> SAMPLE SIZE MDE

> «Классический» подход и калькуляторы размеров выборки

Прежде всего рассмотрим довольно известную формулу, которую вы уже могли видеть:

$$n > \frac{[\Phi^{-1}(1-\alpha/2) + \Phi^{-1}(1-\beta)]^2(\sigma_X^2 + \sigma_Y^2)}{\epsilon^2}$$

Select an Image

α − вероятности ошибки первого рода, она же уровень значимости. Обычно фиксируется от 0.001 до 0.05.

β − вероятности ошибки второго рода, вероятность, когда мы скажем, что эффекта нет, когда на самом деле он есть.

Часто его берут равным 0.1 или 0.2.

σ_x и σ_y − стандартные отклонения в контрольной и экспериментальной группах соответственно. Эти значения можно оценить, подсчитав стандартные отклонения метрики по историческим данным.

ϵ − ожидаемый эффект — эффект, который мы ожидаем получить от эксперимента. Например, ожидаемый эффект может быть таким — повышение CTR на 0.01. Это означает, что если окажется верна альтернативная гипотеза, а истинный эффект будет 0.01 или более, то мы обнаружим его с вероятностью не менее 1 - β.

Φ^{-1} − это обозначение специальной функции из статистики (обратная функция нормального распределения).

Данная формула чаще всего используется в «калькуляторах» размера выборки. Несмотря на очевидные плюсы формулы стоит понимать, что она делает несколько серьёзных допущений. Например то, что мы точно знаем, что наши наблюдения распределены нормально. Эта проблема отчасти решается тем, что мы часто смотрим на статистику и на её распределение, но даже распределение статистики очень часто нам неизвестно.

Также формула предполагает использование t-test'a, что создает дополнительные ограничения.

Тем не менее, данная формула вовсе не лишена смысла, из неё можно выделить несколько полезных свойств:

- чем больше дисперсия в группах -> тем больше наблюдений нам нужно;
- чем меньший эффект мы хотим обнаружить -> тем больше наблюдений нам нужно;
- чем большие вероятности ошибок мы выбираем -> тем меньше наблюдений нам нужно.

> Оценка методом Монте-Карло

На помощь может прийти оценка необходимого размера выборки методом Монте-Карло, данный способ позволяет нам опустить большинство требований.

Выполнить такую оценку можно с помощью предварительного A/A тестирования или на исторических данных:

- Фиксируем количество тестов, которые будем проводить, например, `N = 1000`.
- Выбираем начальный размер выборок, например, `n = 500`
- Семплируем n наблюдений из группы A (которую мы модифицировали) и группы B.
- Проводим между этими группами тестирование и хотим заметить отличия в проценте групп, соответствующем нашей мощности.
- Если получаем меньший процент, то необходимо увеличить количество наблюдений, если получаем больший, то можно попробовать уменьшить количество наблюдений

В нашем примере мы рассмотрели следующий дизайн эксперимента:

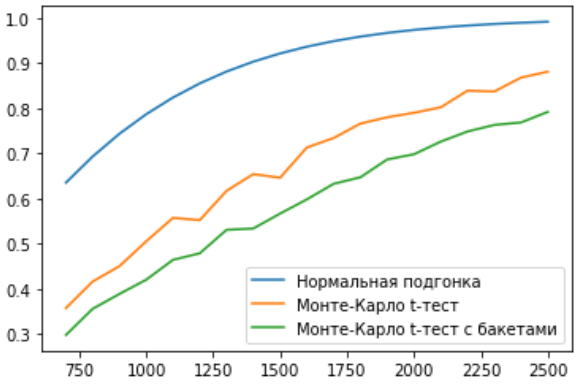
- засемплировали просмотры;
- засемплировали CTR и использовали это как оценку вероятности клика;
- получили CTR как семпл из биномиального распределения.

> Бакетное преобразование и линеаризация

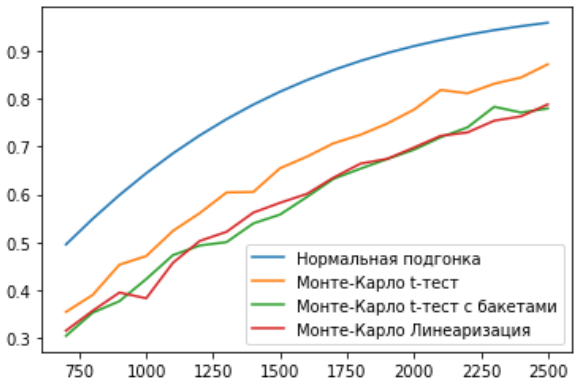
Из предыдущего урока нам может пригодиться бакетное преобразование:

```
def bucketization(ctrs_0, weights_0, ctrs_1, weights_1, n_buckets=100):
    """
    Разбиваем на бакеты с весами
    :param ctrs_0: np.array shape (n_experiments, n_users), CTRs of every user from control group in every experiment
    :param weights_0: np.array (n_experiments, n_users), веса пользователей в контрольной группе
    :param ctrs_1: np.array (n_experiments, n_users), CTRs of every user from treatment group in every experiment
    :param weights_1: np.array (n_experiments, n_users), веса пользователей в целевой группе
    :param n_buckets: int, кол-во бакетов
    :return: np.array shape (n_experiments), средневзвешенные метрики в каждом бакете
    """

    n_experiments, n_users = ctrs_0.shape

    values_0 = np.zeros((n_experiments, n_buckets))
    values_1 = np.zeros((n_experiments, n_buckets))

    for b in np.arange(n_buckets):
```

