Space Science and Social Media: Automating Science Communication on Twitter

Maia Williams
maiaw@bgsu.edu

Follow this and additional works at: https://scholarworks.bgsu.edu/honorsprojects

Part of the Astrophysics and Astronomy Commons, Computer Sciences Commons, and the Social Media Commons

How does access to this work benefit you? Let us know!

Repository Citation
https://scholarworks.bgsu.edu/honorsprojects/620

This work is brought to you for free and open access by the Student Scholarship at ScholarWorks@BGSU. It has been accepted for inclusion in Honors Projects by an authorized administrator of ScholarWorks@BGSU.
SPACE SCIENCE AND SOCIAL MEDIA: AUTOMATING SCIENCE COMMUNICATION
ON TWITTER

MAIA WILLIAMS

HONORS PROJECT

Submitted to the Honors College at Bowling Green State University in partial fulfillment of the
requirements for graduation with
UNIVERSITY HONORS
8.4.2021

Dr. Andrew Layden, Department of Physics and Astronomy, Advisor
Dr. Eric Mandell, Department of Physics and Astronomy, Advisor
Dr. Louisa Ha, School of Media and Communication, Advisor
1. Literature Review

Although science literacy has had many definitions historically, it is often described as the understanding and knowledge of scientific concepts, processes, and reasoning. Liu argues that instead of viewing science literacy as a goal to achieve by the end of schooling, it should be a lifelong process of learning Liu calls “science and the public.” Science and the public bridges formal classroom learning with informal education and advocates for the training of science professionals to better communicate with the public (Liu, 2009). If the goal of science in the public is to encourage lifelong learning, social media could provide a way to consistently expose the public to science.

Similarly, another study suggests that social media can spark interest in students when formal astronomy education is lacking. A group of high school students in Portugal reports that their curriculum fails to teach them about astronomy as well as how to interpret and discuss scientific findings. Despite not actively seeking out science information, seeing information on social media sometimes encourages them to research further. In this way, social media can prompt students to interact with astronomy topics on a deeper level (Anjos & Carvalho, 2020).

Numerous studies have looked at social media as a way of improving science literacy among the public. One such study found that young adults in their mid-twenties use the Internet to research science more than other topics such as health, celebrity news, and political news (Hargittai et al., 2018). Twitter, a microblogging social media site, remains a popular destination for this content. Many prolific scientists use Twitter for outreach and education with the benefit of possibly attracting grants and research assistants (You, 2014). For younger students, posts made by the National Aeronautics and Space Administration (NASA) on Twitter have the potential to engage novices and aid inquiry and critical thinking skills. The wide variety of media
available in the links in each tweet can engage youth more than traditional textbooks do. Furthermore, the behind-the-scenes look at NASA operations and employees makes science seem more accessible and less exclusive (Lesley, 2014).

NASA in particular makes large efforts to engage the public through its social media presence. Prior to the Great American Eclipse of 2017, NASA initiated a social media campaign to teach its followers about the event. A study that compiles tweets sent before, during, and after the eclipse concludes that NASA’s campaign increased sentiment and the use of STEM words among the public. The higher frequency of STEM words related to the eclipse possibly indicates that people became more familiar with the words and their meanings as a result of NASA’s education (Pickering et al., 2018).

The European Space Agency (ESA), the Planetary Society, SETI Institute, and the National Radio Astronomy Observatory (NRAO) are other organizations that research and communicate space science. Comprised of 22 Member States, the ESA researches the Universe and develops satellite technology while promoting collaboration among European governments and industries (European Space Agency, n.d.). The Planetary Society (n.d.) describes itself as a nonprofit organization that encourages public involvement in space exploration, technology, and advocacy. Another nonprofit is the SETI Institute, which aims to discover life and technologically advanced civilizations in the Universe (SETI Institute, n.d.). Finally, the NRAO welcomes all astronomers to observe the Universe in radio wavelengths and learn more about the formation of stars, planets, galaxies, and black holes (NRAO, n.d.).

