8 Collecting the daily access in the SciELO Brazil collection

The Ratchet API[1] has the daily access count for every journal. Our goal is to find a way to obtain and use this data.

In [1]:
```
import json, re
from urllib.request import urlopen
```

In [2]:
```
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

In [3]:
```
%matplotlib inline
```

8.1 Single journal access data in a DataFrame

The entry point to get the access data of a single ISSN is:

`http://ratchet.scielo.org/api/v1/journals/<ISSN>/`

replacing `<ISSN>` by the actual journal ISSN.

That API returns a nested JSON structure, which isn't properly fit when we're reading the data with Pandas. The function below reads the dictionary of a single journal entry (i.e., a single journal of a properly loaded JSON from Ratchet) and converts the access data to a Pandas DataFrame:

In [4]:
```
def build_dataframe_from_dict(ratchet_dict):
    columns = ["total", "abstract", "html", "pdf",
               "pdfsite", "toc", "issues", "journal"]
    dicts = {
        "total": ratchet_dict,
        "pdfsite": ratchet_dict.get("other", {}).get("pdfsite", {}),
        **{key: ratchet_dict.get(key, {})
           for key in columns if key not in ["total", "pdfsite"]},
    }
    series = []
    result = pd.DataFrame()
    for key, jdata in dicts.items():
        pairs = [(pd.Timestamp(f"{pyear[1:]}-{pmonth[1:]}-{pday[1:]}"), count)
            for pyear, ydata in jdata.items()
             if re.match("y\d\d\d\d", pyear)
            for pmonth, mdata in ydata.items()
             if re.match("m\d\d", pmonth)
            for pday, count in mdata.items()
             if re.match("d\d\d", pday)
        ]
        if pairs:
            dates, counts = zip(*pairs)
            series.append(pd.Series(counts, index=dates).rename(key))
    result = pd.DataFrame(series, dtype=object).reindex(columns).T
    result[result.isna()] = 0
    return pd.DataFrame(result, dtype=int).rename_axis("date")
```

[1] http://docs.scielo.org/projects/ratchet/en/latest/api.html

8.1.1 Nauplius example

For example, let's get the access totals for Nauplius[2], whose ISSN is `0104-6497`.

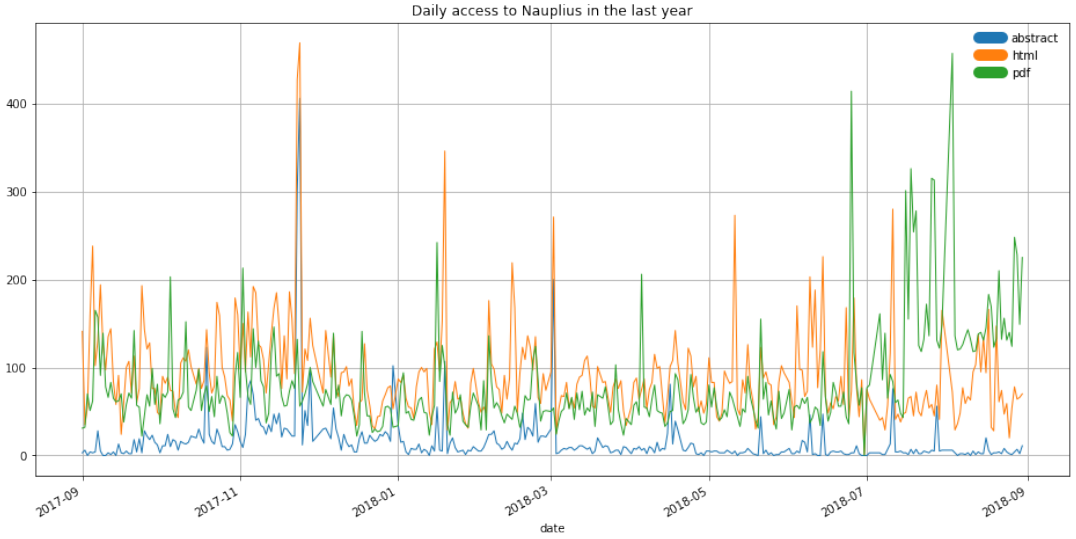
In [5]:
```
nauplius_url = "http://ratchet.scielo.org/api/v1/journals/0104-6497/"
nauplius_dict = json.load(urlopen(nauplius_url))
```

In [6]:
```
nauplius_df = build_dataframe_from_dict(nauplius_dict)
nauplius_df.head()
```

Out [6]:

date	total	abstract	html	pdf	pdfsite	toc	issues	journal
2013-09-10	65	0	4	0	0	3	26	32
2013-09-11	17	0	1	0	0	1	6	9
2013-09-12	184	3	11	5	5	24	23	113
2013-09-13	44	3	7	0	0	9	4	21
2013-09-14	15	1	1	0	0	3	2	8

In [7]:
```
nauplius_df.loc["2017-09":"2018-08", ["abstract", "html", "pdf"]].plot(
    figsize=(16, 8),
    title="Daily access to Nauplius in the last year",
    linewidth=1,
    grid=True,
)
legend = plt.gca().legend_
legend.set_frame_on(False)
for legend_line in legend.legendHandles:
    legend_line.set_linewidth(10)
    legend_line.set_solid_capstyle("round")
```



8.2 Getting all collection journals

The entry point to get the access data of all the journals from the `scl` collection is:

[2] http://www.scielo.br/scielo.php?script=sci_serial&pid=0104-6497&lng=en&nrm=iso

```
http://ratchet.scielo.org/api/v1/general/?type=journal&collection=scl
```

However, that link is effectively broken given it'll always return a *504 - Gateway Time-out* error. Therefore, we have no alternative but to request the journals one by one.

At first, let's create a façade to directly grab the daily access DataFrame.

In [8]:
```
def get_daily_accesses(issn):
    url = f"http://ratchet.scielo.org/api/v1/journals/{issn}/"
    return build_dataframe_from_dict(json.load(urlopen(url)))
```

We can get the ISSNs from either the `journals.csv` report file, or from the ArticleMeta API. Getting from the former is faster, so we'll stick with it.

In [9]:
```
journals = pd.read_csv("tabs_bra/journals.csv")
issns = journals["ISSN SciELO"].unique()
len(issns)
```

Out [9]:
```
366
```

Then we can build a tidy matrix with all the downloaded access data.

