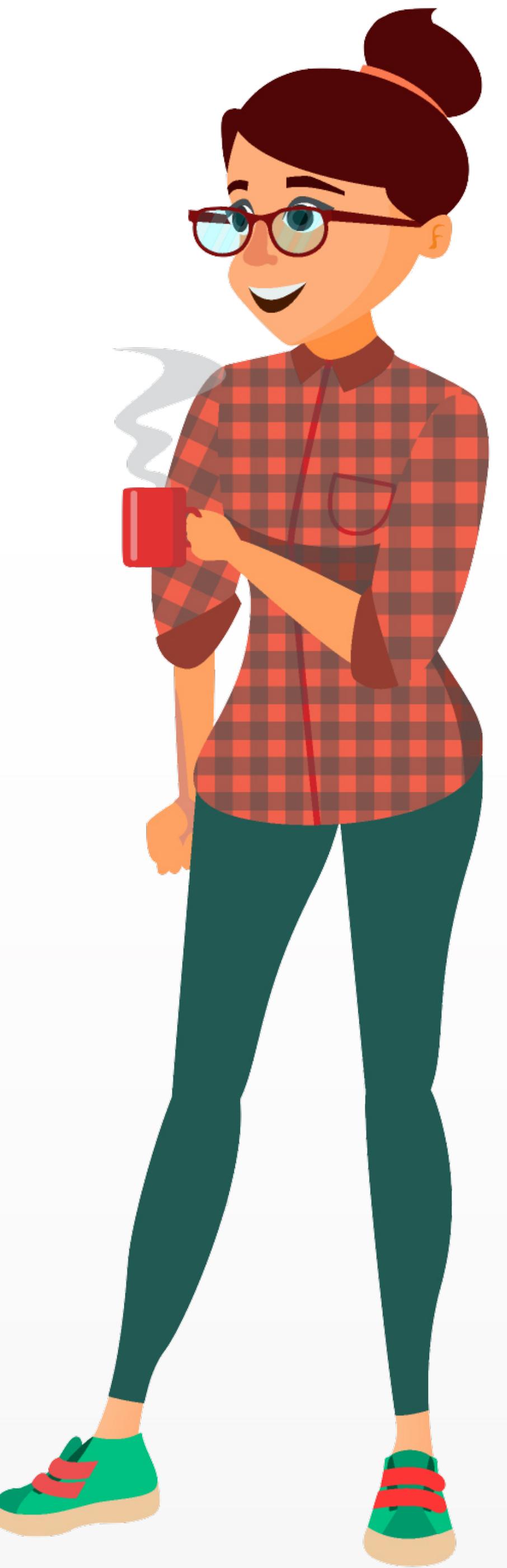


# Practical plumber Patterns

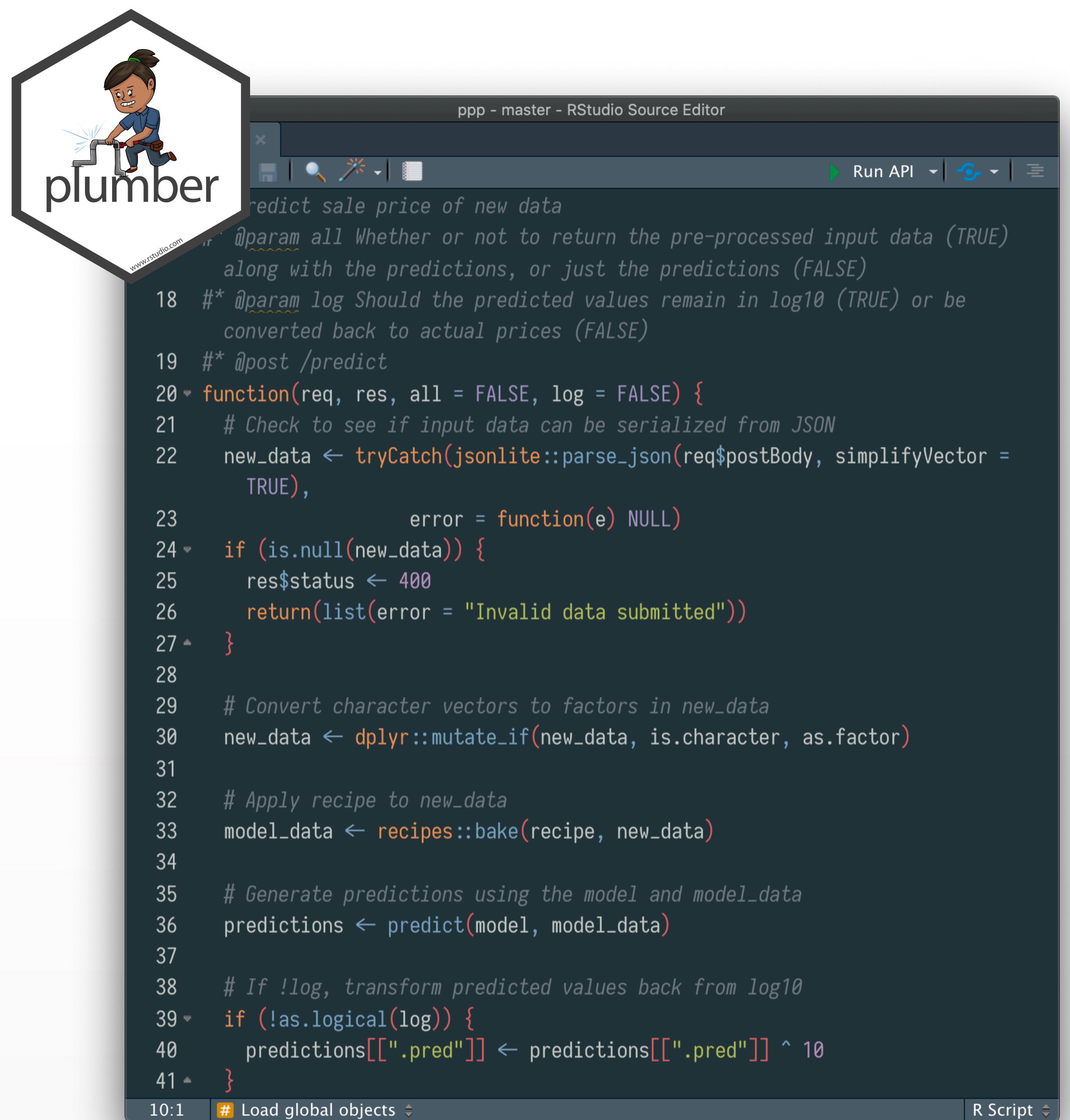
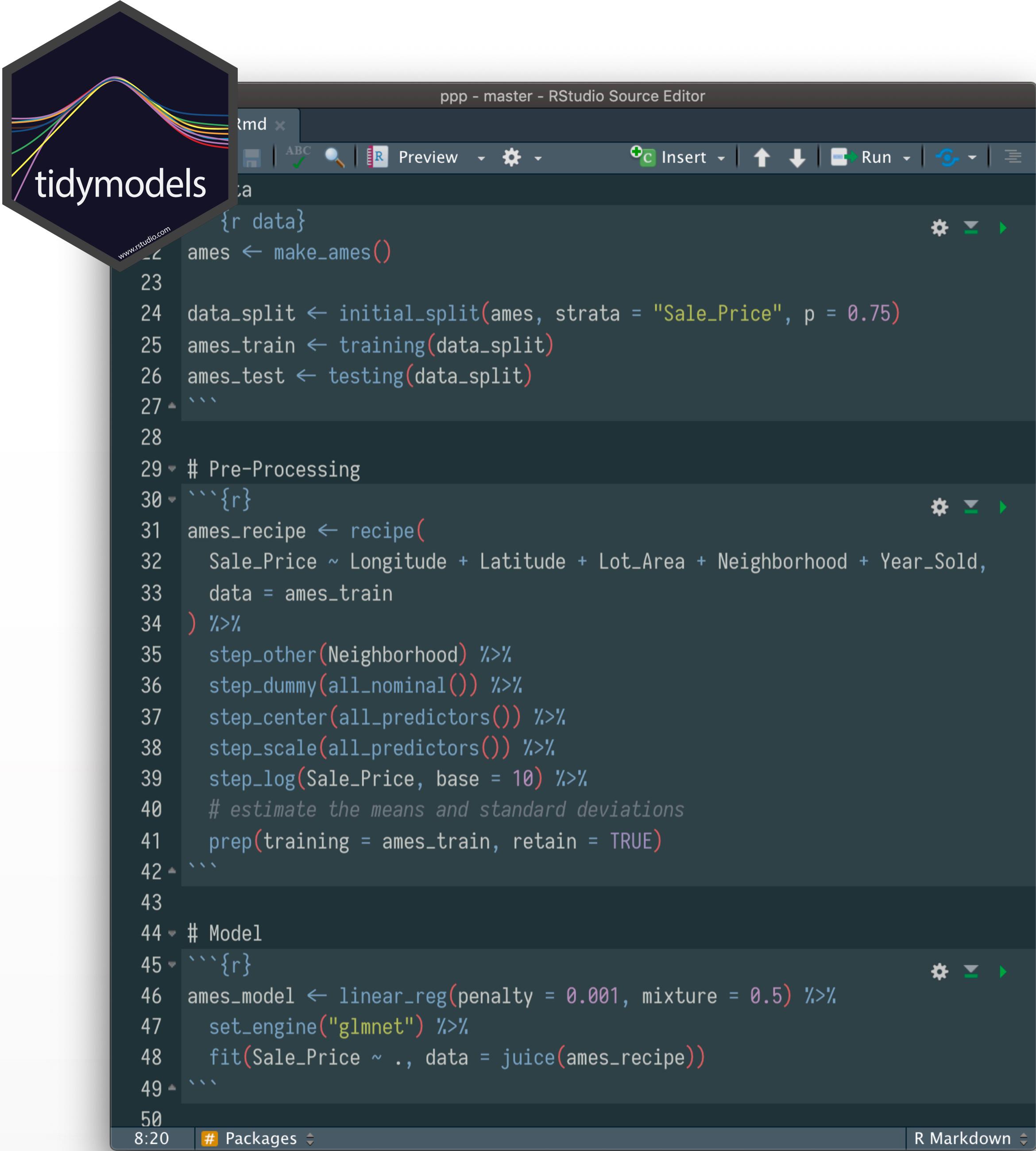


