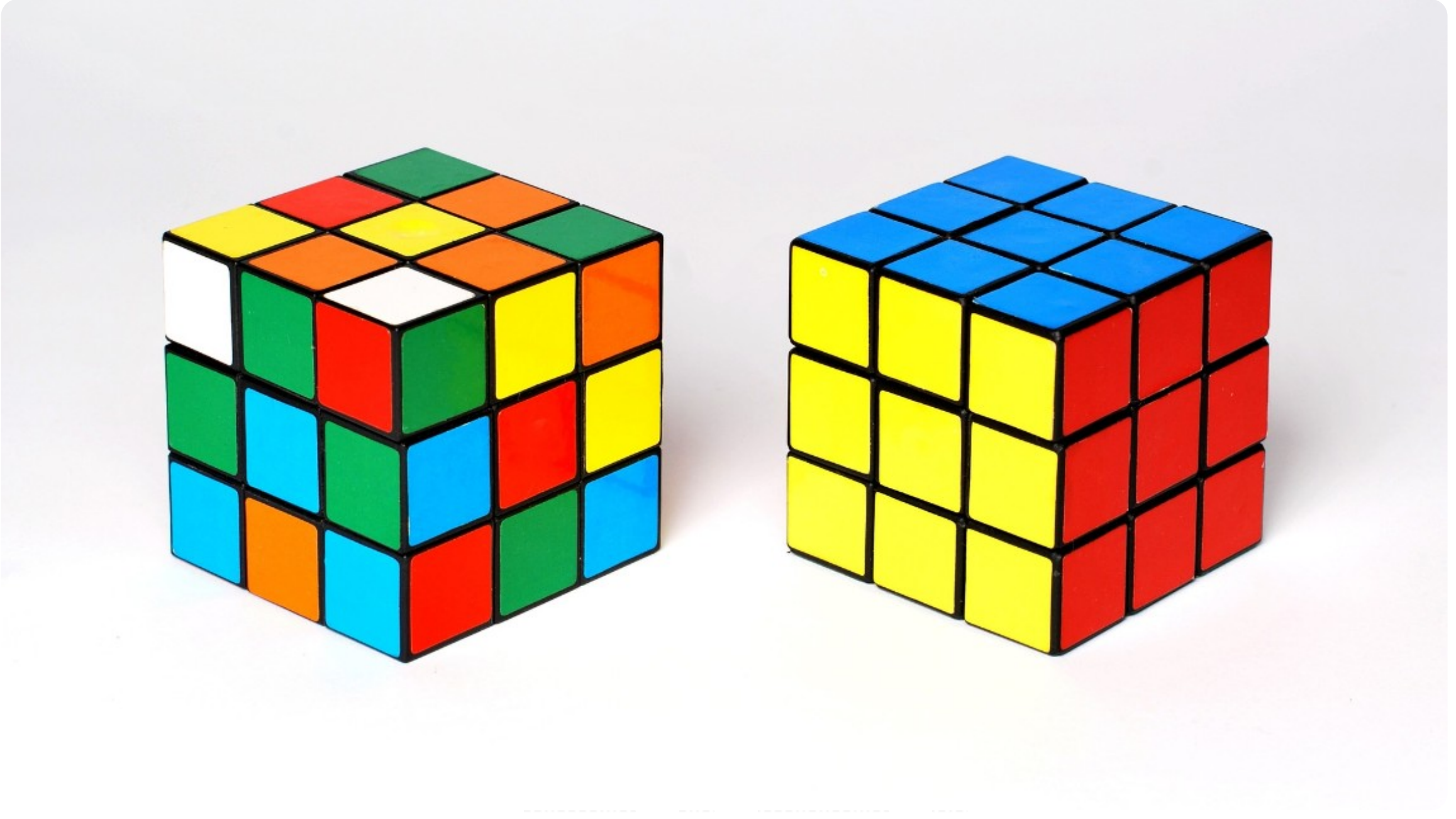
24 | A/B 测试：验证决策效果的利器

戴铭 2019-05-04



00:00 09:58

讲述：冯永吉 大小：9.14M

你好，我是戴铭。今天，我来跟你聊聊验证决策的利器 A/B 测试。

现在 App Store 中的应用，就像商场中的商品一样琳琅满目，可以解决用户各个方面的需求。这时，你要想创新，或者做出比竞品更优秀的功能，是越来越不容易。所以，很多公司都必须去做一些实验，看看有哪些功能可以增强自己 App 的竞争力，又有哪些功能可以废弃掉。而进行这样的实验的主要方法，就是 A/B 测试。

A/B 测试，也叫桶测试或分流测试，指的是针对一个变量的两个版本 A 和 B，来测试用户的不同反应，从而判断出哪个版本更有效，类似统计学领域使用的双样本假设测试。

简单地说，A/B 测试就是检查 App 的不同用户在使用不同版本的功能时，哪个版本的用户反馈最好。

比如，引导用户加入会员的按钮，要设置为什么颜色更能吸引他们加入，这时候我们就需要进行 A/B 测试。产品接触的多了，我们自然清楚一个按钮的颜色，会影响到用户点击它，并进入会员介绍页面的概率。

这里我再和你分享一件有意思的事儿。记得我毕业后去新西兰的那段时间里，认识了一个住在海边的油画家，她在海边还有一间画廊，出售自己的作品还有美院学生的作品。

有一天她要给画廊门面重涂油漆，叫我过去帮忙。涂漆之前问我用什么颜色好，我环顾下了旁边的店面，大多是黑色、灰色和深蓝色，而我觉得卖橄榄球衣服那家的黑底红字，看起来很帅气，于是就说黑色可能不错。

