Define Algo Threshold + Bootstrapping

```
library("dplyr")
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

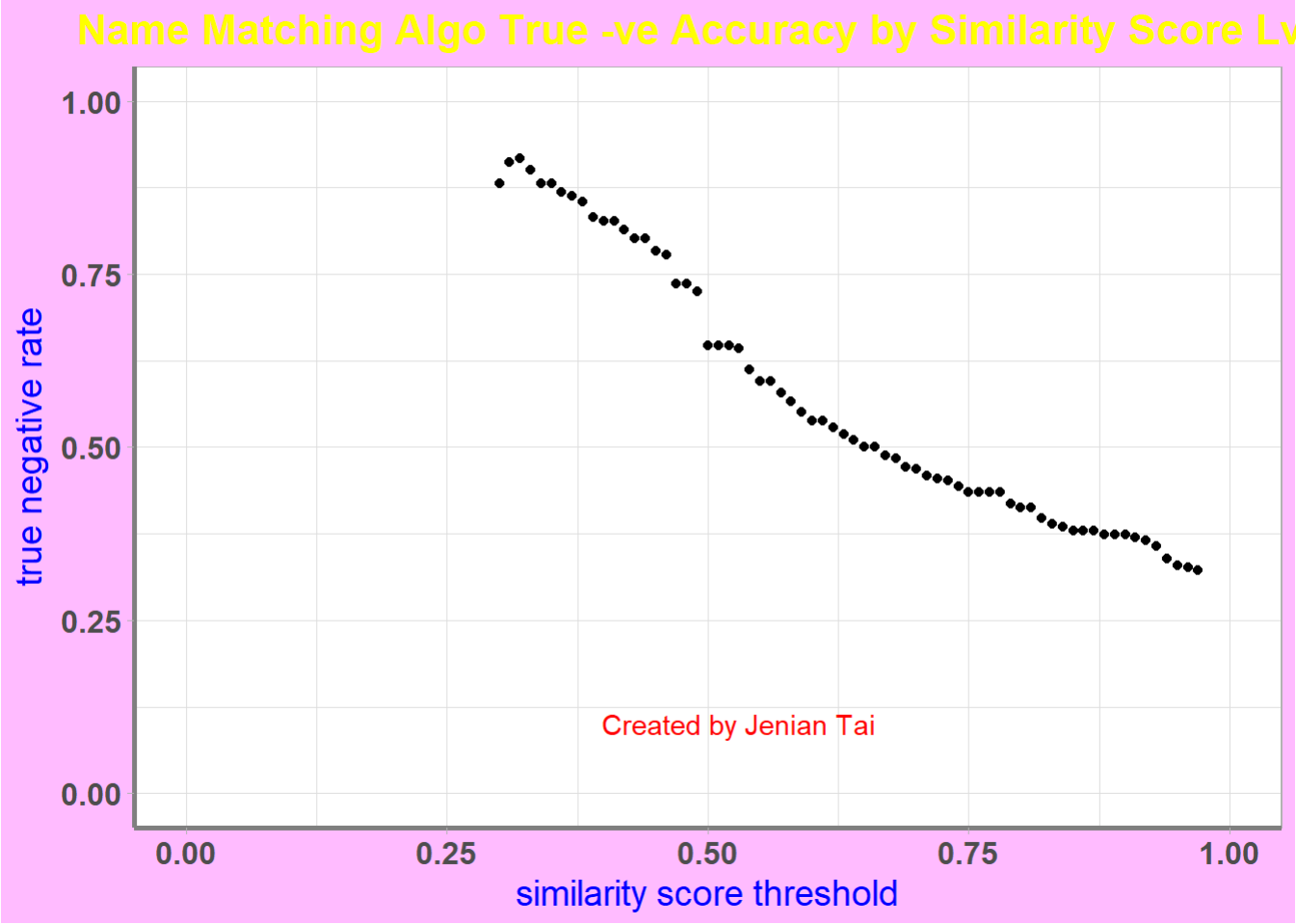
```
library("tidyr")
library("ggplot2")

