Final Project Code

Clark Enge

Simulation plan

1. Set up each scenario

```
# Define all scenarios
scenarios <- expand.grid(
  effect_size = c(0, 0.1, 0.2, 0.3, 0.5),        # Include 0 for Type I error analysis
  sample_size = c(10, 30, 50, 100),
  outlier_percent = c(0, 2.5, 5, 10, 15)
)

# Number of simulations per scenario
n_sim <- 10000

# Set seed for reproducibility
set.seed(12345)
```

2. Generate data

Generate normal data(base mean = 60, SD = 10) Shift group B mean by d * sd add outliers of x% values of 200

```
generate_data <- function(n, effect, outlier_percent, base_mean = 60, sd = 10) {
  n_clean <- round(n * (1 - outlier_percent / 100))
  n_outlier <- n - n_clean

  # Add slight variance and mean noise
  mu1 <- base_mean + rnorm(1, 0, 2)
  mu2 <- base_mean + effect * sd + rnorm(1, 0, 2)
  sd1 <- sd
  sd2 <- sd + abs(effect) * 5 + runif(1, 0, 5)

  # Group A: clean, no outliers
  group1 <- rnorm(n, mu1, sd1)
  group1 <- pmax(group1, 0)  # enforce non-negative time

  # Group B: clean data + positive-skewed outliers
  clean_b <- rnorm(n_clean, mu2, sd2)
  clean_b <- pmax(clean_b, 0)

  outliers_b <- rnorm(n_outlier, mean = mu2 + 100, sd = 40)  # large values only
```

```
  outliers_b <- pmax(outliers_b, 0)

  group2 <- c(clean_b, outliers_b)

  list(
    group1 = group1,
    group2 = group2
  )
}
```

3. Apply the tests

You can also use WRS2::yuen() for robust trimmed mean t-test for formal implementation

```
# Function to run three different t-tests on group data
run_tests <- function(a, b) {
  t_equal <- t.test(a, b, var.equal = TRUE)$p.value
  t_welch <- t.test(a, b, var.equal = FALSE)$p.value
  t_trim  <- t.test(a, b, var.equal = TRUE, trim = 0.1)$p.value

  list(
    t = t_equal,
    welch = t_welch,
    trimmed = t_trim
  )
}

# Quick test
test_data <- generate_data(n = 10, effect = 0.2, outlier_percent = 10)
run_tests(test_data$group1, test_data$group2)
```

```
## $t
## [1] 0.3871183
##
## $welch
## [1] 0.3944064
##
## $trimmed
## [1] 0.3871183
```

