An Overview of the Haas and Farmer School of Business Official Instagram

Chris Lee[1,*]

[1]Market Games, Berkeley California
[*]chris@marketgames.io

ABSTRACT

This paper looks at the Miami Farmer School of Business and Berkeley Haas School of Business Instagram pages. Its purpose is two fold. First it will give benchmarks for an Instagram page targeting the same audience as market games. Secondly it is trail run for a new method of automated data collection. This paper finds that currently Market Games follower engagement is right on target, if not better, than other pages with the same audience. It also shows that the web scraping tool is mostly effective and will allow us to collect larger amounts of data from other accounts on Instagram.

Introduction

The last few weeks the reports have been focused on Market Games' content. These have been limited in their usefulness due to the general lack of data. Furthermore, the last few reports also lacked important context. Without a comparison group it is hard to tell whether results are good or poor. In this analysis I attempt to provide some grounding for our data and give some ways Market Games can improve its own metrics.

Apart from the functional use of this article. This also acts as an experiment on automated data collection. You as the reader might not know this but we have been struggling with efficient means of collecting data from our various sources. So far we have been doing a lot of this by hand. This means that we have had to go through and copy results for each post on various platforms. This is extremely inefficient and not scalable. Here we use a method called web scraping.

Methods

We will discuss the method of automated data acquisition first and separately than the analysis of the data.

Web Scraping

First a little background. Web scraping is the process of automating data collection on websites. The specific program was written to login to Instagram and collect various pieces of information about each post including the post link, title, likes, mentions, and hashtags. This work is not all mine and I referenced and adapted work by user "jnawjux" from github. In order to replicate this you must have a webdriver installed. If you are interested you can do so here: https://github.com/mozilla/geckodriver/releases. With this and the code below (Note: you must change the text with text as it will be unique to your local environment) the data collection should be entirely reproducible. This code was used to scrape the most recent 300 posts from an instagram account.

The Code:

