GITHUB 仓库流行度

分析报告

2016-6-10

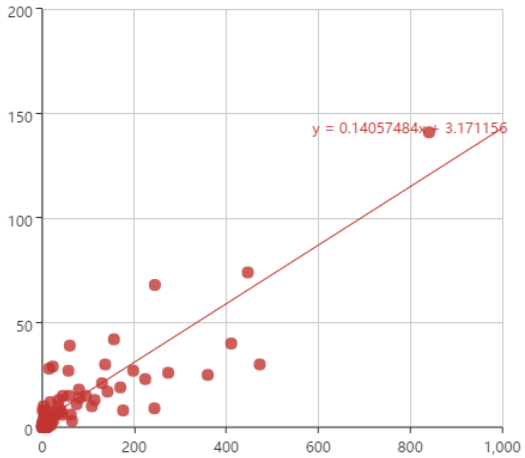
南京大学软件学院 崔浩

引言

通过 Github 提供的 API，我们通过随机抽样获取了 Github 的 98341 个公共仓库，以及于此相关的 330226 位用户，1522728 条贡献记录，将他们保存在云端服务器数据库中。我们依次为基础分析项目语言、项目领域和项目贡献者对项目 star 数量的影响，以及影响的显著性差异，并用逻辑分类算法给出了该项目流行概率的估计。

一、流行度

在仓库可获取的数据中，我们认为仓库的 star 数量和 fork 数量最能代表仓库受欢迎的程度，我们通过统计发现，star 和 fork 之间存在一定的关联，为了统计方便，我们便直接选取 star 的数量作为仓库流行度的衡量标准。



除此之外，我们定义 star 数量位于前 5%的为流行仓库，这样的比例对应到 star 数量上约为大于等于 50（star 数量大于等于 50 的仓库共有 5115 个，占总数的 5.20%）。

根据我们所能获得的数据，我们在这次实验中考虑以下因素对项目的影响：

1、 项目语言：项目语言是指 Github 接口提供的项目的主要语言，在本次实验中没有考虑项目的其他语言分布。

2、 项目领域：根据我们能获取到的数据，我们对项目描述做了词频分析，并选取了约 12 个出现频率最多的领域，以此考察不同的领域对项目流行度的影响。

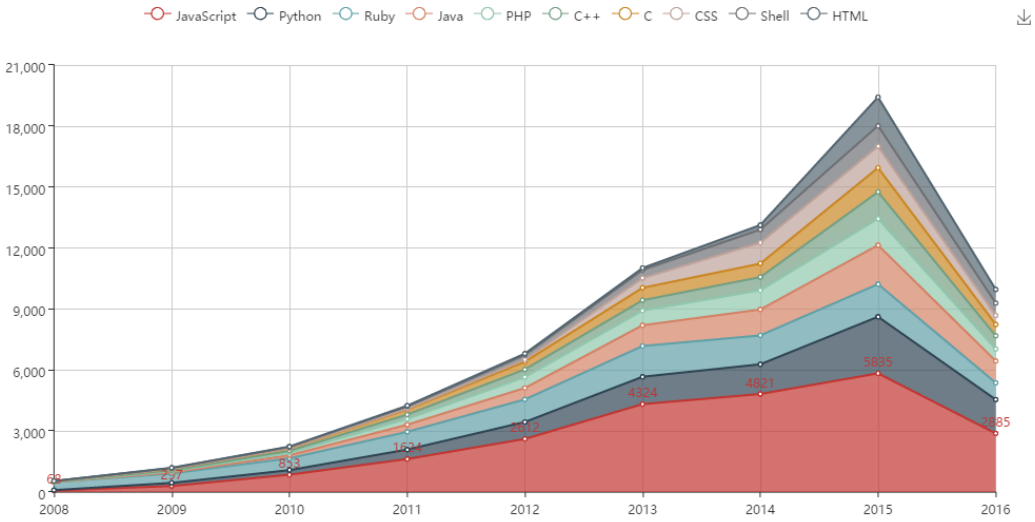
3、 项目贡献者：指的是项目的 contributor，对项目贡献者的考察，是统计贡献者的 follower 数量，我们不妨推测用户的 follower 数量越多，这名用户则越优秀。