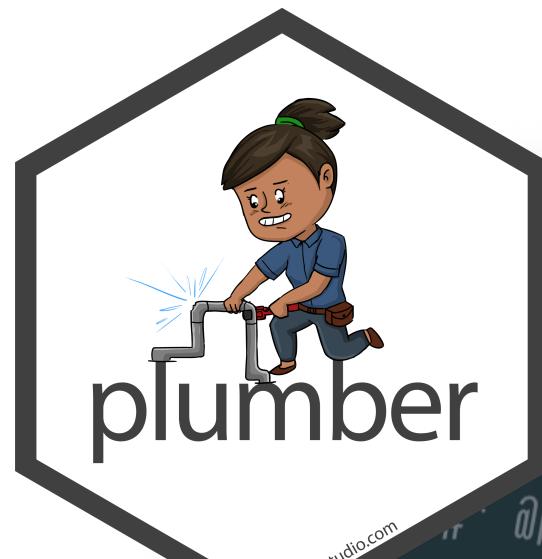
James Blair  
Solutions Engineer @ RStudio

# Meet Susan



rstudio::conf





ppp - master - RStudio Source Editor

```
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)
18  ## @param log Should the predicted values remain in log10 (TRUE) or be
      converted back to actual prices (FALSE)
19  #* @post /predict
20  function(req, res, all = FALSE, log = FALSE) {
21    # Check to see if input data can be serialized from JSON
22    new_data <- tryCatch(jsonlite::parse_json(req$postBody, simplifyVector =
      TRUE),
23                          error = function(e) NULL)
24    if (is.null(new_data)) {
25      res$status <- 400
26      return(list(error = "Invalid data submitted"))
27    }
28
29    # Convert character vectors to factors in new_data
30    new_data <- dplyr::mutate_if(new_data, is.character, as.factor)
31
32    # Apply recipe to new_data
33    model_data <- recipes::bake(recipe, new_data)
34
35    # Generate predictions using the model and model_data
36    predictions <- predict(model, model_data)
37
38    # If !log, transform predicted values back from log10
39    if (!as.logical(log)) {
40      predictions[[".pred"]] <- predictions[[".pred"]] ^ 10
41    }

```

10:1 # Load global objects R Script

Swagger UI

colorado.rstudio.com/rsc/ppp/\_swagger\_/

Incognito

swagger

https://colorado.rstudio.com/rsc/ppp/swagger.json?schemes=https&host=colc

Explore

## Sale Price Model 1.0.0

[ Base url: colorado.rstudio.com/rsc/ppp/] <https://colorado.rstudio.com/rsc/ppp/swagger.json?schemes=https&host=colorado.rstudio.com&path=/rsc/ppp/>

