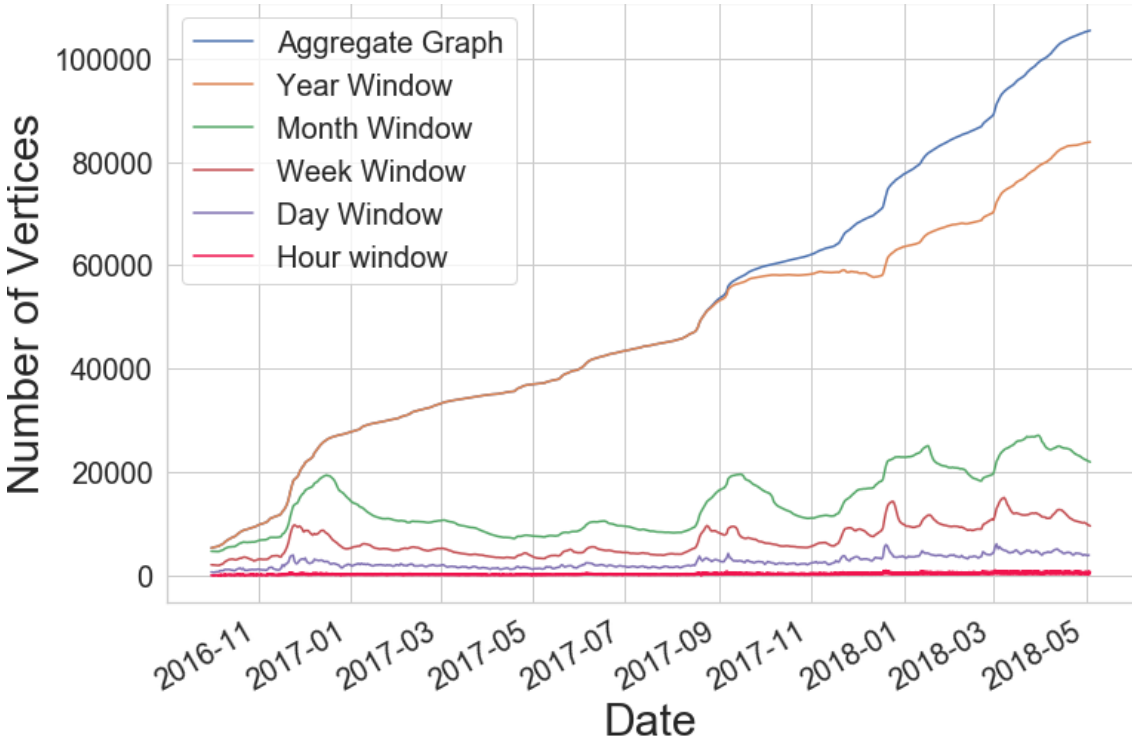
Basic network statistics

Firstly we look at the number of nodes, number of edges and average degree for each window size for the whole period. Note that "number of nodes" means number of individuals who were involved in at least one interaction within the relevant period.

Number of nodes

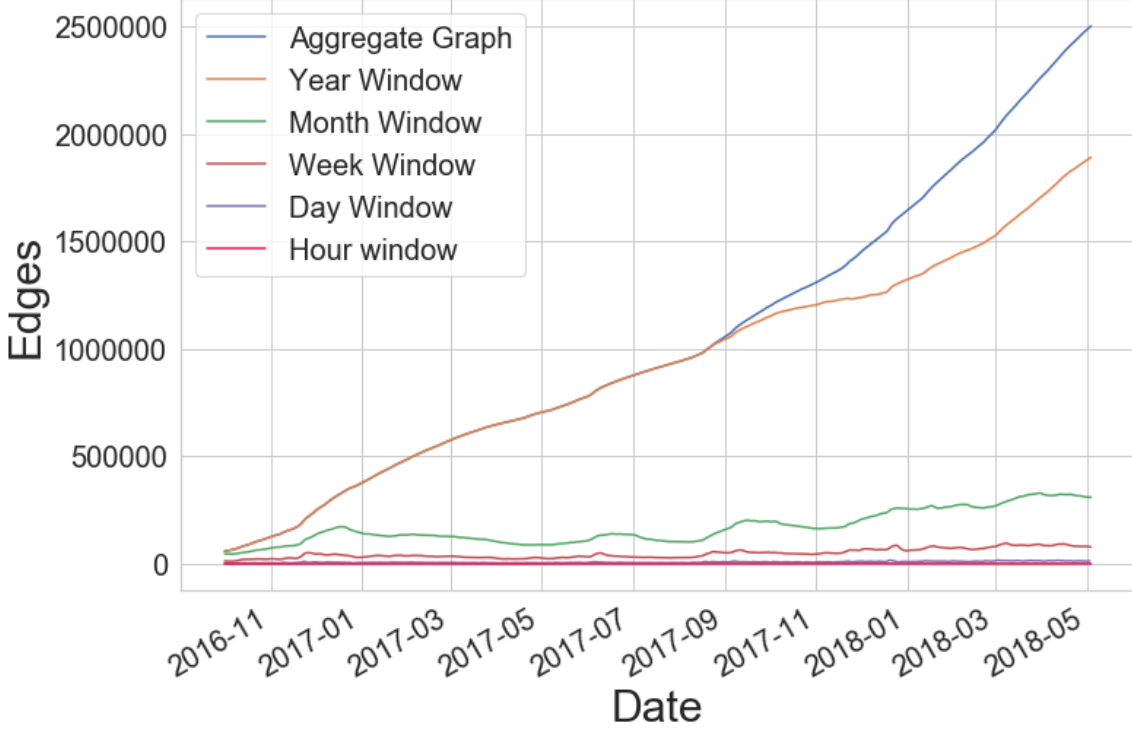
First we examine how the number of nodes is reported according to each window size. We see that:

- The values for the year and the month window diverge by huge scales. This suggests that there is an important distinction to be made about reported "total" number of users vs "active" number of users for social networks.
- On medium scales of a month and week, we see 'bumps' following news events of interest to the Gab community such as the Trump election in Nov 16 and Charlottesville "Unite the Right" rally in Aug 17, which seem to be obscured in larger timescales and just noise in smaller scales. Perhaps weekly/monthly windows are a good candidate size for time series anomaly detection tasks?



Number of edges

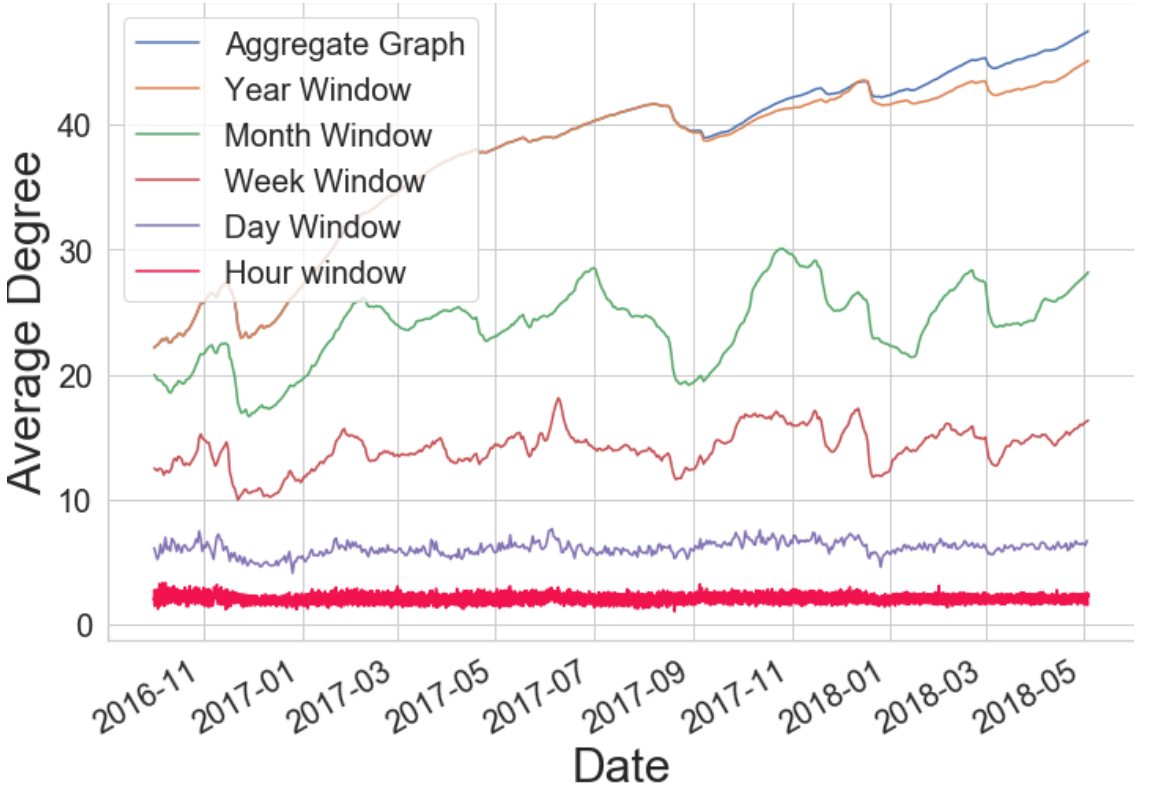
Not much to say here apart from, again, the hugely differing scales of looking at a month vs a year.



Average degree

Calculated as 2x|edges|/|vertices|. We see that:

- For medium sized windows this is largely varying, with the 'dips' being caused by the bursts in number of vertices.
- For daily and hourly windows, it seems to be just constant + random noise (or constant+seasonal, I haven't looked into it yet.) It just seems interesting to me that there's seemingly no trend for the hour one compared to other sizes.



Connected Components Analysis

This section contains the analysis of the size/proportion of the largest connected components, as well as the number of connected components. In the proportion and number, we exclude components comprising just one edge (two nodes) from the calculations.

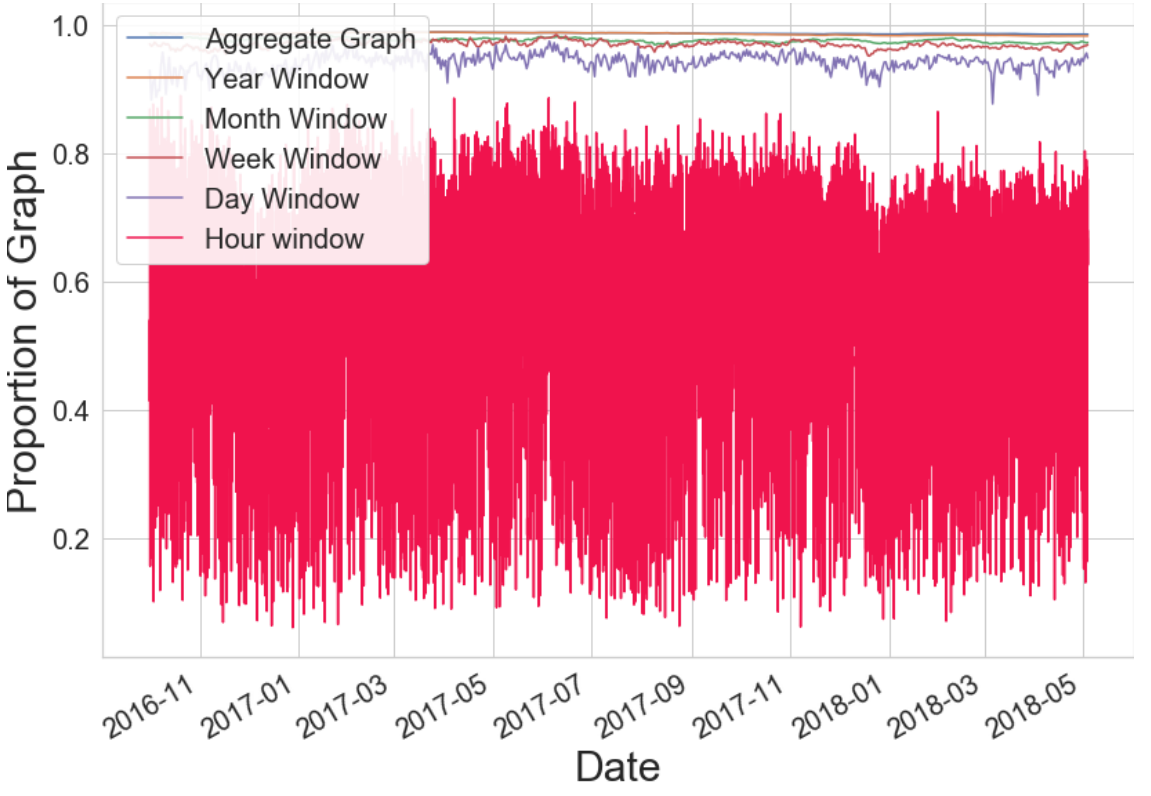
Proportion (without isolated nodes)

We measure the size of the largest connected component as a proportion of the size of the whole graph for that window.

Whole time period

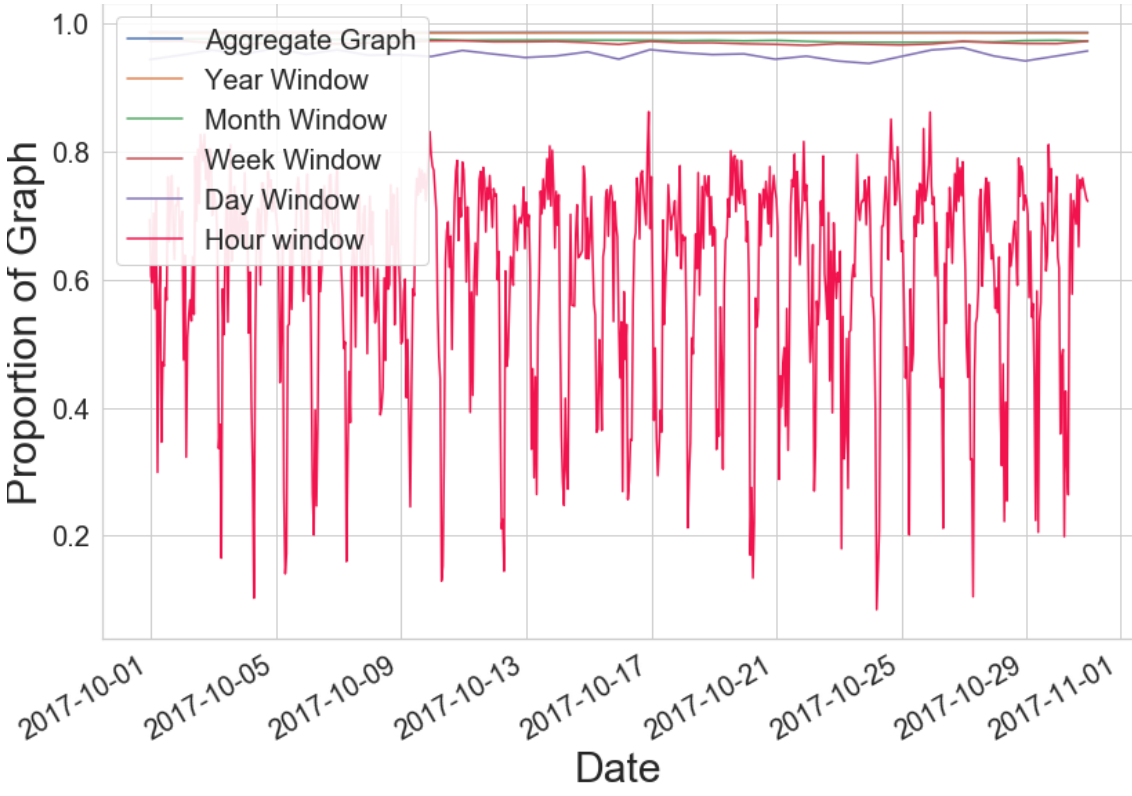
We find that:

- Just a day-length window is large enough to observe a connected component that is always more than 90% of the graph.
- Hourly window is a total mess, look into this later.



