Сейсмическая задача

Подготовка

GeologyIO` и GPNTools` разработаны специально для компании Газпром Нефть НТЦ

Обращаться к Екименко Антону

In[1]:=

```
$HistoryLength = 0;
<<GeologyIO`
<<GPNTools`
SetDirectory[NotebookDirectory[]];
```

Импорт данных

Сам импорт

In[ ]:=
```
trainTable =
Normal[Import["Heff+NTG_train.xlsx", "Dataset", "HeaderLines" -> 1][[1]]];
```

In[ ]:=
```
testTable =
Normal[Import["Heff+NTG_train.xlsx", "Dataset", "HeaderLines" -> 1][[2]]];
```

In[ ]:=
```
horiz = Function[arr, <|
    "toCoord" -> Dispatch[Round[#[[{1, 2}]]] ->
        #[[{3, 4, 5}]]& /@ arr[[All, {3, 6, 7, 8, 9}]]],
    "toIndex" -> Dispatch[Round[#[[{3, 4}]]] ->
        #[[{1, 2}]]& /@ arr[[All, {3, 6, 7, 8, 9}]]]
|>] @
Import["seismic_interpretation_new.charisma"];
```

In[ ]:=
```
cube =
SEGYImport["3D_cube_new.sgy", "Loading" -> "Delayed"];
```

Таблицы

In[ ]:=
```
Dataset[trainTable]
```

Out[ ]=

x	y	Heff1a	Heff1b	Heff2	NTG1a	NTG1b
−1 539 284.	7 357 878.	0.18	0.99	4.12	0.47	0.46
−1 529 063.	7 363 225.	2.51	0.04	0.07	1.0	0.03
−1 533 351.	7 368 928.	2.74	4.4	5.6	0.76	0.32
−1 536 884.	7 365 563.	14.57	5.8	8.38	0.88	0.71
−1 536 957.	7 366 245.	10.78	0.02	5.61	0.84	0.06
−1 536 413.	7 364 685.	7.4	0.0	2.09	0.82	0.73
−1 536 579.	7 366 732.	9.39	5.6	0.01	0.64	0.6
−1 537 977.	7 361 696.	2.81	0.0	3.41	1.0	0.0
−1 536 185.	7 361 024.	8.42	2.16	3.33	0.89	1.0
−1 535 879.	7 361 532.	3.57	1.19	0.72	0.58	1.0
−1 536 511.	7 361 478.	7.43	0.05	0.12	0.75	1.0
−1 537 103.	7 360 694.	16.71	0.01	0.06	0.85	1.0
−1 540 580.	7 370 157.	1.09	0.0	0.08	0.67	0.0
−1 537 313.	7 365 044.	7.76	4.52	7.97	0.82	0.72
−1 540 809.	7 363 851.	1.49	0.0	0.57	1.0	0.0
−1 539 963.	7 362 406.	3.2	0.0	0.53	0.84	0.0
−1 540 446.	7 359 539.	8.45	0.0	0.0	0.82	0.0
−1 537 229.	7 364 760.	7.19	2.01	7.73	0.8	0.71
−1 529 908.	7 358 195.	0.77	3.6	4.14	0.71	1.0
−1 528 459.	7 362 109.	1.7	9.2	5.82	0.43	0.7

