Prioritizing the Domestic Substance List Using Bioactivity - Methods

Marc Beal

2021-05-03

## Setup

Analysis was performed using R version 3.5.3 with the following R libraries:

library(ChemmineOB)  
library(ChemmineR)  
library(data.table)  
library(ggplot2)  
library(httk)  
library(httr)  
library(knitr)  
library(openxlsx)  
library(rcdk)  
library(RColorBrewer)  
library(rgl)  
library(RMySQL)  
library(tcpl)  
library(VennDiagram)  
  
sessionInfo()

## R version 3.5.3 (2019-03-11)  
## Platform: x86\_64-w64-mingw32/x64 (64-bit)  
## Running under: Windows 10 x64 (build 18363)  
##   
## Matrix products: default  
##   
## locale:  
## [1] LC\_COLLATE=English\_Canada.1252 LC\_CTYPE=English\_Canada.1252   
## [3] LC\_MONETARY=English\_Canada.1252 LC\_NUMERIC=C   
## [5] LC\_TIME=English\_Canada.1252   
##   
## attached base packages:  
## [1] grid stats graphics grDevices utils datasets methods   
## [8] base   
##   
## other attached packages:  
## [1] VennDiagram\_1.6.20 futile.logger\_1.4.3 tcpl\_2.0.2   
## [4] RMySQL\_0.10.20 DBI\_1.1.0 rgl\_0.100.54   
## [7] RColorBrewer\_1.1-2 rcdk\_3.5.0 rcdklibs\_2.3   
## [10] rJava\_0.9-12 openxlsx\_4.1.4 knitr\_1.23   
## [13] httr\_1.4.2 httk\_1.10.0 ggplot2\_3.3.2   
## [16] data.table\_1.12.8 ChemmineR\_3.34.1 ChemmineOB\_1.20.0   
##   
## loaded via a namespace (and not attached):  
## [1] bit64\_0.9-7 jsonlite\_1.6.1   
## [3] splines\_3.5.3 gsubfn\_0.7   
## [5] shiny\_1.4.0.2 assertthat\_0.2.1   
## [7] expm\_0.999-4 blob\_1.2.1   
## [9] yaml\_2.2.1 numDeriv\_2016.8-1.1   
## [11] pillar\_1.4.6 RSQLite\_2.2.0   
## [13] lattice\_0.20-41 glue\_1.4.0   
## [15] fingerprint\_3.5.7 chron\_2.3-55   
## [17] digest\_0.6.25 manipulateWidget\_0.10.1  
## [19] promises\_1.1.0 colorspace\_1.4-1   
## [21] htmltools\_0.4.0 httpuv\_1.5.2   
## [23] Matrix\_1.2-18 survey\_4.0   
## [25] pkgconfig\_2.0.3 zlibbioc\_1.28.0   
## [27] purrr\_0.3.4 xtable\_1.8-4   
## [29] mvtnorm\_1.1-0 scales\_1.0.0   
## [31] webshot\_0.5.2 itertools\_0.1-3   
## [33] later\_1.0.0 tibble\_3.0.1   
## [35] sqldf\_0.4-11 ellipsis\_0.3.0   
## [37] DT\_0.16 withr\_2.3.0   
## [39] proto\_1.0.0 survival\_3.1-12   
## [41] magrittr\_1.5 crayon\_1.3.4   
## [43] mime\_0.9 memoise\_1.1.0   
## [45] evaluate\_0.14 msm\_1.6.8   
## [47] tools\_3.5.3 mitools\_2.4   
## [49] formatR\_1.7 lifecycle\_0.2.0   
## [51] stringr\_1.4.0 munsell\_0.5.0   
## [53] zip\_2.0.4 lambda.r\_1.2.4   
## [55] compiler\_3.5.3 rlang\_0.4.5   
## [57] RCurl\_1.98-1.2 iterators\_1.0.13   
## [59] rsvg\_1.3 rjson\_0.2.20   
## [61] htmlwidgets\_1.5.2 crosstalk\_1.1.0.1   
## [63] miniUI\_0.1.1.1 bitops\_1.0-6   
## [65] base64enc\_0.1-3 rmarkdown\_2.4   
## [67] gtable\_0.3.0 deSolve\_1.28   
## [69] R6\_2.4.1 gridExtra\_2.3   
## [71] dplyr\_0.8.5 bit\_1.1-15.2   
## [73] fastmap\_1.0.1 futile.options\_1.0.1   
## [75] stringi\_1.4.6 parallel\_3.5.3   
## [77] Rcpp\_1.0.4.6 vctrs\_0.2.4   
## [79] png\_0.1-7 tidyselect\_1.0.0   
## [81] xfun\_0.13

## Extract All ToxCast Invitrodb\_v3 Data

This step extracts all bioactivity data from the ToxCast databse (version3). The methods used in this first step borrow heavily from the work by **Paul-Friedman et al. 2019** ([Utility of *In Vitro* Bioactivity as a Lower Bound Estimate of *In Vivo* Adverse Effect Levels and in Risk-Based Prioritization](https://academic.oup.com/toxsci/advance-article/doi/10.1093/toxsci/kfz201/5571376)). For this step to work, MySQL needs to be installed and the invitrodb\_v3 database needs to be stored locally.

ToxCastAssayDesc <- read.table("ToxCast\_Assay\_Information.txt", header=TRUE, sep="\t", stringsAsFactors=FALSE)  
  
tcplConf(user="USER1", pass="PASSWORD", drvr = "MySQL", db="invitrodb\_v3", host="localhost")  
  
## Load all chemicals in database  
chem <- tcplLoadChem(field="chem.only")  
chem <- tcplLoadChem(field="casn", val=chem$casn)  
  
## Load chemical level 5 data (activities and hit calls)   
dat5 <- tcplPrepOtpt(tcplLoadData(lvl = 5, fld = 'spid', val = chem$spid, type = 'mc'))  
datToxCast = dat5  
  
## Subsets level 5 data to a single tested sample per chemical and filter to positive hits  
dat5\_all <- tcplSubsetChid(dat5, flag=TRUE)  
dat5 <- dat5\_all[hitc==1]  
  
## Load level 7 uncertainty information needed for hitpercent  
m4.id <- dat5[,m4id]  
dat7 <- tcplPrepOtpt(tcplLoadData(lvl = 7, fld = 'm4id', val = m4.id))  
  
## Load Level 6 Data (caution flags) and add to Level 5 Table  
dat6 <- tcplPrepOtpt(tcplLoadData(lvl = 6, fld = 'spid', val = chem$spid, type='mc'))  
setDT(dat6)  
mc6\_mthds <- dat6[ , .( mc6\_mthd\_id = paste(mc6\_mthd\_id, collapse = ",")),  
 by = m4id]  
mc6\_flags <- dat6[ , .( flag = paste(flag, collapse = ";")),   
 by = m4id]  
dat5$mc6\_flag\_ID <- mc6\_mthds$mc6\_mthd\_id[match(dat5$m4id, mc6\_mthds$m4id)]  
dat5$flags <- mc6\_flags$flag[match(dat5$m4id, mc6\_flags$m4id)]  
  
## Add 7 Data to Level 5 Table  
dat5$hitpct <- dat7$hit\_pct[match(dat5$m4id, dat7$m4id)]  
  
##Extract cyotoxicity data  
cytotox <- tcplCytoPt(chid=chem$chid)  
  
##ToxCast Assay Filtering  
originaldat5 = dat5  
originaldat5[,hitcsum := sum(hitc), by=list(casn)] # Will be used to compare active assays later  
dat5 = dat5[(nflg < 3 | is.na(nflg) == TRUE) | hitpct > 0.5]  
dat5 = dat5[fitc != 36 & fitc != 45]  
  
##Add assay info  
dat5 = merge(dat5[,c("casn", "chid", "chnm", "aeid", "aenm", "hitc", "coff", "modl\_tp", "modl\_ga", "flags", "hitpct")],ToxCastAssayDesc[,c("aeid", "assay\_component\_endpoint\_name", "organism", "tissue", "cell\_format", "intended\_target\_family", "intended\_target\_family\_sub")], by.x="aenm", by.y="assay\_component\_endpoint\_name", all.x=TRUE, all.y=TRUE)  
names(dat5)[names(dat5) == "aeid.x"] = "aeid"  
dat5 = dat5[,-c("aeid.y")]  
dat5 = dat5[order(chnm)]  
dat5 = dat5[!is.na(dat5$chnm),]  
  
##Set AC50 to 100uM for chemicals with no suitable curves after filtering  
dat5[,hitcsum := sum(hitc), by=list(casn)]  
dat5[, modl\_ga\_uM := ifelse(!is.na(modl\_ga), 10^modl\_ga, NA)]  
dat5[, min\_modl\_ga\_uM := min(modl\_ga\_uM, na.rm=TRUE), by=list(chid)]  
  
negatives = originaldat5$chnm[!originaldat5$chnm %in% dat5$chnm]  
  
if (length(negatives) > 0) {  
  
 negatives = unique(negatives)  
  
 print(paste0("Warning! The following chemicals had no curves after filtering and AC50 set to 100 uM: ", negatives))  
  
 negativesMatrix = matrix(ncol = 19, nrow = length(negatives))  
  
 for (i in 1:length(negatives)) {  
 negativesMatrix[i,1] = "NA"  
 negativesMatrix[i,2] = as.character(originaldat5[originaldat5$chnm %in% negatives[i],5][1])  
 negativesMatrix[i,3] = as.character(originaldat5[originaldat5$chnm %in% negatives[i],4][1])  
 negativesMatrix[i,4] = negatives[i]  
 negativesMatrix[i,5] = "NA"  
 negativesMatrix[i,6] = "NA"  
 negativesMatrix[i,7] = "NA"  
 negativesMatrix[i,8] = "NA"  
 negativesMatrix[i,9] = 2  
 negativesMatrix[i,10] = "NA"  
 negativesMatrix[i,11] = "NA"  
 negativesMatrix[i,12] = "NA"  
 negativesMatrix[i,13] = "NA"  
 negativesMatrix[i,14] = "NA"  
 negativesMatrix[i,15] = "NA"  
 negativesMatrix[i,16] = "NA"  
 negativesMatrix[i,17] = 0  
 negativesMatrix[i,18] = 100  
 negativesMatrix[i,19] = 100  
 }  
  
 rownames(negativesMatrix) = negatives  
 colnames(negativesMatrix) = colnames(dat5)  
  
 dat5 = rbind(dat5, negativesMatrix)  
  
}  
  
dat5[, min\_modl\_ga\_uM := min(as.numeric(modl\_ga\_uM), na.rm=TRUE), by=list(chid)]  
dat5[, modl\_ga\_max\_uM := max(as.numeric(modl\_ga\_uM), na.rm=TRUE), by=list(chid)]   
dat5[, modl\_ga\_0.05\_uM := quantile(as.numeric(modl\_ga\_uM), probs=c(0.05)), by=list(chid)]  
dat5[, modl\_ga\_0.5\_uM := quantile(as.numeric(modl\_ga\_uM), probs=c(0.5)), by=list(chid)]  
dat5[, modl\_ga\_0.95\_uM := quantile(as.numeric(modl\_ga\_uM), probs=c(0.95)), by=list(chid)]  
  
##Calculate hit rate  
  
hitRateTable = matrix(nrow=length(unique(datToxCast$chnm)), ncol=6)  
rownames(hitRateTable) = unique(datToxCast$chnm)  
colnames(hitRateTable) = c("casn", "Total Unique Assays Screened", "Active Assay Count Pre-Filtering", "Active Assays After Filtering", "Percent of Active Assays Filtered", "Chemical Name")  
  
for (i in 1:length(unique(datToxCast$chnm))) {  
  
hitRateTable[i,1] = datToxCast[datToxCast$chnm %in% unique(datToxCast$chnm)[i], casn][1]  
hitRateTable[i,2] = length(unique(datToxCast[datToxCast$chnm %in% unique(datToxCast$chnm)[i], aeid]))  
hitRateTable[i,3] = length(unique(originaldat5[originaldat5$chnm %in% unique(datToxCast$chnm)[i], aeid]))  
ifelse(unique(dat5[dat5$chnm %in% unique(datToxCast$chnm)[i], aeid]) != "NA",(hitRateTable[i,4] = length(unique(dat5[dat5$chnm %in% unique(datToxCast$chnm)[i], aeid]))),(hitRateTable[i,4] = 0))  
hitRateTable[i,5] = (1-(as.numeric(hitRateTable[i,4])/as.numeric(hitRateTable[i,3])))\*100  
hitRateTable[i,6] = as.character(unique(datToxCast$chnm)[unique(datToxCast$casn) %in% hitRateTable[i,1]])  
}  
  
hitRateTable <- as.data.table(hitRateTable)

**ToxCast Data Preview:**

kable(head(dat5[,c("casn", "chid", "aeid", "modl\_ga\_uM", "modl\_ga\_0.05\_uM")]))

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| casn | chid | aeid | modl\_ga\_uM | modl\_ga\_0.05\_uM |
| 50505-91-4 | 48888 | 766 | 49.84771 | 0.0001542 |
| 50505-91-4 | 48888 | 804 | 71.12628 | 0.0001542 |
| 50505-91-4 | 48888 | 805 | 35.73573 | 0.0001542 |
| 50505-91-4 | 48888 | 807 | 68.20978 | 0.0001542 |
| 50505-91-4 | 48888 | 1134 | 69.36872 | 0.0001542 |
| 50505-91-4 | 48888 | 2089 | 112.82176 | 0.0001542 |

**Hit Rate Table Preview:**

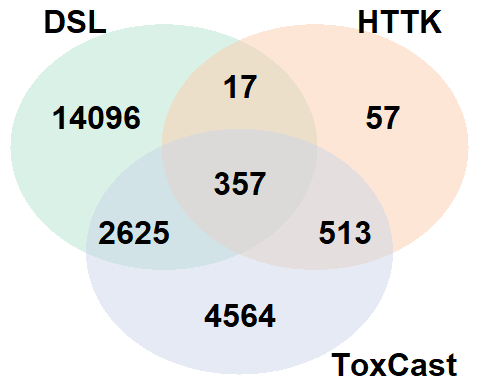
kable(head(hitRateTable[,1:5]), digits=2)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| casn | Total.Unique.Assays.Screened | Active.Assay.Count.Pre.Filtering | Active.Assays.After.Filtering | Percent.of.Active.Assays.Filtered |
| 344423-98-9 | 1 | 1 | 1 | 0.00 |
| 73573-88-3 | 80 | 9 | 9 | 0.00 |
| 75330-75-5 | 720 | 172 | 172 | 0.00 |
| NOCAS\_110050 | 1 | 0 | NA | NA |
| 79902-63-9 | 698 | 228 | 226 | 0.88 |
| NOCAS\_110051 | 1 | 1 | 1 | 0.00 |

## Idenitfy Data Gaps Between DSL, ToxCast, and HTTK

Here the CASRN intersection is established between ToxCast, DSL, and HTTK.

dsl <- readLines("DSL\_casrn.txt")  
  
httkDat <- get\_cheminfo(info="CAS")  
  
toxcast <- dat5$casn  
  
ColScheme <- brewer.pal(3, "Pastel2")  
  
VennPlot <- venn.diagram(  
 x = list(dsl, httkDat, toxcast),  
 category.names = c("DSL" , "HTTK " , "ToxCast"),  
 filename = NULL,  
 output=TRUE,  
   
 # Output features  
 imagetype="png" ,  
 height = 480 ,   
 width = 480 ,   
 resolution = 300,  
 compression = "lzw",  
   
 # Circles  
 lwd = 2,  
 lty = 'blank',  
 fill = ColScheme,  
   
 # Numbers  
 cex = 2,  
 fontface = "bold",  
 fontfamily = "sans",  
   
 # Set names  
 cat.cex = 2,  
 cat.fontface = "bold",  
 cat.default.pos = "outer",  
 cat.pos = c(-27, 27, 135),  
 cat.dist = c(0.055, 0.055, 0.085),  
 cat.fontfamily = "sans",  
 rotation = 1  
)  
  
grid.draw(VennPlot)



The results above show the two main data gaps that need to be addressed to prioritize DSL compounds using this workflow:

1. HTTK Data
2. ToxCast Data

The workflow will address these data gaps and all accompanying information will be supplied to a summary table.

summaryTable <- matrix(nrow=length(dsl), ncol=29)  
colnames(summaryTable) <- c("CASRN", "Compound", "SMILES", "Log10BER95th", "Log10BERMedian", "AED(mg/kg-bw/day)", "ExposureAboveTTC", "TTC(mg/kg-bw/day)", "CramerClass", "ExpoCast95thPercentile", "ExpoCastMedian", "BioactivityConcentration(MicroMolar)", "BioactivitySource", "Css(MicroMolar)", "CssSource", "ADMETOutsideApplicabilityDomain", "LogP\_OpenBabel", "LogP\_RCDK", "LogPViolation", "MolecularWeight", "MolecularWeightViolation", "HBondDonorsCount", "HBondDonorsCountViolation", "HBondAcceptorsCount", "HBondAcceptorsCountViolation", "FractionAbsorbed", "FractionAbsorbedViolation", "FractionBioavailable", "FractionBioavailableViolation")  
rownames(summaryTable) <- dsl  
summaryTable[,"CASRN"] <- dsl  
  
summaryTable[summaryTable[,"CASRN"] %in% get\_cheminfo("CAS"),"CssSource"] <- "in\_vitro"

## First Data Gap - DSL Compounds Lacking HTTK Data

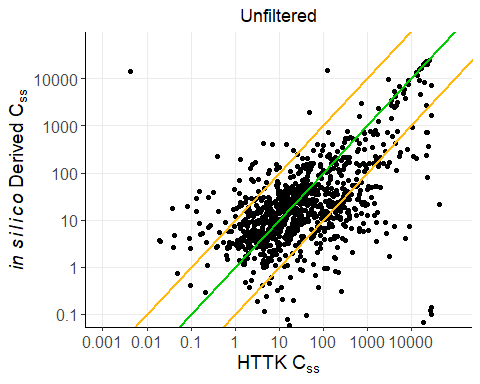
The first data gap to be addressed is lack of HTTK data for DSL compounds. *in silico* models exist and can be used to address this issue. Specifically, ADMET Predictor 10 was used to obtain Human intrinsic clearance (Clint) and fraction unbound in plasma protein (Fup) values. Before applying model to DSL compounds, the performance of these models, in the context of this workflow, are first assessed in the code blocks below.

### High-Throughput Toxicokinetics - Evaluate Performance of *in silico* Predictions

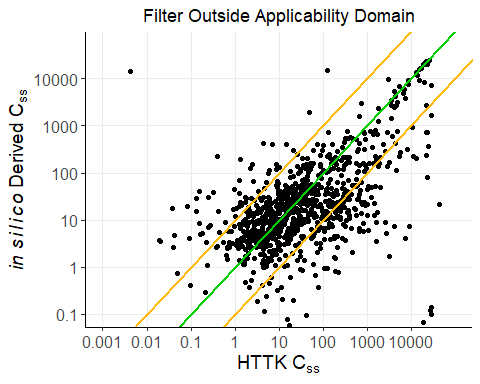
Here we derive Css values using data from HTTK v1.10 and compare it to Css values obtained using *in silico* predictions of Human Clint and Fup.

##Calculate Css based on existing HTTK Data  
CAS <- get\_cheminfo("CAS")  
Css <- rep("NA", length(CAS))  
for (i in 1:length(CAS)) { Css[i] <- calc\_mc\_css(chem.cas=CAS[i],  
 which.quantile=c(0.95), restrictive.clearance=T,   
 model="3compartmentss", output.units="uM", species="Human",   
 method="dr", well.stirred.correction=T, suppress.messages=F,  
 return.samples=F)  
}  
httkDataTable <- cbind(CAS, Css)  
  
##Calculate Css based on existing in silico predictions  
inHTTK <- read.xlsx("ADMET\_HTTK\_Predictions.xlsx", colNames=TRUE)  
chem.physical\_and\_invitro.data <- add\_chemtable(inHTTK,current.table=chem.physical\_and\_invitro.data, data.list=list(CAS="CASRN", Funbound.plasma="Human.Funbound.plasma"),species="Human", reference="ADMET10", overwrite=T)  
chem.physical\_and\_invitro.data <- add\_chemtable(inHTTK,current.table=chem.physical\_and\_invitro.data, data.list=list(CAS="CASRN", Clint="Human.Clint"), species="Human", reference="ADMET10",overwrite=T)  
  
CAS <- inHTTK$CASRN  
Css <- rep("NA", length(CAS))  
for (i in 1:length(CAS)) { Css[i] <- calc\_mc\_css(chem.cas=CAS[i],  
 which.quantile=c(0.95), restrictive.clearance=T,   
 model="3compartmentss", output.units="uM", species="Human",   
 method="dr", well.stirred.correction=T, suppress.messages=F,  
 return.samples=F)  
}  
admetDataTable <- cbind(CAS, Css)  
colnames(admetDataTable) <- c("CAS", "in\_silico\_derived\_Css")  
  
##Restore chem.physical\_and\_invitro.data to default  
rm(chem.physical\_and\_invitro.data)

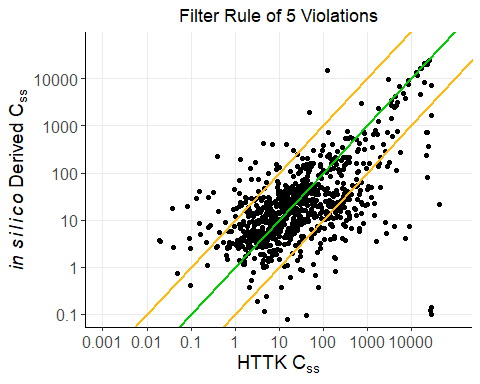
##Merge tables together and compare results  
dataComparison <- merge(httkDataTable, admetDataTable, by.x="CAS", by.y="CAS")  
#Merge returns factors so it is important to convert back to numeric, but first as a string  
dataComparison$Css <- as.numeric(as.character(dataComparison$Css))  
dataComparison$in\_silico\_derived\_Css <- as.numeric(as.character(dataComparison$in\_silico\_derived\_Css))  
  
##Sort dataComparison by absolute log10 difference  
dataComparison$logDifference <- log10(dataComparison$in\_silico\_derived\_Css)-log10(dataComparison$Css)  
dataComparison <- dataComparison[order(abs(dataComparison$logDifference)),]  
  
##Only include chemicals that had ADMET Predictions  
dataComparison <- dataComparison[dataComparison$CAS %in% inHTTK$CASRN,]  
  
#Define plot function to be used multiple times  
comparisonPlotFunction <- function(xData, yData, xLab, yLab, titleLab, straightLine = c(0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000, 100000, 1000000)) {  
  
ggplot() +   
 geom\_point(aes(x=xData, y=yData)) +   
 scale\_x\_log10(breaks=c(0.001,0.01,0.1,1,10,100,1000,10000), labels=c(0.001,0.01,0.1,1,10,100,1000,10000)) +   
 scale\_y\_log10(breaks=c(0.1,1,10,100,1000,10000), labels=c(0.1,1,10,100,1000,10000)) +  
 coord\_cartesian(ylim=c(0.1,50000), xlim=c(0.001,100000)) +  
 ggtitle(titleLab) +  
 ylab(yLab) +   
 xlab(xLab) +  
 geom\_line(aes(x=straightLine, y=straightLine), colour="green3", size=1) +  
 geom\_line(aes(x=straightLine, y=straightLine\*10), colour="darkgoldenrod1", size=1) +  
 geom\_line(aes(x=straightLine, y=straightLine/10), colour="darkgoldenrod1", size=1) +  
 theme\_bw() +  
 theme(panel.border = element\_blank(),   
 panel.grid.major.y = element\_line(),  
 panel.grid.minor.y = element\_blank(),  
 panel.grid.minor.x = element\_blank(),  
 axis.line = element\_line(colour = "black"),  
 axis.text = element\_text(size=12),  
 axis.title = element\_text(size=14, face="bold"),  
 plot.title = element\_text(hjust = 0.5))   
}  
  
comparisonPlotFunction(dataComparison$Css, dataComparison$in\_silico\_derived\_Css, expression(HTTK~C[ss]), expression(italic("in"~silico)~Derived~C[ss]), "Unfiltered")



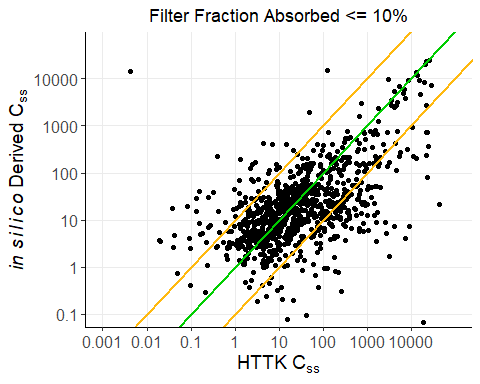
##Remove compounds outside the Applicability Domain of ADMET  
dataComparisonInDomain <- dataComparison[dataComparison$CAS %in% inHTTK[inHTTK$Outside\_Applicability\_Domain == FALSE,"CASRN"],]  
  
comparisonPlotFunction(dataComparisonInDomain$Css, dataComparisonInDomain$in\_silico\_derived\_Css, expression(HTTK~C[ss]), expression(italic("in"~silico)~Derived~C[ss]), "Filter Outside Applicability Domain")



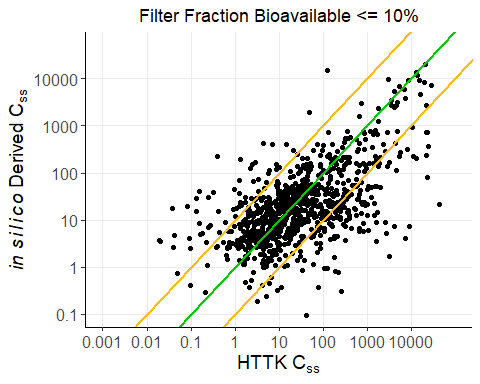
##Remove compounds violating the Rule of 5  
#ruleOf5 function:  
ruleOf5Violations <- function(x) {  
 as.numeric(eval.desc(parse.smiles(x)[[1]], "org.openscience.cdk.qsar.descriptors.molecular.RuleOfFiveDescriptor"))  
}  
  
#HBondDonorCountDescriptor function:  
HBondDonorCountFunction <- function(x) {  
 as.numeric(eval.desc(parse.smiles(x)[[1]], "org.openscience.cdk.qsar.descriptors.molecular.HBondDonorCountDescriptor"))  
}  
  
#HBondAcceptorCountDescriptor function:  
HBondAcceptorCountFunction <- function(x) {  
 as.numeric(eval.desc(parse.smiles(x)[[1]], "org.openscience.cdk.qsar.descriptors.molecular.HBondAcceptorCountDescriptor"))  
}  
  
#XLogPDescriptor function:  
xLogPFunction <- function(x) {  
 as.numeric(eval.desc(parse.smiles(x)[[1]], "org.openscience.cdk.qsar.descriptors.molecular.XLogPDescriptor"))  
}  
  
LipinskiFailuresHTTK <- as.vector(sapply(inHTTK$SMILES, ruleOf5Violations))  
  
inHTTK <- cbind(inHTTK, LipinskiFailuresHTTK)  
  
dataComparisonNoViolations <- dataComparison[dataComparison$CAS %in% inHTTK$CASRN[inHTTK$LipinskiFailuresHTTK == 0],]  
  
comparisonPlotFunction(dataComparisonNoViolations$Css, dataComparisonNoViolations$in\_silico\_derived\_Css, expression(HTTK~C[ss]), expression(italic("in"~silico)~Derived~C[ss]), "Filter Rule of 5 Violations")



##Remove compounds with predicted Fraction Absorbed below 10%  
dataComparisonFa <- dataComparison[dataComparison$CAS %in% inHTTK$CASRN[inHTTK$`%Fraction\_Absorbed` > 10],]  
  
comparisonPlotFunction(dataComparisonFa$Css, dataComparisonFa$in\_silico\_derived\_Css, expression(HTTK~C[ss]), expression(italic("in"~silico)~Derived~C[ss]), "Filter Fraction Absorbed <= 10%")



##Remove compounds with predicted Fraction Bioavailable below 10%  
dataComparisonFb <- dataComparison[dataComparison$CAS %in% inHTTK$CASRN[inHTTK$`%Fraction\_Bioavailable` > 10],]  
  
comparisonPlotFunction(dataComparisonFb$Css, dataComparisonFb$in\_silico\_derived\_Css, expression(HTTK~C[ss]), expression(italic("in"~silico)~Derived~C[ss]), "Filter Fraction Bioavailable <= 10%")



#Evaluate Rule of 5 violations for compound where in silico Css - in vitro Css is highest  
highCssSMILES <- inHTTK[inHTTK$CASRN == dataComparison[rev(order(dataComparison$logDifference))[1],]$CAS,"SMILES"]  
cat(paste0("The chemical with the highest in silico Css to in vitro Css ratio was ", inHTTK[inHTTK$CASRN == dataComparison[rev(order(dataComparison$logDifference))[1],]$CAS,"Compound"]))

## The chemical with the highest in silico Css to in vitro Css ratio was Fulvestrant

cat(paste0("\nlogP: ",   
 as.numeric(eval.desc(parse.smiles(highCssSMILES)[[1]], "org.openscience.cdk.qsar.descriptors.molecular.XLogPDescriptor"))))

##   
## logP: 9.44

cat(paste0("\nMolecular Weight: ",   
 as.numeric(eval.desc(parse.smiles(highCssSMILES)[[1]], "org.openscience.cdk.qsar.descriptors.molecular.WeightDescriptor"))))

##   
## Molecular Weight: 606.773080002774

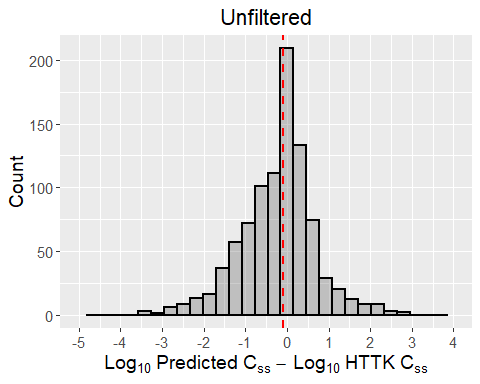
cat(paste0("\nH Bond Donors: ",   
 as.numeric(eval.desc(parse.smiles(highCssSMILES)[[1]], "org.openscience.cdk.qsar.descriptors.molecular.HBondDonorCountDescriptor"))))

##   
## H Bond Donors: 2

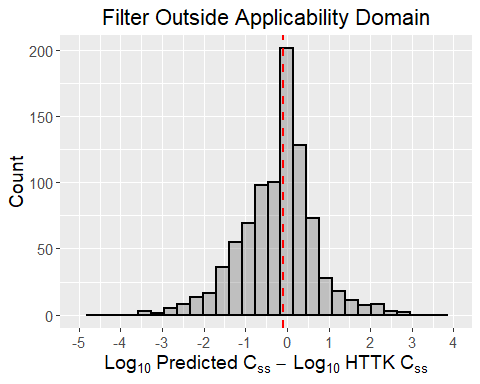
cat(paste0("\nH Bond Acceptors: ",   
 as.numeric(eval.desc(parse.smiles(highCssSMILES)[[1]], "org.openscience.cdk.qsar.descriptors.molecular.HBondAcceptorCountDescriptor"))))