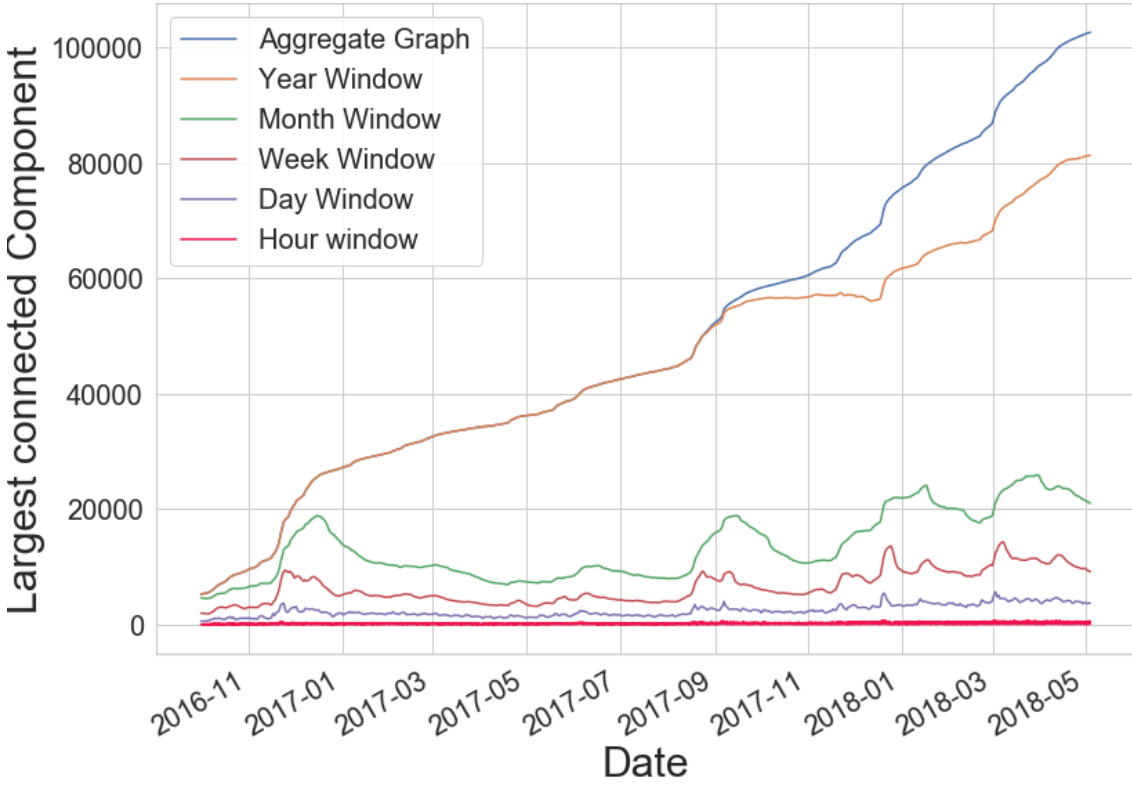
Zoomed in

When zooming in to just a month subset of data, we see that the hourly window size is actually showing diurnal behaviour which we will explore later.



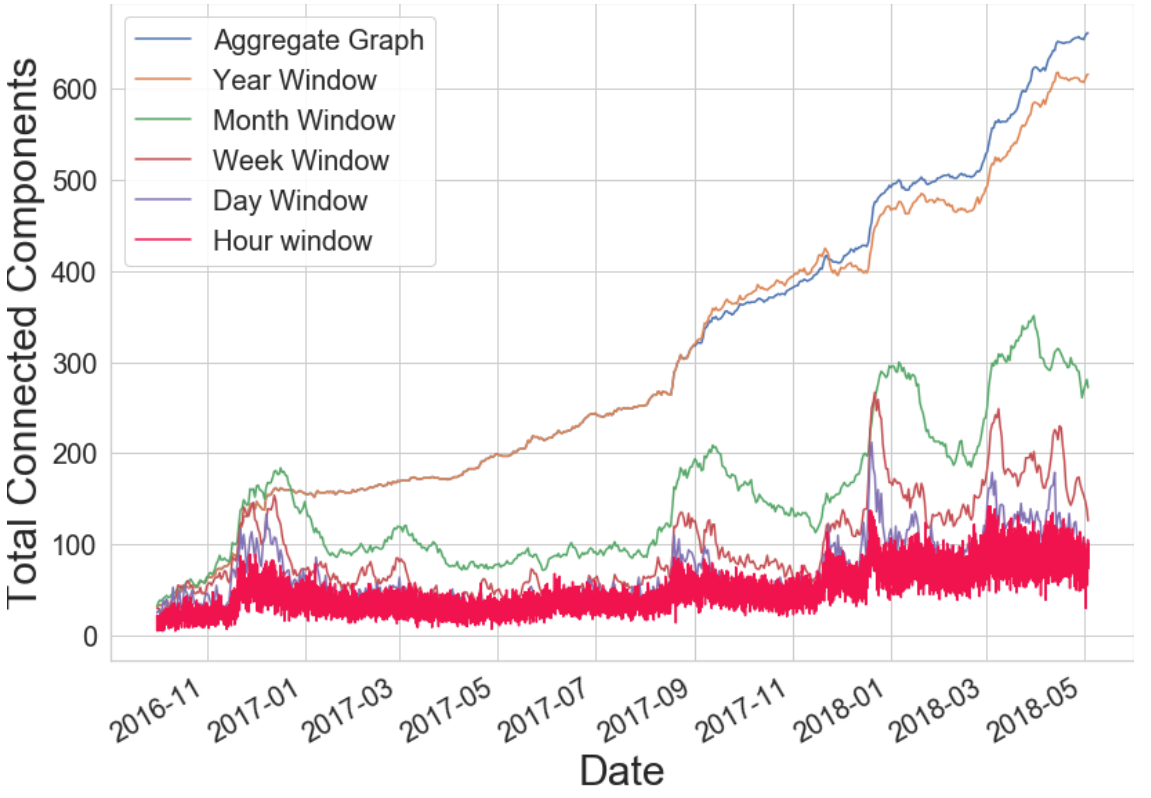
Size of the largest connected component

On the whole, this shows similar behaviour to just the number of vertices.



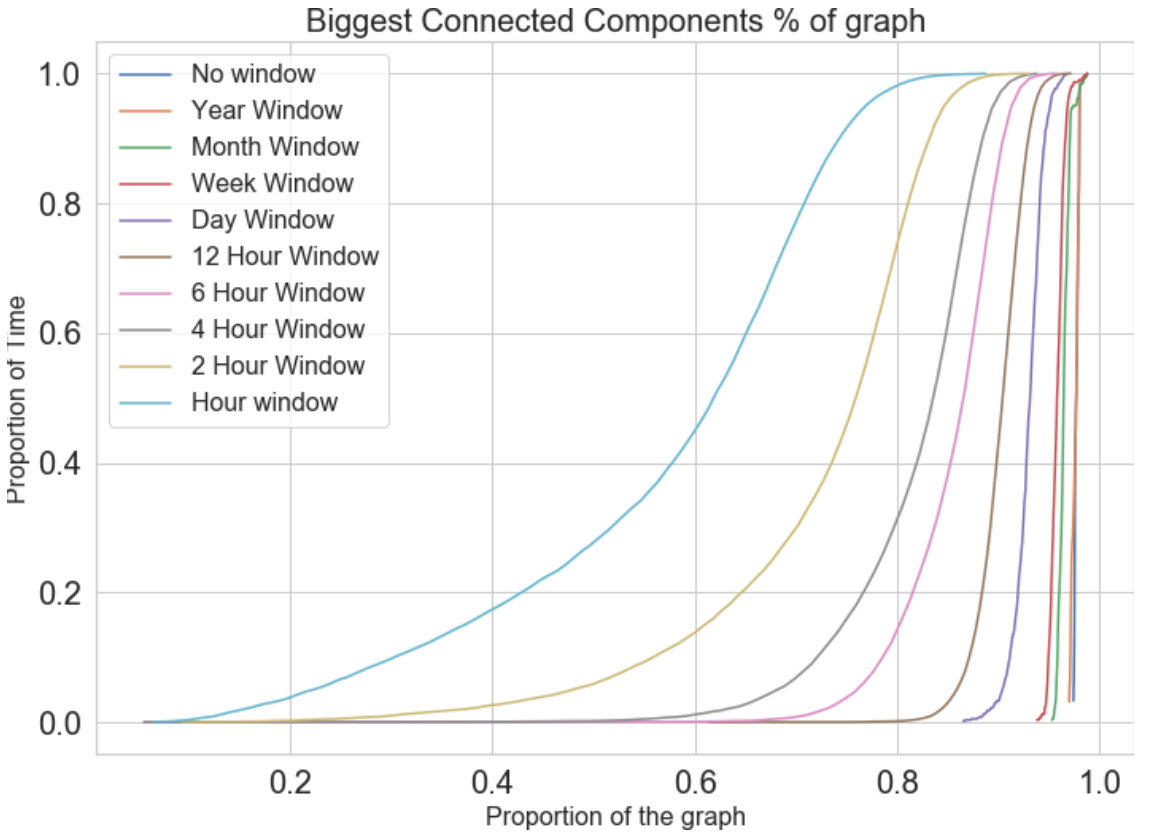
Number of connected components

This seems to be similar in trend to the size of the largest (note the much smaller scale of ~600 components in total).



At what size does the giant component break?

The plot below shows a CDF of the 'proportion' data for different window sizes, with particular attention to sizes between an hour and a day. Would be maybe helpful to provide a takeaway statistic like "for all window sizes, there's a component of size x% for y% of the time.

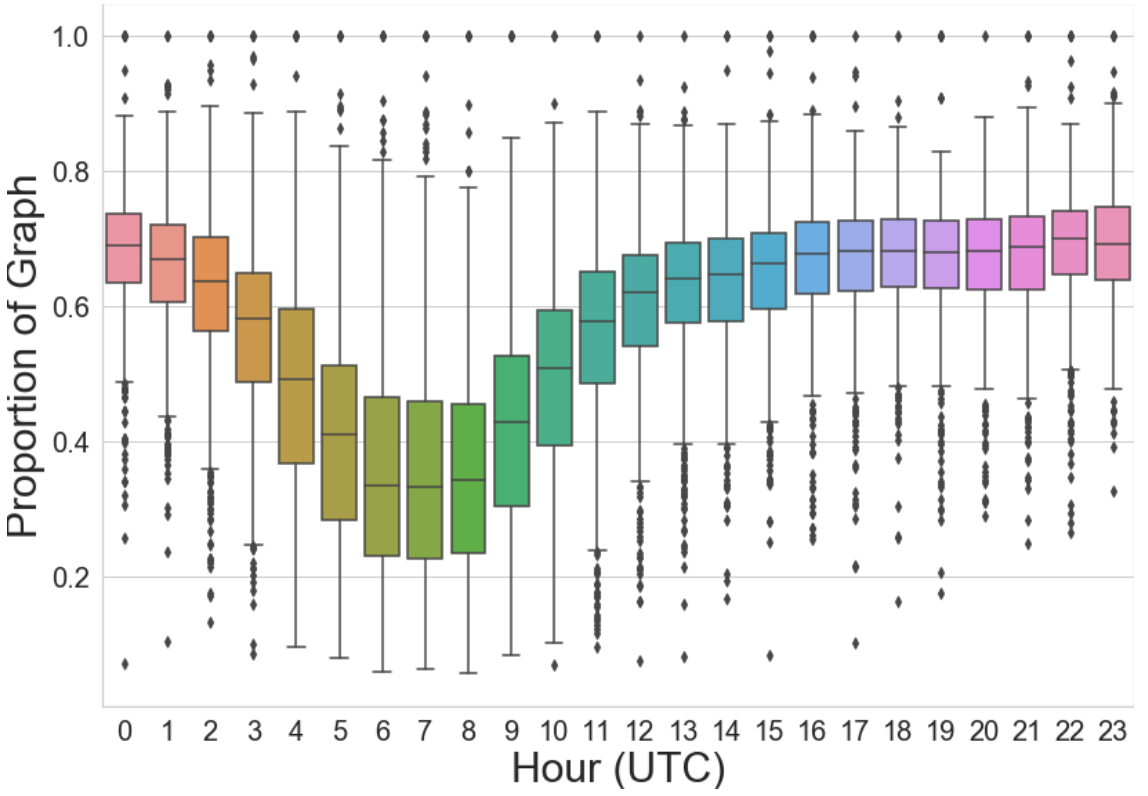


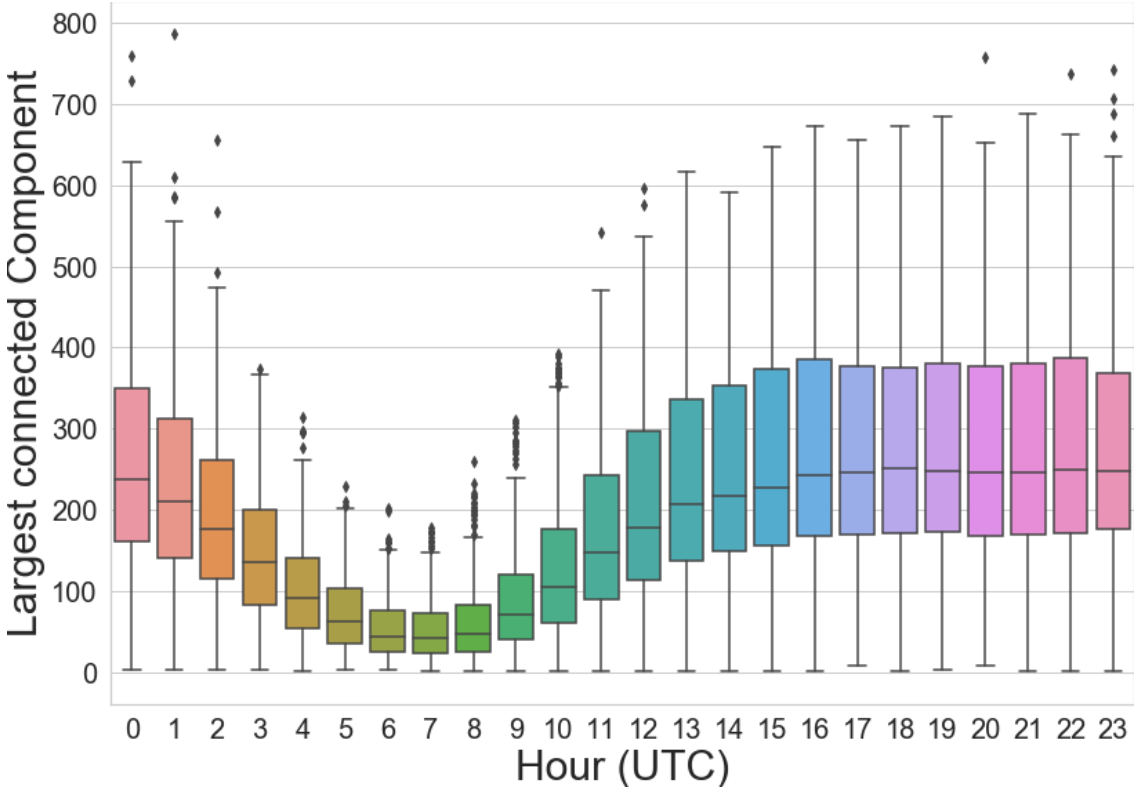
Diurnal activity as shown by the hourly window

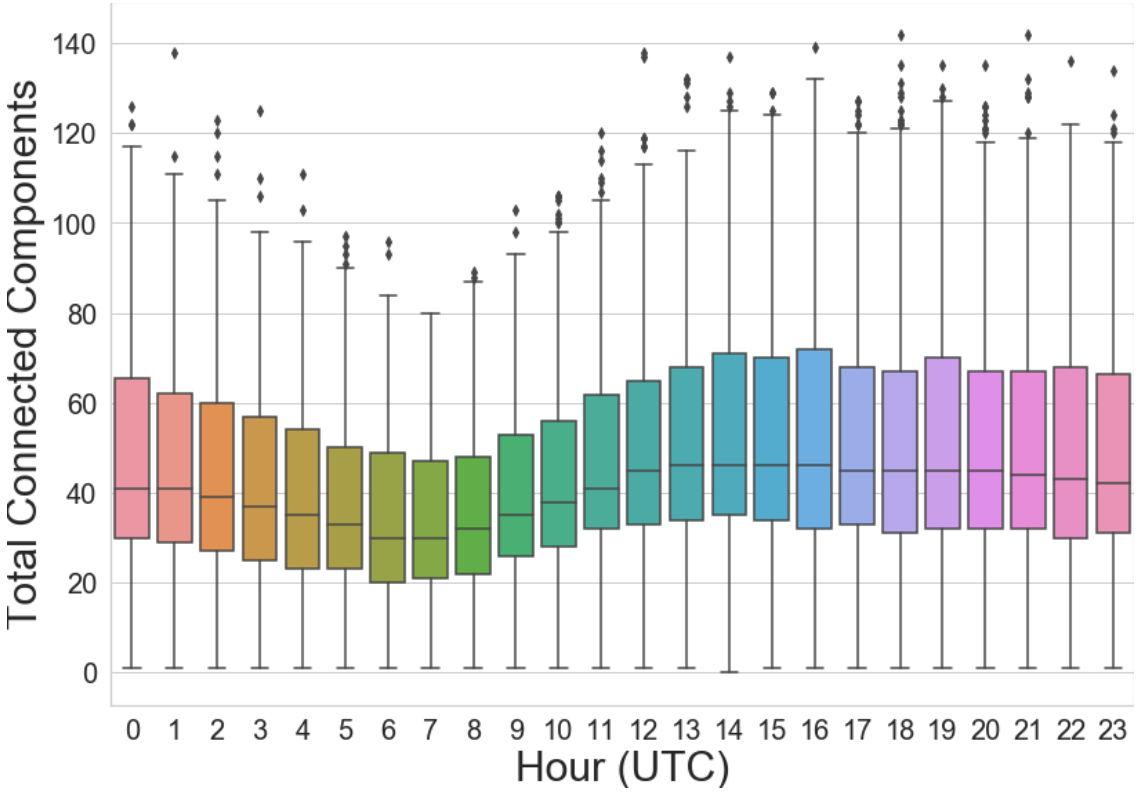
Batching the data by hour of the day we can see some diurnal behaviour as it's a mostly US-based platform.

