



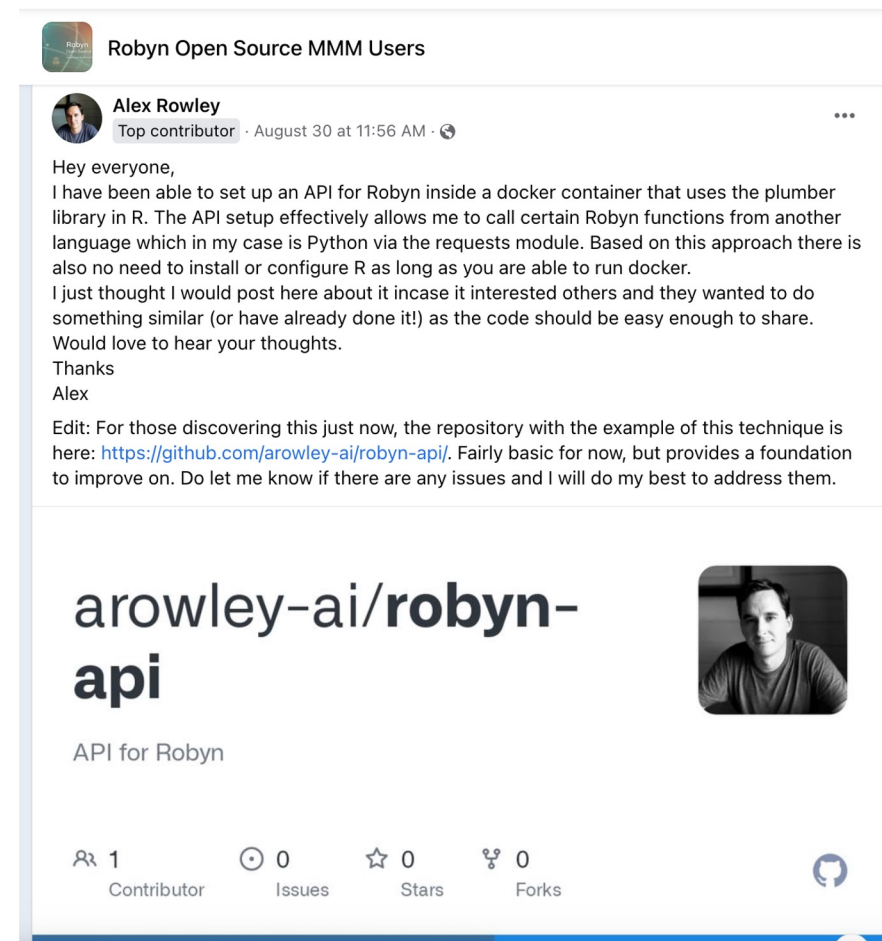
Robyn API for Python beta

RobynをAPI化するソリューションがRobynのオープンソースコミュニティから提案されました

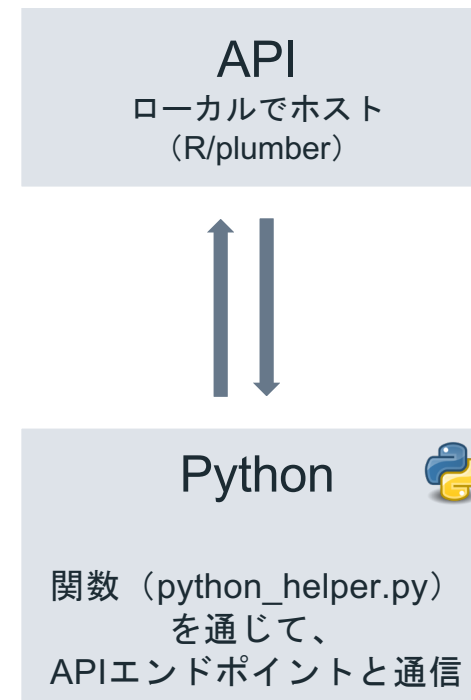
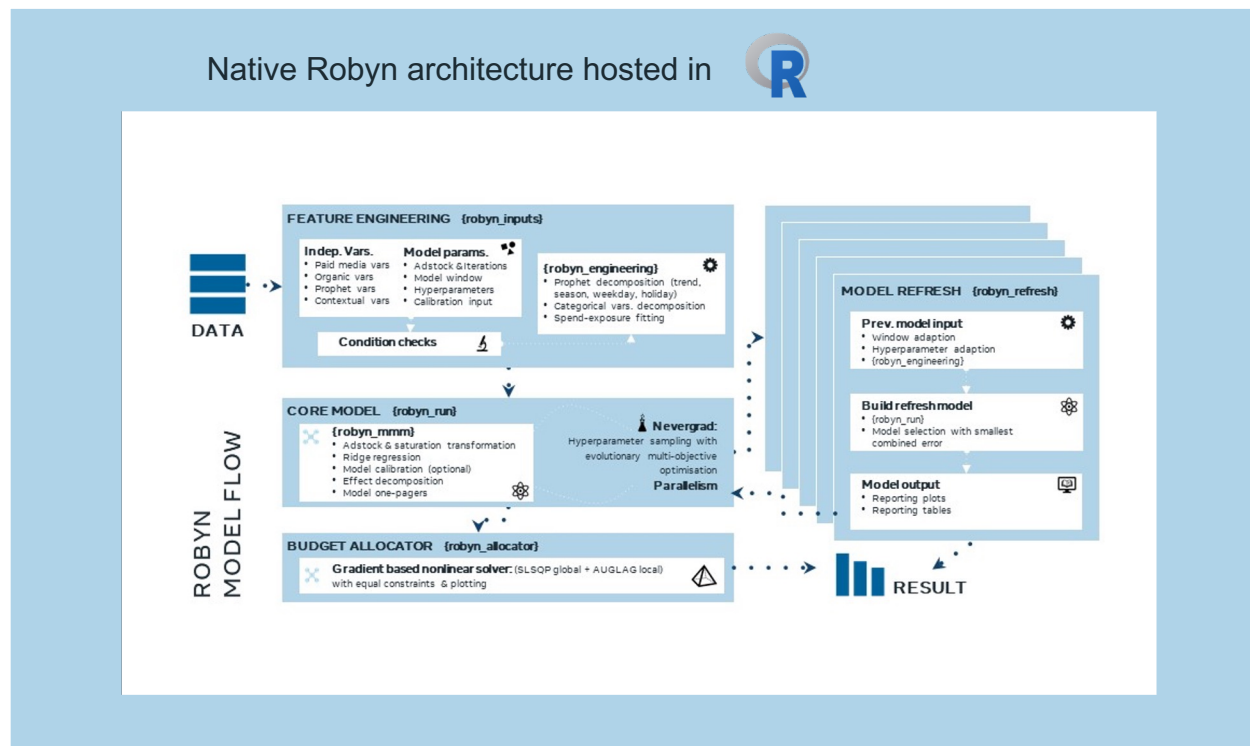
2023年8月、一人のRobynユーザーがFacebookグループにおいて、PythonからRobynを利用できるように、Dockerコンテナとplumberを使用したRobyn APIのプロトタイプを提案しました。

彼はプロトタイプとして動作する最初のソリューションをオープンソース化しました。

Robynチームはこのソリューションを発展させ、Robynの主要な関数をAPIとして利用できるように開発を行いました。



Robyn API for Python アーキテクチャ



このベータ版のAPIソリューションは、追加のコンテナ（dockerなど）に依存せず、PythonからAPIのRスクリプトをネイティブに呼び出します。



jupyter robyn_python_notebook (autosaved)

```
File Edit View Insert Cell Kernel Widgets Help
```