4. Run tests and store results

```
# Initialize results data frame
results_df <- data.frame()

# Loop over all scenarios
for (i in 1:nrow(scenarios)) {
  effect   <- scenarios$effect_size[i]
  sample   <- scenarios$sample_size[i]
  outlier  <- scenarios$outlier_percent[i]
```

```r
  sig_t <- sig_welch <- sig_trim <- 0

  for (sim in 1:n_sim) {
    data <- generate_data(n = sample, effect = effect, outlier_percent = outlier)
    p_vals <- run_tests(data$group1, data$group2)

    sig_t     <- sig_t     + (p_vals$t < 0.05)
    sig_welch <- sig_welch + (p_vals$welch < 0.05)
    sig_trim  <- sig_trim  + (p_vals$trimmed < 0.05)
  }

  # Store power (or Type I error if effect_size == 0)
  results_df <- rbind(results_df, data.frame(
    effect_size = effect,
    sample_size = sample,
    outlier_percent = outlier,
    power_t      = sig_t / n_sim,
    power_welch  = sig_welch / n_sim,
    power_trim   = sig_trim / n_sim
  ))
}

print(results_df)
```

```
##     effect_size sample_size outlier_percent power_t power_welch power_trim
## 1           0.0          10             0.0  0.0799      0.0784     0.0799
## 2           0.1          10             0.0  0.0860      0.0830     0.0860
## 3           0.2          10             0.0  0.0981      0.0947     0.0981
## 4           0.3          10             0.0  0.1102      0.1074     0.1102
## 5           0.5          10             0.0  0.1645      0.1596     0.1645
## 6           0.0          30             0.0  0.1647      0.1641     0.1647
## 7           0.1          30             0.0  0.1677      0.1668     0.1677
## 8           0.2          30             0.0  0.1884      0.1875     0.1884
## 9           0.3          30             0.0  0.2378      0.2370     0.2378
## 10          0.5          30             0.0  0.3655      0.3632     0.3655
## 11          0.0          50             0.0  0.2258      0.2256     0.2258
## 12          0.1          50             0.0  0.2258      0.2256     0.2258
## 13          0.2          50             0.0  0.2790      0.2780     0.2790
## 14          0.3          50             0.0  0.3386      0.3381     0.3386
## 15          0.5          50             0.0  0.4997      0.4986     0.4997
## 16          0.0         100             0.0  0.3290      0.3290     0.3290
## 17          0.1         100             0.0  0.3486      0.3484     0.3486
## 18          0.2         100             0.0  0.3996      0.3993     0.3996
## 19          0.3         100             0.0  0.4804      0.4803     0.4804
## 20          0.5         100             0.0  0.6772      0.6771     0.6772
## 21          0.0          10             2.5  0.0874      0.0847     0.0874
## 22          0.1          10             2.5  0.0852      0.0825     0.0852
## 23          0.2          10             2.5  0.0981      0.0950     0.0981
## 24          0.3          10             2.5  0.1153      0.1111     0.1153
## 25          0.5          10             2.5  0.1644      0.1588     0.1644
## 26          0.0          30             2.5  0.0969      0.0941     0.0969
## 27          0.1          30             2.5  0.1325      0.1294     0.1325
## 28          0.2          30             2.5  0.1841      0.1787     0.1841
```

```
## 29          0.3          30           2.5  0.2468      0.2404      0.2468
## 30          0.5          30           2.5  0.3983      0.3904      0.3983
## 31          0.0          50           2.5  0.1499      0.1479      0.1499
## 32          0.1          50           2.5  0.2049      0.2030      0.2049
## 33          0.2          50           2.5  0.2791      0.2768      0.2791
## 34          0.3          50           2.5  0.3583      0.3551      0.3583
## 35          0.5          50           2.5  0.5570      0.5535      0.5570
## 36          0.0         100           2.5  0.2757      0.2748      0.2757
## 37          0.1         100           2.5  0.3618      0.3598      0.3618
## 38          0.2         100           2.5  0.4522      0.4512      0.4522
## 39          0.3         100           2.5  0.5721      0.5711      0.5721
## 40          0.5         100           2.5  0.7547      0.7539      0.7547
## 41          0.0          10           5.0  0.0842      0.0814      0.0842
## 42          0.1          10           5.0  0.0898      0.0871      0.0898
## 43          0.2          10           5.0  0.1003      0.0976      0.1003
## 44          