B-H for incentive calculation

Contents


```
library(ggplot2)
library(dplyr)
library(purrr)
library(furrr)
library(beepr)
library(tidyr)
library(forcats)
```

Purpose of this document: Using Benjamini-Hochberg to justify the FP penalty being 19. The simulation will give us the expected total payout under B-H and FP penalty = 19, so we can create a grid of α's and $P(null)$'s and see which $alpha$ gives the highest payout.

Executive summary: under B-H, $\alpha = 0.01$ (the lowest tested on the simulation) gives the highest expected total payout. Without any correction strategy, $\alpha = 0.17 \sim 0.2$ maximizes payout.

Setup params

```
alpha <- 0.05     # not using in simulation
K <- 20           # number of data points in a hypothesis/region

# ACHTUNG: mu/sd is c(1.2/3, 1.5/2.5, 1.6/2) in stimuli, picking the average here
mu <- 1.5         # sample mean in a hypothesis/region
sigma <- 2.5

p_null <- 0.5     # the bane of my existance
n_iter <- 5000    # number of iterations in simulation
```

p-value PDF, CDF, invCDF defs

- Definitions of PDF and CDF of the p-value from [@hung_behavior_1997]
- Random draws from the p-value PDF `rp` is sampled through random draws from the uniform [0, 1] quantile space, then looked up through the inverse CDF function (using `uniroot`).
- **Limitation**: These functions are dependent on the μ, σ, K parameters. These parameters are most likely different IRL but we are not using other values yet.

```
# PDF
f_p <- function(x, mu, sigma, K) {
  dnorm(qnorm(1 - x) - sqrt(K) * mu / sigma) / dnorm(qnorm(1 - x))
}

# CDF
F_p <- function(x, mu, sigma, K) {
  1 - pnorm(qnorm(1 - x) - sqrt(K) * mu / sigma)
}
```

```r
# inverse CDF of p-value
F_p_inv <- function(q, mu, sigma, K, l = 0, u = 1){
  uniroot(function(p) F_p(p, mu, sigma, K) - q, lower = l, upper = u)$root
}

# random sample from PDF of p-value using its invCDF
rp <- function( mu, sigma){
  q <- runif(1)
  F_p_inv(q, mu, sigma, K)
}
```

Expected value of number of selections

We can get the expected number of selections under B-H with a few simulation iterations and take the average. Given that all the parameters, such as α, are fixed, the uncertainty comes from the sampling of true μ's from a Binomial and the random sampling of p-value.

- The simulation has `n_iter` of "trials" in our experiment
- For each trial, draw 8 or 12 true μ (0 or μ) from $Bin(n, 1 - P(null))$
- With the μ and pre-specified σ et al., draw 8 or 12 p-values
- Do the B-H and reject hypotheses/regions accordingly
- We get the number of rejections that *should* happen under B-H, for both 8 and 12 regions.

```r
(df <-
  expand.grid(
  nregions = c(8, 12),
  iter = 1:n_iter
) %>%
  uncount(nregions, .remove = FALSE, .id = "panel") %>%
  group_by(iter, nregions) %>%
  mutate(
    mu = mu * rbinom(nregions, 1, 1 - p_null),
    p_raw = map_dbl(mu, ~rp(.x, sigma)),
    p_bh = p.adjust(p_raw, method = "BH"),
  )  %>%
  pivot_longer(starts_with("p_"), names_to = "method", values_to = "p")  %>%
  mutate(
    true = mu == 0, # null hypothesis being true
    reject = p < alpha
  ) %>%
  group_by(iter, nregions, method) %>%
  summarize( tp = sum(!true & reject),
             fp = sum(true & reject),
             tn = sum(true & !reject),
             fn = sum(!true & !reject),
             .groups = "drop_last") %>%
  mutate(fdr =  ifelse(tp * fp != 0, (fp) / (tp + fp), 0),
         pay = (tp - 19 * fp + tn - fn) * 1)
)
```

```
## # A tibble: 20,000 x 9
## # Groups:   iter, nregions [10,000]
##      iter nregions method    tp    fp    tn    fn   fdr   pay
```

```
##    <int>    <dbl> <chr>  <int> <int> <int> <int> <dbl> <dbl>
##  1     1        8 p_bh       5     0     3     0     0     8
##  2     1        8 p_raw      5     0     3     0     0     8
##  3     1       12 p_bh       8     0     2     2     0     8
##  4     1       12 p_raw      8     0     2     2     0     8
##  5     2        8 p_bh       0     0     5     3     0     2
##  6     2        8 p_raw      2     0     5     1     0     6
##  7     2       12 p_bh       8     0     4     0     0    12
##  8     2       12 p_raw      8     0     4     0     0    12
##  9     3        8 p_bh       1     0     5     2     0     4
## 10     3        8 p_raw      2     0     5     1     0     6
## # ... with 19,990 more rows
```

```
df %>%
  group_by(method) %>%
  summarize(mean(fdr))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 2 x 2
##   method `mean(fdr)`
##   <chr>        <dbl>
## 1 p_bh        0.0239
## 2 p_raw       0.0532
```

