This notebook explores some simple one - dimensional examples cases of the general procedure described in: https://arxiv.org/abs/2111.05442

First we create some models (our name for toy MCs) which either do or don' t generate data which is translation invariant on the torus. Histograms of data produced with these models will shortly be displayed.

In[45]:=
```
unifModel[] := RandomReal[]
```

In[46]:=
```
lowModel[] := RandomReal[]^1.3
```

In[48]:=
```
sineModel[] := Module[{x = RandomReal[], δ = 0.1},
   (While[RandomReal[] > (δ Cos[3 × 2 π x] + 1) / (1 + δ), x = RandomReal[]];
    x)
  ]
```

In[49]:=
```
genUnfilteredDatum[model_] := model[]
```

Create some filters (inefficiencies in the detector, etc)

In[50]:=
```
testFilter[x_] :=
 Which[x > 0.2 && x < 0.4, 0 / 10, x > 0.8 && x < 0.9, 2 / 10, True, 1]
```

In[51]:=
```
nullFilter[x_] := 1
```

In[52]:=
```
genFilteredDatum[model_, fil_] := Module[{x = genUnfilteredDatum[model]},
   (While[
     RandomReal[] > fil[x],
     x = genUnfilteredDatum[model]
    ]; x)
  ]
```

In[53]:=
```
(* Here is a function that randomly
   transforms an object in a way such that p(a→b) =
  p(b→a) and that would leave the uniform distibution on [0,1] invariant *)
wideTransform[x_] := Mod[x + RandomReal[], 1];
narrowTransform[x_] := Mod[x + RandomReal[{-1 / 4, 1 / 4}], 1];
veryNarrowTransform[x_] := Mod[x + RandomReal[{-1 / 100, 1 / 100}], 1];
```

Some infrastructure ...

In[56]:=
```
genData[n_, model_, fil_] := Table[genFilteredDatum[model, fil], {i, 1, n}]
```

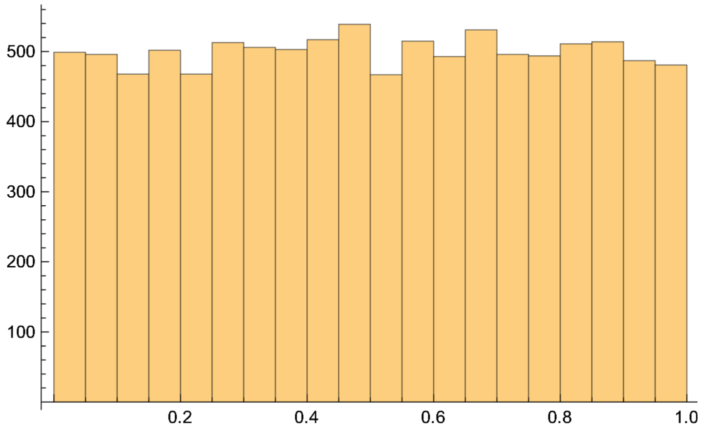
In[57]:=
```
genFilteredDatumPair[model_, fil_, tfm_] :=
 Module[{x = genFilteredDatum[model, fil], y},
  (y = tfm[x];
   While[
    RandomReal[] > fil[y],
    ( x = genFilteredDatum[model, fil] ; y = tfm[x])
   ]; {x, y})
 ]
```

In[58]:=
```
genFilteredDatumPairs[n_, model_, fil_, tfm_] :=
 Table[genFilteredDatumPair[model, fil, tfm], {i, 1, n}]
```

Finally: here is some data generated from the "uniform" model :