In [10]:
```
%%time
accesses = pd.concat([get_daily_accesses(issn).assign(issn=issn)
                      for issn in issns])
```

```
CPU times: user 1min 12s, sys: 801 ms, total: 1min 12s
Wall time: 1min 58s
```

In [11]:
```
# Save the accesses to avoid re-downloading it all
# (as of 2018-09-20, it had 38MB)
accesses.to_csv("scl_daily_access.csv")
```

In [12]:
```
accesses.head()
```

Out [12]:

date	total	abstract	html	pdf	pdfsite	toc	issues	journal	issn
2011-12-31	151	4	46	72	9	1	1	18	1676-5648
2012-01-01	180	3	54	96	8	3	2	14	1676-5648
2012-01-02	615	8	160	374	35	19	2	17	1676-5648
2012-01-03	688	14	196	384	46	19	6	23	1676-5648
2012-01-04	647	12	177	409	26	1	4	18	1676-5648

8.3 Evaluating the access count for the entire collection

In [13]:
```
collection_accesses = accesses.reset_index().groupby("date").sum()
collection_accesses
```

Out [13]:

date	total	abstract	html	pdf	pdfsite	toc	issues	journal
2011-12-30	5	0	2	3	0	0	0	0
2011-12-31	225441	5989	156060	43226	8221	4145	1830	5970
2012-01-01	152307	7400	74829	47314	8582	4522	2126	7534
2012-01-02	343623	7573	162905	119296	24075	10337	5088	14349

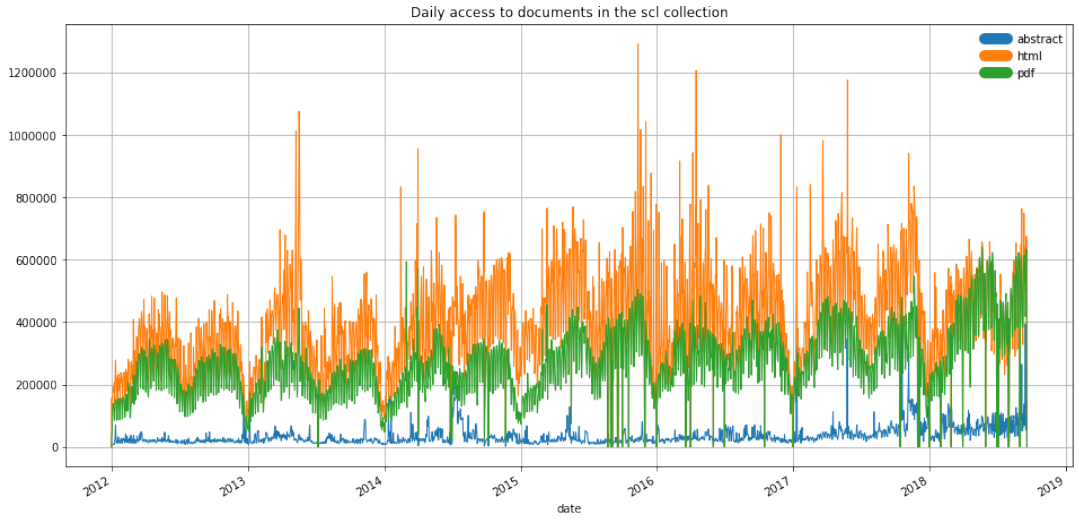
Continued on next page

date	total	abstract	html	pdf	pdfsite	toc	issues	journal
2012-01-03	383105	8967	180467	134192	27812	11660	5622	14385
2012-01-04	398325	8705	190284	139069	28324	10898	5705	15340
2012-01-05	396488	8457	192430	135552	27202	12441	5868	14538
2012-01-06	345391	8641	173659	114398	22035	9708	4528	12422
2012-01-07	333875	7174	207567	85053	16846	6057	2713	8465
2012-01-08	285171	16077	138494	88690	18048	9171	3540	11151
2012-01-09	494699	33372	228460	148660	32131	22580	7759	21737
2012-01-10	593908	69470	278209	151764	33066	29901	8644	22854
2012-01-11	507330	35529	232274	147398	31736	29530	8373	22490
2012-01-12	445707	18640	211135	141340	31054	17254	6664	19620
2012-01-13	378003	13546	181275	120512	26643	14732	6018	15277
2012-01-14	361575	20001	220618	84960	17658	7212	3036	8090
2012-01-15	273460	11428	134305	89064	18718	7808	2957	9180
2012-01-16	468103	23041	217976	148539	33452	18413	7020	19662
2012-01-17	498450	31363	226533	151281	35432	23395	8375	22071
2012-01-18	506176	31585	231393	155216	35368	22012	8548	22054
2012-01-19	439913	13640	211981	144641	33279	14515	6177	15680
2012-01-20	381625	17520	181394	119554	27381	15657	5259	14860
2012-01-21	368721	18281	216394	86868	19509	12605	4005	11059
2012-01-22	288466	12584	138208	92713	20645	9619	3550	11147
2012-01-23	455417	11364	211992	155175	35421	15156	6928	19381
2012-01-24	507045	25447	237568	155939	34286	22306	8631	22868
2012-01-25	500954	27177	229582	155120	33926	23194	8664	23291
2012-01-26	454778	16857	215447	148717	32590	16417	6279	18471
2012-01-27	433941	25181	204735	127125	28190	22850	6994	18866
2012-01-28	359657	10908	217473	88921	18797	9200	3787	10571
...	...	...	...	...	...	...	...	...