##   
## H Bond Acceptors: 3

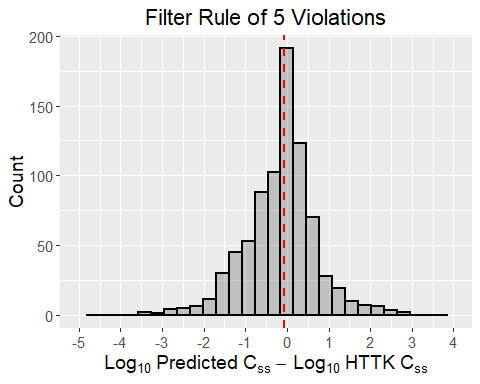
#Define second plot function to be used multiple times  
comparisonPlotFunction2 <- function(inLogDifference, titleLab) {  
  
 suppressWarnings(  
 print(  
 ggplot(inLogDifference, aes(x=as.numeric(value))) +   
 geom\_histogram(aes(y=..count..), alpha=0.3, color="black", size=1) +  
 geom\_vline(aes(xintercept=median(inLogDifference$value)), color="red", linetype="dashed", size=1) +  
 ggtitle(titleLab) +  
 ylab("Count") +  
 scale\_x\_continuous(name=expression(Log["10"]~Predicted~C[ss]~-~Log["10"]~HTTK~C[ss]), limits=c(-5, 4), breaks=-5:4) +  
 theme(text = element\_text(size=14),  
 plot.title = element\_text(hjust = 0.5))  
 )  
 )   
   
}  
  
#Unfiltered  
logDifference <- data.frame(label=1:nrow(dataComparison), value=as.numeric(log10(dataComparison$in\_silico\_derived\_Css)-log10(dataComparison$Css)))  
  
comparisonPlotFunction2(logDifference, "Unfiltered")



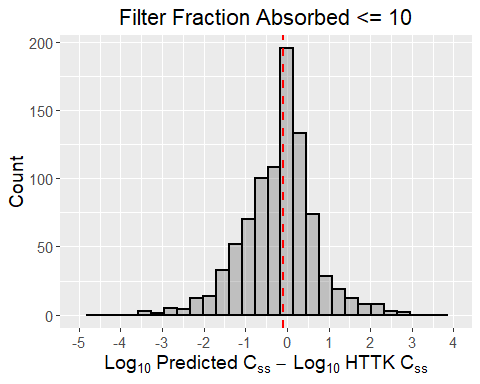
#Filter compounds outside Applicability Domain  
logDifferenceInDomain <- data.frame(label=1:nrow(dataComparisonInDomain), value=as.numeric(log10(dataComparisonInDomain$in\_silico\_derived\_Css)-log10(dataComparisonInDomain$Css)))  
  
comparisonPlotFunction2(logDifferenceInDomain, "Filter Outside Applicability Domain")



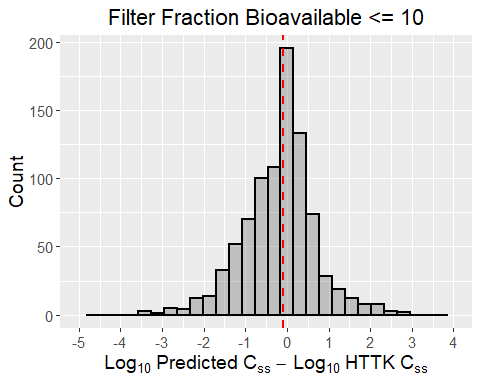
#Filter compounds violating Rule of 5  
logDifferenceNoViolations <- data.frame(label=1:nrow(dataComparisonNoViolations), value=as.numeric(log10(dataComparisonNoViolations$in\_silico\_derived\_Css)-log10(dataComparisonNoViolations$Css)))  
  
comparisonPlotFunction2(logDifferenceNoViolations, "Filter Rule of 5 Violations")



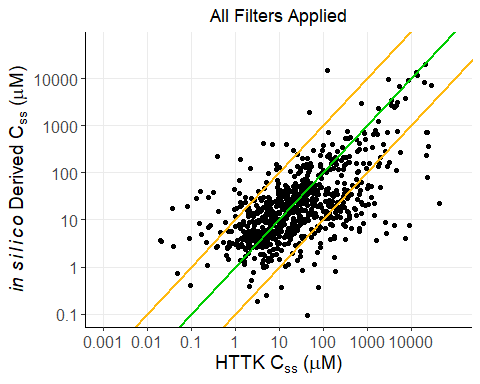
#Filter compounds with low absoprtion  
logDifferenceAbsorbed <- data.frame(label=1:nrow(dataComparisonFa), value=as.numeric(log10(dataComparisonFa$in\_silico\_derived\_Css)-log10(dataComparisonFa$Css)))  
  
comparisonPlotFunction2(logDifferenceAbsorbed, "Filter Fraction Absorbed <= 10")



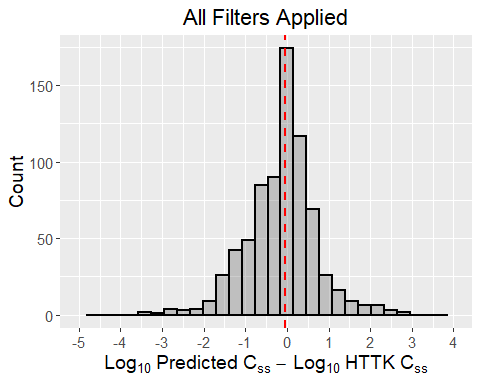
#Filter compounds with low bioavailability  
logDifferenceBioavailable <- data.frame(label=1:nrow(dataComparisonFb), value=as.numeric(log10(dataComparisonFb$in\_silico\_derived\_Css)-log10(dataComparisonFb$Css)))  
  
comparisonPlotFunction2(logDifferenceAbsorbed, "Filter Fraction Bioavailable <= 10")



filteredDataComparison <- dataComparison  
filteredDataComparison <- filteredDataComparison[filteredDataComparison$CAS %in% dataComparisonInDomain$CAS,]  
filteredDataComparison <- filteredDataComparison[filteredDataComparison$CAS %in% dataComparisonNoViolations$CAS,]  
filteredDataComparison <- filteredDataComparison[filteredDataComparison$CAS %in% dataComparisonFa$CAS,]  
filteredDataComparison <- filteredDataComparison[filteredDataComparison$CAS %in% dataComparisonFb$CAS,]  
  
comparisonPlotFunction(filteredDataComparison$Css, filteredDataComparison$in\_silico\_derived\_Css, expression(HTTK~C[ss]~"("\*mu\*"M)"), expression(italic("in"~silico)~Derived~C[ss]~"("\*mu\*"M)"), "All Filters Applied")



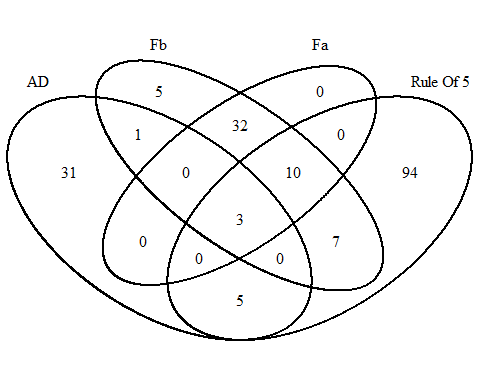
logDifferenceFiltered <- data.frame(label=1:nrow(filteredDataComparison), value=as.numeric(log10(filteredDataComparison$in\_silico\_derived\_Css)-log10(filteredDataComparison$Css)))  
  
comparisonPlotFunction2(logDifferenceFiltered, "All Filters Applied")



httkFilterTable <- matrix(nrow=6, ncol=3)  
rownames(httkFilterTable) <- c("Unfiltered", "AD", "RuleOf5", "Fa", "Fb", "AllFilters")   
colnames(httkFilterTable) <- c("Chemicals Removed", "%Within 10-fold", "%Within 10-fold")  
  
httkFilterTable["Unfiltered",] <- c(0,   
 round(as.numeric(table(abs(logDifference$value)>1)["FALSE"])/length(logDifference$value)\*100,2),   
 round(as.numeric(table(abs(logDifference$value)>2)["FALSE"])/length(logDifference$value)\*100,2))  
httkFilterTable["AD",] <- c(nrow(logDifference)-nrow(logDifferenceInDomain),   
 round(as.numeric(table(abs(logDifferenceInDomain$value)>1)["FALSE"])/length(logDifferenceInDomain$value)\*100,2), round(as.numeric(table(abs(logDifferenceInDomain$value)>2)["FALSE"])/length(logDifferenceInDomain$value)\*100,2))  
httkFilterTable["RuleOf5",] <- c(nrow(logDifference)-nrow(logDifferenceNoViolations),   
 round(as.numeric(table(abs(logDifferenceNoViolations$value)>1)["FALSE"])/length(logDifferenceNoViolations$value)\*100,2),   
 round(as.numeric(table(abs(logDifferenceNoViolations$value)>2)["FALSE"])/length(logDifferenceNoViolations$value)\*100,2))  
httkFilterTable["Fa",] <- c(nrow(logDifference)-nrow(logDifferenceAbsorbed),   
 round(as.numeric(table(abs(logDifferenceAbsorbed$value)>1)["FALSE"])/length(logDifferenceAbsorbed$value)\*100,2), round(as.numeric(table(abs(logDifferenceAbsorbed$value)>2)["FALSE"])/length(logDifferenceAbsorbed$value)\*100,2))  
httkFilterTable["Fb",] <- c(nrow(logDifference)-nrow(logDifferenceBioavailable),   
 round(as.numeric(table(abs(logDifferenceBioavailable$value)>1)["FALSE"])/length(logDifferenceBioavailable$value)\*100,2),   
 round(as.numeric(table(abs(logDifferenceBioavailable$value)>2)["FALSE"])/length(logDifferenceBioavailable$value)\*100,2))  
httkFilterTable["AllFilters",] <- c(nrow(logDifference)-nrow(logDifferenceFiltered),  
 round(as.numeric(table(abs(logDifferenceFiltered$value)>1)["FALSE"])/length(logDifferenceFiltered$value)\*100,2),  
 round(as.numeric(table(abs(logDifferenceFiltered$value)>2)["FALSE"])/length(logDifferenceFiltered$value)\*100,2))  
  
kable(httkFilterTable)

|  |  |  |  |
| --- | --- | --- | --- |
|  | Chemicals Removed | %Within 10-fold | %Within 10-fold |
| Unfiltered | 0 | 75.94 | 94.31 |
| AD | 40 | 75.76 | 94.16 |
| RuleOf5 | 119 | 78.45 | 95.57 |
| Fa | 45 | 77.43 | 95.37 |
| Fb | 58 | 77.89 | 95.88 |
| AllFilters | 188 | 79.68 | 96.64 |

#Show overlap CASRN removed by each filter  
httkVennPlot <- venn.diagram(  
 x = list(dataComparison$CAS[!dataComparison$CAS %in% dataComparisonInDomain$CAS],  
 dataComparison$CAS[!dataComparison$CAS %in% dataComparisonNoViolations$CAS],  
 dataComparison$CAS[!dataComparison$CAS %in% dataComparisonFb$CAS],  
 dataComparison$CAS[!dataComparison$CAS %in% dataComparisonFa$CAS]),  
 category.names = c("AD" , "Rule Of 5", "Fb", "Fa"),  
 filename = NULL,  
 output=TRUE)  
  
grid.draw(httkVennPlot)



#Get the partition information for reporting - order above was changed for graphing reasons  
httkVennPartitions <- get.venn.partitions(list(  
 dataComparison$CAS[!dataComparison$CAS %in% dataComparisonInDomain$CAS],   
 dataComparison$CAS[!dataComparison$CAS %in% dataComparisonNoViolations$CAS],  
 dataComparison$CAS[!dataComparison$CAS %in% dataComparisonFa$CAS],  
 dataComparison$CAS[!dataComparison$CAS %in% dataComparisonFb$CAS]))

The results above show that 75.94% of Css values derived from *in silico* predictions are within 10-fold of the Css derived using HTTK data, and 94.31% are within 100-fold, irrespective of the filters.

The venn diagram shows that there is only 3 chemical in common that all the fiters removed. The filter based on fraction absorbed is redundant, as the fraction bioavailable filter removes all the same chemicals. There are 31 chemicals unique to the applicability domain filter, 94 unique to rule of 5 filter, 0 unique to fraction absorbed filter, and 5 chemicals unique to the fraction bioavailable filter.

Removing 188 chemicals based on the three above filters show that 79.68% of Css values derived from *in silico* predictions are within 10-fold of the Css derived using HTTK data, and 96.64% are within 100-fold.

Collectively, these results provide confidence in the application of the ADMET predictor model.

### Examine the Quality of Predictions for Different ToxPrint Chemotypes

The above results show that the *in silico* derived Css values are concordant with the HTTK Css for the majority of compounds. Here we are examining the ToxPrint ChemoTypes of the compounds with poorer predictions to see if there are any functional groups that should be excluded from the analysis. First, the SMILES for HTTK compounds are obtained and converted to ToxPrint Chemotypes. This step was performed outside of the R workflow using Chemotyper v1.0.r12976 and ToxPrintv2.0\_r711.xml. Below, a Structure-Data File (SDF) for HTTK compounds is created to be used with the Chemotyper software. The ToxPrint Chemotypes were identified for structures that could be resolved and the results matrices were exported as TSV files. The results matrices are available in xlsx format.

smilesTable <- chem.physical\_and\_invitro.data[chem.physical\_and\_invitro.data$CAS %in% get\_cheminfo("CAS"), c("CAS", "SMILES.desalt")]  
smilesTable <- smilesTable[!smilesTable$SMILES.desalt %in% "-",]  
sdfsetHTTK <- smiles2sdf(smilesTable$SMILES.desalt)  
cid(sdfsetHTTK) <- smilesTable$CAS  
for (i in 1:length(sdfsetHTTK)) { attributes(sdfsetHTTK[[i]])$header["Molecule\_Name"] <- smilesTable$CAS[i] }  
write.SDF(sdfsetHTTK, file="HTTK.sdf")

In the next code block, the ToxPrint ChemoTypes are loaded into the R environment and checked to see if there are any enriched ChemoTypes associated discrepancies between *in silico* and *in vitro* derived Css values. In this example, a discrepancy was represented by an *in silico* derived Css >100-fold larger or smaller than the HTTK Css value. On the log-scale, this is a threshold of 2 (log10(*in silico* Css)-log10(HTTK Css)). To evaluate other thresholds, the first variable in the code below, labeled as **predictionThreshold**, can be adjusted.

##Change the value of predictionThreshold as required  
predictionThreshold <- 2  
  
httkFP <- read.xlsx("HTTK\_FingerPrints.xlsx", colNames=TRUE)  
rownames(httkFP) <- httkFP[,1]  
httkFP <- httkFP[,-1]  
  
goodPredCAS <- dataComparison$CAS[!abs(log10(dataComparison$in\_silico\_derived\_Css)-log10(dataComparison$Css))>predictionThreshold]   
badPredCAS <- dataComparison$CAS[abs(log10(dataComparison$in\_silico\_derived\_Css)-log10(dataComparison$Css))>predictionThreshold]  
  
goodPredTable <- httkFP[rownames(httkFP) %in% goodPredCAS,]  
goodPresent <- colSums(goodPredTable)  
  
badPredTable <- httkFP[rownames(httkFP) %in% badPredCAS,]  
badPresent <- colSums(badPredTable)  
  
FishersTable <- rbind(goodPresent, badPresent)  
colnames(FishersTable) <- colnames(httkFP)  
FishersTable <- FishersTable[,colSums(FishersTable)!=0]  
  
goodAbsent <- nrow(goodPredTable)-FishersTable["goodPresent",]  
badAbsent <- nrow(badPredTable)-FishersTable["badPresent",]  
  
FishersTable <- rbind(FishersTable, goodAbsent, badAbsent)  
FishersTable <- FishersTable[c(1,3,2,4),]  
  
pValue <- vector()  
lowerConfInt <- vector()  
upperConfInt <- vector()  
oddsRatio <- vector()  
  
for (i in 1:ncol(FishersTable)) {  
  
 tempResult <- fisher.test(matrix(FishersTable[,i],2,2))  
  
 pValue <- c(pValue, tempResult$p.value)  
 lowerConfInt <- c(lowerConfInt, tempResult$conf.int[1])  
 upperConfInt <- c(upperConfInt, tempResult$conf.int[2])  
 oddsRatio <- c(oddsRatio, tempResult$estimate)  
  
}  
  
adjustedPValue <- p.adjust(pValue, method="holm")  
  
FishersTable <- rbind(FishersTable, pValue, adjustedPValue, oddsRatio, lowerConfInt, upperConfInt)  
  
FishersTable <- t(FishersTable)  
  
enrichedChemoTypes <- FishersTable[FishersTable[,"pValue"]<=0.05,]  
enrichedChemoTypes <- enrichedChemoTypes[enrichedChemoTypes[,"oddsRatio"]<1,]  
enrichedChemoTypes <- enrichedChemoTypes[order(enrichedChemoTypes[,"pValue"]),]  
  
kable(enrichedChemoTypes[,1:7], digits=3)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | goodPresent | goodAbsent | badPresent | badAbsent | pValue | adjustedPValue | oddsRatio |
| bond:CX\_halide\_aromatic-X\_biphenyl | 2 | 876 | 3 | 50 | 0.002 | 0.636 | 0.038 |
| bond:CX\_halide\_alkyl-Cl\_dichloro\_(1\_1-) | 14 | 864 | 5 | 48 | 0.003 | 1.000 | 0.156 |
| ring:hetero\_[6\_6\_6]\_N\_S\_phenothiazine | 5 | 873 | 3 | 50 | 0.008 | 1.000 | 0.096 |
| ring:aromatic\_biphenyl | 19 | 859 | 5 | 48 | 0.009 | 1.000 | 0.213 |
| bond:CX\_halide\_alkyl-Cl\_trichloro\_(1\_1\_1-) | 6 | 872 | 3 | 50 | 0.011 | 1.000 | 0.115 |
| chain:alkaneLinear\_octyl\_C8 | 14 | 864 | 4 | 49 | 0.016 | 1.000 | 0.199 |
| bond:CX\_halide\_generic-X\_dihalo\_(1\_2-) | 52 | 826 | 8 | 45 | 0.016 | 1.000 | 0.355 |
| bond:C(=O)O\_carboxylicEster\_aliphatic | 23 | 855 | 5 | 48 | 0.018 | 1.000 | 0.259 |
| bond:CX\_halide\_alkyl-X\_dihalo\_(1\_1-) | 72 | 806 | 10 | 43 | 0.020 | 1.000 | 0.385 |
| bond:CX\_halide\_aromatic-Cl\_trihalo\_benzene\_(1\_2\_4-) | 8 | 870 | 3 | 50 | 0.021 | 1.000 | 0.154 |
| bond:CX\_halide\_alkyl-X\_trihalo\_(1\_1\_2-) | 16 | 862 | 4 | 49 | 0.023 | 1.000 | 0.228 |
| ring:hetero\_[6]\_Z\_1\_4- | 35 | 843 | 6 | 47 | 0.025 | 1.000 | 0.326 |
| bond:CX\_halide\_aromatic-X\_dihalo\_benzene\_(1\_2-) | 18 | 860 | 4 | 49 | 0.032 | 1.000 | 0.257 |
| bond:CX\_halide\_aromatic-X\_dihalo\_benzene\_(1\_4-) | 11 | 867 | 3 | 50 | 0.041 | 1.000 | 0.212 |
| chain:aromaticAlkane\_Ph-1\_4-C1\_acyclic | 11 | 867 | 3 | 50 | 0.041 | 1.000 | 0.212 |
| bond:CX\_halide\_alkyl-X\_trihalo\_(1\_1\_1-) | 59 | 819 | 8 | 45 | 0.047 | 1.000 | 0.406 |

The table above shows that aromatic halides are most likely to be associated with discrepancies. However, none of these ToxPrintChemoTypes are significantly associated with discrepancies when adjusting for multiple comparisons. In the next code block, the data for the largest discrepancies will be presented.

kable(dataComparison[abs(dataComparison$logDifference)>2,], digits=3)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | CAS | Css | in\_silico\_derived\_Css | logDifference |
| 240 | 153233-91-1 | 67.547 | 0.651 | -2.016 |
| 106 | 119-47-1 | 21.225 | 0.197 | -2.033 |
| 60 | 111-21-7 | 0.042 | 4.682 | 2.046 |
| 633 | 58-38-8 | 178.967 | 1.598 | -2.049 |
| 352 | 2390-60-5 | 106.666 | 0.931 | -2.059 |
| 756 | 75530-68-6 | 665.423 | 5.791 | -2.060 |
| 278 | 1763-23-1 | 1693.475 | 14.533 | -2.066 |
| 125 | 121-75-5 | 0.131 | 15.506 | 2.074 |
| 280 | 17804-35-2 | 1491.461 | 12.522 | -2.076 |
| 720 | 69-09-0 | 127.448 | 15300.687 | 2.079 |
| 279 | 17696-62-7 | 0.321 | 38.970 | 2.084 |
| 290 | 188489-07-8 | 0.248 | 30.755 | 2.093 |
| 759 | 76-44-8 | 192.188 | 1.517 | -2.103 |
| 243 | 155213-67-5 | 124.031 | 0.929 | -2.126 |
| 153 | 1260-17-9 | 818.618 | 5.757 | -2.153 |
| 583 | 54593-83-8 | 1.347 | 194.245 | 2.159 |
| 452 | 35575-96-3 | 0.021 | 3.520 | 2.231 |
| 5 | 100784-20-1 | 1073.310 | 6.018 | -2.251 |
| 563 | 52479-85-3 | 1617.183 | 8.723 | -2.268 |
| 423 | 321-64-2 | 0.020 | 3.668 | 2.269 |
| 787 | 79538-32-2 | 15.876 | 0.079 | -2.304 |
| 696 | 64706-54-3 | 78.879 | 0.375 | -2.323 |
| 244 | 156-10-5 | 0.132 | 28.894 | 2.340 |
| 620 | 57-97-6 | 457.243 | 2.077 | -2.343 |
| 781 | 789-02-6 | 9.045 | 0.040 | -2.356 |
| 192 | 136-45-8 | 0.081 | 19.503 | 2.380 |
| 429 | 33089-61-1 | 193.331 | 0.792 | -2.388 |
| 525 | 50-29-3 | 9.541 | 0.039 | -2.394 |
| 193 | 136-60-7 | 0.159 | 40.377 | 2.404 |
| 232 | 150378-17-9 | 220.088 | 0.760 | -2.462 |
| 133 | 122-66-7 | 2579.788 | 8.868 | -2.464 |
| 700 | 654055-01-3 | 23704.593 | 79.126 | -2.477 |
| 67 | 112-00-5 | 17.655 | 0.057 | -2.492 |
| 677 | 62-73-7 | 1601.307 | 3.594 | -2.649 |
| 529 | 50-52-2 | 43.167 | 0.095 | -2.659 |
| 594 | 55406-53-6 | 0.037 | 17.762 | 2.678 |
| 202 | 138472-01-2 | 2799.807 | 5.168 | -2.734 |
| 39 | 105-87-3 | 0.401 | 225.300 | 2.749 |
| 1 | 100-01-6 | 4280.371 | 7.461 | -2.759 |
| 26 | 103-24-2 | 0.084 | 0.000 | -2.775 |
| 361 | 24602-86-6 | 264.085 | 0.303 | -2.940 |
| 115 | 120-32-1 | 9420.652 | 7.472 | -3.101 |
| 893 | 95-69-2 | 7487.964 | 4.051 | -3.267 |
| 711 | 67485-29-4 | 3256.682 | 1.706 | -3.281 |
| 512 | 486-56-6 | 43659.048 | 21.939 | -3.299 |
| 425 | 32598-13-3 | 29228.525 | 0.136 | -5.333 |
| 736 | 72-54-8 | 27036.997 | 0.122 | -5.346 |
| 149 | 125533-88-2 | 18860.246 | 0.067 | -5.451 |
| 433 | 33284-52-5 | 29137.084 | 0.096 | -5.481 |
| 448 | 35065-27-1 | 23703.806 | 0.020 | -6.068 |
| 469 | 38411-22-2 | 21051.103 | 0.012 | -6.247 |
| 443 | 33979-03-2 | 22365.734 | 0.008 | -6.441 |
| 163 | 129453-61-8 | 0.004 | 14390.002 | 6.522 |

In the next code block, the data for the largest discrepancies will be presented after filtering.

