

株式会社松尾研究所 シニアデータサイエンティスト からあげ氏 Qiita Conference 2025 Autumn 11/6(木)に登壇決定!



概要

普段私たちは天気予報を頼りに、予定や服装を決めたりしますが

「1週間先の天気予報って実際当たってるの？」

— 7 — (Continued from page 6)

このため、丞相がばねくらい当たっているかを確認する

このため、予報がどれくらい当たっているかを確認するのが難しいです。

そこで僕は、Google Apps Script(GAS)を利用して、気象庁の天気予報を定期的に取得して保存するプログラムを組み、天気予報の移り変わりをモニタリングすることを考えました。

本記事はその方法を解説します。

## 大まかな流れ

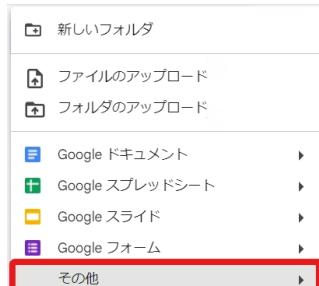
1. 気象庁の天気予報をJSONで取得する（GAS）
  2. 取得したJSON内の情報をスプレッドシートに書き込む（GAS）
  3. スプレッドシートから値を取得し、チニタリング結果を可視化する（Python）

#### ① GASプロジェクトの作成

自分のGoogle driveの任意の場所を開いて、左タブの「新規」をクリック。



「その他」にカーソルを合わせて…



[6] 1.4.2.1.1. 本办法自二〇一〇年十一月一日起施行或之日实施。



## ② 天気予報をJSONで取得するコードを書く

以下のようなURLにリクエストを投げると、天気予報の情報が入ったJSONが返ってきます。

<https://www.jma.go.jp/bosai/forecast/data/forecast/エリアコード.json>

エリアコードは、例えばこちらのサイトを参考にしてみてください。  
京都府の場合は「**260000**」がエリアコードです。

✓ エリアコード一覧をJSONで取得する方法もあります。

▶ 詳しい手順は【ここ】をクリック

GASでリクエストを送るコードは以下の通りです。説明はコメントアウトで書きました。

```
// JSONを取得+保存するだけのコード
functiongetJSON() {
  var now = new Date(); // 取得日時(JSONのファイル名に使う)
  var date = Utilities.formatDate(now, "GMT+9", "yyyyMMddHHmmss"); // ファイル名用に
  var weather_all = {};// 全地域の予報データを格納する辞書

  // 全地域のエリアコード
  var areaCodeList = ["011000", "012000", "016000", "013000", "014100", "015000", "017000",
  // 各地域ごとにJSONを取得するループ
  for (var j=0; j<areaCodeList.length; j++) {
    // リクエスト先URL
    const weatherUrl = 'https://www.jma.go.jp/bosai/forecast/data/forecast/' + areaCodeList[j];
    // リクエスト送信してレスポンスを受ける
    var weatherj = UrlFetchApp.fetch(weatherUrl);
    // JSONをパース
    var weather = JSON.parse(weatherj);
    // さっそく定義したweather_allに格納
    weather_all[areaCodeList[j]] = weather;
  }

  // ----- weather_allをJSONとして保存(任意) -----
  var folderId = "xxxxxxxxxxxxxxxxxxxxxx"; // 保存先のフォルダID
  var weatherString = JSON.stringify(weather_all);
  var fileName = date + ".json";
  var folder = DriveApp.getFolderById(folderId);
  var file = folder.createFile(fileName, weatherString);
  Logger.log("JSON file created and saved: " + file.getUrl());
}
```

✓ フォルダIDとは？

Google driveの任意のフォルダを開き、URLを見てみると下のような形式になっていると思います。

<https://drive.google.com/drive/u/0/folders/xxxxxxxxxxxxxxxxxxxxxx>

このURLのxxxxxxxxxxxxxxxxxxxxxxに相当する部分が**フォルダID**です。

試しに実行してみましょう。

GASを実行するには、上の方にある「実行」をクリックするか「Ctrl+R」を押します。

すると、保存先に {yyyyMMddHHmmss}.json というフォーマットの名前でJSONファイルができるていると思います。この中に天気予報データが入っています。

