LinkUp “RAW 2.0 Enhanced” Change Document

# Purpose: To outline the changes from RAW2.0 to RAW2.0 Enhanced

# Summary:

RAW2.0 Enhanced (R2E or “Gold Copy”) provides many game changing enhancements versus RAW2 (R2). We have:

* Re-envisioned the entire dataset and analytics to streamline usage, data footprint, and usability
* Made the dataset completely *point in time* (PIT), with each day as you would have seen the data on that day.
* Separated mutable from immutable data to streamline files and ingestion
* Added a detailed and complete “scrapelog”, filling the gaps in the R2 company\_scrape\_log.
* Simplified reference data, to 1 user friendly point in time file
* Greatly simplified the file structure. Eliminated file “bloat”.
* Simplified file ingestion significantly
* Dramatically reduced file storage footprint
* Fixed the systemic “reposting” issue.
* Added a significant amount of PIT analytics and metadata.
* Added a simple python downloader.
* Fixed “bad” raw data, including malformed state and country fields, and null date fields.
* Fixed faulty encoding
* Fixed bad “new” created dates, where a hash would show up with 2 or more created dates.
* We remove FTP access, shifting to AWS S3 as it offers far superior access and automated sync capabilities
* AWS Glue
* Removed duplicate ‘hash’
* Deal with hash that have multiple different “created” dates over time. Should never happen.

# RAW:

Let’s start with the actual RAW jobs data and descriptions, the “heart” of our data and analytics.

R2 has a full archive “jobs” file, and a full archive “descriptions” file. It also has a daily “delta” file which combined the daily “jobs” file with the daily “descriptions” file. These files are located in the following directories:

/JDE/Standard/Feeds/Raw Full Job Records

/JDE/Standard/Feeds/Raw Full Job Descriptions

/JDE/Standard/Feeds/Raw Daily Job Records and Descriptions

Then further broken down to each year/month.

For daily files, you download, extract the daily file, which would create 2 new files 🡪 the daily “jobs” file, and the daily “descriptions” file. While the “jobs” file cleaned out all the noisy and legacy “ad” and “hosted” jobs from RAW1, the “descriptions” file did not, leaving the user to remove them.

Typically, a client would download an archive, then append the dailies to keep an up to date data set.

These RAW files were only available back to 2018, with many of the initial files suffering from missing or malformed data fields, as well as periodic changes in encoding. The archives really don’t start until 9/2018, so from a practical perspective that would be your best starting point from a point in time perspective. This is actually less than the previous RAW1 which went back to 2015.

But our daily jobs data history goes back to 2007-08-03. While the data is accurate, in R2 we lost the PIT nature of it. The changes described below create a perfect PIT representation of the raw data, and greatly improve usability, veracity, and data credibility while expanding the overall analytics options.

## A look at the old files vs. the new.

R2E is formatted completely in the ultra slim, fast, and flexible [Apache Arrow “parquet” format](https://parquet.apache.org/), built for Big Data (note, if requested we can provide in .csv format, but you will lose many of the big data benefits of this format). R2, as was Raw1, is in csv and xml.

