条形

聚合 有一个令人激 的特性就是能 十分容易地将它 成 表和 形。本章中， 我 正在通 示例数据来来完成各 各 的聚合分析，最 ，我 将会 聚合功能是非常 大的。

直方 histogram 特 有用。 它本 上是一个条形 ，如果有 建 表或分析 表 的 ，那 我会 无疑 的 里面有一些 表是条形 。 建直方 需要指定一个区 ，如果我 要 建一个直方 ，可以将 隔 20,000。 做将会在 个 $20,000 建一个新桶，然后文 会被分到 的桶中。

于 表 来 ，我 希望知道 个 区 内汽 的 量。我 会想知道 个 区 内汽 所 来的收入，可以通 个区 内已 汽 的 求和得到。

可以用 histogram 和一个嵌套的 sum 度量得到我 想要的答案：

```
GET /cars/transactions/_search
{
   "size" : 0,
   "aggs":{
      "price":{
         "histogram":{ ①
            "field": "price",
            "interval": 20000
         },
         "aggs":{
            "revenue": {
               "sum": { ②
                 "field" : "price"
               }
             }
         }
      }
   }
}
```

① histogram 桶要求 个参数：一个数 字段以及一个定 桶大小 隔。

② sum 度量嵌套在 个 区 内，用来 示 个区 内的 收入。

如我 所 ， 是 price 聚合 建的，它包含一个 histogram 桶。它要求字段的 型必 是数 型的同 需要 定分 的 隔 。 隔 置 20,000 意味着我 将会得到如 [0-19999, 20000-39999, …] 的区 。

接着，我 在直方 内定 嵌套的度量， 个 sum 度量，它会 落入某一具体 区 的文 中 price 字段的 行求和。 可以 我 提供 个 区 的收入，从而可以 到底是普通家用 是奢 。

果如下：

```
{
...
   "aggregations": {
      "price": {
         "buckets": [
            {
               "key": 0,
               "doc_count": 3,
               "revenue": {
                  "value": 37000
               }
            },
            {
               "key": 20000,
               "doc_count": 4,
               "revenue": {
                  "value": 95000
               }
            },
            {
               "key": 80000,
               "doc_count": 1,
               "revenue": {
                  "value": 80000
               }
            }
         ]
      }
   }
}
```

