## **新冠疫情的影响分析**

## 摘要

本文研究的是新冠肺炎疫情带来的影响问题, 根据概率论与数理统计相关原理, 建立了假设检验模型, 以原假设为目标, 得出新型冠状病毒确实让程序员的工作增加了。

自计算机于 1946 年问世以来, 它极大地改变了人们的生活, 为我们带来了各种各样的便利。bilibili 弹幕视频网 (https://www.bilibili.com/) 知名 Python 教师偶尔有点小迷糊曾经说过: “毕竟我们 (人类) 是碳基生物, 而计算机是硅基生物, 脑回路不太一样。程序员写软件的过程, 就是用硅基生物的脑回路来解决碳基生物遇到的问题。” 所以, 无妨说, 是程序员极大地改变了我们的日常生活。感谢程序员!

2019 年年底, 新型冠状病毒 (novel coronavirus) 爆发。英语新闻用 pandemic 一词形容新型冠状病毒, 这反映出这是一个非常恐怖的、流行于全球的传染病。疫情首次在中华人民共和国湖北省武汉市大爆发, 随后全湖北省沦陷, 随后 (几乎) 全中华人民共和国沦陷, 随后......事态已经发展到 “处处皆新冠”。我们中华人民共和国在中国共产党的伟大领导下, 交出了一份近乎完美的答卷。起初, 中华人民共和国的互联网公司是在互联网上工作的。不过, 随着疫情逐渐得到控制, 我们开始复工、复学。但是, 在自由美利坚, 疫情越来越严重, 这都 0202 年 5 月 61 日了, 还没到拐点, 川建国同志还闹着要复工复产呢。这就是自由国度吗, 我们小组 eye 了 eye 了。所以, 像 Facebook, Google, Microsoft, Twitter 等公司都已经长期线上办公了。我们小组联想到最近在蓝点网 (https://www.landiannews.com/) 看到的几则新闻, 决定用数量关系考察新型冠状病毒对程序员带来的影响。

我们也不吝啬地, 表达我们对数学建模天团的赞美。有如此开放的题, 真是照顾我们三个菜鸟了。

最后, 我们对模型的优缺点进行评价, 并提出改进的方向。

关键词：概率论 数理统计 正态分布 假设检验

## 一、问题的背景与重述

自 2019 年底, 新冠肺炎席卷全球, 截至 2020 年 5 月 16 日 16 时, 全球共有 215 个国家和地区发现有确诊病例, 累计确诊病例 4 515 561 例, 累计死亡 307 543 例, 新冠肺炎的传播和流行给全世界人民生命财产带来巨大危害。请你们的小组选择一个或相关联的多个侧面, 定量研究新冠肺炎疫情带来的影响, 所需数据自行查找。

## 二、问题的分析

考虑到 GitHub (https://github.com/) 是全球最大同性交友网站 (没有之一), 我们想到, 可以用 GitHub 的代码提交量 (the number of commits) 反映出程序员的工作是否受到影响。我们三位组员都在使用 Microsoft 的免费、小巧、构建于开源代码的文本编辑器 Visual Studio Code。这是一个富有生命力的软件。为了完备性, 我们也考察 GitHub 自己的 Atom、Adobe 的 Brackets。这三款文本编辑器都与 JavaScript 密切相关。

## 三、模型的假设

1. 假设代码提交量服从正态分布 *N*(*μ*, *σ*2)。

## 四、符号说明

|  |  |  |
| --- | --- | --- |
| 符号 | 表示含义 | 单位 |
| *μ* | 周代码提交量的均值 | 次 |
| *σ* | 周代码提交量的标准差 | 次 |
| *X* | 疫情爆发前周代码提交量 | 次 |
| *Y* | 疫情爆发后周代码提交量 | 次 |

## 五、模型的建立与求解

5.1 模型的建立

5.1.1 JSON

JSON (JavaScript Object Notation, JavaScript 对象表示法) 是一种由 Douglas Crockford 构想和设计、轻量级的数据交换语言, 该语言以易于让人阅读的文字为基础, 用来传输由属性值或者序列性的值组成的数据对象。尽管 JSON 是 JavaScript 的一个子集, 但 JSON 是独立于语言的文本格式, 并且采用了类似于C语言家族的一些习惯。

