Project Name: Time series Competitive Analysis for Product Customer Sentiment Trends

Course: CS439 - Intro to Data Science

Names: ANNA NINASHVILI AND FABIAN RODAS INIGUEZ

NetIDs: an868, fr312

Project statement:

We developed this project with the intent to compare trends of drug discussion volume over time to see how a drug's topic volume is changing(which could be for a variety of reasons) and where competitors are headed instead.
Our project analyzes month-over-month changes in Reddit discussion volume for four Crohn's disease drugs: Humira, Stelara, Skyrizi, and Rinvoq.
Instead of examining sentiment in isolation, we focus on how volumes of specific patient-reported topics(which are inherently positive or negative from a marketing perspective) change over time. Because the reddit posts contain real patient experience, the trends reveal how patient conversations and attitudes evolve in real time. The primary goal developed during this project became: to identify whether increases in negative discussion around one drug are associated with increases in posts about discussion for switching to or starting another drug for competing companies.

Novelty and Importance:

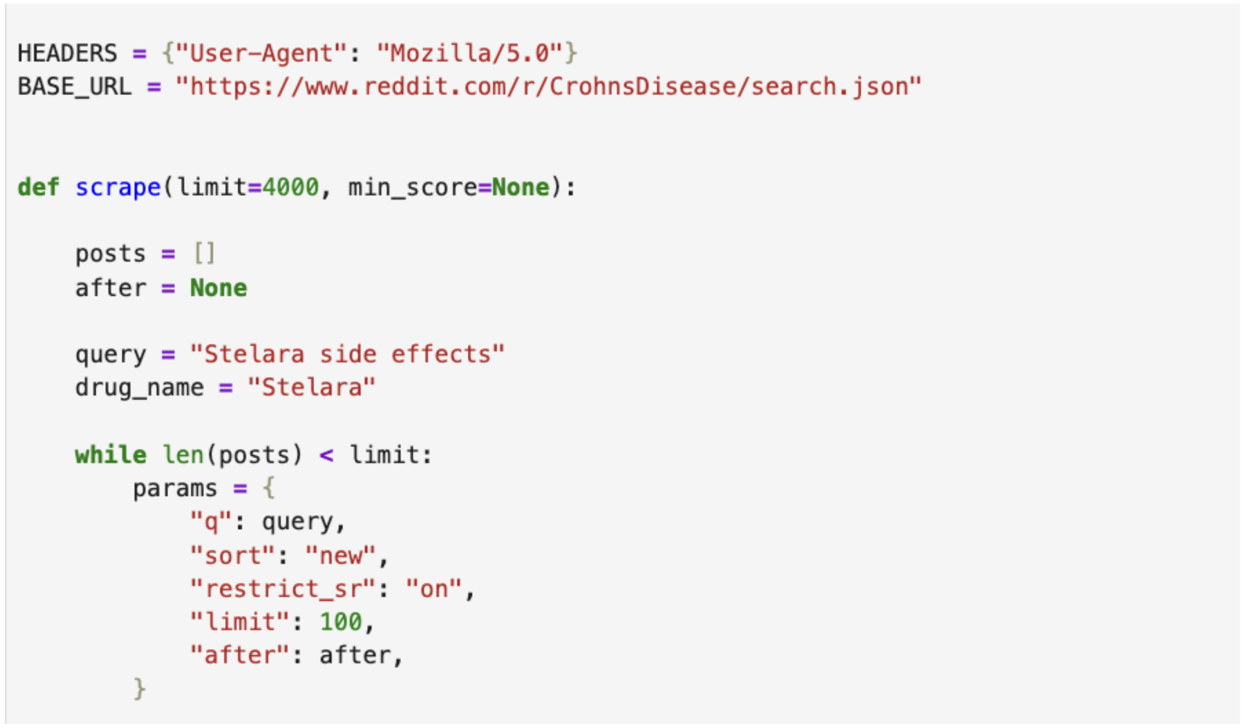
Big pharma companies have limited access to real time patient feedback. Reddit on the other hand are unscripted and reflect topic based signals that can act as early warning systems for a marketing team. Companies often do goal planning around ‘headwinds’ and ‘tailwinds’. Headwinds are obstacles in achieving goals, which is specifically relevant to this experiment. For example, if a company’s post volume contains a lot of talk about negative side effects of their drug, while another company’s post volume contains talk of wanting to switch to another drug, that’s a headwind because it’s a PR obstacle. While they’re trying to win back the favor of their own customers, another company has the ability to take their attention when these customers see these posts. Especially for a niche disease like Crohn’s disease, these people are likely to seek out the experience of others who share the disease because it is so debilitating and the treatments available are not developed enough to allow them to live a regular lifestyle. With that said, a marketing team may wonder if the discussion around their drug has been heading in a specific direction for quite some time now to possibly do some damage, and while that is happening, is the discussion about a competitor able to further compromise us? These are relevant questions to answer for planning a course of action for the company to address the (potential) PR crisis.
That is exactly what this time series project is seeking to answer. Forecasting for the next month gives the marketing team a good idea about where the state of discussion is heading toward.

Contributions:

Project idea: Anna
Data Scraping: Fabian
Data alignment: Anna
Data cleaning and labeling: Fabian
Data unpacking: Fabian
Correlations and line chart analysis: Anna
ETS model research: Fabian
ETS model implementation: Anna
Project results analysis: Anna and Fabian

Reddit Scraper

For each drug, we wrote a code that searches Reddit for posts related to a specific word, such as Stelara. This code uses Reddit's JSON search page, which lets us collect posts without the API. We had to do it this way because Reddit now requires special approval to use their official API, so using the JSON search endpoint was the best option for collecting the data.



```python
HEADERS = {"User-Agent": "Mozilla/5.0"}
BASE_URL = "https://www.reddit.com/r/CrohnsDisease/search.json"


def scrape(limit=4000, min_score=None):

    posts = []
    after = None

    query = "Stelara side effects"
    drug_name = "Stelara"

    while len(posts) < limit:
        params = {
            "q": query,
            "sort": "new",
            "restrict_sr": "on",
            "limit": 100,
            "after": after,
        }
```

snip bit of the code

We used Reddit's API to scrape the first page of each of these 'keywords':

#Stelara_working
#Humira_working
#Skyrizi_working
#Rinvoq_working

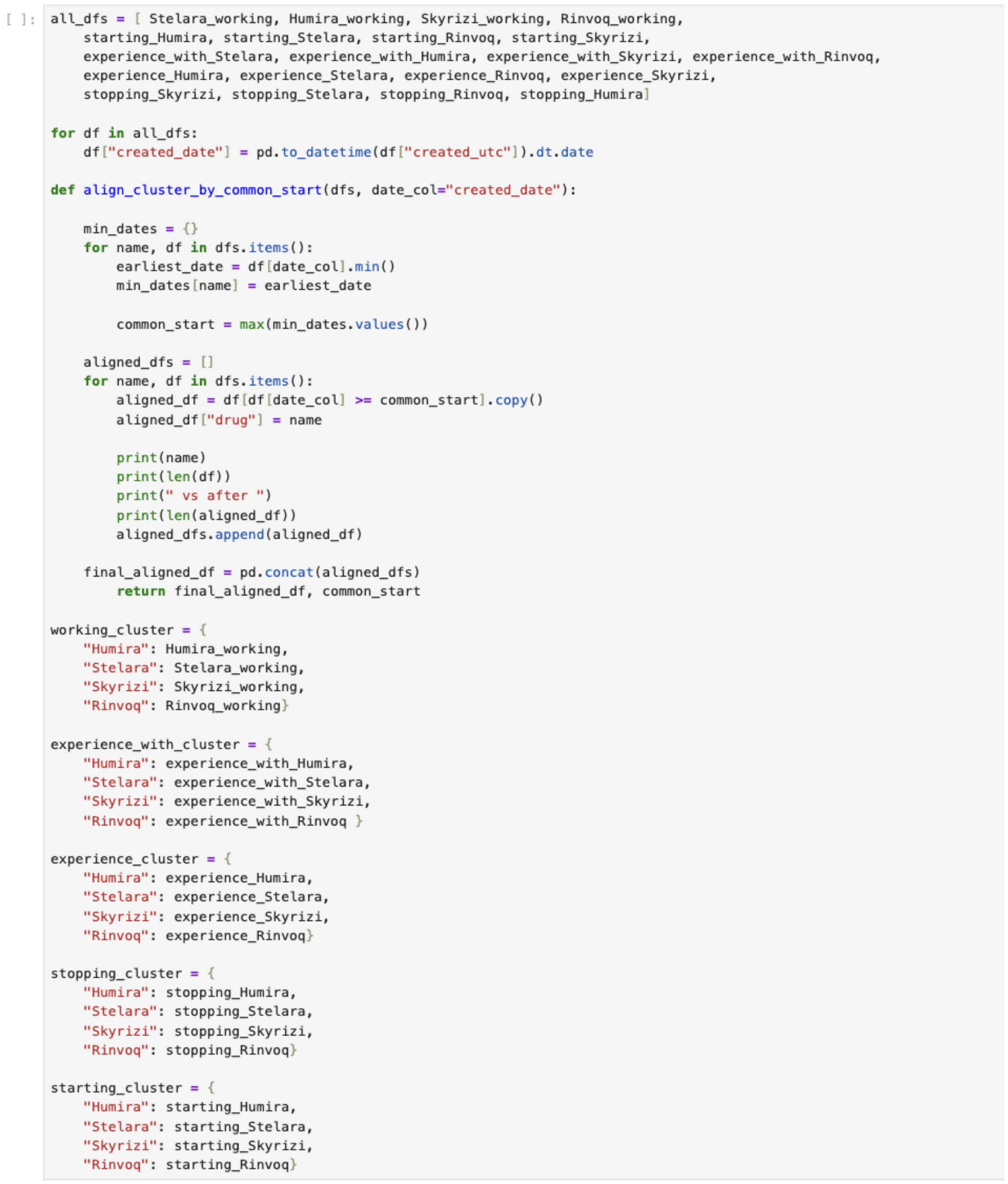
#starting_Humira
#starting_Stelara
#starting_Rinvoq
#starting_Skyrizi

#experience_with_Stelara
#experience_with_Humira
#experience_with_Skyrizi
#experience_with_Rinvoq

#experience_Humira
#experience_Stelara
#experience_Rinvoq
#experience_Skyrizi

#stopping_Skyrizi
#stopping_Stelara
#stopping_Rinvoq
#stopping_Humira

We did this to ensure that we got as close to an unbiased distribution of sentiment and topics as possible.



```
[ ]: all_dfs = [ Stelara_working, Humira_working, Skyrizi_working, Rinvoq_working,
         starting_Humira, starting_Stelara, starting_Rinvoq, starting_Skyrizi,
         experience_with_Stelara, experience_with_Humira, experience_with_Skyrizi, experience_with_Rinvoq,
         experience_Humira, experience_Stelara, experience_Rinvoq, experience_Skyrizi,
         stopping_Skyrizi, stopping_Stelara, stopping_Rinvoq, stopping_Humira]

     for df in all_dfs:
         df["created_date"] = pd.to_datetime(df["created_utc"]).dt.date

     def align_cluster_by_common_start(dfs, date_col="created_date"):

         min_dates = {}
         for name, df in dfs.items():
             earliest_date = df[date_col].min()
             min_dates[name] = earliest_date

             common_start = max(min_dates.values())

         aligned_dfs = []
         for name, df in dfs.items():
             aligned_df = df[df[date_col] >= common_start].copy()
             aligned_df["drug"] = name

             print(name)
             print(len(df))
             print(" vs after ")
             print(len(aligned_df))
             aligned_dfs.append(aligned_df)

         final_aligned_df = pd.concat(aligned_dfs)
             return final_aligned_df, common_start

     working_cluster = {
         "Humira": Humira_working,
         "Stelara": Stelara_working,
         "Skyrizi": Skyrizi_working,
         "Rinvoq": Rinvoq_working}

     experience_with_cluster = {
         "Humira": experience_with_Humira,
         "Stelara": experience_with_Stelara,
         "Skyrizi": experience_with_Skyrizi,
         "Rinvoq": experience_with_Rinvoq }

     experience_cluster = {
         "Humira": experience_Humira,
         "Stelara": experience_Stelara,
         "Skyrizi": experience_Skyrizi,
         "Rinvoq": experience_Rinvoq}

     stopping_cluster = {
         "Humira": stopping_Humira,
         "Stelara": stopping_Stelara,
         "Skyrizi": stopping_Skyrizi,
         "Rinvoq": stopping_Rinvoq}

     starting_cluster = {
         "Humira": starting_Humira,
         "Stelara": starting_Stelara,
         "Skyrizi": starting_Skyrizi,
         "Rinvoq": starting_Rinvoq}
```

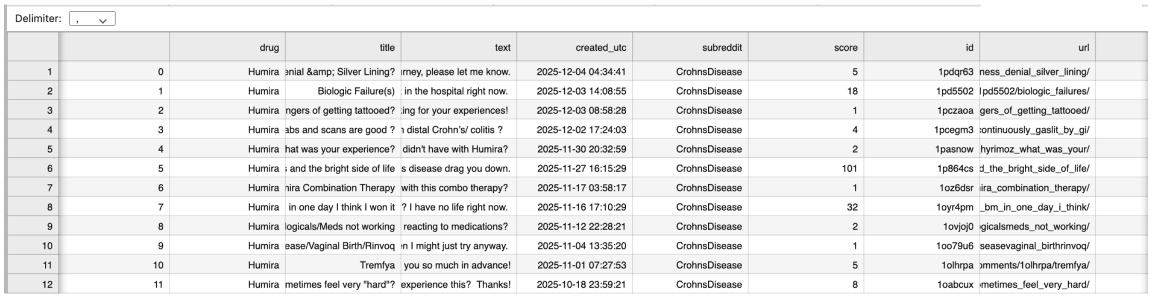
experience_with_aligned, exp_with_start =
align_cluster_by_common_start(experience_with_cluster)
working_aligned, working_start = align_cluster_by_common_start(working_cluster)

```
experience_aligned, exp_start = align_cluster_by_common_start(experience_cluster)
stopping_aligned, stopping_start = align_cluster_by_common_start(stopping_cluster)
starting_aligned, starting_start = align_cluster_by_common_start(starting_cluster)

drug = pd.concat([
    experience_aligned.drop_duplicates(subset=["text", "created_utc"]),
stopping_aligned.drop_duplicates(subset=["text", "created_utc"]),
experience_with_aligned.drop_duplicates(subset=["text", "created_utc"]),
working_aligned.drop_duplicates(subset=["text", "created_utc"]),
starting_aligned.drop_duplicates(subset=["text", "created_utc"])
]).drop_duplicates(subset=["text", "created_utc"])
```

Next we, saw that after examining each dataframe that due to some topic drug combinations being less popular than others, our data ended at all different dates, some spanning back to 2023, and some starting in February of 2025. We needed to have a consistent start date across these dataframes before combining them because we cannot do a time series correlation analysis if entire years are missing for some of our drugs.