names = read.csv("Z:/SharData/Research and Strategic Analysis/Requests/20180523_Rent_Stab_Name_Matching_Research/20180529_sample_data_name_matches.csv",
                 stringsAsFactors = FALSE)

str(names)
```

```
## 'data.frame':    1244 obs. of  5 variables:
##  $ X         : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ stringdist: int  1 1 1 1 1 1 1 1 1 1 ...
##  $ length    : int  40 38 32 32 31 29 29 29 28 28 ...
##  $ similarity: num  0.975 0.974 0.969 0.969 0.968 ...
##  $ match     : chr  "TRUE" "TRUE" "TRUE" "TRUE" ...
```

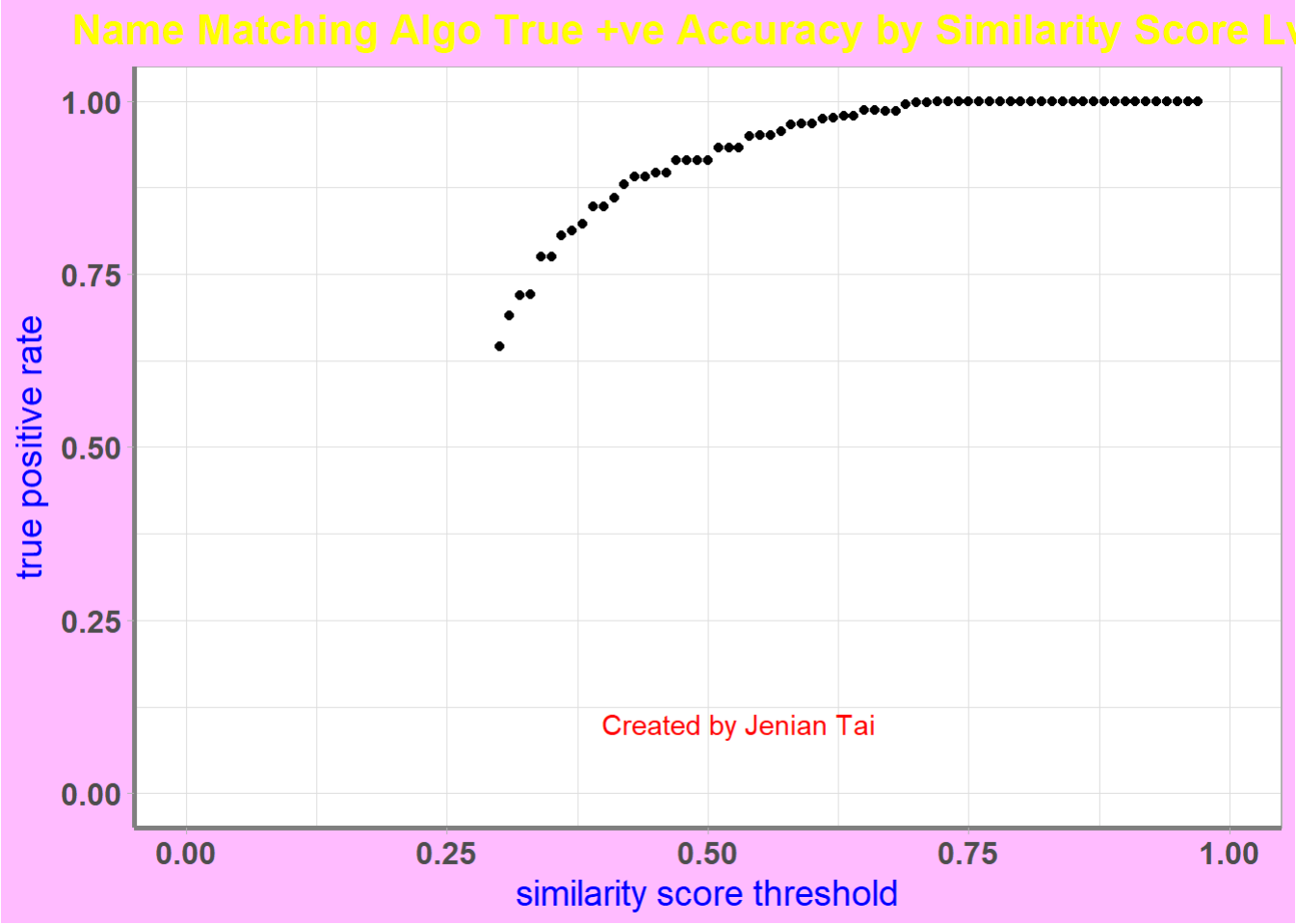
```
table(names$match)
```