Although many organizations use social media to educate, misinformation remains rampant and can worsen when bots infiltrate groups to further divide users. There has been some research on the presence of bots in social media discourse, especially on Twitter. One study
analyzes tweets sent during the United Nations Climate Conference in 2018 and finds that bots potentially reinforce the echo chamber environments created by opposing groups (Tyagi et al., 2020). Another source, which analyzes tweets about climate change after the announcement of the U.S. withdrawal from the Paris Climate Agreement, agrees that bots are possible drivers of online polarization (Marlow et al., 2021). Bots can negatively influence discussions on social media, although more research is needed to determine to what extent. On the contrary, they can also be used for creative and helpful purposes.

2. Research Questions

My project will investigate how social media is used to engage the public in space science and ways to automate this engagement. First, I will analyze the social media presence of space science organizations to determine how they communicate with their audiences. Then, using the programming language Python, I will build a Twitter bot that will tweet pictures taken by the NASA Mars rovers. To understand how social media can be better used for science communication and outreach, the research questions are as follows.

**Research Question 1**: To what extent do science organizations/communicators utilize social media? What level of engagement do they receive?

**Research Question 2a**: Can social media be used to automate science communication?

**Research Question 2b**: What features can be added to a Twitter bot to make it more complex?

**Research Question 2c**: Can the bot utilize Twitter for more interpersonal connections?

**Research Question 3**: Can existing NASA Application Program Interfaces (APIs) be adapted for social media?
3. Analyzing Engagement from Space Science Organizations

3.1 Methodology

Five organizations that research space science and astronomy were selected for engagement analysis on Twitter: the National Aeronautics and Space Administration (NASA), the European Space Agency (ESA), the Planetary Society, the Search for Extraterrestrial Intelligence (SETI) Institute, and the National Radio Astronomy Observatory (NRAO). These organizations can be found on Twitter @NASA, @esa, @exploreplanets, @SETIInstitute, and @TheNRAO, respectively. The initial data collection window was a two-week period from May 24 to June 6, 2021; this time frame was chosen to compensate for any major events or announcements that might receive higher engagement than usual. However, this initial collection did not provide enough tweets from the National Radio Astronomy Observatory to analyze. Therefore, for NRAO only, the collection window was extended another two weeks until June 20.

Twitter users can interact with a tweet through likes, replies, retweets, and quote tweets. Replies indicate the involvement of users with their followers, while retweets indicate the reach of a user (Leavitt et al., 2009). Likes, formally known as favorites, indicate the personal impact of a tweet on users (Bertrand et al., 2015). Quote tweets share the characteristics of replies and retweets. For each collected tweet, the number of likes, replies, retweets, and quote tweets were recorded and summed. The number of followers of each account was also recorded; this number represents the audience and potential reach of a tweet (Leavitt et al., 2009, as cited in Bertrand et al., 2015). The engagement rate for each tweet was calculated from the formula defined by Hwong et al. (2017):
\[ ER = \frac{1p + 0.75r + 0.5a}{nf}, \]

in which “\( p \) is the number of retweets, \( r \) the number of replies, \( a \) the number of likes a tweet received and \( nf \) the number of followers of the Twitter account.” This formula weighs retweets as the most valuable interaction and likes as the least valuable (Hwong et al., 2017). Although quote tweets allow users to share a tweet and add a reaction, they were counted as propagative interactions and weighted the same as retweets.

3.2 Data

The organizations often retweeted or quote tweeted other associated organizations. Only original tweets, quote tweets, and replies to original tweets were counted. For example, if @NASA retweeted a tweet by @NASAJPL, that tweet was not counted, and the interactions were not recorded. The sum of retweets, quote tweets, likes, and replies and the total interactions for each account are given in the following table.