showing 1–20 of 30

In[ ]:=
```
Dataset[testTable]
```

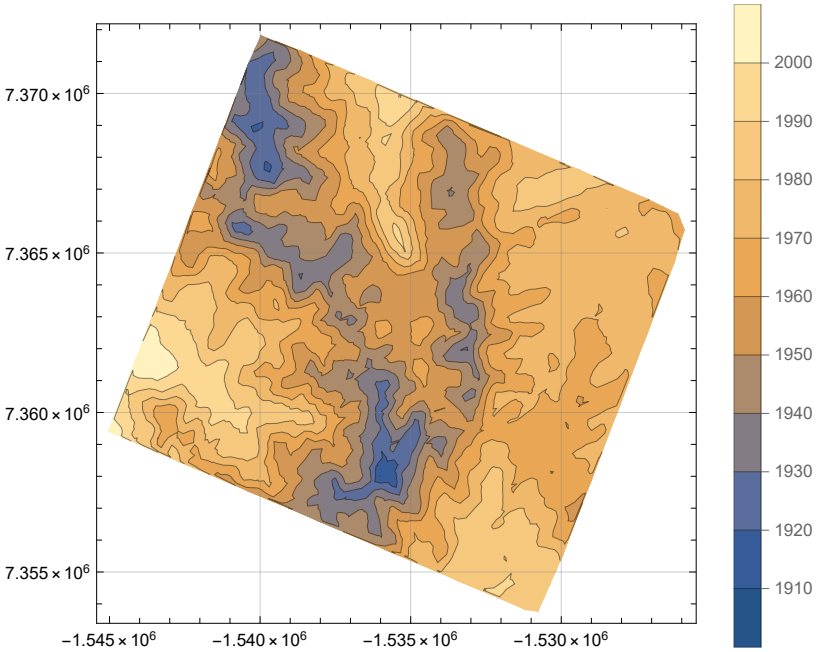
Out[ ]=

x	y	Heff1a	Heff1b	Heff2	NTG1a	NTG1b
−1 539 284.	7 357 878.	0.0	0.0	0.0	0.0	0.0
−1 535 537.	7 362 509.	0.0	0.0	0.0	0.0	0.0
−1 542 815.	7 358 588.	0.0	0.0	0.0	0.0	0.0
−1 539 369.	7 365 372.	0.0	0.0	0.0	0.0	0.0
−1 532 044.	7 360 515.	0.0	0.0	0.0	0.0	0.0
−1 537 476.	7 367 782.	0.0	0.0	0.0	0.0	0.0
−1 526 450.	7 364 392.	0.0	0.0	0.0	0.0	0.0
−1 539 191.	7 371 095.	0.0	0.0	0.0	0.0	0.0

3D-карта глубин

In[ ]:=
```
ListContourPlot[Values[Normal[horiz["toCoord"]][[1 ;; -1 ;; 100]]],
    ImageSize -> Medium,
    GridLines -> Automatic,
    PlotLegends -> Automatic
]
```

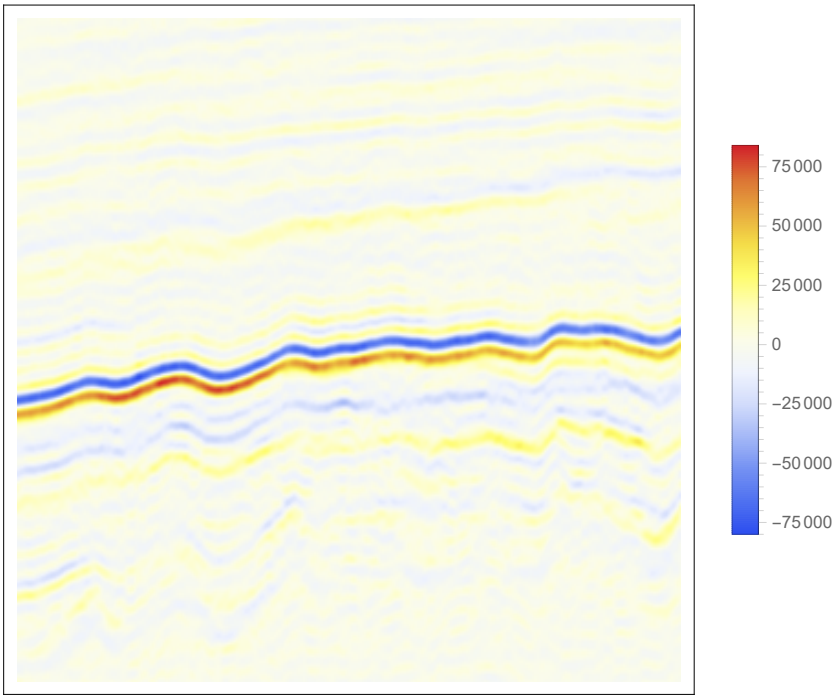
Out[ ]=



Срез куба

In[ ]:=
```
ArrayPlot[
    Transpose[SEGYLoad[cube["TracesUnloaded", 1 ;;
        cube["BinaryHeader", "NumberOfSamplesForReel"]]]],
    ColorFunction -> "TemperatureMap",
    ImageSize -> Medium, PlotLegends -> Automatic
]
```

Out[ ]=



Приведение данных

У нас есть координаты из тестовых данных.

По этим координатам необходимо взять трассы.

К сожалению координаты трасс и скважин не совпадают точно.

Придется ради экономии написать “неадекватный” код:

In[ ]:=
```
Options[WellTraces] = {"count" -> 1};
```

In[ ]:=
```
WellTraces[cube_SEGYData, horiz_Association, table_List, OptionsPattern[]] :=
Block[{$indexes, $traceIndex,
    $ilineMin, $ilineMax, $xlineMin, $xlineMax, $ilineLen, $xlineLen,
    $count = OptionValue["count"],
    $delrt = ("delrt" /. SEGYLoad[cube["traceheadersunloaded", 1]]) / 1000.0,
    $dt = ("dt" /. SEGYLoad[cube["traceheadersunloaded", 1]]) / 1000000.0,
    $ns = "ns" /. SEGYLoad[cube["traceheadersunloaded", 1]]
},
    $indexes = Flatten[Table[
        Select[(g + #& /@ SortBy[Tuples[Range[-100, 100], 2], Abs /* Total]) /.
        horiz["toIndex"], Total[Abs[#]] < 10000&][[1 ;; $count]],
        {g, Round[Normal[Query[All, {"x", "y"} /* Values] @ table]]}
    ], 1];
    {$ilineMin, $ilineMax} = "iline" /.
        SEGYLoad[cube["TraceHeadersUnloaded", {1, -1}]];
    {$xlineMin, $xlineMax} = "xline" /.
        SEGYLoad[cube["TraceHeadersUnloaded", {1, -1}]];
    $ilineLen = $ilineMax - $ilineMin + 1;
    $xlineLen = $xlineMax - $xlineMin + 1;
    $traceIndex[{iline_Integer, xline_Integer}] :=
        (iline - $ilineMin) * $xlineLen + xline - $xlineMin + 1;

    MapThread[
        Association[Append[
            Thread[{"x", "y", "t"} -> #1 / {1, 1, 1000.0}],
            "trace" -> TimeSeries[#2,
                {Range[$delrt, $delrt + $ns * $dt - $dt, $dt]}]
        ]]&,
        {
            $indexes /. horiz["toCoord"],
            SEGYLoad[cube["tracesunloaded", $traceIndex /@ $indexes]]
        }
    ]
]
```

Выберем по одной ближайшей трассе около точек из тестовой таблицы:

In[ ]:=
```
Dataset[WellTraces[cube, horiz, testTable, "count" -> 1]]
```

Out[ ]=

x	y	t	trace
-1539289.	7357875.	1.96537	TimeSeries[Time: 1.65 to 2.25 Data points: 301]
-1535541.	7362506.	1.95689	TimeSeries[Time: 1.65 to 2.25 Data points: 301]
-1542818.	7358575.	1.98516	TimeSeries[Time: 1.65 to 2.25 Data points: 301]
-1539377.	7365369.	1.93811	TimeSeries[Time: 1.65 to 2.25 Data points: 301]
-1532053.	7360508.	1.97382	TimeSeries[Time: 1.65 to 2.25 Data points: 301]
-1537471.	7367766.	1.96148	TimeSeries[Time: 1.65 to 2.25 Data points: 301]
-1526459.	7364394.	1.97138	TimeSeries[Time: 1.65 to 2.25 Data points: 301]
-1539189.	7371089.	1.9338	TimeSeries[Time: 1.65 to 2.25 Data points: 301]

Интерполяция таблицы

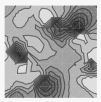
Трассы находятся не точно в скважинах. Если у нас 4 близкие трассы - то есть проблема выбора значений для них.
Попробуем просто использовать IDW на таблице:

In[ ]:=
```
Table[
    trainValues[column] =
    IDWInterpolation[Values[trainTable[[All, {"x", "y", column}]]]],
    {column, {"Heff1a", "Heff1b", "Heff2", "NTG1a", "NTG1b", "NTG2"}}
]
```

Out[ ]=
```
{IDWInterpolatedFunction[
```

```
Range: x: {-1.54081×10^6, -1.52773×10^6} y: {7.35788×10^6, 7.37016×10^6}
Radius: 1.13×10^4    Delta: 0.0167    Beta: 2
],
IDWInterpolatedFunction[
```

```
Range: x: {-1.54081×10^6, -1.52773×10^6} y: {7.35788×10^6, 7.37016×10^6}
Radius: 1.13×10^4    Delta: 0.0092    Beta: 2
],
IDWInterpolatedFunction[
```

```
Range: x: {-1.54081×10^6, -1.52773×10^6} y: {7.35788×10^6, 7.37016×10^6}
Radius: 1.13×10^4    Delta: 0.00838    Beta: 2
],
IDWInterpolatedFunction[
```

```
Range: x: {-1.54081×10^6, -1.52773×10^6} y: {7.35788×10^6, 7.37016×10^6}
Radius: 1.13×10^4    Delta: 0.00099    Beta: 2
],
IDWInterpolatedFunction[
```

```
Range: x: {-1.54081×10^6, -1.52773×10^6} y: {7.35788×10^6, 7.37016×10^6}
Radius: 1.13×10^4    Delta: 0.001    Beta: 2
],
IDWInterpolatedFunction[
```

```
Range: x: {-1.54081×10^6, -1.52773×10^6} y: {7.35788×10^6, 7.37016×10^6}
Radius: 1.13×10^4    Delta: 0.001    Beta: 2
]}
```

Теперь даже если координата трассы не совпадает точно
с координатой известного значения из таблицы,
то все равно можно вычислить приближенно значения толщин.

