R package API

we will have separate csv/sdf files for meta, para etc SMILES but we may explore having just one SQLite file or CSV file for meta, para etc values by incorporating HT type and sigma key/index number.

Base level individual functions:

**Single prediction value**

getfragmentsmile {  
  
1.using fmcs check the sample smiles with every in the csv fragment list (as per user selection for meta, para, ortho sigma)  
#this corresponds to sorting order in sqlite table  
2.find the mcs index and substituted fragment as per whatever attribute user selects for MCS cutoff and mcs type  
3.return mcs index and the type of sigma (HT type), and a sigma value key/index number in our sqlite table in a dataframe or csv with that info  
  
}

getfragmentsmile (samplefragmentsmileobject, HT type=, MCS selection method=, MCS cuttoff=0, description = true/false, output type=)

where

* HT type type = meta, para, ortho, taft sigma, Es
* MCS selection method:
* HighestMCS with lowest weight: take the highest index value with lowest weight of the substituted fragment. In this type we will take the substituted fragment with highest mcs value and replace this value only if we find another fragment with higher mcs index value and ingore ones which are equal.
* tstHighest MCS with Highest weight: take the highest index value with highest weight of the substituted fragment. for example in case of a chain containing halogens, you may get same MCS value for a lower weight chain which may have less halogens.
* n highest: return mcs for user specified “n” highest substituted fragment values INCLUDING equal values
* just equal values: retun a list of fragments if they have equally highest MCS index match.
* MCS cutoff this is a number from 0-1 which specifies at what point below which we should throw an error saying MCS match could not be found. Reccomended setting for meta, para, taft is 0.5 and for sigma ortho is 0 (aka it means to ignore sigma ortho).
* output type is either a CSV file, tab delimited or a dataframe, (list or any other suitable object) with fragment, substituted fragment, MCS index,HT type, sigma value key, and description.

test<-fmcsBatch(sampleSDF[1], batchSDF[1:300], au=0, bu=0)  
testframe<-data.frame(test)  
testframe$Tanimoto\_Coefficient  
add columns to testframe to correspond with index numbers, molecular weight, SMILES as strings etc. After that, use col max for tanimoto column to identify top matches; and return a smaller dataframe with what user has requested in attributes.

using the output from above file:

getsigmavalue {  
  
  
4. using the subsituted fragment, we will use it as a "key" in sqlite table to only extract values which correspond to that fragment  
5.based on sigma type attribute, we will run a specific SQL query to get the value on the fly for a particular sigma value  
6. fill that our dataframe with the value in our sqlite table   
#sqlite table will be sorted in ascending order according to molecular weight with lowest to highest  
5. output the dataframe/file   
}

getsigmavalue (getfragmentfile object, HT type=, MCS selection method=, MCS cuttoff=0, sigma selection method =, output description=FALSE, output type=)

where

* HT type type = meta, para, ortho, taft sigma, Es
* sigma selection method can be
* A: duplicate value of highest occurance aka mode plus single values
* B: avg of distinct values
* C: Hydrowin 1st option plus duplicate value of highest occurance aka mode
* D: Hydrowin 1st option plus distinct avg.
* E: Hansch preffered first plus distinct avg for when hydrowin not available
* F: Hansch preffered first plus duplicate value of highest occurance aka mode plus single values
* G: hydrowin plus hansch preffered
* H: median of distinct values ??

Avg of distict value is available in SQL (ANSI standard)

* Output description (true/false): cell contains how the value was calculated (basically a line with prints all the user selected parameters)
* output type is either a CSV file, tab delimited or a dataframe with fragment, substituted fragment, MCS, sigma value, sigma value type description, and reference.

**Generating substituted fragment library for multiple samples**

getbatchmetasigma(samplefragmentsmilefile, type=, )

putput is a dataframe with fragment, substituted fragment, MCS, sigma value, sigma value type description, and reference.

### AD

The average distance of the test chemical from its five nearest neighbors in the training set is compared with a threshold, which is the 95th percentile of average distance of training chemicals from their five nearest neighbors. If the average distance of a test sample was lesser than or equal to the threshold value, the test sample was retained within the AD. Sahigara 2013 and zang 2017.

2.6 Calculation of k-nearest neighbors for Applicability Domain (AD)  
# The chemical space is the 600 FP bits selected by GA  
> Data<- LogPdata600BitsTraining[, 1:600]  
> Query<- LogPdata600BitsTest[, 1:600]  
# Load package FNN for calculating 5-nearest neighbors  
> library(FNN)  
> Dtraining<-knn.dist(Data, k=5, algorithm=c("kd\_tree", "cover\_tree", "CR", "brute"))  
# The distance of the test chemical from its five nearest neighbors in the training set  
> Dtest<-knnx.dist(Data, Query, k=5, algorithm=c("kd\_tree", "cover\_tree", "CR", "brute"))

### working code for getfragmentsigma

getfragmentsigma <- function(smile = "CC1=CC=CC(=C1)C(=O)N\*", HT.type = "test", sigma.selection = "A", ...){  
   
 # "..." indicates a and b for atoms and bonds mismatch from fmcsR::fmcsbatch; add this explicitly with default =0 instead of ...  
  