Initial box plot

There are definitely some issues with the outliers in the Seaborn boxplotting function (it is a bit black-box and I'm a bit worried how it is deciding that something is an outlier) so happy to let this go, THE BEAUTIFUL COLOURS THOUGH!!


Proportion of Graph
1.0
0.8
0.6
0.4
0.2
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour (UTC)

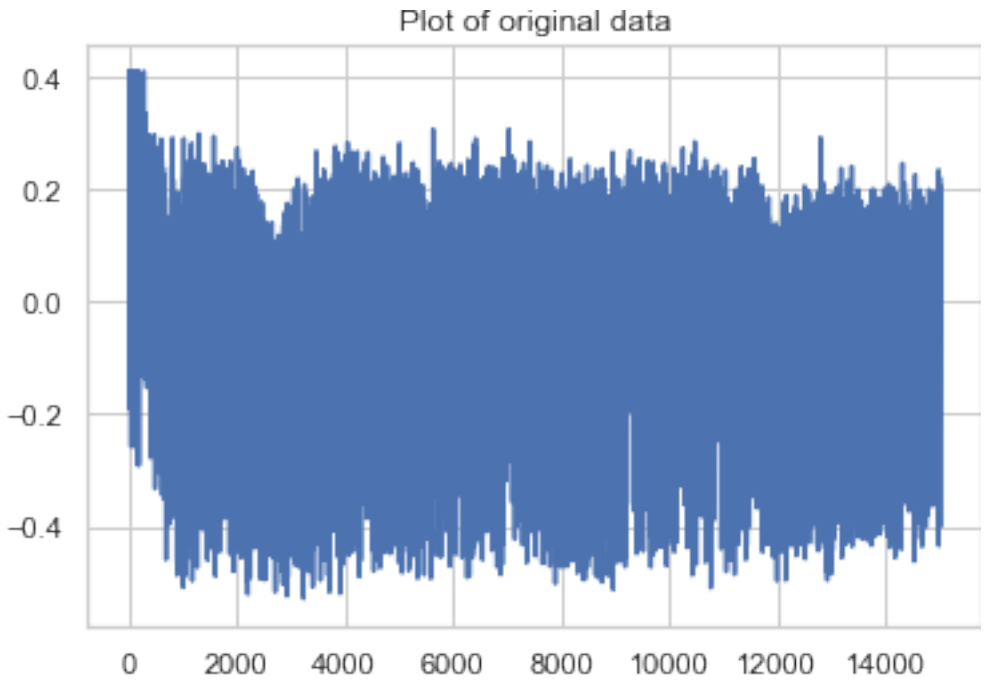

Largest connected Component
800
700
600
500
400
300
200
100
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour (UTC)

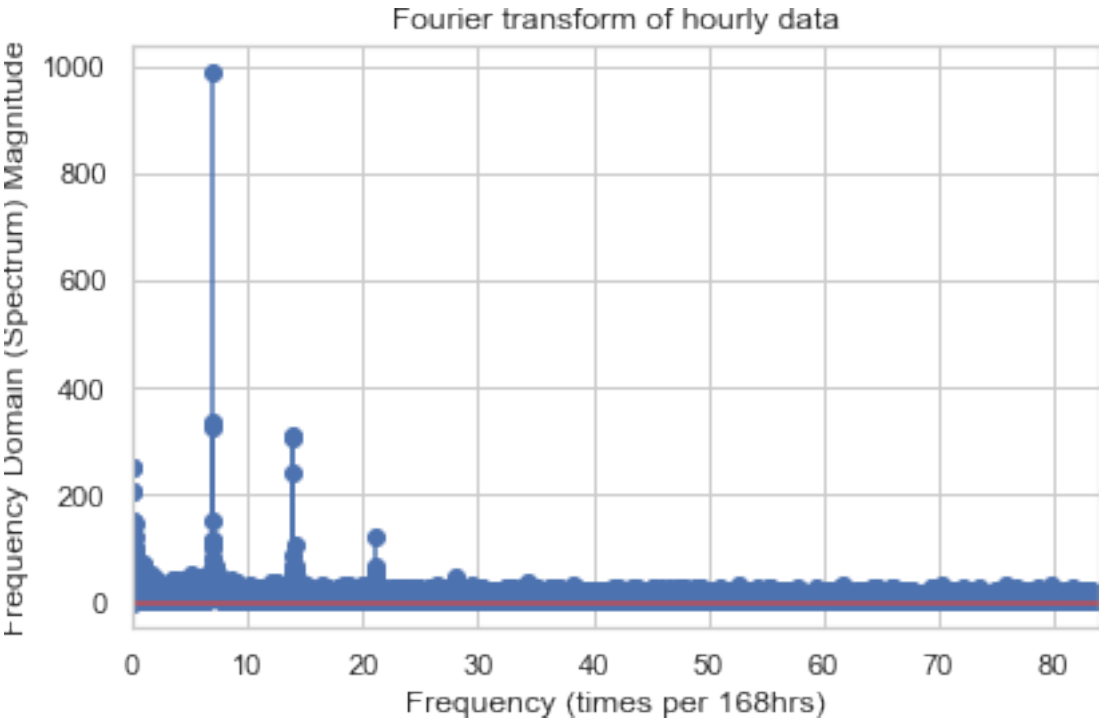


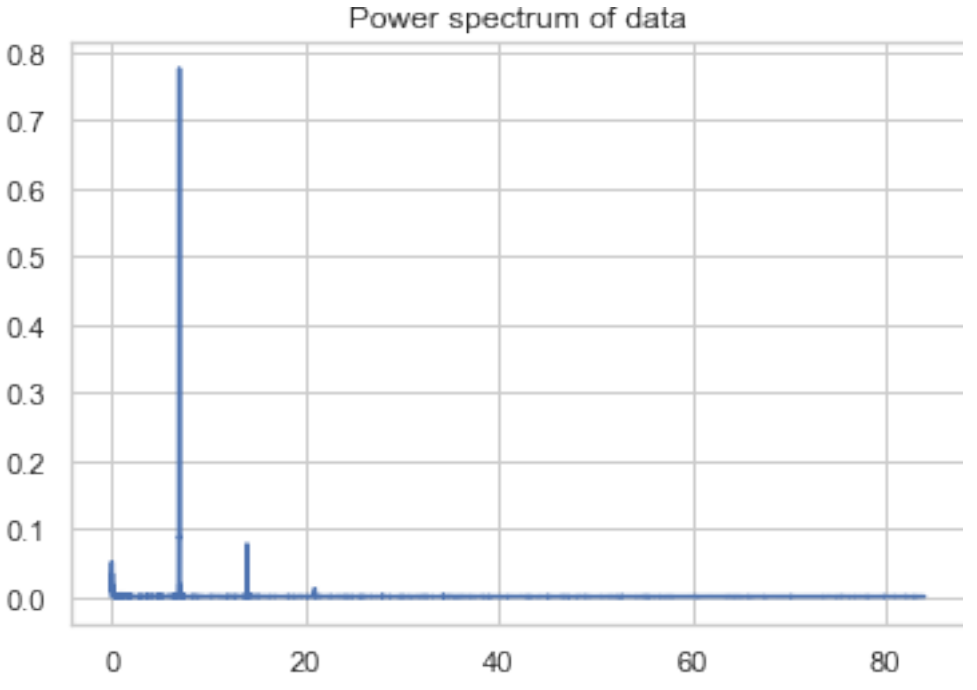
Time-series analysis of the LCC in the hour window

This section explores deeper the existence of periodic/diurnal behaviour in the size/proportion of giant component size using fourier analysis. We will look first at the proportion data.

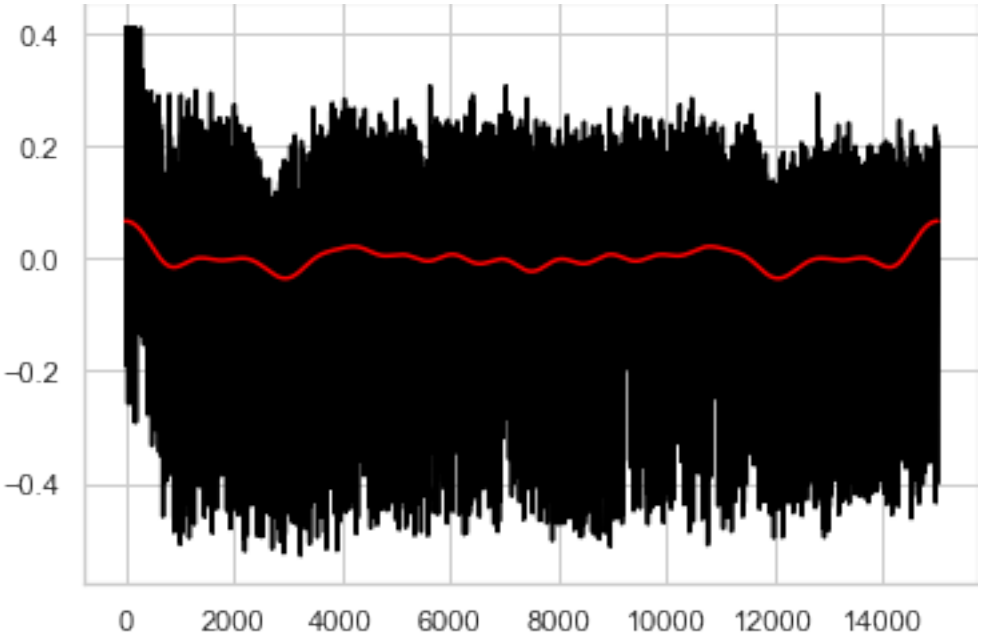
The fourier transform and power spectrum of the shows peaks at the 24hr, 12hr, and 6hr frequencies.


Plot of original data
0.4
0.2
0.0
−0.2
−0.4
0
2000
4000
6000
8000
10000
12000
14000

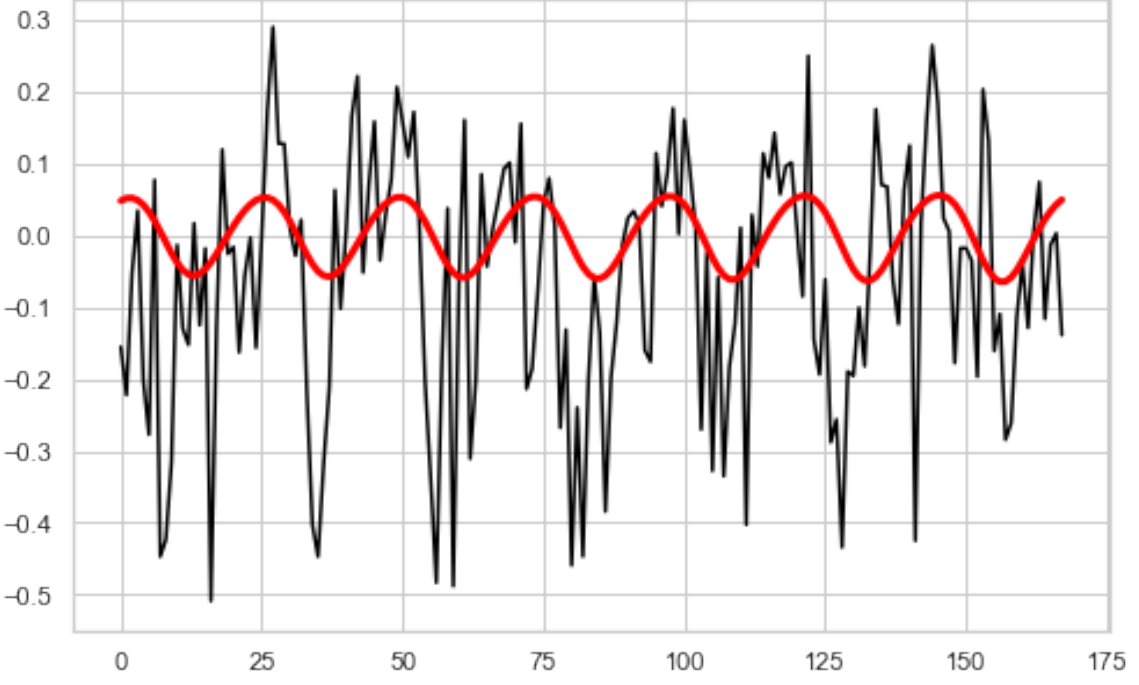

Fourier transform of hourly data
Frequency Domain (Spectrum) Magnitude
1000
800
600
400
200
0
0
10
20
30
40
50
60
70
80
Frequency (times per 168hrs)


Power spectrum of data
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
20
40
60
80

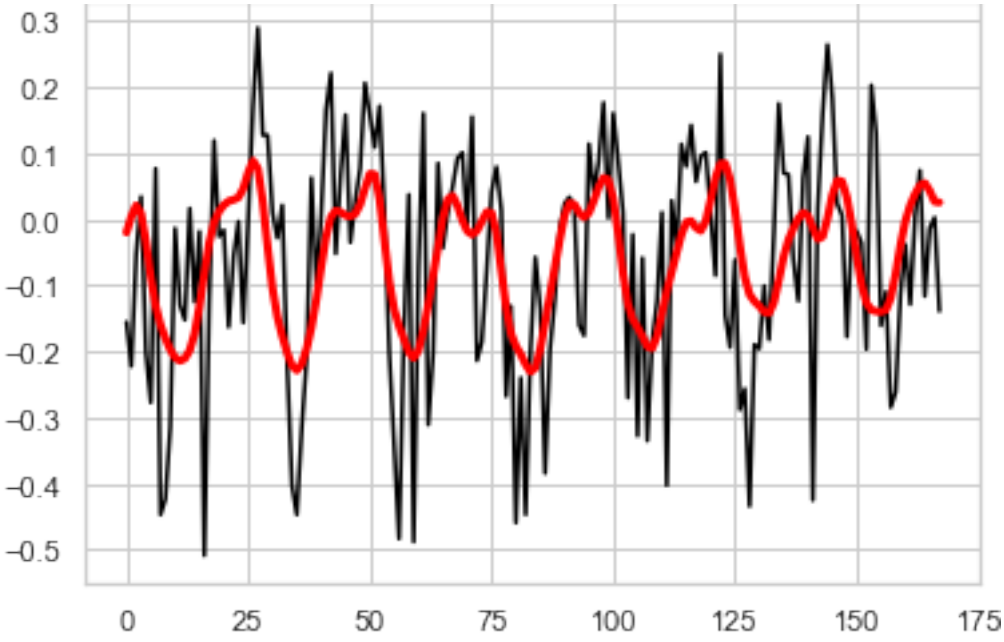
Below is the inverse Fourier transform (IFT) of the first few frequencies, showing the 'hum'.



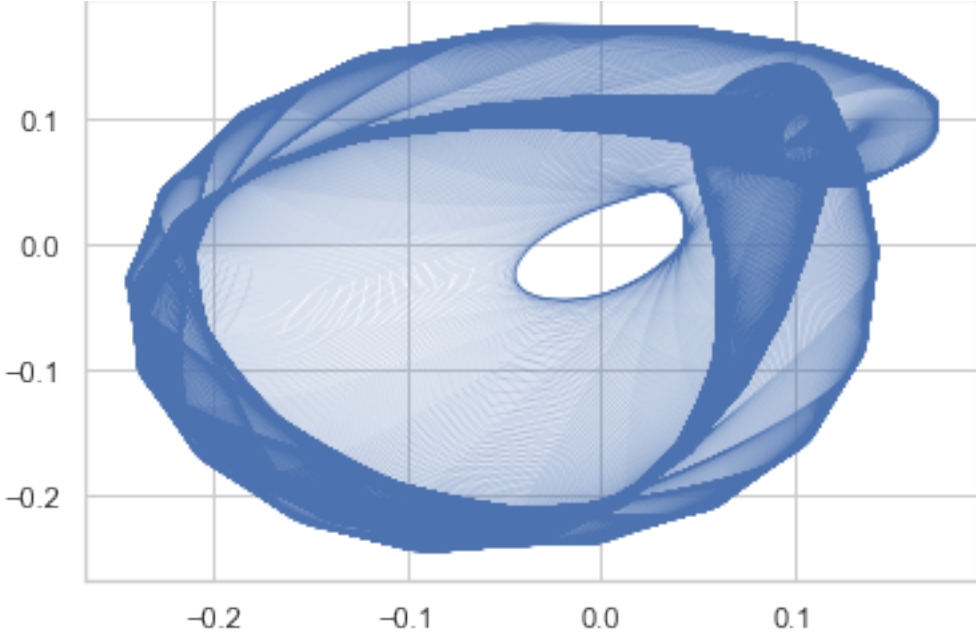
The IFT of the largest magnitude 10 frequencies looks fairly as expected, looking like a sine wave with 24h period.