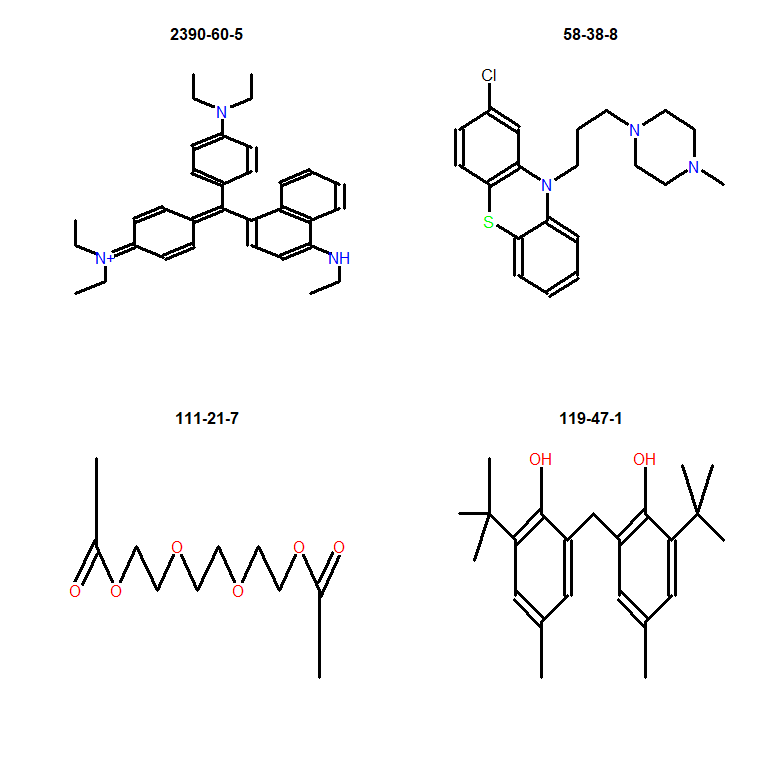
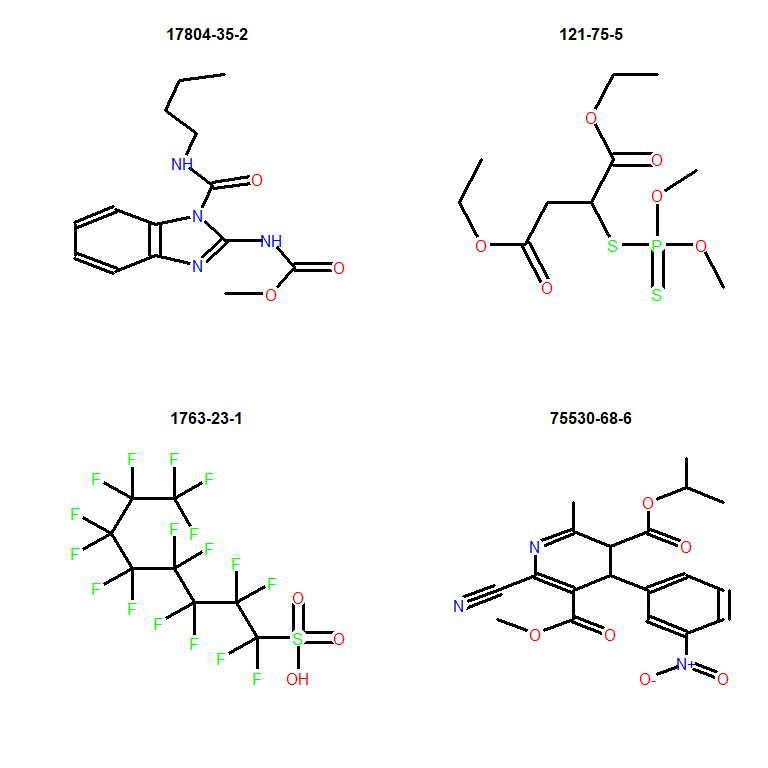
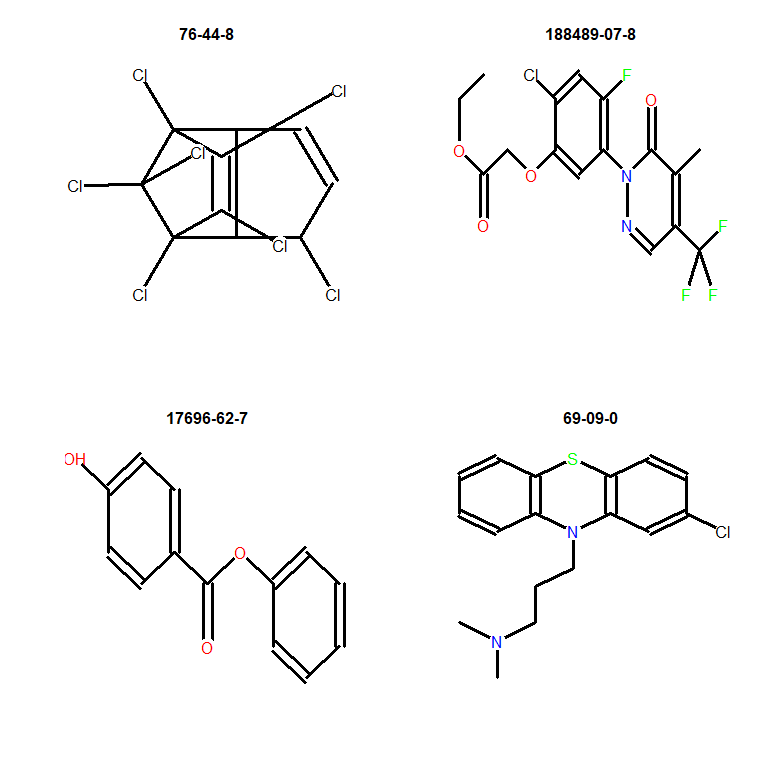
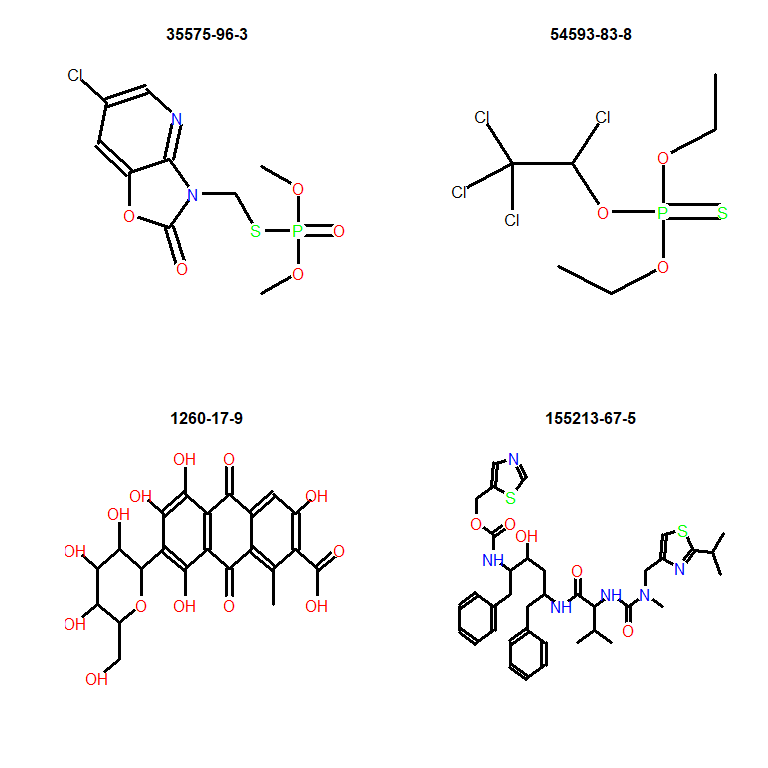
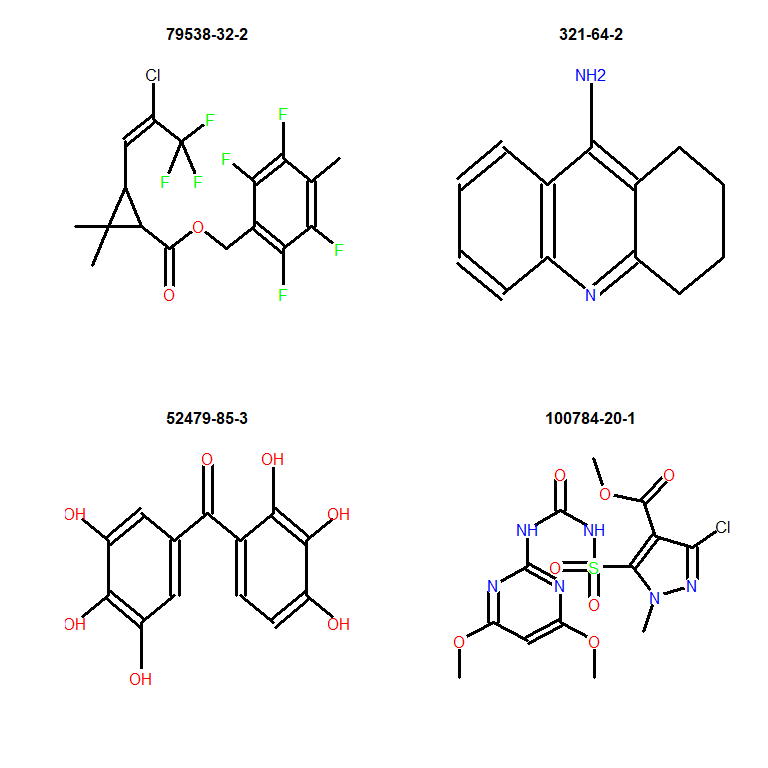
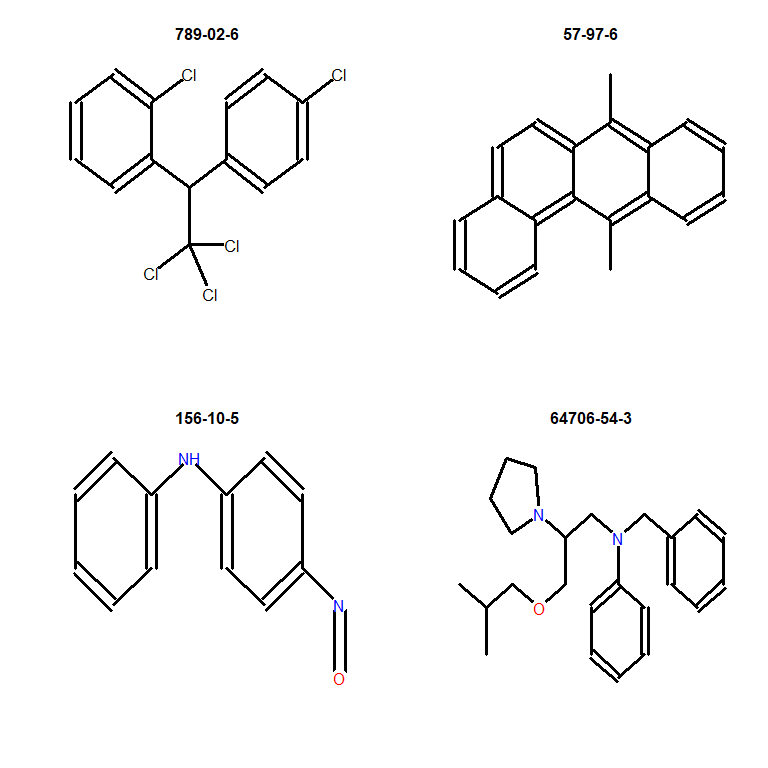
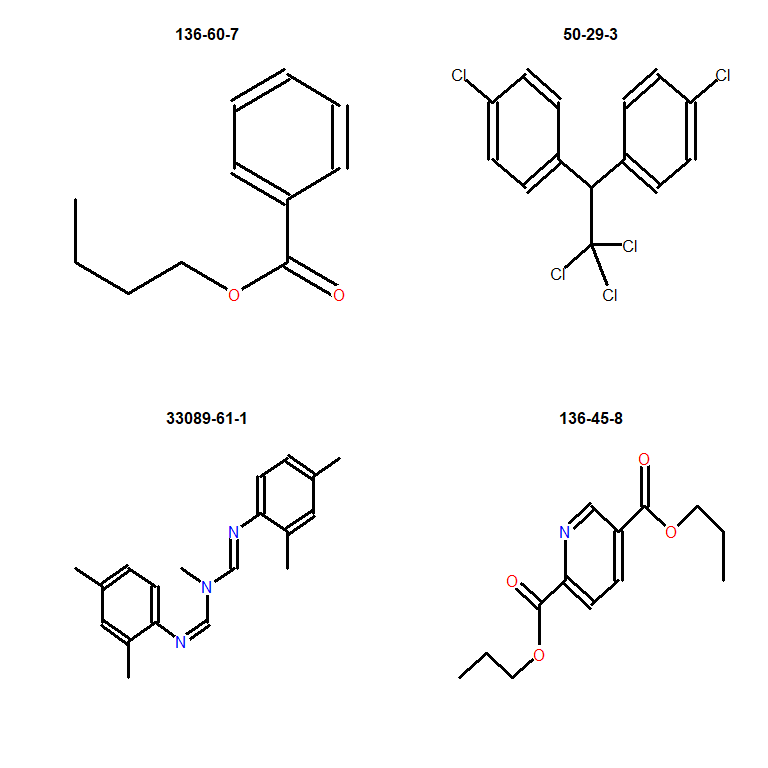
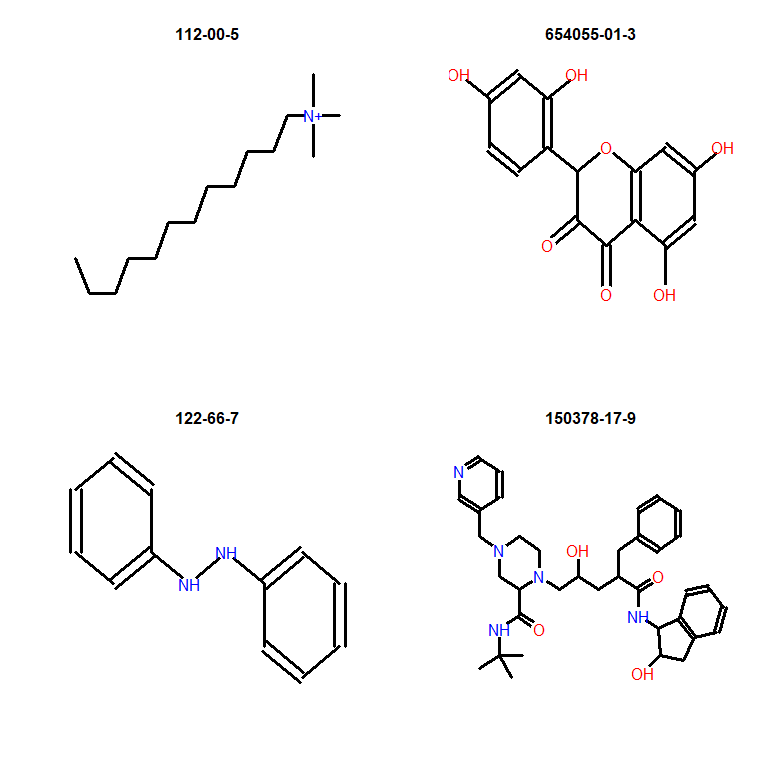
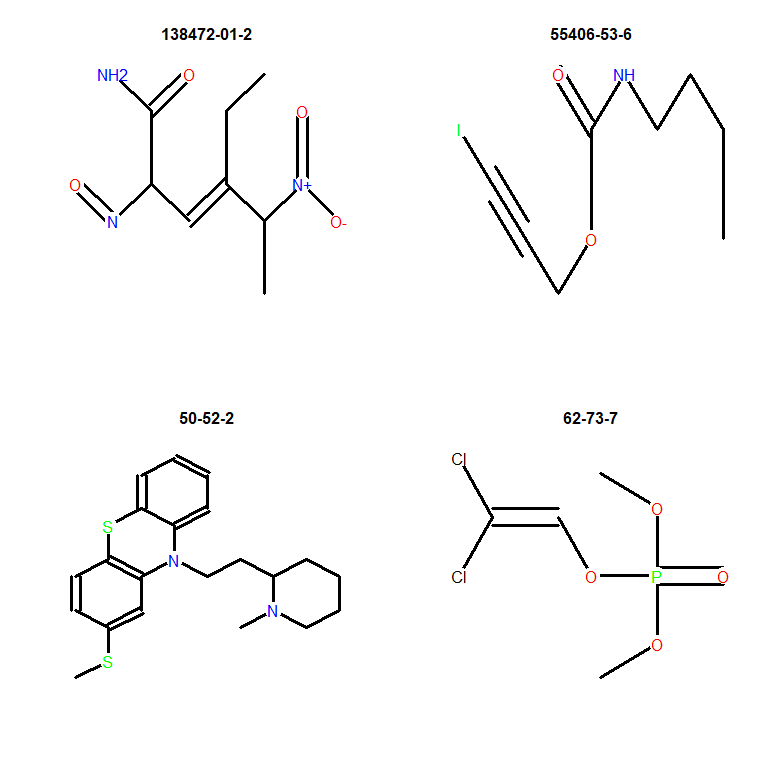
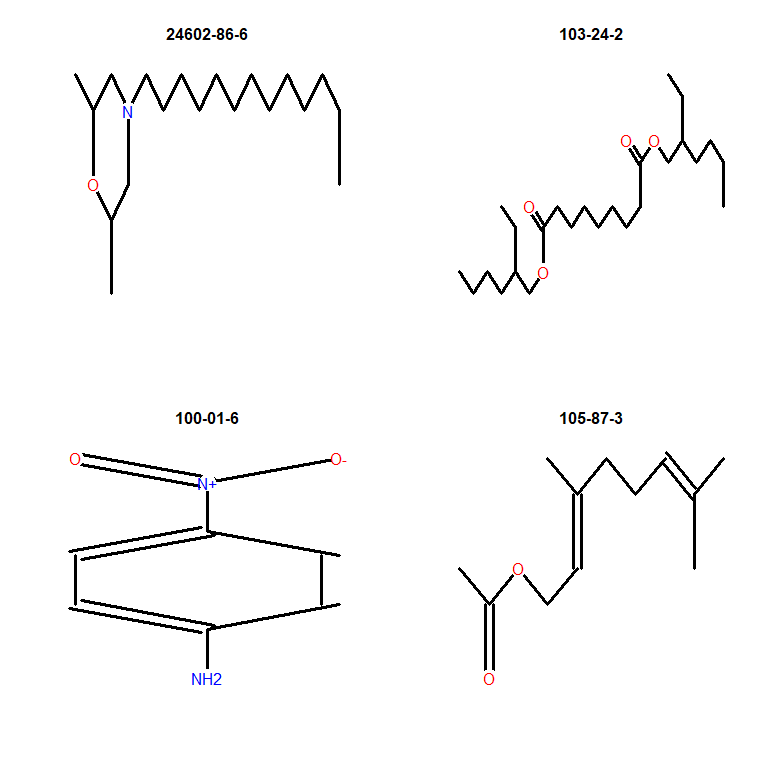
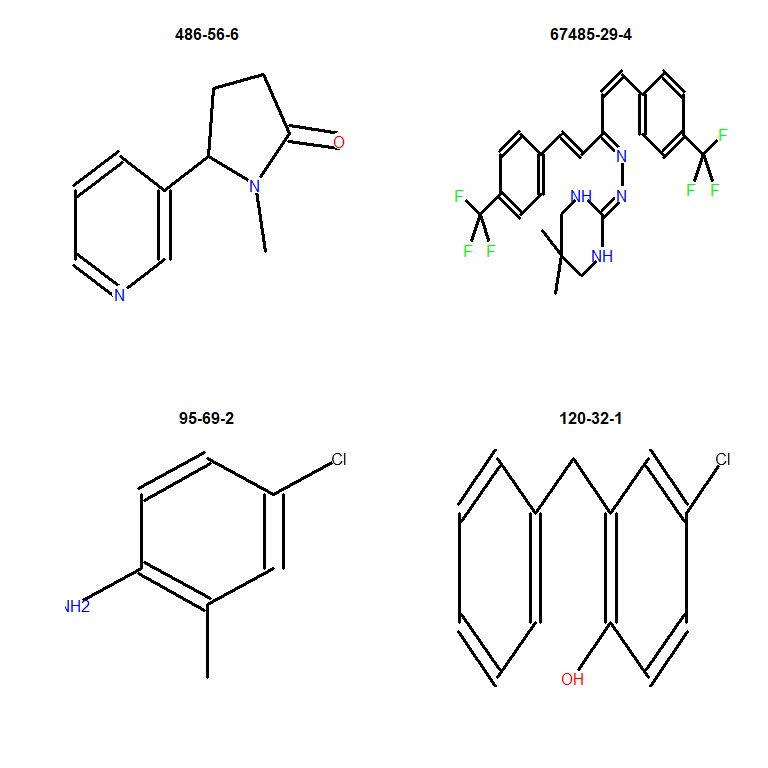
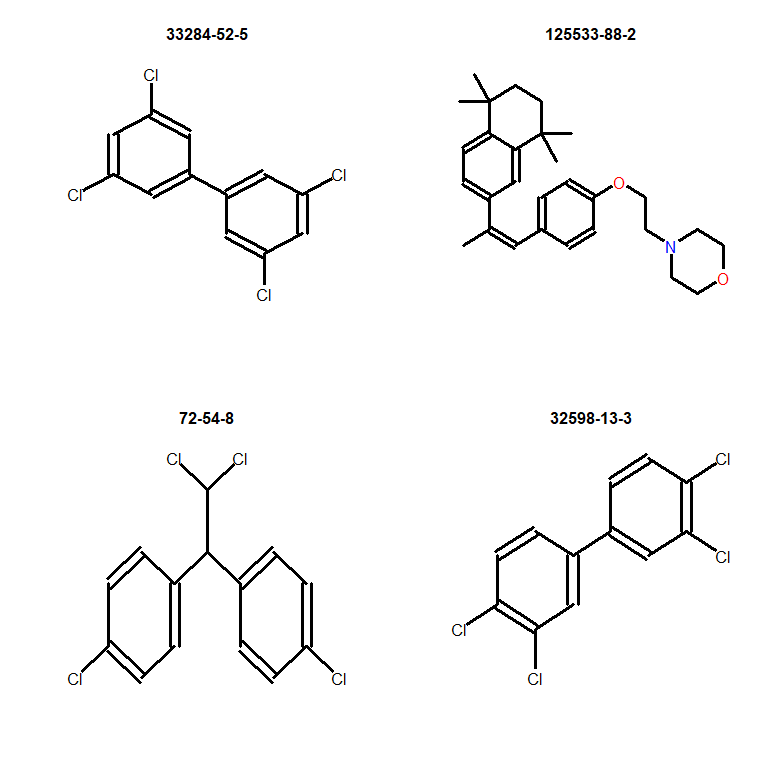
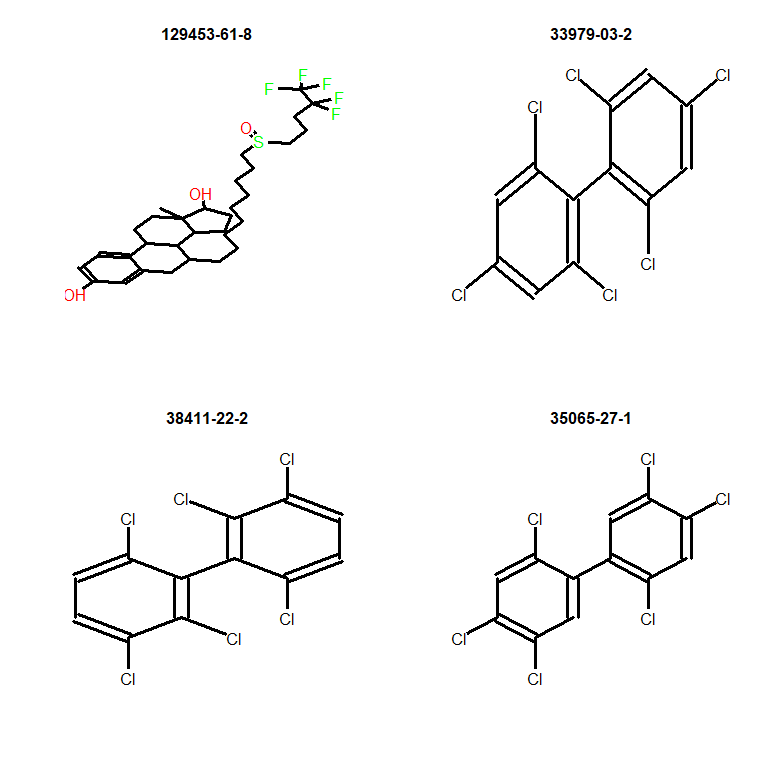
kable(filteredDataComparison[abs(filteredDataComparison$logDifference)>2,], digits=3)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | CAS | Css | in\_silico\_derived\_Css | logDifference |
| 633 | 58-38-8 | 178.967 | 1.598 | -2.049 |
| 756 | 75530-68-6 | 665.423 | 5.791 | -2.060 |
| 280 | 17804-35-2 | 1491.461 | 12.522 | -2.076 |
| 720 | 69-09-0 | 127.448 | 15300.687 | 2.079 |
| 279 | 17696-62-7 | 0.321 | 38.970 | 2.084 |
| 290 | 188489-07-8 | 0.248 | 30.755 | 2.093 |
| 583 | 54593-83-8 | 1.347 | 194.245 | 2.159 |
| 452 | 35575-96-3 | 0.021 | 3.520 | 2.231 |
| 423 | 321-64-2 | 0.020 | 3.668 | 2.269 |
| 696 | 64706-54-3 | 78.879 | 0.375 | -2.323 |
| 244 | 156-10-5 | 0.132 | 28.894 | 2.340 |
| 192 | 136-45-8 | 0.081 | 19.503 | 2.380 |
| 429 | 33089-61-1 | 193.331 | 0.792 | -2.388 |
| 193 | 136-60-7 | 0.159 | 40.377 | 2.404 |
| 133 | 122-66-7 | 2579.788 | 8.868 | -2.464 |
| 700 | 654055-01-3 | 23704.593 | 79.126 | -2.477 |
| 677 | 62-73-7 | 1601.307 | 3.594 | -2.649 |
| 529 | 50-52-2 | 43.167 | 0.095 | -2.659 |
| 594 | 55406-53-6 | 0.037 | 17.762 | 2.678 |
| 202 | 138472-01-2 | 2799.807 | 5.168 | -2.734 |
| 39 | 105-87-3 | 0.401 | 225.300 | 2.749 |
| 1 | 100-01-6 | 4280.371 | 7.461 | -2.759 |
| 115 | 120-32-1 | 9420.652 | 7.472 | -3.101 |
| 893 | 95-69-2 | 7487.964 | 4.051 | -3.267 |
| 512 | 486-56-6 | 43659.048 | 21.939 | -3.299 |

The previous code blocks show that from the 53 chemicals with a log difference greater than 2 (range -6.4411653 to 6.5220195) there are only 25 chemicals remaining (range -3.298855 to 2.7493535)

In the next code block, the unfiltered structures associated with the largest discrepancies are displayed in order of decreasing discrepancy.

worstPredictedCAS <- rev(dataComparison[abs(dataComparison$logDifference)>2,]$CAS)  
worstPredictedSDF <- smiles2sdf(chem.physical\_and\_invitro.data[worstPredictedCAS,"SMILES.desalt"])  
cid(worstPredictedSDF) <- worstPredictedCAS  
  
for (i in 1:length(worstPredictedSDF)) {  
 attributes(worstPredictedSDF[[i]])$header["Molecule\_Name"] <- worstPredictedCAS[i]  
}  
  
##Plot structures beginning with worst predictions first   
plotNum <- 1  
plotNumTop <- plotNum + 3  
while(plotNum < length(worstPredictedSDF)) {  
 if (plotNumTop > length(worstPredictedSDF)) { plotNumTop <- length(worstPredictedSDF)}  
 invisible(plot(worstPredictedSDF[plotNum:(plotNum+3)]))  
 plotNum <- plotNum + 4  
 plotNumTop <- plotNumTop + 4  
}



Viewing of the structures reiterates the point that caution needs to be used when interpreting results for poly-halogenated aromatics, specifically polychlorinated biphenyls, when *in silico* toxicokinetics values from this model are used. To be conservative, we are recommending filters be applied and a Css value of 0.1 be used for any compounds with an *in silico* derived Css below 0.1 as there are only 12 compounds in HTTK with a Css below 0.1.

**Collectively, the results provide confidence that *in silico* predictions of Human Clint and Fup serve as useful parameters in the HTTK model. These data are used below to prioritize DSL compounds tested in the ToxCast database**

## Bridging HTTK Data Gaps for DSL Using *in silico* Predictions

The first part of this workflow is performed outside of the R workflow. ADMET Predictor 10 was used to obtain Human Clint and Fup values, where possible, for DSL compounds based on CASRN. This data was saved in the file Entire\_DSL\_HTTK\_Data.xlsx. In the code below, Open Babel in R (ChemmineOB library) was used to provide LogP and MW based on SMILES:

inHTTKDSL <- read.xlsx("Entire\_DSL\_HTTK\_Data.xlsx", colNames=T)  
  
smiles=inHTTKDSL$SMILES  
obTable = matrix(nrow=length(smiles), ncol=13)  
for (i in 1:length(smiles)) {   
 obTable[i,] = unlist(prop\_OB(forEachMol("SMILES", smiles[i], identity)))   
 }  
colnames(obTable) = colnames(prop\_OB(forEachMol("SMILES", smiles[i], identity)))  
  
inHTTKDSL$logP <- obTable[,"logP"]  
inHTTKDSL$MW <- obTable[,"MW"]  
  
kable(head(inHTTKDSL[,c("CASRN","Human.Clint","Human.Funbound.plasma","logP","MW")]))

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CASRN | Human.Clint | Human.Funbound.plasma | logP | MW |
| 5421-46-5 | 6.970825 | 0.4203196 | -0.9577 | 109.1475 |
| 328-42-7 | 5.416217 | 0.6306497 | -0.8852 | 132.0716 |
| 25360-10-5 | 90.978443 | 0.1004343 | 3.6652 | 160.3201 |
| 100-53-8 | 18.117002 | 0.2091790 | 2.1164 | 124.2034 |
| 759-05-7 | 5.506052 | 0.3529942 | 0.2961 | 116.1152 |
| 328-50-7 | 4.653704 | 0.5071394 | -0.4951 | 146.0981 |

In total, the requisite HTTK data were obtained for 16637 DSL compounds.

### Calculate Steady State Plasma Concentrations

A steady-state plasma concentration is calculated for the 16637 DSL compounds using the code below.

##Import data into HTTK without overwriting existing data  
chem.physical\_and\_invitro.data <- add\_chemtable(inHTTKDSL,current.table=chem.physical\_and\_invitro.data, data.list=list(Compound="HC.Name", CAS="CASRN", Funbound.plasma="Human.Funbound.plasma"),species="Human", reference="ADMET10", overwrite=F)  
chem.physical\_and\_invitro.data <- add\_chemtable(inHTTKDSL,current.table=chem.physical\_and\_invitro.data, data.list=list(Compound="HC.Name", CAS="CASRN", logp="logP"), reference="ChemmineOB", overwrite=F)  
chem.physical\_and\_invitro.data <- add\_chemtable(inHTTKDSL,current.table=chem.physical\_and\_invitro.data, data.list=list(Compound="HC.Name", CAS="CASRN", MW="MW"), reference="ChemmineOB", overwrite=F)  
chem.physical\_and\_invitro.data <- add\_chemtable(inHTTKDSL,current.table=chem.physical\_and\_invitro.data, data.list=list(Compound="HC.Name", CAS="CASRN", Clint="Human.Clint"), species="Human", reference="ADMET10",overwrite=F)  
chem.physical\_and\_invitro.data <- add\_chemtable(inHTTKDSL,current.table=chem.physical\_and\_invitro.data, data.list=list(Compound="HC.Name", CAS="CASRN", SMILES.desalt="SMILES"),species="Human", reference="PubChem", overwrite=F)  
  
  
##Calculate Css  
Css <- vector()  
  
for (i in inHTTKDSL$CASRN) {  
  
 tempCss = "NA" #reset tempCSS each time  
   
 if (i %in% get\_cheminfo("CAS")) { #checks if Css can be modelled  
 tempCss = calc\_mc\_css(chem.cas=i,  
 which.quantile=c(0.95), restrictive.clearance=T,   
 model="3compartmentss", output.units="uM", species="Human",   
 method="dr", well.stirred.correction=T, suppress.messages=F,  
 return.samples=F)  
 }  
   
 Css <- c(Css, tempCss)  
}  
  
##Restore chem.physical\_and\_invitro.data to default  
rm(chem.physical\_and\_invitro.data)  
  
httkResults <- cbind(inHTTKDSL, Css)  
  
#Defaulting Css value to 0.1 to be conservative.   
httkResults$Css[httkResults$Css<0.1] <- 0.1  
  
kable(head(httkResults[,cc("CID", "CASRN", "Human.Clint", "Human.Funbound.plasma", "logP", "MW", "Css")]),digits=2)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| CID | CASRN | Human.Clint | Human.Funbound.plasma | logP | MW | Css |
| 21534 | 5421-46-5 | 6.97 | 0.42 | -0.96 | 109.15 | 5.85419281494965 |
| 970 | 328-42-7 | 5.42 | 0.63 | -0.89 | 132.07 | 4.92410598207412 |
| 520196 | 25360-10-5 | 90.98 | 0.10 | 3.67 | 160.32 | 10.2300640331242 |
| 7509 | 100-53-8 | 18.12 | 0.21 | 2.12 | 124.20 | 5.69134867868717 |
| 49 | 759-05-7 | 5.51 | 0.35 | 0.30 | 116.12 | 9.06219513737892 |
| 51 | 328-50-7 | 4.65 | 0.51 | -0.50 | 146.10 | 5.8862945220756 |

Code below identifies which chemicals pass filtering for HTTK.

LipinskiFailuresDSLHTTK <- as.vector(sapply(inHTTKDSL$SMILES, ruleOf5Violations))  
  
filteredDSLHTTK <- inHTTKDSL[LipinskiFailuresDSLHTTK == 0,]  
filteredDSLHTTK <- filteredDSLHTTK[filteredDSLHTTK$Outside\_Applicability\_Domain == FALSE,]  
filteredDSLHTTK <- filteredDSLHTTK[filteredDSLHTTK$'%Fraction\_Absorbed' > 10,]  
filteredDSLHTTK <- filteredDSLHTTK[filteredDSLHTTK$'%Fraction\_Bioavailable' > 10,]  
filteredDSLHTTK <- filteredDSLHTTK$CASRN

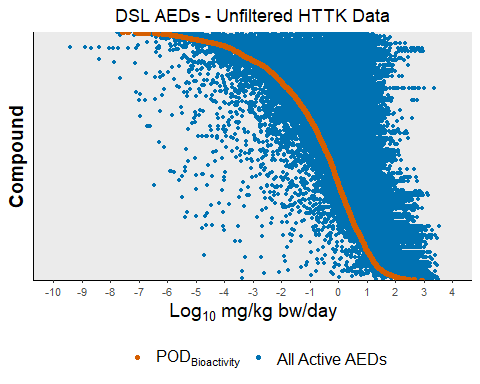
There were 6571 compounds with HTTK data remaining after filtering out of 16637 DSL compounds. Only the filtered data is presented in the mansucript, but all data will be provided in the supplementary material with filtering criteria included.