Once we have the distribution of # of rejections B-H says we should make, we can take the average and get the expected value for number of selections.

```
(expected_nselect <-
   df %>%
   group_by(method, nregions) %>%
   summarise(E_nselect = mean(tp + fp), .groups = "drop")
)
```

```
## # A tibble: 4 x 3
##   method nregions E_nselect
##   <chr>     <dbl>     <dbl>
## 1 p_bh          8      3.09
## 2 p_bh         12      4.54
## 3 p_raw         8      3.63
## 4 p_raw        12      5.40
```

Expected total payout

There have been three iterations of expected total payout

1. $E[n_{FP}] = E[n_{reject}] * P(null)$. This is (even more) problematic now that we're using B-H. It can overestimate n_{FP} because if people are ordering hypotheses/regions by p-values, they should be making false discoveries at a lower rate than $P(null)$.
2. $E[n_{FP}] = E[n_{reject}] * P(true|reject) = E[n_{reject}] * \alpha$. By definition, $P(true|reject) = E\left[\frac{n_{true \wedge reject}}{n_{reject}}\right] = FDR \leq \alpha$, and we say B-H controls FDR at level α. But α here is an upperbound instead of the expected value, see [@benjamini_controlling_1995].

- The problem is, we don't have a good expression for $E[n_{TP}]$. If we write $E[n_{TP}] = E[n_{reject}] * P(\neg true|reject)$, we don't have an expression for $P(\neg true|reject)$ like $FDR \leq \alpha$. If we just use the joint probability $E[n_{TP}] = nP(\neg true, reject)$, $P(\neg true, reject) = 1 - \beta$, but there is no closed-form for power for B-H; [@benjamini_controlling_1995] had to run a simulation. So...

3. Screw it, just use the simulation results. Basically the original simulation.

```
# (expected_payout <- expected_nselect %>%
#   mutate(E_tp = E_nselect * (1 - alpha),
#          E_tn = (nregions - E_nselect) * alpha,
#          E_fp = E_nselect * p_null,
#          E_fn = (nregions - E_nselect) * (1 - p_null),
#          E_payout = E_tp - 19 * E_fp + E_tn - E_fn)
# )

# (expected_payout <- expected_nselect %>%
#   mutate(E_tp = E_nselect * (1 - p_null),
#          E_tn = (nregions - E_nselect) * p_null,
#          E_fp = E_nselect * p_null,
#          E_fn = (nregions - E_nselect) * (1 - p_null),
#          E_payout = E_tp - 19 * E_fp + E_tn - E_fn)
# )
df %>%
  group_by(method, nregions) %>%
  summarize(mean(pay))
```

```
## `summarise()` regrouping output by 'method' (override with `.groups` argument)
```

```
## # A tibble: 4 x 3
## # Groups:   method [2]
##   method nregions `mean(pay)`
##   <chr>     <dbl>       <dbl>
## 1 p_bh          8        4.02
## 2 p_bh         12        6.12
## 3 p_raw         8        2.72
## 4 p_raw        12        4.19
```

Grid search for the good alpha

Putting the above pieces together, we vary *alpha* and $P(null)$ but keep μ, σ and K the same.

