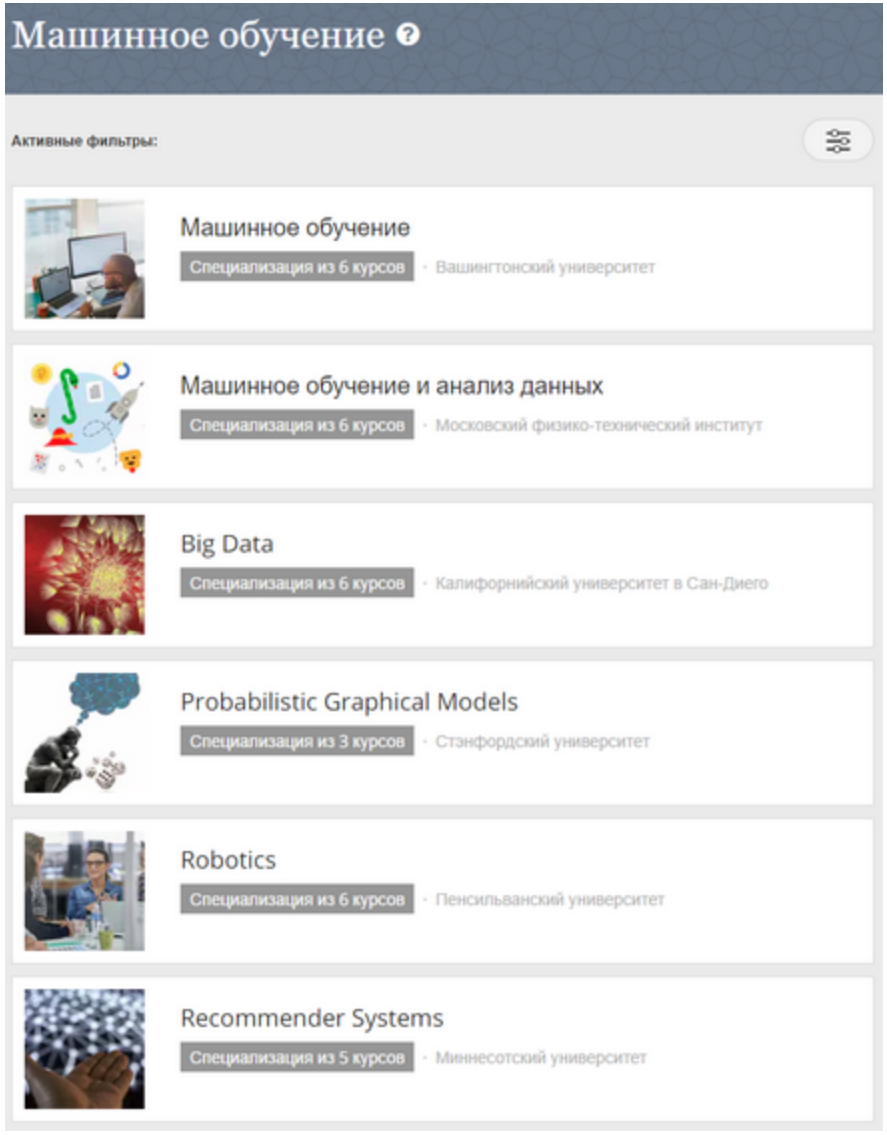
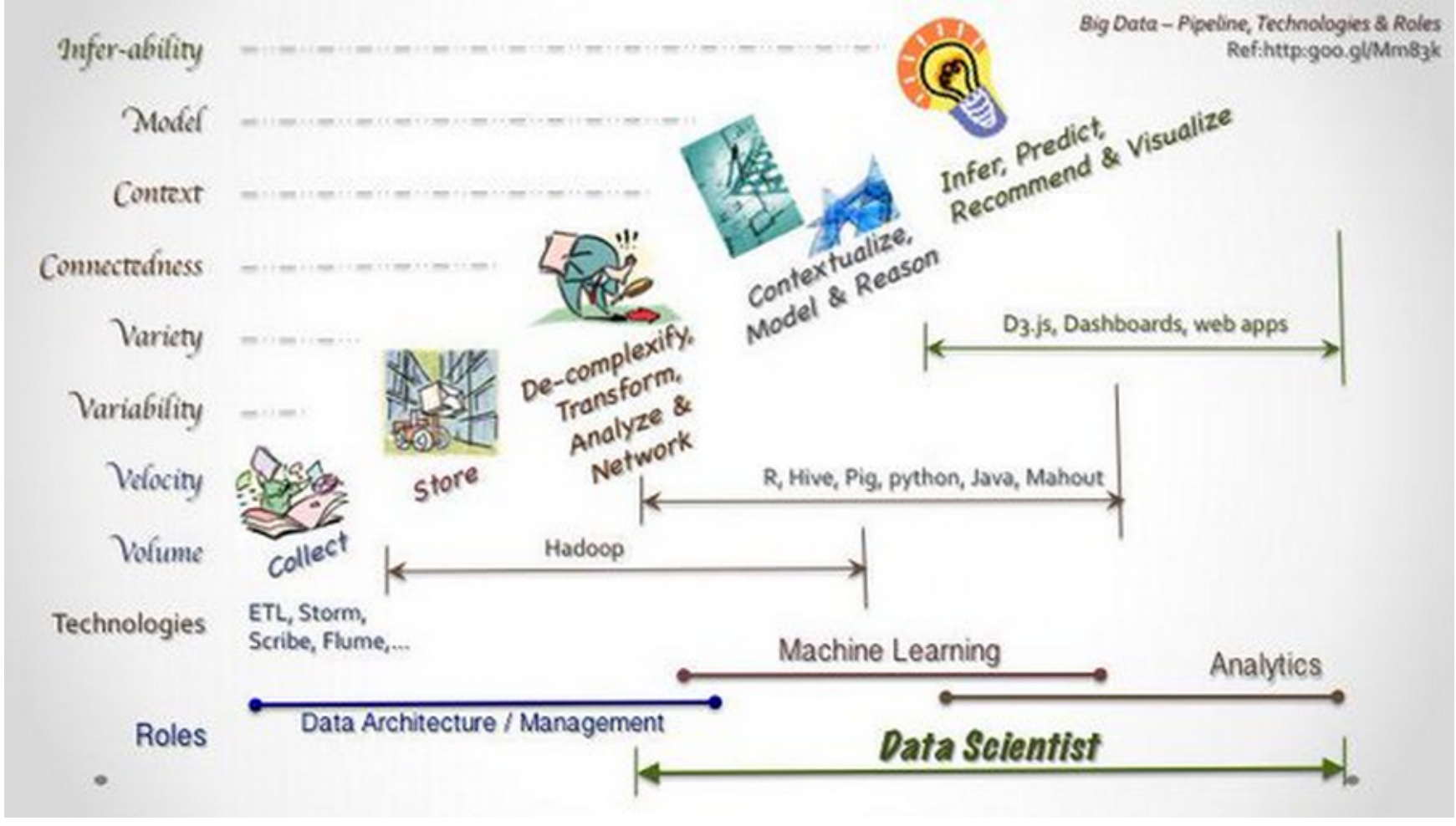
Big Data (Wikipedia's definition)