#This code block adds the HTTK data to the summary table to be reported at the end of the workflow  
  
rownames(httkResults) <- httkResults$CASRN  
overlappingCASRN <- httkResults$CASRN[httkResults$CASRN %in% summaryTable[,"CASRN"]]  
  
summaryTable[overlappingCASRN, "Compound"] <- httkResults[overlappingCASRN, "HC.Name"]  
summaryTable[overlappingCASRN, "SMILES"] <- httkResults[overlappingCASRN, "SMILES"]  
  
summaryTable[overlappingCASRN, "MolecularWeight"] <- httkResults[overlappingCASRN, "MW"]  
summaryTable[as.numeric(summaryTable[, "MolecularWeight"])>=500,"MolecularWeightViolation"] <- "TRUE"  
summaryTable[as.numeric(summaryTable[, "MolecularWeight"])<500,"MolecularWeightViolation"] <- "FALSE"  
  
summaryTable[overlappingCASRN, "LogP\_OpenBabel"] <- httkResults[overlappingCASRN, "logP"]  
summaryTable[overlappingCASRN, "LogP\_RCDK"] <- as.vector(sapply(summaryTable[overlappingCASRN, "SMILES"], xLogPFunction))  
summaryTable[as.numeric(summaryTable[, "LogP\_RCDK"])>5,"LogPViolation"] <- "TRUE"  
summaryTable[as.numeric(summaryTable[, "LogP\_RCDK"])<=5,"LogPViolation"] <- "FALSE"  
  
summaryTable[overlappingCASRN, "ADMETOutsideApplicabilityDomain"] <- httkResults[overlappingCASRN, "Outside\_Applicability\_Domain"]  
  
summaryTable[overlappingCASRN, "FractionAbsorbed"] <- httkResults[overlappingCASRN, "%Fraction\_Absorbed"]  
summaryTable[as.numeric(summaryTable[, "FractionAbsorbed"])<=10,"FractionAbsorbedViolation"] <- "TRUE"  
summaryTable[as.numeric(summaryTable[, "FractionAbsorbed"])>10,"FractionAbsorbedViolation"] <- "FALSE"  
  
summaryTable[overlappingCASRN, "FractionBioavailable"] <- httkResults[overlappingCASRN, "%Fraction\_Bioavailable"]  
summaryTable[as.numeric(summaryTable[, "FractionBioavailable"])<=10,"FractionBioavailableViolation"] <- "TRUE"  
summaryTable[as.numeric(summaryTable[, "FractionBioavailable"])>10,"FractionBioavailableViolation"] <- "FALSE"  
  
summaryTable[overlappingCASRN, "HBondDonorsCount"] <- as.vector(sapply(summaryTable[overlappingCASRN, "SMILES"], HBondDonorCountFunction))  
summaryTable[as.numeric(summaryTable[, "HBondDonorsCount"])>5,"HBondDonorsCountViolation"] <- "TRUE"  
summaryTable[as.numeric(summaryTable[, "HBondDonorsCount"])<=5,"HBondDonorsCountViolation"] <- "FALSE"  
  
summaryTable[overlappingCASRN, "HBondAcceptorsCount"] <- as.vector(sapply(summaryTable[overlappingCASRN, "SMILES"], HBondAcceptorCountFunction))  
summaryTable[as.numeric(summaryTable[, "HBondAcceptorsCount"])>10,"HBondAcceptorsCountViolation"] <- "TRUE"  
summaryTable[as.numeric(summaryTable[, "HBondAcceptorsCount"])<=10,"HBondAcceptorsCountViolation"] <- "FALSE"  
  
summaryTable[overlappingCASRN, "Css(MicroMolar)"] <- httkResults[overlappingCASRN, "Css"]  
summaryTable[overlappingCASRN[!overlappingCASRN %in% get\_cheminfo("CAS")], "CssSource"] <- "in\_silico"

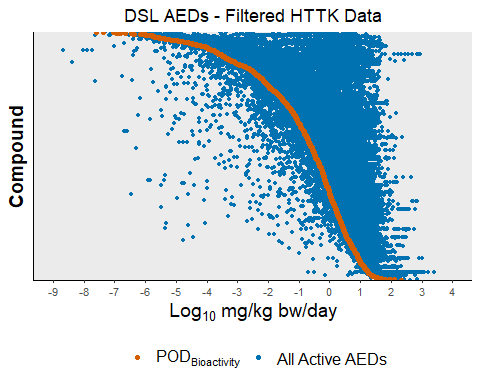
## Prioritize DSL After Bridging HTTK Data Gap

*in vitro* to *in vivo* extrapolation can now be applied to derive Administered Equivalent Doses (AEDs) in mg/kg bw/day for DSL compounds present in ToxCast.

dslDat5 <- merge(httkResults, dat5, by.x="CASRN", by.y="casn")  
dslDat5 <- as.data.table(dslDat5[,c("CASRN", "HC.Name", "SMILES", "Css", "aenm", "chid", "chnm", "modl\_ga\_uM", "modl\_ga\_0.05\_uM")])  
  
#Defaulting Css value to 0.1 to be conservative.   
dslDat5$Css[dslDat5$Css<0.1]= 0.1  
  
Log.AED <- log10(dslDat5$modl\_ga\_uM/as.numeric(dslDat5$Css))  
Log.POD.Bioactivity <- log10(dslDat5$modl\_ga\_0.05\_uM/as.numeric(dslDat5$Css))  
  
AEDs <- cbind(dslDat5, Log.AED, Log.POD.Bioactivity)  
colnames(AEDs)[2] <- "Provided Compound Name"  
  
ggplot() +  
 geom\_point(data = AEDs, aes(x= AEDs$"Log.AED", y = AEDs$"Provided Compound Name", color="Log.AED"), size=1) +   
 geom\_point(data = AEDs, aes(x= AEDs$"Log.POD.Bioactivity", y = AEDs$"Provided Compound Name", color="Log.POD.Bioactivity"), size=1.5) +  
 ylab("Compound") +   
 xlab(expression(Log["10"]~"mg/kg"~"bw/day")) +  
 scale\_x\_continuous(breaks=seq( floor(min(c(AEDs$"Log.AED", AEDs$"Log.POD.Bioactivity"))), ceiling(max(c(AEDs$"Log.AED", AEDs$"Log.POD.Bioactivity"))) ,1)) +  
 scale\_y\_discrete(limits= unique(AEDs$"Provided Compound Name"[rev(order(AEDs$"Log.POD.Bioactivity"))])) +  
 coord\_cartesian(xlim = c(floor(min(c(AEDs$"Log.AED", AEDs$"Log.POD.Bioactivity"))),ceiling(max(c(AEDs$"Log.AED", AEDs$"Log.POD.Bioactivity"))))) +  
 theme\_bw() +  
 theme(panel.border = element\_blank(),   
 panel.grid.major.y = element\_line(),  
 panel.grid.minor.y = element\_blank(),  
 panel.grid.minor.x = element\_blank(),  
 axis.line = element\_line(colour = "black"),  
 axis.text.y=element\_blank(),  
 axis.ticks.y=element\_blank(),  
 axis.text = element\_text(size=8),  
 axis.title = element\_text(size=14, face="bold")) +  
 ggtitle("DSL AEDs - Unfiltered HTTK Data") +  
 scale\_colour\_manual(  
 breaks=c("Log.POD.Bioactivity", "Log.AED"),  
 values=c("#D55E00", "#0072B2"),  
 labels=c(expression(POD[Bioactivity]), "All Active AEDs")) +  
 theme(legend.position="bottom", legend.title=element\_blank(), legend.text=element\_text(size=12), plot.title=element\_text(hjust = 0.5))



AEDsFiltered <- AEDs[AEDs$CASRN %in% filteredDSLHTTK,]  
  
ggplot() +  
 geom\_point(data = AEDsFiltered, aes(x= AEDsFiltered$"Log.AED", y = AEDsFiltered$"Provided Compound Name", color="Log.AED"), size=1) +   
 geom\_point(data = AEDsFiltered, aes(x= AEDsFiltered$"Log.POD.Bioactivity", y = AEDsFiltered$"Provided Compound Name", color="Log.POD.Bioactivity"), size=1.5) +  
 ylab("Compound") +   
 xlab(expression(Log["10"]~"mg/kg"~"bw/day")) +  
 scale\_x\_continuous(breaks=seq( floor(min(c(AEDsFiltered$"Log.AED", AEDsFiltered$"Log.POD.Bioactivity"))), ceiling(max(c(AEDsFiltered$"Log.AED", AEDsFiltered$"Log.POD.Bioactivity"))) ,1)) +  
 scale\_y\_discrete(limits= unique(AEDsFiltered$"Provided Compound Name"[rev(order(AEDsFiltered$"Log.POD.Bioactivity"))])) +  
 coord\_cartesian(xlim = c(floor(min(c(AEDsFiltered$"Log.AED", AEDsFiltered$"Log.POD.Bioactivity"))),ceiling(max(c(AEDsFiltered$"Log.AED", AEDsFiltered$"Log.POD.Bioactivity"))))) +  
 theme\_bw() +  
 theme(panel.border = element\_blank(),   
 panel.grid.major.y = element\_line(),  
 panel.grid.minor.y = element\_blank(),  
 panel.grid.minor.x = element\_blank(),  
 axis.line = element\_line(colour = "black"),  
 axis.text.y=element\_blank(),  
 axis.ticks.y=element\_blank(),  
 axis.text = element\_text(size=8),  
 axis.title = element\_text(size=14, face="bold")) +  
 ggtitle("DSL AEDs - Filtered HTTK Data") +  
 scale\_colour\_manual(  
 breaks=c("Log.POD.Bioactivity", "Log.AED"),  
 values=c("#D55E00", "#0072B2"),  
 labels=c(expression(POD[Bioactivity]), "All Active AEDs")) +  
 theme(legend.position="bottom", legend.title=element\_blank(), legend.text=element\_text(size=12), plot.title=element\_text(hjust = 0.5))



In total, AEDs could be obtained for 1708 compounds with filtering HTTK data and 2974 compounds without filtering.

uniqueAEDs <- as.matrix(AEDs[!duplicated(AEDs$CASRN),])  
  
rownames(uniqueAEDs) <- uniqueAEDs[,"CASRN"]  
overlappingCASRN <- uniqueAEDs[,"CASRN"][uniqueAEDs[,"CASRN"] %in% summaryTable[,"CASRN"]]  
  
summaryTable[overlappingCASRN, "AED(mg/kg-bw/day)"] <- 10^as.numeric(uniqueAEDs[overlappingCASRN, "Log.POD.Bioactivity"])  
summaryTable[overlappingCASRN, "BioactivityConcentration(MicroMolar)"] <- uniqueAEDs[overlappingCASRN, "modl\_ga\_0.05\_uM"]  
summaryTable[overlappingCASRN, "BioactivitySource"] <- "ToxCast"

## Second Data Gap - DSL Compounds Ouside of ToxCast

To address this gap, read-across is used. Specifically, structural fingerprints are compared between ToxCast and DSL compounds to find structurally similar compounds (analogs). The bioactivity concentrations are recorded for the most similar ten or fewer analogs, with a Tanimoto Coefficient above threshold, and the read-across bioactivity concentration is calculated using the Generalized Read-Across (GenRA) equation. Before applying this approach, the ToxCast compounds are used as control. Specifically, each ToxCast compound is iteratively compared against the other compounds to find read-across analogs. The read-across bioactivity concentration is compared against the derived bioactivity concentration for that compound. Different molecular fingerprints (ToxPrint Chemotypes, PubChem Fingerprints, and Extended Connectivity Fingerprints) and Tanimoto Coefficients are explored to optimize the approach. More details are below.

### Obtain ToxPrint Chemotypes for ToxCast Compounds

This step is necessary to allow read-across, using ToxPrint Chemotypes, between ToxCast bioactivity results and DSL compounds outside of ToxCast.

#### Retrieve SMILES for Compounds and Prepare SDF

##Convert GSID to SMILES using PubChem API  
uniqueGSID = unique(dat5$chid)  
uniqueSmiles <- vector()  
for (i in 1:length(uniqueGSID)) {   
 uniqueSmiles <- c(uniqueSmiles,   
 strsplit(content(GET(paste0("https://pubchem.ncbi.nlm.nih.gov/rest/pug/compound/name/DSSTox\_GSID\_",  
 uniqueGSID[i], "/property/CanonicalSMILES/txt")), as="text"), "\n")[[1]][1])  
 }  
  
##This repeats the scrape for instances of Status 503 until SMILES are obtained. Maximum number of tries are set to 20 to prevent infinite loop in the case of internet connection issues.  
counter <- 1  
while (length(grep("Status: 503", uniqueSmiles)) > 0) {  
 if (counter > 20) { break }  
 counter <- counter + 1  
 for (i in grep("Status: 503", uniqueSmiles)) {  
 uniqueSmiles[i] <- strsplit(content(  
 GET(paste0("https://pubchem.ncbi.nlm.nih.gov/rest/pug/compound/name/DSSTox\_GSID\_",   
 uniqueGSID[i], "/property/CanonicalSMILES/txt")), as="text"), "\n")[[1]][1]  
 }  
}  
  
##Remove compounds without SMILES  
uniqueGSID <- uniqueGSID[-grep("Status: 404", uniqueSmiles)]  
uniqueSmiles <- uniqueSmiles[-grep("Status: 404", uniqueSmiles)]  
  
##Convert SMILES to SDFset  
sdfsetToxCast = smiles2sdf(uniqueSmiles)  
cid(sdfsetToxCast) = as.character(uniqueGSID)  
  
for (i in 1:length(sdfsetToxCast)) {   
 attributes(sdfsetToxCast[[i]])$header["Molecule\_Name"] = uniqueGSID[i]   
 }  
  
#Store data as SDF for other applications and to avoid having to scrape data again  
write.SDF(sdfsetToxCast, file="ToxCast.sdf")  
  
#Store SMILES to avoid having to scrape again  
fileConn <- file("toxcastSmiles.txt")  
writeLines(uniqueSmiles, fileConn)  
close(fileConn)  
  
#Store GSID to avoid having to scrape again  
fileConn <- file("toxcastGSID.txt")  
writeLines(as.character(uniqueGSID), fileConn)  
close(fileConn)

#### Convert SDF to ToxPrint Chemotypes

As mentioned previously, this step was performed outside of the R workflow using Chemotyper v1.0.r12976 and ToxPrintv2.0\_r711.xml. The ToxCast.sdf and a previously created DSL SDF were both imported into the ChemoTyper. The ToxPrint Chemotypes were identified for structures that could be resolved and the results matrices were exported as TSV files. The results matrices are available in xlsx format and imported back into the workflow.

toxCastToxPrintFP <- read.xlsx("toxprint\_V2\_vs\_ToxCastSDF.xlsx", colNames=T)  
rownames(toxCastToxPrintFP) <- toxCastToxPrintFP[,1]  
toxCastToxPrintFP <- toxCastToxPrintFP[,-1]

#### Convert SMILES to Other Fingerprints

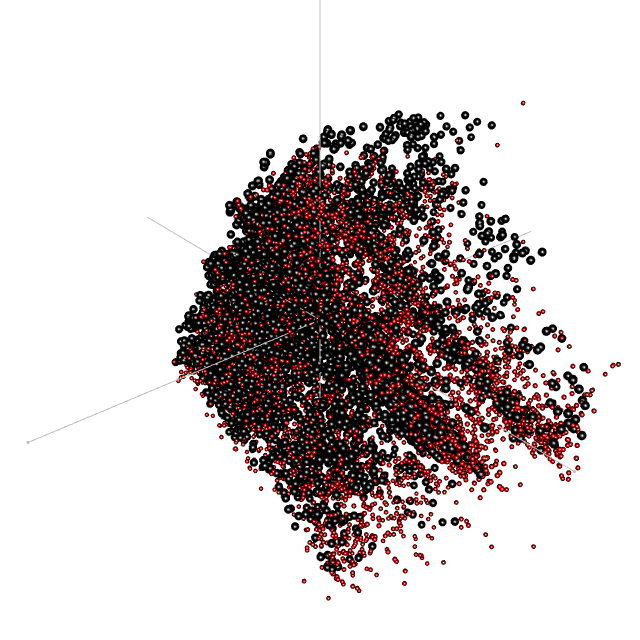
#Pubchem Fingerprints  
toxCastPubChemFP <- matrix(rep( 0, len=881\*length(uniqueSmiles)), nrow=length(uniqueSmiles))  
rownames(toxCastPubChemFP) <- uniqueGSID  
  
for (i in 1:length(uniqueSmiles)) {  
 toxCastPubChemFP[i,attributes(get.fingerprint(parse.smiles(uniqueSmiles[i])[[1]], type="pubchem"))$bits] <- 1  
}  
  
write.xlsx(toxCastPubChemFP, file="toxcastPubChemFP.xlsx")  
  
#Morgan (Extended Connectivity 6) Fingerprints  
toxCastECFP <- matrix(rep( 0, len=1024\*length(uniqueSmiles)), nrow=length(uniqueSmiles))  
rownames(toxCastECFP) <- uniqueGSID  
  
for (i in 1:length(uniqueSmiles)) {  
 toxCastECFP[i,attributes(get.fingerprint(parse.smiles(uniqueSmiles[i])[[1]], type="circular"))$bits] <- 1  
}  
  
write.xlsx(toxCastECFP, file="toxcastECFP.xlsx")

### Visualize Chemical Space Comparison of ToxCast and DSL (Optional)

Here Classical Multidimensioal Scaling is used to visualize in three-dimensional space the comparison of ToxCast and DSL chemical space. DSL compounds outside of this domain do not have suitable read-across analogs.

dslFP <- read.xlsx("toxprint\_V2\_vs\_DSL.xlsx", colNames=T)  
rownames(dslFP) <- dslFP[,2]  
dslFP <- dslFP[,-c(1,2)]  
  
allFP <- rbind(toxCastToxPrintFP, dslFP)  
  
#remove empty rows  
allFP <- allFP[rowSums(allFP)!=0,]  
  
##Classical Multidimensioal Scaling (Principal Coordinates Analysis)  
#It may be necessary here to increase memory ussage with function memory.limit()  
coord <- cmdscale(dist(allFP), k=3)   
colnames(coord) <- c("x","y","z")  
  
ToxCast1\_DSL2 <- rep(1, nrow(coord))  
coord <- cbind(coord, ToxCast1\_DSL2)  
coord[grep("-", rownames(coord)),4] <- 2

#rgl\_init() function modified from custom function found online (Statistical tools for high-throughput data analysis - A complete guide to 3D visualization device system in R - R software and data visualization)  
rgl\_init <- function(new.device = FALSE, bg = "white", width = 640) {   
 if( new.device | rgl.cur() == 0 ) {  
 rgl.open()  
 par3d(windowRect = 50 + c( 0, 0, width, width ) )  
 rgl.bg(color = bg )  
 }  
 rgl.clear(type = c("shapes", "bboxdeco"))  
 rgl.viewpoint(theta = 15, phi = 20, zoom = 0.7)  
}  
  
rgl\_add\_axes <- function(x, y, z, axis.col = "grey",  
 xlab = "", ylab="", zlab="", show.bbox = FALSE, bbox.col = c("#333377","black"))  
 {   
   
 lim <- function(x){c(-max(abs(x)), max(abs(x))) \* 1.1}  
 # Add axes  
 xlim <- lim(x); ylim <- lim(y); zlim <- lim(z)  
 rgl.lines(xlim, c(0, 0), c(0, 0), color = axis.col)  
 rgl.lines(c(0, 0), ylim, c(0, 0), color = axis.col)  
 rgl.lines(c(0, 0), c(0, 0), zlim, color = axis.col)  
   
 # Add a point at the end of each axes to specify the direction  
 axes <- rbind(c(xlim[2], 0, 0), c(0, ylim[2], 0),   
 c(0, 0, zlim[2]))  
 rgl.points(axes, color = axis.col, size = 3)  
   
 # Add axis labels  
 rgl.texts(axes, text = c(xlab, ylab, zlab), color = axis.col,  
 adj = c(0.5, -0.8), size = 2)  
   
   
 # Add bounding box decoration  
 if(show.bbox){  
 rgl.bbox(color=c(bbox.col[1],bbox.col[2]), alpha = 0.5,   
 emission=bbox.col[1], specular=bbox.col[1], shininess=5,   
 xlen = 3, ylen = 3, zlen = 3)   
 }  
}  
  
#Plot MDS  
rgl\_init()  
rgl.viewpoint(theta=50, phi=30, fov=60, zoom=0.5)  
spheres3d(coord[coord[,4]==2,1], coord[coord[,4]==2,2], coord[coord[,4]==2,3], radius=0.02, color=coord[coord[,4]==2,4], alpha=1, shininess=20)   
spheres3d(coord[coord[,4]==1,1], coord[coord[,4]==1,2], coord[coord[,4]==1,3], radius=0.04, color=coord[coord[,4]==1,4], alpha=1, shininess=20)  
rgl\_add\_axes(coord[,1], coord[,2], coord[,3], show.bbox = FALSE)  
aspect3d(1, 1, 1)



In the above figure, ToxCast compounds are represented by black spheres and DSL compounds are represented by red spheres. The quality of the read-across is dependent on the proximity of nearest neighbours in 3D space.

### Test Performance of Read-Across Using ToxCast Compounds as Control

Read-across is limited to compounds with >5 active assays and >5 active Chemotype bits. The code below performs the following steps:

1. Iteratively compare each ToxCast compound to the other compounds matching the above criteria.
2. Record 5th percentile bioactivity concentration for most similar structures (Tanimoto Coefficient > threshold).
3. Use GenRA equation to calculate read-across bioactivity concentration from ten or fewer closest analogeus above threshold.
4. Compare read-across concentration against the derived bioactivity concentration for each compound to evaluate read-across performance.
5. Repeat steps 1-4 with different fingerprints and Tanimoto Coefficient thresholds to optimize approach.

##Ready dat5 for read-across  
uniqueDat5 <- dat5[!duplicated(dat5$casn),c("chid","modl\_ga\_0.05\_uM")]  
uniqueDat5 <- as.data.frame(uniqueDat5)  
rownames(uniqueDat5) <- uniqueDat5[,"chid"]  
  
##Retain only chemicals with >5 active bits  
toxCastToxPrintFP <- toxCastToxPrintFP[rowSums(toxCastToxPrintFP) >5,]  
toxCastPubChemFP <- toxCastPubChemFP[rowSums(toxCastPubChemFP) >5,]  
toxCastECFP <- toxCastECFP[rowSums(toxCastECFP) >5,]  
  
##Retain only chemicals with >5 active assays  
#Add chids to hitRateTable for compounds that have active assays, then filter by number of active assays  
chidHitRateTable <- merge(hitRateTable, dat5[!duplicated(dat5$casn),c("casn","chid")], by="casn")  
chidHitRateTable <- chidHitRateTable[chidHitRateTable$Active.Assays.After.Filtering > 5,]  
  
#Retain only overlap between fingerprints and chidHitRateTable where the number of active assays are >5  
toxCastToxPrintFP <- toxCastToxPrintFP[rownames(toxCastToxPrintFP) %in% chidHitRateTable$chid,]  
toxCastPubChemFP <- toxCastPubChemFP[rownames(toxCastPubChemFP) %in% chidHitRateTable$chid,]  
toxCastECFP <- toxCastECFP[rownames(toxCastECFP) %in% chidHitRateTable$chid,]  
  
#Convert fingerprints to FPset for comparison  
toxCastToxPrintFPset <- as(as.matrix(toxCastToxPrintFP), "FPset")  
toxCastPubChemFPset <- as(as.matrix(toxCastPubChemFP), "FPset")  
toxCastECFPset <- as(as.matrix(toxCastECFP), "FPset")

#Define function for testing read-across  
testGenRAFunction <- function(inFPset, sValue, kValue = 10) {  
  
 referenceChemical <- vector()  
 genRABioactivity <- vector()  
 trueBioactivity <- vector()  
   
 for (i in 1:length(inFPset)) {  
 temp = NA  
 temp = fpSim(inFPset[i], inFPset[-i], sorted=TRUE, method="Tanimoto", addone=0)  
 temp = temp[temp>=sValue]  
   
 #Only retain first k analogs (default of k=10 as per previous work)  
 lengthTemp <- length(temp)  
 if (lengthTemp > kValue) { lengthTemp <- kValue }  
 temp = temp[1:lengthTemp]  
   
 genRABioactivity <- c(genRABioactivity, 10^as.numeric(sum(as.numeric(temp)\*log10(uniqueDat5[names(temp),"modl\_ga\_0.05\_uM"]))/sum(as.numeric(temp))))  
   
 referenceChemical <- c(referenceChemical, cid(inFPset)[i])  
   
 trueBioactivity <- c(trueBioactivity, uniqueDat5[referenceChemical[i],"modl\_ga\_0.05\_uM"])  
 }   
   
 toxCastReadAcrossResults <- cbind(referenceChemical, genRABioactivity, trueBioactivity)  
   
 #remove targets without analogs  
 toxCastReadAcrossResults <- toxCastReadAcrossResults[!is.na(toxCastReadAcrossResults[,"genRABioactivity"]),]  
   
 targetCount <- nrow(toxCastReadAcrossResults)  
   
 percentPredictionWithinTenFold <- as.numeric(table(abs(log10(as.numeric(toxCastReadAcrossResults[,"genRABioactivity"]))-log10(as.numeric(toxCastReadAcrossResults[,"trueBioactivity"])))<1)["TRUE"])/targetCount\*100  
   
 percentPredictionWithinHundredFold <- as.numeric(table(abs(log10(as.numeric(toxCastReadAcrossResults[,"genRABioactivity"]))-log10(as.numeric(toxCastReadAcrossResults[,"trueBioactivity"])))<2)["TRUE"])/targetCount\*100  
   
 return(cbind(targetCount, percentPredictionWithinTenFold, percentPredictionWithinHundredFold))  
   
}

####ToxPrint Chemotypes

#Explore read-across using ToxPrint Chemotype   
sValueVector <- c(0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8)  
  
toxPrintResultsTable <- matrix(ncol=3, nrow=length(sValueVector))  
rownames(toxPrintResultsTable) <- sValueVector  
colnames(toxPrintResultsTable) <- c("Targets", "%Within10Fold", "%Within100Fold")  
  
for (sValueTested in sValueVector) {  
 toxPrintResultsTable[as.character(sValueTested),] <- testGenRAFunction(toxCastToxPrintFPset, sValueTested)  
}  
  
toxPrintResultsTable <- cbind(sValueVector, toxPrintResultsTable)  
colnames(toxPrintResultsTable)[1] <- "s-value"  
  
kable(toxPrintResultsTable, row.names = FALSE)

|  |  |  |  |
| --- | --- | --- | --- |
| s-value | Targets | %Within10Fold | %Within100Fold |
| 0.1 | 4369 | 63.83612 | 89.17372 |
| 0.2 | 4369 | 63.83612 | 89.17372 |
| 0.3 | 4366 | 63.99450 | 89.16628 |
| 0.4 | 4337 | 63.84598 | 88.77104 |
| 0.5 | 4172 | 63.08725 | 87.87152 |
| 0.6 | 3648 | 63.59649 | 86.12939 |
| 0.7 | 2954 | 64.31957 | 85.84970 |
| 0.8 | 2131 | 65.69686 | 84.93665 |

####Pubchem Fingerprints

#Explore read-across using Pubchem fingerprints   
sValueVector <- c(0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8)  
  
pubchemResultsTable <- matrix(ncol=3, nrow=length(sValueVector))  
rownames(pubchemResultsTable) <- sValueVector  
colnames(pubchemResultsTable) <- c("Targets", "%Within10Fold", "%Within100Fold")  
  
for (sValueTested in sValueVector) {  
 pubchemResultsTable[as.character(sValueTested),] <- testGenRAFunction(toxCastPubChemFPset, sValueTested)  
}  
  
pubchemResultsTable <- cbind(sValueVector, pubchemResultsTable)  
colnames(pubchemResultsTable)[1] <- "s-value"  
  
kable(pubchemResultsTable, row.names = FALSE)

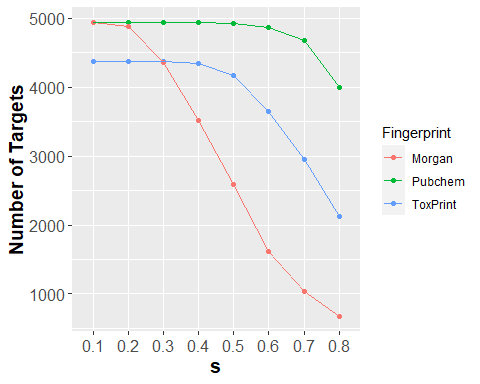
|  |  |  |  |
| --- | --- | --- | --- |
| s-value | Targets | %Within10Fold | %Within100Fold |
| 0.1 | 4945 | 63.74115 | 88.08898 |
| 0.2 | 4945 | 63.72093 | 88.08898 |
| 0.3 | 4943 | 63.70625 | 88.08416 |
| 0.4 | 4941 | 63.77252 | 88.07934 |
| 0.5 | 4927 | 63.75076 | 87.90339 |
| 0.6 | 4871 | 63.80620 | 87.66167 |
| 0.7 | 4671 | 63.81931 | 86.96211 |
| 0.8 | 4002 | 63.99300 | 85.90705 |

####Morgan Fingerprints

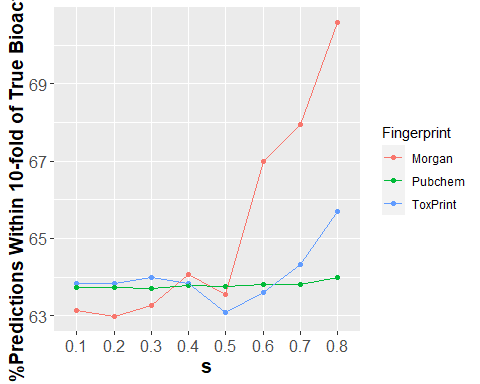
#Explore read-across using Morgan fingerprints  
sValueVector <- c(0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8)  
  
morganResultsTable <- matrix(ncol=3, nrow=length(sValueVector))  
rownames(morganResultsTable) <- sValueVector  
colnames(morganResultsTable) <- c("Targets", "%Within10Fold", "%Within100Fold")  
  
for (sValueTested in sValueVector) {  
 morganResultsTable[as.character(sValueTested),] <- testGenRAFunction(toxCastECFPset, sValueTested)  
}  
  
morganResultsTable <- cbind(sValueVector, morganResultsTable)  
colnames(morganResultsTable)[1] <- "s-value"  
  
kable(morganResultsTable, row.names = FALSE)

|  |  |  |  |
| --- | --- | --- | --- |
| s-value | Targets | %Within10Fold | %Within100Fold |
| 0.1 | 4934 | 63.13336 | 87.75841 |
| 0.2 | 4875 | 62.97436 | 87.50769 |
| 0.3 | 4354 | 63.27515 | 85.48461 |
| 0.4 | 3515 | 64.06828 | 84.92176 |
| 0.5 | 2587 | 63.54851 | 83.91960 |
| 0.6 | 1618 | 66.99629 | 85.47590 |
| 0.7 | 1033 | 67.95741 | 85.67280 |
| 0.8 | 670 | 70.59701 | 87.76119 |

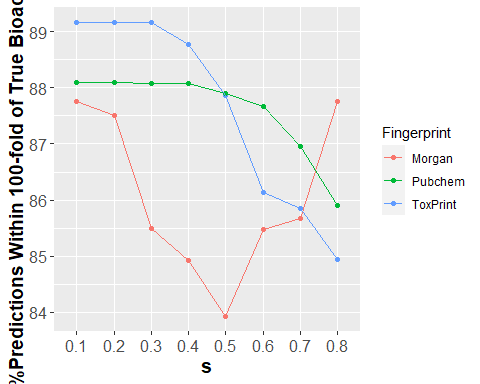
colnames(toxPrintResultsTable)[1] <- "s"  
colnames(pubchemResultsTable)[1] <- "s"  
colnames(morganResultsTable)[1] <- "s"  
  
#Targets Table  
toxPrintTargets <- melt(toxPrintResultsTable, id.vars="s", measure.vars="Targets", variable.name="Fingerprint")  
toxPrintTargets[,colnames(toxPrintTargets)=="Fingerprint"] <- "ToxPrint"  
   
pubchemTargets <- melt(pubchemResultsTable, id.vars="s", measure.vars="Targets", variable.name="Fingerprint")  
pubchemTargets[,colnames(pubchemTargets)=="Fingerprint"] <- "Pubchem"  
  
morganTargets <- melt(morganResultsTable, id.vars="s", measure.vars="Targets", variable.name="Fingerprint")  
morganTargets[,colnames(morganTargets)=="Fingerprint"] <- "Morgan"  
  
meltedTargetsTable <- rbind(toxPrintTargets, pubchemTargets, morganTargets)  
  
ggplot(meltedTargetsTable, aes(x=s, y=value, color=Fingerprint, group=Fingerprint)) +   
 geom\_point() +  
 geom\_line() +   
 ylab("Number of Targets") +  
 theme(axis.text=element\_text(size=12),  
 axis.title=element\_text(size=14,face="bold"))



#Within10 Table  
toxPrintWithin10 <- melt(toxPrintResultsTable, id.vars="s", measure.vars="%Within10Fold", variable.name="Fingerprint")  
toxPrintWithin10[,colnames(toxPrintTargets)=="Fingerprint"] <- "ToxPrint"  
   
pubchemWithin10 <- melt(pubchemResultsTable, id.vars="s", measure.vars="%Within10Fold", variable.name="Fingerprint")  
pubchemWithin10[,colnames(pubchemTargets)=="Fingerprint"] <- "Pubchem"  
  
morganWithin10 <- melt(morganResultsTable, id.vars="s", measure.vars="%Within10Fold", variable.name="Fingerprint")  
morganWithin10[,colnames(morganTargets)=="Fingerprint"] <- "Morgan"  
  
meltedWithin10 <- rbind(toxPrintWithin10, pubchemWithin10, morganWithin10)  
  
ggplot(meltedWithin10, aes(x=s, y=value, color=Fingerprint, group=Fingerprint)) +   
 geom\_point() +  
 geom\_line() +   
 ylab("%Predictions Within 10-fold of True Bioactivity") +  
 theme(axis.text=element\_text(size=12),  
 axis.title=element\_text(size=14,face="bold"))



#Within100 Table  
toxPrintWithin100 <- melt(toxPrintResultsTable, id.vars="s", measure.vars="%Within100Fold", variable.name="Fingerprint")  
toxPrintWithin100[,colnames(toxPrintTargets)=="Fingerprint"] <- "ToxPrint"  
   
pubchemWithin100 <- melt(pubchemResultsTable, id.vars="s", measure.vars="%Within100Fold", variable.name="Fingerprint")  
pubchemWithin100[,colnames(pubchemTargets)=="Fingerprint"] <- "Pubchem"  
  
morganWithin100 <- melt(morganResultsTable, id.vars="s", measure.vars="%Within100Fold", variable.name="Fingerprint")  
morganWithin100[,colnames(morganTargets)=="Fingerprint"] <- "Morgan"  
  
meltedWithin100 <- rbind(toxPrintWithin100, pubchemWithin100, morganWithin100)  
  
ggplot(meltedWithin100, aes(x=s, y=value, color=Fingerprint, group=Fingerprint)) +   
 geom\_point() +  
 geom\_line() +   
 ylab("%Predictions Within 100-fold of True Bioactivity") +  
 theme(axis.text=element\_text(size=12),  
 axis.title=element\_text(size=14,face="bold"))



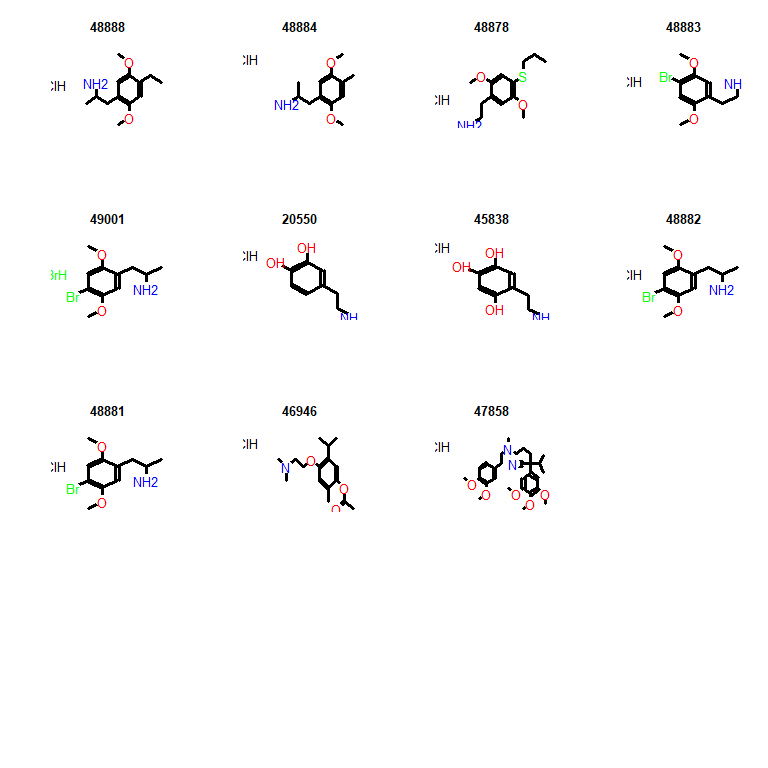
Based on the optimization results above, going forward the ToxPrint chemotype will be used with an s-value of 0.3.