We converted each dataframe's created_utc column into a datetime format and created a created_date column so we could get rid of the extra noise from the timestamp. Inside align_cluster_by_common_start, we looped through each drug dataframe and found its minimum date, then took the maximum of those minimum dates to identify the latest first-observed date shared across all of the drugs. We filtered each dataframe to keep only rows on or after that common start date, added a drug label to each trimmed dataframe, and then concatenated them to produce one aligned dataset

Duplicates were dropped based on an id formed by the text and the timestamp, the only reason we didn't use ID was because we weren't sure if ID provided by the API was related to the user or the post itself uniquely.

1. 

Delimiter: ,

		drug	title	text	created_utc	subreddit	score	id	url
1	0	Humira	enial &amp; Silver Lining?	ırney, please let me know.	2025-12-04 04:34:41	CrohnsDisease	5	1pdqr63	ness_denial_silver_lining/
2	1	Humira	Biologic Failure(s)	in the hospital right now.	2025-12-03 14:08:55	CrohnsDisease	18	1pd5502	Ipd5502/biologic_failures/
3	2	Humira	ngers of getting tattooed?	ing for your experiences!	2025-12-03 08:58:28	CrohnsDisease	1	1pczaoa	gers_of_getting_tattooed/
4	3	Humira	abs and scans are good ?	distal Crohn's/ colitis ?	2025-12-02 17:24:03	CrohnsDisease	4	1pcegm3	continuously_gaslit_by_gi/
5	4	Humira	hat was your experience?	didn't have with Humira?	2025-11-30 20:32:59	CrohnsDisease	2	1pasnow	hyrimoz_what_was_your/
6	5	Humira	and the bright side of life	s disease drag you down.	2025-11-27 16:15:29	CrohnsDisease	101	1p864cs	d_the_bright_side_of_life/
7	6	Humira	nira Combination Therapy	with this combo therapy?	2025-11-17 03:58:17	CrohnsDisease	1	1oz6dsr	ira_combination_therapy/
8	7	Humira	in one day I think I won it	? I have no life right now.	2025-11-16 17:10:29	CrohnsDisease	32	1oyr4pm	_bm_in_one_day_i_think/
9	8	Humira	logicals/Meds not working	reacting to medications?	2025-11-12 22:28:21	CrohnsDisease	2	1ovjoj0	gicalsmeds_not_working/
10	9	Humira	ease/Vaginal Birth/Rinvoq	n I might just try anyway.	2025-11-04 13:35:20	CrohnsDisease	1	1oo79u6	seasevaginal_birthrinvoq/
11	10	Humira	Tremfya	you so much in advance!	2025-11-01 07:27:53	CrohnsDisease	5	1olhrpa	omments/1olhrpa/tremfya/
12	11	Humira	metimes feel very "hard"?	experience this? Thanks!	2025-10-18 23:59:21	CrohnsDisease	8	1oabcux	metimes_feel_very_hard/

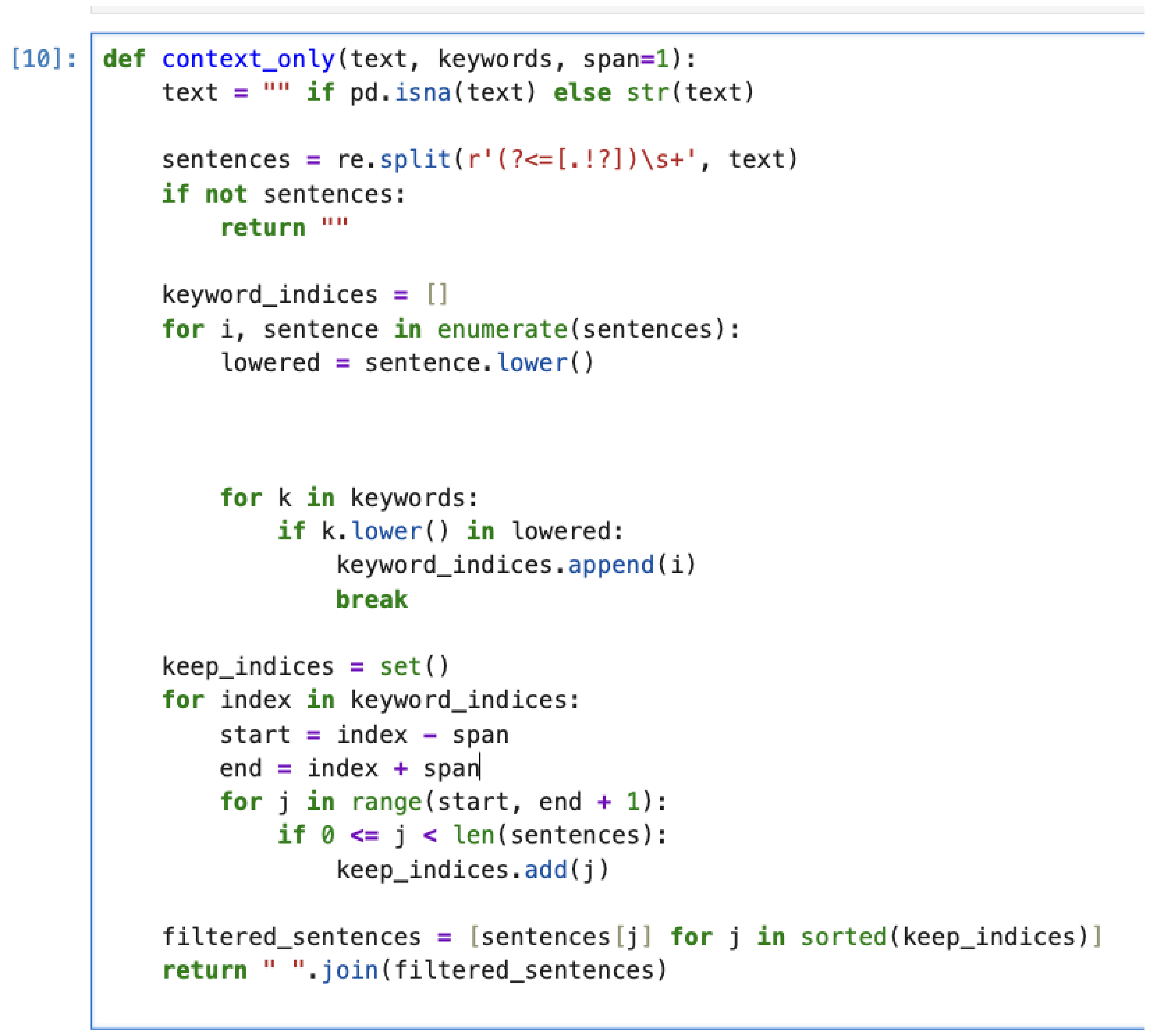
Drug.csv ^

Next, was labeling our topics. After a surface level view of the first few dozen posts, we determined that there were 7 key topics people discussed:

2. **Treatment failure(drug didn't help them)**
3. **Negative side effects**
4. **Insurance issues or affordability**
5. **Asking for peoples advice about going off the drug**
6. **Asking for peoples advice about going on the drug**
7. **Passively mentioning that they had recently started taking it**
8. **Describing the treatment as effective.**

We decided not to get rid of common words or lemant the sentences because phrasing was key to determine what topic they were discussing.

But before we could do topic label, we needed to get rid of extra noise. So we removed all sentences that don't fall within one sentence that contains the drug name.



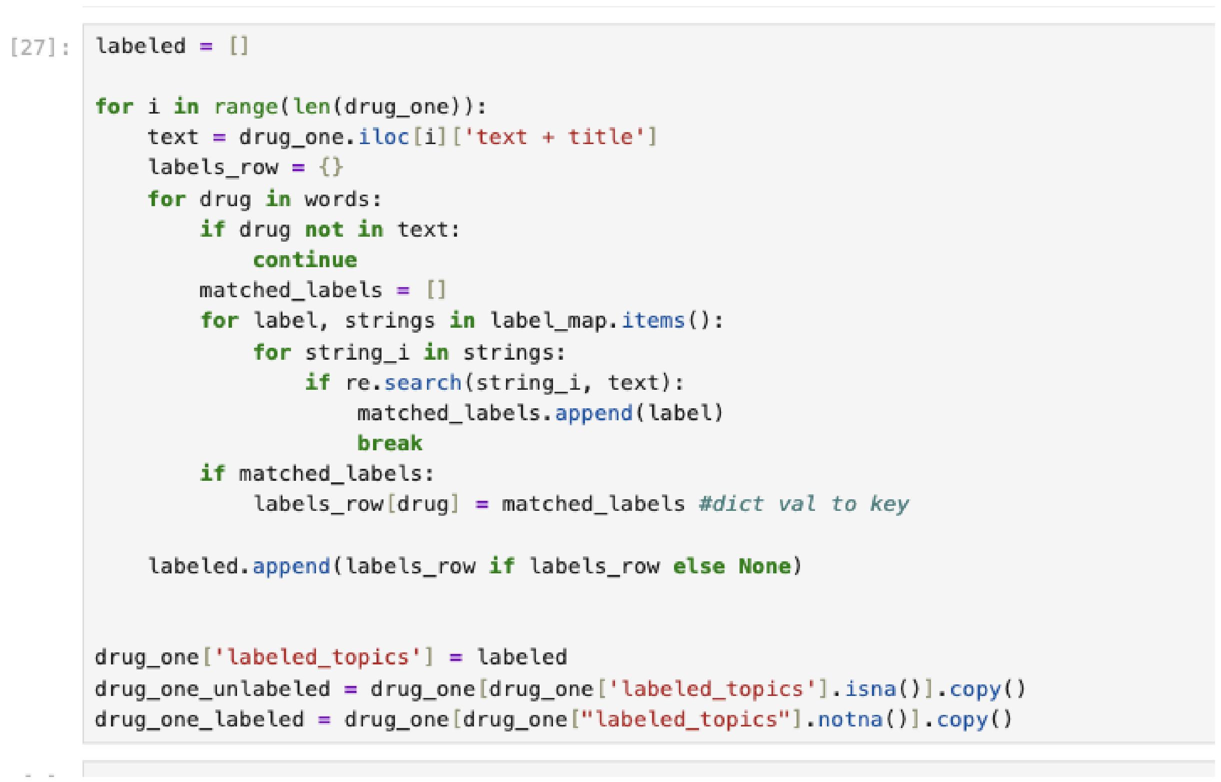
```
[10]: def context_only(text, keywords, span=1):
          text = "" if pd.isna(text) else str(text)

          sentences = re.split(r'(?<=[.!?])\s+', text)
          if not sentences:
              return ""

          keyword_indices = []
          for i, sentence in enumerate(sentences):
              lowered = sentence.lower()



              for k in keywords:
                  if k.lower() in lowered:
                      keyword_indices.append(i)
                      break

          keep_indices = set()
          for index in keyword_indices:
              start = index - span
              end = index + span
              for j in range(start, end + 1):
                  if 0 <= j < len(sentences):
                      keep_indices.add(j)

          filtered_sentences = [sentences[j] for j in sorted(keep_indices)]
          return " ".join(filtered_sentences)
```

We wrote a function that takes text, splits it into sentences using a regular expression, and searches each sentence to see whether it contains any of our drug names. When we found a sentence with a keyword, we recorded its index and then used a context window to look one sentence before and after it. For each keyword hit, we computed the start and end of the window using idx - window and idx + window, and we kept all valid sentence indices that fell inside that range, then we concatenated those together to return the final text.