## Current R2E vs. R2 raw files and schemas. Fields in GREEN are added, in RED are deleted:

## R2E raw files R2 raw files

|  |  |
| --- | --- |
| **jobs\_base-** This is an enhanced “jobs” table, which is just the *immutable* data for a given hash, with some important modifications. It exists as a single file partitioned by ‘created\_pit (see below)’. It is pure PIT, all you do is ingest the file once, and add the created\_date you need every day.   * jobid – the jobid is a “smart” identifier.  The first 8 digits are the created date, *point in time*, meaning the date that the client would have seen that the job was createdà the actual production date (see below “created\_pit”).  The next 5 digits are the company\_id.  The final 5 digits is the specific number of the job that was created that day for that company\_id.  For example 202006090000100001 would be the jobid for Target for the first job that was created on 2020-06-09.  While “hash” is random, this identifier has information and therefore value, flexibility and simplicity. * repostFlag – 0 or 1 based on if a repost or not * company\_id * title * city * state * zip * country * created * createdPIT (this is the actual PIT production created date, used in the jobid, and to fix the created and production mismatch issue.) * parenthash (this is the parent “hash” (see the “hash” in the jobs schema on the right) for the jobid.  With this we can deal perfectly with all reposts) | **jobs**-  Available in:  Raw\_daily\_yyyy-mm-dd.csv.gz  Raw\_job\_archive\_yyyy-mm-dd.tar.gz  This is the primary raw file(s) which shows 1 record for each unique job, which each records changes over time. Some fields are mutable like “last\_checked”, which creates the PIT issue.   * Hash – the unique hash assigned based on unique URL of the job post. Added as parenthash in the jobs\_base file to the left * company\_id * company\_name – moved to mutable PIT reference file * title * city * state * zip * country * created * last\_checked – captured in PIT jobs\_log below * last\_updated * onet\_occupation\_code – moved to the PIT analyticsEnhancedJob file as socCode2010. * url - moved to the PIT analyticsEnhancedJob file as url. |
| **jobs\_log-** With the enhanced *company\_scrape\_log (see below)*, and the *jobs\_base* as a foundation, we create a hash level ***true*** point in time record, all the way back to day 1.  So we would be able to look at 2007-08-03, and see the “daily” view on that date.  **The result though is a perfect PIT dataset, at the atomic “hash” level.**  The dataset is **approaching 5 BILLION records**!  BUT, because of how it is constructed it, it requires only ~17GB of storage space. Pull the whole file up to date, or just select the “scrape\_date(s)” you want.   * jobid – see above * scrapeDate – the PIT scrape date for the job * addremoveflag - The addremoveflag is also SMART!!!  It can be either 1,0,-1, where 1 means the jobid was created (the PIT created date as you would have seen in production-historically many jobs would have a created date of the day before you actually saw the job in our production process. This is true PIT), 0 means it was active and checked (this replaces ALL of the historical possible “last\_checkeds” in history, -1 means it was deleted (the PIT deleted date). | None |
| **company\_scrape\_log**- this is a great meta data enhancement which shows at the company\_id level each day the company was scraped, whether the company scraper was “changed”.   * company\_id, * actionDate, * actionType. (i.e. “scraped”, or “scrape\_changed” etc.) | **Raw\_company\_scrape\_log\_daily\_yyyy\_mm\_dd.csv.gz** – this is a first attempt to make a company level scrape log. This is located in the following directories:   * Raw Daily Company Scrape Log * Raw Full Company Scrape Log   Unfortunately these are incomplete in terms of detail at the company level, as well as actually missing companies. It also does not account for companies that were actually deleted from the scrape cycle. |
| **descriptions-**   * hash * description * hash1 – this is so you can iterate through this large data set easily in the file/partition. Simplifies processing * created\_pit – this is so you can pull specific days or just add a new day. Simplifies processing | **descriptions-**   * hash * description |
| **reference-** this is a great enhancement where we consolidate all of our reference data, PIT. This new file allows us to remove the bloat from the analytics files, as well as add new sources of identifiers at will, such as we did with Factset. This parquet is partitioned by refsource (i.e. Refinitiv/Factset/LinkUp/SmartMarketData etc.) as well as reftype (i.e. ticker, cusip, sedol, LEI, PermID etc.) for a fast and easy way to grab all at once, or just what you want.   * company\_id * refsource – the source of the data * refsource\_id – the primary identifier used by the refsource * reftype – the type of reference data – i.e. URL, or company\_name, sedol, or ticker etc. * start\_date – the start date of the mapping * end\_date – the end date of the mapping * value – the value (i.e. ‘AAPL’) * thrudate – the date that the data was added to the file | The R2 reference files are located in various places like:   * Raw Company Reference * FS Daily Company Reference * Raw Daily PIT Company Reference * Raw Full PIT Ticker Reference * Raw Full PIT Company Reference   Each with various files with different formats, and significant duplication and file bloat. The new file is “reference”, is pure point in time, flexible and SCALEABLE.  This format does not allow for scalability, and is confusing. |
| **auxiliary – this is a file to hold auxiliary “helper” tables, like the 2010 and 2018 SOC code descriptions for example which are already posted. Wholly scaleable and simple to navigate.** | None |