API Description

Schemes

HTTPS

### default

GET /health-check Check status of API

POST /predict Predict sale price of new data

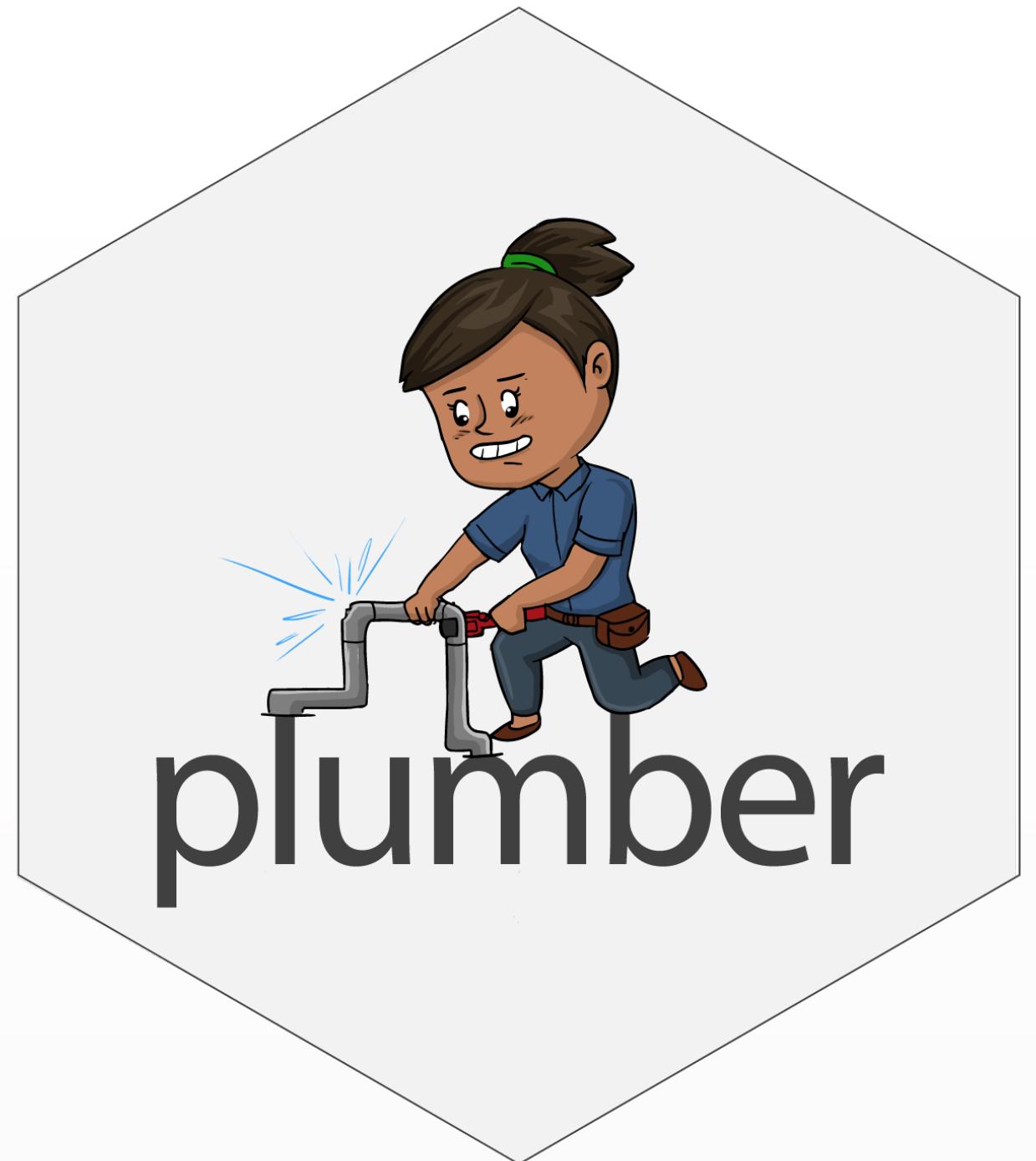
GET / RStudio Connect added this endpoint to redirect to the API docs by default. Once you define a base handler (i.e.: `GET /`), RStudio Connect will stop adding this redirector.

# Test



*Packages are the fundamental  
units of reproducible R code.*

- Hadley Wickham



rstudio::conf





ppp - master - RStudio Source Editor

tree x

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 x

ABC

```
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 x

DESCRIPTION  
LICENSE  
LICENSE.md  
NAMESPACE  
R  
└ <R function files>  
README.md  
inst  
└ plumber  
└ <Plumber Files>  
man  
tests  
└ testthat  
└ <R code tests>  
└ <Plumber tests>  
testthat.R

Text file ▾

ppp - slides - RStudio Source Editor

R predict.R x

Source on Save | Run | Source |

```
28 #' @export
29 predict_sale_price <- function(new_data, all = TRUE, log = FALSE)
29 {
30   # Load saved model and recipe
31   model <- readRDS(system.file("plumber", "model", "ames-model",
31   .rds", package = "ppp"))
32   recipe <- readRDS(system.file("plumber", "model", "ames-recipe",
32   .rds", package = "ppp"))
33
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
44   if (!as.logical(log)) {
45     predictions[[".pred"]] <- 10 ^ predictions[[".pred"]]
46   }
47
```

64:17 (Top Level) ▾

R Script ▾

ppp - master - RStudio Source Editor

tree x

DESCRIPTION  
LICENSE  
LICENSE.md  
NAMESPACE  
R  
  └ <R function files>  
README.md  
inst  
  └ plumber  
    └ <Plumber Files>  
man  
tests  
  └ testthat  
    └ <R code tests>  
    └ <Plumber tests>  
testthat.R

Text file ▾

ppp - slides - RStudio Source Editor

plumber.R x

Run API ▾

```
11 #* Predict sale price of new data
12 #* @param all Whether or not to return the pre-processed input
13 #* data (TRUE) along with the predictions, or just the predictions
14 #* (FALSE)
15 #* @param log Should the predicted values remain in log10 (TRUE)
16 #* or be converted back to actual prices (FALSE)
17 #* @post /predict
18 function(req, res, all = FALSE, log = FALSE) {
19   # Check to see if input data can be serialized from JSON
20   new_data <- tryCatch(jsonlite::parse_json(req$postBody,
21                                             simplifyVector = TRUE),
22                         error = function(e) NULL)
23
24   if (is.null(new_data)) {
25     res$status <- 400
26     return(list(error = "Invalid data submitted"))
27   }
28
29   ppp::predict_sale_price(new_data, all = all, log = log)
30 }
```

4:1 (Top Level) ▾

R Script ▾

ppp - master - RStudio Source Editor

tree x

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

Text file ▾

ppp - slides - RStudio Source Editor

test-predict.R x

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 f rand\_nas(v, p) ▾ R Script ▾

ppp - master - RStudio Source Editor

tree x

ABC

