

What's inside? It's a suite of **Application Programming interfaces (APIs)**, a system which can receive **requests** from your computer keyboard, to carry out given **tasks** on given **resources**. This system is **programming-language-agnostic** (works for curl, Python, R, Javascript, PHP, etc.), relatively fast, and there's no need for additional packages installation when you have an access to it.

What are requests and resources here? In Techtonique/APIs, resources are Statistical/Machine Learning (ML) **model predictions**. In this context, one type of request for a task could be: obtain sales, weather or revenues forecast for the next 5 weeks. In general though, requests for tasks do not have such a long description. They are characterized by a verb and an URL Path – approximately, an endpoint – which both lead to a **response**.

Below in an illustrative example of API. In this example, the resource we're interested in is a list of users that we'd like to manage.

Request type (verb): GET

URL Path: <http://users> | endpoint: users | API Response: display a list of all users

URL Path: <http://users/:id> | endpoint: users/:id | API Response: display a specific user

Request type (verb): POST

URL Path: <http://users> | endpoint: users | API Response: create a new user

Request type (verb): PUT

URL Path: <http://users/:id> | endpoint: users/:id | API Response: update a specific user

Request type (verb): DELETE

URL Path: <http://users/:id> | endpoint: users/:id | API Response: delete a specific user

In Techtonique/APIs, a **resource endpoint** will be for example /MLmodel instead of /users. And since the resources are already implemented – as model predictions – and do not need to be altered (PUT) or deleted (DELETE), **every request for a forecast is a POST request to a /MLmodel**.

So, **how does the tool work?** You start by creating an account (sign up @ <https://www.techtonique.net>) using a valid email address. Then, prepare an input file containing univariate or multivariate time series, with the following format.

Univariate

date;Series1

1980-01-01;12.97065

1981-01-01;15.38714

1982-01-01;13.22957

1983-01-01;12.97065

1984-01-01;15.38714

1985-01-01;11.72288

1986-01-01;10.06177

1987-01-01;10.82279

OR

Multivariate

date;Series1;Series2

1980-01-01;12.97065;7.212725

1981-01-01;15.38714;9.44357

1982-01-01;13.22957;7.53425

1983-01-01;12.97065;7.212725

1984-01-01;15.38714;9.44357

1985-01-01;11.72288;6.415215

1986-01-01;10.06177;5.80699

1987-01-01;10.82279;6.2036

The first column of the input csv file must be named 'date'. In addition, the dates in this column must be regularly spaced, and have the format '%Y-%m-%d' (year, month, day, separated by '-s'). The rest of the procedure is described below for curl, Python and R (in that order). For more details on models' parameters, log in @ <https://www.techtonique.net>.

With curl

Obtain a token with the credentials that you used for signing up @ <https://www.techtonique.net>:

```
curl -u yourusername:yourpassword -i -X GET https://www.techtonique.net/token
```

Use the token for obtaining `dynrm` forecasts. `dynrm` is adapted from [NNETAR](#), but uses an automatic ridge regression to adjust time series' lags and seasonal lags.

```
curl -u token:x -F 'file=@/path/to/univariateinput.csv' "  
https://www.techtonique.net/dynrm?h=10&type\_pi=A"
```

With Python

`ridge2` (cf. second example) is the model from [Moudiki et al. \(2018\)](#) for multivariate time series.

```
import requests
```

```
base_url = "https://www.techtonique.net"
```

```
# same credentials that were used for signing up @  
https://www.techtonique.net
```

```
username = "yourusername"
```

```
password = "yourpassword"
```

```

# 1 - request for a token -----

response_token = requests.get(base_url + '/token', auth=(username,
password))

token = response_token.json()['token']

print("\n")
print(f"token: {token}")


# 2 - request using a token (univariate time series) -----

# for more details on these parameters, log in
params = (
    ('date_formatting', 'ms'),
    ('h', '5'),
)

files = {
    'file': open('/path/to/yourinputfileunivariate.csv', 'rb')
}

response = requests.post(base_url + '/dynrm',
params=params, files=files, auth=(token, 'x'))

print(response.json())


# 3 - request using a token (multivariate time series) -----

# for more details on these parameters, log in
params = (
    ('date_formatting', 'original'),
    ('h', '8'),
    ('nb_hidden', '10')
)

files = {
    'file': open('/path/to/yourinputfilemultivariate.csv', 'rb')
}

response = requests.post(base_url + '/ridge2',
params=params, files=files, auth=(token, 'x'))

print(response.json())

```

With R

```

library(httr)
library(datasets)

```

```

# same credentials that were used for signing up @
https://www.techtonique.net
username <- "yourusername"
password <- "yourpassword"

# 1 - request for a token -----

res_token <- httr::GET(url = 'https://www.techtonique.net/token',
                      httr::authenticate(username, password))

(token <- httr::content(res_token)$token)

# 2 - request a forecast using a token -----

# for more details on these parameters, log in
params = list(
  `date_formatting` = 'original',
  `h` = '15'
)

files = list(
  `file` = upload_file('/path/to/AirPassengers.csv')
)

(res_api <- httr::POST(url = 'https://www.techtonique.net/dynrm', query =
params,
                      body = files, httr::authenticate(token, 'x'))

list_res <- httr::content(res_api)

# 3 - Extract results -----

h <- length(list_res$ranges)

forecast_object <- list()
forecast_object$mean <- forecast_object$upper <- forecast_object$lower
<- rep(0, h)

for (i in 1:h)
{
  forecast_object$mean[i] <- list_res$averages[[i]][[2]]
  forecast_object$lower[i] <- list_res$ranges[[i]][[2]]
  forecast_object$upper[i] <- list_res$ranges[[i]][[3]]
}

```

```
# 4 - Plot results -----
```

```
par(mfrow=c(1, 2))
plot(AirPassengers, main="Dynrm forecast")
xx <- c(1:h, h:1)
yy <- c(forecast_object$lower, rev(forecast_object$upper))
plot(1:h, forecast_object$mean, type='l', main="Dynrm forecast \n using
Tecthonique's dynrm")
polygon(xx, yy, col = "gray90", border = "gray90")
lines(1:h, forecast_object$mean, col="blue")
```