Однако стоит помнить, что для малого числа точек это работает только вбизи известных значений

In[ ]:=
```
trainTable[[1, "Heff1a"]]
trainValues["Heff1a"][Values[trainTable[[1, {"x", "y"}]]] + {100, 100}]
```

Out[ ]= 0.18

Out[ ]= 0.180286

Функции

Здесь стоит определять все вспомогательные функции

Получение куска временного ряда

In[ ]:=
```
traceWindow[{t1_?NumericQ, t2_?NumericQ}] :=
Function[a, TimeSeriesWindow[a["trace"], a["t"] + {t1, t2}]]
```

In[ ]:=
```
traceWindow[dt_?NumericQ] := traceWindow[{-dt, dt} / 2]
```

Разложение трассы на компоненты

In[ ]:=
```
waveleComponent[n_Integer][ts_] :=
TimeSeries[Re[ContinuousWaveletTransform[ts["Values"], MexicanHatWavelet[]][[1, n]]], {ts["Ti
```

Это специальное определение для вырезки окна

In[ ]:=
```
waveleComponent[n_Integer][a_Association] :=
<|"t" -> a["t"], "trace" -> waveleComponent[n][a["trace"]]|>
```

Поиск ближайшего минимума слева/справа

In[ ]:=
```
nearMin["left"][a_Association] :=
Block[{
    $w = traceWindow[{a["trace"]["FirstTime"]-a["t"], 0}][a]["Path"],
    $tmin = 0
},
    Print[$w];
    Table[
        If[$tmin === 0 && $w[[-i, 2]] <=
        $w[[-i + 1, 2]] && $w[[-i, 2]] < $w[[-i - 1, 2]],
            $tmin = $w[[i, 1]]
        ],
        {i, 2, Length[$w] - 1}
    ]; $tmin
]
```

In[ ]:=
```
nearMin["right"][a_Association] :=
Block[{
    $w = traceWindow[{0, a["trace"]["LastTime"]-a["t"]}][a]["Path"],
    $tmin = 0
},
    Print[$w];
    Table[
        If[$tmin === 0 && $w[[i, 2]] <=
        $w[[i + 1, 2]] && $w[[i, 2]] < $w[[i - 1, 2]],
            $tmin = $w[[i, 1]]
        ],
        {i, 2, Length[$w] - 1}
    ]; $tmin
]
```

Создание таблицы с атрибутами

```
DataQuery = Query[All, <|
    "x" -> "x",
    "y" -> "y",
    "t" -> "t",
    "trace" -> "trace",
    "window" -> {"t", "trace"} /*
        traceWindow[{0, 0.03}],
    "windowImg" -> {"t", "trace"} /*
        traceWindow[{0, 0.03}] /* DateListPlot,
    "RMS(0ms)" -> {"t", "trace"} /*
        traceWindow[{0.00,0.01}] /* RootMeanSquare,
    "RMS(10ms)" -> {"t", "trace"} /*
        traceWindow[{0.01,0.02}] /* RootMeanSquare,
    "RMS(20ms)" -> {"t", "trace"} /*
        traceWindow[{0.020,0.03}] /* RootMeanSquare,
    "RMS(0mscwt8)" -> {"t", "trace"} /*
        waveleComponent[8] /* traceWindow[{0.00,0.01}] /* RootMeanSquare,
    "RMS(0mscwt16)" -> {"t", "trace"} /*
        waveleComponent[16] /* traceWindow[{0.00,0.01}] /* RootMeanSquare,
    "RMS(0mscwt20)" -> {"t", "trace"} /*
        waveleComponent[20] /* traceWindow[{0.00,0.01}] /* RootMeanSquare,
    "RMS(10mscwt8)" -> {"t", "trace"} /*
        waveleComponent[8] /* traceWindow[{0.01,0.02}] /* RootMeanSquare,
    "RMS(10mscwt16)" -> {"t", "trace"} /*
        waveleComponent[16] /* traceWindow[{0.01,0.02}] /* RootMeanSquare,
    "RMS(10mscwt20)" -> {"t", "trace"} /*
        waveleComponent[20] /* traceWindow[{0.01,0.02}] /* RootMeanSquare,
    "RMS(20mscwt8)" -> {"t", "trace"} /*
        waveleComponent[8] /* traceWindow[{0.02,0.03}] /* RootMeanSquare,
    "RMS(20mscwt16)" -> {"t", "trace"} /*
        waveleComponent[16] /* traceWindow[{0.02,0.03}] /* RootMeanSquare,
    "RMS(20mscwt20)" -> {"t", "trace"} /*
        waveleComponent[20] /* traceWindow[{0.02,0.03}] /* RootMeanSquare,

    "Heff1a" -> {"x", "y"} /* Values /*
        trainValues["Heff1a"] /* (# * (1 + RandomReal[{-0.025, 0.025}])&),
    "Heff1b" -> {"x", "y"} /* Values /*
        trainValues["Heff1b"] /* (# * (1 + RandomReal[{-0.025, 0.025}])&),
    "Heff2" -> {"x", "y"} /* Values /*
        trainValues["Heff2"] /* (# * (1 + RandomReal[{-0.025, 0.025}])&),
    "NTG1a" -> {"x", "y"} /* Values /* trainValues["NTG1a"] /*
        (# * (1 + RandomReal[{-0.025, 0.025}])&),
    "NTG1b" -> {"x", "y"} /* Values /*
        trainValues["NTG1b"] /* (# * (1 + RandomReal[{-0.025, 0.025}])&),
    "NTG2" -> {"x", "y"} /* Values /* trainValues["NTG2"] /*
        (# * (1 + RandomReal[{-0.025, 0.025}])&)
|>];
```