Encapsulate simulation function

This is the same code as above

```
finding_payout <- function(alpha, p_null){
  df <-
    expand.grid(
      nregions = c(8, 12),
      iter = 1:n_iter
    ) %>%
    uncount(nregions, .remove = FALSE, .id = "panel") %>%
    group_by(iter, nregions) %>%
    mutate(
      mu = mu * rbinom(nregions, 1, 1 - p_null),
      p_raw = map_dbl(mu, ~rp(.x, sigma)),
      p_bh = p.adjust(p_raw, method = "BH"),
    )  %>%
    pivot_longer(starts_with("p_"), names_to = "method", values_to = "p")  %>%
    mutate(
      true = mu == 0, # null hypothesis being true
```

```
      reject = p < alpha
    ) %>%
    group_by(iter, nregions, method) %>%
    summarize( tp = sum(!true & reject),
               fp = sum(true & reject),
               tn = sum(true & !reject),
               fn = sum(!true & !reject),
               .groups = "drop_last") %>%
    mutate(fdr =  ifelse(tp * fp != 0, (fp) / (tp + fp), 0),
           pay = (tp - 19 * fp + tn - fn) * 1,
           power = (tp)/(tp + fn)) # this produces NA's; set to 1?

df %>%
  group_by(method, nregions) %>%
  summarize(E_pay = mean(pay),
            E_fdr = mean(fdr),
            power = mean(power), .groups = "drop")
}
```