果很容易理解，不　　注意到直方　的　　是区　的下限。　　0　代表区　　0-19, 999　，　　20000　代表区　20, 000-39, 999，等等。

NOTE

我　可能会注意到空的区　，比如：$40，000-60，000，没有出　在　　中。　histogram　桶　　会忽略它，因　它有可能会　致不希望的潜在　　出。

我　会在下一小　中　　如何包括空桶。返回空桶 返回空 Buckets 。

可以在　Sales and Revenue per price bracket 中看到以上数据直方　的　形化表示。

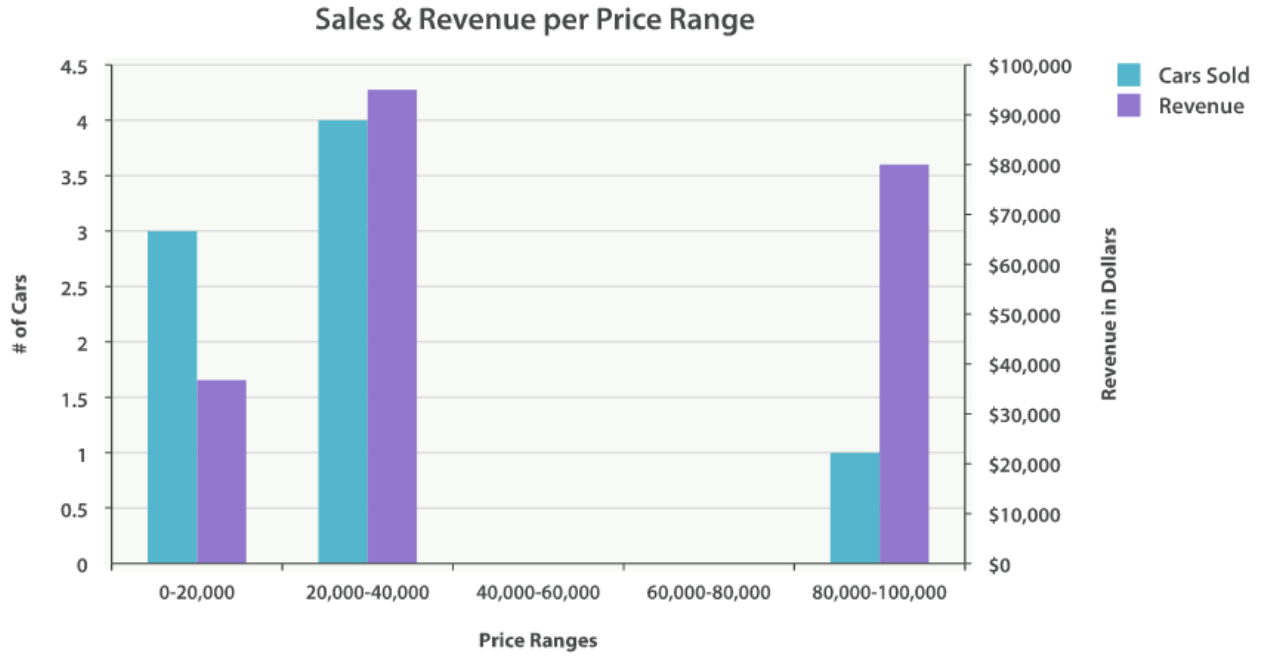

Figure 1. Sales and Revenue per price bracket

当然，我 可以 任何聚合 出的分 和 果 建条形 ，而不只是 直方 桶。 我 以最受 迎 10 汽 以及它 的平均 、 准差 些信息 建一个条形 。 我 会用到 `terms` 桶和 `extended_stats` 度量：

```
GET /cars/transactions/_search
{
  "size" : 0,
  "aggs": {
    "makes": {
      "terms": {
        "field": "make",
        "size": 10
      },
      "aggs": {
        "stats": {
          "extended_stats": {
            "field": "price"
          }
        }
      }
    }
  }
}
```

上述代 会按受 迎度返回制造商列表以及它 各自的 信息。我 其中的 `stats.avg` 、 `stats.count` 和 `stats.std_deviation` 信息特 感 趣，并用 它 算出 准差：

```
std_err = std_deviation / count
```

