TODO:

* A lot of the Add @@ Table commands don't seem to exploit multithreading.
* I need to test the operator norm. Relatedly, one could consider using a different measure of the "slope" of a matrix.
* Currently finding an optimal report usually works, but sometimes fails somewhat spectacularly.
* Plot the bounds
* Are the Scatter plots really plotting the data as intended?
* TODO: the range of the axes of the scatter plots are currently determined by the plot of the trendline, which I think means that some points my not be visible. Make sure that all points are visible!

Definitions

In[54]:=
```
IsDistr[p_] := Total[p] == 1 && And @@ NonNegative[p]
```

In[55]:=
```
IsDistrAppr[p_] := Total[p] == 1 && And @@ Table[p[[i]] ≥ -0.00001, {i, 1, Length[p]}]
```

In[56]:=
```
BrierScore[p_, q_] := Sum[2 * q[[i]] * p[[i]], {i, 1, Length[q]}] - Sum[p[[i]]^2, {i, 1, Length[q]}]
```

In[57]:=
```
logscoreh[0, 0] := 0; logscoreh[0, 0.0] = 0; logscoreh[0, x_] := -∞;
logscoreh[0.0, 0.0] := 0 ; logscoreh[0.0, 0]; logscoreh[0.0, x_] := -∞;
logscoreh[pi_, qi_] := qi * Log[pi];
```

In[60]:=
```
LogScore[p_, q_] := Sum[logscoreh[p[[i]], q[[i]]], {i, 1, Length[q]}]
```

In[61]:=
```
uniform[dim_] := Table[1 / dim, dim]
```

In[62]:=
```
RandomA[dim_] := Transpose[Table[Normalize[RandomInteger[{0, 100}, dim], Total], dim]];
```

In[63]:=
```
RandomDistr[dim_] := Normalize[RandomInteger[{0, 100}, dim], Total]
```

In[64]:=
```
FixedPointDistr[A_] := Last /@
  Last@Solve[A.Array[x, Length[A]] == Array[x, Length[A]] && IsDistr[Array[x, Length[A]]],
    Array[x, Length[A]]]
```

In[65]:=
```
Clear[pmax];
Clear[S];
Clear[A];
OptReport[S_, A_] := Last /@ Last@NMaximize[
    {S[Array[pmax, Length[A]], A.Array[pmax, Length[A]]], IsDistr[Array[pmax, Length[A]]]},
    Array[pmax, Length[A]], Method → {"RandomSearch", "SearchPoints" → 100}]
```

For the Brier scoring rule, we can actually solve for the optimal report exactly using Maximize:

In[66]:=
```
OptReportBrier[A_] :=
 Last /@ Last@Maximize[{BrierScore[Array[pmax, Length[A]], A.Array[pmax, Length[A]]],
     IsDistr[Array[pmax, Length[A]]]}, Array[pmax, Length[A]]]
```

The following definition of the operator norm is from https://mathworld.wolfram.com/OperatorNorm.html.

In[67]:=
```
OperatorNorm[a_List?MatrixQ] := Sqrt[Max[Eigenvalues[Transpose[a].a]]]
```

In[68]:=
```
TangentSpaceOpNorm[A_] := First@N@Maximize[{Norm[A.Array[v, Length[A]]],
      Total[Array[v, Length[A]]] == 0 && Norm[Array[v, Length[A]]] ≤ 1}, Array[v, Length[A]]]
```

In[69]:=
```
TangentSpaceOpNormN[A_] := First@NMaximize[{Norm[A.Array[v, Length[A]]],
     Total[Array[v, Length[A]]] == 0 && Norm[Array[v, Length[A]]] ≤ 1}, Array[v, Length[A]]]
```

For our tests we also use the following function to generate constant functions.

In[70]:=
```
ConstantA[q_] := Table[Table[q[[i]], Length[q]], {i, 1, Length[q]}]
```

In[71]:=
```
AccuracyBoundBrier[A_] := TangentSpaceOpNorm[A]
```

Tests

In[•]:=
```
testdim = 5;
```

Operator norm

Example from https://math.stackexchange.com/questions/2670350/operator-norm-calculation-for-simple-matrix.

In[•]:=
```
OperatorNorm[{{1, 4}, {5, 6}}] == Sqrt[39 + 5 * Sqrt[53]]
```

Out[•]=
```
True
```

Example from https://mathworld.wolfram.com/OperatorNorm.html.

In[•]:=
```
OperatorNorm[{{2, 0, 0}, {3, 0, 2}}] == 4
```

Out[•]=
```
True
```

In[•]:=
```
FullSimplify[OperatorNorm[{{a}}] == Abs[a], Element[a, Reals]]
```

Out[•]=
```
True
```

In[•]:=
```
TangentSpaceOpNorm[IdentityMatrix[3]] == 1
```

Out[•]=
```
True
```

In[•]:=
```
Timing[And @@ Table[A = RandomA[2];
    Abs[TangentSpaceOpNorm[A] - TangentSpaceOpNormN[A]] < 0.00001, {i, 1, 600}]]
```

Out[•]=
```
{56.2362, True}
```

In[•]:=
```
Timing[And @@ Table[A = RandomA[3];
    Abs[TangentSpaceOpNorm[A] - TangentSpaceOpNormN[A]] < 0.00001, {i, 1, 35}]]
```

Out[•]=
```
{56.8949, True}
```

In[•]:=
```
Timing[And @@ Table[A = RandomA[4];
    Abs[TangentSpaceOpNorm[A] - TangentSpaceOpNormN[A]] < 0.00001, {i, 1, 25}]]
```

Out[•]=
```
{71.6548, True}
```

Probability distributions

```
In[•]:= IsDistr[uniform[testdim]]
Out[•]= True

In[•]:= And @@ Table[IsDistr[RandomDistr[testdim]], 100]
Out[•]= True

In[•]:= And @@ Table[IsDistr[RandomA[testdim].RandomDistr[testdim]], 1000]
Out[•]= True
```

The following lets Mathematica show symbolically that constantA works as intended, i.e., that constantA[q].p=q for any distributions q and p.

```
In[•]:= Clear[q];
        FullSimplify[ConstantA[Array[q, testdim]].Array[p, testdim],
          IsDistr[Array[p, testdim]]] == Array[q, testdim]
Out[•]= True
```

Scoring rules

```
In[•]:= -0.23 ≤ LogScore[{0.8, 0.2}, {1, 0}] ≤ -0.21
Out[•]= True
```

Log scoring rule degenerate cases

TODO

```
In[•]:= Simplify[logscoreh[0, x], x > 0] == -∞
Out[•]= True

In[•]:= logscoreh[1, 0] == 0
Out[•]= True

In[•]:= logscoreh[0, 0] == 0
Out[•]= True

In[•]:= LogScore[{1, 0, 0}, {1, 0, 0}] == 0
Out[•]= True

In[•]:= LogScore[{1/2, 1/2, 0}, uniform[3]] == -∞
Out[•]= True
```

Propriety

We here test that scoring rules we defined are indeed proper. We do this specifically by making sure that if the environment probability is a fixed distribution q, then the optimal report is q.

