**Week 4 Writeup**

# Week 3 Summary

1. **[MON]** Fit 88/101 distributions in Scipy (DISTRIBUTIONS.ipynb) and compare GOF using KS & MAE
2. **[TUE]** Weekly meeting & writeup. Filter out multi-robot data (FILTER.ipynb)
3. **[WED]** Use filtered data to fit 88 distributions in Scipy against 25 edges with most data. Select models with low KS & MAE (section 3 of MAIN3.ipynb)
   1. Lognorm, Powerlognorm, Invgauss, Mielke, Fatiguelife, Invgamma
4. **[THU]** Compare models (see Wk3 Writeup for more details)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Distribution** | **Parameters** | **MLE** | **Conjugate priors** | **Common use** | **Difficulty** |
| Lognorm | Mu, sigma | Simple | Transform lognormal to normal | If X is lognormal, then lnX is normal | Easiest |
| Powerlognorm |  | Hard | None that I could find |  | Hard & unsuitable |
| Invgauss (Wald) | Mu, lambda | Simple | Exists, but complicated | Time for Brownian motion to reach a certain level | Medium |
| Mielke  (Dagum / burr) | a, b, p | Hard | None for optimising both parameters | Modelling income & wealth distributions | Hard & unsuitable |
| Fatiguelife | Mu, gamma, beta | Hard | None for optimising both parameters | Models failure times due to crack growth | Hard & unsuitable |
| Invgamma | Alpha, beta | Algorithm exists | Algorithm exists | Used as a prior | Medium |

1. **[FRI]** Fitting distributions (sections 4-7 of MAIN3.ipynb)
   1. Bayesian optimisation of Lognormal
   2. Manual fit to lognormal, invgauss, invgamma using ipywidgets
   3. Test for Gaussian mixture model – unsuitable
   4. Kernel density estimation

**Unfinished:** Network visualisation.

# Week 4 Plans

**Goal for this week:** fit distribution models to the edge data using Bayesian approach. Select the best out of lognormal/invgamma/invgauss

1. Additional filtering
   1. Did the robot start at the origin node?
      1. Order data for each robot and remove data after is\_final == True
      2. Separate by run\_id & robot\_id and order by time
2. Create a new repo with Charlie’s dataset – congestion\_data
3. Models
   1. Implement invgamma & Invgauss
   2. Hard cutoff minimum / offset
   3. Use probabilistic forecasting to a threshold (i.e. train on a subset of data for one edge and test against unused data for the same edge)
4. Where are the opportunities for generalisation? Use KS between each pair of edges in the environment to check for similarities
   1. Same length, same no. of branches
   2. Use Walmart to start / STRANDS
5. Does the generalisation hold if we increase the number of robots on an edge
   1. Walmart\_targeted/ Blenheim
6. Data from Strands robots
   1. How well can we fit noisy data?
   2. We might overfit if the data is very clean
   3. Tsc is office / aaf is hospital
7. Create Network visualisation map
8. Research update for Tues 27 July (30 mins)

**Goal for next week:** start looking at how to transform the posterior of one edge into the prior of another edge based on spatial similarities

# Lognormal distribution fitting

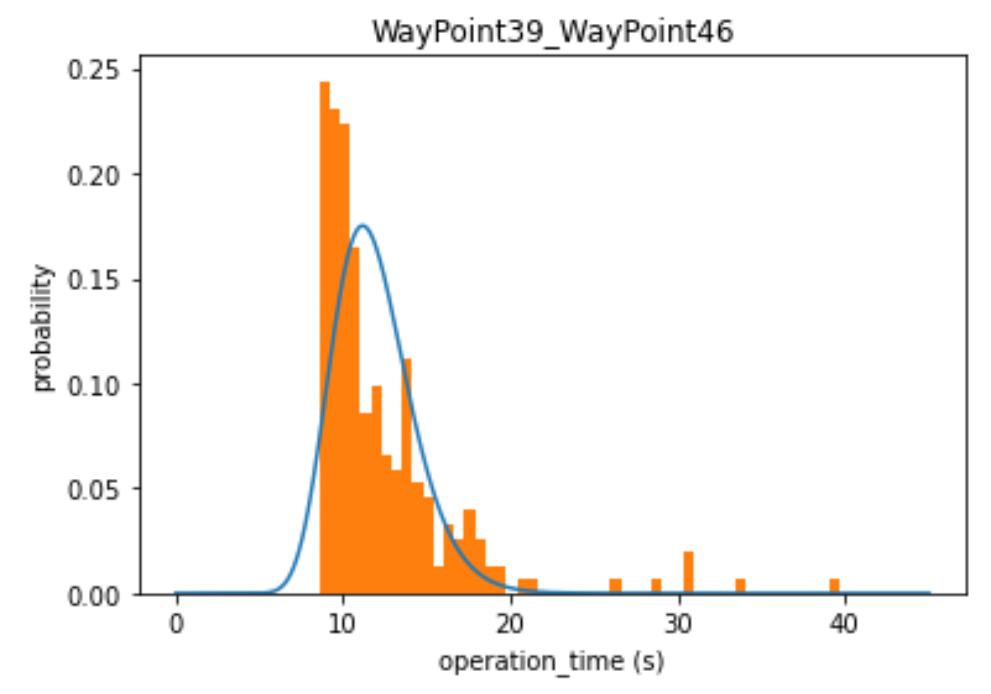
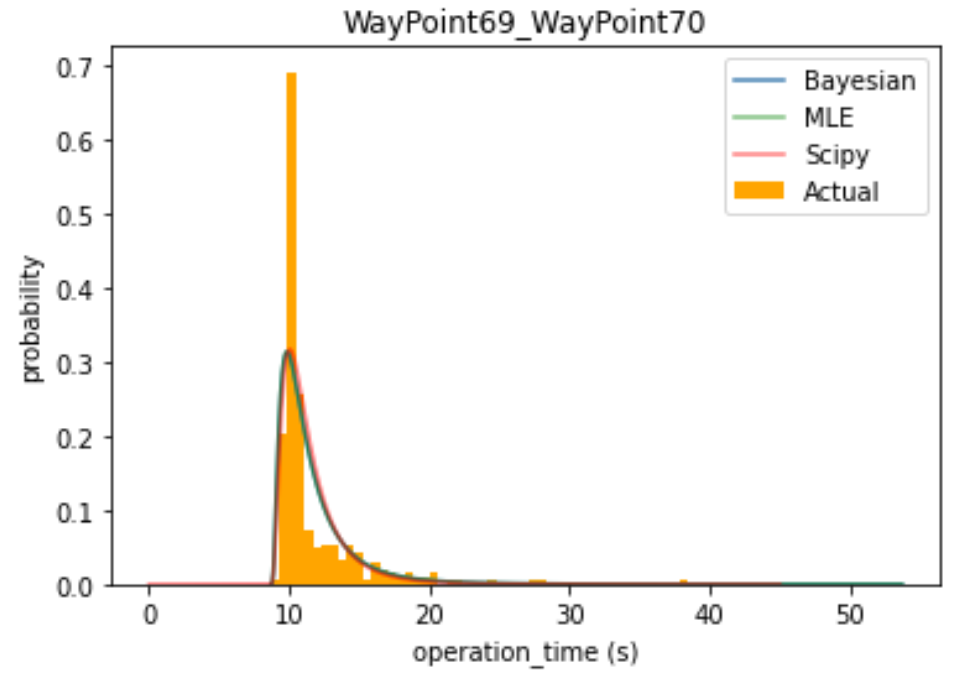
**Section 2 of Wk4/MAIN4.ipynb**

**You get a better fit if you include an offset**. If you do not include the offset, the fact that the lognormal distribution has only 2 parameters will result in losing skew in order to allow the distribution to peak at a higher time.