<table>
<thead>
<tr>
<th>Organization</th>
<th>Number of Followers</th>
<th>Total Retweets</th>
<th>Total Quote Tweets</th>
<th>Total Likes</th>
<th>Total Replies</th>
<th>Total Interactions</th>
</tr>
</thead>
<tbody>
<tr>
<td>NASA</td>
<td>46,900,000</td>
<td>48,276</td>
<td>3382</td>
<td>354,819</td>
<td>6339</td>
<td>412,816</td>
</tr>
<tr>
<td>ESA</td>
<td>1,200,000</td>
<td>4507</td>
<td>867</td>
<td>26,784</td>
<td>479</td>
<td>32,637</td>
</tr>
<tr>
<td>Planetary Society</td>
<td>235,000</td>
<td>966</td>
<td>99</td>
<td>4933</td>
<td>171</td>
<td>6169</td>
</tr>
<tr>
<td>SETI Institute</td>
<td>701,200</td>
<td>1631</td>
<td>246</td>
<td>8570</td>
<td>394</td>
<td>10,841</td>
</tr>
<tr>
<td>NRAO</td>
<td>31,500</td>
<td>573</td>
<td>42</td>
<td>2210</td>
<td>36</td>
<td>2861</td>
</tr>
</tbody>
</table>

*Table 1. Engagement and interactions for space science organizations on Twitter.*
Each account’s total tweets and average tweets per day are recorded in the table below, along with the average engagement rate for the collected tweets. The average engagement rate is multiplied by 1,000,000 to make the data more readable.

<table>
<thead>
<tr>
<th>Organization</th>
<th>Total Number of Tweets</th>
<th>Average Number of Tweets per Day</th>
<th>Ave ER (interactions per million followers)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NASA</td>
<td>84</td>
<td>6</td>
<td>59.4</td>
</tr>
<tr>
<td>ESA</td>
<td>64</td>
<td>4.57</td>
<td>250</td>
</tr>
<tr>
<td>Planetary Society</td>
<td>48</td>
<td>3.43</td>
<td>324.5</td>
</tr>
<tr>
<td>SETI Institute</td>
<td>62</td>
<td>4.43</td>
<td>148.5</td>
</tr>
<tr>
<td>NRAO</td>
<td>43</td>
<td>1.54</td>
<td>1290</td>
</tr>
</tbody>
</table>

*Table 2. Average engagement rates for space science organizations.*

### 3.3 Discussion

These space science organizations utilize Twitter in varied amounts and receive different levels of engagement. The organization with the most Twitter followers is NASA, followed by ESA, SETI Institute, the Planetary Society, and NRAO. In Table 1, the sum of each type of interaction and total interactions follow the same pattern, with a larger following corresponding to more interactions. In addition, the accounts with more followers tweet more each day. NASA has about 46.9 million followers and is the most prolific account, averaging 6 tweets per day. The National Radio Astronomy Observatory has only 31,500 followers and tweets 1.54 times per day on average as shown in Figure 1.
Figure 1. Graph of the log of the number of followers versus tweets per day.

Figure 2. Graph of total interactions versus the number of followers.

Figure 1 shows the number of followers is strongly positively correlated to the average tweets per day ($R^2 = 0.9214$). Likewise, the total interactions an account receives are strongly
correlated to the number of followers ($R^2 = 0.998$); a bigger audience gives more opportunities for interaction. Other variables are likely present, but these relationships suggest that frequent posting is one way to increase engagement. If so, then using bots to automate tweeting could grow an organization’s following and interactions by allowing them to post more times each day. It is also possible that the accounts with the largest followings represent larger, more recognizable organizations; these accounts most likely have more resources and time to devote to social media.

Follower count appears to be a better predictor of tweets per day than of average engagement rate as shown in Figure 3. Despite receiving more interactions, the organizations with more followers generally have lower average engagement rates. NASA, the account with the largest following, has an average engagement rate of 59.4 interactions per million followers, whereas NRAO has an average engagement rate of 1290 interactions per million followers and the smallest following. ESA breaks this pattern by having the second most followers and the third smallest engagement rate on average. This trend makes sense with the engagement rate equation, which divides the weighted interactions by the number of followers. Since NASA has millions of followers and only thousands of interactions on each tweet, their engagement rate per tweet is very low.
Both total interactions and engagement rates among followers are valuable metrics for analyzing engagement. Furthermore, the engagement rate equation given by Hwong et. al (2017) might not be the most effective method of comparing engagement between accounts with different amounts of followers. This equation may be more useful for analyzing tweets from the same account or accounts with similar reach. The equation also provides more insight when determining how features of a tweet affect engagement as the study by Hwong et. al (2017) does. These applications would allow researchers to investigate what factors beyond follower count can increase engagement rates. Future research could develop a model to weigh the value of engagement rate against the sum of interactions.