LogScore:

```
In[•]:= Clear[p];
        N[Last /@ Last@Maximize[LogScore[Array[p, testdim], uniform[testdim]],
              IsDistr[Array[p, testdim]], Array[p, testdim]]] == uniform[testdim]
Out[•]= True

In[•]:= Clear[p];
        Timing[Table[q = RandomDistr[testdim];
          N[Last /@ Last@Maximize[LogScore[Array[p, testdim], q],
                IsDistr[Array[p, testdim]], Array[p, testdim]]] == q, 1000]]
Out[•]= $Aborted

In[•]:= Table[q = RandomDistr[testdim];
         N[Last /@ Last@Maximize[LogScore[Array[p, testdim], q],
               IsDistr[Array[p, testdim]], Array[p, testdim]]] == q, 100]
Out[•]= {True, True, True, True, True, True, True, True, True, True, True, True, True,
         True, True, True, True, True, True, True, True, True, True, True, True, True,
         True, True, True, True, True, True, True, True, True, True, True, True, True,
         True, True, True, True, True, True, True, True, True, True, True, True, True,
         True, True, True, True, True, True, True, True, True, True, True, True, True,
         True, True, True, True, True, True, True, True, True, True, True, True, True,
         True, True, True, True, True, True, True, True, True, True, True, True, True}
```

BrierScore:

```
In[•]:= Clear[p];
        N[Last /@ Last@Maximize[BrierScore[Array[p, testdim], uniform[testdim]],
              IsDistr[Array[p, testdim]], Array[p, testdim]]] == uniform[testdim]
Out[•]= True

In[•]:= Timing[And @@ Table[q = RandomDistr[testdim];
            N[Last /@ Last@Maximize[BrierScore[Array[p, testdim], q],
                  IsDistr[Array[p, testdim]], Array[p, testdim]]] == q, 100]]
Out[•]= {22.4564, True}
```

Finding fixed points

```
In[•]:= Timing[And @@ Table[q = RandomDistr[testdim]; FixedPointDistr[ConstantA[q]] == q, 1000]]
Out[•]= {7.90514, True}

In[•]:= Timing[And @@ Table[IsDistr[FixedPointDistr[RandomA[testdim]]], 1000]]
Out[•]= {7.1437, True}
```

In[•]:=
```
Timing[And @@ Table[A = RandomA[testdim]; p = FixedPointDistr[A]; A.p == p, 1000]]
```

Out[•]=
```
{7.4467, True}
```

Optimal report function

Optimal reports should be probability distributions.

LogScore:

In[•]:=
```
Timing[And @@ Table[IsDistr[OptReport[LogScore, RandomA[testdim]]], 30]]
```

Out[•]=
```
{271.779, True}
```

BrierScore:

In[•]:=
```
Timing[And @@ Table[IsDistrAppr[OptReport[BrierScore, RandomA[testdim]]], 30]]
```

Out[•]=
```
{284.868, True}
```

In[•]:=
```
Timing[And @@ Table[IsDistr[OptReportBrier[RandomA[testdim]]], 5]]
```

Out[•]=
```
{40.7348, True}
```

On randomly generated A, I test whether the “optimal report” is at least as good as the fixed point.

LogScore:

In[•]:=
```
Timing[And @@ Table[A = RandomA[testdim];
    p = OptReport[LogScore, A];
    fp = FixedPointDistr[A];
    LogScore[p, A.p] ≥ LogScore[fp, fp], 30]]
```

Out[•]=
```
{299.523, True}
```

BrierScore:

In[•]:=
```
Timing[And @@ Table[A = RandomA[testdim];
    p = OptReport[BrierScore, A];
    fp = FixedPointDistr[A];
    BrierScore[p, A.p] ≥ BrierScore[fp, fp], 20]]
```

Out[•]=
```
{223.589, True}
```

In[•]:=
```
Timing[And @@ Table[A = RandomA[testdim];
    p = OptReportBrier[A];
    fp = FixedPointDistr[A];
    BrierScore[p, A.p] ≥ BrierScore[fp, fp], 10]]
```

Out[•]=
```
{80.3928, True}
```

On randomly generated A, I test whether the “optimal report” (as found by OptReport) is at least as good as a bunch of randomly generated reports.

LogScore:

In[•]:=
```
Timing[And @@ Table[A = RandomA[testdim]; p = OptReport[LogScore, A];
    And @@ Table[randp = RandomDistr[testdim];
      LogScore[p, A.p] ≥ LogScore[randp, A.randp], 10 000], 10]]
```

Out[•]=
```
{122.265, True}
```

Currently it appears that the above occasionally fails. Basically it seems that for at least some A, OptReport[LogScore,A] occasionally fails to find even a decent solution. I'm not sure why that is.

BrierScore:

In[•]:=
```
Timing[And @@ Table[A = RandomA[testdim]; p = OptReport[BrierScore, A];
    And @@ Table[randp = RandomDistr[testdim];
      BrierScore[p, A.p] ≥ BrierScore[randp, A.randp], 10 000], 10]]
```

Out[•]=
```
{120.944, True}
```

In[•]:=
```
Timing[And @@ Table[A = RandomA[testdim]; p = OptReportBrier[A];
    And @@ Table[randp = RandomDistr[testdim];
      BrierScore[p, A.p] ≥ BrierScore[randp, A.randp], 10 000], 5]]
```

Out[•]=
```
{51.2441, True}
```

If A is simply the matrix that maps each probability distribution onto q, then the optimal report should be approximately q.

LogScore:

In[•]:=
```
Timing[And @@ Table[q = RandomDistr[testdim];
    p = OptReport[LogScore, ConstantA[q]];
    Norm[p - q] ≤ 0.00001, 20]]
```

Experimental`NumericalFunction: The function value Indeterminate is not a number at {pmax[2], pmax[3], pmax[4], pmax[5]} = {0.185578, 0.505298, 0., 0.106244}.

Experimental`NumericalFunction: The function value Indeterminate is not a number at {pmax[2], pmax[3], pmax[4], pmax[5]} = {0.185578, 0.505298, 0., 0.106244}.

Experimental`NumericalFunction: The function value Indeterminate is not a number at {pmax[2], pmax[3], pmax[4], pmax[5]} = {0.0779029, 0., 0.117862, 0.664668}.

General: Further output of Experimental`NumericalFunction::nnum will be suppressed during this calculation.

Out[•]=
```
{187.092, True}
```

BrierScore:

In[•]:=
```
Timing[And @@ Table[q = RandomDistr[testdim];
    p = OptReport[BrierScore, ConstantA[q]];
    Norm[p - q] ≤ 0.00001, 20]]
```

Out[•]=
```
{184.344, True}
```

In[•]:=
```
Timing[And @@ Table[q = RandomDistr[testdim];
    p = OptReportBrier[ConstantA[q]];
    p == q, 100]]
```

Out[•]= {33.0995, True}

We can now test the NMaximized-based approximate best report function against the Maximize-based exact best report function for the Brier scoring rule.

In[•]:=
```
Timing[And @@ Table[A = RandomA[testdim];
    p1 = OptReportBrier[A];
    p2 = OptReport[BrierScore, A];
    Norm[p1 - p2] ≤ 0.001, 5]]
```

Out[•]= {88.8496, True}

Experiment

In[135]:=
```
n = 200;
```

In[136]:=
```
S = BrierScore;
```

%Unfortunately, at dimensions higher than 4, calculating the tangent space operator norm _exactly_ seems to currently cause crashes. So we use dim=4.

In[137]:=
```
dim = 5;
```

We sample n matrices A. For each of them we record four numbers: 1) the operator norm of A; 2) how far the fixed point distribution of A is from the uniform distribution; 3) how far the optimal report is from the fixed point; 4) how far the optimal report is from the true distribution when the optimal report is made; 5) how far the actual report is from the uniform distribution.

In[138]:=
```
Timing[data = Table[A = RandomA[dim];
    fixedpoint = FixedPointDistr[A];
    optrep = OptReportBrier[A];
    {N@TangentSpaceOpNormN[A], N@Norm[fixedpoint - uniform[dim]], N@Norm[
       optrep - fixedpoint], N@Norm[optrep - A.optrep], N@Norm[optrep - uniform[dim]]}, n]]
```

