group project 1

Xinyi Lin

2/8/2019

Design a simulation under this setting to evaluate and compare the effectiveness of these four multiple comparison adjustments regarding (1) controlling family-wise type I error and (2) preserving the study power to detect true effects. Evaluate the impact of (1) the number of outcomes, (2) between-outcome correlations and (3) effect size on the performance of those methods

Evaluating the impact of effect size on the performance of thoese methods

```r
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.1.0     v purrr   0.2.5
## v tibble  1.4.2     v dplyr   0.7.8
## v tidyr   0.8.2     v stringr 1.3.1
## v readr   1.1.1     v forcats 0.3.0
```

```
## -- Conflicts ------------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

1. Generating data

```r
mean_x = 5
effect_size = 0
n_samplesize = 20
n_repeat = 100
method = "bonferroni"    #c("holm", "hochberg", "bonferroni")
l_resample = 200   # number of resample times for each outcomes
m = 5  # number of outcomes
```

```r
create_outcomes = function(mean_x, mean_y){
  #set.seed(123)
  treatment = list(mode = "vector", m)
  control = list(mode = "vector", m)
  p_value = vector(mode = "numeric", m)

  for(i in 1: m){
    treatment[[i]] = as.vector(rnorm(n_samplesize, mean_x, 5))
    control[[i]] = as.vector(rnorm(n_samplesize, mean_y, 5))
    p_value[i] = t.test(treatment[[i]], control[[i]])$p.value
  }

  repeat_unit = tibble(treatment, control, p_value)
  return(repeat_unit)
}

#repeat_unit = create_outcomes(mean_x, 5*effect_size + mean_x)
```

2. Calculating adjusted p-value

```
unadjusted_results = function(){
  if(min(repeat_unit$p_value) < 0.05) {return(1)}
  else {return(0)}
}
```

```
global_results = function(method){
  p_adjust = p.adjust(repeat_unit$p_value, method)
  effect_p = p_adjust[(p_adjust > 0.05)]

  return(length(effect_p)/n_samplesize)
}
```

```
resample_results = function(){
  #repeat_unit = create_outcomes(mean_x, 5*effect_size + mean_x)
  treatment_star = list(mode = "vector", m)
  control_star = list(mode = "vector", m)
  for (j in 1:m) {
    mean_treatment = mean(repeat_unit$treatment[[j]])
    mean_control = mean(repeat_unit$control[[j]])
    treatment_star[[j]] = repeat_unit$treatment[[j]] - mean_treatment
    control_star[[j]] = repeat_unit$control[[j]] - mean_control
  }
  p_min = vector(mode = "numeric", l_resample)
  for (i in 1: l_resample){
    p_resample = vector(mode = "numeric", m)
    # resampling each outcome
    for (j in 1:m) {
      sample_treatment = sample(treatment_star[[j]], n_samplesize, replace = TRUE)
      sample_control = sample(control_star[[j]], n_samplesize, replace = TRUE)
      p_resample[j] = t.test(sample_treatment, sample_control)$p.value
    }
    p_min[i] = min(p_resample)
  }
  p_orig_min = min(repeat_unit$p_value)
  p_num = 0
  for (i in 1:l_resample) {
    if (p_min[i] < p_orig_min) {p_num = p_num + 1}
  }
  p_final = p_num/l_resample
  if (p_final<0.05) {return(1)}
  else {return(0)}
}
```

3. Calculating power and type1error

For resampling process

