Suomen Parhaat Boulderit 2018: Create Boulders Final

March 17, 2018

Google Maps JavaScript API key. See https://developers.google.com/maps/documentation/javascript/get-api-key (https://developers.google.com/maps/documentation/javascript/get-api-key)

In [1]:
```
GOOGLE_MAPS_JAVASCRIPT_API_KEY = "YOUR_API_KEY"
```

Import required modules

In [2]:
```
import json
import time
import numpy as np
import pandas as pd
from geopy.geocoders import GoogleV3
from geopy.exc import GeocoderQueryError, GeocoderQuotaExceeded
```

Load the datafile `spb2018_-_cleaned.csv`, which contains the form responses to the **Suomen Parhaat Boulderit 2018** survey.

In [3]:
```
# Load cleaned dataset
spb2018_df = pd.read_csv("data/survey_-_cleaned.csv")

# Drop duplicates (exclude the Timestamp column from comparisons)
spb2018_df = spb2018_df.drop_duplicates(subset=spb2018_df.columns.values
.tolist()[1:])
spb2018_df.head()
```

Out[3]:

	Timestamp	Suostumus	Ikä	Sukupuoli	Pituus (cm)	Vaikein *ulkona* kiipeämäsi boulder	Boulderin nimi	Olen lähettänyt (kiivennyt) kyseisen boulderin	Ku
0	12/17/2017 18:36:32	Kyllä	35 - 39	Mies	184	7A+	Muistipeli	Kyllä	erila kolm pitkä
1	12/17/2017 18:48:44	Kyllä	30 - 34	Mies	180	7B	Voodoo	Kyllä	Tiuk hänk
2	12/17/2017 18:49:04	Kyllä	30 - 34	Mies	180	8A	One love	Kyllä	Dyn
3	12/17/2017 18:53:03	Kyllä	25 - 29	Mies	190	7A	Bitch slap and male pinch	Kyllä	kuur mieh
4	12/17/2017 19:10:14	Kyllä	25 - 29	Mies	180	8A	Maitomies	Kyllä	Herk klas

5 rows × 29 columns

Load the datafile `boulders_-_prefilled.csv`, which contains manually added details of each voted boulder.

In [4]:
```
boulder_details_df = pd.read_csv("data/boulders_-_prefilled.csv", index_
col="Name")
boulder_details_df.head()
```

Out[4]:

Name	Grade	InFinland	Crag	ApproximateCoordinates	Coc
360 Kickflip	6A	Yes	Djupviksgrottorna	No	60.401326,19
Alcoholocaust	7B	Yes	Uusi Sipoo	No	60.311802,25
Analstacia	7A+	Yes	Rokokallio	No	60.484207,24
Baby Voodoo	6A+	Yes	Djupviksgrottorna	No	60.397900,19
Bitch slap and male pinch	6C	Yes	Itäinen Runsori	No	63.042414,21