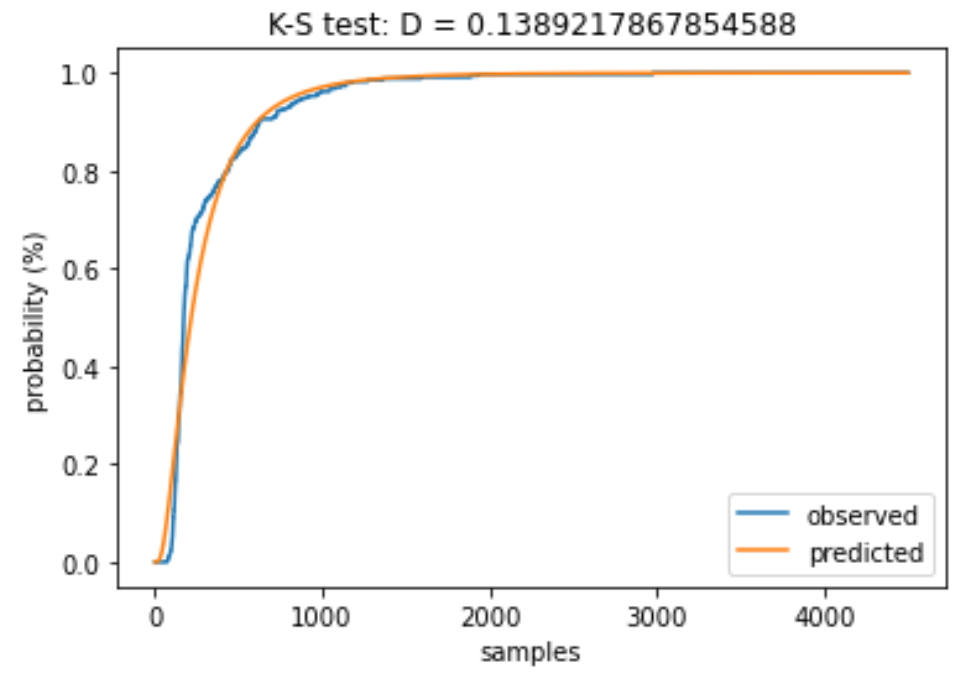
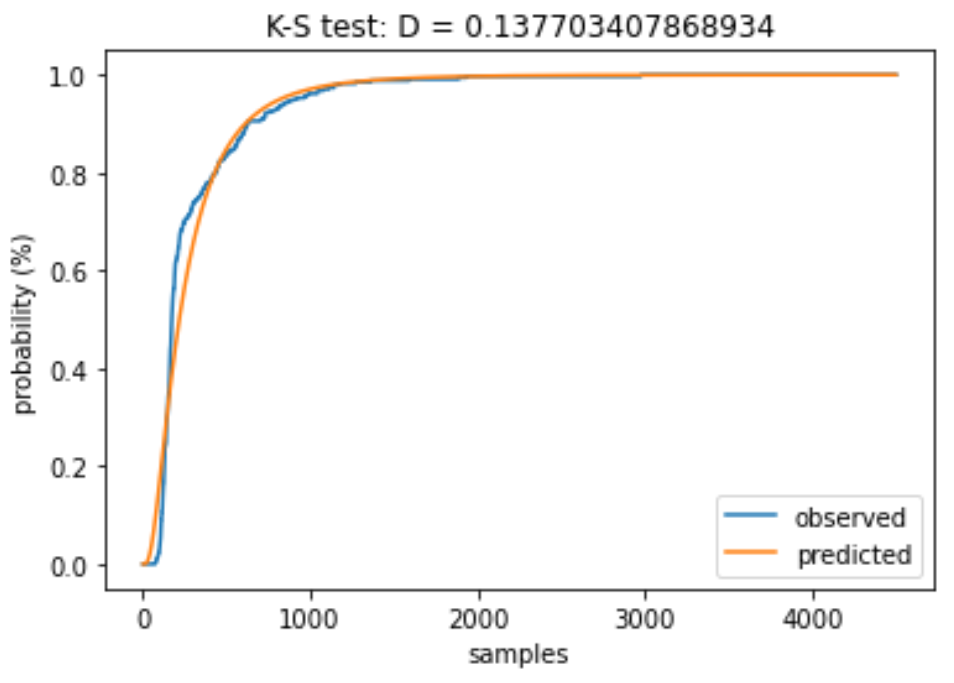
**I used the minimum observed duration as the offset.**

* This reflects the fact that there should be an absolute minimum achievable time.
* However, this is sensitive to outliers. Therefore, data pre-processing could also remove data points that are very far away below the mean. Again, this is not so clear if we do not have many data points in total for an edge transition.

Without offset (left, MAIN3.ipynb) & including offset (right, MAIN4.ipynb):

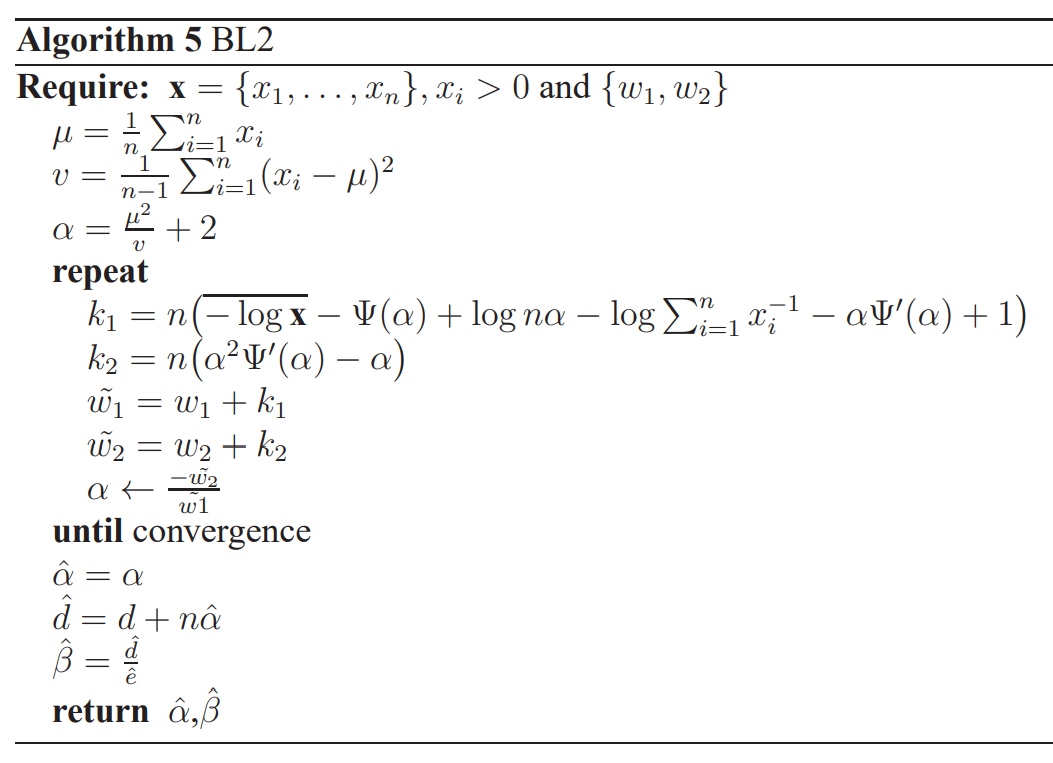
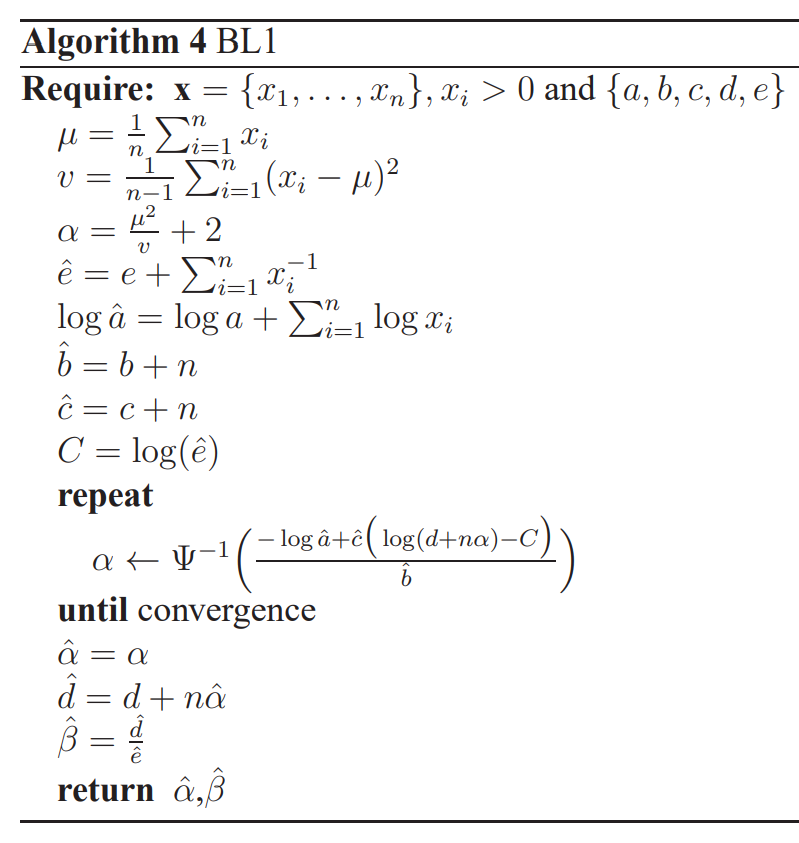
For the offset lognormal distribution, the KS-statistic of the Bayesian method is slightly higher than that of the Scipy fit. Left-hand graph is KS-statistic of scipy fit. Right-hand graph is KS-statistic of Bayesian method.

