AUC Analysis

Stefan Hillmann

6. Oktober 2015

```
library(RMongo)
```

```
## Loading required package: rJava
```

```
library(ggplot2)
library(plyr)
library(reshape2)
library(knitr)
```

```
# Multiple plot function
#
# from: http://www.cookbook-r.com/Graphs/Multiple_graphs_on_one_page_%28ggplot2%29/
#
# ggplot objects can be passed in ..., or to plotlist (as a list of ggplot objects)
# - cols:   Number of columns in layout
# - layout: A matrix specifying the layout. If present, 'cols' is ignored.
#
# If the layout is something like matrix(c(1,2,3,3), nrow=2, byrow=TRUE),
# then plot 1 will go in the upper left, 2 will go in the upper right, and
# 3 will go all the way across the bottom.
#
multiplot <- function(..., plotlist=NULL, file, cols=1, layout=NULL, title="") {
  library(grid)
  library(gridExtra)

  # Make a list from the ... arguments and plotlist
  plots <- c(list(...), plotlist)

  numPlots = length(plots)

  if (numPlots==1) {
    print(plots[[1]])

  } else {
    do.call("grid.arrange", c(plots, ncol=cols, nrow=ceiling(numPlots/cols), top=title))
  }
}
```

Load data

```
cross_validation <- mongoDbConnect("classification_cross_validation", "localhost", 27017)
performance <- dbGetQuery(cross_validation, "performance", '{}', 0, 0)
```

```
colormap <- c("#3B4CC0", "#445ACC", "#4D68D7", "#5775E1", "#6282EA", "#6C8EF1", "#779AF7", "#82A5FB",
```

```
discrete_cm <- function(count) {

  c = 0
  if(is.numeric(count) & length(count) == 1) {
    c = count
  }

  if(length(count) > 1) {
    c = length(count)
  }

  colors <- colormap[round(seq(1, length(colormap), length.out = c))]
  return(colors)
}

# To use for fills, add
# scale_fill_manual(values=colormap)

# To use for line and point colors, add
# scale_colour_manual(values=colormap)
```

Filter just one half of the symmetric (regarding to AUC) data

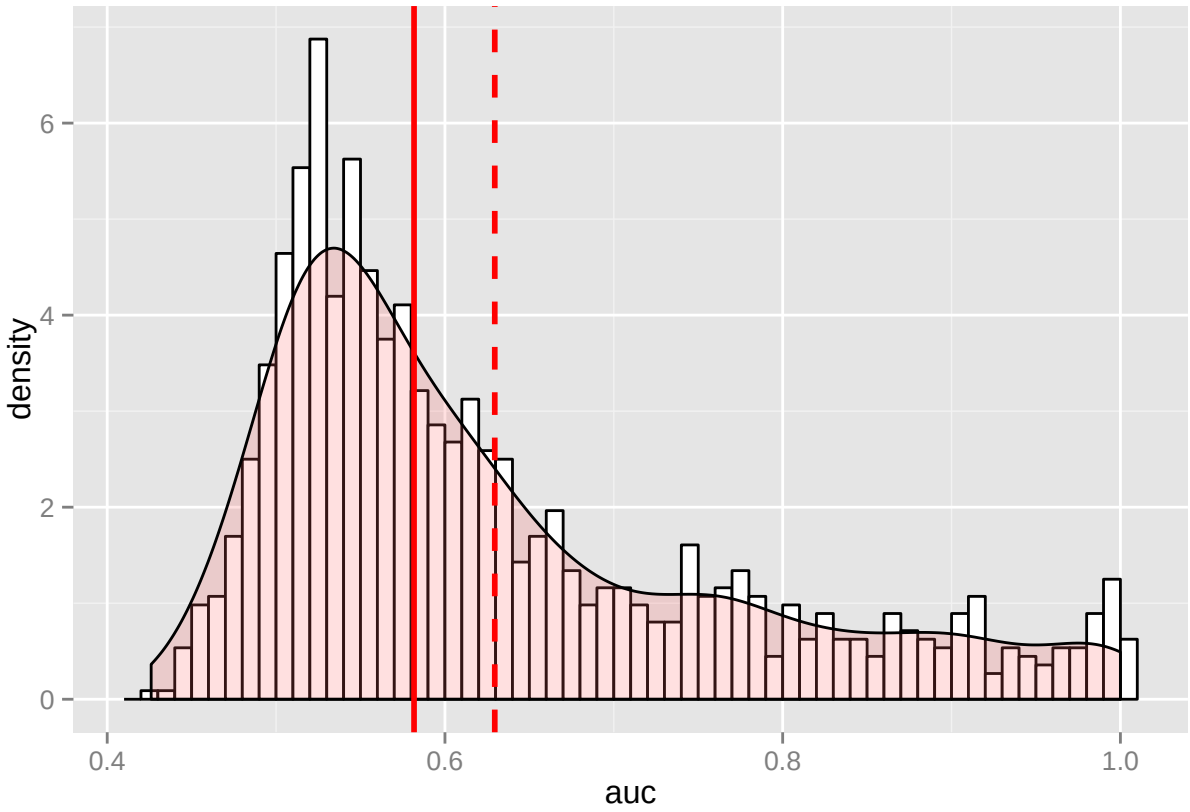
```
#   juged.data <- pr.complete[which(pr.complete$criteria %in% c('juged_bad', 'juged_good')), ]
#   interact_length.data <- pr.complete[which(pr.complete$criteria %in% c('short_interactions', 'long_i
#   real_simulated.data <- pr.complete[which(pr.complete$criteria %in% c('real', 'simulated')), ]
#   success.data <- pr.complete[which(pr.complete$criteria %in% c('task_failed', 'task_successful')), ]
#   word_accuracy.data <- pr.complete[which(pr.complete$criteria %in% c('word_accuracy_100', 'word_accu
#   simulation_quality.data <- pr.complete[which(pr.complete$criteria %in% c('simulation_quality_best',
#   real_vs_worst_sim.data <- pr.complete[which(pr.complete$criteria %in% c('real_vs_simulated_worst',

  cutted <- performance[performance$criteria=='juged_bad',]
  cutted <- rbind(cutted, performance[performance$criteria=='short_interactions',])
  cutted <- rbind(cutted, performance[performance$criteria=='real',])
  cutted <- rbind(cutted, performance[performance$criteria=='task_failed',])
  cutted <- rbind(cutted, performance[performance$criteria=='word_accuracy_100',])
  cutted <- rbind(cutted, performance[performance$criteria=='simulation_quality_best',])
  cutted <- rbind(cutted, performance[performance$criteria=='real_vs_simulated_worst',])

  # for rank order we need only one case for smoothing value, as the smoothing value is
  # not used for rank order
  # select all rows where criteria name is not "rank order" and smoothing_value is not in (0.05, 0.25)
  # That keeps all classifier != "rank order", and for "rank order" only those with smoothin_value == 0
  cutted <- cutted[which( !(cutted$classifier_name == "rank order" & cutted$smoothing_value %in% c(0.05

  # Set names for criteria
  cutted$criteria_name <- 'NA'
  cutted[which(cutted$criteria == 'juged_bad'),]$criteria_name <- 'user jugedment'
  cutted[which(cutted$criteria == 'short_interactions'),]$criteria_name <- 'dialogue length'
  cutted[which(cutted$criteria == 'real'),]$criteria_name <- 'real vs sim. (good)'
  cutted[which(cutted$criteria == 'task_failed'),]$criteria_name <- 'task success'
  cutted[which(cutted$criteria == 'word_accuracy_100'),]$criteria_name <- 'word accuracy'
  cutted[which(cutted$criteria == 'simulation_quality_best'),]$criteria_name <- 'simulation quality'
```

```
cutted[which(cutted$criteria == 'real_vs_simulated_worst'),]$criteria_name <- 'real vs. sim. (bad)'
```

```
temp_1 <- data.frame(c = cutted$classifier_name, s = cutted$criteria_name, n = cutted$n_gram_size,
                     t = cutted$frequency_threshold, l = cutted$smoothing_value, auc = round(cutted$auc,
data = c()
for (size in 1:8) {
  temp_2 <- temp_1[which(temp_1$n == size), ]

  data[[size]] <- dcast(temp_2, t + s + c +n ~ l, value.var = "auc")
}

auc_cols = 5:7
latex_table <-  cbind(data[[1]][,-4], data[[2]][,5:7], data[[3]][,auc_cols], data[[4]][,auc_cols], da
latex_table$c <- revalue(latex_table$c, c("cosine"="cos", "jensen"="j", "mean kullback leibler"="mkl"
latex_table[is.na(latex_table)] <- "" # replace NA (rank order has no AUC values for 2 of 3 smoothing

# Dateinamen zusammenbauen
file_name <- '/home/stefan/git/diss-dokument/cued/Tables/SimEval/cross_validation_results-auc.tex'
fileConn<-file(file_name)

value_rows <- paste(rep("r", 3*8), collapse = "")
lines <- c()
lines <- append(lines, sprintf("\\begin{tabular}{@{}ccc%s@{}} \\toprule", value_rows)) # begin tabula
lines <- append(lines, " &&&  \\multicolumn{24}{c}{$n$} \\\\ \\cmidrule(l){4-27}") # just the column
lines <- append(lines, " &&& \\multicolumn{3}{c}{1} & \\multicolumn{3}{c}{2} & \\multicolumn{3}{c}{3}
# the single lamba columns
lines <- append(lines,  sprintf(" $t$ & $c$ & $d$ & %s \\\\ \\midrule", paste(rep("$\\las$ & $\\lam$ &

for(r in 1:nrow(latex_table)) {

  df_args <- c(latex_table[r,], sep=" & ")
  line <- do.call(paste, df_args)
  line <- paste(c("  ", line, " \\\\"), collapse = "")
  lines <- append(lines, line)
}
lines <- append(lines, "\\bottomrule \n\\end{tabular}")

writeLines(lines, fileConn)
close(fileConn)
```