- **Volume**: big data doesn't sample; it just observes and tracks what happens
- **Velocity**: big data is often available in real-time
- **Variety**: big data draws from text, images, audio, video; plus it completes missing pieces through data fusion
- ***Machine Learning***: big data often doesn't ask why and simply detects patterns
- ***Digital footprint***: big data is often a cost-free byproduct of digital interaction

Big Data on Coursera



Pipelines, Technologies & Roles



Spark

spark-notebook.io

```
import org.apache.spark.sql.SQLContext
import org.apache.spark.mllib.rdd.RDDFunctions._
import scala.util.{Try, Success, Failure}
import java.sql.Timestamp

val sqlContext = new SQLContext(sparkContext)
import sqlContext.implicits._
```

TradingView Ideas on BTCUSD

4000 опубликованных идей получены примерно так:

```
curl -s "https://www.tradingview.com/chart/
                ?stream=bitcoin&time=all&s=0&l=1000"

curl -s "https://www.tradingview.com/chart/
                ?stream=bitcoin&time=all&s=1000&l=1000"

curl -s "https://www.tradingview.com/chart/
                ?stream=bitcoin&time=all&s=2000&l=1000"

curl -s "https://www.tradingview.com/chart/
                ?stream=bitcoin&time=all&s=3000&l=1000"
```