The web scraper was written in python relying mostly on the tool selenium. It is dependent on consistent xpaths to elements. As a broad summary this scrolls down a profile page and compies the links to each individual post. Then it takes the list of link posts and goes to each link. While at each link it copies the type of post (photo or video), the comment, number of linkes, any hashtags, and mentions appearing in the title comment or comments section. It then takes this data and writes it to a csv.

```
import time
import re
import urllib
import pandas as pd
from selenium.webdriver.firefox.options import Options
from selenium.webdriver import Firefox
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.by import By
from selenium.webdriver.support.wait import WebDriverWait
import json


def login(browser, cred=None):
    wait = WebDriverWait(browser, 10)

    login_elem = browser.find_element_by_xpath("/html/body/div[1]/
    section/nav/div[2]/div/div/div[3]/div/span/a[1]")

    login_elem.click()

    second_page_flag = wait.until(EC.presence_of_element_located(
        (By.CLASS_NAME, "KPnG0")))  # util login page appear

    user = browser.find_element_by_name("username")

    passw = browser.find_element_by_name('password')

    if cred is None:
        usr_code = input('Username: ')
        pass_code = input('Password: ')
    else:
        usr_code = cred["id"]
        pass_code =cred["key"]

    ActionChains(browser) \
        .move_to_element(user).click() \
        .send_keys(usr_code) \
        .move_to_element(passw).click() \
        .send_keys(pass_code) \
        .perform()

    login_button_ = browser.find_element_by_xpath(
        "/html/body/div[1]/section/main/div/article/div/div[1]/div
        /form/div/div[3]/button")

    login_button_.click()

    second_page_flag = wait.until(EC.presence_of_element_located(
        (By.CLASS_NAME, "cmbtv")))  #wait until popup

    not_now_ = browser.find_element_by_xpath("/html/body/div[1]/
    section/main/div/div/div/div/button")

    not_now_.click()
```

```
    wait


def recent_post_links(username, post_count=10, cred=None):
    """
    With the input of an account page, scrape the 10 most recent posts urls
    Args:
    username: Instagram username
    post_count: default of 10, set as many or as few as you want
    Returns:
    A list with the unique url links for the most recent posts for
    the provided user
    """
    url = "https://www.instagram.com/" + username + "/"
    firefox_options = Options()
    firefox_options.add_argument("--headless")
    browser = Firefox(executable_path="~my path~")
    browser.get(url)
    login(browser, cred)
    post = 'https://www.instagram.com/p/'
    post_links = []
    while len(post_links) < post_count:
        links = [a.get_attribute('href')
                 for a in browser.find_elements_by_tag_name('a')]
        for link in links:
            if post in link and link not in post_links:
                post_links.append(link)
        scroll_down = "window.scrollTo(0, document.body.scrollHeight);"
        browser.execute_script(scroll_down)
        print("status: " + str(len(post_links)))
        time.sleep(10)
    else:
        browser.stop_client()
        return post_links[:post_count]


def find_hashtags(comment):
    """
    Find hastags used in comment and return them
    Args:
    comment: Instagram comment text
    Returns:
    a list or individual hashtags if found in comment
    """
    hashtags = re.findall('#[A-Za-z]+', comment)
    if (len(hashtags) > 1) & (len(hashtags) != 1):
        return hashtags
    elif len(hashtags) == 1:
        return hashtags[0]
    else:
        return ""


def find_mentions(comment):
```

```
    """
    Find mentions used in comment and return them
    Args:
    comment: Instagram comment text
    Returns:
    a list or individual mentions if found in comment
    """
    mentions = re.findall('@[A-Za-z]+', comment)
    if (len(mentions) > 1) & (len(mentions) != 1):
        return mentions
    elif len(mentions) == 1:
        return mentions[0]
    else:
        return ""


def insta_link_details(url):
    """
    Take a post url and return post details
    Args:
    urls: a list of urls for Instagram posts
    Returns:
    A list of dictionaries with details for each Instagram post, including link,
    post type, like/view count, age (when posted), and initial comment
    """
    firefox_options = Options()
    firefox_options.add_argument("--headless")
    browser = Firefox(executable_path="~my path~")
    browser.get(url)
    try:
        # This captures the standard like count.
        likes = browser.find_element_by_xpath(
            """/html/body/div[1]/section/main/div/div/article/
                div[3]/section[2]/div/div/button/span""").text.split()[0]
        post_type = 'photo'
    except:
        # This captures the like count for videos which is stored
        likes = browser.find_element_by_xpath(
            """/html/body/div[1]/section/main/div/div/article/
                div[3]/section[2]/div/span/span""").text.split()[0]
        post_type = 'video'
    age = browser.find_element_by_css_selector('a time').text
    try:
        comment = browser.find_element_by_xpath(
        """/html/body/div[1]/section/main/div/div[1]/article/
        div[3]/div[1]/ul/div/li/div/div/div[2]/span""").text
    except:
        comment = "NA"

    hashtags = find_hashtags(comment)
    mentions = find_mentions(comment)
    post_details = {'link': url, 'type': post_type, 'likes/views': likes,
                    'age': age, 'comment': comment, 'hashtags': hashtags,
                    'mentions': mentions}
    time.sleep(10)
```

```
    browser.close()
    return post_details


def insta_url_to_img(url, filename="insta.jpg"):
    """
    Getting the actual photo file from an Instagram url
    Args:
    url: Instagram direct post url
    filename: file name for image at url
    Returns:
    image file, saved locally
    """
    firefox_options = Options()
    firefox_options.add_argument("--headless")
    browser = Firefox(executable_path="~my path~")
    browser.get(url)
    try:
        image = browser.find_element_by_xpath(
            """/html/body/span/section/main/div/div/article/
                div[1]/div/div/div[1]/div[1]/img""").get_attribute
                ('src').split(' ')[0]
        urllib.request.urlretrieve(image, filename)
    # If image is not a photo, print notice
    except:
        print("No image")

with open("~path to credentials json~", "r") as read_file:
    cred = json.load(read_file)


urls = recent_post_links("~any instagram account~", 300, cred)
Posts = []
for link in urls:
    if len(Posts % 10 == 0):
        print("Collecting Post " + str(len(Posts) + "..."))

    Posts.append(insta_link_details(link))

pd.DataFrame(Posts).to_csv(path_or_buf="~path to save to~")
```

