Tau specification analysis

Run on the cluster:

```r
start <- Sys.time()


# function to generate the discrete data and covariates
# takes a RNG seed, sample size, number of predictors, lambda (root of the
# non-zero elements in the precision matrix), values to populate the
# extraneous covariate vector with, and boolean for whether the two groups
# should have the same covariance
# returns the continuous data, covariates, and true covariance matrix
generate_discrete <- function(n = 100, p = 10, lambda = 15,
                              cov1 = -0.1, cov2 = 0.1, same = T){

  # generating the precision matrix: Assume two discrete covariate levels, one
  # for each group
  Lam1 <- c(rep(lambda, 4), rep(0, p - 3))
  Lam2 <- c(rep(0, 4), rep(lambda, p - 3))

  # if same is true, the individuals in both groups will have the same
  # covariance matrix
  if (same) Lam2 <- Lam1

  # create covariance matrix for both groups
  Var1 <- solve(Lam1 %*% t(Lam1) + diag(rep(10, p + 1)))
  Var2 <- solve(Lam2 %*% t(Lam2) + diag(rep(10, p + 1)))

  # create the extraneous covariate; individuals in group j have a covariate
  # vector of length p with cov_j as the only entry, j\in {1,2}
  Z <- matrix(c(rep(cov1, n %/% 2), rep(cov2, n %/% 2)), n, p)

  # create the data matrix; individuals in group j are generated from a MVN with
  # 0 mean vector and covariance matrix Var_j, j\in {1,2}
  X1 <- MASS::mvrnorm(n %/% 2, rep(0, p + 1), Var1)
  X2 <- MASS::mvrnorm(n %/% 2, rep(0, p + 1), Var2)
  data_mat <- rbind(X1, X2)

  return(list(data = data_mat, covts = Z, true_precision = list(solve(Var1), solve(Var2))))

}

library(covdepGE)
library(doRNG)

set.seed(1)
```

```r
# number of trials and values of bandwidth to try
n_trials <- 1000
tau <- c(0.01, 0.05, 0.1, 0.5, 1, 5)

# grid of slab variance
sbsq_grid <- c(0.01, 0.05, 0.1, 0.5, 1, 3, 7, 10)

n_cores <- parallel::detectCores() - 10
print(paste("N cores:", n_cores))
doParallel::registerDoParallel(n_cores)

results <- foreach (j = 1:n_trials, .packages = "covdepGE") %dorng%{

  # generate the data
  cont <- generate_discrete(same = F)
  X <- cont$data
  Z <- cont$covts

  # list for storing the results from each bandwidth and the data
  bw_res <- vector("list", length(tau) + 1)
  bw_res[[1]] <- cont
  names(bw_res) <- c("data", tau)

  # loop over each of the bandwidths
  for (bandwidth in tau){
    bw_res[[as.character(bandwidth)]] <- tryCatch(
      covdepGE(X, Z, sbsq = sbsq_grid, CS = T, tau = bandwidth, kde = F, scale = F,
               elbo_tol = 1e-12, alpha_tol = 1e-12, max_iter = 1e2, warnings = F),
      error = function(msg) as.character(msg))
  }

  # save the final results
  bw_res
}

doParallel::stopImplicitCluster()

Sys.time() - start

#save(results, file = "tau_specif_disc_indep_models.Rda")
```

```r
library(ggplot2)
library(latex2exp)
library(covdepGE)
library(foreach)

load(paste0("C:/Users/jacob/OneDrive/Documents/TAMU/Research/An approximate ",
            "Bayesian approach to covariate dependent/covdepGE/dev/",
            "analyses_demos_experiments/hyperparameter_specification/",
            "tau_specif_disc_dep_models.Rda"))

n_trials <- 1000
tau <- c(0.01, 0.05, 0.1, 0.5, 1, 5)
```

```
names(results) <- paste("Trial", 1:n_trials)

# get all of the models
mods <- lapply(results, `[`, as.character(tau))

# look at the ELBO
elbo <- lapply(lapply(mods, lapply, `[[`, "model_details"), lapply, `[[`, "ELBO")
ELBO_vec <- unlist(elbo)
length(ELBO_vec[ELBO_vec < -65e3])
```

```
## [1] 78
```

```
length(ELBO_vec[ELBO_vec > -65e3])
```

```
## [1] 5922
```

```
summary(ELBO_vec[ELBO_vec < -65e3])
```

```
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## -4.912e+20 -7.368e+07 -8.220e+06 -1.889e+19 -4.563e+05 -1.721e+05
```

```
summary(ELBO_vec[ELBO_vec > -65e3])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -62403  -56876  -55774  -51046  -44996  -41318
```

```
# filter out the models that blew up
elbo_stable <- lapply(elbo, function(elbo) elbo > -65e3)
mods_unstable <- lapply(1:n_trials, function(trial_ind)
  mods[[trial_ind]][!elbo_stable[[trial_ind]]])
names(mods_unstable) <- names(elbo_stable)

# analyze the models that blew up

# filter out those of length 0
mods_unstable0 <- mods_unstable[lapply(mods_unstable, length) != 0]

# filter out the trials with instability
mods_stable <- mods[setdiff(names(mods), names(mods_unstable0))]
length(mods_stable)
```

```
## [1] 974
```

```
# look at the mu distribution for those that blew up
unstable_mu_mat <- lapply(mods_unstable0, lapply, `[[`, "mu_matrices")
max_trial_bw <- lapply(unstable_mu_mat, lapply, sapply, function(mu_mat) max(abs(mu_mat)))
max_var <- as.numeric(sapply(max_trial_bw, lapply, max))
summary(max_var)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
##         3         5        16   5701779        65 148233900
```

```
stable_mu_mat <- lapply(mods_stable, lapply, `[[`, "mu_matrices")
max_trial_bw <- lapply(stable_mu_mat, lapply, sapply, function(mu_mat) max(abs(mu_mat)))
max_var <- as.numeric(sapply(max_trial_bw, lapply, max))
summary(max_var)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.06855 0.29695 0.79934 0.64354 0.97444 1.27277
```

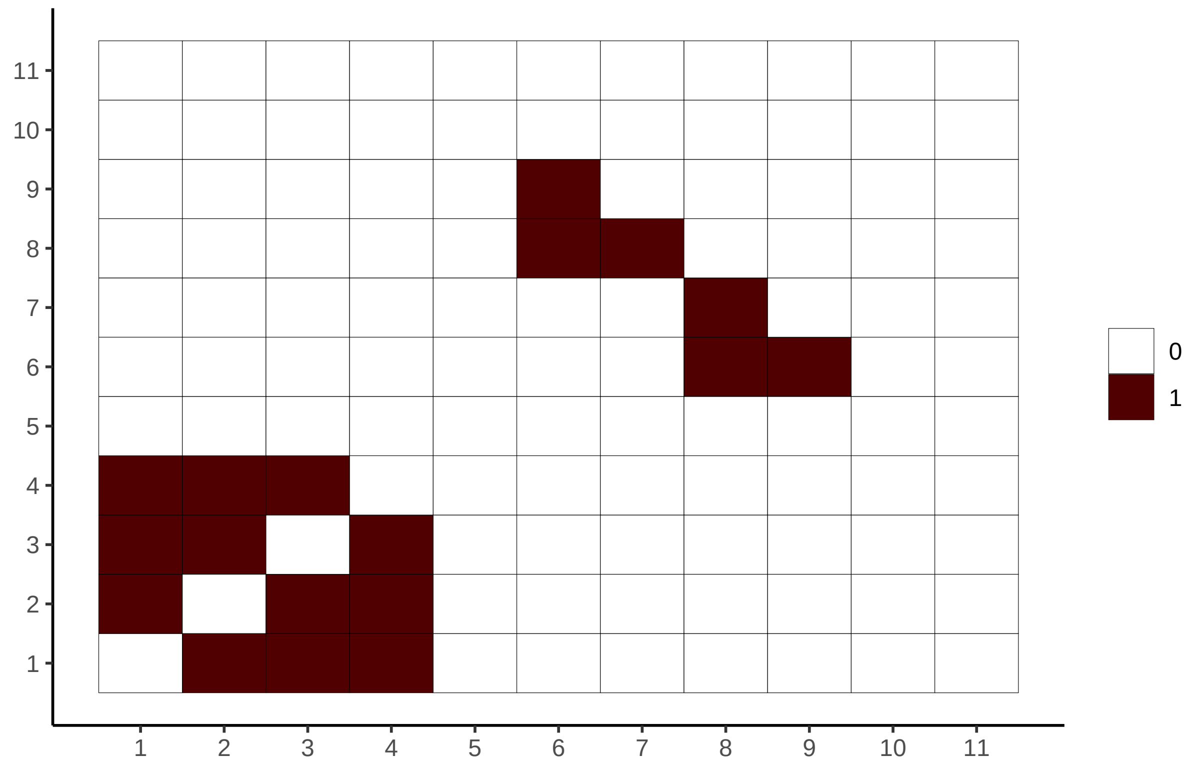
```
# visualize the unique graphs for a model that blew up
sapply(lapply(mods_unstable0, lapply, `[[`, "model_details"), lapply, `[[`, "ELBO")
```

```
##      Trial 19      Trial 64 Trial 155    Trial 237 Trial 238 Trial 279
## 0.01 -950512394296 -1236622 -26216098512 -941824.7 -1903959  -155229331
## 0.05 -950512394296 -1236622 -26216098512 -941824.7 -1903959  -155229331
## 0.1  -950512110169 -1236620 -26216092993 -941824.7 -1903959  -155229225
##      Trial 286     Trial 320 Trial 342 Trial 354 Trial 390 Trial 523 Trial 533
## 0.01 -4.911797e+20 -8225943  -11830767 -204224.5 -30034282 -2004566  -333612.7
## 0.05 -4.911797e+20 -8225943  -11830767 -204224.5 -30034282 -2004566  -333612.7
## 0.1  -4.911793e+20 -8225939  -11830762 -204224.4 -30034271 -2004564  -333612.1
##      Trial 582 Trial 588   Trial 590 Trial 594 Trial 619     Trial 803
## 0.01 -439050.3 -7388180127 -73323665 -13645440 -2.20076e+11  -73675532
## 0.05 -439050.3 -7388180127 -73323665 -13645440 -2.20076e+11  -73675532
## 0.1  -439050.3 -7388176215 -73323643 -13645439 -220075769618 -73675456
##      Trial 807 Trial 840 Trial 857 Trial 887 Trial 929 Trial 964 Trial 971
## 0.01 -456307.2 -34063307 -435736.1 -8214636  -321110.6 -172075.8 -3480541
## 0.05 -456307.2 -34063307 -435736.1 -8214636  -321110.6 -172075.8 -3480541
## 0.1  -456306.7 -34063177 -435735.3 -8214632  -321110.5 -172075.7 -3480539
```

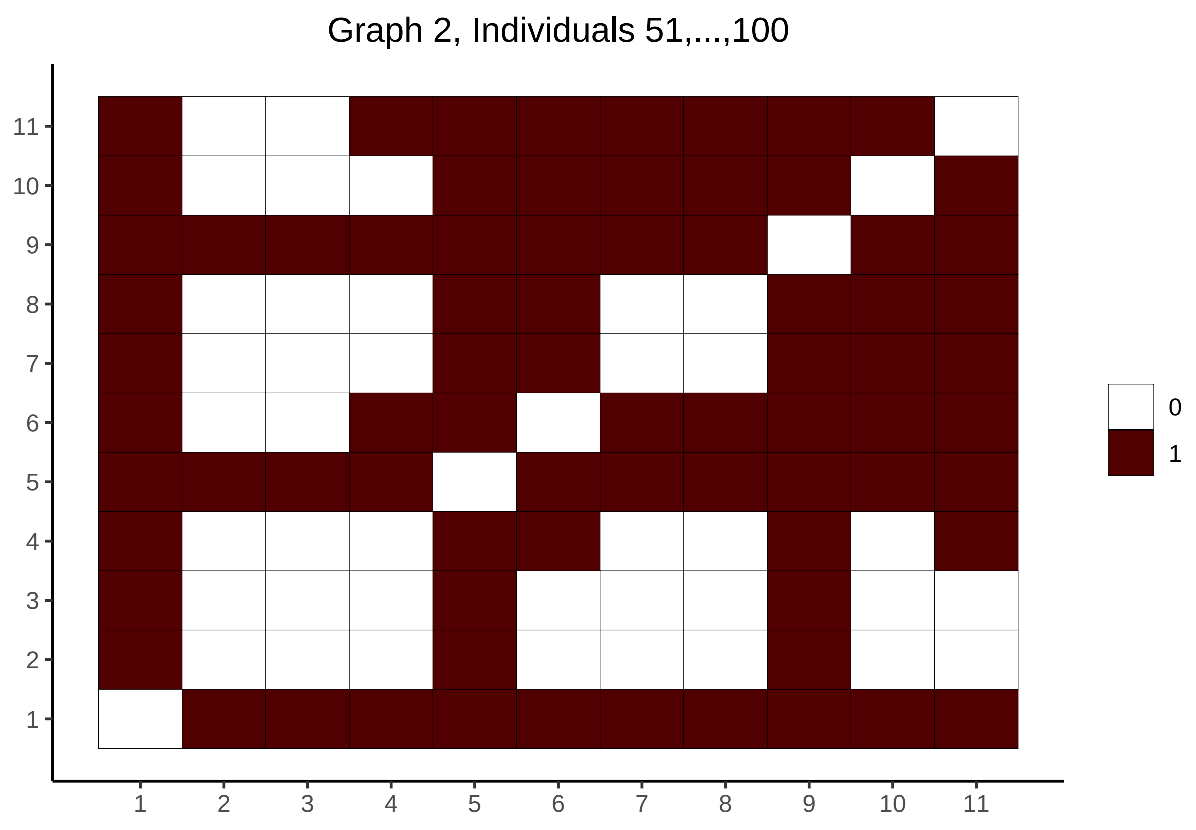
```
plot(mods_unstable0[["Trial 286"]]$`0.01`, title_sum = T)
```