※ 保存はしなくとも①以降に支障はありません。保存たくない人はコメントアウトしておいてください。

## ③ JSONファイルの中身をスプレッドシートに書き込む

②ではJSONを取得するコードを書きましたが、これを解析しやすい形でスプレッドシートに書き込む処理を行っていきます。

GASからスプレッドシートを操作するには1つだけ準備が必要です。

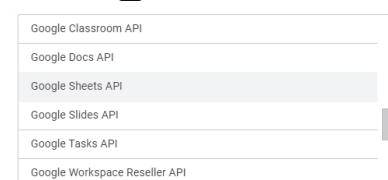
GASプロジェクトを開き、左タブにある「サービス」の「+」マークをクリックします。



サービス一覧の中から「Google Sheets API」を選択肢「追加」をクリックします。  
これで準備完了です。

サービスを追加

高度なタスクを実施するために、プロジェクトでGoogleの拡張サービスを利用することができます。



データを書き込む用のスプレッドシートは先に作っておいてください。  
そのうえで、以下のコードをGASで実行してみてください。

```
// JSONを取得+スプシに書き込むコード
function writeJSON() {
  // スプレッドシートとの接続
  const SHEET_ID = "xxxxxxxxxxxxxxxxxxxxxx"; // スpreadsheet ID
  const SHEET_NAME = "シート1"; // シート名
  var spreadsheet = SpreadsheetApp.openById(SHEET_ID);
```

```

var sheet = spreadsheet.getSheetByName(SHEET_NAME);

var areaCodeList = ["011000","012000","016000","013000","014100","015000","017000",
for (var j=0; j<areaCodeList.length; j++) {
  // JSONを取得 (getJSON()と同様)
  console.log(areaCodeList[j]);
  const weatherUrl = 'https://www.jma.go.jp/bosai/forecast/data/forecast/' + areaCodeList[j];
  var weatherj = UrlFetchApp.fetch(weatherUrl);
  var weather = JSON.parse(weatherj);
}

// JSON解析
var sub = weather[1]['timeSeries']; // 選択予報のみを取り出す
var reportDatetime = weather[1]['reportDatetime']; // 予報更新日時
var dates = sub[0]['timeDefines']; // 予報対象日時

var df = [];
for (var i=0; i<dates.length; i++) {
  var line = [];
  line.push(reportDatetime); // 予報更新日時
  line.push(areaCodeList[i]); // エリアコード
  line.push(dates[i]); // 予報対象日時
  line.push(sub[0]['areas'][0]['weatherCodes'][i]); // 天気コード
  line.push(sub[0]['areas'][0]['pops'][i]); // 降水確率
  line.push(sub[0]['areas'][0]['reliabilities'][i]); // 予報の信頼度
  line.push(sub[1]['areas'][0]['tempMin'][i]); // 最低気温
  line.push(sub[1]['areas'][0]['tempMinUpper'][i]); // 最低気温の信頼上限
  line.push(sub[1]['areas'][0]['tempMinLower'][i]); // 最低気温の信頼下限
  line.push(sub[1]['areas'][0]['tempMax'][i]); // 最高気温
  line.push(sub[1]['areas'][0]['tempMaxUpper'][i]); // 最高気温の信頼上限
  line.push(sub[1]['areas'][0]['tempMaxLower'][i]); // 最高気温の信頼下限
  df.push(line);
}
sheet.getRange(sheet.getLastRow()+1, 1, df.length, line.length).setValues(df);
Utilities.sleep(100); // 100ミリ秒待機
}
}

```

☑ スプレッドシートIDとは？  
作成したスプレッドシートを開き、URLを見てみると下のような形式になっていると思います。

<https://docs.google.com/spreadsheets/d/xxxxxxxxxxxxxxxxxxxx/edit#gid=0>

このURLのxxxxxxxxxxxxxxxxxxxxに相当する部分がスプレッドシートIDです。

実行してみると、取得した天気予報がスプレッドシートに書き込まれていると思います。

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	reportDatetime	areaCode	targetDate	weather	pop	relat	tempL	tempH	tempM	tempL	tempH	tempM	tempMaxLower
2	2023-09-09T17:00:00+09:00	11000	2023-09-10T00:00:00+09:00		201								
3	2023-09-09T17:00:00+09:00	11000	2023-09-10T00:00:00+09:00	212	60		18	20	17	24	26	22	
4	2023-09-09T17:00:00+09:00	11000	2023-09-12T00:00:00+09:00	202	60 C		20	22	18	25	28	24	