0.3          10           5.0  0.1153      0.1112      0.1153
## 45          0.5          10           5.0  0.1623      0.1552      0.1623
## 46          0.0          30           5.0  0.1317      0.1244      0.1317
## 47          0.1          30           5.0  0.1826      0.1744      0.1826
## 48          0.2          30           5.0  0.2581      0.2469      0.2581
## 49          0.3          30           5.0  0.3257      0.3141      0.3257
## 50          0.5          30           5.0  0.4973      0.4818      0.4973
## 51          0.0          50           5.0  0.1837      0.1805      0.1837
## 52          0.1          50           5.0  0.2475      0.2440      0.2475
## 53          0.2          50           5.0  0.3514      0.3464      0.3514
## 54          0.3          50           5.0  0.4426      0.4377      0.4426
## 55          0.5          50           5.0  0.6400      0.6352      0.6400
## 56          0.0         100           5.0  0.4310      0.4282      0.4310
## 57          0.1         100           5.0  0.5498      0.5476      0.5498
## 58          0.2         100           5.0  0.6586      0.6573      0.6586
## 59          0.3         100           5.0  0.7607      0.7586      0.7607
## 60          0.5         100           5.0  0.8932      0.8917      0.8932
## 61          0.0          10          10.0  0.0246      0.0201      0.0246
## 62          0.1          10          10.0  0.0327      0.0254      0.0327
## 63          0.2          10          10.0  0.0486      0.0383      0.0486
## 64          0.3          10          10.0  0.0606      0.0506      0.0606
## 65          0.5          10          10.0  0.1097      0.0870      0.1097
## 66          0.0          30          10.0  0.2277      0.2111      0.2277
## 67          0.1          30          10.0  0.3070      0.2895      0.3070
## 68          0.2          30          10.0  0.3909      0.3719      0.3909
## 69          0.3          30          10.0  0.4642      0.4452      0.4642
## 70          0.5          30          10.0  0.6338      0.6157      0.6338
## 71          0.0          50          10.0  0.4928      0.4830      0.4928
## 72          0.1          50          10.0  0.5972      0.5885      0.5972
## 73          0.2          50          10.0  0.6854      0.6775      0.6854
## 74          0.3          50          10.0  0.7688      0.7607      0.7688
## 75          0.5          50          10.0  0.8817      0.8775      0.8817
## 76          0.0         100          10.0  0.8106      0.8078      0.8106
## 77          0.1         100          10.0  0.8733      0.8714      0.8733
## 78          0.2         100          10.0  0.9278      0.9267      0.9278
## 79          0.3         100          10.0  0.9598      0.9588      0.9598
## 80          0.5         100          10.0  0.9875      0.9873      0.9875
## 81          0.0          10          15.0  0.0459      0.0276      0.0459
## 82          0.1          10          15.0  0.0641      0.0411      0.0641
```

```
## 83          0.2          10           15.0  0.0915      0.0612      0.0915
## 84          0.3          10           15.0  0.1171      0.0817      0.1171
## 85          0.5          10           15.0  0.1843      0.1273      0.1843
## 86          0.0          30           15.0  0.3742      0.3522      0.3742
## 87          0.1          30           15.0  0.4699      0.4461      0.4699
## 88          0.2          30           15.0  0.5511      0.5281      0.5511
## 89          0.3          30           15.0  0.6304      0.6087      0.6304
## 90          0.5          30           15.0  0.7752      0.7588      0.7752
## 91          0.0          50           15.0  0.8528      0.8456      0.8528
## 92          0.1          50           15.0  0.9015      0.8962      0.9015
## 93          0.2          50           15.0  0.9358      0.9312      0.9358
## 94          0.3          50           15.0  0.9559      0.9533      0.9559
## 95          0.5          50           15.0  0.9867      0.9858      0.9867
## 96          0.0         100           15.0  0.9762      0.9757      0.9762
## 97          0.1         100           15.0  0.9889      0.9888      0.9889
## 98          0.2         100           15.0  0.9938      0.9934      0.9938
## 99          0.3         100           15.0  0.9976      0.9975      0.9976
## 100         0.5         100           15.0  0.9992      0.9992      0.9992
```