```r
set.seed(123)
count = vector(mode = "numeric", n_repeat)
#p_final = vector("numeric", n_repeat)
for (i in 1:n_repeat){
  repeat_unit = create_outcomes(mean_x, 5*effect_size + mean_x)
  count[i] = resample_results()
}

power = mean(count)
```

For other process

```r
set.seed(123)
count = vector(mode = "numeric", n_repeat)
for (i in 1: n_repeat){
  repeat_unit = create_outcomes(mean_x, 5*effect_size + mean_x)
  #print(repeat_unit$p_value)
  count[i] = global_results(method)
}

power = mean(count)
```

evaluate influence of effec size

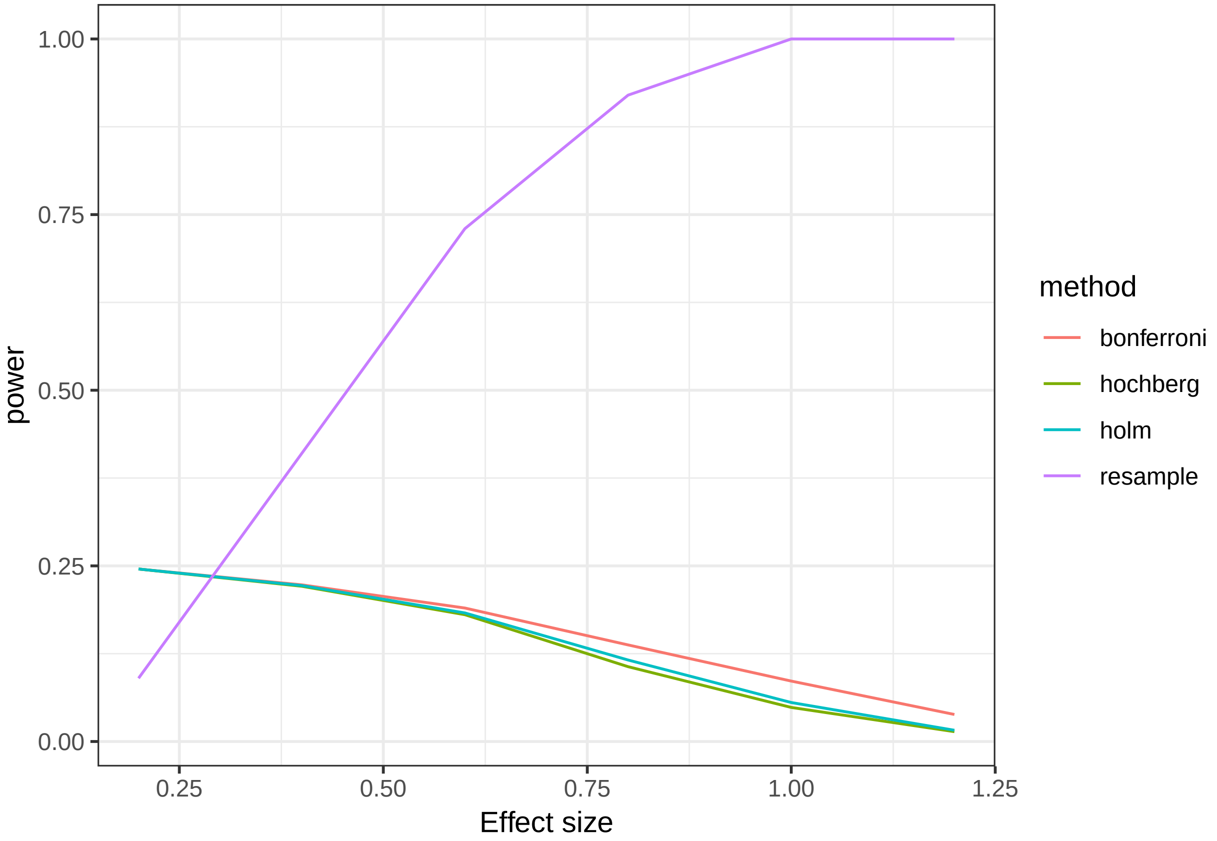
```r
set.seed(123)
effect_size_v = seq(0.2,1.2,0.2)
power_results = NULL
plot_restults = NULL
for (i in 1:length(effect_size_v)) {
  count_holm = 0
  count_hochberg = 0
  count_bonferroni = 0
  count_resample = 0
  power_holm = 0
  power_hochberg = 0
  power_bonferroni = 0
  power_resample = 0
  for (j in 1:n_repeat) {
    repeat_unit = create_outcomes(mean_x, 5*effect_size_v[i] + mean_x)
    count_resample = count_resample + resample_results()
    count_holm = count_holm + global_results("holm")
    count_hochberg = count_hochberg + global_results("hochberg")
    count_bonferroni = count_bonferroni + global_results("bonferroni")
  }
  power_resample = count_resample/n_repeat
  power_holm = count_holm/n_repeat
  power_hochberg = count_hochberg/n_repeat
  power_bonferroni = count_bonferroni/n_repeat
  power_results = cbind(power_holm, power_hochberg, power_bonferroni, power_resample)
  plot_restults = rbind(plot_restults, power_results)
}
plot_restults = cbind(effect_size_v, plot_restults)
#plot_restults_10 = plot_restults
#plot_restults_100 = plot_restults
#plot_restults_1000 = plot_restults
write.csv(plot_restults, file = "n20.cvs")
```

Draw the picture

