T-Test Vs Wilcoxon

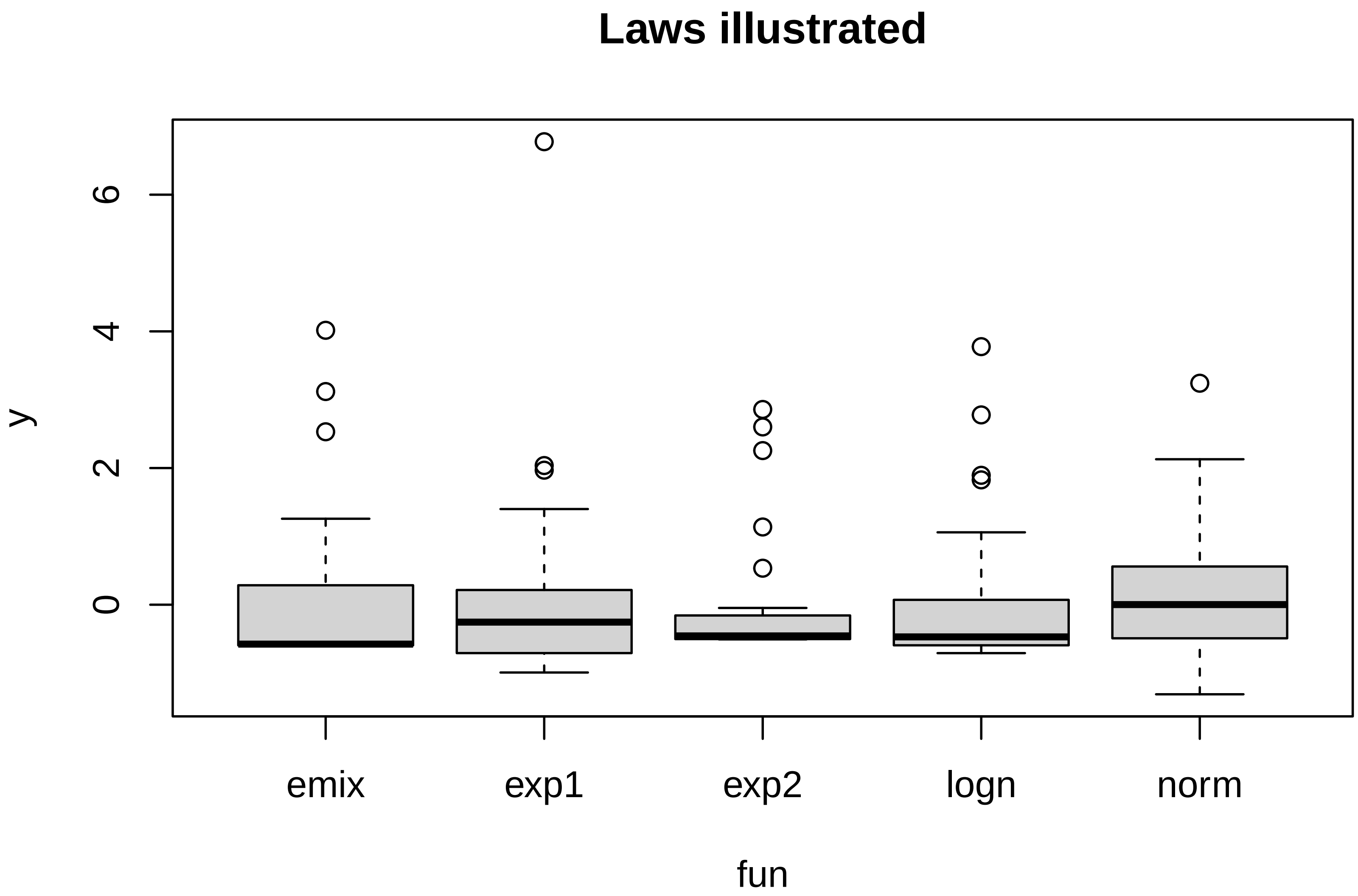
Lukas Graz

2023-09-20

Define a list of laws (i.e. distributions) with `mean` and `sd`.

```r
# functions
n <- 40 # per group
laws <- list(
  norm = function(mean=0, sd=1) mean + sd *  rnorm(n, 0, 1),
  logn = function(mean=0, sd=1) mean + sd * (rlnorm(n, sdlog=1) - 1.65) / 1.95,
  exp1 = function(mean=0, sd=1) mean + sd * (rexp(n) - 1),
  exp2 = function(mean=0, sd=1) mean + sd * (rexp(n)^2 - 2) / 3.94,
  emix = function(mean=0, sd=1) mean + sd * (c(rep(0, n/2), rexp(n/2)) - 0.5) / 0.84
)
```

```r
set.seed(123)
get_data_all_laws <- function(){
  dat <- lapply(names(laws), function(fname){
    f <- laws[[fname]]
    data.frame(y=f(), fun=fname)
  })
  dat <- do.call(rbind, dat)
  dat
}
boxplot(y ~ fun, get_data_all_laws(), main= "Laws illustrated")
```



verify mean and standard deviation

```
sapply(laws, function(f) mean(replicate(10000, mean(f(0, 1)))))
```

```
##      norm      logn      exp1      exp2      emix
##  1.55e-03 -2.89e-03 -1.66e-03 -8.53e-05 -1.32e-03
```

```
sapply(laws, function(f) mean(replicate(10000, mean(f(1, 2)))))
```

```
##  norm  logn  exp1  exp2  emix
## 1.003 1.002 0.998 0.994 0.997
```

```
sapply(laws, function(f) mean(replicate(10000,   sd(f(0, 1)))))
```

```
##  norm  logn  exp1  exp2  emix
## 0.994 1.008 0.977 0.999 1.000
```

```
sapply(laws, function(f) mean(replicate(10000,   sd(f(1, 2)))))
```

```
## norm logn exp1 exp2 emix
## 1.99 2.01 1.96 1.99 2.00
```