Out[138]=
```
{1711.68, {{0.542223, 0.129581, 0.0275126, 0.0252574, 0.12897},
  {0.342585, 0.0903047, 0.0164769, 0.0160977, 0.088455},
  {0.45257, 0.121087, 0.0379322, 0.0343038, 0.125734},
  {0.342338, 0.0688611, 0.0163879, 0.017432, 0.0686312},
  {0.43701, 0.0428935, 0.021652, 0.0169124, 0.060716},
  {0.422679, 0.0861956, 0.0386408, 0.0364599, 0.104971},
  {0.460777, 0.117808, 0.164479, 0.10443, 0.24507},
  {0.334905, 0.0983142, 0.0203055, 0.019206, 0.111027},
  {0.455463, 0.133668, 0.114574, 0.0838968, 0.21263},
  {0.398079, 0.124268, 0.150554, 0.0932058, 0.251797},
  {0.396924, 0.0826371, 0.0188924, 0.0190567, 0.0826978},
```

```
{0.419295, 0.165369, 0.0771623, 0.0569426, 0.172676},
{0.453406, 0.111597, 0.0727396, 0.0568762, 0.147064},
{0.335379, 0.108166, 0.0340778, 0.0282938, 0.126861},
{0.413562, 0.0712348, 0.0107279, 0.0136811, 0.0728719},
{0.335803, 0.107958, 0.0288268, 0.0325898, 0.101427},
{0.335648, 0.0770789, 0.0216432, 0.0202132, 0.0905293},
{0.345109, 0.124789, 0.0350942, 0.0349805, 0.135527},
{0.362414, 0.163266, 0.123949, 0.0867666, 0.274251},
{0.414087, 0.0645294, 0.0137217, 0.0173231, 0.0635487},
{0.336465, 0.10569, 0.018068, 0.0155822, 0.109201},
{0.286383, 0.155948, 0.0242866, 0.0279222, 0.146553},
{0.340697, 0.130437, 0.0153087, 0.0138948, 0.144805},
{0.368262, 0.22006, 0.147252, 0.107107, 0.362933},
{0.420712, 0.0948249, 0.0393527, 0.0348376, 0.131285},
{0.311599, 0.110765, 0.0450161, 0.0346417, 0.153629},
{0.439349, 0.164375, 0.0880296, 0.0657706, 0.233728},
{0.620919, 0.187277, 0.0712109, 0.0587207, 0.242322},
{0.476118, 0.0725804, 0.0270204, 0.0219645, 0.0928426},
{0.373645, 0.0798621, 0.0275429, 0.0211027, 0.0920013},
{0.274961, 0.110223, 0.0212041, 0.0238741, 0.10587},
{0.351014, 0.118889, 0.024256, 0.0240502, 0.128302},
{0.368399, 0.158777, 0.0809164, 0.0616631, 0.230622},
{0.419171, 0.0752284, 0.0343958, 0.0297114, 0.0971063},
{0.355385, 0.0502165, 0.0194874, 0.0145219, 0.0625949},
{0.380193, 0.0425954, 0.0122252, 0.0113627, 0.0487471},
{0.318669, 0.105699, 0.0512125, 0.0386909, 0.149994},
{0.281796, 0.0785347, 0.0133792, 0.0123129, 0.0881013},
{0.255869, 0.0820938, 0.0125807, 0.013749, 0.0742814},
{0.465396, 0.159523, 0.0667834, 0.0608947, 0.173421},
{0.389247, 0.152744, 0.0263337, 0.0298841, 0.160798},
{0.553487, 0.0906866, 0.0333096, 0.0382141, 0.100915},
{0.422482, 0.117769, 0.0179797, 0.0213675, 0.12859},
{0.222316, 0.105294, 0.0182062, 0.0185408, 0.106377},
{0.336923, 0.168525, 0.0697456, 0.0597101, 0.225852},
{0.412762, 0.148828, 0.0463062, 0.0473207, 0.176626},
{0.349599, 0.154397, 0.048502, 0.0395867, 0.179878},
{0.479533, 0.229039, 0.0790707, 0.0765232, 0.285592},
{0.383218, 0.128125, 0.0826327, 0.0657469, 0.182519},
{0.463752, 0.180387, 0.110738, 0.0736804, 0.272016},
{0.231172, 0.143794, 0.0169523, 0.0181613, 0.133347},
{0.274789, 0.067949, 0.00757917, 0.00784818, 0.0672308},
{0.290117, 0.0996232, 0.0457309, 0.0374534, 0.135112},
{0.431428, 0.0947804, 0.011794, 0.0123646, 0.100331},
```

```
{0.414241, 0.180104, 0.0691008, 0.0557357, 0.240324},
{0.376507, 0.0703642, 0.0112729, 0.0123922, 0.072329},
{0.438182, 0.275862, 0.249638, 0.169918, 0.525028},
{0.348001, 0.08673, 0.0119495, 0.0109728, 0.0924896},
{0.461255, 0.148465, 0.0679982, 0.0467611, 0.165576},
{0.353523, 0.132877, 0.0654502, 0.051616, 0.175831},
{0.32368, 0.0681992, 0.0157819, 0.0166444, 0.0607758},
{0.446777, 0.0697794, 0.0144862, 0.0127217, 0.0710682},
{0.346325, 0.25111, 0.077193, 0.0626305, 0.311616},
{0.358933, 0.155748, 0.0382566, 0.0308707, 0.179027},
{0.429013, 0.112999, 0.0498328, 0.0488559, 0.119179},
{0.339513, 0.0946096, 0.0218966, 0.0223089, 0.0906285},
{0.370377, 0.0453207, 0.013214, 0.013604, 0.0473691},
{0.344445, 0.159248, 0.0322027, 0.0302457, 0.186116},
{0.379, 0.153862, 0.114452, 0.0805511, 0.235986},
{0.323015, 0.0839437, 0.0299273, 0.0247243, 0.105833},
{0.412122, 0.0918897, 0.0615961, 0.0456186, 0.148014},
{0.423039, 0.150537, 0.0538373, 0.0464008, 0.197567},
{0.363289, 0.0544484, 0.0167708, 0.0163724, 0.0667026},
{0.494693, 0.0877932, 0.0175984, 0.0184073, 0.0847332},
{0.443578, 0.0743329, 0.0465899, 0.0411428, 0.10344},
{0.50031, 0.118868, 0.0486876, 0.0411876, 0.115062},
{0.46153, 0.1145, 0.0790471, 0.0589309, 0.174083},
{0.30136, 0.157355, 0.0446984, 0.0365902, 0.197849},
{0.416418, 0.147917, 0.0748339, 0.0551255, 0.220392},
{0.407157, 0.124266, 0.119582, 0.0810946, 0.222833},
{0.477216, 0.145512, 0.150127, 0.0951366, 0.277039},
{0.453576, 0.147867, 0.240343, 0.140719, 0.380704},
{0.363452, 0.136395, 0.0259191, 0.024111, 0.154254},
{0.438629, 0.0536381, 0.0152462, 0.0162153, 0.0588393},
{0.297935, 0.037983, 0.00387523, 0.00372018, 0.0386509},
{0.247549, 0.131477, 0.031246, 0.029794, 0.137743},
{0.391448, 0.0819965, 0.00994053, 0.0112702, 0.0815834},
{0.493106, 0.133337, 0.0472715, 0.0418087, 0.16647},
{0.473662, 0.123494, 0.0301207, 0.0405892, 0.1088},
{0.389226, 0.172805, 0.235836, 0.149272, 0.391619},
{0.300476, 0.0810373, 0.0161837, 0.0169766, 0.0742004},
{0.289626, 0.0613194, 0.0191844, 0.0168091, 0.0643442},
{0.315435, 0.0393047, 0.0164794, 0.0144171, 0.0459524},
{0.440321, 0.089378, 0.0268062, 0.0305228, 0.0834741},
{0.339493, 0.100682, 0.0302847, 0.0270732, 0.124808},
{0.432083, 0.0897978, 0.0377049, 0.0357285, 0.104282},
{0.297058, 0.1748, 0.0490285, 0.0445852, 0.202307},
```

