DMC@ISU: Iowa State University's 2015 Data Mining Cup Team

Categorical Similarity Measures

Spring 2015, One Week Left

Name:	Ian Mouzon
email:	imouzon@iastate.edu
Due Date:	May 8, 2015

I am using the following packages:

```
library(ggplot2)
library(lubridate)
library(xtable)
library(foreach)
library(rCharts)
library(magrittr)
library(tidyr)
library(dplyr)
library(reshape2)
library(gtools)
library(sqldf)
library(missForest)
source("./R/renm.R")
```

and my working directory is set to `dmc2015/ian`.

0.1 Categorical Similarity

Oh My Gosh - did you know that you can compare categorical variables to each other? You can create kind of like a distance on them.
One simple example based on the Jaccard Measure:
Let $\mathbf{u}$ and $\mathbf{v}$ be two multidimensional categorical variables taking values in $A = \{a_1, a_2, \ldots, a_n\}$. Then we can think of these as subsets of the the set A in which case we can describe a similarity between the coupons using

$$J(\mathbf{u}, \mathbf{v}) = \frac{|\mathbf{u} \cap \mathbf{v}|}{\mathbf{u} \cup \mathbf{v}}$$

In this document I am calculating some of these features for the categorical variables `categoryIDs`.

0.2 Getting Tranthe Data and Manipulations

I am using our new clean data - so should you

```
d = readRDS("../data/clean_data/universalCleanData.rds")
```

I can melt the columns by coupon using the following:

```
source("./r/stackCoupons2.R")
dm = stackCoupons2(d, idcols = c(1:4, 32:49))
```

```
## using the following as id:
##  orderID,
##  orderTime,
##  userID,
##  couponsReceived,
##  basketValue,
##  couponsReceivedDate,
##  couponsReceivedTime,
##  couponsReceivedDoW,
##  couponsReceivedWeekend,
##  couponsReceivedFriSat,
##  orderTimeDate,
##  orderTimeTime,
##  orderTimeDoW,
##  orderTimeWeekend,
##  orderTimeFriSat,
##  batchID,
##  couponsExpire,
##  couponsSent,
##  TimeBtwnSentRec,
##  TimeBtwnRecExpire,
##  TimeBtwnRecOrder,
##  TimeBtwnOrderExpire
##
## using the following as measure columns:
##  couponID1,
##  price1,
##  basePrice1,
##  reward1,
##  premiumProduct1,
##  brand1,
##  productGroup1,
##  categoryIDs1,
##  coupon1Used,
##  couponID2,
##  price2,
##  basePrice2,
##  reward2,
##  premiumProduct2,
##  brand2,
##  productGroup2,
##  categoryIDs2,
##  coupon2Used,
##  couponID3,
##  price3,
##  basePrice3,
##  reward3,
##  premiumProduct3,
##  brand3,
##  productGroup3,
##  categoryIDs3,
##  coupon3Used
```

I and can split the columns of product group using:

```
source("./r/splitColumn.R")
dmc = splitColumn(dm, "categoryIDs", "orderID", splitby = ":")

## Loading required package: tcltk

dmcs = dmc[, c(1, 23, 32:37)] %>% mutate(couponNum = as.numeric(gsub("cpn",
    "", couponID))) %>% arrange(orderID, couponID)

# the number of coupons
ncpns = length(unique(dmc$couponID))
```

Consider the unique coupon IDs and create a matrix with the columns 0 to 1 for whether or not the coupon has the given category:

```
dmcsu = unique(dmcs[, c(2, 9, 4:8)]) %>% arrange(couponNum)

dmcsuc = matrix(NA, nrow = nrow(dmcsu), ncol = 31)
for (i in 1:nrow(dmcsuc)) dmcsuc[i, ] = 1 * (paste0("cat", 1:31) %in% dmcsu[i,
    2:6])

catIndMat = dmcsu[, "couponID"] %>% cbind(data.frame(dmcsuc))
names(catIndMat) = c("couponID", paste0("cat", 1:31))
```

and save the jaccard matrix from this:

```
# these are the jaccard similarities
jaccard_func = function(cpn1, cpn2) sum(cpn1 * cpn2)/sum(as.numeric(cpn1 + cpn2 >
    0))

dmcsuck = matrix(0, ncpns, ncpns)
for (i in 1:ncpns) for (j in i:ncpns) dmcsuck[i, j] = jaccard_func(dmcsuc[i,
    ], dmcsuc[j, ])
```

0.3 Jaccard Similarity Used To Describe Order

We can use these jaccard similaritys to describe our orders. Consider, for example, the mean Jaccard similarity:

```
dmcsucks = d %>% arrange(orderID) %>% select(couponID1, couponID2, couponID3)

JaccardSummary = function(i) {
    x = as.numeric(dmcsucks[i, ])
    rows = combn(x, 2)[1, ]
    cols = combn(x, 2)[2, ]
    sapply(1:3, function(j) dmcsuck[rows[j], cols[j]])
}

jac.d = d %>% select(orderID) %>% cbind(t(sapply(1:nrow(dmcsucks), function(i) JaccardSummary(i)))) %>%
    data.frame %>% renm(c("orderID", "jaccard12", "jaccard13", "jaccard23"))

head(jac.d)

##   orderID jaccard12 jaccard13 jaccard23
## 1       1 0.0000000      0.25      0.25
## 2       2 0.0000000      0.00      1.00
```

```
## 3       3 0.3333333      0.00      0.00
## 4       4 0.0000000      0.00      0.00
## 5       5 0.0000000      0.00      0.00
## 6       6 0.0000000      0.00      0.00
```

0.4 Term Frequency, Document Frequency

We can also consider comparing the coupons like a set of documents, using `tf-idf`:

```
# parameters
N = ncpns
nt = colSums(dmcsuc)

# term frequency raw frequency
ftd = dmcsuc

## log normalization
lnormM = log(2) * dmcsuc

# document frequency inverse frequency
ivf = log(N/nt)

## inverse frequency smooth
lnorm = log(1 + N/nt)

## inverse frequency max
invmax = log(1 + max(nt)/nt)

## probabilistic inverse frequency
pif = log((N - nt)/nt)

ftd_lnorm = matrix(sapply(1:N, function(i) ftd[i, ] * lnorm), nrow = N, byrow = TRUE)
ftd_ivf = matrix(sapply(1:N, function(i) ftd[i, ] * ivf), nrow = N, byrow = TRUE)
ftd_invmax = matrix(sapply(1:N, function(i) ftd[i, ] * invmax), nrow = N, byrow = TRUE)
ftd_pif = matrix(sapply(1:N, function(i) ftd[i, ] * pif), nrow = N, byrow = TRUE)

lnorm_lnorm = matrix(sapply(1:N, function(i) lnormM[i, ] * lnorm), nrow = N,
    byrow = TRUE)
lnorm_ivf = matrix(sapply(1:N, function(i) lnormM[i, ] * ivf), nrow = N, byrow = TRUE)
lnorm_invmax = matrix(sapply(1:N, function(i) lnormM[i, ] * invmax), nrow = N,
    byrow = TRUE)
lnorm_pif = matrix(sapply(1:N, function(i) lnormM[i, ] * pif), nrow = N, byrow = TRUE)
```