2018-08-21	1548417	125385	653653	610258	10635	17549	9301	121636
2018-08-22	1352492	69731	549938	586141	9061	13445	8436	115740
2018-08-23	1349190	100246	549611	559453	7872	12443	7669	111896
2018-08-24	1208159	107687	486725	475430	6433	12337	7572	111975
2018-08-25	856643	34563	312899	383789	5812	6969	5386	107225
2018-08-26	1075708	92784	426287	426855	4534	9062	6143	110043
2018-08-27	1432692	68008	599682	619275	4733	14027	8412	118555
2018-08-28	1411581	68844	573298	623172	4371	13407	8249	120240
2018-08-29	1492594	118631	616353	611515	3550	14948	8260	119337
2018-08-30	1414441	109344	579403	583262	3192	13578	7660	118002
2018-08-31	895	62	425	319	5	4	5	75
2018-09-01	866528	38467	324313	377309	3917	6815	6199	109508
2018-09-02	1095764	91967	434509	441412	3137	7208	6293	111238
2018-09-03	1545886	129406	649031	598335	3660	35346	9105	121003
2018-09-04	1434437	85846	587234	614996	3785	13558	8052	120966
2018-09-05	1799870	264639	764140	559685	3471	56109	10039	141787
2018-09-06	1111364	50661	428293	473339	1982	11662	7419	138008
2018-09-07	1012653	80890	376236	387011	2317	31416	6719	128064
2018-09-08	919296	58981	327936	383358	3614	9742	6380	129285
2018-09-09	1115466	105025	430741	435628	2244	9315	5604	126909
2018-09-10	1450453	68993	623706	595691	2787	13911	8602	136763
2018-09-11	1660814	112587	749664	610700	3151	36688	8854	139170
2018-09-12	1563269	137942	654267	613354	3215	16643	9829	128019
2018-09-13	1334611	67464	548346	570163	3099	12932	8478	124129
2018-09-14	1347717	142803	522307	494605	2909	47186	10227	127680
2018-09-15	1634685	394595	640161	418208	4147	53105	6799	117670
2018-09-16	1074175	54021	430230	452933	2903	8758	6048	119282

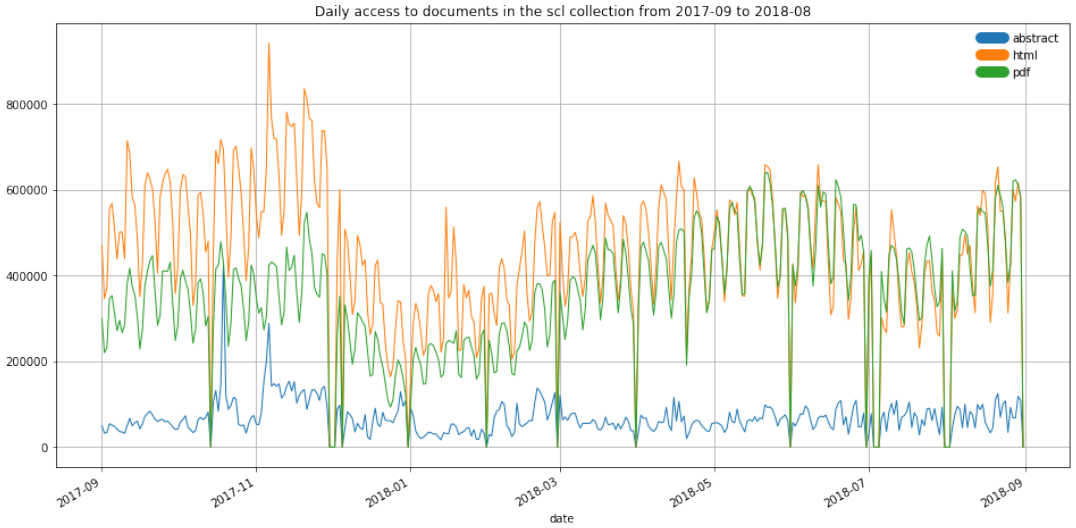
Continued on next page

date	total	abstract	html	pdf	pdfsite	toc	issues	journal
2018-09-17	1585446	133221	675839	617419	3555	16602	8888	129922
2018-09-18	1603385	131722	670879	634299	3471	20976	8862	133176
2018-09-19	1147	165	523	365	2	7	1	84

In [14]:
```
collection_accesses[["abstract", "html", "pdf"]].plot(
    figsize=(16, 8),
    title="Daily access to documents in the scl collection",
    linewidth=1,
    grid=True,
)
legend = plt.gca().legend_
legend.set_frame_on(False)
for legend_line in legend.legendHandles:
    legend_line.set_linewidth(10)
    legend_line.set_solid_capstyle("round")
```

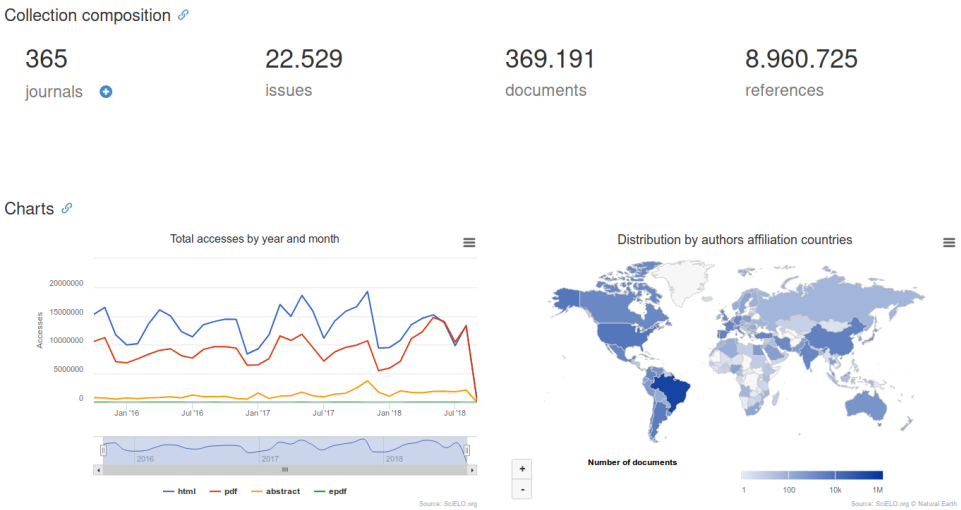


In [15]:
```
collection_accesses.loc["2017-09":"2018-08", ["abstract", "html", "pdf"]].plot(
    figsize=(16, 8),
    title="Daily access to documents in the scl collection "
          "from 2017-09 to 2018-08",
    linewidth=1,
    grid=True,
)
legend = plt.gca().legend_
legend.set_frame_on(False)
for legend_line in legend.legendHandles:
    legend_line.set_linewidth(10)
    legend_line.set_solid_capstyle("round")
```



8.4 Reproducing the SciELO Analytics Charts

In https://analytics.scielo.org we have a home/dashboard like:



Let's reproduce the left hand side of it, but the number of issues. The journal count we got from the `journals.csv`, but we could have done the same with the ArticleMeta API, which would get the current data (the notebook regarding the deindexing reason in scl has more information on how to do that).