5. Compute differences between methods and run paired t-tests between methods

Compute differences

```
results_df$diff_trim_vs_t  <- results_df$power_trim - results_df$power_t

results_df$diff_welch_vs_t <- results_df$power_welch - results_df$power_t

results_df$diff_trim_vs_welch <- results_df$power_trim - results_df$power_welch
```

Pair t-tests between methods

```
# Full dataset comparisons
t1 <- t.test(results_df$power_trim, results_df$power_t, paired = TRUE)
t2 <- t.test(results_df$power_welch, results_df$power_t, paired = TRUE)
t3 <- t.test(results_df$power_trim, results_df$power_welch, paired = TRUE)

# Optional: on outlier-only subset
outlier_df <- subset(results_df, outlier_percent > 0)
t1_out <- t.test(outlier_df$power_trim, outlier_df$power_t, paired = TRUE)
t2_out <- t.test(outlier_df$power_welch, outlier_df$power_t, paired = TRUE)
t3_out <- t.test(outlier_df$power_trim, outlier_df$power_welch, paired = TRUE)
```

Format into table

```
library(knitr)

# Build table from the full dataset
diff_summary <- data.frame(
  Comparison = c("Trimmed vs t-test", "Welch vs t-test", "Trimmed vs Welch"),
  Mean_Difference = round(c(
    mean(results_df$power_trim - results_df$power_t),
```

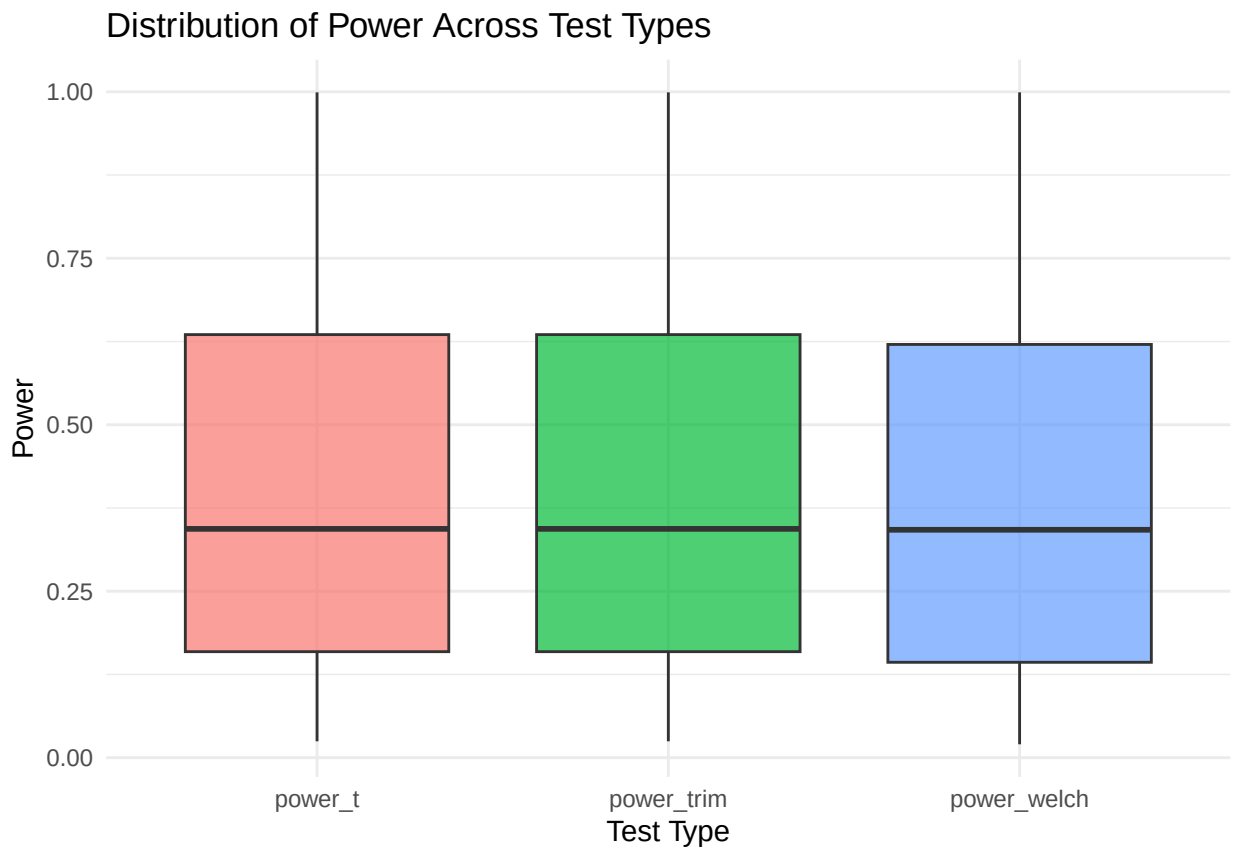
```
    mean(results_df$power_welch - results_df$power_t),
    mean(results_df$power_trim - results_df$power_welch)
  ), 4),
  p_value = signif(c(t1$p.value, t2$p.value, t3$p.value), 4)
)

kable(diff_summary, caption = "Paired t-test Comparison of Average Power Between A/B Testing Methods")
```