Overview

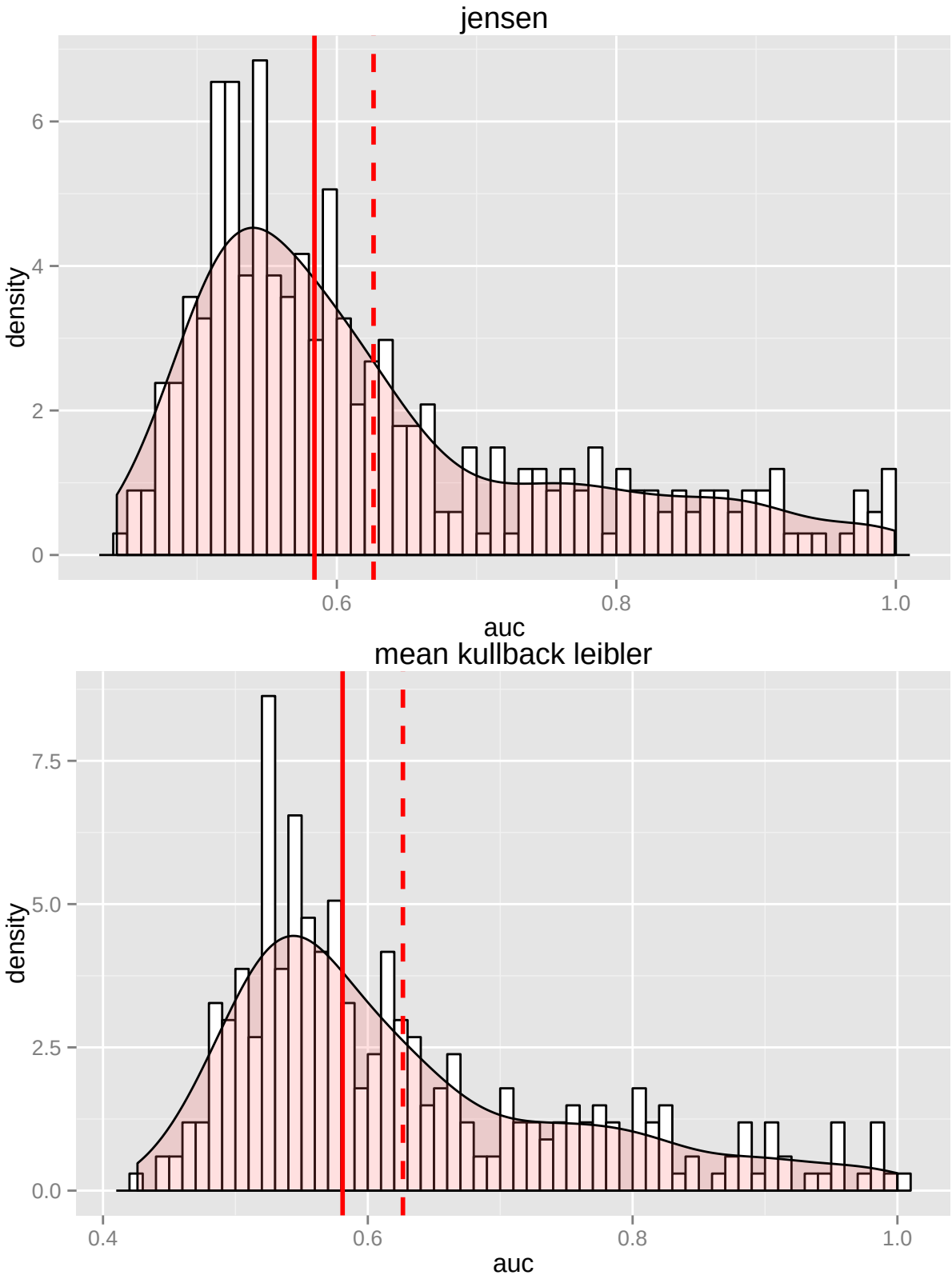
Histogram of AUC for all 2688 scenarios.