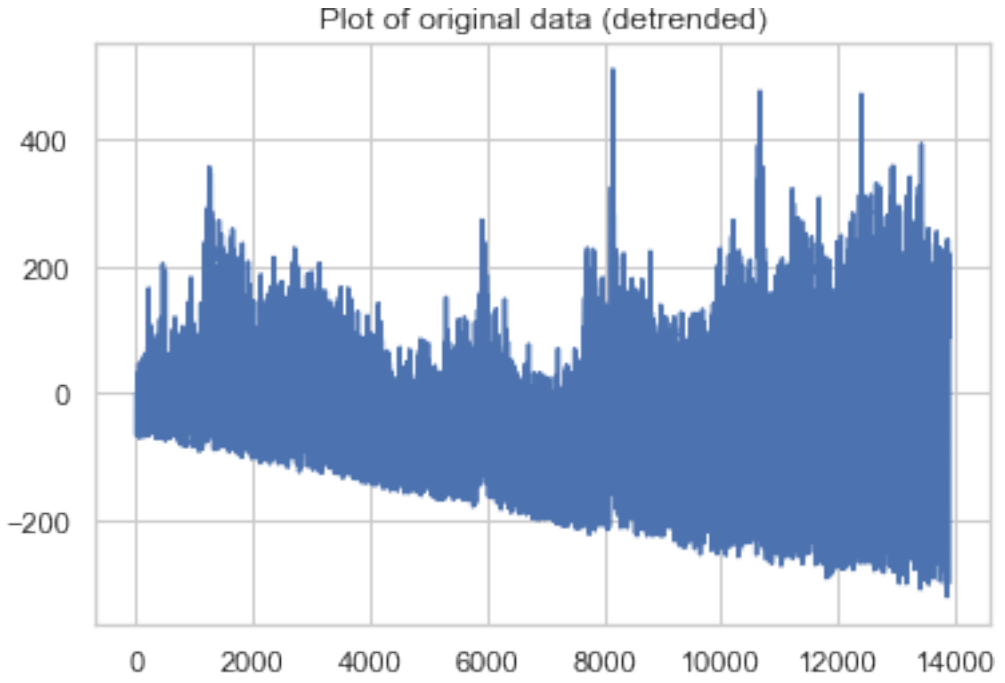
If we allow the IFT of the largest 240 harmonics we actually see a second slightly smaller peak slightly before a bigger peak, suggesting the presence of a European userbase.

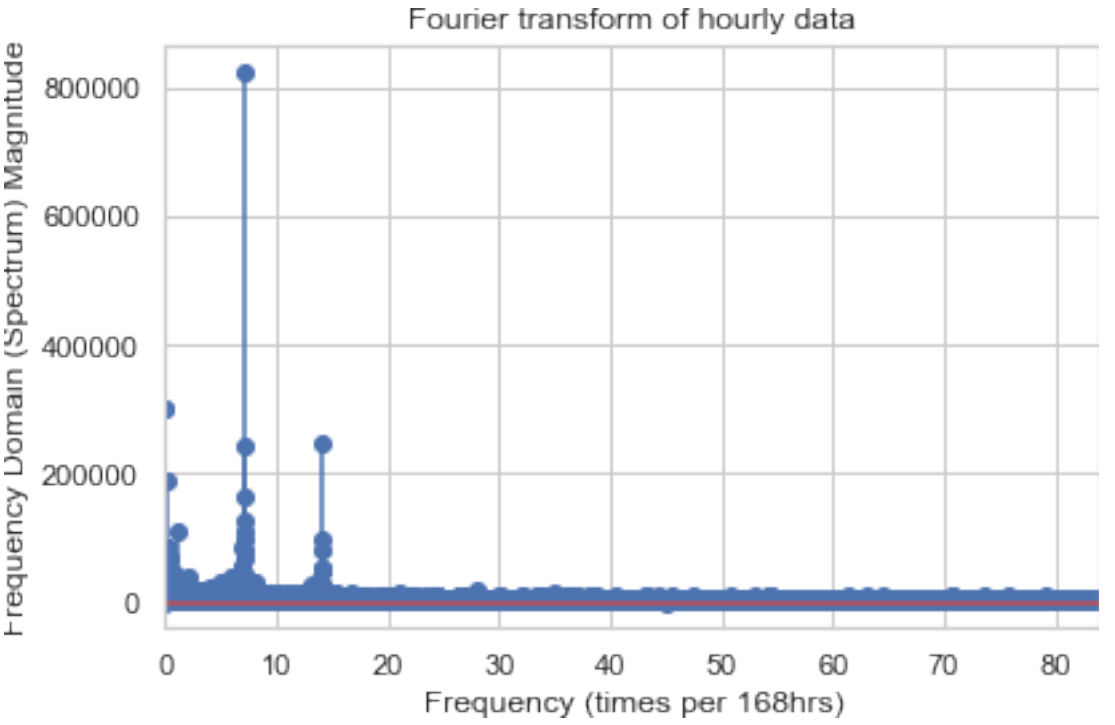


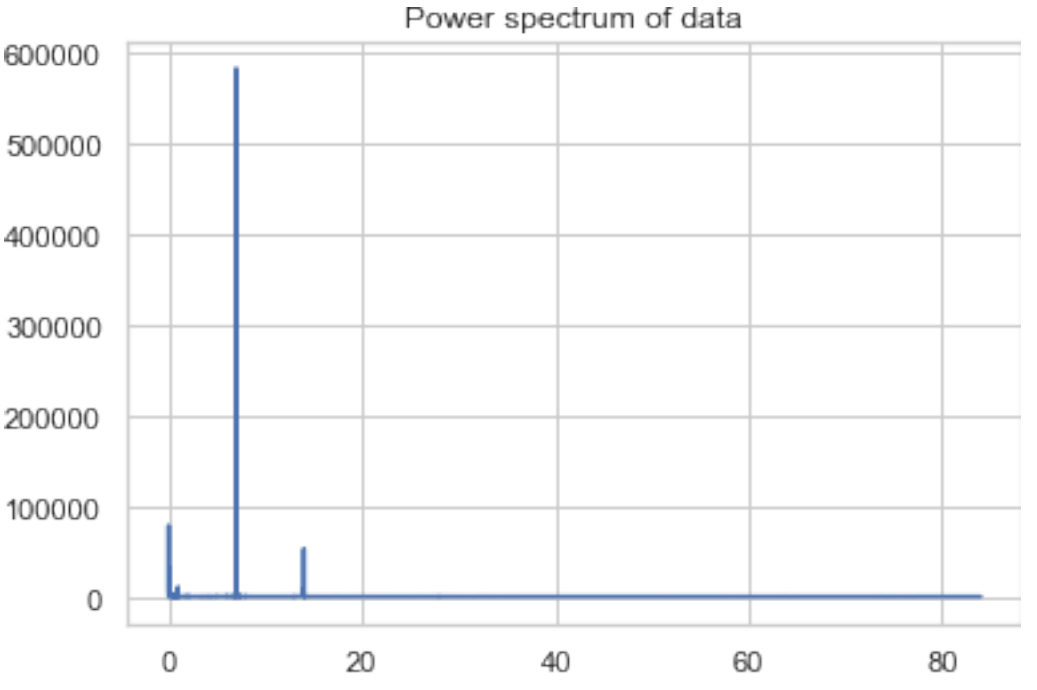
The following plot is the red line in the previous plot plotted against a 4hr behind version of itself -- it doesn't really add anything but I just want to keep the picture for it somewhere as it's quite pretty!

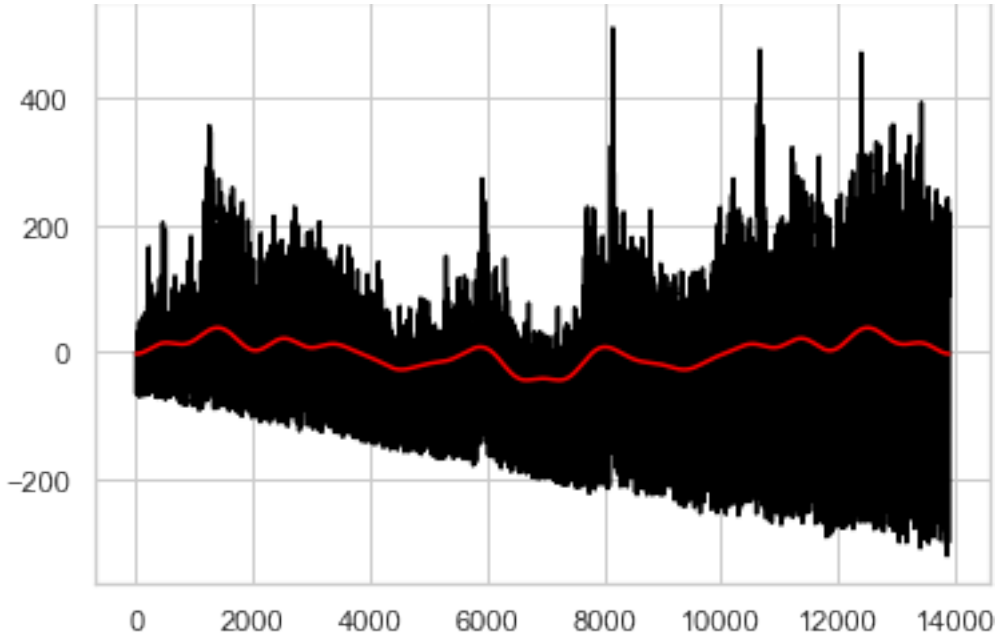


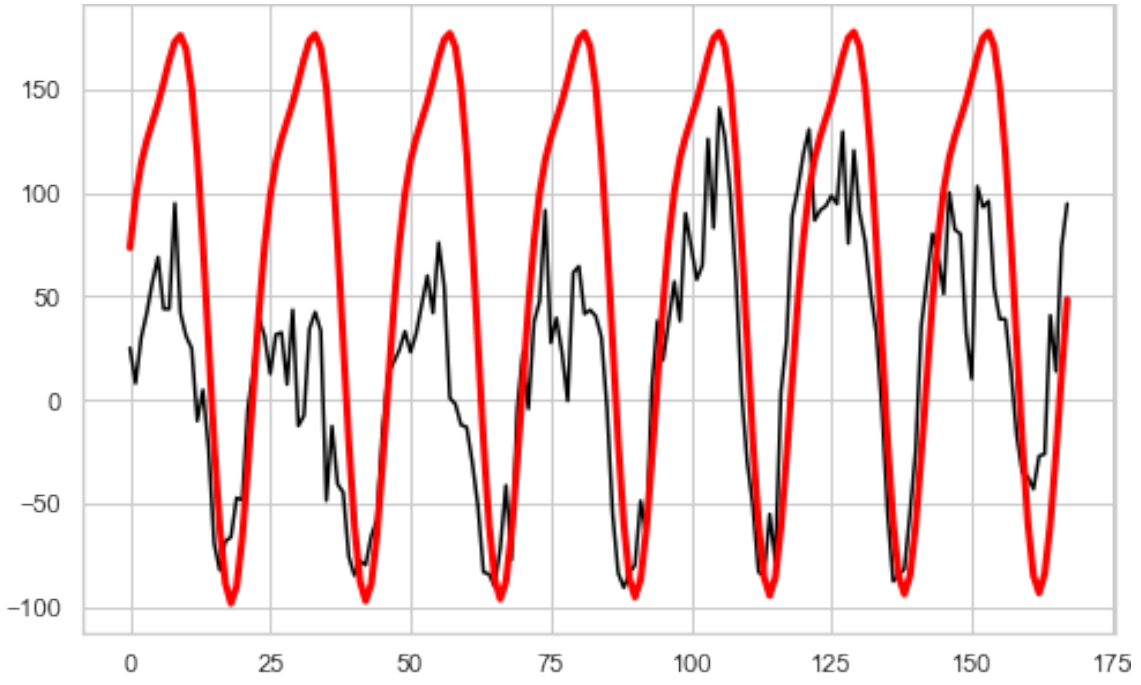
This following subsection definitely needs more attention from me but is essentially doing the same process but for the absolute size of the LCC, and shows that if we look at the absolute size rather than proportion, we see not just a 24hr frequency but a weekly frequency (component is smaller at weekends)/

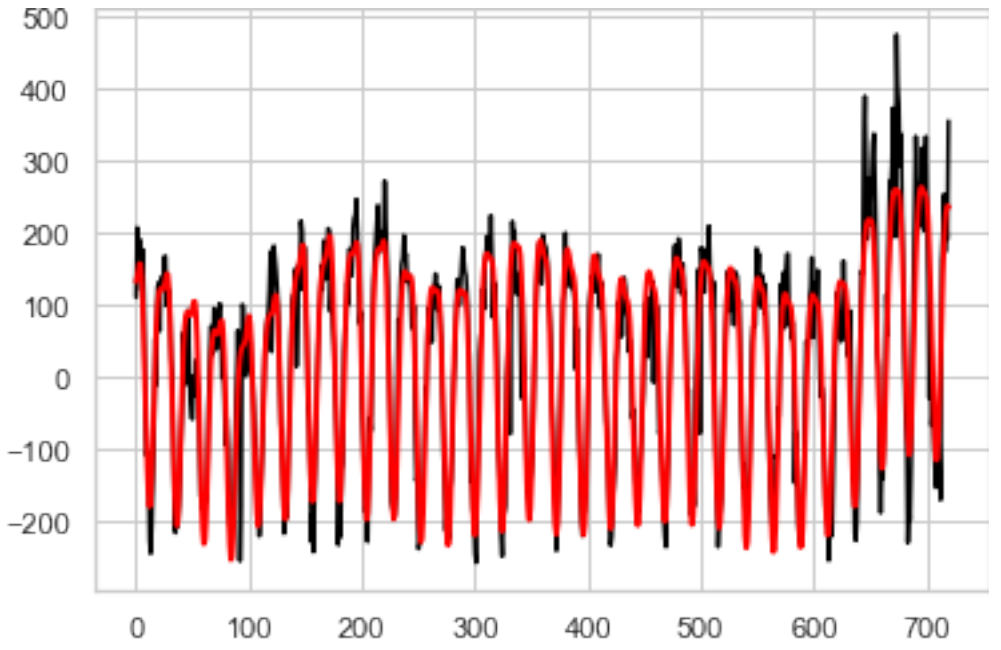

Plot of original data (detrended)
400
200
0
−200
0
2000
4000
6000
8000
10000
12000
14000

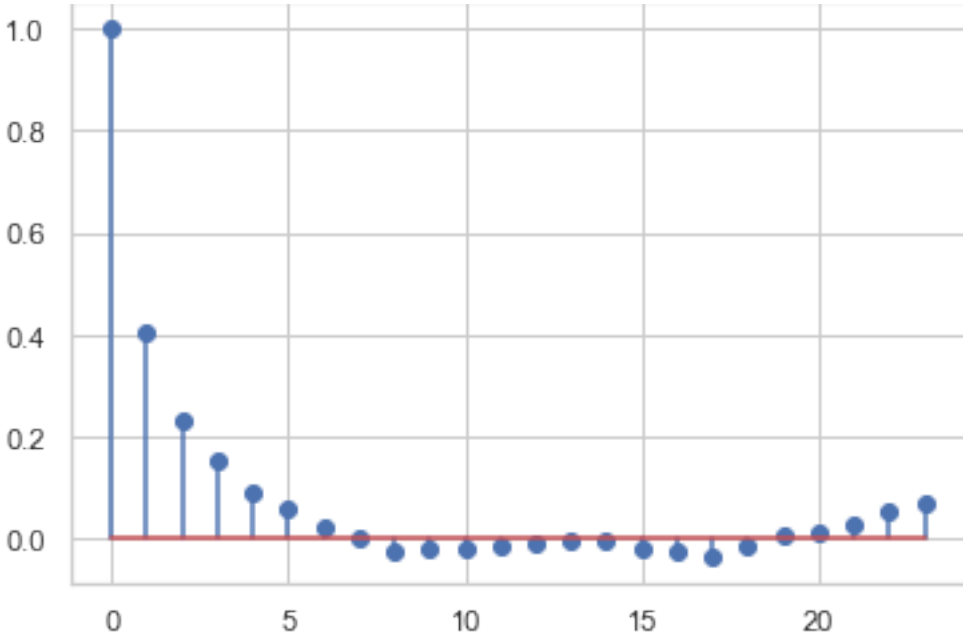

Fourier transform of hourly data
Frequency Domain (Spectrum) Magnitude
800000
600000
400000
200000
0
0
10
20
30
40
50
60
70
80
Frequency (times per 168hrs)