Add column *VotedBy*

In [5]:
```
"""
# Simpler but slower (appr. four times) implementation
# 533 ms ± 95.2 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
def add_column_votedby(column_name="VotedBy"):
    # Gender mappings from Finnish to English
    gender_dict = {
        "Mies": "Male",
        "Nainen": "Female"
    }

    # Iterate over boulders
    for index, row in boulder_details_df.iterrows():
        boulder_name = index
        gender_s = spb2018_df.loc[(spb2018_df["Boulderin nimi"] == bould
er_name) | (spb2018_df["Boulderin nimi.1"] == boulder_name) | (spb2018_d
f["Boulderin nimi.2"] == boulder_name), "Sukupuoli"]
        boulder_details_df.loc[boulder_name, column_name] = gender_dict
[gender_s.iloc[0]] if gender_s.nunique() == 1 else "Both"
"""
"""
# More complex but faster (appr. four times) implementation
# 136 ms ± 5.42 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)
def add_column_votedby(column_name="VotedBy"):
    # Initialize the new column
    boulder_details_df[column_name] = ""

    # Gender mappings from Finnish to English
    gender_dict = {
        "Mies": "Male",
```

```
        "Nainen": "Female"
    }

    def update_genders(gender, boulder_names):
        for boulder_name in boulder_names:
            previous_gender = boulder_details_df.loc[boulder_name, colum
n_name]
            if previous_gender == "" or previous_gender == gender:
                boulder_details_df.loc[boulder_name, column_name] = gend
er
            else:
                boulder_details_df.loc[boulder_name, column_name] = "Bot
h"

    # Iterate over form responses
    for index, row in spb2018_df.iterrows():
        gender = gender_dict[row["Sukupuoli"]]
        boulder_names = [row["Boulderin nimi"], row["Boulderin nimi.1"],
 row["Boulderin nimi.2"]]
        boulder_names = [boulder_name for boulder_name in boulder_names
 if pd.notnull(boulder_name)]
        update_genders(gender, boulder_names)
"""
# Typical implementation
# 430 ms ± 78.2 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
def add_column_votedby(column_name="VotedBy"):
    # Gender mappings from Finnish to English
    gender_dict = {
        "Mies": "Male",
        "Nainen": "Female"
    }

    def set_voted_by(row):
        boulder_name = row.name
        gender_s = spb2018_df.loc[(spb2018_df["Boulderin nimi"] == bould
er_name) | (spb2018_df["Boulderin nimi.1"] == boulder_name) | (spb2018_d
f["Boulderin nimi.2"] == boulder_name), "Sukupuoli"]
        return gender_dict[gender_s.iloc[0]] if gender_s.nunique() == 1
else "Both"

    boulder_details_df[column_name] = boulder_details_df.apply(set_voted
_by, axis=1)

add_column_votedby()
boulder_details_df.head()
```

Out[5]:

Name	Grade	InFinland	Crag	ApproximateCoordinates	Coc
360 Kickflip	6A	Yes	Djupviksgrottorna	No	60.401326,19
Alcoholocaust	7B	Yes	Uusi Sipoo	No	60.311802,25
Analstacia	7A+	Yes	Rokokallio	No	60.484207,24
Baby Voodoo	6A+	Yes	Djupviksgrottorna	No	60.397900,19
Bitch slap and male pinch	6C	Yes	Itäinen Runsori	No	63.042414,21

Add column *Votes*.

In [6]:
```
def add_column_votes(column_name="Votes"):
    boulder_name_columns = [spb2018_df["Boulderin nimi"], spb2018_df["Bo
ulderin nimi.1"], spb2018_df["Boulderin nimi.2"]]
    all_voted_boulders_s = pd.concat(boulder_name_columns, ignore_index=
True).dropna()
    boulder_votes_s = all_voted_boulders_s.value_counts()
    boulder_details_df[column_name] = boulder_votes_s

add_column_votes()
boulder_details_df.sort_values(by=["Votes"], ascending=[False]).loc[boul
der_details_df["Votes"] >= 3]
```

Out[6]:

Name	Grade	InFinland	Crag	ApproximateCoordinates	
Kun Jari koskee	7A	Yes	Uusi Sipoo	No	60.31180
Ruoska	7A	Yes	Tahmela boulder park	No	61.49687
Kaljala	6A+	Yes	Taljala	No	61.07631
Supermartikainen	7C	Yes	Djupviksgrottorna	No	60.40135
Puerto Rico	6C	Yes	Fågelberget	No	60.41302

Add columns *Latitude* and *Longitude*.