{0.376222, 0.0675185, 0.0207708, 0.0205001, 0.0756285},
{0.273075, 0.10453, 0.00830945, 0.0099718, 0.101325},
{0.396659, 0.130045, 0.0159787, 0.016699, 0.125453},
{0.408842, 0.104601, 0.0744141, 0.0546682, 0.173371},
{0.451721, 0.139316, 0.166951, 0.0971312, 0.276387},
{0.53614, 0.16313, 0.0971028, 0.082415, 0.236511},
{0.423513, 0.111049, 0.0440778, 0.0419592, 0.138836},
{0.316878, 0.0471096, 0.00476236, 0.00410697, 0.0491257},
{0.389682, 0.0450493, 0.00997289, 0.0118592, 0.0406561},
{0.398739, 0.108359, 0.0352327, 0.0394434, 0.113477},
{0.367873, 0.0806419, 0.02871, 0.0253032, 0.104583},
{0.438822, 0.173307, 0.0296073, 0.0357505, 0.176086},
{0.461609, 0.0845305, 0.049418, 0.040312, 0.116896},
{0.37642, 0.13725, 0.0352307, 0.0283492, 0.158127},
{0.54372, 0.136713, 0.362856, 0.226547, 0.495587},
{0.393603, 0.103848, 0.031332, 0.0239155, 0.109626},
{0.329271, 0.0713524, 0.0269171, 0.0208737, 0.0897809},
{0.335579, 0.0844702, 0.0522819, 0.0376839, 0.131543},
{0.392922, 0.106016, 0.0666728, 0.0470607, 0.160384},
{0.346513, 0.1247, 0.0371359, 0.0293105, 0.149816},
{0.457799, 0.0890652, 0.256482, 0.144278, 0.324603},
{0.422422, 0.146927, 0.0505913, 0.0487807, 0.188212},
{0.341183, 0.0421615, 0.00818256, 0.00890504, 0.038303},
{0.477777, 0.116006, 0.0439537, 0.0487792, 0.102882},
{0.440316, 0.115053, 0.0293374, 0.033437, 0.116894},
{0.319541, 0.0795078, 0.0268722, 0.0222731, 0.09873},
{0.391382, 0.15914, 0.0228523, 0.0215677, 0.17722},
{0.408418, 0.132911, 0.0470848, 0.0415916, 0.164965},
{0.483669, 0.132532, 0.0704862, 0.0544742, 0.193897},
{0.202792, 0.0340603, 0.00610662, 0.00592698, 0.0382848},
{0.372719, 0.0706755, 0.012342, 0.0142792, 0.0627665},
{0.406776, 0.111271, 0.0252718, 0.0185854, 0.110406},
{0.529807, 0.124025, 0.250819, 0.137435, 0.291433},
{0.357404, 0.129225, 0.0132929, 0.0122758, 0.131548},
{0.519437, 0.066343, 0.031123, 0.0301717, 0.0826295},
{0.308608, 0.046164, 0.0132147, 0.0131477, 0.0540149},
{0.326256, 0.120714, 0.0217531, 0.0218127, 0.129658},
{0.444258, 0.0484708, 0.0157797, 0.0132227, 0.0541356},
{0.427614, 0.11159, 0.107822, 0.0765893, 0.20527},
{0.483954, 0.122452, 0.203719, 0.123791, 0.27036},
{0.45838, 0.0914713, 0.040969, 0.0294818, 0.122559},
{0.326667, 0.101021, 0.0438679, 0.0342577, 0.125127},
{0.295427, 0.127893, 0.0276112, 0.0290316, 0.13657},

```
{0.400309, 0.0985245, 0.0181391, 0.021749, 0.0962477},
{0.258895, 0.0933685, 0.0270903, 0.02262, 0.11498},
{0.63772, 0.376194, 0.524086, 0.224131, 0.894427},
{0.379062, 0.103465, 0.0317477, 0.0321192, 0.117707},
{0.508875, 0.0879796, 0.0140996, 0.0169487, 0.0865996},
{0.277986, 0.10455, 0.0205991, 0.0212243, 0.098221},
{0.48596, 0.0475535, 0.0857251, 0.0494984, 0.120932},
{0.641882, 0.105427, 0.025803, 0.0386556, 0.107695},
{0.541589, 0.168825, 0.285307, 0.173774, 0.418644},
{0.287725, 0.16613, 0.028714, 0.0323681, 0.167091},
{0.503954, 0.122927, 0.0459006, 0.0371049, 0.13766},
{0.346971, 0.0781991, 0.0478895, 0.0325762, 0.113074},
{0.268271, 0.126845, 0.0450323, 0.0396026, 0.162539},
{0.326081, 0.103977, 0.0208997, 0.0191215, 0.117446},
{0.434879, 0.0771437, 0.132596, 0.0841853, 0.208179},
{0.3221, 0.106384, 0.0112398, 0.0123756, 0.108881},
{0.348302, 0.136719, 0.10725, 0.0740671, 0.233884},
{0.390694, 0.0912582, 0.0402404, 0.026953, 0.0999306},
{0.313739, 0.120165, 0.0169951, 0.0163624, 0.130453},
{0.384398, 0.134625, 0.0366189, 0.0364529, 0.155685},
{0.37733, 0.110135, 0.0813752, 0.0581953, 0.172119},
{0.578546, 0.141687, 0.713647, 0.399431, 0.819329},
{0.355774, 0.070773, 0.0229892, 0.0163896, 0.0818173},
{0.451236, 0.141502, 0.0911396, 0.060893, 0.194408},
{0.469055, 0.112268, 0.0219173, 0.02181, 0.1289},
{0.466733, 0.0976167, 0.0727353, 0.0557772, 0.150342},
{0.3939, 0.130097, 0.0348959, 0.0334674, 0.139131},
{0.457571, 0.128363, 0.0680845, 0.0565514, 0.153734},
{0.331533, 0.109804, 0.0240913, 0.0195444, 0.128599},
{0.317001, 0.166744, 0.0413699, 0.0359324, 0.194809},
{0.430622, 0.115076, 0.0358288, 0.0322727, 0.146057},
{0.400024, 0.133537, 0.0229717, 0.0210024, 0.141977},
{0.410384, 0.0878852, 0.0806532, 0.0518744, 0.155394},
{0.288169, 0.0810856, 0.0181173, 0.0148093, 0.0898577},
{0.368717, 0.1658, 0.0566747, 0.0451035, 0.197512},
{0.369156, 0.103071, 0.00990583, 0.00971486, 0.101233},
{0.487071, 0.0699914, 0.0845366, 0.0487963, 0.137321},
{0.412344, 0.12829, 0.0617049, 0.0473254, 0.188032},
{0.50956, 0.0956565, 0.131709, 0.0857642, 0.199361},
{0.415938, 0.0845864, 0.0246449, 0.0219876, 0.0989019},
{0.357556, 0.119498, 0.0579581, 0.0482875, 0.16792},
{0.362359, 0.226778, 0.122309, 0.0892503, 0.340405},
{0.480145, 0.0576893, 0.0129379, 0.0143693, 0.0549643},
```

```
{0.345858, 0.174404, 0.017136, 0.0163167, 0.186648},
{0.431837, 0.137548, 0.0414099, 0.0318305, 0.164157},
{0.350872, 0.0953691, 0.0317381, 0.0300011, 0.11318},
{0.405988, 0.139809, 0.0674083, 0.0514592, 0.203719},
{0.347197, 0.0755491, 0.0222373, 0.0183876, 0.0820053},
{0.313772, 0.055366, 0.0198017, 0.0163308, 0.0628301},
{0.530544, 0.109492, 0.102574, 0.0807206, 0.196703},
{0.34987, 0.0978935, 0.0107557, 0.00806043, 0.0980351},
{0.437705, 0.0907449, 0.0117474, 0.0152103, 0.0844911},
{0.518167, 0.118324, 0.132216, 0.0865737, 0.236223},
{0.42176, 0.0874645, 0.066811, 0.049374, 0.12745},
{0.332179, 0.177827, 0.0826272, 0.0617604, 0.247119},
{0.496166, 0.0631841, 0.0255228, 0.0246579, 0.07928},
{0.526394, 0.138796, 0.401065, 0.216661, 0.530553},
{0.339362, 0.110815, 0.0475663, 0.0424936, 0.139377},
{0.425805, 0.0924543, 0.0502133, 0.0357931, 0.109767},
{0.413562, 0.0339243, 0.0114132, 0.0095043, 0.0380853}}
```

Analyzing and visualizing the results

Averages, medians, maxes, mins, etc.