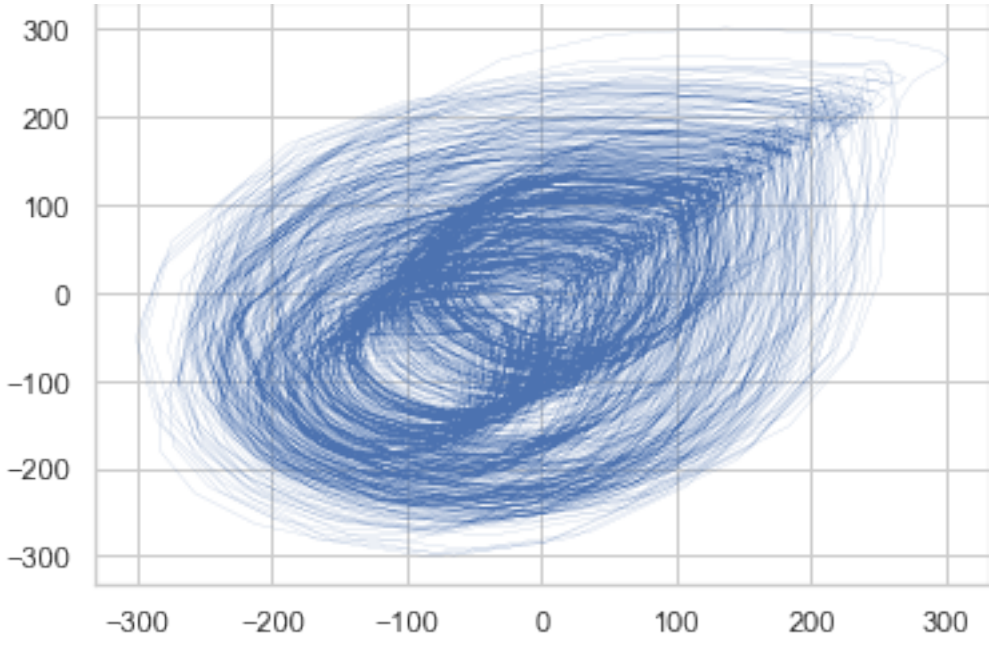

Power spectrum of data
600000
500000
400000
300000
200000
100000
0
0
20
40
60
80


400
200
0
−200
0
2000
4000
6000
8000
10000
12000
14000


150
100
50
0
−50
−100
0
25
50
75
100
125
150
175


500
400
300
200
100
0
−100
−200
0
100
200
300
400
500
600
700


1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20


300
200
100
0
−100
−200
−300
−300
−200
−100
0
100
200
300

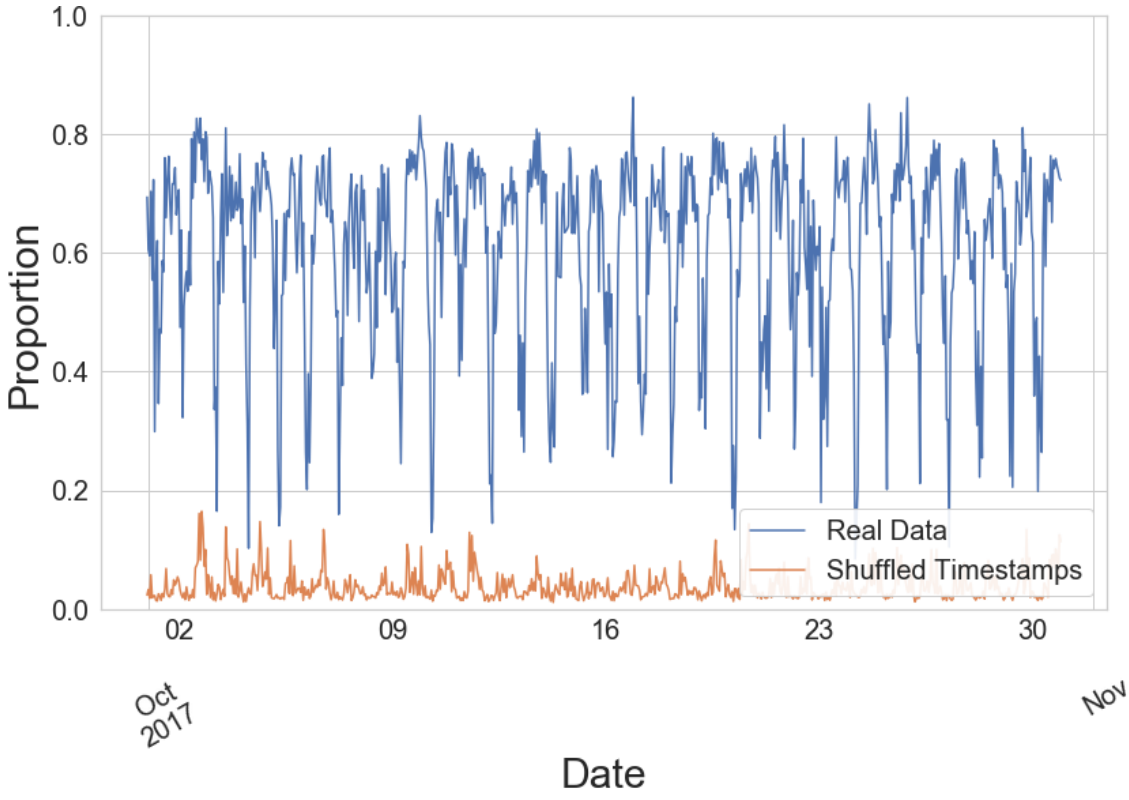
Comparison with shuffled timestamps null model

Keep the same links but reorder their timestamps randomly, so that the rate of edge activity is conserved and that the aggregate graph is identical. For more detail on this null model, take a look at Temporal Networks (https://arxiv.org/pdf/1108.1780.pdf), P. Holme, J. Saramaki (2011) under the heading Randomly Permuted Times (p17).

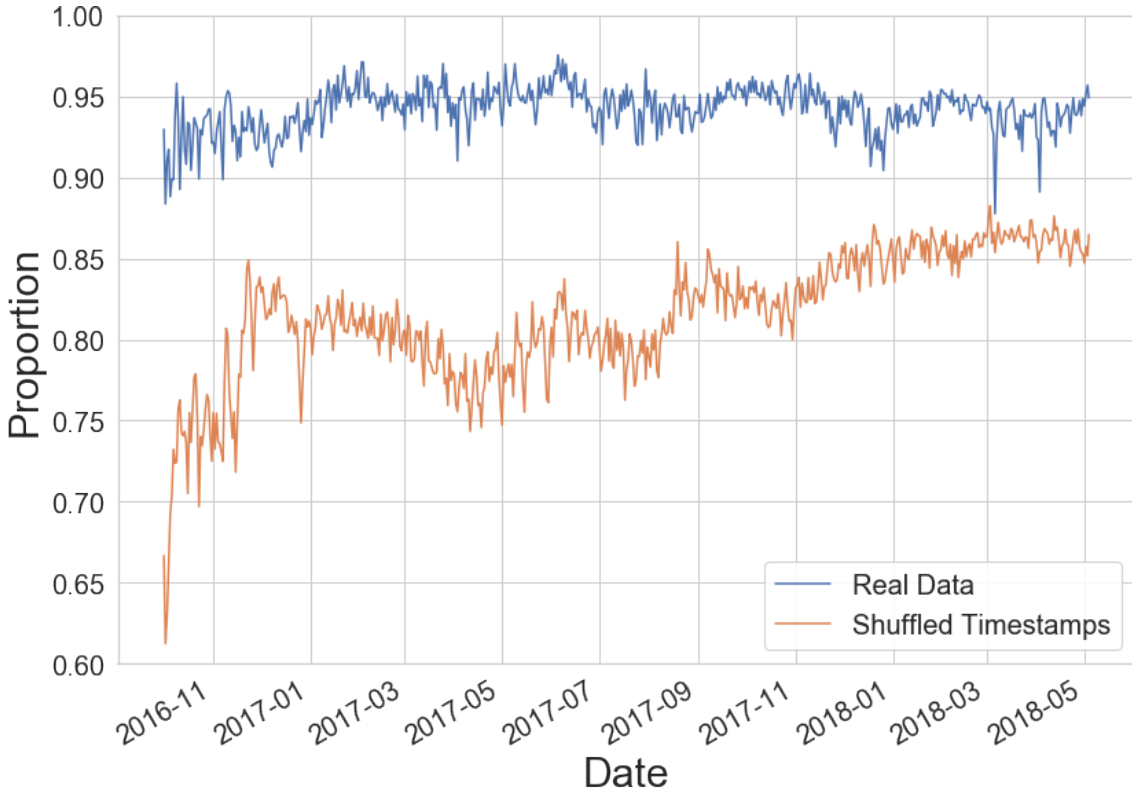
Proportion

We find that the value for the proportion is always smaller in the shuffled timestamps case than the real data, only slightly so for window sizes greater than a day, but largely so for the hour window. My thoughts are that this is due to the 'memory effect'/'edge persistence' in the real data, i.e. pairwise interactions are fairly bursty, and the chance of an interaction between two users decreases the longer it's been since the last interaction. In this way, we might expect to see a higher number of unique nodes by randomly sampling a number of edges throughout the whole time period than sampling the same number of edges but within a small time slice (i.e. a larger denominator in the 'proportion' for the shuffled than for unshuffled).

