- p值的问题
  - p值到底是干什么的？ 在假设检验问题中有一个非常基本的问题。每一个假设检验都有一个原假设和一个备择假设。假定原假设是事实， 那么备择假设就有了一个可以计算出来的发生概率，如果这个发生概率非常小，但是实际上却发生了，我们就拒绝原假设，认为原假设不正确， 并把原假设正确备择假设成立的概率叫做p值。
  - 因为p值很容易根据一个异常案例决定是否改变现有的对事实的认知，所以被广泛使用来推翻现有的认知，但是p值的误用也是广泛存在的。
  - 这篇论文主要对一些常见的计算p值的方法进行汇总分析，来论述为什么p值会被如此广泛的误用。
  - 这篇论文主要涉及了信息论和非参数统计，经过我的搜索，并没有太多的同类文献可以参考，所以实际上基本上没有参考来源和同行研究。
  - 为了让文章的观点更容易接受，我们会从案例的角度来论述这篇文章
  - p值是怎么工作的
    - 0假设（null hypothesis)H0: a的均值和b的均值一样大, 备择假设H1(alter hypothesis): a的均值和b的均值不一样大
      - 我们计算在0假设的情况下，现实情况发生的可能性，0假设是欧洲一年喝的啤酒的数量是8升左右，备择假设是欧洲人喝的啤酒数量和8升相差很远。
      - 实际的情况，我们有10个人喝啤酒，他们喝啤酒的数量是5 7 8 8 8 2 3 4 3 4
      - 我们按照平均水平是8升的情况下按照一定的计算方法计算现实情况，或者比现实情况更为远离均值的情况发生的概率，也就得到了一个p值，
      - 通过简单的计算我们实际上的喝酒的情况平均水平达不到8升，接着我们计算出来了一个在0假设下小于现实情况的概率，假定这个概率是0.01
      - 因为如果平均值是8升的话，我们的得到现实情况是这样的概率是0.01相对来说比较小，所以，我们不认为0假定均值为8升是对的，
  - 如果我们想要思考一个问题，我们必须有一个度量的尺子，来度量事情的复杂度。
    下面采用的方法： 在每一个需要人为判断的环节标记一个bit的信息，然后计算整体的计算过程中需要多少bit的信息， 以信息为例
  - 问题复杂度的度量的方法，在有分支的情况下，不管左右分支的先验信息，每条边分支的概率都假设成1/2.
  - R语言的例子1 判断组函数的中位数是否是一个数
    - x=c(274,279,290,326,329,341,378,405,436,500, 515,541,558,566,618,708,760,867,868,869, 888,915,932,942,960,975,976,1014,1025,1095, 1118,1166,1193,1194,1243,1277,1304,1327,1343,1398, 1407,1409,1417,1467,1477,1512,1530,1623,1710,1921) #代表一个有多个数值的样本
      #检验中位数是否是1200
      n=length(x) # 计算一共有多少数据
      median(x) #计算实际的这一组数据的中位数的数值
      p0=0.5 q0=1200 alpha=0.05 初始化参数，0假设的中位数， 以及显著性水平0.05
      sfu=sum(x<=q0) #计数所有小于0假设的中位数的数量
      low=qbinom(alpha/2,n,p0) 计算给定显著性水平置信区间的下分位点
      upp=qbinom(1-alpha/2,n,p0)计算给定显著性水平置信区间的上分位点
      p=2*min(pbinom(sfu,n,p0), 1-pbinom(sfu-1,n,p0)) 计算p值
      - 说明
        - 说明： 给出一组数据，检验中位数是否是某一个数，解出p值得到结论。
        - 这是一个最简单的例子， 也是一个最为常用的例子， 我们来检查一下计算p值的几个环节，以及可能出错的部分。
      - 需要人为判断的环节
        - sfu=sum(x<=q0) #计数所有小于0假设的中位数的数量
          因为不同人的习惯不同，这个地方会涉及到有一部分人默认计数了大于假定中位数的数量，他的代码可能是这样的。
          sfu=sum(x>=q0)
          也就是任何审阅代码的人必须切实的理解这一部分， 已检查后来的计算是否出现了稿件的作者是不是将第一种sfu当成 第二种sfu进行计算
        - p=2*min(pbinom(sfu,n,p0), 1-pbinom(sfu-1,n,p0)) 计算p值
          这里有两个需要判断的环节，一个是前面的默认计数对象sfu， 如果前面的计数sfu是小于假定中位数的数量，那么就应该使用上面的代码，否者应该使用下面的代码
          p=2*min(1-pbinom(sfu,n,p0), pbinom(sfu-1,n,p0))