The Analysis

The first aim of the analysis of Farmer and Haas' Instagram accounts is to gain perspective on our data. With some of the calculations there are a few assumptions. First he data for follower count at the time of posting is not available. So in order to do some of the calculations there needed to be a current follow count. This was estimated by doing linear regression on the likes over time and using the slope of the line as an estimator for follower growth. This assumption will be accurate if the underlying like percentage of these accounts has remained unchanged over time (causing any overall changes in likes to be due to followers changing). This seems like a reasonable assumption (see figures 1 and 2).

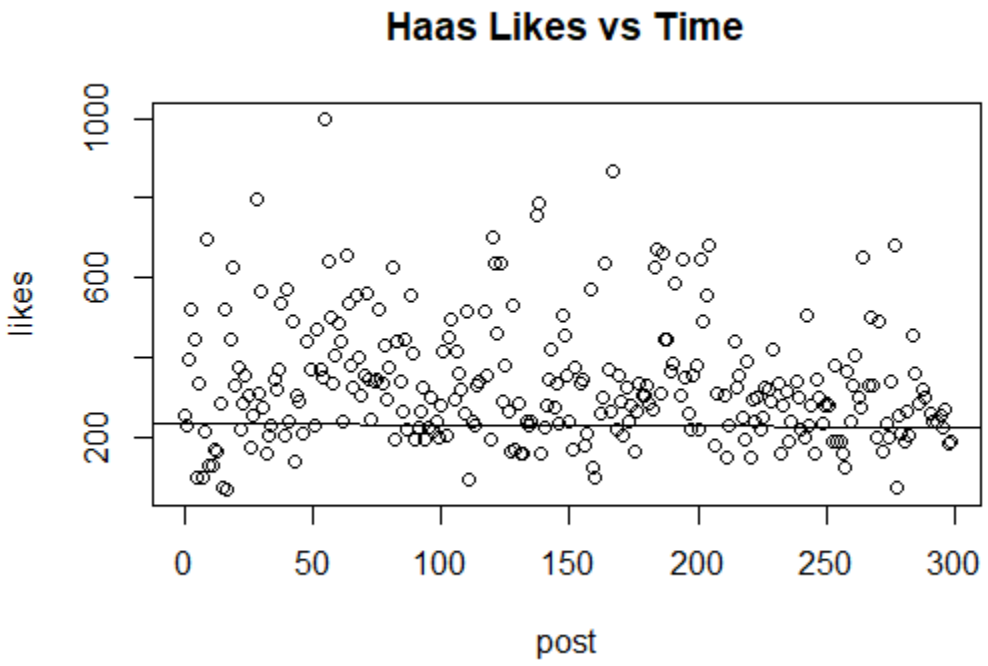

Figure 1. This shows the relationship between the post age and the likes. The black solid line is the regression line mentioned before. Note that the left side is the most recent post (number indicates posts ago)