下面我们对这三个因素进行一些简单、直观的统计。

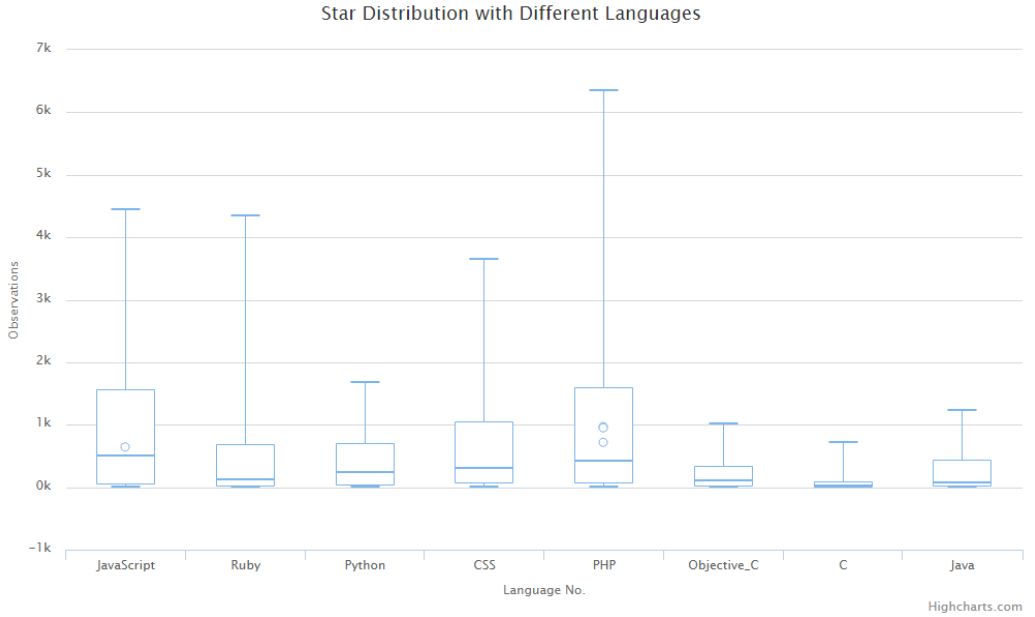
二、项目语言

在考察流行仓库的语言之前，我们先统计了 Github 仓库语言的发展情况：我们将仓库按创建时间分类，并统计了每一年不同语言仓库的创建数量。

从下图中可以看到，Github 仓库创建数量逐渐增多（2016 年截止至 5 月），JavaScript 和 HTML 随时间所占比例越来越大，而 Github 传统的 Ruby 语言所占比例却在不断下滑，这说明了 Github 开始告别传统的 Ruby 社区走向多元化，而 WEB 技术的流行使得 Github 上的 WEB 项目逐渐增多。