她想了想摇头说：我觉得橙色好，因为这附近都是暗色调，如果用了明亮的橙色可能更容易吸引游客。结果呢，后来一段时间进店的人确实多了，而且画也卖得多了。

当然了，我举这个例子的目的不是说用了橙色就一定能够提高用户进店率。试想一下，如果这个画廊周围都是花花绿绿的店面，你还能够保证橙色会吸引用户吗。

实际情况往往要比选择门面颜色更复杂，也只有有专业经验的人才可以做出正确的决策，但并不是每个人都是有相关领域经验的专家。所以，就有了 A/B 测试这一利器，来辅助我们进行决策。

知乎上有个关于A/B 测试的问答，里面列举了很多关于实际案例，有兴趣的话你可以去看看。接下来，我和你说说 iOS 中的 A/B 测试。

App 开发中的 A/B 测试

从 App 开发层面看，新版本发布频繁，基本上是每月或者每半月会发布一个版本。那么，新版本发布后，我们还需要观察界面调整后情况如何，性能问题修复后线上情况如何，新加功能使用情况如何等。这时，我们就需要进行 A/B 测试来帮助我们分析这些情况，通过度量每个版本的测试数据，来确定下一个版本应该如何迭代。

对于 App 版本迭代的情况简单说就是，新版本总会在旧版本的基础上做修改。这里，我们可以把旧版本理解为 A/B 测试里的 A 版本，把新版本理解为 B 版本。在 A/B 测试中 A 版本和 B 版本会同时存在，B 版本一开始是将小部分用户放到 B 测试桶里，逐步扩大用户范围，通过分析 A 版本和 B 版本的数据，看哪个版本更接近期望的目标，最终确定用哪个版本。