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

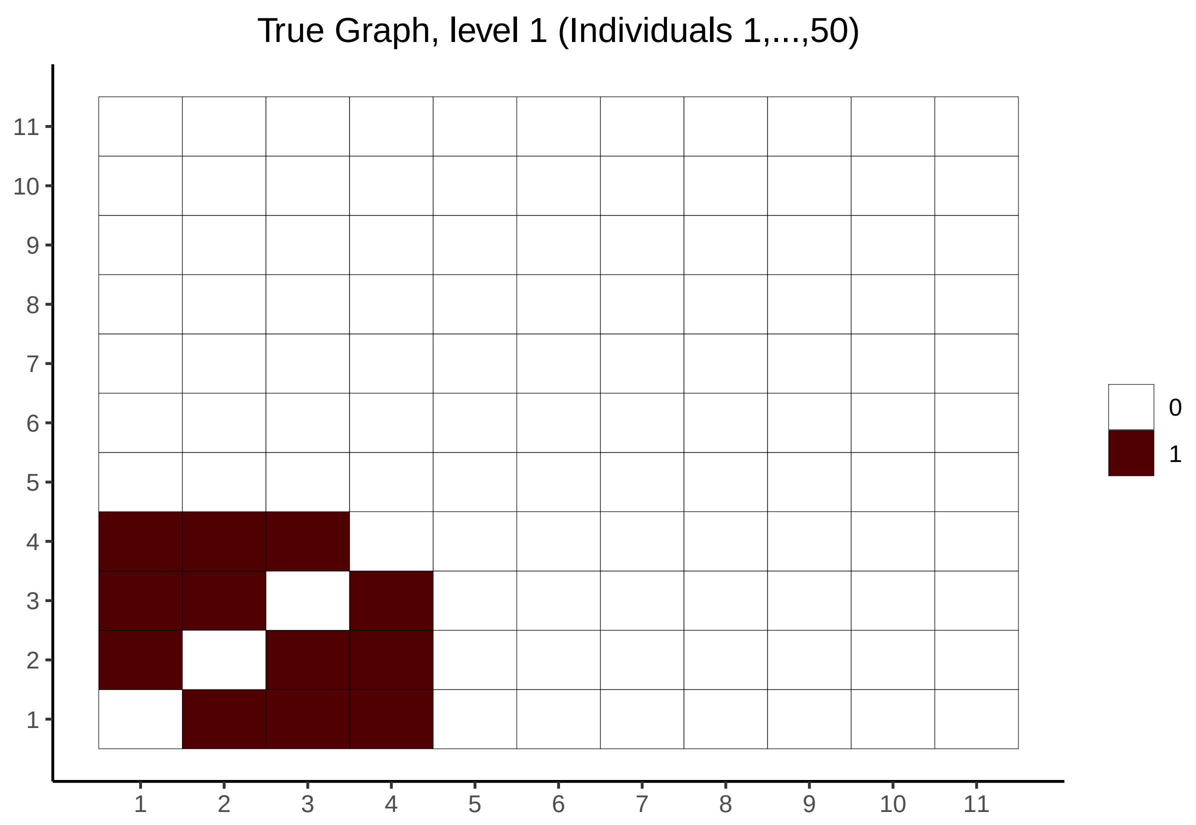
Graph 1, Individuals 1,...,50



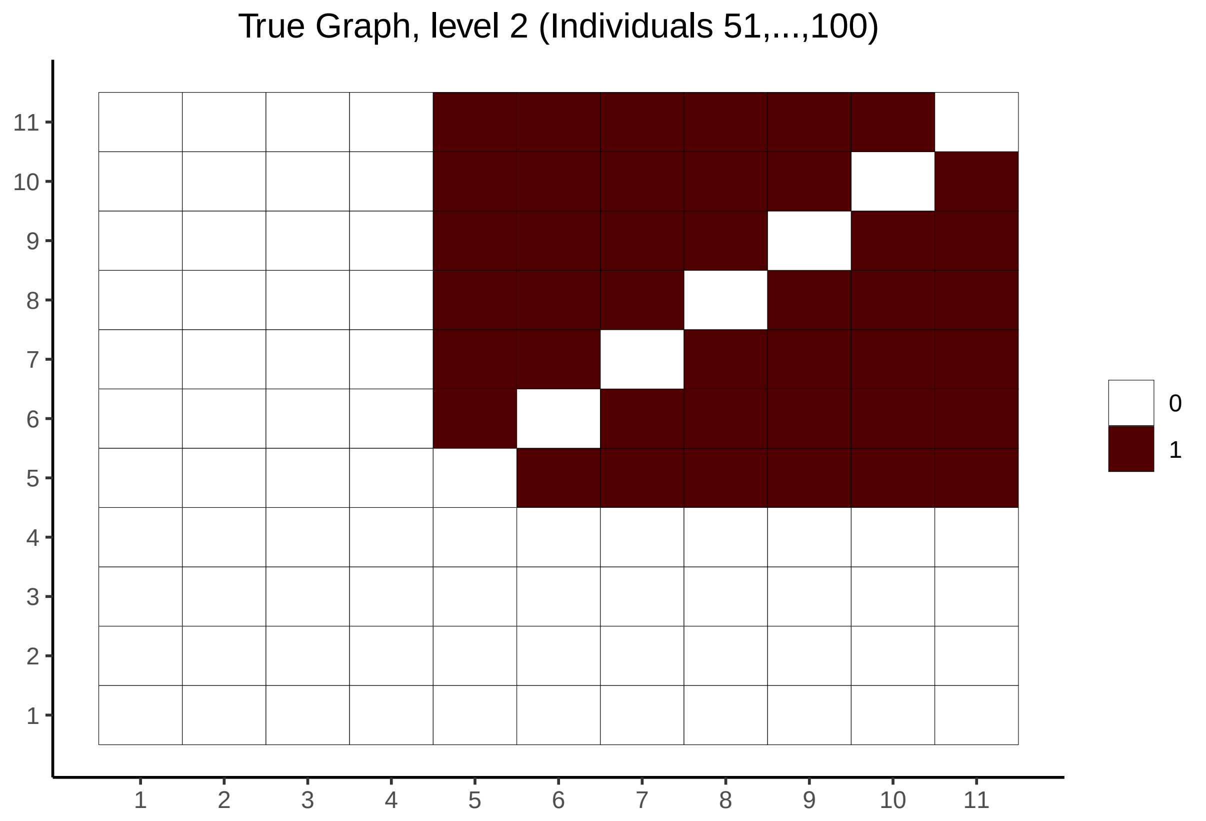
```
##
## [[2]]
```



```
# get the true precision structure and visualize
true_prec <- results$`Trial 1`$data$true_precision
true_graph <- lapply(true_prec, function(prec) ((prec - diag(diag(prec))) != 0) * 1)
true_graph1 <- true_graph[[1]]
true_graph2 <- true_graph[[2]]
gg_adjMat(true_graph1) + ggtitle("True Graph, level 1 (Individuals 1,...,50)") +
  theme(plot.title = element_text(hjust = 0.5))
```

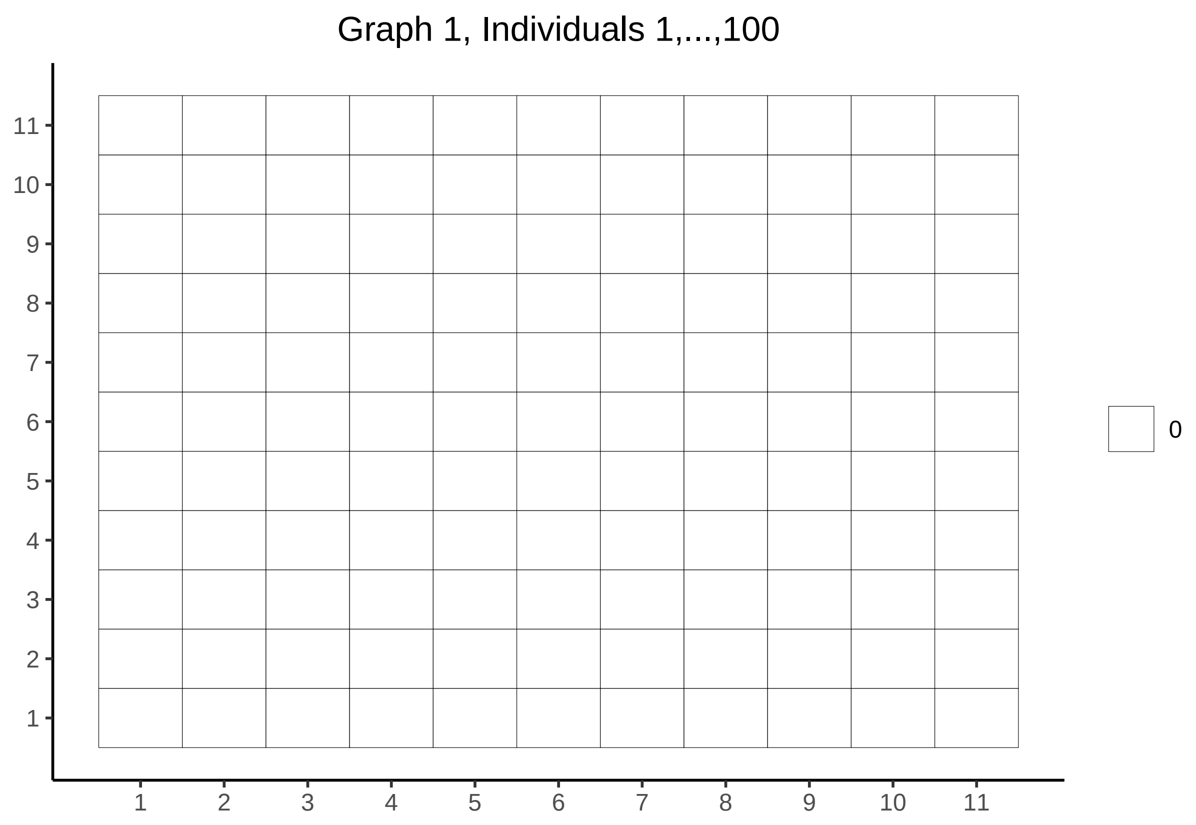


```
gg_adjMat(true_graph2) + ggtitle("True Graph, level 2 (Individuals 51,...,100)") +
  theme(plot.title = element_text(hjust = 0.5))
```



```r
# find the true number of 0's and 1's in one graph and across all structures
# within each of the covariate levels
n <- nrow(results$`Trial 1`$data$data)
true1_graph1 <- sum(true_graph1 == 1)
true0_graph1 <- sum(true_graph1 == 0)
true1_graph2 <- sum(true_graph2 == 1)
true0_graph2 <- sum(true_graph2 == 0)
true1_tot1 <- true1_graph1 * n / 2
true0_tot1 <- true0_graph1 * n / 2
true1_tot2 <- true1_graph2 * n / 2
true0_tot2 <- true0_graph2 * n / 2

# find the estimated graphs for each model
mod_graphs <- lapply(mods_stable, lapply, `[[`, "graphs")
mod_graphs1 <- lapply(mod_graphs, lapply, `[`, 1:50)
mod_graphs2 <- lapply(mod_graphs, lapply, `[`, 51:100)

# find the true 1's and 0's for each estimated graph within each level
true_pos_gr1 <- lapply(
  mod_graphs1, lapply, function(model) sum(sapply(
    model, function(graph) sum(graph == 1 & true_graph1 == 1))))
true_neg_gr1 <- lapply(
  mod_graphs1, lapply, function(model) sum(sapply(
    model, function(graph) sum(graph == 0 & true_graph1 == 0))))
true_pos_gr2 <- lapply(
  mod_graphs2, lapply, function(model) sum(sapply(
    model, function(graph) sum(graph == 1 & true_graph2 == 1))))
true_neg_gr2 <- lapply(
  mod_graphs2, lapply, function(model) sum(sapply(
    model, function(graph) sum(graph == 0 & true_graph2 == 0))))

# find sensitivity and specificity
sens1 <- lapply(true_pos_gr1, lapply, lapply, `/`, true1_tot1)
sens2 <- lapply(true_pos_gr2, lapply, lapply, `/`, true1_tot2)
spec1 <- lapply(true_neg_gr1, lapply, lapply, `/`, true0_tot1)
spec2 <- lapply(true_neg_gr2, lapply, lapply, `/`, true0_tot2)

min_spec <- round(min(unlist(c(spec1, spec2))), 2)
max_spec <- max(unlist(c(spec1, spec2)))
min_sens <- round(min(unlist(c(sens1, sens2))))
max_sens <- round(max(unlist(c(sens1, sens2))))

# display the results from a model that had zero sensitivity
plot(results$`Trial 83`$`1`, title_sum = T)
```

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



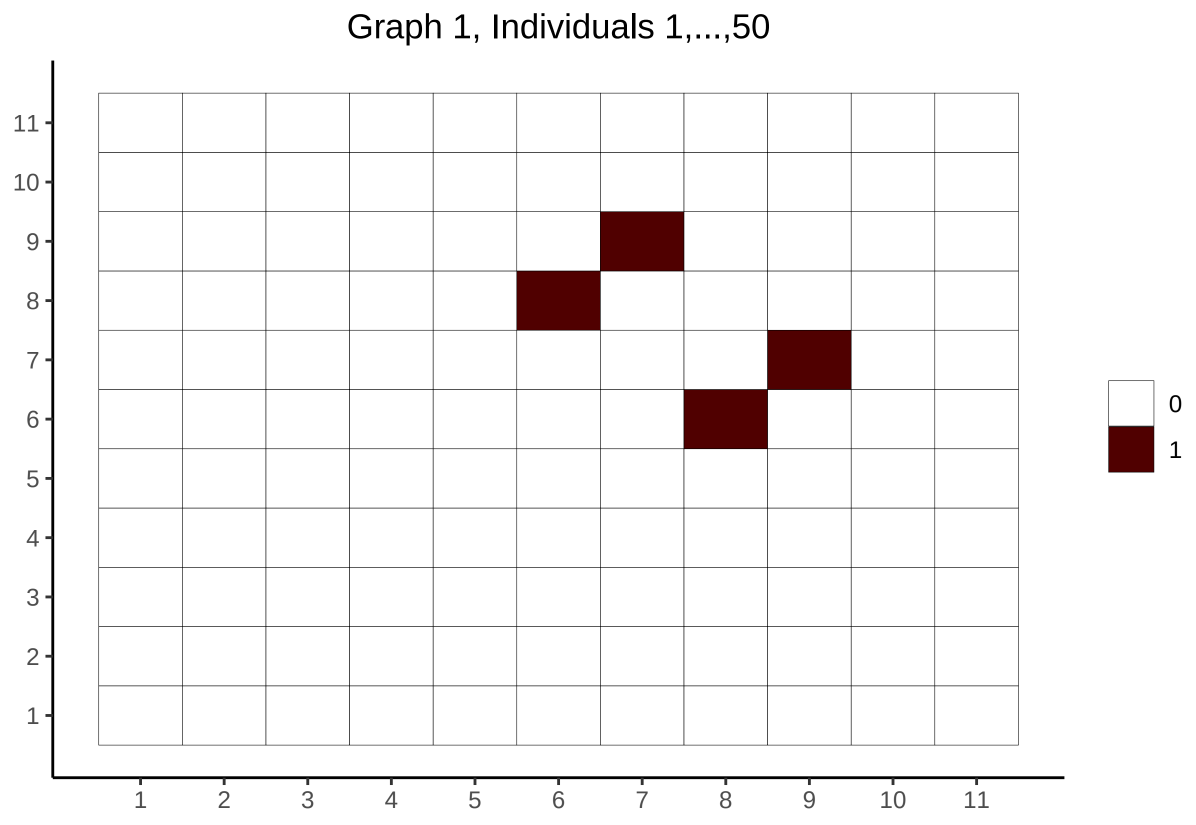
```r
# analyze models with 0 sensitivity for level 1
sens_0 <- lapply(sens1, lapply, `==`, 0)
mods_sens0 <- lapply(1:length(mods_stable), function(trial_ind)
  mods_stable[[trial_ind]][unlist(sens_0[[trial_ind]], F)])
```

```r
# filter out zero length elements
mods_sens0 <- mods_sens0[sapply(mods_sens0, length) != 0]

# how many unique graphs were estimated by these models?
table(unlist(sapply(mods_sens0, lapply,
                    function(mod) mod$model_details$num_unique)))
```