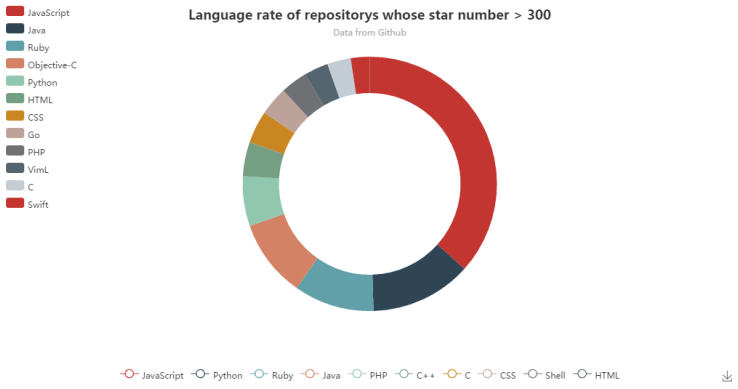


语言对流行度的预测究竟有什么样的影响呢？我们仅仅改变项目语言这一变量，进行了抽样，统计了不同语言的项目的 star 数情况。

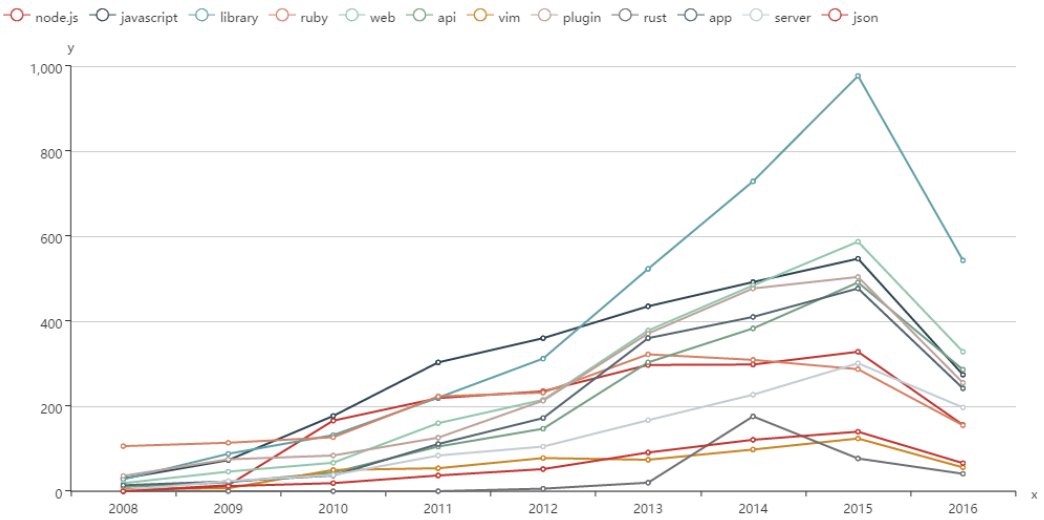


从结果来看，如果一个项目的语言是 JavaScript 或 PHP 等语言，我们更愿意相信他们比一个 C 语言的项目更容易受到关注（尽管每种语言都有不受关注的项目）。