Define the tests used

```
pval_t <- function(d) t.test(y ~ group,d)$p.value
pval_w <- function(d) coin::pvalue(coin::wilcox_test(y ~ group, d))
pval_m <- function(d) coin::pvalue(coin::median_test(y ~ group, d))
```

```
# retuns data with f1() for group "A" and f2(mean2, sd2) for group "B"
get_data <- function(f1, f2, mean2=0, sd2=1){
  d <- rbind(
    data.frame(
      y=f1(),
      group="A"
```

```r
    ),
    data.frame(
      y=f2(mean2, sd2),
      group="B"
    ))
  d$group <- as.factor(d$group)
  d
}

# get power of tests
get_power <- function(f1, f2, nsim=1000, mean2=0, sd2=1) {
  data_list <- replicate(nsim, get_data(f1, f2, mean2=mean2, sd2=sd2), simplify = FALSE)
  c(t = mean(sapply(data_list, pval_t) < 0.05),
    w = mean(sapply(data_list, pval_w) < 0.05),
    m = mean(sapply(data_list, pval_m) < 0.05))
}

# get all combinations of functions
fun_comb <- expand.grid(names(laws), names(laws)) |> as.matrix()
rnames <- apply(fun_comb, 1, paste0, collapse="_")

sim <- function(nsim=1000, mean2=0, sd2=1) {
  fun_comb_list <- split(fun_comb, row(fun_comb))
  coverage <- parallel::mclapply(fun_comb_list, function(f_names){
    f1 <- laws[[f_names[1]]]
    f2 <- laws[[f_names[2]]]
    get_power(f1, f2, nsim=nsim, mean2=mean2, sd2=sd2)
  })
  coverage <- do.call(rbind, coverage)
  rownames(coverage) <- rnames
  colnames(coverage) <- paste0(
    colnames(coverage), " ", as.character(mean2), " ", as.character(sd2))
  coverage |> as.data.frame()
}
```

Simulation

```r
nsim <- 1000
```

```r
set.seed(4321)
null <- sim(nsim=nsim)
```

```r
set.seed(4321)
s_diff <- sim(nsim=nsim, sd2=2)
```

```r
set.seed(4321)
s_difff <- sim(nsim=nsim, sd2=5)
```

```r
set.seed(4321)
mu_diff <- sim(nsim=nsim, mean2 = 0.1)
```

```r
set.seed(4321)
mu_difff <- sim(nsim=nsim, mean2 = 0.2)
```

```r
results <- cbind(
  null,
  s_diff,
  s_difff,
  mu_diff,
  mu_difff
)
results * 100
```

	t 0 1	w 0 1	m 0 1	t 0 2	w 0 2	m 0 2	t 0 5	w 0 5	m 0 5	t 0.1 1	w 0.1 1	m 0.1 1	t 0.2 1	w 0.2 1	m 0.2 1
norm__norm	4.2	4.3	3.8	4.0	6.0	5.1	5.0	7.8	10.0	6.3	5.7	4.9	14.7	14.6	10.0
logn__norm	6.3	15.1	32.4	4.1	6.7	17.2	4.2	8.7	12.2	10.9	24.5	44.8	21.7	41.7	62.7
exp1__norm	5.6	8.9	18.3	5.2	6.5	12.5	4.7	8.6	13.3	8.2	18.2	30.8	16.3	32.0	43.4
exp2__norm	5.2	15.4	45.6	5.5	9.3	24.5	5.0	8.4	14.0	11.7	29.0	65.5	21.9	45.4	78.4
emix__norm	3.6	11.5	48.6	5.0	7.1	22.5	5.0	8.2	10.9	6.8	24.3	64.6	14.8	40.7	75.8
norm__logn	5.4	12.5	28.9	7.4	40.3	52.9	9.6	77.3	76.7	5.4	7.3	19.4	9.2	6.6	9.8
logn__logn	5.4	4.9	4.3	6.7	57.4	32.5	8.6	77.6	68.2	7.7	18.2	10.1	14.6	55.0	27.9
exp1__logn	3.6	6.7	5.2	7.1	47.6	26.8	9.8	77.1	67.0	6.5	20.0	6.9	11.0	38.7	16.1
exp2__logn	3.3	8.7	6.1	6.4	57.6	33.8	8.9	79.4	73.1	6.7	10.0	25.2	17.1	51.0	54.1
emix__logn	2.2	2.2	9.6	7.3	54.6	0.7	9.4	76.4	15.3	3.7	34.0	32.1	12.1	91.3	58.8
norm__exp1	5.1	8.8	18.5	6.3	28.9	35.0	6.7	50.7	52.1	6.8	5.8	11.2	11.5	6.9	7.6
logn__exp1	5.4	7.9	6.0	5.5	35.2	19.7	7.3	50.6	44.0	9.5	8.2	10.3	19.6	24.3	21.7
exp1__exp1	3.8	4.3	4.0	6.5	30.8	15.1	6.4	51.5	40.1	7.4	11.0	6.8	14.3	29.1	13.1
exp2__exp1	5.2	12.5	7.8	4.8	33.9	17.2	5.7	51.4	44.6	10.4	6.1	20.2	20.4	18.3	44.9
emix__exp1	3.0	4.4	11.8	3.7	35.2	1.6	6.7	49.5	5.1	5.1	6.8	24.0	13.2	27.5	46.0
norm__exp2	6.6	15.6	47.3	10.2	52.2	83.3	12.9	95.2	95.3	4.3	7.8	29.4	10.0	6.0	16.6
logn__exp2	3.9	9.4	7.8	8.7	85.1	68.5	13.6	96.1	94.0	6.5	36.3	10.0	15.7	67.7	27.1
exp1__exp2	4.3	10.1	7.7	10.4	73.7	61.4	12.8	98.1	93.2	6.0	28.7	7.3	12.9	47.8	15.3
exp2__exp2	4.2	5.3	5.0	9.3	91.2	70.6	12.9	96.0	93.9	6.9	66.0	27.1	18.0	93.2	64.9
emix__exp2	2.3	62.9	10.1	8.2	84.2	5.7	12.3	96.7	49.8	3.5	85.0	36.6	12.2	93.5	64.0
norm__emix	3.7	11.3	50.4	2.9	49.5	77.9	4.5	90.7	89.1	4.3	5.9	31.1	7.8	4.4	16.5
logn__emix	2.1	3.8	12.1	3.4	85.7	64.3	4.0	90.6	86.3	4.5	9.7	2.9	17.4	45.6	0.4
exp1__emix	2.5	4.8	11.6	2.9	88.2	58.4	4.2	91.4	86.8	5.5	13.9	4.5	11.3	32.6	1.0
exp2__emix	2.1	60.4	9.6	2.1	85.3	65.2	4.5	91.1	88.3	6.0	1.4	1.1	17.1	69.9	0.3
emix__emix	1.3	0.0	0.0	2.1	88.7	2.3	4.8	91.5	16.9	2.8	83.7	0.0	9.1	94.9	0.4