Twitter rounds the number of likes above 10,000 to the hundreds place, the number of followers above 1,000 to the hundreds place, and the number of followers above 1,000,000 to the hundred thousands place. The significant figures for total interactions and average engagement
rate reflect this rounding. Given the interactions for each tweet were recorded manually, the sample size was limited. Further research could compare engagement between more accounts with a larger sample of tweets.

4. Building the Twitter Bot

4.1 Methodology

The Twitter bot was written in Python using PyCharm CE, an integrated development environment for writing and running code. The following libraries need to be installed in PyCharm and imported into the Python file for the bot to run.

Random: randomly generates numbers within a given range.
Requests and urllib.request: send HTTP requests and retrieve URLs.
Os: interacts with a computer’s operating system to save files.
Nasapy: a Python wrapper for accessing the NASA API.
Tweepy: a Python library for accessing the Twitter API.

```python
import tweepy
import random
import requests
import urllib.request
import os
import nasapy
```

To access NASA APIs, an API key can be generated by signing up at https://api.nasa.gov/. This project utilizes the Mars Rover Photos API. The rover photos API was chosen because it is one of the few NASA APIs that provides images, which can make a tweet more appealing. Mars is also particularly interesting to the public and the scientific community
due to its climate history, potential to harbor life, and accessibility compared to other planets (National Aeronautics and Space Administration [NASA], n.d.).

To access the Twitter API, one must apply for a developer account. Once approved, access tokens can be generated by creating an app in the developer portal. Four keys are necessary to utilize the Twitter API: an API consumer key, API consumer secret key, access token, and access secret token. These keys should be kept confidential and can be regenerated if needed.

In the first few lines of code for the bot, the NASA API key is stored in a variable, and the Twitter keys are stored in a dictionary. Then, the tweepy library authenticates the Twitter API request with the consumer and access keys (Vachhani, n.d.).

```python
auth = tweepy.OAuthHandler(
    twitter_auth_keys["consumer_key"],
    twitter_auth_keys["consumer_secret"]
)
auth.set_access_token(
    twitter_auth_keys["access_token"],
    twitter_auth_keys["access_token_secret"]
)
api = tweepy.API(auth)
```

The bot works by randomly choosing one of four NASA Mars Rovers – Spirit, Opportunity, Curiosity, or Perseverance – to request photos from. Similarly, the script randomly picks a Martian sol to query by. The NASA API returns a list of dictionaries; each dictionary contains a link to an image, the name of the rover and camera that captured the image, the Earth date and Martian sol on which the image was taken, and other related information about the rover. Figures 4 and 5 show the information returned by the API.
Figure 4. Sample of NASA API response in JSON format.

```json
{
  "photos": [
    {
      "id": 102693,
      "sol": 1000,
      "camera": {
        "id": 20,
        "name": "FHAZ",
        "rover_id": 5,
        "full_name": "Front Hazard Avoidance Camera"
      },
      "earth_date": "2015-05-30",
      "rover": {
        "id": 5,
        "name": "Curiosity",
        "landing_date": "2012-08-06",
        "launch_date": "2011-11-26",
        "status": "active"
      }
    }
  ]
}
```

Figure 5. Sample of NASA API response in JSON Pretty Print format.