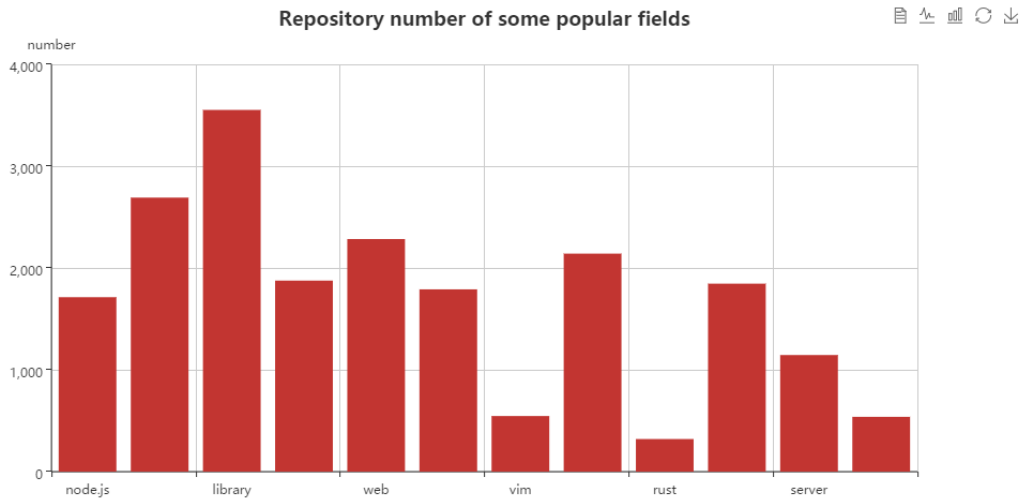
此外，我们统计了流行仓库中各个语言所占的比例：



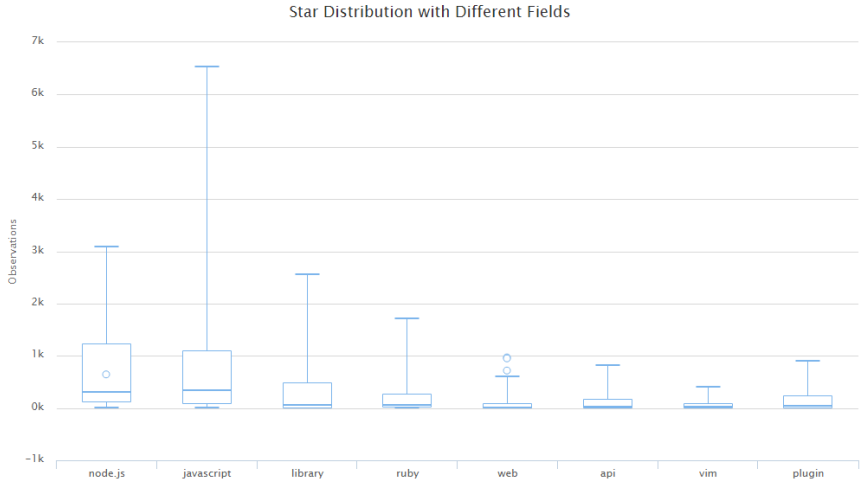
三、项目领域



可以看到，标有 web 和 library 关键词的项目在 2012 年以后增长迅猛，标注 node.js 或 json 关键词的项目增长平稳，vim 和 json 的增长趋势几乎完全一致，我们没有具体探究这样的原因，仅仅描述这样的现象。