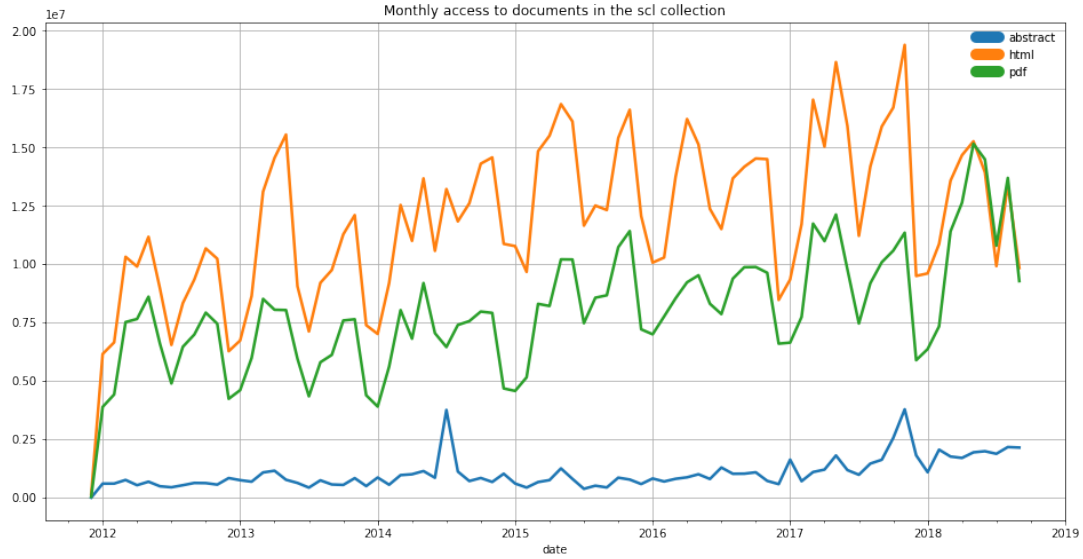
The total accesses shown are by month. Pandas allows us to group by the year and month (using `collection_accesses.index.year` and `collection_accesses.index.month` in the `groupby`), but it also has a monthly grouper, so let's use it.

In [16]:
```
monthly_data = (collection_accesses
    .groupby(pd.Grouper(freq="M"))
    .sum()
)
monthly_data[["abstract", "html", "pdf"]].plot(
    figsize=(16, 8),
    title="Monthly access to documents in the scl collection",
    linewidth=2.5,
    grid=True,
```

```
)
legend = plt.gca().legend_
legend.set_frame_on(False)
for legend_line in legend.legendHandles:
    legend_line.set_linewidth(10)
    legend_line.set_solid_capstyle("round")
monthly_data.tail()
```

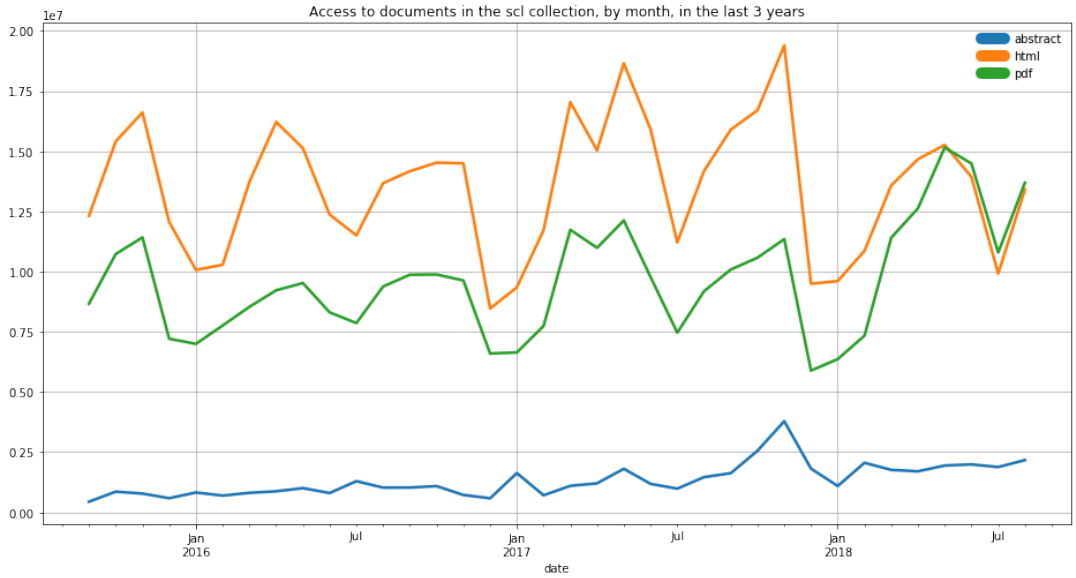
Out [16]:

date	total	abstract	html	pdf	pdfsite	toc	issues	journal
2018-05-31	36611020	1944622	15263095	15155345	95756	409505	231829	3510868
2018-06-30	34278168	1991711	13936406	14491219	67733	413571	214659	3162869
2018-07-31	26470272	1882551	9918115	10792576	81643	480489	187428	3127470
2018-08-31	33371045	2171366	13414688	13694333	180712	405378	212873	3291695
2018-09-30	24156966	2149395	9838355	9278810	57366	407979	142398	2282663



In [17]:

```
monthly_data.loc["2015-09":"2018-08", ["abstract", "html", "pdf"]].plot(
    figsize=(16, 8),
    title="Access to documents in the scl collection, by month, "
          "in the last 3 years",
    linewidth=2.5,
    grid=True,
)
legend = plt.gca().legend_
legend.set_frame_on(False)
for legend_line in legend.legendHandles:
    legend_line.set_linewidth(10)
    legend_line.set_solid_capstyle("round")
```



8.4.1 Week day analysis

In which day of the week does SciELO Brazil articles have more accesses?