          这个问题的真正简单之处是，即便作者搞错了前后参数的对应关系， 2*min() 这个取两个数中较小值的函数也能够防止出错，
          但是对于审核的人仍然意味着需要搞清每一个参数的真实意义。
          以及需要放在论文中的位置
      - 总结
        - 总结： 这个计算一共需要做统计计算的人搞清楚两处参数的位置，即便搞错了，也能得到正确的结果，
          对于作者来只需要按着流程使用，不需做反复的检查，0bit的复杂度
          对于统计审核的人容易搞反的地方有两处，需要检查的地方有两个地方，检查的复杂度是2bit
  - R语言中的例子2
    判断一个函数的0.25分位点是不是一个数值
    - x=c(274,279,290,326,329,341,378,405,436,500, 515,541,558,566,618,708,760,867,868,869, 888,915,932,942,960,975,976,1014,1025,1095, 1118,1166,1193,1194,1243,1277,1304,1327,1343,1398, 1407,1409,1417,1467,1477,1512,1530,1623,1710,1921)
      #检验0.25分位数是否是1000
      n=length(x) 计算一共有多少数据
      p0=0.25 q0=1000 alpha=0.05 初始化参数，0假设的中位数， 以及显著性水平0.05
      sfu=sum(x<=q0) #计数所有小于0假设的中位数的数量
      p=2*min(pbinom(sfu,n,p0), 1-pbinom(sfu-1,n,p0)) 计算p值
      - 说明
        - 判断一个函数的0.25分位点是不是一个数值
          sfu=sum(x<=q0) #计数所有小于0.25分位数的数量
          因为不同人的习惯不同，这个地方会涉及到有一部分人默认计数了大于假定中位数的数量，他的代码可能是这样的。
          sfu=sum(x>=q0)
          也就是任何审阅代码的人必须切实的理解这一部分， 已检查后来的计算是否出现了稿件的作者是不是将第一种sfu当成 第二种sfu进行计算
      - 需要人为判断的环节
        - p=2*min(pbinom(sfu,n,p0), 1-pbinom(sfu-1,n,p0)) 计算p值
          这里有两个需要判断的环节，一个是前面的默认计数对象sfu， 如果前面的计数sfu是小于假定0.25分位数的数量，那么就应该使用上面的代码，否者应该使用下面的代码
          p=2*min(1-pbinom(sfu,n,p0), pbinom(sfu-1,n,p0))
          这个案例并没有类似于上面的检验中位数时的防止出错机制，如果弄错了顺序，就会真正的出错
          但是对于审核的人意味着需要搞清每一个参数的真实意义。
          以及前后的承接关系
        - 总结： 这个计算一共需要做统计计算的人搞清楚两处参数的位置，即便搞错了，也能得到正确的结果，
          对于作者来只需要按着流程使用，不需做反复的检查，0bit的复杂度
          对于统计审核的人容易搞反的地方有两处，需要检查的地方有两个地方，检查的复杂度是2bit
      - 总结：需要人为判断的环节 和上面类似，不过这个案例没有防止出错的机制，如果作者搞错了，就真的搞错了
        也就是对于作者来说复杂度提升到了2bit，
        对于审核人员来说，仍然维持2bit
  - R语言中的例子3
    判断一个函数的0.25分位点是不是小于一个给定的数值多少
    - x=c(274,279,290,326,329,341,378,405,436,500, 515,541,558,566,618,708,760,867,868,869, 888,915,932,942,960,975,976,1014,1025,1095, 1118,1166,1193,1194,1243,1277,1304,1327,1343,1398, 1407,1409,1417,1467,1477,1512,1530,1623,1710,1921) #原假设： 0.25分位数就是1000
      #备择假设0.25分位数小于1000
      n=length(x) 计算一共有多少数据
      p0=0.25 q0=1000 alpha=0.05 初始化参数，0假设的中位数， 以及显著性水平0.05
      sfu=sum(x<=q0) #计数所有小于0假设的中位数的数量
      p = 1-pbinom(sfu,n,p0) 计算p值
      - 说明
        - 判断一个函数的0.25分位点是不是小于一个给定的数值多少