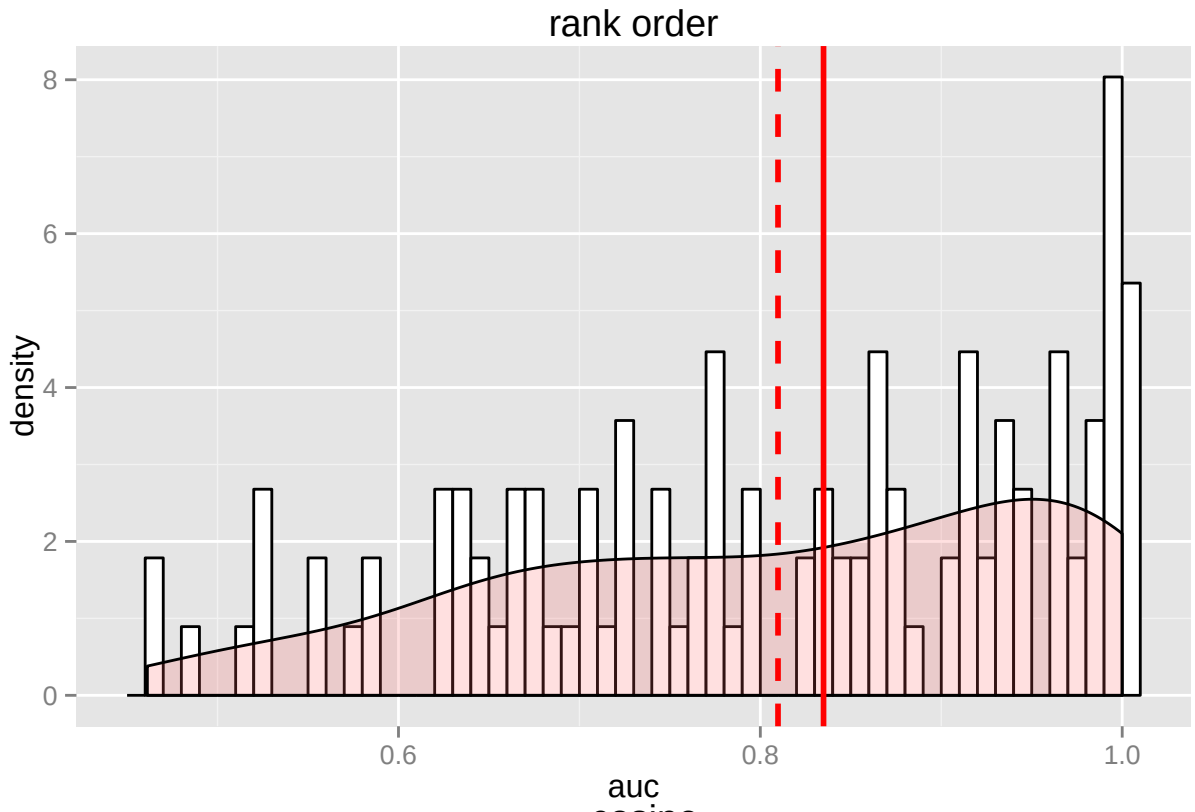


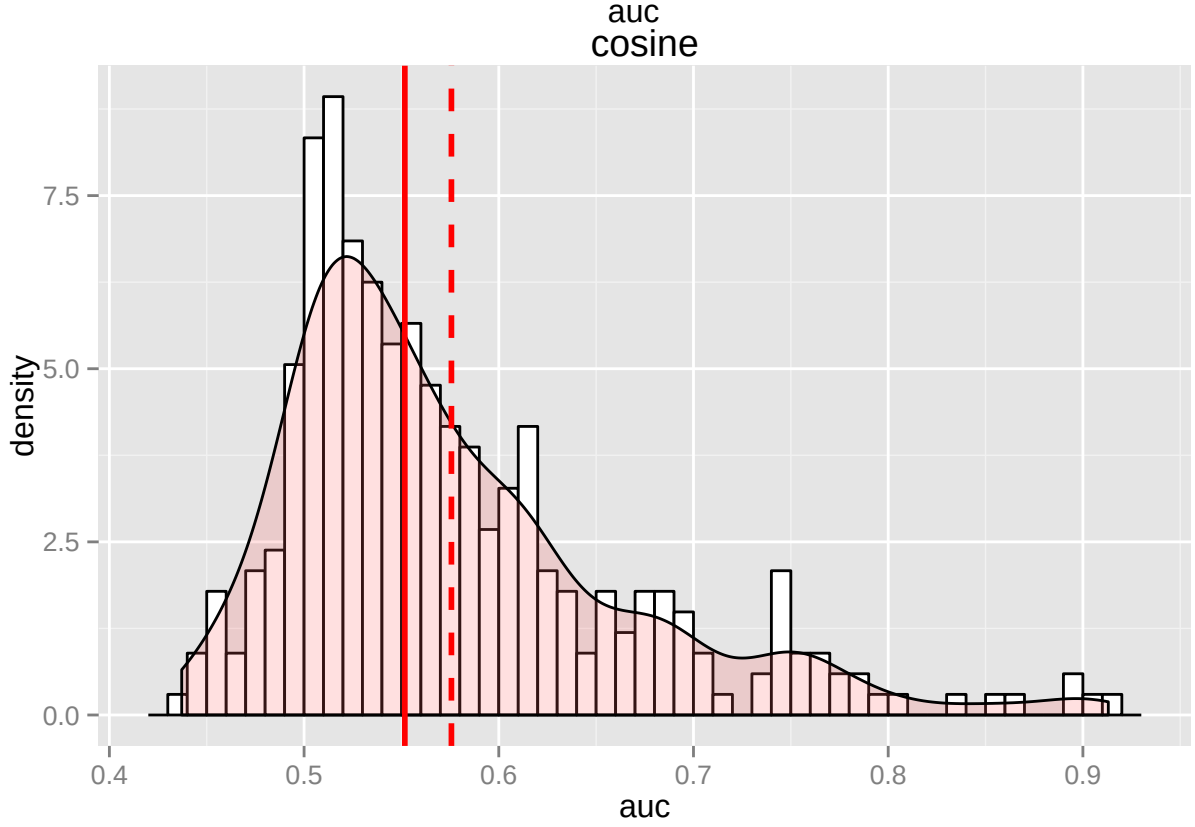
Mean: 0.6295113

Median: 0.581706

Distribution of AUC values per Distance Measure

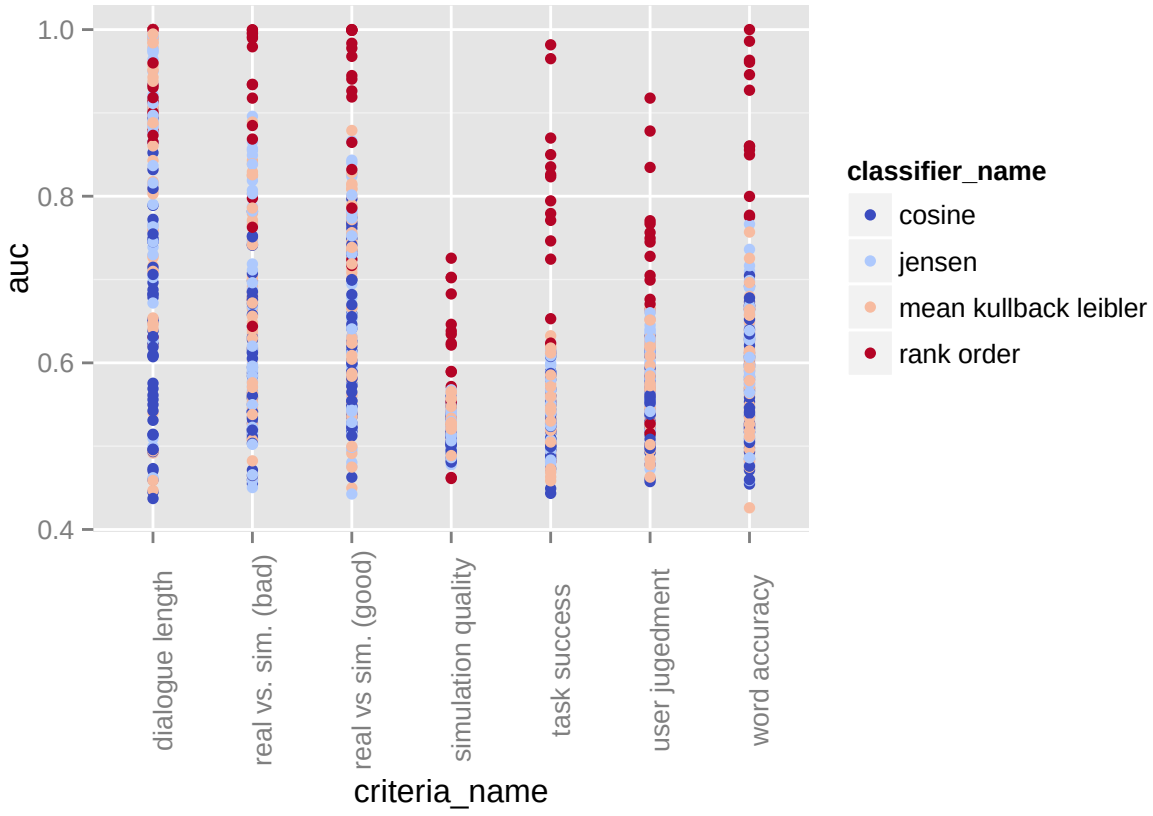



rank order
density
auc
0
2
4
6
8
0.6
0.8
1.0


cosine
density
auc
0.0
2.5
5.0
7.5
0.4
0.5
0.6
0.7
0.8
0.9

AUC in dependency of measures and criteria

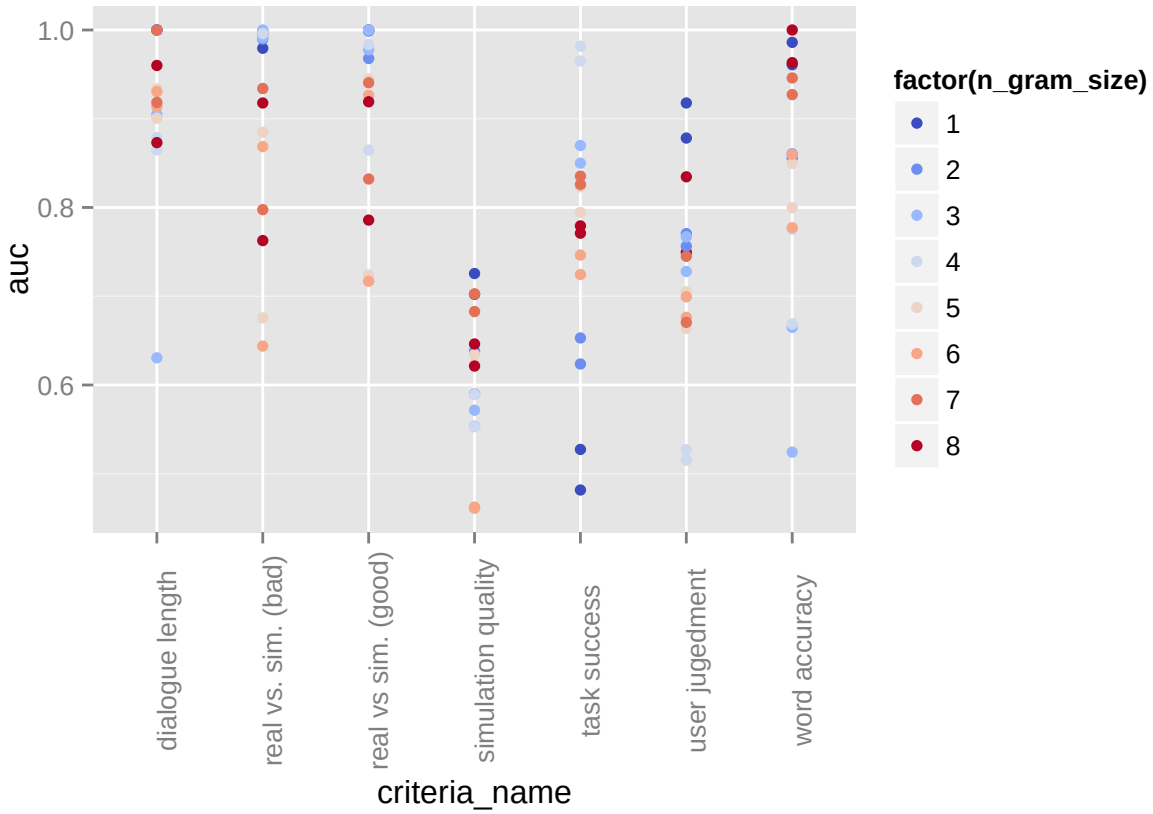
```
ggplot(cutted, aes(x=criteria_name, y=auc, color=classifier_name)) + geom_point() +
theme(axis.text.x = element_text(angle=90)) +
scale_colour_manual(values = discrete_cm(unique(cutted$classifier_name)))
```



Rank Order

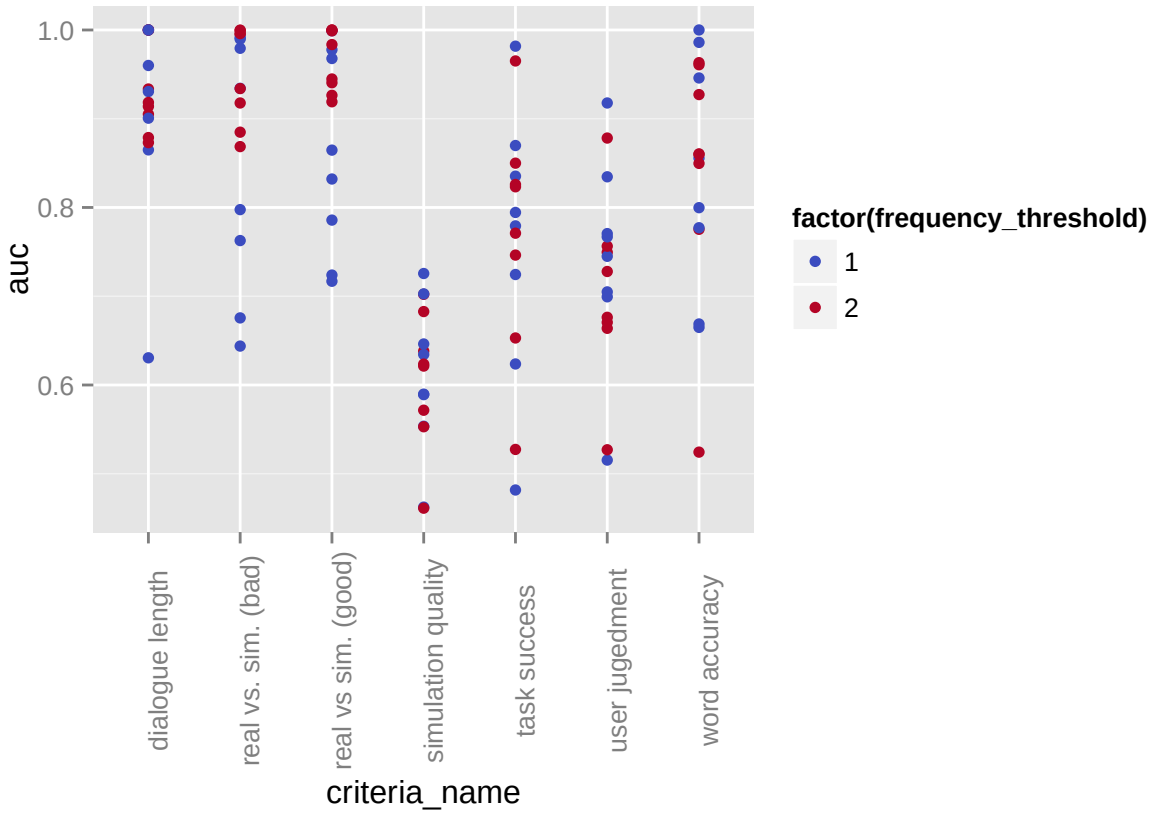
AUC in dependency of n-gram size and criteria

```
ggplot(cutted[which(cutted$classifier_name == 'rank order'),], aes(x=criteria_name, y=auc, color=facto
geom_point() +
theme(axis.text.x = element_text(angle=90)) +
scale_colour_manual(values = discrete_cm(unique(cutted$n_gram_size)))
```



