Calibration Diagnostics

Alan Aw

4/19/2022

1 Introduction

We perform simple diagnostic tests of our MOCHIS software. These include

- checking that our implementation agrees with standard two-sample tests when applicable;
- checking that (our implementation in R of) the test controls Type I Error

```
## Setup
library(tidyverse) # for data analysis, plotting
library(grid) # for plotting
tab_mur_dir <- "~/Documents/research/spacing_stats/030222/"
diagnostic_dir <- "~/Documents/research/spacing_stats/032922"
source("~/Documents/research/spacing_stats/012522/main_draft0.R")
```

1.1 Version Log

- (April 19, 2022) We implement a resampling approach for the small n, small k and large n, small k scenario. We are working to fix all implementations relying on approximation theory, since they appear to suffer from catastrophic cancellation and precision arithmetic issues.

2 Agreement with Standard Tests

A special case of MOCHIS is the widely used two-sample test of stochastic dominance, known as the Mann-Whitney or Wilcoxon rank sum test (see explanation and example R code).

We compare our implementation of MOCHIS with `stats::wilcox.test`, which is the standard choice provided by base R. We construct our two samples of size n and k by drawing each sample i.i.d. from a standard Gaussian distribution before taking an absolute value (i.e., $\boldsymbol{z} = (|z_1|, \ldots, |z_k|)$ where $z_i \overset{\text{iid}}{\sim} N(0,1)$). We vary $(n,k) \in \{(50,10),(50,20),(50,50),(100,10),(100,20),(100,50),(500,10),(500,20),(500,50)\}$, effectively allowing our experiment to cover our implementation of both the resampling routine for small k and the Gaussian approximation for large k and n.

```
## Helper function for computing p-values
getPValues <- function(n, k, n_draws = 800) {
  # Create dataframe to return
  to_return <- data.frame(SEED = numeric(),
                          WILCOX = numeric(),
                          MOCHIS = numeric())
  # Generate n_draws p-values using both MOCHIS and stats::wilcox.test
  for (seed in 1:n_draws) {
    # Set seed
    set.seed(seed)
    # Generate null samples (X,Y)
    x0 <- abs(rnorm(k))
    y0 <- abs(rnorm(n))

    to_return <- rbind(to_return,
                       data.frame(SEED = seed,
                                  WILCOX = wilcox.test(x=x0,y=y0,alternative="two.sided")$p.value,
                                  MOCHIS = mochis.test(x = x0,
                                                       y = y0,
                                                       p = 1,
                                                       wList = k:0,
                                                       alternative = "two.sided",
                                                       approx = "resample",
                                                       n_mom = 100,
                                                       python_backend = FALSE)))
  }
  # Return
  return(to_return)
}

## Compute p-values
results_list <- list()
for (n in c(50,100,500)) {
  for (k in c(10,20,50)) {
    results_list[[paste0("n",n,"_k",k)]] <- getPValues(n = n, k = k)
  }
}

save(results_list, file = "mw_vs_mochis.RData")
```

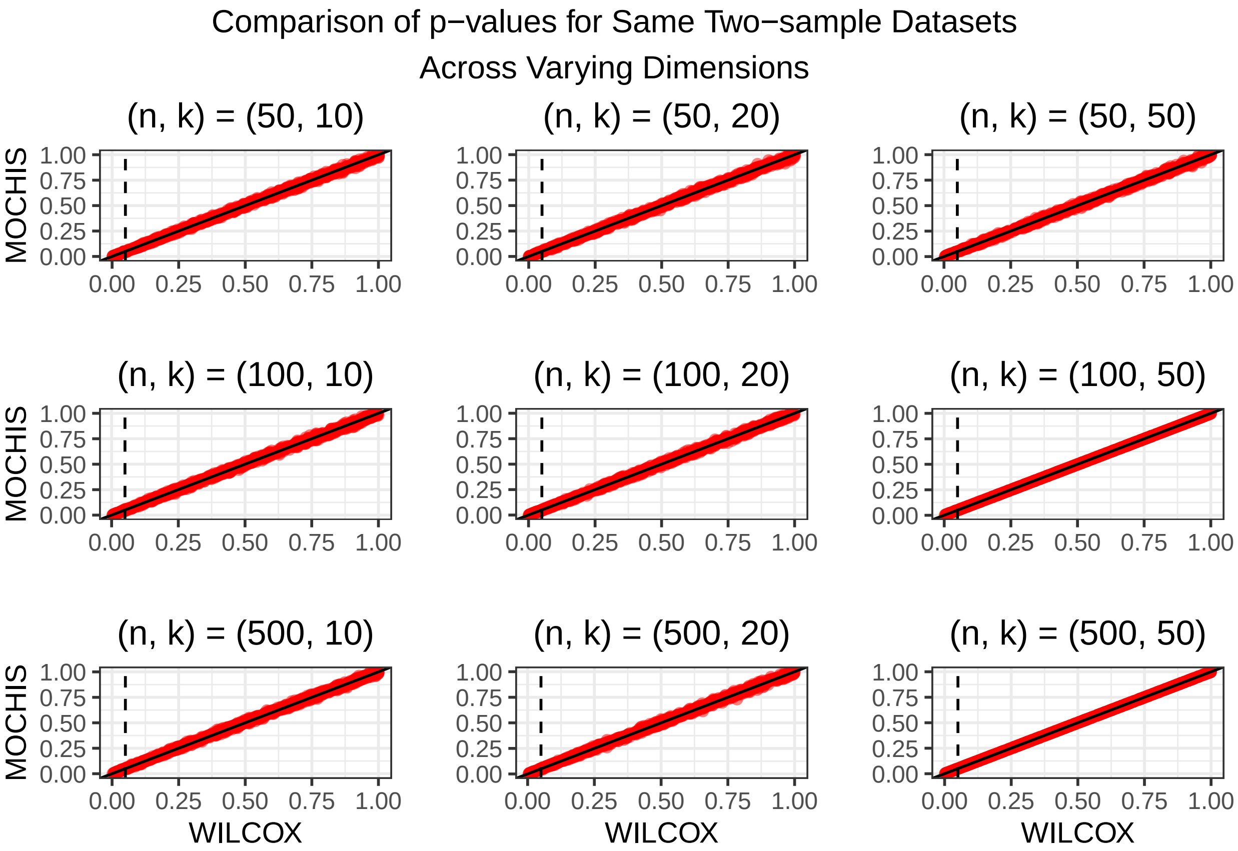
```
## Load data and generate plots
load("mw_vs_mochis.RData")

## Plot
my_plot_list <- list()
for (n in c(50,100,500)) {
  for (k in c(10,20,50)) {
    my_plot_list[[paste0("n",n,"_k",k)]]<- ggplot(results_list[[paste0("n",n,"_k",k)]],
                                                  aes(x =  WILCOX, y = MOCHIS)) +
      geom_point(alpha = 0.5, colour = "red") +
      geom_abline(slope = 1, intercept = 0) +
      geom_vline(xintercept = 0.05, lty = "dashed") +
      theme_bw() +
      ggtitle(paste0("(n, k) = (", n, ", ", k, ")")) +
      theme(plot.title = element_text(hjust = 0.5))

    # Compute squared difference
    rmse <- mean((results_list[[paste0("n",n,"_k",k)]]$WILCOX -
                    results_list[[paste0("n",n,"_k",k)]]$MOCHIS)^2)
    print(paste0("The root mean squared difference in p-values for (n,k) = (",
                 n, ",", k,") is ", rmse))
  }
}
```

```
## [1] "The root mean squared difference in p-values for (n,k) = (50,10) is 0.000126209661506754"
## [1] "The root mean squared difference in p-values for (n,k) = (50,20) is 0.000141583658527787"
## [1] "The root mean squared difference in p-values for (n,k) = (50,50) is 0.000140173572210388"
## [1] "The root mean squared difference in p-values for (n,k) = (100,10) is 0.000145810877445218"
## [1] "The root mean squared difference in p-values for (n,k) = (100,20) is 0.000124171987823092"
## [1] "The root mean squared difference in p-values for (n,k) = (100,50) is 1.51324251422152e-06"
## [1] "The root mean squared difference in p-values for (n,k) = (500,10) is 0.000136522264593647"
## [1] "The root mean squared difference in p-values for (n,k) = (500,20) is 0.000140424272969241"
## [1] "The root mean squared difference in p-values for (n,k) = (500,50) is 8.0643710741506e-08"
```

```
gridExtra::grid.arrange(my_plot_list[['n50_k10']] + xlab(""),
                        my_plot_list[['n50_k20']] + xlab("") + ylab(""),
                        my_plot_list[['n50_k50']] + xlab("") + ylab(""),
                        my_plot_list[['n100_k10']] + xlab(""),
                        my_plot_list[['n100_k20']] + xlab("") + ylab(""),
                        my_plot_list[['n100_k50']] + xlab("") + ylab(""),
                        my_plot_list[['n500_k10']],
                        my_plot_list[['n500_k20']] + ylab(""),
                        my_plot_list[['n500_k50']] + ylab(""),
                        top = textGrob("Comparison of p-values for Same Two-sample Datasets\nAcross Var
                        ncol = 3)
```



We see that the Gaussian approximation matches Mann-Whitney well whenever used appropriately (in cases where $n \geqslant 100$ and $k \geqslant 50$ such that $k/n > 10^{-3}$). The resampling implementation leads to p-values that differ from Mann-Whitney on the order of 10^{-4}. (Such an order of magnitude would matter only when performing multiple testing on the order of hundreds of tests. We will fix our implementation such that it defaults to using `stats::wilcox.test` whenever the choice of test statistic corresponds to the Mann-Whitney test.) On the whole, there is good agreement between our implementation and `stats::wilcox.test`.

3 Type I Error Control

Perhaps more important that good agreement is whether our implementation controls the Type I error rate, also known as false positive rate (FPR). Indeed, an implementation failing to control the FPR can lead to spurious discoveries that waste research time, especially when the results are piped into downstream analysis. Here again, we simulate samples (X, Y), with $X \in \mathbb{R}^k$ and $Y \in \mathbb{R}^n$ and $k \leqslant n$. We draw each element of X and Y from the same distribution (magnitudes of standard Gaussian) to match the null hypothesis. By varying the values of (k, n), we examine the control of Type I Error of MOCHIS.

```r
## Helper function for getting FPR metrics
## Helper function for getting FPR metrics
getFPR <- function(n, # length(y)
                   k, # length(x)
                   p, # choice of exponent
                   w_vec, # choice of weight vector
                   plot = FALSE, n_draws = 800) {
  # Enforce length(w_vec) = k+1
  assertthat::assert_that(length(w_vec) == k + 1, msg = "Length of w_vec must be (k+1).")

  # Generate n_draws p-values to compute FPP
  p_values_vec <- sapply(1:n_draws, function(seed) {
    # Set seed
    set.seed(seed)
    # Generate null samples (X,Y)
    x0 <- abs(rnorm(k))
    y0 <- abs(rnorm(n))

    # Compute p-value using mochis.test
    p_value <- mochis.test(x = x0,
                           y = y0,
                           p = p,
                           wList = w_vec,
                           alternative = "two.sided",
                           approx = "resample",
                           n_mom = 100,
                           python_backend = FALSE)
    if (p_value > 1 | p_value < 0) {
      print(paste0("Abnormal p-value detected for seed ", seed, "..."))
    }
    return(p_value)
  })

  # Compute variance of empirical distribution of p-values
  # Should be close to 1/12
  emp_var <- var(p_values_vec)
  emp_mean <- mean(p_values_vec)

  # Compute FPP vector (i.e., for each alpha, what's the FPP?)
  alpha_vec <- seq(0, 1, length.out = 201)
  fpp_vec <- c()
  for (alpha in alpha_vec) {
    fpp_vec <- c(fpp_vec, mean(p_values_vec < alpha))
  }
  fpp_df <- data.frame(ALPHA = alpha_vec, FPP = fpp_vec)
  if (plot) {
    message(date(), paste0(": Generating plot for (n,k,p) = (",
                           n,", ",k,", ",p,")"))
    plot(ggplot(fpp_df, aes(x = ALPHA, y = FPP)) +
           geom_line(colour = "blue") +
           geom_abline(slope = 1, intercept = 0) +
           theme_bw() +
           xlab(expression(paste("Sig. Threshold, ",alpha))) +
           ylab("False Positive Proportion") +
           ggtitle("Probability-Probability Plot") +
           theme(plot.title = element_text(hjust = 0.5,
                                           face = "bold")))
  }

  # Return fpp_df and emp_var
  return(list(EMP_MEAN = emp_mean,
              EMP_VAR = emp_var,
              FPP_DF = fpp_df,
              PVALUES = p_values_vec))
}
```