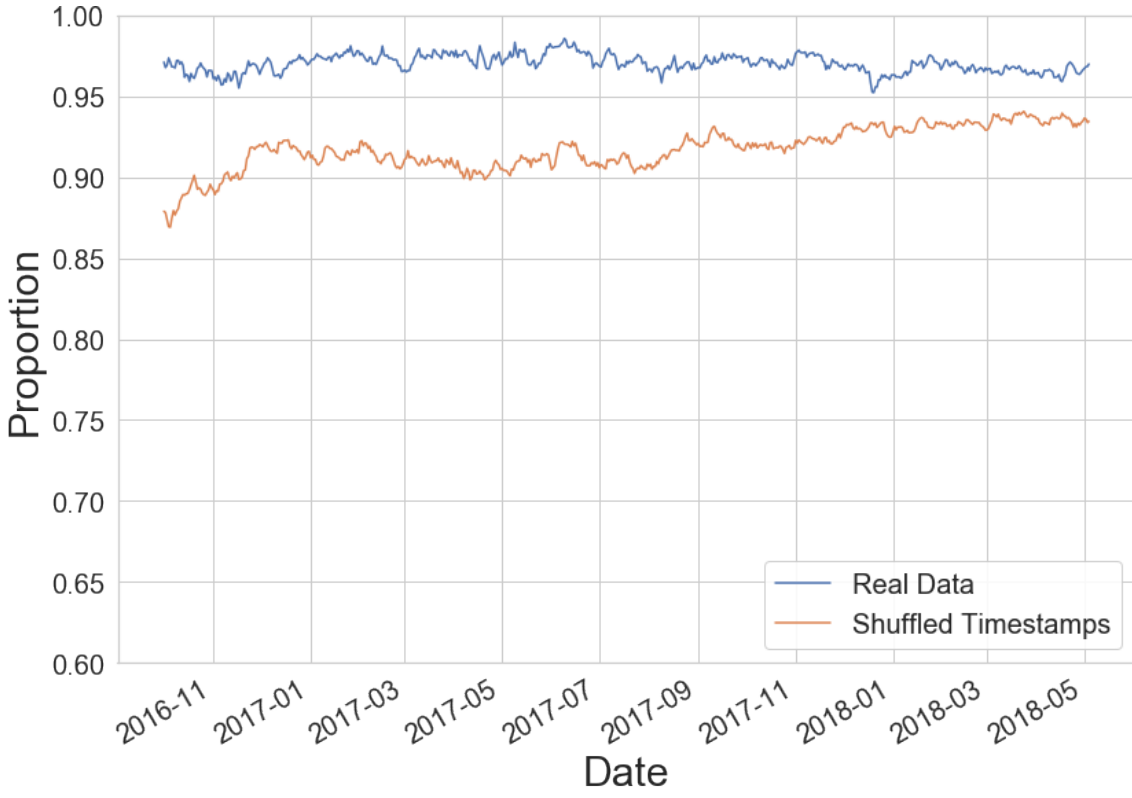
Proportion: Hour window



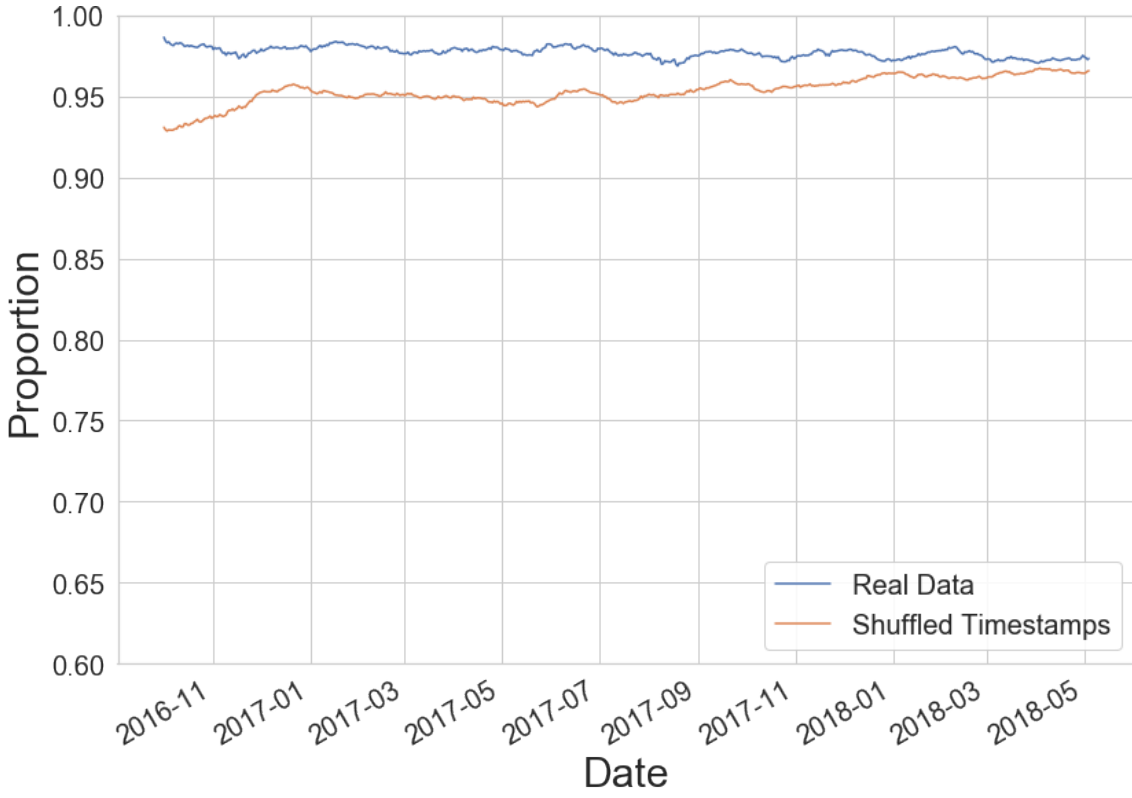
Proportion: Day window



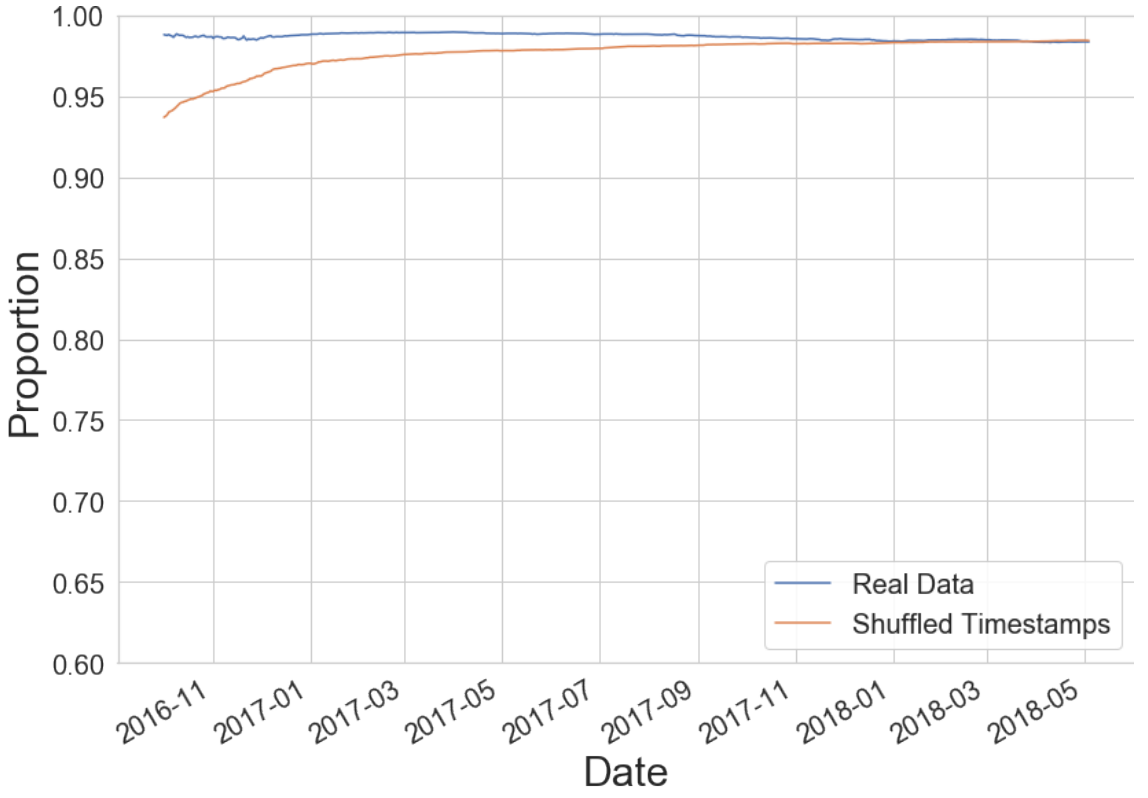
Proportion: Week Window



Proportion: Month window



Proportion: Year window

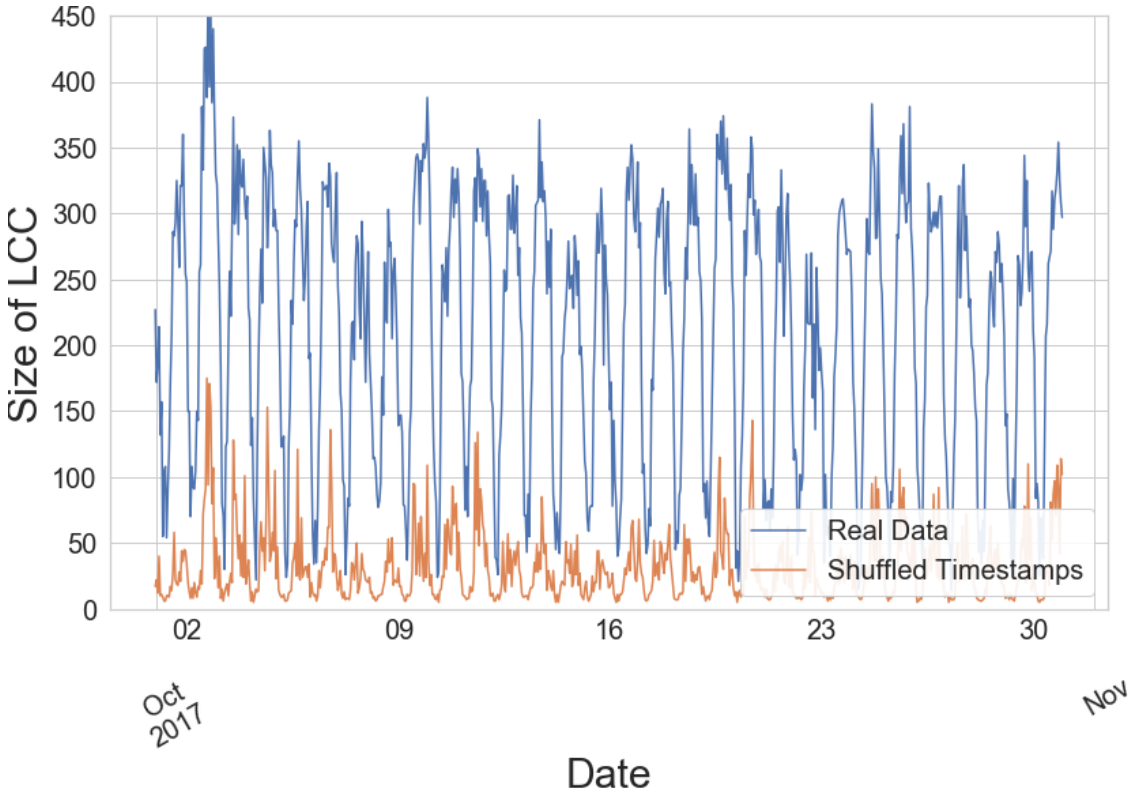


Size of the largest connected component

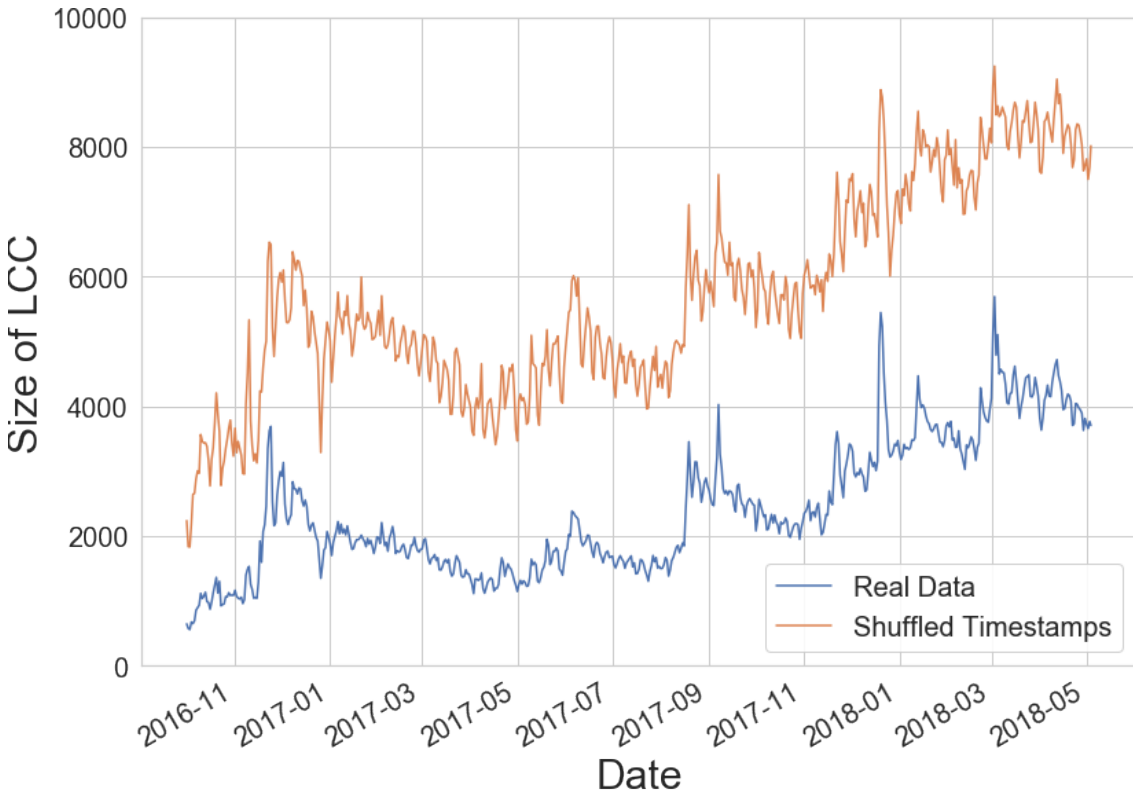
For this part we instead plot the absolute size of the largest connected component for different windows. Like the proportion, the LCC size is smaller for the shuffled data than the real in the hour window, but confusingly this order switches going up to the day, week and month windows.

For the day, week and month window, I suspect that 'memory effect' might explain this too, in that you're more likely to sample 'weak ties' if sampling from the whole time period as opposed to the same number but from a small time interval.