In[59]:=
```
Histogram[genData[10 000, unifModel, nullFilter]]
```

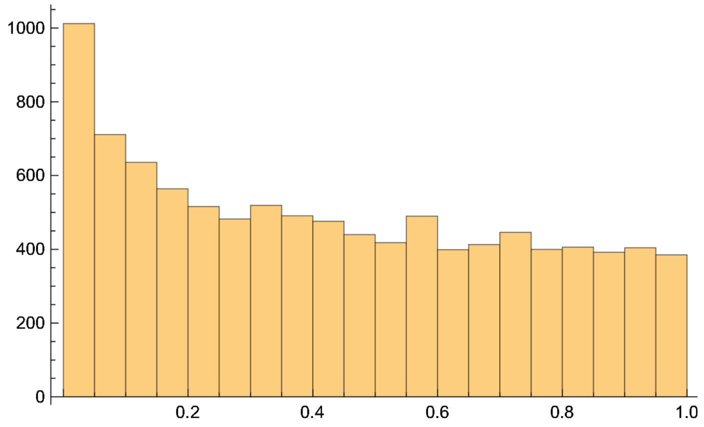
Out[59]=



Here is some data generated from a non-uniform model called the “lowModel” (because it is biased low) :

In[60]:=
```
Histogram[genData[10 000, lowModel, nullFilter]]
```

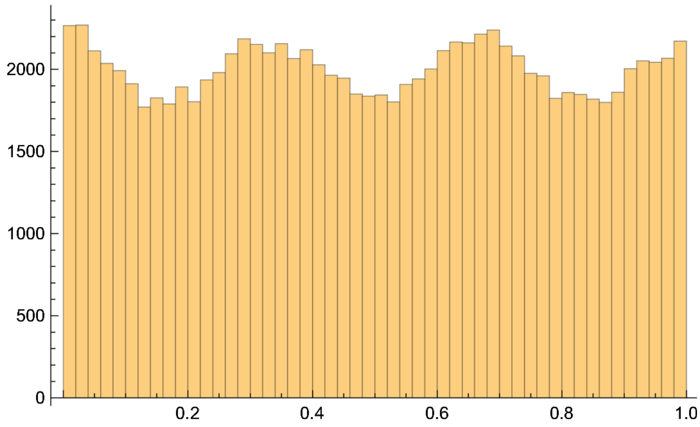
Out[60]=



Here is some data generated from a non-uniform model called the "sineModel" (because it has a sine or cos-like variation) :

In[61]:= `Histogram[genData[100 000, sineModel, nullFilter], 50]`

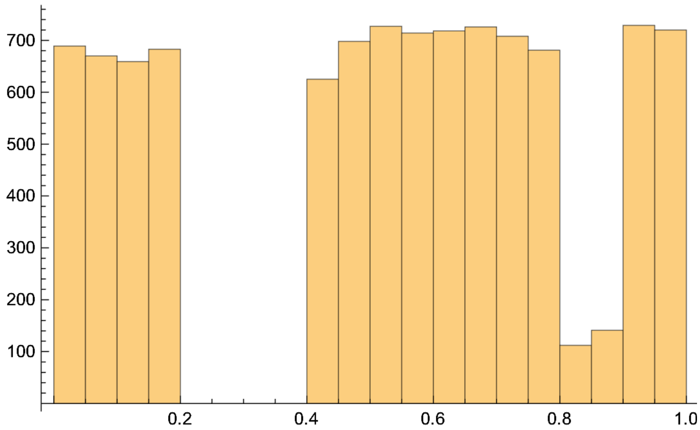
Out[61]=



Here is some data generated from the "uniform" model but taking our filter into account:

In[62]:= `Histogram[genData[10 000, unifModel, testFilter]]`

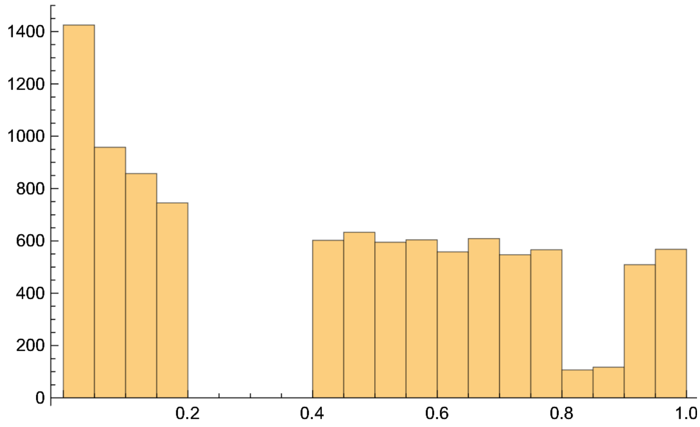
Out[62]=



Here is some data generated from the "lowModel” taking our filter into account:

In[63]:=
```
Histogram[genData[10 000, lowModel, testFilter]]
```

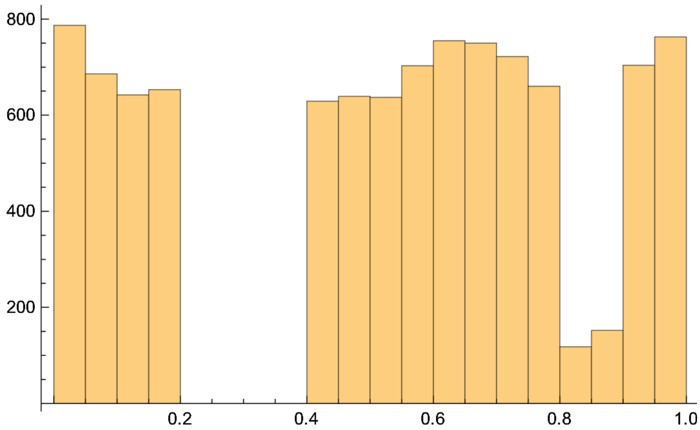
Out[63]=



Here is some data generated from the "sineModel” taking our filter into account:

In[64]:=
```
Histogram[genData[10 000, sineModel, testFilter]]
```

Out[64]=

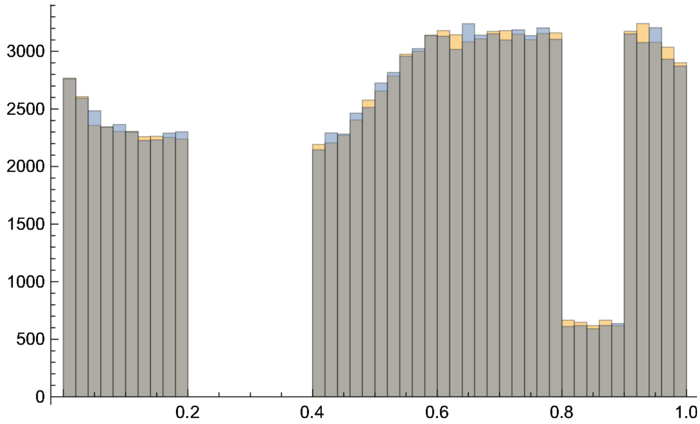


"filtered datum pairs" are what you get by generating a data point x, filtering it, then transforming x to y, then applying the filter to the transformed point y . Here we throw the entire pair (x,y) away if the transformed event “y” fails the filter . In a more nuanced approach the event - pair could be retained but weighted according to the filter probability. Note that “y” is called “ x’ ” in the paper.

Check that uniform model's filtered (x) and filtered-transformed-filtered (y) distributions are THE SAME AS EACH OTHER (which is the sign of symmetry preservation in the original) when we use (say) the narrow transform:

In[65]:=
```
Histogram[(genFilteredDatumPairs[100 000,
    unifModel, testFilter, narrowTransform] // Transpose), 50]
```

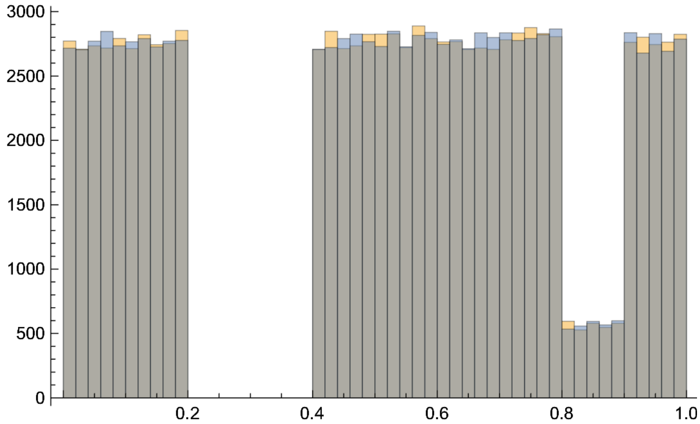
Out[65]=



Check that uniform model's x and y distributions are also THE SAME AS EACH OTHER (which is the sign of symmetry preservation in the original) when we use (say) the wide transform:

In[66]:=
```
Histogram[(genFilteredDatumPairs[100 000,
    unifModel, testFilter, wideTransform] // Transpose), 50]
```

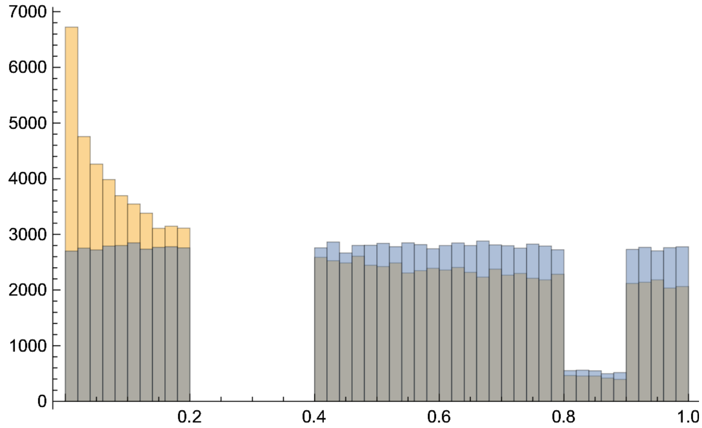
Out[66]=



Check that low model x and y distributions are DIFFERENT TO EACH OTHER (which is the sign of symmetry violation in the original) when we use (say) the wide transform:

In[67]:=
```
Histogram[(genFilteredDatumPairs[100 000,
    lowModel, testFilter, wideTransform] // Transpose), 50]
```

Out[67]=



Let' s take the (log of the) ratio of the yellow histogram above to the blue histogram above and call it "betterZetaForLowWide" . This is only something we are generating for illustrative purposes in this notebook. In reality, no histogram ratio needs to be taken. Instead a neural net (or similar) would learn a good zeta for any given problem. Here we are sidestepping those Net shenanigans by creating a function that makes good zetas for us from some histograms of the data:

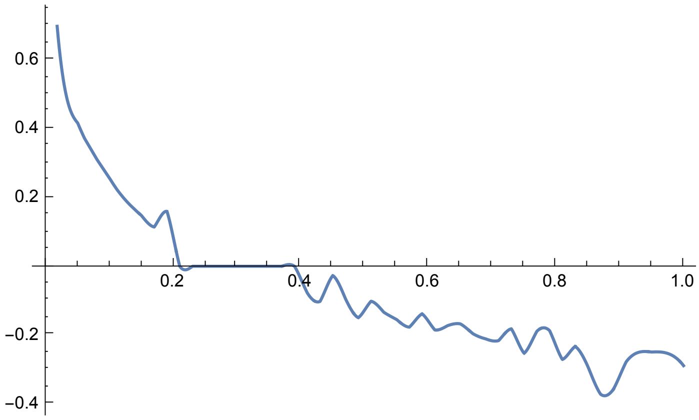
In[68]:=
```
(
 removeZeroData[x_] := {x⟦1⟧,  x⟦2⟧ /. {0 → 1}};

 extractGoodZeta[filteredDatumPairs_, bins_] :=
  Module[{n = Length[filteredDatumPairs], l1 =
     removeZeroData[HistogramList[(filteredDatumPairs // Transpose)⟦1⟧, bins]],
    l2 = removeZeroData[HistogramList[
       (filteredDatumPairs // Transpose)⟦2⟧, bins]]
   },
   Interpolation[ {
      Table[ ( l1⟦1⟧⟦i⟧ + l1⟦1⟧⟦i + 1⟧) / 2 , {i, 1, bins}],
       Log[l1⟦2⟧ / l2⟦2⟧]
     } // Transpose]
  ]
)
```

In[69]:=
```
betterZetaForLowWide = extractGoodZeta[
   genFilteredDatumPairs[100 000, lowModel, testFilter, wideTransform], 50];
Plot[betterZetaForLowWide[x], {x, 0, 1}]
```

InterpolatingFunction: Input value {0.0000204286} lies outside the range of data in the interpolating function. Extrapolation will be used.