```
##
##   FALSE    TRUE UNKNOWN
##     401     803      40
```

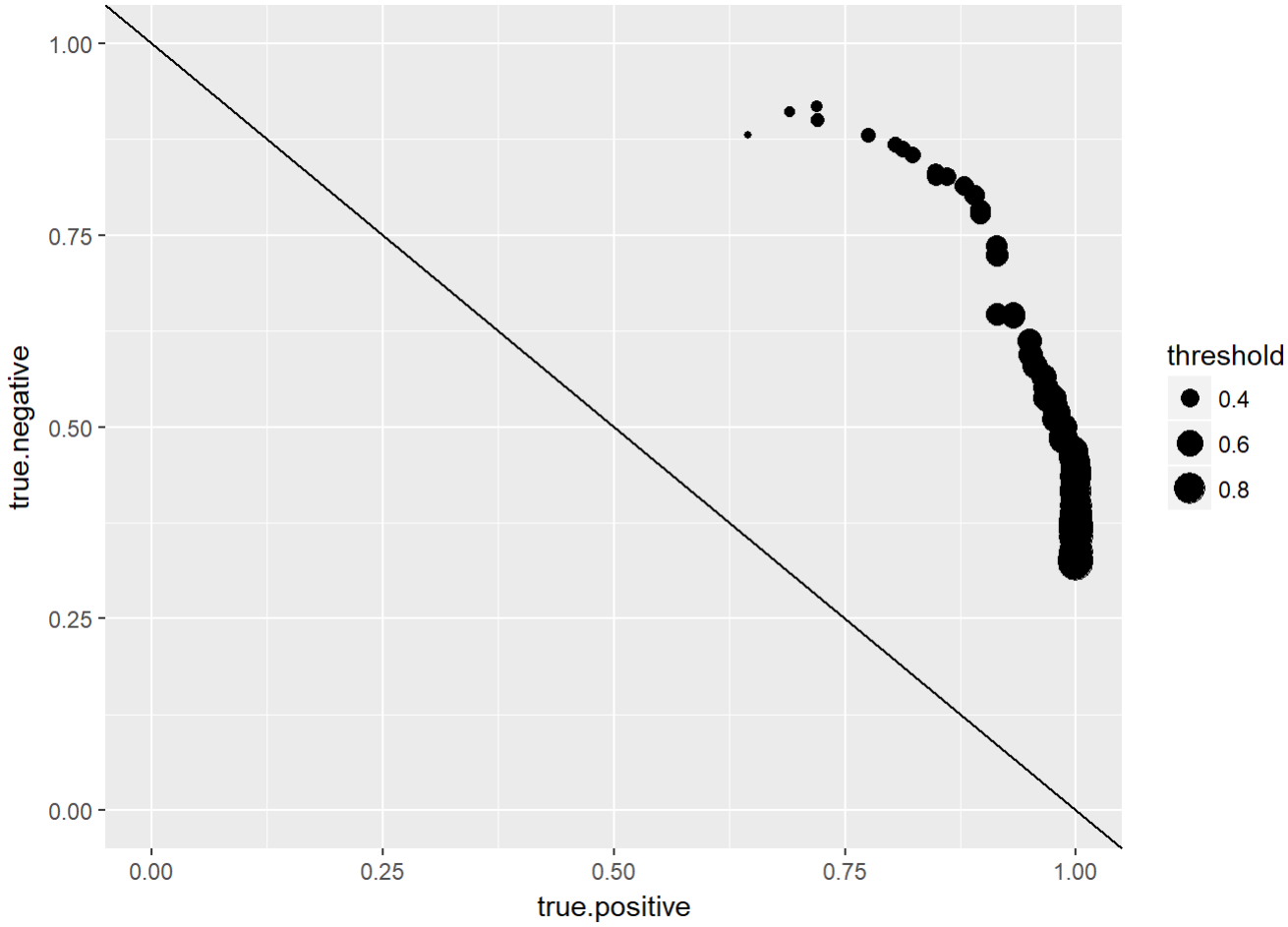
```
#
#   FALSE    TRUE UNKNOWN
#   401     803      40

min.score = min(names$similarity)
max.score = max(names$similarity)