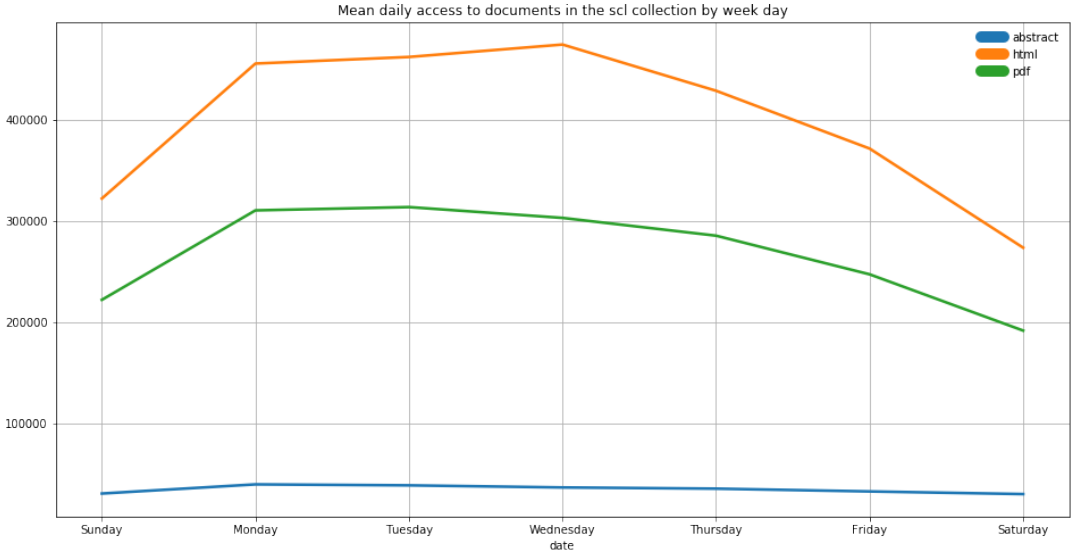
In [18]:
```
collection_accesses_weekday = (
    collection_accesses.groupby([collection_accesses.index.weekday,
                                 collection_accesses.index.weekday_name])
                       .mean()
                       .reset_index(0, drop=True)
).iloc[range(-1, 6)]
collection_accesses_weekday[["abstract", "html", "pdf"]].plot(
    figsize=(16, 8),
    title="Mean daily access to documents in the scl collection by week day",
    xticks=range(7),
    linewidth=2.5,
    grid=True,
)
legend = plt.gca().legend_
legend.set_frame_on(False)
for legend_line in legend.legendHandles:
    legend_line.set_linewidth(10)
    legend_line.set_solid_capstyle("round")
collection_accesses_weekday
```

Out [18]:

date	total	abstract	html	pdf	pdfsite	toc	issues	journal
Sunday	667275.151862	30531.512894	321898.217765	221847.630372	29626.472779	15768.830946	4335.234957	43267.252149
Monday	929982.943020	39583.225071	455172.199430	310183.421652	41974.880342	22807.145299	7481.689459	52780.381766
Tuesday	938702.829060	38615.227920	461691.746439	313432.242165	42595.868946	23383.911681	7319.373219	51664.458689
Wednesday	934482.729345	36531.743590	473964.897436	302749.301994	42004.840456	22194.301994	7123.737892	49913.905983

Continued on next page

date	total	abstract	html	pdf	pdfsite	toc	issues	journal
Thursday	866744.737143	35349.480000	428322.605714	285194.960000	39702.020000	22451.720000	6708.942857	49015.008571
Friday	760144.531609	32603.298851	371228.370690	247018.795977	34579.117816	21581.054598	6332.298851	46801.594828
Saturday	581277.145714	29985.014286	273333.277143	191460.714286	26959.111429	15213.351429	4160.202857	40165.474286



8.5 Daily access by thematic area

As we have the data labeled for each ISSN, we can apply some filter using some information about the journal. For example, we can look for plots for each thematic area.

In [19]:
```
areas = [field for field in journals.columns
              if field.startswith("title is")]
areas
```

Out [19]:
```
['title is agricultural sciences',
 'title is applied social sciences',
 'title is biological sciences',
 'title is engineering',
 'title is exact and earth sciences',
 'title is health sciences',
 'title is human sciences',
 'title is linguistics, letters and arts',
 'title is multidisciplinary']
```

At first we need an *edge list* of all thematic areas that an ISSN is assigned to.

In [20]:
```
stacked_areas = journals.set_index("ISSN SciELO")[areas].stack()
stacked_areas = (stacked_areas[stacked_areas == 1]
    .reset_index(1)["level_1"]
    .rename("area")
    .apply(lambda x: x[len("title is "):].capitalize())
    .rename_axis("issn")
    .reset_index()
```

```
)
stacked_areas.head(15)
```

Out [20]:

	issn	area
0	1676-5648	Applied social sciences
1	0101-8108	Health sciences
2	0034-7701	Human sciences
3	0102-261X	Exact and earth sciences
4	1516-9332	Health sciences
5	0104-5687	Health sciences
6	0103-1759	Exact and earth sciences
7	0101-8175	Biological sciences
8	0101-3122	Agricultural sciences
9	2179-6491	Health sciences
10	1980-6523	Health sciences
11	0373-5524	Biological sciences
12	0373-5524	Exact and earth sciences
13	0100-4239	Biological sciences
14	0100-4239	Exact and earth sciences

That can be joined with the Ratchet data, giving us the desired data to be able to plot the accesses count by thematic area.

In [21]:
```
accesses_by_area = (pd.merge(accesses.reset_index(), stacked_areas)
    .groupby(["date", "area"])
    .sum()
    .stack()
    .rename("count")
    .reset_index()