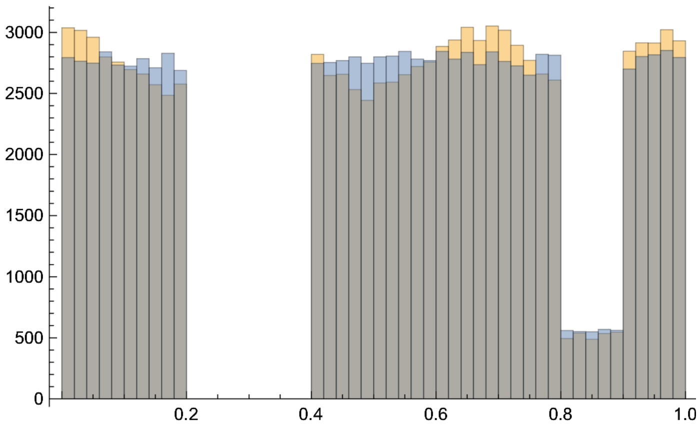
Out[69]=



Check that sine model x and x' distributions are DIFFERENT TO EACH OTHER (which is the sign of symmetry violation in the original) when we use (say) the wide transform:

In[70]:=
```
Histogram[(genFilteredDatumPairs[100 000,
     sineModel, testFilter, wideTransform] // Transpose), 50]
```

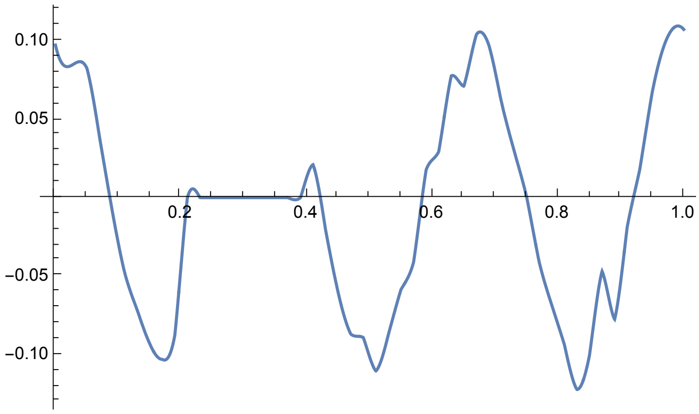
Out[70]=



In[71]:=
```
betterZetaForSineWide = extractGoodZeta[
   genFilteredDatumPairs[1 000 000, sineModel, testFilter, wideTransform], 50];
Plot[betterZetaForSineWide[x], {x, 0, 1}]
```

InterpolatingFunction: Input value {0.0000204286} lies outside the range of data in the interpolating function. Extrapolation will be used.

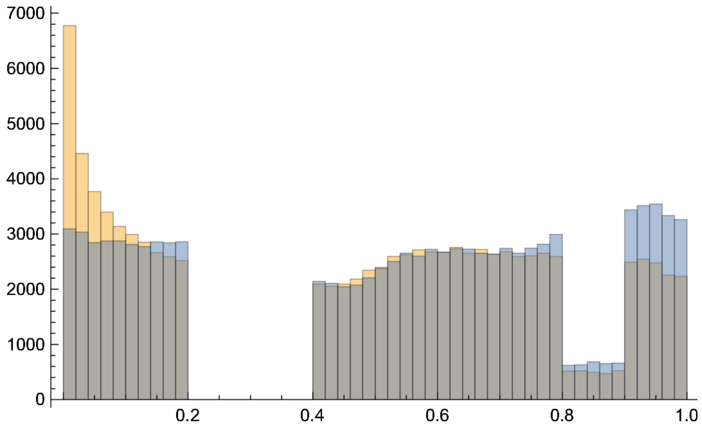
Out[71]=



Check that low model x and x’ distributions are DIFFERENT TO EACH OTHER (which is the sign of symmetry violation in the original) when we use (say) the narrow transform:

In[72]:=
```
Histogram[(genFilteredDatumPairs[100 000,
    lowModel, testFilter, narrowTransform] // Transpose), 50]
```