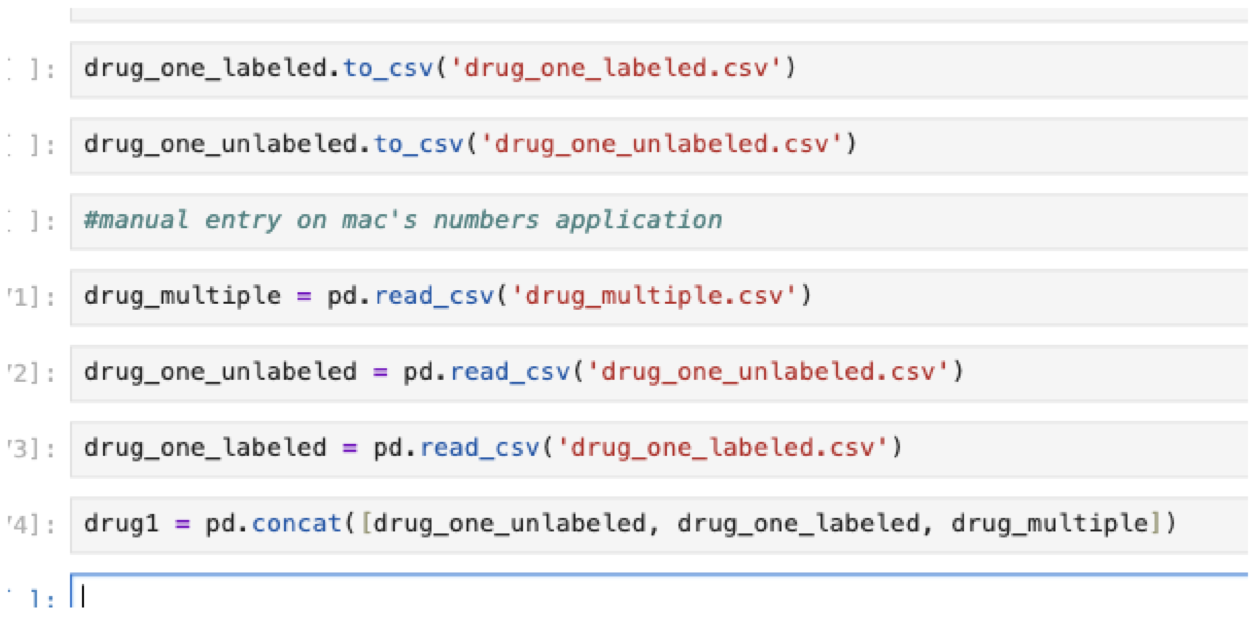
We used this regex topic to string mapping dictionary to assign topics:

label_map = { "treatment_failure":
    [

```
        r"didn[’']?t work", r"(humira|stelara|skyrizi|rinvoq) failed", r"stopped working",
                r"no response",
        r"developed antibodies", r"tried .{1,12} (humira|stelara|skyrizi|rinvoq)[s]?",
                r"not work"],

    "negative_side_effects":
         [
        r"gave me .*", r"made symptoms worse", r"bad reaction", r"pain", r"developed", r"started
develop"],

    "insurance_cost_access":
         [
        r"insurance won[’']?t cover", r"can[’']?t afford", r"biosimilar.*insurance",
        r"copay", r"prior auth", r"formulary", r"coverage", r"too expensive"],

        "switching_off": [
        r"switch(ing)? (from|off)", r"stopping", r"coming off", r"anyone stop",
        r"getting off", r"try another drug", r"experience"],

            "new_start": [
        r"starting", r"first dose", r"about to begin", r"adding", r"\bfirst\b"],

            "switching_to": [
        r"switching to (humira|stelara|skyrizi|rinvoq)", r"move (me)? to
(humira|stelara|skyrizi|rinvoq)",
        r"considering .* vs", r"experience with", r"trying (humira|stelara|skyrizi|rinvoq)"],

    "treatment_effective": [
        r"remission", r"works", r"feel better", r"been working", r"effective "]
}
```



```python
labeled = []

for i in range(len(drug_one)):
    text = drug_one.iloc[i]['text + title']
    labels_row = {}
    for drug in words:
        if drug not in text:
            continue
        matched_labels = []
        for label, strings in label_map.items():
            for string_i in strings:
                if re.search(string_i, text):
                    matched_labels.append(label)
                    break
        if matched_labels:
            labels_row[drug] = matched_labels #dict val to key

    labeled.append(labels_row if labels_row else None)


drug_one['labeled_topics'] = labeled
drug_one_unlabeled = drug_one[drug_one['labeled_topics'].isna()].copy()
drug_one_labeled = drug_one[drug_one["labeled_topics"].notna()].copy()
```



```python
drug_one_labeled.to_csv('drug_one_labeled.csv')
```

```python
drug_one_unlabeled.to_csv('drug_one_unlabeled.csv')
```

```python
#manual entry on mac's numbers application
```

```python
drug_multiple = pd.read_csv('drug_multiple.csv')
```

```python
drug_one_unlabeled = pd.read_csv('drug_one_unlabeled.csv')
```

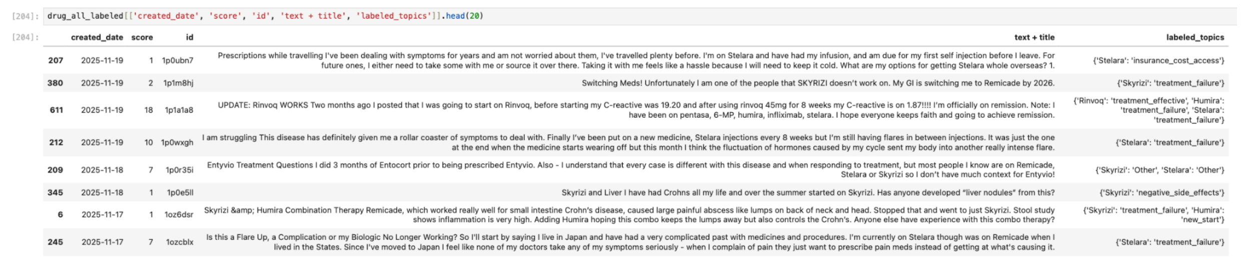
```python
drug_one_labeled = pd.read_csv('drug_one_labeled.csv')
```

```python
drug1 = pd.concat([drug_one_unlabeled, drug_one_labeled, drug_multiple])
```

However, we were only able to label a couple hundred with this algorithm, we needed to unfortunately manually label the rest because for how many stratifications we were doing, having whatever low n counts of data be accurate was key.

We joined the regex labeled and manually labeled data after performing both.

We entered a dictionary including the drug mentioned and its topic assigned, because quite a few posts mentioned more than one drug, so a dictionary containing a key and list of values for each drug was comprehensive enough.
All posts with more than one drug mentioned were manually labeled as well.

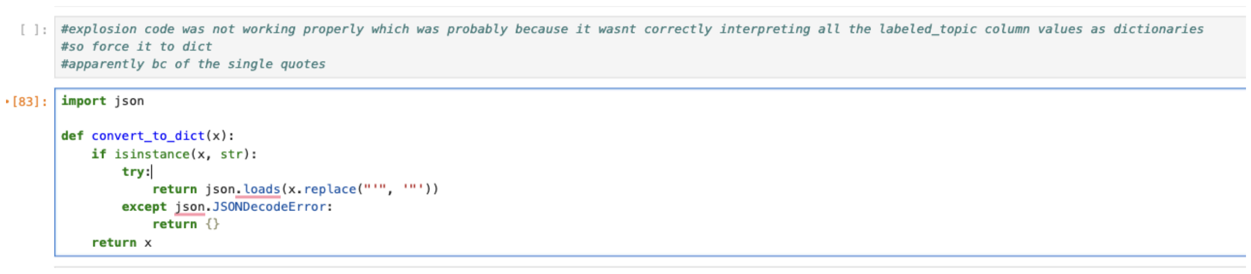


```
[204]: drug_all_labeled[['created_date', 'score', 'id', 'text + title', 'labeled_topics']].head(20)
```

[204]:

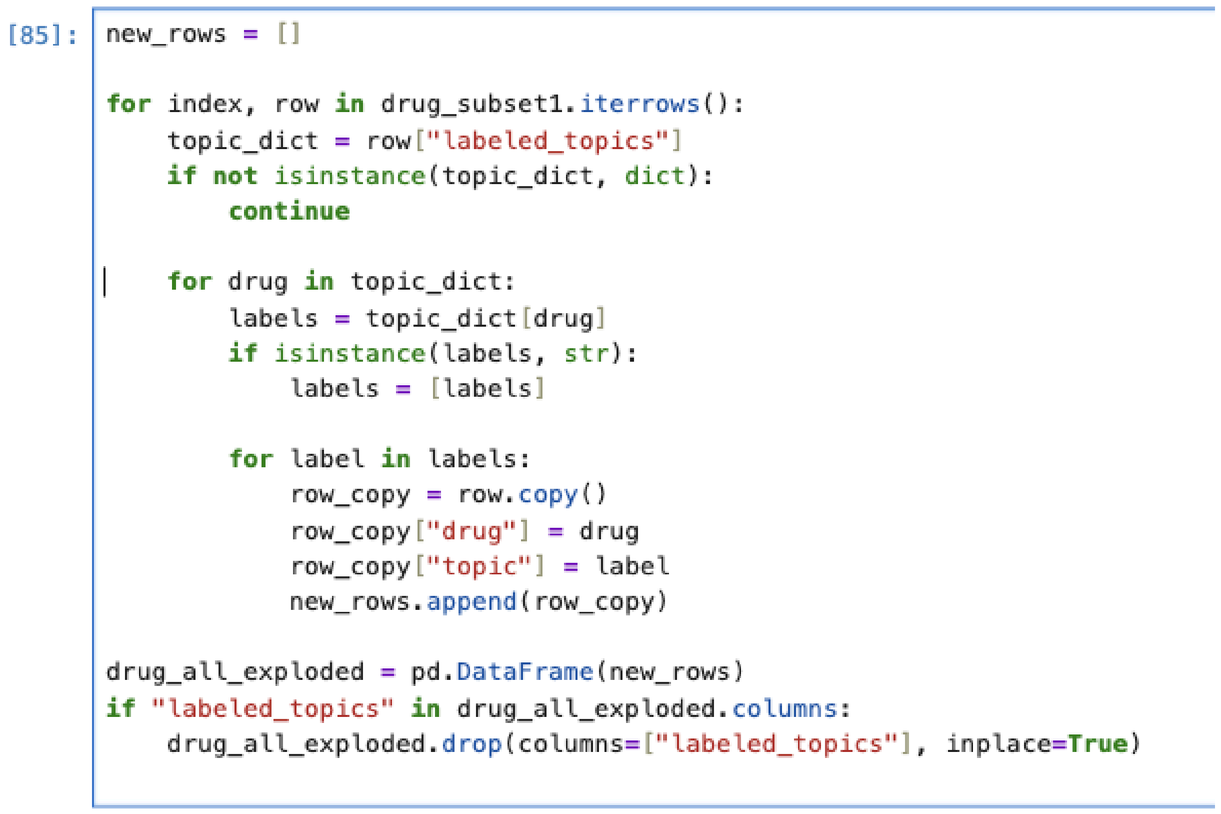
	created_date	score	id	text + title	labeled_topics
207	2025-11-19	1	1p0ubn7	Prescriptions while travelling I've been dealing with symptoms for years and am not worried about them, I've travelled plenty before. I'm on Stelara and have had my infusion, and am due for my first self injection before I leave. For future ones, I either need to take some with me or source it over there. Taking it with me feels like a hassle because I will need to keep it cold. What are my options for getting Stelara whole overseas? 1.	{'Stelara': 'insurance_cost_access'}
380	2025-11-19	2	1p1m8hj	Switching Meds! Unfortunately I am one of the people that SKYRIZI doesn't work on. My GI is switching me to Remicade by 2026.	{'Skyrizi': 'treatment_failure'}
611	2025-11-19	18	1p1a1a8	UPDATE: Rinvoq WORKS Two months ago I posted that I was going to start on Rinvoq, before starting my C-reactive was 19.20 and after using rinvoq 45mg for 8 weeks my C-reactive is on 1.87!!!! I'm officially on remission. Note: I have been on pentasa, 6-MP, humira, infliximab, stelara. I hope everyone keeps faith and going to achieve remission.	{'Rinvoq': 'treatment_effective', 'Humira': 'treatment_failure', 'Stelara': 'treatment_failure'}
212	2025-11-19	10	1p0wxgh	I am struggling This disease has definitely given me a rollar coaster of symptoms to deal with. Finally I've been put on a new medicine, Stelara injections every 8 weeks but I'm still having flares in between injections. It was just the one at the end when the medicine starts wearing off but this month I think the fluctuation of hormones caused by my cycle sent my body into another really intense flare.	{'Stelara': 'treatment_failure'}
209	2025-11-18	7	1p0r35i	Entyvio Treatment Questions I did 3 months of Entocort prior to being prescribed Entyvio. Also - I understand that every case is different with this disease and when responding to treatment, but most people I know are on Remicade, Stelara or Skyrizi so I don't have much context for Entyvio!	{'Skyrizi': 'Other', 'Stelara': 'Other'}
345	2025-11-18	1	1p0e5ll	Skyrizi and Liver I have had Crohns all my life and over the summer started on Skyrizi. Has anyone developed "liver nodules" from this?	{'Skyrizi': 'negative_side_effects'}
6	2025-11-17	1	1oz6dsr	Skyrizi &amp; Humira Combination Therapy Remicade, which worked really well for small intestine Crohn's disease, caused large painful abscess like lumps on back of neck and head. Stopped that and went to just Skyrizi. Stool study shows inflammation is very high. Adding Humira hoping this combo keeps the lumps away but also controls the Crohn's. Anyone else have experience with this combo therapy?	{'Skyrizi': 'treatment_failure', 'Humira': 'new_start'}
245	2025-11-17	7	1ozcblx	Is this a Flare Up, a Complication or my Biologic No Longer Working? So I'll start by saying I live in Japan and have had a very complicated past with medicines and procedures. I'm currently on Stelara though was on Remicade when I lived in the States. Since I've moved to Japan I feel like none of my doctors take any of my symptoms seriously - when I complain of pain they just want to prescribe pain meds instead of getting at what's causing it.	{'Stelara': 'treatment_failure'}

A subset of our data frame.



```
[ ]: #explosion code was not working properly which was probably because it wasnt correctly interpreting all the labeled_topic column values as dictionaries
     #so force it to dict
     #apparently bc of the single quotes
```