 sdf1 <- ChemmineR::smiles2sdf(smile)  
   
 if (HT.type == "meta") {  
 batchSDF <- ChemmineR::read.SDFset("meta.SDF")  
 molwt <- read.csv("meta\_mol\_wt.csv")  
 } else if (HT.type == "para") {  
 batchSDF <- ChemmineR::read.SDFset("para.SDF")  
 mol\_wt <- read.csv("para\_mol\_wt.csv")  
# inserting a test file fopr testing, remove in production code  
 } else if (HT.type == "test") {  
 batchSDF <- ChemmineR::read.SDFset("test.SDF")  
 molwt <- read.csv("test\_mol\_wt.csv")  
# end of test code  
 } else if (HT.type == "ortho") {  
 batchSDF <- ChemmineR::read.SDFset("ortho.SDF")  
 molwt <- read.csv("ortho\_mol\_wt.csv")  
 } else if (HT.type == "taft") {  
 batchSDF <- ChemmineR::read.SDFset("taft.SDF")  
 molwt <- read.csv("taft\_mol\_wt.csv")  
 } else if (HT.type == "es") {  
 batchSDF <- ChemmineR::read.SDFset("es.SDF")  
 molwt <- read.csv("es\_mol\_wt.csv")  
 } else if (HT.type == "induction") {  
 batchSDF <- ChemmineR::read.SDFset("induction.SDF")  
 molwt <- read.csv("induction\_wt.csv")  
 } else {  
 stop("Specify valid HT.type")  
 }  
   
 fmcsoutput < -fmcsR::fmcsBatch(sampleSDF[1], batchSDF)  
 fmcsoutputframe <- data.frame(fmcsoutput, molwt)  
 fmcsoutputframe <- fmcsoutputframe[order(-fmcsoutputframe$Tanimoto\_Coefficient),]  
 return(fmcsoutputframe)  
  
 if (sigma.selection =="A") {  
 # A: duplicate value of highest occurance aka mode plus single values  
 } else if (sigma.selection == "B") {  
 #B: avg of distinct values   
 } else if (sigma.selection == "C") {  
 #C: Hydrowin 1st option plus duplicate value of highest occurance aka mode  
 } else if (sigma.selection == "D") {  
 #D: Hydrowin 1st option plus distinct avg  
 } else if (sigma.selection == "E") {  
 #E: Hansch preffered first plus distinct avg for when hydrowin not available   
 } else if (sigma.selection == "F") {  
 #F: Hansch preffered first plus duplicate value of highest occurance aka mode plus single values  
 } else if (sigma.selection =="G") {  
 #G: hydrowin plus hansch preffered  
 } else if (sigma.selection == "H") {  
 #H: median of distinct values ??  
 } else {  
 stop ("Specify valid sigma.selection")  
 }  
   
}

### helper function to fill dataframe with sigma values

# fillqsardataframe ()  
 # returns a dataframe containing test chemicals with substituted fragments, mcs index values, functional group type (carbamate or acid ester)  
  
fillqsardataframe <- function (inputcsvfile = "test.csv", sigma.selection = "A", ...) {  
 # reading the csv file as a dataframe  
   
 qsardataframe <- read.csv("inputcsvfile")  
   
 # initializing the iterator  
   
 i = 1  
   
 while (qsardataframe$compoundsmile[i] =! NULL) {  
   
 if (qsardataframe$r1meta1[i] || qsardataframe$r1ortho1[i] || qsardataframe$r1para1[i] != NULL) {  
 # if the structure is aromatic, than we will set these to default values  
 qsardataframe$r1.sub.taft[i] <- "\*C1=CC=CC=C1"  
 qsardataframe$r1.sub.es[i] <- "\*C1=CC=CC=C1"  
 qsardataframe$r1.taft.mcs[i] = 1  
 qsardataframe$r1.es.mcs[i] = 1  
 qsardataframe$r1.taft.value[i] = 0.1 # check value again  
 qsardataframe$r1.es.value[i] = 0.1 # check value again  
   
 } else {  
 # if its not aromatic, than we will need to call getfragmentsigma   
 # coercing dataframe object smiles into a list  
 x <- list (qsardataframe$r1taft[i])  
 # coercing the list containign smiles into ChemmineR's SMIset  
 y <- as(x, "SMIset")  
 #converting SMIset into SDFSet required for evaluation in getfragmentsigma  
   
 z <- ChemmineR::smiles2sdf(y)  
   
 # getting sigma value from getsigmavalue function for r1taft  
   
 t <- getfragmentsigma (smile = z, HT.type = "taft")  
   
 # getting taft steric factor for r1taft  
   
 es <- getfragmentsigma (smile = z, HT.type = "es")  
   
 qsardataframe$r1.sub.taft[i] <- t$fragment[1]  
 qsardataframe$r1.sub.es[i] <- es$fragment[1]  
 qsardataframe$r1.taft.mcs[i] <- t$Tanimoto\_Coefficient[1]  
 qsardataframe$r1.es.mcs[i] <- es$Tanimoto\_Coefficient[1]  
 qsardataframe$r1.taft.value[i] <- t$sigmavalue[1]  
 qsardataframe$r1.es.value[i] <- es$sigmavalue[1]  
 }  
   
 if (qsardataframe$r2meta1[i] || qsardataframe$r2ortho1[i] || qsardataframe$r2para1[i] =! NULL) {  
 qsardataframe$r1.sub.taft[i] <- "\*C1=CC=CC=C1"  
 qsardataframe$r1.sub.es[i] <- "\*C1=CC=CC=C1"  
 qsardataframe$r1.taft.mcs[i] = 1  
 qsardataframe$r1.es.mcs[i] = 1  
 qsardataframe$r1.taft.value[i] = 0.12  
 qsardataframe$r1.es.value[i] = es$sigmavalue[1]  
   