Confidence intervals of ratios

```r
prop <- function(ratio, nsim){
  confint_ <- prop.test(round(ratio*nsim), nsim)$conf.int[1:2]
  names(confint_) <- c("lower", "upper")
   c(
    ratio= ratio,
    confint_
)}
sapply(c(0:4/40, 3:10/20), prop, nsim) |> as.data.frame() * 100
```

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13
ratio	0.000	2.50	5.00	7.50	10.00	15.0	20.0	25.0	30.0	35.0	40.0	45.0	50.0
lower	0.000	1.66	3.77	5.98	8.24	12.9	17.6	22.4	27.2	32.1	37.0	41.9	46.9
upper	0.477	3.72	6.59	9.36	12.07	17.4	22.6	27.8	33.0	38.1	43.1	48.1	53.1

this gives an idea of the uncertainty of a ratio given 1000 simulations

redo analysis but with groupwise equal medians

Define a list of laws (i.e. distributions) with `median` and `sd`

```r
# functions
n <- 40 # per group
# mean == to keep notation consistent
```

```r
laws <- list(
  norm = function(mean=0, sd=1) mean + sd *  rnorm(n, 0, 1)
  ,logn = function(mean=0, sd=1) mean + sd * (rlnorm(n, sdlog=1) - 1.02) / 1.95
  ,exp1 = function(mean=0, sd=1) mean + sd * (rexp(n) - 0.706)
  ,exp2 = function(mean=0, sd=1) mean + sd * (rexp(n)^2 - 0.523) / 3.94
  ,emix = function(mean=0, sd=1) mean + sd * (c(rep(0, n/2), rexp(n/2)) -0.025) / 0.84
)
```

verify **median** and standard deviation

```r
sapply(laws, function(f) mean(replicate(10000, median(f(0, 1)))))
```

```
##      norm      logn      exp1      exp2      emix
## -0.000142 -0.000216  0.002144  0.000204  0.000246
```

```r
sapply(laws, function(f) mean(replicate(10000, median(f(1, 2)))))
```

```
##  norm  logn  exp1  exp2  emix
## 1.000 1.001 0.998 1.000 0.999
```

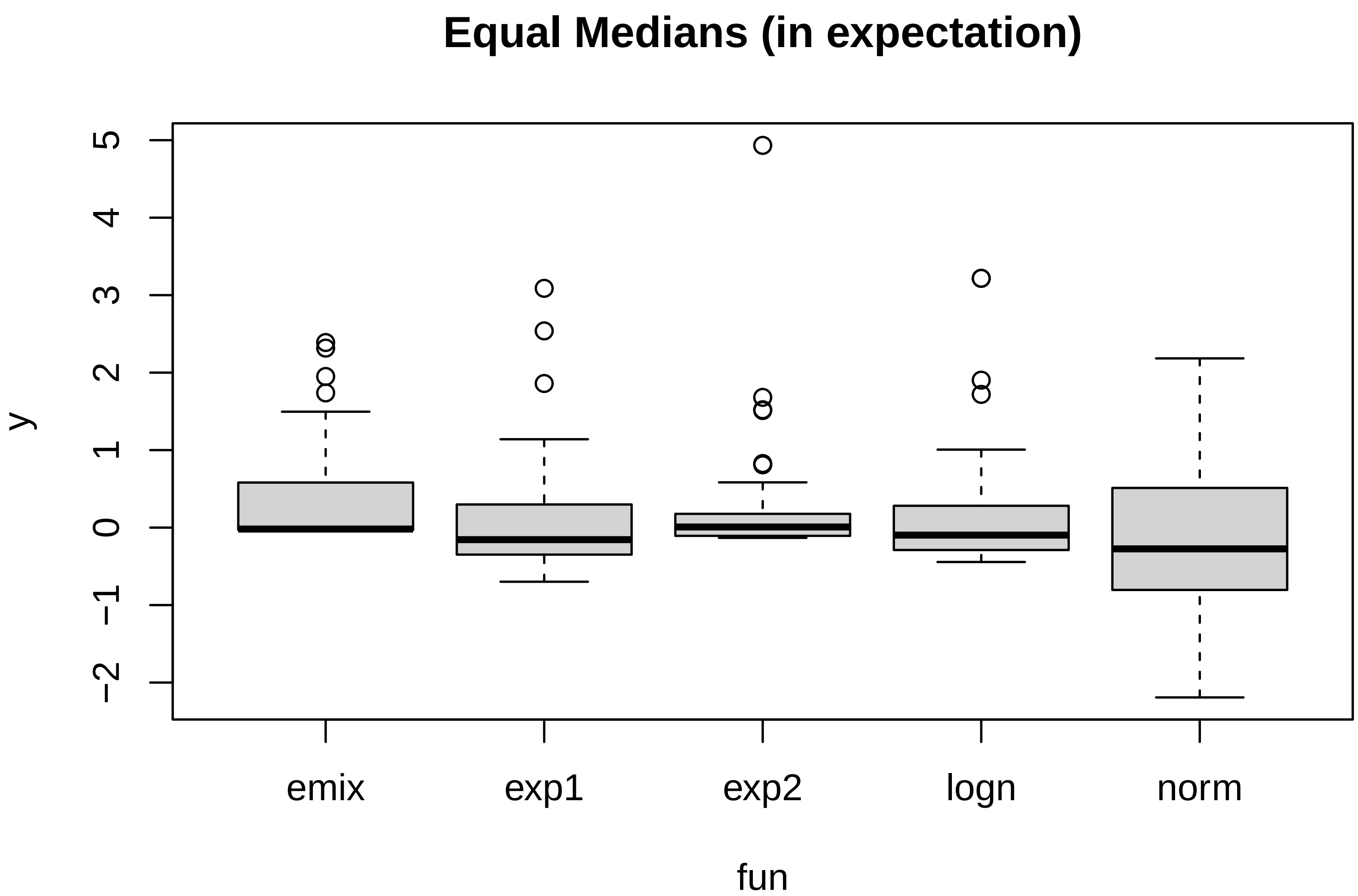
```r
sapply(laws, function(f) mean(replicate(10000,     sd(f(0, 1)))))
```

```
##  norm  logn  exp1  exp2  emix
## 0.996 1.006 0.978 1.010 1.003
```

```r
sapply(laws, function(f) mean(replicate(10000,     sd(f(1, 2)))))
```

```
## norm logn exp1 exp2 emix
## 1.99 2.00 1.95 1.99 2.01
```

```r
boxplot(y ~ fun, get_data_all_laws(), main= "Equal Medians (in expectation)")
```



```r
set.seed(4321)
null_median <- sim(nsim=nsim)
```

```r
set.seed(4321)
s_diff_median <- sim(nsim=nsim, sd2=2)
```

```r
set.seed(4321)
s_difff_median <- sim(nsim=nsim, sd2=5)
```

```r
set.seed(4321)
mu_diff_median <- sim(nsim=nsim, mean2 = 0.1)
```

```r
set.seed(4321)
mu_difff_median <- sim(nsim=nsim, mean2 = 0.2)
```

```r
results_median <- cbind(
  null_median,
  s_diff_median,
  s_difff_median,
  mu_diff_median,
  mu_difff_median
)
results_median * 100
```