In [7]:
```
def add_columns_latitude_and_longitude(column_names=["Latitude", "Longit
ude"]):
    boulder_details_df[[column_names[0], column_names[1]]] = boulder_det
ails_df["Coordinates"].str.split(",", expand=True).astype(float)

add_columns_latitude_and_longitude()
boulder_details_df.head()
```

Out[7]:

	Grade	InFinland	Crag	ApproximateCoordinates	Coc
Name					
360 Kickflip	6A	Yes	Djupviksgrottorna	No	60.401326,19
Alcoholocaust	7B	Yes	Uusi Sipoo	No	60.311802,25
Analstacia	7A+	Yes	Rokokallio	No	60.484207,24
Baby Voodoo	6A+	Yes	Djupviksgrottorna	No	60.397900,19
Bitch slap and male pinch	6C	Yes	Itäinen Runsori	No	63.042414,21

Add column *GradeNumeric*.

In [8]:
```
def add_column_gradenumeric(column_name="GradeNumeric"):
    # Grade mappings from Font to numeric
    grade_dict = {
        "?":    0,
        "1":    1,
        "2":    2,
        "3":    3,
        "4":    4,
        "4+":   5,
        "5":    6,
        "5+":   7,
        "6A":   8,
        "6A+":  9,
        "6B":  10,
        "6B+": 11,
        "6C":  12,
        "6C+": 13,
        "7A":  14,
        "7A+": 15,
        "7B":  16,
        "7B+": 17,
        "7C":  18,
        "7C+": 19,
        "8A":  20,
        "8A+": 21,
        "8B":  22,
        "8B+": 23,
        "8C":  24,
        "8C+": 25,
        "9A":  26
    }

    boulder_details_df[column_name] = boulder_details_df.apply(lambda ro
w: str(grade_dict[row["Grade"]]) if pd.notnull(row["Grade"]) else np.nan
, axis=1)
    boulder_details_df[column_name] = boulder_details_df[column_name].as
type(int)

add_column_gradenumeric()
boulder_details_df.head()
```

Out[8]:

	Grade	InFinland	Crag	ApproximateCoordinates	Coc
Name					
360 Kickflip	6A	Yes	Djupviksgrottorna	No	60.401326,19
Alcoholocaust	7B	Yes	Uusi Sipoo	No	60.311802,25
Analstacia	7A+	Yes	Rokokallio	No	60.484207,24
Baby Voodoo	6A+	Yes	Djupviksgrottorna	No	60.397900,19
Bitch slap and male pinch	6C	Yes	Itäinen Runsori	No	63.042414,21

Add column *Adjectives*

In [9]:
```
def add_column_adjectives(column_name="Adjectives"):
    def set_adjectives(row):
        boulder_name = row.name
        adjectives1_s = spb2018_df.loc[(spb2018_df["Boulderin nimi"] ==
boulder_name), "Kuvaile boulderia kolmella (3) adjektiivilla"]
        adjectives2_s = spb2018_df.loc[(spb2018_df["Boulderin nimi.1"] =
= boulder_name), "Kuvaile boulderia kolmella (3) adjektiivilla.1"]
        adjectives3_s = spb2018_df.loc[(spb2018_df["Boulderin nimi.2"] =
= boulder_name), "Kuvaile boulderia kolmella (3) adjektiivilla.2"]
        adjectives_s = adjectives1_s.append(adjectives2_s).append(adject
ives3_s)
        adjectives = ",".join(adjectives_s)
        # Clean adjectives
        adjectives = ",".join(sorted(list(set([adjective.strip().lower()
 for adjective in adjectives.split(",")]))))
        return adjectives

    boulder_details_df[column_name] = boulder_details_df.apply(set_adjec
tives, axis=1)

add_column_adjectives()
boulder_details_df.head()
```

Out[9]:

	Grade	InFinland	Crag	ApproximateCoordinates	Coc
Name					
360 Kickflip	6A	Yes	Djupviksgrottorna	No	60.401326,19
Alcoholocaust	7B	Yes	Uusi Sipoo	No	60.311802,25
Analstacia	7A+	Yes	Rokokallio	No	60.484207,24
Baby Voodoo	6A+	Yes	Djupviksgrottorna	No	60.397900,19
Bitch slap and male pinch	6C	Yes	Itäinen Runsori	No	63.042414,21

