Appendix: Computing the null distribution of the Wilcoxon two-sample rank-sum statistic in R
by Mario Cortina Borja and Julian Stander

Original article: [http://www.significancemagazine.com/2590](http://www.significancemagazine.com/2590)

```r
y_1 <- c(68, 68, 59, 72, 64, 67, 70, 74)  # Original Data from W1945
y_2 <- c(60, 67, 61, 62, 67, 63, 56, 58)
n_1 <- length(y_1)
n_2 <- length(y_2)
y <- c(y_1, y_2)  # Pooled vector of observations
sample <- c(rep(1, n_1), rep(2, n_2))  # Sample identities
r <- rank(y)  # Ranks of combined data
sum_rank_1 <- sum(r[sample == 1])  # Sum of the ranks in the two samples
sum_rank_2 <- sum(r[sample == 2])

# **** One-tailed test looking at the alternative
# that the distribution of Sample 2 has a location shift
to the left of the distribution of Sample 1 *****
# Consider all the permutations - How many?
choose(n_1 + n_2, n_1) ### 12870 in total
# Let's consider them all, selecting out the members of the second group
all_comb <- combn(n_1 + n_2, n_2)
# Generates all combinations of n_1 + n_2 elements, taken n_2 at a time
dim(all_comb) # 8 rows by 12870 columns
# So each row is a possible combination for the second sample
# Function to sum the ranks
rank_sum <- function(sample_2_labels){  sum(r[sample_2_labels])}
# Check
T_stat_obs <- rank_sum((n_1+1):(n_1 + n_2))
# to the left of the distribution of Sample 1
T_stat_obs # 45, replicating W1945
# Apply this to all the combinations
T_stat <- apply(all_comb, 2, rank_sum)

# Display the results
# p-value
p_value <- length(T_stat[T_stat <= T_stat_obs]) / length(T_stat)
p_value ### gives p = 0.00676

require(ggplot2)
ggplot(data=T_stat_df, aes(T_stat)) +
  geom_histogram(breaks=seq(min(T_stat), max(T_stat), by=1),
                 fill=colours, colour =gray(0.5)) + theme_bw() +
  labs(x = 'T statistic', y = 'probability density') +
  geom_text(aes(x = T_stat_obs, y = 0.02,
                label=paste(c("T", "p"), c(T_stat_obs, round(p_value,4)),
                sep='=', collapse=' '),
                color='black')) + theme(legend.position="none") +
  geom_segment(aes(x = T_stat_obs, y = 0.018, xend = T_stat_obs, yend = 0.018), col='red')

require(coin)  # considers ties; function pvalue is in this library
w_exact <- wilcox_test(y ~ factor(sample), alternative = "greater",
                       distribution = "exact")  # Permutational
pvalue(w_exact) # 0.00676, same as with our approach above

set.seed(1111) # to be able to replicate following result
w_approximate <- wilcox_test(y ~ factor(sample), alternative = "greater",
                             distribution = "approximate")  # Monte Carlo
pvalue(w_approximate) # Gives 0.008, 99% CI = (0.0059, 0.0106)

w_asymptotic <- wilcox_test(y ~ factor(sample), alternative = "greater",
                           distribution = "asymptotic")  # Normal approximation
pvalue(w_asymptotic)  # 0.0077
```