3.1 Small n and small k

Check for Mann-Whitney.

```
## Mann-Whitney
n_vec <- c(20,30,40)
k_vec <- c(10,20,40)
diagnostic_list <- list()
for (i in 1:length(k_vec)) {
  this_k_list <- list()
  k <- k_vec[i]
  wList <- k:0
  for (j in 1:length(n_vec)) {
    n <- n_vec[j]
    if (k > n) {
      message(date(), ": k exceeds n, skipping...")
    } else if (file.exists(paste0("n",n,"_k",k,"_mw_resample.RData"))) {
      message(date(), paste0(": RData already exists for (n,k) = (",
                             n,", ",k,"), skipping..."))
    } else {
      message(date(), paste0(": Generating Mann-Whitney results for (n,k) = (",
                             n,", ",k,")"))
      start_time <- Sys.time()
      mw_check <- getFPR(n=n, k=k, p=1, w_vec=wList, plot=FALSE)
      message(date(), paste0(": Empirical mean = ",
                   mw_check$EMP_MEAN,
                   ", Empirical variance = ",
                   mw_check$EMP_VAR))
      this_k_list[[j]] <- mw_check
      message(date(), paste0(": Saving data for (n,k) = (", n,", ",k,")"))
      save(mw_check, file = paste0("n",n,"_k",k,"_mw_resample.RData"))
      end_time <- Sys.time()
      print(end_time - start_time)
```

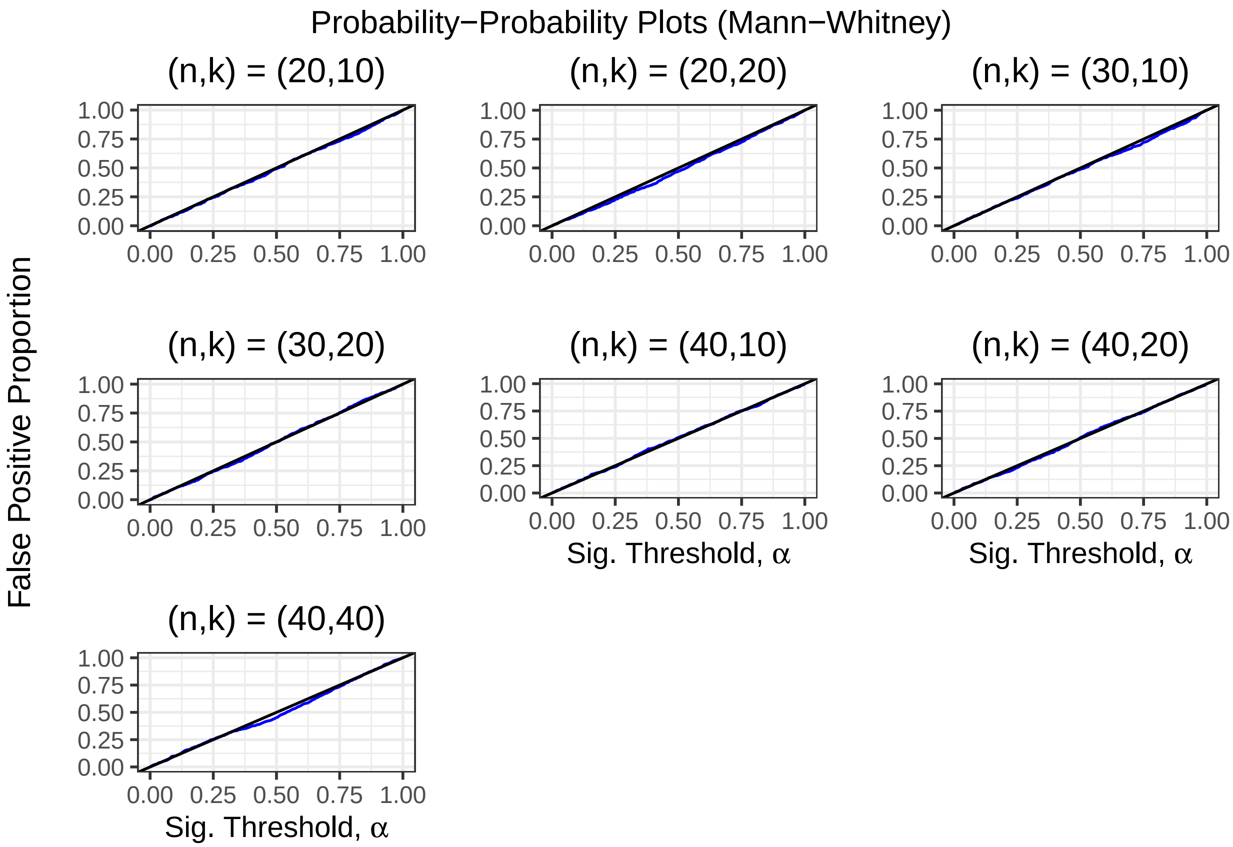
```r
    }
  }
  diagnostic_list[[i]] <- this_k_list
}

# Save
save(diagnostic_list, file = "small_k_small_n_mw_resample.RData")
```

```r
## Plot
my_plot_list <- list()
for (n in c(20,30,40)) {
  for (k in c(10,20,40)) {
    if (k > n) {
      message(date(), ": k exceeds n, skipping...")
    } else {
      load(paste0("results/resampling/n",n,"_k",k,"_mw_resample.RData"))

      my_plot_list[[paste0("n",n,"_k",k)]] <- ggplot(mw_check$FPP_DF,
                                                   aes(x =  ALPHA, y = FPP)) +
      geom_line(colour = "blue") +
      geom_abline(slope = 1, intercept = 0) +
      theme_bw() +
      xlab(expression(paste("Sig. Threshold, ",alpha))) +
      ylab("False Positive Proportion") +
      ggtitle(paste0("(n,k) = (", n, ",", k,")")) +
      theme(plot.title = element_text(hjust = 0.5))
    }
  }
}

gridExtra::grid.arrange(my_plot_list[['n20_k10']] + xlab("") + ylab(""),
                        my_plot_list[['n20_k20']] + xlab("") + ylab(""),
                        my_plot_list[['n30_k10']] + xlab("") + ylab(""),
                        my_plot_list[['n30_k20']] + xlab("") + ylab(""),
                        my_plot_list[['n40_k10']] + ylab(""),
                        my_plot_list[['n40_k20']] + ylab(""),
                        my_plot_list[['n40_k40']] + ylab(""),
                        left = textGrob("False Positive Proportion", rot = 90, vjust = 1),
                        top = textGrob("Probability-Probability Plots (Mann-Whitney)"),
                        nrow = 3)
```



Check for dispersion shift.

```
## Scale-shift alternative
n_vec <- c(10,20,40)
k_vec <- c(10,20,40)
diagnostic_list <- list()
for (i in 1:length(k_vec)) {
  this_k_list <- list()
  k <- k_vec[i]
  wList <- sapply(1:(k+1), function(x) {(x/(k+1)-0.5)^2})
  for (j in 1:length(n_vec)) {
    n <- n_vec[j]
    if (k > n) {
      message(date(), ": k exceeds n, skipping...")
    } else if (file.exists(paste0("n",n,"_k",k,"_quad_kernel_resample.RData"))) {
      message(date(), paste0(": RData already exists for (n,k) = (",
                             n,", ",k,"), skipping..."))
    } else {
      message(date(), paste0(": Generating dispersion shift results for (n,k) = (",
                             n,", ",k,")"))
      start_time <- Sys.time()
      mw_check <- getFPR(n=n, k=k, p=1, w_vec=wList, plot=FALSE)
      message(paste0("Empirical mean = ",
                     mw_check$EMP_MEAN,
                     ", Empirical variance = ",
                     mw_check$EMP_VAR))
      this_k_list[[j]] <- mw_check
```

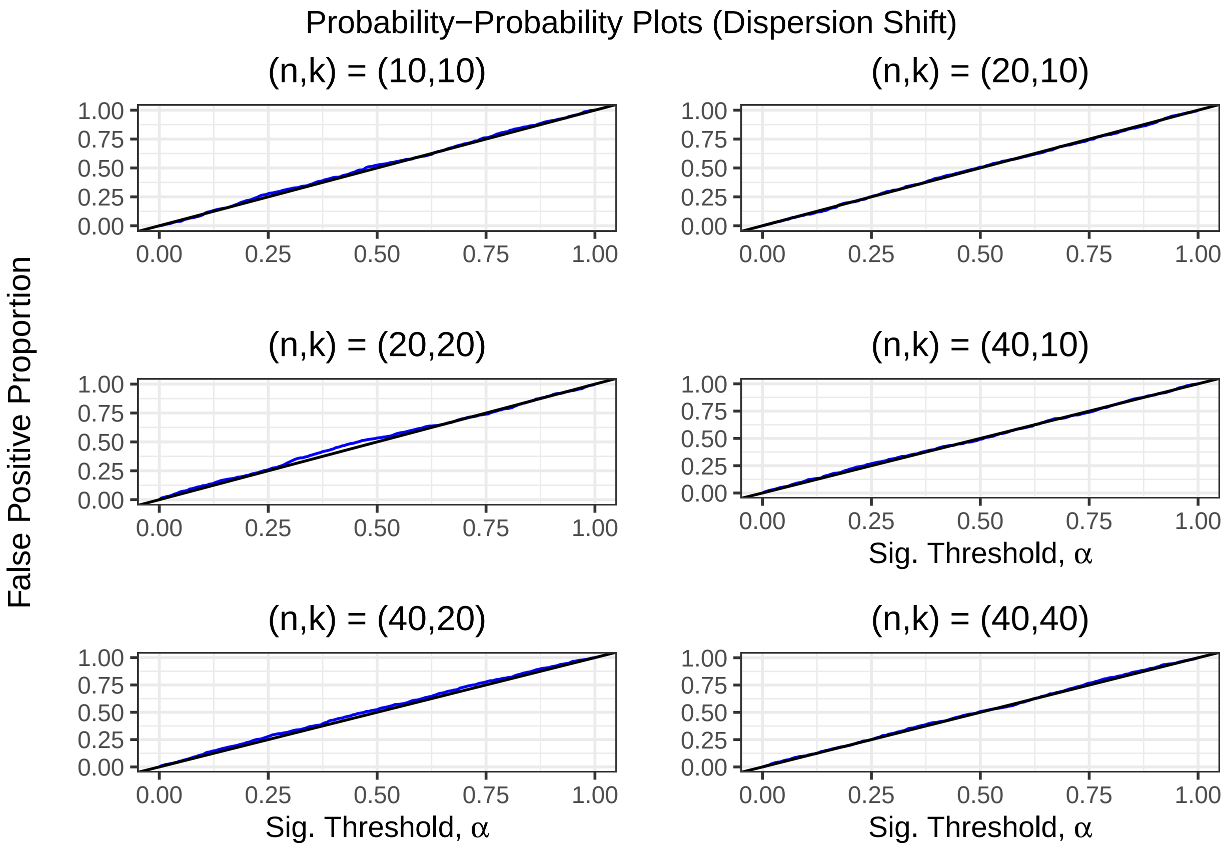
```r
      message(date(), paste0(": Saving data for (n,k) = (", n,", ",k,")"))
      save(mw_check, file = paste0("n",n,"_k",k,"_quad_kernel_resample.RData"))
      end_time <- Sys.time()
      print(end_time - start_time)
    }
  }
  diagnostic_list[[i]] <- this_k_list
}

# Save
save(diagnostic_list, file = "small_k_small_n_quad_kernel_resample.RData")
```

```r
## Plot
my_plot_list <- list()
for (n in c(10,20,40)) {
  for (k in c(10,20,40)) {
    if (k > n) {
      message(date(), ": k exceeds n, skipping...")
    } else {
      load(paste0("results/resampling/n",n,"_k",k,"_quad_kernel_resample.RData"))

      my_plot_list[[paste0("n",n,"_k",k)]] <- ggplot(mw_check$FPP_DF,
                                                   aes(x =  ALPHA, y = FPP)) +
      geom_line(colour = "blue") +
      geom_abline(slope = 1, intercept = 0) +
      theme_bw() +
      xlab(expression(paste("Sig. Threshold, ",alpha))) +
      ylab("False Positive Proportion") +
      ggtitle(paste0("(n,k) = (", n, ",", k,")")) +
      theme(plot.title = element_text(hjust = 0.5))
    }
  }
}

gridExtra::grid.arrange(my_plot_list[['n10_k10']] + xlab("") + ylab(""),
                        my_plot_list[['n20_k10']] + xlab("") + ylab(""),
                        my_plot_list[['n20_k20']] + xlab("") + ylab(""),
                        my_plot_list[['n40_k10']] + ylab(""),
                        my_plot_list[['n40_k20']] + ylab(""),
                        my_plot_list[['n40_k40']] + ylab(""),
                        left = textGrob("False Positive Proportion", rot = 90, vjust = 1),
                        top = textGrob("Probability-Probability Plots (Dispersion Shift)"),
                        nrow = 3)
```



On the whole, we see that our implementation achieves good FPR control for small values of n and k.