JSON 数据格式与语言无关。即便它源自 JavaScript, 但当前很多编程语言都支持 JSON 格式数据的生成和解析。JSON 的官方 MIME 类型是 application/json, 文件扩展名是 .json。

我们将使用 JSONArray 解析 API 返回的 JSON 数据。

5.1.2 API

API (Application Programming Interface, 应用程序接口) 是一些预先定义的函数, 或指软件系统不同组成部分衔接的约定。用来提供应用程序与开发人员基于某软件或硬件得以访问的一组例程, 而又无需访问原码, 或理解内部工作机制的细节。

我们将使用 Java 编写模拟 HTTPS 请求的爬虫, 对 API 返回的 JSON 文本解析。选用 Java 是因为 Java 有丰富的文档与社区支持, 且过程可高度自定义。为方便使用, 我们将一些操作打包为 public static 方法。这从某种角度上说, 也可以是 API。

5.1.3 正态分布

若随机变量 *X* 满足  
我们称 *X* 服从参数为 *μ*, *σ*2 的正态分布, 记作 *X* ~ *N*(*μ*, *σ*2)。正态分布是一个非常常见的连续概率分布。正态分布在统计学上十分重要, 经常用在自然和社会科学来代表一个不明的随机变量。正态分布有一个非常重要的性质: 在特定条件下, 大量统计独立的随机变量的平均值的分布趋于正态分布, 这就是中心极限定理。中心极限定理的重要意义在于, 根据这一定理的结论, 其他概率分布可以用正态分布作为近似。

5.1.4 *t* 分布

若随机变量 *X* 满足  
我们称 *X* 服从自由度为 *n* 的 *t* 分布, 记作 *X* ~ *t*(*n*)。*t* 分布用于根据小样本来估计呈正态分布且变方差未知的总体的平均值。如果总体方差已知 (例如在样本数量足够多时), 则应该用正态分布来估计总体均值。

5.1.5 假设检验

假设检验是推论统计中用于检验统计假设的一种方法。而 “统计假设” 是可通过观察一组随机变量的模型进行检验的科学假说。一旦能估计未知参数, 就会希望根据结果对未知的真正参数值做出适当的推论。统计上对参数的假设, 就是对一个或多个参数的论述。而其中欲检验其正确性的为零假设, 零假设通常由研究者决定, 反映研究者对未知参数的看法。相对于零假设的其他有关参数之论述是备择假设, 它通常反应了执行检定的研究者对参数可能数值的另一种 (对立的) 看法 (换句话说, 备择假设通常才是研究者最想知道的)。假设检验的种类包括: *t* 检验, *Z* 检验, 卡方检验, *F* 检验等等。

5.1.6 *t* 检验

设 , , 其中总体均值与方差均未知。设 有 个样本, 有 个样本。我们选取原假设 , 备选假设 为 的对立。由于了解的信息过少, 且样本量也不是很大, 这使问题变得复杂, 而且在实际应用里 (就比如现在这情况) 也不少。这是历史上著名的 Behrens-Fisher 问题, 至今还有学者研究。下面介绍一个近似方法。令  
其中 表示离 最近的整数 (若小数部分为 0.5, 取偶数)。从而  
近似服从自由度为 的 分布。取显著性水平 , 则拒绝域 , 其中 是下述方程的唯一实数解:  
若 , 则拒绝原假设, 反之接受原假设。

5.2 模型的求解

首先, 我们打算直接使用 GitHub API 进行数据的获取。但是, 由于众所周知的原因, GitHub 的某些页面被拦截在万里长城之外。所以, 我们退而求其次, 使用中华人民共和国广东省深圳市奥思网络科技有限公司的码云 (gitee.com)。我们使用码云的 import repository 功能, 成功地导入了 Atom, Brackets, Visual Studio Code 的源代码 repositories。码云也提供了与 GitHub API 相似的 API, 从而我们的 JSON 模型仍可用, 并且, 获取数据的时间也缩短了不少。

我们首先实现 X509TrustManager 接口启用 HTTPS 请求, 然后伪造 User Agent, 欺骗那些反爬虫的 API。得到 API 返回的文本后, 我们使用 JSONArray 解析, 得到 JSON 数据, 从而得出 JSON 数据的长度, 也就是指定范围的 commits 数。