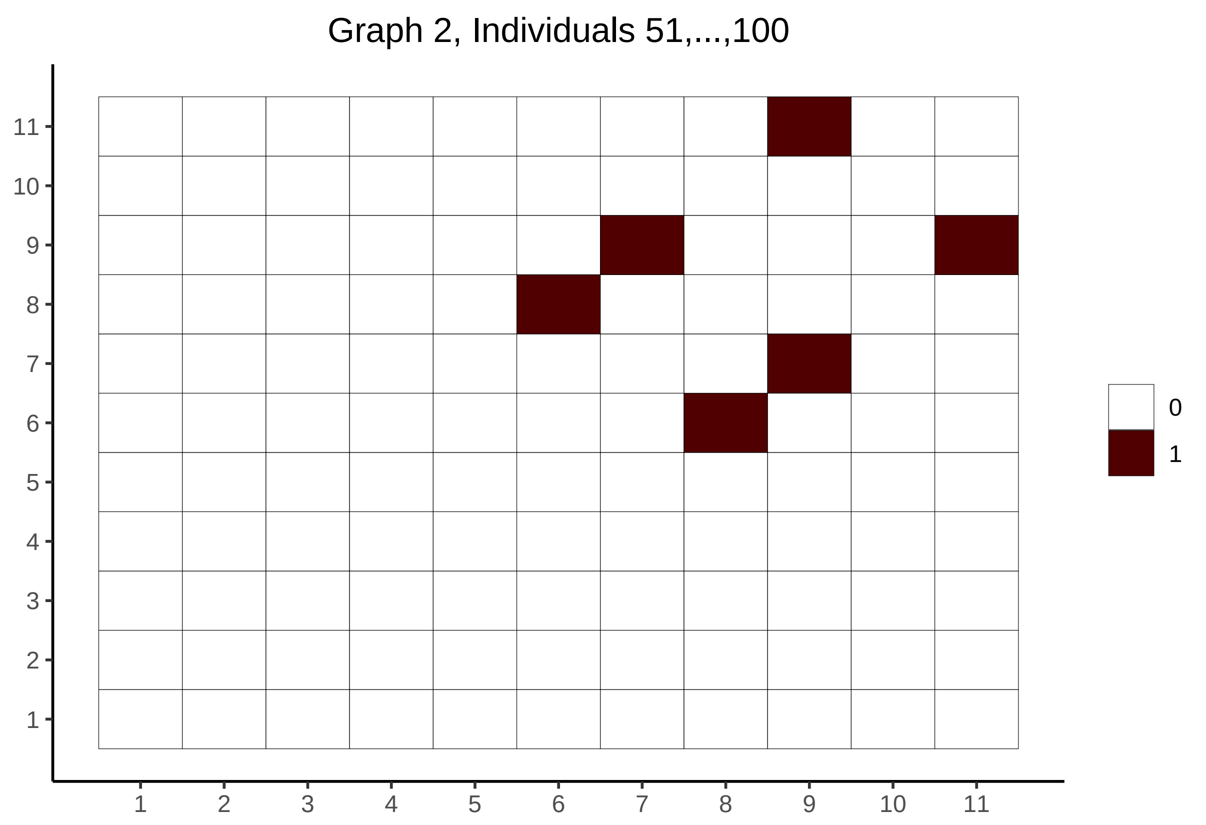
```
##
##   1   2
## 369 146
```

```r
# display one of the graphs where there were two unique structures
plot(mods_sens0[[3]]$`1`, title_sum = T)
```

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

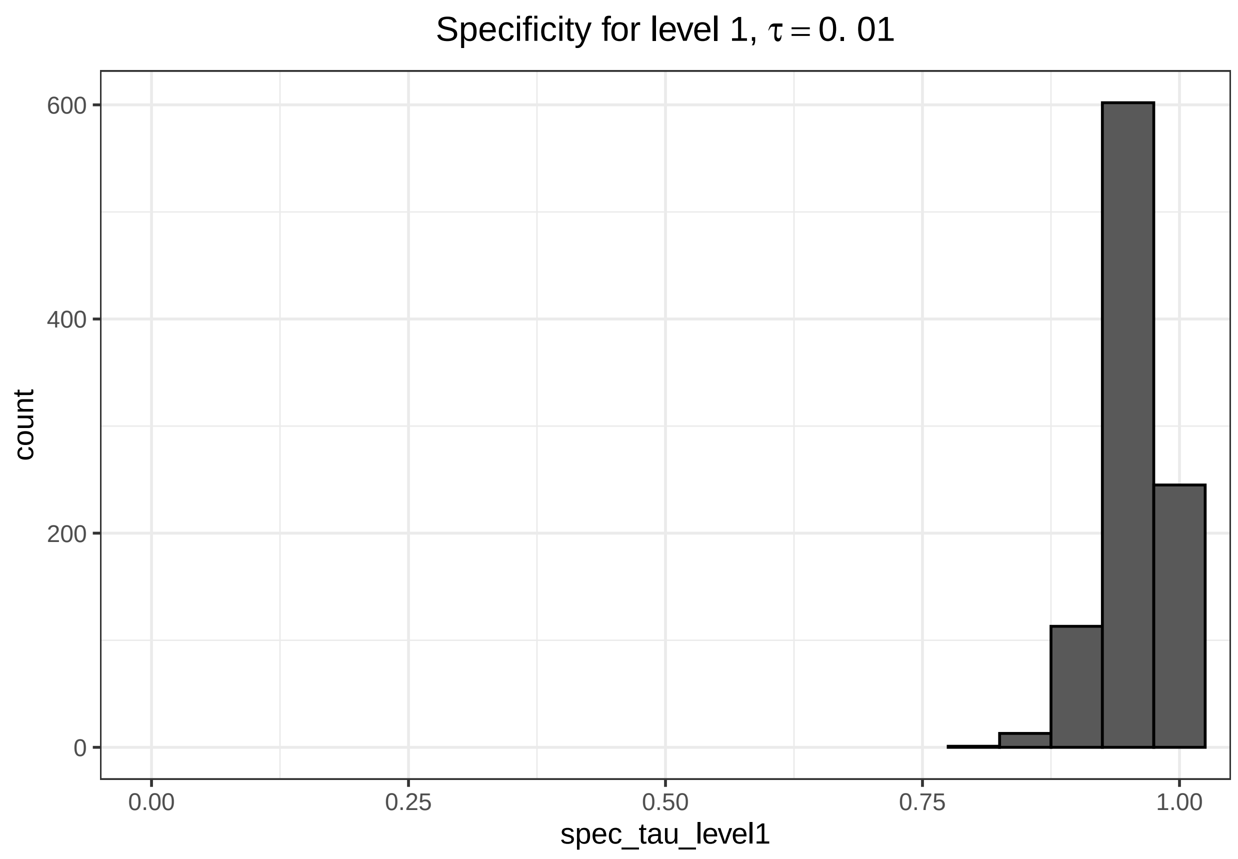


```
##
## [[2]]
```

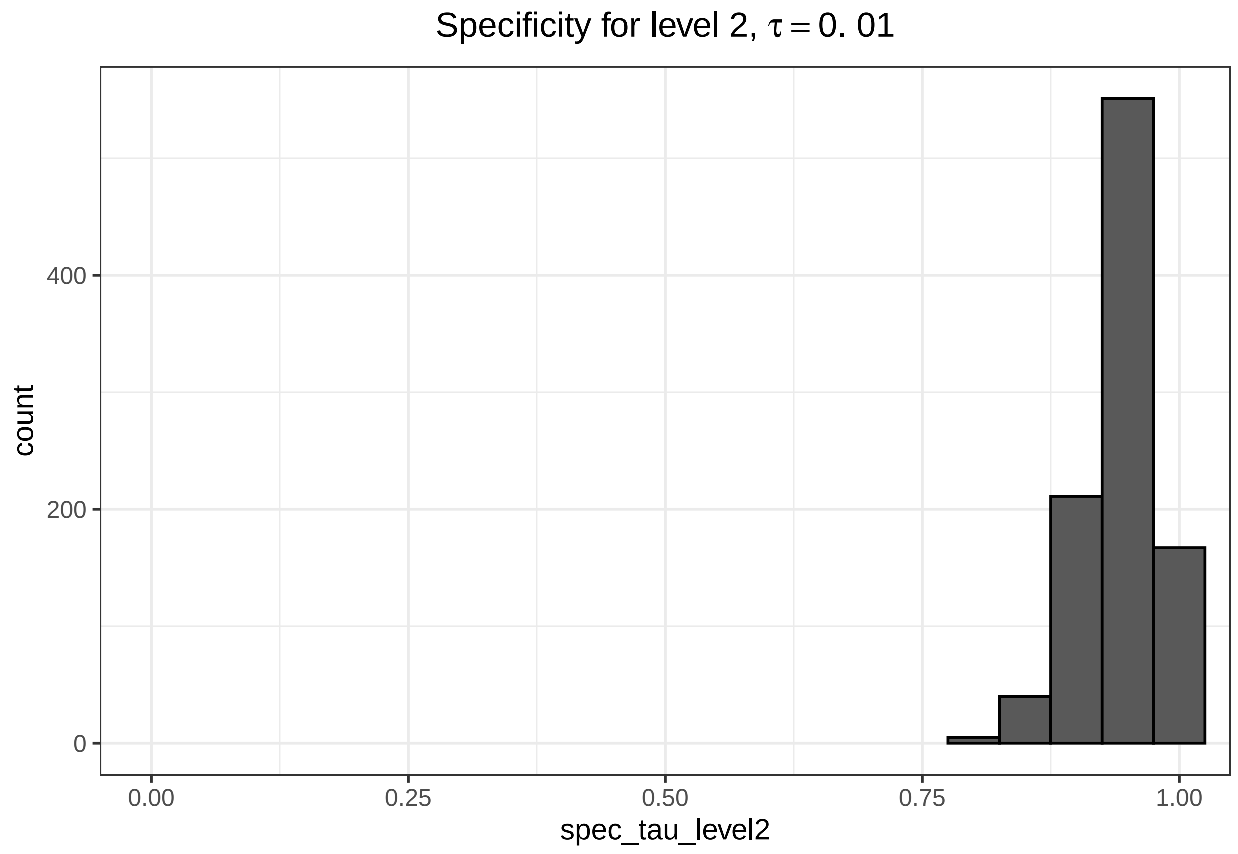


```r
# get visualizations and summaries for each of the specificities and sensitivities
for (bandwidth in as.character(tau)){
  cat("\n\nResults for tau =", bandwidth, "\n\nWeights: ")
  print(unique(as.numeric(results$`Trial 1`[[bandwidth]]$weights)))
  cat("\n")

  # get sensitivity and specificity
  spec_tau_level1 <- unlist(sapply(spec1, `[[`, bandwidth))
  spec_tau_level2 <- unlist(sapply(spec2, `[[`, bandwidth))
  sens_tau_level1 <- unlist(sapply(sens1, `[[`, bandwidth))
  sens_tau_level2 <- unlist(sapply(sens2, `[[`, bandwidth))

  # visualize the specificities
  plot_spec1 <- ggplot() + geom_histogram(aes(x = spec_tau_level1), binwidth = 0.05, color = "black") +
    coord_cartesian(xlim = c(min_sens, max_spec)) +
    ggtitle(TeX(paste0("Specificity for level 1, $\\tau = $", bandwidth))) + theme_bw() +
    theme(plot.title = element_text(hjust = 0.5))
  plot_spec2 <- ggplot() + geom_histogram(aes(x = spec_tau_level2), binwidth = 0.05, color = "black") +
    coord_cartesian(xlim = c(min_sens, max_spec)) +
    ggtitle(TeX(paste0("Specificity for level 2, $\\tau = $", bandwidth))) + theme_bw() +
    theme(plot.title = element_text(hjust = 0.5))
  print(plot_spec1)
  print(summary(spec_tau_level1))
  print(plot_spec2)
  print(summary(spec_tau_level2))
  # visualize the sensitivities
  plot_sens1 <- ggplot() + geom_histogram(aes(x = sens_tau_level1), binwidth = 0.05, color = "black") +
    coord_cartesian(xlim = c(min_sens, max_sens)) +
    ggtitle(TeX(paste0("Sensitivity for level 1, $\\tau = $", bandwidth))) + theme_bw() +
    theme(plot.title = element_text(hjust = 0.5))
  plot_sens2 <- ggplot() + geom_histogram(aes(x = sens_tau_level2), binwidth = 0.05, color = "black") +
    coord_cartesian(xlim = c(min_sens, max_sens)) +
    ggtitle(TeX(paste0("Sensitivity for level 2, $\\tau = $", bandwidth))) + theme_bw() +
    theme(plot.title = element_text(hjust = 0.5))
  print(plot_sens1)
  print(summary(sens_tau_level1))
  print(plot_sens2)
  print(summary(sens_tau_level2))

  cat("\n", rep("-", 80), sep = "")

}
```

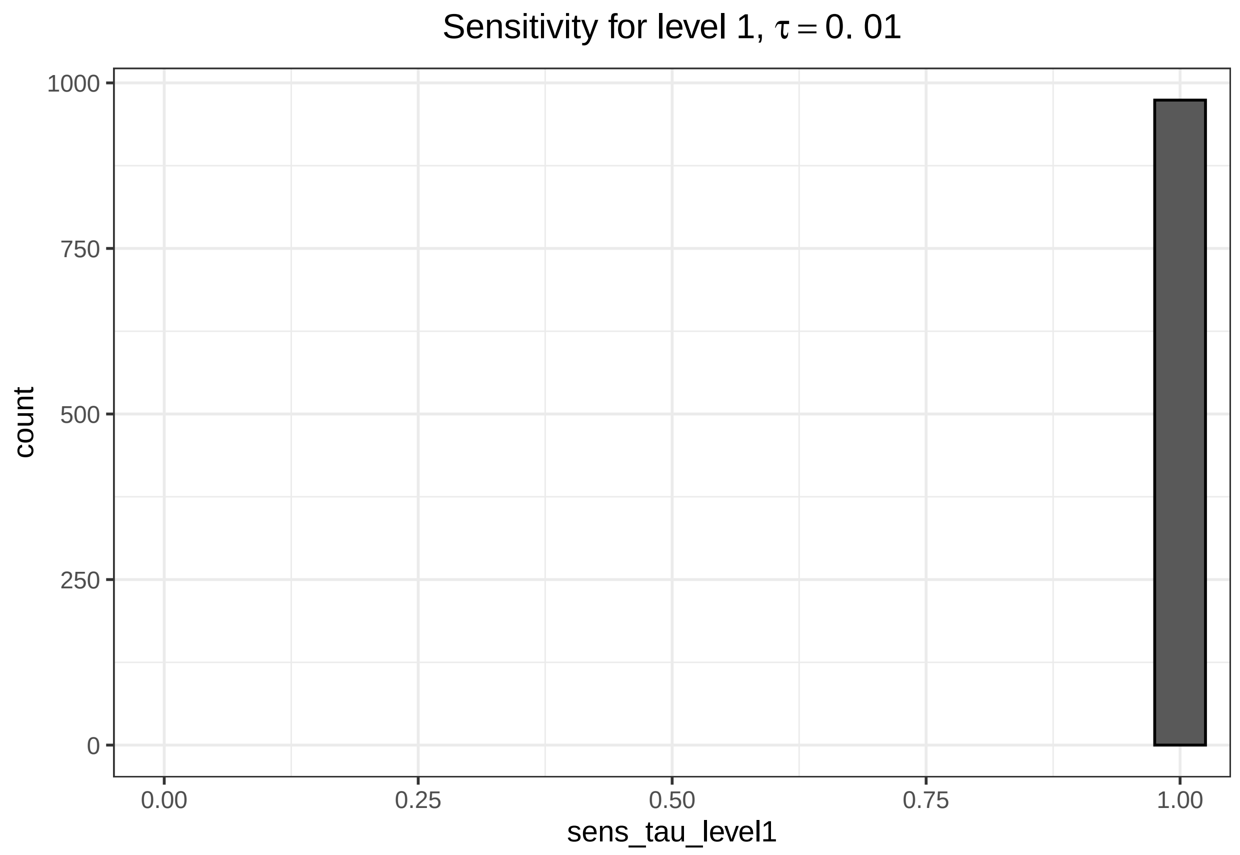
```
##
##
## Results for tau = 0.01
##
## Weights: [1] 2 0
```