# InvGamma Distribution fitting – method 1

**Section 3 of Wk4/MAIN4.ipynb**

This method of fitting the Inverse Gamma distribution uses the BL1 & BL2 approximation algorithms from LLera’s paper on “Estimating an Inverse Gamma distribution” (see below).



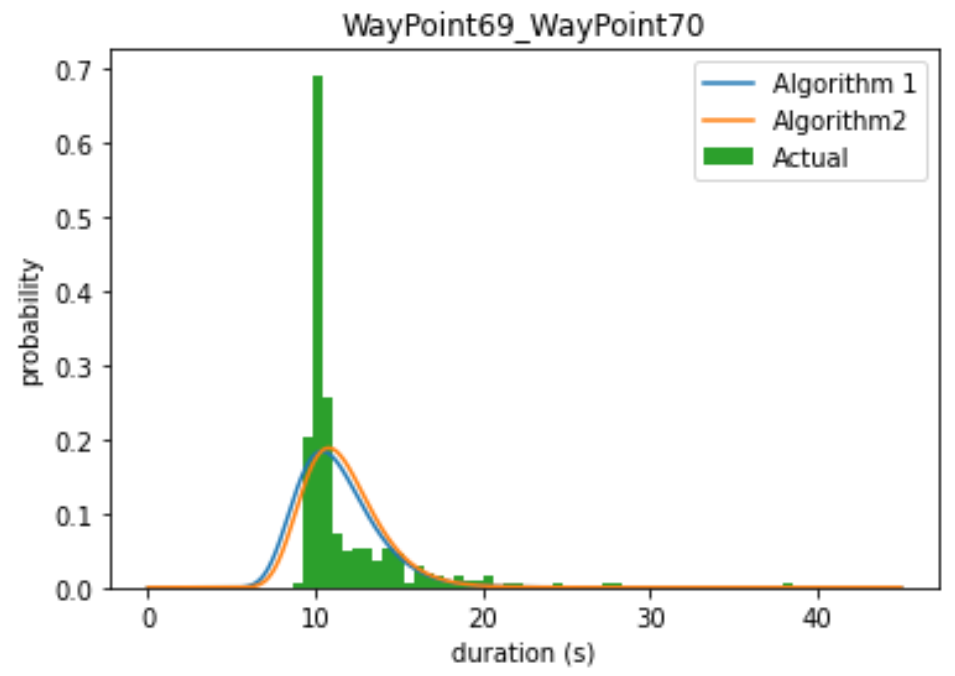
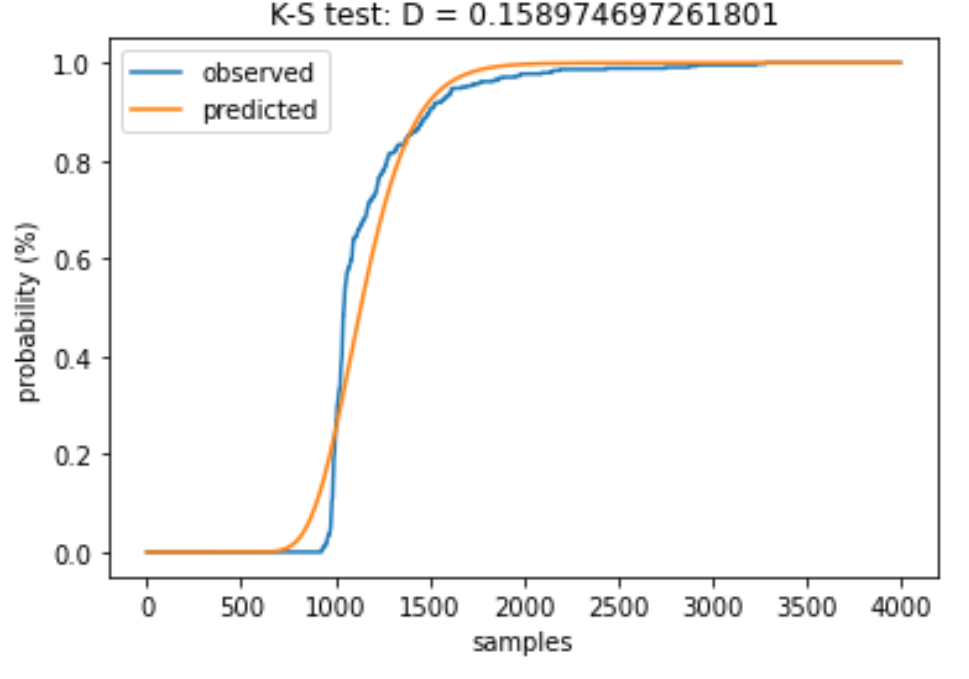
These algorithms are fast (approx 1ms) since optimisation occurs only for 1 parameter of the invgamma distribution. The other parameter is directly calculated from the observed data and the other parameter.

However, these algorithms result in a poor fit since they do not work if we offset the data along the t-axis.

BL2 has better performance than BL1.