Выбор по 6 трасс около каждой скважины для тренировки

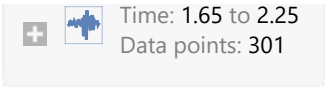
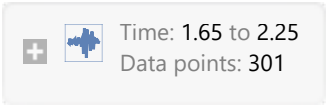
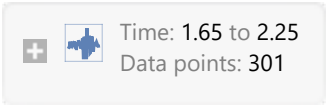
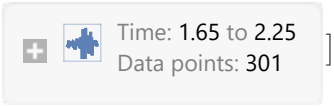
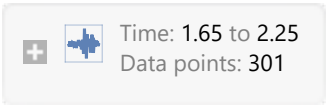
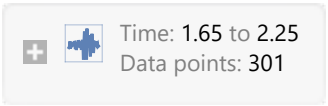
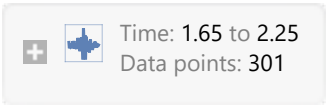
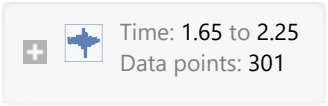
In[ ]:=
```
trainTraces = WellTraces[cube, horiz, trainTable, "count" -> 6];
```

Таблица с данным для тренировки

In[ ]:=
```
Dataset[trainDataset = DataQuery @ trainTraces]
```

Out[ ]=

x	y	t	trace	window
−1 539 289.	7 357 875.	1.96537	TimeSeries[Time: 1.65 to 2.25 Data points: 301]	TimeSer
−1 539 280.	7 357 898.	1.96452	TimeSeries[Time: 1.65 to 2.25 Data points: 301]	TimeSer
−1 539 266.	7 357 866.	1.96496	TimeSeries[Time: 1.65 to 2.25 Data points: 301]	TimeSer
−1 539 312.	7 357 884.	1.9656	TimeSeries[Time: 1.65 to 2.25 Data points: 301]	TimeSer
−1 539 257.	7 357 889.	1.96411	TimeSeries[Time: 1.65 to 2.25 Data points: 301]	TimeSer
−1 539 298.	7 357 852.	1.96575	TimeSeries[Time: 1.65 to 2.25 Data points: 301]	TimeSer
−1 529 078.	7 363 224.	1.97355	TimeSeries[Time: 1.65 to 2.25 Data points: 301]	TimeSer
−1 529 055.	7 363 215.	1.97298	TimeSeries[Time: 1.65 to 2.25 Data points: 301]	TimeSer
−1 529 069.	7 363 247.	1.97354	TimeSeries[Time: 1.65 to 2.25 Data points: 301]	TimeSer
−1 529 046.	7 363 238.	1.97311	TimeSeries[Time: 1.65 to 2.25 Data points: 301]	TimeSer
−1 529 064.	7 363 192.	1.97305	TimeSeries[Time: 1.65 to 2.25 Data points: 301]	TimeSer
−1 529 022.	7 363 228.	1.97264	TimeSeries[Time: 1.65 to 2.25 Data points: 301]	TimeSer

−1533354.	7368921.	1.96055	TimeSeries []	TimeSer
−1533345.	7368944.	1.96153	TimeSeries []	TimeSer
−1533377.	7368930.	1.96	TimeSeries []	TimeSer
−1533331.	7368912.	1.96116	TimeSeries []	TimeSer
−1533322.	7368935.	1.96211	TimeSeries []	TimeSer
−1533368.	7368953.	1.96095	TimeSeries []	TimeSer
−1536881.	7365569.	1.94215	TimeSeries []	TimeSer
−1536890.	7365546.	1.94165	TimeSeries []	TimeSer

showing 1–20 of 180

Тренировка

Специальная функция для тестирование обчения

In[ ]:=
```
checkPredictor[data_List, attrs: {__String}, key_String, {n_Integer, m_Integer}, method_Strin
Block[{$trainDataTest, $trainSample, $trainCheck, $predictorTest,
    $data = Query[All, Append[attrs, key]] @ data},
    Table[
        $trainSample = RandomSample[$data, n];
        $trainCheck = Complement[$data, $trainSample];
        Clear[$predictorTest];
        $predictorTest = Predict[$trainSample -> key, Method -> method];
        Transpose[{$trainCheck[[All, key]], $predictorTest[$trainCheck]}],
        {m}
    ]
]
```