为了方便后期处理, 我们还调用 BufferedWriter 与 FileWriter 进行文件写入。我们把每日的 commits 数保存为 txt 文件, 这样, 每次使用数据时, 就无需等待网络连接。此外, 由于码云 API 限制, 每次最多返回 100 条 commits 记录, 所以, 我们还对commits 数不低于 100 的日期进行复查, 得到真实数据文件。

随后, 我们调用 BufferedReader 与 FileReader 进行文件读取。因为数据的保存格式为 yyyyMMdd,N, 其中 yyyy, MM, dd 分别表示年、月、日, N 表示日 commits 数, 所以我们还使用了 substring 方法, 截取出我们需要的 commits 数。考虑到这些编辑器的公司都在自由美利坚, 而自由美利坚的工作日是周一至周五, 所以, 我们以一周为基本周期, 考虑周 commits 数。我们使用 DescriptiveStatistics 进行数理统计, 节约我们写计算均值、标准差、分位数等方法的时间。通过 TDistribution 与假设检验的方法, 我们得到了如下结果。

atom:

Before COVID-19:

mean: 7.2

standard deviation: 6.67753500059794

After COVID-19:

mean: 6.15

standard deviation: 6.761150633933313

(y^bar - x^bar - mu\_2 + mu\_1)/s0 is approximately t-distributed.

s0 = 2.1248838977570914

degree of freedom = 38

alpha = 0.05

t\_alpha = -1.685954460167519

t = -0.49414464531842

It can be drawn that after\_mean >= before\_mean.

================

brackets:

Before COVID-19:

mean: 0.9500000000000002

standard deviation: 2.0894471693929506

After COVID-19:

mean: 2.85

standard deviation: 5.274317117185084

(y^bar - x^bar - mu\_2 + mu\_1)/s0 is approximately t-distributed.

s0 = 1.2685466196856108

degree of freedom = 25

alpha = 0.05

t\_alpha = -1.7081407612528856

t = 1.497777039105496

It can be drawn that after\_mean >= before\_mean.

================

vscode:

Before COVID-19:

mean: 301.55

standard deviation: 89.09632575565557

After COVID-19:

mean: 346.2

standard deviation: 80.76580829456041

(y^bar - x^bar - mu\_2 + mu\_1)/s0 is approximately t-distributed.

s0 = 26.889841067428772

degree of freedom = 38

alpha = 0.05

t\_alpha = -1.685954460167519

t = 1.6604783898884325

It can be drawn that after\_mean >= before\_mean.

这说明, 疫情使程序员工作变多了。虽然这也不难从定性的角度理解, 不过, 定量分析给我们的论据提供了强有力的支撑。值得注意的是, Visual Studio Code 比 Atom, Brackets 都更有活力。这得益于 Visual Studio Code 有着较 Atom 好的性能, 且插件与 Google Chrome 浏览器的插件类似, 都使用 JavaScript 等前端技术, 所以越来越多的开发者使用 Visual Studio Code。由于 Brackets 功能没有 Atom 或 Visual Studio Code 丰富, 且有更轻量的对手, Brackets 并没有那么活跃。由于 Microsoft 收购了 GitHub, 且 Atom 与 Visual Studio Code 的功能高度重合, Atom 也在走下坡路。

## 六、模型的评价与改进

6.1 模型的评价

6.1.1 模型的优点

1. 本文采用概率论与数理统计模型对 GitHub 的代码提交量进行研究, 具有较为严谨的推导过程, 且模型具有普适性。

2. 在研究影响时, 模型参数能够根据要求通过使用计算机编程语言写出的程序随时进行调增加、削减或改动, 使得到结果的过程加快, 提供了极大的方便。

6.1.2 模型的缺点

1. 考虑的影响因素较少, 在处理问题时可能存在一些误差, 结果不够精确。

6.2 模型的改进

由于本文假设代码提交量服从正态分布 *N*(*μ*, *σ*2), 近似得出比较结果, 但也有小概率会犯错。因此, 可以增加样本, 探求代码提交量服从于何种分布, 构造合适的统计量进行求解。

## 七、模型的应用与推广

本文主要运用到概率论与数理统计相关知识。概率论与数理统计在各种各样想到的与没想到的方面都有应用。我们可以将此方法推广到各种不同情况下的概率与统计问题中。这个模型具有很强的适用性, 基本上稍作改动即可研究其他事件对提交量的影响, 或者研究对其他因素的影响。这个模型也具有很强的推广意义, 例如: 研究新型冠状病毒对全国 (或全球) 经济的影响。

1. 参考文献

[1] 偶尔有点小迷糊. 小迷糊的 Python 入门教程 (9) 循环: 简单粗暴但有效 [EB/OL]. https://www.bilibili.com/video/BV1B5411x7Pj, 2020-05-03.