```
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
```

Text file ▾

ppp - master - RStudio Source Editor

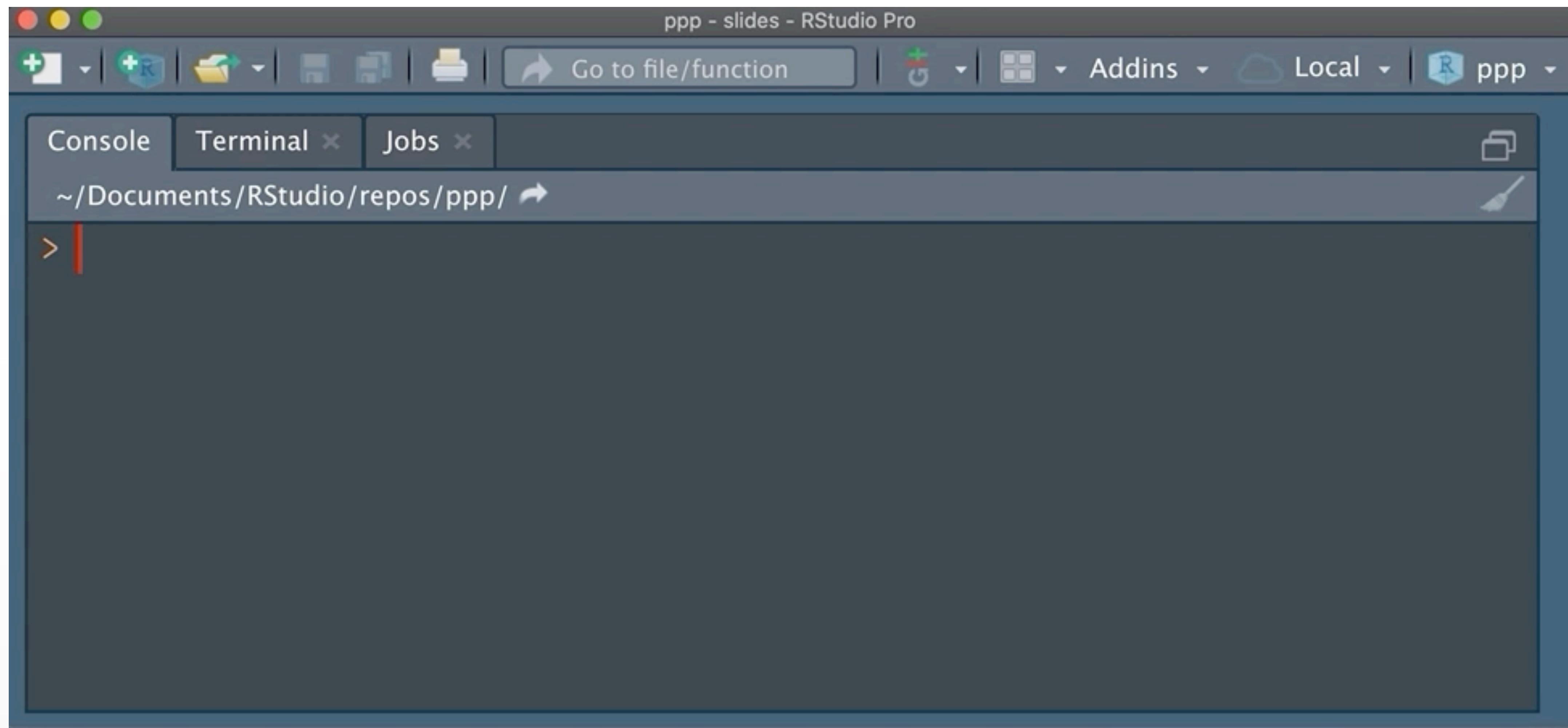
test-plumber.R x

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
20   # listening. At most
21   # wait 15 seconds for the API to initialize
22   max_s <- 15
23   for (i in 1:max_s) {
24     if (pred_api$is_alive())
25       break
26     Sys.sleep(1)
27   }
28   expect_true(pred_api$is_alive())
29 })
30
31 - test_that("Swagger UI is available", {
32   url <- paste0(root_path, "/swagger-ui.html")
33   res <- httr::GET(url)
34   expect_equal(res$status_code, 200)
35   content_type <- content_type(res)
36   expect_equal(content_type, "text/html; charset=UTF-8")
37 })
38
39 - test_that("Health Check endpoint returns OK", {
40   url <- paste0(root_path, "/health")
41   res <- httr::GET(url)
42   expect_equal(res$status_code, 200)
43   content_type <- content_type(res)
44   expect_equal(content_type, "application/json; charset=UTF-8")
45   content <- content(res, "text")
46   expect_equal(content, "{'status': 'ok'}")
47 })
48
49 - test_that("Plumber API handles invalid requests", {
50   url <- paste0(root_path, "/invalid-endpoint")
51   res <- httr::GET(url)
52   expect_equal(res$status_code, 404)
53   content_type <- content_type(res)
54   expect_equal(content_type, "text/plain; charset=UTF-8")
55   content <- content(res, "text")
56   expect_equal(content, "404: Not Found")
57 })
58
59 - test_that("Plumber API handles invalid JSON input", {
60   url <- paste0(root_path, "/invalid-json")
61   res <- httr::POST(url, body = "invalid json")
62   expect_equal(res$status_code, 400)
63   content_type <- content_type(res)
64   expect_equal(content_type, "text/plain; charset=UTF-8")
65   content <- content(res, "text")
66   expect_equal(content, "400: Bad Request")
67 })
68
69 - test_that("Plumber API handles invalid JSON output", {
70   url <- paste0(root_path, "/invalid-json-output")
71   res <- httr::GET(url)
72   expect_equal(res$status_code, 400)
73   content_type <- content_type(res)
74   expect_equal(content_type, "text/plain; charset=UTF-8")
75   content <- content(res, "text")
76   expect_equal(content, "400: Bad Request")
77 })
78
79 - test_that("Plumber API handles invalid JSON output", {
80   url <- paste0(root_path, "/invalid-json-output")
81   res <- httr::GET(url)
82   expect_equal(res$status_code, 400)
83   content_type <- content_type(res)
84   expect_equal(content_type, "text/plain; charset=UTF-8")
85   content <- content(res, "text")
86   expect_equal(content, "400: Bad Request")
87 })
88
89 - test_that("Plumber API handles invalid JSON output", {
90   url <- paste0(root_path, "/invalid-json-output")
91   res <- httr::GET(url)
92   expect_equal(res$status_code, 400)
93   content_type <- content_type(res)
94   expect_equal(content_type, "text/plain; charset=UTF-8")
95   content <- content(res, "text")
96   expect_equal(content, "400: Bad Request")
97 })
98
99 - test_that("Plumber API handles invalid JSON output", {
100  url <- paste0(root_path, "/invalid-json-output")
101  res <- httr::GET(url)
102  expect_equal(res$status_code, 400)
103  content_type <- content_type(res)
104  expect_equal(content_type, "text/plain; charset=UTF-8")
105  content <- content(res, "text")
106  expect_equal(content, "400: Bad Request")
107 })
```

(Top Level) ▾

R Script ▾



rstudio::conf

# Deploy



The screenshot shows the RStudio interface with a dark theme. The top bar displays the file name "plumber.R". The toolbar includes standard icons for file operations, search, and code navigation. On the right side of the toolbar, there are buttons for "Run API" and other tools. The main code editor area contains the following R code:

```
11  #'* Predict sale price of new data
12  #'* @param all Whether or not to return the pre-processed input data (TRUE)
13  #'* @param log Should the predicted values remain in log10 (TRUE) or be
14  #'* converted back to actual prices (FALSE)
15  #'@post /predict
16  # Check to see if input data can be serialized from JSON
17  new_data <- tryCatch(jsonlite::parse_json(req$postBody, simplifyVector = TRUE
18
19  error = function(e) NULL)
```

The code uses the plumber package to create a REST API endpoint for predicting house prices. It includes parameters for returning raw data or predictions, and an option to keep predictions in log10 or convert them back to actual prices. The code also handles JSON input from the POST body.

