power_effsize

January 14, 2019

1 1 Effect Size Calculation

1.1 1.1 Cohen's d for t-tests

Suppose we are interested in whether births are more likely on a weekday or a weekend. We will use a two-sample t-test to investigate this.

```
In [ ]: library(fivethirtyeight)
        library(ggplot2)
        library(dplyr)
        library(lsr)

        us_births <- US_births_2000_2014
```

```
In [ ]: head(us_births)
```

```
In [ ]: # Add a column indicating if it's a weekend birth

        us_births <- us_births %>%
                       mutate(is_weekend = day_of_week == "Sat" | day_of_week == "Sun")
```

```
In [ ]: ggplot(us_births, aes(x = is_weekend, y = births, fill = is_weekend)) +
            geom_boxplot(width = 0.15, colour = "#777777")
```

```
In [ ]: t.test(births ~ is_weekend, us_births)
```

```
In [ ]: cohensD(births ~ factor(is_weekend), us_births)
```

2 1.2 ω^2 for ANOVA

Now suppose we want to test the difference in births across all days of the week.

```
In [ ]: ggplot(us_births, aes(x = day_of_week, y = births, fill = day_of_week)) +
            geom_boxplot(width = 0.2, colour = "#777777") +
            geom_jitter(width = 0.2, alpha = 0.1, aes(colour = day_of_week))
```

```
In [ ]: fm1 <- aov(births ~ day_of_week, us_births)
```

```
In [ ]: anova(fm1)
```

```
In [ ]: etaSquared(fm1)
```

---

3 2 Power

3.1 2.1 Power & Sample Size

Suppose we want to know the necessary sample size in a two-sample t-test with Cohen's $d = 0.5$, 80% power $\alpha = 0.05$.

```
In [ ]: library(pwr)

        pwr.t.test(d = 0.5, power = 0.80, sig.level = 0.05)
```

Suppose we had a paired t-test design with Cohen's $d = 0.5$, 80% power $\alpha = 0.05$.

```
In [ ]: pwr.t.test(d = 0.5, power = 0.80, sig.level = 0.05, type = "paired")
```

3.2 2.2 Simulation for Power Analysis

For certain species of models, there is not a closed form solution for estimating power. In these cases, we use Monte Carlo simulations to generate our estimates of power. One common example of cases such as this is when we are estimating the power for linear or generalized linear models with many or multiple coefficients.

3.2.1 2.2.1 Simulating Linear Model

```
In [ ]: library(paramtest)
```

```
In [ ]: # Define function lm_test() to simulate linear models
        # and the power of their respective coefficients

        lm_test <- function(simNum, N, b1, b0 = 0, xm = 0, xsd = 1) {
            x <- rnorm(N, xm, xsd)
            y <- rnorm(N, b0 + b1*x, sqrt(1 - b1^2))  # var. approx. 1 after accounting
                                                       # for explained variance by x
            model <- lm(y ~ 1 + x)

            # pull output from model
            est <- coef(summary(model))["x", "Estimate"]
            se <- coef(summary(model))["x", "Std. Error"]
```

```
    p <- coef(summary(model))["x", "Pr(>|t|)"]

    res <- c(xm = mean(x), xsd = sd(x), ym = mean(y), ysd = sd(y), est = est, se = se, p

    return(res)
}

# We vary N from 100 to 500; we are also setting coefficient
# of x predicting y to be approx. 0.15 across all simulations
power_lm <- grid_search(lm_test,
                        params = list(N = c(100, 200, 300, 400, 500)),
                        n.iter = 1000,
                        output = "data.frame",
                        b1 = 0.15,
                        parallel = "snow",
                        ncpus = 4)

results(power_lm) %>%
    group_by(N.test) %>%
    summarise(power = mean(sig))
```

3.2.2 2.2.2 Simulating Linear Model with Interaction

```
In [ ]: lm_test_interaction <- function(simNum, N, b1, b2, b3, b0 = 0, x1m = 0, x1sd = 1, x2m =

    x1 <- rnorm(N, x1m, x1sd)
    x2 <- rnorm(N, x2m, x2sd)
    yvar <- sqrt(1 - b1^2 - b2^2 - b3^2)  # residual variance
    y <- rnorm(N, b0 + b1*x1 + b2*x2 + b3*x1*x2, yvar)
    model <- lm(y ~ x1 * x2)

    # pull output from model (two main effects and interaction)
    est_x1 <- coef(summary(model))['x1', 'Estimate']
    p_x1 <- coef(summary(model))['x1', 'Pr(>|t|)']

    est_x2 <- coef(summary(model))['x2', 'Estimate']
    p_x2 <- coef(summary(model))['x2', 'Pr(>|t|)']

    est_x2x3 <- coef(summary(model))['x1:x2', 'Estimate']
    p_x2x3 <- coef(summary(model))['x1:x2', 'Pr(>|t|)']

    res <- c(est_x1 = est_x1, p_x1 = p_x1, sig_x1 = (p_x1 < 0.05), est_x2 = est_x2, p_x2
        sig_x2 = (p_x2 < 0.05), est_x2x3 = est_x2x3, p_int = p_x2x3, sig_x2x3 = p_x2x3 <

    return(res)
}

# We vary N from 100 to 500; setting coefficient of x1 = 0.15,
# coefficient of x2 = 0, and coefficient of interaction = 0.3
power_lm_int <- grid_search(lm_test_interaction,
                            params = list(N = c(100, 200, 300, 400, 500)),
                            n.iter = 1000,
                            output = "data.frame",
                            b1 = 0.15,
                            b2 = 0,
                            b3 = 0.3,
                            parallel = "snow",
                            ncpus = 4)

results(power_lm_int) %>%
    group_by(N.test) %>%
    summarise(
        power_x1 = mean(sig_x1),
        power_x2 = mean(sig_x2),
        power_x2x3 = mean(sig_x2x3))
```

4 3 Post-Hoc Tests

```
In [ ]: fm1 <- aov(births ~ day_of_week, us_births)
```

```
In [ ]: pairwise.t.test(us_births$births, us_births$day_of_week, "fdr")
```

```
In [ ]: TukeyHSD(fm1)
```