建 表如 Average price of all makes, with error bars 。

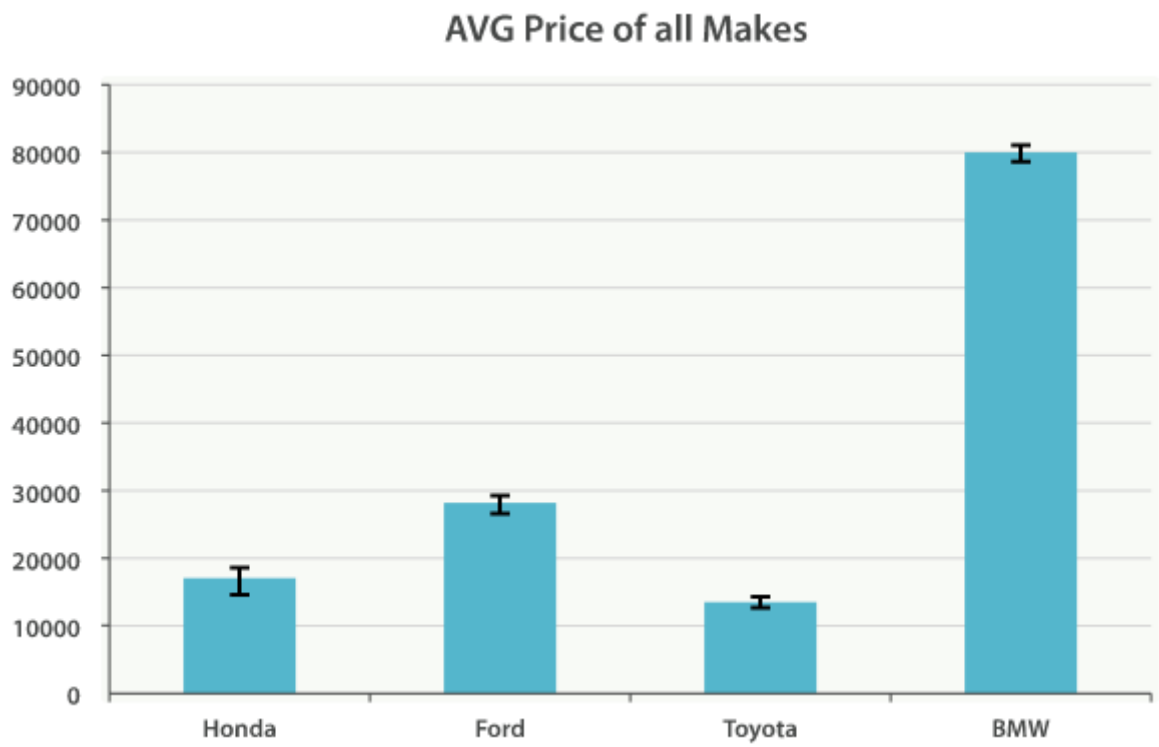

Figure 2. Average price of all makes, with error bars

按

如果搜索是在 Elasticsearch 中使用 率最高的，那 建按 的 date_histogram 随其后。什 会想用 date_histogram ？

假 的数据 。 无 是什 数据（Apache 事件日志、股票 交易 、棒球 ）只要有 都可以 行 date_histogram 分析。当 的数据有 ， 是想在 度上 建指分析：

- 今年 月 多少台汽 ？
- 只股票最近 12 小 的 格是多少？
- 我 站上周 小 的平均 延 是多少？

然通常的 histogram 都是条形 ，但 date_histogram 向于 成 状 以展示 序列。 多公司用 Elasticsearch 只是 了分析 序列数据。`date_histogram` 分析是它 最基本的需要。

`date_histogram` 与 通常的 `histogram` 似。 但不是在代表数 的数 字段上 建 buckets ，而是在 上 建 buckets。 因此 一个 bucket 都被定 成一个特定的日期大小 (比如， `1个月` 或 `2.5 天`)。

可以用通常的 **histogram** 行 分析 ？

从技 上来 ，是可以的。 通常的 `histogram` bucket（桶）是可以 理日期的。但是它不能自 日期。 而用 `date_histogram` ， 可以指定 段如 `1 个月` ，它能 明地知道 2 月的天数比 12 月少。 `date_histogram` 具有 外一个 ，即能合理地 理 区， 可以使用客 端的 区 行 定制，而不是用服 器端 区。

通常的 histogram 会把日期看做是数字， 意味着 必 以微秒 位指明 隔。 外聚合并不知道日 隔，使得它 于日期而言几乎没什 用 。

我 的第一个例子将 建一个 的折 来回答如下 ： 月 多少台汽 ？

```
GET /cars/transactions/_search
{
   "size" : 0,
   "aggs": {
      "sales": {
         "date_histogram": {
            "field": "sold",
            "interval": "month", ①
            "format": "yyyy-MM-dd" ②
         }
      }
   }
}
```

① 隔要求是日 (如 个 bucket 1 个月)。

② 我 提供日期格式以便 buckets 的 便于 。

我 的 只有一个聚合， 月 建一个 bucket。 我 可以得到 个月 的汽 数量。 外 提供了一个 外的 `format` 参数以便 buckets 有 "好看的" 。 然而在内部，日期 然是被表示成数 。 可能会使得 UI 者抱怨，因此可以提供常用的日期格式 行格式化以更方便 。

果既符合 期又有一点出人意料（看看 是否能 到意外之 ）：

```
{
   ...
   "aggregations": {
      "sales": {
         "buckets": [
            {
               "key_as_string": "2014-01-01",
               "key": 1388534400000,
               "doc_count": 1
            },
            {
               "key_as_string": "2014-02-01",
               "key": 1391212800000,
               "doc_count": 1
            },
            {
               "key_as_string": "2014-05-01",
               "key": 1398902400000,
               "doc_count": 1
            },
            {
               "key_as_string": "2014-07-01",
               "key": 1404172800000,
               "doc_count": 1
            },
            {
               "key_as_string": "2014-08-01",
               "key": 1406851200000,
               "doc_count": 1
            },
            {
               "key_as_string": "2014-10-01",
               "key": 1412121600000,
               "doc_count": 1
            },
            {
               "key_as_string": "2014-11-01",
               "key": 1414800000000,
               "doc_count": 2
            }
         ]
...
}
```

聚合 果已 完全展示了。正如 所 ，我 有代表月 的 buckets， 个月的文 数目，以及美化后的 `key_as_string` 。

返回空 Buckets

注意到 果末尾 的奇怪之 了 ？

是的， 果没 。 我 的 果少了一些月 ！ date_histogram （和 histogram 一 ） 只会返回文 数目非零的 buckets。

意味着 的 histogram 是返回最少 果。通常， 并不想要 。 于很多 用， 可能想直接把果 入到 形 中，而不想做任何后期加工。

事 上，即使 buckets 中没有文 我 也想返回。可以通 置 个 外参数来 效果：

```
GET /cars/transactions/_search
{
   "size" : 0,
   "aggs": {
      "sales": {
         "date_histogram": {
            "field": "sold",
            "interval": "month",
            "format": "yyyy-MM-dd",
            "min_doc_count" : 0, ①
            "extended_bounds" : { ②
                "min" : "2014-01-01",
                "max" : "2014-12-31"
            }
         }
      }
   }
}
```