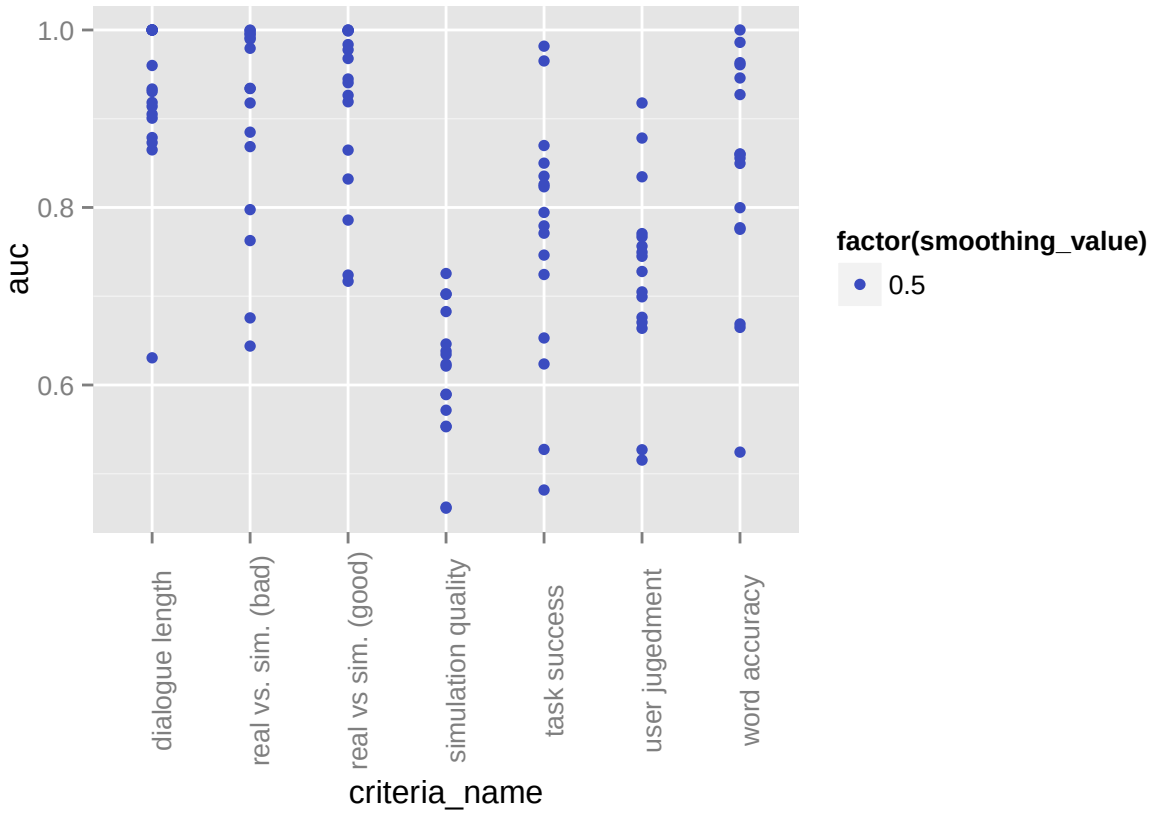
AUC in dependency of frequency threshold and criteria

```
ggplot(cutted[which(cutted$classifier_name == 'rank order'),], aes(x=criteria_name, y=auc, color=facto
geom_point() +
theme(axis.text.x = element_text(angle=90)) +
scale_colour_manual(values = discrete_cm(unique(cutted$frequency_threshold)))
```



AUC in dependency of frequency threshold and criteria

```
ggplot(cutted[which(cutted$classifier_name == 'rank order'),], aes(x=criteria_name, y=auc, color=facto
geom_point() +
theme(axis.text.x = element_text(angle=90)) +
scale_colour_manual(values = discrete_cm(unique(cutted$smoothing_value)))
```



Details about different AUC, Scenarios, Measures

Tables with 5 best classifiers per criteria

```
criteria <- unique(cutted$criteria) # get criteria for filtering
for (i in 1:length(criteria)) {
  crit <- criteria[i]
  cutted.crit <- cutted[which(cutted$criteria == crit),] # get data for current criteria
  cutted.crit <- arrange(cutted.crit, desc(auc)) # sort data for criteria by auc (descending)
  print(kable(cutted.crit[1:5, c(2, 3, 5, 12, 13, 14, 15)])) # print the firs 5 rows (i.e. 5 best cla
}
```

criteria	auc	classifier_name	n_gram_size	frequency_threshold	smoothing_value	criteria_name
juged_bad	0.9177433	rank order	1	1	0.5	user jugedment
juged_bad	0.8782300	rank order	1	2	0.5	user jugedment
juged_bad	0.8345715	rank order	8	1	0.5	user jugedment
juged_bad	0.7705642	rank order	2	1	0.5	user jugedment
juged_bad	0.7669035	rank order	3	1	0.5	user jugedment

criteria	auc	classifier_name	n_gram_size	frequency_threshold	smoothing_value	criteria_nam
short_interactions	1	rank order	1	1	0.5	dialogue leng
short_interactions	1	rank order	1	2	0.5	dialogue leng

criteria	auc	classifier_name	n_gram_size	frequency_threshold	smoothing_value	criteria_nam
short_interactions	1	rank order	2	1	0.5	dialogue leng
short_interactions	1	rank order	2	2	0.5	dialogue leng
short_interactions	1	mean kullback leibler	3	2	0.5	dialogue leng

criteria	auc	classifier_name	n_gram_size	frequency_threshold	smoothing_value	criteria_name
real	0.9999903	rank order	1	2	0.5	real vs sim. (good)
real	0.9998838	rank order	3	2	0.5	real vs sim. (good)
real	0.9990991	rank order	1	1	0.5	real vs sim. (good)
real	0.9990022	rank order	2	2	0.5	real vs sim. (good)
real	0.9835685	rank order	4	2	0.5	real vs sim. (good)

criteria	auc	classifier_name	n_gram_size	frequency_threshold	smoothing_value	criteria_name
task_failed	0.9817352	rank order	4	1	0.5	task success
task_failed	0.9650852	rank order	4	2	0.5	task success
task_failed	0.8698073	rank order	3	1	0.5	task success
task_failed	0.8499276	rank order	3	2	0.5	task success
task_failed	0.8352823	rank order	7	1	0.5	task success

criteria	auc	classifier_name	n_gram_size	frequency_threshold	smoothing_value	criteria_na
word_accuracy_100	1.0000000	rank order	8	1	0.5	word accur
word_accuracy_100	0.9860368	rank order	1	1	0.5	word accur
word_accuracy_100	0.9631610	rank order	8	2	0.5	word accur
word_accuracy_100	0.9607843	rank order	1	2	0.5	word accur
word_accuracy_100	0.9459299	rank order	7	1	0.5	word accur

criteria	auc	classifier_name	n_gram_size	frequency_threshold	smoothing_value	criteri
simulation_quality_best	0.7256023	rank order	1	1	0.5	simula
simulation_quality_best	0.7026837	rank order	7	1	0.5	simula
simulation_quality_best	0.7021459	rank order	1	2	0.5	simula
simulation_quality_best	0.6827253	rank order	7	2	0.5	simula
simulation_quality_best	0.6462170	rank order	8	1	0.5	simula

criteria	auc	classifier_name	n_gram_size	frequency_threshold	smoothing_value	criter
real_vs_simulated_worst	0.9997836	rank order	3	2	0.5	real v
real_vs_simulated_worst	0.9995059	rank order	2	2	0.5	real v
real_vs_simulated_worst	0.9958861	rank order	4	2	0.5	real v
real_vs_simulated_worst	0.9951369	rank order	1	2	0.5	real v
real_vs_simulated_worst	0.9914379	rank order	3	1	0.5	real v

Table with statistical information per criteria and measure information

criteria_name	classifier_name	mean	median	sd	min	max
dialogue length	cosine	0.652	0.635	0.135	0.437	0.913
dialogue length	jensen	0.836	0.880	0.140	0.462	0.999
dialogue length	mean kullback leibler	0.811	0.870	0.156	0.447	1.000
dialogue length	rank order	0.919	0.924	0.091	0.631	1.000
real vs. sim. (bad)	cosine	0.594	0.574	0.085	0.455	0.782
real vs. sim. (bad)	jensen	0.652	0.610	0.124	0.450	0.896
real vs. sim. (bad)	mean kullback leibler	0.669	0.669	0.112	0.465	0.889
real vs. sim. (bad)	rank order	0.898	0.934	0.118	0.644	1.000
real vs sim. (good)	cosine	0.615	0.594	0.087	0.463	0.797
real vs sim. (good)	jensen	0.653	0.642	0.118	0.443	0.866
real vs sim. (good)	mean kullback leibler	0.675	0.705	0.112	0.450	0.879
real vs sim. (good)	rank order	0.911	0.943	0.098	0.717	1.000
simulation quality	cosine	0.516	0.513	0.020	0.481	0.560
simulation quality	jensen	0.524	0.522	0.020	0.478	0.568
simulation quality	mean kullback leibler	0.528	0.527	0.018	0.487	0.566
simulation quality	rank order	0.610	0.622	0.078	0.461	0.726
task success	cosine	0.532	0.521	0.043	0.443	0.614
task success	jensen	0.538	0.532	0.041	0.470	0.616
task success	mean kullback leibler	0.540	0.540	0.047	0.458	0.632
task success	rank order	0.766	0.787	0.139	0.482	0.982
user jugedment	cosine	0.549	0.555	0.041	0.457	0.632
user jugedment	jensen	0.582	0.579	0.047	0.473	0.663
user jugedment	mean kullback leibler	0.572	0.572	0.048	0.463	0.666
user jugedment	rank order	0.725	0.736	0.107	0.515	0.918
word accuracy	cosine	0.571	0.562	0.067	0.455	0.705
word accuracy	jensen	0.598	0.595	0.073	0.459	0.766
word accuracy	mean kullback leibler	0.589	0.593	0.070	0.426	0.757
word accuracy	rank order	0.839	0.858	0.133	0.524	1.000