### File Sizes

|  |  |
| --- | --- |
| **jobs\_base** size is ~8GB zipped, and that is IT. Every day you just consume the new data which is easily ingested via the created\_pit partition, and is very small (MBs) | **jobs** - All Dailies are ~500GB zipped, and only go back to 2016. This file set grows ~300MB per day ZIPPED. These do include descriptions  All jobs archives are ~520GB, and only go back to 2016 as well. These archives grow almost 1GB per month. So over 1TB of data stored here, growing 10GB per month. (these are stats vs. Raw1, R2 has significantly less history so while lower foot print, growing roughly the same) |
| **jobs\_log** size is ~ 15GB zipped and that is IT. Every day you just consume the new data which is easily ingested via the scrape\_date partition, and is very small (~40KB) | NONE |
| **scrapelog** size is inconsequential at 150MB growing very little per day | raw\_company\_scrape\_log\_daily\_2020-08-20.csv.gz is ~47KB per day. The monthly archive is 70MB, every month. |
| **descriptions** size is around 70GB zipped, total. Adding just MBs per day. No archives, deltas, etc. | **descriptions** size is included in the above “jobs dailies”, **but has its own archive, which represents ~3.2TB of data, adding 125GB zipped per month! (using Raw 1 stats as R2 archive uses the Raw1 file)** |
| **reference** size is inconsequential at <10MB. One file. No bloat. | The R2 reference files are split up in various archive and daily formats, with different files for different sources (i.e. Factset, or LinkUp). The sizes are similar, but grow much quicker with archives and dailies separate. |

The end result is the following RAW data file profile:

R2E Files (all parquet) R2 Files (csv and xml)

|  |  |
| --- | --- |
| jobs\_base | linkup\_raw\_daily\_yyyy-mm-dd.tar.gz  raw\_job\_archive\_yyyy-mm-dd.tar.gz |
| jobs\_log | linkup\_raw\_daily\_yyyy-mm-dd.csv  raw\_job\_archive\_yyyy-mm-dd.tar.gz |
| descriptions | linkup\_raw\_daily\_yyyy-mm-dd\_descriptions.xml  linkup\_job\_descriptions\_yyyy-mm-dd.tar.gz |
| scrapelog | -raw\_company\_scrape\_log\_daily\_yyyy-mm-dd.csv.gz  -raw\_company\_scrape\_log\_full\_yyyy-mm-dd.csv.gz |
| reference | -fs\_company\_reference\_daily\_yyyy-mm-dd.csv.gz  -raw\_pit\_company\_reference\_full\_yyyy-mm-dd.csv.gz  -raw\_company\_scrape\_log\_full\_yyyy-mm-dd.csv.gz  -company\_reference\_yyyy-mm-dd.csv.gz  - raw\_pit\_company\_reference\_daily\_yyyy-mm-dd.csv.gz  - fs\_company\_reference\_daily\_yyyy-mm-dd.csv.gz |
| auxiliary | none |
| #files/size footprint/daily growth – **6/<100GB/<100MB** | **1000’s/~4+TB/~11GB** |

## RAW Daily Data

We spent a lot of time cleaning up Title, City, State, and Country fields which were wrong in the initial R2. Still not perfect, but much improved…

## Reposts!

What is the repost issue? Historically, on 2016-01-23, we saw the raw jobs data for 2010-01-01 as it was on 2016-01-23. We had no record backwards of While differences are fairly insignificant, there is a systemic upward “drift” in job counts over time, based on the analytics and data capture methodology, due to companies “reposting” jobs they had previously removed for a period. For instance, company XYZ may have 10 jobs active on 2010-01-01. On the next “scrape” on 2010-01-02 they have 9 jobs, having removed 1. There are 10 scrape days where no changes occurred, so on 2010-01-12 we still show 9 jobs, and the time series for the period would show 10, then 9,9,9...9. Now company XYZ adds back the EXACT SAME JOB (based on the unique URL) on 2010-01-13. For 2010-01-13 we will show 10 jobs again. *BUT*… the time series will now adjust back that that job existed for ALL the days, so it would look like 10,then 10,10,10…10. A “drift” updward of 1 job.

Again, NOT a PIT representation of what we knew on a specific day. We fix this nuance and many other things in R2E.

We eliminated the repost issue by capturing when companies remove jobs, and if they add back, creating a “new” job – that still references the original “hash”. This is accomplished with the jobid, and the parenthash in the jobs\_base\_file. In doing this, we eliminate the “drift” that occurs non-PIT. We also eliminate the the problem of having a deleted date be NULL, then have a date, then suddenly be NULL again. Our goal is to capture when companies have jobs actually posted. This modified look gives you just that.