R Studio Connect

JB

## Content

JB Options

**ppp-ide**

jb james

Type: API Deployed: Today at 12:15 AM

**plumber-load-test-test**

ca cole

Type: API Deployed: Yesterday at 4:44 PM

RStudio Connect

colorado.rstudio.com/rsc/connect/#/apps/4252/info

Content / ppp-deploy

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.

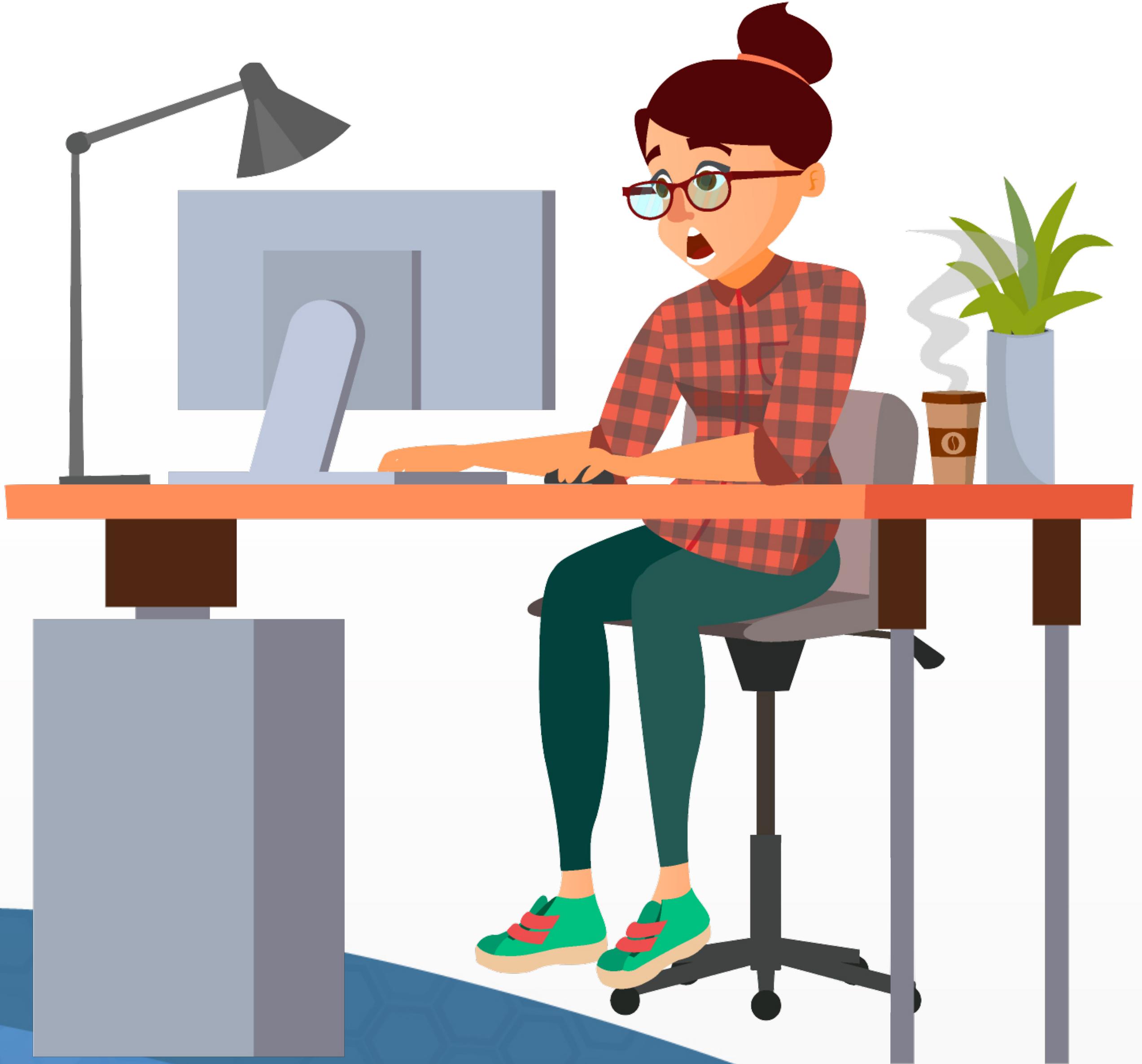
This content was generated from Git  
Repo: <https://github.com/blairj09/PPP>  
Branch: master  
Directory: /inst/plumber/pkg-api

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



rstudio::conf

ppp - slides - RStudio Pro

```
entrypoint.R
24 pr$registerHooks(
25   list(
26     preroute = function() {
27       # Start timer for log info
28       tictoc::tic()
29     },
30     postroute = function(req, res) {
31       end ← tictoc::toc(quiet = TRUE)
32       # Log details about the request and the response
33       log_info('{{convert_empty(req$REMOTE_ADDR)} } {{convert_empty
34         (req$HTTP_USER_AGENT)}}} {{convert_empty(req$HTTP_HOST)}} {{convert_empty
35         (req$REQUEST_METHOD)}} {{convert_empty(req$PATH_INFO)}} {{convert_empty
36         (res$status)}} {{round(end$toc - end$tic, digits = getOption("digits", 5
37           ))}}')
38   )
39 )
40
41 pr
```

Define actions that take place  
on the Plumber router

ppp - slides - RStudio Pro

entrypoint.R

Source on Save | Run | Source | R Script

```
24 pr$registerHooks(  
25   list(  
26     preroute = function() {  
27       # Start timer for log info  
28       tictoc::tic()  
29     },  
30     postroute = function(req, res) {  
31       end ← tictoc::toc(quiet = TRUE)  
32       # Log details about the request and the response  
33       log_info('{{convert_empty(req$REMOTE_ADDR)} "{{convert_empty  
34         (req$HTTP_USER_AGENT)}" {{convert_empty(req$HTTP_HOST)} {{convert_empty  
35         (req$REQUEST_METHOD)} {{convert_empty(req$PATH_INFO)} {{convert_empty  
36         (res$status)} {{round(end$toc - end$tic, digits = getOption("digits", 5  
37           ))}}')  
38     }  
39   )  
40 )  
41  
42 pr
```