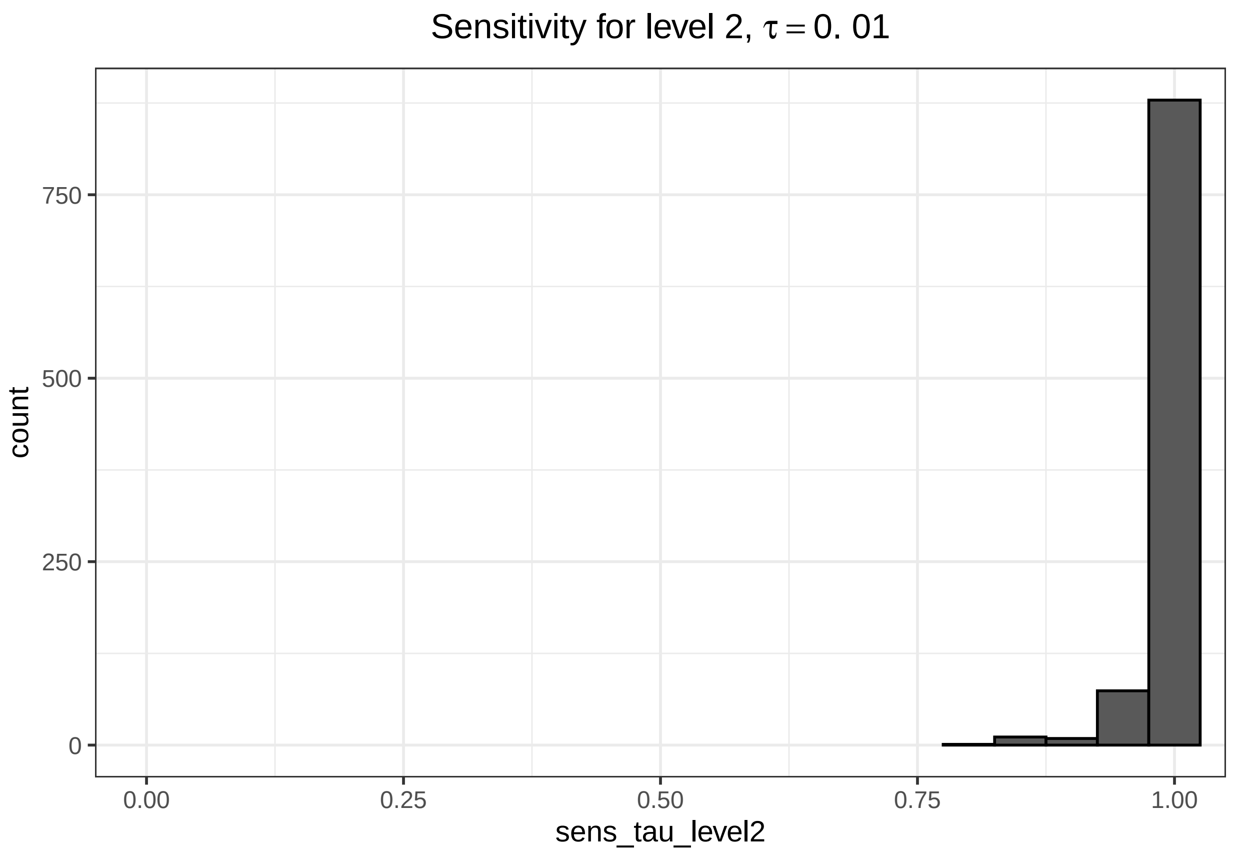
```
##     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.7798  0.9266  0.9633  0.9516  0.9817  1.0000
```



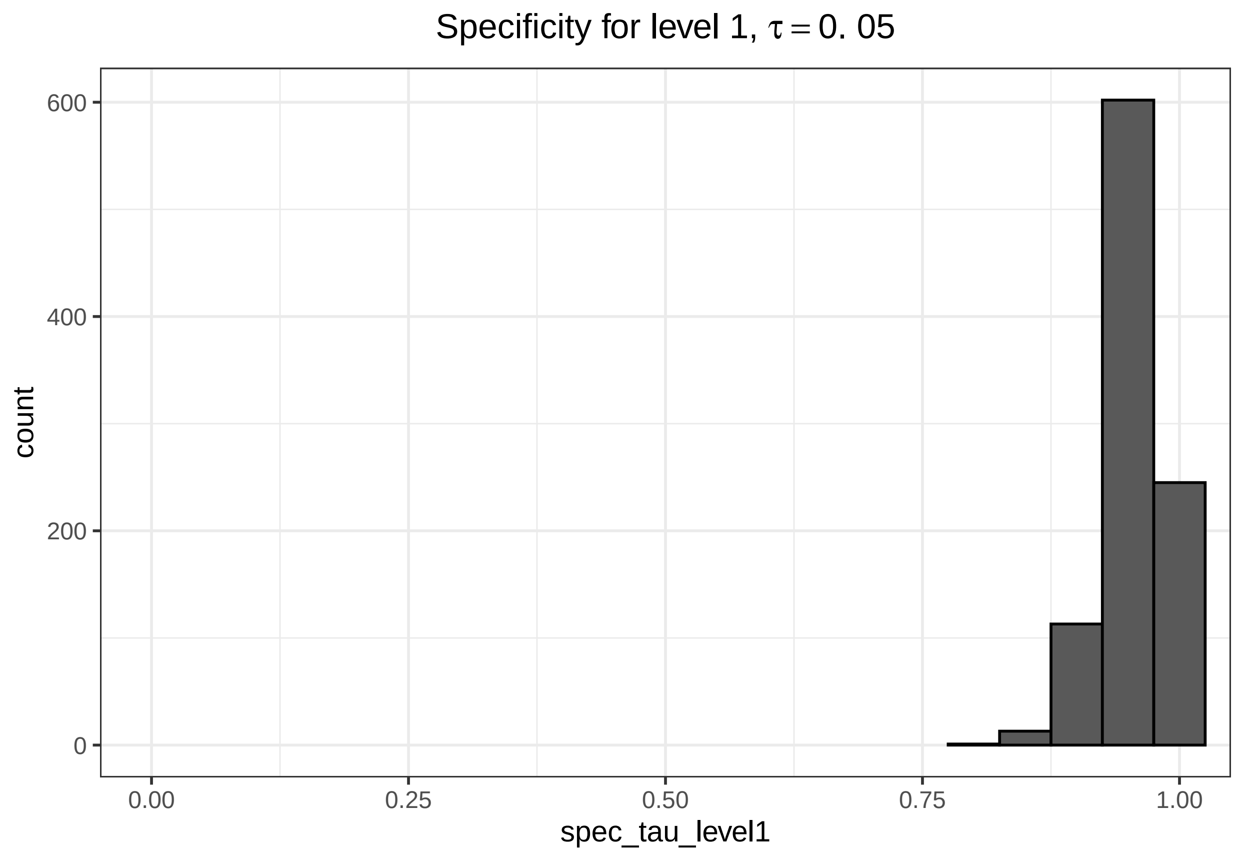
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.7975  0.9241  0.9747  0.9553  0.9747  1.0000
```



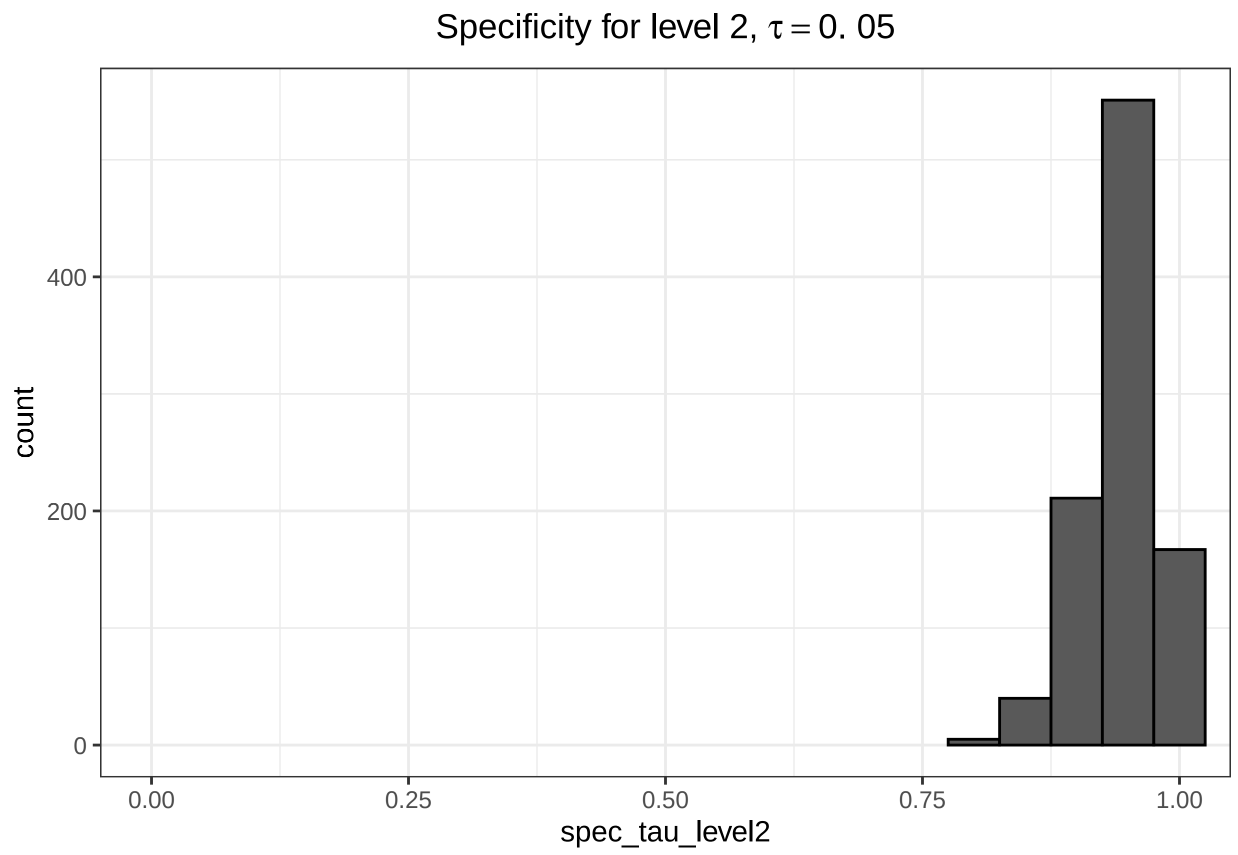
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##       1       1       1       1       1       1
```