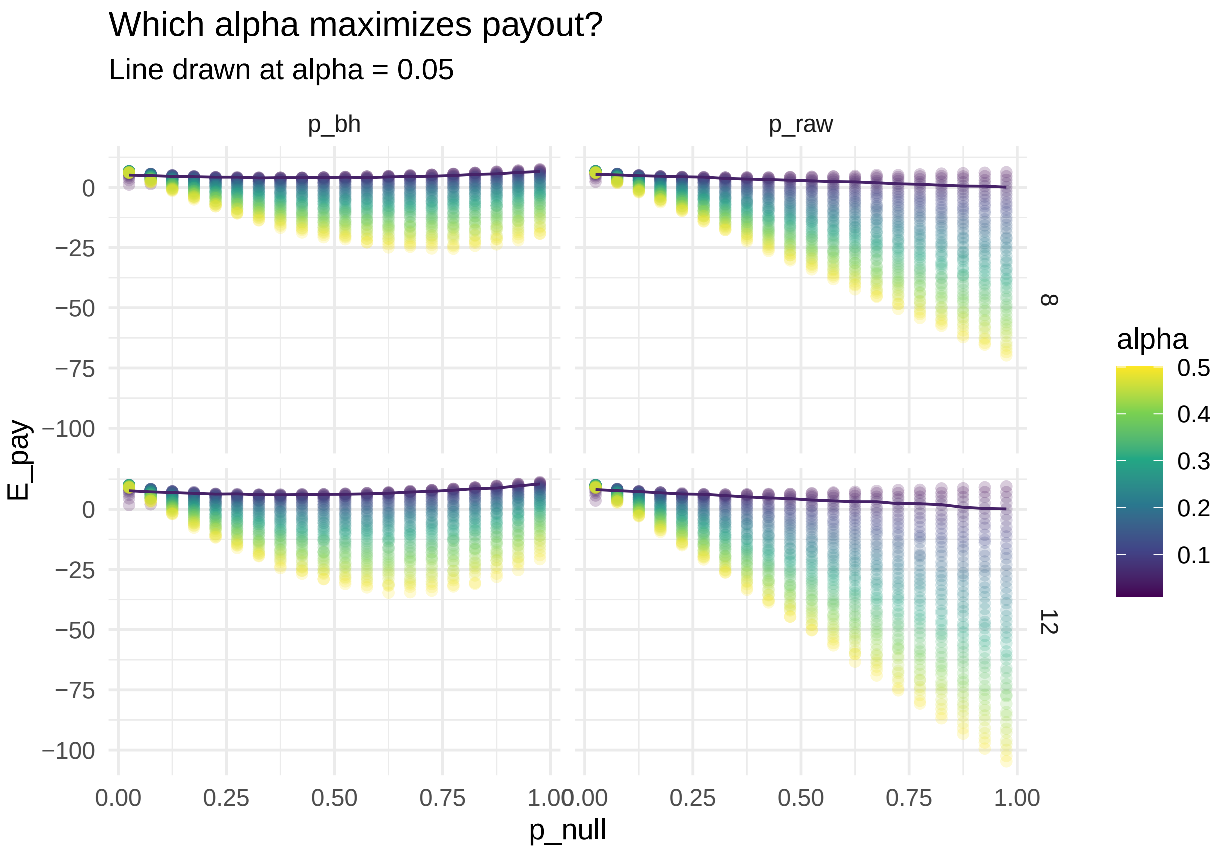
setting seed https://davisvaughan.github.io/furrr/articles/articles/gotchas.html#argument-evaluation

```
plan(multisession, workers = 5)
options <- furrr_options(seed = 123)


(sim_df <- expand.grid(
    alpha = seq(from = 0.01, to = 0.5, by = 0.01),
    p_null = ppoints(20)