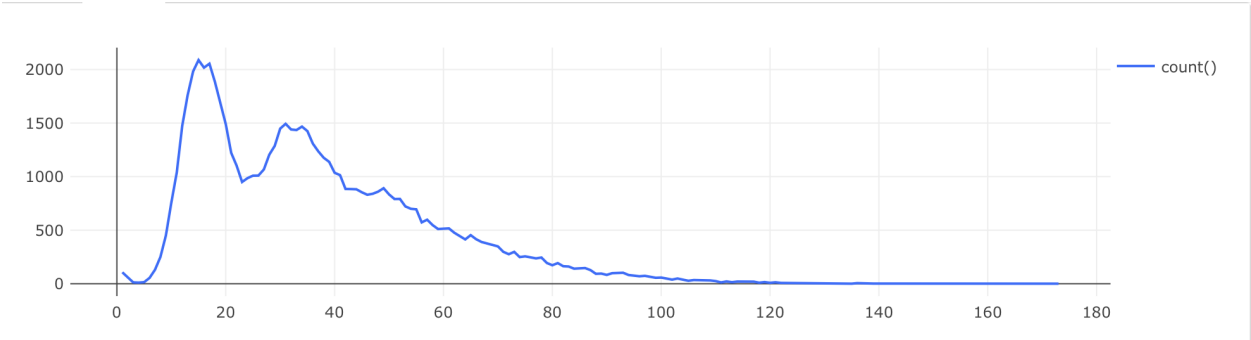
```
        ind = np.arange(b * n_users / n_buckets, b * n_users / n_buckets + n_users / n_buckets).astype(np.int)
        values_0[:, b] = np.sum(ctrs_0[:, ind] * weights_0[:, ind], axis=1) / np.sum(weights_0[:, ind], axis=1)
        values_1[:, b] = np.sum(ctrs_1[:, ind] * weights_1[:, ind], axis=1) / np.sum(weights_1[:, ind], axis=1)

    return values_0, values_1
```



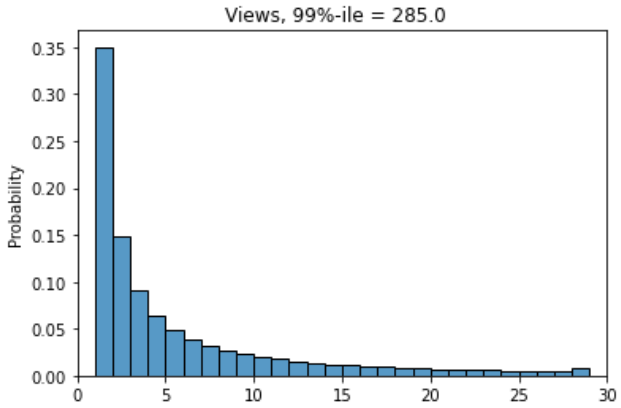
Линеаризация тоже, на первый взгляд, не приносит никакого результата:



Подход с бакенами не очень подходит конкретно под наш набор данных, график показов выглядит вот так:



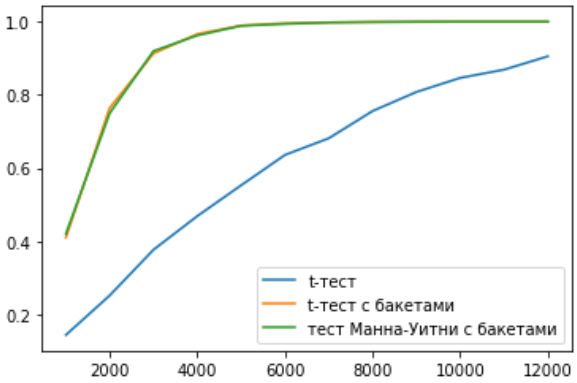
но гистограмма просмотров и конверсий также довольно часто выглядит следующим образом и имеет сильное смещение в сторону нуля:



И в таком случае t-test после бакетного преобразования уже будет гораздо более чувствительным. При предрассчитанном sample size мы увидели статзначимые различия во всех экспериментах при бакетном преобразовании и увидели различия в 88% при обычном способе проведения эксперимента.

> Результаты расчета на более экстремальных условиях

Мы также можем промоделировать то, что наш эффект повлиял на пользователей не одинаково, а только **в среднем**, в этом поможет бета-распределение, а также уменьшение конверсии. Проведем синтетические тесты:



На графике можно заметить, что простой t-test требует гораздо больше наблюдений только для того, чтобы сойтись на уровень мощности 80%, чем критерии с бакетизацией, которые показывают хорошую мощность довольно быстро.