左から

- 予報更新日時（毎日11時と17時に更新されるようです）
- エリアコード
- 予報対象日
- 天気コード（晴れ・曇り・雨・晴れ時々曇りなどを表すコード）
- 降水確率
- 予報の信頼度（A・B・C）
- 最低気温
- 最低気温の信頼上限
- 最低気温の信頼下限
- 最高気温
- 最高気温の信頼上限
- 最高気温の信頼下限

を表しています。

## ④ GAS定期実行の設定をする

気象庁の天気予報は（週間予報の場合）毎日11時と17時に2回更新されますが、逐一手動で実行するわけにもいきません。

そこで、定期実行の設定を行います。

GASプロジェクトを開き、左端の時計マークをクリックします。



すると「トリガー」の設定画面になるので、左下の「トリガーを追加」をクリックします。

下のような画面が出てくるので、下記画像のように設定してください。

実行する関数を選択 writeJSON	エラー通知設定 毎日通知を受け取る
デプロイ時に実行 Head	
イベントのソースを選択 時間主導型	
時間ベースのトリガーのタイプを選択 日付ベースのタイマー	
時刻を選択 午後5時～6時 (GMT+09:00)	

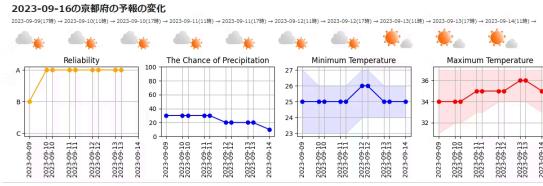
ここで設定したのは「毎日17時～18時の間のどこかでwriteJSONを実行してください」ということです。  
同様に「午後5時～6時」の部分を「午前11時～12時」に変えたトリガーを作成してください。

これで定期実行の設定が完了です。

## ⑤ Pythonでスプレッドシートを読み込み、予報の推移をモニタリングする

データがたまってきたら、結果を可視化します。  
結果の可視化にはPythonを使用し、今回はColaboratoryで実行します。

最終的なイメージは下記画像のような感じです。



上には天気アイコンが表示され、2023年9月16日の京都府の予報がどう移り変わっているかが分かります。その下には「予報の信頼度」「降水確率」「最低気温」「最高気温」が折れ線グラフで表示されます。塗りつぶしている部分は信頼区間を表します。

これをPythonで実装していきます。使用するライブラリは以下のとおりです。

```
from google.colab import auth
import gspread
from google.auth import default

import os
import json
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from IPython.display import HTML
```

### 5-1.スプレッドシートから値を取得する

Pythonでスプレッドシートと連携する方法（認証する方法）はいくつかあるのですが、今回はそのうちの1つを紹介します。

次のコードを実行すると取得できます。  
「認証」の部分でポップアップが出てきますが、全部「承認する」でOKです。

```
# 認証
auth.authenticate_user()
creds, _ = default()
gc = gspread.authorize(creds)

# データを取得
spreadsheet_id = 'xxxxxxxxxxxxxxxxxxxxxx'
worksheet_name = 'シート1'
url = f'https://docs.google.com/spreadsheets/d/{spreadsheet_id}/'
ss = gc.open_by_url(url)
st = ss.worksheet(worksheet_name)
data = st.get_all_values()
```

上記の方法だと、ノートブックを開くごとに認証の許可をもとめられてやや面倒です。  
認証を省略したい人はこちらの記事を参考にしてみてください。

【もう迷わない】Pythonでスプレッドシートに読み書きする初期設定まとめ  
<https://tanuhack.com/operate-spreadsheet/>

これでdataにスプレッドシートの値が格納されます。

### 5-2.お天気アイコンを表示する

お天気アイコンは以下のようなURLから取得することができます。画像形式はSVGです。

```
https://www.jma.go.jp/bosai/forecast/img/天気コード.svg
```