  ) %>%
    split(1:nrow(.)) %>%
    future_map_dfr(~cbind(.x, finding_payout(.$alpha, .$p_null), row.names = NULL), .options = options)
)

beep()
saveRDS(sim_df, "sim_n5000_alpha_pnull.rds")
```

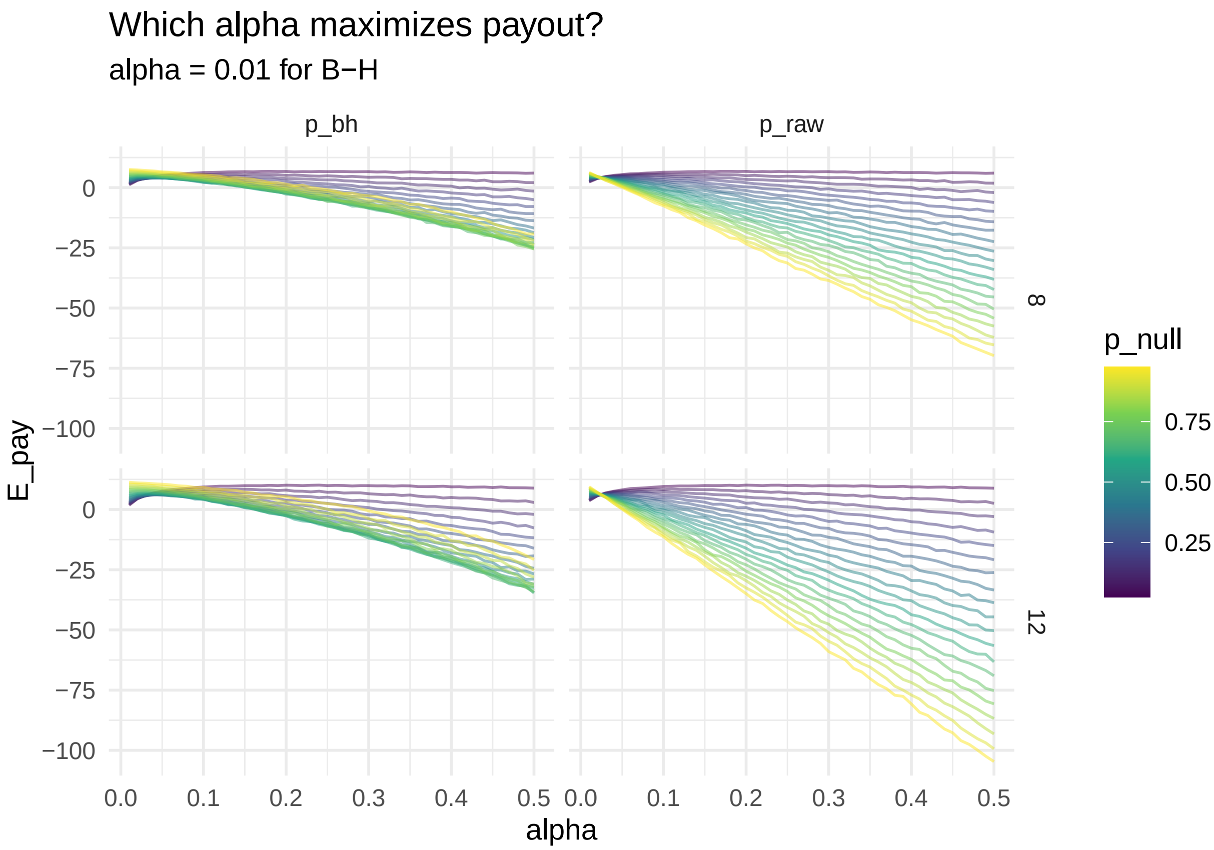
```
sim_df <- readRDS("sim_n5000_alpha_pnull.rds")
```