[2] 山外的鸭子哥. 疫情让远程办公真实效果得到印证 推特正在考虑无限期的远程办公方案 [EB/OL]. https://www.landiannews.com/archives/73897.html, 2020-05-14.

[3] 山外的鸭子哥. 谷歌和脸书同时宣布其员工除非必须到岗否则可选在家办公至 2021 年 [EB/OL]. https://www.landiannews.com/archives/73897.html, 2020-05-09.

[4] Wikipedia. JSON [EB/OL]. https://zh.jinzhao.wiki/zh-cn/JSON, 2020-03-23.

[5] 百度百科. API [EB/OL]. https://baike.baidu.com/item/api/10154, 2020-05-16.

[6] Wikipedia. 正态分布 [EB/OL]. https://zh.jinzhao.wiki/zh-cn/%E6%AD%A3%E6%80%81%E5%88%86%E5%B8%83, 2020-05-12.

[7] Wikipedia. 学生 t - 分布 [EB/OL]. https://zh.jinzhao.wiki/zh-cn/%E5%AD%A6%E7%94%9Ft-%E5%88%86%E5%B8%83, 2019-08-19.

[8] Wikipedia. 假设检验 [EB/OL]. https://zh.jinzhao.wiki/zh-cn/%E6%AD%A3%E6%80%81%E5%88%86%E5%B8%83, 2019-07-07.

附件:

代码可在下面二个网址的任意一个内获取。

https://github.com/septsea/covid\_19\_mm

https://gitee.com/septsea/covid\_19\_mm

*File: MyX509TrustManager.java*

package numberOfCommits

import java.security.cert.CertificateException

import java.security.cert.X509Certificate

import javax.net.ssl.X509TrustManager

/\*\*

\* This is a trust manager doing nothing but enabling HTTPS connections.

\*/

public class MyX509TrustManager implements X509TrustManager

@Override

public void checkClientTrusted(final X509Certificate[] chain,

final String authType) throws CertificateException

@Override

public void checkServerTrusted(final X509Certificate[] chain,

final String authType) throws CertificateException

@Override

public X509Certificate[] getAcceptedIssuers()

return null

*File: Crawler.java*

package numberOfCommits

import java.io.BufferedReader

import java.io.BufferedWriter

import java.io.FileReader

import java.io.FileWriter

import java.io.InputStreamReader

import java.net.URL

import java.text.ParseException

import java.text.SimpleDateFormat

import java.util.Calendar

import javax.net.ssl.HttpsURLConnection

import javax.net.ssl.SSLContext

import javax.net.ssl.TrustManager

import org.json.JSONArray

/\*\*

\* The core.

\*/

public class Crawler

/\*\*

\* Do an HTTPS request.

\*

\* @param requestUrl the request HTTPS URL.

\* @param requestMethod the request method, one of "GET", "POST", "HEAD",

\* "OPTIONS", "PUT", "DELETE", "TRACE". If you do not know

\* what to use, set it "GET".

\* @param outputStr string to be written to the server. If no string is

\* needed to be written, set it {@code null}.

\* @return the string sent by the server.