#Iteratively compare each ToxCast compound against the others  
referenceChemical <- vector()  
genRA <- vector()  
similarChemicals <- vector()  
similarScores <- vector()  
bioactivityValues <- vector()  
  
  
for (i in 1:nrow(toxCastToxPrintFP)) {  
 temp = NA  
 temp = fpSim(toxCastToxPrintFPset[i], toxCastToxPrintFPset[-i], sorted=TRUE, method="Tanimoto", addone=0)  
 temp = temp[temp>=0.3]  
  
 lengthTemp <- length(temp)  
 if (lengthTemp > 10) { lengthTemp <- 10 }  
 temp = temp[1:lengthTemp]  
  
 referenceChemical <- c(referenceChemical, rownames(toxCastToxPrintFP)[i])  
   
 genRA <- c(genRA, 10^as.numeric(sum(as.numeric(temp)\*log10(uniqueDat5[names(temp),"modl\_ga\_0.05\_uM"]))/sum(as.numeric(temp))))  
   
 similarChemicals <- c(similarChemicals, paste(names(temp)[1:lengthTemp], collapse=";"))  
   
 similarScores <- c(similarScores, paste(as.numeric(temp)[1:lengthTemp], collapse=";"))  
   
 bioactivityValues <- c(bioactivityValues, paste(uniqueDat5[names(temp),"modl\_ga\_0.05\_uM"][1:lengthTemp],collapse=";"))  
   
}  
  
toxCastReadAcrossResults <- cbind(referenceChemical, similarChemicals, similarScores, bioactivityValues, genRA)

There were 4369 ToxCast compounds tested using read-across based on the Generalized Read-Across equation (TC>0.5), and a read-across bioactivity concentration could be derived for 4366 of those compounds.

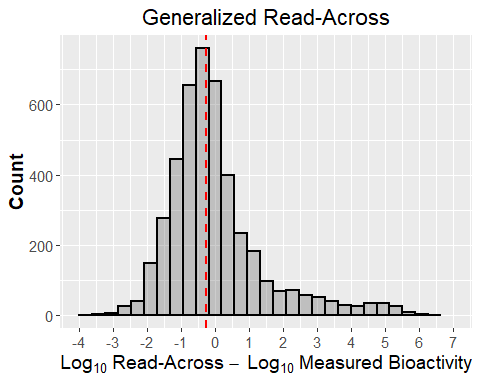
Below is an example of the structures used in read-across. The compound with the DSSTox\_GSID of 48888 is compared against the other 10 compounds.

##Plot first example  
invisible(plot(sdfsetToxCast[c(referenceChemical[1], strsplit(similarChemicals[1], ";")[[1]])]))



In the next code block, the read-across bioactivity concentration is compared against the empirical bioactivity concentration derived from the data for that compound.

##Add empirical bioactivity concentration to table  
empiricalBioactivity <- uniqueDat5[toxCastReadAcrossResults[,"referenceChemical"],"modl\_ga\_0.05\_uM"]  
toxCastReadAcrossResults <- cbind(toxCastReadAcrossResults, empiricalBioactivity)  
  
#Examine generalized read-across approach  
##Remove chemicals that failed read-across  
toxCastReadAcrossResultsOriginal <- toxCastReadAcrossResults  
toxCastReadAcrossResults <- toxCastReadAcrossResults[!is.na(toxCastReadAcrossResults[,"genRA"]),]  
  
##Plot comparison  
bioactivityComparisonGenRA <- log10(as.numeric(toxCastReadAcrossResults[,"genRA"])) - log10(as.numeric(toxCastReadAcrossResults[,"empiricalBioactivity"]))  
bioactivityComparisonGenRA = data.frame(label=1:length(bioactivityComparisonGenRA), value=as.numeric(bioactivityComparisonGenRA))  
  
ggplot(bioactivityComparisonGenRA, aes(x=as.numeric(value))) +   
 geom\_histogram(aes(y=..count..), alpha=0.3, color="black", size=1) +  
 geom\_vline(aes(xintercept=median(value)), color="red", linetype="dashed", size=1) +  
 scale\_x\_continuous(name=expression(Log["10"]~"Read-Across"~-~Log["10"]~Measured~Bioactivity), limits=c(-7, 7), breaks=-7:7) +  
 scale\_x\_continuous(name=expression(Log["10"]~"Read-Across"~-~Log["10"]~Measured~Bioactivity), limits=c(floor(min(as.numeric(bioactivityComparisonGenRA$value))), ceiling(max(as.numeric(bioactivityComparisonGenRA$value)))), breaks=seq(floor(min(as.numeric(bioactivityComparisonGenRA$value))), ceiling(max(as.numeric(bioactivityComparisonGenRA$value))), 1)) +  
 ylab("Count") +  
 ggtitle("Generalized Read-Across") +  
 theme(text = element\_text(size=14),   
 axis.title = element\_text(size=14, face="bold"),   
 plot.title=element\_text(hjust = 0.5))



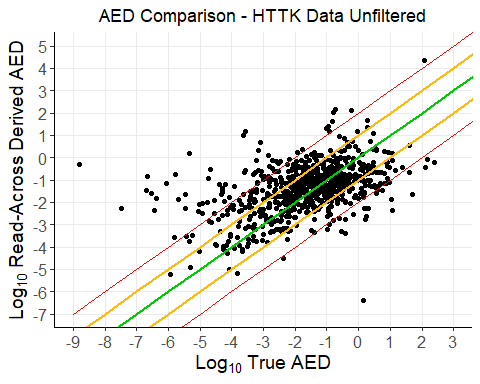
The results above show that 63.99% of bioactivity values derived from read-across are within 10-fold of the bioactivity values derived from ToxCast, and 89.17% are within 100-fold.

Note: The range of fifth percentile bioactivity concentrations in ToxCast, using this approach, is 8.808771210^{-7} to 342.1581373 on the arithmetic scale, which is a 8.5893115 magnitude difference.

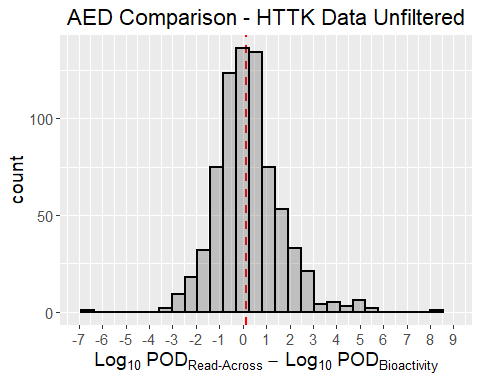
In the absence of any other information, this approach can be useful for prioritizing chemicals for assessment in the majority of cases.

### Test Combined Performance of Read-Across and *in silico* HTTK Data by Comparing *in silico*-Derived AEDs with *in vitro*-Derived AEDs

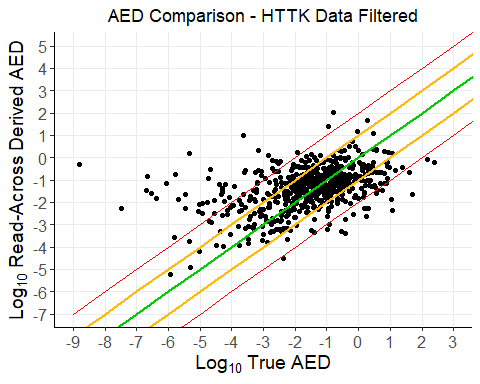
#Compare in silico AED (in siilico HTTK and read-across) with true AED (based on in vitro HTTK and ToxCast data)  
matrixChidHitRateTable <- as.matrix(chidHitRateTable)  
matrixChidHitRateTable[,"chid"] <- gsub(" ", "", matrixChidHitRateTable[,"chid"], fixed = TRUE)  
  
aedComparison <- merge(matrixChidHitRateTable[,c("casn","chid")], toxCastReadAcrossResults[,c("referenceChemical", "genRA")], by.x="chid", by.y="referenceChemical")  
aedComparison <- merge(aedComparison, uniqueDat5, by="chid")  
aedComparison <- merge(aedComparison, dataComparison[,c("CAS", "Css", "in\_silico\_derived\_Css")], by.x="casn", by.y="CAS")  
empiricalAED <- aedComparison$modl\_ga\_0.05\_uM/aedComparison$Css  
genRAAED <- as.numeric(as.character(aedComparison$genRA))/aedComparison$in\_silico\_derived\_Css  
  
aedComparison <- cbind(aedComparison, empiricalAED, genRAAED)  
  
  
#Plot difference  
straightLineLog <- -9:5  
  
ggplot() +   
 geom\_point(aes(x=log10(aedComparison$empiricalAED), y=log10(aedComparison$genRAAED))) +   
 scale\_x\_continuous(breaks=c(-9:3), labels=c(-9:3)) +   
 scale\_y\_continuous(breaks=c(-7:5), labels=c(-7:5)) +  
 coord\_cartesian(ylim=c(-7,5), xlim=c(-9,3)) +  
 ylab(expression(Log["10"]~"Read-Across"~Derived~AED)) +   
 xlab(expression(Log["10"]~True~AED)) +  
 ggtitle("AED Comparison - HTTK Data Unfiltered") +  
 geom\_line(aes(x=straightLineLog, y=straightLineLog), colour="green3", size=1) +  
 geom\_line(aes(x=straightLineLog, y=straightLineLog+1), colour="darkgoldenrod1", size=1) +  
 geom\_line(aes(x=straightLineLog, y=straightLineLog-1), colour="darkgoldenrod1", size=1) +  
 geom\_line(aes(x=straightLineLog, y=straightLineLog+2), colour="red", size=0.5) +  
 geom\_line(aes(x=straightLineLog, y=straightLineLog-2), colour="red", size=0.5) +  
 theme\_bw() +  
 theme(panel.border = element\_blank(),   
 panel.grid.major.y = element\_line(),  
 panel.grid.minor.y = element\_blank(),  
 panel.grid.minor.x = element\_blank(),  
 axis.line = element\_line(colour = "black"),  
 axis.text = element\_text(size=12),  
 axis.title = element\_text(size=14, face="bold"),  
 plot.title=element\_text(hjust = 0.5))



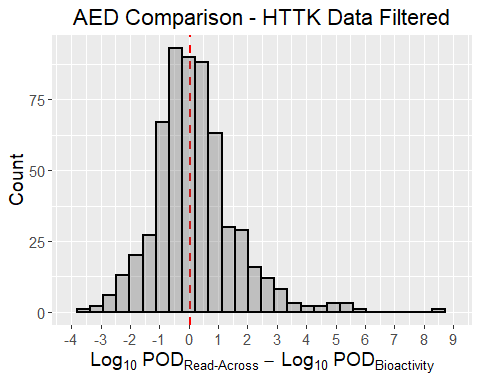
aedComparisonLogDifferenceGenRA <- log10(as.numeric(aedComparison$genRAAED)) - log10(as.numeric(aedComparison$empiricalAED))  
aedComparisonLogDifferenceGenRA = data.frame(label=1:length(aedComparisonLogDifferenceGenRA), value=as.numeric(aedComparisonLogDifferenceGenRA))  
  
ggplot(aedComparisonLogDifferenceGenRA, aes(x=as.numeric(value))) +   
 ggtitle("AED Comparison - HTTK Data Unfiltered") +  
 geom\_vline(aes(xintercept=median(value)), color="red", linetype="dashed", size=1) +  
 geom\_histogram(aes(y=..count..), alpha=0.3, color="black", size=1) +  
 scale\_x\_continuous(name=expression(Log["10"]~POD["Read-Across"]~-~Log["10"]~POD[Bioactivity]), limits=c(floor(min(as.numeric(aedComparisonLogDifferenceGenRA$value))), ceiling(max(as.numeric(aedComparisonLogDifferenceGenRA$value)))), breaks=seq(floor(min(as.numeric(aedComparisonLogDifferenceGenRA$value))), ceiling(max(as.numeric(aedComparisonLogDifferenceGenRA$value))), 1)) +  
 theme(text = element\_text(size=14), plot.title=element\_text(hjust = 0.5))



#Filter based on HTTK data that are out of domain or violate other filters  
  
aedComparisonFiltered <- aedComparison[aedComparison$casn %in% filteredDataComparison$CAS,]  
  
ggplot() +   
 geom\_point(aes(x=log10(aedComparisonFiltered$empiricalAED), y=log10(aedComparisonFiltered$genRAAED))) +   
 scale\_x\_continuous(breaks=c(-9:3), labels=c(-9:3)) +   
 scale\_y\_continuous(breaks=c(-7:5), labels=c(-7:5)) +  
 coord\_cartesian(ylim=c(-7,5), xlim=c(-9,3)) +  
 ylab(expression(Log["10"]~"Read-Across"~Derived~AED)) +   
 xlab(expression(Log["10"]~True~AED)) +  
 ggtitle("AED Comparison - HTTK Data Filtered") +  
 geom\_line(aes(x=straightLineLog, y=straightLineLog), colour="green3", size=1) +  
 geom\_line(aes(x=straightLineLog, y=straightLineLog+1), colour="darkgoldenrod1", size=1) +  
 geom\_line(aes(x=straightLineLog, y=straightLineLog-1), colour="darkgoldenrod1", size=1) +  
 geom\_line(aes(x=straightLineLog, y=straightLineLog+2), colour="red", size=0.5) +  
 geom\_line(aes(x=straightLineLog, y=straightLineLog-2), colour="red", size=0.5) +  
 theme\_bw() +  
 theme(panel.border = element\_blank(),   
 panel.grid.major.y = element\_line(),  
 panel.grid.minor.y = element\_blank(),  
 panel.grid.minor.x = element\_blank(),  
 axis.line = element\_line(colour = "black"),  
 axis.text = element\_text(size=12),  
 axis.title = element\_text(size=14, face="bold"),  
 plot.title=element\_text(hjust = 0.5))



aedComparisonLogDifferenceGenRAFiltered <- log10(as.numeric(aedComparisonFiltered$genRAAED)) - log10(as.numeric(aedComparisonFiltered$empiricalAED))  
aedComparisonLogDifferenceGenRAFiltered = data.frame(label=1:length(aedComparisonLogDifferenceGenRAFiltered), value=as.numeric(aedComparisonLogDifferenceGenRAFiltered))  
  
ggplot(aedComparisonLogDifferenceGenRAFiltered, aes(x=as.numeric(value))) +   
 ggtitle("AED Comparison - HTTK Data Filtered") +  
 geom\_vline(aes(xintercept=median(value)), color="red", linetype="dashed", size=1) +  
 geom\_histogram(aes(y=..count..), alpha=0.3, color="black", size=1) +  
 scale\_x\_continuous(name=expression(Log["10"]~POD["Read-Across"]~-~Log["10"]~POD[Bioactivity]), limits=c(floor(min(as.numeric(aedComparisonLogDifferenceGenRAFiltered$value))), ceiling(max(as.numeric(aedComparisonLogDifferenceGenRAFiltered$value)))), breaks=seq(floor(min(as.numeric(aedComparisonLogDifferenceGenRAFiltered$value))), ceiling(max(as.numeric(aedComparisonLogDifferenceGenRAFiltered$value))), 1)) +  
 ylab("Count")+  
 theme(text = element\_text(size=14), plot.title=element\_text(hjust = 0.5))



The results above show that of 580 compounds where read-across could be applied, and HTTK data passed filtering, 79.4827586206897% of predicted AEDs were within 10-fold of the true AED and 91.2068965517241% were witin 100-fold.

Without filtering data, based on HTTK criteria, there were 733 compounds where read-across could be applied, and 76.5347885402456% of predicted AEDs were within 10-fold of the true AED and 90.450204638472% were witin 100-fold.

Note: The range of AEDs in ToxCast, obtained using 5th percentile bioactivity and in vitro HTTK data, is 1.568490510^{-9} to 246.0605717 on the arithmetic scale, which is a 11.1955601 magnitude difference.

## Apply Read-Across Approach to DSL Compounds Where Applicable

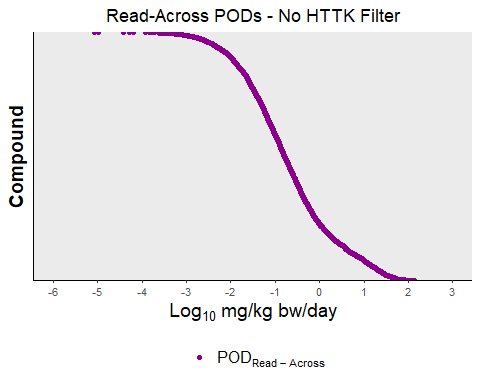
For DSL compounds not covered by ToxCast, each compound is iteratively compared against ToxCast compounds, by ToxPrint Chemotype, to idenitfy structural analogs. The bioactivity concentration is calculated using the Generalized Read-Across equation and the first ten analogs above the Tanimoto Coefficient threshold of 0.3.

##Retain only chemicals with >5 active Chemotypebits  
originalDslFP <- dslFP  
dslFP <- dslFP[rowSums(dslFP) > 5,]  
  
##Eliminate compounds present in ToxCast  
dslFP <- dslFP[!rownames(dslFP) %in% dat5$casn,]  
  
#Convert fingerprints to FPset for comparison  
dslFPset <- as(as.matrix(dslFP), "FPset")  
  
#Iteratively compare each DSL compound against the others  
referenceChemical <- vector()  
genRA <- vector()  
similarChemicals <- vector()  
similarScores <- vector()  
bioactivityValues <- vector()  
  
for (i in 1:nrow(dslFP)) {  
 temp = NA  
 temp = fpSim(dslFPset[i], toxCastToxPrintFPset, sorted=TRUE, method="Tanimoto", addone=0)  
 temp = temp[temp>=0.3]  
  
 lengthTemp <- length(temp)  
 if (lengthTemp > 10) { lengthTemp <- 10 }  
 temp = temp[1:lengthTemp]  
   
 referenceChemical <- c(referenceChemical, rownames(dslFP)[i])  
   
 genRA <- c(genRA, 10^as.numeric(sum(as.numeric(temp)\*log10(uniqueDat5[names(temp),"modl\_ga\_0.05\_uM"]))/sum(as.numeric(temp))))  
   
 similarChemicals <- c(similarChemicals, paste(names(temp)[1:lengthTemp], collapse=";"))  
   
 similarScores <- c(similarScores, paste(as.numeric(temp)[1:lengthTemp], collapse=";"))  
   
 bioactivityValues <- c(bioactivityValues, paste(uniqueDat5[names(temp),"modl\_ga\_0.05\_uM"][1:lengthTemp],collapse=";"))  
   
}  
  
dslReadAcrossResults <- cbind(referenceChemical, similarChemicals, similarScores, bioactivityValues, genRA)

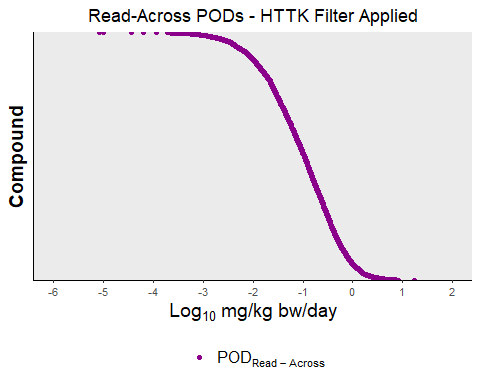
There were 9954 DSL compounds, absent in ToxCast, that were tested using read-across. Of these, a biaoctivity concentration could be derived for 9937 compounds. After applying HTTK filters, 4093 compounds remained.

Below the *in silico* derived Css values from earlier are used in the *in vitro* to *in vivo* extrapolation to provide AEDs. AEDs are limited to chemicals that pass the HTTK filter.

##Merge read-across and httk data  
dslReadAcrossDat5 <- merge(httkResults, dslReadAcrossResults[,c("referenceChemical","genRA")], by.x="CASRN", by.y="referenceChemical")  
  
#Merge returns factors so it is important to convert back to numeric, but first as a string  
dslReadAcrossDat5$Css <- as.numeric(as.character(dslReadAcrossDat5$Css))  
dslReadAcrossDat5$genRA <- as.numeric(as.character(dslReadAcrossDat5$genRA))  
  
dslReadAcrossDat5 <- as.data.table(dslReadAcrossDat5[,c("CASRN", "HC.Name", "SMILES", "Css", "genRA")])  
  
#Defaulting Css value to 0.1 to be conservative.   
dslReadAcrossDat5$Css[dslReadAcrossDat5$Css<0.1]= 0.1  
  
Log.POD.ReadAcross <- log10(as.numeric(dslReadAcrossDat5$genRA)/as.numeric(dslReadAcrossDat5$Css))  
  
AEDsReadAcross <- cbind(dslReadAcrossDat5, Log.POD.ReadAcross)  
colnames(AEDsReadAcross)[2] <- "Provided Compound Name"  
  
##Remove NAs  
AEDsReadAcross <- AEDsReadAcross[!is.na(AEDsReadAcross$Log.POD.ReadAcross),]  
  
##Plot read-across AEDs (HTTK filter applied)  
ggplot() +  
 geom\_point(data = AEDsReadAcross, aes(x= AEDsReadAcross$"Log.POD.ReadAcross", y = AEDsReadAcross$"CASRN", color="Log.POD.ReadAcross"), size=1.5) +  
 ggtitle("Read-Across PODs - No HTTK Filter") +  
 ylab("Compound") +   
 xlab(expression(Log["10"]~"mg/kg"~"bw/day")) +  
 scale\_x\_continuous(breaks=seq(floor(min(c(AEDsReadAcross$"Log.POD.ReadAcross"))), ceiling(max(c(AEDsReadAcross$"Log.POD.ReadAcross"))) ,1)) +  
 scale\_y\_discrete(limits= AEDsReadAcross$"CASRN"[rev(order(AEDsReadAcross$"Log.POD.ReadAcross"))]) +  
 coord\_cartesian(xlim = c(floor(min(c(AEDsReadAcross$"Log.POD.ReadAcross"))),ceiling(max(c(AEDsReadAcross$"Log.POD.ReadAcross"))))) +  
 theme\_bw() +  
 theme(panel.border = element\_blank(),   
 panel.grid.major.y = element\_line(),  
 panel.grid.minor.y = element\_blank(),  
 panel.grid.minor.x = element\_blank(),  
 axis.line = element\_line(colour = "black"),  
 plot.title = element\_text(hjust = 0.5),  
 axis.text.y=element\_blank(),  
 axis.ticks.y=element\_blank(),  
 axis.text = element\_text(size=8),  
 axis.title = element\_text(size=14, face="bold")) +  
 scale\_colour\_manual(  
 breaks=c("Log.POD.ReadAcross"),  
 values=c("darkmagenta"),  
 labels=c(expression(POD[Read~-~Across]))) +  
 theme(legend.position="bottom", legend.title=element\_blank(), legend.text=element\_text(size=12))



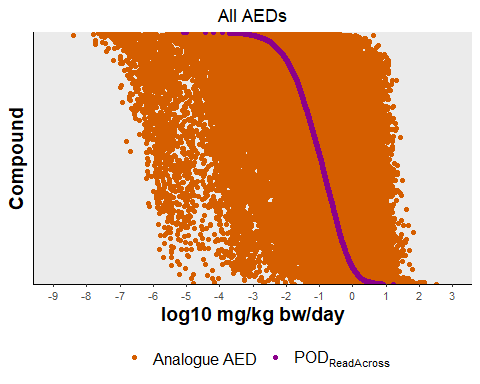
##Plot read-across AEDs (HTTK filter applied)  
AEDsReadAcrossFiltered <- AEDsReadAcross[AEDsReadAcross$CASRN %in% filteredDSLHTTK,]  
   
ggplot() +  
 geom\_point(data = AEDsReadAcrossFiltered, aes(x= AEDsReadAcrossFiltered$"Log.POD.ReadAcross", y = AEDsReadAcrossFiltered$"CASRN", color="Log.POD.ReadAcross"), size=1.5) +  
 ggtitle("Read-Across PODs - HTTK Filter Applied") +  
 ylab("Compound") +   
 xlab(expression(Log["10"]~"mg/kg"~"bw/day")) +  
 scale\_x\_continuous(breaks=seq(floor(min(c(AEDsReadAcrossFiltered$"Log.POD.ReadAcross"))), ceiling(max(c(AEDsReadAcrossFiltered$"Log.POD.ReadAcross"))) ,1)) +  
 scale\_y\_discrete(limits= AEDsReadAcrossFiltered$"CASRN"[rev(order(AEDsReadAcrossFiltered$"Log.POD.ReadAcross"))]) +  
 coord\_cartesian(xlim = c(floor(min(c(AEDsReadAcrossFiltered$"Log.POD.ReadAcross"))),ceiling(max(c(AEDsReadAcrossFiltered$"Log.POD.ReadAcross"))))) +  
 theme\_bw() +  
 theme(panel.border = element\_blank(),   
 panel.grid.major.y = element\_line(),  
 panel.grid.minor.y = element\_blank(),  
 panel.grid.minor.x = element\_blank(),  
 axis.line = element\_line(colour = "black"),  
 plot.title = element\_text(hjust = 0.5),  
 axis.text.y=element\_blank(),  
 axis.ticks.y=element\_blank(),  
 axis.text = element\_text(size=8),  
 axis.title = element\_text(size=14, face="bold")) +  
 scale\_colour\_manual(  
 breaks=c("Log.POD.ReadAcross"),  
 values=c("darkmagenta"),  
 labels=c(expression(POD[Read~-~Across]))) +  
 theme(legend.position="bottom", legend.title=element\_blank(), legend.text=element\_text(size=12))



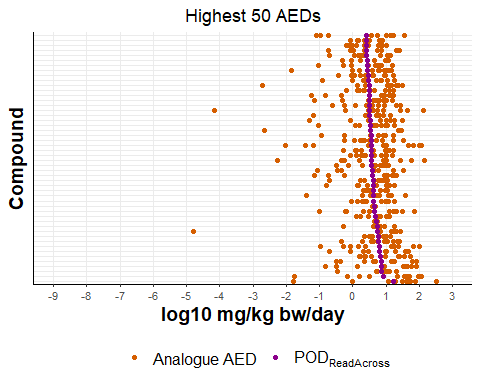
In the next code blocks, the plots include the AEDs used in the read-across. The AEDs use the bioactivity concentration of the analogue but the HTTK parameters for the target.

dslReadAcrossAllAnalogues <- dslReadAcrossResults  
  
#Remove NAs  
dslReadAcrossAllAnalogues <- dslReadAcrossAllAnalogues[!is.na(dslReadAcrossAllAnalogues[,"genRA"]),]  
  
##Create melted table for plotting  
#First - All values  
CASRN <- vector()  
Category <- vector()  
value <- vector()  
  
for (i in 1:nrow(dslReadAcrossAllAnalogues)) {  
 tempActivities <- strsplit(dslReadAcrossAllAnalogues[i,"bioactivityValues"],";")[[1]]  
   
 for (currentActivityValue in tempActivities) {  
 CASRN <- c(CASRN, dslReadAcrossAllAnalogues[i, "referenceChemical"])  
 Category <- c(Category, "ReadAcrossAnalogueAED")  
 value <- c(value, currentActivityValue)  
 }  
   
}  
  
meltReadAcrossData <- cbind(CASRN, Category, value)  
  
#Second - Bioactivity  
CASRN <- vector()  
Category <- vector()  
value <- vector()  
  
for (i in 1:nrow(dslReadAcrossAllAnalogues)) {  
 CASRN <- c(CASRN, dslReadAcrossAllAnalogues[i, "referenceChemical"])  
 Category <- c(Category, "Log.POD.ReadAcross")  
 value <- c(value, dslReadAcrossAllAnalogues[i, "genRA"])  
}  
  
meltBioactivityReadAcross <- cbind(CASRN, Category, value)  
  
meltReadAcrossTable <- rbind(meltReadAcrossData, meltBioactivityReadAcross)  
  
meltReadAcrossTable[,"value"] <- as.numeric(meltReadAcrossTable[,"value"])  
meltBioactivityReadAcross$"value" <- as.numeric(meltBioactivityReadAcross[,"value"])  
meltReadAcrossTable <- as.data.frame(meltReadAcrossTable)

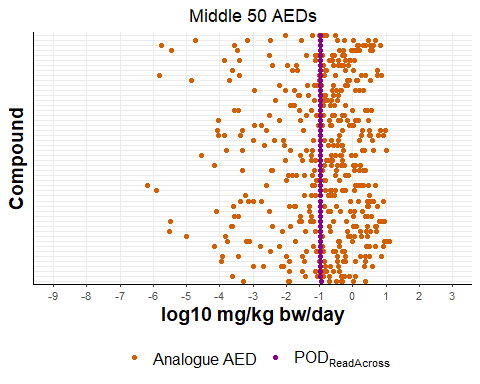
meltReadAcrossTable$value <- as.character(meltReadAcrossTable$value)  
meltReadAcrossTable$value <- as.numeric(meltReadAcrossTable$value)  
  
##Merge read-across and httk data  
meltReadAcrossTableMerge <- merge(httkResults[,c("CASRN", "Css")], meltReadAcrossTable[,c("CASRN","Category", "value")], by.x="CASRN", by.y="CASRN")  
  
#Merge returns factors so it is important to convert back to numeric, but first as a string  
meltReadAcrossTableMerge$Css <- as.numeric(as.character(meltReadAcrossTableMerge$Css))  
meltReadAcrossTableMerge$value <- as.numeric(as.character(meltReadAcrossTableMerge$value))  
  