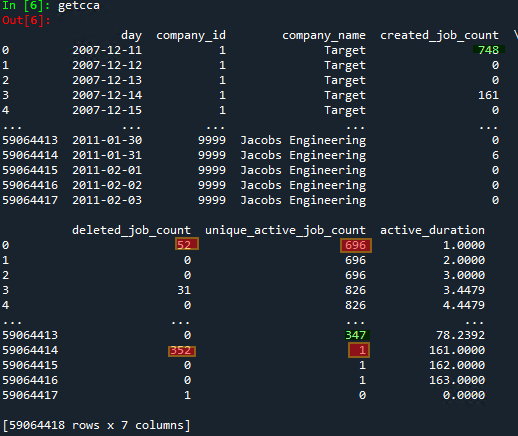
## Last\_checked, deleted and created

In R2, the team made enhancements to try and get a more accurate picture of when a job was “active. Specifically, we added an “official” delete date based on the actual time a job was scraped, and it DIDN’T exist anymore. In Raw1, we created an “inferred” delete date based on all of the inferred scrape information we had down to the company\_id level. Very good, but not perfectly accurate.

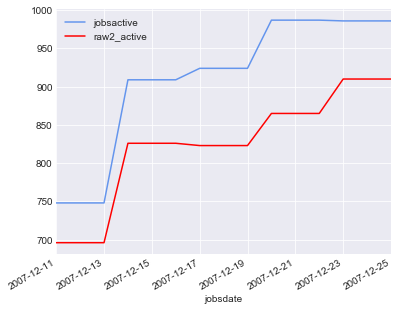
Unfortunately, in R2, where we DIDN’T have an actual delete date, the team inferred” that the delete date was the “last\_checked” date. This happened for some 20+million jobs, not insignificant. Not to mention that in doing this, we were creating a situation where EITHER the last\_checked was accurate, OR the deleted date was accurate, but both could not be. If the “last\_checked” is defined as the last time we scraped a job AND IT EXISTED, and if the “deleted” was the first time we scraped AND IT DID NOT EXIST, you could not have both with the same date as a job could not EXIST and NOT EXIST at the same time. In addition, a delete date, much like a created date, should be immutable. A job being deleted, should not be undeleted, as this is confusing, but also creates the reposting issue from the previous section, among other things like look ahead bias.

In addition, while R2 made a great enhancement in formalizing the scrape cycle to 00:00 UTC, and did not allow partial scrapes to be published in the current days activity (like Raw1 did where you could get some jobs on 1 day and some on the next in two different scrape cycles), there was an unintended consequence. Basically, jobs had a “created” date assigned based on the actual time of the jobs scrape, BUT the last\_checked would be assigned at the END of the company\_id level scrape. So this created a situation where point in time, you were seeing jobs created the “day before” you would have actually known they were created. This leads to analytics mismatch, where a company’s previous day active jobs, plus today’s created jobs, minus today’s deleted jobs, should equal today’s active jobs. This would NOT be the case in these instances, affecting the veracity of the analytics we created (or a client tried to create).

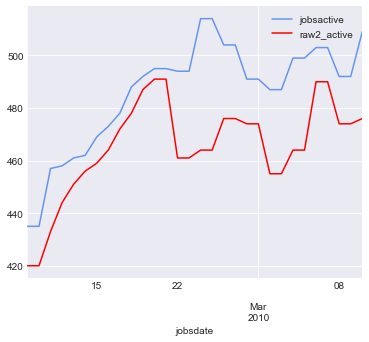
A random view of just the beginning and end of the R2 “core\_company\_analytics” file clearly shows the issues:



Notice, on the FIRST day that Target was scraped, 2007-12-11, there were 748 jobs created. We actually scraped them and saw they existed. Yet somehow 52 of those jobs are showing up deleted on the same day, which is not what really happened. What actually happened was we did not have a formal “delete” date, so we assigned the last\_checked date to the deleted date. Yet in reality those 52 jobs were active that day, and removed sometime between the scrape on 2007-12-11 and the next scrape on 2007-12-14. The jobs DID exist, and should have been counted. This not only understates the actual jobs that were active on 2007-12-11, but inaccurately inflates the number of jobs deleted, and is a clear 3 day look ahead bias. In addition, by doing this we do not follow the basic tenet that active jobs today = active jobs yesterday + created jobs today – deleted jobs today, or what I refer to as “analytic mismatch”. You can see the issue easily in the chart which shows the systemic understatement of the analytics found in R2:



If you look at the bottom of the file, the first analytics shown for company\_id 9999 show similar inaccuracies. If there were 347 active jobs on 2011-01-30, then how could you have “deleted” a number higher than that (352)on the following day? This suggests that 5 of the jobs deleted were NEVER active… You can see the issue again in the below chart for company\_id 9999. This is systemic across all the data.