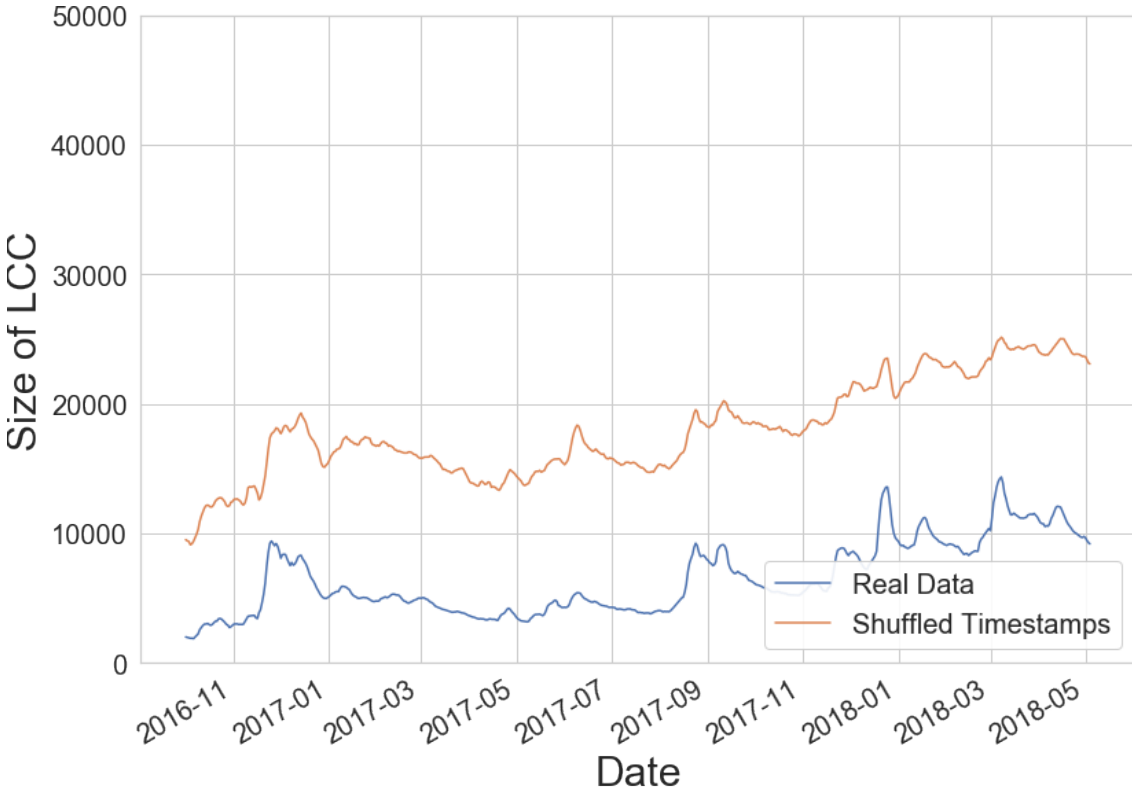
Raw size of LCC: Hour window



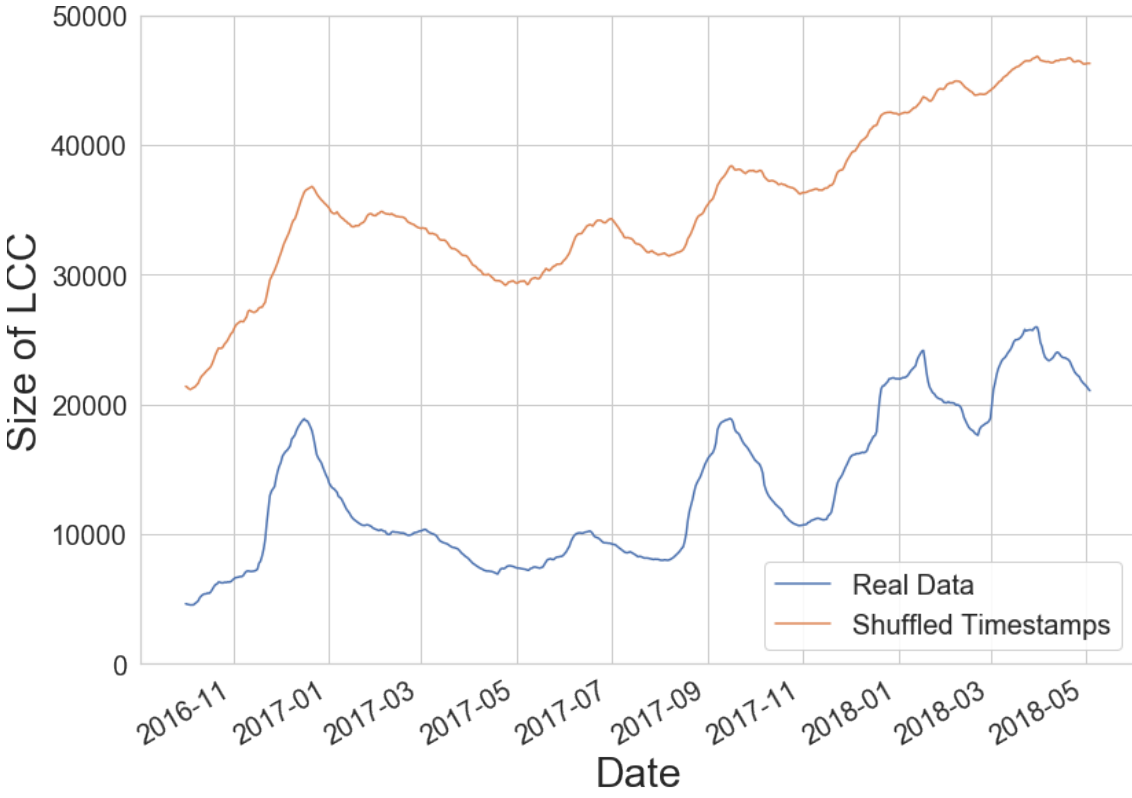
Raw size of LCC: Day window



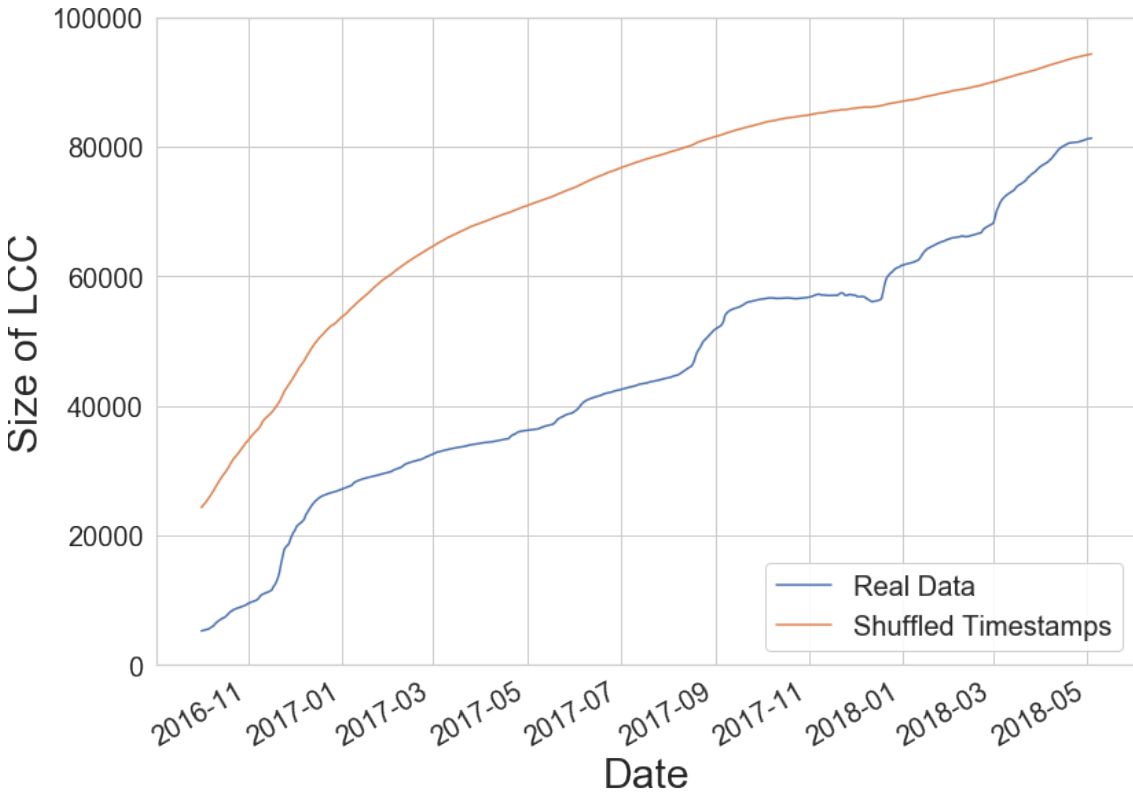
Raw size of LCC: Week window



Raw size of LCC: Month window



Raw size of LCC: Year window



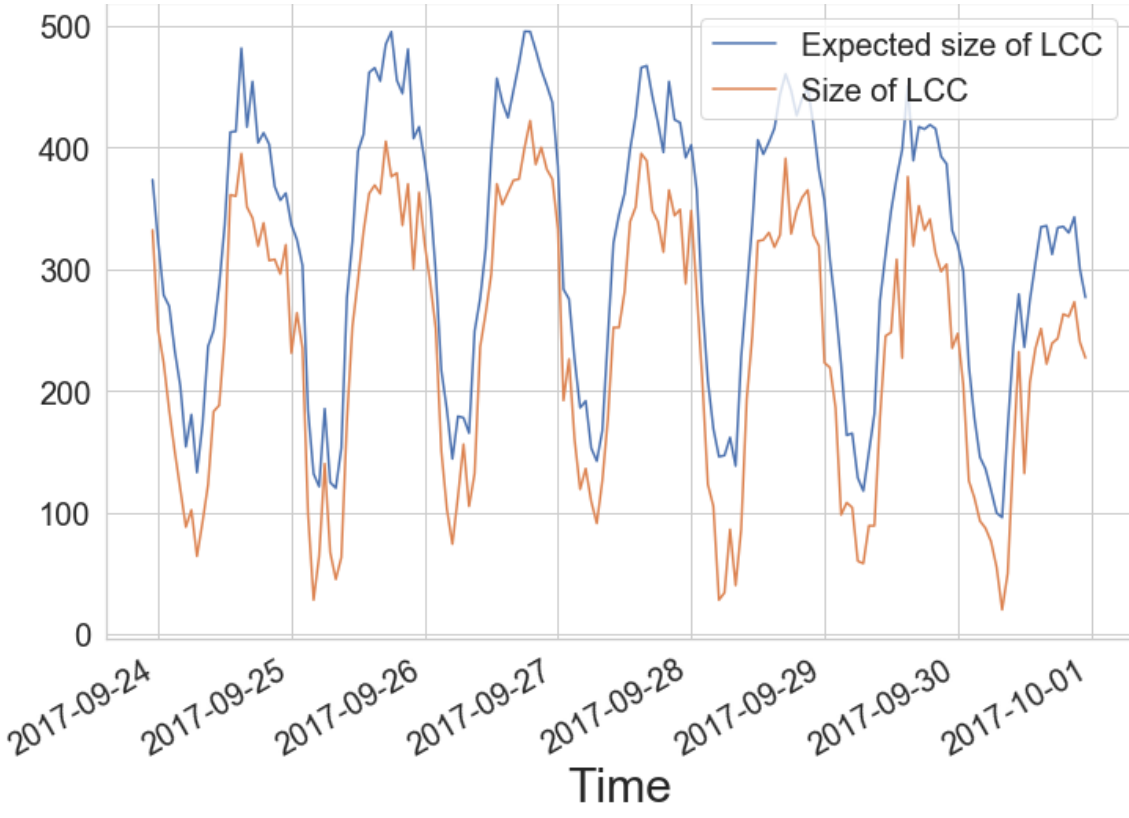
Comparison with Erdos-Renyi Reference model

We compare also (on a smaller time interval because of time constraints!) the size of the LCC for the real data with that of an Erdos-Renyi random graph with the same number of nodes and edges.

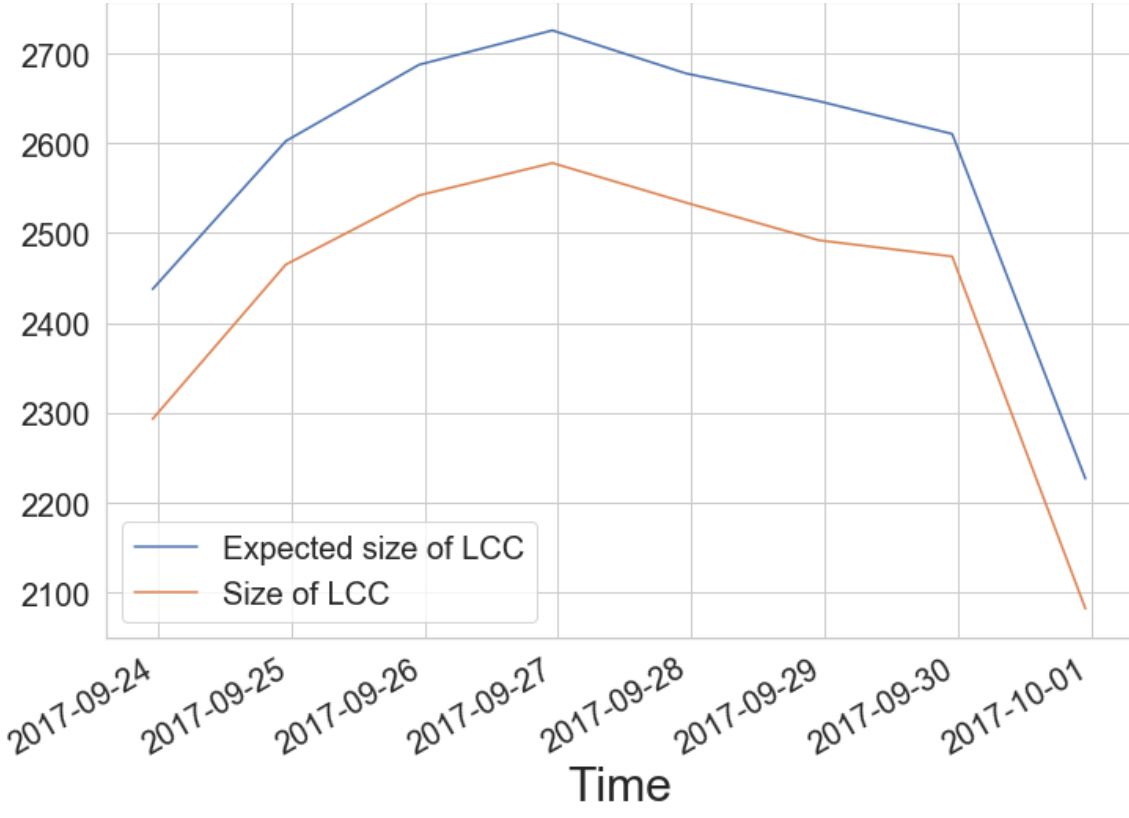
Specifically we generate, for each window size and time, a graph with the same number of nodes and edges as the real data, with the edges assigned at random, and compare the size of the LCC of this with the real data.

For each window size, the LCC is consistently overestimated by the E-R model, which may be due to some strong underlying community structure, whereby the second, third etc largest connected components in the real data are non-negligible in size.

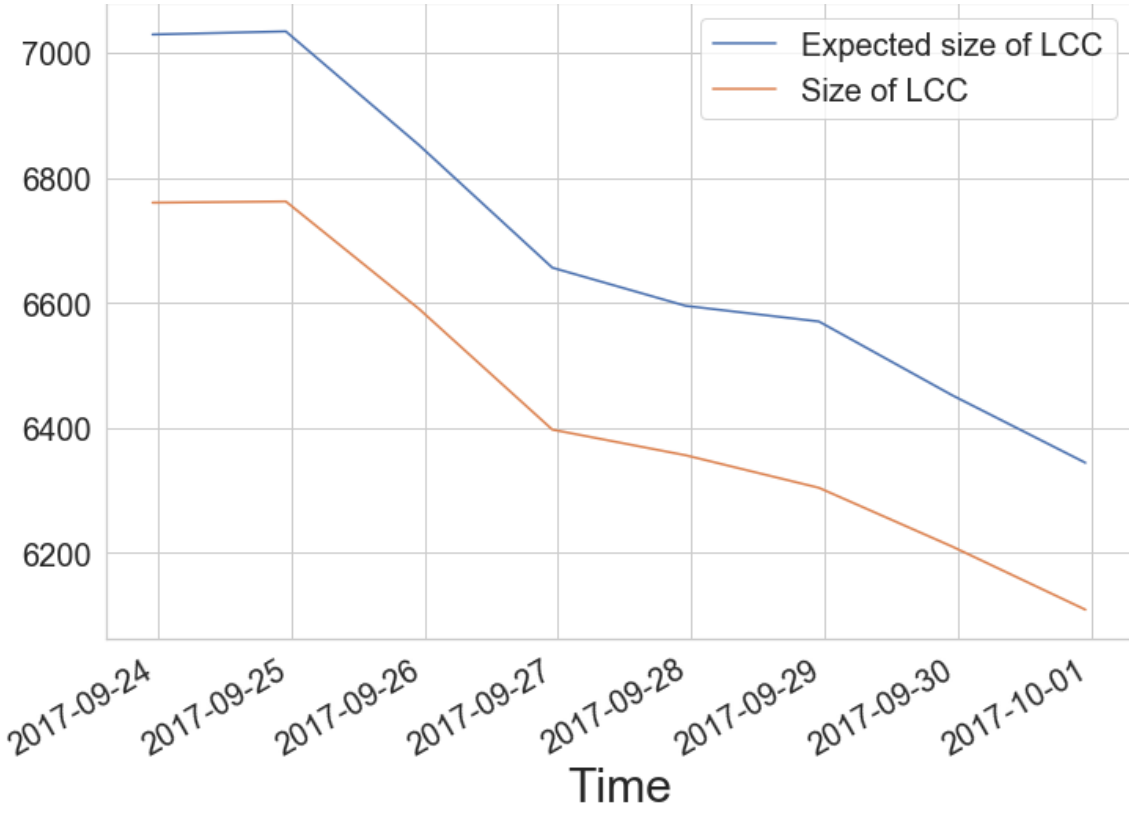
Hour window



Day window



Week window



Month window

Dynamics of top 20 users

For each window size and window, we obtain the top 20 users in terms of in-degree. Is it the case that some users dominate for long periods, or is it more dynamic?

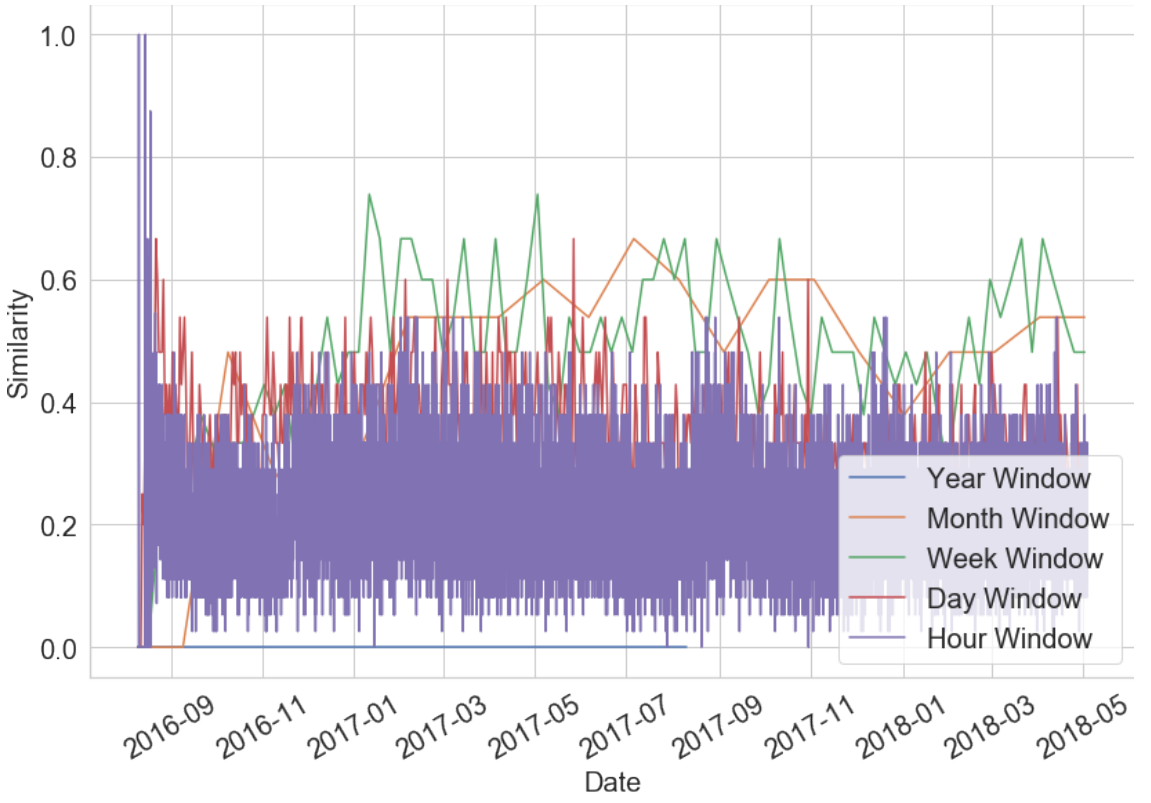
Jaccard Similarity.

For two sets A and B, the Jaccard similarity is given by |A n B| / |A u B| measuring their percentage overlap.

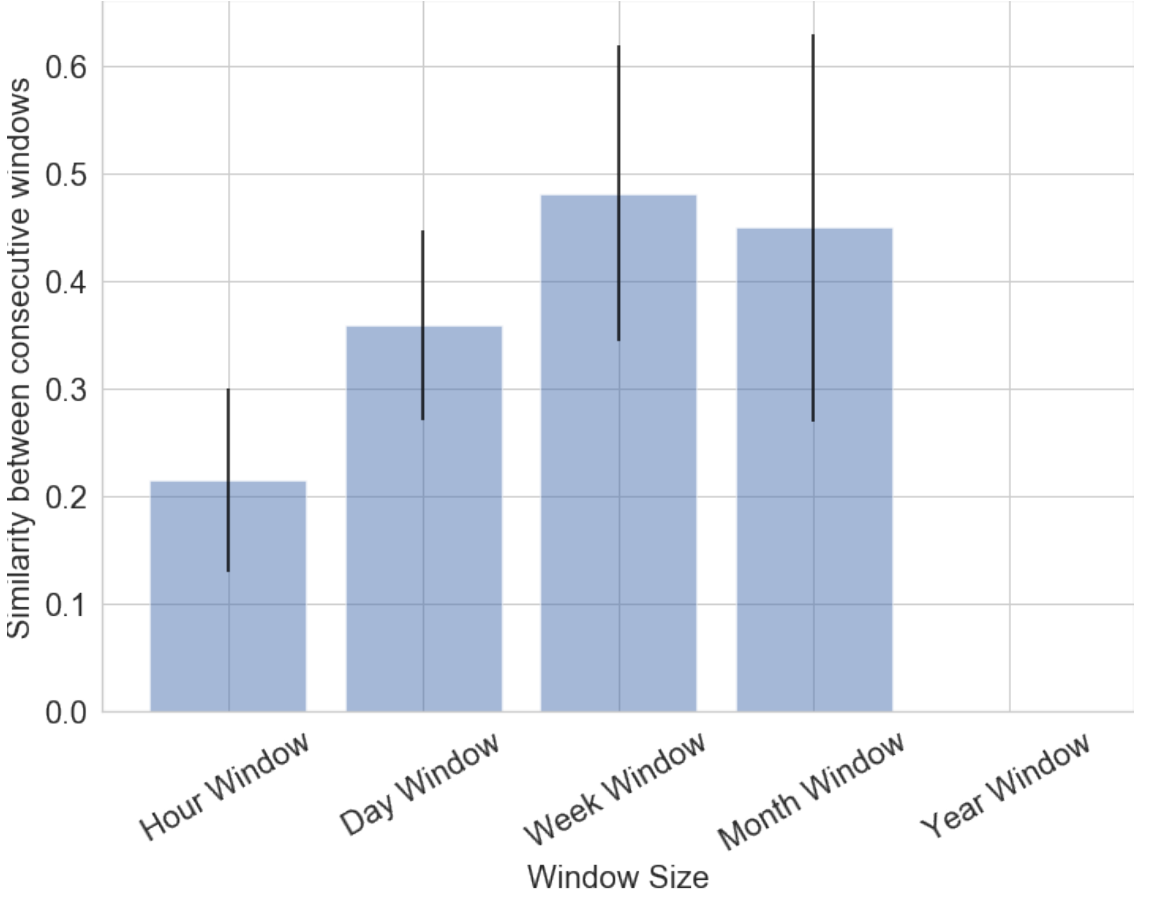
Stability over consecutive windows

For each window size, we calculate the Jaccard similarity between the pairs of consecutive non-overlapping windows' top 20 users. The lower plot shows the mean JS for each window size, but as the error bars overlap and as the number of datapoints drops vastly for each window theres nothing much concrete we can say about it yet.

```
[0.         0.44928546 0.4808423  0.35890023 0.213
73447]
[0.         0.17999554 0.13703931 0.0882071  0.085
3066 ]
```



