U.S. FOOD & DRUG ADMINISTRATION

adaptIVPT: Adaptive Designs for IVPT Data with Mixed Scaled Average Bioequivalence

Daeyoung Lim
DBVIII, Office of Biostatistics
Office of Translational Sciences
Center for Drug Evaluation and Research, FDA
Coauthors: Elena Rantou, Jessica Kim, Nam Hee Choi, Sungwoo Choi, Stella Grosser

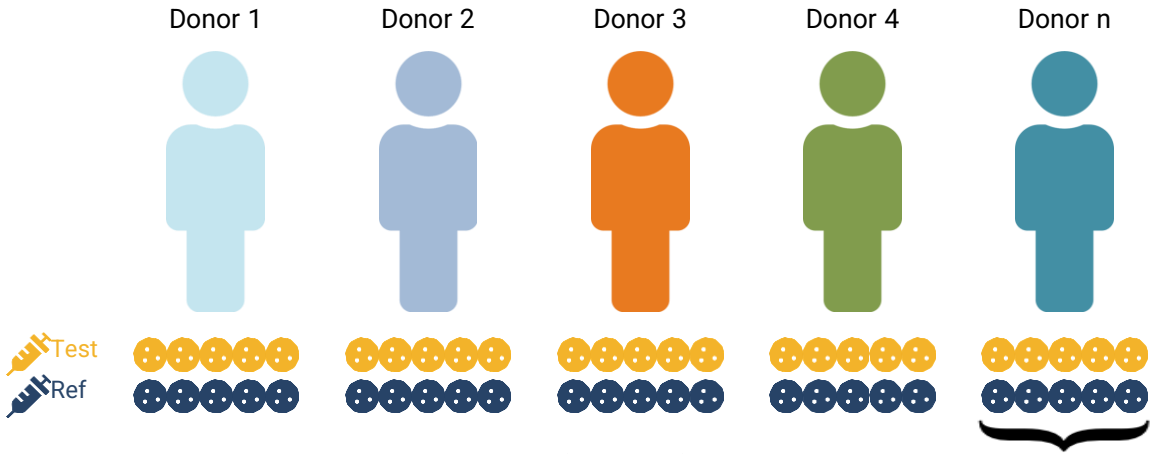
Disclaimer: *The information in this presentation represents the opinions of the speaker and does not necessarily represent FDA's position or policy.*

adaptIVPT

- is an R package that will help you design and analyze in vitro permeation test (IVPT) data
- contains 4 functions:
  - `msabe` for hypothesis testing (mixed-scaled average bioequivalence)
  - `prms` for calculating passing rates (passing rate mixed scaled)
  - `rss` for reestimating sample size (hence adaptive)
  - `PRsurface` for plotting passing-rate surfaces (passing-rate surface)