3.2 Large n and small k

Check for Mann-Whitney.

```
## Mann-Whitney
n_vec <- c(100,200,500)
k_vec <- c(10,20,40)
diagnostic_list <- list()
for (i in 1:length(k_vec)) {
  this_k_list <- list()
  k <- k_vec[i]
  wList <- k:0
  for (j in 1:length(n_vec)) {
    n <- n_vec[j]
    if (k > n) {
      message(date(), ": k exceeds n, skipping...")
    } else if (file.exists(paste0("n",n,"_k",k,"_mw_resample.RData"))) {
      message(date(), paste0(": RData already exists for (n,k) = (",
                             n,", ",k,"), skipping..."))
    } else {
      message(date(), paste0(": Generating Mann-Whitney results for (n,k) = (",
                             n,", ",k,")"))
      start_time <- Sys.time()
      mw_check <- getFPR(n=n, k=k, p=1, w_vec=wList, plot=FALSE)
      message(date(), paste0(": Empirical mean = ",
                    mw_check$EMP_MEAN,
                   ", Empirical variance = ",
                   mw_check$EMP_VAR))
      this_k_list[[j]] <- mw_check
      message(date(), paste0("Saving data for (n,k) = (", n,", ",k,")"))
      save(mw_check, file = paste0("n",n,"_k",k,"_mw_resample.RData"))
      end_time <- Sys.time()
      print(end_time - start_time)
    }
  }
  diagnostic_list[[i]] <- this_k_list
}

# Save
save(diagnostic_list, file = "small_k_large_n_mw_resample.RData")
```

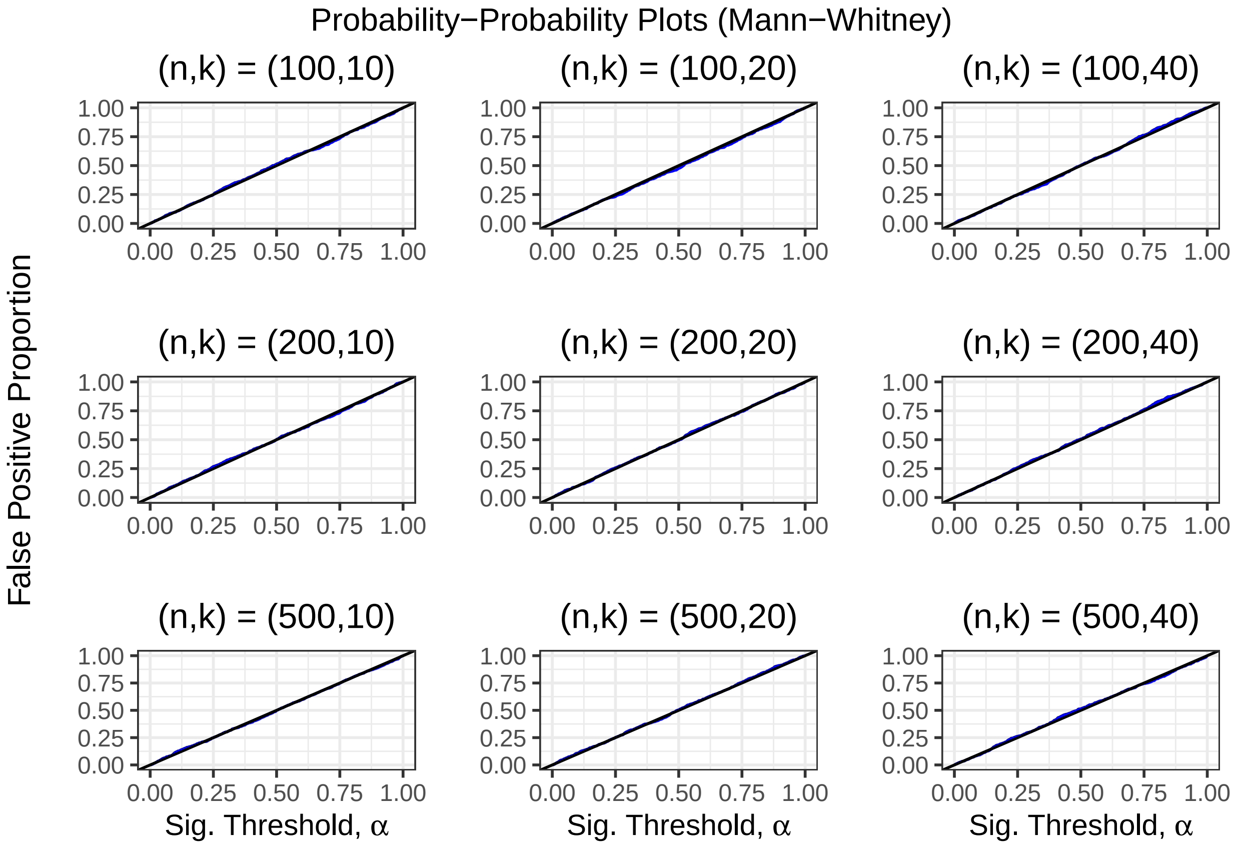
```
## Plot
my_plot_list <- list()
for (n in c(100,200,500)) {
  for (k in c(10,20,40)) {
    if (k > n) {
      message(date(), ": k exceeds n, skipping...")
    } else {
      load(paste0("results/resampling/n",n,"_k",k,"_mw_resample.RData"))

      my_plot_list[[paste0("n",n,"_k",k)]] <- ggplot(mw_check$FPP_DF,
                                                     aes(x =  ALPHA, y = FPP)) +
        geom_line(colour = "blue") +
        geom_abline(slope = 1, intercept = 0) +
        theme_bw() +
        xlab(expression(paste("Sig. Threshold, ",alpha))) +
        ylab("False Positive Proportion") +
        ggtitle(paste0("(n,k) = (", n, ",", k,")")) +
        theme(plot.title = element_text(hjust = 0.5))

      print(paste0("Empirical mean for (n,k) = (", n,", ",k,") is ", mw_check$EMP_MEAN))
      print(paste0("Empirical variance for (n,k) = (", n,", ",k,") is ", mw_check$EMP_VAR))
    }
  }
}
```

```
## [1] "Empirical mean for (n,k) = (100, 10) is 0.498811"
## [1] "Empirical variance for (n,k) = (100, 10) is 0.0861496937962453"
## [1] "Empirical mean for (n,k) = (100, 20) is 0.5092565"
## [1] "Empirical variance for (n,k) = (100, 20) is 0.0848396453644556"
## [1] "Empirical mean for (n,k) = (100, 40) is 0.4985465"
## [1] "Empirical variance for (n,k) = (100, 40) is 0.0797663839176471"
## [1] "Empirical mean for (n,k) = (200, 10) is 0.498783"
## [1] "Empirical variance for (n,k) = (200, 10) is 0.0863440421637046"
## [1] "Empirical mean for (n,k) = (200, 20) is 0.4981485"
## [1] "Empirical variance for (n,k) = (200, 20) is 0.0840608550165206"
## [1] "Empirical mean for (n,k) = (200, 40) is 0.492862"
## [1] "Empirical variance for (n,k) = (200, 40) is 0.081691140481602"
## [1] "Empirical mean for (n,k) = (500, 10) is 0.500935"
## [1] "Empirical variance for (n,k) = (500, 10) is 0.0854552688610764"
## [1] "Empirical mean for (n,k) = (500, 20) is 0.4952685"
## [1] "Empirical variance for (n,k) = (500, 20) is 0.0826386077674593"
## [1] "Empirical mean for (n,k) = (500, 40) is 0.497454"
## [1] "Empirical variance for (n,k) = (500, 40) is 0.086025503363204"
```

```
gridExtra::grid.arrange(my_plot_list[['n100_k10']] + xlab("") + ylab(""),
                        my_plot_list[['n100_k20']] + xlab("") + ylab(""),
                        my_plot_list[['n100_k40']] + xlab("") + ylab(""),
                        my_plot_list[['n200_k10']] + xlab("") + ylab(""),
                        my_plot_list[['n200_k20']] + xlab("") + ylab(""),
                        my_plot_list[['n200_k40']] + xlab("") + ylab(""),
                        my_plot_list[['n500_k10']] + ylab(""),
                        my_plot_list[['n500_k20']] + ylab(""),
                        my_plot_list[['n500_k40']] + ylab(""),
                        left = textGrob("False Positive Proportion", rot = 90, vjust = 1),
                        top = textGrob("Probability-Probability Plots (Mann-Whitney)"),
                        nrow = 3)
```



Check for dispersion shift.

```
## Scale-shift alternative
n_vec <- c(100,200,500)
k_vec <- c(10,20,40)
diagnostic_list <- list()
for (i in 1:length(k_vec)) {
  this_k_list <- list()
  k <- k_vec[i]
  wList <- sapply(1:(k+1), function(x) {(x/(k+1)-0.5)^2})
```

```r
  for (j in 1:length(n_vec)) {
    n <- n_vec[j]
    if (k > n) {
      message(date(), ": k exceeds n, skipping...")
    } else if (file.exists(paste0("n",n,"_k",k,"_quad_kernel_resample.RData"))) {
      message(date(), paste0(": RData already exists for (n,k) = (",
                             n,", ",k,"), skipping..."))
    } else {
      message(date(), paste0(": Generating dispersion shift results for (n,k) = (",
                             n,", ",k,")"))
      start_time <- Sys.time()
      mw_check <- getFPR(n=n, k=k, p=1, w_vec=wList, plot=FALSE)
      message(date(), paste0(": Empirical mean = ",
                   mw_check$EMP_MEAN,
                   ", Empirical variance = ",
                   mw_check$EMP_VAR))
      this_k_list[[j]] <- mw_check
      message(date(), paste0("Saving data for (n,k) = (", n,", ",k,")"))
      save(mw_check, file = paste0("n",n,"_k",k,"_quad_kernel_resample.RData"))
      end_time <- Sys.time()
      print(end_time - start_time)
    }
  }
  diagnostic_list[[i]] <- this_k_list
}

# Save
save(diagnostic_list, file = "small_k_large_n_quad_kernel_resample.RData")
```