The bot picks an entry from this list and prints the rover’s name, the Earth date, the camera name, and the image URL. Images are then saved to a directory in Python. Using the nasapy library allows the bot to query the NASA API in fewer lines of code. However, this
library has not been updated to give photos from Perseverance. Therefore, an “if-else” statement was created so the bot can access photos from all four rovers. Finally, the bot is hosted on the cloud website PythonAnywhere, which runs the file at a scheduled time (Iyanu, 2018). This process results in a bot that tweets a photo of the Martian surface once a day from the Twitter handle @marsmosaicbot.

4.2 Spirit, Opportunity, and Curiosity

This part of the bot is based on a program utilizing NASA’s Astronomy Picture of the Day API (Shukla, 2020). To fetch images taken by Spirit, Opportunity, and Curiosity, the nasapy library assigns the API output to a variable called “roverpic”. These rovers have been operational on Mars for at least 2000 sols, so the bot picks a Martian sol between 0 and 2000. The “camera” variable is randomly assigned one of three values that all three rovers share: front hazard avoidance camera (“FHAZ”), rear hazard avoidance camera (“RHAZ”), or navigation camera (“NAVCAM”) (Schlegel, 2019).

```
if rover == "curiosity" or rover == "opportunity" or rover == "spirit":
    sol_1 = random.randrange(2001)
    nasa = nasapy.Nasa(key=api_key)
    camera_1 = random.choice(['FHAZ', 'RHAZ', 'NAVCAM'])
    roverpic = nasa.mars_rover(sol=sol_1, camera=camera_1, rover=rover)
```

The urllib.request library is used to query the API. Image URLs are stored in a key called “img_src” inside each dictionary. If the length of the “roverpic” list is 0, then no image was taken on the chosen sol. In this case, the bot tweets a message saying there is no image available from that rover’s camera on that sol. This message is shown in Figure 6.

```
if len(roverpic) == 0:
```
When the length of the list is not 0, then at least one image was taken by the chosen rover and camera on the chosen sol; the bot will tweet a randomly chosen image from the list and a message containing information about it. As shown in Figure 7, the pictures often contain parts of the rover that captured them.

```python
else:
    random_pic = random.choice(roverpic)
    index = roverpic.index(random_pic)
    urllib.request.urlretrieve(roverpic[index]['img_src'],
    filename=os.path.join(image_dir, title))
    image = api.media_upload(filename=os.path.join(image_dir, title))
    tweet = "Taken by " + roverpic[index]['rover']['name'] + " on " +
    roverpic[index]['earth_date'] + " with " + \
    roverpic[index]['camera']['full_name']
    status = api.update_status(status=tweet,
    media_ids=[image.media_id])
```

Figure 6. Tweet from the bot when no image is available.
Figure 7. Tweet from the bot when an image from Spirit, Opportunity, or Curiosity is available.

4.3 Perseverance

Images taken by Perseverance are fetched using a function. This function is based on one written by Sonawane (2020) and manually calls the API with a URL containing the rover’s name, camera, and the Martian sol. This URL is given in the Mars Rover Photos API repository on GitHub (Cerami 2015). As the latest rover, Perseverance has spent far less time on Mars. Thus, the bot randomly picks a sol between 0 and 120 to show photos from. Perseverance has many more cameras than the previous rovers. Therefore, the bot picks between the left navigation camera (“navcam_left”), the two mast cameras (“mcz_right” and “mcz_left”), and the right rear hazard avoidance camera (“rear_hazcam_right”). Using the requests library to retrieve the URL, the API response is assigned to a variable called “response”.

The “img_src” keys are stored inside a dictionary called “photos”, which in turn is stored inside a list called “response”. Once again, the script chooses an image dictionary and retrieves the image URL. The if-else statement that determines what the bot tweets is nearly the same as the one used for the other rovers. For Perseverance, the bot checks the length of the list “response[“photos”]” and tweets if an image is available from the chosen camera and sol. One of these tweets is given in Figure 8.
4.4 Latest Photos from Perseverance: Replying to Mentions

To retrieve the latest images taken by the Perseverance rover, a function like the one described in section 4.3 was created. This function queries the Rover Photos API using the “latest photos” endpoint and the name of the rover.

```
1 rover = "perseverance"
2 def fetchpercylatest():
3     mrp_url = "https://api.nasa.gov/mars-photos/api/v1/rovers/perseverance/latest_photos?"
4     mrp_url = mrp_url + "api_key=" + api_key
5     response = requests.get(mrp_url).json()
```

Unlike the random image generator, this function returns a dictionary called “latest photos” inside the list of dictionaries called “response”. Images should always be available.
through this endpoint, so it is not necessary to check the length of the list. The script requests the image URL for the first image in the list.