在较流行的项目中，library，web，app，plugin 等关键词频频出现。

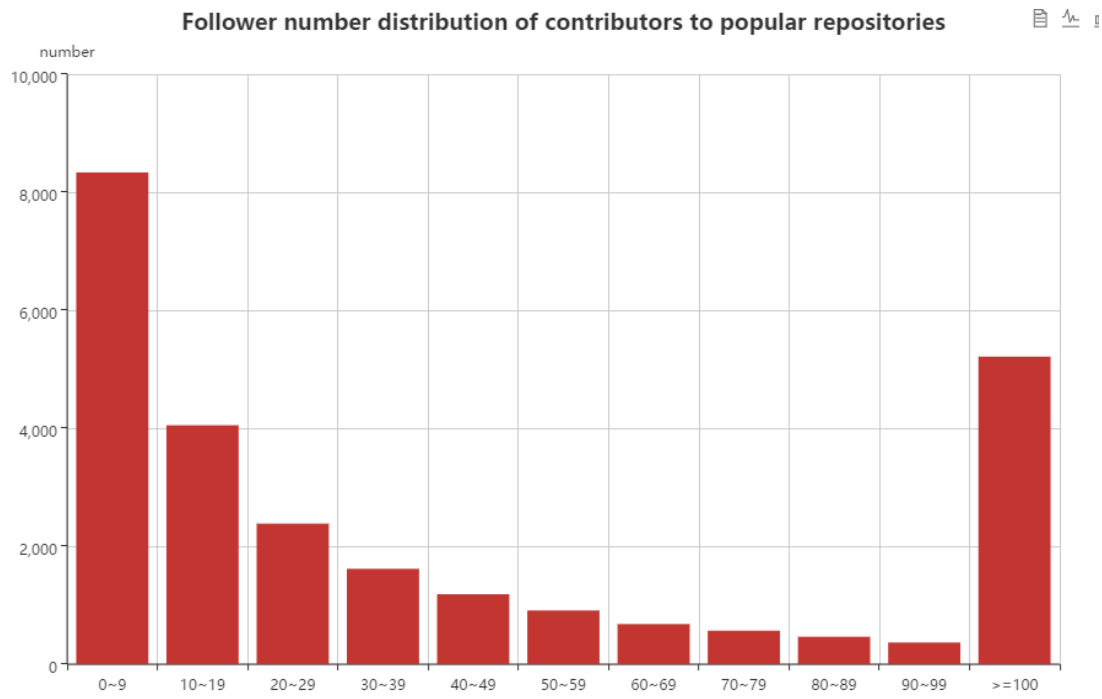


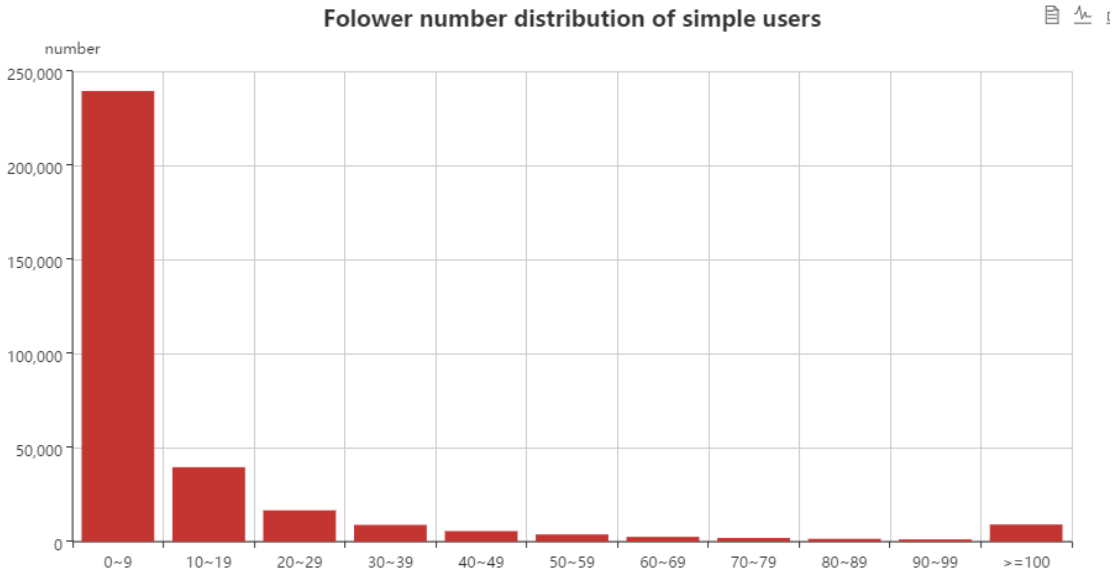
然而，在针对不同领域的随机抽样中，node.js， JavaScript， library， plugin 却表现地更加出色。

四、项目贡献者

敏捷软件开发一直把优秀的团队成员放在更重要的位置，那么，项目贡献者的优秀程度对项目流行的影响究竟有多大呢？

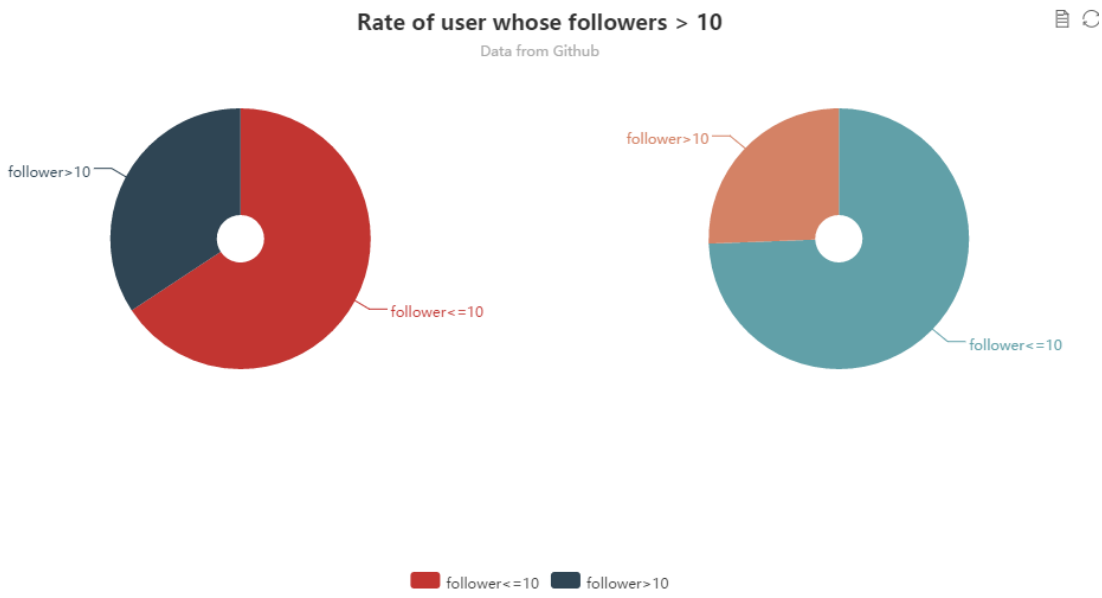
我们将流行项目的贡献者关注人数的分布与普通 Github 项目贡献者关注人数分布做了对比。