Issues in R2 raw data and analytics summary:

* Blanket assignment of deleted dates equal to last\_checked for 20+million records. An impossibility, affecting veracity, and creating look ahead bias.
* Created dates equaling deleted dates, in general an impossibility affecting millions of jobs, and suggesting those jobs ***actually never existed*** for the analytics.
* Created dates that were point in time and accurate, mixed with last\_checked dates which were at the company\_id level, and therefore could be on different days, creating analytics mismatch among other things.
* Deleted dates that could be assigned, then subsequently nullified based on company reposts. Affecting point in time analytics, creating drift, affecting veracity, affecting job count truth.
* False and malformed country codes
* False and malformed state values
* Incomplete scrape log

We fix all of these and adjust all the dates to their relevant “production” dates – ***the date YOU would have known that the job was scraped***. With this we get accurate representation of PIT scrapes, as well as delete dates, as represented in the enhanced scrapelog and further in the job\_log. We do this for the created\_date as well. So, as an example, company XYZ is scraped from 11:00PM on 2020-01-01 to 1:00AM 2020-01-02. The created for a job here can be 2020-01-01, while the last\_checked can be 2020-01-02. This created mismatch errors which we adjust in RAW1, but here we need no adjustment, it is true. While the adjustments on edge are relatively insignificant, they do make the data “truer”.

## Descriptions Cleaned up!

The captured Description unstructured text can at times get a little messy. There are instances of “runaway” records where the text fields are giant, representing a clear error. While we try hard to deal with this in the normal scrape operations, some slip through. We deal with this in R2E by cutting them off, and limiting them to a reasonable size. We also clean up html noise and other things

## “Removed” Company Scrapes

Here we capture company\_id level scrapes that have been “removed” from production, and we deal appropriately with removing jobs for these from the analytics post the removal. In R1, these jobs could persist in the analytics because we had no “next scrape” to infer a delete date. You will see these in the scrapelog described above.

## Scraper “Changed”

We identify when we make a change to the scrape code, as an addition to the scrapelog that can be used to help verify a “strange” change in the analytics.

## Added the SOC/ONET code

In R2 we have mapped all our jobs to the [2010 Bureau of labor Statistics SOC CODE](https://www.bls.gov/soc/), and the enhanced [O\*Net](https://www.onetonline.org/) code. This replaces our limited proprietary category\_id. In R2E this is available in the **analyticsEnhancedJob** file described below. We expect to add the 2018 codes as well, and various levels of salary data connected to this code in the near future. The beauty is, when we add, it will just be a new field in the **analyticsEnhancedJob** file. In R2 SOC code was available in the ‘daily jobs’ file, thus was repetitive. Also, the R2 format was unscalable (i.e. if you wanted to add SOC Code for 2018 you bloat the daily file even for those that don’t care).

## Added the job URL

In R2 we have added the job level unique URL and in R2E it is available in the **analyticsEnhancedJob** file as well. This can be used as a verification tool among other things. In R2 this was also in the ‘daily jobs’ file, again more repetitive bloat.

## We have added additonal PIT identifiers

We have added Refinitiv identifiers in addition to Factset. We provide the ticker, sedol, cusip and isin and can provide RIC, TRBC, LEI and other identifiers to those licensed. We have greatly simplified usage of this to 1 file vs. the many in R2, and it is completely scalable, allowing us to add further sources without having to change or add files.

## We have an enhanced “Downloader”

We have a simplified downloader for R2E. All you need to do is enter the directory you want the data, enter your credentials, pick the files, and go. You can also adjust it to capture just the current days data.

# Analytics! The Fun Part!

With this new Gold Copy, job level, PIT raw data, we can do some great things with analytics. Let’s break down the files as we did for the raw data above. These files are also all in parquet, but are designed to be extremely flexible to consume. They also allow us to provide complete PIT looks of the analytics all the way back to 2007, whereas R1 only allowed for a “pseudo” PIT look back to 2016, and R2 had no part time analytics and only analytics in general back to November 2019. If you like data, buckle up!