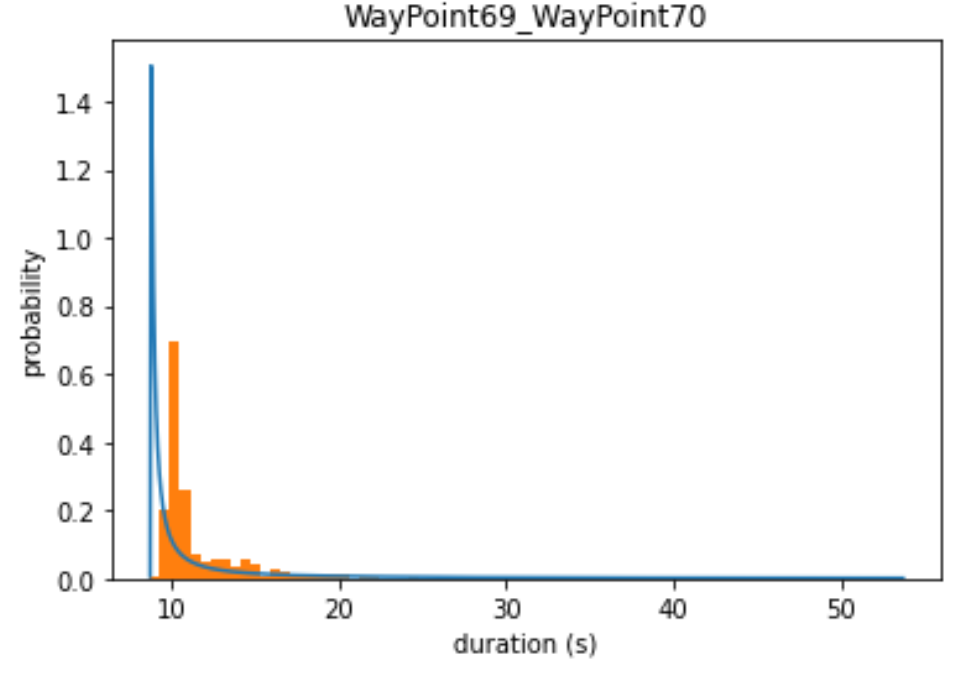
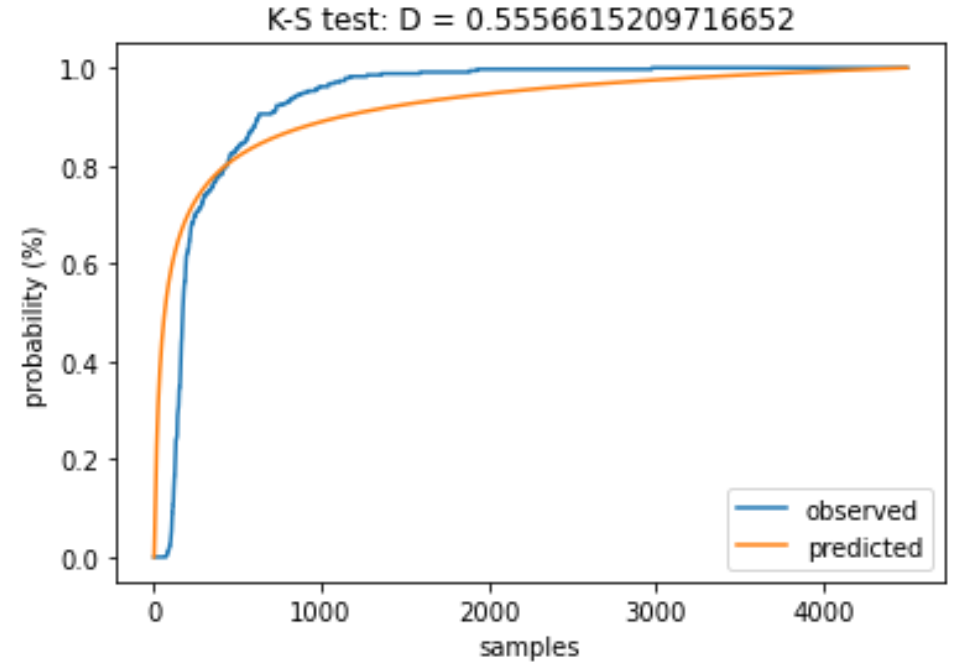
I tried 3 approaches for using these algorithms:

1. No offset

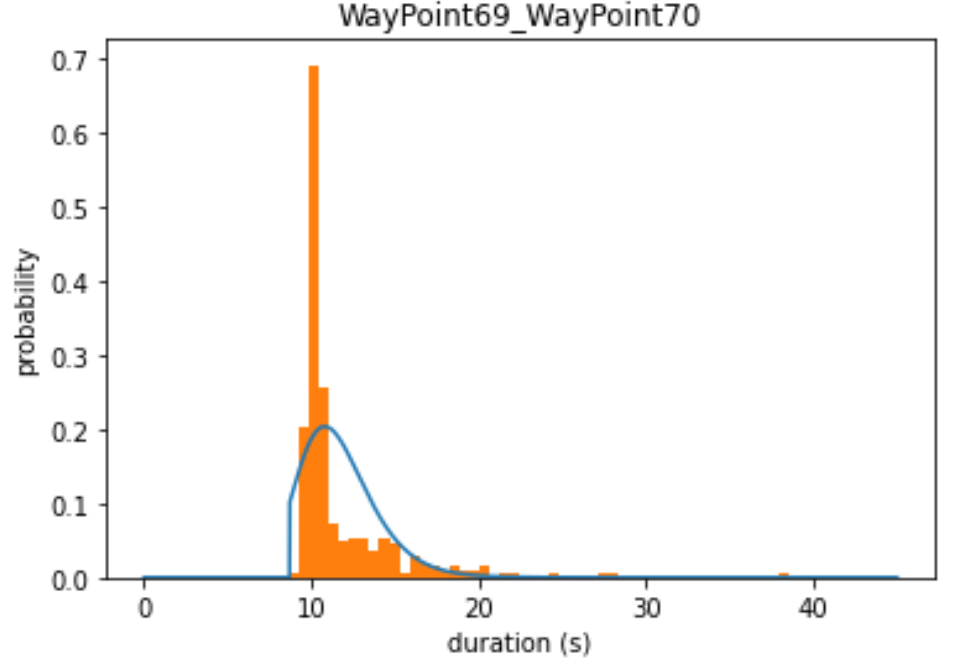
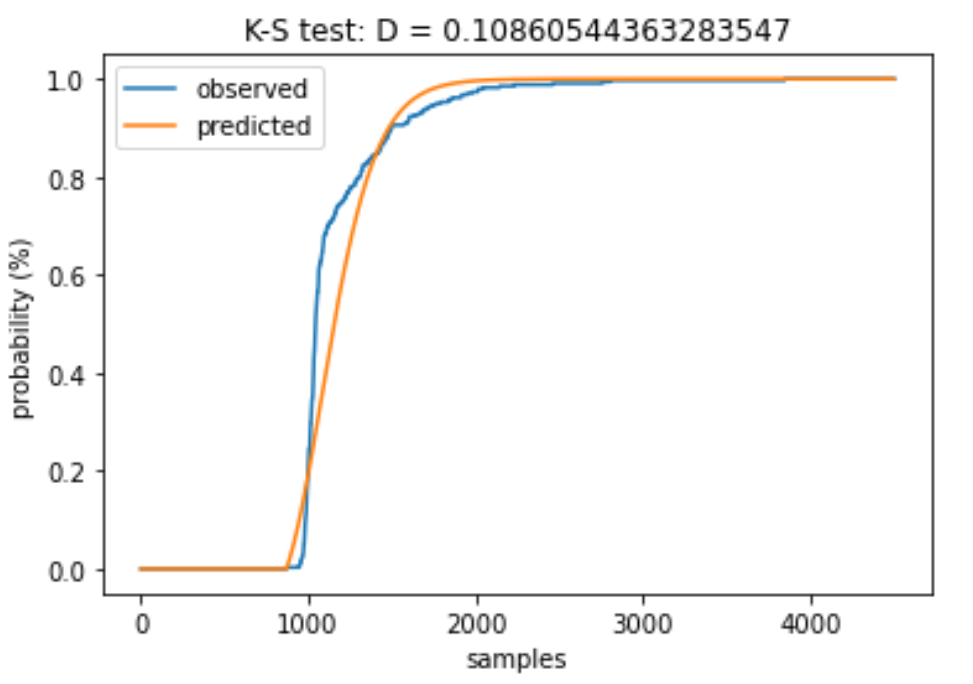
1. With offset on t-axis

Subtract the offset (equal to the minimum observed duration) from the duration data. Obtain parameters for modified dataand shift predicted distribution back along t-axis.