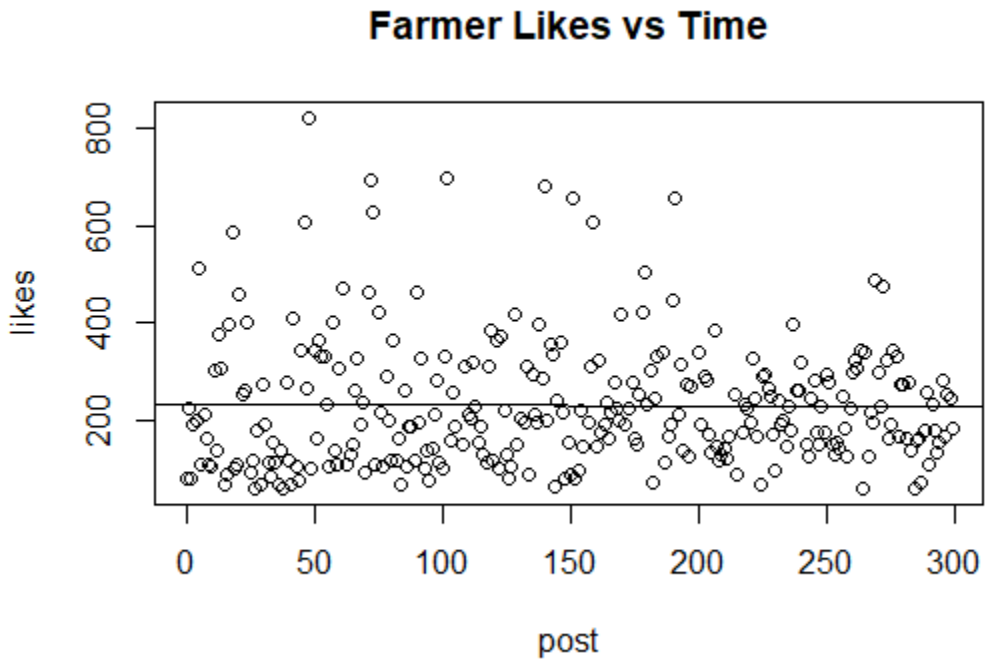

Figure 2. This shows the relationship between the post age and the likes. The black solid line is the regression line mentioned before. Note that the left side is the most recent post (number indicates posts ago)

Definitions

Here we will define the definitions that will be used to contextualize current Market Games behavior and performance.

1. Like rate =
$$\frac{1}{300}\sum_{k=0}^{300}\frac{\text{Likes}}{\text{Current followers estimate}}$$
2. Like trend: This is given by the slope of the line $a+bX=Y$. Let likes $=y$, post number $=x$, and $n=300$. Then
$$a=\frac{(\sum y)(\sum x^2)-(\sum x)(\sum xy)}{n(\sum x^2)-(\sum x)^2}$$
$$b=\frac{n(\sum xy)-(\sum x)(\sum y)}{n(\sum x^2)-(\sum x)^2}$$
3. Post frequency =
$$\frac{1}{300}(\text{Time post 1}-\text{Time post 300})$$
4. Mean caption length =
$$\frac{1}{300}\sum_{k=0}^{300}\text{Caption length of post k}$$
5. Mean hashtags per post =
$$\frac{1}{300}\sum_{k=0}^{300}\text{Number of hashtags on post k}$$

6. Mean mentions per post =

$$\frac{1}{k = 300} \sum_{k=0}^{300} \text{Number of mentions on post k}$$

All calculations are done with R with the aid of packages dplyr, ggplot2, stringr, tidyr, and gridExtra.

Results

Like the methods section we will discuss both the data acquisition strategy and the data.

Web Scraping

The whole process of writing this tool took longer than expected. There were some problems with locating some of the elements as Instagram has a modern style web site. This meant that many of the elements are loaded with javascript. The workaround used for this has some unintended side effects. Mainly this does not have the benefit of working silently. While this can be annoying (it causes windows to pop up in front of whatever you are working on), it is good for demonstrative purposes. There was also an issue where Instagram would require a login after a certain number of pictures were viewed. I have never had to automate a login procedure so although it works currently there may be unforeseen bugs. Another result of this is in order to use it, the user must have access to the Market Games (or other Instagram account) credentials. Now the successes of the experiment. The tool scraps about one post every 11 seconds. This can definitely be pushed faster, however it should not been done without a good reason. It is generally considered good practice to be "gentle" on website by limiting requests on a website to not overly burden the page. Even at this speed the tool seems very helpful. It is fully automated from the point of start (and possible supplying login credentials) to a formatted csv file. So although it is limited in speed and it is difficult to use your computer while it is working, it still saves time and effort as it doesn't need over sight to run. As an example I started the job to collect the 300 posts from each account, went to the grocery store (leaving my computer at home) and came back to two nice csv files. I also labored to make this tool portable. It should work for any public Instagram profile or private profile if we have access to the credentials of an account that follows the private page. It should also be noted that the csv that are created upon completion do usually need some cleaning. Here is a sample workflow for using the tool.