The average distance to the fixed point:

In[139]:=
```
Mean[Table[data[[i]][[3]], {i, 1, n}]]
```

Out[139]=
```
0.0604947
```

In[140]:=
```
Median[Table[data[[i]][[3]], {i, 1, n}]]
```

Out[140]=
```
0.034995
```

In[141]:=
```
Max[Table[data[[i]][[3]], {i, 1, n}]]
```

Out[141]=
```
0.713647
```

In[142]:=
```
Min[Table[data[[i]][[3]], {i, 1, n}]]
```

Out[142]=
```
0.00387523
```

In[143]:=
```
Quartiles[Table[data[[i]][[3]], {i, 1, n}]]
```

Out[143]=
```
{0.0196446, 0.034995, 0.0685926}
```

In[144]:=
```
StandardDeviation[Table[data[[i]][[3]], {i, 1, n}]]
```

Out[144]=
```
0.0829831
```

The average distance of the optimal report to the true distribution (under reporting the fixed point):

In[145]:=
```
Mean[Table[data[[i]][[4]], {i, 1, n}]]
```

Out[145]=
```
0.044482
```

In[146]:=
```
Median[Table[data[[i]][[4]], {i, 1, n}]]
```

Out[146]=
```
0.0324722
```

In[147]:=
```
Max[Table[data[[i]][[4]], {i, 1, n}]]
```

Out[147]=
```
0.399431
```

In[148]:=
```
Min[Table[data[[i]][[4]], {i, 1, n}]]
```

Out[148]=
```
0.00372018
```

In[149]:=
```
Quartiles[Table[data[[i]][[4]], {i, 1, n}]]
```

Out[149]=
```
{0.018474, 0.0324722, 0.0517452}
```

In[150]:=
```
StandardDeviation[Table[data[[i]][[4]], {i, 1, n}]]
```

Out[150]=
```
0.0452963
```

Distances to bounds

In[151]:=
```
preciseBrierBound[Lf_, distreptounif_] := Lf * distreptounif
```

In[152]:=
```
diststobounds = Table[data[[i]][[1]] * data[[i]][[5]], {i, 1, n}]
```

Out[152]=
```
{0.0699308, 0.0303033, 0.0569035, 0.023495, 0.0265335, 0.0443693, 0.112923, 0.0371834,
 0.0968451, 0.100235, 0.0328248, 0.0724022, 0.0666795, 0.0425466, 0.0301371,
 0.0340594, 0.0303859, 0.0467716, 0.0993926, 0.0263147, 0.0367422, 0.0419703,
 0.0493348, 0.133655, 0.055233, 0.0478707, 0.102688, 0.150462, 0.044204, 0.0343758,
 0.02911, 0.0450359, 0.0849609, 0.0407042, 0.0222453, 0.0185333, 0.0477985,
 0.0248265, 0.0190063, 0.0807092, 0.06259, 0.0558549, 0.054327, 0.0236493,
 0.0760946, 0.0729043, 0.0628853, 0.136951, 0.0699448, 0.126148, 0.030826,
 0.0184743, 0.0391983, 0.0432856, 0.0995521, 0.0272323, 0.230058, 0.0321865,
 0.076373, 0.0621604, 0.0196719, 0.0317516, 0.10792, 0.0642588, 0.0511294,
 0.0307696, 0.0175444, 0.0641067, 0.0894385, 0.0341856, 0.0609999, 0.0835786,
 0.0242323, 0.0419169, 0.0458836, 0.0575666, 0.0803443, 0.0596239, 0.0917751,
 0.0907279, 0.132207, 0.172678, 0.056064, 0.0258086, 0.0115155, 0.0340982,
 0.0319356, 0.0820874, 0.0515344, 0.152428, 0.0222955, 0.0186357, 0.014495,
 0.0367554, 0.0423714, 0.0450584, 0.0600968, 0.0284531, 0.0276693, 0.0497623,
 0.0708812, 0.12485, 0.126803, 0.0587987, 0.0155668, 0.015843, 0.0452478, 0.0384734,
 0.0772704, 0.0539602, 0.0595221, 0.269461, 0.0431491, 0.0295623, 0.044143,
 0.0630182, 0.0519131, 0.148603, 0.079505, 0.0130683, 0.0491547, 0.0514705,
 0.0315483, 0.0693609, 0.0673747, 0.0937821, 0.00776384, 0.0233942, 0.0449106,
 0.154403, 0.0470157, 0.0429208, 0.0166694, 0.0423018, 0.0240502, 0.0877766,
 0.130842, 0.0561787, 0.0408747, 0.0403466, 0.0385289, 0.0297678, 0.570394,
 0.0446182, 0.0440684, 0.0273041, 0.0587681, 0.0691273, 0.226733, 0.0480763,
 0.0693745, 0.0392335, 0.0436046, 0.038297, 0.0905328, 0.0350705, 0.0814622,
 0.0390423, 0.0409282, 0.0598451, 0.0649456, 0.47402, 0.0291084, 0.0877239,
 0.0604612, 0.0701697, 0.0548035, 0.0703442, 0.0426347, 0.0617547, 0.0628954,
 0.0567942, 0.0637712, 0.0258942, 0.0728258, 0.0373709, 0.0668849, 0.077534,
 0.101586, 0.0411371, 0.0600408, 0.123349, 0.0263908, 0.0645535, 0.0708891,
 0.0397116, 0.0827074, 0.028472, 0.0197143, 0.10436, 0.0342995, 0.0369822,
 0.122403, 0.0537532, 0.0820877, 0.039336, 0.27928, 0.0472993, 0.0467392, 0.0157506}
```

In[153]:=
```
Median[diststobounds]
```

Out[153]=
```
0.0495485
```

In[154]:=
```
Max[diststobounds]
```

Out[154]=
```
0.570394
```

In[155]:=
```
Min[diststobounds]
```

Out[155]=
```
0.00776384
```

In[156]:=
```
Quartiles[diststobounds]
```
Out[156]=
```
{0.0342426, 0.0495485, 0.072865}
```

Scatter Plots

Accuracy of optimal report against distance of fixed point to uniform

In[157]:=
```
subdata24 = Table[{data[[i]][[2]], data[[i]][[4]]}, {i, 1, n}];
```

In[158]:=
```
minx = Min[Table[subdata24[[i]][[1]], {i, 1, n}]]
```
Out[158]=
```
0.0339243
```

In[159]:=
```
maxx = Max[Table[subdata24[[i]][[1]], {i, 1, n}]]
```
Out[159]=
```
0.376194
```

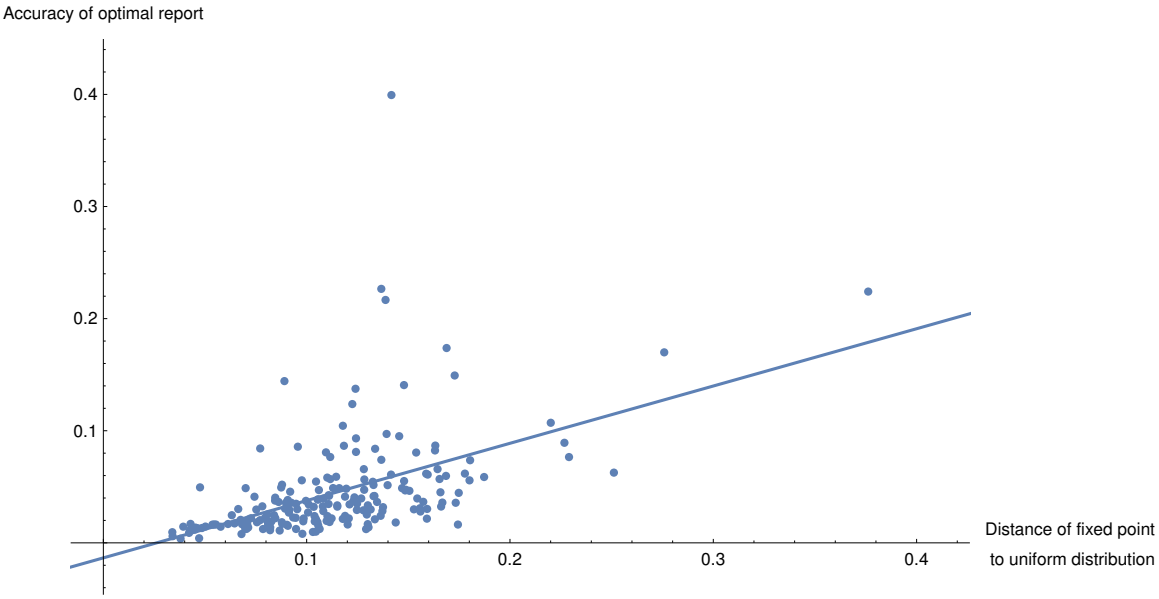
In[160]:=
```
miny = Min[Table[subdata24[[i]][[2]], {i, 1, n}]]
```
Out[160]=
```
0.00372018
```

In[161]:=
```
maxy = Max[Table[subdata24[[i]][[2]], {i, 1, n}]]
```
Out[161]=
```
0.399431
```

In[162]:=
```
trend = Fit[subdata24, {1, x}, x]
```
Out[162]=
```
-0.0134019 + 0.51097 x
```