总的来说，A/B 测试就是以数据驱动的可回退的灰度方案，客观、安全、风险小，是一种成熟的试错机制。

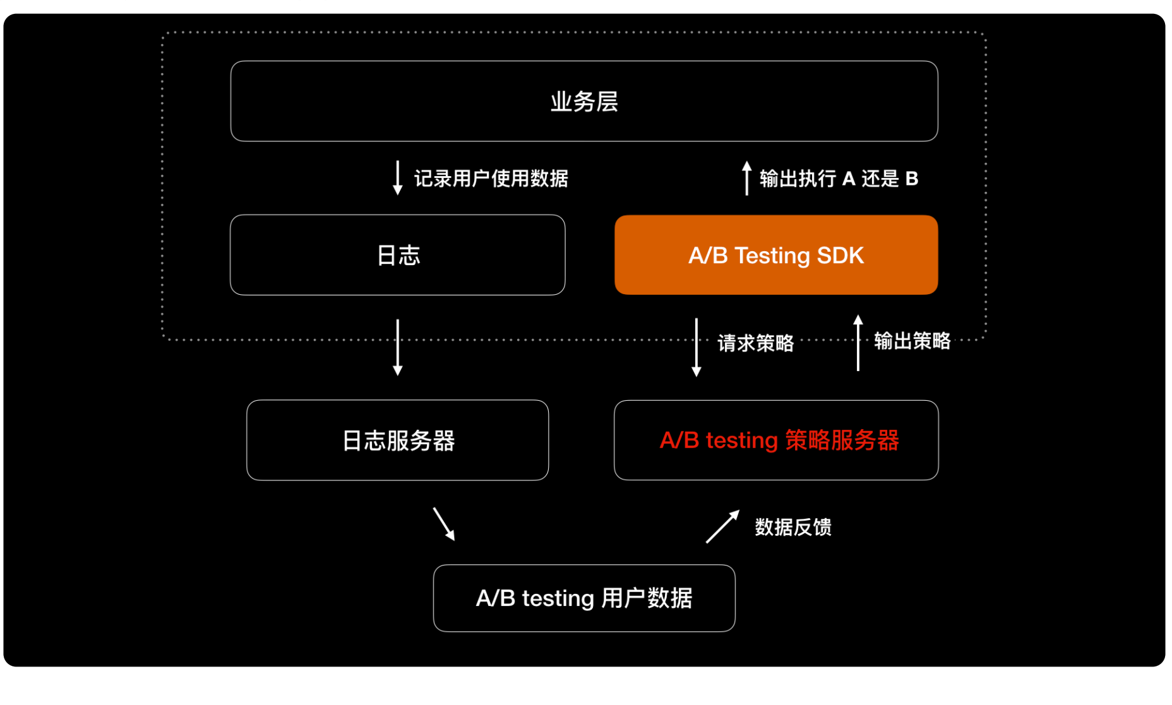
A/B 测试全景设计

一个 A/B 测试框架主要包括三部分：

1. 策略服务，为策略制定者提供策略；
2. A/B 测试 SDK，集成在客户端内，用来处理上层业务去走不同的策略；
3. 日志系统，负责反馈策略结果供分析人员分析不同策略执行的结果。

其中，策略服务包含了决策流程、策略维度。A/B 测试 SDK 将用户放在不同测试桶里，测试桶可以按照系统信息、地址位置、发布渠道等来划分。日志系统和策略服务，主要是用作服务端处理的，这里我就不再展开了。

下图是 A/B 测试方案的结构图：



今天我主要跟你说下客户端内的 A/B 测试 SDK。从 iOS 开发者的角度看 A/B 测试，如何设计或选择一个好用的 A/B 测试 SDK 框架才是我们最关心的。

A/B 测试 SDK

谈到 A/B 测试 SDK 框架，我们需要首先要考虑的是生效机制。生效机制主要分为冷启动生效和热启动生效，相对于冷启动，热启动落实策略要及时些。但是，考虑到一个策略可能关联到多个页面或者多个功能，冷启动可以保持策略整体一致性。