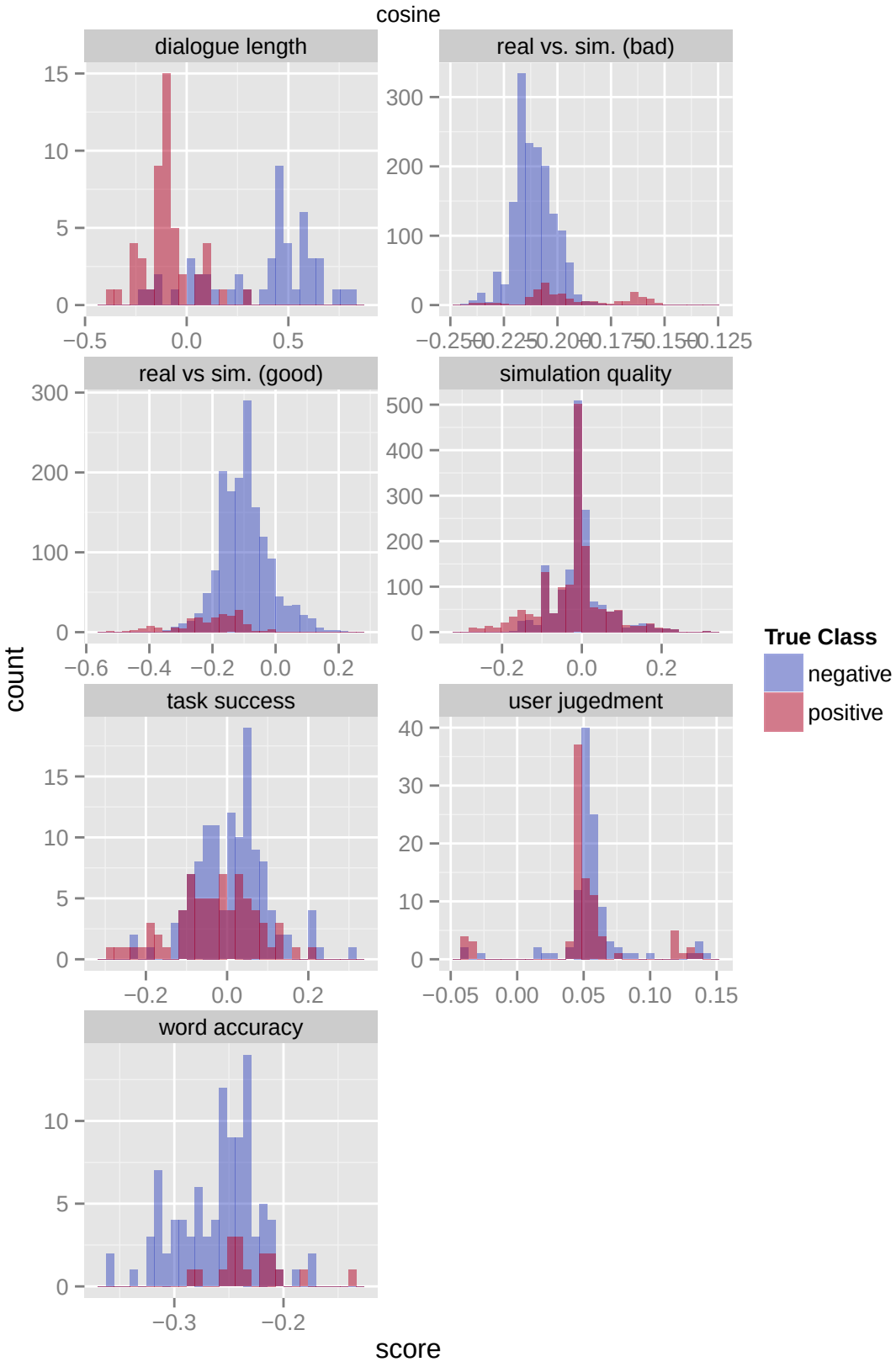
Best configuration of each classifier for each criteria

```
# get each best classifier for each criteria
cutted.all_best <- ddply(cutted, c("classifier_name", "criteria_name"), function(X) X[X$auc == max(X$a

cutted.all_best <- cutted.all_best[ c(-23, -24, -25, -26),] # take only one of the 5 best for rank or

kable(cutted.all_best[, c(5, 15, 12, 13, 14, 3)])
```

	classifier_name	criteria_name	n_gram_size	frequency_threshold	smoothing_value	auc
1	cosine	dialogue length	1	1	0.05	0.9130612
2	cosine	real vs. sim. (bad)	5	1	0.50	0.7817780
3	cosine	real vs sim. (good)	1	2	0.50	0.7968258
4	cosine	simulation quality	3	1	0.05	0.5601430
5	cosine	task success	2	2	0.05	0.6142109
6	cosine	user jugedment	8	1	0.50	0.6324289
7	cosine	word accuracy	8	1	0.25	0.7046940
8	jensen	dialogue length	3	2	0.50	0.9991837
9	jensen	real vs. sim. (bad)	3	1	0.50	0.8958958
10	jensen	real vs sim. (good)	3	1	0.50	0.8661780

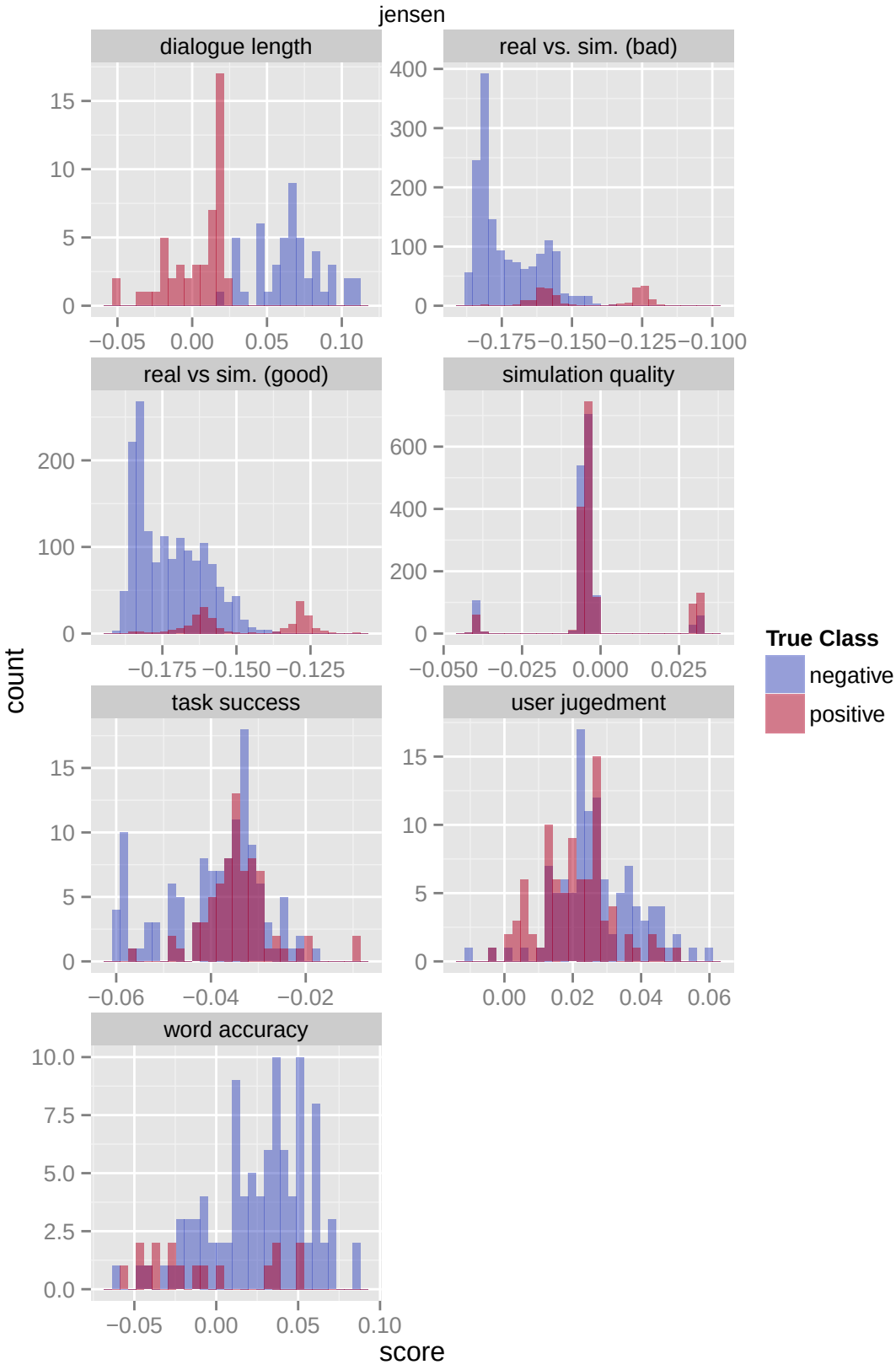
	classifier_name	criteria_name	n_gram_size	frequency_threshold	smoothing_value	auc
11	jensen	simulation quality	5	1	0.50	0.5676446
12	jensen	task success	5	1	0.25	0.6159929
13	jensen	user jugedment	7	2	0.25	0.6630060
14	jensen	word accuracy	5	2	0.05	0.7664884
15	mean kullback leibler	dialogue length	3	2	0.50	1.0000000
16	mean kullback leibler	real vs. sim. (bad)	3	1	0.50	0.8890242
17	mean kullback leibler	real vs sim. (good)	3	1	0.50	0.8788911
18	mean kullback leibler	simulation quality	7	2	0.05	0.5658552
19	mean kullback leibler	task success	2	1	0.50	0.6324758
20	mean kullback leibler	user jugedment	7	2	0.50	0.6661283
21	mean kullback leibler	word accuracy	5	2	0.05	0.7569816
22	rank order	dialogue length	1	1	0.50	1.0000000
27	rank order	real vs. sim. (bad)	3	2	0.50	0.9997836
28	rank order	real vs sim. (good)	1	2	0.50	0.9999903
29	rank order	simulation quality	1	1	0.50	0.7256023
30	rank order	task success	4	1	0.50	0.9817352
31	rank order	user jugedment	1	1	0.50	0.9177433
32	rank order	word accuracy	8	1	0.50	1.0000000

```
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```



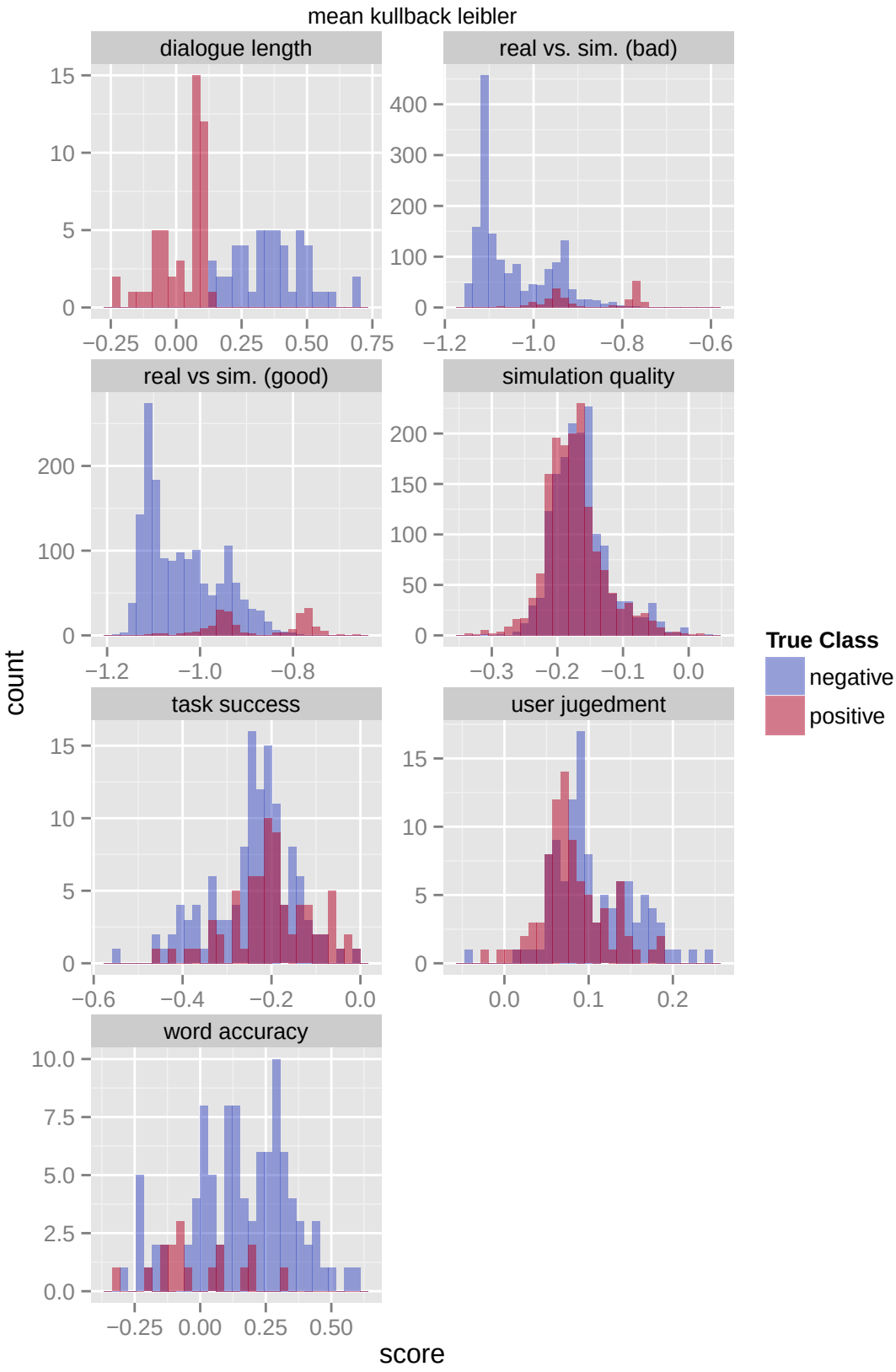