          这个案例比较复杂，通过逆推法来决定p值是如何确认的
          最后的p = 1-pbinom(sfu,n,p0)是如何确认的，p值计算的过程
          首先p值一定是过小拒绝原假设，
          我们构造一个小的p值，
          因为拒绝原假设会导致一个必然很小的p值，
          拒绝原假设的情况是 实际0.25分位数小于1000
          实际0.25分位数小于1000，也就会造成实际小于1000的数量比较多
          根据分布函数的规律，应该用1-pbinom(sfu,n,p0)得到一个小值，
          所以 p 值的计算应该取 1-pbinom(sfu,n,p0)
          整个过程一共有8个思考步骤，中间设计防止出错的机制，和第一个例子和第二个例子不同，简单的改了一下问题要求的内容，需要人为进行检查是否正确的环节变得空前的多，而且整个问题彻底失去了防止出错的机制。
          整个个问题的复杂程度是8bit
      - 总结：需要人为判断的环节 和上面类似，不过这个案例没有防止出错的机制，如果作者搞错了，就真的搞错了
        也就是对于作者来说复杂度提升到了8bit，
        对于审核人员来说，仍然维持8bit
      - 出现的灾难性后果： 人为检查的环节过多，是造成p值在使用过程中耗时比较长的首要原因，不过也因此造成了另一个更为严重的问题， 采用p值的研究即便有错误，检查起来也难如登天。
        而通过几个论文审核人员进行统计审核的排错能力也会大大下降。
        假定一片论文有5个的逻辑错误的地方，而这五个逻辑错误的的地方又相互关联，前后承接，这样一共要在32中情况中找出来一个对的情况，即便有三个人同时参与这篇论文的统计审核，每个人犯错的概率30%，能够保证这篇论文是对的概率也不过1-,3^3 =0.973， 0.027的概率把事实搞成了另外一种情况，这也就意味着， 即便是三个人检查，也有0.027^32 = 0.864的概率是另外的一种结果， 换句话说，因为p值的没有一个能够自动纠错的机制，统计审核的成本变得惊人的高。 在三个人检查的情况下，依然有相当大的犯错概率，而现实中是不可能有这么多人对一篇论文进行检查的。论文的统计意义因此而大打折扣。
  - 论文的总结
    - 上面的例子是p值使用中最为简单的例子，即便是简单的p值的统计可靠性，也很依赖人为检查，这个时候因为审阅论文的人对论文本身的不熟悉的。
      导致从头到尾进行统计审核，审核的时间会比写论文的时间还要长，如果进行了半年统计审核，到头来发现这个论文根本就站不住脚。也就是会面临大量的沉没成本
  - 结论
    - 我们仅仅拥有p值计算的知识是不够的，在绝大多数的关于p值使用过程中， 都会给出正确的使用方式， 但是在正确的道路上前行需要检查众多的参数对应关系，导致的出错概率很大。
    - tips: 人为检查的环节过多，是造成p值在使用过程中耗时比较长的首要原因，不过也因此造成了另一个更为严重的问题， 采用p值的研究即便有错误，检查起来也难如登天。
      而通过几个论文审核人员进行统计审核的排错能力也会大大下降。
      假定一片论文有5个的逻辑错误的地方，而这五个逻辑错误的的地方又相互关联，前后承接，这样一共要在32中情况中找出来一个对的情况，即便有三个人同时参与这篇论文的统计审核，每个人犯错的概率30%，能够保证这篇论文是对的概率也不过1-,3^3 =0.973， 0.027的概率把事实搞成了另外一种情况，这也就意味着， 即便是三个人检查，也有0.027^32 = 0.864的概率是另外的一种结果， 换句话说，因为p值的没有一个能够自动纠错的机制，统计审核的成本变得惊人的高。 在三个人检查的情况下，依然有相当大的犯错概率，而现实中是不可能有这么多人对一篇论文进行检查的。论文的统计意义因此而大打折扣。
    - history tips： 在很多我以前的高中时代的数学计算中，比如乘法运算中，也会出现大量的问题，没有防出错机制， 每一次计算都要同时审核是不是算错，进行多次验证， 这样就造成了计算比想象的要费时，因为真实的计算是在复杂的情况中搜索到正确的答案
    - later for deal: 我或许要计算一下真正的计算中容易范的错误。
  - 待处理的观点
    - P值真正复杂的地方是没有任何一个地方提供一个完整的解决方案，没有通用的解决方案的后果就是参数列表不一致，绝对多数类似于 phypper 的程序没有任何的防止参数代入错误的防呆机制，一次参数的检查可能没有什么，但是如果出现大量的参数的检查，就会变成非常复杂的的问题。
    - 在这里举一个例子搞定p值使用的问题， 具体的读及其中涉及的可能的错误。 p值的意义， 我精确的知道信息论，以及香农熵的计算方法， 所以我可以对整个步骤计算一下信息熵，然后评估一下，
    - tips: 探索新的事物会让我感到劳累，所以要对自己的精力的下降保持警觉