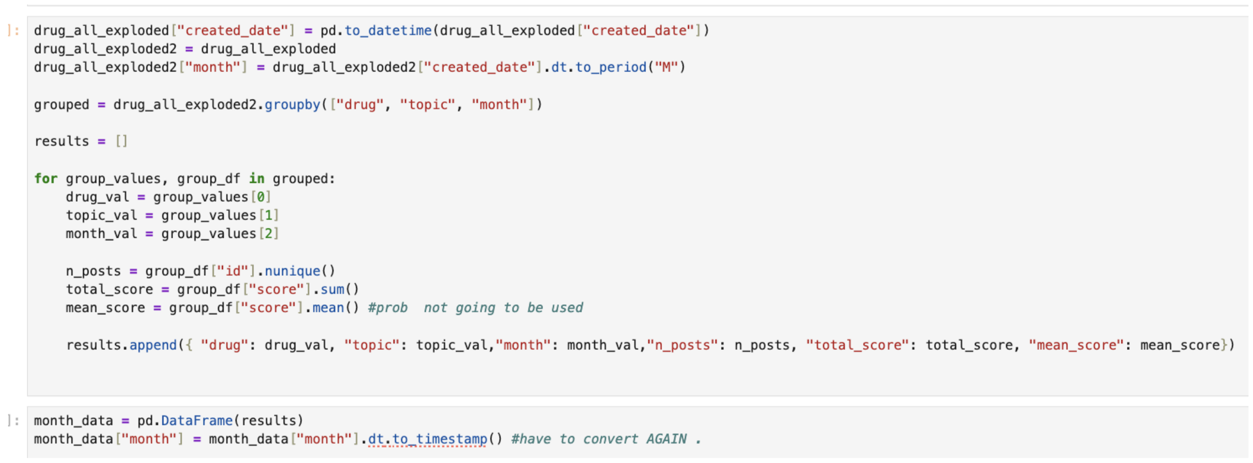
```
[83]: import json

      def convert_to_dict(x):
          if isinstance(x, str):
              try:
                  return json.loads(x.replace("'", '"'))
              except json.JSONDecodeError:
                  return {}
          return x
```


[85]:
```
new_rows = []

for index, row in drug_subset1.iterrows():
    topic_dict = row["labeled_topics"]
    if not isinstance(topic_dict, dict):
        continue

    for drug in topic_dict:
        labels = topic_dict[drug]
        if isinstance(labels, str):
            labels = [labels]

        for label in labels:
            row_copy = row.copy()
            row_copy["drug"] = drug
            row_copy["topic"] = label
            new_rows.append(row_copy)

drug_all_exploded = pd.DataFrame(new_rows)
if "labeled_topics" in drug_all_exploded.columns:
    drug_all_exploded.drop(columns=["labeled_topics"], inplace=True)
```

We then had to unload our dictionary column into a drug and topic column as seen above ^

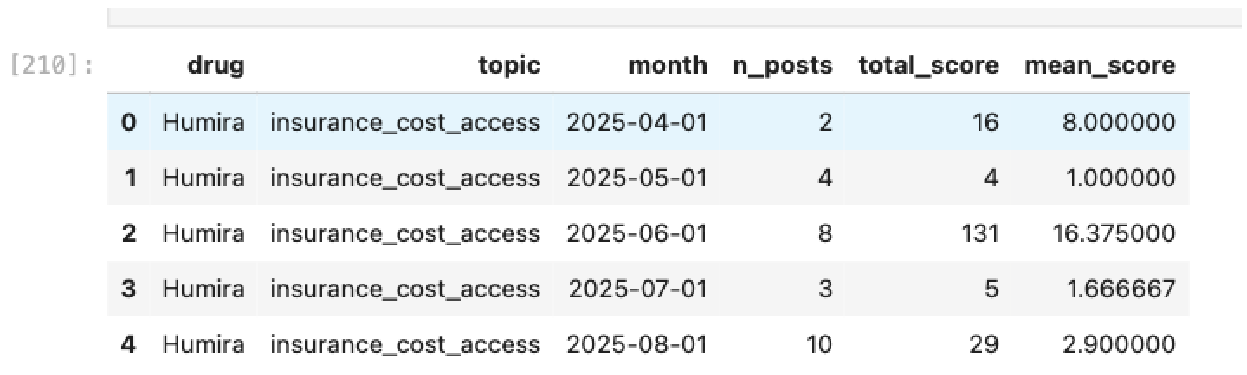
We wrote a function that unpacked the nested topic dictionaries so each drug–topic pair became its own row. Each post originally had a labeled_topics column containing a dictionary like {"Stelara": "insurance_cost_access"} or sometimes a drug mapped to multiple labels. We looped through the dataframe row by row, pulled out this dictionary, and then iterated over every (drug, labels) entry inside it. When a label was a single string, we converted it into a list so the code could treat single-label and multi-label cases the same way. For every individual label, we made a full copy of the original row, replaced the drug field with the drug key from the dictionary, added a topic column containing that label, and stored this new row. We then combined all the new rows into a one dataframe.

Because our goal is to forecast a month or two outward, we aggregated the number of posts(rows) for each drug and topic combination by month.



```
]: drug_all_exploded["created_date"] = pd.to_datetime(drug_all_exploded["created_date"])
   drug_all_exploded2 = drug_all_exploded
   drug_all_exploded2["month"] = drug_all_exploded2["created_date"].dt.to_period("M")

   grouped = drug_all_exploded2.groupby(["drug", "topic", "month"])

   results = []

   for group_values, group_df in grouped:
       drug_val = group_values[0]
       topic_val = group_values[1]
       month_val = group_values[2]

       n_posts = group_df["id"].nunique()
       total_score = group_df["score"].sum()
       mean_score = group_df["score"].mean() #prob  not going to be used

       results.append({ "drug": drug_val, "topic": topic_val,"month": month_val,"n_posts": n_posts, "total_score": total_score, "mean_score": mean_score})
```

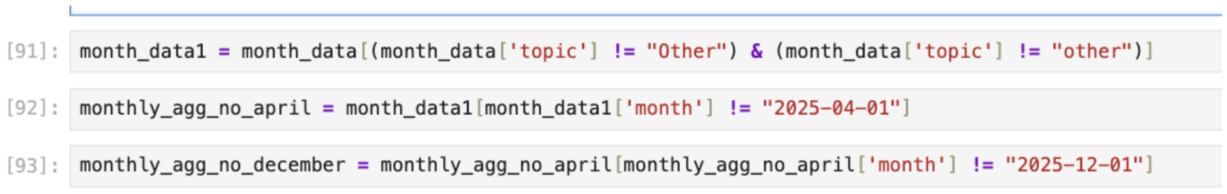
```
]: month_data = pd.DataFrame(results)
   month_data["month"] = month_data["month"].dt.to_timestamp() #have to convert AGAIN .
```



[210]:

	drug	topic	month	n_posts	total_score	mean_score
0	Humira	insurance_cost_access	2025-04-01	2	16	8.000000
1	Humira	insurance_cost_access	2025-05-01	4	4	1.000000
2	Humira	insurance_cost_access	2025-06-01	8	131	16.375000
3	Humira	insurance_cost_access	2025-07-01	3	5	1.666667
4	Humira	insurance_cost_access	2025-08-01	10	29	2.900000

We also got rid of values in december and april because our data started on april 17th and ends december 7th, so those are incomplete months. It is unfortunate to only work with 7 months of data, but that is what we are limited to. And since the forecast is only supposed to be a month or two out, if there is a trend it should capture it.

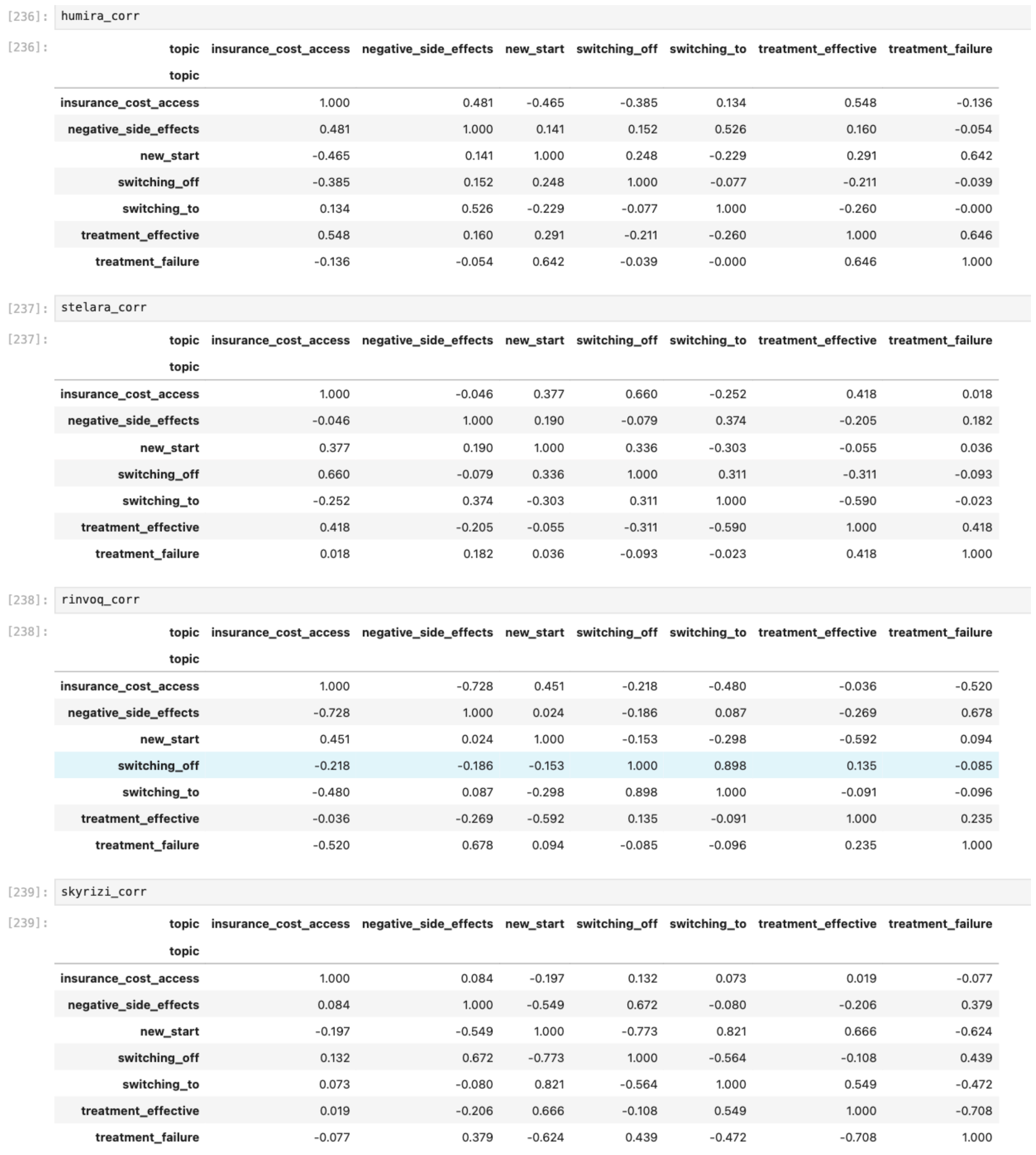


```
[91]: month_data1 = month_data[(month_data['topic'] != "Other") & (month_data['topic'] != "other")]

[92]: monthly_agg_no_april = month_data1[month_data1['month'] != "2025-04-01"]

[93]: monthly_agg_no_december = monthly_agg_no_april[monthly_agg_no_april['month'] != "2025-12-01"]
```

For initial exploration, we had to see if the counts of the topics within drugs are correlated.

[94]:
```
def topic_corr__drug(df, drug_name):
    sub = df[df["drug"] == drug_name]

    wide = sub.pivot_table(index="month",  columns="topic", values="n_posts", fill_value=0 )
    return wide.corr()
```

[95]:
```
humira_corr = topic_corr__drug(monthly_agg_no_december, "Humira")
stelara_corr = topic_corr__drug(monthly_agg_no_december, "Stelara")
skyrizi_corr = topic_corr__drug(monthly_agg_no_december, "Skyrizi")
rinvoq_corr = topic_corr__drug(monthly_agg_no_december, "Rinvoq")
```



[236]:
```
humira_corr
```

[236]:

topic	insurance_cost_access	negative_side_effects	new_start	switching_off	switching_to	treatment_effective	treatment_failure
topic							
insurance_cost_access	1.000	0.481	-0.465	-0.385	0.134	0.548	-0.136
negative_side_effects	0.481	1.000	0.141	0.152	0.526	0.160	-0.054
new_start	-0.465	0.141	1.000	0.248	-0.229	0.291	0.642
switching_off	-0.385	0.152	0.248	1.000	-0.077	-0.211	-0.039
switching_to	0.134	0.526	-0.229	-0.077	1.000	-0.260	-0.000
treatment_effective	0.548	0.160	0.291	-0.211	-0.260	1.000	0.646
treatment_failure	-0.136	-0.054	0.642	-0.039	-0.000	0.646	1.000

[237]:
```
stelara_corr
```

[237]:

topic	insurance_cost_access	negative_side_effects	new_start	switching_off	switching_to	treatment_effective	treatment_failure
topic							
insurance_cost_access	1.000	-0.046	0.377	0.660	-0.252	0.418	0.018
negative_side_effects	-0.046	1.000	0.190	-0.079	0.374	-0.205	0.182
new_start	0.377	0.190	1.000	0.336	-0.303	-0.055	0.036
switching_off	0.660	-0.079	0.336	1.000	0.311	-0.311	-0.093
switching_to	-0.252	0.374	-0.303	0.311	1.000	-0.590	-0.023
treatment_effective	0.418	-0.205	-0.055	-0.311	-0.590	1.000	0.418
treatment_failure	0.018	0.182	0.036	-0.093	-0.023	0.418	1.000

[238]:
```
rinvoq_corr
```

[238]:

topic	insurance_cost_access	negative_side_effects	new_start	switching_off	switching_to	treatment_effective	treatment_failure
topic							
insurance_cost_access	1.000	-0.728	0.451	-0.218	-0.480	-0.036	-0.520
negative_side_effects	-0.728	1.000	0.024	-0.186	0.087	-0.269	0.678
new_start	0.451	0.024	1.000	-0.153	-0.298	-0.592	0.094
switching_off	-0.218	-0.186	-0.153	1.000	0.898	0.135	-0.085
switching_to	-0.480	0.087	-0.298	0.898	1.000	-0.091	-0.096
treatment_effective	-0.036	-0.269	-0.592	0.135	-0.091	1.000	0.235
treatment_failure	-0.520	0.678	0.094	-0.085	-0.096	0.235	1.000