    .rename(columns={"level_2": "type"})
)
accesses_by_area
```

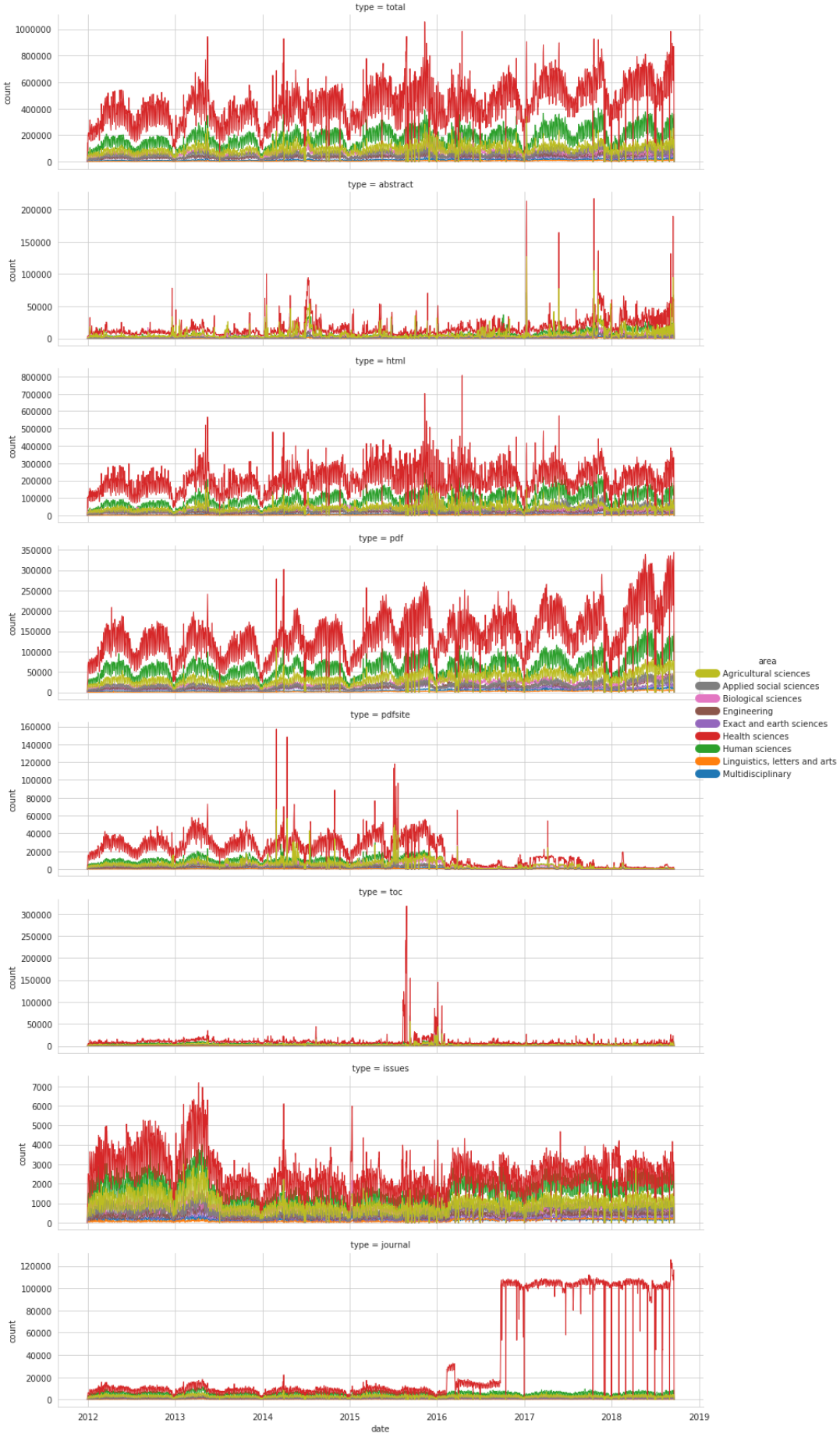
Out [21]:

	date	area	type	count
0	2011-12-30	Biological sciences	total	1
1	2011-12-30	Biological sciences	abstract	0
2	2011-12-30	Biological sciences	html	0
3	2011-12-30	Biological sciences	pdf	1
4	2011-12-30	Biological sciences	pdfsite	0
5	2011-12-30	Biological sciences	toc	0
6	2011-12-30	Biological sciences	issues	0
7	2011-12-30	Biological sciences	journal	0
8	2011-12-30	Health sciences	total	3
9	2011-12-30	Health sciences	abstract	0
10	2011-12-30	Health sciences	html	2
11	2011-12-30	Health sciences	pdf	1
12	2011-12-30	Health sciences	pdfsite	0
13	2011-12-30	Health sciences	toc	0
14	2011-12-30	Health sciences	issues	0
15	2011-12-30	Health sciences	journal	0
16	2011-12-30	Human sciences	total	2
17	2011-12-30	Human sciences	abstract	0
18	2011-12-30	Human sciences	html	1

Continued on next page

	date	area	type	count
19	2011-12-30	Human sciences	pdf	1
20	2011-12-30	Human sciences	pdfsite	0
21	2011-12-30	Human sciences	toc	0
22	2011-12-30	Human sciences	issues	0
23	2011-12-30	Human sciences	journal	0
24	2011-12-31	Agricultural sciences	total	28408
25	2011-12-31	Agricultural sciences	abstract	1330
26	2011-12-31	Agricultural sciences	html	18101
27	2011-12-31	Agricultural sciences	pdf	6296
28	2011-12-31	Agricultural sciences	pdfsite	982
29	2011-12-31	Agricultural sciences	toc	551
...	...	...	...	...
176002	2018-09-19	Health sciences	html	274
176003	2018-09-19	Health sciences	pdf	212
176004	2018-09-19	Health sciences	pdfsite	0
176005	2018-09-19	Health sciences	toc	2
176006	2018-09-19	Health sciences	issues	0
176007	2018-09-19	Health sciences	journal	79
176008	2018-09-19	Human sciences	total	189
176009	2018-09-19	Human sciences	abstract	12
176010	2018-09-19	Human sciences	html	89
176011	2018-09-19	Human sciences	pdf	81
176012	2018-09-19	Human sciences	pdfsite	0
176013	2018-09-19	Human sciences	toc	4
176014	2018-09-19	Human sciences	issues	1
176015	2018-09-19	Human sciences	journal	2
176016	2018-09-19	Linguistics, letters and arts	total	11
176017	2018-09-19	Linguistics, letters and arts	abstract	0
176018	2018-09-19	Linguistics, letters and arts	html	10
176019	2018-09-19	Linguistics, letters and arts	pdf	1
176020	2018-09-19	Linguistics, letters and arts	pdfsite	0
176021	2018-09-19	Linguistics, letters and arts	toc	0
176022	2018-09-19	Linguistics, letters and arts	issues	0
176023	2018-09-19	Linguistics, letters and arts	journal	0
176024	2018-09-19	Multidisciplinary	total	18
176025	2018-09-19	Multidisciplinary	abstract	1
176026	2018-09-19	Multidisciplinary	html	10
176027	2018-09-19	Multidisciplinary	pdf	7
176028	2018-09-19	Multidisciplinary	pdfsite	0
176029	2018-09-19	Multidisciplinary	toc	0
176030	2018-09-19	Multidisciplinary	issues	0
176031	2018-09-19	Multidisciplinary	journal	0

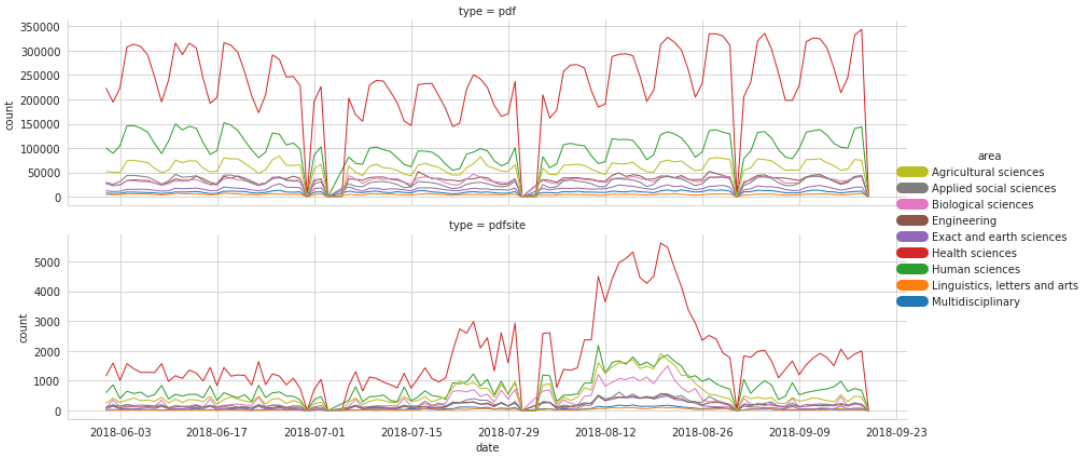
In [22]:
```
areas_order = sorted(accesses_by_area["area"].unique())
with sns.axes_style("whitegrid"):
    sns.FacetGrid(accesses_by_area,
                  row="type", hue="area", hue_order=areas_order[::-1],
                  aspect=4, sharey=False) \
       .map(sns.lineplot, "date", "count", linewidth=1) \
       .add_legend(label_order=areas_order)
    for legend_line in plt.gcf().legends[0].legendHandles:
        legend_line.set_linewidth(10)
```


type = total
type = abstract
type = html
type = pdf
type = pdfsite
type = toc
type = issues
type = journal
count
date
area
Agricultural sciences
Applied social sciences
Biological sciences
Engineering
Exact and earth sciences