① 个参数 制返回空 buckets。

② 个参数 制返回整年。

个参数会 制返回一年中所有月 的 果，而不考 果中的文 数目。 min_doc_count 非常容易理解：它 制返回所有 buckets，即使 buckets 可能 空。

extended_bounds 参数需要一点解 。 min_doc_count 参数 制返回空 buckets，但是 Elasticsearch 只返回 的数据中最小 和最大 之 的 buckets。

因此如果 的数据只落在了 4 月和 7 月之 ，那 只能得到 些月 的 buckets（可能 空也可能不空）。因此 了得到全年数据，我 需要告 Elasticsearch 我 想要全部 buckets， 即便那些 buckets 可能落在最小日期 之前 或 最大日期 之后 。

extended_bounds 参数正是如此。一旦 加上了 个 置， 可以把得到的 果 易地直接 入到 的 形 中，从而得到 似 汽 的 表。

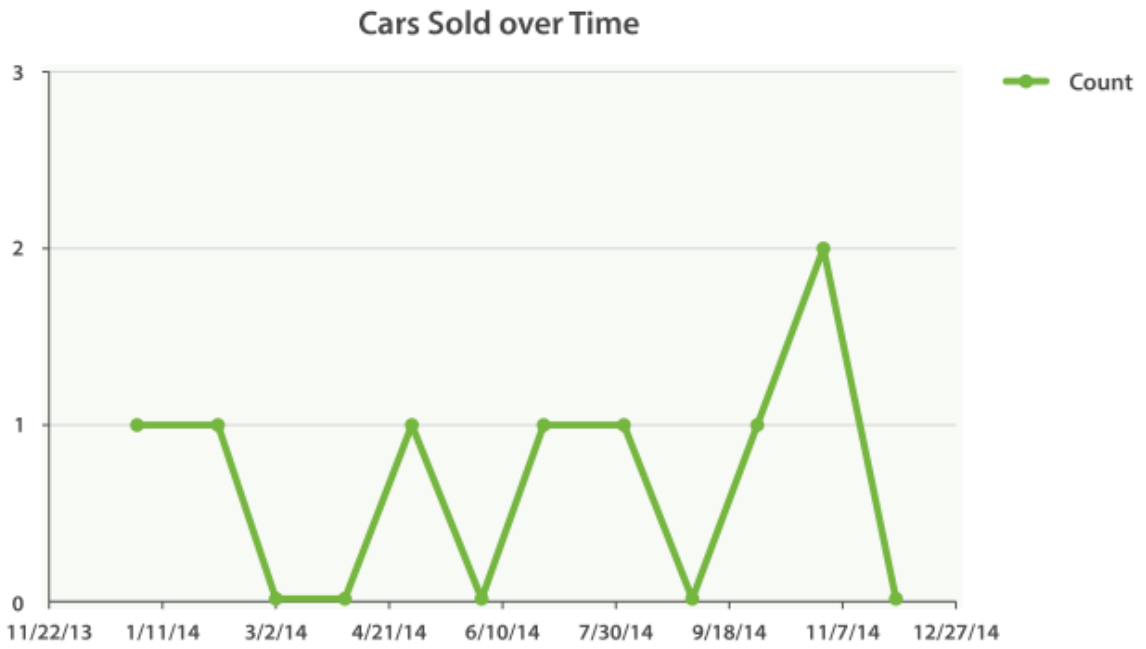

Figure 3. 汽

展例子

正如我 已 很多次，buckets 可以嵌套 buckets 中从而得到更 的分析。 作 例子，我 建聚合以便按季度展示所有汽 品牌 。同 按季度、按 个汽 品牌 算 ，以便可以出 品牌最 ：

```
GET /cars/transactions/_search
{
   "size" : 0,
   "aggs": {
      "sales": {
         "date_histogram": {
            "field": "sold",
            "interval": "quarter", ①
            "format": "yyyy-MM-dd",
            "min_doc_count" : 0,
            "extended_bounds" : {
                "min" : "2014-01-01",
                "max" : "2014-12-31"
            }
         },
         "aggs": {
            "per_make_sum": {
               "terms": {
                  "field": "make"
               },
               "aggs": {
                  "sum_price": {
                     "sum": { "field": "price" } ②
                  }
               }
            },
            "total_sum": {
               "sum": { "field": "price" } ③
            }
         }
      }
   }
}
```