In[163]:=
```
Show[{Plot[Evaluate@trend, {x, minx - 0.05, maxx + 0.05},
    PlotRange → {{minx - 0.05, maxx + 0.05}, {miny - 0.05, maxy + 0.05}},
    AxesLabel → {"Distance of fixed point\n to uniform distribution",
      "Accuracy of optimal report"}], ListPlot[subdata24]}, ImageSize → 600]
```

Out[163]=



Distance of optimal report to fixed point against distance of fixed point to uniform

In[164]:=
```
subdata23 = Table[{data[[i]][[2]], data[[i]][[3]]}, {i, 1, n}];
```

In[165]:=
```
minx = Min[Table[subdata23[[i]][[1]], {i, 1, n}]]
```

Out[165]=
```
0.0339243
```

In[166]:=
```
maxx = Max[Table[subdata23[[i]][[1]], {i, 1, n}]]
```

Out[166]=
```
0.376194
```

In[167]:=
```
miny = Min[Table[subdata23[[i]][[2]], {i, 1, n}]]
```

Out[167]=
```
0.00387523
```

In[168]:=
```
maxy = Max[Table[subdata23[[i]][[2]], {i, 1, n}]]
```

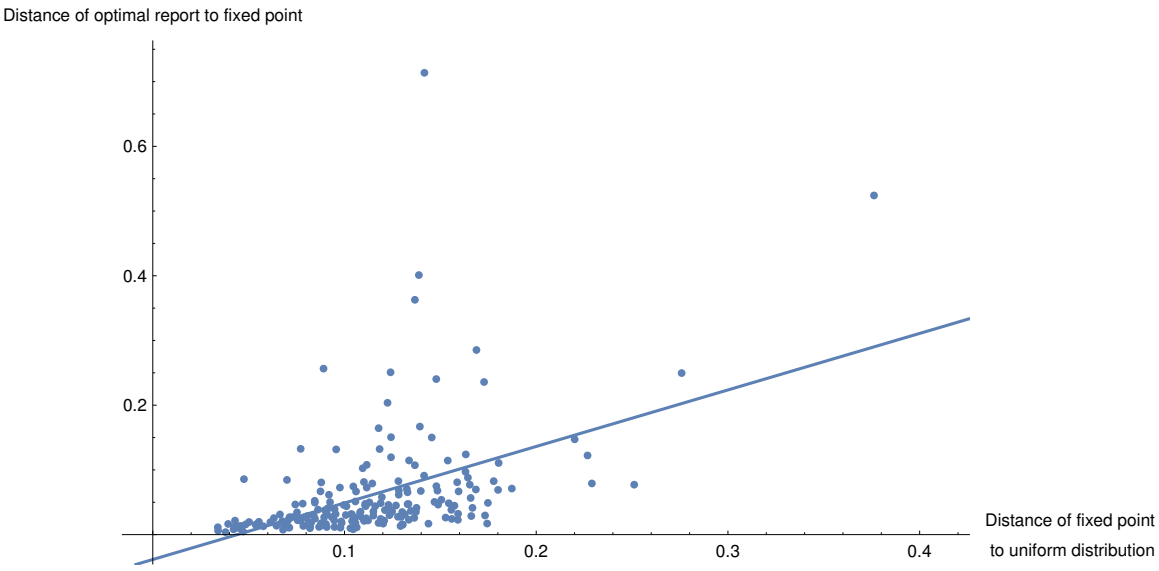
Out[168]=
```
0.713647
```

In[169]:=
```
trend = Fit[subdata23, {1, x}, x]
```

Out[169]=
```
-0.038445 + 0.87339 x
```

In[170]:=
```
Show[{Plot[Evaluate@trend, {x, minx - 0.05, maxx + 0.05},
    PlotRange → {{minx - 0.05, maxx + 0.05}, {miny - 0.05, maxy + 0.05}},
    AxesLabel → {"Distance of fixed point\n to uniform distribution",
      "Distance of optimal report to fixed point"}],
  ListPlot[subdata23]}, ImageSize → 600]
```

Out[170]=



Accuracy of the optimal report against operator norm of A

In[196]:=
```
subdata14 = Table[{data[[i]][[1]], data[[i]][[4]]}, {i, 1, n}];
```

In[197]:=
```
minx = Min[Table[subdata14[[i]][[1]], {i, 1, n}]]
```

Out[197]=
```
0.202792
```

In[198]:=
```
maxx = Max[Table[subdata14[[i]][[1]], {i, 1, n}]]
```

Out[198]=
```
0.641882
```

In[199]:=
```
miny = Min[Table[subdata14[[i]][[2]], {i, 1, n}]]
```

Out[199]=
```
0.00372018
```

In[200]:=
```
maxy = Max[Table[subdata14[[i]][[2]], {i, 1, n}]]
```

Out[200]=
```
0.399431
```

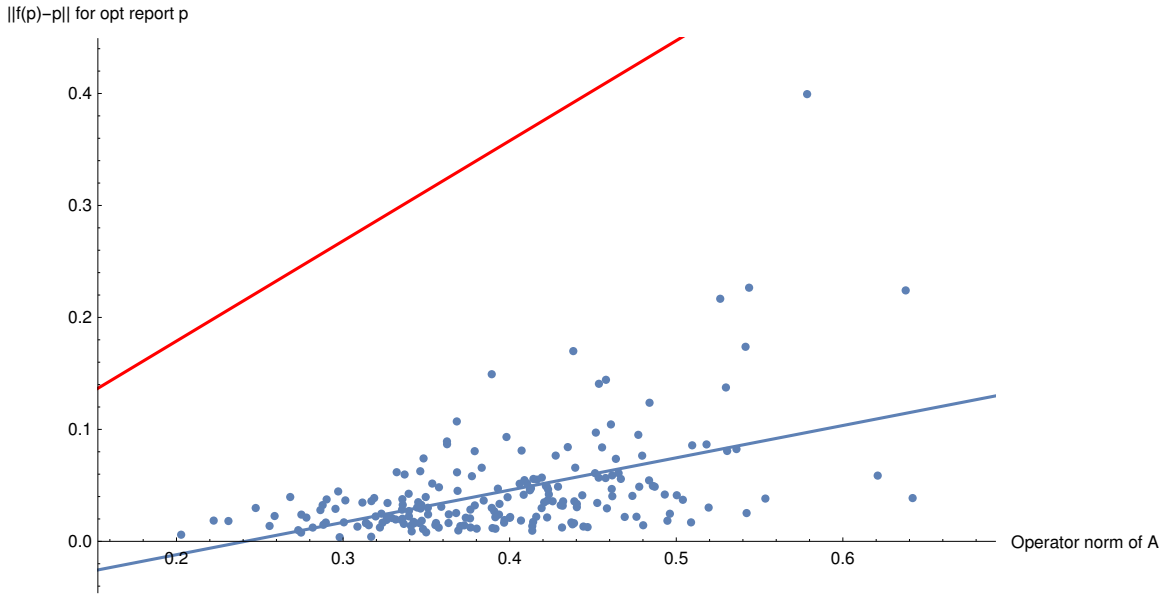
In[201]:=
```
trend = Fit[subdata14, {1, x}, x]
```

Out[201]=
```
-0.0696001 + 0.28845 x
```

In[229]:=
```
accbound[x_] := If[S == BrierScore, Sqrt[(dim - 1) / dim] * x, {}]
```

In[230]:=
```
Show[{Plot[Evaluate@trend, {x, minx - 0.05, maxx + 0.05},
    PlotRange → {{minx - 0.05, maxx + 0.05}, {miny - 0.05, maxy + 0.05}},
    AxesLabel → {"Operator norm of A", "||f(p)-p|| for opt report p"}],
  Plot[accbound[x], {x, minx - 0.05, maxx + 0.05}, PlotStyle → Red],
  ListPlot[subdata14]}, ImageSize → 600]
```

Out[230]=



Distance of optimal report to fixed point against operator norm of A

In[179]:=
```
subdata13 = Table[{data[[i]][[1]], data[[i]][[3]]}, {i, 1, n}];
```