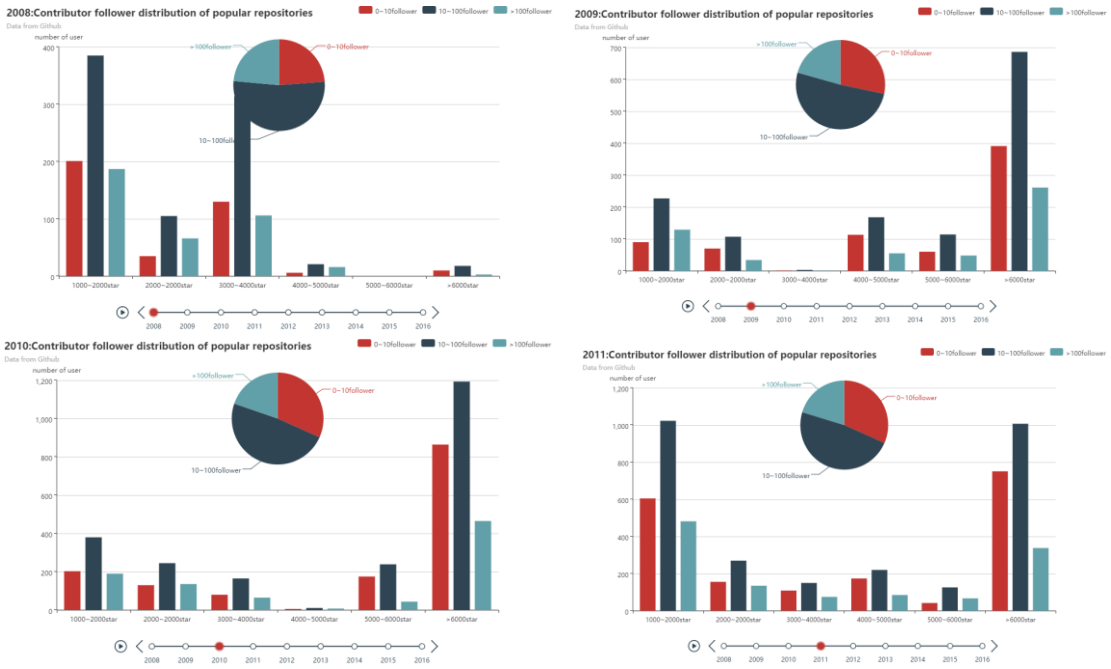
可以看到，尽管他们都有大量的普通贡献者（即关注者人数小于 10 的贡献者），但在中等和优秀的贡献者区间里，流行项目的优秀贡献者占有更高的比例。尤其是关注人数超过