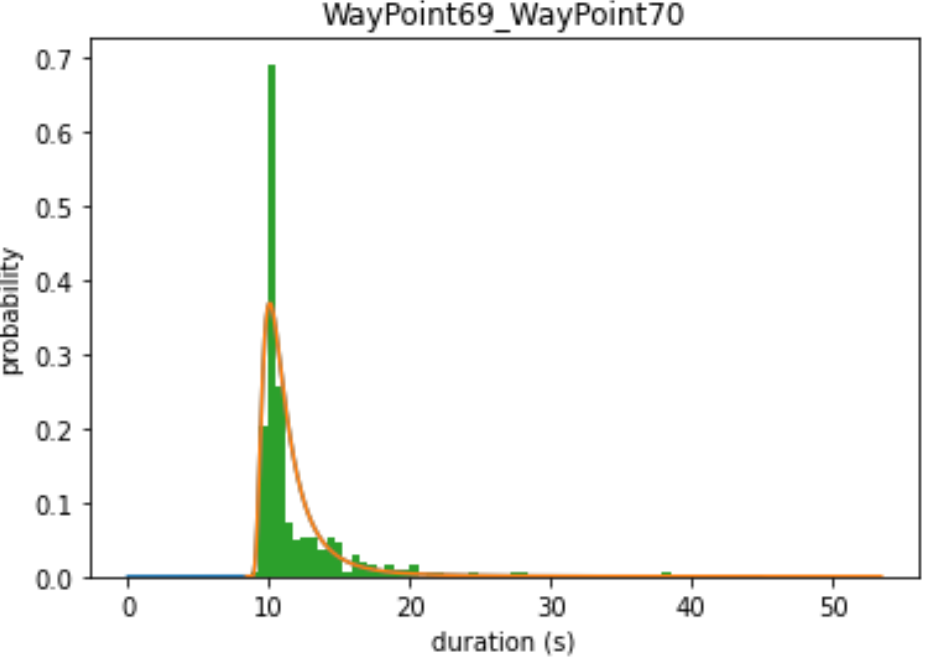
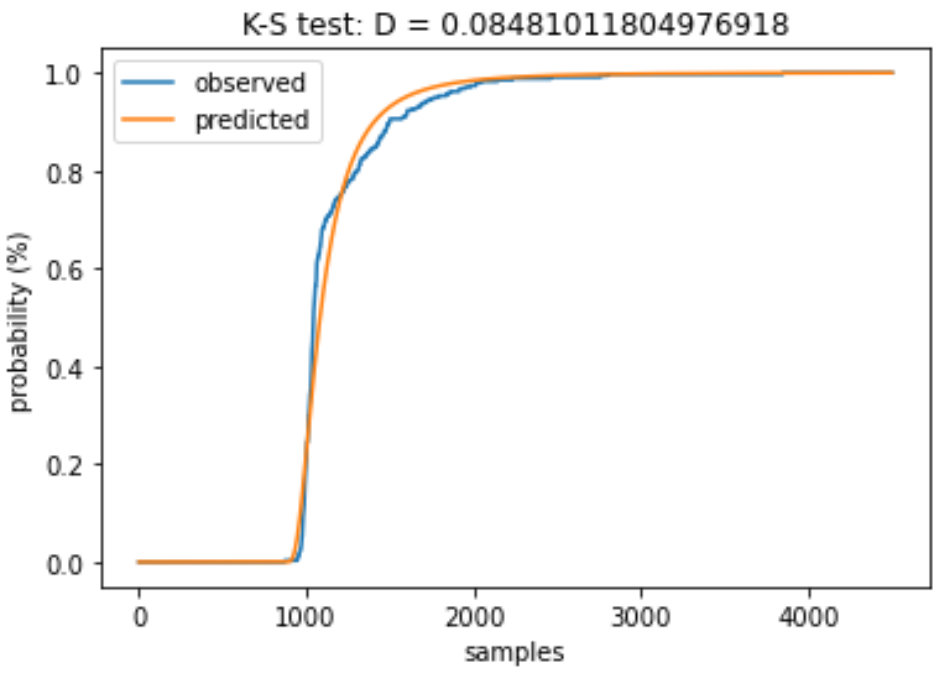
 

1. Hard cutoff

Fit the observed data. Make all probabilities below the minimum observed duration zero and normalise.

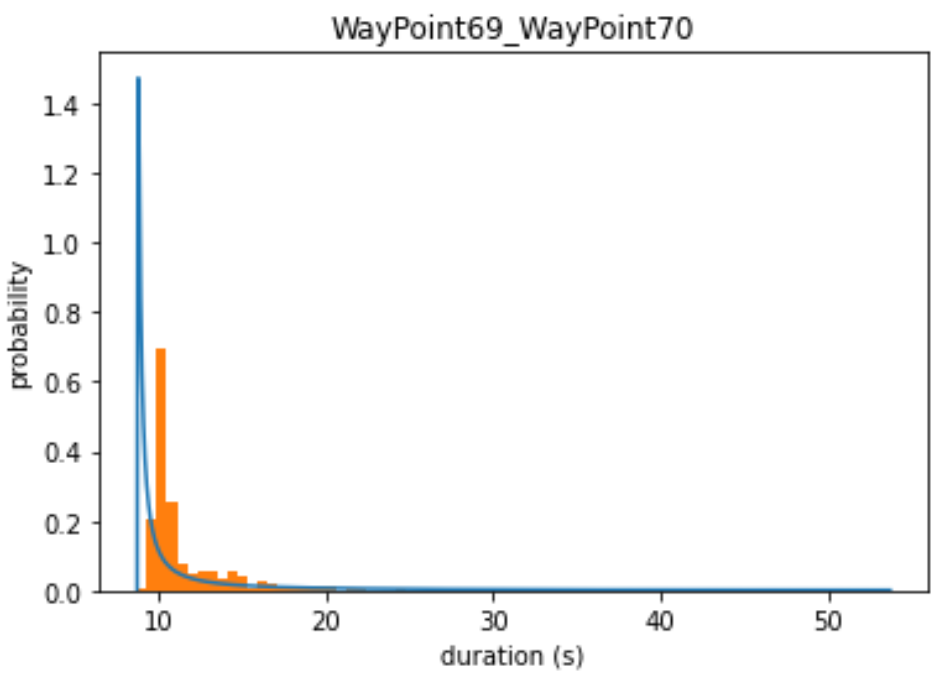
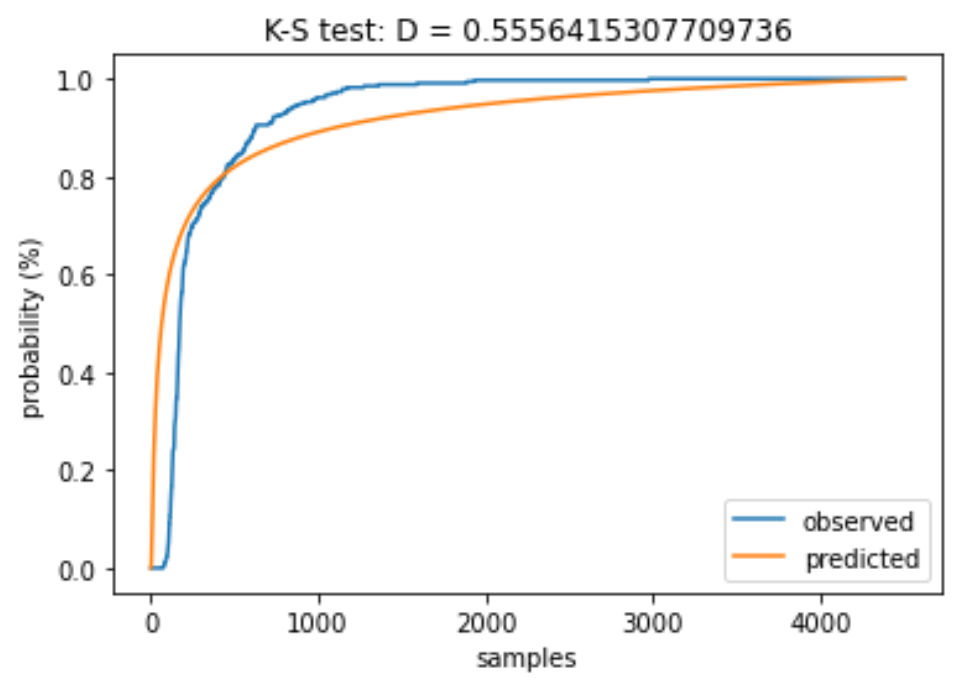
 