0.5 Cosine Similarity

And we can get the cosine similiarty matrix:

```
self_multiply = function(X) X %*% t(X)

cosine_similarity = function(sim_mat) {
    (sim_mat %*% t(sim_mat))/(sim_mat %*% t(sim_mat) %>% sqrt %>% diag %>% matrix(ncol = 1) %>%
        self_multiply)
}
```

```
sim_ftd_lnorm = cosine_similarity(ftd_lnorm)
sim_ftd_ivf = cosine_similarity(ftd_ivf)
sim_ftd_invmax = cosine_similarity(ftd_invmax)
sim_ftd_pif = cosine_similarity(ftd_pif)

sim_lnorm_lnorm = cosine_similarity(lnorm_lnorm)
sim_lnorm_ivf = cosine_similarity(lnorm_ivf)
sim_lnorm_invmax = cosine_similarity(lnorm_invmax)
sim_lnorm_pif = cosine_similarity(lnorm_pif)
```

which we can calculate as follows:

```
dmcsucks = d %>% arrange(orderID) %>% select(couponID1, couponID2, couponID3)

CosSimSummary = function(i, sim_mat) {
    x = as.numeric(dmcsucks[i, ])
    rows = combn(x, 2)[1, ]
    cols = combn(x, 2)[2, ]
    sapply(1:3, function(j) sim_mat[rows[j], cols[j]])
}

# sim_ftd_lnorm
cos.d1 = d %>% select(orderID) %>% cbind(t(sapply(1:nrow(dmcsucks), function(i) CosSimSummary(i,
    sim_ftd_lnorm)))) %>% data.frame %>% renm(c("orderID", "sim_ftd_lnorm_cosSim12",
    "sim_ftd_lnorm_cosSim13", "sim_ftd_lnorm_cos.sim23"))

# sim_ftd_ivf
cos.d2 = d %>% select(orderID) %>% cbind(t(sapply(1:nrow(dmcsucks), function(i) CosSimSummary(i,
    sim_ftd_ivf)))) %>% data.frame %>% renm(c("orderID", "sim_ftd_ivf_cosSim12",
    "sim_ftd_ivf_cosSim13", "sim_ftd_ivf_cos.sim23"))

# sim_ftd_infmax
cos.d3 = d %>% select(orderID) %>% cbind(t(sapply(1:nrow(dmcsucks), function(i) CosSimSummary(i,
    sim_ftd_invmax)))) %>% data.frame %>% renm(c("orderID", "sim_ftd_invmax_cosSim12",
    "sim_ftd_invmax_cosSim13", "sim_ftd_invmax_cos.sim23"))

# sim_lnorm_pif
cos.d4 = d %>% select(orderID) %>% cbind(t(sapply(1:nrow(dmcsucks), function(i) CosSimSummary(i,
    sim_ftd_pif)))) %>% data.frame %>% renm(c("orderID", "sim_ftd_pif_cosSim12",
    "sim_ftd_pif_cosSim13", "sim_ftd_pif_cos.sim23"))

# sim_lnorm_lnorm
cos.d5 = d %>% select(orderID) %>% cbind(t(sapply(1:nrow(dmcsucks), function(i) CosSimSummary(i,
    sim_lnorm_lnorm)))) %>% data.frame %>% renm(c("orderID", "sim_lnorm_lnorm_cosSim12",
    "sim_lnorm_lnorm_cosSim13", "sim_lnorm_lnorm_cos.sim23"))

# sim_lnorm_ivf
cos.d6 = d %>% select(orderID) %>% cbind(t(sapply(1:nrow(dmcsucks), function(i) CosSimSummary(i,
    sim_lnorm_ivf)))) %>% data.frame %>% renm(c("orderID", "sim_lnorm_ivf_cosSim12",
    "sim_lnorm_ivf_cosSim13", "sim_lnorm_ivf_cos.sim23"))

# sim_lnorm_infmax
cos.d7 = d %>% select(orderID) %>% cbind(t(sapply(1:nrow(dmcsucks), function(i) CosSimSummary(i,
    sim_lnorm_invmax)))) %>% data.frame %>% renm(c("orderID", "sim_lnorm_invmax_cosSim12",
```

```
    "sim_lnorm_invmax_cosSim13", "sim_lnorm_invmax_cos.sim23"))

# sim_lnorm_pif
cos.d8 = d %>% select(orderID) %>% cbind(t(sapply(1:nrow(dmcsucks), function(i) CosSimSummary(i,
    sim_lnorm_pif)))) %>% data.frame %>% renm(c("orderID", "sim_lnorm_pif_cosSim12",
    "sim_lnorm_pif_cosSim13", "sim_lnorm_pif_cos.sim23"))
```

0.6 Results

Consider the chances that coupons were used when the coupons had more similarity.

```
simFeatures = function(sim_mat) {
    d_A = dm %>% select(orderID, couponID, couponCol, couponUsed) %>% renm(c("orderID",
        "couponIDA", "couponColA", "couponUsedA")) %>% left_join(sim_mat, by = "orderID") %>%
        gather(sim_meas, value, -orderID, -couponIDA, -couponColA, -couponUsedA) %>%
        arrange(orderID, couponColA) %>% filter((grepl("12", sim_meas) & couponColA ==
        1) | (grepl("13", sim_meas) & couponColA == 1) | (grepl("23", sim_meas) &
        couponColA == 2))

    d_B = dm %>% select(orderID, couponID, couponCol, couponUsed) %>% renm(c("orderID",
        "couponIDB", "couponColB", "couponUsedB")) %>% left_join(sim_mat, by = "orderID") %>%
        gather(sim_meas, value, -orderID, -couponIDB, -couponColB, -couponUsedB) %>%
        arrange(orderID, couponColB) %>% filter((grepl("23", sim_meas) & couponColB ==
        3) | (grepl("13", sim_meas) & couponColB == 3) | (grepl("12", sim_meas) &
        couponColB == 2))

    d_AB = d_A %>% left_join(d_B, by = c("orderID", "sim_meas", "value")) %>%
        mutate(nUsed = factor(couponUsedA + couponUsedB))

    ggplot(data = d_AB, aes(x = value, color = factor(nUsed))) + geom_density()

    p = d_AB %>% filter(!is.na(couponUsedA)) %>% mutate(usedColA = paste0("coupon",
        couponColA, "Used=", couponUsedA)) %>% mutate(usedColB = paste0("coupon",
        couponColB, "Used=", couponUsedB)) %>% mutate(usedA = paste0("couponAUsed=",
        couponUsedA)) %>% mutate(usedB = paste0("couponBUsed=", couponUsedB)) %>%
        mutate(fill = paste0("cpn", couponColA, "Xcpn", couponColB)) %>% mutate(nUsed = factor(couponUs
        couponUsedB)) %>% mutate(similarity = value)

    p1 = p %>% ggplot(aes(fill = usedA, x = similarity, alpha = 0.2)) + facet_grid(usedB ~
        ., scales = "free_y") + geom_density()

    p2 = p %>% ggplot(aes(fill = interaction(usedA, usedB), x = fill, y = log(similarity),
        alpha = 0.2)) + geom_boxplot()

    return(list(p1 = p1, p2 = p2))
}
```