天気コードは、スプレッドシートの列名で言うと「weatherCode」に相当します。  
これに対応するURLにアクセスして画像をとってくればいいわけです。

しかし、そう簡単にはいきません。天気コード「203」の画像を見ようとする…

```
https://www.jma.go.jp/bosai/forecast/img/203.svg
https://www.jma.go.jp
```

「404 Not Found」が返ってきます。実は、天気コードと画像ファイル名はピミョーに合致していないのです…。

なので天気コードとファイル名の対応表を取得する必要があります。

その手順を以下に示します。

1. 気象庁の「<https://www.jma.go.jp/bosai/forecast/>」にアクセス

2. F12キーを押してデベロッパーツールを開く

3. 上のタブから「Console」を選択



4. コンソールに Forecast.Const.TELOPS と打ち込み、Enterキー

↓ Forecast.Const.TELOPS

```

5. weatherCodeを一覧か表示されるので、これをコピー (Copy objectをクリック)
> ForecastConst.TELOPS
< [100: Array(5), 101: Array(5), 102: Array(5),
  00: Array(5), 097: Array(5), 105: Array(5), 11
  0: Array(5), 111: Array(5), 112: Array(5), 11
  3: Array(5), 114: Array(5), 115: Array(5), 11
  6: Array(5), 117: Array(5), 118: Array(5), 11
  9: Array(5), 120: Array(5), 119: Array(5), 11
  2: Array(5), 123: Copy object
  5: Array(5), 126: Array(5),
  8: Array(5), 130:
  2: Array(5), 140: Store object as global variable
  0: Array(5), 181:
  1: Array(5), 202: Expand recursively
  4: Array(5), 203:
  7: Array(5), 208: Collapse children
  0: Array(5), 211: Array(5), 212: Array(5), 21
  3: Array(5), 214: Array(5), 215: Array(5), 21
  6: Array(5), 217: Array(5), 218: Array(5), 21
]

```

6. ローカルでテキストファイルを新規作成し、貼り付け

```

{
  "100": [
    "100.svg",
    "500.svg",
    "100",
    "晴",
    "CLEAR"
  ],
  "101": [
    "101.svg",
    "501.svg",
    "100",
    "晴時々曇",
    "PARTLY CLOUDY"
  ],
}

```

7. 拡張子を .json に変更して保存し (weatherCode.json)、Google driveにアップロード

このJSONを参照すれば、天気コードと画像の対応関係がわかります。

これでお天気アイコンを表示する準備は整いました。Pythonコードを書きましょう。

```

# カレントディレクトリ(必要に応じて変更)
PATH = './'

# スプレッドシートから取得したデータをpd.DataFrameに変換
df = pd.DataFrame(data[1:], columns=data[0])
df = df.replace('', np.nan)
df = df.astype({'pop':'float','tempMin':'float','tempMinUpper':'float','tempMinLower':'float'})

# 天気コードと画像の対応表(JSON)を読み込む
with open(PATH + '/weatherCode.json', 'r') as f:
    weatherCode_dict = json.load(f)

# areaCodeと地域名の対応表(JSON)を読み込む(→手順②を参照)
with open(PATH + '/areaCode.json', 'r') as f:
    areaCode_dict = json.load(f)

def get_weather_icon(weatherCode):
    """weatherCode入力すると、天気アイコンを表示するタグを返す関数"""
    img_url = f'https://www.jma.go.jp/bosai/forecast/img/{weatherCode_dict[str(weatherCode)]}'
    img = f'{date[:10]} ({date[11:13]})  
'
display(HTML(date_html))

# 天気アイコン
icon = ''
for weatherCode in sub['weatherCode'].values:
    if weatherCode != np.nan:
        icon += get_weather_icon(weatherCode)
icon = f'<div class="image-container">{icon}</div>'
display(HTML(icon))