Compare to Scipy fit (MLE)

# InvGamma Distribution fitting – model 2

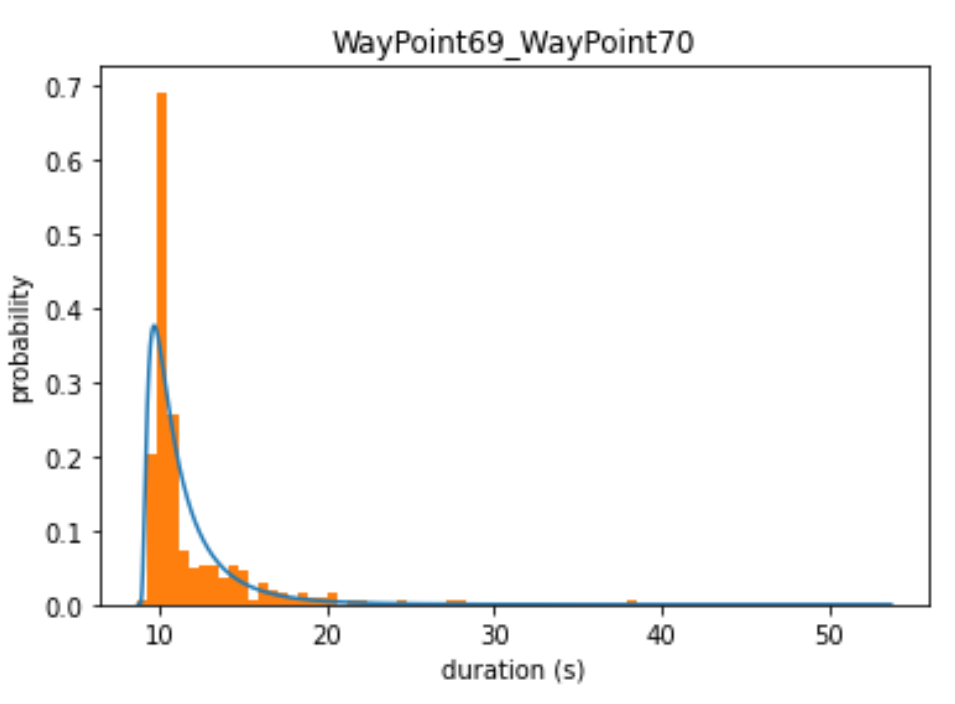
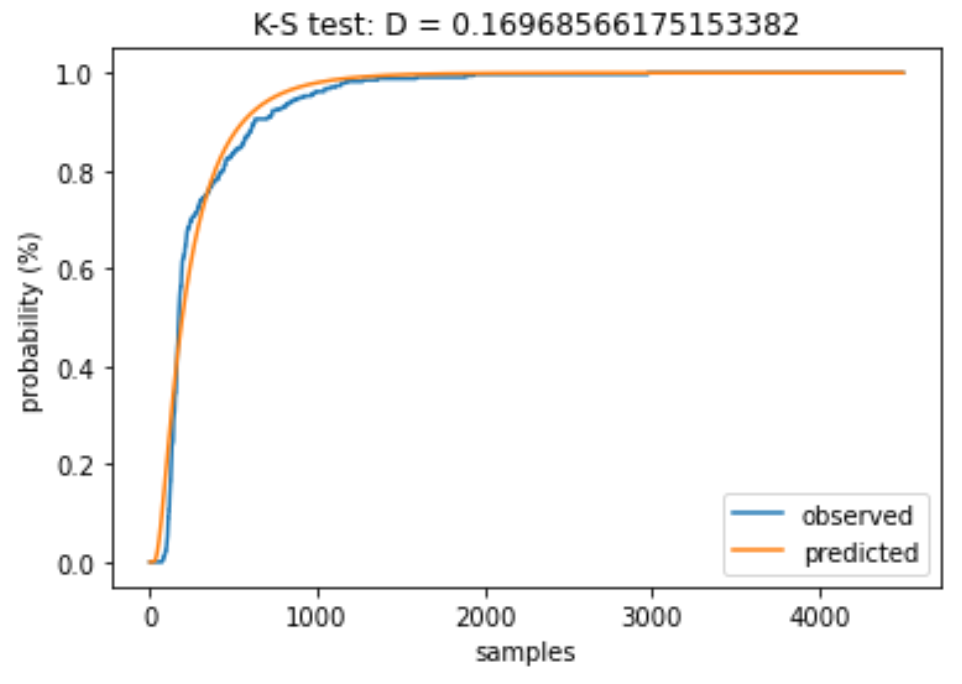
Using Pandey’s paper on “Bayesian Estimation of Inverse Gaussian Distribution” as inspiration, I used a prior formed from a product of the single-parameter conjugate priors in Llera’s paper. This assumes independence of the parameters. However, this may not be conjugate and also does not work well.

# InvGauss Distribution fitting

Uses the method from Pandey’s paper on “Bayesian Estimation of Inverse Gaussian Distribution”.

However, the Bayesian optimisation actually returns the reciprocal of the Lambda parameter of the Inverse Gaussian distribution, rather than the Lambda parameter.