Add column *MainHoldTypes*

In [10]:
```
def add_column_main_hold_types(column_name="MainHoldTypes"):
    def set_main_hold_types(row):
        boulder_name = row.name
        main_hold_types1_s = spb2018_df.loc[(spb2018_df["Boulderin nimi"
] == boulder_name), "Boulderin pääotetyypit"]
        main_hold_types2_s = spb2018_df.loc[(spb2018_df["Boulderin nimi.
1"] == boulder_name), "Boulderin pääotetyypit.1"]
        main_hold_types3_s = spb2018_df.loc[(spb2018_df["Boulderin nimi.
2"] == boulder_name), "Boulderin pääotetyypit.2"]
        main_hold_types_s = main_hold_types1_s.append(main_hold_types2_s
).append(main_hold_types3_s)
        main_hold_types = ",".join(main_hold_types_s)
        # Clean main_hold_types
        main_hold_types = ",".join(sorted(list(set([main_hold_type.strip
().lower() for main_hold_type in main_hold_types.split(",")])))
        return main_hold_types

    boulder_details_df[column_name] = boulder_details_df.apply(set_main_
hold_types, axis=1)

add_column_main_hold_types()
boulder_details_df.head()
```

Out[10]:

	Grade	InFinland	Crag	ApproximateCoordinates	Coc
Name					
360 Kickflip	6A	Yes	Djupviksgrottorna	No	60.401326,19
Alcoholocaust	7B	Yes	Uusi Sipoo	No	60.311802,25
Analstacia	7A+	Yes	Rokokallio	No	60.484207,24
Baby Voodoo	6A+	Yes	Djupviksgrottorna	No	60.397900,19
Bitch slap and male pinch	6C	Yes	Itäinen Runsori	No	63.042414,2

Add column *MainProfiles*

In [11]:
```
def add_column_main_profiles(column_name="MainProfiles"):
    def set_main_profiles(row):
        boulder_name = row.name
        main_profiles1_s = spb2018_df.loc[(spb2018_df["Boulderin nimi"]
== boulder_name), "Boulderin pääprofiilit"]
        main_profiles2_s = spb2018_df.loc[(spb2018_df["Boulderin nimi.1"
] == boulder_name), "Boulderin pääprofiilit.1"]
        main_profiles3_s = spb2018_df.loc[(spb2018_df["Boulderin nimi.2"
] == boulder_name), "Boulderin pääprofiilit.2"]
        main_profiles_s = main_profiles1_s.append(main_profiles2_s).appe
nd(main_profiles3_s)
        main_profiles = ",".join(main_profiles_s)
        # Clean main_profiles
        main_profiles = ",".join(sorted(list(set([main_profile.strip().l
ower() for main_profile in main_profiles.split(",")]))))
        return main_profiles

    boulder_details_df[column_name] = boulder_details_df.apply(set_main_
profiles, axis=1)

add_column_main_profiles()
boulder_details_df.head()
```

Out[11]:

Name	Grade	InFinland	Crag	ApproximateCoordinates	Coc
360 Kickflip	6A	Yes	Djupviksgrottorna	No	60.401326,19
Alcoholocaust	7B	Yes	Uusi Sipoo	No	60.311802,25
Analstacia	7A+	Yes	Rokokallio	No	60.484207,24
Baby Voodoo	6A+	Yes	Djupviksgrottorna	No	60.397900,19
Bitch slap and male pinch	6C	Yes	Itäinen Runsori	No	63.042414,2