```

可視化はHTMLを用いました。HTMLを文字列型として書いて、それをHTML()にツッコめば表示してくれます。

### 5-3. 信頼度・降水確率・最低/最高気温のグラフを表示する

単純にmatplotlibでプロットするだけです。  
デザインはご自分の好みで調整してください。

```

# 信頼度
plt.figure(figsize=(15,2))
plt.subplot(1,4,1)
plt.plot(pd.to_datetime(sub['reportDatetime']), sub['reliability'].map({'A':3,'B':2}),
         color='blue', marker='o')
plt.xticks(pd.to_datetime(sub['reportDatetime']), rotation=90)
plt.yticks(ticks=[1,2,3], labels=['C','B','A'])
plt.ylim(0,3.1)
plt.title('Reliability')
plt.grid()

# 降水確率
plt.subplot(1,4,2)
plt.plot(pd.to_datetime(sub['reportDatetime']), sub['pop'], color='blue', marker='o')
plt.xticks(pd.to_datetime(sub['reportDatetime']), rotation=90)
plt.ylim(0,100)
plt.title('The Chance of Precipitation')
plt.grid()

# 最低気温
plt.subplot(1,4,3)
plt.plot(pd.to_datetime(sub['reportDatetime']), sub['tempMin'], color='blue', marker='o')
plt.fill_between(pd.to_datetime(sub['reportDatetime']), sub['tempMinLower'], sub['tempMinUpper'],
                color='blue', alpha=0.2)
plt.xticks(pd.to_datetime(sub['reportDatetime']), rotation=90)
plt.title('Minimum Temperature')
plt.grid()

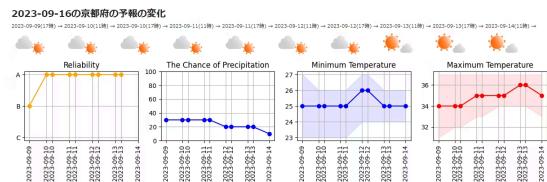
# 最高気温
plt.subplot(1,4,4)
plt.plot(pd.to_datetime(sub['reportDatetime']), sub['tempMax'], color='red', marker='o')
plt.fill_between(pd.to_datetime(sub['reportDatetime']), sub['tempMaxLower'], sub['tempMaxUpper'],
                color='red', alpha=0.2)
plt.xticks(pd.to_datetime(sub['reportDatetime']), rotation=90)
plt.title('Maximum Temperature')
plt.grid()

plt.show()

```

## ⑥ 完成！

できましたー！



予想最高/最低気温の推移を見てみると面白いですね。

日が経つにつれて信頼区間(?)がだんだん狭まっていることが見てとれます。

## 全体のコード

全体のコードも載せておきます（コメントアウトなし）。

トグルをクリックすると開きます。

▶ GAS

▶ Python

GitHub - matsuda-tkm/monitoring-jma-forecast: 気象庁の天気予報をGAS+Pythonで追跡する  
<https://github.com>

## 参考サイト

- 「Pythonと気象庁の天気予報JSONを使って全国の天気予報を取得してみる」  
リンク：<https://www.gis-py.com/entry/weather-json>
- 「気象庁JSONファイルにあるweatherCode一覧」  
リンク：<https://www.3z.jp/blog/web-develop/weather-code-list/>
- 【もう迷わない】Pythonでスプレッドシートに読み書きする初期設定まとめ  
リンク：<https://tanuhack.com/operate-spreadsheet/>

♡ 7 ○ 5 □ 0

X F B

新規登録して、もっと便利にQiitaを使ってみよう

- あなたにマッチした記事をお届けします
- 便利な情報をあとで効率的に読み返せます
- ダークテーマを利用できます

[ログインすると使える機能について](#)

[新規登録](#) [ログイン](#)



@matsuda\_tkmのピックアップ記事

【GAS】気象庁の予報がどう変わっていくかをモニタリングしてみた

♡ 7 Python JSON GAS 気象庁 2023年09月14日

入力した歌詞がどのアーティストっぽいかを判定しその根拠も提示するAIを作った

♡ 9 Python 自然言語処理 CNN 作ってみた XAI 2023年11月18日

【論文解説】GraphCast：GNNで気象予報（概要編）

♡ 9 論文読み DeepMind 気象データ GNN 2023年12月20日



@matsuda\_tkm (TAKUMI MATSUDA)

Python・機械学習に関する記事を書いています。

[フォロー](#) [リツイート](#) [引用](#) [X](#) [RSS](#)