 } else {  
 # getting sigma value from getsigmavalue function for r2taft  
 t <- getfragmentsigma (smile = qsardataframe$r2taft[i], HT.type = "taft")  
   
 # getting taft steric factor for r1taft  
   
 es <- getfragmentsigma (smile = qsardataframe$r2taft[i], HT.type = "es")  
   
 qsardataframe$r2.sub.taft[i] <- t$fragment[1]  
 qsardataframe$r2.sub.es[i] <- es$fragment[1]  
 qsardataframe$r2.taft.mcs[i] <- t$Tanimoto\_Coefficient[1]  
 qsardataframe$r2.es.mcs[i] <- es$Tanimoto\_Coefficient[1]  
 qsardataframe$r2.taft.value[i] <- t$sigmavalue[1]  
 qsardataframe$r2.es.value[i] <- es$sigmavalue[1]  
 }  
   
 if (qsardataframe$r1meta1[i] =! NULL) {  
 # getting sigma value from getsigmavalue function for r1meta1  
   
 } else if (qsardataframe$r1meta2[i] =! NULL) {  
 # getting sigma value from getsigmavalue function for r1meta2  
   
 } else if (qsardataframe$r2meta1[i] =! NULL) {  
 # getting sigma value from getsigmavalue function for r2meta1  
   
 } else if (qsardataframe$r2meta2[i] =! NULL) {  
 # getting sigma value from getsigmavalue function for r2meta2  
   
 } else if (qsardataframe$r1ortho1[i] =! NULL) {  
 # getting sigma value from getsigmavalue function for r1ortho1  
   
 } else if (qsardataframe$r1ortho2[i] =! NULL) {  
 # getting sigma value from getsigmavalue function for r1ortho2  
   
 } else if (qsardataframe$r2ortho1[i] =! NULL) {  
 # getting sigma value from getsigmavalue function for r2ortho1  
   
 } else if (qsardataframe$r2ortho2 [i] =! NULL) {  
 # getting sigma value from getsigmavalue function for r2ortho2  
   
 } else if (qsardataframe$r1para1 [i] =! NULL) {  
 # getting sigma value from getsigmavalue function for r1para1  
   
 } else if (qsardataframe$r2para1[i] =! NULL) {  
 # getting sigma value from getsigmavalue function for r2para1  
   
 }  
   
 # moving to next chemical  
 i++  
 }  
   
}

### Carbamate QSAR function

# creating a function with these parameters  
# gethydrolysisrate (inputcsvfile, MCS.cuttoff=0,onlyortho.cutoff=0, regression.method = "SVR", sigma.selection = "A", outputtype = CSV or R dataframe, ...)  
  
get.hydrolysisrate <- function (inputcsvfile = "test.csv", sigma.selection = "A", MCS.cuttoff=0, regression.method = "SVR", outputtype = CSV or R dataframe, ...){  
 # Calling the helper function to autofill a dataframe which will become test set for our QSARs  
 qsardataframe <- fillqsardataframe (inputcsvfile, sigma.selection = "A", ...)  
   
 # Evaluating if the autofilled dataframe does in fact have sigma value higher than the user specified cutoff  
 if (qsardataframe$meta1mcs || meta2mcs || r1taftmcs.... < MCS.cuttoff) {  
 stop ("qsar evaluation terminated since one of the fragments has mcs tanimoto coefficient value below the user specified MCS cuttoff value")  
 }  
   
 # Evaluating if the autofilled dataframe does in fact have sigma value higher than the user specified cutoff  
 if (qsardataframe$ortho1mcs || ortho2mcs ||...... < onlyortho.cuttoff) {  
 stop ("qsar evaluation terminated since ortho fragments have mcs tanimoto coefficient value below the user specified MCS cuttoff value")  
 }  
   
 # Based on functional group label in qsardataframe, subsetting the master sheet to only keep those variables necassary for a particular qsar  
 if (qsardataframe$funcgroup == "carbamate") {  
 # subset the dataframe to only keep variables used for carbamates and assign it as test set  
 } else if (qsardataframe$funcgroup == "acidester") {  
 # subset the dataframe to only keep variables used for carboxylic acid esters and assign it as test set  
 } else {  
 stop ("given file doesnt contain a functional group with available QSAR model")  
 }  
   
 if (regression.method =="SVR") {  
 # support vector regression using e1071 library   
 # call AD function  
 } else if (regression.method == "pls") {  
 # partial least squares using pls library   
 # call AD function  
 } else if (regression.method == "RF") {  
 # random forest regression using randomforest library  
 # call AD function  
 } else if (regression.method == "MLR") {  
 # multiple linear regression using base R linear model functions  
 # call AD function  
 } else {  
 stop ("Specify valid regression.method")  
 }

