Document —> 1 —> Microsoft

Basic
Controlled group - same thing doing as before
Treatment Group -> group with new feature

Experiment on a part vs other still same
Lime example 1700 british captain

Yes/No widget change 5 star ratings ->
problem since article helps or not

Overall Evaluation Criterion (OEC) =
Outcome

1. Factor -> a/b -> 1 rezultat
2. Variant -> difference between control and treatment group

If bug/problem -> all users considered control group

3. Experminetation Unit -> On who you experiment -> Users*COOKIES*
4. Null Hypothesis -> H0 -> Random fluctuations between control and treatment groups that got the outcome x

5. Confidence Level -> failing to reject H0
6. Power -> measure ability to acknowledge a difference that exists
7. A/A Testing -> Same thing for everyone -> Get data from users
8. Standard Deviation -> Measure of Variability
9. Standard Error

Outcome setting

1. Confidence Level -> means that 5% of time there is no difference between groups(Type 1 error)
2. Power -> 80-95% there is a difference
3. Standard Error -> increase sample size will reduce standard error
4. Have an expectation rather than checking results

Ways of doing the experiment

1. Treatment Ramp-Up -> increase treatment patients while reducing control patients
2. Automation -> Use algorithms and automation for quick response to the

user personalisation(Amazon recommendations for example)
3. Software Migration -> When migrate software use the expectation of null hyphotesis that nothing is changed between group A and B

Limitations

1. Metrics can be good but why and how do we see them or conceive them(different perspective viewing)
2. Long term testing -> test for example ads on people and stop showing certain ads to some people if they do not click on them if they see them many times
3. Newness Effects -> when new feature is released -> people click on multiple things to know how to use the feature so do not get that much data on short term but on long term rather when they are already used to the feature(weeks)
4. New release needs improvements(quick/bug setup) -> test on small percentage of people
5. Consistency -> Multiple versions based

on computer and the cookies
6. If a feature is in the news -> 100% users needs to get it -> ex. facebook like with flower things -> some get it some not but it is in the news -> people who not get it are sad and want to use it

Implementation Architecture

1. Randomisation Algorithm -> Users will get a variant and stick to it -> not correlation between experiments -> Caching the user type of experiment for the next time he/she goes to that page

OR -> Each user gets an id linked to their account so that we recognise their type of experiment based on ID

Analysis

1. Mine the Data -> machine learning check -> might find walls in our testing road -> x % of people have browser problem with a code -> solve code -> continue test the essential part
2. Speed Matters -> tests make website slow since NO optimisation -> AMAZON

1% sell drop / 100 msec more wait -
>GOOGLE 20% revenue drop / 500
msec more wait time
3. Test One Factor At A Time -> Take care
of the variables you take into
consideration

Execution

1. A/A Testing to be sure 95% users same
2. Automate Ramp-Up + Abort -> Abort if
the new featured website is having worse
results than the current one
3. Determine Minimum Sample Size ->
Check and estimate what you want to
get from the test
4. Assign 50% Users to Treatment -> do
50/50 or 99/1(25x longer test period)
5. Beware Days of the Week Effect -> some
users access certain websites in the
weekend more -> dayOfTheWeek
variable to be added

Culture

1. Agree on Outcome Upfront -> agree on
how a experiment is going to be

evaluated before it is done
2. Do not release without testing (“no hurt will be done mentality")
3. Calculate the cost of a new variant in the feature ROI -> do not do 1 mil variant if none of them can get revenue to cover them
4. Running frequent experiments rather than once in a while

By running experiments (in the past there was no possibility) we can learn about the users more so that we can improve the platform more

---

Document —> 4 —> Amazon - Own Choice

A/B Testing Use to see the Choices of clicks a user do!

Too many Choices -> Need to reduce decision space

Use of Bayesian Model to determine path through website -> probability taken by testing websites and checking the patterns users to through

Bandit Algorithm -> The most unsuccessful path(outfit in the example) used not to have combinations of that again -> a horse mask for a date is not good for any combination of underwear + socks + shirt + pants

Hill Climbing Algorithms -> we are dressed in a certain way and now check each component at a time -> keep everything but keep changing shirts and check how that fits with the other clothes that you currently wear(out of 5 webpage elements keep 4 and then switch the options for one of them and see how that fits with the other 4)

Model Interaction -> fast generalisation -> match clothes and try to see if they are good(make content that would match

together*widgets in a checkout page*) but the problem is that what if I dress to go to football and you dress for restaurant?

Very important to run the Bayesian on a long time period so that the probabilities will be more accurate