所以我的结论是，**如果一个策略只在一个地方生效的话，可以使用热启动生效机制；而如果一个策略在多个地方生效的话，最好使用冷启动生效机制。**

除了生效机制，A/B 测试 SDK 框架对于业务方调用接口的设计也很重要。你所熟悉的著名 AFNetworking 网络库和 Alamofire 网络库的作者 Mattt ，曾编写过一个叫作SkyLab的 A/B 测试库。

SkyLab 使用的是 NSUserDefault 保存策略，使得每个用户在使用过程中，不管是在哪个测试桶里，都能够保持相同的策略。 SkyLab 对外的调用接口，和 AFNetworking 一样使用的是 Block ，来接收版本 A 和 B 的区别处理。这样设计的接口易用性非常高。

通过 SkeyLab 原理的学习，你能够体会到如何设计一个优秀易用的接口。这，对你开发公用库的帮助会非常大。

接下来，我们先看看 SkeyLab 接口使用代码，示例如下：

复制代码

```
// A/B Test
[SkyLab abTestWithName:@"Title" A:^{
    self.titleLabel.text = NSLocalizedString(@"Hello, World!", nil);
} B:^{
    self.titleLabel.text = NSLocalizedString(@"Greetings, Planet!", nil);
}];

```

可以看出，Mattt 这个人的接口设计功底有多强了。你一看这两个 block 参数名称，就知道是用来做 A/B 测试的，简单明了。接下来，我们再进入接口看看 Mattt 是具体怎么实现的。

复制代码

```
+ (void)abTestWithName:(NSString *)name
                     A:(void (^)())A
                     B:(void (^)())B
{
    [self splitTestWithName:name conditions:[NSArray arrayWithObjects:@"A", @"B"
        if ([choice isEqualToString:@"A"] && A) {
            // 执行版本 A
            A();
        } else if ([choice isEqualToString:@"B"] && B) {
            // 执行版本 B
            B();
        }
    }];
}

```

你会发现 SkyLab:abTestWithName:A:B: 方法只是一个包装层，里面真正的实现是 SkyLab:splitTestWithName:conditions:block 方法，其定义如下：

复制代码

```
+ (void)splitTestWithName:(NSString *)name
               conditions:(id <NSFastEnumeration>)conditions
                    block:(void (^)(id condition))block;

```

通过定义你会发现，conditions 参数是个 id 类型，通过类型约束，即使用 NSFastEnumeration 协议进行了类型限制。Mattt 是希望这个参数能够接收字典和数组，而字典和数组都遵循 NSFastEnumeration 协议的限制，两者定义如下：

复制代码

```
@interface NSDictionary<__covariant KeyType, __covariant ObjectType> : NSObject

@interface NSArray<__covariant ObjectType> : NSObject <NSCopying, NSMutableCopyi

```

在这里，我和你介绍这个接口的设计方式，是因为这个设计非常赞，非常值得我们学习。类型约束，是苹果公司首先在 Swift 泛型引入的一个特性，后来引入到了 Objective-C 中。

而之所以设计 conditions 这个支持数组和字典的参数，本来是为了扩展这个 SkyLab 框架，使其不仅能够支持 A/B 测试，还能够支持更为复杂的 Multivariate testing或 Multinomial testing。Multivariate testing 和 Multinomial testing 的区别在于，支持更多版本变体来进行测试验证。

接下来，我们再看看 SkyLab 是如何做人群测试桶划分的。

SkyLab 使用的是随机分配方式，会将分配结果通过 NSUserDefaults 进行持续化存储，以确保测试桶的一致性。其实测试桶分配最好由服务端来控制，这样服务端能够随时根据用户群的维度分布分配测试桶。

如果你所在项目缺少服务端支持的话，SkyLab 对测试桶的分配方式还是非常值得借鉴的。SkyLab 对 A/B 测试的测试桶分配代码如下：

复制代码

```
static id SLRandomValueFromArray(NSArray *array) {
    if ([array count] == 0) {
        return nil;
    }
    // 使用 arc4random_uniform 方法随机返回传入数组中某个值
    return [array objectAtIndex:(NSUInteger)arc4random_uniform([array count])];
}