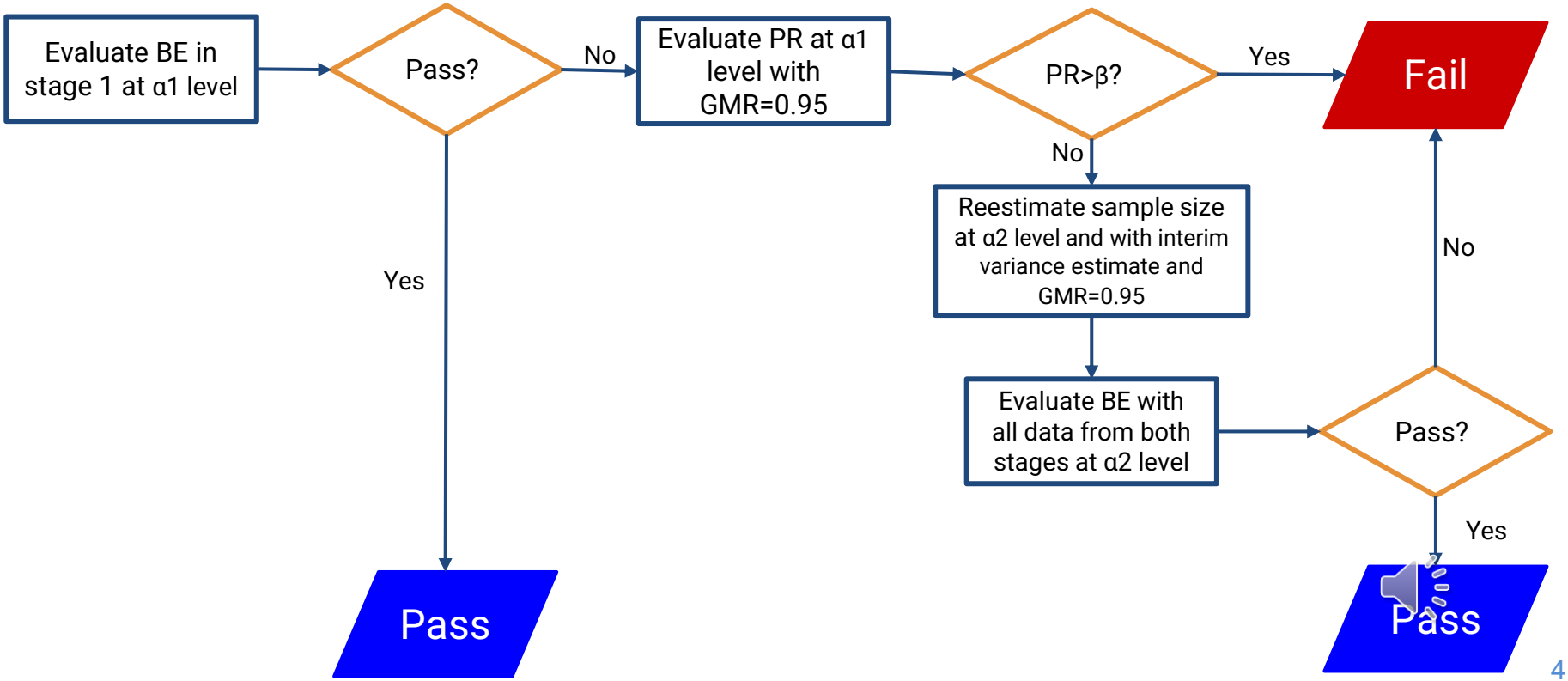
Background – Design

- Replicate design for in vitro permeation tests (Balanced Data)



- In vitro permeation test (IVPT):
  - uses human excised skin
  - measures drug concentration

Background – Adaptive Design



To do this, use this

- To evaluate BE
  - `adaptIVPT` provides `msabe`
- To compute passing rate
  - use `prms`
- To reestimate the sample size
  - use `rss`
- To visualize your product's passing rate, relative to theoretical PRs
  - use `PRsurface`

msabe

Input

- `Test` – an *n*-by-*r* matrix of test product data
- `Reference` – an n-by-r matrix of reference product data
- `params` – (Optional) a list of FDA regulatory constants and the significance level
  - `sigma_W0`: FDA regulatory constant (read more here)
  - `m`: FDA regulatory constant
  - `sig_level`: significance level

Output

- `parameters` – a list of parameter values used for testing
- `fout` – the test result and related estimators
- `runtime` – total elapsed time

Using msabe

Input

```
library(adaptIVPT)
n <- 6
r <- 3
# assume bio-inequivalent
# (mu_WT = 3, mu_WR=1 => GMR = 3 >> 1.25)
Test <- matrix(rnorm(n*r, 3, 0.5), n, r)
# assume highly variable reference drug
Reference <- matrix(rnorm(n*r, 1, 0.9),
n, r)
out_test <- msabe(Test, Reference)

# print test outcome and estimators
print(out_test$fout)
```

Output

```
> print(out_test$fout)
     test      Ibar
0.00000000 2.26890032
     S2_WR     S2_I
0.39122724 0.05707996
lowerbound upperbound
0.00000000 5.77618622
```

test is 0, meaning failed BE test

Ibar is the estimate of log(GMR)

S2_I is the SE of Ibar

rss

Input

- `n` – The number of donors
- `r` – The number of replicates from each donor per treatment
- `params` – (Optional) a list of additional tuning parameters
  - `sigma_W0`, `m`: FDA regulatory constants (read more here)
  - `GMR`: assumed geometric mean ratio
  - `sig_level`: significance level
  - `nmax`: upper limit for reestimated sample size
  - `target_power`: PR to aim (0.8 by default)
- `nsim` – Monte Carlo sample size for estimation
- `ncores` – CPU cores for parallel computing

Output

- `parameters` – a list of parameter values used
- `rss` – reestimated sample size
- `runtime` – total elapsed time

Using rss

Input

```
library(adaptIVPT)
out <- rss(10, 3, S_WR=0.22, nsim=10000)

# print the reestimated sample size
print(out$rss)

# now let's assume highly variable,
S_WR=0.60 and aim for higher power
out_hv <- rss(10, 3, S_WR=0.60,
nsim=10000, params=list(target_power =
0.9))

print(out_hv$rss)
```