Using the Web Scraper

1. **Dependencies** The project requires a few things to run locally. First geckodriver (see introduction or google), python 3.x, and selenium. You can see requirements in the import section of the code as well. If anybody actually intends to set up this tool let me know, there are a few quirks that you need to make sure you do that I will spare you the details of here.

2. **Adding path information** The tool requires you to enter in some information that will vary depending on how your computer is organized. This includes specifying where the csv will be saved (and the name you want to give the file), where the geckodriver file is located, and possible where your credential file is located (this one is optional).

3. **Starting the tool** I currently am running this tool in a virtual environment, I am not sure how this will work outside this environment. The general procedure will be start the program. A Firefox browser should open. Do not close it. You should see that the search bar is orange with a robot in it. This is good.

4. **Entering credentials** The webdriver should navigate to a login page. Click back to the console where the python script is running. There should be a prompt for username and password. It is also has an option to provide the credentials with using json. Once the credentials are provided the tool should begin collecting data.

5. **Data Collection** This should take about 11 seconds per post. Do not exit the browsers of close the popup pages. It will be hard to use the computer during the task.

6. **Output** The csv should appear in you desired location. The data should have the format

	X	link	type	likes.views	age	comment	hashtags	mentions
post entries								

Definitions:

- **X** The post number
- **link** Link to the post

- **type** Video or photo
- **likes.views** This will be the like count if the post is a photo or the view count if a video.
- **age** This is the age of the post. It is scraped from direct text and is not in a very usable form. It needs to be cleaned before most tools can parse it. This is because forms will vary widely. Examples formats include 6w, 3 days ago, and October 6.
- **comment** This is the caption. I apologize, in hindsight the label is confusing.
- **hashtags** A string representation of a python list of hashtags contained in the post. This also needs to be cleaned to be used. Example "['#instagram','#MarketGames']"
- **mentions** A string representation of a python list of mentions contained in the post. Same structure as hashtags, see above.

Farmer and Haas Analysis

The following table was computer using R

Overview

	Like rate	Like trend	Post frequency	Caption length	Hashtags per post	Mentions per post
Haas	0.019	+0.227	3.01	351.67	2.76	1.217
Farmer	0.036	+0.0217	0.84	265.40	3.14	1.240
Market Games	0.047	+0.0362*	4.20	458.615	1.769	1.154

* Limited Data. For a look at Market Games likes over time see figure 3.

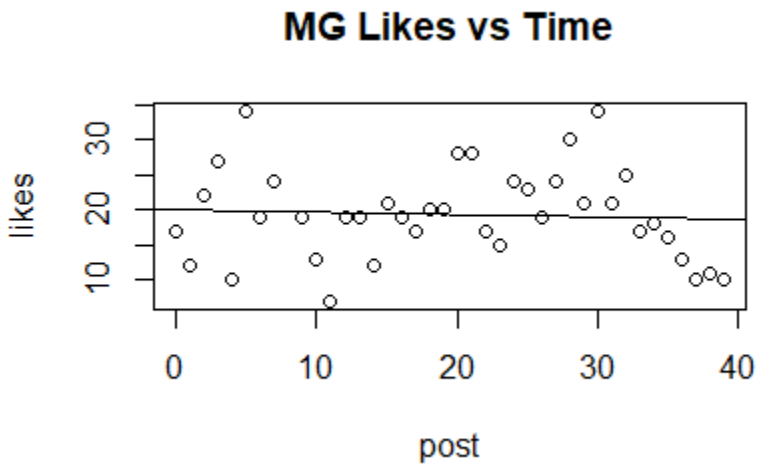

Figure 3. Notice the number of data points as well as the cyclical trend (this does not show up in linear regression). Note that the left side is the most recent post (number indicates posts ago)

Distribution of Likes

Here we can see how the number of likes are distributed. The points above are outliers. They are the ones that are the most likely to have a meaningful difference than the rest.