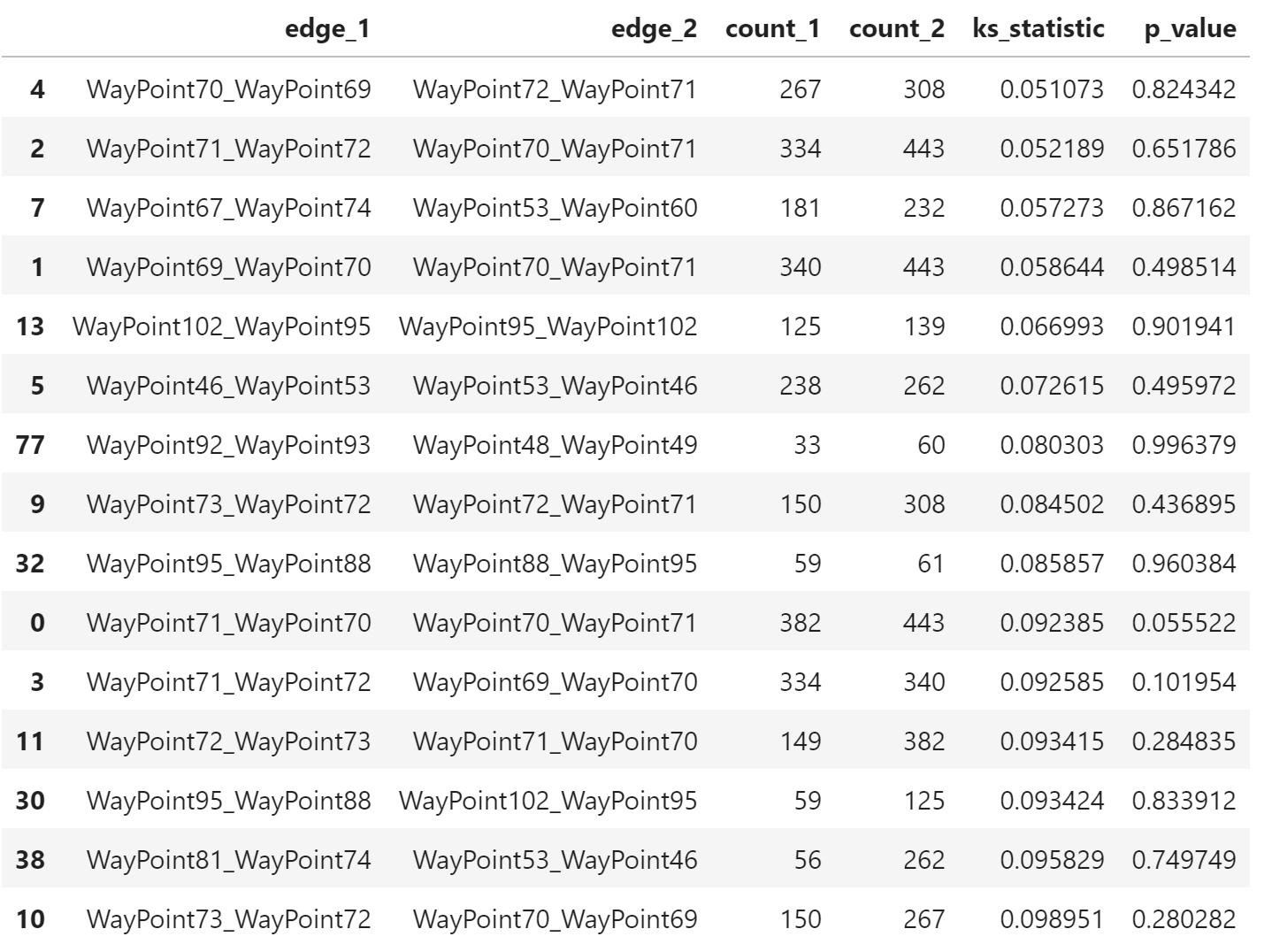
 

# Summary

|  |  |  |
| --- | --- | --- |
| **Fitting method** | **KS-statistic** | **CVM-Statistic** |
| Lognormal Bayesian | 0.13770 | 0.03538 |
| Lognormal MLE | 0.13892 | 0.03503 |
| Lognormal Scipy | 0.10549 | 0.04058 |
| Invgamma No Offset | 0.17431 | 0.10897 |
| Invgamma Offset | 0.55566 | 0.37515 |
| Invgamma Cutoff | 0.10861 | 0.13077 |
| Invgamma Scipy | 0.08481 | 0.02981 |
| Invgamma Product of Priors | 0.55564 | 0.37079 |
| Invgauss Offset | 0.16969 | 0.02910 |
| Invgauss Scipy | 0.10426 | 0.04541 |

# KS Test between edges

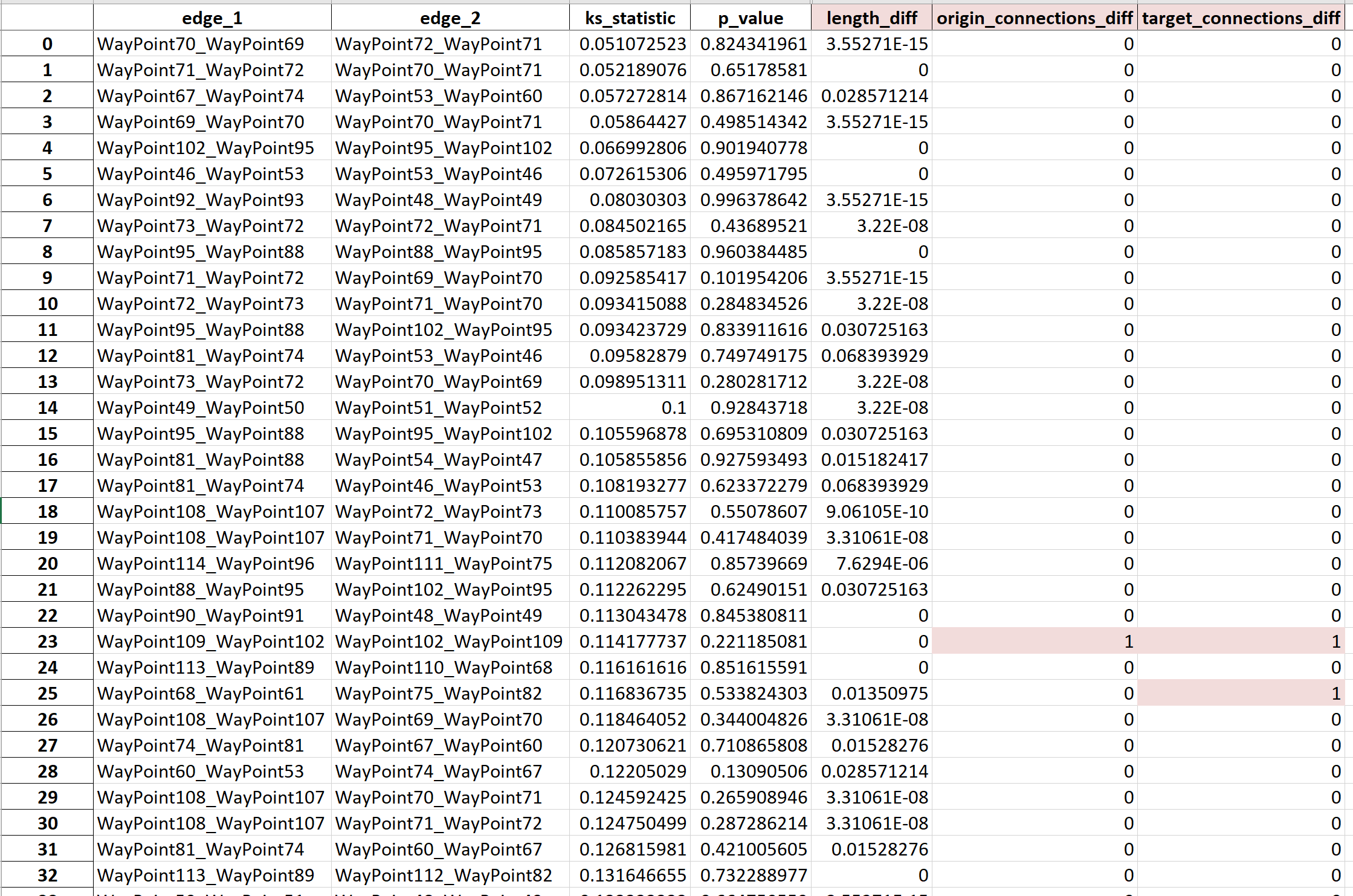
Sorted by lowest ks-stat (favours edges with high number of samples)