Список атрибутов на котором будет производиться обучение

In[ ]:=
```
trainAttrs = {"t",
    "RMS(0ms)", "RMS(10ms)", "RMS(20ms)",
    "RMS(0mscwt8)", "RMS(0mscwt16)", "RMS(0mscwt20)",
    "RMS(10mscwt8)", "RMS(10mscwt16)", "RMS(10mscwt20)",
    "RMS(20mscwt8)", "RMS(20mscwt16)", "RMS(20mscwt20)"
};
```

Тренировка на случайной выборке 25/30 трасс

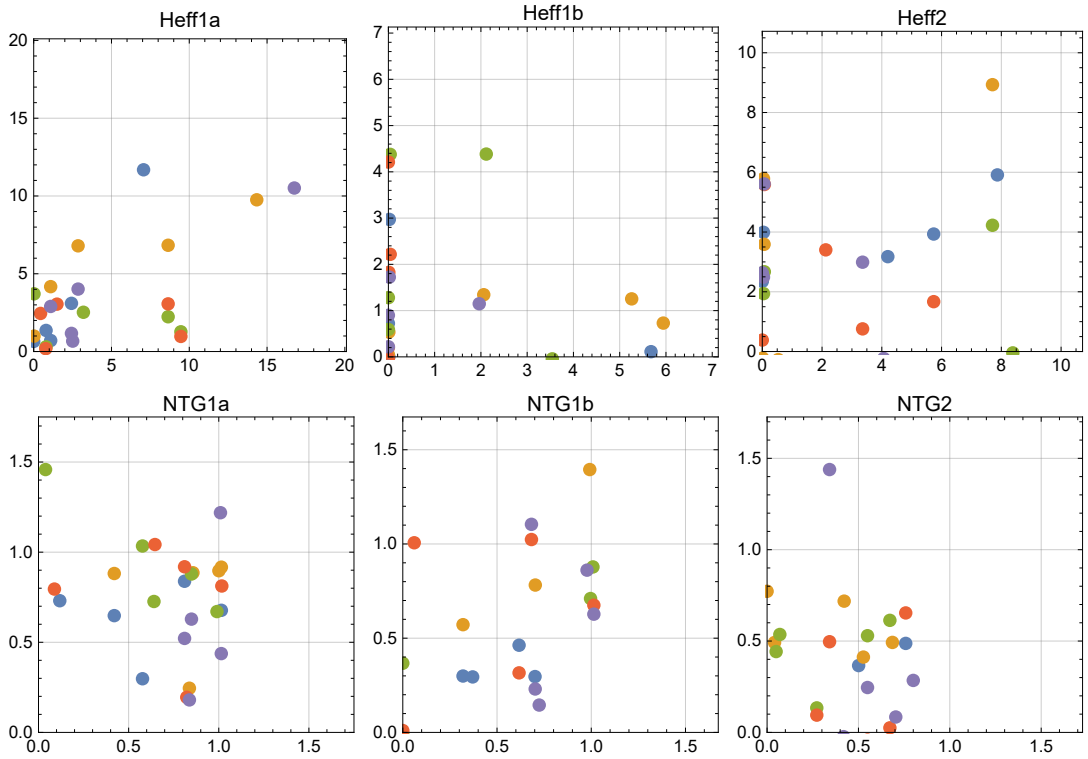
Нейросеть восстанавливает данные 5 раз на разных выборках

In[ ]:=
```
checkRes = Association[Table[
    key ->
    checkPredictor[trainDataset[[1 ;; -1 ;; 6]], trainAttrs, key, {25, 5},
        "NeuralNetwork"],
    {key, {"Heff1a", "Heff1b", "Heff2", "NTG1a", "NTG1b", "NTG2"}}
]];
```

Построим результат

In[ ]:=
```
Grid[ArrayReshape[Table[ListPlot[checkRes[key],
    ImageSize -> Small,
    Frame -> True,
    PlotRange -> {{0, 1.2Max[Flatten[checkRes[key]]]},
        {0, 1.2Max[MinMax[Flatten[checkRes[key]]]]}},
    GridLines -> Automatic,
    AspectRatio -> 1,
    PlotStyle -> PointSize[Large],
    PlotLabel -> key
], {key, Keys[checkRes]}], {2, 3}]]
```

Out[ ]=



Посмотрим на корреляцию каждой выборки

In[ ]:=
```
Dataset[Map[Association,
    Transpose[Map[Thread, Normal[Map[Transpose /*
    Apply[Correlation]] /@ checkRes]]]
]]
```

Out[ ]=

Heff1a	Heff1b	Heff2	NTG1a	NTG1b	NTG2
0.987855	−0.525686	0.730428	0.0707037	−0.433943	0.890087
0.871857	0.550772	0.735079	0.00826604	0.016425	−0.468106
−0.206603	−0.201788	−0.210088	−0.921674	0.91201	0.339783
0.043051	0.19063	−0.333551	−0.157041	0.261543	0.186578
0.933074	0.295049	−0.639318	0.530521	0.24421	−0.528443

