# *How to Calculate and Report Effect Sizes*

*Aaron R. Caldwell*

*2020-03-02*

## *Introduction*

In our manuscript we have argued that sport and exercise scientists should avoid the temptation of reporting a "default" effect size. Instead we suggested that the effect size be chosen on a case-by-case basis and reported in a way that helps communicate the results appropriately. With that said, we realize many sport scientists may not know how to calculate these effect sizes in a efficient manner. Therefore, we have written this document to show how [R Core Team, 2019] can be used to produce these analyses.

### *Loading the Right Tools*

None of the following packages are 'necessary', and all of these analyses can be performed in 'base' R. But, I use these packages to make the analysis pipeline simpler and a little 'cleaner' too. In total, I need the tidyverse [Wickham et al., 2019], boot [Canty and Ripley, 2019], broom [Robinson and Hayes, 2019], and MASS [Venables and Ripley, 2002]. You will likely not need MASS or broom for your own analyses, but it is necessary for this document in order to create a simulated dataset (MASS) and make the tables in the document (broom).

```
#LOAD PACKAGES
library(tidyverse)
library(boot)
library(MASS)
library(broom)
```

## *Generate Some Data*

Okay, now I can generate data we similar to what we used in the manuscipt. First, let us create the "Interpretatable Raw Differences" example. In this example, we have a pre-to-post design of athletes where we are measuring $\dot{V}O_2$ (L·min$^{-1}$). Further, we assume we are sampling from a population wherein the standard deviation of 0.2, correlation pre/post of .9, and a mean increase of .3 (L·min$^{-1}$).

```
set.seed(20200303)
var_raw = .2^2
cor_raw = matrix(c(1,.8,.8,1),nrow=2)
Sigma_raw = cor_raw*var_raw
```

```
df_raw = mvrnorm(
  n = 10,
  Sigma = Sigma_raw,
  mu = c(3.9, 4.1)
  ) %>%
  as.data.frame() %>%
  rename(pre = V1,
         post = V2)

#Add a change score column
df_raw = df_raw %>% mutate(pre = round(pre,2),
                           post = round(post,2)) %>%
  mutate(diff = post-pre)
```

Table 1: Raw Differences Data

| pre | post | diff |
|---|---|---|
| 3.85 | 4.15 | 0.30 |
| 3.75 | 4.11 | 0.36 |
| 4.03 | 4.14 | 0.11 |
| 3.95 | 4.08 | 0.13 |
| 4.22 | 4.40 | 0.18 |
| 4.20 | 4.11 | -0.09 |
| 3.74 | 3.99 | 0.25 |
| 4.13 | 4.17 | 0.04 |
| 3.98 | 4.05 | 0.07 |
| 4.05 | 4.20 | 0.15 |

Now, we want to create "Uninterpretable Raw Differences" dataset. This time we are creating a dummy dataset of VAS scores with 30 'point' reduction, with a standard deviation of 25, and correlation of .7.

```
set.seed(20201113)

var_stan = 25^2
cor_stan = matrix(c(1,.7,.7,1),nrow=2)
Sigma_stan = cor_stan*var_stan
df_stan = mvrnorm(
  n = 10,
  Sigma = Sigma_stan,
  mu = c(50,25)
  ) %>%
  as.data.frame() %>%
```

```
  rename(pre = V1,
         post = V2)

#Add a change score column
df_stan = df_stan %>% mutate(pre = round(pre,2),
                             post = round(post,2)) %>%
  mutate(diff = post-pre)
```

Table 2: Standardized Differences Data

| pre | post | diff |
|---|---|---|
| 12.32 | -11.38 | -23.70 |
| 43.37 | 13.83 | -29.54 |
| 19.95 | 36.84 | 16.89 |
| 72.90 | 27.84 | -45.06 |
| 15.75 | -9.23 | -24.98 |
| 54.57 | 1.96 | -52.61 |
| 25.51 | -23.94 | -49.45 |
| 60.76 | 22.24 | -38.52 |
| 37.29 | 19.39 | -17.90 |
| 65.86 | 15.61 | -50.25 |

### *The Simple Analysis*

Now we have our data, we can perform a simple *t*-test on the differences scores for both datasets.

```
t_raw = t.test(df_raw$diff)
knitr::kable(tidy(t_raw),caption="Raw Differences t-test",
             digits = 4)
```

Table 3: Raw Differences t-test

| estimate | statistic | p.value | parameter | conf.low | conf.high | method | alternative |
|---|---|---|---|---|---|---|---|
| 0.15 | 3.6075 | 0.0057 | 9 | 0.0559 | 0.2441 | One Sample t-test | two.sided |

```
t_stan = t.test(df_stan$diff)
knitr::kable(tidy(t_stan),caption="Standardized Differences t-test",
             digits = 4)
```

Table 4: Standardized Differences t-test

| estimate | statistic | p.value | parameter | conf.low | conf.high | method | alternative |
|---|---|---|---|---|---|---|---|
| -31.512 | -4.7349 | 0.0011 | 9 | -46.5672 | -16.4568 | One Sample t-test | two.sided |

Okay, so we now have our *t*-statistics, *p*-values, mean differences, and confidence intervals. Pretty painless to get this far, but maybe we want (or need) to get the standardized mean difference (SMD) and common langauge effect sizes (CLES).