## ♪ 今日のトレンド記事

✿ @Sakai\_path

2025年10月21日

レガシーC#コード対比集（昔こう→今こう）

C# .NET リファクタリング .NETFramework レガシーコード

♡ 277

□

『生成AI開発の珍プレー好プレー大賞！（珍プレー多め）』

@TooMe in KDDIアジャイル開発センター株式会社

2025年10月23日

GitHub Copilotを使っている人は全員"copilot-instrucions.md"を作成してください

githubcopilot copilot-instrucions.md

♡ 144

□

✿ @kunitomo926 in Kaoエンジニアコミュニティβ

2025年10月22日

Claude x MCPで社内DBに話しかけてみた

TypeScript MCP AzureSQLDatabase Claude MCPサーバー

♡ 41

□

✿ @kazaf01 in 株式会社クラベス

2025年10月21日

Webサーバーサイド言語としてのRustについて紹介

Rust WEBサーバー axum Dioxus

♡ 22

□



@sun\_22  
2025年10月21日

## Dockerを使ってワードプレスの環境構築をする

WordPress Docker wp-cli docker-compose  
♡ 28



[トレンド一覧を見る](#)

関連記事 Recommended by LOGLY



Cloud Functions with Puppeteer + Google Apps Script...  
by howdy39



GASでQiita APIを叩いて結果をGoogleスプレッドシートに自動入力する手順を詳しくメモ  
し...  
by kurararara



GoogleAppScriptを使ってDiscordに朝と夕方に天気予報を流す  
by proudust



Qiitaの指標をGASで集計してNotion Chartsで可視化する  
by ry02132



「人が辞めない組織」に必要なたった一つの条件。社員の離職を止めたリーダーの「気づき」  
PR ビズヒント



指示待ち組織を蘇らせたリーダー。社員の主体性を引き出すために貰いたい「二つの行動」  
PR ビズヒント

☞ この記事は以下の記事からリンクされています

气象庁の天気予報追跡Webアプリを作った 2023年12月09日

## コメント

この記事にコメントはありません。

いいね以上の気持ちをコメントで

[ログイン](#)

[新規登録](#)

## Qiita Conference 2025 Autumn 11月5日(水)~7日(金)開催！



Qiita Conferenceは、AI時代のエンジニアに贈るQiita最大規模の  
テックカンファレンスです！  
基調講演ゲスト(仮称略)  
placere、牛尾 剛、Esteban Suarez、和田 卓人、seyaa、ミノ  
動、市谷 肇啓、からあげ、岩瀬 義昌、まつもとゆきひろ、みの  
るん and more...

[イベント詳細を見る →](#)

## 記事投稿キャンペーン開催中



「Vibe Coding」で創る、インストール不要の生成AI  
「Skywork」での開発体験を投稿しよう！  
2025年9月26. Fri. - 10.30.Thu.

[詳細を見る](#)



「生成AI開発の珍プレー好プレー大賞！」(珍プレー多  
め)」  
2025年9月30.Tue. - 11.10.Mon.

[詳細を見る](#)

[すべて見る](#)

Qiita

How developers code is here.



© 2021-2025 Qiita Inc.

### ガイドとヘルプ

[About](#)

[利用規約](#)

[プライバシーポリシー](#)

[ガイドライン](#)

[メディアキット](#)

[ご意見・ご要望](#)

[ヘルプ](#)

[広告掲載](#)

### コンテンツ

[リリースノート](#)

[公式イベント](#)

[公式コラム](#)

[アドベントカレンダー](#)

[Qiita Tech Festa](#)

[Qiita 表彰プログラム](#)

[エンジニア白書](#)

[API](#)

### 公式アカウント

[Qiita \(キータ\) 公式](#)

[Qiita マイルストーン](#)

[Qiita 人気の投稿](#)

[Facebook](#)

[YouTube](#)

[Qiita ポッドキャスト](#)

[API](#)

### Qiita 関連サービス

[Qiita Team](#)

[Qiita Zine](#)

[Qiita 公式ショッピング](#)

[Qiita Blog](#)

[ニュースリリース](#)

### 運営

[運営会社](#)

[採用情報](#)

[Qiita Team](#)