### Old vs new logic

Old logic for obtaining substituted fragment with highest tanimoto mcs index:

MCS = batchSDF[1]  
Tanimoto=0  
Index=0  
for(i in 1:300)  
  
{  
   
 test3<-fmcs(batchSDF[i], sampleSDF[1], fast=TRUE)  
 s<-parse\_number(test3[4])  
   
 if (s > Tanimoto)  
   
 {  
 Tanimoto=s  
 MCS=batchSDF[i]  
 Index=i   
 }  
}  
Tanimoto  
Index  
smiles<-sdf2smiles(MCS[1])  
  
write.SMI(smiles, file="smiles.smi")

#### New logic:

testfun<-function(sdf1, sdf2, ...){  
  
 batchSDF<-ChemmineR::read.SDFset(sdfstr=sdf1)  
 sampleSDF<-ChemmineR::read.SDFset(sdfstr=sdf2)  
 test<-fmcsR::fmcsBatch(sampleSDF[1], batchSDF[1:300])  
 testframe<-data.frame(test)  
 return(testframe)  
}  
testfun()

with all parameters:

testfun<-function(sdf1, sdf2, molwtcsv, HTtype, MCSselectionmethod, MCScuttoff=0, description = TRUE, outputtype, ...){  
   
 batchSDF<-ChemmineR::read.SDFset(sdfstr=sdf1)  
 sampleSDF<-ChemmineR::read.SDFset(sdfstr=sdf2)  
 test<-fmcsR::fmcsBatch(sampleSDF[1], batchSDF[1:300])  
 testframe<-data.frame(test)  
 return(testframe)  
}  
testfun(sdf1 = "batch2.SDF", sdf2 = "sample.SDF", au=1, bu=0)