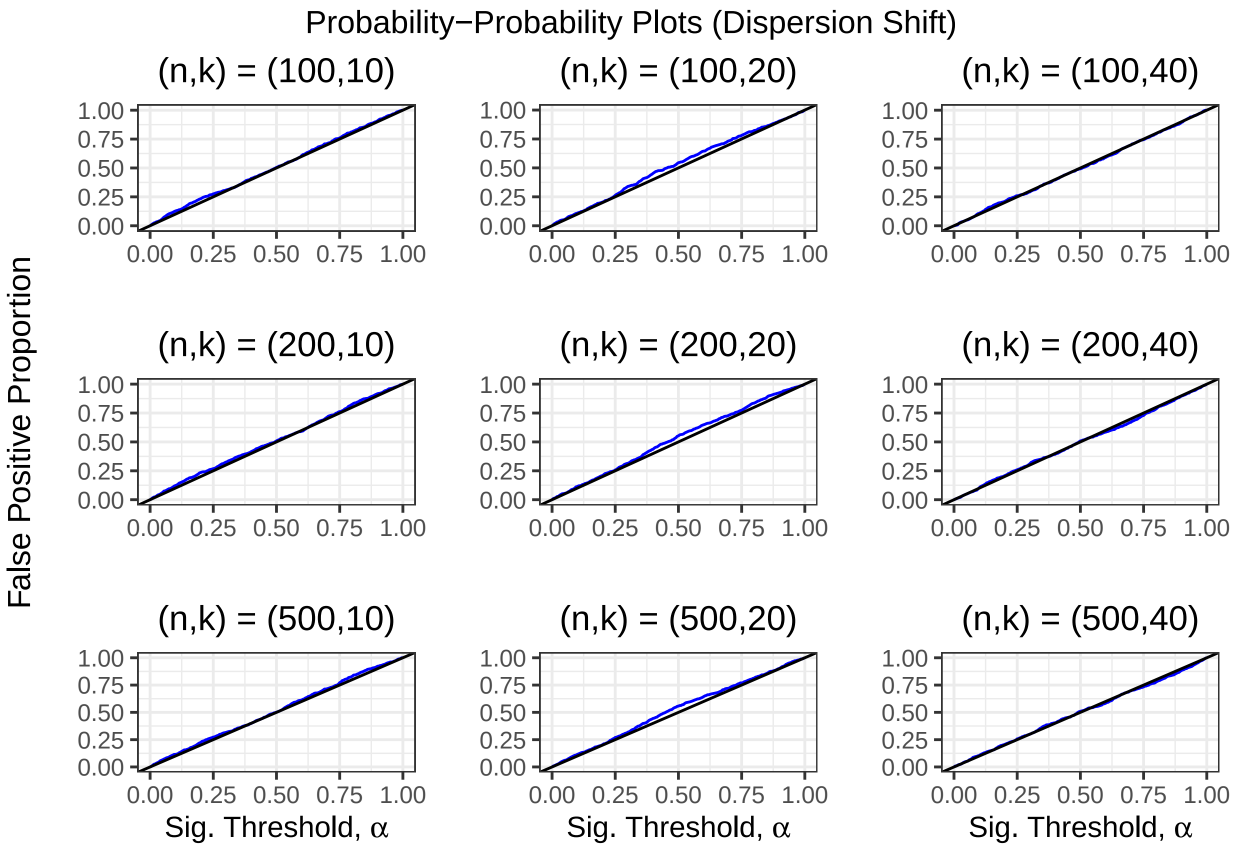
```r
## Plot
my_plot_list <- list()
for (n in c(100,200,500)) {
  for (k in c(10,20,40)) {
    if (k > n) {
      message(date(), ": k exceeds n, skipping...")
    } else {
      load(paste0("results/resampling/n",n,"_k",k,"_quad_kernel_resample.RData"))

      my_plot_list[[paste0("n",n,"_k",k)]] <- ggplot(mw_check$FPP_DF,
                                                     aes(x =  ALPHA, y = FPP)) +
        geom_line(colour = "blue") +
        geom_abline(slope = 1, intercept = 0) +
        theme_bw() +
        xlab(expression(paste("Sig. Threshold, ",alpha))) +
        ylab("False Positive Proportion") +
        ggtitle(paste0("(n,k) = (", n, ",", k,")")) +
        theme(plot.title = element_text(hjust = 0.5))

      print(paste0("Empirical mean for (n,k) = (", n,", ",k,") is ", mw_check$EMP_MEAN))
      print(paste0("Empirical variance for (n,k) = (", n,", ",k,") is ", mw_check$EMP_VAR))
    }
  }
}
```

```
## [1] "Empirical mean for (n,k) = (100, 10) is 0.487306"
## [1] "Empirical variance for (n,k) = (100, 10) is 0.0854874068225282"
## [1] "Empirical mean for (n,k) = (100, 20) is 0.4754705"
## [1] "Empirical variance for (n,k) = (100, 20) is 0.0825365889659574"
## [1] "Empirical mean for (n,k) = (100, 40) is 0.500257"
## [1] "Empirical variance for (n,k) = (100, 40) is 0.0854332859584481"
## [1] "Empirical mean for (n,k) = (200, 10) is 0.4843105"
## [1] "Empirical variance for (n,k) = (200, 10) is 0.0852224037944931"
## [1] "Empirical mean for (n,k) = (200, 20) is 0.4732225"
## [1] "Empirical variance for (n,k) = (200, 20) is 0.0790324058010013"
## [1] "Empirical mean for (n,k) = (200, 40) is 0.5036765"
## [1] "Empirical variance for (n,k) = (200, 40) is 0.0875161411241552"
## [1] "Empirical mean for (n,k) = (500, 10) is 0.4844635"
## [1] "Empirical variance for (n,k) = (500, 10) is 0.0822630345609512"
## [1] "Empirical mean for (n,k) = (500, 20) is 0.4760695"
## [1] "Empirical variance for (n,k) = (500, 20) is 0.0826141959146433"
## [1] "Empirical mean for (n,k) = (500, 40) is 0.502725"
## [1] "Empirical variance for (n,k) = (500, 40) is 0.0882884841802253"
```

```
gridExtra::grid.arrange(my_plot_list[['n100_k10']] + xlab("") + ylab(""),
                        my_plot_list[['n100_k20']] + xlab("") + ylab(""),
                        my_plot_list[['n100_k40']] + xlab("") + ylab(""),
                        my_plot_list[['n200_k10']] + xlab("") + ylab(""),
                        my_plot_list[['n200_k20']] + xlab("") + ylab(""),
                        my_plot_list[['n200_k40']] + xlab("") + ylab(""),
                        my_plot_list[['n500_k10']] + ylab(""),
                        my_plot_list[['n500_k20']] + ylab(""),
                        my_plot_list[['n500_k40']] + ylab(""),
                        left = textGrob("False Positive Proportion", rot = 90, vjust = 1),
                        top = textGrob("Probability-Probability Plots (Dispersion Shift)"),
                        nrow = 3)
```



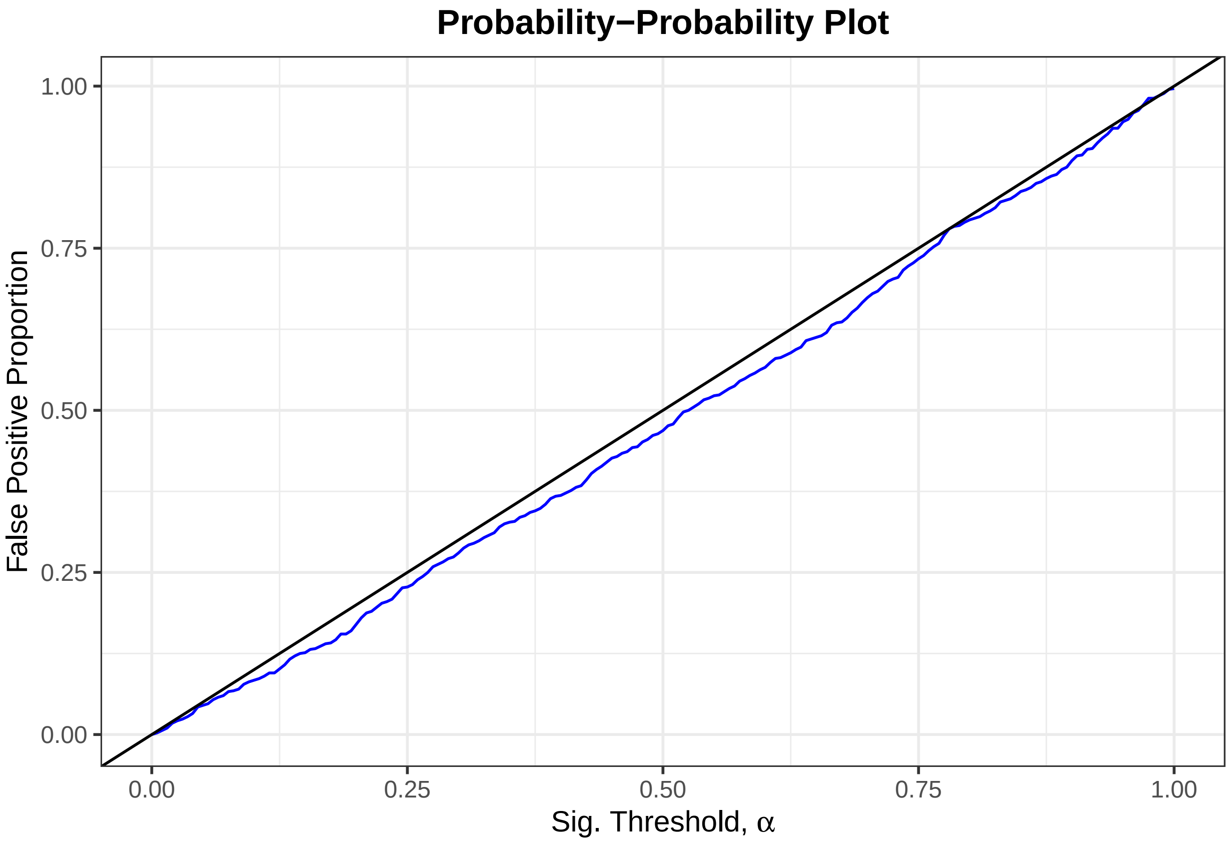
On the whole, we see that our implementation achieves good FPR control for large values of n and small values of k.

3.3 Large n and large k

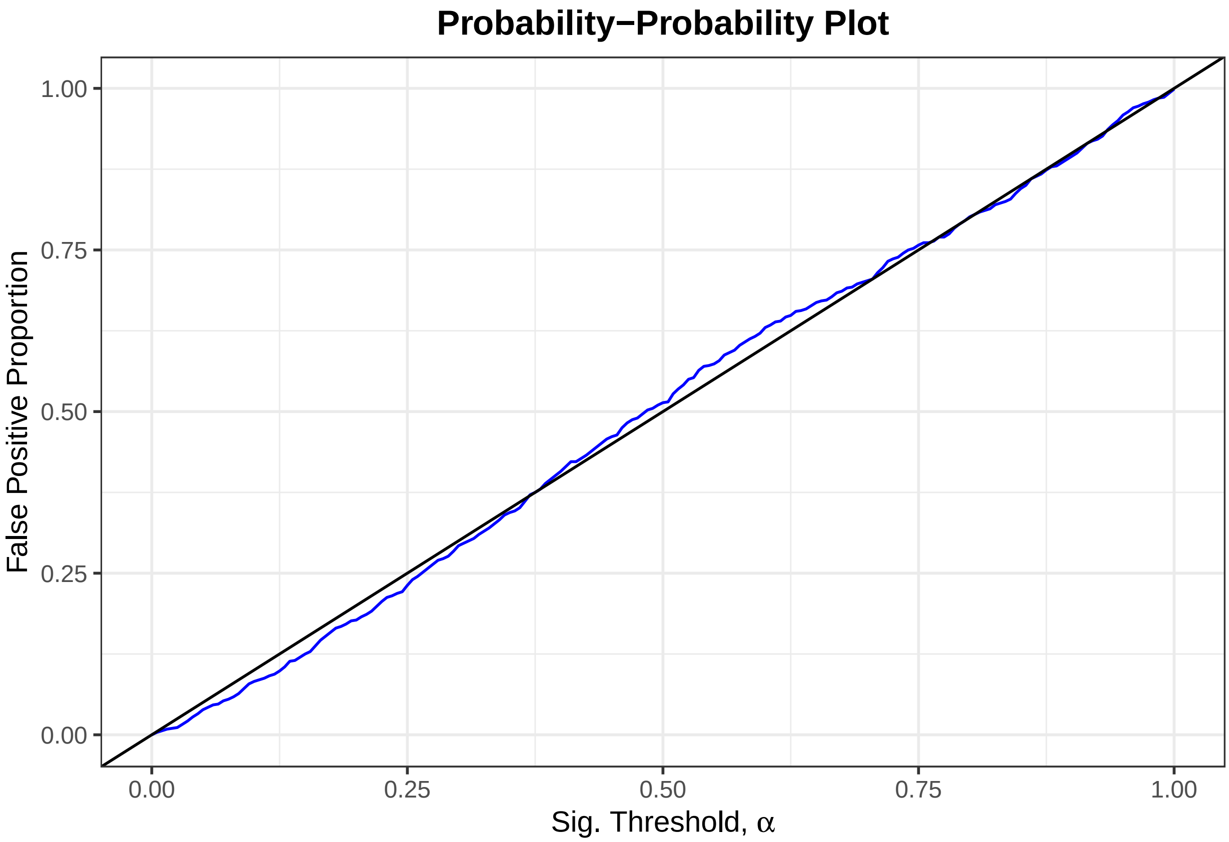
Check for Mann-Whitney.

```
## Mann-Whitney
n_vec <- c(100,200,500)
k_vec <- c(50,100,200)
diagnostic_list <- list()
for (i in 1:length(k_vec)) {
  this_k_list <- list()
  k <- k_vec[i]
  wList <- k:0
  for (j in 1:length(n_vec)) {
    n <- n_vec[j]
    if (k > n) {
      message("k exceeds n, skipping...")
    } else {
      print(paste0("Generating Mann-Whitney plot for (n,k) = (", n,", ",k,")"))
      mw_check <- getFPR(n=n, k=k, p=1, w_vec=wList, plot=TRUE)
      print(paste0("Empirical mean = ",
                   mw_check$EMP_MEAN,
                   ", Empirical variance = ",
                   mw_check$EMP_VAR))
      this_k_list[[j]] <- mw_check
    }
  }
  diagnostic_list[[i]] <- this_k_list
}
```