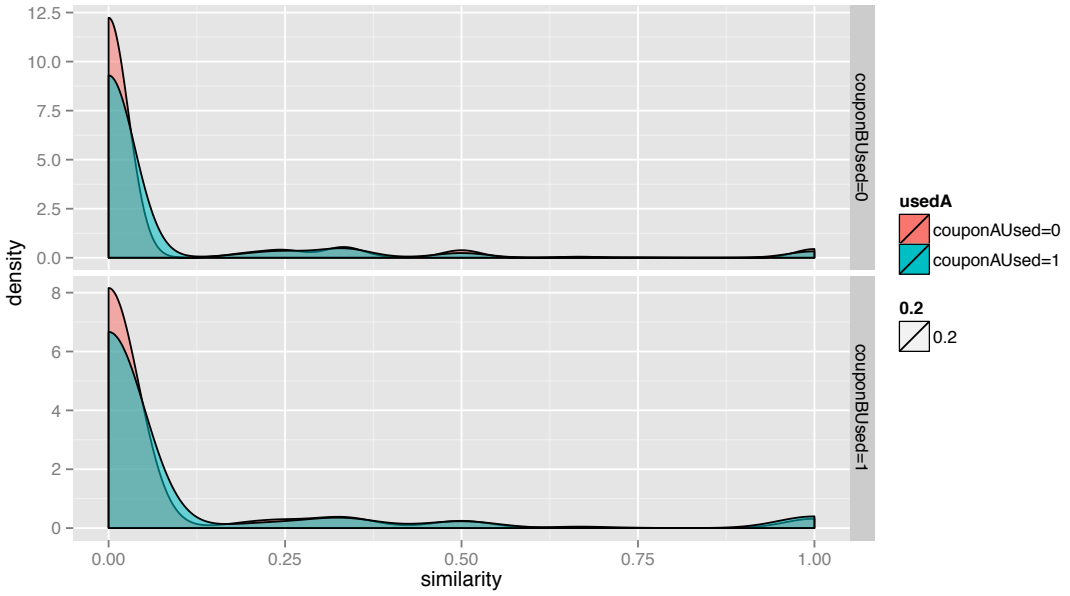
0.6.1 Jaccard Measure

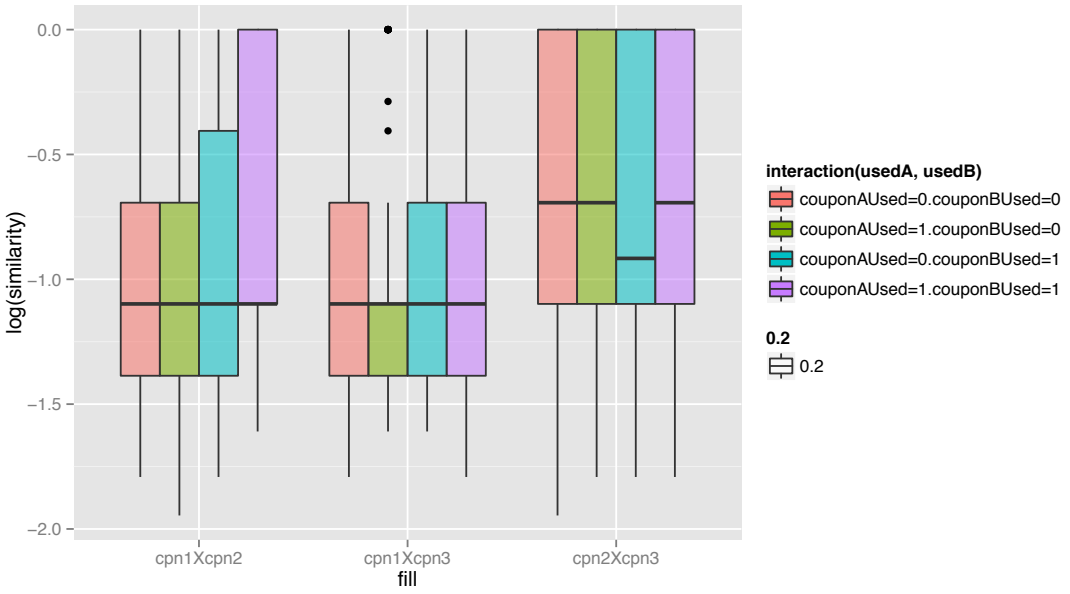
```
# figures
figures = simFeatures(jac.d)

# print
```

```
print(figures$p1)
print(figures$p2)
```

```
## Warning in loop_apply(n, do.ply): Removed 15573 rows containing non-finite
## values (stat_boxplot).
```

```
# save
saveRDS(jac.d, file = "../features/feature_files/universal/jaccard_similarity.rds")
```