Table 1: Paired t-test Comparison of Average Power Between A/B Testing Methods

Comparison	Mean_Difference	p_value
Trimmed vs t-test	0.0000	NaN
Welch vs t-test	-0.0068	0
Trimmed vs Welch	0.0068	0

Visualize power differences

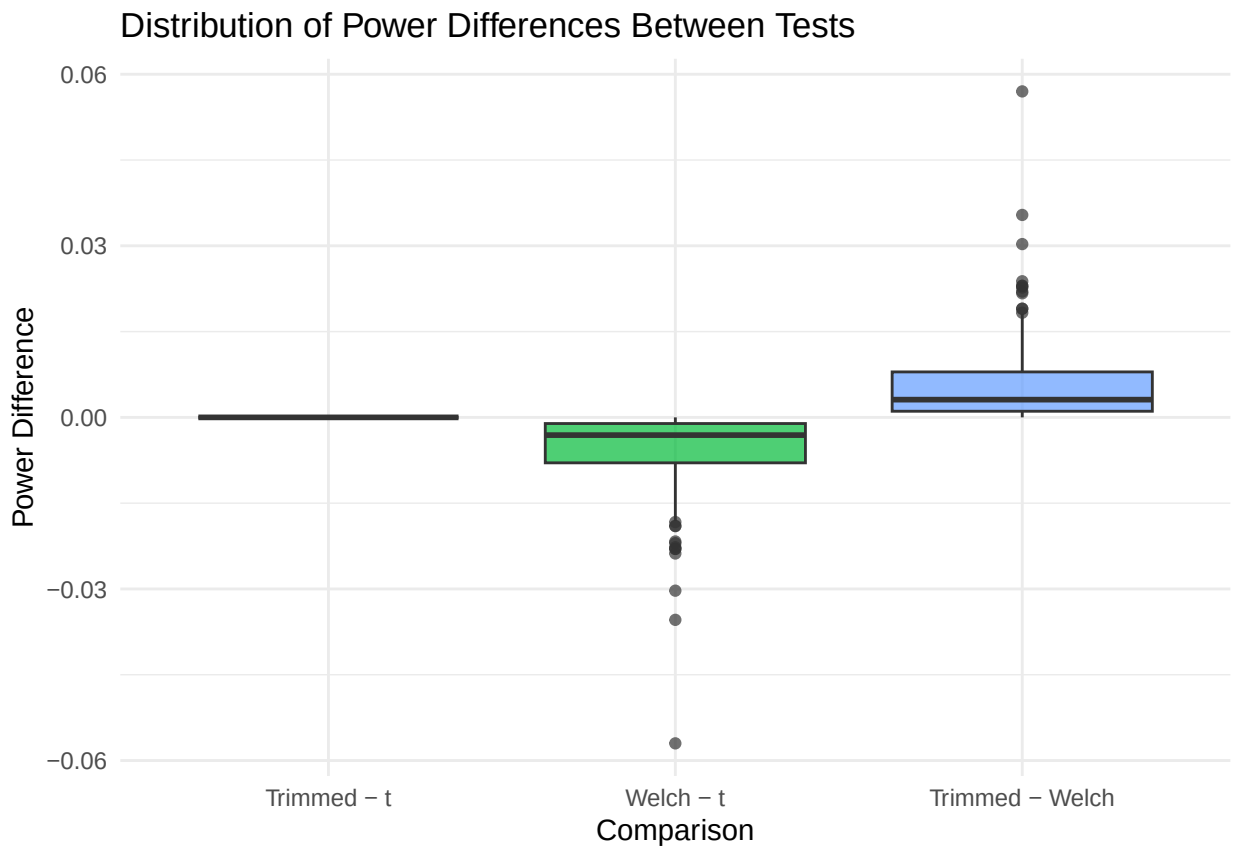
```
library(ggplot2)
library(tidyr)

# Reshape to long format for boxplot
results_long <- pivot_longer(
  results_df,
  cols = c("power_t", "power_welch", "power_trim"),
  names_to = "test_type",
  values_to = "power"
)

power_diff <- ggplot(results_long, aes(x = test_type, y = power, fill = test_type)) +
  geom_boxplot(alpha = 0.7) +
  labs(title = "Distribution of Power Across Test Types",
       x = "Test Type", y = "Power") +
  theme_minimal() +
  theme(legend.position = "none")
power_diff
```