```
library(ggplot2)

results_plot =
  as.tibble(plot_restults) %>%
  gather(key = "method", value = "power", power_holm:power_resample) %>%
  mutate(method = str_replace(method, "power_", "")) %>%
  ggplot(aes(x = effect_size_v, y = power, color = method)) +
  geom_line() +
  xlab("Effect size") +
  theme_bw()

results_plot
```



```
ggsave("n20_plot.png", results_plot, width = 8, height = 5)
```

change outcome sizes

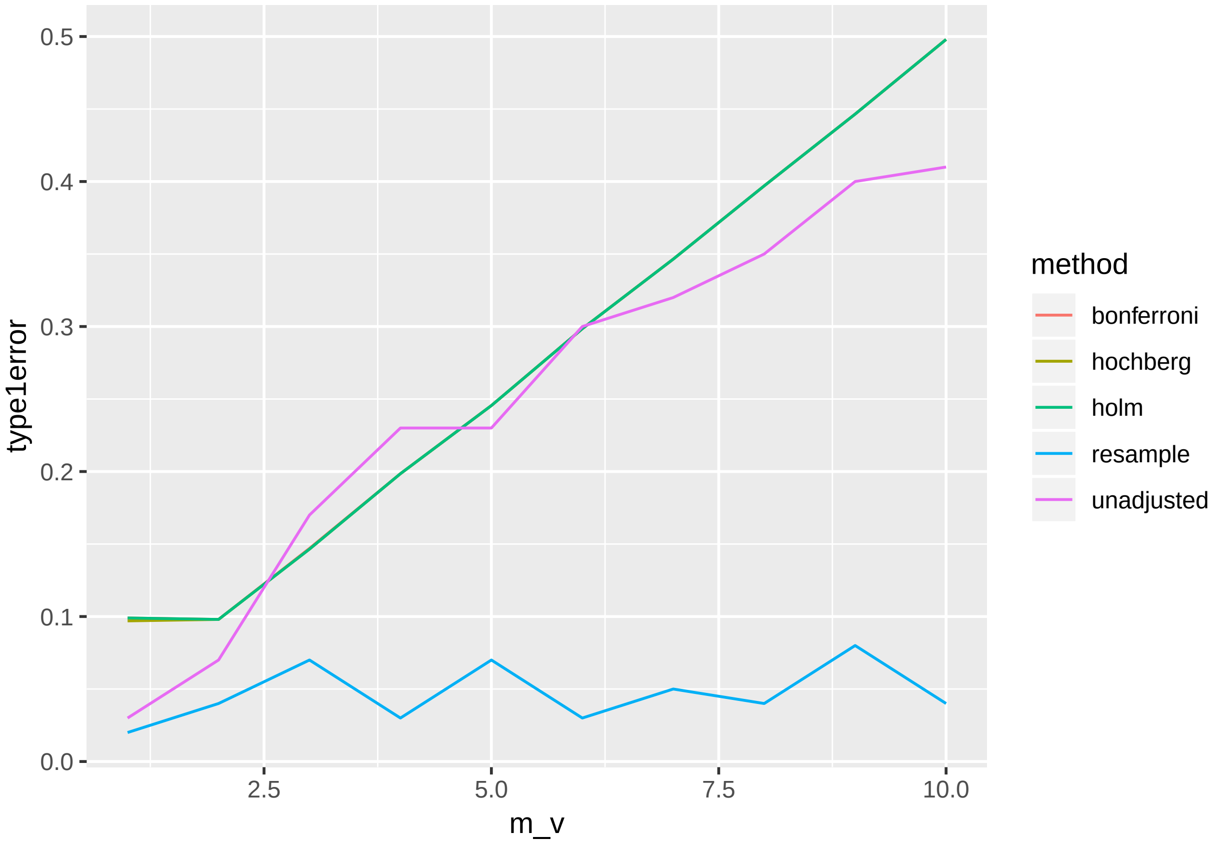
for type1error. . . . . .