# initiate a range of similarity scale with increment of 0.01
sim.range = seq(min.score, max.score, .01)
# create empty array for match rates
results = c()
results2 =c()

# return true positive and true negative match rates
for (i in sim.range){

  #match.rate = rate of true-positive out of total obs., which is >= each similarity threshold
  temp.df = subset(names, names$similarity >= i)
  temp.list = temp.df$match[temp.df$match==TRUE]
  match.rate =  length(temp.list) / length(temp.df$X)
  results = c(results, match.rate)

  #match.rate2 = rate of true-negative which is < each similarity threshold
  temp.df2 = subset(names, names$similarity <= i)
  temp.list2 = temp.df2$match[temp.df2$match==FALSE]
  match.rate2 =  length(temp.list2) / length(temp.df2$X)
  results2 = c(results2, match.rate2)

}

# store results of match rates and assign column names
df.results = as.data.frame(cbind(sim.range, results, results2))
names(df.results) = c("threshold","true.positive","true.negative")


# plot of true negative match rates
ggplot(df.results, aes(x=threshold, y=true.negative)) +
  geom_point()+
  scale_x_continuous(limits=c(0,1)) +
  scale_y_continuous(limits=c(0,1)) +
  labs(y="true negative rate", x="similarity score threshold")+
  ggtitle("Name Matching Algo True -ve Accuracy by Similarity Score Lvls")+
  theme_light()+
  theme(plot.title = element_text(size=16, face="bold", colour = "yellow", hjust=.5))+
  theme(axis.title.x = element_text(size=14,  colour="blue"))+
  theme(axis.title.y = element_text(size=14,  colour="blue"))+
  theme(axis.text.x = element_text(size=12,  face="bold"))+
  theme(axis.text.y = element_text(size=12,  face="bold"))+
  theme(axis.line = element_line(colour="grey50", size=1))+
  theme(plot.background = element_rect(fill="plum1"))+
  annotate("text", x=.53, y=0.1, label = "Created by Jenian Tai", colour="red")
```