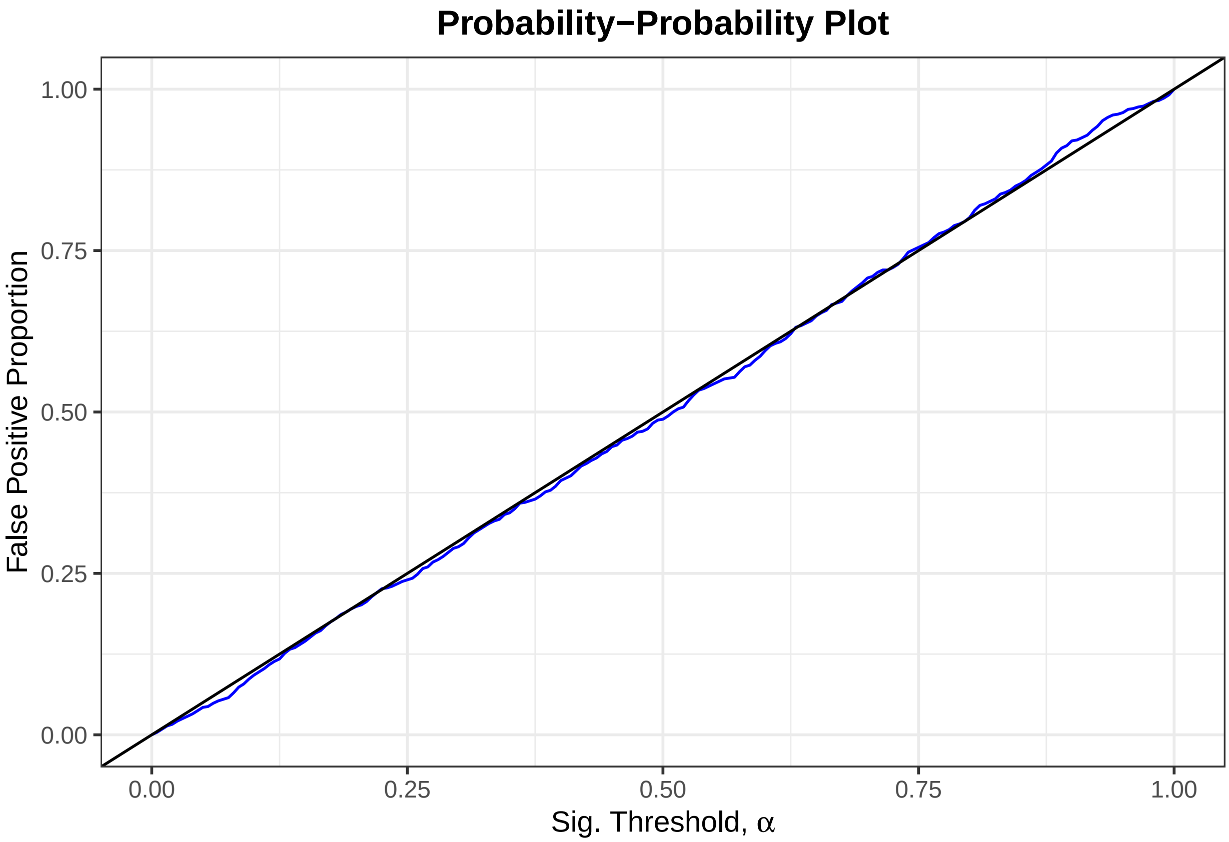
```
## [1] "Generating Mann-Whitney plot for (n,k) = (100, 50)"
```



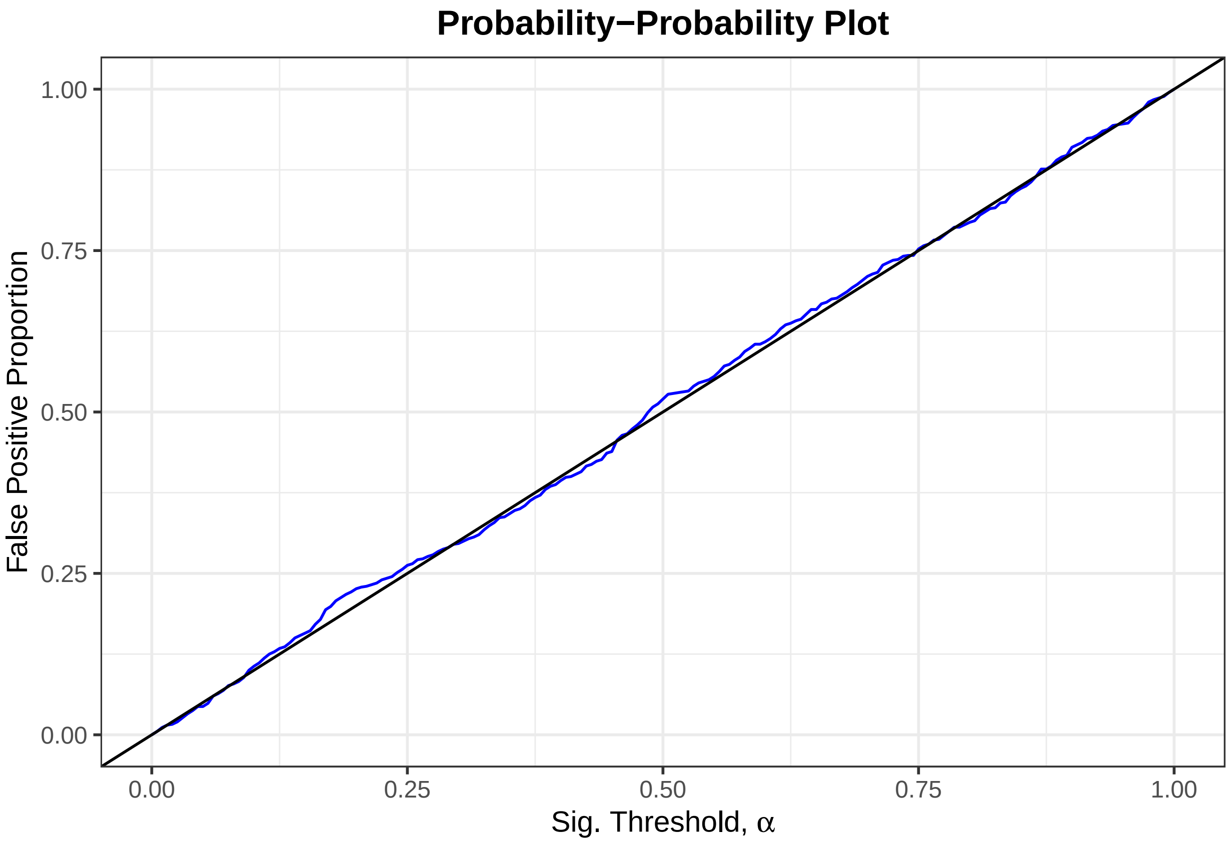
```
## [1] "Empirical mean = 0.520694828605934, Empirical variance = 0.0819835992128552"
## [1] "Generating Mann-Whitney plot for (n,k) = (200, 50)"
```



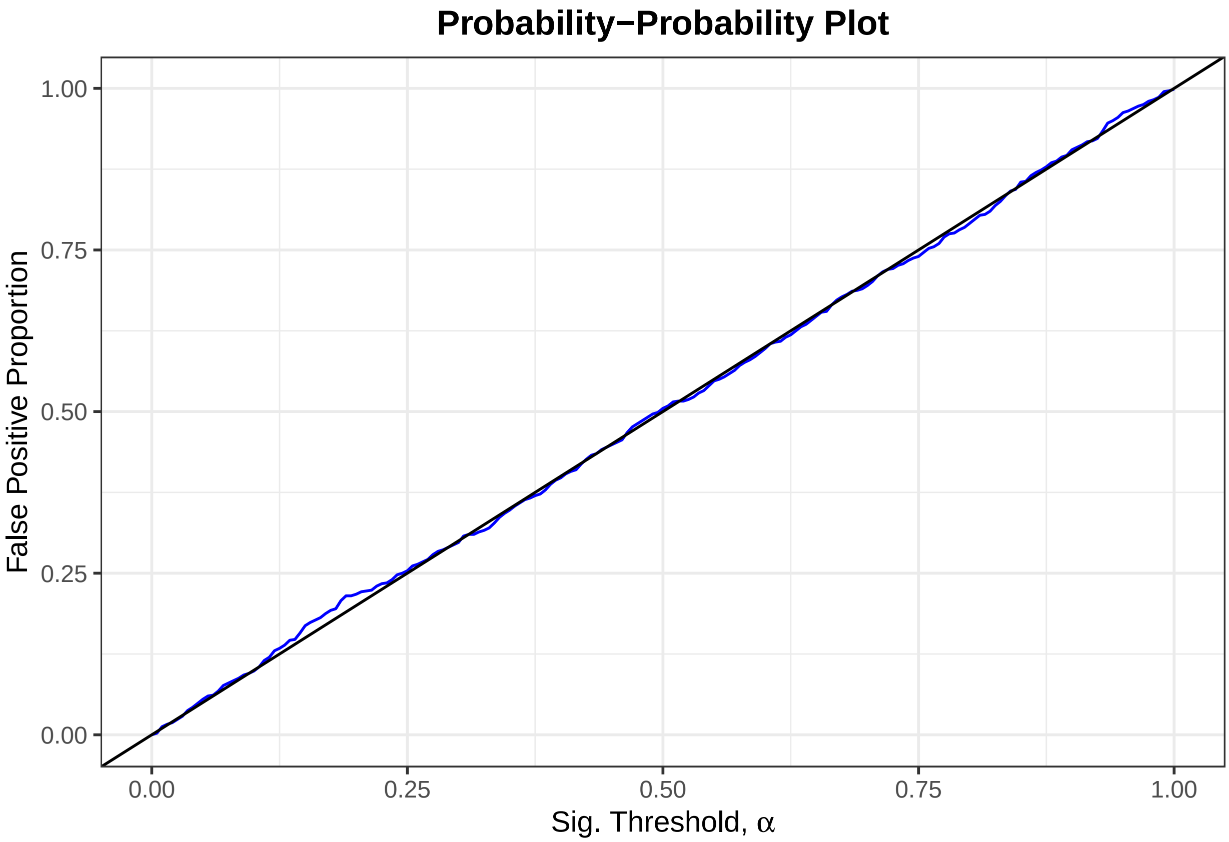
```
## [1] "Empirical mean = 0.500173630340617, Empirical variance = 0.0790514698388856"
## [1] "Generating Mann-Whitney plot for (n,k) = (500, 50)"
```



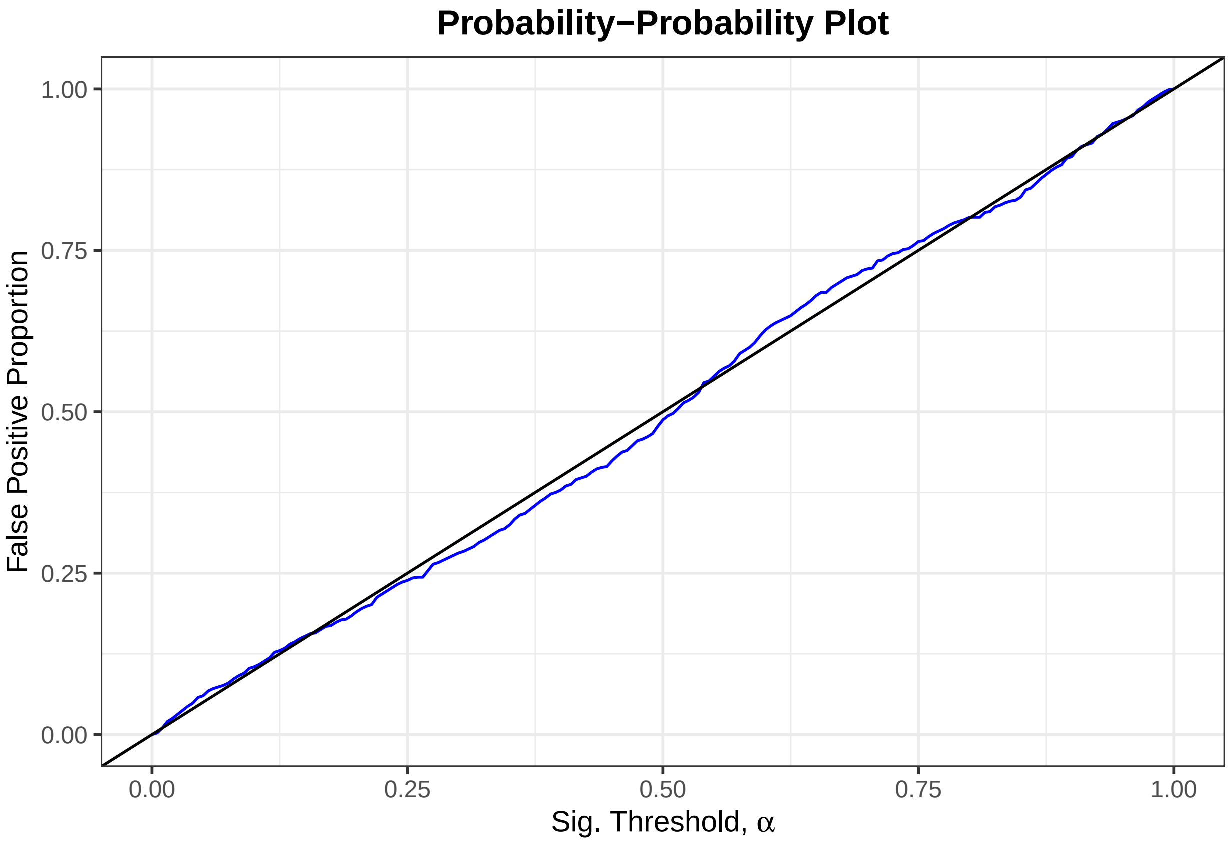
```
## [1] "Empirical mean = 0.501801679329254, Empirical variance = 0.0803828117920205"
## [1] "Generating Mann-Whitney plot for (n,k) = (100, 100)"
```