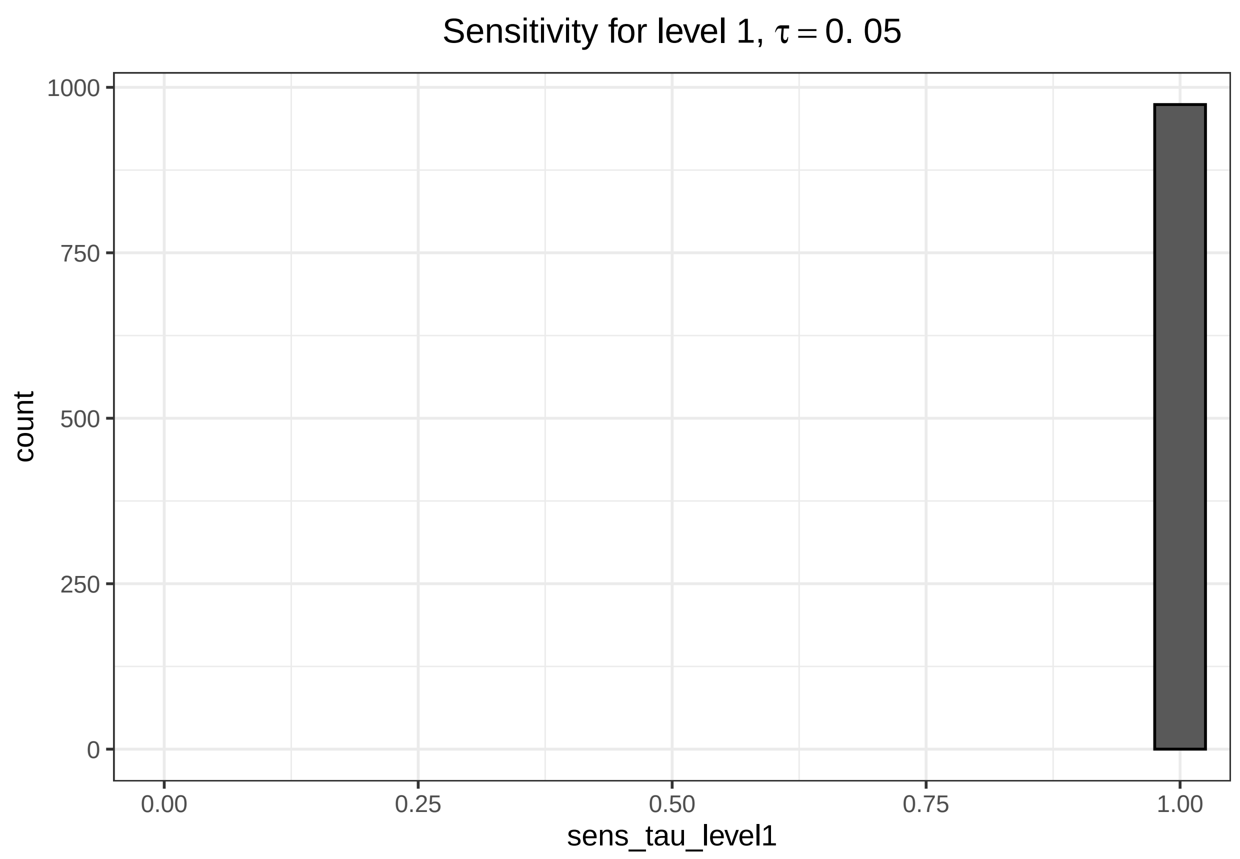
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.8095  1.0000  1.0000  0.9937  1.0000  1.0000
##
## --------------------------------------------------------------------------------
##
## Results for tau = 0.05
##
## Weights: [1] 2.000000e+00 3.609703e-35
```



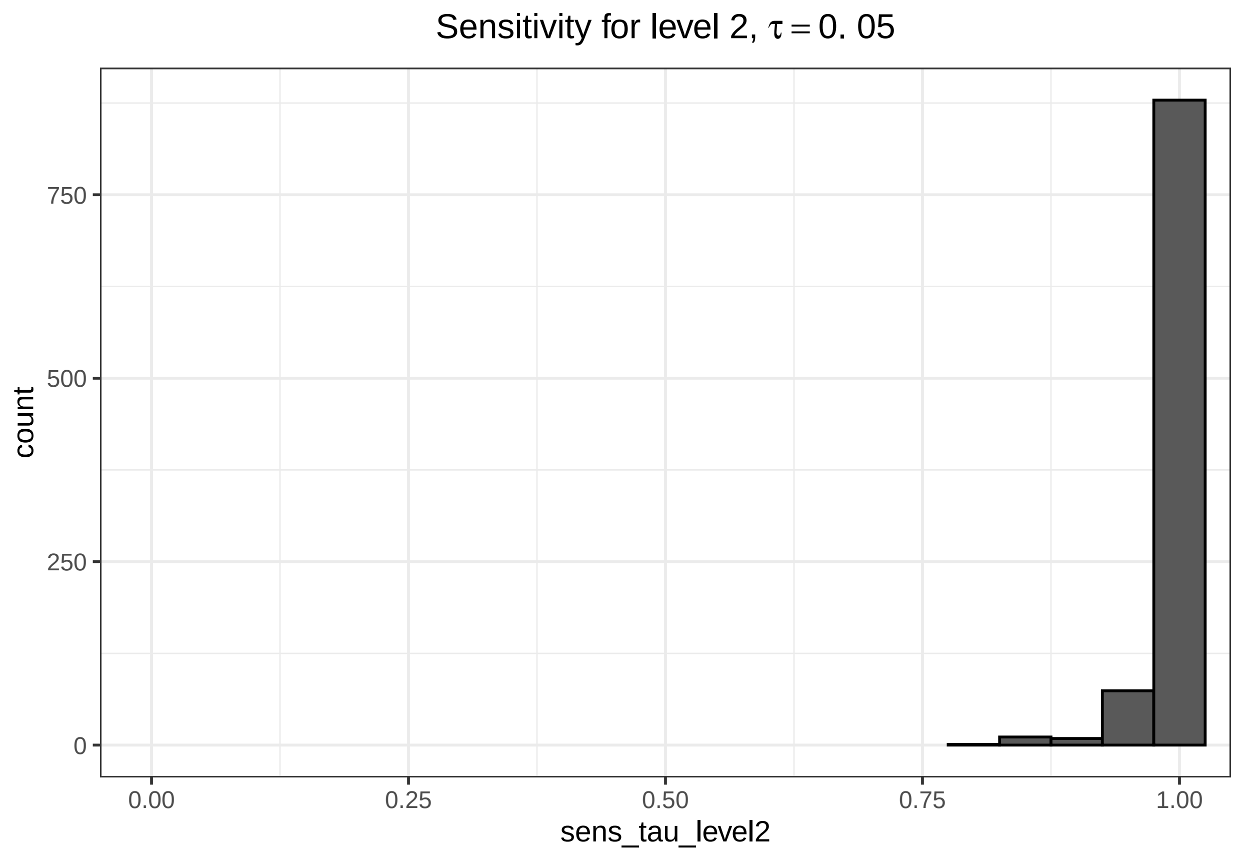
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.7798  0.9266  0.9633  0.9516  0.9817  1.0000
```



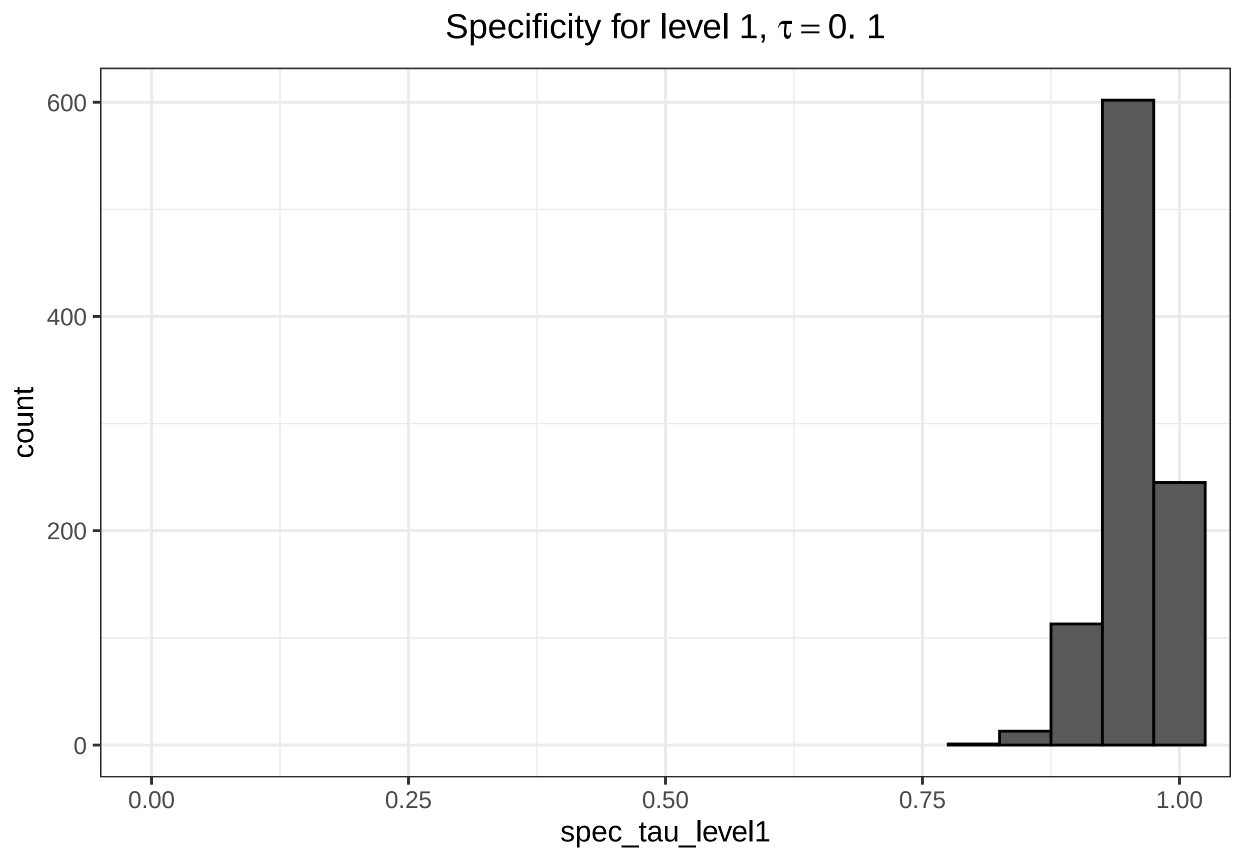
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.7975  0.9241  0.9747  0.9553  0.9747  1.0000
```



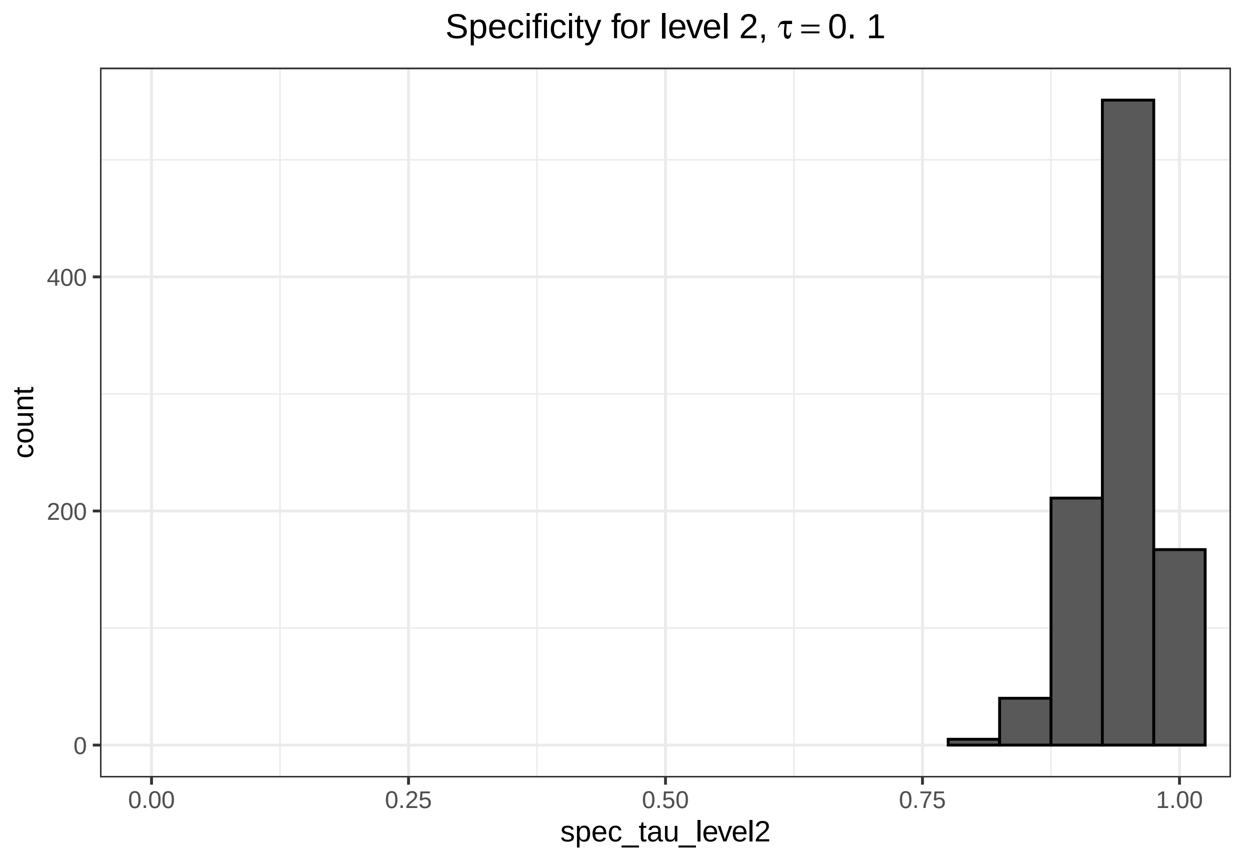
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       1       1       1       1       1       1
```



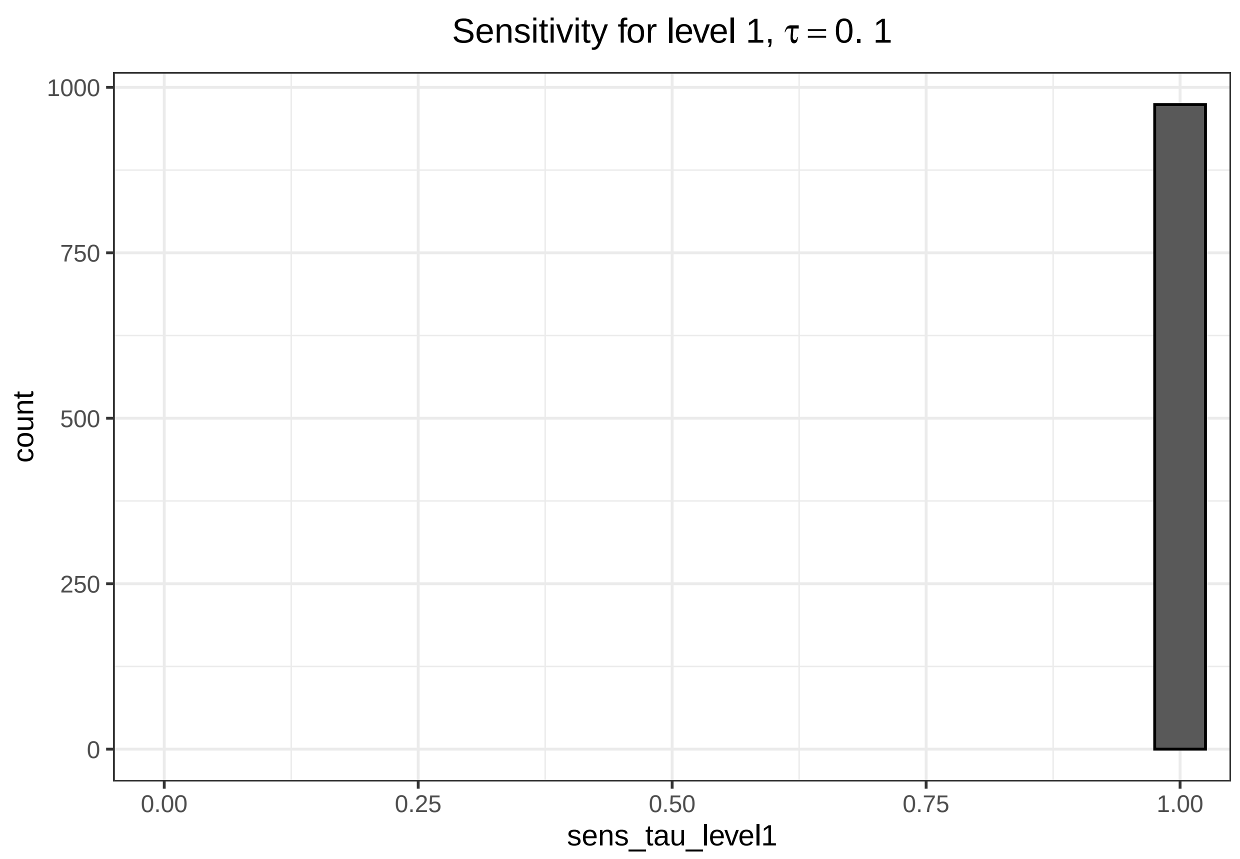
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.8095  1.0000  1.0000  0.9937  1.0000  1.0000
##
## --------------------------------------------------------------------------------
##
## Results for tau = 0.1
##
## Weights: [1] 2.000000e+00 4.122307e-09
```



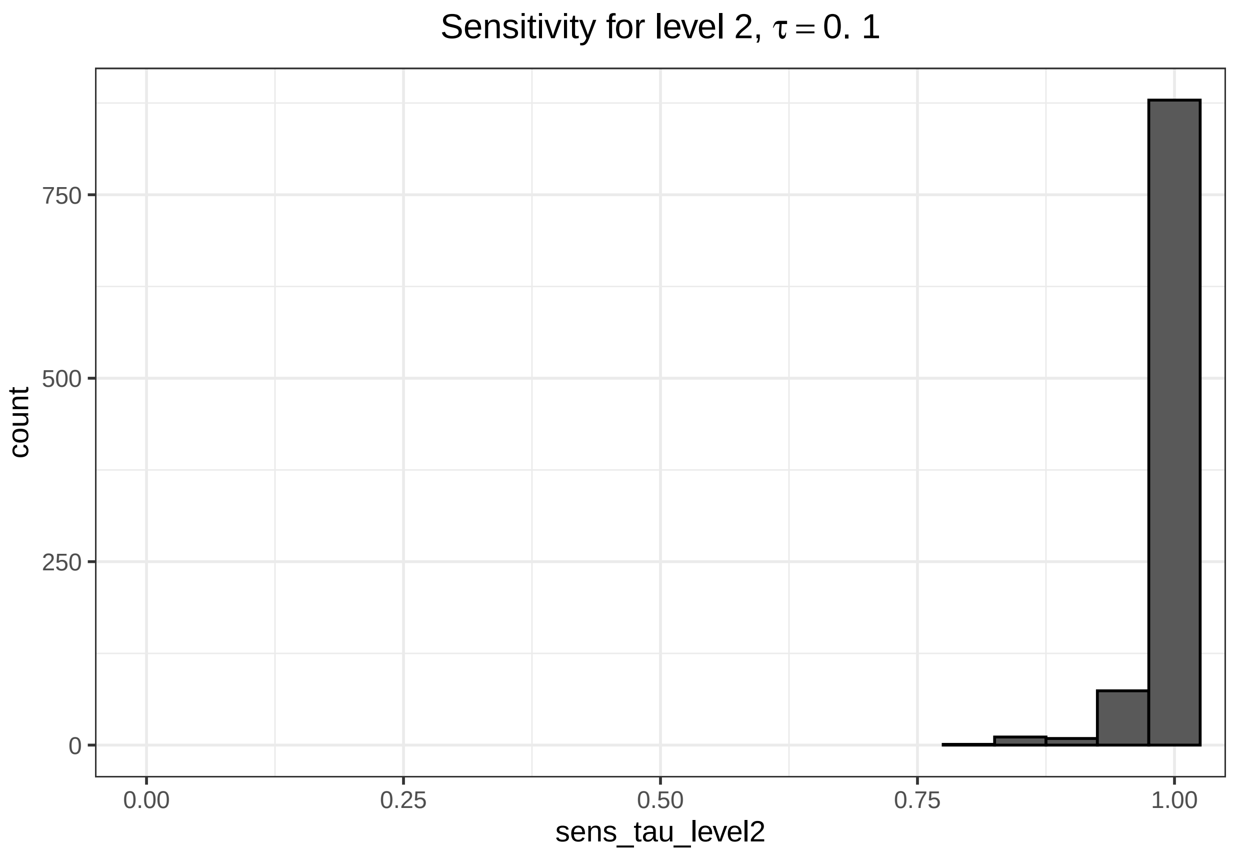
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.7798  0.9266  0.9633  0.9516  0.9817  1.0000
```