```r
set.seed(123)
m_v = c(1:10)
power_results = NULL
plot_restults = NULL
for (i in 1:length(m_v)) {
  m = m_v[i]
  count_unadjusted = 0
  count_holm = 0
  count_hochberg = 0
  count_bonferroni = 0
  count_resample = 0
  power_holm = 0
  power_hochberg = 0
  power_bonferroni = 0
  power_resample = 0
  for (j in 1:n_repeat) {
    repeat_unit = create_outcomes(mean_x, 5*effect_size + mean_x)
    count_unadjusted = count_unadjusted + unadjusted_results()
    count_resample = count_resample + resample_results()
    count_holm = count_holm + global_results("holm")
    count_hochberg = count_hochberg + global_results("hochberg")
    count_bonferroni = count_bonferroni + global_results("bonferroni")
  }
  power_unadjusted = count_unadjusted/n_repeat
  power_resample = count_resample/n_repeat
  power_holm = count_holm/n_repeat
  power_hochberg = count_hochberg/n_repeat
  power_bonferroni = count_bonferroni/n_repeat
  power_results = cbind(power_unadjusted, power_holm, power_hochberg, power_bonferroni, power_resample)
  plot_restults = rbind(plot_restults, power_results)
}
plot_restults = cbind(m_v, plot_restults)
#plot_restults_10 = plot_restults
#plot_restults_100 = plot_restults
#plot_restults_1000 = plot_restults
```

```r
library(ggplot2)

results_plot =
  as.tibble(plot_restults) %>%
  gather(key = "method", value = "type1error", power_unadjusted:power_resample) %>%
  mutate(method = str_replace(method, "power_", "")) %>%
  ggplot(aes(x = m_v, y = type1error, color = method)) +
  geom_line()