① 注意我　把　　隔从 `month` 改成了 `quarter` 。

②　算　品牌的　　金　。

③ 也　算所有全部品牌的　　金　。

得到的　果（截去了一大部分）如下：

```
{
....
"aggregations": {
   "sales": {
      "buckets": [
         {
            "key_as_string": "2014-01-01",
            "key": 1388534400000,
            "doc_count": 2,
            "total_sum": {
               "value": 105000
            },
            "per_make_sum": {
               "buckets": [
                  {
                     "key": "bmw",
                     "doc_count": 1,
                     "sum_price": {
                        "value": 80000
                     }
                  },
                  {
                     "key": "ford",
                     "doc_count": 1,
                     "sum_price": {
                        "value": 25000
                     }
                  }
               ]
            }
         },
...
}
```

我 把 果 成 ，得到如 按品牌分布的 季度 所示的 的折 和 个品牌（ 季度）的柱状 。

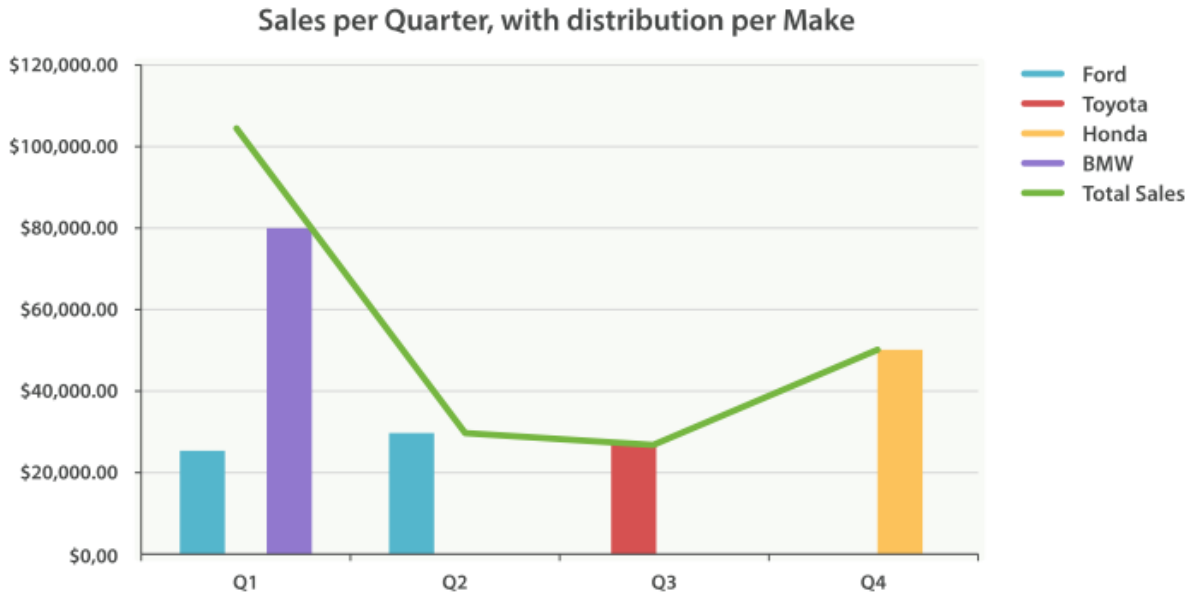

Figure 4. 按品牌分布的　季度

潜力无

　些很明　都是　　例子，但　表聚合其　是潜力无　的。 如 Kibana—用聚合　建的　　分析面板 展示了 Kibana 中用各　聚合　建的面板。

Figure 5. Kibana—用聚合　建的　　分析面板

因　聚合的　　性，　似　　的面板很容易　　、操作和交互。　使得它　成　需要分析数据又不会　建 Hadoop 作　的非技　人　的理想工具。

当然，　了　建　似　　Kibana　　的　大面板，　可能需要更深的知　，比如基于　　、

以及排序的聚合。

限定的聚合

所有聚合的例子到目前 止， 可能已 注意到，我 的搜索 求省略了一个 `query` 。 整个 求只不 是一个聚合。

聚合可以与搜索 求同 行，但是我 需要理解一个新概念： 。 情况下，聚合与 是同一 行操作的，也就是 ，聚合是基于我 匹配的文 集合 行 算的。

我 看看第一个聚合的示例：

```
GET /cars/transactions/_search
{
    "size" : 0,
    "aggs" : {
        "colors" : {
            "terms" : {
              "field" : "color"
            }
        }
    }
}
```