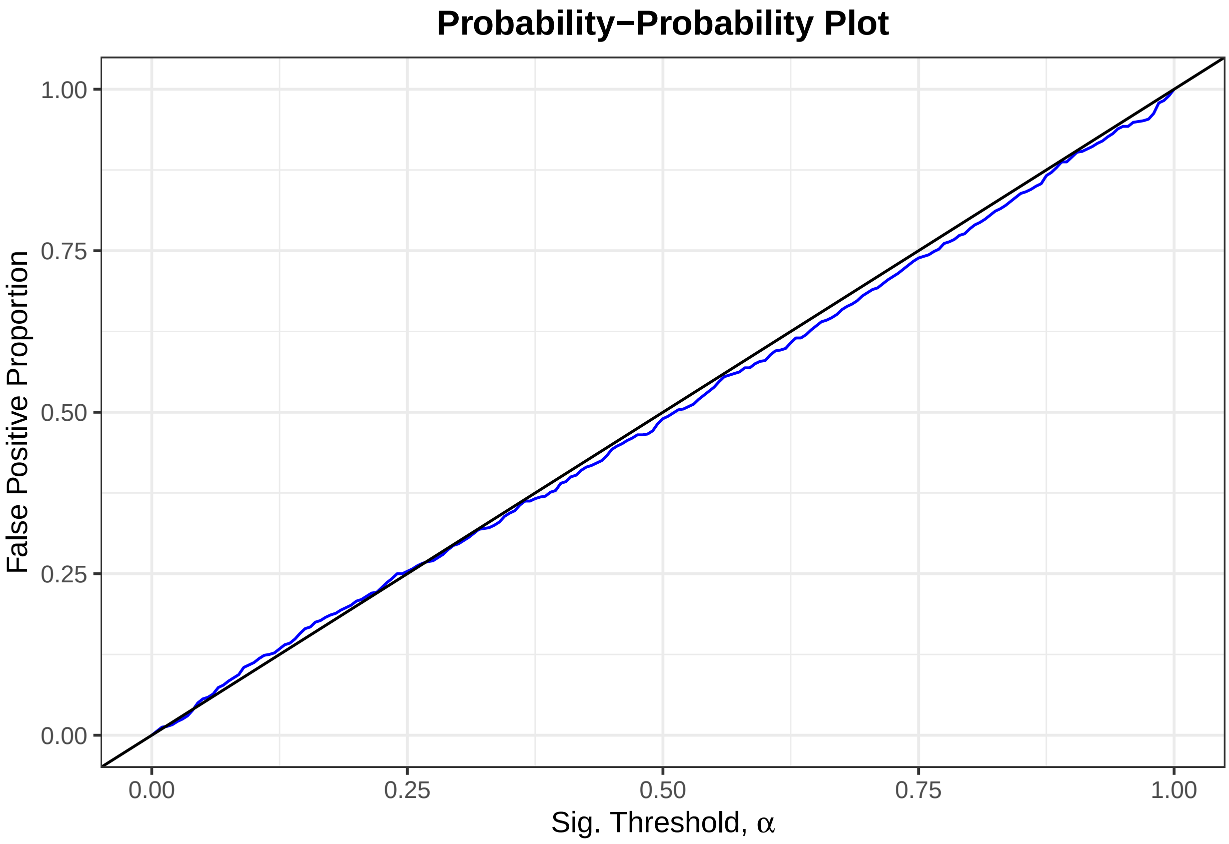
```
## [1] "Empirical mean = 0.496124939788777, Empirical variance = 0.0839873348232607"
## [1] "Generating Mann-Whitney plot for (n,k) = (200, 100)"
```



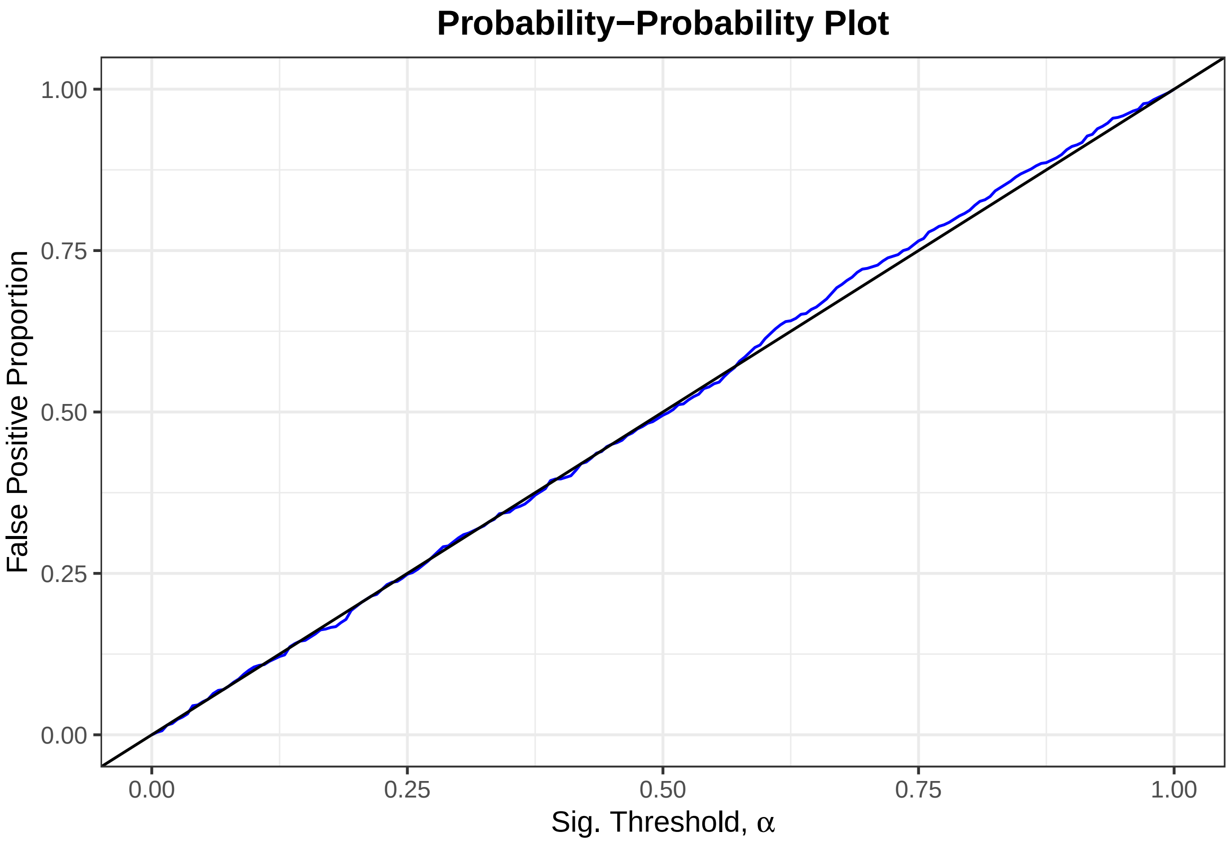
```
## [1] "Empirical mean = 0.499008305052796, Empirical variance = 0.0846430219562476"
## [1] "Generating Mann-Whitney plot for (n,k) = (500, 100)"
```



```
## [1] "Empirical mean = 0.501507927716777, Empirical variance = 0.0814808748971005"
## [1] "Generating Mann-Whitney plot for (n,k) = (200, 200)"
```



```
## [1] "Empirical mean = 0.507080631315207, Empirical variance = 0.0874017897698296"
## [1] "Generating Mann-Whitney plot for (n,k) = (500, 200)"
```

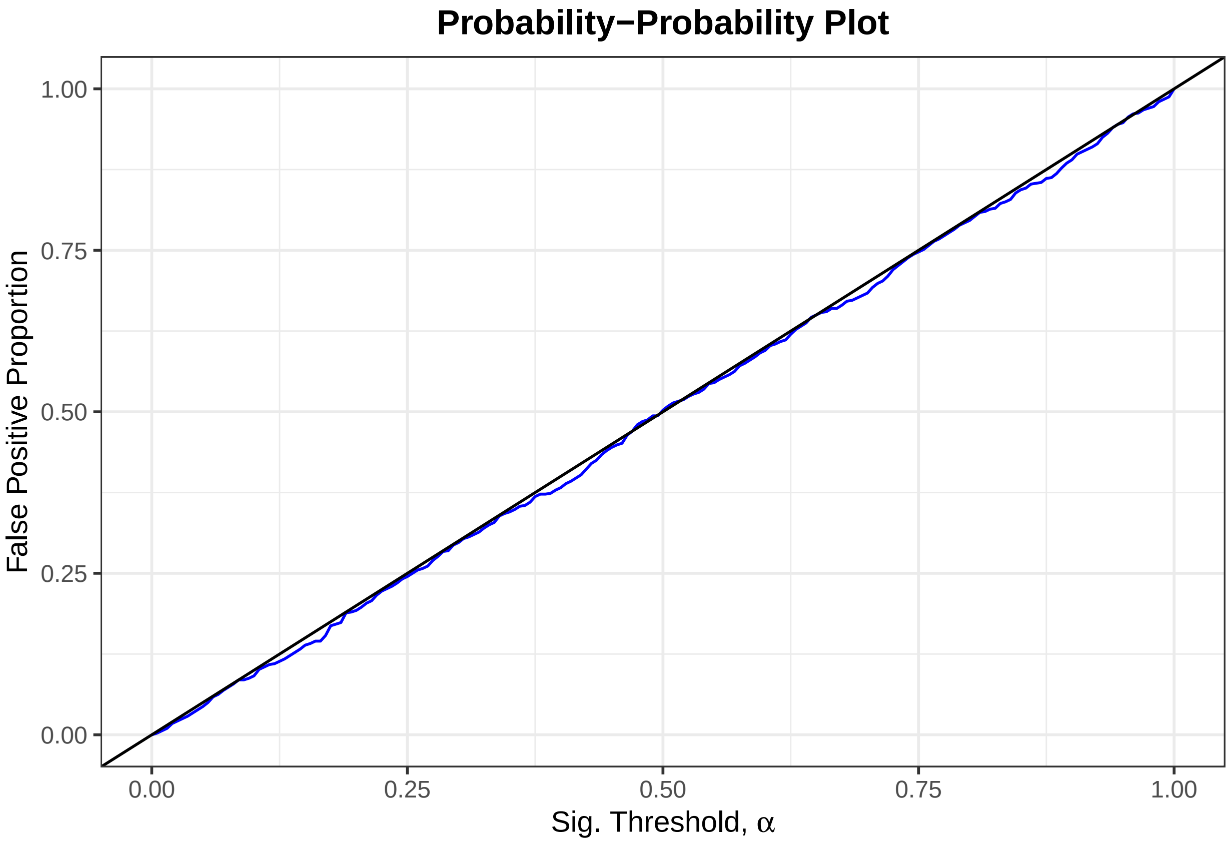


```
## [1] "Empirical mean = 0.495159737266385, Empirical variance = 0.0801284744212768"
```