## Query\_Size Target\_Size MCS\_Size Tanimoto\_Coefficient  
## CMP1 11 1 1 0.09090909  
## CMP2 11 2 1 0.08333333  
## CMP3 11 2 1 0.08333333  
## CMP4 11 2 1 0.08333333  
## CMP5 11 2 1 0.08333333  
## CMP6 11 4 2 0.15384615  
## CMP7 11 3 1 0.07692308  
## CMP8 11 2 1 0.08333333  
## CMP9 11 3 2 0.16666667  
## CMP10 11 4 3 0.25000000  
## CMP11 11 5 3 0.23076923  
## CMP12 11 6 3 0.21428571  
## CMP13 11 7 3 0.20000000  
## CMP14 11 8 3 0.18750000  
## CMP15 11 9 3 0.17647059  
## CMP16 11 5 3 0.23076923  
## CMP17 11 6 3 0.21428571  
## CMP18 11 7 3 0.20000000  
## CMP19 11 7 3 0.20000000  
## CMP20 11 6 3 0.21428571  
## CMP21 11 8 7 0.58333333  
## CMP22 11 8 3 0.18750000  
## CMP23 11 9 7 0.53846154  
## CMP24 11 6 4 0.30769231  
## CMP25 11 6 4 0.30769231  
## CMP26 11 4 3 0.25000000  
## CMP27 11 5 4 0.33333333  
## CMP28 11 3 1 0.07692308  
## CMP29 11 3 1 0.07692308  
## CMP30 11 4 1 0.07142857  
## CMP31 11 4 1 0.07142857  
## CMP32 11 5 2 0.14285714  
## CMP33 11 6 3 0.21428571  
## CMP34 11 5 2 0.14285714  
## CMP35 11 5 3 0.23076923  
## CMP36 11 3 1 0.07692308  
## CMP37 11 3 1 0.07692308  
## CMP38 11 3 1 0.07692308  
## CMP39 11 3 1 0.07692308  
## CMP40 11 4 2 0.15384615  
## CMP41 11 5 3 0.23076923  
## CMP42 11 5 3 0.23076923  
## CMP43 11 4 2 0.15384615  
## CMP44 11 4 2 0.15384615  
## CMP45 11 5 2 0.14285714  
## CMP46 11 6 2 0.13333333  
## CMP47 11 6 2 0.13333333  
## CMP48 11 4 2 0.15384615  
## CMP49 11 5 3 0.23076923  
## CMP50 11 3 2 0.16666667  
## CMP51 11 5 2 0.14285714  
## CMP52 11 6 4 0.30769231  
## CMP53 11 5 2 0.14285714  
## CMP54 11 6 3 0.21428571  
## CMP55 11 9 6 0.42857143  
## CMP56 11 3 1 0.07692308  
## CMP57 11 4 1 0.07142857  
## CMP58 11 5 1 0.06666667  
## CMP59 11 7 1 0.05882353  
## CMP60 11 5 1 0.06666667  
## CMP61 11 6 1 0.06250000  
## CMP62 11 4 3 0.25000000  
## CMP63 11 5 3 0.23076923  
## CMP64 11 5 3 0.23076923  
## CMP65 11 6 3 0.21428571  
## CMP66 11 5 4 0.33333333  
## CMP67 11 6 4 0.30769231  
## CMP68 11 6 4 0.30769231  
## CMP69 11 4 3 0.25000000  
## CMP70 11 5 3 0.23076923  
## CMP71 11 6 3 0.21428571  
## CMP72 11 7 3 0.20000000  
## CMP73 11 7 3 0.20000000  
## CMP74 11 7 3 0.20000000  
## CMP75 11 8 3 0.18750000  
## CMP76 11 9 3 0.17647059  
## CMP77 11 6 3 0.21428571  
## CMP78 11 8 3 0.18750000  
## CMP79 11 8 3 0.18750000  
## CMP80 11 9 3 0.17647059  
## CMP81 11 7 3 0.20000000  
## CMP82 11 8 3 0.18750000  
## CMP83 11 9 3 0.17647059  
## CMP84 11 10 3 0.16666667  
## CMP85 11 10 3 0.16666667  
## CMP86 11 12 3 0.15000000  
## CMP87 11 10 3 0.16666667  
## CMP88 11 11 3 0.15789474  
## CMP89 11 9 7 0.53846154  
## CMP90 11 10 7 0.50000000  
## CMP91 11 14 7 0.38888889  
## CMP92 11 5 4 0.33333333  
## CMP93 11 5 3 0.23076923  
## CMP94 11 6 3 0.21428571  
## CMP95 11 4 1 0.07142857  
## CMP96 11 4 1 0.07142857  
## CMP97 11 4 1 0.07142857  
## CMP98 11 4 1 0.07142857  
## CMP99 11 4 2 0.15384615  
## CMP100 11 4 2 0.15384615  
## CMP101 11 4 2 0.15384615  
## CMP102 11 5 2 0.14285714  
## CMP103 11 4 2 0.15384615  
## CMP104 11 5 3 0.23076923  
## CMP105 11 6 3 0.21428571  
## CMP106 11 7 3 0.20000000  
## CMP107 11 9 7 0.53846154  
## CMP108 11 5 2 0.14285714  
## CMP109 11 4 1 0.07142857  
## CMP110 11 4 1 0.07142857  
## CMP111 11 5 2 0.14285714  
## CMP112 11 6 3 0.21428571  
## CMP113 11 7 2 0.12500000  
## CMP114 11 4 2 0.15384615  
## CMP115 11 5 3 0.23076923  
## CMP116 11 6 3 0.21428571  
## CMP117 11 7 3 0.20000000  
## CMP118 11 8 3 0.18750000  
## CMP119 11 5 3 0.23076923  
## CMP120 11 6 3 0.21428571  
## CMP121 11 7 3 0.20000000  
## CMP122 11 8 3 0.18750000  
## CMP123 11 9 3 0.17647059  
## CMP124 11 12 3 0.15000000  
## CMP125 11 9 3 0.17647059  
## CMP126 11 10 3 0.16666667  
## CMP127 11 11 3 0.15789474  
## CMP128 11 9 3 0.17647059  
## CMP129 11 10 3 0.16666667  
## CMP130 11 10 3 0.16666667  
## CMP131 