Output

```
> print(out$rss)
[1] 9
```

This means even if we start out with n=10, it suffices to have n=9, r=3 to attain PR=80%. (In adaptive design, it means there's no need to go to stage 2)

```
> print(out_hv$rss)
[1] 17
```

n=17 is required if the reference drug is highly variable and target power is higher, using the adaptive design (= 7 more donors in stage 2)

prms

Input

- `n` – The number of donors
- `r` – The number of replicates from each donor per treatment
- `params` – (Optional) a list of additional tuning parameters
  - `sigma_W0`, `m`: FDA regulatory constants (read more here)
  - `GMR`: assumed geometric mean ratio μ_T/μ_R
  - `sig_level`: significance level
  - `sigma_WR`: true sd of the reference population $(N(\mu_R, \sigma^2_{WR}))$
  - `sigma_WT`: true sd of the test population $(N(\mu_T, \sigma^2_{WT}))$
- `nsim` – Monte Carlo sample size for estimation
- `ncores` – CPU cores for parallel computing

Output

- `parameters` – a list of parameter values used
- `passing_rate` – estimated passing rate
- `runtime` – total elapsed time

Using prms

Input

```
library(adaptIVPT)
out <- prms(6, 3, nsim = 10000,
            params = list(sigma_WR = 1,
                          GMR = 0.81))

# print the passing rate
print(out$passing_rate)
```

Output

```
> print(out$passing_rate)
[1] 0.3825
### with highly variable reference drug
(sigma_WR=1) and GMR close to the ABE limits,
the passing rate is not very high
```

PRsurface

- `n` – The number of donors
- `r` – The number of replicates from each donor per treatment
- `observed_GMR` – observed (estimated) GMR
- `observed_sigmaWR` – observed (estimated) reference population standard deviation
- `GMR_grid` – vector of GMR values used for plotting; defaults to `seq(0.75, 1.3, length.out=100)`
- `sigmaWR_grid` – vector of sigmaWR values used for plotting; defaults to `seq(0.2, 1, length.out=100)`
- `nsim` – Monte Carlo sample size for estimation
- `ncores` – CPU cores for parallel computing
- `verbose` – TRUE/FALSE for progress bar
- `params` – see `prms`

- Pops up a 3D figure
- But also invisibly returns:
  - `GMR` – a vector of GMR grid values
  - `sigmaWR` – a vector of sigmaWR grid values
  - `z` – a matrix of passing rates (`length(GMR)` by `length(sigmaWR)`)

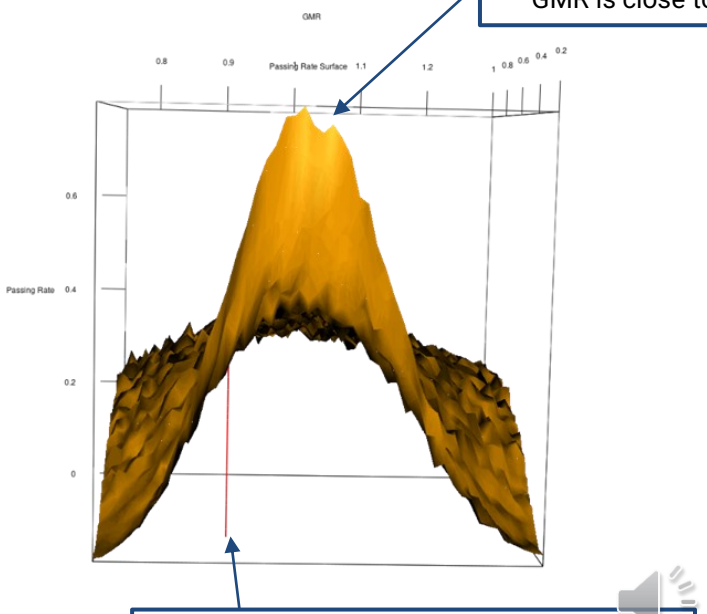
Using PRsurface

Input

```
library(adaptIVPT)
out <- PRsurface(6, 3, observed_GMR =
0.91, observed_sigmaWR = 0.4, nsim=1000,
verbose = TRUE)

# This will automatically generate a 3d
plot (takes some time!)
```

Output



Conclusion

- Test bioequivalence with `msabe`
- Calculate passing rates with `prms`
- Reestimate sample size (during interim analyses) with `rss`
- And visualize your data in terms of passing rates with `PRsurface`

For more information

- Visit github.com/daeyounglim/adaptIVPT for source code
- Check out FDA IVPT guidance
- Please email me at daeyoung.lim@uconn.edu if you have any questions :)
- `install.packages("adaptIVPT")`

U.S. FOOD & DRUG ADMINISTRATION

Thank you!

Disclaimer: *The information in this presentation represents the opinions of the speaker and does not necessarily represent FDA's position or policy.*


FDA
U.S. FOOD & DRUG
ADMINISTRATION