### *Let's Bootstrap*

While it is easy to calculate the SMD or CLES in R, it is quite another thing to calculate the confidence intervals. Unlike the mean difference, there is no simple formula for calculating the SMD.

First, we must create a function to caclulate all the SMD statistics we would like to obtain.

```
# function to obtain R-Squared from the data
SMD <- function(data, indices) {
  d <- data[indices,] # allows boot to select sample
  sd_pre = sd(d$pre, na.rm = TRUE)
  sd_post = sd(d$post, na.rm = TRUE)
  sd_av = (sd_pre+sd_post)/2
  m_diff = mean(d$diff, na.rm = TRUE)
  sd_diff = sd(d$diff, na.rm = TRUE)
  cor_prepost = cor(d$pre,d$post,
                    method = "pearson")
  dz = m_diff/sd_diff
  dav = m_diff/sd_av
  glass = m_diff/sd_pre
  drm = m_diff/sqrt((sd_pre^2+sd_post^2)-(2*cor_prepost*sd_pre*sd_post))*sqrt(2*(1-cor_prepost))
  CLES = pnorm(dz)
  result = c(dz,drm,dav,glass,CLES)

}
```

Second, we can start bootstrapping these statistics. In this we select each value based on their index (1-5) from the function above. We must also specify the *type* of bootstrap confidence intervals. In this case, I have decided to report the bias-corrected and accelerated (BCa) intervals as it will likely provide our most accurate estimate.

```
raw_boot = boot(df_raw, SMD, R = 2000)
#Extract the values
```

```
dz_raw = boot.ci(raw_boot, type="bca", index=1)
drm_raw = boot.ci(raw_boot, type="bca", index=2)
dav_raw = boot.ci(raw_boot, type="bca", index=3)
glass_raw = boot.ci(raw_boot, type="bca", index=4)
CLES_raw = boot.ci(raw_boot, type="bca", index=5)

#Repeat for the standardized dataset
stan_boot = boot(df_stan, SMD, R = 2000)
dz_stan = boot.ci(stan_boot, type="bca", index=1)
drm_stan = boot.ci(stan_boot, type="bca", index=2)
dav_stan = boot.ci(stan_boot, type="bca", index=3)
glass_stan = boot.ci(stan_boot, type="bca", index=4)
CLES_stan = boot.ci(stan_boot, type="bca", index=5)
```

From these statistics we can extract the effect size estimate from bootstrapped values by calling on the `$t0`

For example:

```
dz_raw$t0
```

```
## [1] 1.140795
```

And the upper limit and lower limit of the 95% confidence interval can also be found in the `dz_raw$bca` object. The last and second-to-last numbers representing the upper limit and lower limit of the 95% confidence interval.

```
dz_raw$bca
```

```
##      conf
## [1,] 0.95 4.33 1824.97 0.2679737 2.011248
```

Finally, we can put all the SMD calculations in a summary table. Note, that the effect sizes are negative in the 2nd table are negative, that is because there was a decrease from pre-to-post, and the CLES should be reported as $1 - CLES$ from the table.

Table 5: Effect Sizes for 'Raw' Differences Dataset

| Effect Size | Estimate | Lower Limit | Upper Limit |
|---|---|---|---|
| d(z) | 1.1408 | 0.2680 | 2.0112 |
| d(rm) | 0.9688 | 0.4511 | 1.6086 |
| d(av) | 1.0689 | 0.4581 | 1.7875 |
| Glass's Delta | 0.8771 | 0.3087 | 1.3134 |
| CLES | 0.8730 | 0.5740 | 0.9761 |

Table 6: Effect Sizes for 'Standardized' Differences Dataset

| Effect Size | Estimate | Lower Limit | Upper Limit |
|---|---|---|---|
| d(z) | -1.4973 | -3.1976 | -0.4122 |
| d(rm) | -1.5140 | -2.4391 | -0.7780 |
| d(av) | -1.5235 | -2.3101 | -0.6902 |
| Glass's Delta | -1.4297 | -2.1534 | -0.6963 |
| CLES | 0.0672 | 0.0007 | 0.3649 |

## *References*

Angelo Canty and B. D. Ripley. *boot: Bootstrap R (S-Plus) Functions*, 2019. R package version 1.3-24.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2019. URL `https://www.R-project.org/`.

David Robinson and Alex Hayes. *broom: Convert Statistical Analysis Objects into Tidy Tibbles*, 2019. URL `https://CRAN.R-project.org/package=broom`. R package version 0.5.3.

W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer, New York, fourth edition, 2002. URL `http://www.stats.ox.ac.uk/pub/MASS4`. ISBN 0-387-95457-0.

Hadley Wickham, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D'Agostino McGowan, Romain François, Garrett Grolemund, Alex Hayes, Lionel Henry, Jim Hester, Max Kuhn, Thomas Lin Pedersen, Evan Miller, Stephan Milton Bache, Kirill Müller, Jeroen Ooms, David Robinson, Dana Paige Seidel, Vitalie Spinu, Kohske Takahashi, Davis Vaughan, Claus Wilke, Kara Woo, and Hiroaki Yutani. Welcome to the tidyverse. *Journal of Open Source Software*, 4(43):1686, 2019. DOI: 10.21105/joss.01686.