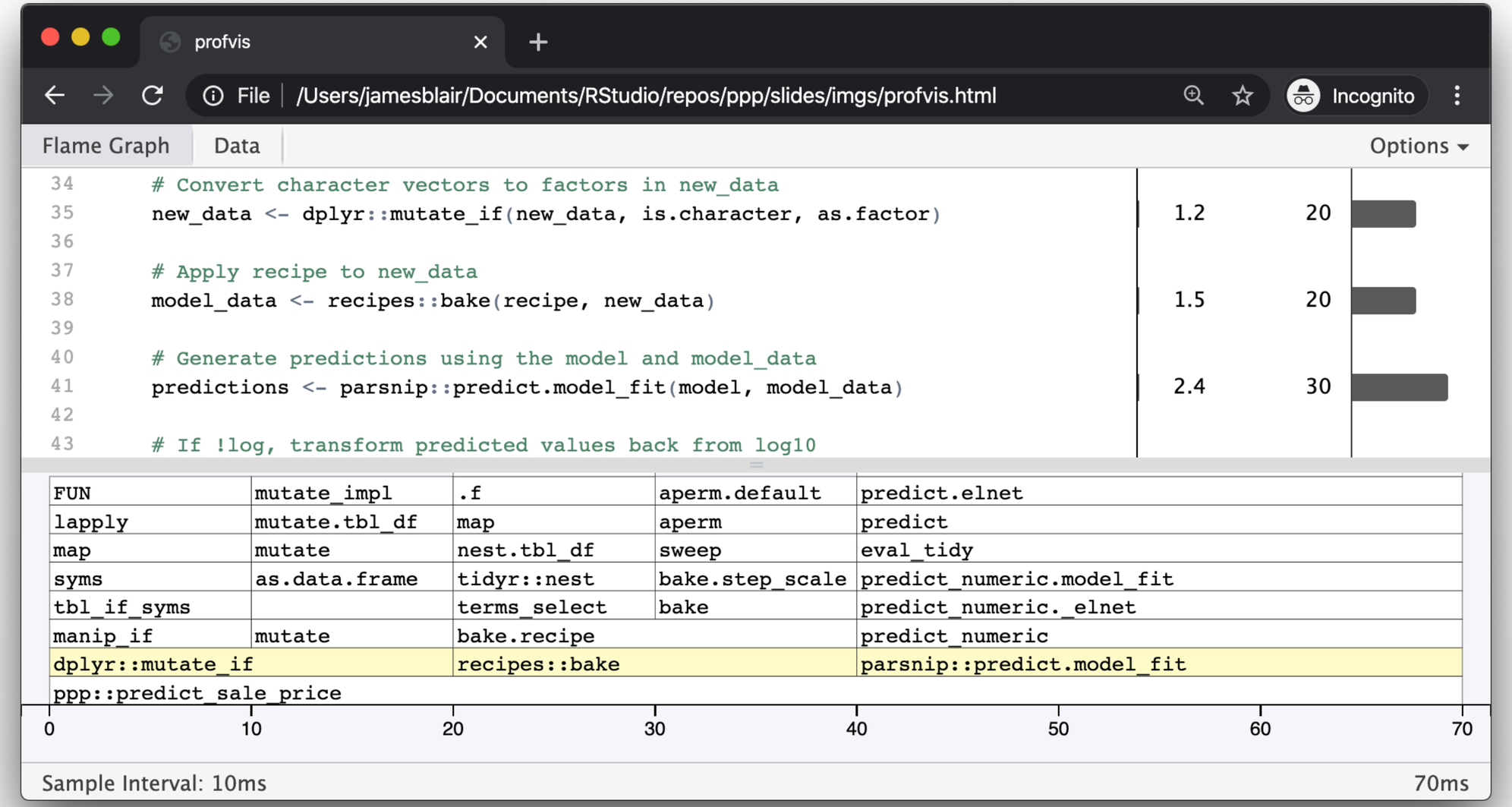
Load testing

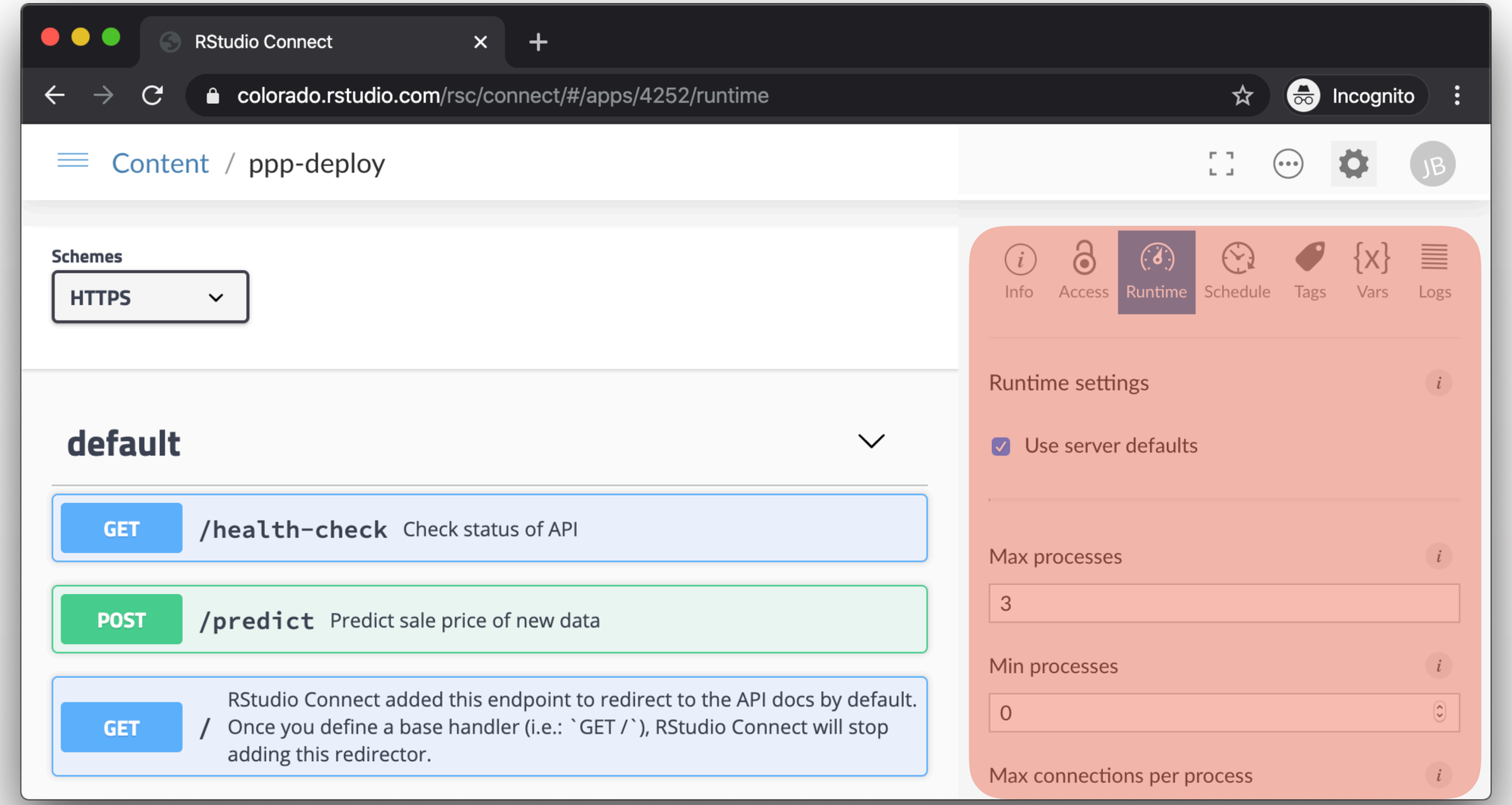
APACHE JMeter™

Gatling

HEY

LOCUST


profvis
File /Users/jamesblair/Documents/RStudio/repos/ppp/slides/imgs/profvis.html
Incognito
Flame Graph
Data
Options
34 # Convert character vectors to factors in new_data
35 new_data <- dplyr::mutate_if(new_data, is.character, as.factor)
36
37 # Apply recipe to new_data
38 model_data <- recipes::bake(recipe, new_data)
39
40 # Generate predictions using the model and model_data
41 predictions <- parsnip::predict.model_fit(model, model_data)
42
43 # If !log, transform predicted values back from log10
1.2 20
1.5 20
2.4 30
FUN mutate_impl .f aperm.default predict.elnet
lapply mutate.tbl_df map aperm predict
map mutate nest.tbl_df sweep eval_tidy
syms as.data.frame tidyr::nest bake.step_scale predict_numeric.model_fit
tbl_if_syms terms_select bake predict_numeric._elnet
manip_if mutate bake.recipe predict_numeric
dplyr::mutate_if recipes::bake parsnip::predict.model_fit
ppp::predict_sale_price
0 10 20 30 40 50 60 70
Sample Interval: 10ms
70ms


RStudio Connect
colorado.rstudio.com/rsc/connect/#/apps/4252/runtime
Incognito
Content / ppp-deploy
JB
Schemes
HTTPS
default
GET /health-check Check status of API
POST /predict Predict sale price of new data
GET / RStudio Connect added this endpoint to redirect to the API docs by default. Once you define a base handler (i.e.: `GET /`), RStudio Connect will stop adding this redirector.
Info
Access
Runtime
Schedule
Tags
Vars
Logs
Runtime settings
Use server defaults
Max processes
3
Min processes
0
Max connections per process

Yes thank you for the raise, and the extra vacation, and the private office, and the company car, and the private jet…

Things on my computer

Things others can easily access and use

Value

Thank you

rplumber.io

github.com/blairj09/ppp