```
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```

```
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```



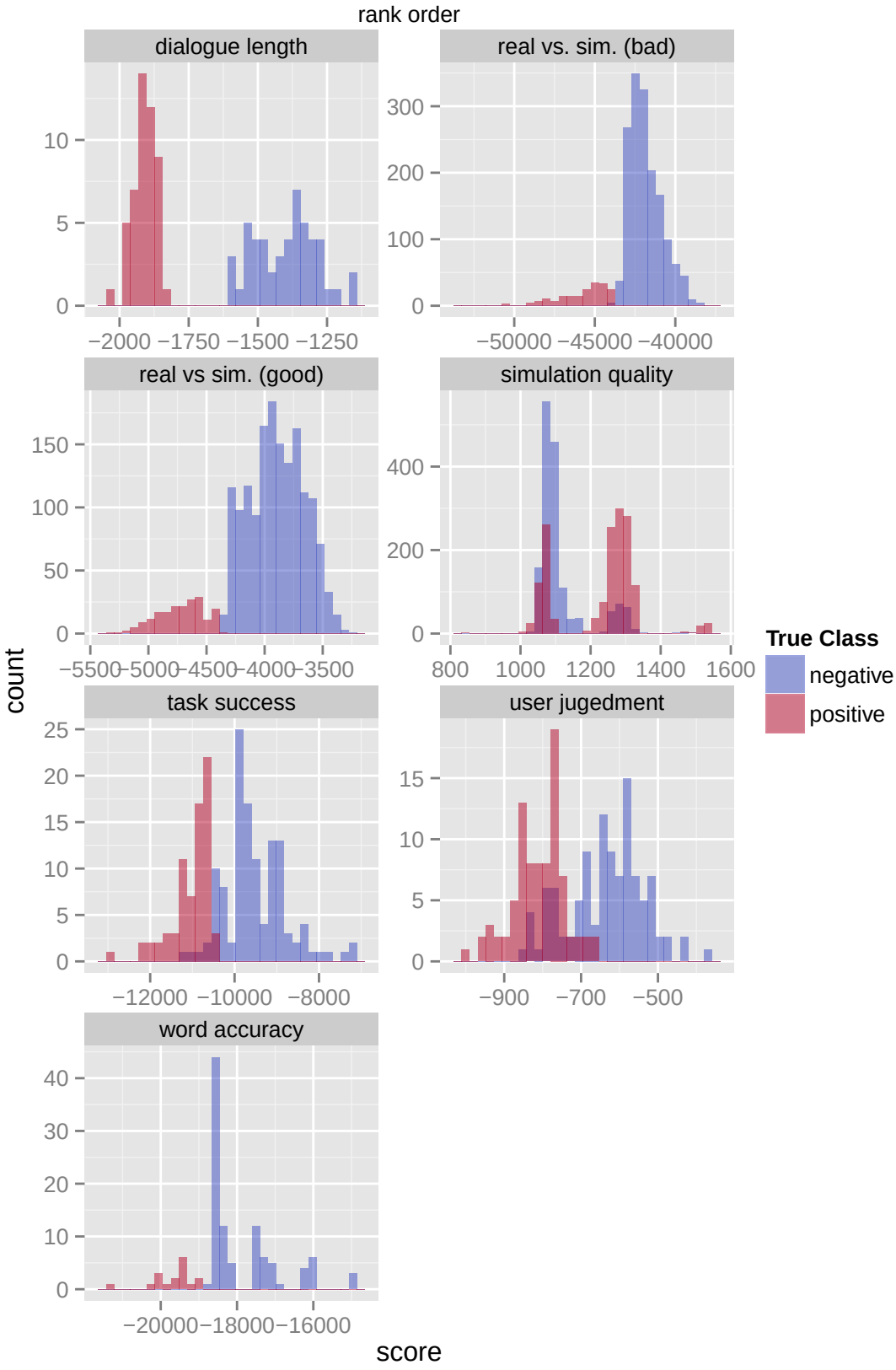

```
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```

```
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```




```
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```

```
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```


rank order
dialogue length
real vs. sim. (bad)
real vs sim. (good)
simulation quality
task success
user jugedment
word accuracy
count
score
True Class
negative
positive

Rank Order Results

All Rank Order results

```
cutted.ro <- cutted[which(cutted$classifier_name == "rank order"),]
kable(arrange(cutted.ro[, c(3, 12, 13, 15)], criteria_name, desc(auc)) )
```

auc	n_gram_size	frequency_threshold	criteria_name
1.0000000	1	1	dialogue length
1.0000000	1	2	dialogue length
1.0000000	2	1	dialogue length
1.0000000	2	2	dialogue length
1.0000000	7	1	dialogue length
0.9600000	8	1	dialogue length
0.9334694	5	2	dialogue length
0.9306122	6	1	dialogue length
0.9183673	7	2	dialogue length
0.9138776	6	2	dialogue length
0.9051020	3	2	dialogue length
0.9006122	5	1	dialogue length
0.8787755	4	2	dialogue length
0.8730612	8	2	dialogue length
0.8648980	4	1	dialogue length
0.6306122	3	1	dialogue length
0.9997836	3	2	real vs. sim. (bad)
0.9995059	2	2	real vs. sim. (bad)
0.9958861	4	2	real vs. sim. (bad)
0.9951369	1	2	real vs. sim. (bad)
0.9914379	3	1	real vs. sim. (bad)
0.9896764	2	1	real vs. sim. (bad)
0.9793723	1	1	real vs. sim. (bad)
0.9342450	4	1	real vs. sim. (bad)
0.9339560	7	2	real vs. sim. (bad)
0.9177393	8	2	real vs. sim. (bad)
0.8848489	5	2	real vs. sim. (bad)
0.8685837	6	2	real vs. sim. (bad)
0.7975894	7	1	real vs. sim. (bad)
0.7627196	8	1	real vs. sim. (bad)
0.6755893	5	1	real vs. sim. (bad)
0.6438420	6	1	real vs. sim. (bad)
0.9999903	1	2	real vs sim. (good)
0.9998838	3	2	real vs sim. (good)
0.9990991	1	1	real vs sim. (good)
0.9990022	2	2	real vs sim. (good)
0.9835685	4	2	real vs sim. (good)
0.9776899	3	1	real vs sim. (good)
0.9678652	2	1	real vs sim. (good)
0.9447866	5	2	real vs sim. (good)
0.9405273	7	2	real vs sim. (good)
0.9263869	6	2	real vs sim. (good)
0.9191020	8	2	real vs sim. (good)

auc	n_gram_size	frequency_threshold	criteria_name
0.8646280	4	1	real vs sim. (good)
0.8320799	7	1	real vs sim. (good)
0.7857821	8	1	real vs sim. (good)
0.7238569	5	1	real vs sim. (good)
0.7168610	6	1	real vs sim. (good)
0.7256023	1	1	simulation quality
0.7026837	7	1	simulation quality
0.7021459	1	2	simulation quality
0.6827253	7	2	simulation quality
0.6462170	8	1	simulation quality
0.6382240	2	2	simulation quality
0.6343631	5	1	simulation quality
0.6235439	5	2	simulation quality
0.6213131	8	2	simulation quality
0.5896823	2	1	simulation quality
0.5891454	4	1	simulation quality
0.5715709	3	2	simulation quality
0.5535095	3	1	simulation quality
0.5529557	4	2	simulation quality
0.4624019	6	1	simulation quality
0.4612081	6	2	simulation quality
0.9817352	4	1	task success
0.9650852	4	2	task success
0.8698073	3	1	task success
0.8499276	3	2	task success
0.8352823	7	1	task success
0.8259272	7	2	task success
0.8233099	5	2	task success
0.7944092	5	1	task success
0.7792627	8	1	task success
0.7710213	8	2	task success
0.7463526	6	2	task success
0.7244682	6	1	task success
0.6529680	2	2	task success
0.6236775	2	1	task success
0.5273973	1	2	task success
0.4816795	1	1	task success
0.9177433	1	1	user jugedment
0.8782300	1	2	user jugedment
0.8345715	8	1	user jugedment
0.7705642	2	1	user jugedment
0.7669035	3	1	user jugedment
0.7564061	2	2	user jugedment
0.7495693	8	2	user jugedment
0.7449935	7	1	user jugedment
0.7278747	3	2	user jugedment
0.7048880	5	1	user jugedment
0.6993432	6	1	user jugedment
0.6762489	6	2	user jugedment
0.6705426	7	2	user jugedment
0.6638674	5	2	user jugedment
0.5270241	4	2	user jugedment

auc	n_gram_size	frequency_threshold	criteria_name
0.5153962	4	1	user jugedment
1.0000000	8	1	word accuracy
0.9860368	1	1	word accuracy
0.9631610	8	2	word accuracy
0.9607843	1	2	word accuracy
0.9459299	7	1	word accuracy
0.9272133	7	2	word accuracy
0.8603684	2	2	word accuracy
0.8597742	6	2	word accuracy
0.8556150	2	1	word accuracy
0.8496732	5	2	word accuracy
0.7997623	5	1	word accuracy
0.7771836	6	1	word accuracy
0.7754011	4	2	word accuracy
0.6687463	4	1	word accuracy
0.6648841	3	1	word accuracy
0.5243613	3	2	word accuracy