\*/

public static String httpsRequest(final String requestUrl,

final String requestMethod, final String outputStr)

StringBuffer buffer = null

try

// Create SSLContext

final var sslContext = SSLContext.getInstance("SSL")

final var tm = new TrustManager[] { new MyX509TrustManager() }

// Initialize

sslContext.init(null, tm, new java.security.SecureRandom())

// Get SSLSocketFactory object

final var ssf = sslContext.getSocketFactory()

final var url = new URL(requestUrl)

final var conn = (HttpsURLConnection) url.openConnection()

conn.setDoOutput(true)

conn.setDoInput(true)

conn.setUseCaches(false)

// Cheat the API

final var key = "User-Agent"

var value = "Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit"

value +=

"/537.36 (KHTML, like Gecko) Chrome/61.0.3163.100 Safari/537.36)"

conn.setRequestProperty(key, value)

conn.setRequestMethod(requestMethod)

// Set up SSLSoctetFactory

conn.setSSLSocketFactory(ssf)

conn.connect()

// Write a string to server

if (null != outputStr)

final var os = conn.getOutputStream()

os.write(outputStr.getBytes("utf-8"))

os.close()

// Read content sent by server

final var is = conn.getInputStream()

final var isr = new InputStreamReader(is, "utf-8")

final var br = new BufferedReader(isr)

buffer = new StringBuffer()

String line = null

while ((line = br.readLine()) != null)

buffer.append(line)

catch (final Exception e)

e.printStackTrace()

return buffer.toString()

/\*\*

\* Give how many commits there are on a specified date.

\*

\* @param owner the owner of the repository.

\* @param repo the name of the repository.

\* @param date Only commits on this date will be returned. This is a timestamp

\* in ISO 8601 format: YYYY-MM-DDTHH:MM:SSZ.

\* @return the number of commits.

\*/

public static int numberOfCommits(final String owner, final String repo,

final String date)

final var since = date

final var until = addDays(date, 1)

return numberOfCommits(owner, repo, since, until)

/\*\*

\* Give how many commits there are during a specified period.

\*

\* @param owner the owner of the repository.

\* @param repo the name of the repository.

\* @param since Only commits after this date will be returned. This is a

\* timestamp in ISO 8601 format: YYYY-MM-DDTHH:MM:SSZ.

\* @param until Only commits before this date will be returned. This is a

\* timestamp in ISO 8601 format: YYYY-MM-DDTHH:MM:SSZ.

\* @return the number of commits.

\*/

public static int numberOfCommits(final String owner, final String repo,

final String since, final String until)

final var link = new StringBuffer()

link.append("https://gitee.com/api/v5/repos/")

link.append(owner)

link.append("/")

link.append(repo)

link.append("/commits?&since=")

link.append(since)

link.append("&until=")

link.append(until)

// At most 100 records per page.

// So if there is a date when num of commits >= 100,

// recheck the date manually.

link.append("&page=1&per\_page=100")

final var json = new JSONArray(httpsRequest(link.toString(), "GET", null))

return json.length()

/\*\*

\* Read a txt file.

\*

\* @param txtFile the path of the txt file.

\* @return the string of the file.

\*/

public static String txtReader(final String txtFile)

final var sb = new StringBuffer()

try

final var br = new BufferedReader(new FileReader(txtFile))

String s = null

while ((s = br.readLine()) != null)

sb.append(s)

sb.append("\n")

br.close()

catch (final Exception e)

e.printStackTrace()

return sb.toString()

/\*\*

\* Write a string to a file.

\*

\* @param txtFile the txt file to be written to.

\* @param str the string to be written.

\* @param append if {@code true}, then data will be written to the end of the

\* file rather than the beginning.

\*/

public static void txtWriter(final String txtFile, final String str,

final boolean append)

try

final var bw = new BufferedWriter(new FileWriter(txtFile, append))

bw.write(str)

bw.close()

catch (final Exception e)

e.printStackTrace()

/\*\*

\* Add or subtract the specified amount of time to the given calendar field,

\* based on the calendar's rules.

\*

\* @param date the date to be added.

\* @param n the amount of days to be added to the date. Negative integers are

\* allowed.

\* @return the new date.

\*/

public static String addDays(final String date, final int n)

final var sdf = new SimpleDateFormat("yyyyMMdd")

String newDate = null

try

final var originalDate = sdf.parse(date)

final var c = Calendar.getInstance()

c.setTime(originalDate)

c.add(Calendar.DATE, n)

newDate = sdf.format(c.getTime())

catch (final ParseException e)

e.printStackTrace()

return newDate

*File: FetchInstance.java*

package numberOfCommits

public class FetchInstance

public static void main(final String... args)

fetch("atom")

fetch("brackets")

fetch("vscode")

public static void fetch(final String repo)

// It takes long time to load GitHub,

// so we manually clone the repositories.