Plot the differences in power between pairs

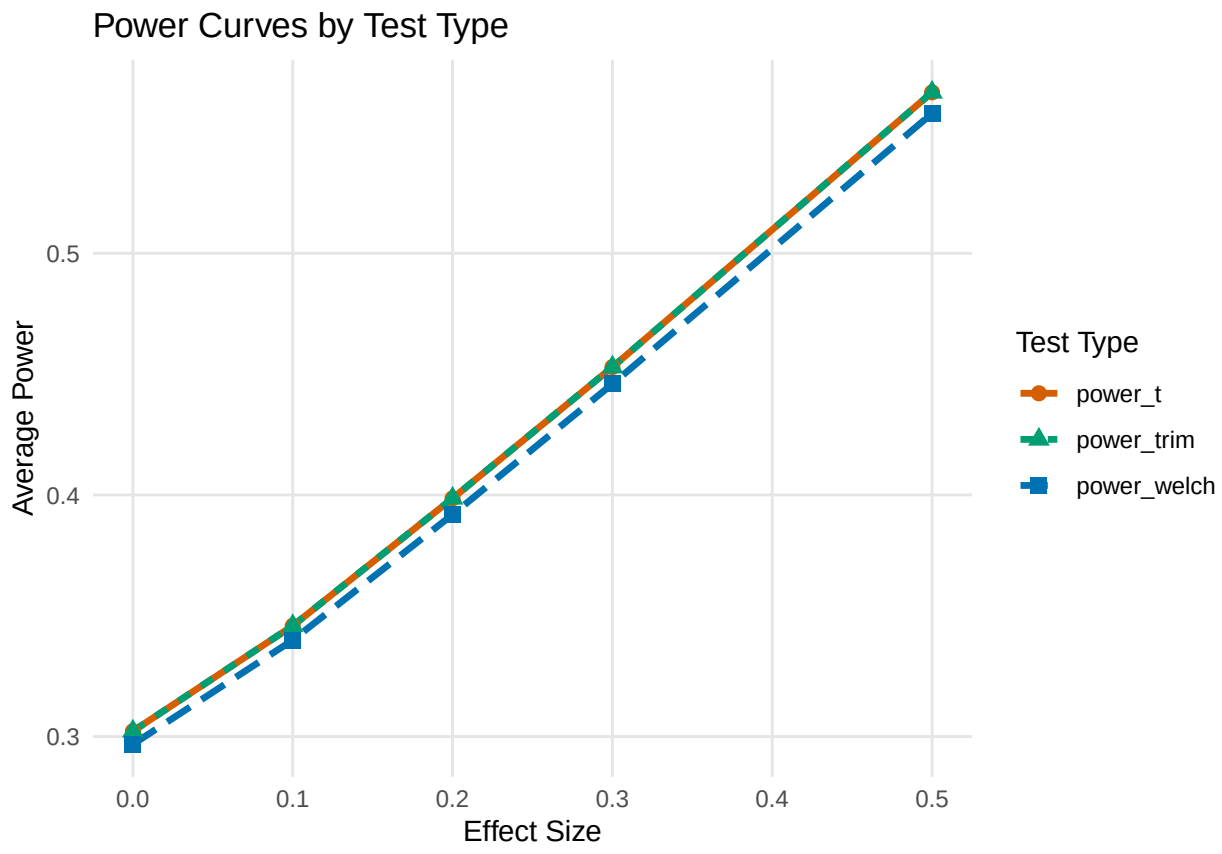
```
# Compute pairwise differences
results_df$diff_trim_vs_t <- results_df$power_trim - results_df$power_t
results_df$diff_welch_vs_t <- results_df$power_welch - results_df$power_t
results_df$diff_trim_vs_welch <- results_df$power_trim - results_df$power_welch

# Combine into long format
diffs_long <- pivot_longer(
  results_df,
  cols = c(diff_trim_vs_t, diff_welch_vs_t, diff_trim_vs_welch),
  names_to = "comparison",
  values_to = "difference"
)

# Rename for cleaner plot labels
diffs_long$comparison <- factor(diffs_long$comparison,
  levels = c("diff_trim_vs_t", "diff_welch_vs_t", "diff_trim_vs_welch"),
  labels = c("Trimmed - t", "Welch - t", "Trimmed - Welch")
)

# Plot distribution of differences
dist_differences <- ggplot(diffs_long, aes(x = comparison, y = difference, fill = comparison)) +
  geom_boxplot(alpha = 0.7) +
  labs(title = "Distribution of Power Differences Between Tests",
       x = "Comparison", y = "Power Difference") +
```

```
  theme_minimal() +
  theme(legend.position = "none")
dist_differences
```