[239]:
```
skyrizi_corr
```

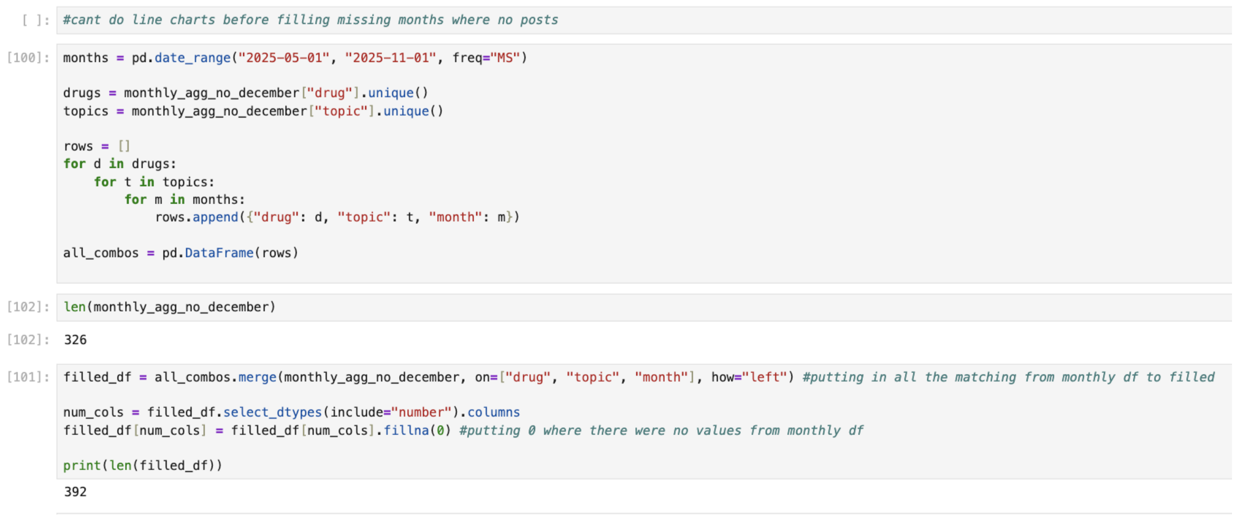
[239]:

topic	insurance_cost_access	negative_side_effects	new_start	switching_off	switching_to	treatment_effective	treatment_failure
topic							
insurance_cost_access	1.000	0.084	-0.197	0.132	0.073	0.019	-0.077
negative_side_effects	0.084	1.000	-0.549	0.672	-0.080	-0.206	0.379
new_start	-0.197	-0.549	1.000	-0.773	0.821	0.666	-0.624
switching_off	0.132	0.672	-0.773	1.000	-0.564	-0.108	0.439
switching_to	0.073	-0.080	0.821	-0.564	1.000	0.549	-0.472
treatment_effective	0.019	-0.206	0.666	-0.108	0.549	1.000	-0.708
treatment_failure	-0.077	0.379	-0.624	0.439	-0.472	-0.708	1.000

We are seeing strong r values, but they could be articially inflated by very small n’s, so a p value is necessary to confirm.

We wanted to see the time series line charts, but before that we had to fill in any missing months for consistency, since some months had no counts of a specific drug topic combination.

That new dataframe was called filled_df



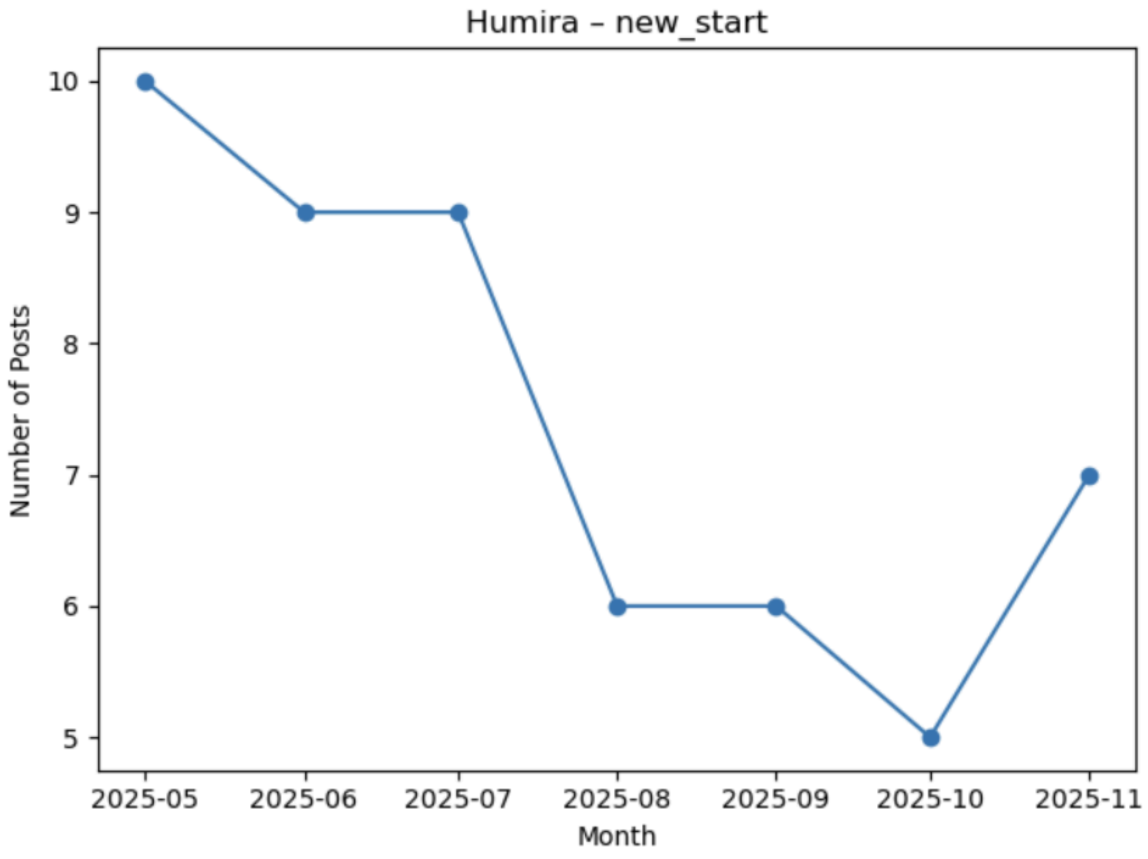
```
[ ]: #cant do line charts before filling missing months where no posts
```

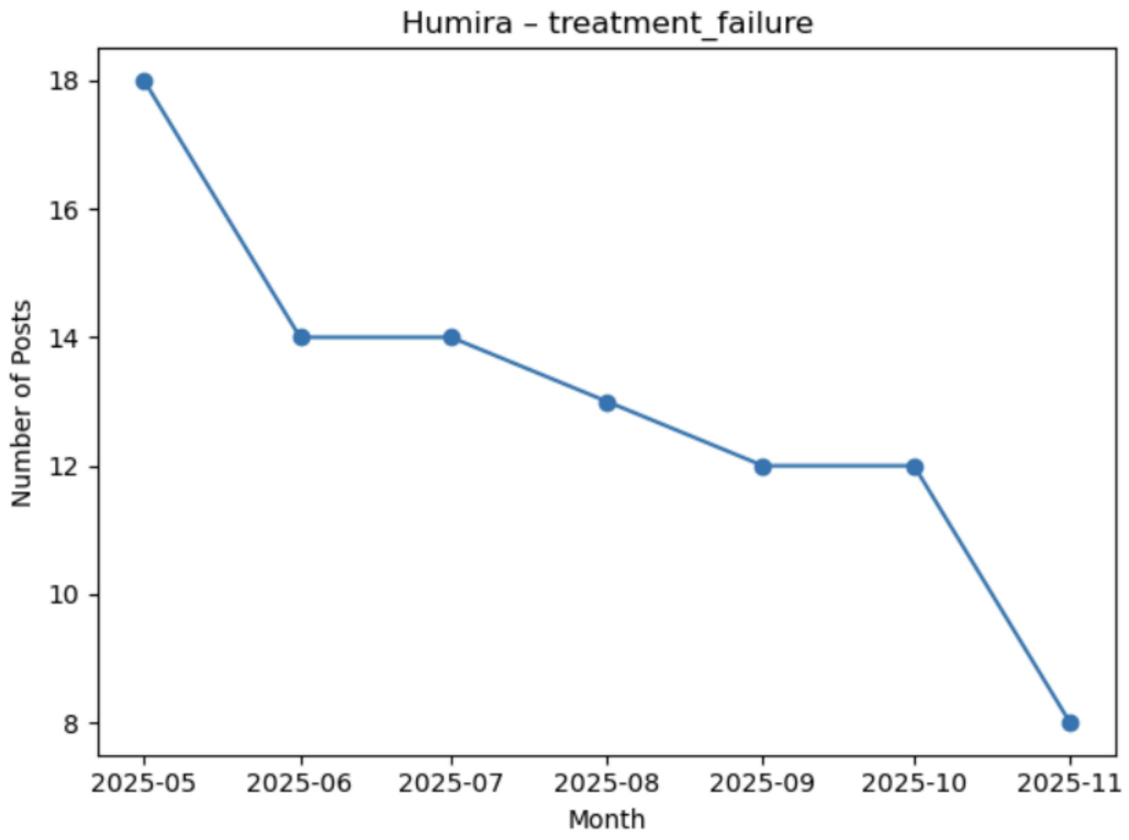
```
[100]: months = pd.date_range("2025-05-01", "2025-11-01", freq="MS")

       drugs = monthly_agg_no_december["drug"].unique()
       topics = monthly_agg_no_december["topic"].unique()

       rows = []
       for d in drugs:
           for t in topics:
               for m in months:
                   rows.append({"drug": d, "topic": t, "month": m})

       all_combos = pd.DataFrame(rows)
```

```
[102]: len(monthly_agg_no_december)
[102]: 326
```

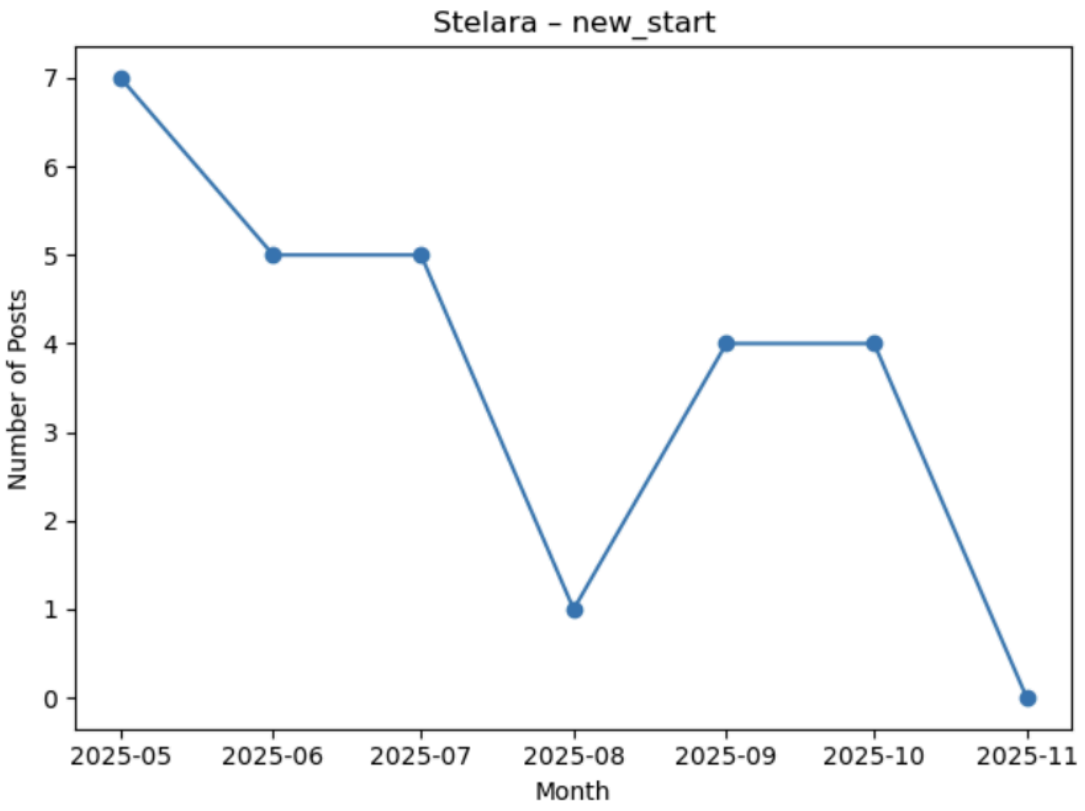
```
[101]: filled_df = all_combos.merge(monthly_agg_no_december, on=["drug", "topic", "month"], how="left") #putting in all the matching from monthly df to filled

       num_cols = filled_df.select_dtypes(include="number").columns
       filled_df[num_cols] = filled_df[num_cols].fillna(0) #putting 0 where there were no values from monthly df

       print(len(filled_df))
       392
```

Then we looked at line charts to watch out for trends.