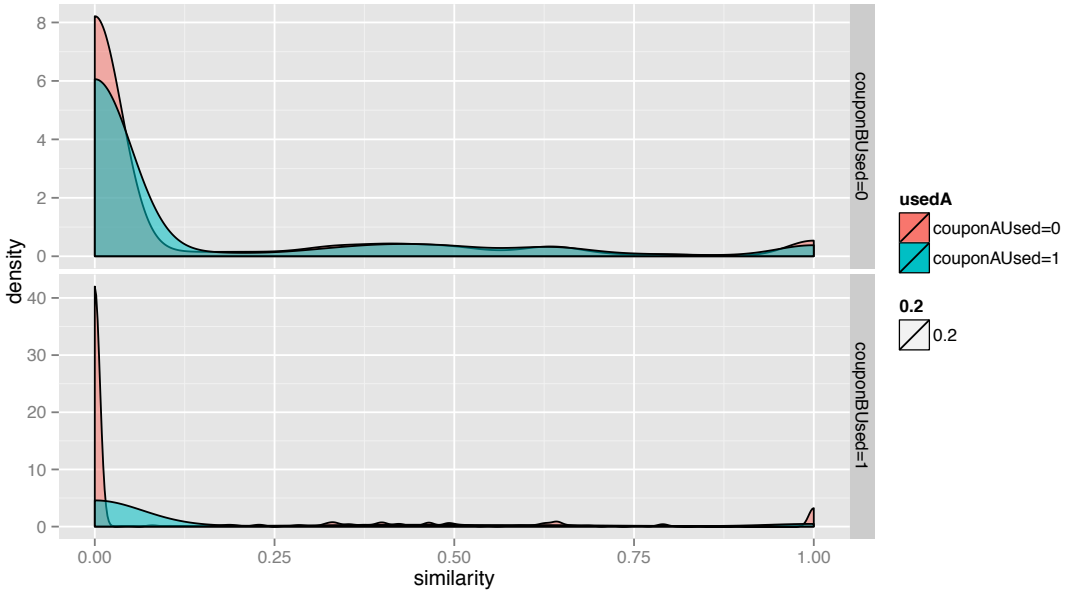


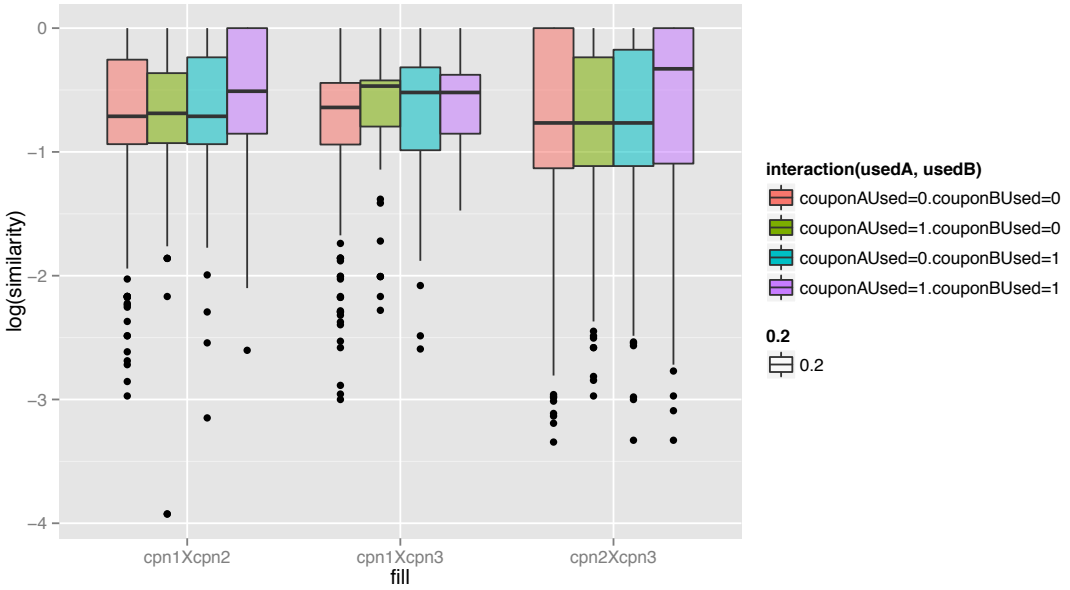


0.6.2 tf-idf using ftd and lnorm

```
# similarity matrix
sim_mat = cos.d1
```

```
# figures
figures = simFeatures(sim_mat)

# print
print(figures$p1)
print(figures$p2)

## Warning in loop_apply(n, do.ply): Removed 13870 rows containing non-finite
## values (stat_boxplot).

# save
saveRDS(sim_mat, file = "../features/feature_files/universal/ftd_lnorm_similarity.rds")
```