我 可以看到聚合是隔 的。 中，Elasticsearch "没有指定 " 和 " 所有文 " 是等 的。前面 个 内部会 化成下面的 个 求：

```
GET /cars/transactions/_search
{
    "size" : 0,
    "query" : {
        "match_all" : {}
    },
    "aggs" : {
        "colors" : {
            "terms" : {
              "field" : "color"
            }
        }
    }
}
```

因 聚合 是 内的 果 行操作的，所以一个隔 的聚合 上是在 `match_all` 的 果 操作，即所有的文 。

一旦有了 的概念，我 就能更 一 聚合 行自定 。我 前面所有的示例都是 所有 数据 算 信息的： 量最高的汽 ，所有汽 的平均 ，最佳 月 等等。

利用 ，我 可以 “福特在 有多少 色？” 如此 的 。可以 的在 求中加上一个 （本例中 `match` ）：

```
GET /cars/transactions/_search
{
    "query" : {
        "match" : {
            "make" : "ford"
        }
    },
    "aggs" : {
        "colors" : {
            "terms" : {
              "field" : "color"
            }
        }
    }
}
```

因 我 没有指定 `"size" : 0` ，所以搜索 果和聚合 果都被返回了：

```
{
...
   "hits": {
      "total": 2,
      "max_score": 1.6931472,
      "hits": [
         {
            "_source": {
               "price": 25000,
               "color": "blue",
               "make": "ford",
               "sold": "2014-02-12"
            }
         },
         {
            "_source": {
               "price": 30000,
               "color": "green",
               "make": "ford",
               "sold": "2014-05-18"
            }
         }
      ]
   },
   "aggregations": {
      "colors": {
         "buckets": [
            {
               "key": "blue",
               "doc_count": 1
            },
            {
               "key": "green",
               "doc_count": 1
            }
         ]
      }
   }
}
```

看上去 并没有什 ，但却 高大上的 表 来 至 重要。 加入一个搜索 可以将任何静 的 表板 成一个 数据搜索 。 用 可以搜索数据， 看所有 更新的形（由于聚合的支持以及 的限定）。 是 Hadoop 无法做到的！

全局桶

通常我 希望聚合是在 内的，但有 我 也想要搜索它的子集，而聚合的 象却是 所有 数据。

例如，比方 我 想知道福特汽 与 所有 汽 平均 的比 。我 可以用普通的聚合（

内的）得到第一个信息，然后用 全局 桶 得第二个信息。

全局 桶包含 所有 的文 ，它无 的 。因 它 是一个桶，我 可以像平常一 将聚合嵌套在内：

```
GET /cars/transactions/_search
{
    "size" : 0,
    "query" : {
        "match" : {
            "make" : "ford"
        }
    },
    "aggs" : {
        "single_avg_price": {
            "avg" : { "field" : "price" } ①
        },
        "all": {
            "global" : {}, ②
            "aggs" : {
                "avg_price": {
                    "avg" : { "field" : "price" } ③
                }

            }
        }
    }
}
```

① 聚合操作在 内（例如：所有文 匹配 ford ）

② global 全局桶没有参数。

③ 聚合操作 所有文 ，忽略汽 品牌。

single_avg_price 度量 算是基于 内所有文 ，即所有 福特 汽 。avg_price 度量是嵌套在 全局 桶下的， 意味着它完全忽略了 并 所有文 行 算。聚合返回的平均 是所有汽 的平均 。

如果能一直 持 到 里， 知道我 有个真言：尽可能的使用 器。它同 可以 用于聚合，在下一章中，我 会展示如何 聚合 果 行 而不是 做限定。

和聚合

聚合 限定 有一个自然的 展就是 。因 聚合是在 果 内操作的，任何可以 用于 的 器也可以 用在聚合上。

如果我 想 到 在 $10,000 美元之上的所有汽 同 也 些 算平均 ， 可以 地使用一个 constant_score 和 filter 束：

```
GET /cars/transactions/_search
{
    "size" : 0,
    "query" : {
        "constant_score": {
            "filter": {
                "range": {
                    "price": {
                        "gte": 10000
                    }
                }
            }
        }
    },
    "aggs" : {
        "single_avg_price": {
            "avg" : { "field" : "price" }
        }
    }
}
```