```
# plot of true positive match rates
ggplot(df.results, aes(x=threshold, y=true.positive)) +
  geom_point()+
  scale_x_continuous(limits=c(0,1)) +
  scale_y_continuous(limits=c(0,1)) +
  labs(y="true positive rate", x="similarity score threshold")+
  ggtitle("Name Matching Algo True +ve Accuracy by Similarity Score Lvls")+
  theme_light()+
  theme(plot.title = element_text(size=16, face="bold", colour = "yellow", hjust=.5))+
  theme(axis.title.x = element_text(size=14,  colour="blue"))+
  theme(axis.title.y = element_text(size=14,  colour="blue"))+
  theme(axis.text.x = element_text(size=12,  face="bold"))+
  theme(axis.text.y = element_text(size=12,  face="bold"))+
  theme(axis.line = element_line(colour="grey50", size=1))+
  theme(plot.background = element_rect(fill="plum1"))+
  annotate("text", x=.53, y=0.1, label = "Created by Jenian Tai", colour="red")
```



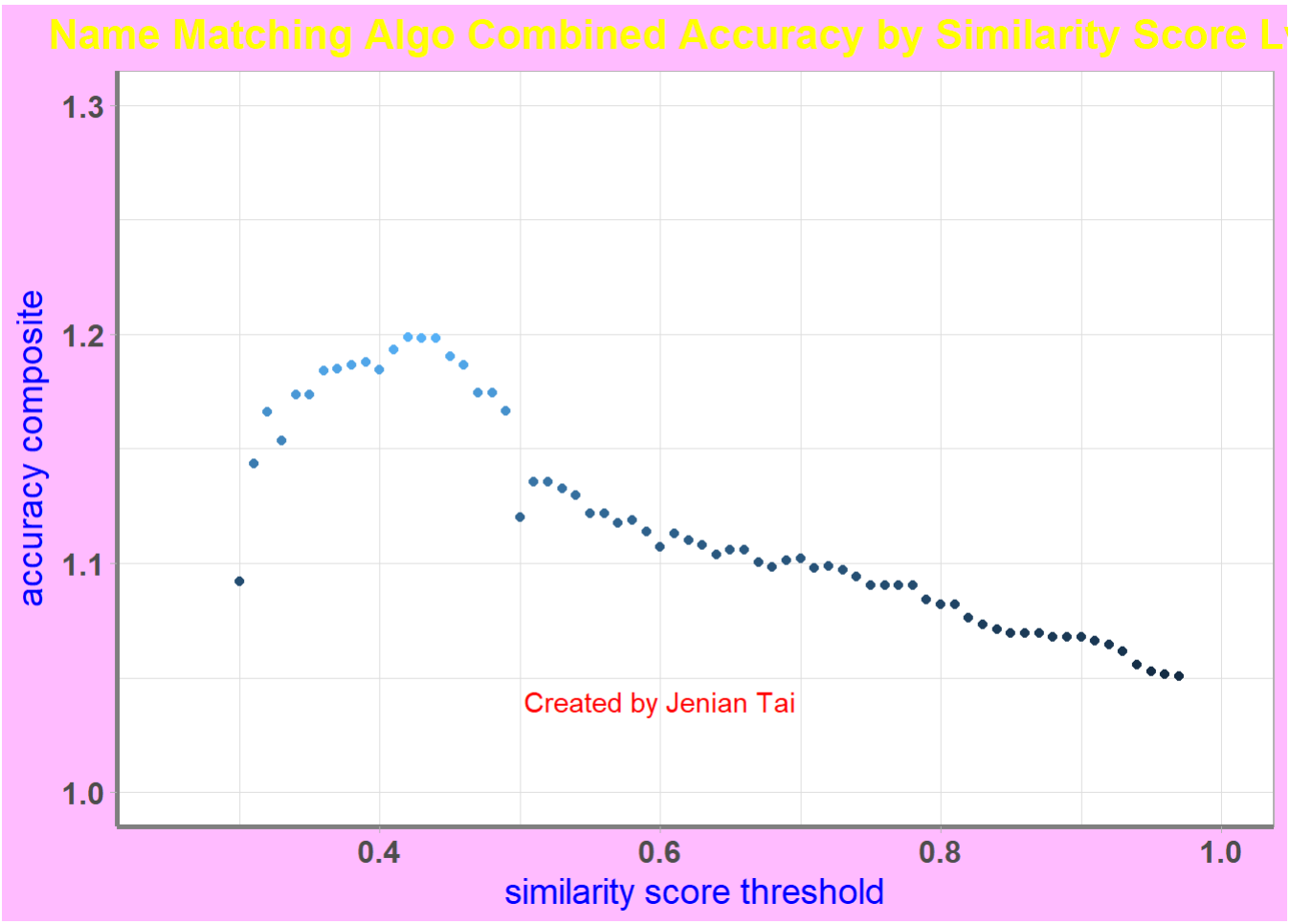