Users can mention the bot by writing @marsmosaicbot in a tweet; the Twitter API mentions function lists these mentions so the bot can reply to the most recent one. If there is a new mention, the user’s screen name and the id number of the tweet are stored in variables.

```python
def checkmentions():
    global last_mention
    mentions = api.mentions_timeline()
    most_recent_mention = mentions[0]
    if most_recent_mention != last_mention:
        user = most_recent_mention.user.screen_name
        tweet_id = most_recent_mention.id_str
```

The bot then tweets the image as a reply to the mention, along with a description of what Earth date the image was taken on. Finally, the bot stores the mention id number to compare to more recent mentions the next time the script runs (Davis & Eaton, 2016).
4.5 *Latest Photos from Perseverance: Direct Messaging*

Direct messaging works the same as replying to mentions but instead uses the API function for listing direct messages. This function returns messages both sent and received by the bot (Roesslein, 2020). The id of the user who sends the bot a message is stored in a variable. To avoid the bot messaging itself, the script compares the sender and recipient ids of the most recent message to ensure they are not the same.

```python
1 def checkmessages():
2     global last_message
3     messages = api.list_direct_messages()
```
If the ids are different, the bot sends the first image from the Perseverance latest photos list and a message saying what Earth date it was taken on.

![Bot's reply to a direct message.](image)

*Figure 10. The bot’s reply to a direct message.*

One limitation of the direct message feature is that message requests from users that the bot does not follow must be accepted manually.

4.6 Tweeting Video and Sound
One of the additional functions planned for the bot was the ability to tweet videos and sound bites. These videos would be taken by Perseverance, the only rover outfitted with video cameras. However, the Mars Rover Photos API only returns images. Another NASA API, the Image and Video Library, provides videos related to the Perseverance rover. Some of the videos are taken by Perseverance, while most are informational videos and podcasts about Perseverance and Mars. There does not appear to be a way to separate videos from Perseverance from videos about Perseverance when querying the API. Therefore, it does not seem possible for the bot to tweet videos and sound bites taken by the Mars rovers at this time. Future research could investigate how to tweet these informational videos as well as images from the rovers.

5. Conclusion

Space science organizations use social media to varying degrees, as shown by the sample of tweets from five organizations. NASA’s Twitter account has the most followers out of the five accounts by far and receives the most interactions on its tweets. However, larger accounts receive lower average engagement rates. Accounts that tweet more each day have more followers, and accounts with more followers receive more interactions in total; these trends support the use of automated posting as one way to generate more interactions. Furthermore, the equation used to calculate engagement rate might be better suited for analyzing the effect of variables besides follower count. Both total interactions and engagement rate can be considered valuable metrics that a bot could aim to improve.

This project found that it is possible to use social media to somewhat automate science communication. The Twitter bot created was able to tweet images taken by all four available NASA Mars rovers at a scheduled time once a day. The bot was also able to include captions in
the tweet that explain the source of the image and the Earth date on which it was taken. To show that NASA Application Program Interfaces can be adapted for social media, the bot retrieves the pictures by querying the NASA Mars Rover Photos API with a randomly chosen sol and rover. In addition, the bot can use Twitter for interpersonal connections by replying to mentions and direct messages. Further research is needed to allow the bot to tweet videos and sound bites from the rovers using the NASA APIs or another method. This code and explanation can serve as a guide for experimenting with NASA APIs and building a Twitter bot with introductory knowledge of Python.