	t 0	w 0	m 0	t 0	w 0	m 0	t 0	w 0	m 0	t 0.1	w 0.1	m 0.1	t 0.2	w 0.2	m 0.2
	1	1	1	2	2	2	5	5	5	1	1	1	1	1	1
norm_norm	6.3	5.6	5.2	5.9	6.3	5.5	4.4	9.5	11.3	7.3	6.8	5.5	12.8	13.1	11.0
logn_norm	27.6	18.3	5.6	12.2	13.7	7.7	6.9	10.5	11.0	11.7	7.0	7.4	5.1	5.8	15.8
exp1_norm	23.8	12.6	4.2	12.0	12.2	6.7	5.8	9.3	9.8	10.9	6.0	6.5	5.5	6.1	12.8
exp2_norm	34.2	23.6	7.7	14.6	16.5	10.7	7.6	11.3	14.3	16.1	9.1	9.4	7.3	6.9	18.4
emix_norm	72.4	54.0	0.3	35.7	31.6	1.7	11.7	12.6	4.8	54.8	34.8	2.0	29.9	16.1	6.0
norm_logn	23.0	15.9	6.4	39.5	14.9	5.2	46.4	8.8	7.2	45.1	32.7	8.2	62.4	52.7	14.5
logn_logn	5.3	6.3	4.9	7.7	12.3	6.0	24.5	10.4	9.1	6.0	18.7	10.9	16.8	52.7	26.2
exp1_logn	3.8	9.4	4.3	8.7	6.5	4.6	23.6	8.7	5.9	8.9	25.5	8.7	14.8	47.6	20.8
exp2_logn	4.8	22.9	5.4	6.4	19.1	8.3	21.4	15.0	11.5	6.3	6.5	11.8	12.0	27.9	39.2
emix_logn	19.8	59.4	0.2	4.4	40.1	0.3	11.8	20.8	2.2	8.8	20.4	1.0	3.5	2.7	6.0
norm_exp1	25.1	15.0	4.2	34.8	10.2	5.0	40.0	6.4	7.5	41.3	26.7	7.0	58.7	41.6	10.9
logn_exp1	3.9	8.7	4.6	10.1	12.1	8.1	22.6	9.4	10.2	6.5	4.9	6.3	15.4	17.6	18.1
exp1_exp1	5.3	5.7	4.9	9.6	9.5	5.2	24.4	9.2	8.7	9.7	12.8	7.2	15.1	27.0	12.2
exp2_exp1	4.2	23.3	8.1	6.4	15.4	8.9	21.7	11.3	12.1	5.2	7.4	11.1	11.5	5.8	21.2
emix_exp1	21.4	50.5	0.8	4.7	31.2	2.9	13.4	19.4	3.9	8.7	32.1	1.1	3.5	12.0	5.7
norm_exp2	33.2	23.7	8.9	51.1	29.9	4.7	71.6	27.8	4.2	51.9	40.7	12.6	72.3	59.7	21.6
logn_exp2	3.4	21.2	5.6	15.1	13.2	4.2	43.1	9.8	7.6	9.3	51.7	16.7	22.7	81.8	44.9
exp1_exp2	5.7	24.2	7.3	15.6	22.9	3.9	43.6	6.0	5.6	11.3	43.8	15.0	21.4	64.6	29.7
exp2_exp2	3.0	5.5	5.2	9.4	19.2	7.2	37.0	17.3	11.9	6.5	65.5	25.0	16.3	93.9	65.7
emix_exp2	16.4	64.3	0.2	2.3	48.3	0.4	23.2	28.9	1.0	6.3	6.8	1.3	2.2	63.4	13.2
norm_emix	73.2	52.6	1.0	97.4	76.7	0.5	100.0	87.9	0.0	87.2	72.3	0.8	96.4	88.8	2.9
logn_emix	21.9	58.6	0.4	66.5	73.8	0.0	97.9	70.5	0.0	38.8	87.9	2.5	52.0	95.9	10.5
exp1_emix	21.9	52.2	0.6	68.9	72.2	0.5	99.4	81.9	0.0	39.1	73.2	1.5	59.1	86.6	4.3
exp2_emix	14.4	60.6	0.1	58.9	66.2	0.0	97.1	0.0	0.0	30.9	94.4	4.8	46.6	99.2	25.7
emix_emix	1.2	0.0	0.0	22.7	7.3	0.0	95.8	0.2	0.0	3.3	83.1	0.0	10.6	93.4	0.1