results_plot
```



```
ggsave("type1error_plot.pdf", results_plot, width = 8, height = 5)
```

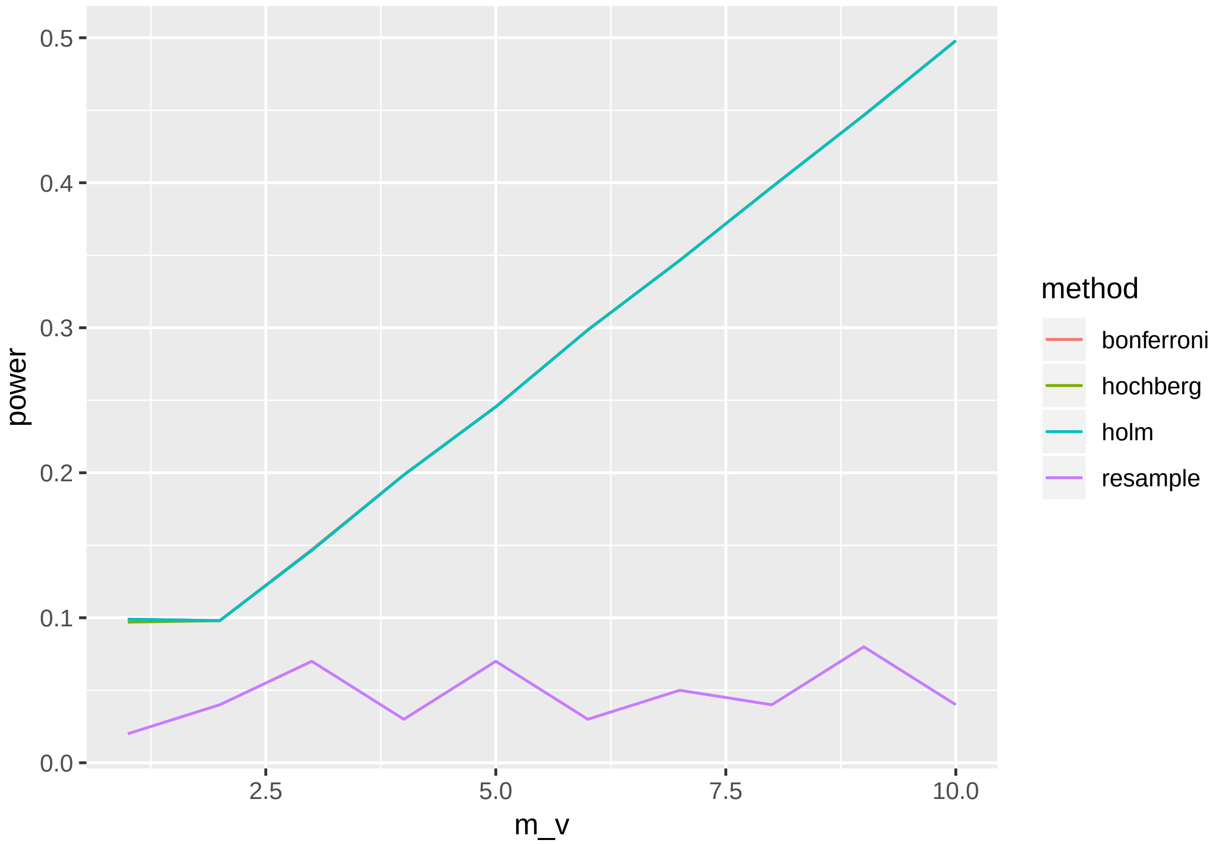
for power. . . . . .

```
set.seed(123)
m_v = c(1:10)
power_results = NULL
plot_restults = NULL
for (i in 1:length(m_v)) {
  m = m_v[i]
  #count_unadjusted = 0
  count_holm = 0
  count_hochberg = 0
  count_bonferroni = 0
  count_resample = 0
  power_holm = 0
  power_hochberg = 0
  power_bonferroni = 0
  power_resample = 0
  for (j in 1:n_repeat) {
    repeat_unit = create_outcomes(mean_x, 5*effect_size + mean_x)
    #count_unadjusted = count_unadjusted + unadjusted_results()
    count_resample = count_resample + resample_results()
    count_holm = count_holm + global_results("holm")
    count_hochberg = count_hochberg + global_results("hochberg")
    count_bonferroni = count_bonferroni + global_results("bonferroni")
  }
  #power_unadjusted = count_unadjusted/n_repeat
  power_resample = count_resample/n_repeat
  power_holm = count_holm/n_repeat
  power_hochberg = count_hochberg/n_repeat
  power_bonferroni = count_bonferroni/n_repeat
  power_results = cbind(power_unadjusted, power_holm, power_hochberg, power_bonferroni, power_resample)
  plot_restults = rbind(plot_restults, power_results)
}
plot_restults = cbind(m_v, plot_restults)
#plot_restults_10 = plot_restults
#plot_restults_100 = plot_restults
#plot_restults_1000 = plot_restults
```

```
library(ggplot2)

results_plot =
  as.tibble(plot_restults) %>%
  gather(key = "method", value = "power", power_holm:power_resample) %>%
  mutate(method = str_replace(method, "power_", "")) %>%
  ggplot(aes(x = m_v, y = power, color = method)) +
  geom_line()

results_plot
```



```
ggsave("power_plot.pdf", results_plot, width = 8, height = 5)
```