```
  cutted.ro.wide <- dcast(cutted.ro, criteria_name + n_gram_size ~ frequency_threshold, value.var="auc"
  cutted.ro.wide$auc_threshold_diff <- cutted.ro.wide$`1` - cutted.ro.wide$`2`
  print(sprintf("Times where AUC of threshold 1 better: %s", nrow(cutted.ro.wide[which(cutted.ro.wide$au
```

```
## [1] "Times where AUC of threshold 1 better: 24"
```

```
  print(sprintf("Times where AUC of threshold 2 better: %s", nrow(cutted.ro.wide[which(cutted.ro.wide$au
```

```
## [1] "Times where AUC of threshold 2 better: 30"
```

```
  print(sprintf("Times where AUC of threshold 1 and 2 are equal: %s", nrow(cutted.ro.wide[which(cutted.
```

```
## [1] "Times where AUC of threshold 1 and 2 are equal: 2"
```

Analysis of smoothing factor's influence in Cosine, Jensen and Mean Kullback-Leibler

```
  cutted.probmeas <- cutted[which(cutted$classifier_name != "rank order"), ] # rows of probability dist
  cutted.probmeas.wide <- dcast(cutted.probmeas, criteria_name + n_gram_size + frequency_threshold ~ sm
```

```
## Aggregation function missing: defaulting to length
```

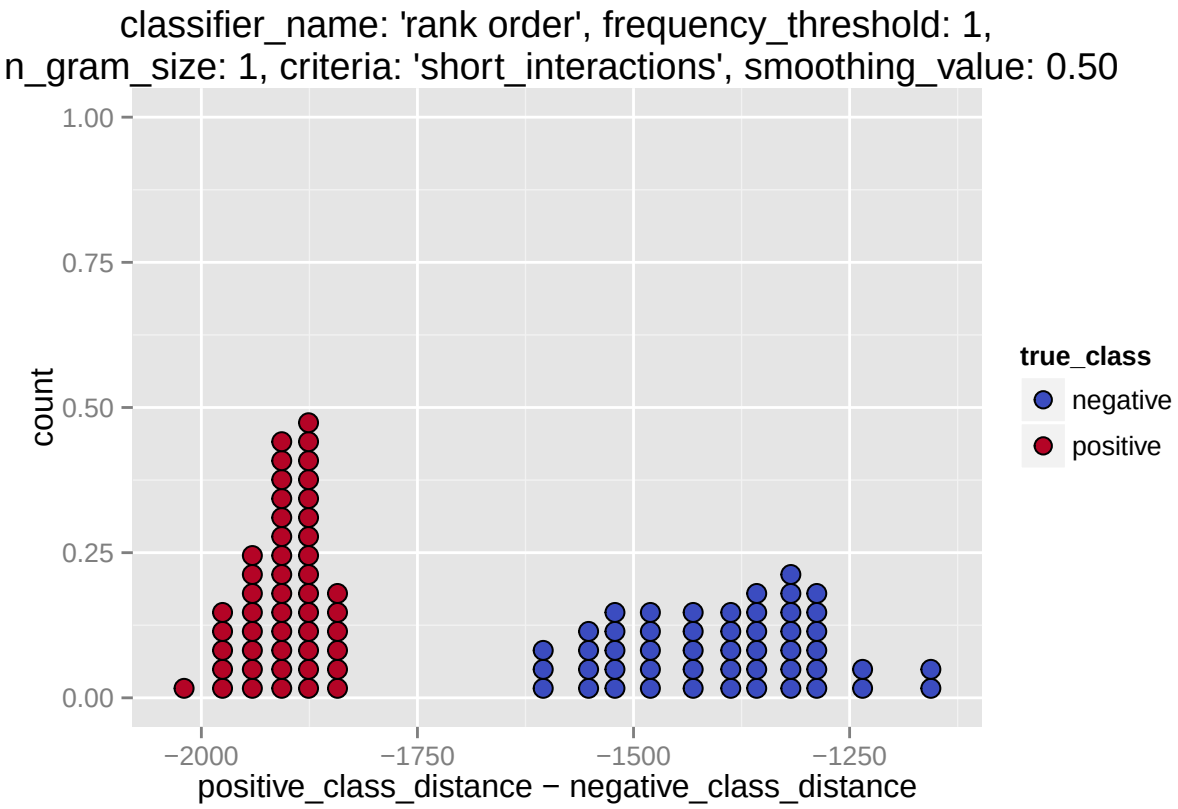
Detailed view on selected scenarios

Histogram for t=1, n=8, word_accuracy_60 and s = 0.25

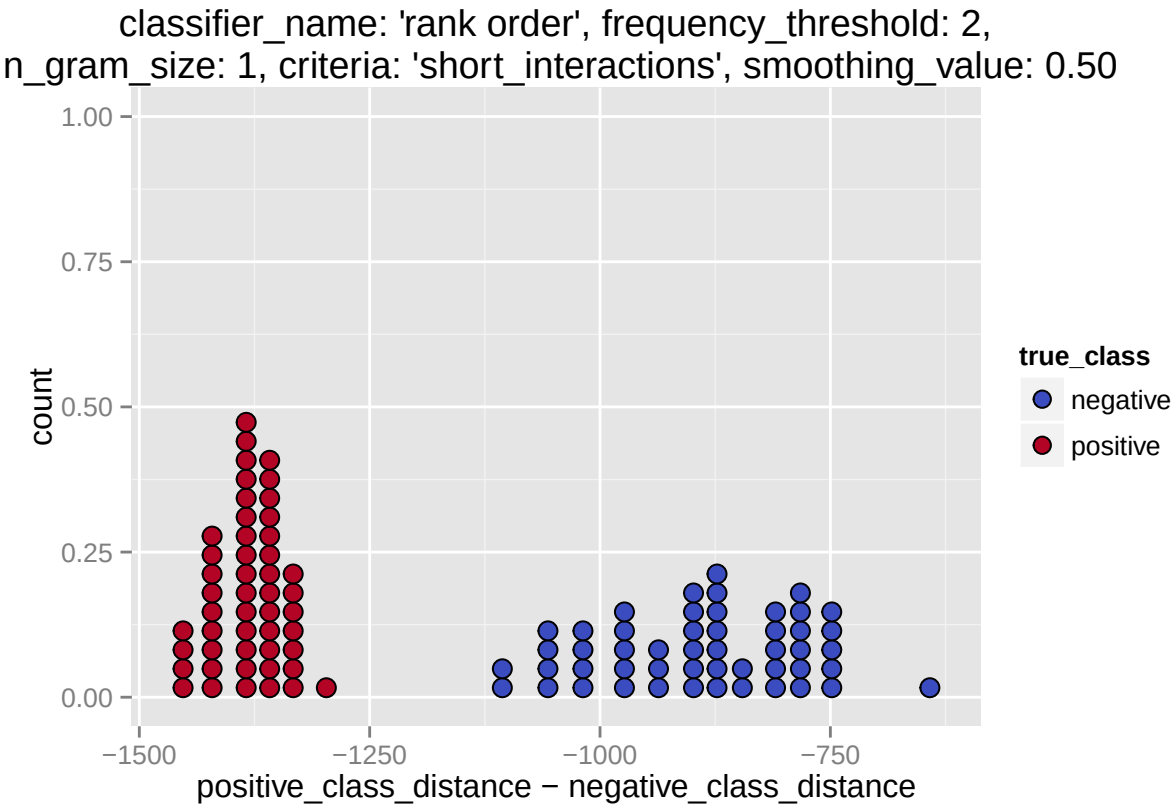
Get scenatios with AUC = 1.

```
  # get all performance entries with auc = 1
  p <- cutted[which(cutted$auc == 1),]
    #dbGetQuery(cross_validation, "performance", "{auc: 1}", 0, 0)
```

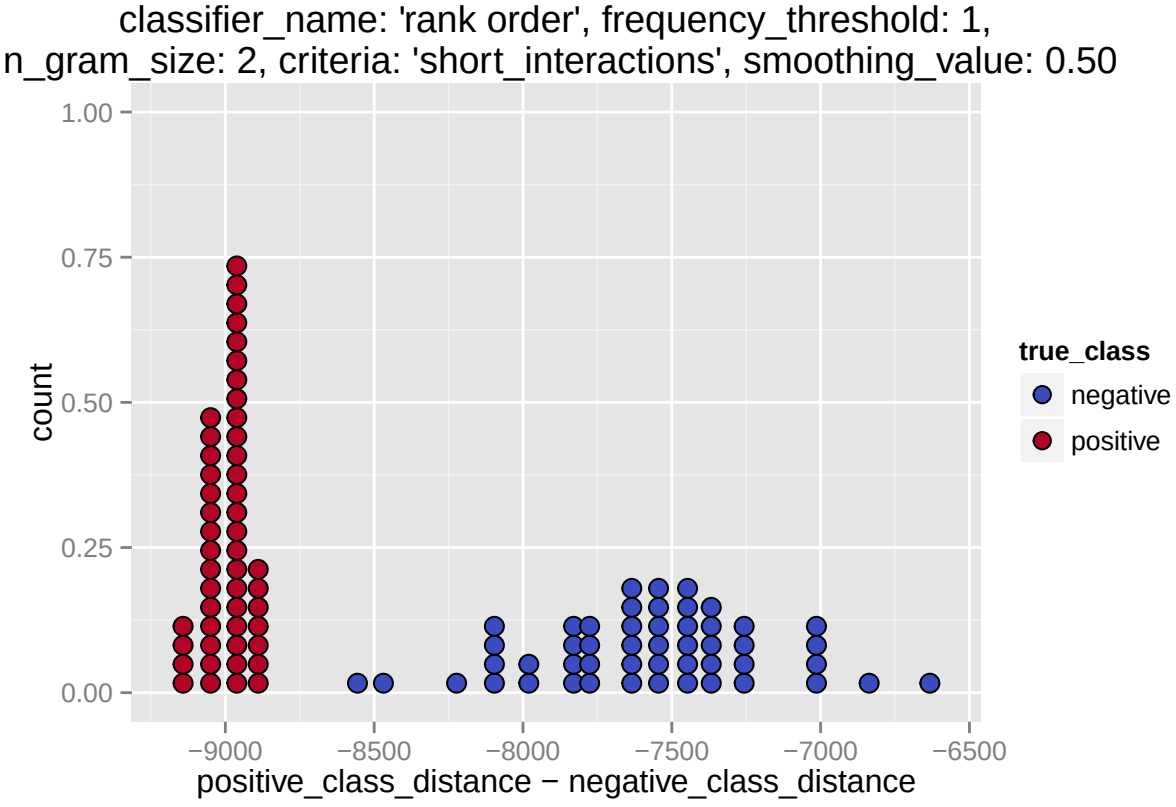
```
## stat_bindot: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```