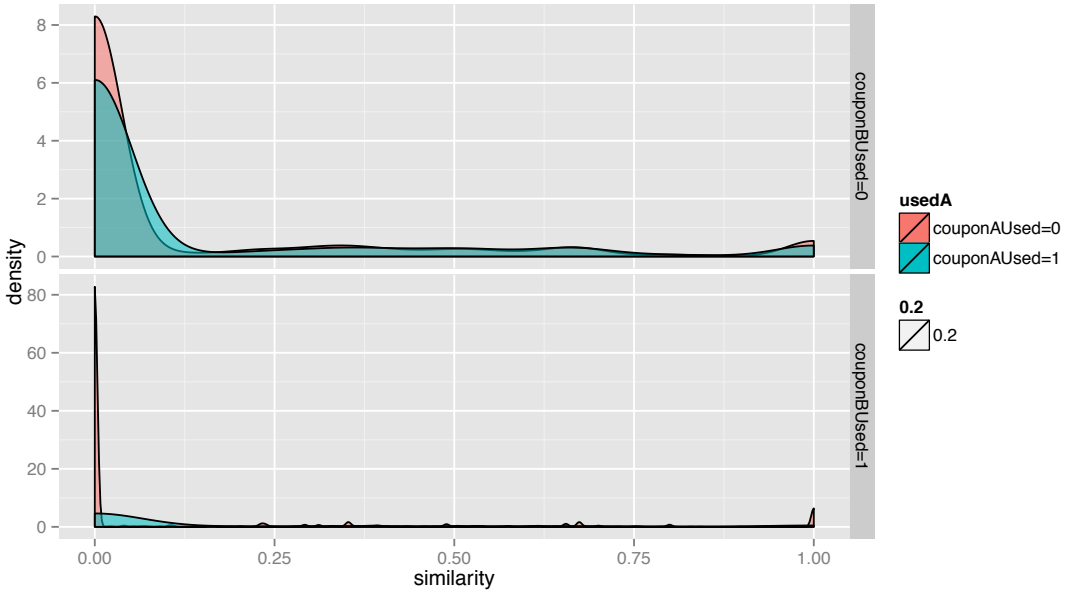


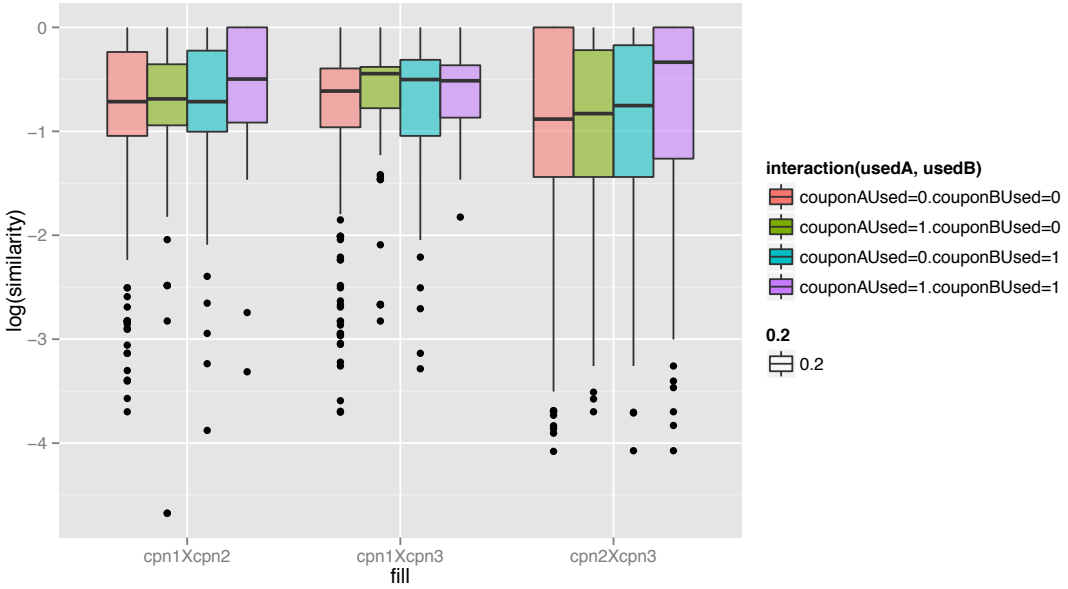
0.6.3 tf-idf using ftd and ivf

```
# similarity matrix
sim_mat = cos.d2

# figures
figures = simFeatures(sim_mat)

# print
print(figures$p1)
print(figures$p2)
```

```
## Warning in loop_apply(n, do.ply): Removed 13870 rows containing non-finite
## values (stat_boxplot).
```

```
# save
saveRDS(sim_mat, file = "../features/feature_files/universal/ftd_ivf_similarity.rds")
```



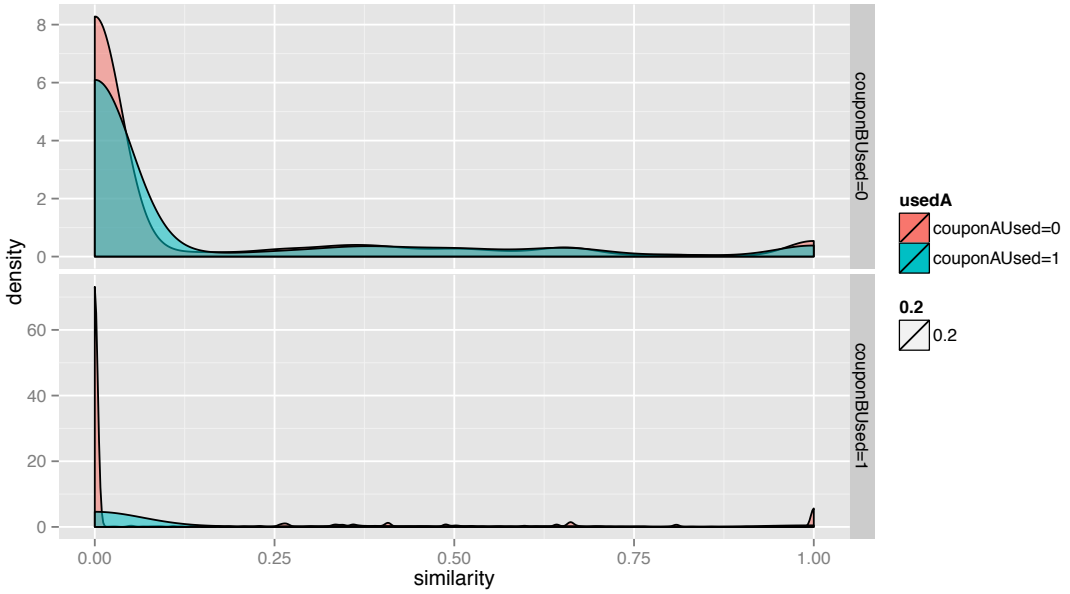


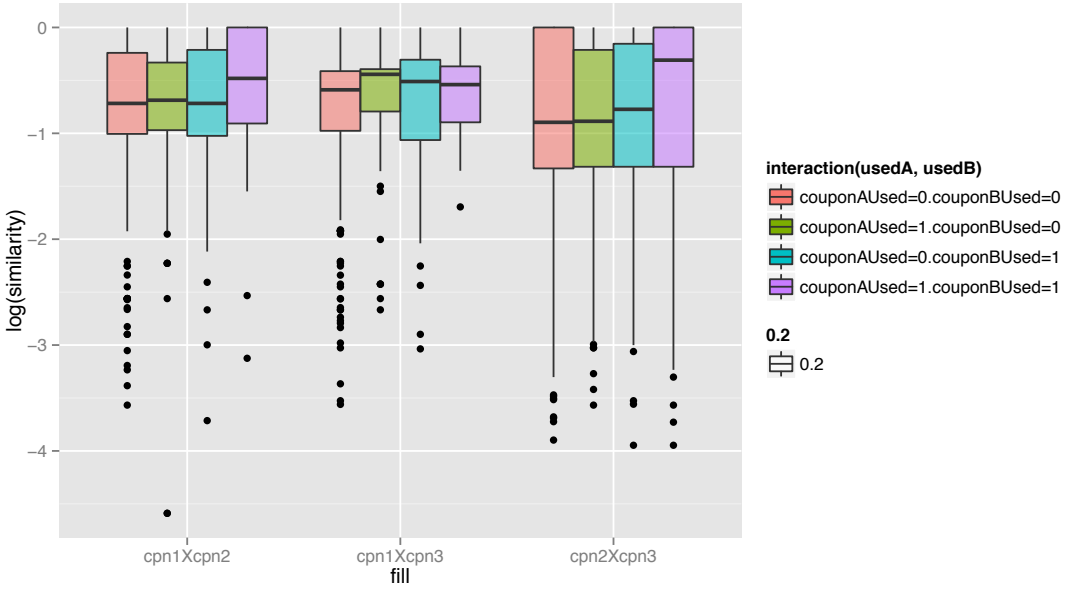
0.6.4 tf-idf using ftd and invmax

```
# similarity matrix
sim_mat = cos.d3

# figures
figures = simFeatures(sim_mat)

# print
print(figures$p1)
print(figures$p2)
```

```
## Warning in loop_apply(n, do.ply): Removed 13870 rows containing non-finite
## values (stat_boxplot).
```

```
# save
saveRDS(sim_mat, file = "../features/feature_files/universal/ftd_invmax_similarity.rds")
```