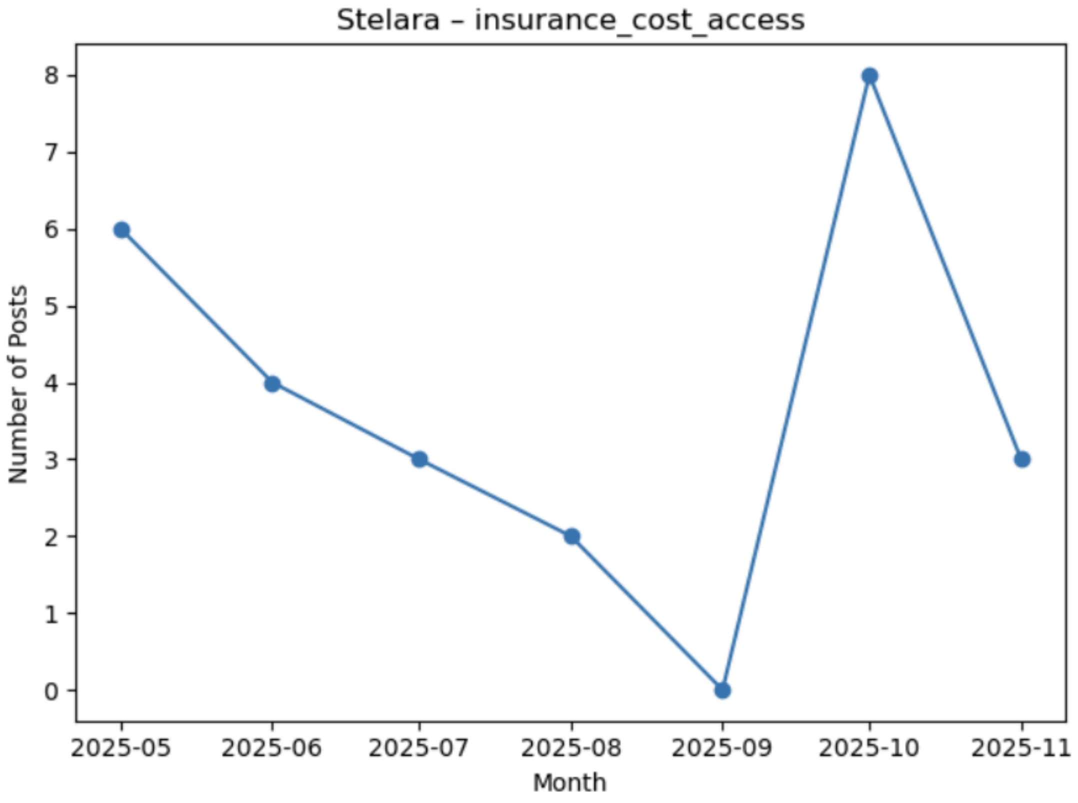

Humira – new_start
Number of Posts
10
9
8
7
6
5
2025-05
2025-06
2025-07
2025-08
2025-09
2025-10
2025-11
Month


Humira – treatment_failure
Number of Posts
18
16
14
12
10
8
2025-05
2025-06
2025-07
2025-08
2025-09
2025-10
2025-11
Month

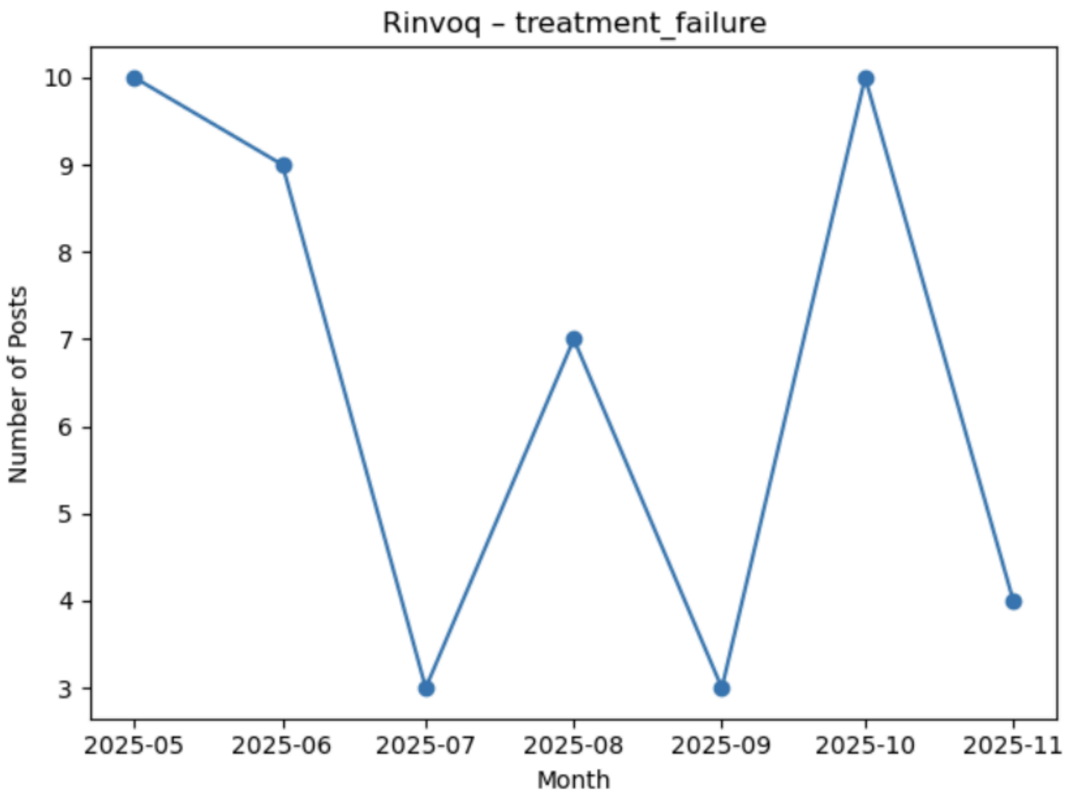
Month



There were unfortunately only a few that looked like they had a trend. (above)

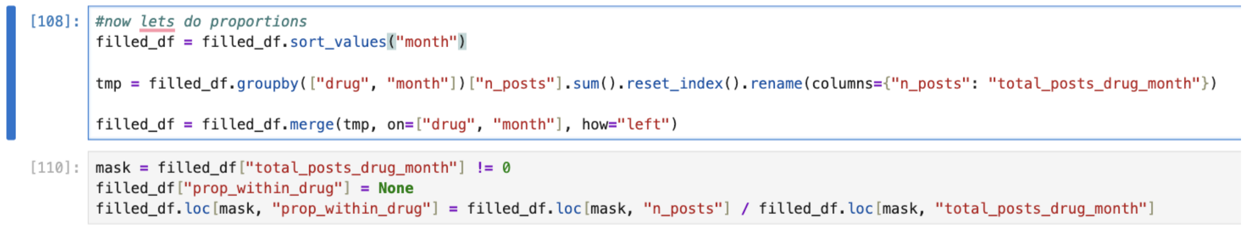


Stelara – insurance_cost_access
Number of Posts
0
1
2
3
4
5
6
7
8
2025-05
2025-06
2025-07
2025-08
2025-09
2025-10
2025-11
Month



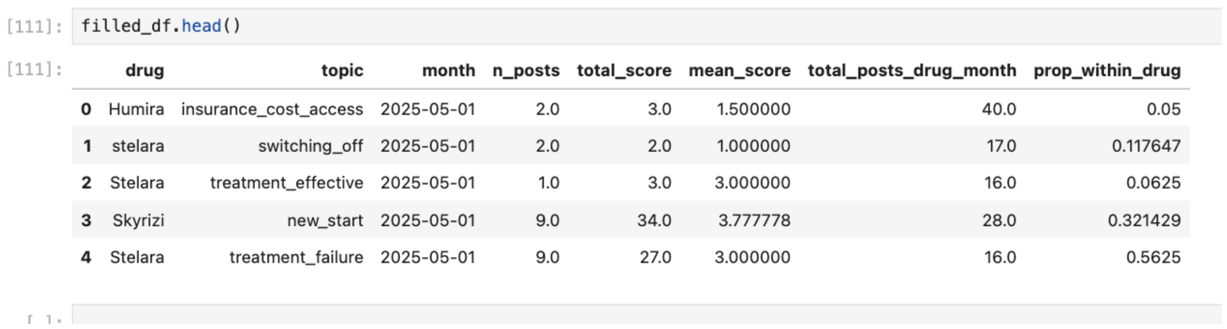
The others were too random fluctuation and didn’t have a strong seasonality either. (examples above)

We realized that plain counts may be biased by reddit activity going up in a certain period broadly, creating bias for comparison between drugs. So we feature engineered a proportion column that for each drug and moth the topic proportions add to 1. This way we can see how the topic’s post volume changes relative to the other topics of that drug.



```
[108]: #now lets do proportions
       filled_df = filled_df.sort_values("month")

       tmp = filled_df.groupby(["drug", "month"])["n_posts"].sum().reset_index().rename(columns={"n_posts": "total_posts_drug_month"})

       filled_df = filled_df.merge(tmp, on=["drug", "month"], how="left")
```

```
[110]: mask = filled_df["total_posts_drug_month"] != 0
       filled_df["prop_within_drug"] = None
       filled_df.loc[mask, "prop_within_drug"] = filled_df.loc[mask, "n_posts"] / filled_df.loc[mask, "total_posts_drug_month"]
```



```
[111]: filled_df.head()
```

	drug	topic	month	n_posts	total_score	mean_score	total_posts_drug_month	prop_within_drug
0	Humira	insurance_cost_access	2025-05-01	2.0	3.0	1.500000	40.0	0.05
1	stelara	switching_off	2025-05-01	2.0	2.0	1.000000	17.0	0.117647
2	Stelara	treatment_effective	2025-05-01	1.0	3.0	3.000000	16.0	0.0625
3	Skyrizi	new_start	2025-05-01	9.0	34.0	3.777778	28.0	0.321429
4	Stelara	treatment_failure	2025-05-01	9.0	27.0	3.000000	16.0	0.5625

EXPERIMENTAL DESIGN(at first)

At this point we had come up with several specific hypothesis that we believed would be the most useful of all topic drug combination comparisons:

If these correlations were positive, then that would be useful information for a company doing this analysis, because they can see which competitor's post volume consists of people interested in the product increases when their own drug's post volume consists of a negative topic. High correlation signals further analysis into the association.

insurance [drug1] vs switching to [drug2]

insurance [drug1] vs new start [drug2]

treatment_fail [drug 1] vs switching to [drug2]

treatment_fail [drug 1] vs new start [drug2]

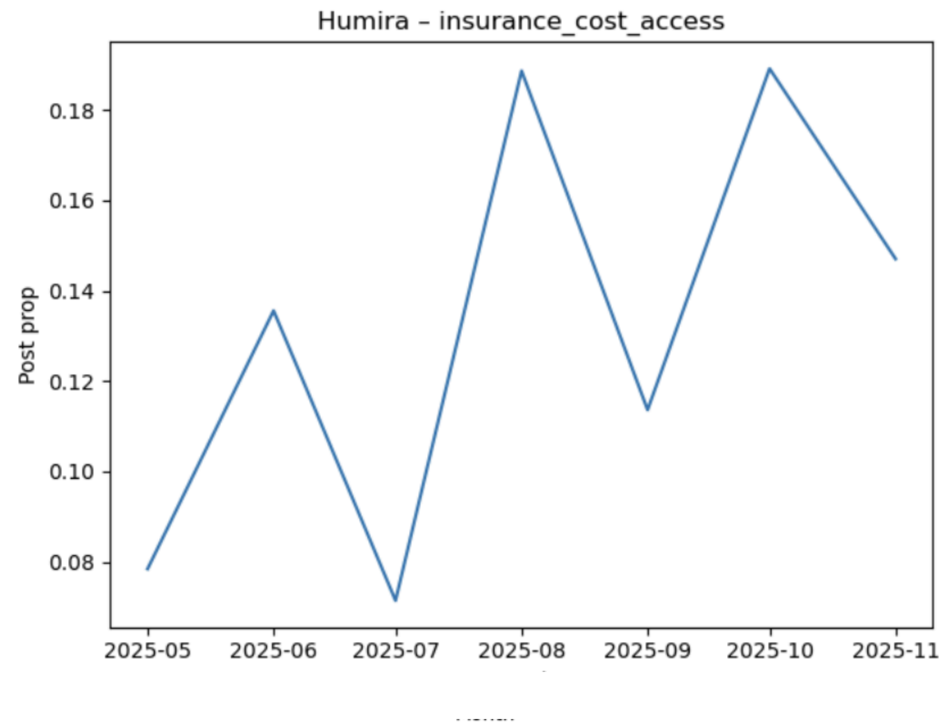
negative_side_effects [drug1] vs switching to [drug2]

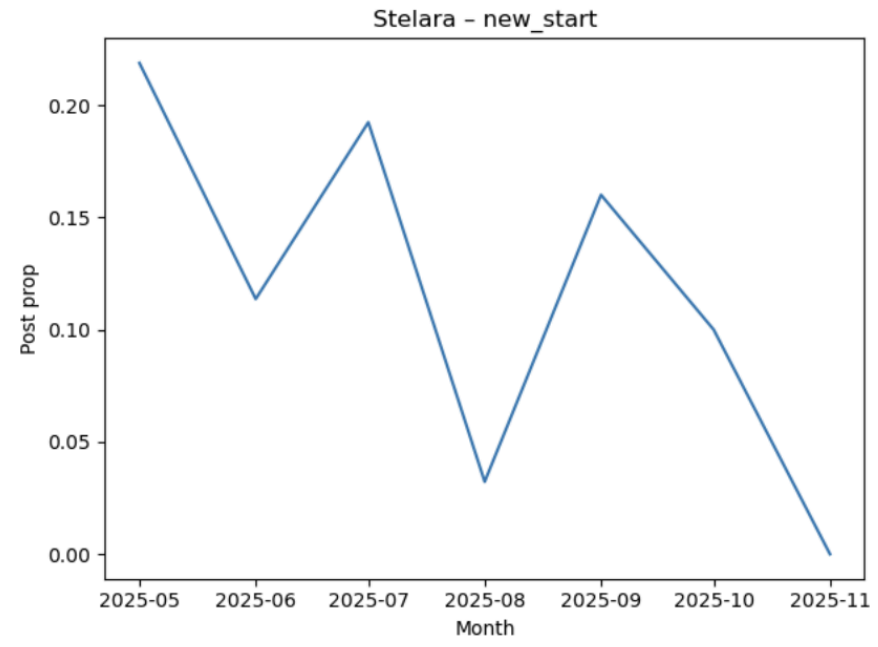
negative_side_effects [drug1] vs new start [drug2]

switching_off [drug 1] vs switching_to [drug 2]

switching_off [drug 1] vs new start [drug 2]

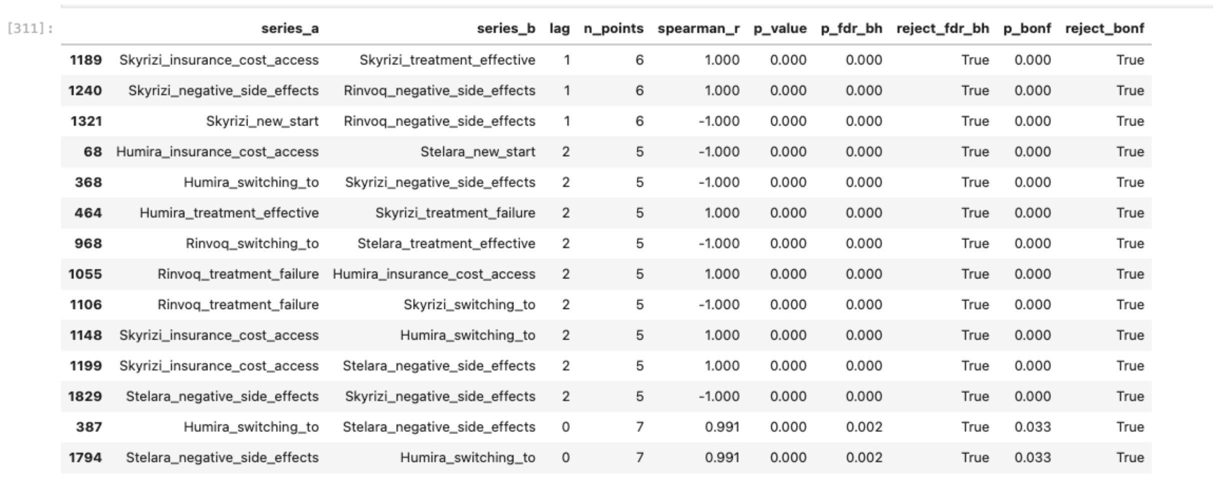
Then we looked at the line graphs for this new column





These two were the only two that seemed interesting side by side.