In[180]:=
```
minx = Min[Table[subdata13[[i]][[1]], {i, 1, n}]]
```

Out[180]=
```
0.202792
```

In[181]:=
```
maxx = Max[Table[subdata13[[i]][[1]], {i, 1, n}]]
```

Out[181]=
```
0.641882
```

In[182]:=
```
miny = Min[Table[subdata13[[i]][[2]], {i, 1, n}]]
```

Out[182]=
```
0.00387523
```

In[183]:=
```
maxy = Max[Table[subdata13[[i]][[2]], {i, 1, n}]]
```

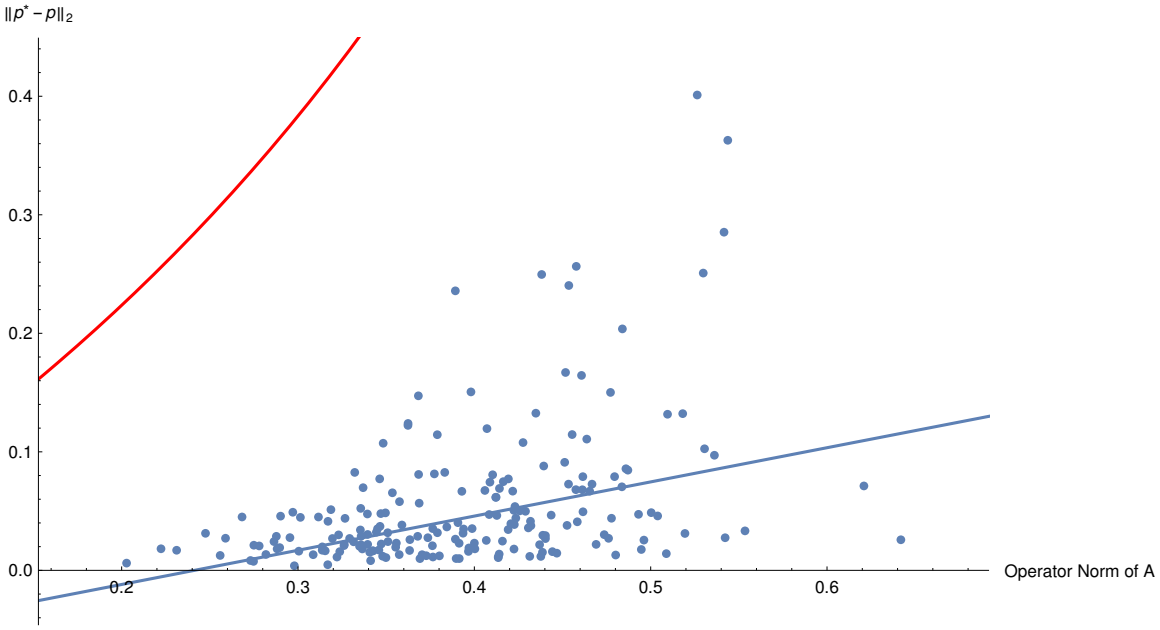
Out[183]=
```
0.713647
```

In[184]:=
```
trend = Fit[subdata13, {1, x}, x]
```

Out[184]=
```
-0.140752 + 0.508839 x
```

In[238]:=
```
fpdistbound[x_] := If[S == BrierScore, Sqrt[(dim - 1) / dim] * (x / (1 - x)), {}]
```

In[271]:=
```
Show[{Plot[Evaluate@trend, {x, minx - 0.05, maxx + 0.05},
    PlotRange → {{minx - 0.05, maxx + 0.05}, {miny - 0.05, maxy + 0.05}},
    AxesLabel → {"Operator Norm of A",
      TraditionalForm[Norm[ToExpression["p^*-p", TeXForm, HoldForm], 2]]}],
  ListPlot[subdata13], Plot[fpdistbound[x], {x, minx - 0.05, maxx + 0.05},
    PlotStyle → Red]}, ImageSize → 600]
```



Accuracy of the optimal report against distance of optimal report to fixed point, along with the plot

In[186]:=
```
subdata54 = Table[{data[[i]][[5]], data[[i]][[4]]}, {i, 1, n}];
```

In[187]:=
```
minx = Min[Table[subdata54[[i]][[1]], {i, 1, n}]]
```

Out[187]=
```
0.0380853
```

In[188]:=
```
subdata54
```

Out[188]=
```
{{0.12897, 0.0252574}, {0.088455, 0.0160977}, {0.125734, 0.0343038},
 {0.0686312, 0.017432}, {0.060716, 0.0169124}, {0.104971, 0.0364599},
 {0.24507, 0.10443}, {0.111027, 0.019206}, {0.21263, 0.0838968}, {0.251797, 0.0932058},
 {0.0826978, 0.0190567}, {0.172676, 0.0569426}, {0.147064, 0.0568762},
 {0.126861, 0.0282938}, {0.0728719, 0.0136811}, {0.101427, 0.0325898},
 {0.0905293, 0.0202132}, {0.135527, 0.0349805}, {0.274251, 0.0867666},
 {0.0635487, 0.0173231}, {0.109201, 0.0155822}, {0.146553, 0.0279222},
 {0.144805, 0.0138948}, {0.362933, 0.107107}, {0.131285, 0.0348376},
```

{0.153629, 0.0346417}, {0.233728, 0.0657706}, {0.242322, 0.0587207},
{0.0928426, 0.0219645}, {0.0920013, 0.0211027}, {0.10587, 0.0238741},
{0.128302, 0.0240502}, {0.230622, 0.0616631}, {0.0971063, 0.0297114},
{0.0625949, 0.0145219}, {0.0487471, 0.0113627}, {0.149994, 0.0386909},
{0.0881013, 0.0123129}, {0.0742814, 0.013749}, {0.173421, 0.0608947},
{0.160798, 0.0298841}, {0.100915, 0.0382141}, {0.12859, 0.0213675},
{0.106377, 0.0185408}, {0.225852, 0.0597101}, {0.176626, 0.0473207},
{0.179878, 0.0395867}, {0.285592, 0.0765232}, {0.182519, 0.0657469},
{0.272016, 0.0736804}, {0.133347, 0.0181613}, {0.0672308, 0.00784818},
{0.135112, 0.0374534}, {0.100331, 0.0123646}, {0.240324, 0.0557357},
{0.072329, 0.0123922}, {0.525028, 0.169918}, {0.0924896, 0.0109728},
{0.165576, 0.0467611}, {0.175831, 0.051616}, {0.0607758, 0.0166444},
{0.0710682, 0.0127217}, {0.311616, 0.0626305}, {0.179027, 0.0308707},
{0.119179, 0.0488559}, {0.0906285, 0.0223089}, {0.0473691, 0.013604},
{0.186116, 0.0302457}, {0.235986, 0.0805511}, {0.105833, 0.0247243},
{0.148014, 0.0456186}, {0.197567, 0.0464008}, {0.0667026, 0.0163724},
{0.0847332, 0.0184073}, {0.10344, 0.0411428}, {0.115062, 0.0411876},
{0.174083, 0.0589309}, {0.197849, 0.0365902}, {0.220392, 0.0551255},
{0.222833, 0.0810946}, {0.277039, 0.0951366}, {0.380704, 0.140719},
{0.154254, 0.024111}, {0.0588393, 0.0162153}, {0.0386509, 0.00372018},
{0.137743, 0.029794}, {0.0815834, 0.0112702}, {0.16647, 0.0418087}, {0.1088, 0.0405892},
{0.391619, 0.149272}, {0.0742004, 0.0169766}, {0.0643442, 0.0168091},
{0.0459524, 0.0144171}, {0.0834741, 0.0305228}, {0.124808, 0.0270732},
{0.104282, 0.0357285}, {0.202307, 0.0445852}, {0.0756285, 0.0205001},
{0.101325, 0.0099718}, {0.125453, 0.016699}, {0.173371, 0.0546682},
{0.276387, 0.0971312}, {0.236511, 0.082415}, {0.138836, 0.0419592},
{0.0491257, 0.00410697}, {0.0406561, 0.0118592}, {0.113477, 0.0394434},
{0.104583, 0.0253032}, {0.176086, 0.0357505}, {0.116896, 0.040312},
{0.158127, 0.0283492}, {0.495587, 0.226547}, {0.109626, 0.0239155},
{0.0897809, 0.0208737}, {0.131543, 0.0376839}, {0.160384, 0.0470607},
{0.149816, 0.0293105}, {0.324603, 0.144278}, {0.188212, 0.0487807},
{0.038303, 0.00890504}, {0.102882, 0.0487792}, {0.116894, 0.033437},
{0.09873, 0.0222731}, {0.17722, 0.0215677}, {0.164965, 0.0415916},
{0.193897, 0.0544742}, {0.0382848, 0.00592698}, {0.0627665, 0.0142792},
{0.110406, 0.0185854}, {0.291433, 0.137435}, {0.131548, 0.0122758},
{0.0826295, 0.0301717}, {0.0540149, 0.0131477}, {0.129658, 0.0218127},
{0.0541356, 0.0132227}, {0.20527, 0.0765893}, {0.27036, 0.123791},
{0.122559, 0.0294818}, {0.125127, 0.0342577}, {0.13657, 0.0290316},
{0.0962477, 0.021749}, {0.11498, 0.02262}, {0.894427, 0.224131}, {0.117707, 0.0321192},
{0.0865996, 0.0169487}, {0.098221, 0.0212243}, {0.120932, 0.0494984},
{0.107695, 0.0386556}, {0.418644, 0.173774}, {0.167091, 0.0323681},
{0.13766, 0.0371049}, {0.113074, 0.0325762}, {0.162539, 0.0396026},
{0.117446, 0.0191215}, {0.208179, 0.0841853}, {0.108881, 0.0123756},

```
{0.233884, 0.0740671}, {0.0999306, 0.026953}, {0.130453, 0.0163624},
{0.155685, 0.0364529}, {0.172119, 0.0581953}, {0.819329, 0.399431},
{0.0818173, 0.0163896}, {0.194408, 0.060893}, {0.1289, 0.02181}, {0.150342, 0.0557772},
{0.139131, 0.0334674}, {0.153734, 0.0565514}, {0.128599, 0.0195444},
{0.194809, 0.0359324}, {0.146057, 0.0322727}, {0.141977, 0.0210024},
{0.155394, 0.0518744}, {0.0898577, 0.0148093}, {0.197512, 0.0451035},
{0.101233, 0.00971486}, {0.137321, 0.0487963}, {0.188032, 0.0473254},
{0.199361, 0.0857642}, {0.0989019, 0.0219876}, {0.16792, 0.0482875},
{0.340405, 0.0892503}, {0.0549643, 0.0143693}, {0.186648, 0.0163167},
{0.164157, 0.0318305}, {0.11318, 0.0300011}, {0.203719, 0.0514592},
{0.0820053, 0.0183876}, {0.0628301, 0.0163308}, {0.196703, 0.0807206},
{0.0980351, 0.00806043}, {0.0844911, 0.0152103}, {0.236223, 0.0865737},
{0.12745, 0.049374}, {0.247119, 0.0617604}, {0.07928, 0.0246579}, {0.530553, 0.216661},
{0.139377, 0.0424936}, {0.109767, 0.0357931}, {0.0380853, 0.0095043}}
```