Parse Time

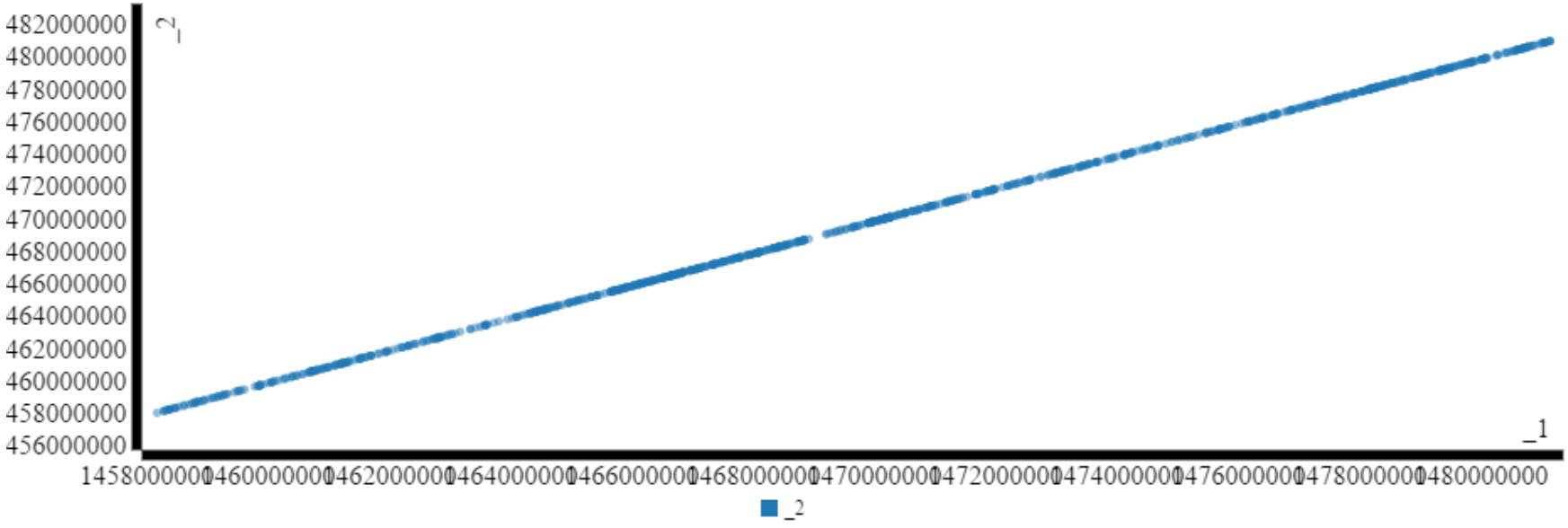
```
val btcIdeas =
          sparkContext.textFile("data/1000_1_BTC_ideas.json")
  .union(sparkContext.textFile("data/1000_2_BTC_ideas.json"))
  .union(sparkContext.textFile("data/1000_3_BTC_ideas.json"))
  .union(sparkContext.textFile("data/1000_4_BTC_ideas.json"))
```

```
import java.lang.Double
val times = btcIdeas
  .flatMap(_.split("data-timestamp"))
  .map(_.split("\\\"")(1))
  .map(x => x.substring(0, x.length - 1))
  .flatMap(x => Try(Double.parseDouble(x)).toOption)
```

```
times: org.apache.spark.rdd.RDD[Double] = MapPartitionsRDD
```

Идеи по BTCUSD на шкале времени

```
ScatterChart(times.map(x => (x,x)).collect)
```



Время первой и последней

```
val t = Seq(times.collect.min, times.collect.max)
  .map(x => new Timestamp(x.toLong * 1000));
```

```
t: Seq[java.sql.Timestamp] =
      List(2014-07-28 19:30:18.0, 2016-12-07 19:59:13.0)
```

Yahoo News

В `data/yahoo.txt` - манипуляциями с html в текстовом редакторе собраны новости по BTCUSD в виде:

```
http://finance.yahoo.com/news/exclusive-mona-el-isa-google-1716
http://finance.yahoo.com/news/exclusive-mona-el-isa-google-1614
http://finance.yahoo.com/news/payments-marijuana-industry-block
http://finance.yahoo.com/news/youre-money-ny-judge-rules-162025
...
```