Add column *MainSkillsNeeded*

In [12]:
```
def add_column_main_skills_needed(column_name="MainSkillsNeeded"):
    def set_main_skills_needed(row):
        boulder_name = row.name
        main_skills_needed1_s = spb2018_df.loc[(spb2018_df["Boulderin ni
mi"] == boulder_name), "Boulderin kiipeämiseen vaadittavat pääkyvyt"]
        main_skills_needed2_s = spb2018_df.loc[(spb2018_df["Boulderin ni
mi.1"] == boulder_name), "Boulderin kiipeämiseen vaadittavat pääkyvyt.1"
]
        main_skills_needed3_s = spb2018_df.loc[(spb2018_df["Boulderin ni
mi.2"] == boulder_name), "Boulderin kiipeämiseen vaadittavat pääkyvyt.2"
]
        main_skills_needed_s = main_skills_needed1_s.append(main_skills_
needed2_s).append(main_skills_needed3_s)
        main_skills_needed = ",".join(main_skills_needed_s)
        # Clean main_skills_needed
        main_skills_needed = ",".join(sorted(list(set([main_skill_needed
.strip().lower() for main_skill_needed in main_skills_needed.split(","
)]))))
        return main_skills_needed

    boulder_details_df[column_name] = boulder_details_df.apply(set_main_
skills_needed, axis=1)

add_column_main_skills_needed()
boulder_details_df.head()
```

Out[12]:

Name	Grade	InFinland	Crag	ApproximateCoordinates	Coc
360 Kickflip	6A	Yes	Djupviksgrottorna	No	60.401326,19
Alcoholocaust	7B	Yes	Uusi Sipoo	No	60.311802,25
Analstacia	7A+	Yes	Rokokallio	No	60.484207,24
Baby Voodoo	6A+	Yes	Djupviksgrottorna	No	60.397900,19
Bitch slap and male pinch	6C	Yes	Itäinen Runsori	No	63.042414,21

Add column *Comments*

In [13]:
```
def add_column_comments(column_name="Comments"):
    def set_comments(row):
        boulder_name = row.name
        comments1_s = spb2018_df.loc[(spb2018_df["Boulderin nimi"] == bo
ulder_name), "Kuvaile boulderia omin sanoin (vapaaehtoinen)"]
        comments2_s = spb2018_df.loc[(spb2018_df["Boulderin nimi.1"] ==
boulder_name), "Kuvaile boulderia omin sanoin (vapaaehtoinen).1"]
        comments3_s = spb2018_df.loc[(spb2018_df["Boulderin nimi.2"] ==
boulder_name), "Kuvaile boulderia omin sanoin (vapaaehtoinen).2"]
        comments_s = comments1_s.append(comments2_s).append(comments3_s)
        comments = []
        for index, value in comments_s.iteritems():
            if pd.notnull(value):
                comments.append(value.strip())
        return ",".join("\"{}\"".format(comment) for comment in comments
)

    boulder_details_df[column_name] = boulder_details_df.apply(set_comme
nts, axis=1)

add_column_comments()
boulder_details_df.head()
```

Out[13]:

	Grade	InFinland	Crag	ApproximateCoordinates	Coc
Name					
360 Kickflip	6A	Yes	Djupviksgrottorna	No	60.401326,19
Alcoholocaust	7B	Yes	Uusi Sipoo	No	60.311802,25
Analstacia	7A+	Yes	Rokokallio	No	60.484207,24
Baby Voodoo	6A+	Yes	Djupviksgrottorna	No	60.397900,19
Bitch slap and male pinch	6C	Yes	Itäinen Runsori	No	63.042414,21