## Current R2E vs. R2 analytics files and schemas. Fields in GREEN are added, in RED are deleted:

## R2E analytics files R1 analytics files

|  |  |
| --- | --- |
| **analyticsCore-** this file will contain the core analytics for any aggregate (company\_id, ticker, portfolio, GIC industry, Country, State etc.). Sweet and simple. Each aggreagate is the top level, and is given a “smart” code, which links to a table. You simply look at the table description, and download the “smart” code to get all the available aggregate data.   * jobsdate * active * created * removed * durationActive – the average length of time that the current active jobs have been posted for the aggregate * durationClosed – the average length of time that the “removed” jobs *were* posted | core\_company\_analytics\_yyyy-mm-dd.csv.gz  core\_ticker\_analytics\_yyyy-mm-dd.csv.gz  A summary of the fields:   * day * company\_id/stock\_ticker * company\_name * created\_job\_count * deleted\_job\_count * unique\_active\_job\_count * active\_duration |
| **analyticsEnhanced-** this file provides secondary enhanced analytics   * jobsdate * word\_parttime * word\_workfromhome * constituentsActive * constituentsActiveCount * constituentsAdded * constituentsAddedCount * constituentsRemoved * constituentsRemovedCount * indexEqualWeighted (divisor adjusted) * indexJobsWeighted (divisor adjusted) * durationModified1 * rank * rankPer(%) * Much more coming! | None |
| **analyticsFirst (coming soon)-** This file allows us to highlight “firsts” such as  “First part-time job” or “first SOC code” or “first Country”.  A client can quickly determine a company, ticker, portfolio, industry, any aggregate’s first day that they were in a City, State, Zip, Country, specific SOC code etc….   * jobsdate * firstType * value | None |

This format offers a flexible, scalable, and concise way to access ANY aggregates that we create. The footprint is a mere percentage of the current analytics files in R1. You access the aggregate (company, ticker etc.), and have access underneath it to all of the above files. You can select what you want. All the files are 100% PIT as we would have known on any given date. So no delta files. No PIT files. No archives. Just grab the file, and update based on the jobsdate “partition” that you need. In fact, we are considering offering portfolio level aggregate services to our customers. This is a massive improvement over the solid R1 analytics offering.

## Initial Aggregates

* MACRO
  + Company\_id
  + City
  + State
  + Zip
  + Country
  + Parttime (coming)
  + Fulltime (coming)
  + WorkFromHome (coming)
  + Ticker (RIC)
  + Ticker (Factset)
  + Sedol (Factset)
  + Naics sector
  + Naics subsector
  + Naics industry group
  + Naics industry
  + Naics sub industry
  + Trbc economic sector (requires Refinitiv License)
  + Trbc business sector (requires Refinitiv License)
  + Trbc industry group (requires Refinitiv License)
  + Trbc industry (requires Refinitiv License)
  + Soc major
  + Soc minor
  + Soc broad
  + Soc detailed
* USMacro1000
  + With all the breakdowns in Macro above

## Hash Level Analytics

To allow us to scale our hash level analytics, we have created the *analyticsEnhancedJob* file, for job level analytics.

|  |  |
| --- | --- |
| **analyticsEnhancedJob -** This file allows for us to add any level of information at the hash level. This file replaces the ‘hash’ file in R1   * jobid, * analyticType * value   Where the analyticType can be:   * url * soc\_2010 * soc\_2018 * word\_parttime * word\_workfromhome * soc2010 pit salary * soc2010 pit salary by region * soc2010 pit salary by industry * soc2018 pit salary * soc2018 pit salary by region * soc2018 pit salary by industry * soc2019 pit salary * soc2019 pit salary by region * soc2019 pit salary by industry * …..much more | This file replaces the ‘hash’ file in R1, which was GBs per day.  (some like soc, url are found duplicative in the R1 jobs file) |

# File Delivery

## Timing

R2 RAW files are typically available around 12:30AM UTC. The R2E RAW data and reference files will be available roughly 45 minutes after the original R2 RAW files are posted.

The R2 company and ticker analytics are typically available around **7 hours** after the R2 RAW files, or 7:30AM UTC. The R2E Analytics are available within roughly **2 hours**, or 2:30AM UTC.

