29.4: Comparing Paired Observations (Section 28.5)

Let's look at how to perform a paired t-test in R. In this case, let's generate some data for a set of individuals on two tests, where each indivdual varies in their overall ability, but there is also a practice effect such that performance on the second test is generally better than the first.

First, let's see how big of a sample we will require to find a medium (d=0.5) sized effect. Let's say that we want to be extra sure in our results, so we will find the sample size that gives us 95% power to find an effect if it's there:

```
0.50.95paired_power <- pwr.t.test(d=, power=, type='paired', alternative='greater')
paired_power
```

```
##
##      Paired t test power calculation
##
##               n = 45
##               d = 0.5
##       sig.level = 0.05
##           power = 0.95
##     alternative = greater
##
## NOTE: n is number of *pairs*
```

Now let's generate a dataset with the required number of subjects:

```
subject_id <- seq(paired_power$n)
# we code the tests as 0/1 so that we can simply
# multiply this by the effect to generate the data
test_id <- c(0,1)
repeat_effect <- 5
noise_sd <- 5

subject_means <- rnorm(paired_power$n, mean=100, sd=15)
paired_data <- crossing(subject_id,test_id) %>%
  mutate(subMean=subject_means[subject_id],
         score=subject_means +
           test_id*repeat_effect +
           rnorm(paired_power$n, mean=noise_sd))
```

Let's perform a paired t-test on these data. To do that, we need to separate the first and second test data into separate variables, which we can do by converting our *long* data frame into a *wide* data frame.

```
paired_data_wide <- paired_data %>%
  spread(test_id, score) %>%
  rename(test1=`0`,
         test2=`1`)

glimpse(paired_data_wide)
```

```
## Observations: 44
## Variables: 4
## $ subject_id <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,…
## $ subMean    <dbl> 116, 95, 103, 91, 97, 91, 89, 97, 99, …
## $ test1      <dbl> 121, 108, 102, 94, 105, 111, 110, 89, …
## $ test2      <dbl> 104, 101, 102, 107, 108, 101, 157, 126…
```

Now we can pass those new variables into the `t.test()` function:

```
paired_ttest_result <- t.test(paired_data_wide$test1,
                              paired_data_wide$test2,
                              type='paired')
paired_ttest_result
```

```
##
##  Welch Two Sample t-test
##
## data:  paired_data_wide$test1 and paired_data_wide$test2
## t = -1, df = 73, p-value = 0.2
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -10.5   2.3
## sample estimates:
## mean of x mean of y
##       108       112
```

This analysis is a bit trickier to perform using the linear model, because we need to estimate a separate intercept for each subject in order to account for the overall differences between subjects. We can't do this using `lm()` but we can do it using a function called `lmer()` from the `lme4` package. To do this, we need to add `(1|subject_id)` to the formula, which tells `lmer()` to add a separate intercept ("1") for each value of `subject_id` .

```
paired_test_lmer <- lmer(score ~ test_id + (1|subject_id),
                         data=paired_data)
summary(paired_test_lmer)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: score ~ test_id + (1 | subject_id)
##    Data: paired_data
##
## REML criterion at convergence: 719
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.5424 -0.6214 -0.0929  0.7349  2.9793
##
## Random effects:
##  Groups     Name        Variance Std.Dev.
##  subject_id (Intercept)   0       0.0
##  Residual               228      15.1
## Number of obs: 88, groups:  subject_id, 44
##
## Fixed effects:
##             Estimate Std. Error     df t value Pr(>|t|)
## (Intercept)   107.59       2.28  86.00   47.26   <2e-16 ***
## test_id         4.12       3.22  86.00    1.28      0.2
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##         (Intr)
## test_id -0.707
## convergence code: 0
## boundary (singular) fit: see ?isSingular
```

This gives a similar answer to the standard paired t-test. The advantage is that it's more flexible, allowing us to perform *repeated measures* analyses, as we will see below.