6. Acknowledgements

The author of this paper would like to acknowledge Chris Cerami, Pratik Shukla, and Ajinkya Sonawane, who wrote the code that aided in modeling this bot.
References


European Space Agency (n.d.). *ESA facts*. https://www.esa.int/About_Us/Corporate_news/ESA_facts


7. Appendix

The code for the bot is given below with the necessary API keys omitted for confidentiality. This code can also be found on GitHub: https://github.com/maiaformars/mars-rover-twitter-bot.

7.1 Tweeting Images from Opportunity, Spirit, Curiosity, and Perseverance

```python
import tweepy
import random
import requests
import urllib.request
import os
import nasapy

api_key = "API_KEY"

twitter_auth_keys = {
    "consumer_key": "CONSUMER_KEY",
    "consumer_secret": "CONSUMER_SECRET",
    "access_token": "ACCESS_TOKEN",
    "access_token_secret": "ACCESS_TOKEN_SECRET"
}

auth = tweepy.OAuthHandler(
    twitter_auth_keys["consumer_key"],
    twitter_auth_keys["consumer_secret"]
)

auth.set_access_token(
    twitter_auth_keys["access_token"],
    twitter_auth_keys["access_token_secret"]
)

api = tweepy.API(auth)

# Checks if directory for saving images exists, creates one if not
image_dir = "./Astro_Images"

dir_res = os.path.exists(image_dir)
if dir_res is False:
    os.makedirs(image_dir)
```
# Randomly picks a rover to query by
rover = random.choice(['curiosity', 'opportunity', 'spirit', 'perseverance'])

# When Curiosity, Opportunity, or Spirit is the chosen rover
if rover == "curiosity" or rover == "opportunity" or rover == "spirit":
    # Randomly picks a Martian sol and camera to query by
    sol_1 = random.randrange(2001)
    nasa = nasapy.Nasa(key=api_key)
    camera_1 = random.choice(['FHAZ', 'RHAZ', 'NAVCAM'])
    roverpic = nasa.mars_rover(sol=sol_1, camera=camera_1, rover=rover)
    title = "sol_" + str(sol_1) + "_" + rover + "_" + camera_1 + ".jpg"
    # Tweets a message if no image is available
    if len(roverpic) == 0:
        tweet = "No image from " + rover + " " + camera_1 + " available from sol " + str(sol_1)
        status = api.update_status(status=tweet)
    # Otherwise, randomly picks an image from the list, tweets it and a description
    else:
        random_pic = random.choice(roverpic)
        index = roverpic.index(random_pic)
        urllib.request.urlretrieve(roverpic[index]['img_src'], filename=os.path.join(image_dir, title))
        image = api.media_upload(filename=os.path.join(image_dir, title))
        tweet = "Taken by " + roverpic[index]['rover']['name'] + " on " + roverpic[index]['earth_date'] + " with " + \
            roverpic[index]['camera']['full_name']
        status = api.update_status(status=tweet, media_ids=[image.media_id])

# When Perseverance is the chosen rover
else:
    def fetchpercy():
        mrp_url = "https://api.nasa.gov/mars-photos/api/v1/rovers/"
        # Randomly picks a Martian sol and camera to query by
        sol_2 = str(random.randrange(121))
        rover = "perseverance"
        camera_2 = random.choice(['navcam_left', 'mcz_right', 'mcz_left', "rear_hazcam_right"])
        mrp_url = mrp_url + rover + "/photos?api_key=" + api_key + 
            ";sol=" + sol_2
        mrp_url = mrp_url + "&camera=" + camera_2
        response = requests.get(mrp_url).json()
# Tweets a message if no image is available
if len(response['photos']) == 0:
    tweet = "No image from " + rover + " " + camera_2 + " available from sol " + str(sol_2)
    status = api.update_status(status=tweet)
    # Otherwise, randomly picks an image from the list, tweets it and a description
else:
    random_pic = random.choice(response['photos'])
    index = response['photos'].index(random_pic)
    urllib.request.urlretrieve
    response['photos'][index]['img_src'], filename=os.path.join(image_dir, 102 title))
    image = api.media_upload(filename=os.path.join(image_dir, 104 title))
    tweet = "Taken by Perseverance on " +
    response['photos'][index]['earth_date'] + " with " + \
    response['photos'][index]['camera'] ['full_name']
    status = api.update_status(status=tweet,
    media_ids=[image.media_id])
    fetchpercy()