Adding context from the topological map (length of each edge, number of connections to each node):

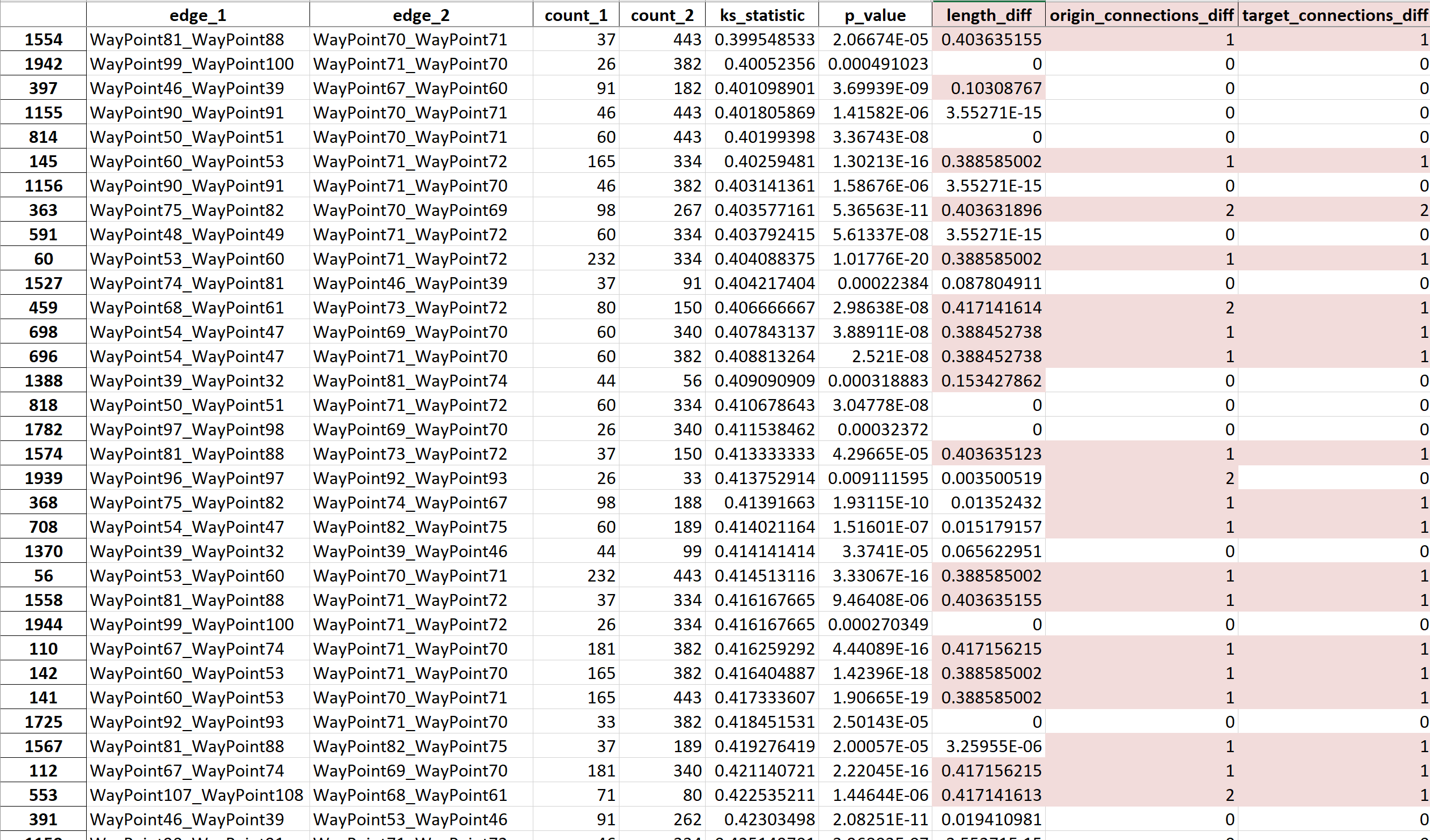
**High p\_value corresponds to length difference < 0.1m & no difference in number of connections to end nodes**

(see similar\_edges\_with\_context.xlsx)



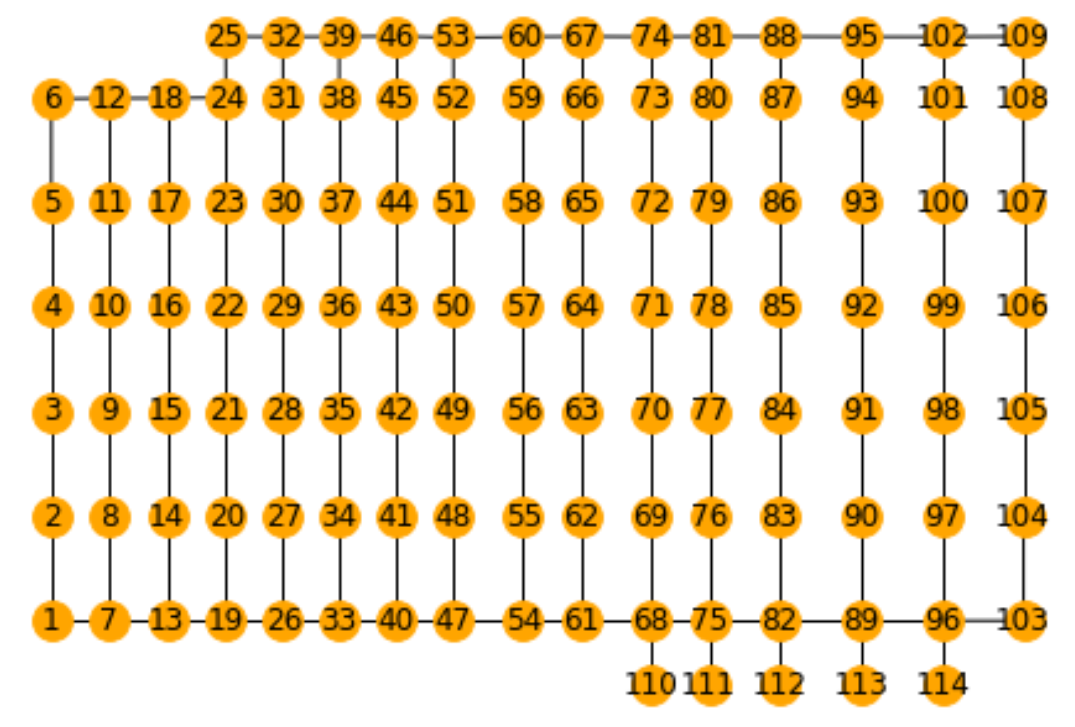
**Low p\_value corresponds to length differences > 0.1m & differences in number of connections to end nodes**

(see dissimilar\_edges\_with\_context.xlsx)



# Topological map

For Walmart map ("walmart\_map.yaml"):



For Bleinheim map ("blenheim\_map.yaml")