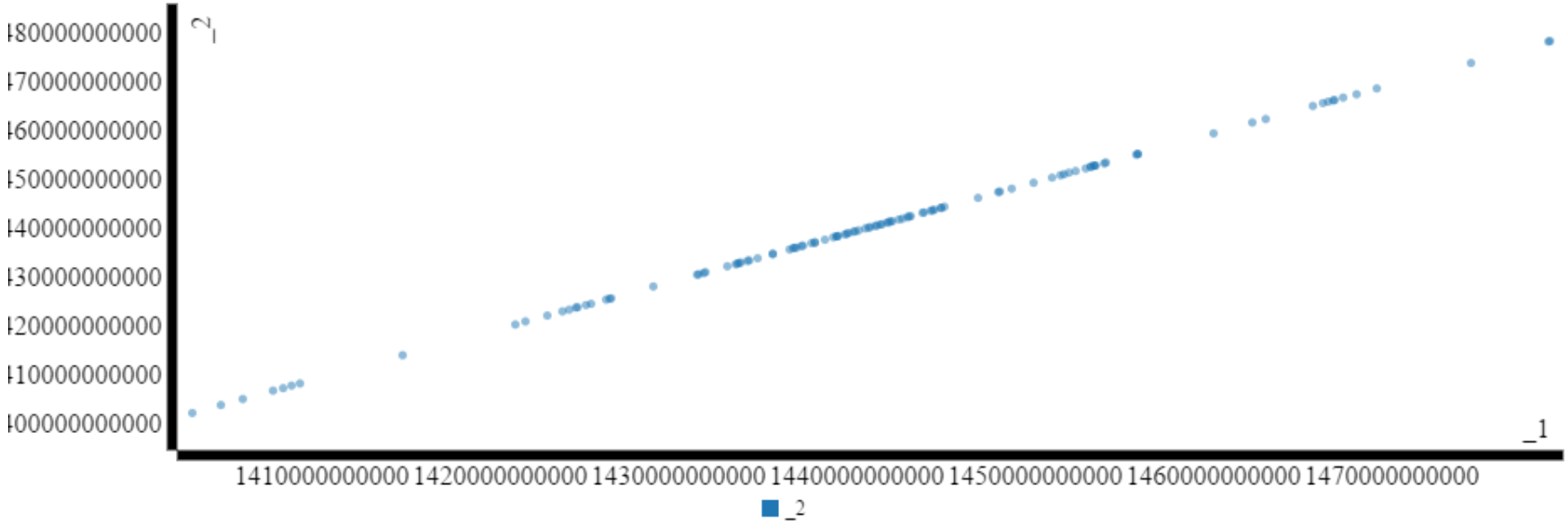
```
val btcNewsUrls = sparkContext.textFile("data/yahoo.txt")
```

Parse Time

```
import java.text.SimpleDateFormat
val dateFormatter =
        new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss.SSS'Z'")
val yahooTimes = btcNewsUrls
  .flatMap(x => Try{
    val s = scala.io.Source.fromURL(x, "utf-8")
    val res = s.getLines
                .filter(_.contains("datetime"))
                .take(1)
                .toList
    s.close()
    res
  }.toOption)
  .flatMap(x => x)
  .flatMap(x => "datetime=\"(.*?)\"".r
                .findFirstMatchIn(x)
                .map(_.group(1)))
  .map(dateFormatter.parse(_))
  .map(_.getTime)
  .collect
```

Yahoo новости по BTCUSD на шкале времени

```
ScatterChart(yahooTimes.map(x => (x,x)))
```



Время первой и последней

```
val t = Seq(yahooTimes.min, yahooTimes.max)
  .map(x => new Timestamp(x.toLong));
```

```
t: Seq[java.sql.Timestamp] =
      List(2014-06-13 18:02:44.0, 2016-11-11 17:16:02.0)
```