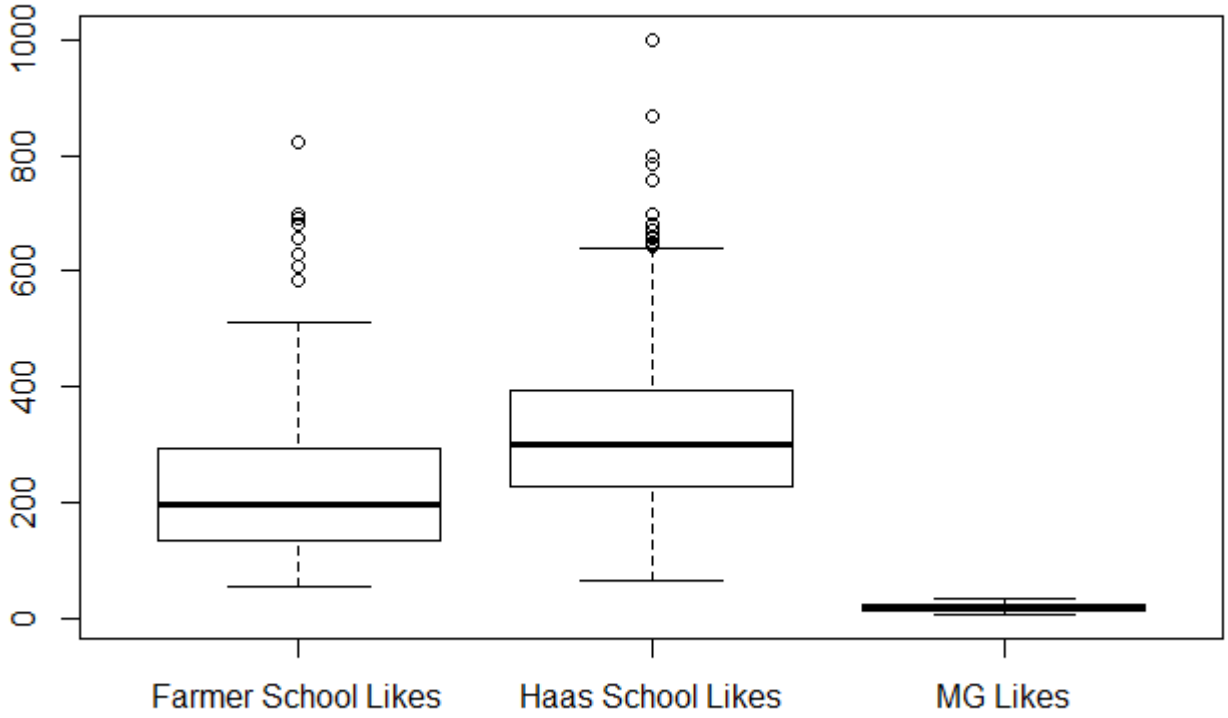

Figure 4

Outliers

For ease here are the links for the outlier posts:

Farmer

- "https://www.instagram.com/p/CGU8Rz0jamT/"
- "https://www.instagram.com/p/CFvQVcsDxDD/"
- "https://www.instagram.com/p/CFt7e-5j4ye/"
- "https://www.instagram.com/p/CFACVnHjN9j/"
- "https://www.instagram.com/p/CE9dI_uDqeq/"
- "https://www.instagram.com/p/CD_bowVDg1Z/"
- "https://www.instagram.com/p/CCD5O_kjvWw/"
- "https://www.instagram.com/p/CBNxghpjgf5/"
- "https://www.instagram.com/p/CAp4fxPjoJc/"
- "https://www.instagram.com/p/B_v5VzYD_LS/"