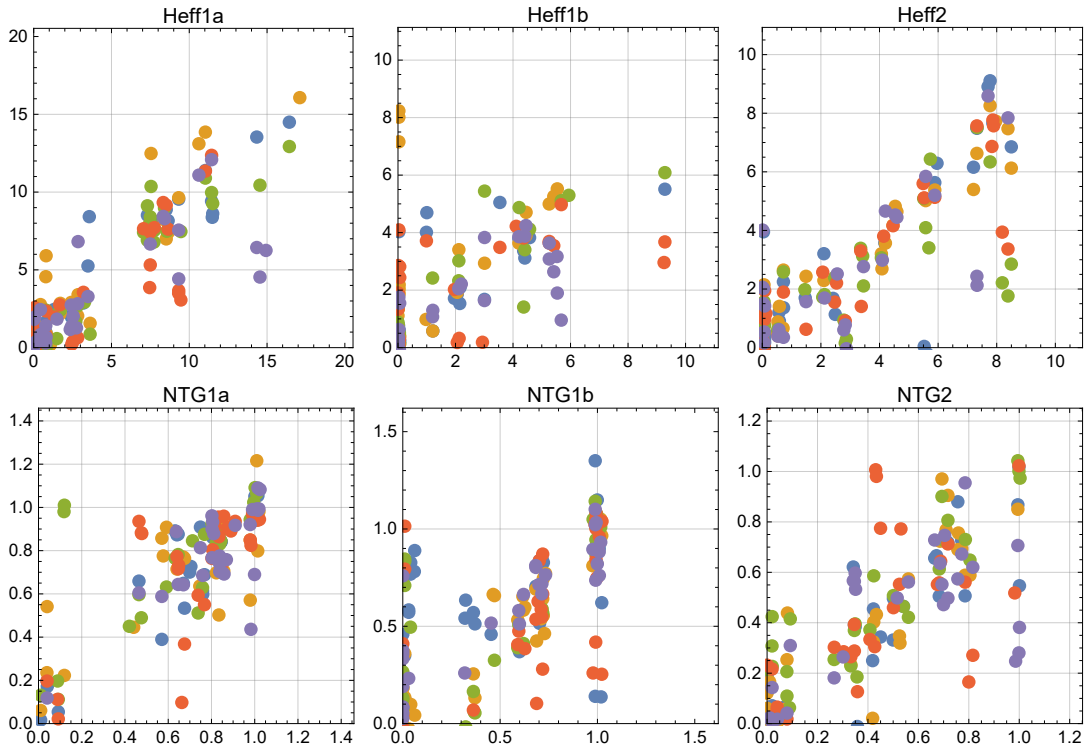
Теперь все тоже самое, но выберем побольше трасс

In[ ]:=
```
checkRes2 = Association[Table[
    key ->
    checkPredictor[trainDataset[[1 ;; -1 ;; 2]], trainAttrs, key,
        {60, 5}, "NeuralNetwork"],
    {key, {"Heff1a", "Heff1b", "Heff2", "NTG1a", "NTG1b", "NTG2"}}
]];
```

Картинки

In[ ]:=
```
Grid[ArrayReshape[Table[ListPlot[checkRes2[key],
    ImageSize -> Small,
    Frame -> True,
    PlotRange -> {{0, 1.2Max[Flatten[checkRes2[key]]]},
        {0, 1.2Max[MinMax[Flatten[checkRes2[key]]]]}},
    GridLines -> Automatic,
    AspectRatio -> 1,
    PlotStyle -> PointSize[Large],
    PlotLabel -> key
], {key, Keys[checkRes2]}], {2, 3}]]
```

Out[ ]=



Местами невероятная корреляция

In[•]:=
```
Dataset[Map[Association, Transpose[Map[Thread,
Normal[Map[Transpose /* Apply[Correlation]] /@ checkRes2]]]]]
```

Out[•]=

Heff1a	Heff1b	Heff2	NTG1a	NTG1b	NTG2
0.956702	0.737285	0.760024	0.918898	0.313861	0.810774
0.923712	0.4664	0.955107	0.824804	0.915616	0.848662
0.940584	0.505324	0.684241	0.68604	0.729	0.898872
0.829969	0.582599	0.883156	0.730327	0.433423	0.613969
0.783515	0.796504	0.73994	0.828887	0.880263	0.682619

Что ж...

Попробуем натренировать сеть...