Сам BTCUSD в это время

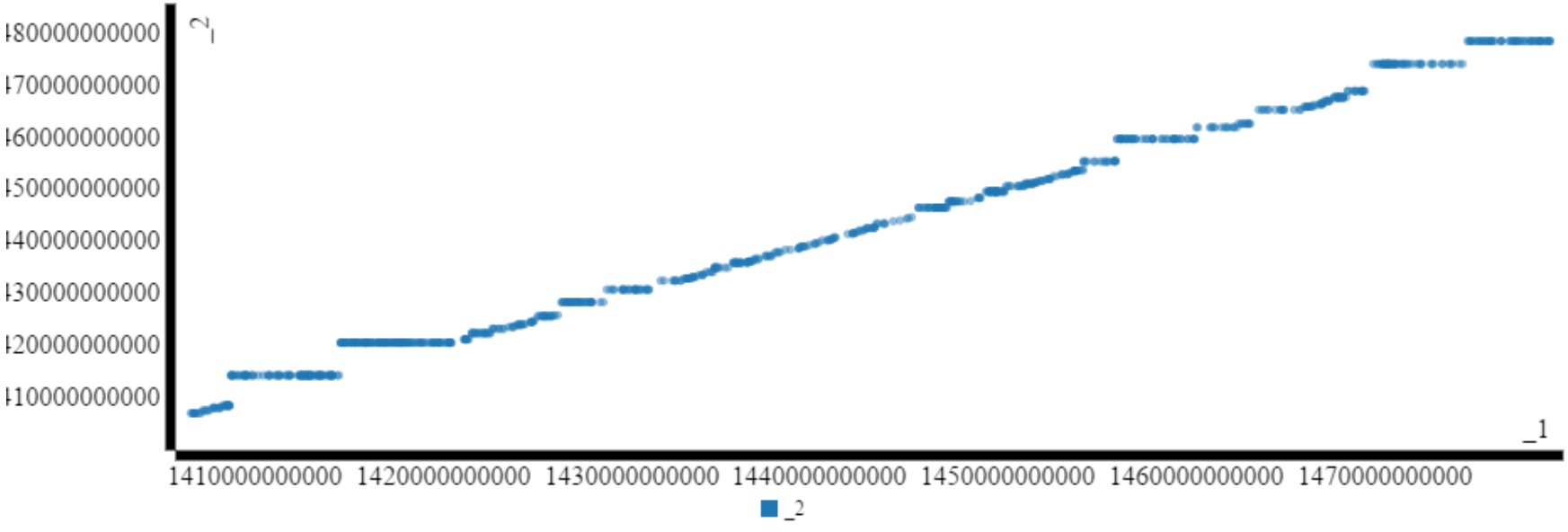
Compare Ideas and News

Приведём к одному формату

```
val newsTimes = yahooTimes.sorted
val ideasTimes = times.map(_ * 1e3).collect.sorted
```

Выведем новости и идеи на одном графике.

```
val both = for {
  newsPair <- newsTimes.sliding(2)
  ti <- ideasTimes if ti >= newsPair(0) && ti < newsPair(1)
} yield (ti, newsPair(1))
// select only 22% to be able to plot scatter chart
ScatterChart(both.filter(x => scala.math.random > 0.78).toSeq)
```



Рост вверх - новость, рост вправо - идея.