```
# plot of positve accuracy against negative accuracy
ggplot(df.results, aes(x=true.positive, y=true.negative)) +
  geom_point(aes(size=threshold))+
  scale_x_continuous(limits=c(0,1)) +
  scale_y_continuous(limits=c(0,1)) +
  geom_abline(intercept = 1, slope=-1)
```



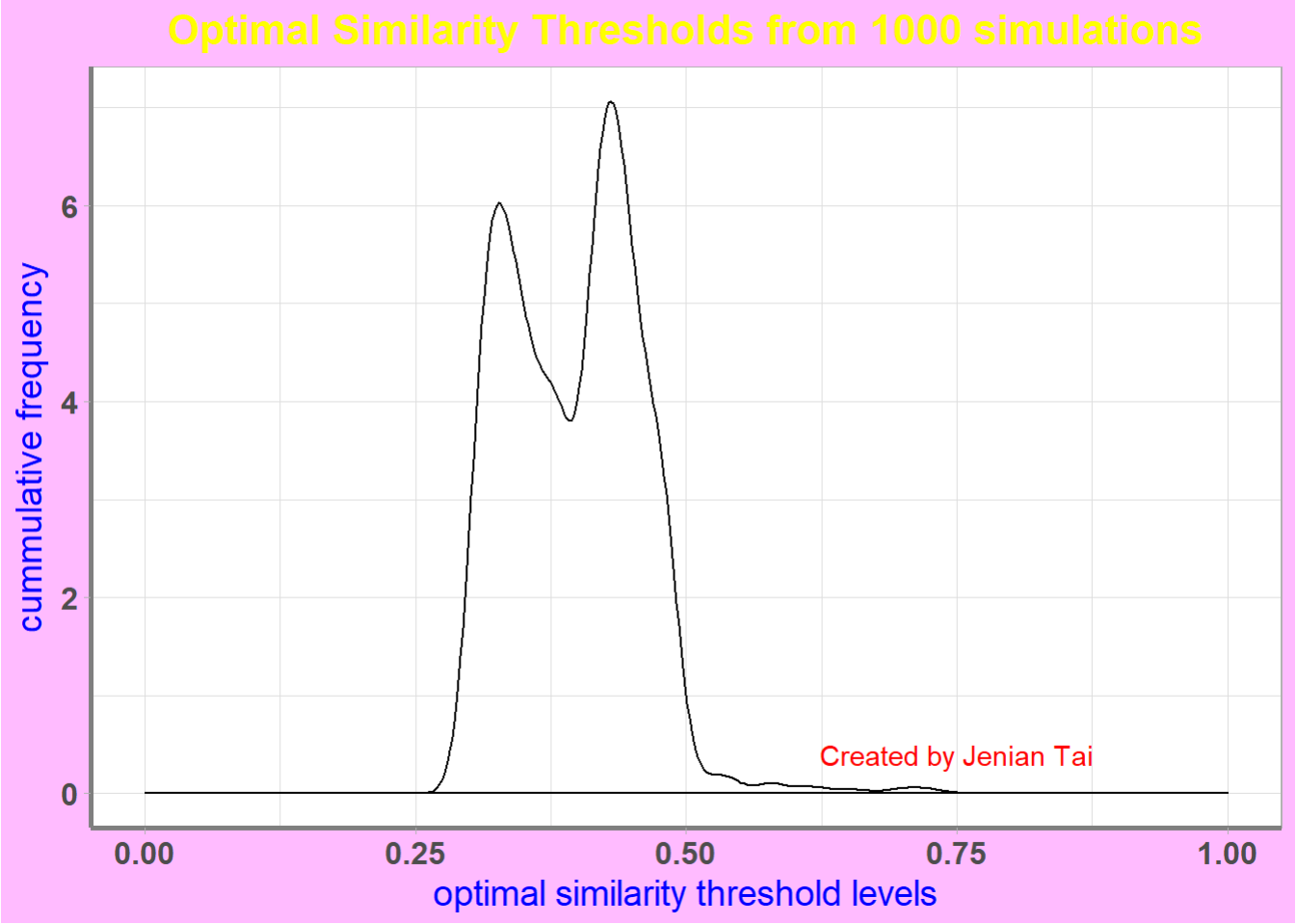

```
#there is a diminishing marginal-benefit to increasing the threshold for the true-positive
#there is a approximately constant/linear loss to increasing the threshold for the true-negative
#what is the optimal threshold for maximizing the true-positive/true-negative???
#what is the practical significance of a false negative or a false positive???