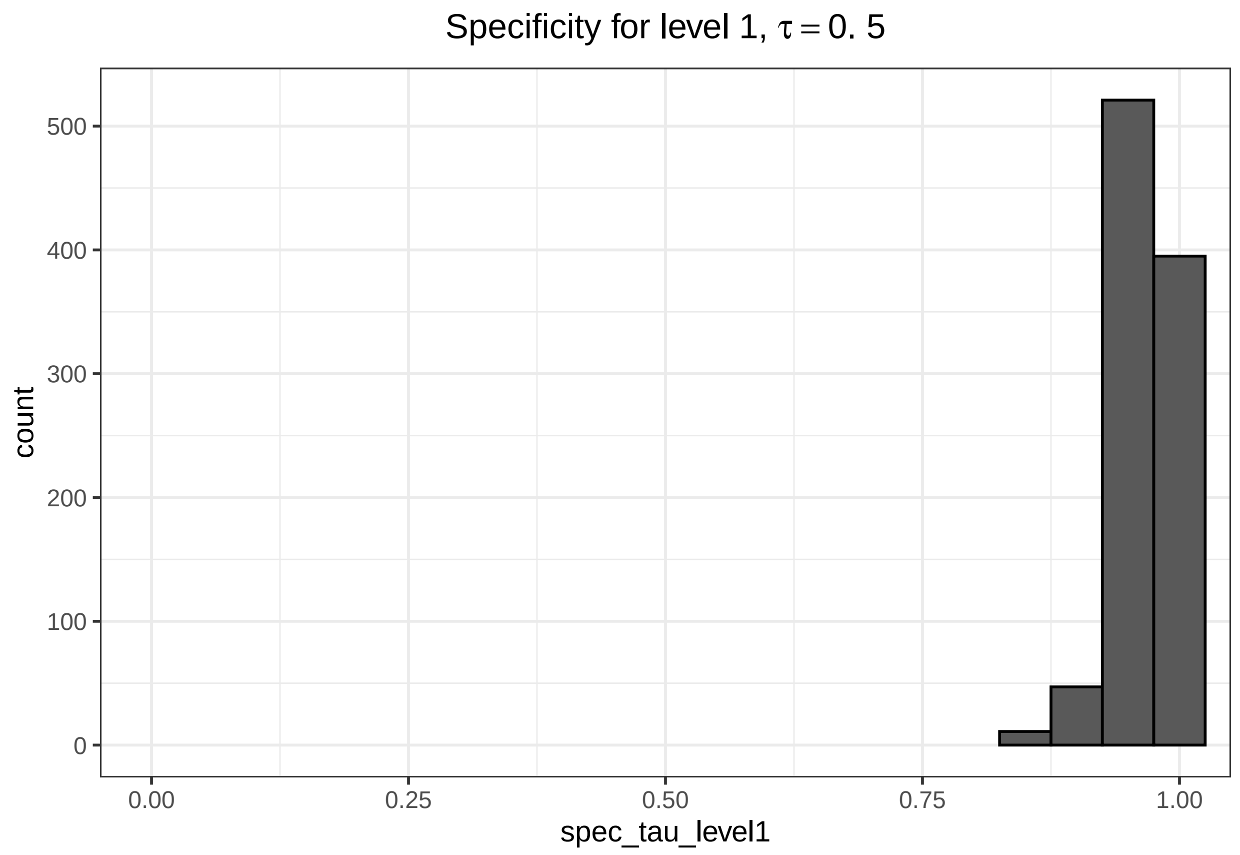
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.7975  0.9241  0.9747  0.9553  0.9747  1.0000
```



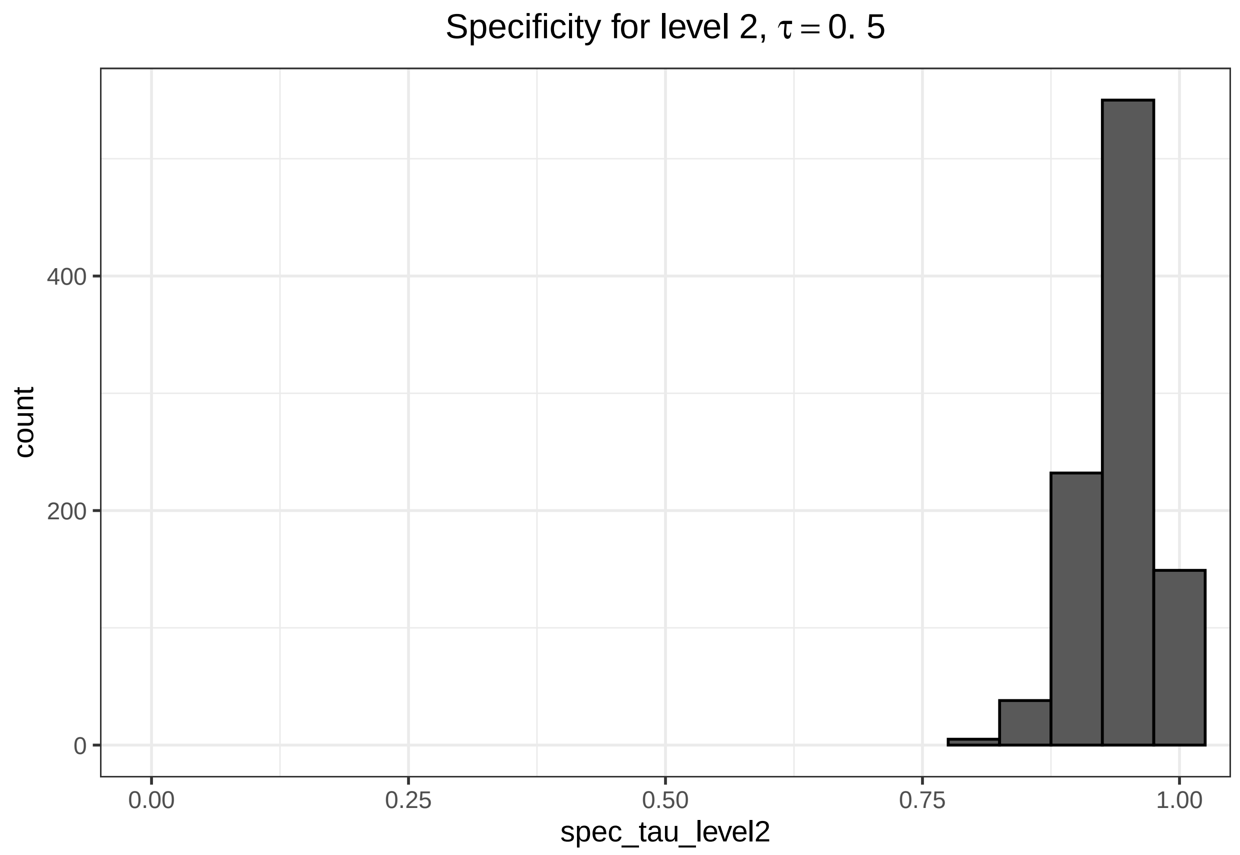
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       1       1       1       1       1       1
```



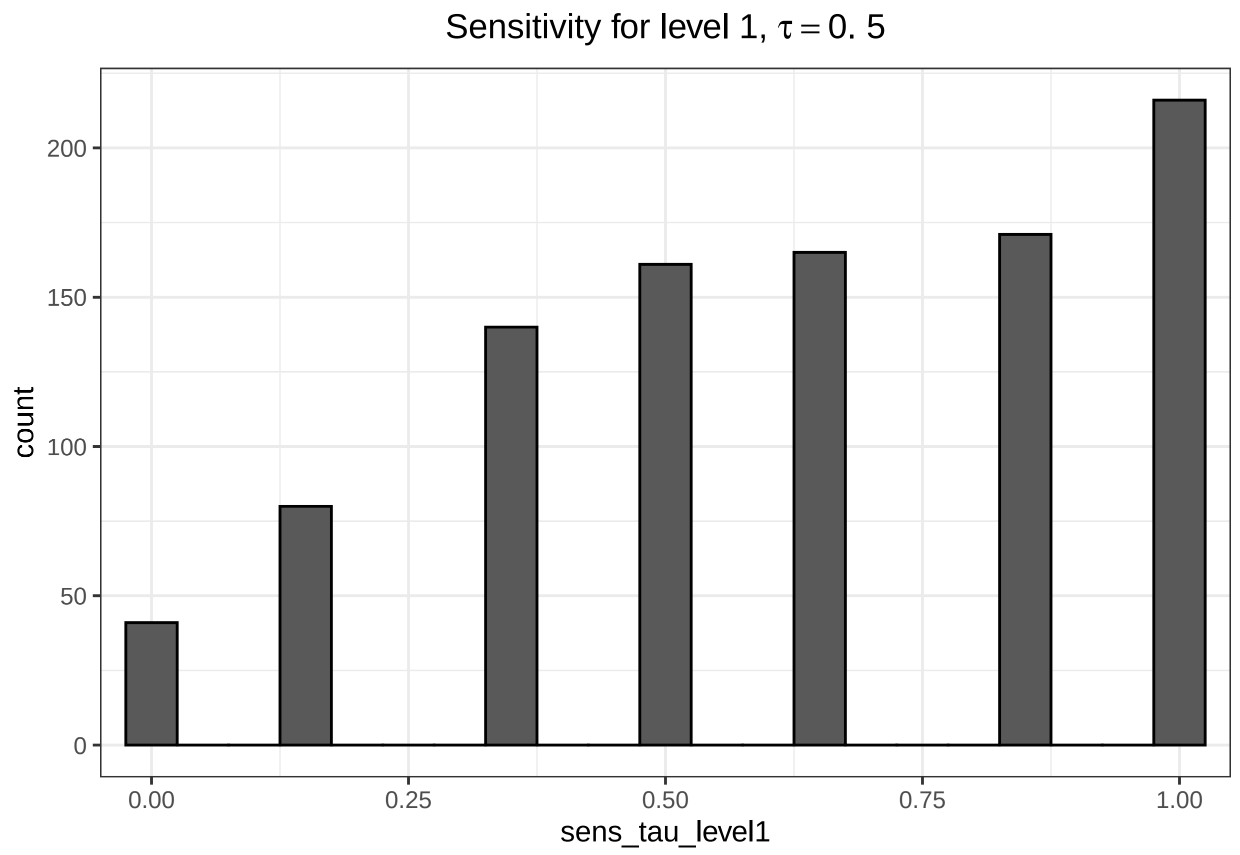
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.8095  1.0000  1.0000  0.9937  1.0000  1.0000
##
## --------------------------------------------------------------------------------
##
## Results for tau = 0.5
##
## Weights: [1] 1.379949 0.620051
```



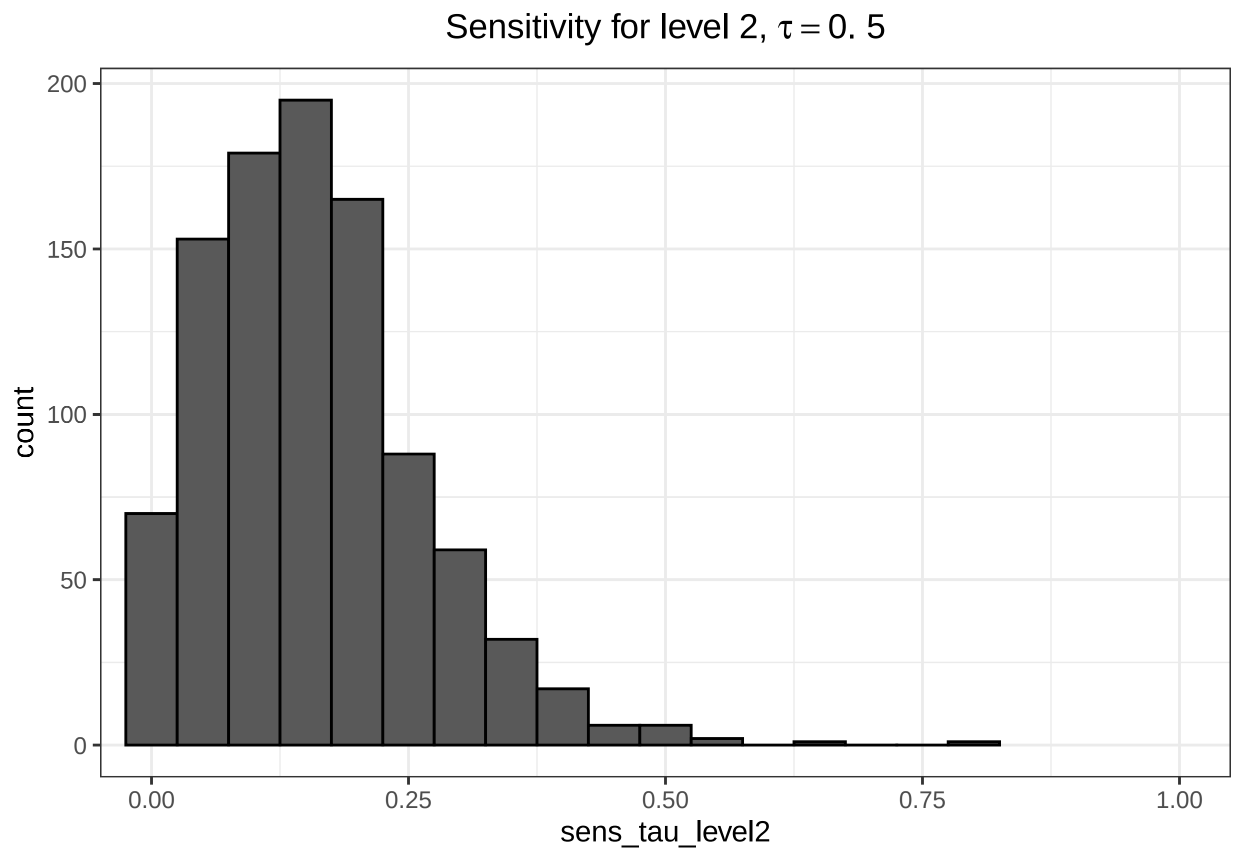
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.8349  0.9450  0.9633  0.9622  0.9817  1.0000
```