In[•]:=
```
ClearAll[predictors];
predictors = Association[Table[key ->
    Predict[trainDataset[[All, Append[trainAttrs, key]]] -> key,
        Method -> "NeuralNetwork"],
{key, {"Heff1a", "Heff1b", "Heff2", "NTG1a", "NTG1b", "NTG2"}}]]
```

Out[•]=
```
⟨|Heff1a → PredictorFunction[ Input type: Mixed (number: 13)
                              Method: NeuralNetwork ],
 Heff1b → PredictorFunction[ Input type: Mixed (number: 13)
                             Method: NeuralNetwork ],
 Heff2 → PredictorFunction[ Input type: Mixed (number: 13)
                            Method: NeuralNetwork ],
 NTG1a → PredictorFunction[ Input type: Mixed (number: 13)
                            Method: NeuralNetwork ],
 NTG1b → PredictorFunction[ Input type: Mixed (number: 13)
                            Method: NeuralNetwork ],
 NTG2 → PredictorFunction[ Input type: Mixed (number: 13)
                           Method: NeuralNetwork ]|⟩
```

... и применить к тестовым данным:

In[ ]:=
```
Dataset[testDataset = Query[All, 1 ;; -7] @ DataQuery @ WellTraces[cube, horiz, testTable]]
```

Out[ ]=

x	y	t	trace	window
-1539289.	7357875.	1.96537	TimeSeries[Time: 1.65 to 2.25 Data points: 301]	TimeSerie
-1535541.	7362506.	1.95689	TimeSeries[Time: 1.65 to 2.25 Data points: 301]	TimeSerie
-1542818.	7358575.	1.98516	TimeSeries[Time: 1.65 to 2.25 Data points: 301]	TimeSerie
-1539377.	7365369.	1.93811	TimeSeries[Time: 1.65 to 2.25 Data points: 301]	TimeSerie
-1532053.	7360508.	1.97382	TimeSeries[Time: 1.65 to 2.25 Data points: 301]	TimeSerie
-1537471.	7367766.	1.96148	TimeSeries[Time: 1.65 to 2.25 Data points: 301]	TimeSerie
-1526459.	7364394.	1.97138	TimeSeries[Time: 1.65 to 2.25 Data points: 301]	TimeSerie
-1539189.	7371089.	1.9338	TimeSeries[Time: 1.65 to 2.25 Data points: 301]	TimeSerie

Результат

In[ ]:=
```
Dataset[result = MapThread[
    Association[Join[#1, #2]]&, {
        Normal[testTable[[All, {"x", "y"}]]],
        Transpose[Table[Thread[key ->
            predictors[key][testDataset]], {key, Keys[predictors]}]]
    }
]]
```

Out[ ]=

x	y	Heff1a	Heff1b	Heff2	NTG1a	NTG1b
−1 539 284.	7 357 878.	−0.187327	1.04766	4.37168	0.460234	0.482812
−1 535 537.	7 362 509.	1.2479	1.24062	4.18475	0.482495	0.988867
−1 542 815.	7 358 588.	1.84826	3.60372	1.23574	1.04567	0.0847745
−1 539 369.	7 365 372.	1.15101	0.214173	0.498101	0.856533	0.0336066
−1 532 044.	7 360 515.	0.171697	1.66847	1.90643	0.639741	0.779828
−1 537 476.	7 367 782.	4.52852	2.70697	1.93935	0.830215	0.659242
−1 526 450.	7 364 392.	0.787204	2.43255	3.03887	0.875112	0.322729
−1 539 191.	7 371 095.	4.30582	5.25369	−0.315776	0.745791	0.407852

Ответ

In[ ]:=
```
Dataset[answer =
Normal[First[Import["Heff+NTG_test_answer.xlsx", "Dataset", "HeaderLines" -> 1]]]]
```

Out[ ]=

x	y	Heff1a	Heff1b	Heff2	NTG1a	NTG1b
−1 539 284.	7 357 878.	0.18	0.99	4.12	0.47	0.46
−1 535 537.	7 362 509.	1.42	0.01	3.29	1.0	1.0
−1 542 815.	7 358 588.	0.01	0.0	2.9	0.06	0.0
−1 539 369.	7 365 372.	0.2	0.0	0.09	0.5	0.0
−1 532 044.	7 360 515.	0.01	2.4	9.99	0.07	1.0
−1 537 476.	7 367 782.	4.05	4.36	4.19	0.8	0.57
−1 526 450.	7 364 392.	1.1	0.01	4.09	1.0	0.0
−1 539 191.	7 371 095.	0.15	3.37	0.98	0.27	0.51

Ошибка

```
Total[MapThread[Total[((#1 - #2))^2] / Length[#1]&,
{Values[answer[[All, 3 ;; 5]]], Values[result[[All, 3 ;; 5]]]}]] / Length[answer]
```

Out[ ]= 5.26253

И сохраним результат

In[ ]:=
```
Export["Heff+NTG_test_result.xlsx", Dataset[result]]
```

Out[ ]= Heff+NTG_test_result.xlsx