5:16 (Top Level) R Script

Pre-route actions

rstudio::conf

ppp - slides - RStudio Pro

entrypoint.R

```
24 pr$registerHooks(  
25   list(  
26     preroute = function() {  
27       # Start timer for log info  
28       tictoc::tic()  
29     },  
30     postroute = function(req, res) {  
31       end ← tictoc::toc(quiet = TRUE)  
32       # Log details about the request and the response  
33       log_info('{{convert_empty(req$REMOTE_ADDR)} } {{convert_empty  
34         (req$HTTP_USER_AGENT)} } {{convert_empty(req$HTTP_HOST)} } {{convert_empty  
35         (req$REQUEST_METHOD)} } {{convert_empty(req$PATH_INFO)} } {{convert_empty  
36         (res$status)} } {{round(end$toc - end$tic, digits = getOption("digits", 5  
37           ))}}')  
38   })  
39 )  
38 pr
```

5:16 (Top Level) R Script

## Post-route actions

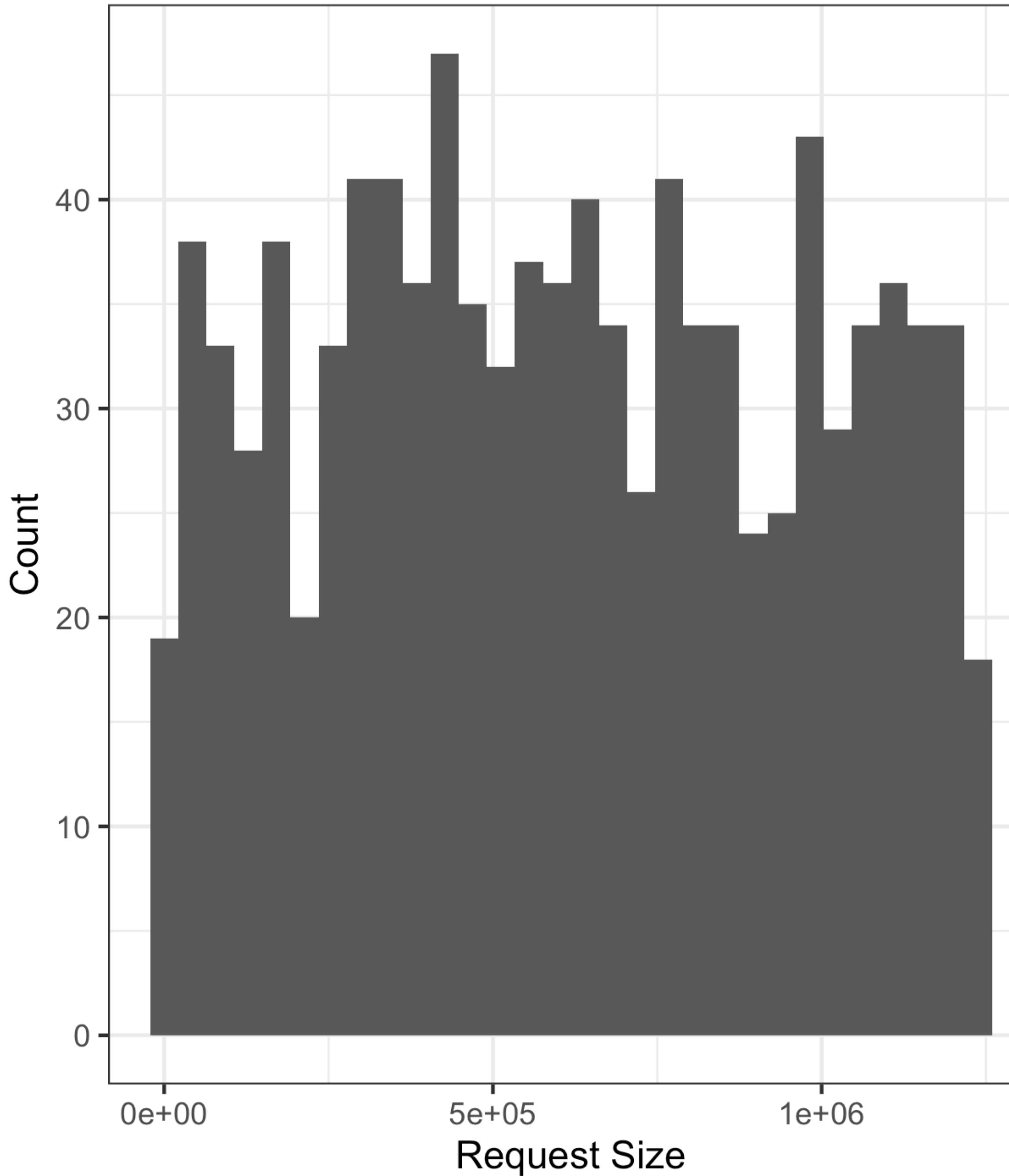
Source

Console Terminal x Jobs x

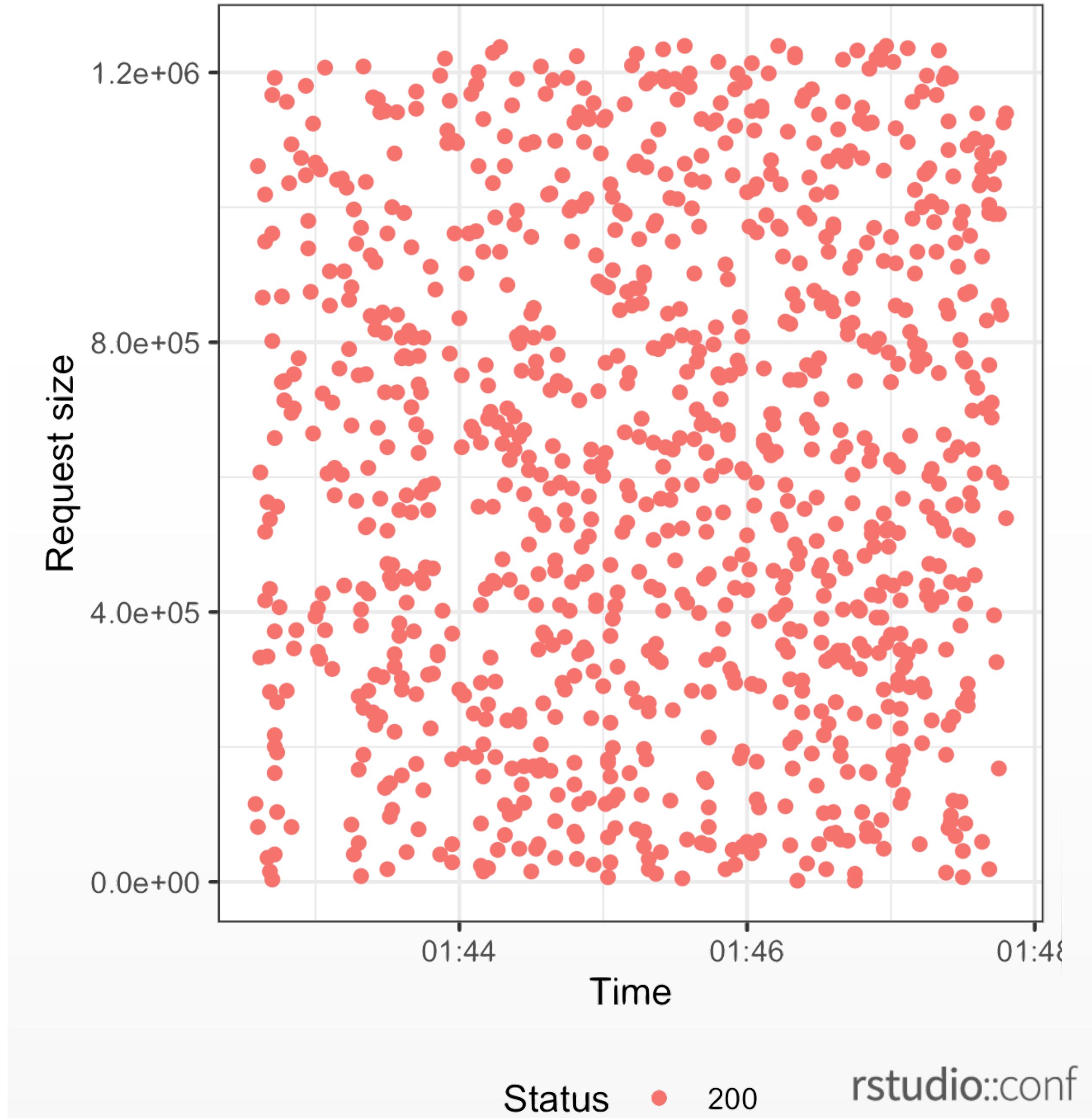
← run-ppp.R 1:45

```
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



## Request size over time



**615**

Total Requests