#Defaulting Css value to 0.1 to be conservative.   
meltReadAcrossTableMerge$Css[meltReadAcrossTableMerge$Css<0.1]= 0.1  
  
meltReadAcrossTableMerge$value <- log10(as.numeric(meltReadAcrossTableMerge$value)/as.numeric(meltReadAcrossTableMerge$Css))  
  
meltReadAcrossTableMerge <- as.data.frame(meltReadAcrossTableMerge)  
meltReadAcrossTableMerge$value <- as.numeric(as.character(meltReadAcrossTableMerge$value))  
meltReadAcrossTableMerge <- meltReadAcrossTableMerge[order(meltReadAcrossTableMerge$Category),]  
  
#Reverse order so POD\_Read-Across is on top in plot  
meltReadAcrossTableMerge <- meltReadAcrossTableMerge[nrow(meltReadAcrossTableMerge):1,]  
  
#Limit plot to chemicals passing HTTK filter  
meltReadAcrossTableMerge <- meltReadAcrossTableMerge[meltReadAcrossTableMerge$CASRN %in% filteredDSLHTTK,]  
  
#Define plot function  
plotAEDsReadAcrossFunction <- function(numToDisplay=NA, titleLab) {  
   
 if(is.na(numToDisplay)) { numToDisplay <- 1:length(AEDsReadAcross$CASRN)}  
   
 ggplot() +  
 geom\_point(data = meltReadAcrossTableMerge, aes(x= meltReadAcrossTableMerge$"value", y = meltReadAcrossTableMerge$"CASRN", color=meltReadAcrossTableMerge$"Category")) + ylab("Compound") +   
 xlab("log10 mg/kg bw/day") +  
 ggtitle(titleLab) +  
 scale\_x\_continuous(breaks=seq( floor(min(c(meltReadAcrossTableMerge$"value"))), ceiling(max(c(meltReadAcrossTableMerge$"value", meltReadAcrossTableMerge$"value"))) ,1)) +  
 scale\_y\_discrete(limits= unique(AEDsReadAcrossFiltered$"CASRN"[rev(order(AEDsReadAcrossFiltered$"Log.POD.ReadAcross"))][numToDisplay])) +  
 coord\_cartesian(xlim = c(floor(min(c(meltReadAcrossTableMerge$"value", meltReadAcrossTableMerge$"value"))),ceiling(max(c(meltReadAcrossTableMerge$"value", meltReadAcrossTableMerge$"value"))))) +  
 theme\_bw() +  
 theme(panel.border = element\_blank(),   
 panel.grid.major.y = element\_line(),  
 panel.grid.minor.y = element\_blank(),  
 panel.grid.minor.x = element\_blank(),  
 axis.line = element\_line(colour = "black"),  
 axis.text.y=element\_blank(),  
 axis.ticks.y=element\_blank(),  
 axis.text = element\_text(size=8),  
 axis.title = element\_text(size=14, face="bold"),  
 plot.title = element\_text(hjust = 0.5)) +  
 scale\_colour\_manual(  
 breaks=c("ReadAcrossAnalogueAED", "Log.POD.ReadAcross"),  
 values=c("#D55E00", "darkmagenta"),  
 labels=c("Analogue AED", expression(POD[ReadAcross]))) +  
 theme(legend.position="bottom", legend.title=element\_blank(), legend.text=element\_text(size=12))   
   
}  
  
#Display AEDs based on different positions  
plotAEDsReadAcrossFunction(titleLab = "All AEDs")



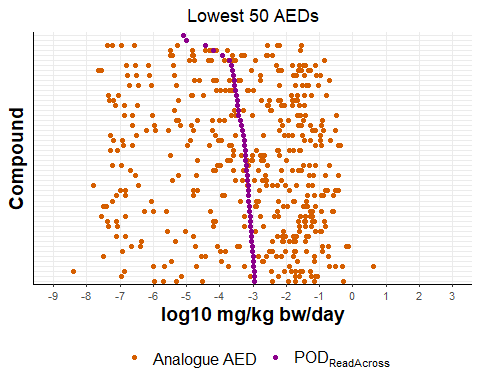
plotAEDsReadAcrossFunction(numToDisplay = 1:50, titleLab = "Highest 50 AEDs")



plotAEDsReadAcrossFunction(numToDisplay = (length(AEDsReadAcrossFiltered$CASRN)/2-24):(length(AEDsReadAcrossFiltered$CASRN)/2+25), titleLab = "Middle 50 AEDs")



plotAEDsReadAcrossFunction(numToDisplay = (length(AEDsReadAcrossFiltered$CASRN)-49):length(AEDsReadAcrossFiltered$CASRN), titleLab = "Lowest 50 AEDs")



matrixAEDsReadAcross <- as.matrix(AEDsReadAcross)  
  
rownames(matrixAEDsReadAcross) <- matrixAEDsReadAcross[,"CASRN"]  
overlappingCASRN <- matrixAEDsReadAcross[,"CASRN"][matrixAEDsReadAcross[,"CASRN"] %in% summaryTable[,"CASRN"]]  
  
summaryTable[overlappingCASRN, "AED(mg/kg-bw/day)"] <- 10^as.numeric(matrixAEDsReadAcross[overlappingCASRN, "Log.POD.ReadAcross"])  
summaryTable[overlappingCASRN, "BioactivityConcentration(MicroMolar)"] <- matrixAEDsReadAcross[overlappingCASRN, "genRA"]  
summaryTable[overlappingCASRN, "BioactivitySource"] <- "Read-Across"

## Combine DSL PODs

##Merge all bioactivity data, layered with Log.AED first, then Read-Across, then Bioactivity AED  
summaryResults <- melt.data.table(AEDsReadAcross, id.vars="CASRN", measure.vars="Log.POD.ReadAcross", variable.name="Group")  
  
summaryResults <- rbind(summaryResults, melt.data.table(AEDs[!duplicated(AEDs$CASRN),], id.vars="CASRN", measure.vars=c("Log.POD.Bioactivity"), variable.name="Group"))

In total, 12828 AEDs could be derived for DSL compounds. There were 2974 AEDs from ToxCast data and 9854 from read-across.

After applying filters, there were 5801 AEDs for DSL compounds. There were 1708 AEDs from ToxCast and 4093 from read-across.

For the filtered data, the minimum POD was -7.5887972 and the maximum POD was 2.3391495.

## Increasing the Confidence of Workflow Through Comparisons with *in vivo* Points of Departure

The next code block extracts traditional PODs (NOAELs, LOAELs, and BMDLs) from the ToxValDB. Data was from the ToxValDB version updated on September 17, 2020. Data were filtered by response type, units, exposure route, risk assessment class, and study type. Methods are described below. Note: the full database is required to run the code block.

podTable <- read.xlsx("toxval\_pod\_summary\_human health\_2020-09-17.xlsx")  
  
#Limit to DSL compounds  
podTable <- podTable[podTable$casrn %in% dsl,]  
  
#Only retain certain columns  
podTable <- podTable[,c("casrn", "toxval\_numeric", "toxval\_units", "toxval\_type", "exposure\_route", "risk\_assessment\_class", "study\_type", "source", "species\_common")]  
  
#Only keep units with mass per kg  
podTable <- podTable[podTable$toxval\_units %in% c("mg/kg-day", "mg/kg"),]  
  
#Only keep the most common response types associated with NOAEL, BMDL, and LOAEL. Synonyms converted accordingly.  
podTable[grepl("BMDL", podTable$toxval\_type),"toxval\_type"] <- "BMDL"  
podTable[podTable$toxval\_type %in% c("NOEC", "NOAEC", "NOEL", "NEL", "HNEL"), "toxval\_type"] <- "NOAEL"  
podTable[podTable$toxval\_type %in% c("LOEC", "LOAEC", "LOEL", "LEL"), "toxval\_type"] <- "LOAEL"  
podTable <- podTable[podTable$toxval\_type %in% c("NOAEL", "BMDL", "LOAEL"),]  
  
#Limit exposure route to oral or gavage  
podTable <- podTable[podTable$exposure\_route %in% c("oral", "oral/gavage"),]  
  
#Filter by risk\_assessment\_class  
podTable <- podTable[podTable$risk\_assessment\_class %in% c("developmental", "reproductive", "reproductive developmental", "subchronic", "chronic", "repeat dose"),]  
  
#Filter by study\_type  
podTable <- podTable[podTable$study\_type %in% c("developmental", "Developmental", "developmental reproductive", "repeat-dose", "repeat dose", "Reproduction", "reproductive", "Reproductive", "reproductive developmental", "subchronic", "chronic", "repeat dose"),]  
  
#Convert values to log10 scale  
podTable$toxval\_numeric <- log10(podTable$toxval\_numeric)  
colnames(podTable)[colnames(podTable) == "toxval\_numeric"] <- "LogTraditionalPOD"  
  
#Store information  
write.xlsx(x=podTable, file="ToxValDB\_POD\_Table.xlsx")

Below the PODs derived from the workflow are compared against the Traditional PODs from ToxValDB. The lowest Traditional POD from NOAELs, LOAELs, and BMD are used for comparison where possible.

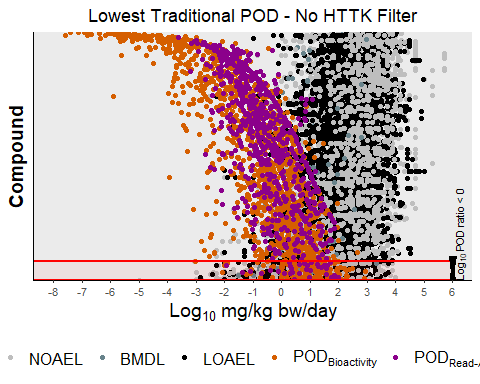
#Load file  
podTable <- read.xlsx(xlsxFile="ToxValDB\_POD\_Table.xlsx", colNames=TRUE)  
  
#Merge podTable with summaryResults for comparison  
podTable <- podTable[podTable$casrn %in% summaryResults$CASRN,]   
colnames(podTable)[colnames(podTable) %in% "LogTraditionalPOD"] <- "value"  
colnames(podTable)[colnames(podTable) %in% "toxval\_type"] <- "Group"  
colnames(podTable)[colnames(podTable) %in% "casrn"] <- "CASRN"  
toxValDB <- as.data.frame(rbind(summaryResults[summaryResults$CASRN %in% unique(podTable$CASRN),], podTable[,c("CASRN", "Group","value")]))  
  
#Compare results  
casrnForComparison <- unique(toxValDB$CASRN)  
bioactivityForComparison <- rep(NA, length(casrnForComparison))  
loaelForComparison <- rep(NA, length(casrnForComparison))  
noaelForComparison <- rep(NA, length(casrnForComparison))  
bmdlForComparison <- rep(NA, length(casrnForComparison))  
lowestTraditionalForComparison <- rep(NA, length(casrnForComparison))  
  
for (i in 1:length(casrnForComparison)) {  
 bioactivityForComparison[i] <- toxValDB[toxValDB$CASRN %in% casrnForComparison[i] & (toxValDB$Group == "Log.POD.Bioactivity" | toxValDB$Group =="Log.POD.ReadAcross"),"value"]  
   
 if (length(toxValDB[toxValDB$CASRN == casrnForComparison[i] & toxValDB$Group == "LOAEL", "value"]) > 0) {  
 loaelForComparison[i] <- min(toxValDB[toxValDB$CASRN == casrnForComparison[i] & toxValDB$Group == "LOAEL", "value"])  
 }  
 if (length(toxValDB[toxValDB$CASRN == casrnForComparison[i] & toxValDB$Group == "BMDL", "value"]) > 0) {  
 bmdlForComparison[i] <- min(toxValDB[toxValDB$CASRN == casrnForComparison[i] & toxValDB$Group == "BMDL", "value"])  
 }  
 if (length(toxValDB[toxValDB$CASRN == casrnForComparison[i] & toxValDB$Group == "NOAEL", "value"]) > 0) {  
 noaelForComparison[i] <- min(toxValDB[toxValDB$CASRN == casrnForComparison[i] & toxValDB$Group == "NOAEL", "value"])  
 }  
 lowestTraditionalForComparison[i] <- min(toxValDB[toxValDB$CASRN == casrnForComparison[i] & (toxValDB$Group == "NOAEL" | toxValDB$Group == "BMDL" | toxValDB$Group == "NOAEL"), "value"])  
}  
  
  
#Establish differences and plot orders  
podDifferenceTVDBLowest <- lowestTraditionalForComparison - bioactivityForComparison  
plotOrderToxValLowest <- casrnForComparison[which(podDifferenceTVDBLowest != "Inf")]  
podDifferenceTVDBLowest <- podDifferenceTVDBLowest[podDifferenceTVDBLowest != "Inf"]  
plotOrderToxValLowest <- plotOrderToxValLowest[order(podDifferenceTVDBLowest)]  
podDifferenceTVDBLowest <- podDifferenceTVDBLowest[order(podDifferenceTVDBLowest)]  
  
podDifferenceTVDBNOAEL <- noaelForComparison - bioactivityForComparison  
plotOrderToxValNOAEL <- casrnForComparison[which(!is.na(podDifferenceTVDBNOAEL))]  
podDifferenceTVDBNOAEL <- podDifferenceTVDBNOAEL[!is.na(podDifferenceTVDBNOAEL)]  
plotOrderToxValNOAEL <- plotOrderToxValNOAEL[order(podDifferenceTVDBNOAEL)]  
podDifferenceTVDBNOAEL <- podDifferenceTVDBNOAEL[order(podDifferenceTVDBNOAEL)]  
  
podDifferenceTVDBLOAEL <- loaelForComparison - bioactivityForComparison  
plotOrderToxValLOAEL <- casrnForComparison[which(!is.na(podDifferenceTVDBLOAEL))]  
podDifferenceTVDBLOAEL <- podDifferenceTVDBLOAEL[!is.na(podDifferenceTVDBLOAEL)]  
plotOrderToxValLOAEL <- plotOrderToxValLOAEL[order(podDifferenceTVDBLOAEL)]  
podDifferenceTVDBLOAEL <- podDifferenceTVDBLOAEL[order(podDifferenceTVDBLOAEL)]  
  
podDifferenceTVDBBMDL <- bmdlForComparison - bioactivityForComparison  
plotOrderToxValBMDL <- casrnForComparison[which(!is.na(podDifferenceTVDBBMDL))]  
podDifferenceTVDBBMDL <- podDifferenceTVDBBMDL[!is.na(podDifferenceTVDBBMDL)]  
plotOrderToxValBMDL <- plotOrderToxValBMDL[order(podDifferenceTVDBBMDL)]  
podDifferenceTVDBBMDL <- podDifferenceTVDBBMDL[order(podDifferenceTVDBBMDL)]

podDifferenceTable <- rbind(  
 c(  
 length(podDifferenceTVDBLowest),  
 table(podDifferenceTVDBLowest>0)["TRUE"],  
 table(podDifferenceTVDBLowest>0)["FALSE"],  
 min(podDifferenceTVDBLowest),  
 median(podDifferenceTVDBLowest),  
 max(podDifferenceTVDBLowest)  
 ),  
 c(  
 length(podDifferenceTVDBNOAEL),  
 table(podDifferenceTVDBNOAEL>0)["TRUE"],  
 table(podDifferenceTVDBNOAEL>0)["FALSE"],  
 min(podDifferenceTVDBNOAEL),  
 median(podDifferenceTVDBNOAEL),  
 max(podDifferenceTVDBNOAEL)  
 ),  
 c(  
 length(podDifferenceTVDBBMDL),  
 table(podDifferenceTVDBBMDL>0)["TRUE"],  
 table(podDifferenceTVDBBMDL>0)["FALSE"],  
 min(podDifferenceTVDBBMDL),  
 median(podDifferenceTVDBBMDL),  
 max(podDifferenceTVDBBMDL)  
 ),  
 c(  
 length(podDifferenceTVDBLOAEL),  
 table(podDifferenceTVDBLOAEL>0)["TRUE"],  
 table(podDifferenceTVDBLOAEL>0)["FALSE"],  
 min(podDifferenceTVDBLOAEL),  
 median(podDifferenceTVDBLOAEL),  
 max(podDifferenceTVDBLOAEL)  
 ),  
 c(  
 length(podDifferenceTVDBLowest[plotOrderToxValLowest %in% filteredDSLHTTK]),  
 table(podDifferenceTVDBLowest[plotOrderToxValLowest %in% filteredDSLHTTK]>0)["TRUE"],  
 table(podDifferenceTVDBLowest[plotOrderToxValLowest %in% filteredDSLHTTK]>0)["FALSE"],  
 min(podDifferenceTVDBLowest[plotOrderToxValLowest %in% filteredDSLHTTK]),  
 median(podDifferenceTVDBLowest[plotOrderToxValLowest %in% filteredDSLHTTK]),  
 max(podDifferenceTVDBLowest[plotOrderToxValLowest %in% filteredDSLHTTK])  
 ),  
 c(  
 length(podDifferenceTVDBNOAEL[plotOrderToxValNOAEL %in% filteredDSLHTTK]),  
 table(podDifferenceTVDBNOAEL[plotOrderToxValNOAEL %in% filteredDSLHTTK]>0)["TRUE"],  
 table(podDifferenceTVDBNOAEL[plotOrderToxValNOAEL %in% filteredDSLHTTK]>0)["FALSE"],  
 min(podDifferenceTVDBNOAEL[plotOrderToxValNOAEL %in% filteredDSLHTTK]),  
 median(podDifferenceTVDBNOAEL[plotOrderToxValNOAEL %in% filteredDSLHTTK]),  
 max(podDifferenceTVDBNOAEL[plotOrderToxValNOAEL %in% filteredDSLHTTK])  
 ),  
 c(  
 length(podDifferenceTVDBBMDL[plotOrderToxValBMDL %in% filteredDSLHTTK]),  
 table(podDifferenceTVDBBMDL[plotOrderToxValBMDL %in% filteredDSLHTTK]>0)["TRUE"],  
 table(podDifferenceTVDBBMDL[plotOrderToxValBMDL %in% filteredDSLHTTK]>0)["FALSE"],  
 min(podDifferenceTVDBBMDL[plotOrderToxValBMDL %in% filteredDSLHTTK]),  
 median(podDifferenceTVDBBMDL[plotOrderToxValBMDL %in% filteredDSLHTTK]),  
 max(podDifferenceTVDBBMDL[plotOrderToxValBMDL %in% filteredDSLHTTK])  
 ),  
 c(  
 length(podDifferenceTVDBLOAEL[plotOrderToxValLOAEL %in% filteredDSLHTTK]),  
 table(podDifferenceTVDBLOAEL[plotOrderToxValLOAEL %in% filteredDSLHTTK]>0)["TRUE"],  
 table(podDifferenceTVDBLOAEL[plotOrderToxValLOAEL %in% filteredDSLHTTK]>0)["FALSE"],  
 min(podDifferenceTVDBLOAEL[plotOrderToxValLOAEL %in% filteredDSLHTTK]),  
 median(podDifferenceTVDBLOAEL[plotOrderToxValLOAEL %in% filteredDSLHTTK]),  
 max(podDifferenceTVDBLOAEL[plotOrderToxValLOAEL %in% filteredDSLHTTK])  
 )  
)  
  
colnames(podDifferenceTable) <- c("POD Comparisons", "Number Protective", "Number Non-Proective", "Minimum Log10Traditional-Log10Bioactivity", "Median Log10Traditional-Log10Bioactivity", "Maximum Log10Traditional-Log10Bioactivity")  
  
rownames(podDifferenceTable) <- c("Lowest POD", "NOAEL", "BMDL", "LOAEL", "Lowest POD Filtered", "NOAEL Filtered", "BMDL Filtered", "LOAEL Filtered")  
  
kable(podDifferenceTable)

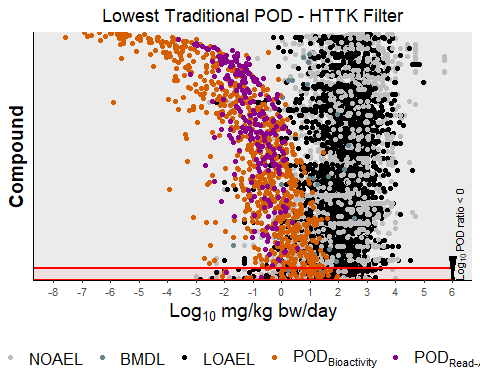
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | POD Comparisons | Number Protective | Number Non-Proective | Minimum Log10Traditional-Log10Bioactivity | Median Log10Traditional-Log10Bioactivity | Maximum Log10Traditional-Log10Bioactivity |
| Lowest POD | 2248 | 2077 | 171 | -6.892886 | 2.236523 | 9.909845 |
| NOAEL | 2230 | 2068 | 162 | -6.892886 | 2.250510 | 9.909845 |
| BMDL | 96 | 76 | 20 | -2.106526 | 1.422560 | 7.875401 |
| LOAEL | 1218 | 1149 | 69 | -3.779193 | 2.496555 | 11.151416 |
| Lowest POD Filtered | 1042 | 992 | 50 | -4.333721 | 2.382890 | 9.909845 |
| NOAEL Filtered | 1037 | 990 | 47 | -4.333721 | 2.393010 | 9.909845 |
| BMDL Filtered | 42 | 37 | 5 | -1.331193 | 1.448308 | 5.538780 |
| LOAEL Filtered | 610 | 591 | 19 | -2.076111 | 2.611749 | 8.995609 |

The median Log10POD difference was 2.2365231 for unfiltered data and 2.3828905 for filtered data.

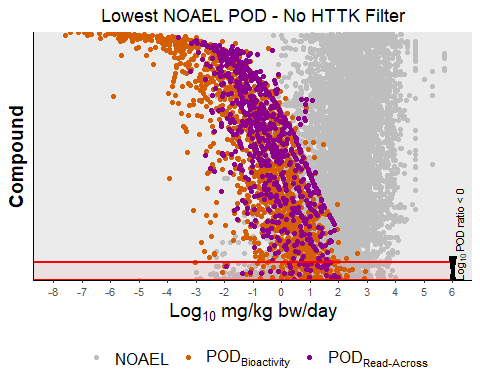
toxValDB <- toxValDB[nrow(toxValDB):1,]  
  
#Define plot function  
plotPODDifferenceFunction <- function(inData, titleLab,   
 plotBreaks=c("NOAEL", "BMDL", "LOAEL", "Log.POD.Bioactivity", "Log.POD.ReadAcross"),  
 plotLabels=c("NOAEL" = "NOAEL", "BMDL" = "BMDL", "LOAEL" = "LOAEL", "Log.POD.Bioactivity" = expression(POD[Bioactivity]), "Log.POD.ReadAcross" = expression(POD["Read-Across"])),  
 plotColours=c("NOAEL" = "gray", "BMDL" = "lightblue4", "LOAEL" = "black", "Log.POD.Bioactivity" = "#D55E00", "Log.POD.ReadAcross" = "darkmagenta"),   
 inPlotOrder=plotOrderToxValLowest,  
 inPODDifference=podDifferenceTVDBLowest) {  
  
 ggplot() +  
 geom\_point(data = inData, aes(x=value, y=CASRN, color=Group)) +   
 ggtitle(titleLab) +  
 ylab("Compound") +   
 xlab(expression(Log["10"]~"mg/kg"~"bw/day")) +  
 scale\_x\_continuous(breaks=seq(floor(min(inData$value)), ceiling(max(inData$value)), 1)) +  
 scale\_y\_discrete(limits=inPlotOrder) +  
 coord\_cartesian(xlim = c(floor(min(inData$value)), ceiling(max(inData$value)))) +  
 theme\_bw() +  
 theme(panel.border = element\_blank(),   
 panel.grid.major.y = element\_line(),  
 panel.grid.minor.y = element\_blank(),  
 panel.grid.minor.x = element\_blank(),  
 plot.title = element\_text(hjust = 0.5),   
 axis.line = element\_line(colour = "black"),  
 axis.text.y=element\_blank(),  
 axis.ticks.y=element\_blank(),  
 axis.text = element\_text(size=8),  
 axis.title = element\_text(size=14, face="bold")) +  
 scale\_colour\_manual(  
 breaks=plotBreaks,  
 values=plotColours,  
 labels=plotLabels) +  
 theme(legend.position="bottom", legend.title=element\_blank(), legend.text=element\_text(size=12)) +  
 annotate("rect",   
 xmin = floor(min(inData$value)-1),   
 xmax = ceiling(max(inData$value)),   
 ymin = 0,   
 ymax = as.numeric(table(inPODDifference < 0)["TRUE"]),   
 fill = "red",   
 alpha = 0.05,   
 size = 1,   
 color = "red") +  
 geom\_segment(data = inData, aes(x = ceiling(max(inData$value[!is.na(inData$value)])),  
 y= as.numeric(table(inPODDifference < 0)["TRUE"])+0.6,  
 xend= ceiling(max(inData$value[!is.na(inData$value)])),  
 yend=0),  
 color="black",  
 arrow=arrow(type="closed",  
 angle=8,  
 ends="both")) +  
 annotate(geom="text", x=ceiling(max(inData$value[!is.na(inData$value)])) + 0.25,  
 y = 0.1,  
 label = expression(Log["10"]~POD~ratio~"<"~0),  
 color="black",  
 size=3,  
 fontface="bold",  
 angle=90,  
 vjust=0.5,  
 hjust=0)   
}  
  
#Plot Minimum Traditional POD  
plotPODDifferenceFunction(inData=toxValDB, titleLab="Lowest Traditional POD - No HTTK Filter")



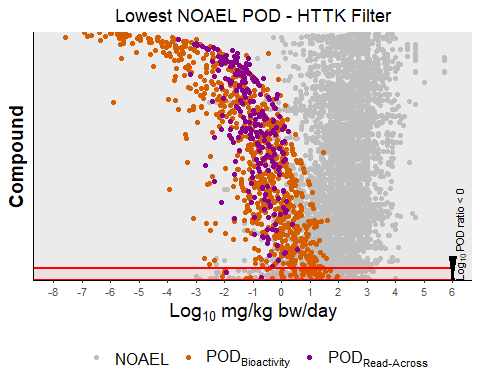
plotPODDifferenceFunction(inData=toxValDB, titleLab="Lowest Traditional POD - HTTK Filter",   
 inPlotOrder =plotOrderToxValLowest[plotOrderToxValLowest %in% filteredDSLHTTK],   
 inPODDifference=podDifferenceTVDBLowest[which(plotOrderToxValLowest %in% filteredDSLHTTK)])



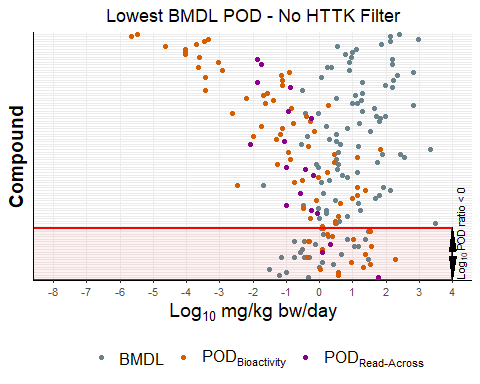
#Plot Minimum NOAEL  
plotPODDifferenceFunction(inData=toxValDB[toxValDB$Group %in% c("NOAEL", "Log.POD.Bioactivity", "Log.POD.ReadAcross"),], titleLab="Lowest NOAEL POD - No HTTK Filter", plotBreaks=c("NOAEL", "Log.POD.Bioactivity", "Log.POD.ReadAcross"), plotLabels=c("NOAEL" = "NOAEL", "Log.POD.Bioactivity" = expression(POD[Bioactivity]), "Log.POD.ReadAcross" = expression(POD["Read-Across"])),   
plotColours=c("NOAEL" = "gray", "Log.POD.Bioactivity" = "#D55E00", "Log.POD.ReadAcross" = "darkmagenta"),   
inPlotOrder=plotOrderToxValNOAEL,   
inPODDifference=podDifferenceTVDBNOAEL)



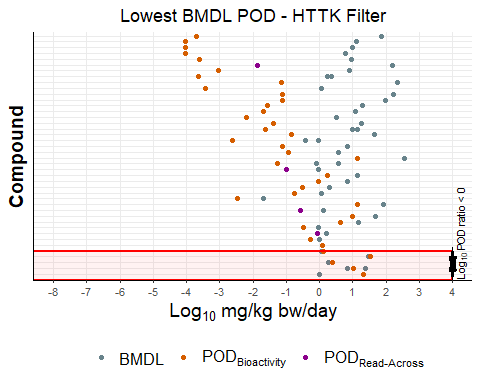
plotPODDifferenceFunction(inData=toxValDB[toxValDB$Group %in% c("NOAEL", "Log.POD.Bioactivity", "Log.POD.ReadAcross"),], titleLab="Lowest NOAEL POD - HTTK Filter", plotBreaks=c("NOAEL", "Log.POD.Bioactivity", "Log.POD.ReadAcross"), plotLabels=c("NOAEL" = "NOAEL", "Log.POD.Bioactivity" = expression(POD[Bioactivity]), "Log.POD.ReadAcross" = expression(POD["Read-Across"])),   
plotColours=c("NOAEL" = "gray", "Log.POD.Bioactivity" = "#D55E00", "Log.POD.ReadAcross" = "darkmagenta"),   
inPlotOrder=plotOrderToxValNOAEL[plotOrderToxValNOAEL %in% filteredDSLHTTK],   
inPODDifference=podDifferenceTVDBNOAEL[which(plotOrderToxValNOAEL %in% filteredDSLHTTK)])



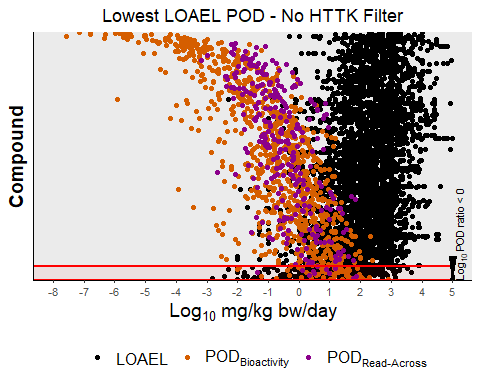
#Plot Minimum BMDL  
plotPODDifferenceFunction(inData=toxValDB[toxValDB$Group %in% c("BMDL", "Log.POD.Bioactivity", "Log.POD.ReadAcross"),], titleLab="Lowest BMDL POD - No HTTK Filter", plotBreaks=c("BMDL", "Log.POD.Bioactivity", "Log.POD.ReadAcross"), plotLabels=c("BMDL" = "BMDL", "Log.POD.Bioactivity" = expression(POD[Bioactivity]), "Log.POD.ReadAcross" = expression(POD["Read-Across"])),   
plotColours=c("BMDL" = "lightblue4", "Log.POD.Bioactivity" = "#D55E00", "Log.POD.ReadAcross" = "darkmagenta"),   
inPlotOrder=plotOrderToxValBMDL,   
inPODDifference=podDifferenceTVDBBMDL)