# calculate distance from the origin (0,0), store in new column "composite"
df.results <- na.omit(df.results)
df.results$composite = sqrt(df.results$true.positive**2+df.results$true.negative**2)
df.results$threshold[df.results$composite==max(df.results$composite)]
```

```
## [1] 0.42
```

```
# scatter plot of max accuracy
ggplot(df.results, aes(x=threshold, y=composite)) +
  geom_point(aes(color=composite))+
  scale_x_continuous(limits=c(.25,1)) +
  scale_y_continuous(limits=c(1,1.3)) +
  labs(y="accuracy composite", x="similarity score threshold")+
  ggtitle("Name Matching Algo Combined Accuracy by Similarity Score Lvls")+
  theme_light()+
  theme(plot.title = element_text(size=16, face="bold", colour = "yellow", hjust=.5))+
  theme(axis.title.x = element_text(size=14,  colour="blue"))+
  theme(axis.title.y = element_text(size=14,  colour="blue"))+
  theme(axis.text.x = element_text(size=12,  face="bold"))+
  theme(axis.text.y = element_text(size=12,  face="bold"))+
  theme(axis.line = element_line(colour="grey50", size=1))+
  theme(plot.background = element_rect(fill="plum1"))+
  theme(legend.position="none")+
  annotate("text", x=.6, y=1.04, label = "Created by Jenian Tai", colour="red")
```



```
# create empty array for maximum composites/accuracies
optimal.threshold = c()

# bootstrapping with 1000 replications each with sample size of 100
for (k in 1:1000){
  resultsb = c()
  results2b = c()
  set.seed(k)
  names.sub = sample_n(names, 100)

  for (i in sim.range){

    #match.rate = rate of true-positive
    temp.df = subset(names.sub, names.sub$similarity>= i)
    temp.list = temp.df$match[temp.df$match==TRUE]
    match.rate =  length(temp.list) / length(temp.df$X)
    resultsb = c(resultsb, match.rate)

    #match.rate2 = rate of true-negative
    temp.df2 = subset(names.sub, names.sub$similarity <= i)
    temp.list2 = temp.df2$match[temp.df2$match==FALSE]
    match.rate2 =  length(temp.list2) / length(temp.df2$X)
    results2b = c(results2b, match.rate2)
  }
  temp.results = as.data.frame(cbind(sim.range, resultsb, results2b))
  names(temp.results) = c("threshold","true.positive","true.negative")
  temp.results$composite = sqrt(temp.results$true.positive**2+temp.results$true.negative**2)
  max.accuracy = max(na.omit(temp.results$composite))

  optimal.threshold = c(optimal.threshold,(((temp.results$threshold[temp.results$composite == ma
x.accuracy])))))
}


# plot the density/cumulative accuracy rates of thresholds
opts = as.numeric(na.omit(optimal.threshold))
ggplot() + geom_density(aes(x=opts)) +
  scale_x_continuous(limits=c(0,1))+
  labs(y="cummulative frequency", x="optimal similarity threshold levels")+
  ggtitle("Optimal Similarity Thresholds from 1000 simulations")+
  theme_light()+
  theme(plot.title = element_text(size=16, face="bold", colour = "yellow", hjust=.5))+
  theme(axis.title.x = element_text(size=14,  colour="blue"))+
  theme(axis.title.y = element_text(size=14,  colour="blue"))+
  theme(axis.text.x = element_text(size=12,  face="bold"))+
  theme(axis.text.y = element_text(size=12,  face="bold"))+
  theme(axis.line = element_line(colour="grey50", size=1))+
  theme(plot.background = element_rect(fill="plum1"))+
  theme(legend.position="none")+
  annotate("text", x=.75, y=0.4, label = "Created by Jenian Tai", colour="red")
```




```
# find peak value in desity plot
density(opts)$x[which.max(density(opts)$y)]
```

```
## [1] 0.4304316
```

```
quantile(opts, seq(0,1,.1))
```

```
##   0%  10%  20%  30%  40%  50%  60%  70%  80%  90% 100%
## 0.30 0.32 0.33 0.35 0.37 0.40 0.42 0.43 0.45 0.47 0.73
```

```
sqrt(var(opts))
```

```
## [1] 0.06135743
```