// Hence the owner is not the original owner.

final var owner = "septsea"

// It is 140 days from the start date = 20191229

// to the end date = 20200516, end date included

var date = "20200516"

var i = 0

while (i < 140)

final var data = date + "," + Crawler.numberOfCommits(owner, repo, date)

Crawler.txtWriter(repo + "\_After.txt", data + "\n", true)

System.out.println(data)

date = Crawler.addDays(date, -1)

i -=- 1

// It is 140 days from the start date = 20190811

// to the end date = 20191228, end date included

date = "20191228"

i = 0

while (i < 140)

final var data = date + "," + Crawler.numberOfCommits(owner, repo, date)

Crawler.txtWriter(repo + "\_Before.txt", data + "\n", true)

System.out.println(data)

date = Crawler.addDays(date, -1)

i -=- 1

*File: Stat.java*

package numberOfCommits

import org.apache.commons.math3.distribution.TDistribution

import org.apache.commons.math3.stat.descriptive.DescriptiveStatistics

public class Stat

public static void main(final String... args)

statAnalysis("atom")

System.out.println("\n================\n")

statAnalysis("brackets")

System.out.println("\n================\n")

statAnalysis("vscode")

public static void statAnalysis(final String repo)

// Each file has 140 pieces of information

var size = 140

final var before\_day = new int[size]

final var after\_day = new int[size]

final var data\_0 = Crawler.txtReader(repo + "\_Before.txt").split("\n")

final var data\_1 = Crawler.txtReader(repo + "\_After.txt").split("\n")

var i = 0

while (i < size)

// Each line is like "20200202,100"

before\_day[i] = Integer.parseInt(data\_0[i].substring(9))

after\_day[i] = Integer.parseInt(data\_1[i].substring(9))

i -=- 1

final var before\_week = new double[size / 7]

final var after\_week = new double[size / 7]

i = 0

while (i < 20)

var s = 0

var t = 0

var j = 0

while (j < 7)

s -=- before\_day[i \* 7 + j]

t -=- after\_day[i \* 7 + j]

j -=- 1

before\_week[i] = s

after\_week[i] = t

i -=- 1

final var before = new DescriptiveStatistics()

final var after = new DescriptiveStatistics()

i = 0

// We analyse weekly data

size = 20

while (i < size)

before.addValue(before\_week[i])

after.addValue(after\_week[i])

i -=- 1

final var before\_mean = before.getMean()

final var before\_std = Math.sqrt(before.getVariance())

final var after\_mean = after.getMean()

final var after\_std = Math.sqrt(after.getVariance())

System.out.println(repo + ":")

System.out.println()

System.out.println("Before COVID-19:")

System.out.println("mean: " + before\_mean)

System.out.println("standard deviation: " + before\_std)

System.out.println()

System.out.println("After COVID-19:")

System.out.println("mean: " + after\_mean)

System.out.println("standard deviation: " + after\_std)

final var s0 = Math.sqrt((Math.pow(before\_std, 2)

+ Math.pow(after\_std, 2)) / size)

final var tmp1 = Math.pow(before\_std, 4) + Math.pow(after\_std, 4)

final var tmp2 = Math.pow(s0, 4) \* size \* size \* (size - 1)

final var l = (int) Math.round(tmp2 / tmp1)

System.out.println()

System.out.println(

"(y^bar - x^bar - mu\_2 + mu\_1)/s0 is approximately t-distributed."

)

System.out.println()

System.out.println("s0 = " + s0)

System.out.println("degree of freedom = " + l)

// Perform a fixed significance level test with alpha = 0.05

// Hypothesis: after\_mean >= before\_mean

final var alpha = 0.05

final var tDist = new TDistribution(l)

final var t\_alpha = tDist.inverseCumulativeProbability(alpha)

final var t1 = (after\_mean - before\_mean) / s0

System.out.println()

System.out.println("alpha = " + alpha)

System.out.println("t\_alpha = " + t\_alpha)

System.out.println("t = " + t1)

System.out.println()

if (t1 <= t\_alpha)

System.out.println("It cannot be drawn that after\_mean >= before\_mean.")

else

System.out.println("It can be drawn that after\_mean >= before\_mean.")