Statistics

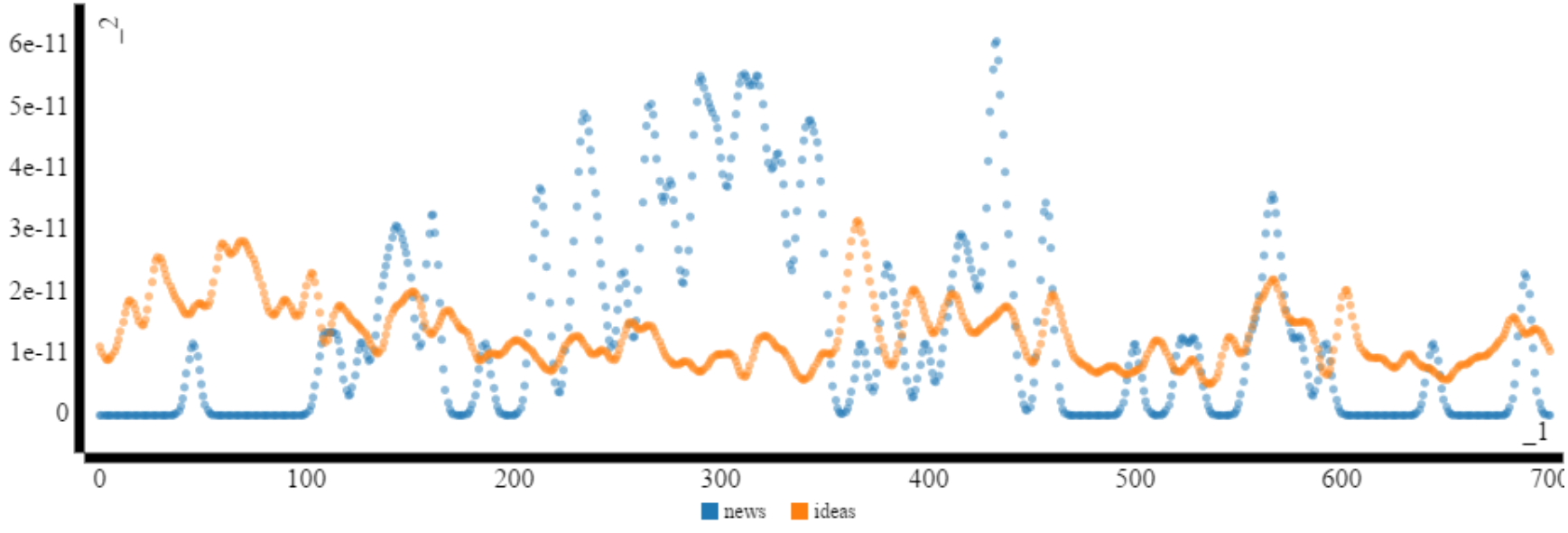
Сравнивать точки во времени не совсем понятно как.

Будем сравнивать плотности вероятностей.

```
import org.apache.spark.mllib.stat.KernelDensity
import org.apache.spark.rdd.RDD

def myKernelDensity(arr: Array[Double]): KernelDensity = {
  val data = sc.parallelize(arr)
  new KernelDensity()
    .setSample(data)
    .setBandwidth(3e8)
}

val kdi = myKernelDensity(ideasTimes)
val kdn = myKernelDensity(newsTimes)

// Find density estimates for the given values
// 14100 to 14800 * 1e8 - time: 06 Sep 2014 - 24 Nov 2016
val densitiesi = kdi.estimate(
                    (14100 to 14800).map(_ * 1e8).toArray)
val densitiesn = kdn.estimate(
                    (14100 to 14800).map(_ * 1e8).toArray)
```

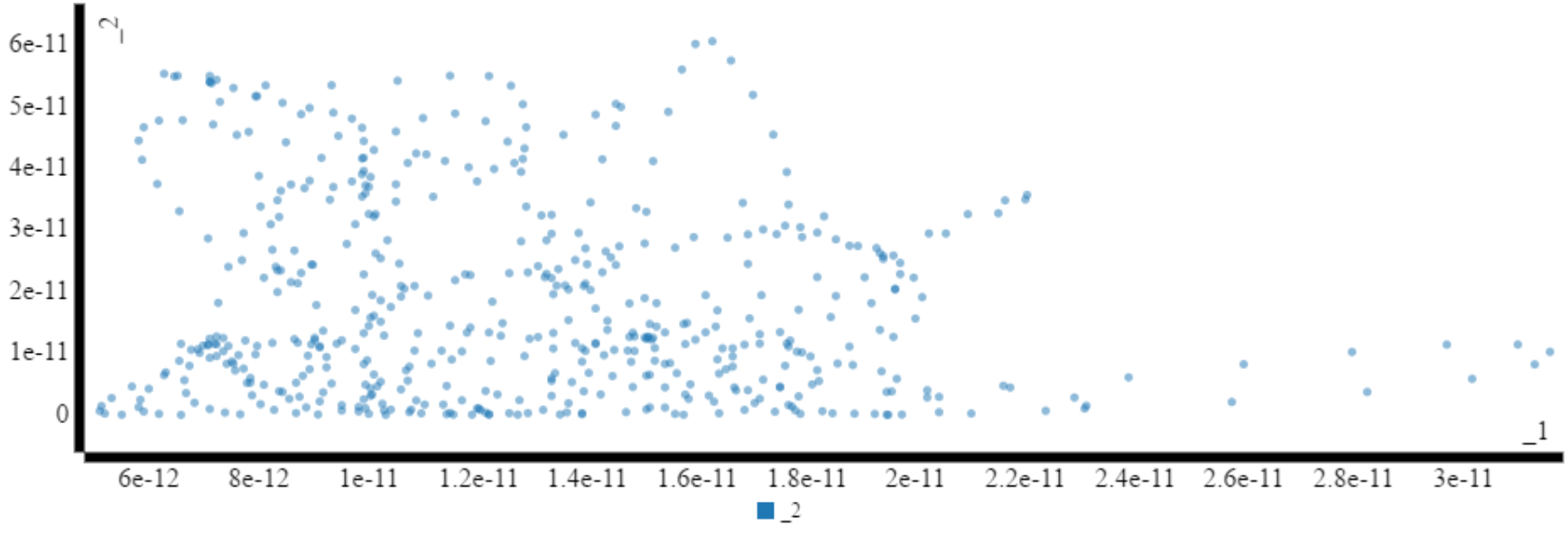
На графике

```
val newsAndIdeasForChart = (densitiesn zip densitiesi)
  .zipWithIndex
  .flatMap{
    case ((x,y), i) => Seq((i, x, "news"), (i, y, "ideas"))
  }
ScatterChart(newsAndIdeasForChart,
             fields=Some(("_1", "_2")),
             groupField=Some("_3"),
             maxPoints=2000)
```