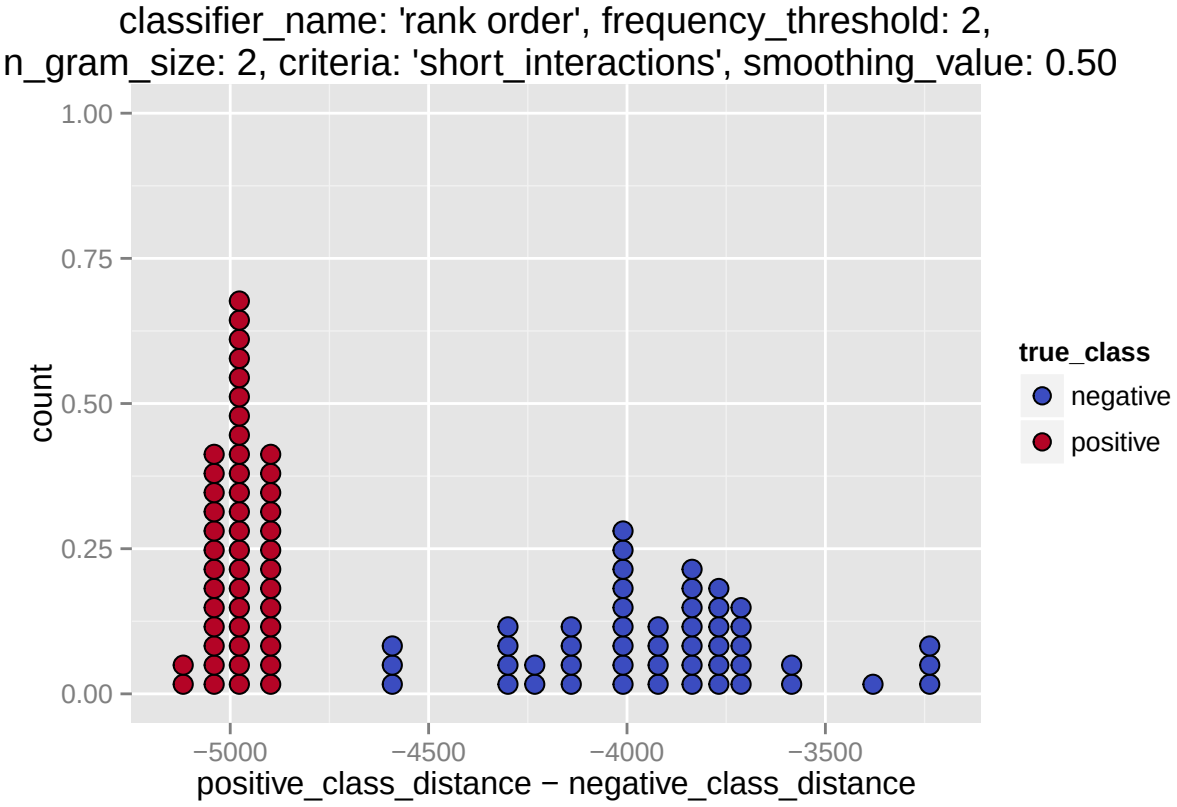

```
## stat_bindot: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```




```
## stat_bindot: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```

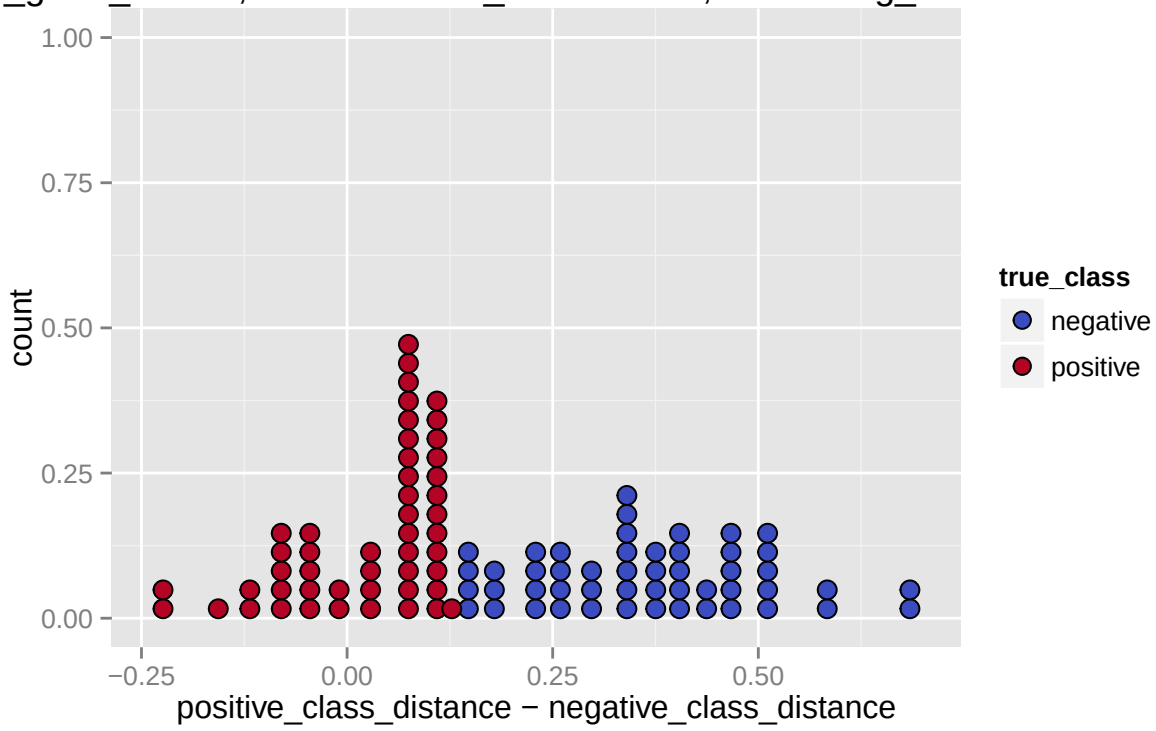


```
## stat_bindot: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```



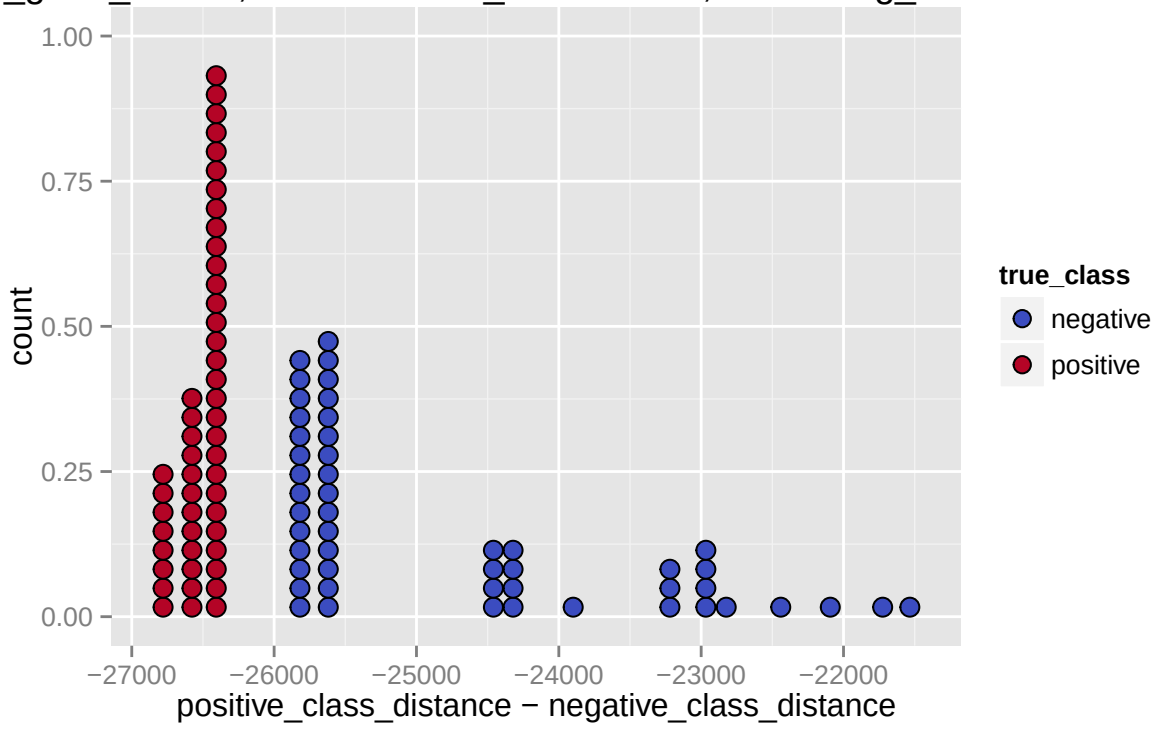

```
## stat_bindot: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```

classifier_name: 'mean kullback leibler', frequency_threshold: 2, n_gram_size: 3, criteria: 'short_interactions', smoothing_value: 0.50




```
## stat_bindot: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```

classifier_name: 'rank order', frequency_threshold: 1, n_gram_size: 7, criteria: 'short_interactions', smoothing_value: 0.50



```
## stat_bindot: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```