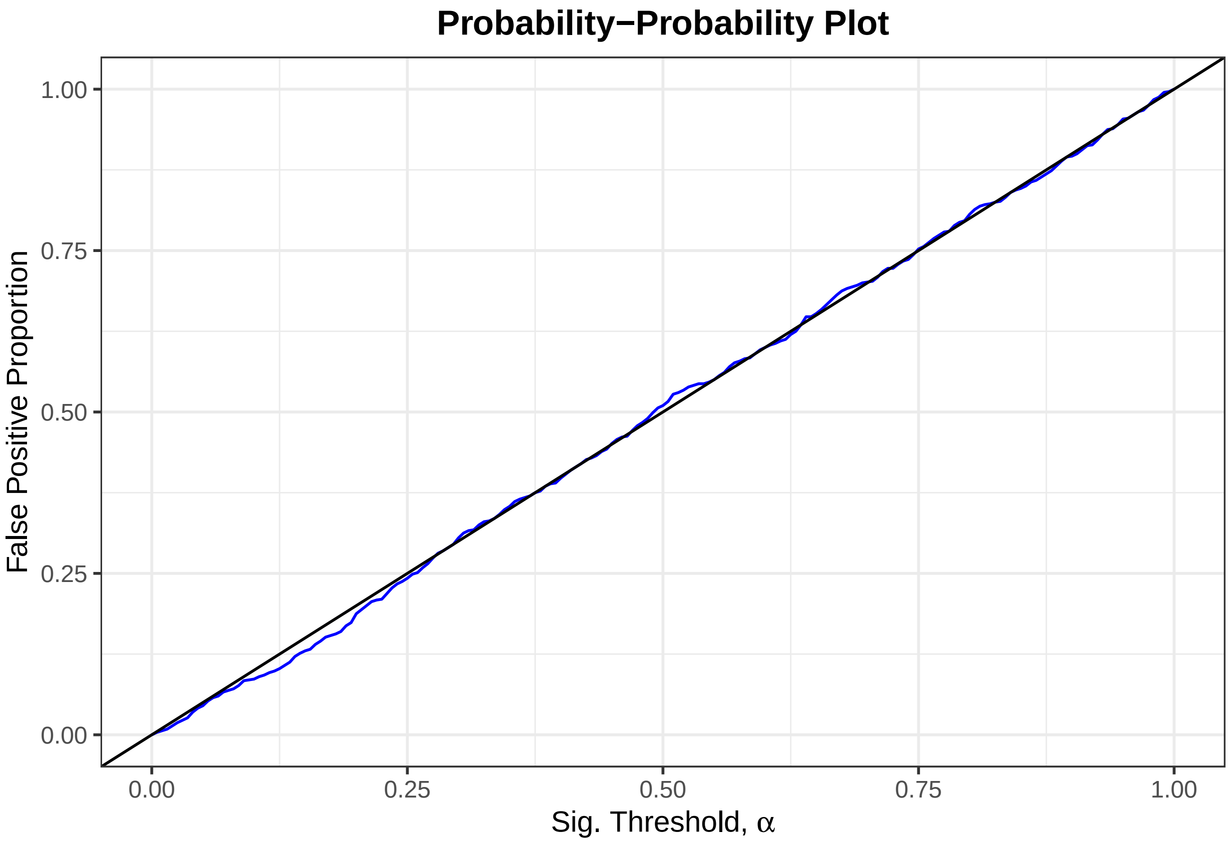
Check for dispersion shift.

```
## Scale-shift alternative
n_vec <- c(100,200,500)
k_vec <- c(50,100,200)
diagnostic_list <- list()
for (i in 1:length(k_vec)) {
  this_k_list <- list()
  k <- k_vec[i]
  wList <- sapply(1:(k+1), function(x) {(x/(k+1)-0.5)^2})
  for (j in 1:length(n_vec)) {
    n <- n_vec[j]
    if (k > n) {
      message("k exceeds n, skipping...")
    } else {
      print(paste0("Generating Mann-Whitney plot for (n,k) = (", n,", ",k,")"))
      mw_check <- getFPR(n=n, k=k, p=1, w_vec=wList, plot=TRUE)
      print(paste0("Empirical mean = ",
                   mw_check$EMP_MEAN,
                   ", Empirical variance = ",
                   mw_check$EMP_VAR))
      this_k_list[[j]] <- mw_check
    }
  }
  diagnostic_list[[i]] <- this_k_list
}
```

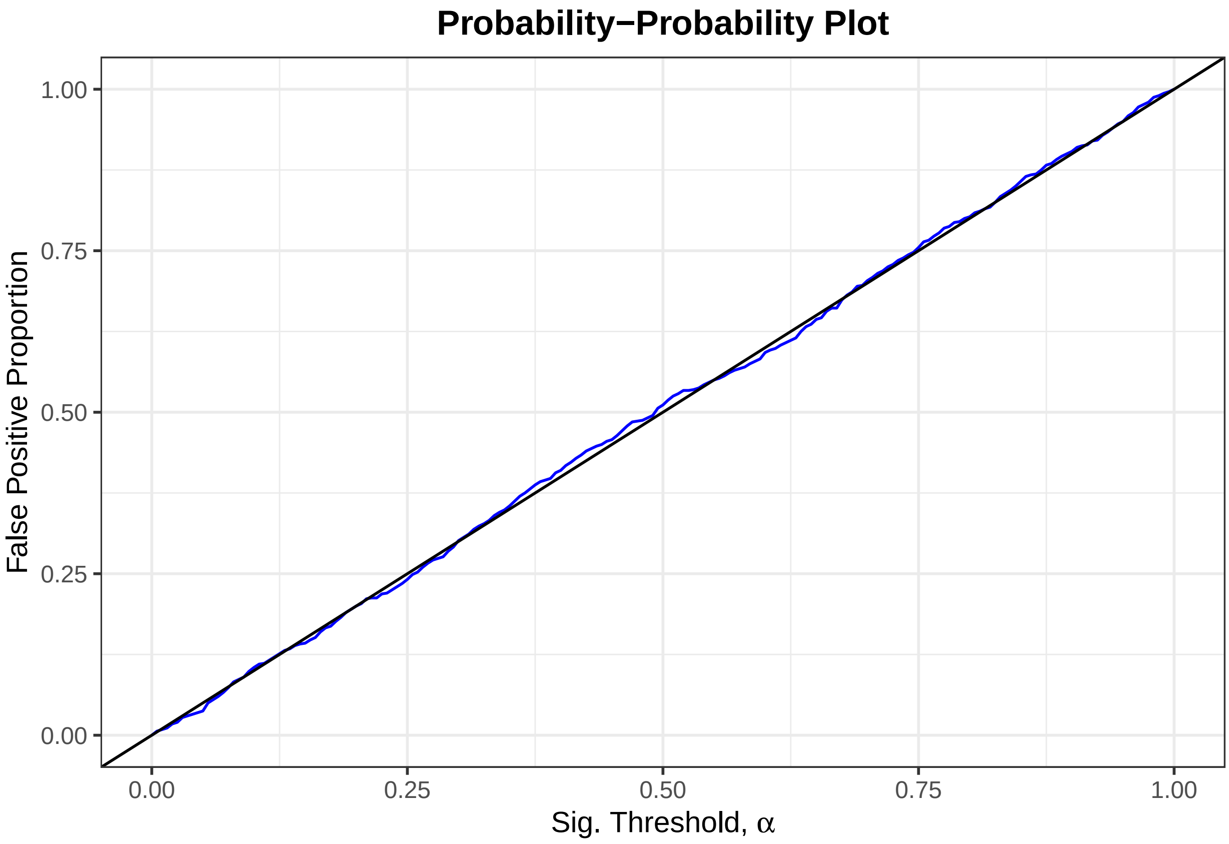
```
## [1] "Generating Mann-Whitney plot for (n,k) = (100, 50)"
```



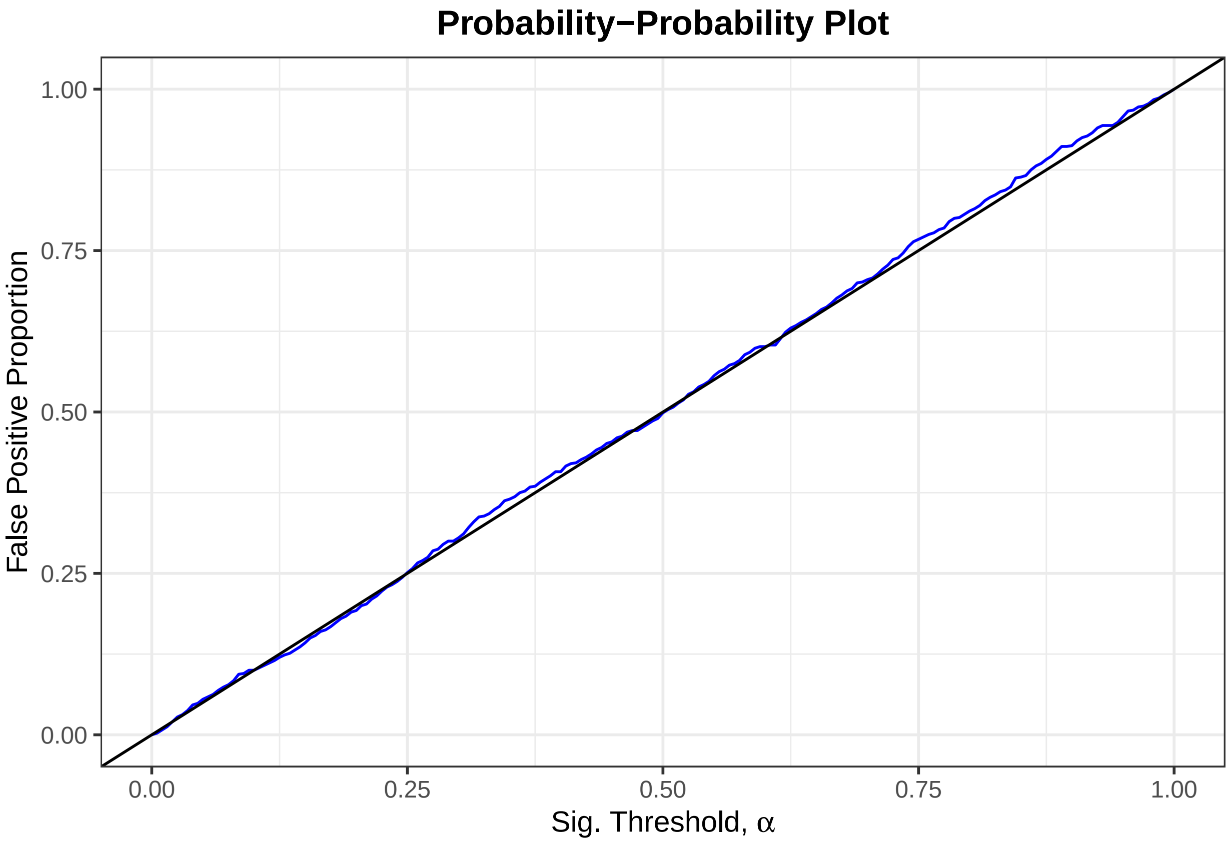
```
## [1] "Empirical mean = 0.505972256831217, Empirical variance = 0.0833340460887009"
## [1] "Generating Mann-Whitney plot for (n,k) = (200, 50)"
```



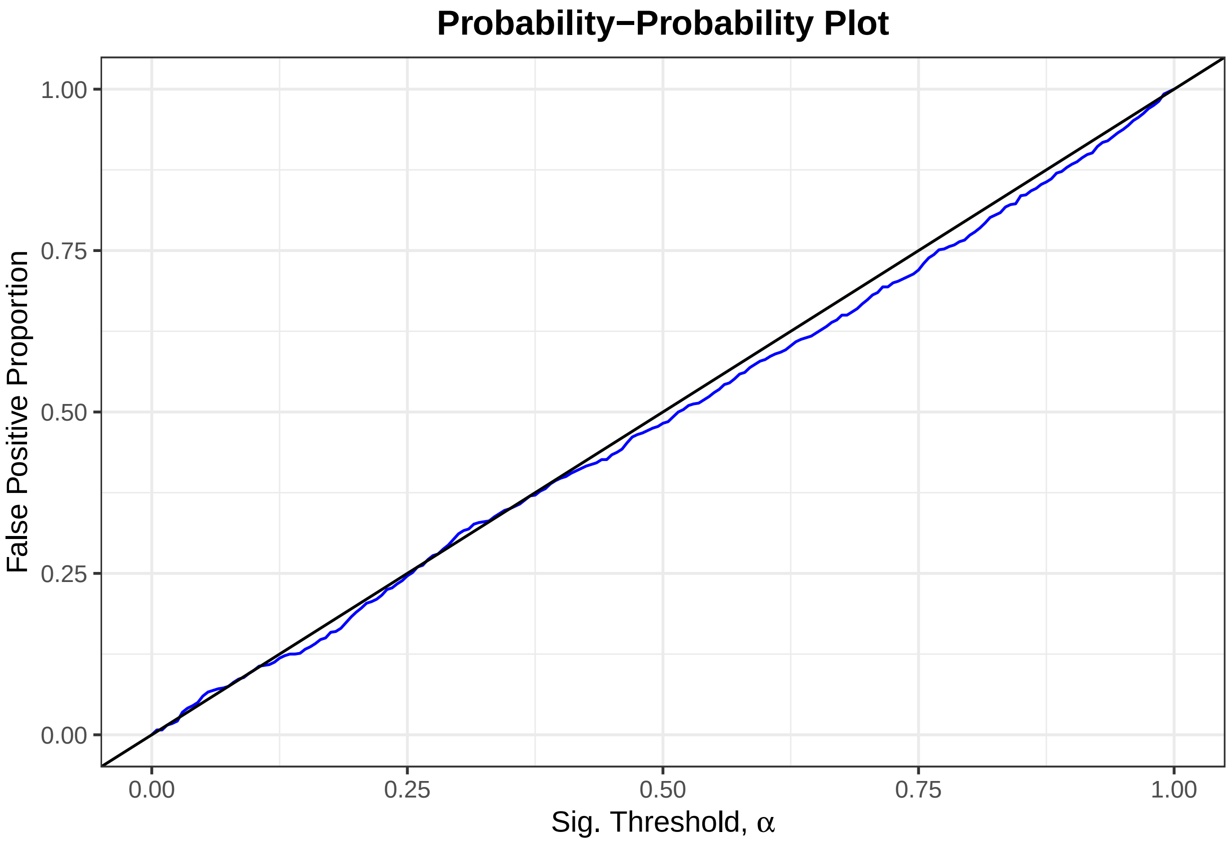
```
## [1] "Empirical mean = 0.502026200597895, Empirical variance = 0.0811951954062257"
## [1] "Generating Mann-Whitney plot for (n,k) = (500, 50)"
```