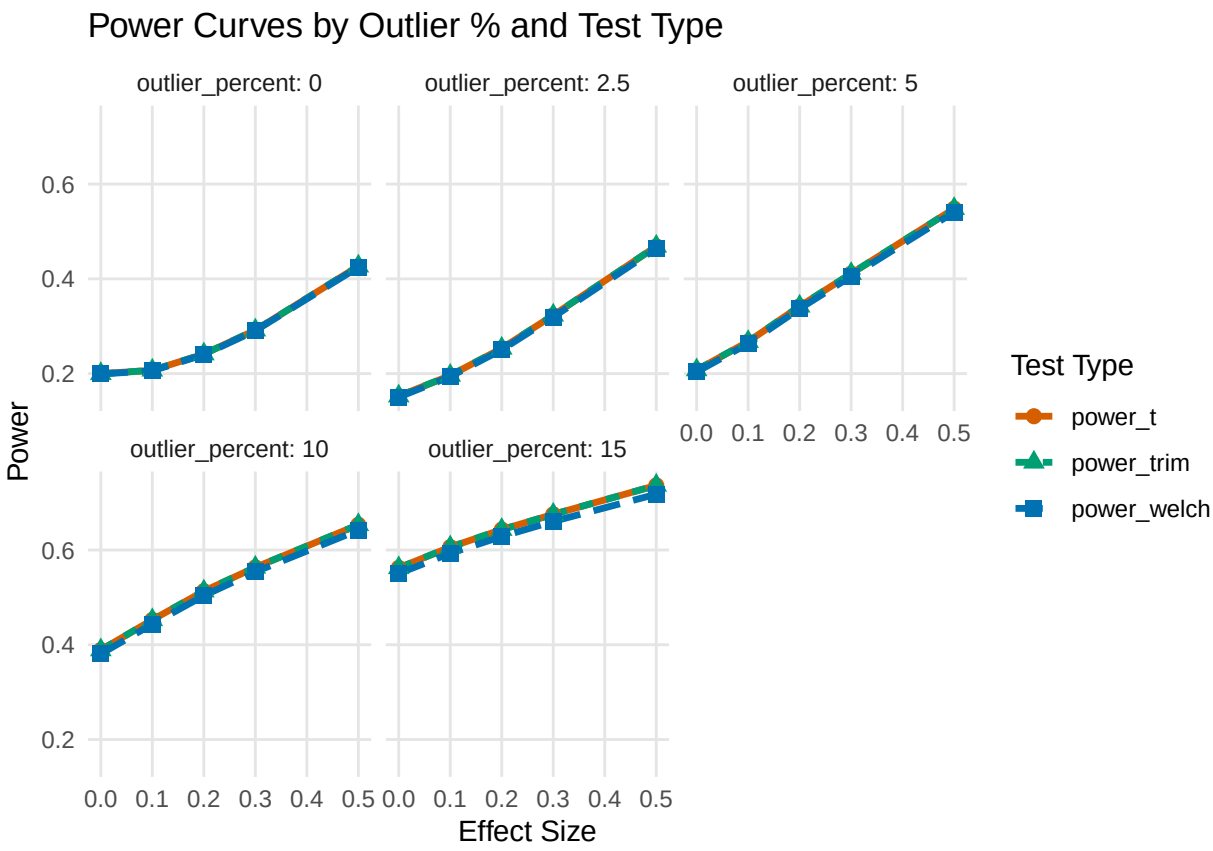
Power Curves by Effect Size and Test Type

```
# Plot power curves
power_curves_es_tt <- ggplot(results_long, aes(x = effect_size, y = power,
                        color = test_type, linetype = test_type, shape = test_type,
                        group = test_type)) +
  stat_summary(fun = mean, geom = "line", linewidth = 1.2) +
  stat_summary(fun = mean, geom = "point", size = 2.5) +
  scale_color_manual(values = c("power_t" = "#D55E00", "power_trim" = "#009E73", "power_welch" = "#0072
  labs(title = "Power Curves by Test Type",
       x = "Effect Size", y = "Average Power",
       color = "Test Type", linetype = "Test Type", shape = "Test Type") +
  theme_minimal() +
  theme(panel.grid.minor = element_blank(),
        panel.grid.major = element_line(color = "gray90"))
power_curves_es_tt
```