11 8 3 0.18750000  
## CMP132 11 11 3 0.15789474  
## CMP133 11 15 7 0.36842105  
## CMP134 11 16 7 0.35000000  
## CMP135 11 20 7 0.29166667  
## CMP136 11 5 3 0.23076923  
## CMP137 11 6 3 0.21428571  
## CMP138 11 5 3 0.23076923  
## CMP139 11 5 2 0.14285714  
## CMP140 11 5 1 0.06666667  
## CMP141 11 5 1 0.06666667  
## CMP142 11 5 1 0.06666667  
## CMP143 11 5 1 0.06666667  
## CMP144 11 5 1 0.06666667  
## CMP145 11 8 2 0.11764706  
## CMP146 11 3 2 0.16666667  
## CMP147 11 4 3 0.25000000  
## CMP148 11 4 3 0.25000000  
## CMP149 11 5 4 0.33333333  
## CMP150 11 5 4 0.33333333  
## CMP151 11 5 4 0.33333333  
## CMP152 11 6 4 0.30769231  
## CMP153 11 6 4 0.30769231  
## CMP154 11 8 4 0.26666667  
## CMP155 11 4 2 0.15384615  
## CMP156 11 4 2 0.15384615  
## CMP157 11 5 2 0.14285714  
## CMP158 11 5 2 0.14285714  
## CMP159 11 5 3 0.23076923  
## CMP160 11 7 4 0.28571429  
## CMP161 11 6 2 0.13333333  
## CMP162 11 9 7 0.53846154  
## CMP163 11 4 3 0.25000000  
## CMP164 11 5 4 0.33333333  
## CMP165 11 6 2 0.13333333  
## CMP166 11 6 2 0.13333333  
## CMP167 11 9 5 0.33333333  
## CMP168 11 3 1 0.07692308  
## CMP169 11 4 2 0.15384615  
## CMP170 11 9 7 0.53846154  
## CMP171 11 7 2 0.12500000  
## CMP172 11 3 2 0.16666667  
## CMP173 11 4 2 0.15384615  
## CMP174 11 6 2 0.13333333  
## CMP175 11 4 3 0.25000000  
## CMP176 11 6 4 0.30769231  
## CMP177 11 9 8 0.66666667  
## CMP178 11 7 3 0.20000000  
## CMP179 11 4 3 0.25000000  
## CMP180 11 5 3 0.23076923  
## CMP181 11 10 6 0.40000000  
## CMP182 11 5 2 0.14285714  
## CMP183 11 6 2 0.13333333  
## CMP184 11 7 6 0.50000000  
## CMP185 11 11 7 0.46666667  
## CMP186 11 15 8 0.44444444  
## CMP187 11 19 7 0.30434783  
## CMP188 11 6 4 0.30769231  
## CMP189 11 6 4 0.30769231  
## CMP190 11 2 2 0.18181818  
## CMP191 11 3 3 0.27272727  
## CMP192 11 4 4 0.36363636  
## CMP193 11 5 5 0.45454545  
## CMP194 11 6 5 0.41666667  
## CMP195 11 8 7 0.58333333  
## CMP196 11 4 4 0.36363636  
## CMP197 11 5 5 0.45454545  
## CMP198 11 10 10 0.90909091  
## CMP199 11 7 4 0.28571429  
## CMP200 11 5 4 0.33333333  
## CMP201 11 3 2 0.16666667  
## CMP202 11 3 2 0.16666667  
## CMP203 11 4 3 0.25000000  
## CMP204 11 6 4 0.30769231  
## CMP205 11 6 5 0.41666667  
## CMP206 11 2 1 0.08333333  
## CMP207 11 3 1 0.07692308  
## CMP208 11 4 2 0.15384615  
## CMP209 11 5 3 0.23076923  
## CMP210 11 6 3 0.21428571  
## CMP211 11 7 3 0.20000000  
## CMP212 11 5 3 0.23076923  
## CMP213 11 5 3 0.23076923  
## CMP214 11 5 2 0.14285714  
## CMP215 11 8 3 0.18750000  
## CMP216 11 8 6 0.46153846  
## CMP217 11 5 3 0.23076923  
## CMP218 11 8 3 0.18750000  
## CMP219 11 5 1 0.06666667  
## CMP220 11 6 1 0.06250000  
## CMP221 11 4 1 0.07142857  
## CMP222 11 4 1 0.07142857  
## CMP223 11 5 1 0.06666667  
## CMP224 11 5 1 0.06666667  
## CMP225 11 6 1 0.06250000  
## CMP226 11 4 1 0.07142857  
## CMP227 11 6 1 0.06250000  
## CMP228 11 4 1 0.07142857  
## CMP229 11 6 2 0.13333333  
## CMP230 11 6 1 0.06250000  
## CMP231 11 6 1 0.06250000  
## CMP232 11 8 2 0.11764706  
## CMP233 11 11 6 0.37500000  
## CMP234 11 14 6 0.31578947  
## CMP235 11 4 1 0.07142857  
## CMP236 11 4 1 0.07142857  
## CMP237 11 5 1 0.06666667  
## CMP238 11 5 1 0.06666667  
## CMP239 11 5 1 0.06666667  
## CMP240 11 5 1 0.06666667  
## CMP241 11 11 3 0.15789474  
## CMP242 11 7 1 0.05882353  
## CMP243 11 9 2 0.11111111  
## CMP244 11 11 3 0.15789474  
## CMP245 11 15 6 0.30000000  
## CMP246 11 5 1 0.06666667  
## CMP247 11 5 1 0.06666667  
## CMP248 11 15 6 0.30000000  
## CMP249 11 2 1 0.08333333  
## CMP250 11 3 1 0.07692308  
## CMP251 11 4 2 0.15384615  
## CMP252 11 5 3 0.23076923  
## CMP253 11 6 3 0.21428571  
## CMP254 11 7 3 0.20000000  
## CMP255 11 5 3 0.23076923  
## CMP256 11 9 7 0.53846154  
## CMP257 11 8 6 0.46153846  
## CMP258 11 8 3 0.18750000  
## CMP259 11 4 2 0.15384615  