In[189]:=
```
maxx = Max[Table[subdata54[[i]][[1]], {i, 1, n}]]
```

Out[189]=
```
0.894427
```

In[190]:=
```
miny = Min[Table[subdata54[[i]][[2]], {i, 1, n}]]
```

Out[190]=
```
0.00372018
```

In[191]:=
```
maxy = Max[Table[subdata54[[i]][[2]], {i, 1, n}]]
```

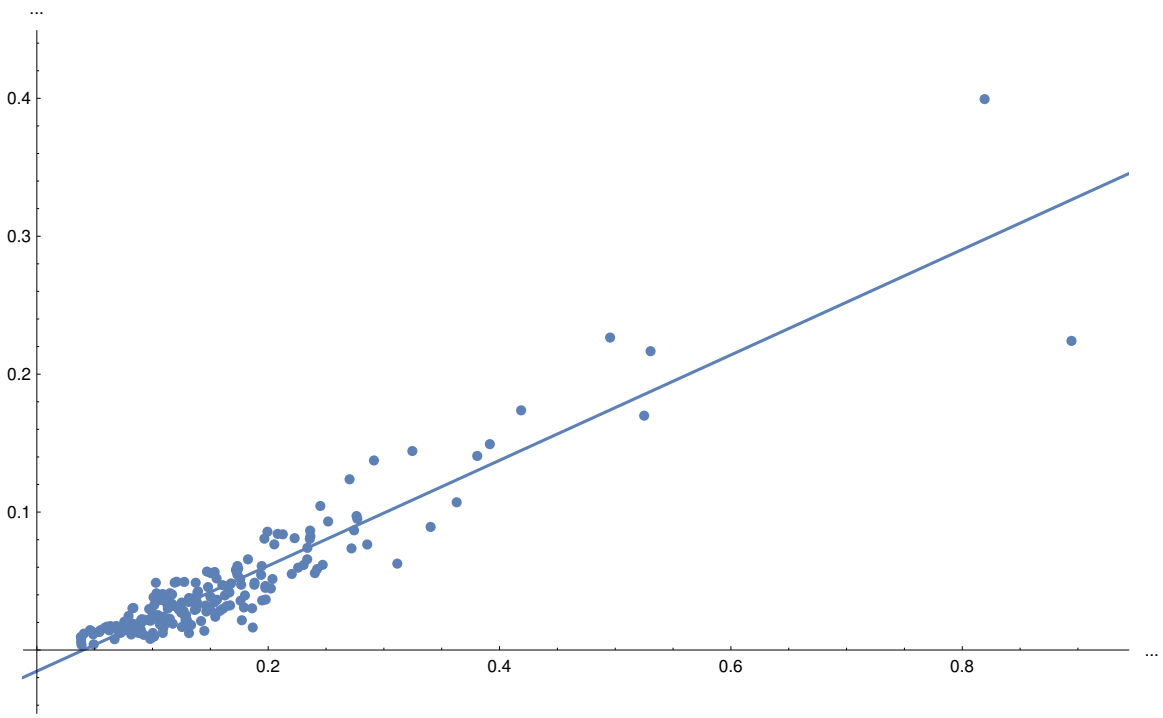
Out[191]=
```
0.399431
```

In[192]:=
```
trend = Fit[subdata54, {1, x}, x]
```

Out[192]=
```
-0.0153442 + 0.382159 x
```

In[193]:=
```
Show[{Plot[Evaluate@trend, {x, minx - 0.05, maxx + 0.05},
    PlotRange → {{minx - 0.05, maxx + 0.05}, {miny - 0.05, maxy + 0.05}},
    AxesLabel → {"...", "..."}], ListPlot[subdata54]}, ImageSize → 600]
```

Out[193]=



Appendix A: Finding optimal reports with Mathematica in the Log Scoring rule case seems difficult

In[ ]:=
```
Clear[x];
A = RandomA[3];
Timing[TimeConstrained[Solve[D[LogScore[Array[x, 3], A.Array[x, 3]], x[1]] == 0 &&
     D[LogScore[Array[x, 3], A.Array[x, 3]], x[2]] == 0 &&
     D[LogScore[Array[x, 3], A.Array[x, 3]], x[3]] == 0], 300]]
```

Out[ ]=
```
$Aborted
```

In[•]:=
```
A = RandomA[3];
TimeConstrained[Maximize[{LogScore[Array[pmax, 3], A.Array[pmax, 3]],
   IsDistr[Array[pmax, 3]]}, Array[pmax, 3]], 500]
```

Out[•]=
$$\text{Maximize}\left[\left\{\text{Log}[\text{pmax}[3]]\left(\frac{37\ \text{pmax}[1]}{82}+\frac{9\ \text{pmax}[2]}{35}\right)+\text{Log}[\text{pmax}[2]]\left(\frac{35\ \text{pmax}[1]}{82}+\frac{54\ \text{pmax}[2]}{175}+\frac{11\ \text{pmax}[3]}{85}\right)+\right.\right.$$
$$\left.\text{Log}[\text{pmax}[1]]\left(\frac{5\ \text{pmax}[1]}{41}+\frac{76\ \text{pmax}[2]}{175}+\frac{74\ \text{pmax}[3]}{85}\right),$$
$$\left.\text{pmax}[1]+\text{pmax}[2]+\text{pmax}[3]==1\ \&\&\ \text{NonNegative}[\text{pmax}[1]]\ \&\&\ \text{NonNegative}[\text{pmax}[2]]\ \&\&\ \text{NonNegative}[\text{pmax}[3]]\right\},\ \{\text{pmax}[1],\ \text{pmax}[2],\ \text{pmax}[3]\}\right]$$