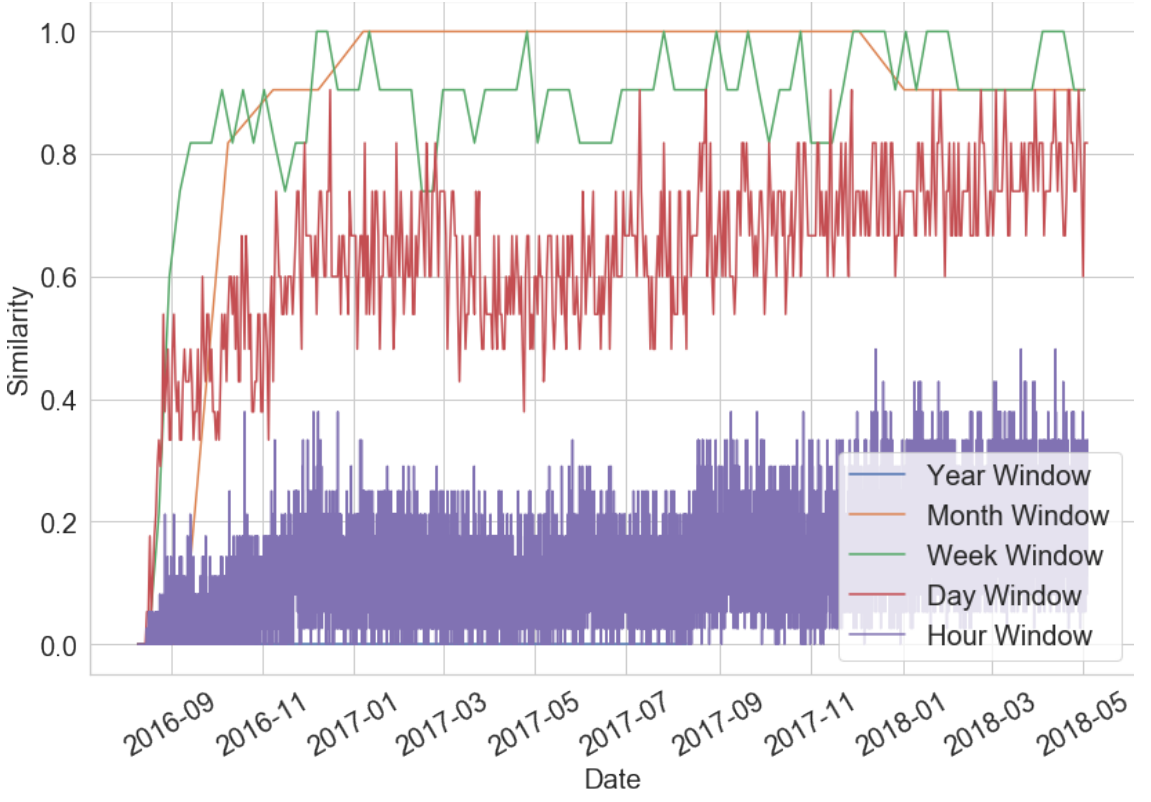

```
['Hour Window', 'Day Window', 'Week Window', 'Mont
h Window', 'Year Window']
```



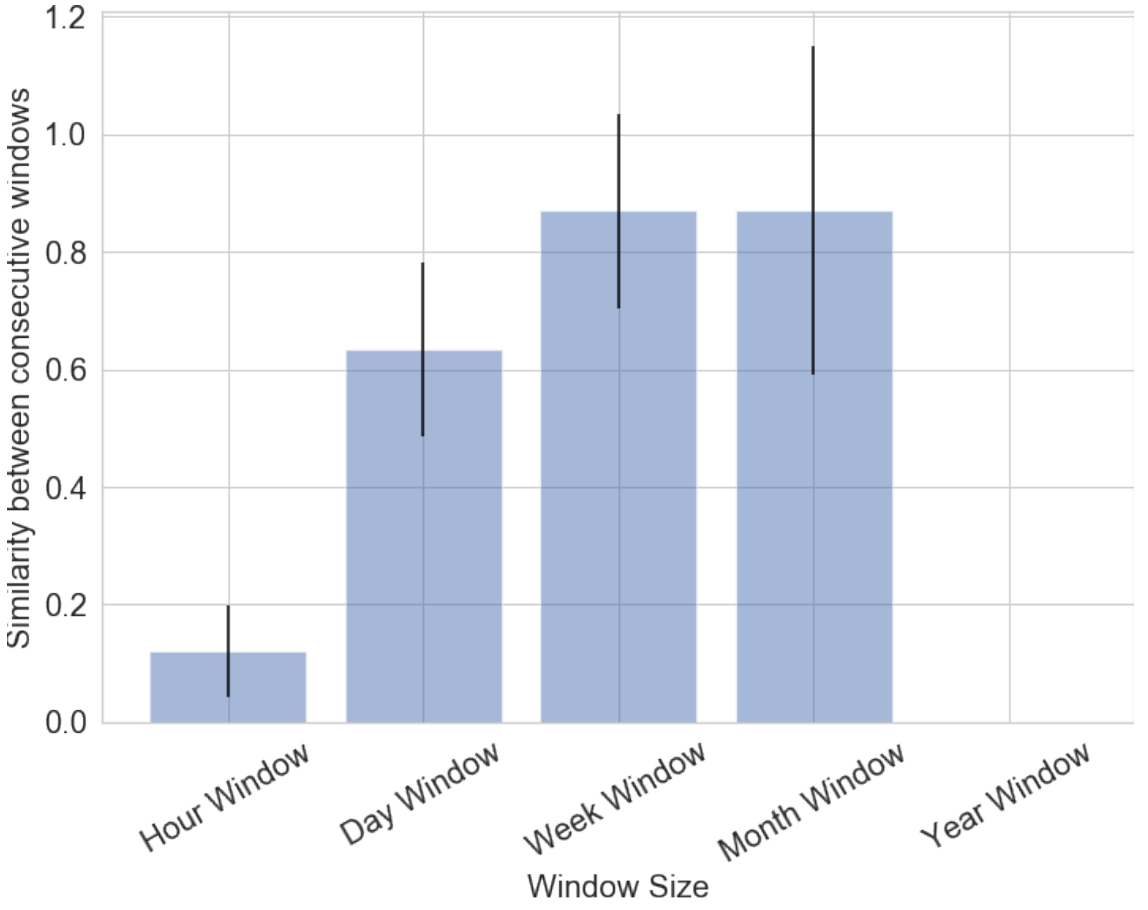
Effect of shuffling timestamps

As with the connected components, we look at the effect of randomly shuffling the timestamps. It seems to have a "smoothing effect", suggesting that in the original data, many of the users who reach the top 20 may only do so for a short period of time.

```
[0.         0.87052342 0.8693952  0.63457222 0.120
76518]
[0.         0.28023836 0.16617867 0.14761091 0.076
94133]
```




```
['Hour Window', 'Day Window', 'Week Window', 'Mont
h Window', 'Year Window']
```

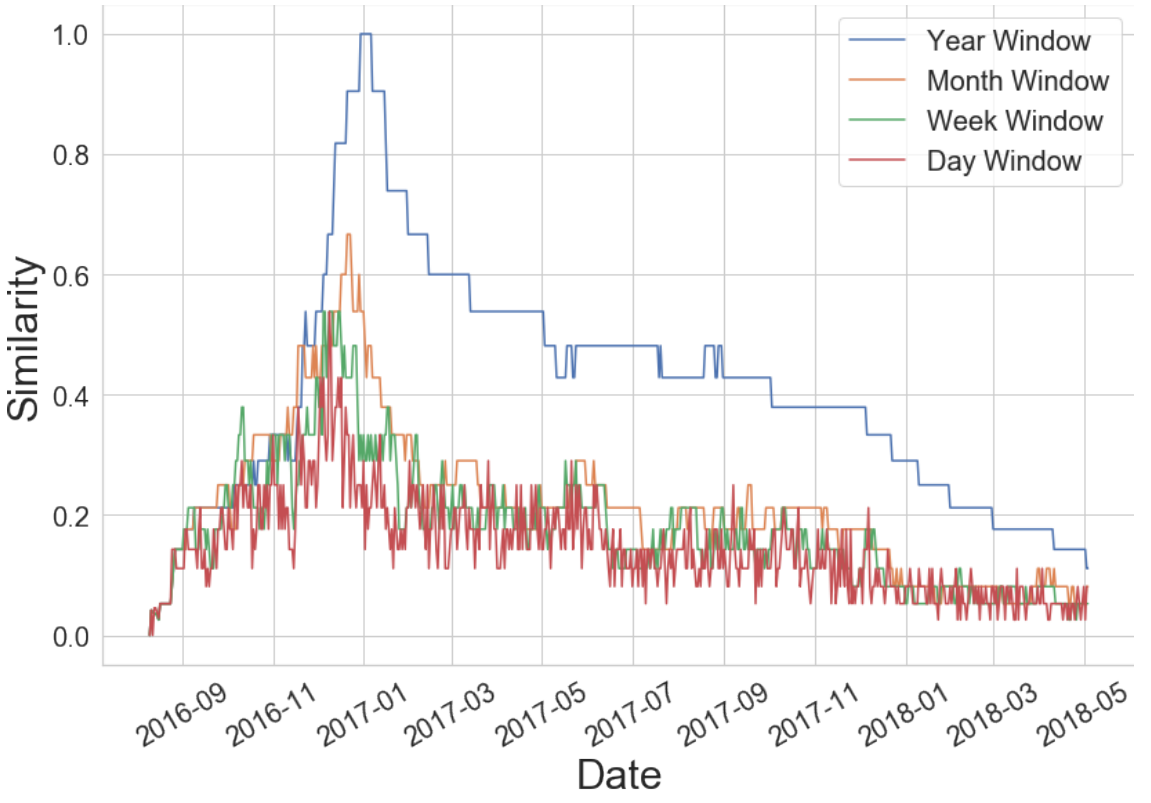


Comparison with reference point

We also compute the similarity (for the real dataset) between top 20 users in each window size and the top 20 all-time top users at a reference point around Nov 16. We find:

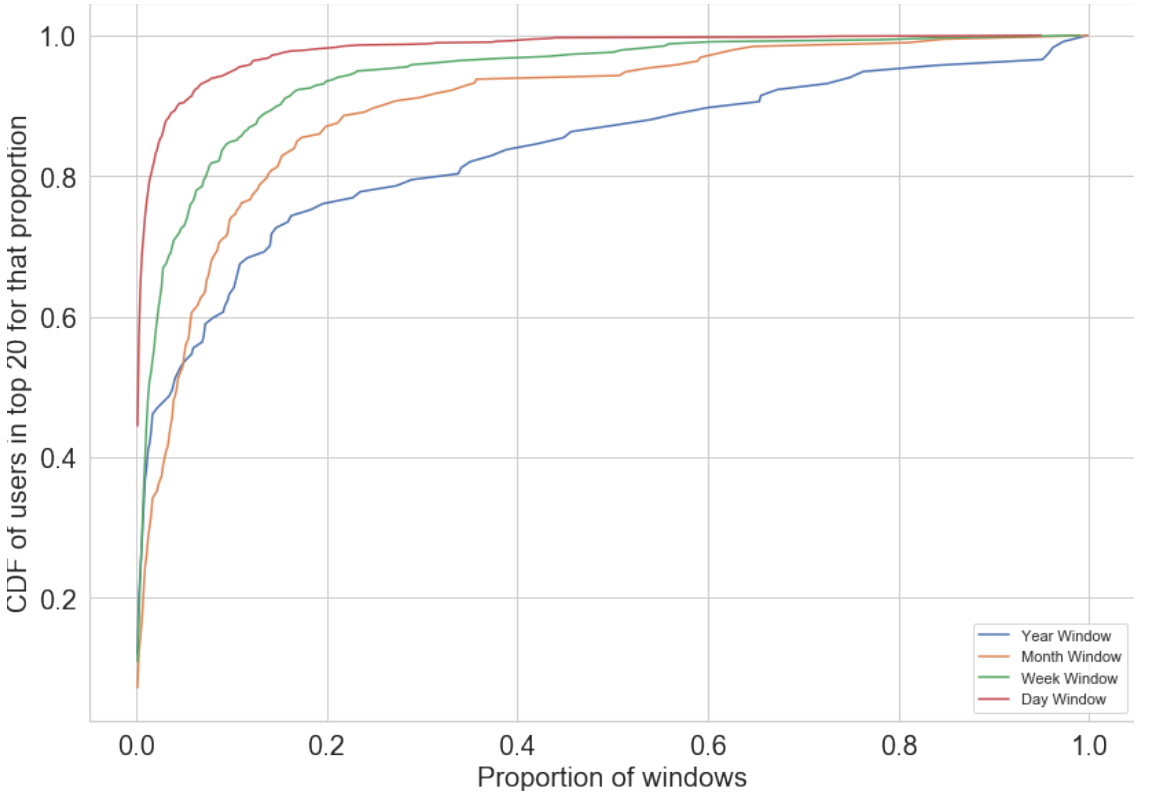
- By the end of the time series, even in the largest window nearly all the top 20 users have been replaced, with the JS dropping to 0.1
- Even at the same time as the reference point, there are different top 20 users in smaller window sizes.

```
31536000000
2592000000
604800000
86400000
```

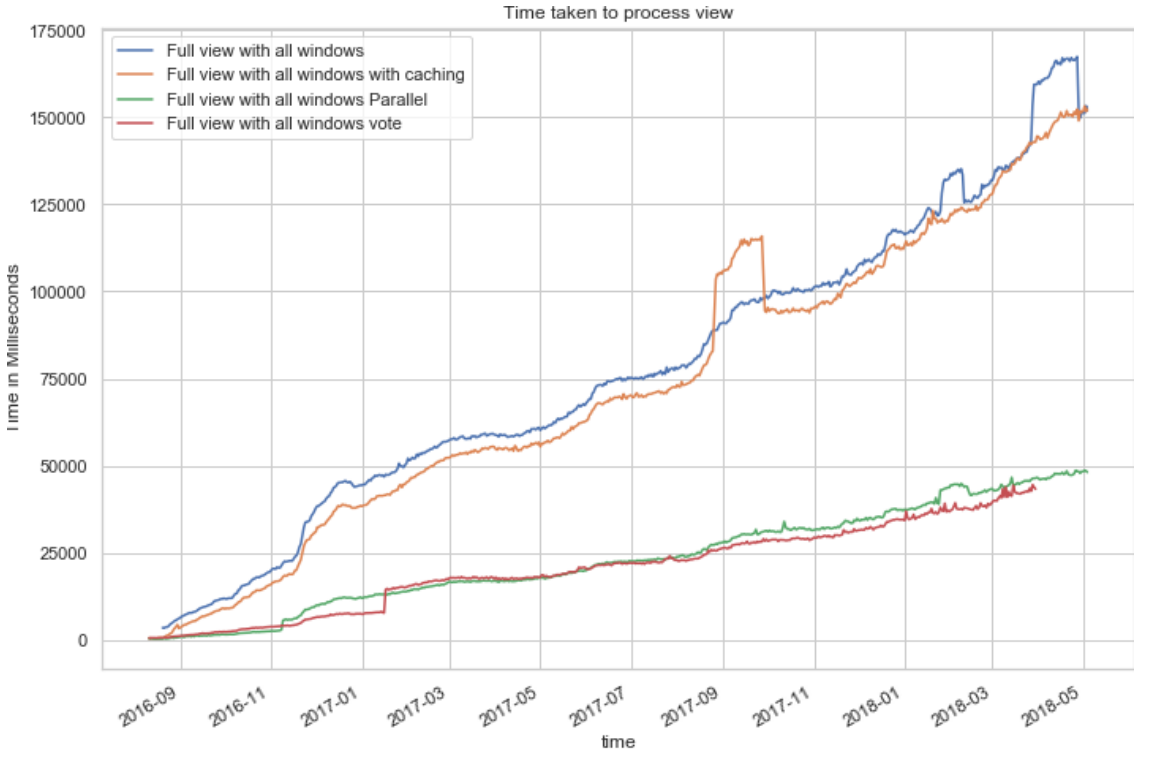


For how long do users enter the top 20?

For each user who has ever been in the top 20 in any window, we count the number of windows (for each size) in which they appear in the top 20. For example, 90% of these users appear in the daily top 20 for less than 10% of the time period. Needs ironing out a bit I think as the explanation takes *me* ages to get my head around!


CDF of users in top 20 for that proportion
Proportion of windows
Year Window
Month Window
Week Window
Day Window

OLD STUFF


Time taken to process view
Time in Milliseconds
time
Full view with all windows
Full view with all windows with caching
Full view with all windows Parallel
Full view with all windows vote