7.2 Replying to Mentions

Lines 11-25 from section 7.1 are the same for the following code and have been omitted.

```python
import requests
import urllib.request
import os
import tweepy

api = tweepy.API(auth)

api_key = "API_KEY"
rover = "perseverance"

# Assumes the last mention was a null value
last_mention = None

# Checks for the most recent mention
def checkmentions():
```
```python
global last_mention
mentions = api.mentions_timeline()
most_recent_mention = mentions[0]
if most_recent_mention != last_mention:
    user = most_recent_mention.user.screen_name
tweet_id = most_recent_mention.id_str

# Queries the NASA API for a recent image from Perseverance

def fetchpercylatest():
    mrp_url = "https://api.nasa.gov/mars-photos/api/v1/rovers/perseverance/latest_photos?"
    mrp_url = mrp_url + "api_key=" + api_key
    response = requests.get(mrp_url).json()
    image_dir = "./Astro_Images"
    title = response["latest_photos"][0]["earth_date"] + ".jpg"
    if "img_src" in response["latest_photos"][0]:
        urllib.request.urlretrieve(response["latest_photos"][0]["img_src"],
                                    filename=os.path.join(image_dir, title))
        image = api.media_upload(filename=os.path.join(image_dir,
                                  title))
        # Replies to most recent mention with a recent image from Perseverance
        status = "@" + user + " Here's one of the latest photos taken by Perseverance on " + \ response["latest_photos"][0]["earth_date"]
        reply = api.update_status(status=status,
                                   in_reply_to_status_id=tweet_id, auto_populate_reply_metadata=True,
                                   media_ids=[image.media_id])
        print("reply sent!")
    else:
        fetchpercylatest()
else:
    print("no new mentions")

# Updates last_mention to the most recent mention
last_mention = most_recent_mention
checkmentions()
```
7.3 Direct Messaging

Lines 11-25 from section 7.1 are the same for the following code and have been omitted.

```python
import requests
import urllib.request
import os
import tweepy

api = tweepy.API(auth, wait_on_rate_limit=True,
                 wait_on_rate_limit_notify=True)

# Assumes the last message was a null value
last_message = None

api_key = "API_KEY"
rover = "perseverance"

# Checks for the most recent message
def checkmessages():
    global last_message
    messages = api.list_direct_messages()
    most_recent_message = messages[0]
    bot_id = most_recent_message.message_create['target']['recipient_id']
    sender_id = most_recent_message.message_create['sender_id']
    if most_recent_message != last_message and bot_id != sender_id:
        user_id = sender_id

        # Queries the NASA API for a recent image from Perseverance
        def fetchpercylatest():
            mrp_url = "https://api.nasa.gov/mars-photos/api/v1/rovers/perseverance/latest_photos?"
            mrp_url = mrp_url + "api_key=" + api_key
            response = requests.get(mrp_url).json()
            image_dir = "./Astro_Images"
            title = response["latest_photos"]['earth_date'] + "_
            rover + ".png"
            if "img_src" in response["latest_photos"][0]:
                urllib.request.urlretrieve(response["latest_photos"][0]["img_src"],
                                            filename=os.path.join(image_dir, title))
```
image = api.media_upload(filename=os.path.join(image_dir, title))

# Replies to most recent message with a recent image from Perseverance

text = "Here's one of the latest photos taken by Perseverance on " + \
response["latest_photos"][0]["earth_date"]
dm = api.send_direct_message(recipient_id=user_id, text=text,
attachment_type="media", attachment_media_id=image.media_id)
print("message sent!")

fetchpercylatest()

else:
    print("no new messages")

# Updates last_message to the most recent message
last_message = most_recent_message

checkmessages()