T-Test Vs Wilcoxon

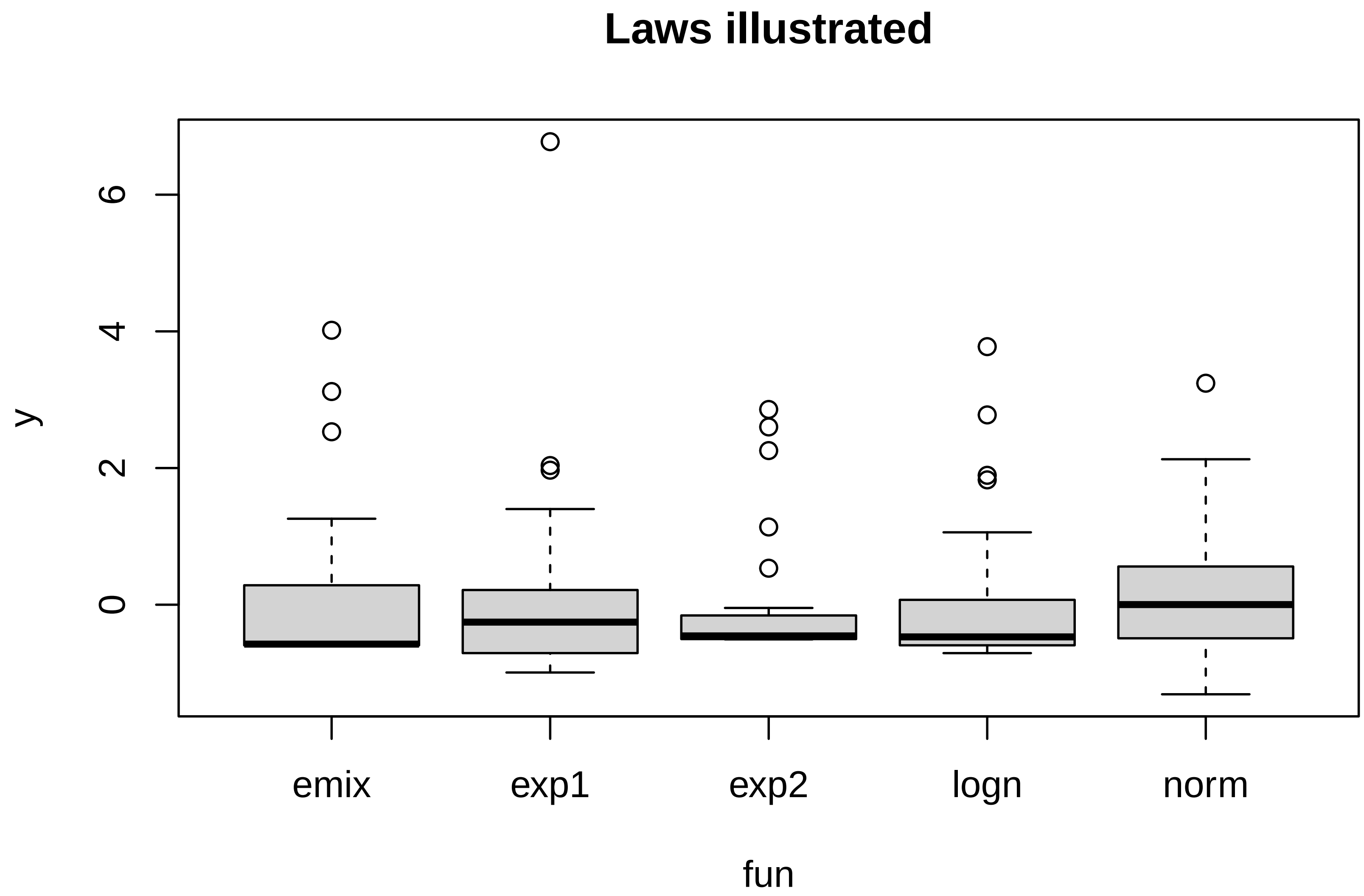
Lukas Graz

2023-09-20

Define a list of laws (i.e. distributions) with `mean` and `sd`.

```
# functions
n <- 40 # per group
laws <- list(
  norm = function(mean=0, sd=1) mean + sd *  rnorm(n, 0, 1),
  logn = function(mean=0, sd=1) mean + sd * (rlnorm(n, sdlog=1) - 1.65) / 1.95,
  exp1 = function(mean=0, sd=1) mean + sd * (rexp(n) - 1),
  exp2 = function(mean=0, sd=1) mean + sd * (rexp(n)^2 - 2) / 3.94,
  emix = function(mean=0, sd=1) mean + sd * (c(rep(0, n/2), rexp(n/2)) - 0.5) / 0.84
)
```

```
set.seed(123)
get_data_all_laws <- function(){
  dat <- lapply(names(laws), function(fname){
    f <- laws[[fname]]
    data.frame(y=f(), fun=fname)
  })
  dat <- do.call(rbind, dat)
  dat
}
boxplot(y ~ fun, get_data_all_laws(), main= "Laws illustrated")
```



verify mean and standard deviation

```r
sapply(laws, function(f) mean(replicate(10000, mean(f(0, 1)))))
```

```
##      norm      logn      exp1      exp2      emix
##  1.55e-03 -2.89e-03 -1.66e-03 -8.53e-05 -1.32e-03
```

```r
sapply(laws, function(f) mean(replicate(10000, mean(f(1, 2)))))
```

```
##  norm  logn  exp1  exp2  emix
## 1.003 1.002 0.998 0.994 0.997
```

```r
sapply(laws, function(f) mean(replicate(10000,   sd(f(0, 1)))))
```

```
##  norm  logn  exp1  exp2  emix
## 0.994 1.008 0.977 0.999 1.000
```

```r
sapply(laws, function(f) mean(replicate(10000,   sd(f(1, 2)))))
```

```
## norm logn exp1 exp2 emix
## 1.99 2.01 1.96 1.99 2.00
```