Add columns *AreaLevel1*, *AreaLevel2*, and *AreaLevel3*

In [14]:
```
def add_columns_arealevel1_arealevel2_and_arealevel3(column_names=["Area
Level1", "AreaLevel2", "AreaLevel3"]):
    boulder_details_df.drop(columns=[column_names[0], column_names[1], c
olumn_names[2]], inplace=True, errors="ignore")
    geolocator = GoogleV3(api_key=GOOGLE_MAPS_JAVASCRIPT_API_KEY)

    def extract_administrative_area_levels(location_results, approximate
Location, area_levels_dict):
        # List of location result types that we are interested in
        location_result_types = ["administrative_area_level_1", "adminis
trative_area_level_2", "administrative_area_level_3"]

        # Iterate over location results
        for location_result in location_results:
            location_result_json = location_result.raw
            # Extract data only from those location results that we are
 interested in
            if any(location_result_type in location_result_json["types"]
 for location_result_type in location_result_types):
                # Extract location result type
                location_result_type = location_result_json["types"][0]
                # Iterate over address components
                for address_component in location_result_json["address_c
omponents"]:
                    # Extract data only from the matched location result
 type
                    if location_result_type in address_component["types"
]:
                        # Extract the name of the administrative area le
vel 1
                        if location_result_type == location_result_types
[0]:
                            area_levels_dict["AreaLevel1"] = address_com
ponent["long_name"]
                        # Extract the name of the administrative area le
vel 2
                        if location_result_type == location_result_types
[1] and approximateLocation == "No":
                            area_levels_dict["AreaLevel2"] = address_com
ponent["long_name"]
                        # Extract the name of the administrative area le
vel 3
                        if location_result_type == location_result_types
[2] and approximateLocation == "No":
                            area_levels_dict["AreaLevel3"] = address_com
ponent["long_name"]
        return area_levels_dict

    def get_area_levels(row):
        # Area levels template
        area_levels_dict = {
            column_names[0]: "",
```

```
            column_names[1]: "",
            column_names[2]: ""
        }

        geocoded = False
        while geocoded is not True:
            # Reverse geocode coordinates
            try:
                location_results = geolocator.reverse(row["Coordinates"
], language="fi")
                area_levels_dict = extract_administrative_area_levels(lo
cation_results, row["ApproximateCoordinates"], area_levels_dict)
                geocoded = True
            except GeocoderQueryError as gqe:
                print("Geocoding error with {}: {}".format(row.name, str
(gqe)))
                print("Skipping {}".format(row.name))
                geocoded = True
            except GeocoderQuotaExceeded as gqe:
                print("Geocoding quota exceeded: {}".format(str(gqe)))
                print("Backing off for a bit")
                time.sleep(30 * 60) # sleep for 30 minutes
                print("Back in action")

        return pd.Series(area_levels_dict)

    boulder_area_levels_df = boulder_details_df[["Coordinates", "Approxi
mateCoordinates"]].apply(get_area_levels, axis=1)
    return pd.merge(boulder_details_df, boulder_area_levels_df, how="out
er", left_index=True, right_index=True)

boulder_details_df = add_columns_arealevel1_arealevel2_and_arealevel3()
boulder_details_df.head()
```

Out[14]:

Name	Grade	InFinland	Crag	ApproximateCoordinates	Coc
360 Kickflip	6A	Yes	Djupviksgrottorna	No	60.401326,19
Alcoholocaust	7B	Yes	Uusi Sipoo	No	60.311802,25
Analstacia	7A+	Yes	Rokokallio	No	60.484207,24
Baby Voodoo	6A+	Yes	Djupviksgrottorna	No	60.397900,19
Bitch slap and male pinch	6C	Yes	Itäinen Runsori	No	63.042414,21

5 rows × 21 columns

Create boulders final file `boulders_-_final.csv`.

In [15]:
```
def create_boulders_final():
    boulder_details_reset_df = boulder_details_df.reset_index()
    boulder_details_reset_df = boulder_details_reset_df[["Votes", "Voted
By", "Name", "Grade", "GradeNumeric", "InFinland", "AreaLevel1", "AreaLe
vel2", "AreaLevel3", "Crag", "ApproximateCoordinates", "Coordinates", "L
atitude", "Longitude", "Url27crags", "UrlVideo", "UrlStory", "MainProfil
es", "MainHoldTypes", "MainSkillsNeeded", "Adjectives", "Comments"]]
    boulder_details_reset_df = boulder_details_reset_df.sort_values(by=[
"Votes", "GradeNumeric", "Name"], ascending=[False, False, True])
    boulder_details_reset_df.to_csv("data/boulders_-_final.csv", index=F
alse)

create_boulders_final()
```