```

代码中的 array 参数就是包含 A 和 B 两个版本的数组，随机返回 A 版本或 B 版本，然后保存返回版本。实现代码如下：

复制代码

```
condition = SLRandomValueFromArray(mutableCandidates);
// 判断是否需要立刻进行同步保存
BOOL needsSynchronization = ![condition isEqual:[[NSUserDefaults standardUserDef
// 通过 NSUserDefaults 进行保存
[[NSUserDefaults standardUserDefaults] setObject:condition forKey:SLUserDefaults
if (needsSynchronization) {
    [[NSUserDefaults standardUserDefaults] synchronize];
}

```

持久化存储后，当前用户就命中了 A 和 B 版本中的一个，后续的使用会一直按照某个版本来，操作的关键数据会通过日志记录，并反馈到统计后台。至此，你就可以通过 A、B 版本的数据比较，来决策哪个版本更优了。

小结

今天我跟你说了 A/B 测试在产品中的重要性，特别是在 App 版本迭代时，A/B 测试可以帮助我们判断新版本的功能更新是否能够更好地服务用户。然后，我为你展示了 A/B 测试方案的全景设计，并针对其中 iOS 开发者最关注的 A/B 测试 SDK 的设计做了详细分享。

通过 Mattt 设计的 SkyLab 这个 A/B 测试 SDK 框架，你会发现好的接口设计不是凭空想出来的，而是需要一定的知识积累。比如，将泛型的类型约束引入到 Objective-C 中以提高接口易用性，这需要了解 Swift 才能够做到的。

今天我在看评论区的留言时，有同学问我现在应该学习 Objective-C 还是 Swift，为什么？我想，我们今天对 SkyLab 接口的分析应该就是最好的回答了。知识的学习最好结合工作需求来，无论是 Objective-C 还是 Swift，最重要的还是代码设计能力。

课后作业

今天我留给你一个作业，前面我提到 Swift 是值得学习的，那么今天的作业就是参照 SkyLab，使用 Swift 来写一个 A/B 测试 SDK。

感谢你的收听，欢迎你在评论区给我留言分享你的观点，也欢迎把它分享给更多的朋友一起阅读。

吴开

由作者筛选后的优质留言将会公开显示，欢迎踊跃留言。

Command + Enter 发表 0/2000字 提交留言

精选留言(7)

Mhy

对于现在想学iOS开发我觉得要分两种 一种是以iOS开发谋生的 一种是想扩充自己技能树或者想做个全栈的 对于第一种OC和Swift应该是要都去了解 对于第二种 我觉得直接上手Swift就好了 Swift设计的理念就是让开发更简单(但是从现在来看 加入各种语法糖 想要包含其他语言的优秀特性反而让他变得越发不纯粹 相比较Go做的就很好 一直秉承着简单干净的风格) Swift也是苹果的亲儿子 跟着苹果爸爸走肯定不会错的 我想要问老师的问题是对于iOS开发的出路到底在哪 我觉得未来移动开发的趋势肯定是要偏向大前端了 大前端的天然优势是更少的成本还有更快的迭代速度 这也是企业一直在追求的 请问老师的看法呢

2019-05-04 7

尘归心

我和Mhy同学有相同的疑问，特别是以后5G时代的到来，网速不在是限制前端体验的元素，原生的意义是否会被近一步削弱，就算是很精良的纺织工也一样会被纺织机所替代，所以现在对于继续IOS原生开发很是迷茫

2019-05-05 5

鼠辈

最近一直在找一个好的AB测试的SDK，不知道作者之前用过什么好的AB测试的SDK(三方的，可以后台控制的)

2019-05-05 3

data

学习优秀的框架提升自己的水平，并且逐步掌握底层技术，因为很多技术点都是相同的，就算换语言，你也更容易入门的

2019-05-05 1