正如我 在前面章 中 那 ，从根本上 ，使用 non-scoring 和使用 match 没有任何区 。 （包括了一个 器）返回一 文 的子集，聚合正是操作 些文 。使用 filtering query 会忽略 分，并有可能会 存 果数据等等。

桶

但是如果我 只想 聚合 果 ？ 假 我 正在 汽 商 建一个搜索 面， 我 希望 示用 搜索的 果，但是我 同 也想在 面上提供更 富的信息，包括（与搜索匹配的）上个月度汽 的平均 。

里我 无法 的做 限定，因 有 个不同的条件。搜索 果必 是 ford ，但是聚合 果必 足 ford AND sold > now - 1M 。

了解决 个 ，我 可以用一 特殊的桶，叫做 filter （注： 桶） 。 我 可以指定一个 桶，当文 足 桶的条件 ，我 将其加入到桶内。

果如下：

```
GET /cars/transactions/_search
{
   "size" : 0,
   "query":{
      "match": {
         "make": "ford"
      }
   },
   "aggs":{
      "recent_sales": {
         "filter": { ①
            "range": {
               "sold": {
                  "from": "now-1M"
               }
            }
         },
         "aggs": {
            "average_price":{
               "avg": {
                  "field": "price" ②
               }
            }
         }
      }
   }
}
```

① 使用　　桶在　　基　上　用　　器。

② avg 度量只会　ford 和上个月　出的文　　算平均　　。

因　　filter　桶和其他桶的操作方式一　，所以可以随意将其他桶和度量嵌入其中。所有嵌套的　件都会"　承"　个　　，　使我　可以按需　　聚合　　出　　部分。

后　　器

目前　止，我　可以同　　搜索　果和聚合　果　行　　（不　算得分的　filter　　），以及　　聚合　果的一部分　行　　（ filter 桶）。

我　可能会想，"只　　搜索　果，不　　聚合　果　？" 答案是使用 post_filter 。

它是接收一个　　器的　　搜索　求元素。　个　　器在　　之后　　行（　正是　　器的名字的由来：它在　　之后　　post　　行）。正因　它在　　之后　行，它没有任何影　，所以　聚合也不会有任何影　。

我　可以利用　个行　　条件　用更多的　　器，而不会影　其他的操作，就如　UI　上的各个分面。　我　　汽　　商　　外一个搜索　面，　个　面允　用　搜索汽　同　可以根据　色来　　。色的　　是通　聚合　得的：

```
GET /cars/transactions/_search
{
    "size" : 0,
    "query": {
        "match": {
            "make": "ford"
        }
    },
    "post_filter": {    ①
        "term" : {
            "color" : "green"
        }
    },
    "aggs" : {
        "all_colors": {
            "terms" : { "field" : "color" }
        }
    }
}
```

① post_filter 元素是 top-level 而且　　命中　果　行　　。

　　部分　到所有的 ford 汽　，然后用 terms 聚合　建一个　色列表。因　聚合　　　　行操作，色列表与福特汽　有的　色相　　。

最后，post_filter 会　　搜索　果，只展示　色 ford 汽　。　在　　行　后　生，所以聚合不受影　。

　通常　　UI　　的　一致性很重要，可以想象用　在界面商　　了一　　色（比如：色），期望的是搜索　果已　被　　了，而　不是　　界面上的　　。如果我　　用 filter　　，界面会　上　成只　示　色作　　，　不是用　想要的！

WARNING

性能考　（*Performance consideration*）

当　需要　搜索　果和聚合　果做不同的　　，　才　　使用 post_filter ，　有　用　会在普通搜索使用 post_filter 。

不要　　做！ post_filter 的特性是在　　之后　　行，任何　　性能　来的好　（比如　存）都会完全失去。

在我　需要不同　　，post_filter 只与聚合一起使用。

小

　合　　型的　　（如：搜索命中、聚合或　者兼有）通常和我　期望如何表　用　交互有　。　　合的　　器（或　合）取决于我　期望如何将　果呈　　用　。

- 在 filter　　中的 non-scoring　　，同　影　搜索　果和聚合　果。
- filter 桶影　聚合。

- post_filter 只影 搜索 果。

多桶排序

多 桶（ terms 、 histogram 和 date_histogram ） 生成很多桶。 Elasticsearch 是如何决定 些桶展示 用 的 序 ？