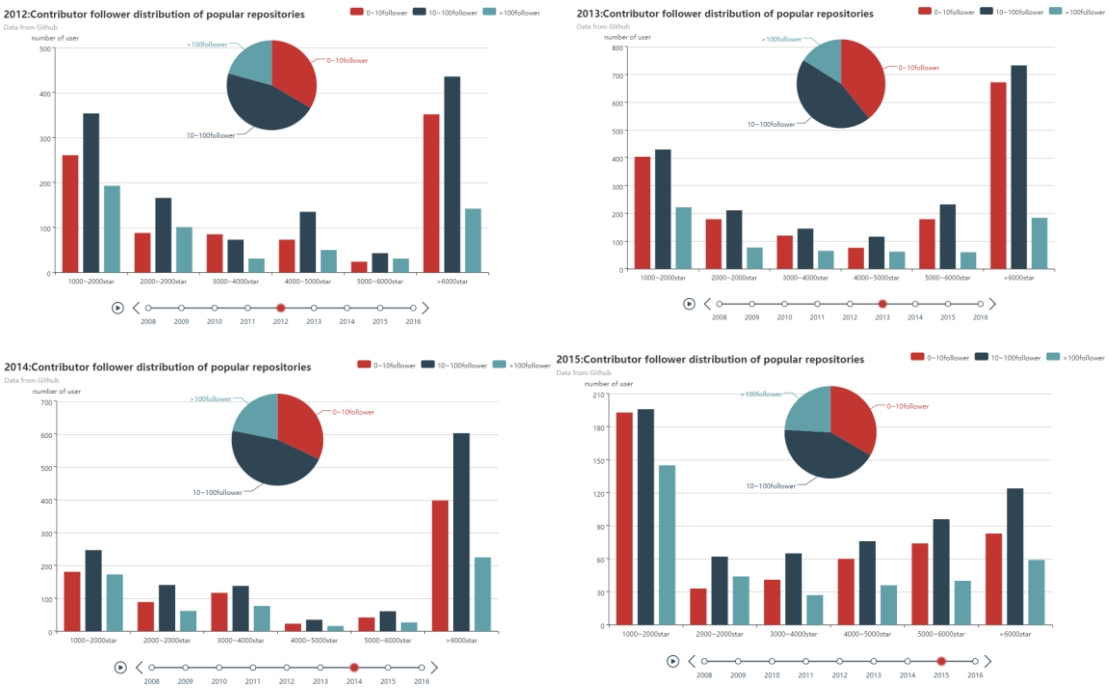
100 人这样的区间里，流行仓库有更多这样优秀的贡献者。这样的差距究竟有多大呢？我们用饼图说明两者的差异：



左边是流行项目贡献者的统计。拥有多于 10 个关注的贡献者占据了 34.26%，而右边的是 Github 普通项目的水平，拥有多于 10 个关注的贡献者占据了 25.59%。

我们还想知道，对于极其优秀的项目（star 数量超过 1000 的项目），这样的分布会有什么变化呢？我们根据年份的变化统计 0-10，10-100，>100 关注的贡献者人数总和，并按照仓库的 star 数分类。





可以看到，尽管随着年份增长，普通贡献者群体的比例略有上升，但非常流行的仓库的贡献者主力军永远都是中等的用户（关注人数在 10 到 100 之间的），而优秀的用户（关注人数大于 100）也一直占据着很大的比重。