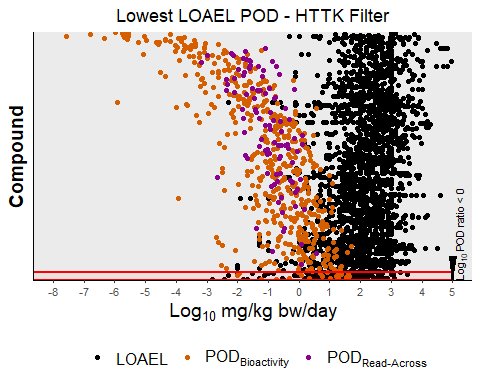
plotPODDifferenceFunction(inData=toxValDB[toxValDB$Group %in% c("BMDL", "Log.POD.Bioactivity", "Log.POD.ReadAcross"),], titleLab="Lowest BMDL POD - HTTK Filter", plotBreaks=c("BMDL", "Log.POD.Bioactivity", "Log.POD.ReadAcross"), plotLabels=c("BMDL" = "BMDL", "Log.POD.Bioactivity" = expression(POD[Bioactivity]), "Log.POD.ReadAcross" = expression(POD["Read-Across"])),   
plotColours=c("BMDL" = "lightblue4", "Log.POD.Bioactivity" = "#D55E00", "Log.POD.ReadAcross" = "darkmagenta"),   
inPlotOrder=plotOrderToxValBMDL[which(plotOrderToxValBMDL %in% filteredDSLHTTK)],   
inPODDifference=podDifferenceTVDBBMDL[which(plotOrderToxValBMDL %in% filteredDSLHTTK)])



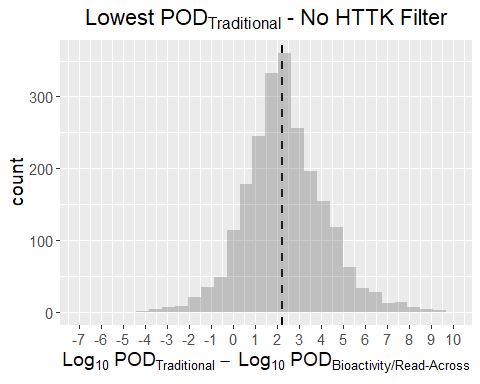
#Plot Minimum LOAEL  
plotPODDifferenceFunction(inData=toxValDB[toxValDB$Group %in% c("LOAEL", "Log.POD.Bioactivity", "Log.POD.ReadAcross"),], titleLab="Lowest LOAEL POD - No HTTK Filter", plotBreaks=c("LOAEL", "Log.POD.Bioactivity", "Log.POD.ReadAcross"), plotLabels=c("LOAEL" = "LOAEL", "Log.POD.Bioactivity" = expression(POD[Bioactivity]), "Log.POD.ReadAcross" = expression(POD["Read-Across"])),   
plotColours=c("LOAEL" = "black", "Log.POD.Bioactivity" = "#D55E00", "Log.POD.ReadAcross" = "darkmagenta"),   
inPlotOrder=plotOrderToxValLOAEL,   
inPODDifference=podDifferenceTVDBLOAEL)



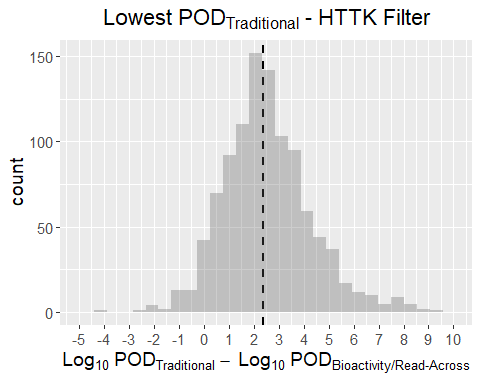
plotPODDifferenceFunction(inData=toxValDB[toxValDB$Group %in% c("LOAEL", "Log.POD.Bioactivity", "Log.POD.ReadAcross"),], titleLab="Lowest LOAEL POD - HTTK Filter", plotBreaks=c("LOAEL", "Log.POD.Bioactivity", "Log.POD.ReadAcross"), plotLabels=c("LOAEL" = "LOAEL", "Log.POD.Bioactivity" = expression(POD[Bioactivity]), "Log.POD.ReadAcross" = expression(POD["Read-Across"])),   
plotColours=c("LOAEL" = "black", "Log.POD.Bioactivity" = "#D55E00", "Log.POD.ReadAcross" = "darkmagenta"),   
inPlotOrder=plotOrderToxValLOAEL[plotOrderToxValLOAEL %in% filteredDSLHTTK],   
inPODDifference=podDifferenceTVDBLOAEL[which(plotOrderToxValLOAEL %in% filteredDSLHTTK)])



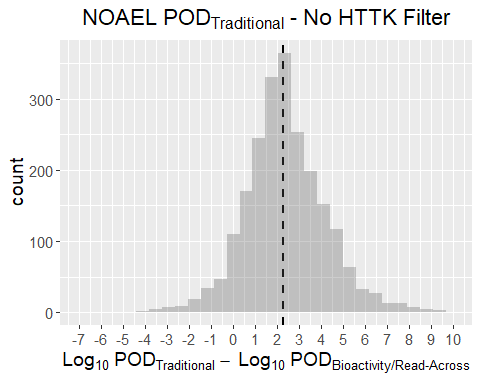
plotPODDifferenceFunction2 <- function(inPODDifference, titleLab) {  
  
podDifferenceTVDF = data.frame(label=1:length(inPODDifference), value=as.numeric(inPODDifference))  
   
 suppressWarnings(  
 print(  
 ggplot(podDifferenceTVDF, aes(x=as.numeric(value))) +  
 geom\_vline(aes(xintercept=median(value)),  
 color="black", linetype="dashed", size=1) +  
 geom\_histogram(aes(y=..count..), alpha=0.3) +  
 ggtitle(titleLab) +  
 scale\_x\_continuous(name=expression(Log["10"]~POD[Traditional]~-~Log["10"]~POD["Bioactivity/Read-Across"]), limits=c(floor(min(inPODDifference)), ceiling(max(inPODDifference[inPODDifference != Inf]))), breaks=seq(floor(min(inPODDifference)), ceiling(max(inPODDifference[inPODDifference != Inf])), 1)) +  
 theme(text = element\_text(size=14), plot.title = element\_text(hjust = 0.5))  
 )  
 )   
}  
  
#Lowest POD  
plotPODDifferenceFunction2(podDifferenceTVDBLowest, expression(Lowest~POD[Traditional]~"-"~No~HTTK~Filter))



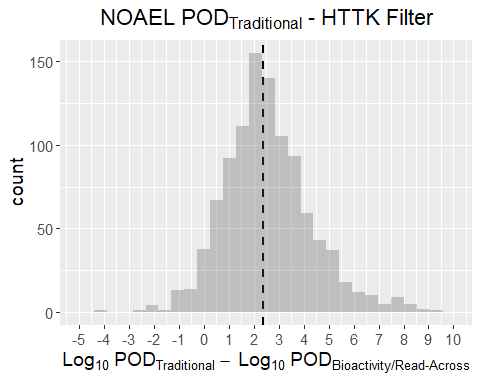
plotPODDifferenceFunction2(podDifferenceTVDBLowest[plotOrderToxValLowest %in% filteredDSLHTTK], expression(Lowest~POD[Traditional]~"-"~HTTK~Filter))



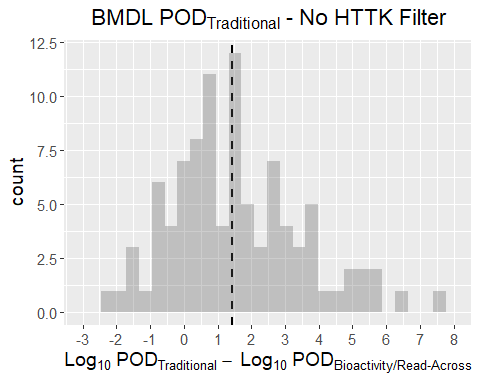
#NOAEL  
plotPODDifferenceFunction2(podDifferenceTVDBNOAEL, expression(NOAEL~POD[Traditional]~"-"~No~HTTK~Filter))



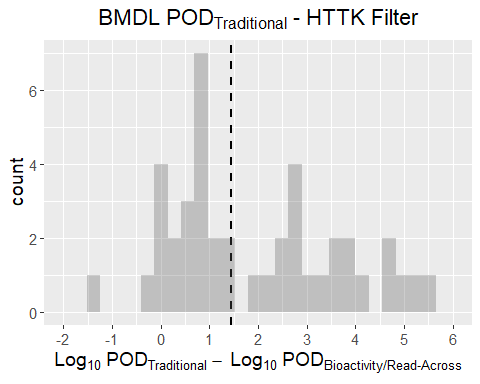
plotPODDifferenceFunction2(podDifferenceTVDBNOAEL[plotOrderToxValNOAEL %in% filteredDSLHTTK], expression(NOAEL~POD[Traditional]~"-"~HTTK~Filter))



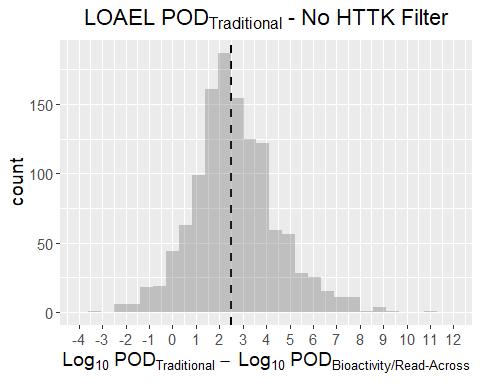
#BMDL  
plotPODDifferenceFunction2(podDifferenceTVDBBMDL, expression(BMDL~POD[Traditional]~"-"~No~HTTK~Filter))



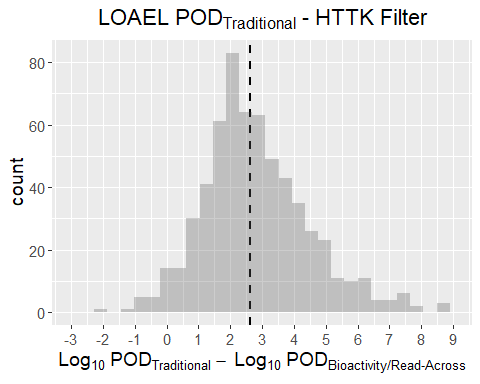
plotPODDifferenceFunction2(podDifferenceTVDBBMDL[plotOrderToxValBMDL %in% filteredDSLHTTK], expression(BMDL~POD[Traditional]~"-"~HTTK~Filter))



#LOAEL  
plotPODDifferenceFunction2(podDifferenceTVDBLOAEL, expression(LOAEL~POD[Traditional]~"-"~No~HTTK~Filter))

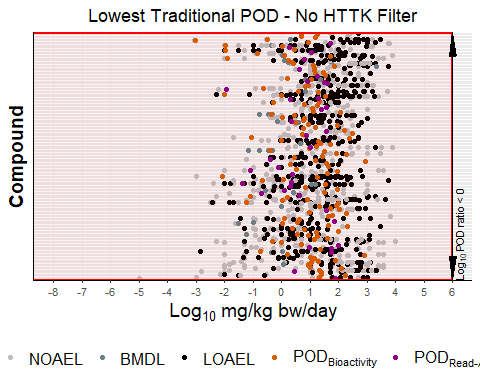


plotPODDifferenceFunction2(podDifferenceTVDBLOAEL[plotOrderToxValLOAEL %in% filteredDSLHTTK], expression(LOAEL~POD[Traditional]~"-"~HTTK~Filter))



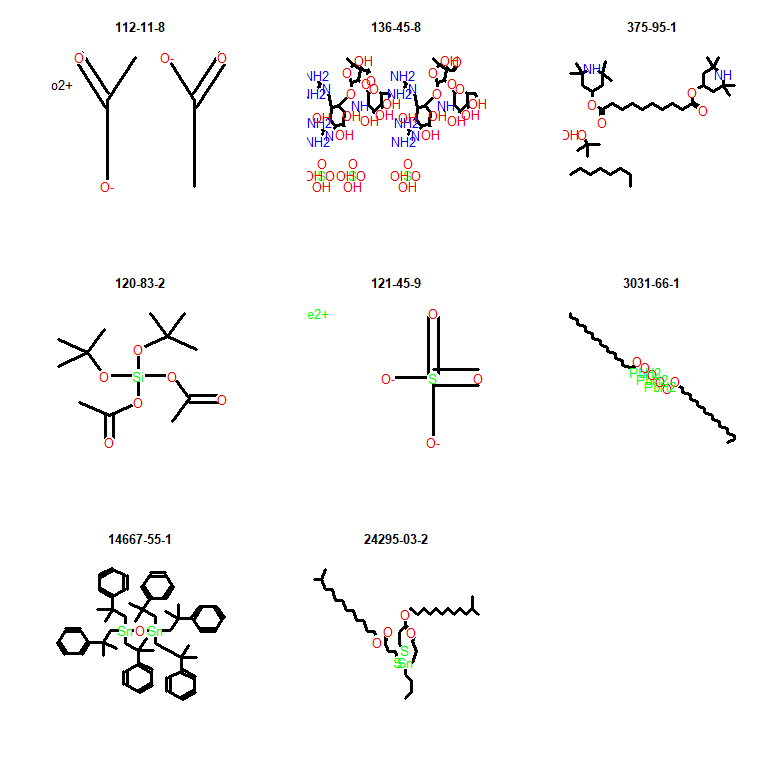
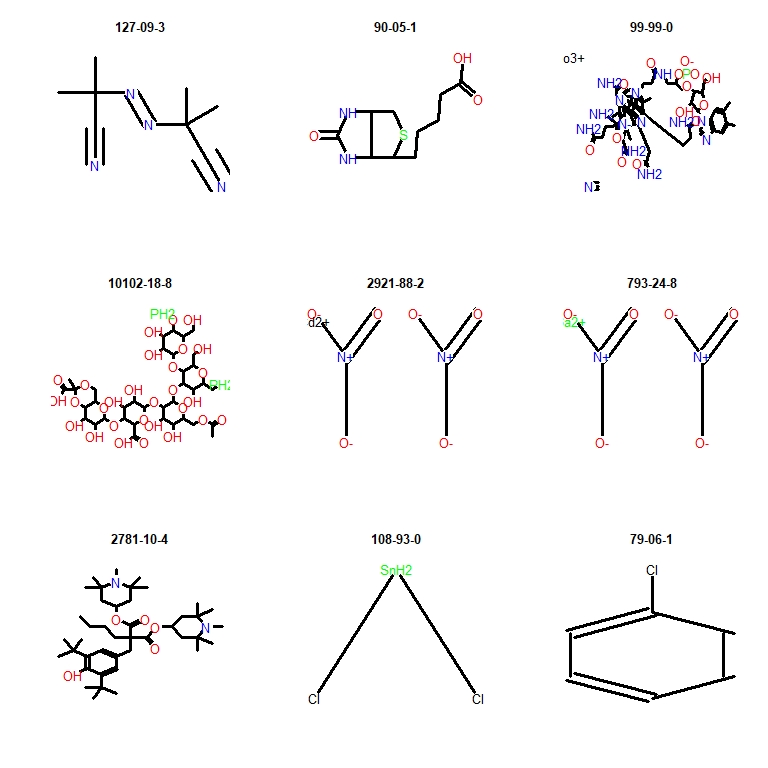
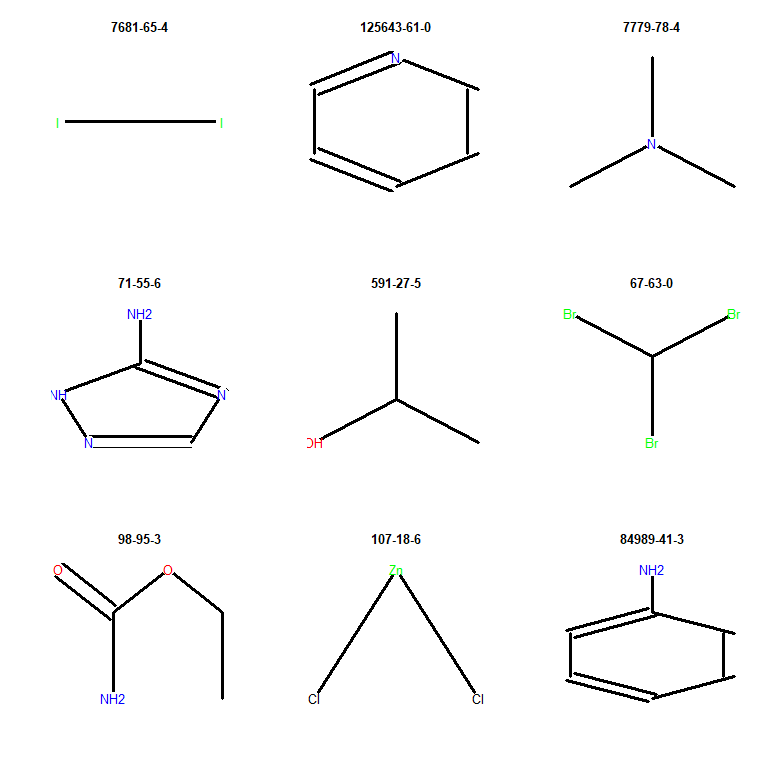
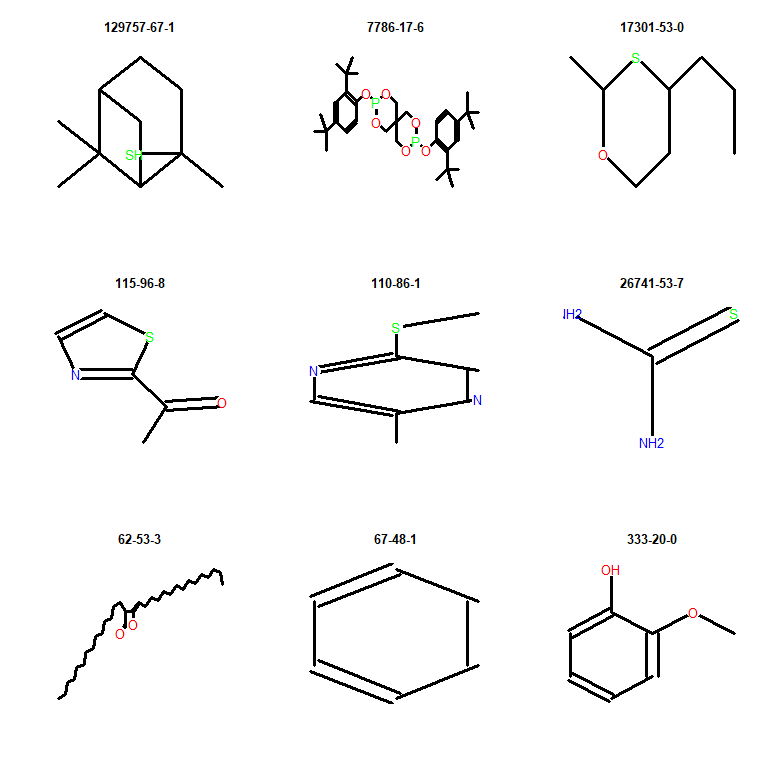
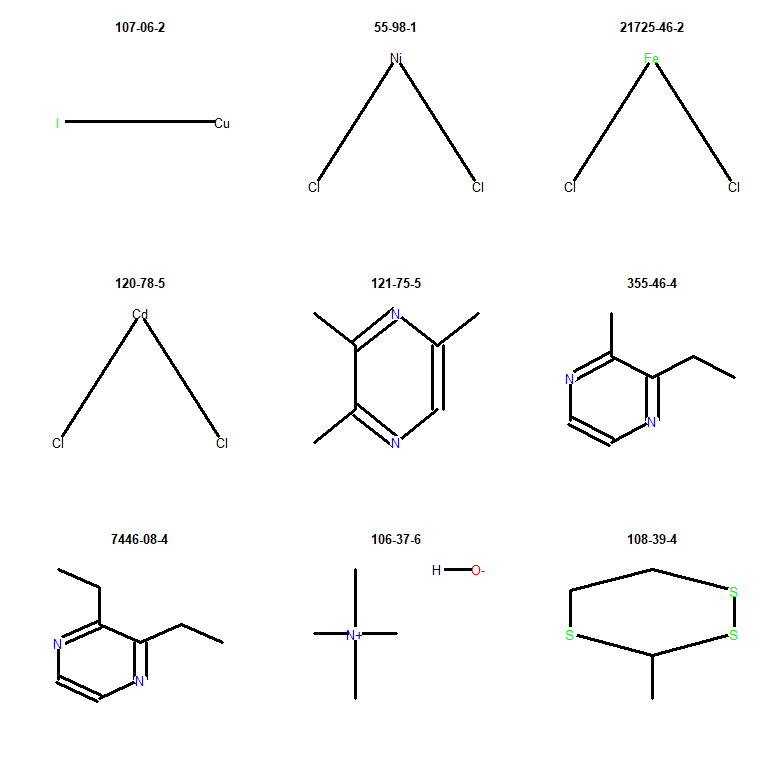
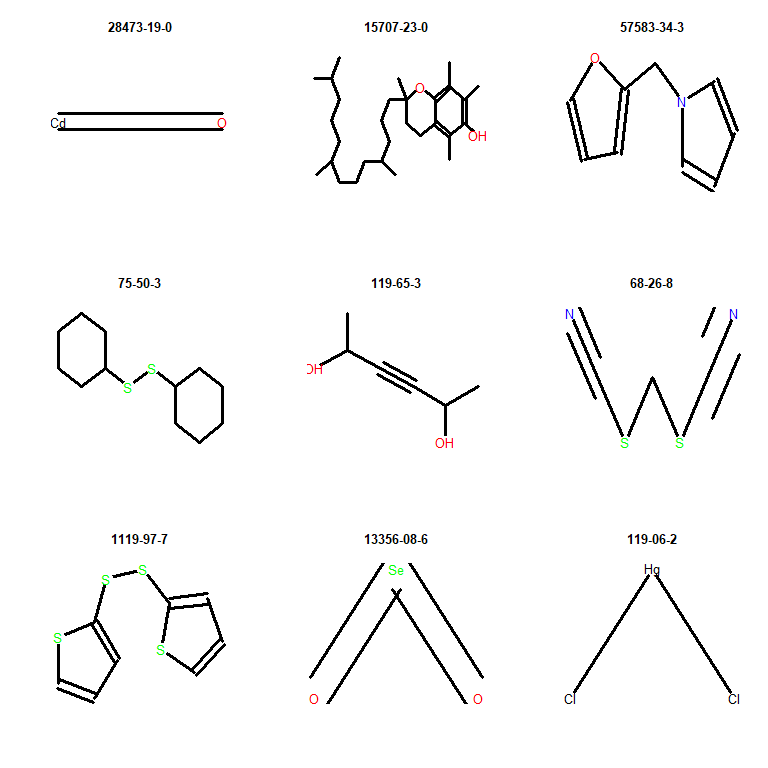
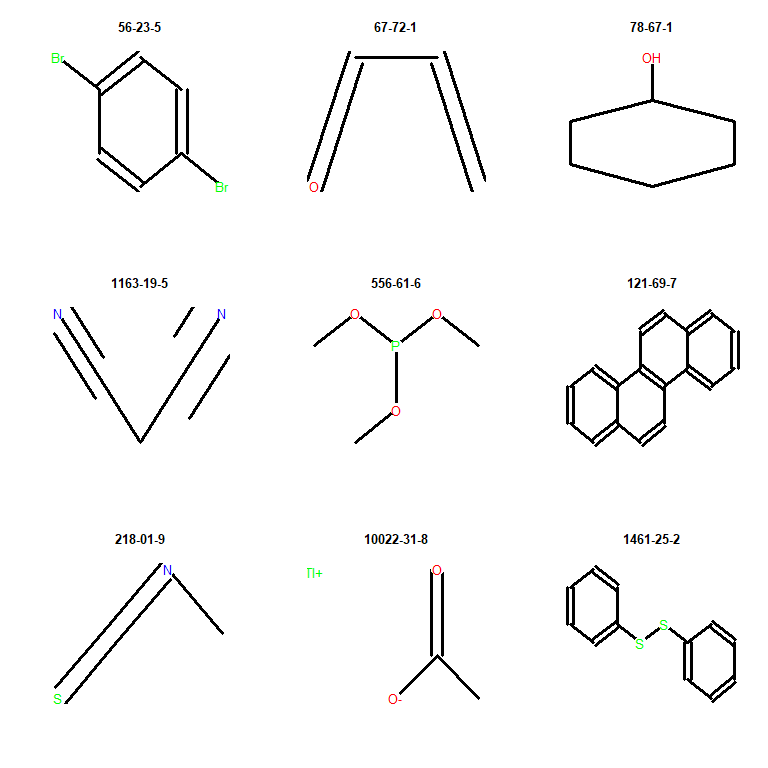
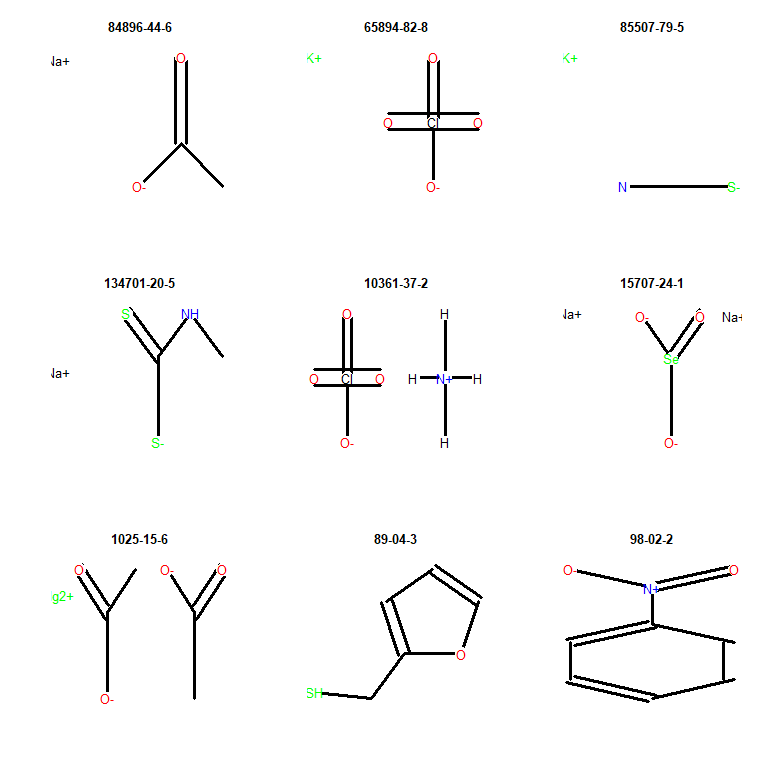
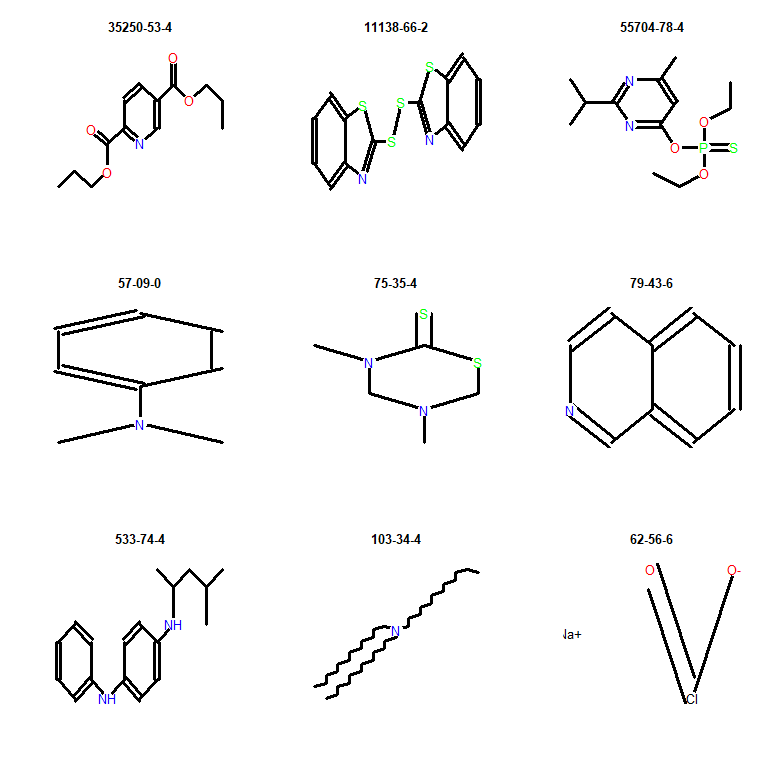
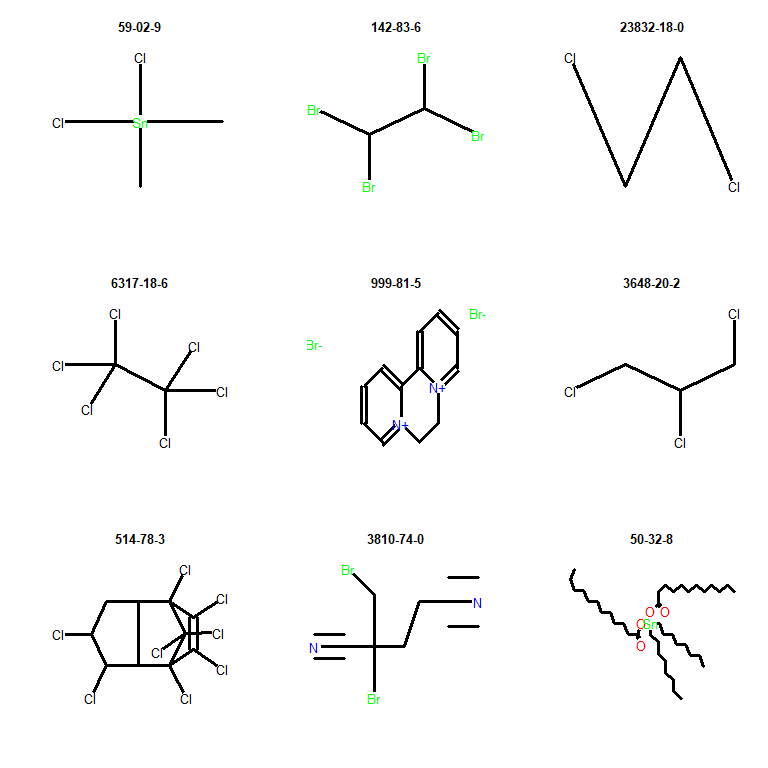
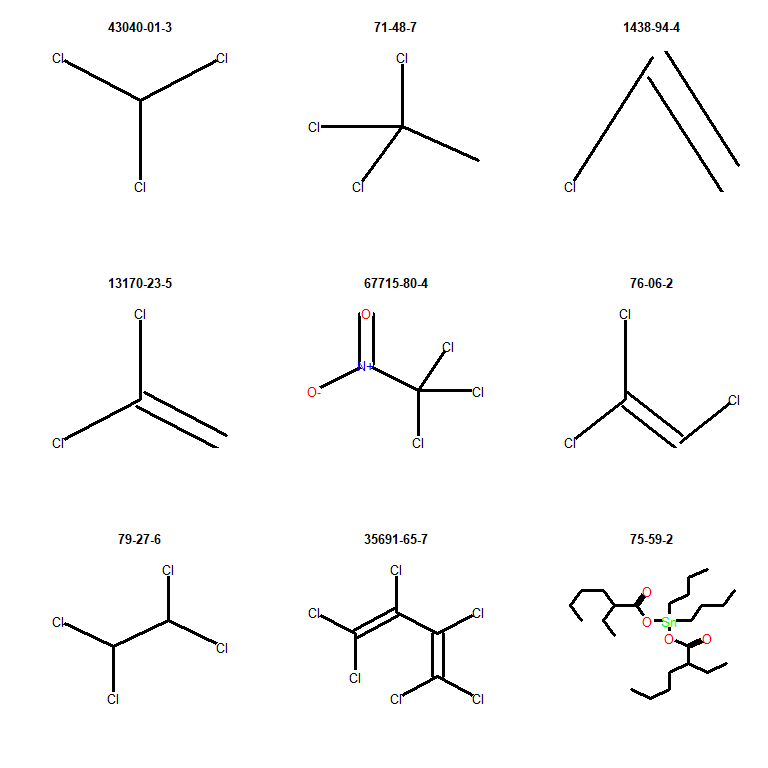
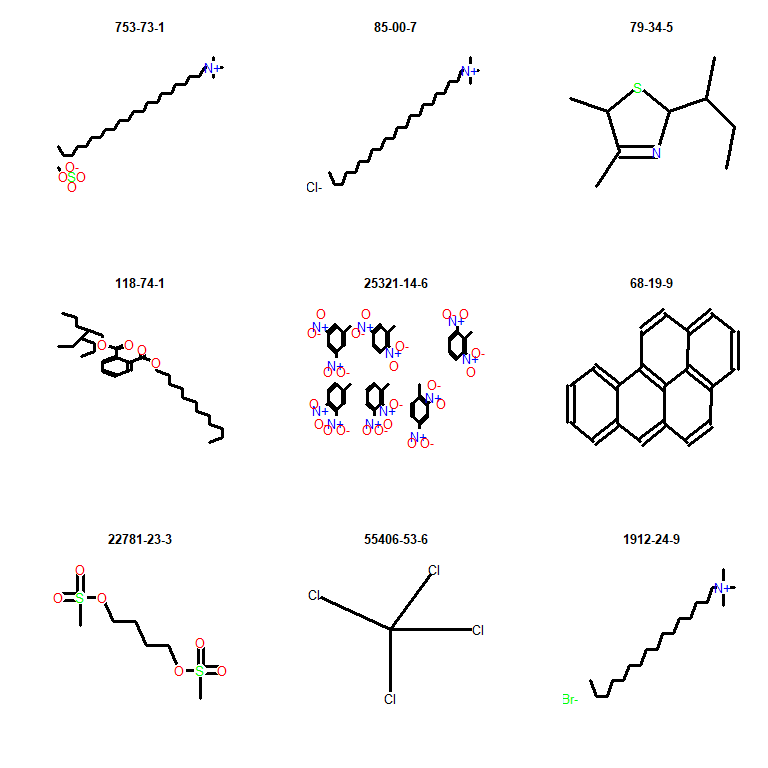
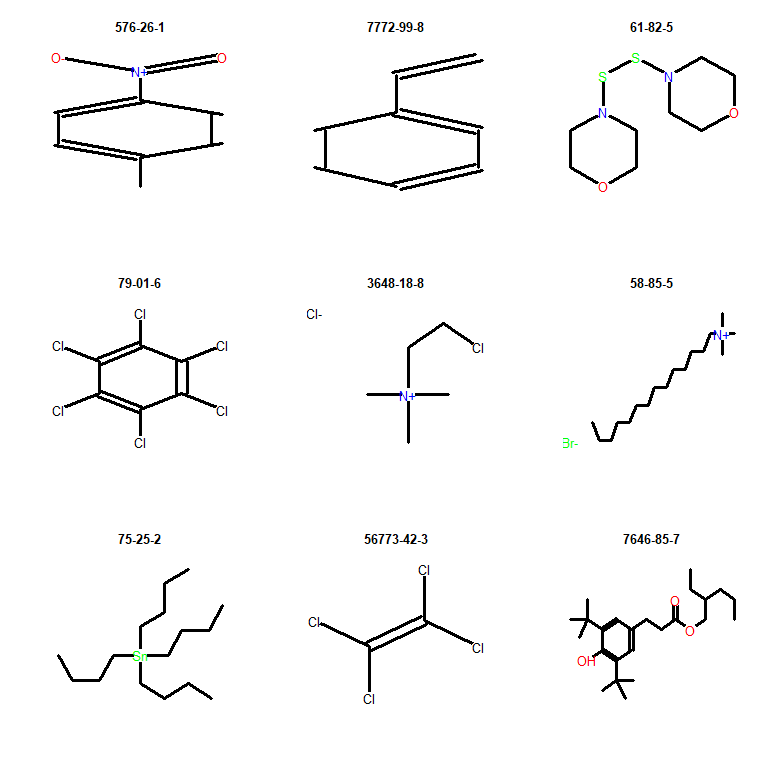
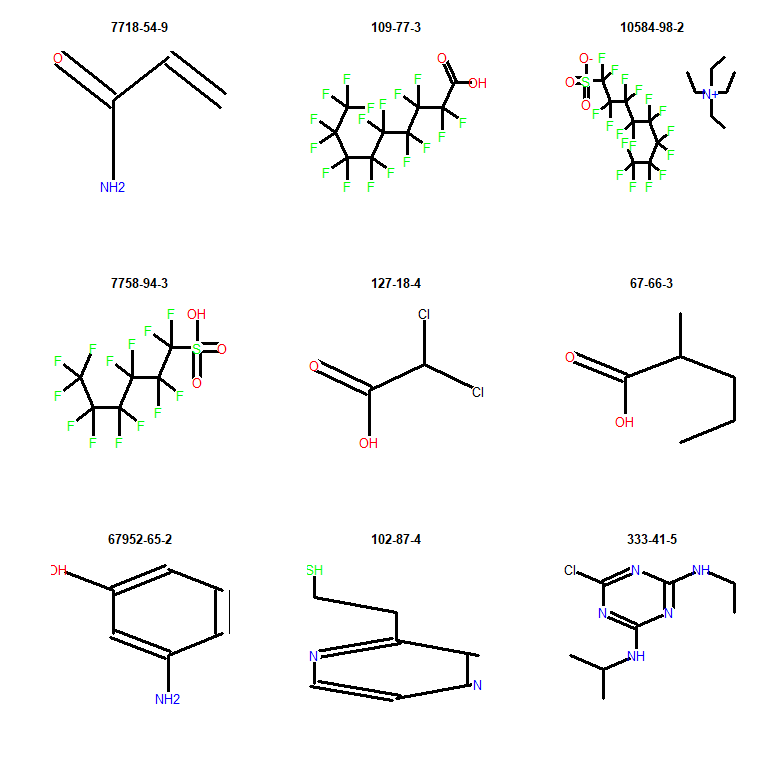
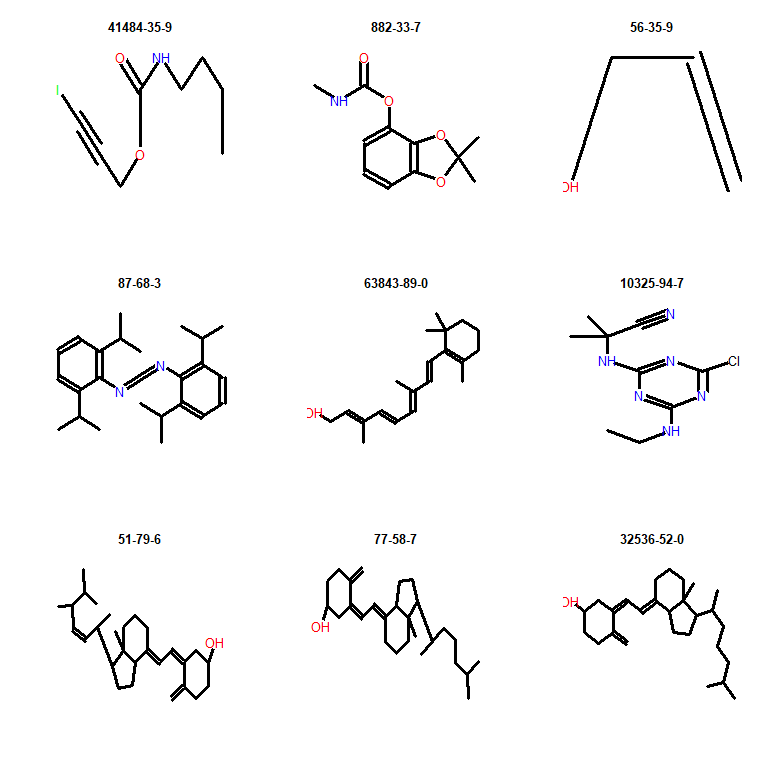
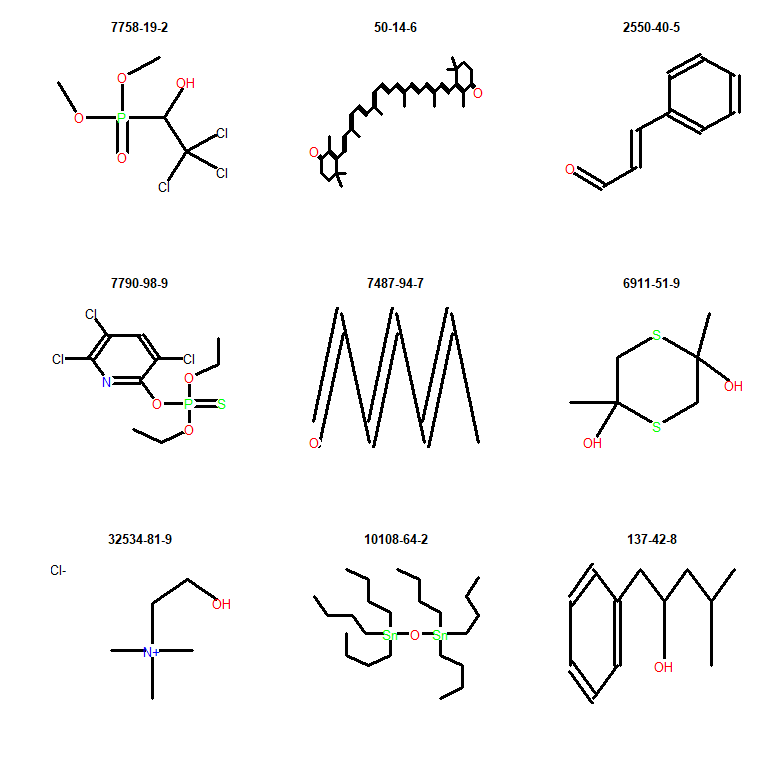
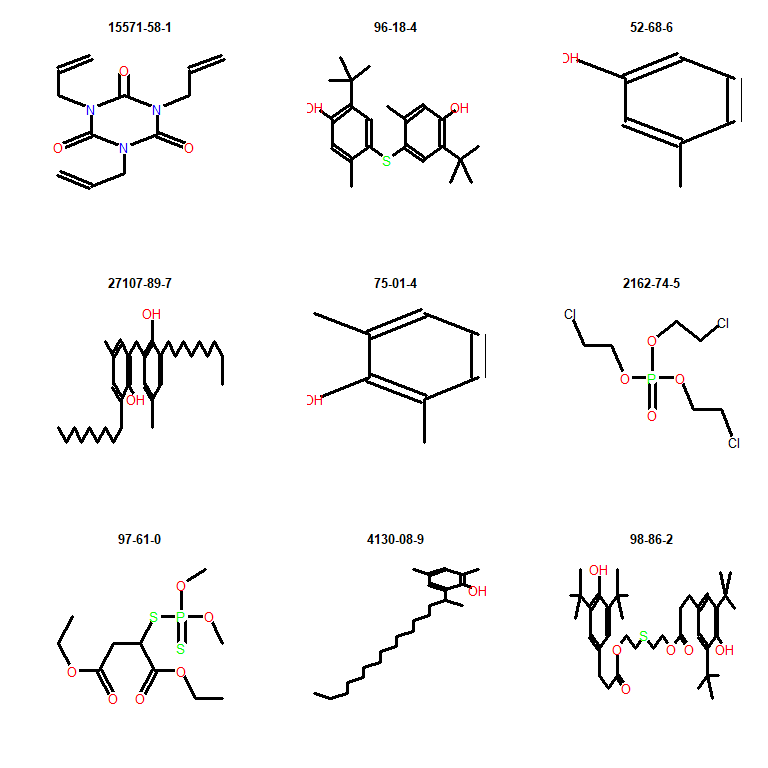
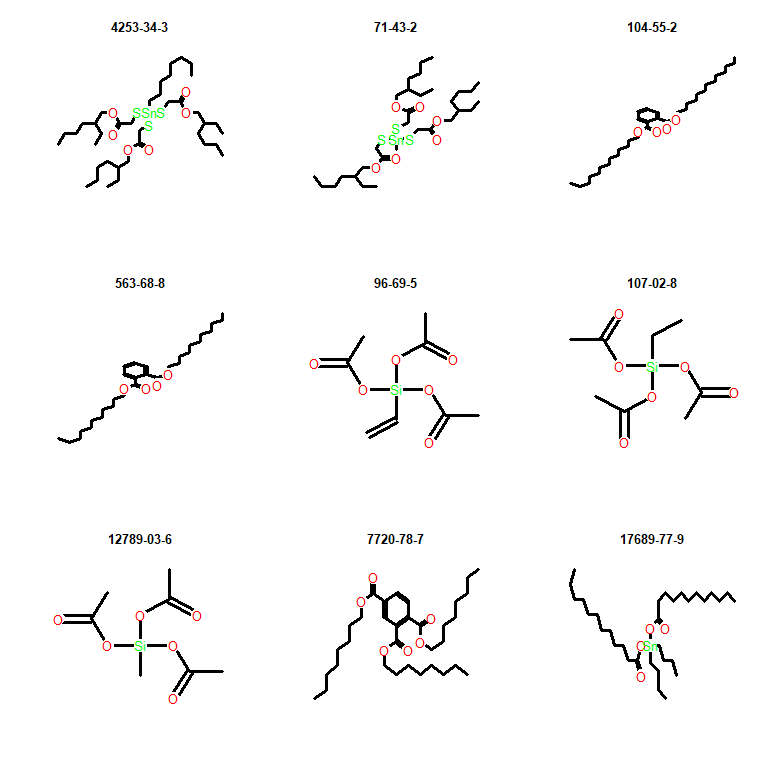
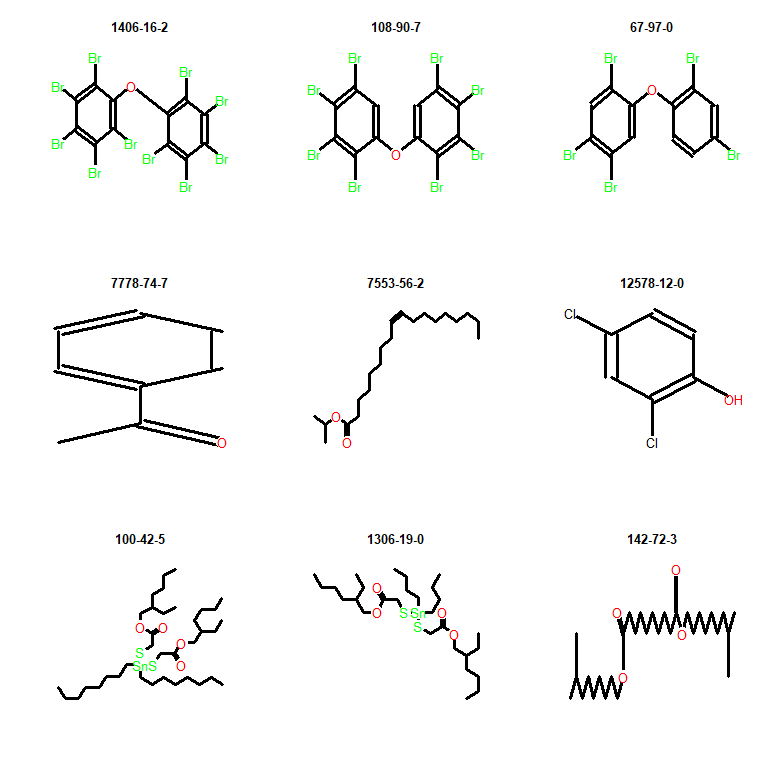
Below is a plot of the compounds where the POD based on bioactivity or read-across were not protective.