## AWS S3

All R2E files are available on AWS S3, are AWS GLUE ready, and can be either downloaded locally or synced with an existing AWS bucket. The file topology is quite simple with the following “directories”, where extraction is merely updating the directory. You also have the option of extracting JUST certain parquet “partitions”:

* jobs\_base
* jobs\_log
* descriptions
* reference
* scrapelog
* auxiliary
* analyticsCore
* analyticsEnhancedJob

The current RAW2 requires you to go to JDE.  Then decide whether what you are looking for is in Feeds, or MarketReports (onboarding needed).  Once you learn that you would go to MarketReports, you then need to decide whether to go to Reports or PostProcess (more onboarding).  If you go to PostProcess, you see this:

# 

If you click into any of these, you get something that looks like this:

# 

If you back out of this because you are completely lost and no amount of onboarding can help, you instead go to the reports directory, you get this:

# 

Where if you look closely you see the Core\_Company\_Analytics and Core\_Ticker\_Analytics I begged for and ultimately got!  But you also have the mess below those, which if you click into you get this:

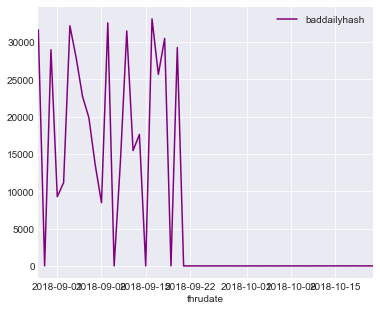
# 

**Not very user friendly…**

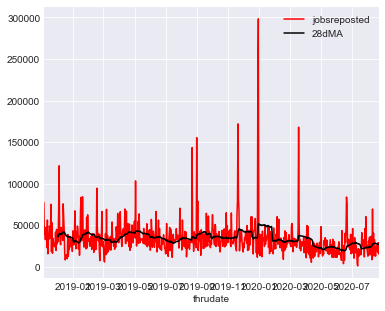
# Appendix 1: RAW2 RAW data supplementary charts

## BAD Daily hash – where company\_id or dates are malformed

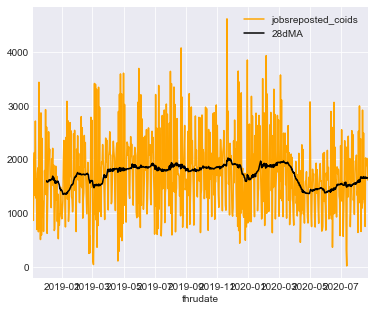
There was a period in 2018 when there were daily files with sizeable “bad” rows.



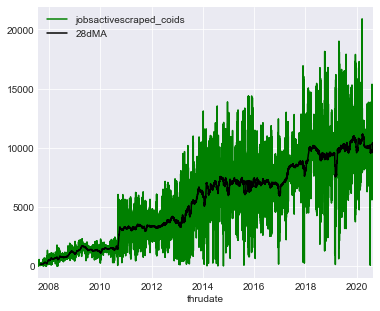
## Reposts – Deletes occur when a company is scraped, and a job that was active the previous scrape is not there. A repost happens when that same job is “reposted”, meaning in a subsequent scrape of the same company, it appears again. Reposts really kick in in the dataset in 2018.



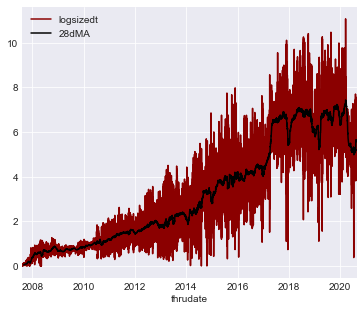
And the # of company\_ids with reposts:



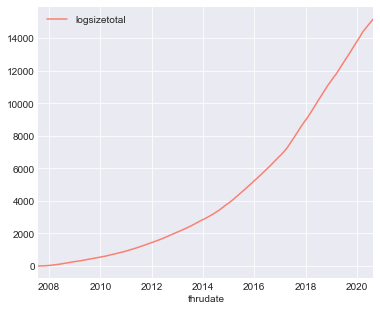
## Company\_ids scraped that had active jobs



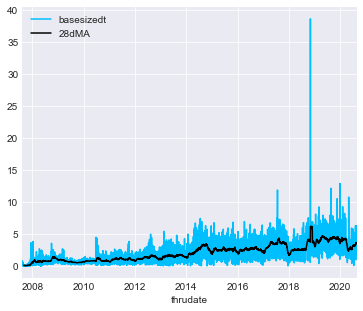
## Daily jobs\_log file size in MB



## Total jobs\_log file size in GB



## Daily jobs\_base file size in MB



## Bad “New” created dates. Multiple created dates for same hash