五、显著性分析

上面的三个因素，都对项目的流行造成了显著影响吗？如果是，造成的影响究竟有多大呢？我们对此进行了单因素方差分析。

1、 实验数据

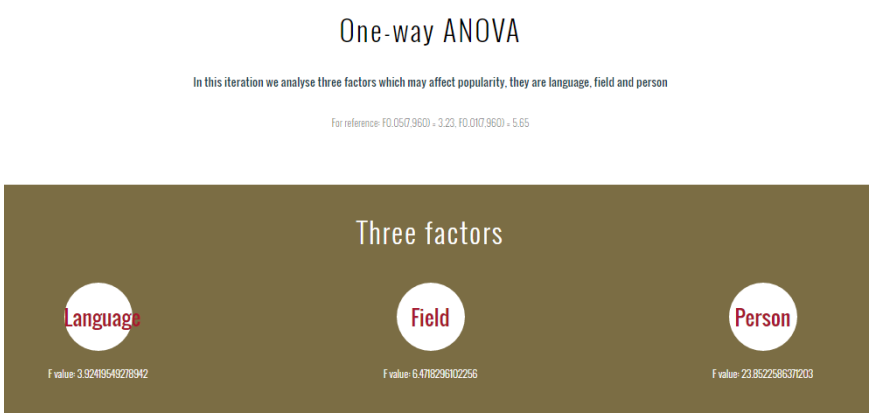
项目语言：选出 Github 最常用的 8 种语言，对于这 8 种语言进行 8 次随机抽样，每次抽取 121 个数据（star 数量），即 m = 8， r = 121；

项目领域：选出出现频率最高的 8 个领域，对于这 8 个领域进行 8 次随机抽样，每次抽取 121 个数据（star 数量），即 m = 8，r = 121；

项目贡献者：因素水平：选取优秀贡献者（follower>100）人数分别为 0，1，…，7 的仓库，每次抽取 121 个数据（star 数量），即 m=8，r=121；

说明：因样本总量足够大，每次抽取 121 个可以看作是进行 121 次伯努利试验，即看成放回抽样。

2、 实验结果



图中可以看到，语言 F = 3.92， 领域 F = 6.47， 贡献者 F = 23.85
查 F 分布表可知，$F_{0.05}$（7，960）= 3.23， $F_{0.01}$（7，960）= 5.65，即语言因素对流行度影响为显著差异（*），领域和贡献者对流行度影响为高度显著差异（**）。

六、逻辑分类算法

限于水平和已有的数据条件，我们选择贡献者的关注者人数作为参考因素，使用贡献者关注人数的总和和平均值作为回归项， 即 X_i = [1, sum(followers), avg(followers)]

我们的思路是使用 2013 年之前的仓库数据作为回归项（是否流行已经基本稳定），流行的标准为是否能获得大于等于 50 个 star，即 y = 1 (if star >= 50)， y = 0 (if star < 50)

我们希望通过逻辑函数计算出 P（y=1 | X，theta），即在 X 和参数 theta 已知的情况下计算出 star>50 的条件概率。

逻辑函数为：

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}},$$

（以下参考机器学习理论）

接下来要做的就是在已知分布的情况下通过样本数据求出 theta 的具体值。使用极大似然估计法可得：

$$\begin{aligned} L(\theta) &= p(\vec{y} \mid X; \theta) \\ &= \prod_{i=1}^{m} p(y^{(i)} \mid x^{(i)}; \theta) \\ &= \prod_{i=1}^{m} \left(h_\theta(x^{(i)})\right)^{y^{(i)}} \left(1 - h_\theta(x^{(i)})\right)^{1-y^{(i)}} \end{aligned}$$

$$\begin{aligned} \ell(\theta) &= \log L(\theta) \\ &= \sum_{i=1}^{m} y^{(i)} \log h(x^{(i)}) + (1 - y^{(i)}) \log(1 - h(x^{(i)})) \end{aligned}$$

代价函数为：J（theta）= -1/m * l（theta）
在最终的梯度下降算法中，每个周期的变化公式和线性回归一样：

$$\theta_j := \theta_j + \alpha \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)}$$

但这里的 h（x）指的是 sigmoid（X*theta），这是和线性回归不同的地方。为了省去调学习率的工作，我们直接使用了 matlab 提供的函数：

```
[theta, cost, existFlag] = ...
    fminunc(@(t)(costFunction(t, X, y)), initial_theta, options);
```

计算得到 theta0 = -2.502 ， theta1=0.0016284， theta2=0.0088649

Theta1 较小的原因是这里使用了 follower 的和，也就是说我们判断仓库的流行度可能的时候对仓库贡献者 followers 的总和和平均值都有很高的要求。

我们使用这样的数据对仓库的流行度进行预测，完成我们最初的目的。