的，桶会根据 doc_count 降序排列。 是一个好的 行 ，因 通常我 想要 到文 中与 条件相 的最大 ： 、人口数量、 率。但有些 候我 希望能修改 个 序，不同的桶有着不同的 理方式。

内置排序

些排序模式是桶 固有的 能力：它 操作桶生成的数据 ，比如 doc_count 。 它 共享相同的 法，但是根据使用桶的不同会有些 微差 。

我 做一个 terms 聚合但是按 doc_count 的升序排序：

```
GET /cars/transactions/_search
{
    "size" : 0,
    "aggs" : {
        "colors" : {
            "terms" : {
              "field" : "color",
              "order": {
                "_count" : "asc" ①
              }
            }
        }
    }
}
```

① 用 字 _count ，我 可以按 doc_count 的升序排序。

我 聚合引入了一个 order 象， 它允 我 可以根据以下几个 中的一个 行排序：

_count
按文 数排序。 terms 、 histogram 、 date_histogram 有效。

_term
按 的字符串 的字母 序排序。只在 terms 内使用。

_key
按 个桶的 数 排序（理 上与 _term 似）。 只在 histogram 和 date_histogram 内使用。

按度量排序

有 ，我 会想基于度量 算的 果 行排序。 在我 的汽 分析 表 中，我 可能想按照汽 色 建一个 条状 表，但按照汽 平均 的升序 行排序。

我　可以　加一个度量，再指定 order 参数引用　个度量即可：

```
GET /cars/transactions/_search
{
    "size" : 0,
    "aggs" : {
        "colors" : {
            "terms" : {
              "field" : "color",
              "order": {
                "avg_price" : "asc" ②
              }
            },
            "aggs": {
                "avg_price": {
                    "avg": {"field": "price"} ①
                }
            }
        }
    }
}
```

①　算　个桶的平均　　。

② 桶按照　算平均　的升序排序。

我　可以采用　　方式用任何度量排序，只需　　的引用度量的名字。不　有些度量会　出多个　。`extended_stats` 度量是一个很好的例子：它　出好几个度量　。

如果我　想使用多　度量　行排序，　我　只需以　心的度量　　使用点式路径：

```
GET /cars/transactions/_search
{
    "size" : 0,
    "aggs" : {
        "colors" : {
            "terms" : {
              "field" : "color",
              "order": {
                "stats.variance" : "asc" ①
              }
            },
            "aggs": {
                "stats": {
                    "extended_stats": {"field": "price"}
                }
            }
        }
    }
}
```

① 使用 . 符号，根据感 趣的度量 行排序。

在上面 个例子中，我 按 个桶的方差来排序，所以 色 方差最小的会排在 果集最前面。

基于“深度”度量排序

在前面的示例中，度量是桶的直接子 点。平均 是根据 个 term 来 算的。 在一定条件下，我 也有可能 更深 的度量 行排序，比如 子桶或从 桶。

我 可以定 更深的路径，将度量用尖括号（ > ）嵌套起来，像 ： my_bucket>another_bucket>metric 。

需要提醒的是嵌套路径上的 个桶都必 是 的。 filter 桶生成 一个 桶：所有与条件匹配的文 都在桶中。 多 桶（如：terms ） 生成 多桶，无法通 指定一个 定路径来 。

目前，只有三个 桶： filter 、 global 和 reverse_nested 。 我 快速用示例 明，建一个汽 的直方 ，但是按照 色和 色（不包括 色） 各自的方差来排序：

```
GET /cars/transactions/_search
{
    "size" : 0,
    "aggs" : {
        "colors" : {
            "histogram" : {
              "field" : "price",
              "interval": 20000,
              "order": {
                "red_green_cars>stats.variance" : "asc" ①
              }
            },
            "aggs": {
                "red_green_cars": {
                    "filter": { "terms": {"color": ["red", "green"]}}, ②
                    "aggs": {
                        "stats": {"extended_stats": {"field" : "price"}} ③
                    }
                }
            }
        }
    }
}
```

① 按照嵌套度量的方差 桶的直方 行排序。

② 因 我 使用 器 `filter`，我 可以使用嵌套排序。

③ 按照生成的度量 果 行排序。

本例中，可以看到我 如何 一个嵌套的度量。 `stats` 度量是 `red_green_cars` 聚合的子 点，而 `red_green_cars` 又是 `colors` 聚合的子 点。 了根据 个度量排序，我 定 了路径 `red_green_cars>stats.variance`。我 可以 做，因 `filter` 桶是个 桶。