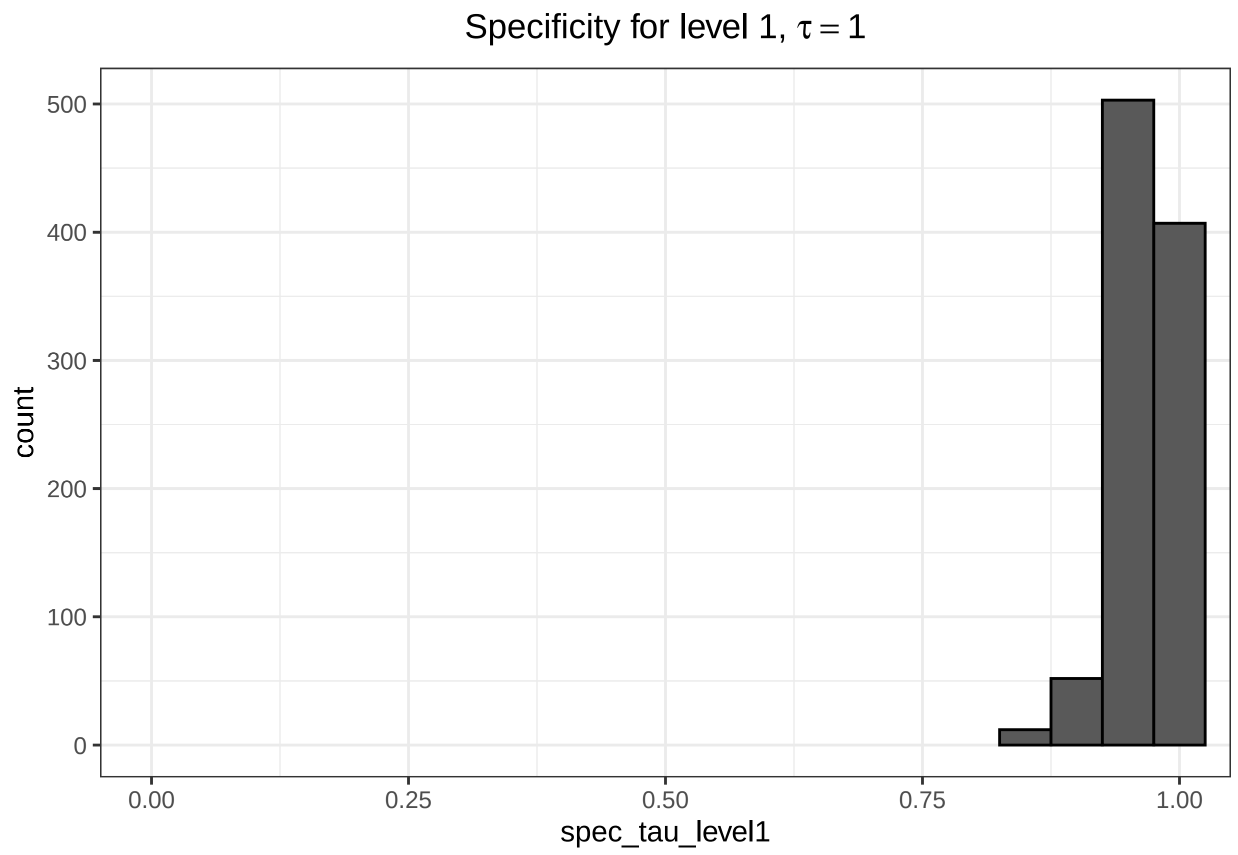
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.7975  0.9241  0.9494  0.9521  0.9747  1.0000
```



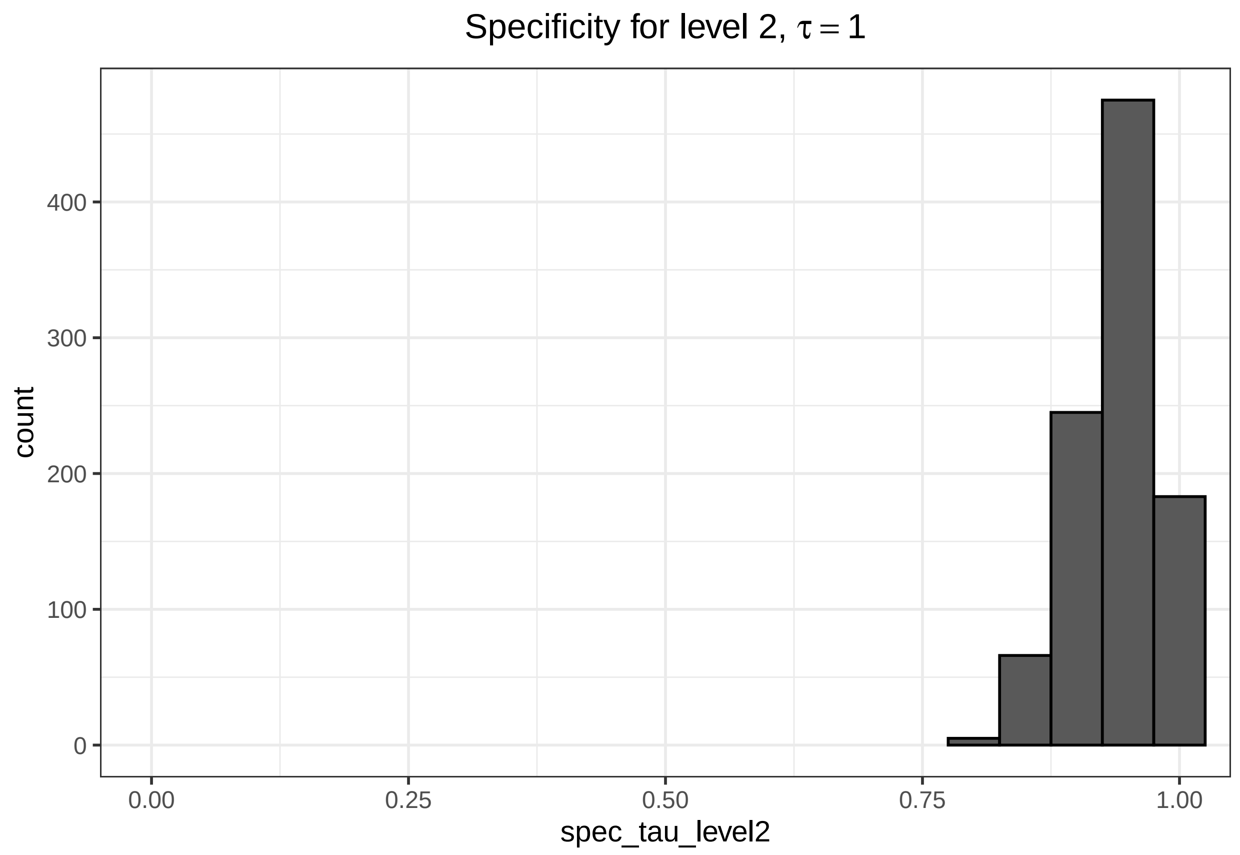
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.3333  0.6667  0.6253  0.8333  1.0000
```



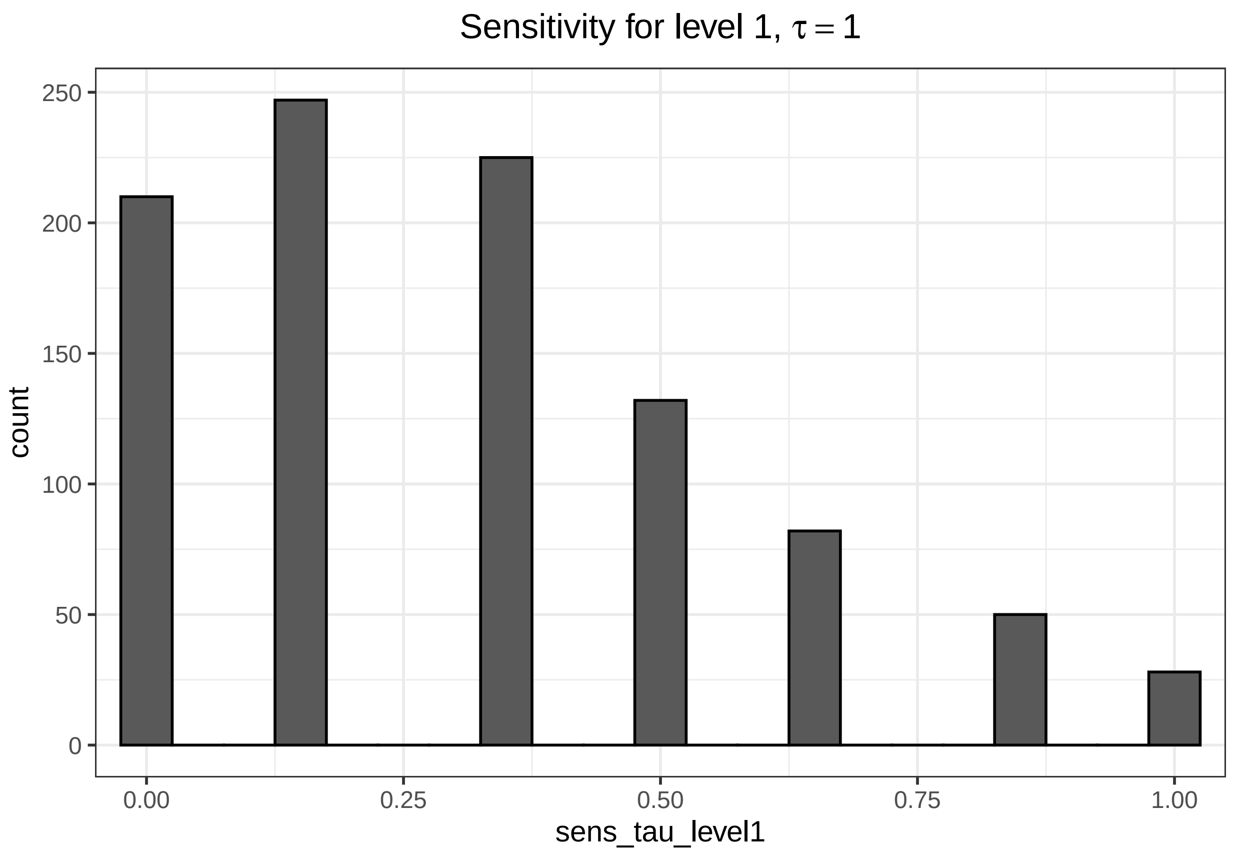
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.09524 0.14286 0.15063 0.19048 0.80952
##
## --------------------------------------------------------------------------------
##
## Results for tau = 1
##
## Weights: [1] 1.099668 0.900332
```



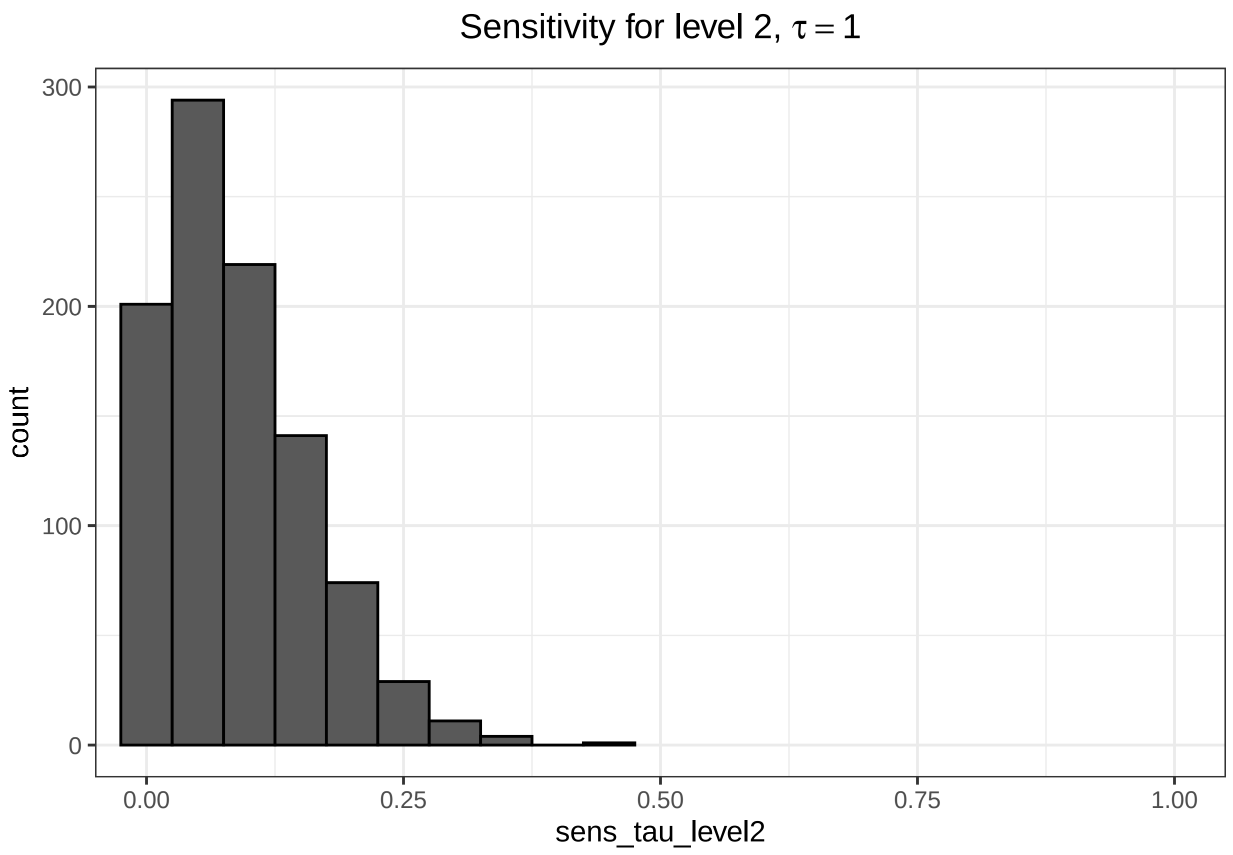
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.8349  0.9450  0.9633  0.9630  0.9817  1.0000
```