## CMP260 11 5 3 0.23076923  
## CMP261 11 4 1 0.07142857  
## CMP262 11 4 1 0.07142857  
## CMP263 11 4 1 0.07142857  
## CMP264 11 5 1 0.06666667  
## CMP265 11 3 1 0.07692308  
## CMP266 11 6 1 0.06250000  
## CMP267 11 5 2 0.14285714  
## CMP268 11 4 1 0.07142857  
## CMP269 11 5 1 0.06666667  
## CMP270 11 5 3 0.23076923  
## CMP271 11 5 3 0.23076923  
## CMP272 11 5 1 0.06666667  
## CMP273 11 5 1 0.06666667  
## CMP274 11 5 1 0.06666667  
## CMP275 11 4 1 0.07142857  
## CMP276 11 6 2 0.13333333  
## CMP277 11 5 2 0.14285714  
## CMP278 11 7 3 0.20000000  
## CMP279 11 6 3 0.21428571  
## CMP280 11 5 1 0.06666667  
## CMP281 11 4 1 0.07142857  
## CMP282 11 5 1 0.06666667  
## CMP283 11 4 1 0.07142857  
## CMP284 11 8 1 0.05555556  
## CMP285 11 7 1 0.05882353  
## CMP286 11 7 3 0.20000000  
## CMP287 11 6 3 0.21428571  
## CMP288 11 5 1 0.06666667  
## CMP289 11 4 1 0.07142857  
## CMP290 11 6 2 0.13333333  
## CMP291 11 5 2 0.14285714  
## CMP292 11 7 2 0.12500000  
## CMP293 11 6 2 0.13333333  
## CMP294 11 10 6 0.40000000  
## CMP295 11 9 6 0.42857143  
## CMP296 11 6 2 0.13333333  
## CMP297 11 6 1 0.06250000  
## CMP298 11 5 1 0.06666667  
## CMP299 11 6 1 0.06250000  
## CMP300 11 7 1 0.05882353  
## Overlap\_Coefficient  
## CMP1 1.0000000  
## CMP2 0.5000000  
## CMP3 0.5000000  
## CMP4 0.5000000  
## CMP5 0.5000000  
## CMP6 0.5000000  
## CMP7 0.3333333  
## CMP8 0.5000000  
## CMP9 0.6666667  
## CMP10 0.7500000  
## CMP11 0.6000000  
## CMP12 0.5000000  
## CMP13 0.4285714  
## CMP14 0.3750000  
## CMP15 0.3333333  
## CMP16 0.6000000  
## CMP17 0.5000000  
## CMP18 0.4285714  
## CMP19 0.4285714  
## CMP20 0.5000000  
## CMP21 0.8750000  
## CMP22 0.3750000  
## CMP23 0.7777778  
## CMP24 0.6666667  
## CMP25 0.6666667  
## CMP26 0.7500000  
## CMP27 0.8000000  
## CMP28 0.3333333  
## CMP29 0.3333333  
## CMP30 0.2500000  
## CMP31 0.2500000  
## CMP32 0.4000000  
## CMP33 0.5000000  
## CMP34 0.4000000  
## CMP35 0.6000000  
## CMP36 0.3333333  
## CMP37 0.3333333  
## CMP38 0.3333333  
## CMP39 0.3333333  
## CMP40 0.5000000  
## CMP41 0.6000000  
## CMP42 0.6000000  
## CMP43 0.5000000  
## CMP44 0.5000000  
## CMP45 0.4000000  
## CMP46 0.3333333  
## CMP47 0.3333333  
## CMP48 0.5000000  
## CMP49 0.6000000  
## CMP50 0.6666667  
## CMP51 0.4000000  
## CMP52 0.6666667  
## CMP53 0.4000000  
## CMP54 0.5000000  
## CMP55 0.6666667  
## CMP56 0.3333333  
## CMP57 0.2500000  
## CMP58 0.2000000  
## CMP59 0.1428571  
## CMP60 0.2000000  
## CMP61 0.1666667  
## CMP62 0.7500000  
## CMP63 0.6000000  
## CMP64 0.6000000  
## CMP65 0.5000000  
## CMP66 0.8000000  
## CMP67 0.6666667  
## CMP68 0.6666667  
## CMP69 0.7500000  
## CMP70 0.6000000  
## CMP71 0.5000000  
## CMP72 0.4285714  
## CMP73 0.4285714  
## CMP74 0.4285714  
## CMP75 0.3750000  
## CMP76 0.3333333  
## CMP77 0.5000000  
## CMP78 0.3750000  
## CMP79 0.3750000  
## CMP80 0.3333333  
## CMP81 0.4285714  
## CMP82 0.3750000  
## CMP83 0.3333333  
## CMP84 0.3000000  
## CMP85 0.3000000  
## CMP86 0.2727273  
## CMP87 0.3000000  
## CMP88 0.2727273  
## CMP89 0.7777778  
## CMP90 0.7000000  
## CMP91 0.6363636  
## CMP92 0.8000000  
## CMP93 0.6000000  
## CMP94 0.5000000  
## CMP95 0.2500000  
## CMP96 0.2500000  
## CMP97 0.2500000  
## CMP98 0.2500000  
## CMP99 0.5000000  
## CMP100 0.5000000  
## CMP101 0.5000000  
## CMP102 0.4000000  
## CMP103 0.5000000  
## CMP104 0.6000000  
## CMP105 0.5000000  
## CMP106 0.4285714  
## CMP107 0.7777778  
## CMP108 0.4000000  
## CMP109 0.2500000  
## CMP110 0.2500000  
## CMP111 0.4000000  
## CMP112 0.5000000  
## CMP113 0.2857143  
## CMP114 0.5000000  
## CMP115 0.6000000  
## CMP116 0.5000000  
## CMP117 0.4285714  
## CMP118 0.3750000  
## CMP119 0.6000000  
## CMP120 0.5000000  
## CMP121 0.4285714  
## CMP122 0.3750000  
## CMP123 0.3333333  
## CMP124 0.2727273  