Haas

- "https://www.instagram.com/p/CGkx_BiD9vx/"
- "https://www.instagram.com/p/CGC8Ln9j8XH/"
- "https://www.instagram.com/p/CFhzlymjiVx/"
- "https://www.instagram.com/p/CFZtT-wDl5F/"
- "https://www.instagram.com/p/CDB1JXejDyP/"
- "https://www.instagram.com/p/CCV5-UdDsbe/"
- "https://www.instagram.com/p/CCOEhIrDArj/"
- "https://www.instagram.com/p/CAaMYRgDhKC/"
- "https://www.instagram.com/p/B_5TiHzDYkP/"
- "https://www.instagram.com/p/B_2ZAuMDCY8/"
- "https://www.instagram.com/p/B_nAEKBDwib/"
- "https://www.instagram.com/p/B_VY43Ljwis/"
- "https://www.instagram.com/p/B_SB_tKjylx/"
- "https://www.instagram.com/p/B9Rq-sng4uz/"
- "https://www.instagram.com/p/B9CPa6BA8tJ/"

Topics

Here is the most commonly used hashtags for each school. These can give a sample of what topics each tends to cover.

	hashtags	avg_likes	n
1	#haasmba	352.4722	72
2		319.9400	50
3	#berkeleyhaas	346.8800	50
4	#haasome	305.4000	40
5	#proud	332.4000	25
6	#gobears	375.8000	20
7	#classof	407.2632	19
8	#beyondyourself	319.2778	18
9	#HaasPOV	401.5333	15
10	#haasalumni	266.2857	14

Figure 5. The most common hashtags for Haas. Blank means a post without hashtags

	hashtags	avg_likes	n
1	#MyFSB	230.9562	251
2	#BeyondReady	219.7814	183
3	#LoveandHonor	239.6636	110
4	#MiamiOH	258.6286	105
5	#FYIC	211.5000	32
6	#HealthyTogether	173.6190	21
7		168.0833	12
8	#MoveInMiami	192.3333	6
9	#FallforMiami	294.6000	5
10	#InfiniteLearning	191.6000	5

Figure 6. The most common hashtags for Farmer. Blank means a post without hashtags

Discussion

Web Scraping

This was a little bit of a pain to build and it occasionally still runs into problems. However, all things considered, it does seem like it could be very useful. If the lower data quality is good enough this tool could be used to automate data collection from our Instagram with ease. It also could be applied to many different accounts for a variety of uses. One possibility that comes to mind is monitoring competitors. Furthermore this specific case does not represent the full extent of what is possible with these methods. With more time and work the data quality could be increased. More types of data could be collected (such as all the follower comments or individual accounts that like posts). Some downsides to this is it is very dependent on a specific website structure. This means that it is dependant on Instagram not changing its website structure. This also means that if I would also have to be pretty much rebuilt from scratch to use on different platform such as LinkedIn or Facebook. It is also kind of difficult for me to build. A majority of my time this past few weeks working on this report was spent building this.

Farmer and Haas Analysis

The value in the data for these two pages lies in the contextualization it provides for Market Games' own data. Assuming that our audiences are the same we can get a better idea of how the Market Games content is fairing. We can see that the average time between posts is the largest of the three account (although it has picked up recently). We also see that the captions are much longer than Farmer and Haas'. However, I think the most promising figure is the like rate. It seems that, according to the estimation method used, that the engagement for Market Games is the highest. It is also worth thinking about the topics section that contains the top hashtags. This could provide some guidance about what each school cares about and what sort of "character" each account has. This is worth thinking about for yourself but here is what comes to mind for me. Haas has a strong sense of identity of its own outside of Berkeley. Contrast this with Farmer which talks much less about the Farmer school of business and more about Miami as a whole. Haas seems both proud of itself and its community. Haas may then respond well to content that celebrates them and their school, things that is very important to Haas. While Farmer also seems to have a sense of community, the focus of it's page seems much more geared toward preparedness and learning. Farmer then may respond better to content that highlights Market Games' effectiveness at building marketable skills.

There is much more to be said about this data that could be explored further now that the web scraping tool is functional. The outlier posts could be examined to see if there is a trend. We also could test specific hypothesis about the data and what posts do well. Things relating to the strategies we are currently using to engage followers. Another thing that may be useful to explore is do some text analysis on the captions not just the hashtags. This would take some time but it could give a much clearer idea of what the character of each student population is. Now that we also have the ability to collect large numbers of data points it would be interesting to try and build a model that would predict an images likes. Depending how good we could get the model, this could be used to test new content ideas.