**21.96**

Requests per Second

**100%**

Success

**0.0413**

Average Execution Time (S)

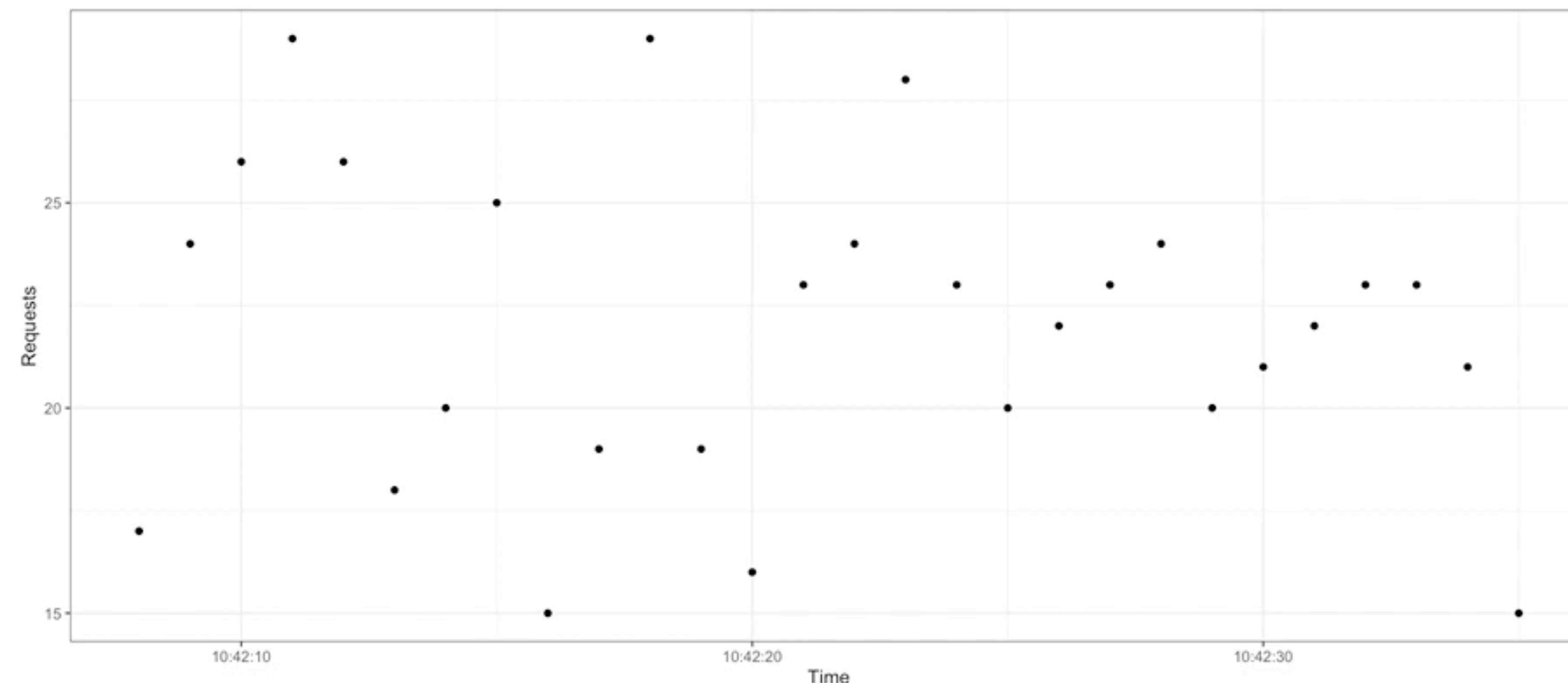
**615**

POST localhost:5762/predict

**615**

Mozilla/5.0 (apple-x86\_64-darwin18.6.0) Siege/4.0.4@127.0.0.1

## Requests Overview

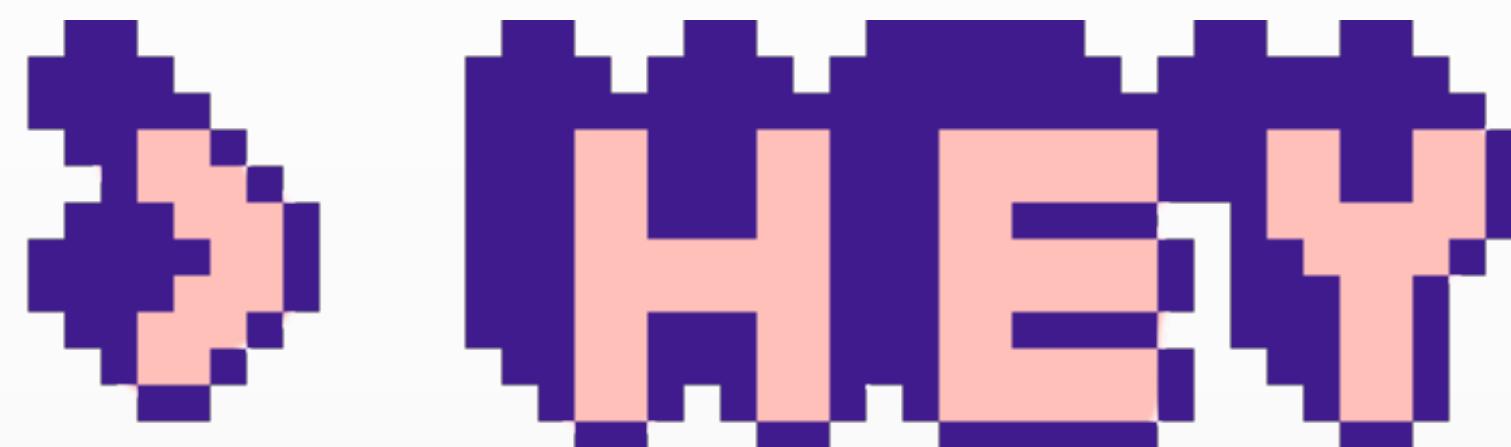




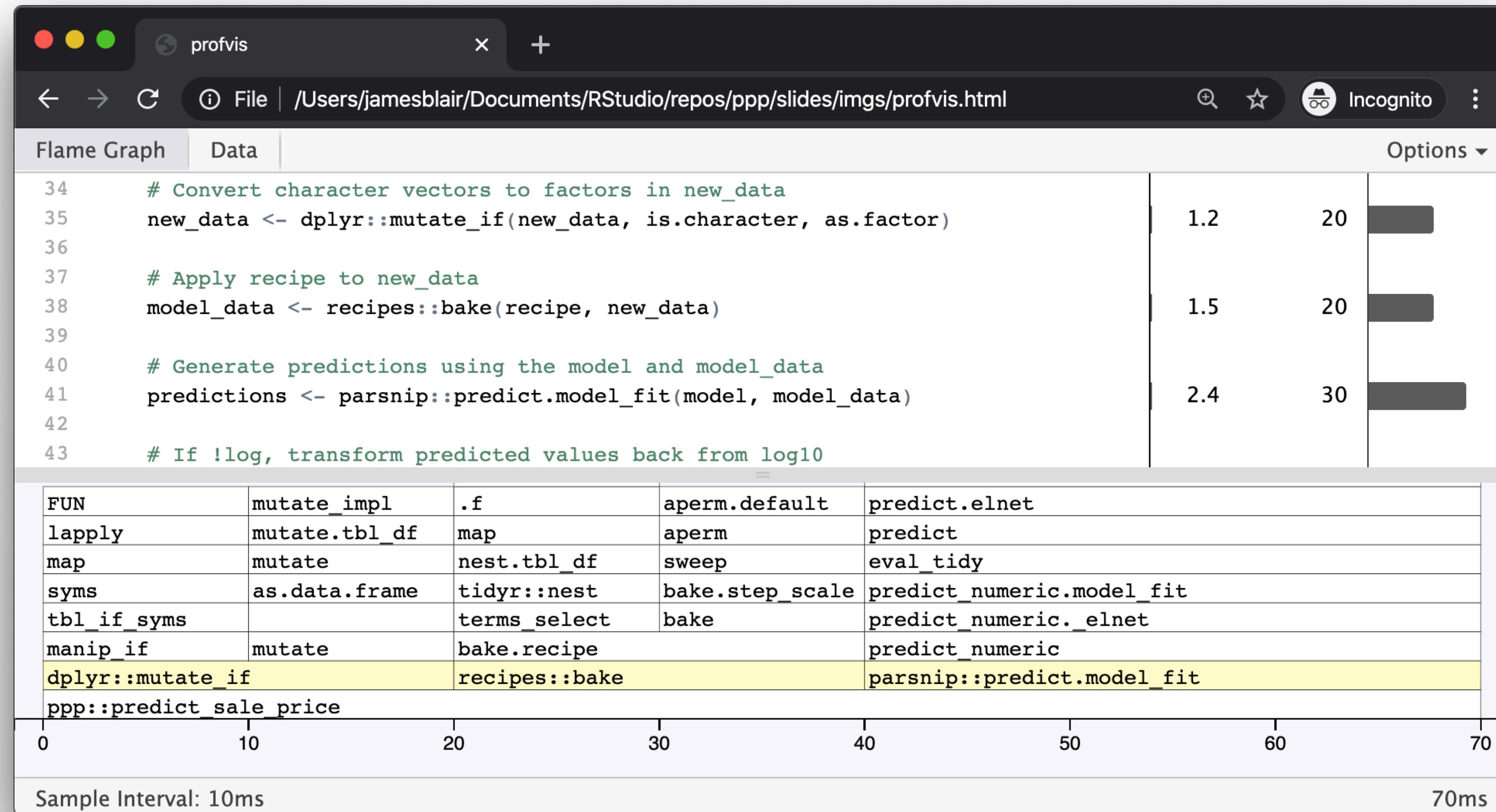
A large, diverse crowd of people of various ages, ethnicities, and styles are gathered together, looking towards the right side of the frame. They are dressed in a variety of clothing, including business suits, casual shirts, and athletic wear. In the center-right of the image, a white speech bubble is tilted diagonally. Inside the bubble, there is a cartoon illustration of a dark-skinned person with short brown hair tied up in a bun, wearing a blue shirt and jeans, kneeling on the ground and working on a pipe with a wrench. Blue sparks are flying from the pipe, indicating it is leaking or being repaired. Below the illustration, the word "plumber" is written in a bold, black, sans-serif font.

plumber

# Load testing



LOCUST



RStudio Connect

colorado.rstudio.com/rsc/connect/#/apps/4252/runtime

Content / ppp-deploy

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.

Runtime settings

Use server defaults

Max processes

3

Min processes

0

Max connections per process

This screenshot shows the RStudio Connect interface for managing an application named 'ppp-deploy'. On the left, the 'Content' section displays a list of API endpoints under the 'default' scheme. The 'https' dropdown is selected. The first endpoint is a GET request to '/health-check' for checking API status. The second endpoint is a POST request to '/predict' for predicting sale prices. A note indicates that the trailing slash endpoint is a redirect to the API documentation. On the right, the 'Runtime' tab is active, showing runtime settings with 'Use server defaults' checked. It also includes configuration for maximum and minimum processes (set to 3 and 0 respectively) and maximum 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](https://github.com/blairj09/ppp)