Health sciences
Human sciences
Linguistics, letters and arts
Multidisciplinary

In [23]:
```
with sns.axes_style("whitegrid"):
    sns.FacetGrid(accesses_by_area[accesses_by_area["date"] >= "2017-09"],
                  row="type", hue="area", hue_order=areas_order[::-1],
                  aspect=4, sharey=False) \
       .map(sns.lineplot, "date", "count", linewidth=1) \
       .add_legend(label_order=areas_order)
    for legend_line in plt.gcf().legends[0].legendHandles:
        legend_line.set_linewidth(10)
```


type = total
type = abstract
type = html
type = pdf
type = pdfsite
type = toc
type = issues
type = journal
count
date
2017-10
2017-12
2018-02
2018-04
2018-06
2018-08
2018-10
area
Agricultural sciences
Applied social sciences
Biological sciences
Engineering
Exact and earth sciences
Health sciences
Human sciences
Linguistics, letters and arts
Multidisciplinary

In [24]:
```
with sns.axes_style("whitegrid"):
    sns.FacetGrid(accesses_by_area[
                        (accesses_by_area["date"] >= "2018-06") &
                        accesses_by_area["type"].isin(["pdf", "pdfsite"])
                    ],
                    row="type", hue="area", hue_order=areas_order[::-1],
                    aspect=4, sharey=False) \
        .map(sns.lineplot, "date", "count", linewidth=1) \
        .add_legend(label_order=areas_order)
    for legend_line in plt.gcf().legends[0].legendHandles:
        legend_line.set_linewidth(10)
```



8.6 Accesses report

A faster approach to get access data would be the `accesses_by_journals.csv` report file. It doesn't have the daily access, but an yearly access summary.

In [25]:
```
accesses_by_journals = pd.read_csv("tabs_bra/accesses_by_journals.csv")
accesses_by_journals.columns
```

Out [25]:
```
Index(['extraction date', 'study unit', 'collection', 'ISSN SciELO', 'ISSN's',
       'title at SciELO', 'title thematic areas',
       'title is agricultural sciences', 'title is applied social sciences',
       'title is biological sciences', 'title is engineering',
       'title is exact and earth sciences', 'title is health sciences',
       'title is human sciences', 'title is linguistics, letters and arts',
       'title is multidisciplinary', 'title current status', 'publishing year',
       'accesses year', 'accesses to html', 'accesses to abstract',
       'accesses to pdf', 'accesses to epdf', 'total accesses'],
      dtype='object')
```

In [26]:
```
accesses_fields = [field for field in accesses_by_journals
                         if "accesses" in field and "to" in field]
accesses_fields
```

Out [26]:

```
['accesses to html',
 'accesses to abstract',
 'accesses to pdf',
 'accesses to epdf',
 'total accesses']
```

In [27]:

```
ajdata = (accesses_by_journals
    .set_index(["accesses year"] + accesses_fields)
    [areas]
    .rename_axis("area", axis="columns")
    .stack()
    .rename("temp")
)
ajdata = ajdata[ajdata == 1].reset_index().drop(columns=["temp"])
ajdata[::5000]
```

Out [27]:

	accesses year	accesses to html	accesses to abstract	accesses to pdf	accesses to epdf	total accesses	area
0	2011	2	0	16	0	18	title is applied social sciences
5000	2015	29792	1331	13466	117	44706	title is human sciences
10000	2018	47970	18586	32856	53	99465	title is engineering
15000	2011	12	1	12	0	25	title is engineering
20000	2013	27117	2954	0	0	30071	title is biological sciences
25000	2011	22	3	9	0	34	title is agricultural sciences
30000	2015	208075	6624	91073	1584	307356	title is health sciences
35000	2017	28125	761	13861	11	42758	title is applied social sciences