Define the tests used

```r
pval_t <- function(d) t.test(y ~ group,d)$p.value
pval_w <- function(d) coin::pvalue(coin::wilcox_test(y ~ group, d))
pval_m <- function(d) coin::pvalue(coin::median_test(y ~ group, d))
```

```r
# retuns data with f1() for group "A" and f2(mean2, sd2) for group "B"
get_data <- function(f1, f2, mean2=0, sd2=1){
  d <- rbind(
    data.frame(
      y=f1(),
      group="A"
```

```
    ),
    data.frame(
      y=f2(mean2, sd2),
      group="B"
    ))
  d$group <- as.factor(d$group)
  d
}

# get power of tests
get_power <- function(f1, f2, nsim=1000, mean2=0, sd2=1) {
  data_list <- replicate(nsim, get_data(f1, f2, mean2=mean2, sd2=sd2), simplify = FALSE)
  c(t = mean(sapply(data_list, pval_t) < 0.05),
    w = mean(sapply(data_list, pval_w) < 0.05),
    m = mean(sapply(data_list, pval_m) < 0.05))
}

# get all combinations of functions
fun_comb <- expand.grid(names(laws), names(laws)) |> as.matrix()
rnames <- apply(fun_comb, 1, paste0, collapse="_")

sim <- function(nsim=1000, mean2=0, sd2=1) {
  fun_comb_list <- split(fun_comb, row(fun_comb))
  coverage <- parallel::mclapply(fun_comb_list, function(f_names){
    f1 <- laws[[f_names[1]]]
    f2 <- laws[[f_names[2]]]
    get_power(f1, f2, nsim=nsim, mean2=mean2, sd2=sd2)
  })
  coverage <- do.call(rbind, coverage)
  rownames(coverage) <- rnames
  colnames(coverage) <- paste0(
    colnames(coverage), " ", as.character(mean2), " ", as.character(sd2))
  coverage |> as.data.frame()
}
```

Simulation

```
nsim <- 1000
```

```
set.seed(4321)
null <- sim(nsim=nsim)
```

```
set.seed(4321)
s_diff <- sim(nsim=nsim, sd2=2)
```

```
set.seed(4321)
s_difff <- sim(nsim=nsim, sd2=5)
```

```
set.seed(4321)
mu_diff <- sim(nsim=nsim, mean2 = 0.1)
```

```
set.seed(4321)
mu_difff <- sim(nsim=nsim, mean2 = 0.2)
```

```
results <- cbind(
  null,
  s_diff,
  s_difff,
  mu_diff,
  mu_difff
)
results * 100
```

	t 0 1	w 0 1	m 0 1	t 0 2	w 0 2	m 0 2	t 0 5	w 0 5	m 0 5	t 0.1 1	w 0.1 1	m 0.1 1	t 0.2 1	w 0.2 1	m 0.2 1
norm_norm	4.2	4.3	3.8	4.0	6.0	5.1	5.0	7.8	10.0	6.3	5.7	4.9	14.7	14.6	10.0
logn_norm	6.3	15.1	32.4	4.1	6.7	17.2	4.2	8.7	12.2	10.9	24.5	44.8	21.7	41.7	62.7
exp1_norm	5.6	8.9	18.3	5.2	6.5	12.5	4.7	8.6	13.3	8.2	18.2	30.8	16.3	32.0	43.4
exp2_norm	5.2	15.4	45.6	5.5	9.3	24.5	5.0	8.4	14.0	11.7	29.0	65.5	21.9	45.4	78.4
emix_norm	3.6	11.5	48.6	5.0	7.1	22.5	5.0	8.2	10.9	6.8	24.3	64.6	14.8	40.7	75.8
norm_logn	5.4	12.5	28.9	7.4	40.3	52.9	9.6	77.3	76.7	5.4	7.3	19.4	9.2	6.6	9.8
logn_logn	5.4	4.9	4.3	6.7	57.4	32.5	8.6	77.6	68.2	7.7	18.2	10.1	14.6	55.0	27.9
exp1_logn	3.6	6.7	5.2	7.1	47.6	26.8	9.8	77.1	67.0	6.5	20.0	6.9	11.0	38.7	16.1
exp2_logn	3.3	8.7	6.1	6.4	57.6	33.8	8.9	79.4	73.1	6.7	10.0	25.2	17.1	51.0	54.1
emix_logn	2.2	2.2	9.6	7.3	54.6	0.7	9.4	76.4	15.3	3.7	34.0	32.1	12.1	91.3	58.8
norm_exp1	5.1	8.8	18.5	6.3	28.9	35.0	6.7	50.7	52.1	6.8	5.8	11.2	11.5	6.9	7.6
logn_exp1	5.4	7.9	6.0	5.5	35.2	19.7	7.3	50.6	44.0	9.5	8.2	10.3	19.6	24.3	21.7
exp1_exp1	3.8	4.3	4.0	6.5	30.8	15.1	6.4	51.5	40.1	7.4	11.0	6.8	14.3	29.1	13.1
exp2_exp1	5.2	12.5	7.8	4.8	33.9	17.2	5.7	51.4	44.6	10.4	6.1	20.2	20.4	18.3	44.9
emix_exp1	3.0	4.4	11.8	3.7	35.2	1.6	6.7	49.5	5.1	5.1	6.8	24.0	13.2	27.5	46.0
norm_exp2	6.6	15.6	47.3	10.2	52.2	83.3	12.9	95.2	95.3	4.3	7.8	29.4	10.0	6.0	16.6
logn_exp2	3.9	9.4	7.8	8.7	85.1	68.5	13.6	96.1	94.0	6.5	36.3	10.0	15.7	67.7	27.1
exp1_exp2	4.3	10.1	7.7	10.4	73.7	61.4	12.8	98.1	93.2	6.0	28.7	7.3	12.9	47.8	15.3
exp2_exp2	4.2	5.3	5.0	9.3	91.2	70.6	12.9	96.0	93.9	6.9	66.0	27.1	18.0	93.2	64.9
emix_exp2	2.3	62.9	10.1	8.2	84.2	5.7	12.3	96.7	49.8	3.5	85.0	36.6	12.2	93.5	64.0
norm_emix	3.7	11.3	50.4	2.9	49.5	77.9	4.5	90.7	89.1	4.3	5.9	31.1	7.8	4.4	16.5
logn_emix	2.1	3.8	12.1	3.4	85.7	64.3	4.0	90.6	86.3	4.5	9.7	2.9	17.4	45.6	0.4
exp1_emix	2.5	4.8	11.6	2.9	88.2	58.4	4.2	91.4	86.8	5.5	13.9	4.5	11.3	32.6	1.0
exp2_emix	2.1	60.4	9.6	2.1	85.3	65.2	4.5	91.1	88.3	6.0	1.4	1.1	17.1	69.9	0.3
emix_emix	1.3	0.0	0.0	2.1	88.7	2.3	4.8	91.5	16.9	2.8	83.7	0.0	9.1	94.9	0.4