Not Trusted Python 3 (ipykernel) O

```
# "min_candidates": 100, # top pareto models for clustering. Default to 100
# "collation_constraint": 0.1, # range [0.01, 0.1] & default of 0.1
"csv_out": "pareto", # "pareto", "all", or NULL (for none)
"clusters": True, # Set to TRUE to cluster similar models by RMSE.
"export": True, # create_files, # this will create files locally
"plot_folder": robyn_directory, # path for plots exports and files creation
"plot_pareto": create_files # Set to FALSE to deactivate plotting and saving model one-pagers
}

>>> Build the payload for the robyn_outputs()
payload = {
    'InputCollect': json.dumps(InputCollect),
    'OutputModels': json.dumps(OutputModels),
    'jsonOutputsArgs': json.dumps(outputsArgs)
}

In [28]: M Get response
OutputCollect = robyn_api('robyn_outputs',payload,payload)

>>> Running Pareto calculations for 500 models on auto fronts...
>>> Automatically selected 6 Pareto-fronts to contain at least 100 pareto-optimal models (100)
>>> Calculating response curves for all models' media variables (540)...
>>> Pareto-Front: 1 [21 models]

00:00:01 =====> 100% | 21
>>> Pareto-Front: 2 [17 models]

00:00:01 =====> 100% | 17
00:00:00 ==> 4.70K | 1

>>> Pareto-Front: 3 [21 models]

00:00:02 =====> 100% | 6.67K
00:00:00 ==>

>>> Pareto-Front: 4 [15 models]

00:00:01 =====> 100% | 6.67K
00:00:00 ==>
```

```
In [29]: M For i in OutputCollect['clusters']['models']:
        print(i['solid'])

1_38_3
1_48_3
1_41_4
1_26_2
1_34_3
1_40_1
1_40_6
1_41_4
1_45_5
1_46_2
1_46_4
1_46_5
1_47_9
1_48_8
1_49_5
1_50_1
1_52_2
1_54_4
1_54_5
```

```
In [30]: M OutputCollect['allSolutions']

Out[30]: [{"1_14_7",
            "1_21_5",
            "1_26_2",
            "1_29_2",
            "1_33_2",
            "1_34_3",
            "1_40_1",
            "1_40_6",
            "1_41_4",
            "1_45_5",
            "1_46_2",
            "1_46_4",
            "1_46_5",
            "1_47_9",
            "1_48_8",
            "1_49_5",
            "1_50_1",
            "1_52_2",
            "1_54_4",
            "1_54_5"}]
```

jupyter robyn_python_notebook (autosaved)

```
File Edit View Insert Cell Kernel Widgets Help
```

Not Trusted Python 3 (ipykernel) L

```
In [31]: M If get_response
allocator = robyn_api('robyn_allocator',payload=payload)

>>> Running budget allocator for model in 1_solid ...
Automatically picked data_range = 1st-4th
Time Window: 2018-12-10T00:00:00Z to 2018-12-10T00:00:00Z (4 weeks)
Featuring Top 1000 (https://api.robyn.ai/v1/models/2018-12-10T00:00:00Z/1000) as realized best ross-gps

In [32]: M plot_individualize(OutputModelName='ross',largestOfTopallocated,graphType='all'allocate,'new_store'(1000, 1500))

Budget Allocation Change for Model ID: 1_S_3
Add Time: 1500 - set to 1500, start at 1500, end at 1500, step = 100, offset = 100, color = red, size = 100, shape = circle, stroke-width = 2px
Data Range: 2018-12-10T00:00:00Z to 2018-12-10T00:00:00Z (4 weeks)
Total Budget Optimization Result

New Response
```

The figure displays the results of a budget allocation optimization process. At the top, there's a summary of the total budget optimization result, showing a comparison between 'Before' and 'After' states across various channels (TVC, PTV, etc.). Below this, a bar chart titled 'Budget Allocation per Channel' shows the distribution of budget across different channels, comparing 'Before' and 'After' states. The 'After' state shows a more optimized allocation. Further down, there are several line graphs showing 'Unallocated Response Curve for Selected Allocation Period' for different time windows and allocation periods. These graphs illustrate how the unallocated response changes over time and across different allocation periods.



Status: OSの違いによる注意点

プロセス	OS	
	Windows	Linux & Mac
APIの起動・モニタリング・停止	CMDコマンド (Jupyter Notebook上でコマンドを実行する)	Pythonのsubprocessを利用
ロギング	ログファイルに出力	Jupyter Notebook上のコンソールに出力