В другом виде

```
ScatterChart(
  (densitiesi zip densitiesn).filter(_._2 >= 1e-13)
)
```



Correlation

```
import org.apache.spark.mllib.linalg._
import org.apache.spark.mllib.stat.Statistics
```

```
val correlation: Double = Statistics.corr(
  sc.parallelize(densitiesi),
  sc.parallelize(densitiesn),
  "spearman"
)
```

```
correlation: Double = -0.10929748013913299
```

Отбрасывая историю

Отбросим начало истории, $2/7$

```
val correlation: Double = Statistics.corr(
  sc.parallelize(densitiesi drop 200),
  sc.parallelize(densitiesn drop 200),
  "spearman"
)
```

```
correlation: Double = 0.12755330772717519
```

Уже лучше. Отбросим ещё $2/7$

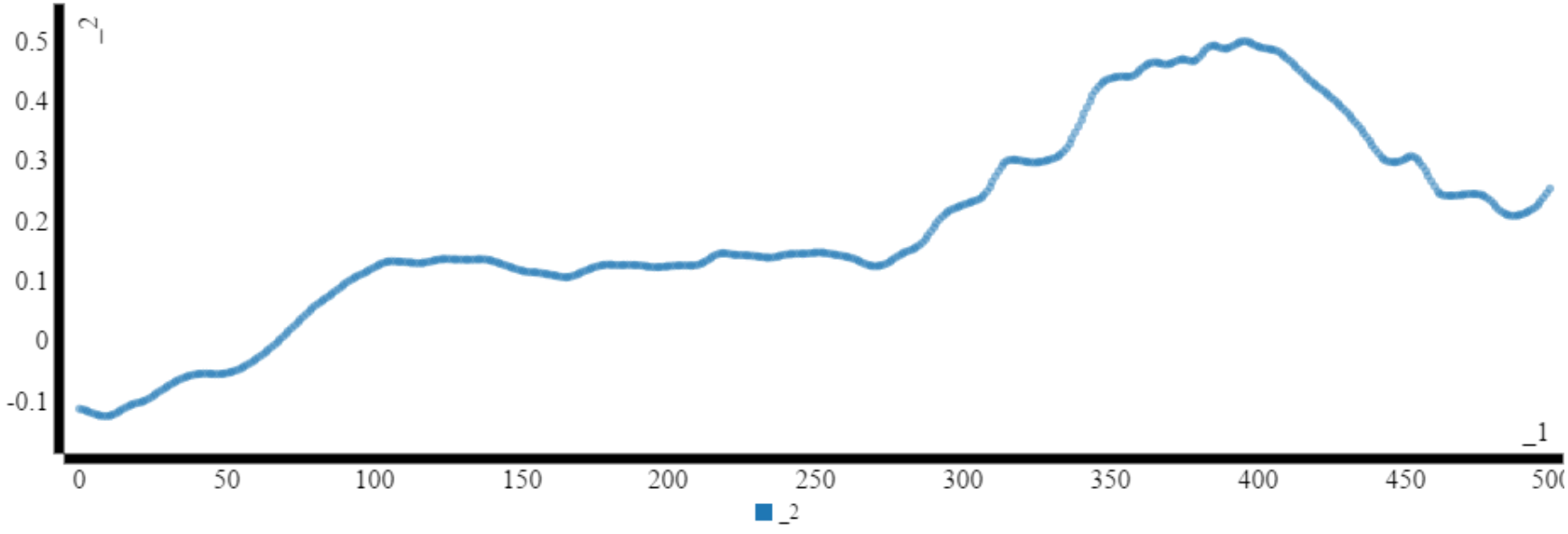
```
val correlation: Double = Statistics.corr(
  sc.parallelize(densitiesi drop 400),
  sc.parallelize(densitiesn drop 400),
  "spearman"
)
```

```
correlation: Double = 0.4965642120085375
```