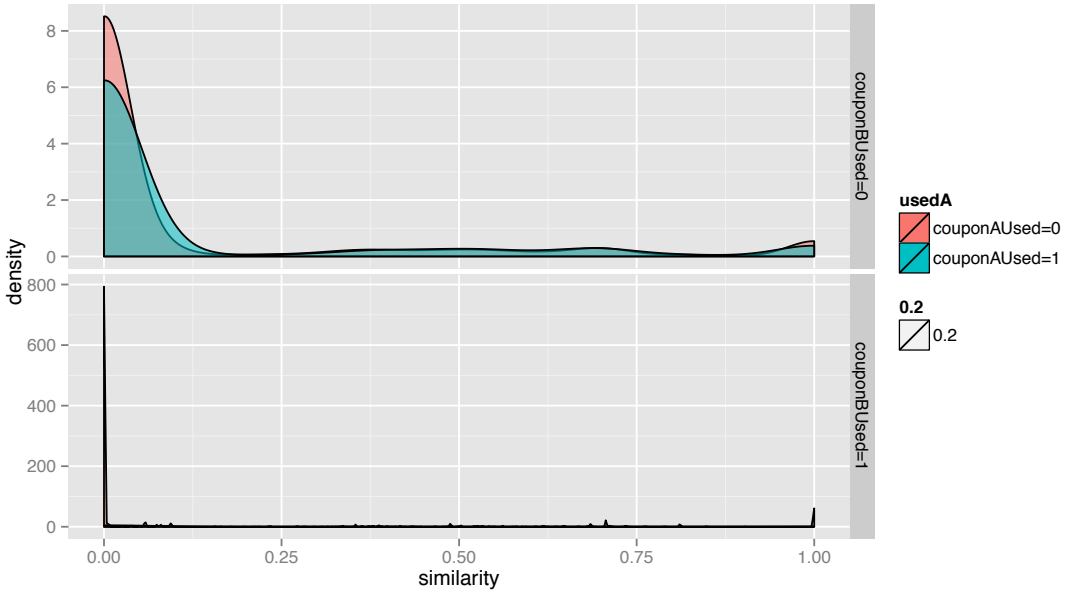


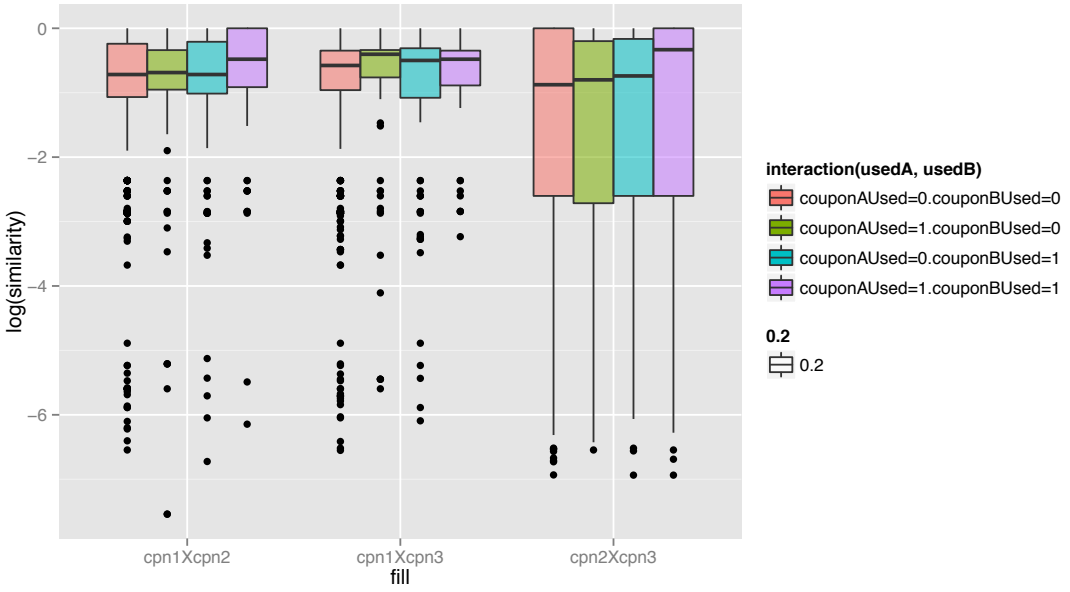
0.6.5 tf-idf using ftd and pif

```
# similarity matrix
sim_mat = cos.d4

# figures
figures = simFeatures(sim_mat)

# print
print(figures$p1)
print(figures$p2)
```

```
## Warning in loop_apply(n, do.ply): Removed 13870 rows containing non-finite
## values (stat_boxplot).
```

```
# save
saveRDS(sim_mat, file = "../features/feature_files/universal/ftd_pif_similarity.rds")
```





0.6.6 tf-idf using lnorm and lnorm

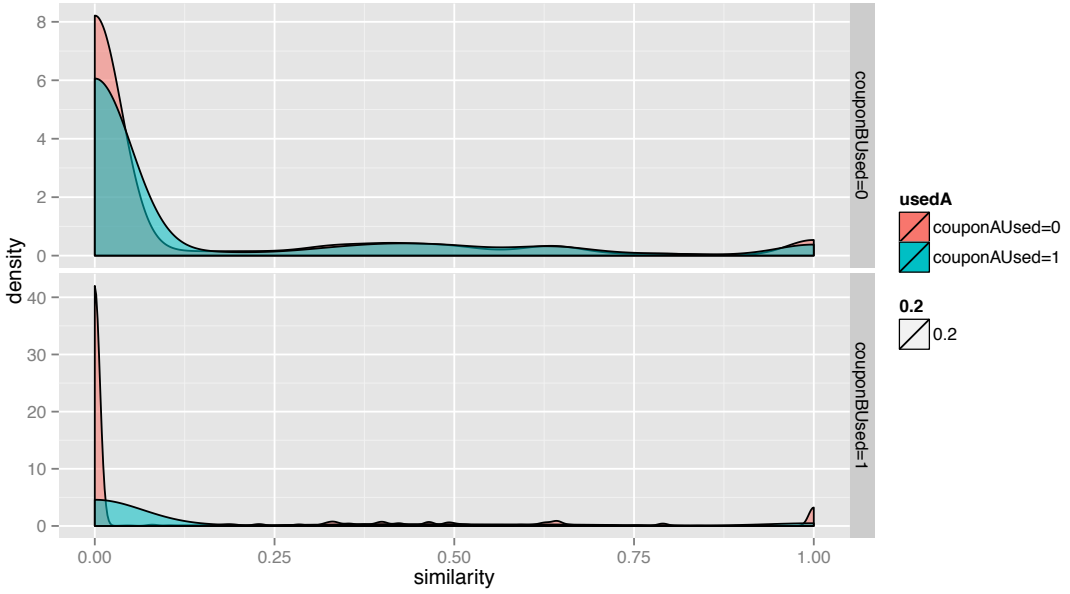
```
# similarity matrix
sim_mat = cos.d5
```