Facet by outlier percent

```
power_curves_outlier <- ggplot(results_long, aes(x = effect_size, y = power,
                         color = test_type, linetype = test_type, shape = test_type,
                         group = test_type)) +
  stat_summary(fun = mean, geom = "line", linewidth = 1.2) +
  stat_summary(fun = mean, geom = "point", size = 2.5) +
  facet_wrap(~ outlier_percent, labeller = label_both) +
  scale_color_manual(values = c("power_t" = "#D55E00", "power_trim" = "#009E73", "power_welch" = "#0072
  labs(title = "Power Curves by Outlier % and Test Type",
       x = "Effect Size", y = "Power",
       color = "Test Type", linetype = "Test Type", shape = "Test Type") +
  theme_minimal() +
  theme(panel.grid.minor = element_blank(),
        panel.grid.major = element_line(color = "gray90"))
power_curves_outlier
```



Takeaways

1.

The Standard t-test performs well under ideal conditions. When the assumptions of normality, equal variances, and balanced sample sizes are met, all three tests – the standard t-test, the Welch's t-test, and the trimmed t-test – all behave similarly, with power increasing as effect size grows

2.

The trimmed t-test has an increased power in the high outlier percentages of around 10-15%, meaning it outpeforms the other two tests, espeically with small to moderate sample sizes. It does this by trimming outliers and avoids skew by cause of extreme values.

3.)

The Welch's t-test helps with unequal variances. When the generate_data() function creates variable standard deviations and variances, the Welch's test accounts for this even if it is marginally more conservative than the other two tests, meaning it loses power.

4.)

Both the paired t-tests and boxplots show real but small differences where the trimmed t-test outperfomrs the standard t in high outlier conditions. The Welch's test performs worse than the standard t, but at the cost of being more conservative as expected. The differences are stronger with smaller sample sizes and higher effect sizes.

In the real world

You would want to use the standard t-test if you have clean data and equal variance. You would use the Welch's test with unequal group sizes or different variances. You would use the trimmed t-test if you have a known skewed distriubtion or have outliers.

All in all, the default test for A/B testing in production should be Welch, just to be safe, or Trimmed if you know that there will be a skewed distribution or have a lot of outliers.

Default to the Welch's test as your go-to, but consider using the trimmed t-test for user behavior data where there is often skew and outliers.

Saving of results

```
saveRDS(results_df, "results_df.rds")
saveRDS(diff_summary, "diff_summary.rds")
saveRDS(power_curves_es_tt, 'power_curves_es_tt.rds')
saveRDS(power_curves_outlier, 'power_curves_outlier.rds')
saveRDS(dist_differences, 'dist_differences.rds')
saveRDS(power_diff, 'power_diff.rds')
```