Results

```
sim_df %>%
  ggplot(aes(p_null, E_pay, color = alpha)) +
  geom_point(alpha = 0.2) +
  geom_line(data = sim_df %>% filter(alpha == 0.05)) +
  facet_grid(nregions ~ method) +
  scale_color_viridis_c() +
  theme_minimal()  +
  labs(title = "Which alpha maximizes payout?", subtitle = "Line drawn at alpha = 0.05")
```



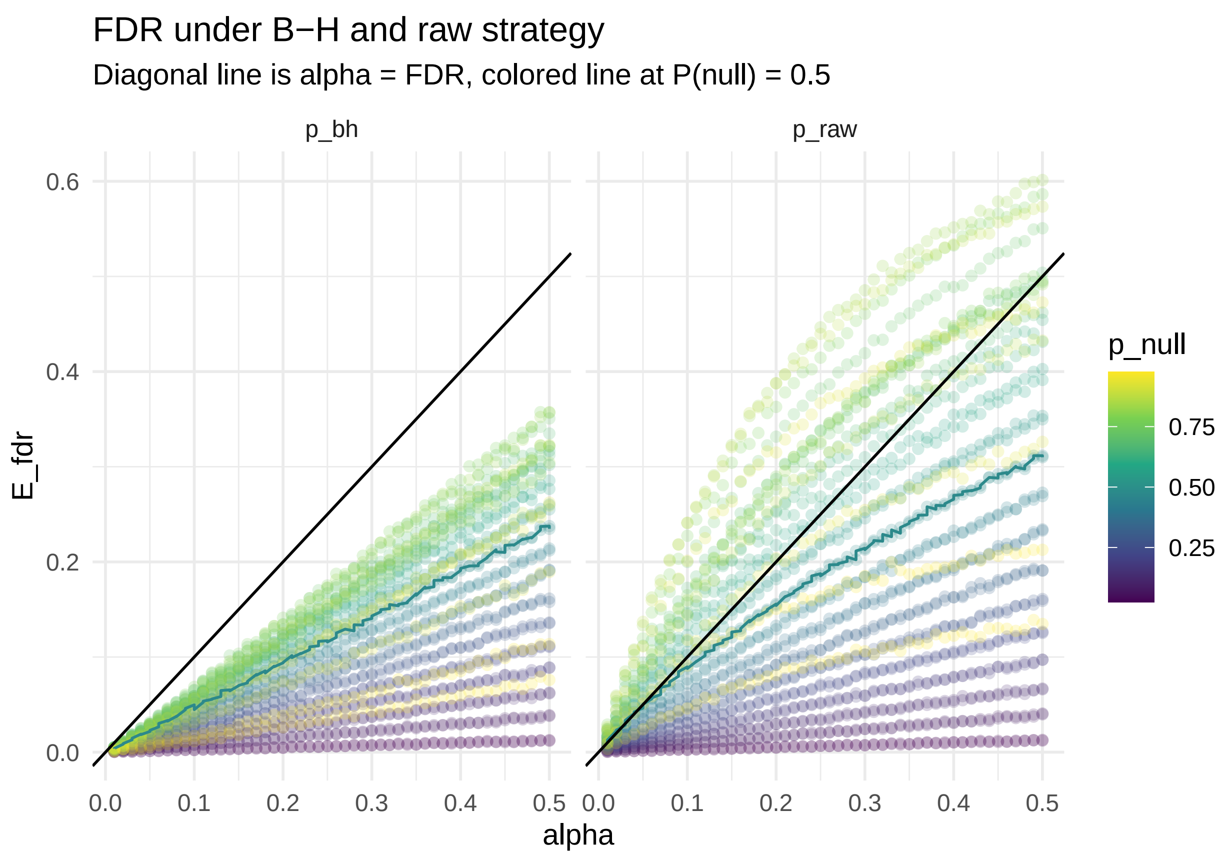

```
sim_df %>%
  ggplot(aes(alpha, E_pay, color = p_null)) +
  # geom_point(alpha = 0.2) +
  geom_line(aes(group = p_null), alpha = 0.5)  +
  # geom_line(data = sim_df %>% filter(p_null %in% c(0.025, 0.475, 0.975)), aes(group = p_null)) +
  facet_grid(nregions ~ method) +
  scale_color_viridis_c() +
  theme_minimal() +
  labs(title = "Which alpha maximizes payout?", subtitle = "alpha = 0.01 for B-H")
```




```
sim_df %>%
  group_by(nregions, method) %>%
  slice_max(E_pay)
```