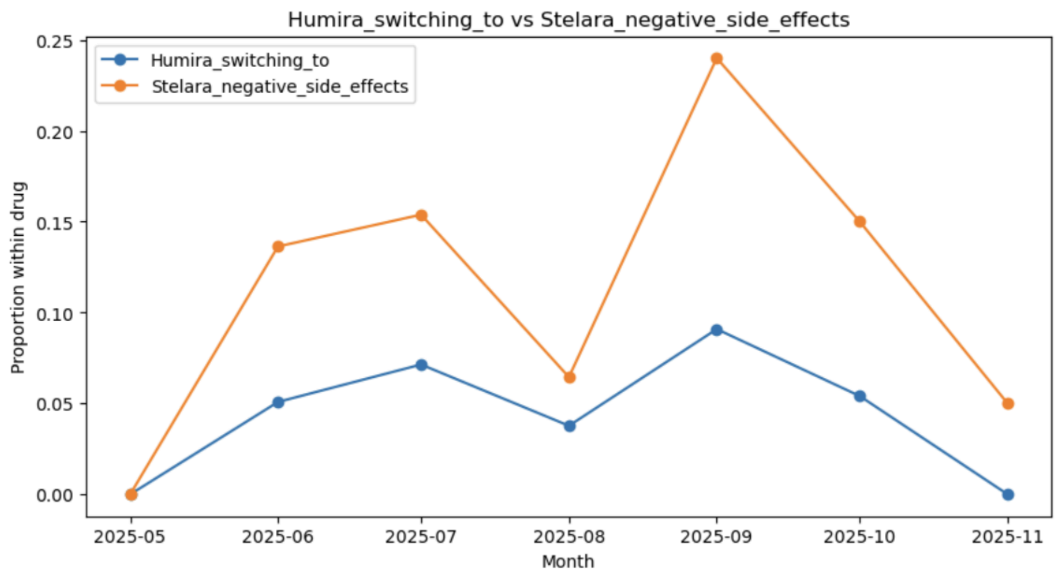
Now for the actual testing of these correlations. Not only did we choose to test the correlations aligning months from different drug-topic combinations, but we included lags as well, since it could be the case that it takes some a month or two for negative sentiment about one drug to be associated with an increase in positive sentiment about another, vice versa. We tested all possible combinations using Pearson Correlation coefficients(a standard metric) and did a multiple testing correction to the p-value. We had to do this because we had many combinations of drug-topic comparisons, which inflates the probability of incorrectly rejecting the null hypothesis by getting a low p-value by chance.



[311]:

	series_a	series_b	lag	n_points	spearman_r	p_value	p_fdr_bh	reject_fdr_bh	p_bonf	reject_bonf
1189	Skyrizi_insurance_cost_access	Skyrizi_treatment_effective	1	6	1.000	0.000	0.000	True	0.000	True
1240	Skyrizi_negative_side_effects	Rinvoq_negative_side_effects	1	6	1.000	0.000	0.000	True	0.000	True
1321	Skyrizi_new_start	Rinvoq_negative_side_effects	1	6	-1.000	0.000	0.000	True	0.000	True
68	Humira_insurance_cost_access	Stelara_new_start	2	5	-1.000	0.000	0.000	True	0.000	True
368	Humira_switching_to	Skyrizi_negative_side_effects	2	5	-1.000	0.000	0.000	True	0.000	True
464	Humira_treatment_effective	Skyrizi_treatment_failure	2	5	1.000	0.000	0.000	True	0.000	True
968	Rinvoq_switching_to	Stelara_treatment_effective	2	5	-1.000	0.000	0.000	True	0.000	True
1055	Rinvoq_treatment_failure	Humira_insurance_cost_access	2	5	1.000	0.000	0.000	True	0.000	True
1106	Rinvoq_treatment_failure	Skyrizi_switching_to	2	5	-1.000	0.000	0.000	True	0.000	True
1148	Skyrizi_insurance_cost_access	Humira_switching_to	2	5	1.000	0.000	0.000	True	0.000	True
1199	Skyrizi_insurance_cost_access	Stelara_negative_side_effects	2	5	1.000	0.000	0.000	True	0.000	True
1829	Stelara_negative_side_effects	Skyrizi_negative_side_effects	2	5	-1.000	0.000	0.000	True	0.000	True
387	Humira_switching_to	Stelara_negative_side_effects	0	7	0.991	0.000	0.002	True	0.033	True
1794	Stelara_negative_side_effects	Humira_switching_to	0	7	0.991	0.000	0.002	True	0.033	True

This resulted in an interesting relationship between people reporting negative side effects on Stelara and also going on Reddit to ask for people's experiences with going on Humira.



We overlaid the changes in proportions overtime in a line chart and observed what seemed like a meaningful relationship, however there are only two season cycles, which isn't enough to legitimately determine seasonality.

There is also no clear trend for either.

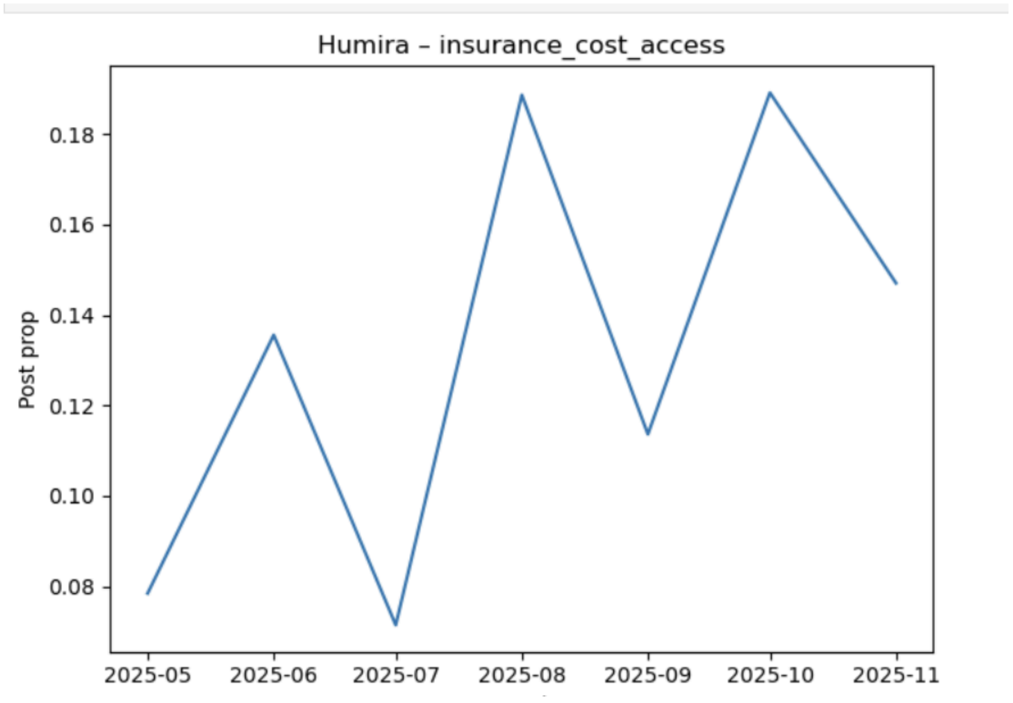
For small data sets like these, official tests for trend and seasonality are not as reliable, so it is best to look visually for these.

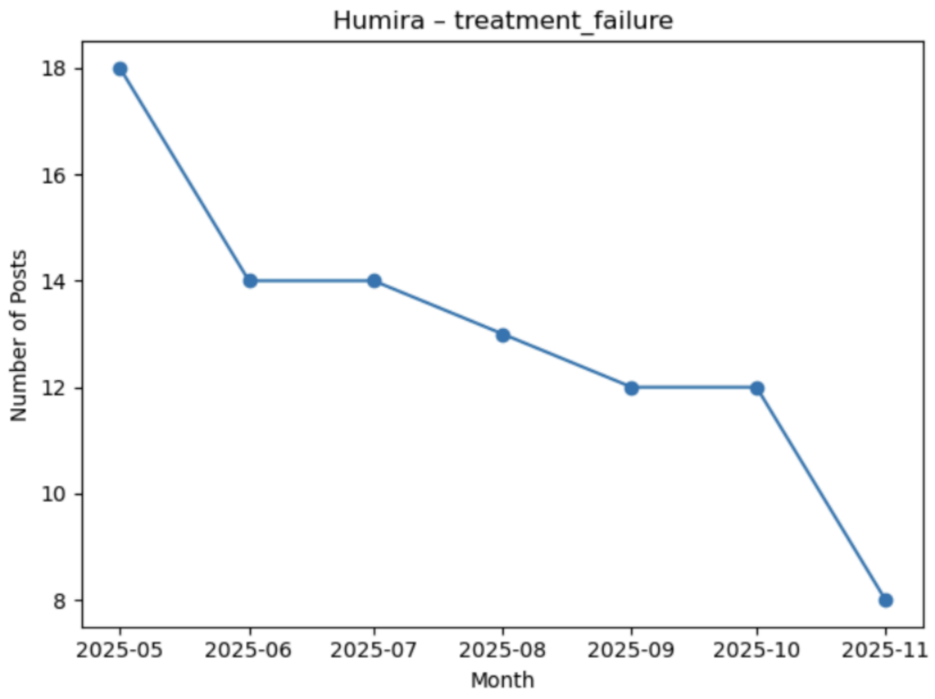
We were considering splitting the data into biweekly, but we realized that that would just introduce noise because volume of topics are unlikely to change within such short periods.

To demonstrate the unreliability of this time series, we can still demonstrate with an ETS model for individual line charts with the strongest visual trends (we cannot do seasons with ETS because it requires at least 12 per cycle).

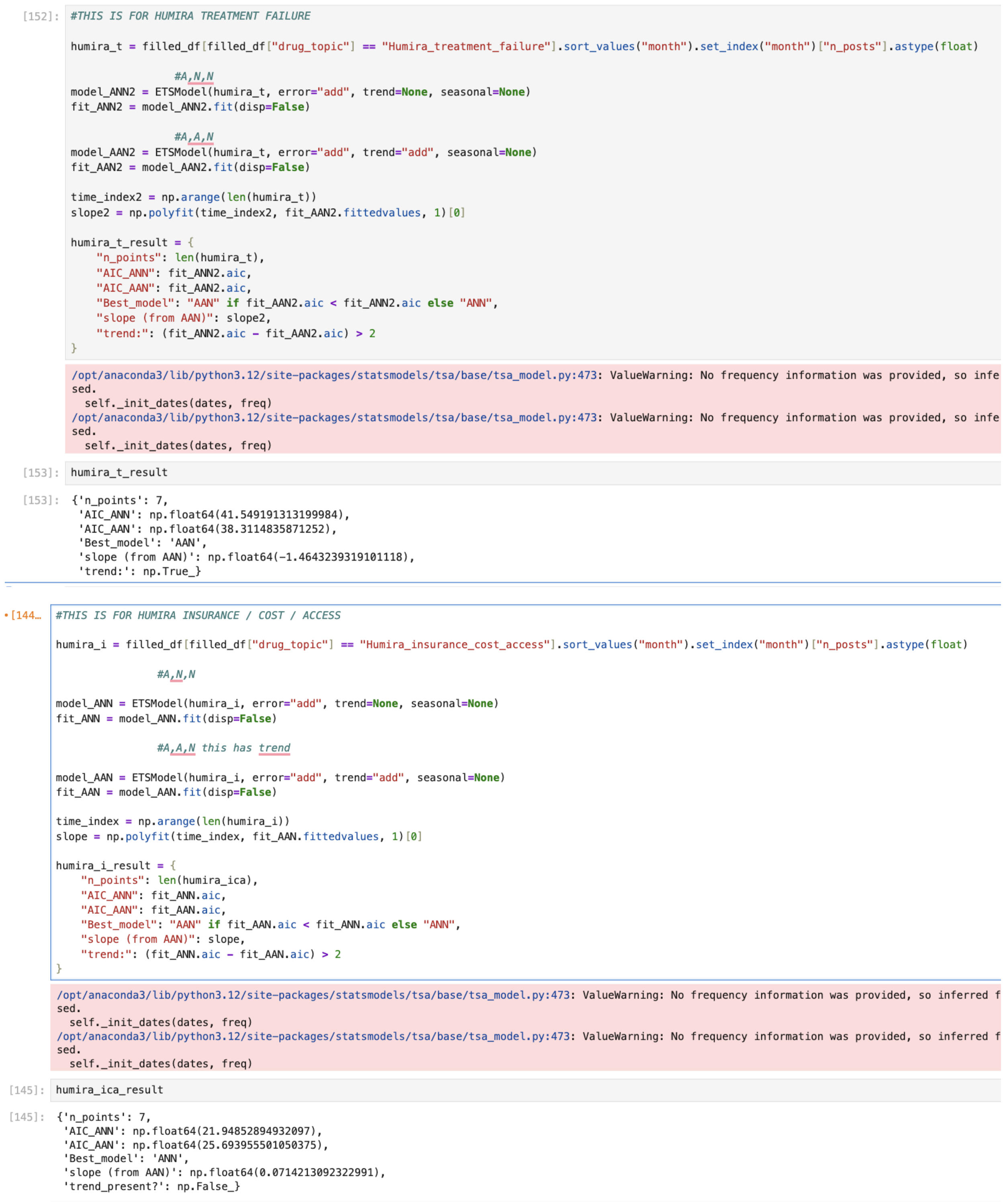
These include:

- Humira (insurance cost access)





Since these are both analysis internal to the drug, there is no need to use proportions and we can revert back to n posts.



```
[152]: #THIS IS FOR HUMIRA TREATMENT FAILURE

       humira_t = filled_df[filled_df["drug_topic"] == "Humira_treatment_failure"].sort_values("month").set_index("month")["n_posts"].astype(float)

                       #A,N,N
       model_ANN2 = ETSModel(humira_t, error="add", trend=None, seasonal=None)
       fit_ANN2 = model_ANN2.fit(disp=False)

                       #A,A,N
       model_AAN2 = ETSModel(humira_t, error="add", trend="add", seasonal=None)
       fit_AAN2 = model_AAN2.fit(disp=False)

       time_index2 = np.arange(len(humira_t))
       slope2 = np.polyfit(time_index2, fit_AAN2.fittedvalues, 1)[0]

       humira_t_result = {
           "n_points": len(humira_t),
           "AIC_ANN": fit_ANN2.aic,
           "AIC_AAN": fit_AAN2.aic,
           "Best_model": "AAN" if fit_AAN2.aic < fit_ANN2.aic else "ANN",
           "slope (from AAN)": slope2,
           "trend:": (fit_ANN2.aic - fit_AAN2.aic) > 2
       }

       /opt/anaconda3/lib/python3.12/site-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency information was provided, so infe
       sed.
         self._init_dates(dates, freq)
       /opt/anaconda3/lib/python3.12/site-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency information was provided, so infe
       sed.
         self._init_dates(dates, freq)

[153]: humira_t_result

[153]: {'n_points': 7,
        'AIC_ANN': np.float64(41.549191313199984),
        'AIC_AAN': np.float64(38.3114835871252),
        'Best_model': 'AAN',
        'slope (from AAN)': np.float64(-1.4643239319101118),
        'trend:': np.True_}

• [144…  #THIS IS FOR HUMIRA INSURANCE / COST / ACCESS

        humira_i = filled_df[filled_df["drug_topic"] == "Humira_insurance_cost_access"].sort_values("month").set_index("month")["n_posts"].astype(float)

                        #A,N,N

        model_ANN = ETSModel(humira_i, error="add", trend=None, seasonal=None)
        fit_ANN = model_ANN.fit(disp=False)

                        #A,A,N this has trend

        model_AAN = ETSModel(humira_i, error="add", trend="add", seasonal=None)
        fit_AAN = model_AAN.fit(disp=False)

        time_index = np.arange(len(humira_i))
        slope = np.polyfit(time_index, fit_AAN.fittedvalues, 1)[0]

        humira_i_result = {
            "n_points": len(humira_ica),
            "AIC_ANN": fit_ANN.aic,
            "AIC_AAN": fit_AAN.aic,
            "Best_model": "AAN" if fit_AAN.aic < fit_ANN.aic else "ANN",
            "slope (from AAN)": slope,
            "trend:": (fit_ANN.aic - fit_AAN.aic) > 2
        }

        /opt/anaconda3/lib/python3.12/site-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency information was provided, so inferred f
        sed.
          self._init_dates(dates, freq)
        /opt/anaconda3/lib/python3.12/site-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency information was provided, so inferred f
        sed.
          self._init_dates(dates, freq)

[145]:  humira_ica_result

[145]:  {'n_points': 7,
         'AIC_ANN': np.float64(21.94852894932097),
         'AIC_AAN': np.float64(25.693955501050375),
         'Best_model': 'ANN',
         'slope (from AAN)': np.float64(0.0714213092322991),
         'trend_present?': np.False_}
```

Choosing of the ETS model:

After we explored the correlations, we wanted to check whether the patterns we saw could actually be predicted. Since our dataset is very small, only 7 months. We wanted to use the ETS model because it is one of the simplest time-series models that can still detect trend and seasonality without needing a large number of observations. ETS stand for Error, Trend and Seasonality, and it breaks the time series into 3 parts.

Error: how far the prediction is from the true value.

Trend: whether the series is going up or down over time

Seasonality: repeating patterns, monthly/weekly cycles.

Why ETS fits our dataset better than the traditional ARIMA?

ARIMA usually needs a lot of observations to estimate autoregressive and moving averages parameters. Since we only have 7 months, ARIMA becomes unstable and overfits. ETS is more flexible with small data because it doesnt require the data to be stationary and can directly estimate a tren even with a short time window.

ETS Equation

ETS is a Time-series model, the equation behind it is actually very similar to what we've learned in linear regression. In regression, we predict:

Y= Intercept+ (slope x X) + error

ETS works the same way, except the predictors come from the time series itself.

There's 2 ETS Forms

The most common are:

ETS(A,N,N)

A = Additive Error

N = No trend

N = No seasonality

This is when the data doesn't show a clear upward or downward pattern

And

ETS (A,A,N)

A = Additive error

A= Additive trend

N = No seasonalty

This model tries to estimate a slope, which is helpful if the series is increasing or decreasing.

Interpreting AIC for ETS

AIC is a model selection metric that compares how well differente models fit the data.

Lower AIC = Better model Fit

Since we only have 7 months of data, it is almost impossible to formally test whether the series is white noise. Normally, if you want to check this, you would look for repeated patterns or consistent movement, but since we only have 7 points, almost anything can look random.

AIC is calculated with $2k - 2\ln(L)$

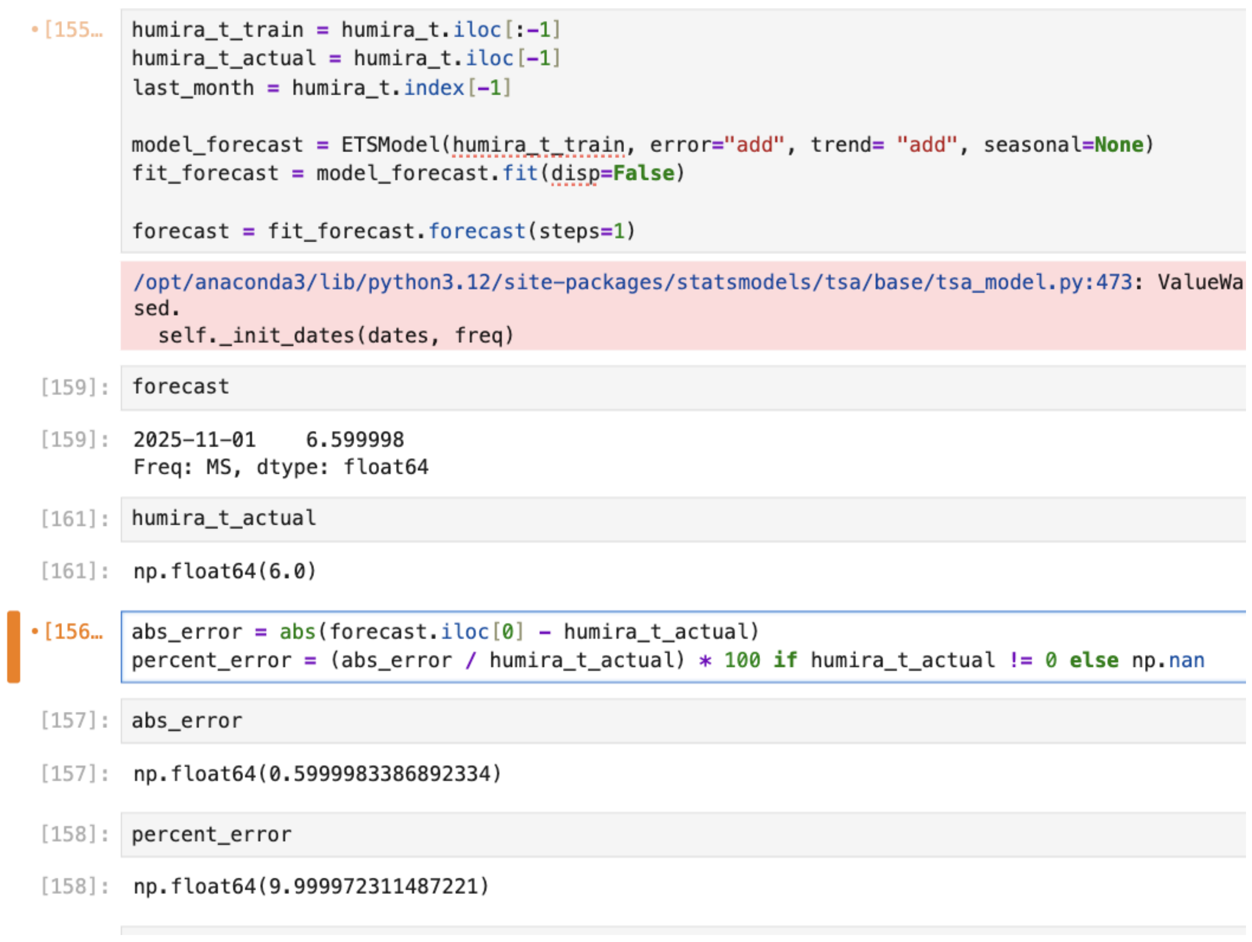
Where k is the number of parameters and the L is the likelihood, which is how well the model fits to the data

In our case, the AIC acts comparatively, in one case demonstrating that the model performs better with NO trend for humira insurance post volume, and better WITH trend for humira treatment failure post volume. Which again, has to be taken with a grain of salt because we don't have a lot of statistical power or stability due to low n.

Good looking, yet unstable time series results:

For this analysis, it is best to stick to our visual preliminary assessment of the line graphs, and only try to make conclusions about Humira's treatment failure post volume.

We can assess the forecast of the last month of our series, by removing November, training on May through October, and forecasting November.



```
•[155…  humira_t_train = humira_t.iloc[:-1]
        humira_t_actual = humira_t.iloc[-1]
        last_month = humira_t.index[-1]

        model_forecast = ETSModel(humira_t_train, error="add", trend= "add", seasonal=None)
        fit_forecast = model_forecast.fit(disp=False)

        forecast = fit_forecast.forecast(steps=1)

        /opt/anaconda3/lib/python3.12/site-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWa
        sed.
          self._init_dates(dates, freq)

[159]:  forecast

[159]:  2025-11-01    6.599998
        Freq: MS, dtype: float64

[161]:  humira_t_actual

[161]:  np.float64(6.0)

•[156…  abs_error = abs(forecast.iloc[0] - humira_t_actual)
        percent_error = (abs_error / humira_t_actual) * 100 if humira_t_actual != 0 else np.nan

[157]:  abs_error

[157]:  np.float64(0.5999983386892334)

[158]:  percent_error

[158]:  np.float64(9.999972311487221)
```

Results and Evaluation:

Our forecast was .6 off from the actual number of posts, which was 6 as opposed to 6.6.

When we look a this from a percentage error point of view, we were only 10% off from the actual value, which means for this particular topic-drug combination, the trend was quite strong as a predictor in the model despite so few points.

We can conclude that the volume of posts about humira failing as a treatment has been going down for the past few months and is trending to continue to, which is a positive result for the company. Still.

So even though we didn't have enough data for a competitive analysis, this result on its own is useful to a company who wants to track what customers are saying about the product and its effectiveness over time.

For AbbVie(Humira's parent company), their marketing team can use this information as a tailwind for their business planning, in that it is a positive business reporting to see that there is a downward trend in the discussion of Humira being a failed treatment of Crohn’s disease and that it is projected to remain this way. It is also the case that no competitor has a trend(let alone downward) as strong as this, which is a comparative tailwind.

To reiterate, our proportion correlation test p-values were unfortunately not reliable as we were seeing extreme correlations of -1 and 1 indicating very small samples that aligned proportions by chance. This is a type I error.

We determined that most of the drug-topic combinations did not have a visual trend, which was likely the result of insufficient data. We learned a valuable lesson during this project that even though you are only planning to forecast into the near future, having a couple months of data is not sufficient. Trend can only be determined with at least 3 times as many points as what we had. If we can’t safely determine whether the pattern we are seeing in the past few months is actually particular to recent consumer attitude changes vs consistently random pattern, then our abilities are limited. We were ambitious with our goal and were hindered by the lack of access to the data we needed, but we definitely saw a preview of what this analysis could yield if other trend lines looked similar to Humira’s treatment failure discussion.

Conclusion:

Marketing teams interested in doing this analysis should use more data, but follow our general algorithm for comparing ETS models with and without trend (and seasonality if applicable), and compare AIC scores. Then, correlations of the forecasts can be performed to test whether the relationship is worth noting for immediate business planning of headwinds and tailwinds.