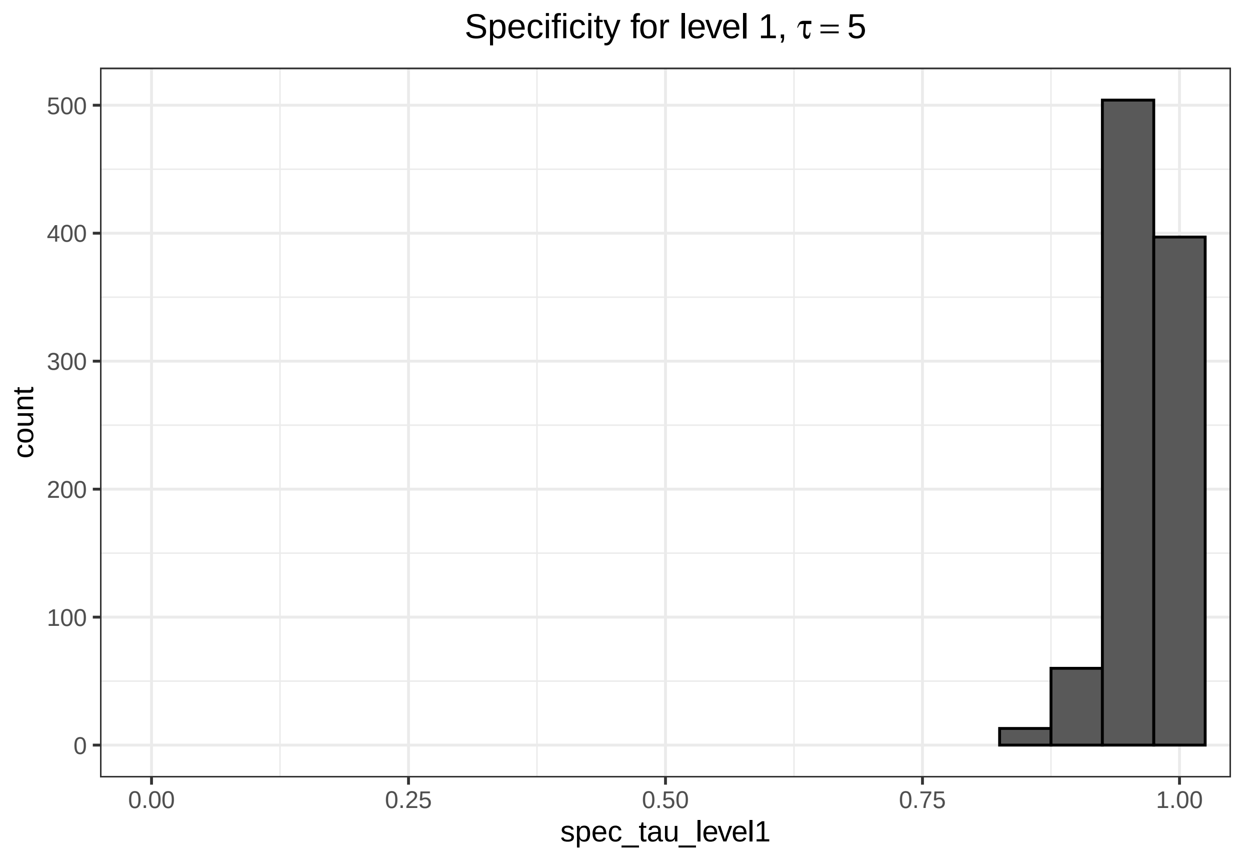
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.7975  0.9241  0.9494  0.9498  0.9747  1.0000
```



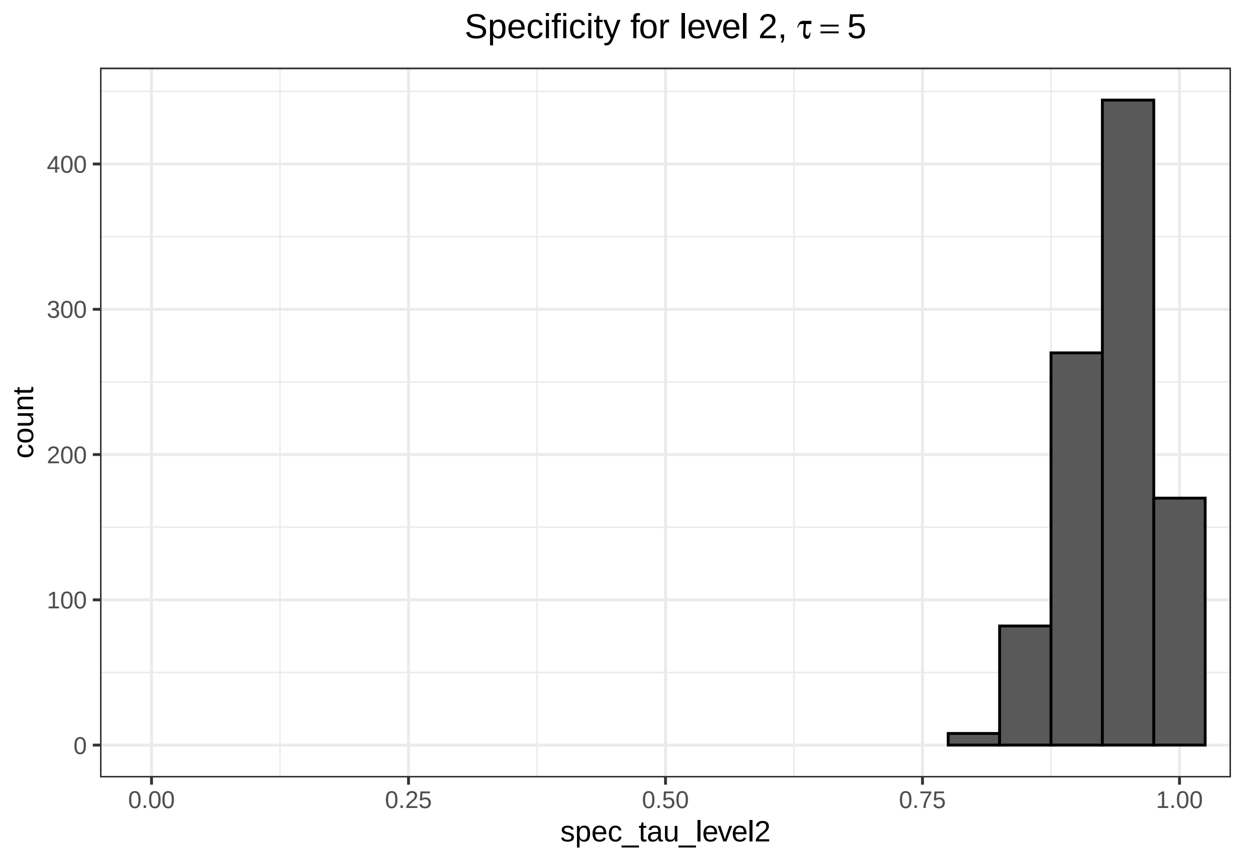
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.1667  0.3333  0.3147  0.5000  1.0000
```



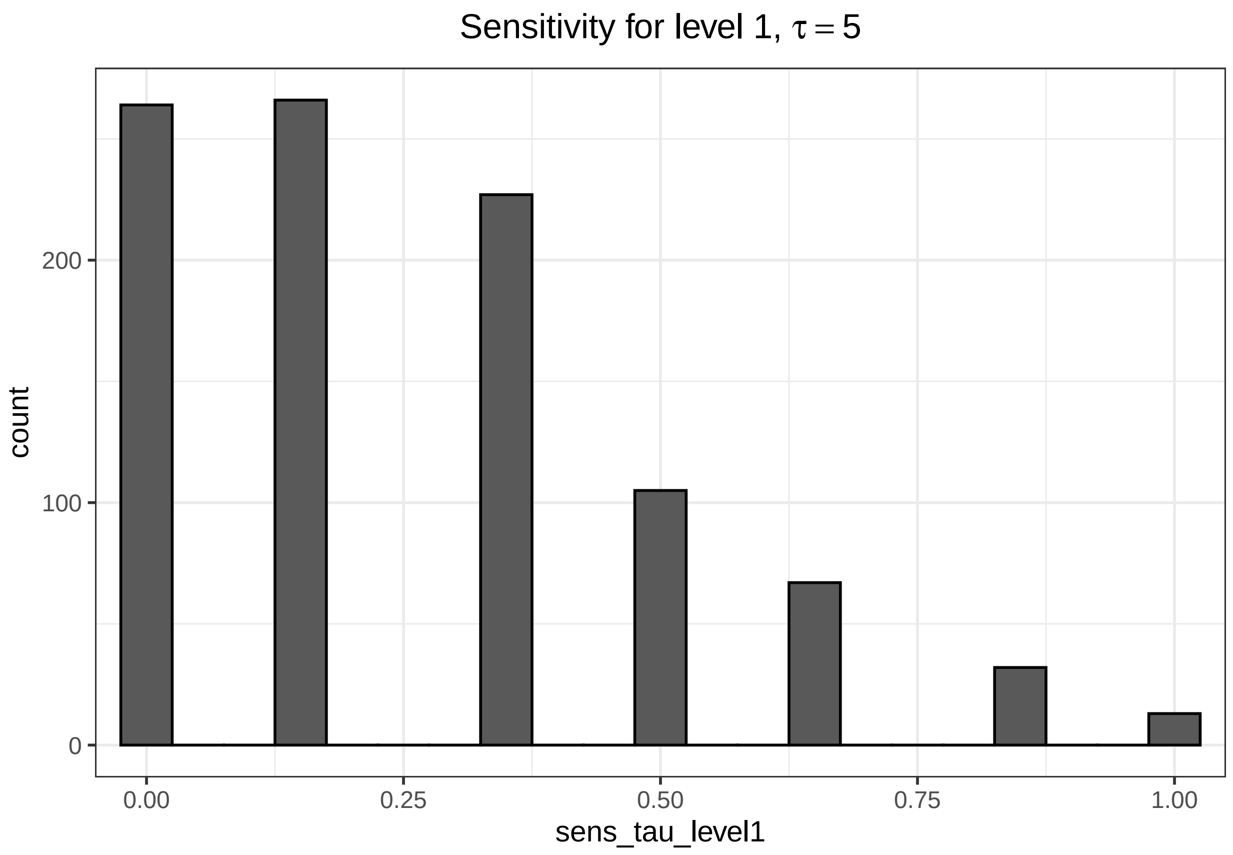
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.04762 0.04762 0.08306 0.14286 0.42857
##
## --------------------------------------------------------------------------------
##
## Results for tau = 5
##
## Weights: [1] 1.004 0.996
```



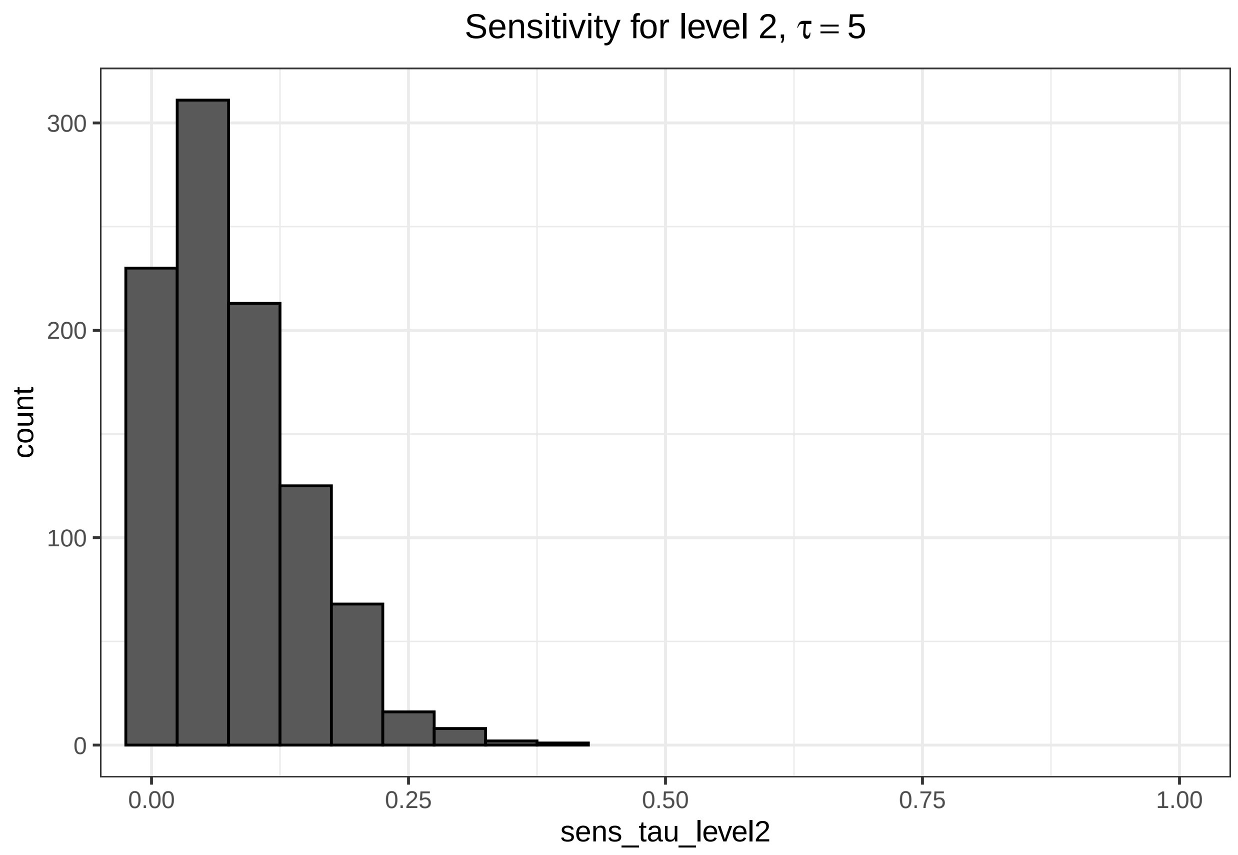
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.8349  0.9450  0.9633  0.9611  0.9817  1.0000
```



```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.8228  0.9241  0.9494  0.9463  0.9747  1.0000
```



```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.1667  0.2637  0.3333  1.0000
```



```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.04762 0.04762 0.07500 0.09524 0.38095
##
## --------------------------------------------------------------------------------
```