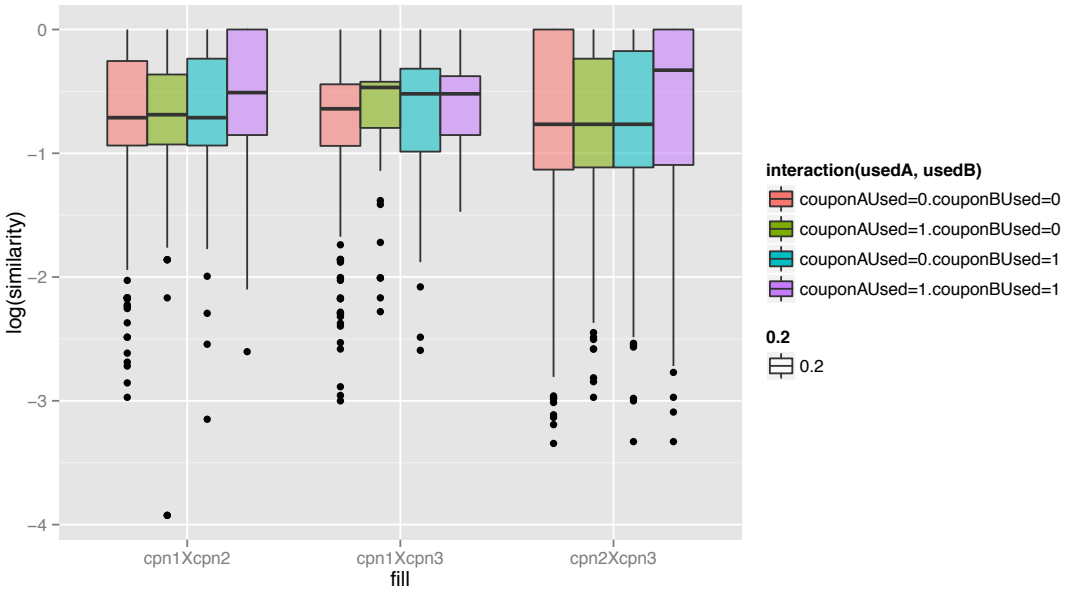
```
# figures
figures = simFeatures(sim_mat)
```

```
# print
print(figures$p1)
print(figures$p2)
```

```
## Warning in loop_apply(n, do.ply): Removed 13870 rows containing non-finite
## values (stat_boxplot).
```

```
# save
saveRDS(sim_mat, file = "../features/feature_files/universal/lnorm_lnorm_similarity.rds")
```