Отбрасывая историю (2)

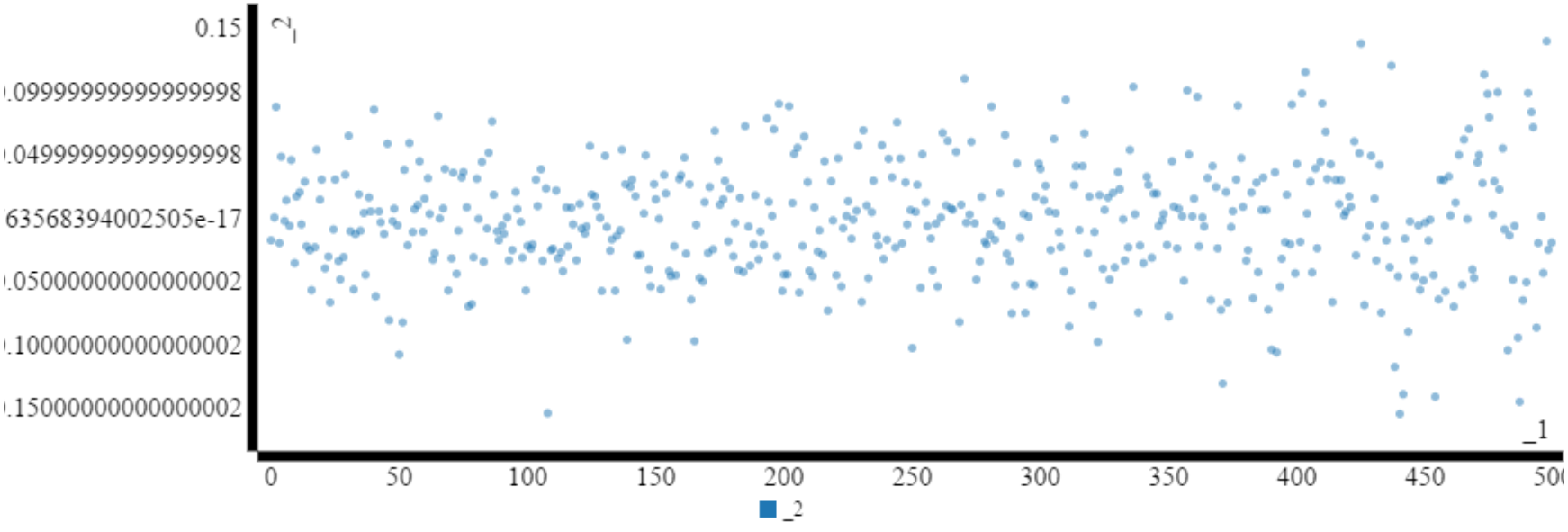
```
val correlations = for (i <- 1 to 500) yield Statistics.corr(
    sc.parallelize(densitiesi drop i),
    sc.parallelize(densitiesn drop i),
    "spearman"
  )
```

```
ScatterChart(correlations.toArray)
```



Убедимся, что мы не сильно обманываемся (?)

```
import scala.util.Random
val correlationsBad = for (i <- 1 to 500) yield Statistics.corr
    sc.parallelize(densitiesi drop i),
    sc.parallelize(Random.shuffle(densitiesi drop i)),
    "spearman"
  )
ScatterChart(correlationsBad.toArray)
```



Overfitting