In [28]:

```
ajdata_tidy = (
    ajdata
    .groupby(["accesses year", "area"])
    .sum()
    .rename_axis("type", axis="columns")
    .stack()
    .rename("count")
    .reset_index()
    .assign(area=lambda df: df["area"].str.replace("title is ", ""),
            type=lambda df: df["type"].str.replace("accesses to ", "")
                                      .str.replace(" accesses", ""))
)
ajdata_tidy
```

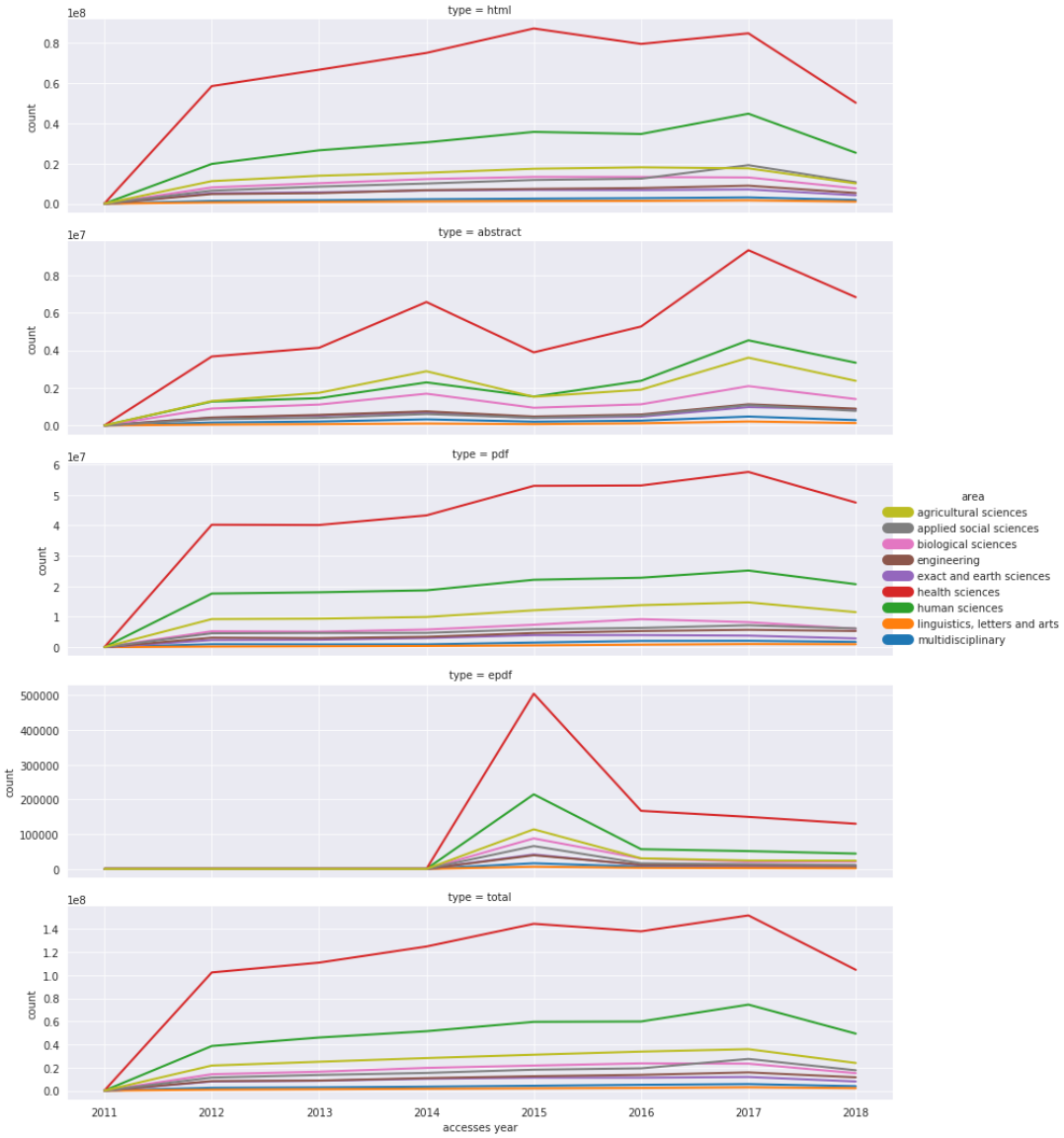
Out [28]:

	accesses year	area	type	count
0	2011	agricultural sciences	html	18048
1	2011	agricultural sciences	abstract	1325
2	2011	agricultural sciences	pdf	6231
3	2011	agricultural sciences	epdf	0
4	2011	agricultural sciences	total	25604
5	2011	applied social sciences	html	3827
6	2011	applied social sciences	abstract	180
7	2011	applied social sciences	pdf	1835
8	2011	applied social sciences	epdf	0
9	2011	applied social sciences	total	5842
10	2011	biological sciences	html	24371
11	2011	biological sciences	abstract	1074
12	2011	biological sciences	pdf	4769
13	2011	biological sciences	epdf	0
14	2011	biological sciences	total	30214
15	2011	engineering	html	5869
16	2011	engineering	abstract	477
17	2011	engineering	pdf	2174
18	2011	engineering	epdf	0
19	2011	engineering	total	8520
20	2011	exact and earth sciences	html	5394
21	2011	exact and earth sciences	abstract	553
22	2011	exact and earth sciences	pdf	1678
23	2011	exact and earth sciences	epdf	0
24	2011	exact and earth sciences	total	7625
25	2011	health sciences	html	105118
26	2011	health sciences	abstract	3279
27	2011	health sciences	pdf	22651
28	2011	health sciences	epdf	0
29	2011	health sciences	total	131048
...	...	...	...	...
330	2018	engineering	html	5342225
331	2018	engineering	abstract	893117
332	2018	engineering	pdf	5309616
333	2018	engineering	epdf	9630
334	2018	engineering	total	11554588
335	2018	exact and earth sciences	html	4215254
336	2018	exact and earth sciences	abstract	856885
337	2018	exact and earth sciences	pdf	2865870
338	2018	exact and earth sciences	epdf	8433
339	2018	exact and earth sciences	total	7946442
340	2018	health sciences	html	50244334
341	2018	health sciences	abstract	6822204
342	2018	health sciences	pdf	47506109
343	2018	health sciences	epdf	130127
344	2018	health sciences	total	104702774
345	2018	human sciences	html	25342659
346	2018	human sciences	abstract	3333890
347	2018	human sciences	pdf	20685944
348	2018	human sciences	epdf	44039
349	2018	human sciences	total	49406532
350	2018	linguistics, letters and arts	html	1022854
351	2018	linguistics, letters and arts	abstract	125274
352	2018	linguistics, letters and arts	pdf	958338

Continued on next page

	accesses year	area	type	count
353	2018	linguistics, letters and arts	epdf	1834
354	2018	linguistics, letters and arts	total	2108300
355	2018	multidisciplinary	html	1807694
356	2018	multidisciplinary	abstract	283093
357	2018	multidisciplinary	pdf	1710317
358	2018	multidisciplinary	epdf	4832
359	2018	multidisciplinary	total	3805936

In [29]:
```
label_order = sorted(ajdata_tidy["area"].unique())
with sns.axes_style("darkgrid"):
    sns.FacetGrid(ajdata_tidy,
                  row="type", hue="area", hue_order=label_order[::-1],
                  aspect=4, sharey=False) \
       .map(sns.lineplot, "accesses year", "count", linewidth=2) \
       .add_legend(label_order=label_order)
    for legend_line in plt.gcf().legends[0].legendHandles:
        legend_line.set_linewidth(10)
```



Here we have the *epdf* information and another total (the one of these 4 types), when compared with the Ratchet API data.