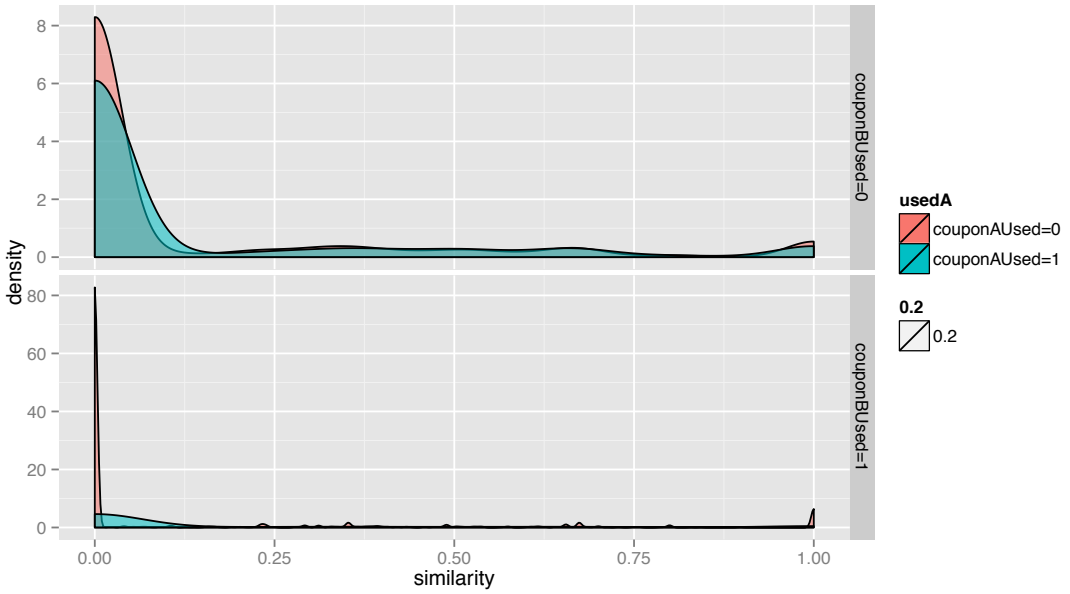


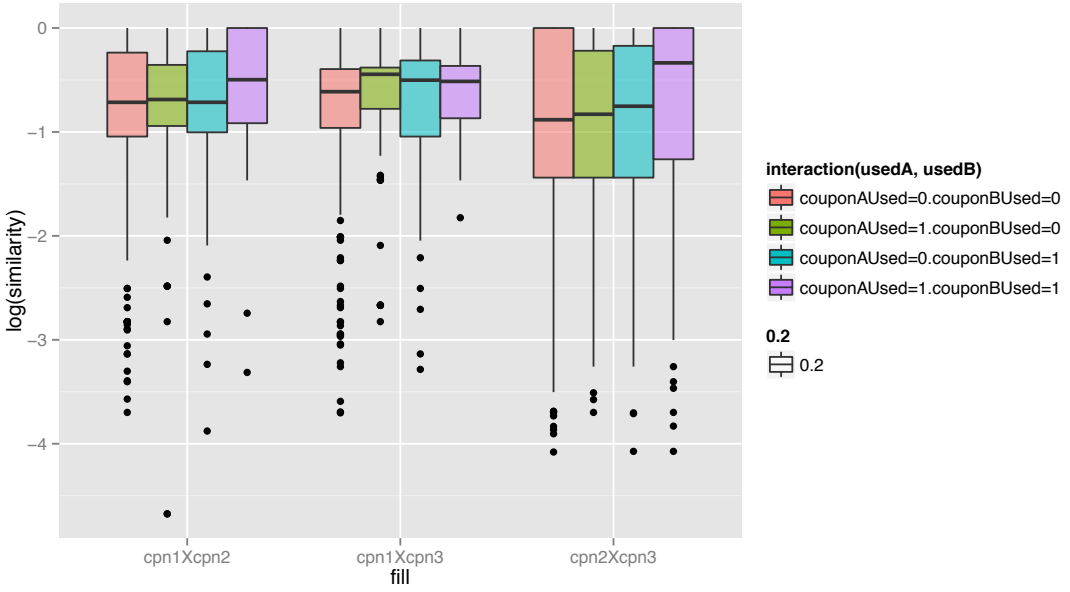


0.6.7 tf-idf using lnorm and ivf

```
# similarity matrix
sim_mat = cos.d6
```

```
# figures
figures = simFeatures(sim_mat)

# print
print(figures$p1)
print(figures$p2)

## Warning in loop_apply(n, do.ply): Removed 13870 rows containing non-finite
## values (stat_boxplot).

# save
saveRDS(sim_mat, file = "../features/feature_files/universal/lnorm_ivf_similarity.rds")
```



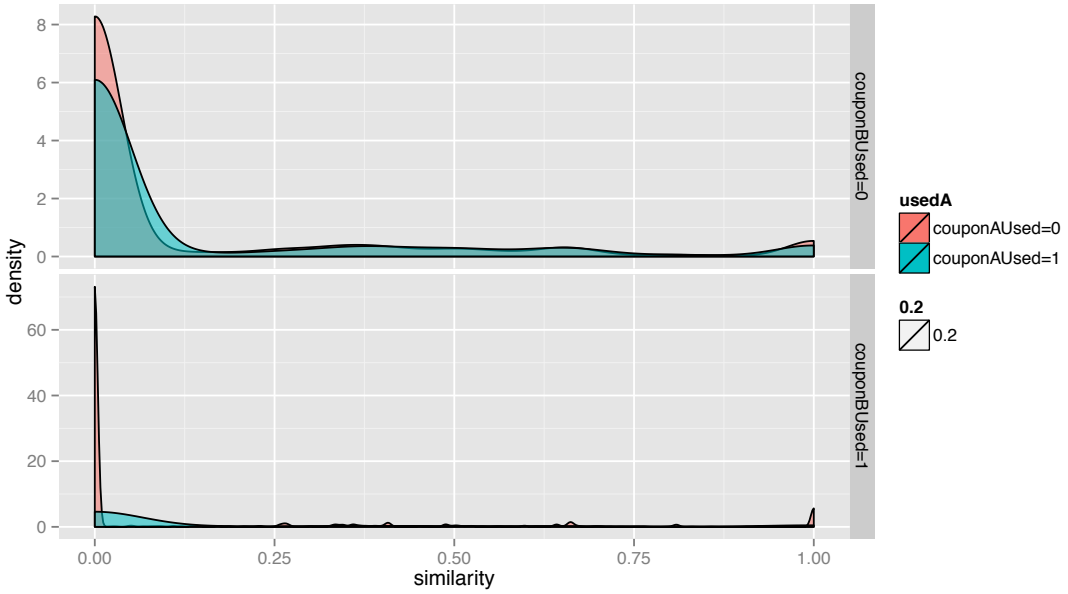


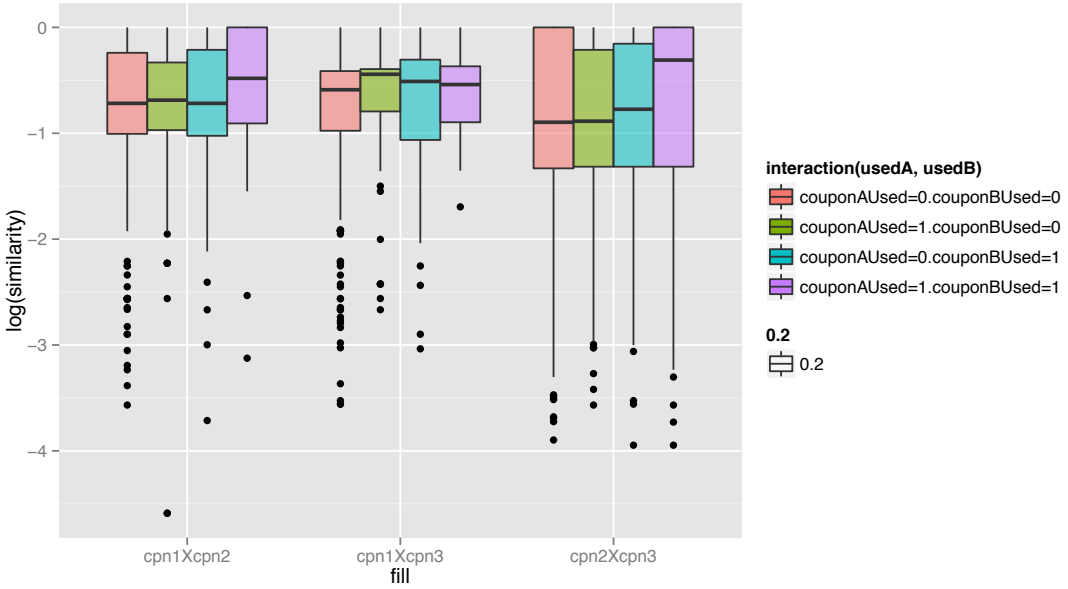
0.6.8 tf-idf using lnorm and invmax

```
# similarity matrix
sim_mat = cos.d7

# figures
figures = simFeatures(sim_mat)

# print
print(figures$p1)
print(figures$p2)
```

```
## Warning in loop_apply(n, do.ply): Removed 13870 rows containing non-finite
## values (stat_boxplot).
```

```
# save
saveRDS(sim_mat, file = "../features/feature_files/universal/lnorm_invmax_similarity.rds")
```





0.6.9 tf-idf using lnorm and pif

```
# similarity matrix
sim_mat = cos.d8

# figures
figures = simFeatures(sim_mat)

# print
print(figures$p1)
print(figures$p2)
```

```
## Warning in loop_apply(n, do.ply): Removed 13870 rows containing non-finite
## values (stat_boxplot).
```

```
# save
saveRDS(sim_mat, file = "../features/feature_files/universal/lnorm_pif_similarity.rds")
```