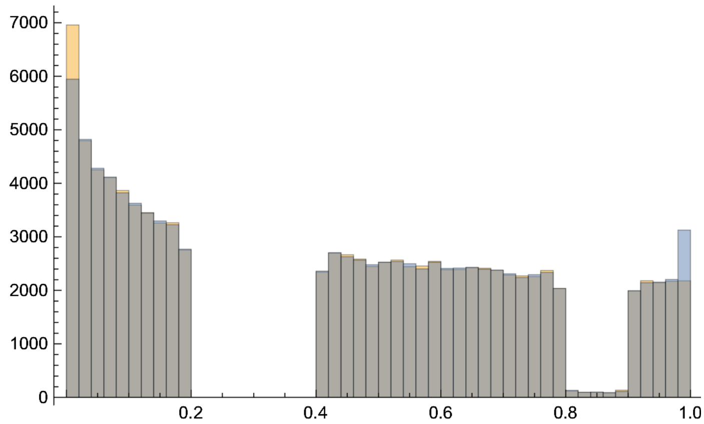
Out[72]=



Check that low model x and x' distributions are DIFFERENT TO EACH OTHER (which is the sign of symmetry violation in the original) when we use (say) the very narrow transform:

In[73]:=
```
Histogram[(genFilteredDatumPairs[100 000,
    lowModel, testFilter, veryNarrowTransform] // Transpose), 50]
```

Out[73]=



Could compare histograms, but that requires binning, etc . Better to work on the pairs themselves

In[86]:=
```
controlZeta[event_] := 1
```

In[74]:=
```
dumbZeta[event_] := If[event < 0.3, 1, -1]
 (* In reality one should get a neural net (or similar) to optimise Zeta
    to make it super-terrific at testing for symmetry violatioin. The
    example here is not trained at all but (by eye) has some hope ... *)
```

In[75]:=
```
whichIsRealForOneEventPair[eventPair_, zeta_] :=
 zeta[eventPair⟦1⟧] - zeta[eventPair⟦2⟧]
```

In[76]:=
```
whichIsRealForEventPairs[eventPairs_, zeta_] := Table[
  whichIsRealForOneEventPair[eventPairs⟦i⟧, zeta], {i, 1, Length[eventPairs]}]
```

In[77]:=
```
stats[x_] := Module[{m = Mean[x], s = StandardDeviation[x], n = Length[x]},
  {
   {"Standard Deviation", s},
   {"Mean", m},
   {"Mean uncertainty", s / Sqrt[n]},
   {"Sigmas mean is from zero", m / (s / Sqrt[n])}
  }
 ]
```

Check mean of zeta is NOT significantly different from zero when underlying model is UNIFORM:

In[78]:=
```
whichIsRealForEventPairs[genFilteredDatumPairs[100 000, unifModel,
    testFilter, wideTransform], dumbZeta] // stats // N // TableForm
```

Out[78]//TableForm=

Standard Deviation	1.26919
Mean	-0.00306
Mean uncertainty	0.00401354
Sigmas mean is from zero	-0.76242

Repeating same check for a different zeta is NOT significantly different from zero when underlying model is UNIFORM:

In[79]:=
```
whichIsRealForEventPairs[genFilteredDatumPairs[100 000, unifModel, testFilter,
    wideTransform], betterZetaForLowWide] // stats // N // TableForm
```

InterpolatingFunction: Input value {0.00324314} lies outside the range of data in the interpolating function. Extrapolation will be used.

InterpolatingFunction: Input value {0.00971032} lies outside the range of data in the interpolating function. Extrapolation will be used.

InterpolatingFunction: Input value {0.00537926} lies outside the range of data in the interpolating function. Extrapolation will be used.

General: Further output of InterpolatingFunction::dmval will be suppressed during this calculation.

Out[79]//TableForm=

Standard Deviation	0.377376
Mean	-0.00100147
Mean uncertainty	0.00119337
Sigmas mean is from zero	-0.839193

Check mean of zeta IS significantly different from zero when underlying model is NON-UNIFORM : (Note that as zeta is not properly trained, the real evidence for symm violation should be even greater)

In[80]:=
```
whichIsRealForEventPairs[genFilteredDatumPairs[100 000, lowModel, testFilter,
    wideTransform], betterZetaForLowWide] // stats // N // TableForm
```

InterpolatingFunction: Input value {0.00739024} lies outside the range of data in the interpolating function. Extrapolation will be used.