## CMP125 0.3333333  
## CMP126 0.3000000  
## CMP127 0.2727273  
## CMP128 0.3333333  
## CMP129 0.3000000  
## CMP130 0.3000000  
## CMP131 0.3750000  
## CMP132 0.2727273  
## CMP133 0.6363636  
## CMP134 0.6363636  
## CMP135 0.6363636  
## CMP136 0.6000000  
## CMP137 0.5000000  
## CMP138 0.6000000  
## CMP139 0.4000000  
## CMP140 0.2000000  
## CMP141 0.2000000  
## CMP142 0.2000000  
## CMP143 0.2000000  
## CMP144 0.2000000  
## CMP145 0.2500000  
## CMP146 0.6666667  
## CMP147 0.7500000  
## CMP148 0.7500000  
## CMP149 0.8000000  
## CMP150 0.8000000  
## CMP151 0.8000000  
## CMP152 0.6666667  
## CMP153 0.6666667  
## CMP154 0.5000000  
## CMP155 0.5000000  
## CMP156 0.5000000  
## CMP157 0.4000000  
## CMP158 0.4000000  
## CMP159 0.6000000  
## CMP160 0.5714286  
## CMP161 0.3333333  
## CMP162 0.7777778  
## CMP163 0.7500000  
## CMP164 0.8000000  
## CMP165 0.3333333  
## CMP166 0.3333333  
## CMP167 0.5555556  
## CMP168 0.3333333  
## CMP169 0.5000000  
## CMP170 0.7777778  
## CMP171 0.2857143  
## CMP172 0.6666667  
## CMP173 0.5000000  
## CMP174 0.3333333  
## CMP175 0.7500000  
## CMP176 0.6666667  
## CMP177 0.8888889  
## CMP178 0.4285714  
## CMP179 0.7500000  
## CMP180 0.6000000  
## CMP181 0.6000000  
## CMP182 0.4000000  
## CMP183 0.3333333  
## CMP184 0.8571429  
## CMP185 0.6363636  
## CMP186 0.7272727  
## CMP187 0.6363636  
## CMP188 0.6666667  
## CMP189 0.6666667  
## CMP190 1.0000000  
## CMP191 1.0000000  
## CMP192 1.0000000  
## CMP193 1.0000000  
## CMP194 0.8333333  
## CMP195 0.8750000  
## CMP196 1.0000000  
## CMP197 1.0000000  
## CMP198 1.0000000  
## CMP199 0.5714286  
## CMP200 0.8000000  
## CMP201 0.6666667  
## CMP202 0.6666667  
## CMP203 0.7500000  
## CMP204 0.6666667  
## CMP205 0.8333333  
## CMP206 0.5000000  
## CMP207 0.3333333  
## CMP208 0.5000000  
## CMP209 0.6000000  
## CMP210 0.5000000  
## CMP211 0.4285714  
## CMP212 0.6000000  
## CMP213 0.6000000  
## CMP214 0.4000000  
## CMP215 0.3750000  
## CMP216 0.7500000  
## CMP217 0.6000000  
## CMP218 0.3750000  
## CMP219 0.2000000  
## CMP220 0.1666667  
## CMP221 0.2500000  
## CMP222 0.2500000  
## CMP223 0.2000000  
## CMP224 0.2000000  
## CMP225 0.1666667  
## CMP226 0.2500000  
## CMP227 0.1666667  
## CMP228 0.2500000  
## CMP229 0.3333333  
## CMP230 0.1666667  
## CMP231 0.1666667  
## CMP232 0.2500000  
## CMP233 0.5454545  
## CMP234 0.5454545  
## CMP235 0.2500000  
## CMP236 0.2500000  
## CMP237 0.2000000  
## CMP238 0.2000000  
## CMP239 0.2000000  
## CMP240 0.2000000  
## CMP241 0.2727273  
## CMP242 0.1428571  
## CMP243 0.2222222  
## CMP244 0.2727273  
## CMP245 0.5454545  
## CMP246 0.2000000  
## CMP247 0.2000000  
## CMP248 0.5454545  
## CMP249 0.5000000  
## CMP250 0.3333333  
## CMP251 0.5000000  
## CMP252 0.6000000  
## CMP253 0.5000000  
## CMP254 0.4285714  
## CMP255 0.6000000  
## CMP256 0.7777778  
## CMP257 0.7500000  
## CMP258 0.3750000  
## CMP259 0.5000000  
## CMP260 0.6000000  
## CMP261 0.2500000  
## CMP262 0.2500000  
## CMP263 0.2500000  
## CMP264 0.2000000  
## CMP265 0.3333333  
## CMP266 0.1666667  
## CMP267 0.4000000  
## CMP268 0.2500000  
## CMP269 0.2000000  
## CMP270 0.6000000  
## CMP271 0.6000000  
## CMP272 0.2000000  
## CMP273 0.2000000  
## CMP274 0.2000000  
## CMP275 0.2500000  
## CMP276 0.3333333  
## CMP277 0.4000000  
## CMP278 0.4285714  
## CMP279 0.5000000  
## CMP280 0.2000000  
## CMP281 0.2500000  
## CMP282 0.2000000  
## CMP283 0.2500000  
## CMP284 0.1250000  
## CMP285 0.1428571  
## CMP286 0.4285714  
## CMP287 0.5000000  
## CMP288 0.2000000  
## CMP289 0.2500000  
## CMP290 0.3333333  
## CMP291 0.4000000  
## CMP292 0.2857143  
## CMP293 0.3333333  
## CMP294 0.6000000  
## CMP295 0.6666667  
## CMP296 0.3333333  
## CMP297 0.1666667  
## CMP298 0.2000000  
## CMP299 0.1666667  
## CMP300 0.1428571