Confidence intervals of ratios

```
prop <- function(ratio, nsim){
  confint_ <- prop.test(round(ratio*nsim), nsim)$conf.int[1:2]
  names(confint_) <- c("lower", "upper")
   c(
    ratio= ratio,
    confint_
)}
sapply(c(0:4/40, 3:10/20), prop, nsim) |> as.data.frame() * 100
```

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13
ratio	0.000	2.50	5.00	7.50	10.00	15.0	20.0	25.0	30.0	35.0	40.0	45.0	50.0
lower	0.000	1.66	3.77	5.98	8.24	12.9	17.6	22.4	27.2	32.1	37.0	41.9	46.9
upper	0.477	3.72	6.59	9.36	12.07	17.4	22.6	27.8	33.0	38.1	43.1	48.1	53.1

this gives an idea of the uncertainty of a ratio given 1000 simulations

redo analysis but with groupwise equal medians

Define a list of laws (i.e. distributions) with `median` and `sd`

```
# functions
n <- 40 # per group
# mean == to keep notation consistent
```

```r
laws <- list(
  norm = function(mean=0, sd=1) mean + sd *  rnorm(n, 0, 1)
  ,logn = function(mean=0, sd=1) mean + sd * (rlnorm(n, sdlog=1) - 1.02) / 1.95
  ,exp1 = function(mean=0, sd=1) mean + sd * (rexp(n) - 0.706)
  ,exp2 = function(mean=0, sd=1) mean + sd * (rexp(n)^2 - 0.523) / 3.94
  ,emix = function(mean=0, sd=1) mean + sd * (c(rep(0, n/2), rexp(n/2)) -0.025) / 0.84
)
```

verify **median** and standard deviation

```r
sapply(laws, function(f) mean(replicate(10000, median(f(0, 1)))))
```

```
##      norm      logn      exp1      exp2      emix
## -0.000142 -0.000216  0.002144  0.000204  0.000246
```

```r
sapply(laws, function(f) mean(replicate(10000, median(f(1, 2)))))
```

```
##  norm  logn  exp1  exp2  emix
## 1.000 1.001 0.998 1.000 0.999
```

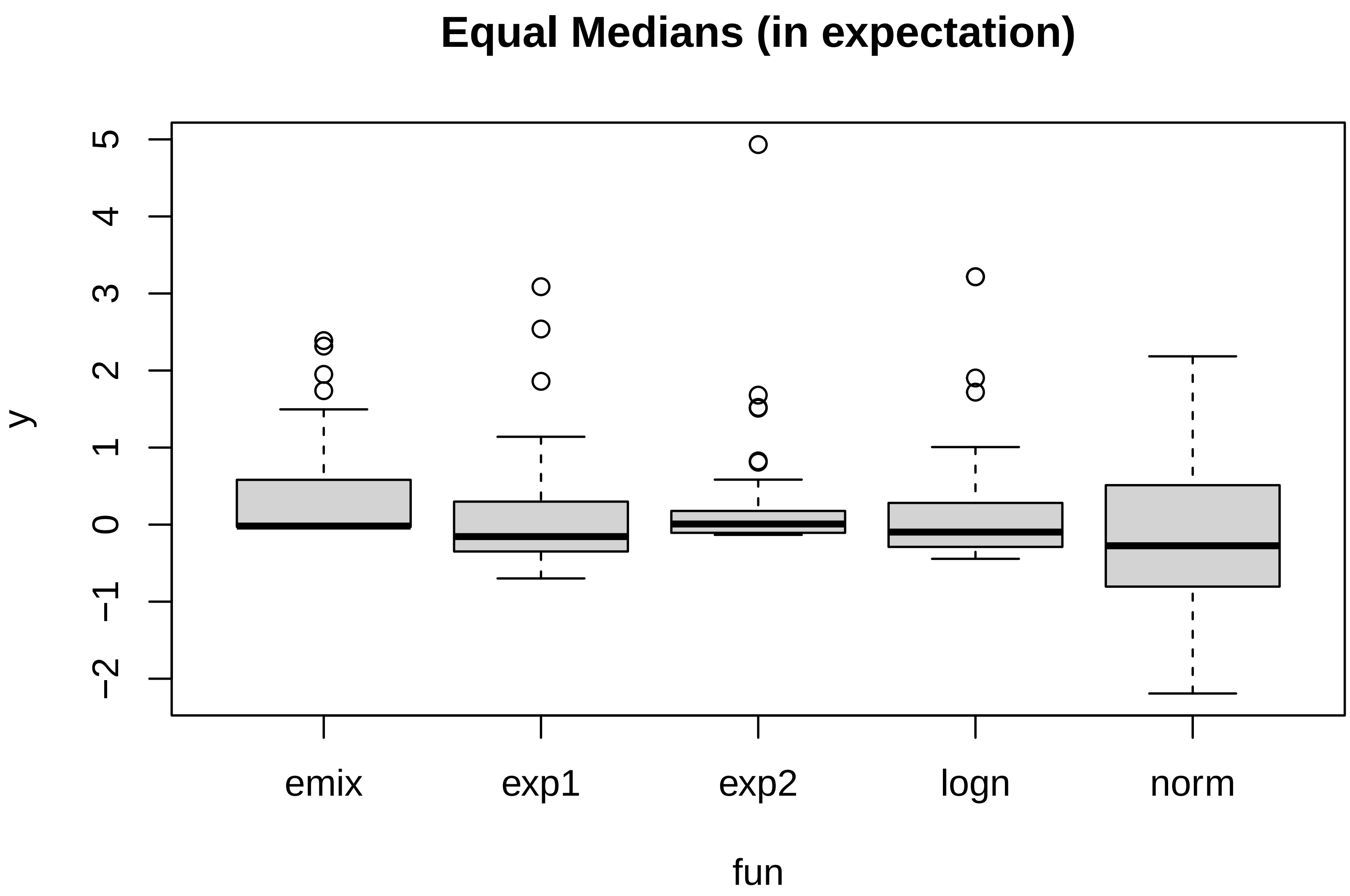
```r
sapply(laws, function(f) mean(replicate(10000,     sd(f(0, 1)))))
```

```
##  norm  logn  exp1  exp2  emix
## 0.996 1.006 0.978 1.010 1.003
```

```r
sapply(laws, function(f) mean(replicate(10000,     sd(f(1, 2)))))
```

```
## norm logn exp1 exp2 emix
## 1.99 2.00 1.95 1.99 2.01
```

```r
boxplot(y ~ fun, get_data_all_laws(), main= "Equal Medians (in expectation)")
```



```r
set.seed(4321)
null_median <- sim(nsim=nsim)
```

```r
set.seed(4321)
s_diff_median <- sim(nsim=nsim, sd2=2)
```

```r
set.seed(4321)
s_difff_median <- sim(nsim=nsim, sd2=5)
```

```r
set.seed(4321)
mu_diff_median <- sim(nsim=nsim, mean2 = 0.1)
```

```r
set.seed(4321)
mu_difff_median <- sim(nsim=nsim, mean2 = 0.2)
```

```r
results_median <- cbind(
  null_median,
  s_diff_median,
  s_difff_median,
  mu_diff_median,
  mu_difff_median
)
results_median * 100
```