InterpolatingFunction: Input value {0.997438} lies outside the range of data in the interpolating function. Extrapolation will be used.

InterpolatingFunction: Input value {0.00492533} lies outside the range of data in the interpolating function. Extrapolation will be used.

General: Further output of InterpolatingFunction::dmval will be suppressed during this calculation.

Out[80]//TableForm=

Standard Deviation	0.433209
Mean	0.0899533
Mean uncertainty	0.00136993
Sigmas mean is from zero	65.6627

Check mean of zeta IS significantly different from zero when underlying model is NON-UNIFORM : (Note that as testZeta is not properly trained -- zeta here is cruder than previous guessr)

In[81]:=
```
whichIsRealForEventPairs[genFilteredDatumPairs[100 000, lowModel,
    testFilter, wideTransform], dumbZeta] // stats // N // TableForm
```

Out[81]//TableForm=

Standard Deviation	1.32621
Mean	0.24662
Mean uncertainty	0.00419385
Sigmas mean is from zero	58.8052

Similar check to above, but using the narrow transform rather than the wide transform. Note that the evidence for symmetry violation is smaller:

In[82]:=
```
whichIsRealForEventPairs[genFilteredDatumPairs[100 000, lowModel,
    testFilter, narrowTransform], dumbZeta] // stats // N // TableForm
```

Out[82]//TableForm=

Standard Deviation	0.886931
Mean	0.1241
Mean uncertainty	0.00280472
Sigmas mean is from zero	44.2468

And for a VERY narrow transform the evidence is smaller still ....

In[83]:=
```
whichIsRealForEventPairs[genFilteredDatumPairs[100 000, lowModel,
    testFilter, veryNarrowTransform], dumbZeta] // stats // N // TableForm
```

Out[83]//TableForm=

Standard Deviation	0.235164
Mean	0.01728
Mean uncertainty	0.000743653
Sigmas mean is from zero	23.2366

... so wide transforms are good .

Can we see non - uniformity in the Sine model?

Control: should see nothing at all using the control zeta:

In[87]:=
```
whichIsRealForEventPairs[genFilteredDatumPairs[100 000, sineModel,
    testFilter, wideTransform], controlZeta] // stats // N // TableForm
```

Power: Infinite expression $\frac{1}{0}$ encountered.

Infinity: Indeterminate expression 0 ComplexInfinity encountered.

Out[87]//TableForm=

Standard Deviation	0.
Mean	0.
Mean uncertainty	0.
Sigmas mean is from zero	Indeterminate

Evidence is very weak using the dumb zeta: (this is expected)

In[84]:=
```
whichIsRealForEventPairs[genFilteredDatumPairs[100 000, sineModel,
    testFilter, wideTransform], dumbZeta] // stats // N // TableForm
```

Out[84]//TableForm=

Standard Deviation	1.26287
Mean	-0.00772
Mean uncertainty	0.00399354
Sigmas mean is from zero	-1.93312

But (as expected) evidence is much stronger using a better trained zeta (albeit here created by cheat rather than by Neural Net or actual training process):

In[85]:=
```
whichIsRealForEventPairs[genFilteredDatumPairs[100 000, sineModel, testFilter,
    wideTransform], betterZetaForSineWide] // stats // N // TableForm
```

InterpolatingFunction: Input value {0.994032} lies outside the range of data in the interpolating function. Extrapolation will be used.

InterpolatingFunction: Input value {0.992071} lies outside the range of data in the interpolating function. Extrapolation will be used.

InterpolatingFunction: Input value {0.990863} lies outside the range of data in the interpolating function. Extrapolation will be used.

General: Further output of InterpolatingFunction::dmval will be suppressed during this calculation.

Out[85]//TableForm=

Standard Deviation	0.100113
Mean	0.00470969
Mean uncertainty	0.000316586
Sigmas mean is from zero	14.8765