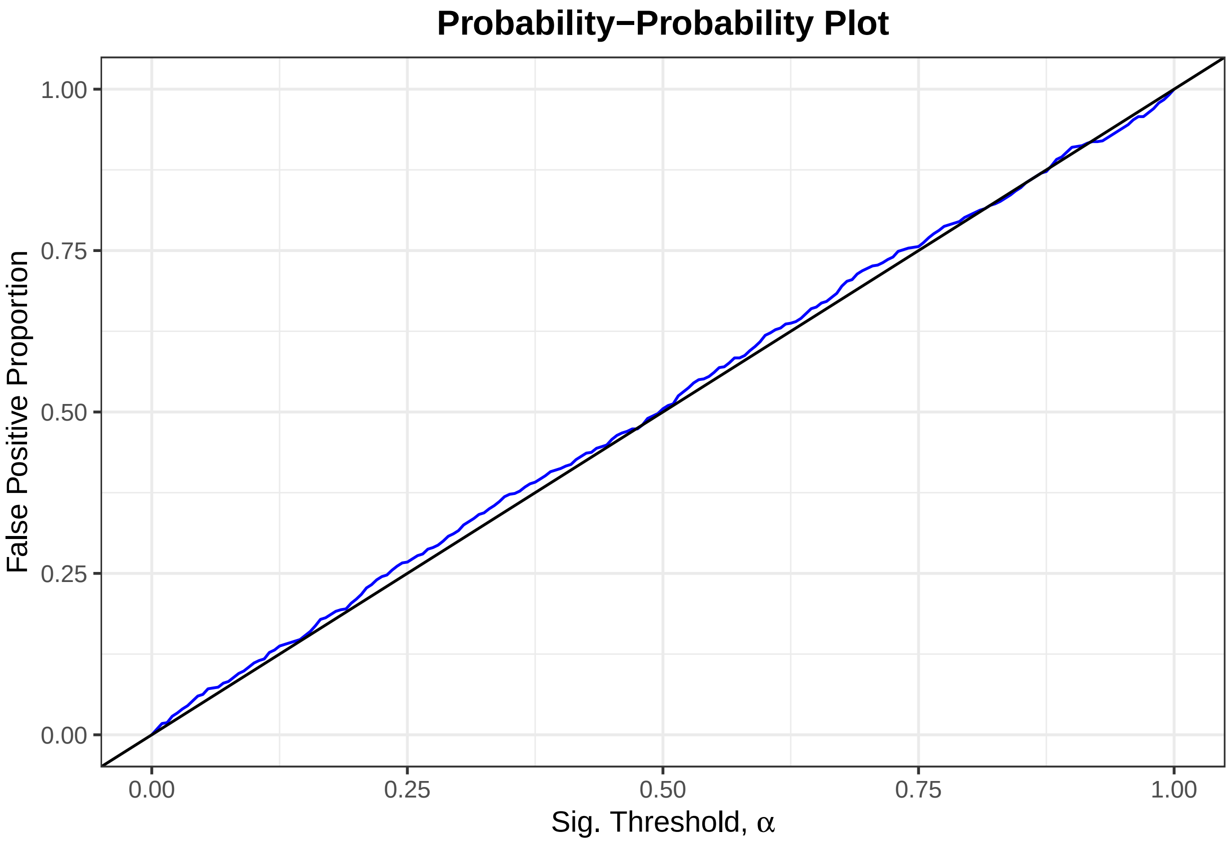
```
## [1] "Empirical mean = 0.498672196213225, Empirical variance = 0.082450827927078"
## [1] "Generating Mann-Whitney plot for (n,k) = (100, 100)"
```



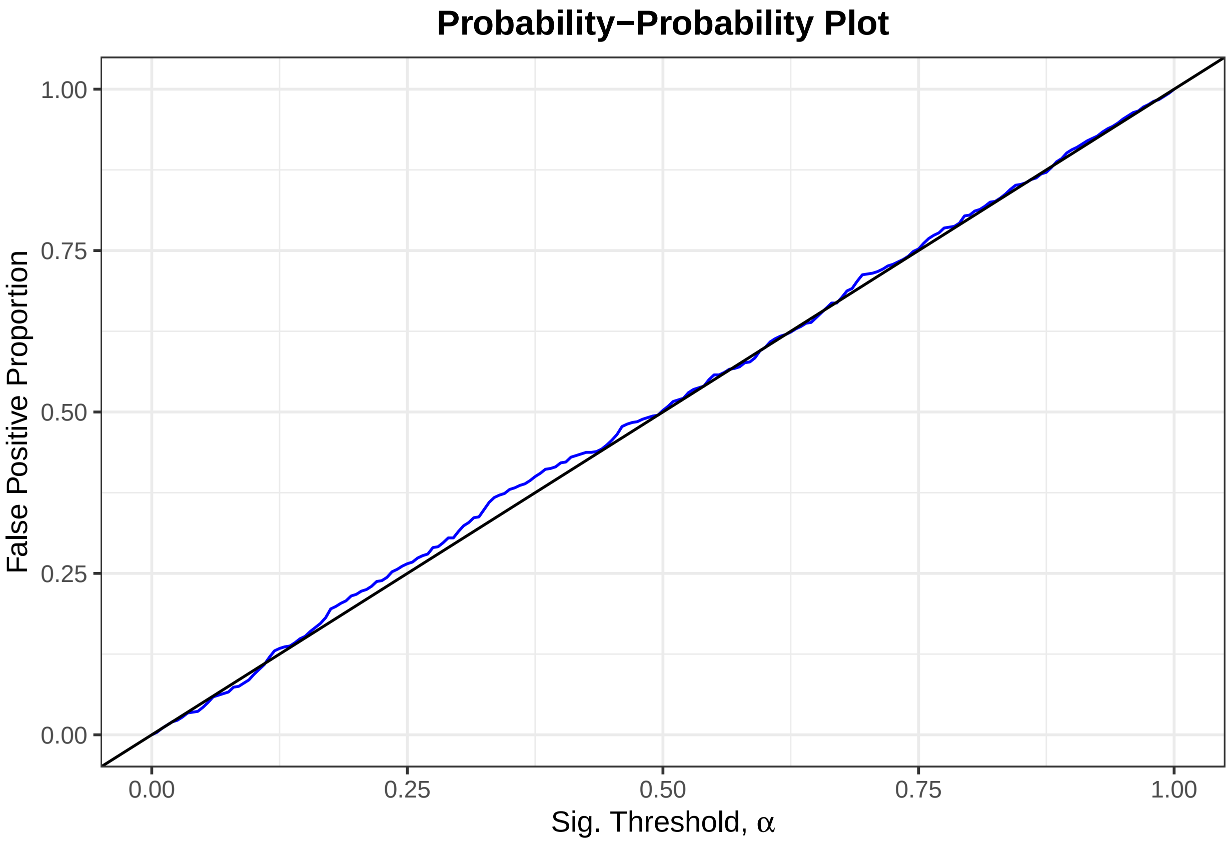
```
## [1] "Empirical mean = 0.494708382847003, Empirical variance = 0.0812917118656229"
## [1] "Generating Mann-Whitney plot for (n,k) = (200, 100)"
```



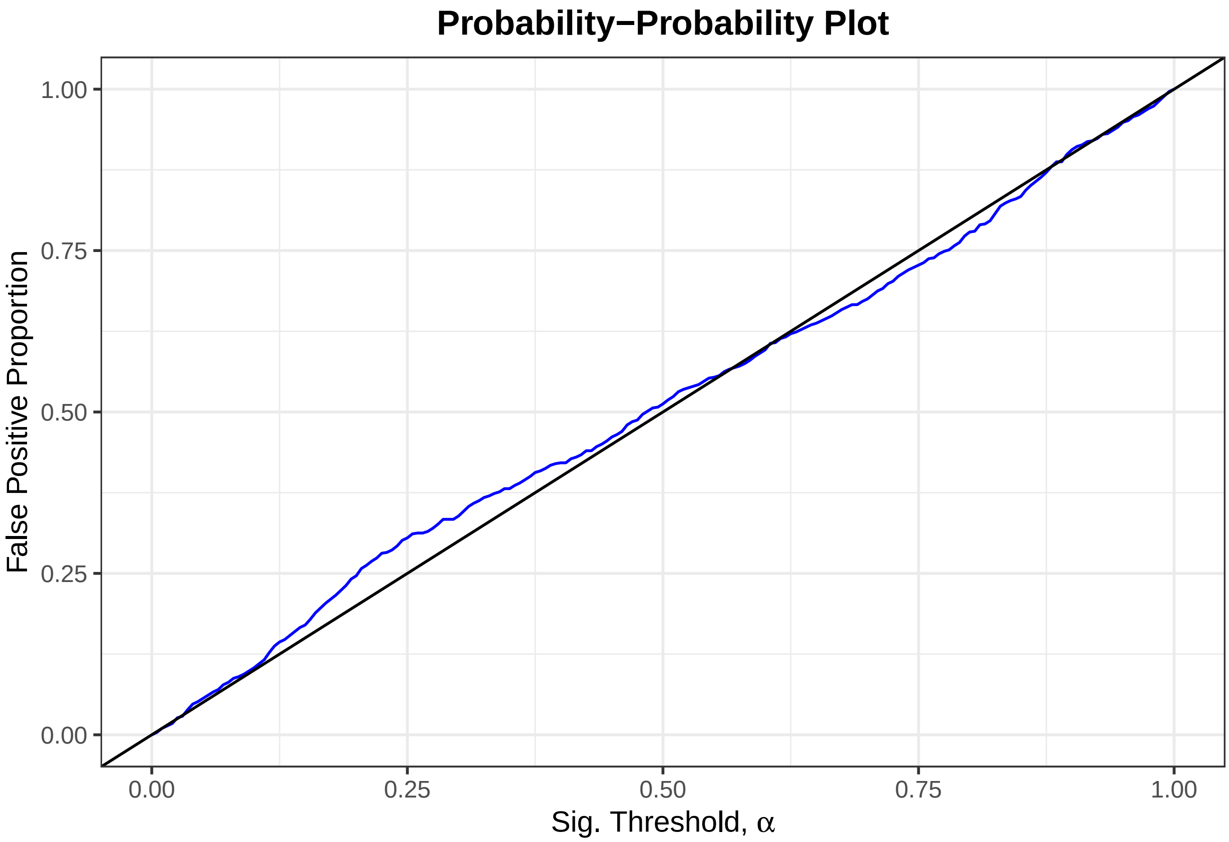
```
## [1] "Empirical mean = 0.512202505380208, Empirical variance = 0.0868928821454392"
## [1] "Generating Mann-Whitney plot for (n,k) = (500, 100)"
```



```
## [1] "Empirical mean = 0.490543477057158, Empirical variance = 0.0856527389297154"
## [1] "Generating Mann-Whitney plot for (n,k) = (200, 200)"
```



```
## [1] "Empirical mean = 0.493337014431638, Empirical variance = 0.0843880579780724"
## [1] "Generating Mann-Whitney plot for (n,k) = (500, 200)"
```



```
## [1] "Empirical mean = 0.491604105939967, Empirical variance = 0.0914715363734669"
```

On the whole, we see that our implementation achieves good FPR control for large values of n and k.