	t 0 1	w 0 1	m 0 1	t 0 2	w 0 2	m 0 2	t 0 5	w 0 5	m 0 5	t 0.1 1	w 0.1 1	m 0.1 1	t 0.2 1	w 0.2 1	m 0.2 1
norm_norm	6.3	5.6	5.2	5.9	6.3	5.5	4.4	9.5	11.3	7.3	6.8	5.5	12.8	13.1	11.0
logn_norm	27.6	18.3	5.6	12.2	13.7	7.7	6.9	10.5	11.0	11.7	7.0	7.4	5.1	5.8	15.8
exp1_norm	23.8	12.6	4.2	12.0	12.2	6.7	5.8	9.3	9.8	10.9	6.0	6.5	5.5	6.1	12.8
exp2_norm	34.2	23.6	7.7	14.6	16.5	10.7	7.6	11.3	14.3	16.1	9.1	9.4	7.3	6.9	18.4
emix_norm	72.4	54.0	0.3	35.7	31.6	1.7	11.7	12.6	4.8	54.8	34.8	2.0	29.9	16.1	6.0
norm_logn	23.0	15.9	6.4	39.5	14.9	5.2	46.4	8.8	7.2	45.1	32.7	8.2	62.4	52.7	14.5
logn_logn	5.3	6.3	4.9	7.7	12.3	6.0	24.5	10.4	9.1	6.0	18.7	10.9	16.8	52.7	26.2
exp1_logn	3.8	9.4	4.3	8.7	6.5	4.6	23.6	8.7	5.9	8.9	25.5	8.7	14.8	47.6	20.8
exp2_logn	4.8	22.9	5.4	6.4	19.1	8.3	21.4	15.0	11.5	6.3	6.5	11.8	12.0	27.9	39.2
emix_logn	19.8	59.4	0.2	4.4	40.1	0.3	11.8	20.8	2.2	8.8	20.4	1.0	3.5	2.7	6.0
norm_exp1	25.1	15.0	4.2	34.8	10.2	5.0	40.0	6.4	7.5	41.3	26.7	7.0	58.7	41.6	10.9
logn_exp1	3.9	8.7	4.6	10.1	12.1	8.1	22.6	9.4	10.2	6.5	4.9	6.3	15.4	17.6	18.1
exp1_exp1	5.3	5.7	4.9	9.6	9.5	5.2	24.4	9.2	8.7	9.7	12.8	7.2	15.1	27.0	12.2
exp2_exp1	4.2	23.3	8.1	6.4	15.4	8.9	21.7	11.3	12.1	5.2	7.4	11.1	11.5	5.8	21.2
emix_exp1	21.4	50.5	0.8	4.7	31.2	2.9	13.4	19.4	3.9	8.7	32.1	1.1	3.5	12.0	5.7
norm_exp2	33.2	23.7	8.9	51.1	29.9	4.7	71.6	27.8	4.2	51.9	40.7	12.6	72.3	59.7	21.6
logn_exp2	3.4	21.2	5.6	15.1	13.2	4.2	43.1	9.8	7.6	9.3	51.7	16.7	22.7	81.8	44.9
exp1_exp2	5.7	24.2	7.3	15.6	22.9	3.9	43.6	6.0	5.6	11.3	43.8	15.0	21.4	64.6	29.7
exp2_exp2	3.0	5.5	5.2	9.4	19.2	7.2	37.0	17.3	11.9	6.5	65.5	25.0	16.3	93.9	65.7
emix_exp2	16.4	64.3	0.2	2.3	48.3	0.4	23.2	28.9	1.0	6.3	6.8	1.3	2.2	63.4	13.2
norm_emix	73.2	52.6	1.0	97.4	76.7	0.5	100.0	87.9	0.0	87.2	72.3	0.8	96.4	88.8	2.9
logn_emix	21.9	58.6	0.4	66.5	73.8	0.0	97.9	70.5	0.0	38.8	87.9	2.5	52.0	95.9	10.5
exp1_emix	21.9	52.2	0.6	68.9	72.2	0.5	99.4	81.9	0.0	39.1	73.2	1.5	59.1	86.6	4.3
exp2_emix	14.4	60.6	0.1	58.9	66.2	0.0	97.1	0.0	0.0	30.9	94.4	4.8	46.6	99.2	25.7
emix_emix	1.2	0.0	0.0	22.7	7.3	0.0	95.8	0.2	0.0	3.3	83.1	0.0	10.6	93.4	0.1