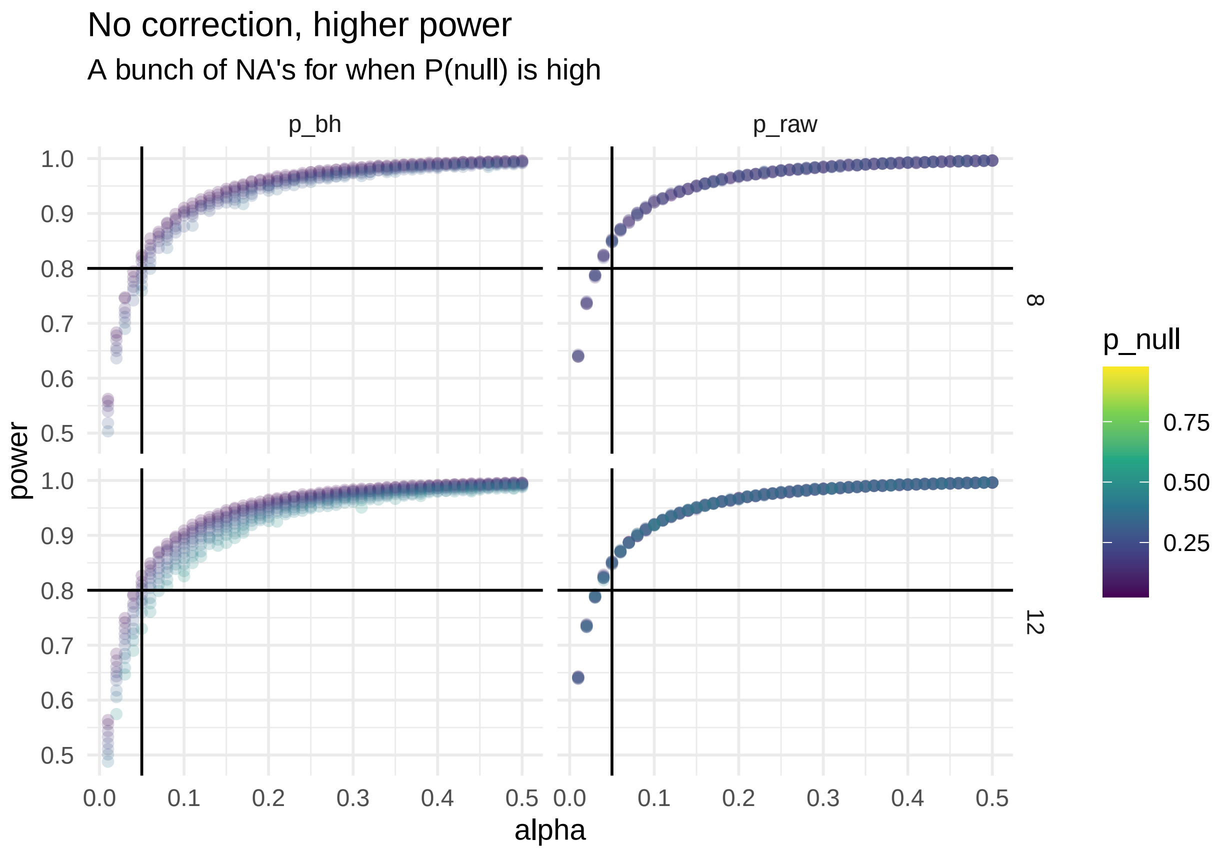
```
## # A tibble: 5 x 7
## # Groups:   nregions, method [4]
##   alpha p_null method nregions E_pay   E_fdr   power
##   <dbl>  <dbl> <chr>     <dbl> <dbl>   <dbl>   <dbl>
## 1  0.01  0.975 p_bh          8  7.54 0.0008   NaN
## 2  0.17  0.025 p_raw         8  6.75 0.00393   0.957
## 3  0.18  0.025 p_raw         8  6.75 0.00441   0.962
## 4  0.01  0.975 p_bh         12 11.3  0.00113 NaN
## 5  0.2   0.025 p_raw        12 10.1  0.00490   0.968
```

```
sim_df %>%
  ggplot(aes(alpha, E_fdr, color = p_null)) +
  geom_point(alpha = 0.2) +
  geom_line(data = sim_df %>% filter(p_null == 0.475)) +
  geom_abline(aes(intercept = 0, slope = 1)) +
  facet_grid(. ~ method) +
  scale_color_viridis_c() +
  theme_minimal() +
  labs(title = "FDR under B-H and raw strategy", subtitle = "Diagonal line is alpha = FDR, colored line
```




```
sim_df %>%
  ggplot(aes(alpha, power)) +
  geom_point(aes(color = p_null), alpha = 0.2) +
  geom_vline(aes(xintercept = 0.05)) +
  geom_hline(aes(yintercept = 0.8)) +
  facet_grid(nregions ~ method) +
  scale_color_viridis_c() +
  theme_minimal() +
  labs(title="No correction, higher power", subtitle = "A bunch of NA's for when P(null) is high")
```

```
## Warning: Removed 2430 rows containing missing values (geom_point).
```



No correction, higher power
A bunch of NA's for when P(null) is high
p_bh
p_raw
power
1.0
0.9
0.8
0.7
0.6
0.5
8
12
alpha
0.0
0.1
0.2
0.3
0.4
0.5
p_null
0.75
0.50
0.25