plotPODDifferenceFunction(inData=toxValDB, inPlotOrder=plotOrderToxValLowest[which(podDifferenceTVDBLowest<0)], inPODDifference=podDifferenceTVDBLowest[which(podDifferenceTVDBLowest<0)], titleLab="Lowest Traditional POD - No HTTK Filter")



Structures of compounds with bioactivity-based POD > than traditional POD (ToxValDB) without filtering:

nonProtectiveCasToxVal <- plotOrderToxValLowest[1:as.numeric(table(podDifferenceTVDBLowest < 0)["TRUE"])]  
  
nonConservativePODSmilesToxVal <- inHTTKDSL$SMILES[inHTTKDSL$CASRN %in% nonProtectiveCasToxVal]  
  
sdfsetPODTV <- smiles2sdf(nonConservativePODSmilesToxVal)  
cid(sdfsetPODTV) <- nonProtectiveCasToxVal  
  
for (i in 1:length(sdfsetPODTV)) {   
 attributes(sdfsetPODTV[[i]])$header["Molecule\_Name"] <- nonProtectiveCasToxVal[i]   
}  
  
#Check for chemicals without a bondblock  
lackingBondBlock <- vector()  
for (i in 1:length(sdfsetPODTV)) {   
 lackingBondBlock <- c(lackingBondBlock, rownames(attributes(sdfsetPODTV[[i]])$bondblock)[1])   
}  
lackingBondBlock <- which(lackingBondBlock %in% "0")  
  
if (length(lackingBondBlock) > 0) {  
 sdfsetPODTV <- sdfsetPODTV[-lackingBondBlock]  
}  
  
##Plot structures beginning with worst predictions first   
plotNum <- 1  
plotNumTop <- plotNum + 8  
while(plotNum < length(sdfsetPODTV)) {  
 if (plotNumTop > length(sdfsetPODTV)) { plotNumTop <- length(sdfsetPODTV)}  
 invisible(plot(sdfsetPODTV[plotNum:plotNumTop]))  
 plotNum <- plotNum + 9  
 plotNumTop <- plotNumTop + 9  
}



Check for enriched chemotypes for chemicals with non-protective PODs (no filtering):

protectiveCasToxVal <- plotOrderToxValLowest[1:as.numeric(table(podDifferenceTVDBLowest < 0)["FALSE"])]  
nonProtectiveCasToxVal <- plotOrderToxValLowest[1:as.numeric(table(podDifferenceTVDBLowest < 0)["TRUE"])]  
  
protectivePredPODTable <- originalDslFP[rownames(originalDslFP) %in% protectiveCasToxVal,]  
protectivePresent <- colSums(protectivePredPODTable)  
  
nonProtectivePredPODTable <- originalDslFP[rownames(originalDslFP) %in% nonProtectiveCasToxVal,]  
nonProtectivePresent <- colSums(nonProtectivePredPODTable)  
  
FishersPODTable <- rbind(protectivePresent, nonProtectivePresent)  
colnames(FishersPODTable) <- colnames(originalDslFP)  
FishersPODTable <- FishersPODTable[,colSums(FishersPODTable)!=0]  
  
protectiveAbsent <- nrow(protectivePredPODTable)-FishersPODTable["protectivePresent",]  
nonProtectiveAbsent <- nrow(nonProtectivePredPODTable)-FishersPODTable["nonProtectivePresent",]  
  
FishersPODTable <- rbind(FishersPODTable, protectiveAbsent, nonProtectiveAbsent)  
FishersPODTable <- FishersPODTable[c(1,3,2,4),]  
  
pValuePOD <- vector()  
lowerConfIntPOD <- vector()  
upperConfIntPOD <- vector()  
oddsRatioPOD <- vector()  
  
for (i in 1:ncol(FishersPODTable)) {  
  
 tempResult <- fisher.test(matrix(FishersPODTable[,i],2,2))  
  
 pValuePOD <- c(pValuePOD, tempResult$p.value)  
 lowerConfIntPOD <- c(lowerConfInt, tempResult$conf.int[1])  
 upperConfIntPOD <- c(upperConfInt, tempResult$conf.int[2])  
 oddsRatioPOD <- c(oddsRatio, tempResult$estimate)  
  
}  
  
adjustedPValuePOD <- p.adjust(pValuePOD, method="holm")  
  
FishersPODTable <- rbind(FishersPODTable, pValuePOD, adjustedPValuePOD, oddsRatioPOD, lowerConfIntPOD, upperConfIntPOD)  
  
FishersPODTable <- t(FishersPODTable)  
  
enrichedChemoTypesPOD <- FishersPODTable[FishersPODTable[,"pValuePOD"]<=0.05,]  
enrichedChemoTypesPOD <- enrichedChemoTypesPOD[enrichedChemoTypesPOD[,"oddsRatioPOD"]<1,]  
enrichedChemoTypesPOD <- enrichedChemoTypesPOD[order(enrichedChemoTypesPOD[,"pValuePOD"]),]  
  
kable(enrichedChemoTypesPOD[,1:7], digits=4)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | protectivePresent | protectiveAbsent | nonProtectivePresent | nonProtectiveAbsent | pValuePOD | adjustedPValuePOD | oddsRatioPOD |
| bond:metal\_group\_III\_other\_Sn\_generic | 21 | 1965 | 13 | 143 | 0.0000 | 0.0001 | 0.9291 |
| atom:element\_metal\_poor\_metal | 26 | 1960 | 14 | 142 | 0.0000 | 0.0001 | 0.0000 |
| bond:X[any]\_halide | 295 | 1691 | 49 | 107 | 0.0000 | 0.0003 | 0.7400 |
| bond:CS\_sulfide | 98 | 1888 | 23 | 133 | 0.0000 | 0.0048 | 0.1153 |
| bond:COH\_alcohol\_pri-alkyl | 234 | 1752 | 5 | 151 | 0.0003 | 0.1565 | 0.4185 |
| bond:CX\_halide\_alkyl-X\_dihalo\_(1\_1-) | 60 | 1926 | 14 | 142 | 0.0006 | 0.3149 | 0.5390 |
| bond:CX\_halide\_alkyl-X\_generic | 100 | 1886 | 19 | 137 | 0.0008 | 0.3779 | 0.0000 |
| bond:CX\_halide\_alkyl-X\_dihalo\_(1\_2-) | 35 | 1951 | 10 | 146 | 0.0011 | 0.5273 | 0.4138 |
| bond:COH\_alcohol\_generic | 545 | 1441 | 25 | 131 | 0.0014 | 0.6576 | 0.2387 |
| bond:NC=O\_aminocarbonyl\_generic | 227 | 1759 | 6 | 150 | 0.0019 | 0.9213 | 0.5995 |
| bond:metal\_group\_III\_other\_generic\_oxy | 14 | 1972 | 6 | 150 | 0.0022 | 1.0000 | 0.9022 |
| bond:metal\_group\_III\_other\_Sn\_oxy | 9 | 1977 | 5 | 151 | 0.0022 | 1.0000 | 0.4825 |
| bond:CX\_halide\_alkenyl-Cl\_dichloro\_(1\_1-) | 5 | 1981 | 4 | 152 | 0.0026 | 1.0000 | 0.4138 |
| bond:CX\_halide\_alkenyl-X\_dihalo\_(1\_1-) | 5 | 1981 | 4 | 152 | 0.0026 | 1.0000 | 0.3838 |
| bond:CX\_halide\_alkyl-Cl\_trichloro\_(1\_1\_1-) | 15 | 1971 | 6 | 150 | 0.0029 | 1.0000 | 0.2985 |
| bond:CX\_halide\_alkyl-X\_trihalo\_(1\_1\_1-) | 47 | 1939 | 10 | 146 | 0.0068 | 1.0000 | 0.1539 |
| bond:CS\_sulfide\_di- | 13 | 1973 | 5 | 151 | 0.0076 | 1.0000 | 0.1562 |
| bond:CX\_halide\_alkenyl-X\_acyclic\_generic | 13 | 1973 | 5 | 151 | 0.0076 | 1.0000 | 0.3584 |
| bond:CX\_halide\_aromatic-X\_ether\_aromatic\_(Ph-O-Ph)\_generic | 4 | 1982 | 3 | 153 | 0.0107 | 1.0000 | 0.2985 |
| chain:alkaneLinear\_ethyl\_C2(H\_gt\_1) | 899 | 1087 | 54 | 102 | 0.0119 | 1.0000 | 0.2387 |
| bond:CX\_halide\_alkyl-X\_trihalo\_(1\_1\_2-) | 22 | 1964 | 6 | 150 | 0.0133 | 1.0000 | 0.0385 |
| bond:CN\_amine\_pri-NH2\_generic | 130 | 1856 | 3 | 153 | 0.0154 | 1.0000 | 0.5070 |
| ring:aromatic\_benzene | 709 | 1277 | 41 | 115 | 0.0184 | 1.0000 | 0.5995 |
| bond:CX\_halide\_alkenyl-X\_dihalo\_(1\_2-) | 11 | 1975 | 4 | 152 | 0.0196 | 1.0000 | 0.6171 |
| bond:C(=O)O\_carboxylicAcid\_generic | 163 | 1823 | 5 | 151 | 0.0202 | 1.0000 | 0.9652 |
| bond:C(=O)O\_carboxylicEster\_acyclic | 266 | 1720 | 11 | 145 | 0.0248 | 1.0000 | 0.5609 |
| ring:hetero\_[6]\_Z\_1\_4- | 42 | 1944 | 8 | 148 | 0.0256 | 1.0000 | 0.5390 |
| bond:metal\_transition\_Cd\_generic | 2 | 1984 | 2 | 154 | 0.0287 | 1.0000 | 0.0598 |
| ring:fused\_PAH\_benzophenanthrene | 2 | 1984 | 2 | 154 | 0.0287 | 1.0000 | 0.3755 |
| bond:CX\_halide\_alkyl-X\_trihalo\_(1\_2\_3-) | 22 | 1964 | 5 | 151 | 0.0419 | 1.0000 | 0.2570 |
| chain:oxy-alkaneLinear\_ethylenOxide\_EO1(O) | 47 | 1939 | 0 | 156 | 0.0455 | 1.0000 | 0.1761 |
| ring:fused\_PAH\_phenanthrene | 3 | 1983 | 2 | 154 | 0.0455 | 1.0000 | 0.9652 |

Check for enriched chemotypes for chemicals with non-protective PODs (filtering):

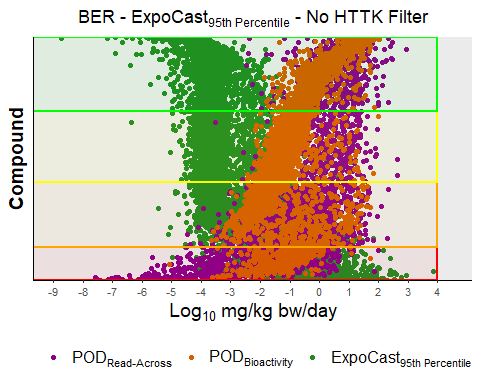
protectiveCasToxVal <- protectiveCasToxVal[protectiveCasToxVal %in% filteredDSLHTTK]  
nonProtectiveCasToxVal <- nonProtectiveCasToxVal[nonProtectiveCasToxVal %in% filteredDSLHTTK]  
  
protectivePredPODTable <- originalDslFP[rownames(originalDslFP) %in% protectiveCasToxVal,]  
protectivePresent <- colSums(protectivePredPODTable)  
  
nonProtectivePredPODTable <- originalDslFP[rownames(originalDslFP) %in% nonProtectiveCasToxVal,]  
nonProtectivePresent <- colSums(nonProtectivePredPODTable)  
  
FishersPODTable <- rbind(protectivePresent, nonProtectivePresent)  
colnames(FishersPODTable) <- colnames(originalDslFP)  
FishersPODTable <- FishersPODTable[,colSums(FishersPODTable)!=0]  
  
protectiveAbsent <- nrow(protectivePredPODTable)-FishersPODTable["protectivePresent",]  
nonProtectiveAbsent <- nrow(nonProtectivePredPODTable)-FishersPODTable["nonProtectivePresent",]  
  
FishersPODTable <- rbind(FishersPODTable, protectiveAbsent, nonProtectiveAbsent)  
FishersPODTable <- FishersPODTable[c(1,3,2,4),]  
  
pValuePOD <- vector()  
lowerConfIntPOD <- vector()  
upperConfIntPOD <- vector()  
oddsRatioPOD <- vector()  
  
for (i in 1:ncol(FishersPODTable)) {  
  
 tempResult <- fisher.test(matrix(FishersPODTable[,i],2,2))  
  
 pValuePOD <- c(pValuePOD, tempResult$p.value)  
 lowerConfIntPOD <- c(lowerConfInt, tempResult$conf.int[1])  
 upperConfIntPOD <- c(upperConfInt, tempResult$conf.int[2])  
 oddsRatioPOD <- c(oddsRatio, tempResult$estimate)  
  
}  
  
adjustedPValuePOD <- p.adjust(pValuePOD, method="holm")  
  
FishersPODTable <- rbind(FishersPODTable, pValuePOD, adjustedPValuePOD, oddsRatioPOD, lowerConfIntPOD, upperConfIntPOD)  
  
FishersPODTable <- t(FishersPODTable)  
  
enrichedChemoTypesPOD <- FishersPODTable[FishersPODTable[,"pValuePOD"]<=0.05,]  
enrichedChemoTypesPOD <- enrichedChemoTypesPOD[enrichedChemoTypesPOD[,"oddsRatioPOD"]<1,]  
enrichedChemoTypesPOD <- enrichedChemoTypesPOD[order(enrichedChemoTypesPOD[,"pValuePOD"]),]  
  
kable(enrichedChemoTypesPOD[,1:7], digits=4)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | protectivePresent | protectiveAbsent | nonProtectivePresent | nonProtectiveAbsent | pValuePOD | adjustedPValuePOD | oddsRatioPOD |
| ring:hetero\_[6]\_N\_pyrazine | 10 | 914 | 5 | 45 | 0.0006 | 0.2249 | 0.4138 |
| bond:C(=O)O\_carboxylicEster\_acyclic | 126 | 798 | 0 | 50 | 0.0017 | 0.6324 | 0.3755 |
| bond:C(=O)O\_carboxylicEster\_alkyl | 117 | 807 | 0 | 50 | 0.0027 | 1.0000 | 0.5609 |
| bond:CS\_sulfide\_dialkyl | 18 | 906 | 5 | 45 | 0.0049 | 1.0000 | 0.2056 |
| ring:hetero\_[6]\_Z\_1\_3\_5- | 11 | 913 | 4 | 46 | 0.0055 | 1.0000 | 0.2387 |
| bond:C(=O)O\_carboxylicEster\_aliphatic | 97 | 827 | 0 | 50 | 0.0069 | 1.0000 | 0.2589 |
| bond:CX\_halide\_alkyl-X\_dihalo\_(1\_2-) | 12 | 912 | 4 | 46 | 0.0071 | 1.0000 | 0.7784 |
| bond:CX\_halide\_alkyl-X\_ethyl\_generic | 12 | 912 | 4 | 46 | 0.0071 | 1.0000 | 0.5099 |
| bond:CX\_halide\_alkyl-X\_primary | 12 | 912 | 4 | 46 | 0.0071 | 1.0000 | 0.4787 |
| bond:CX\_halide\_generic-X\_dihalo\_(1\_2-) | 32 | 892 | 6 | 44 | 0.0105 | 1.0000 | 0.8093 |
| bond:COH\_alcohol\_pri-alkyl | 96 | 828 | 0 | 50 | 0.0117 | 1.0000 | 0.8319 |
| bond:CS\_sulfide | 47 | 877 | 7 | 43 | 0.0168 | 1.0000 | 0.2387 |
| bond:CX\_halide\_alkyl-X\_dihalo\_(1\_1-) | 27 | 897 | 5 | 45 | 0.0204 | 1.0000 | 0.3449 |
| bond:CX\_halide\_alkyl-Cl\_ethyl | 11 | 913 | 3 | 47 | 0.0311 | 1.0000 | 0.3584 |
| bond:CX\_halide\_alkyl-X\_tetrahalo\_(1\_1\_2\_2-) | 4 | 920 | 2 | 48 | 0.0339 | 1.0000 | 0.4058 |
| chain:alkeneCyclic\_ethene\_generic | 67 | 857 | 0 | 50 | 0.0429 | 1.0000 | 0.4185 |

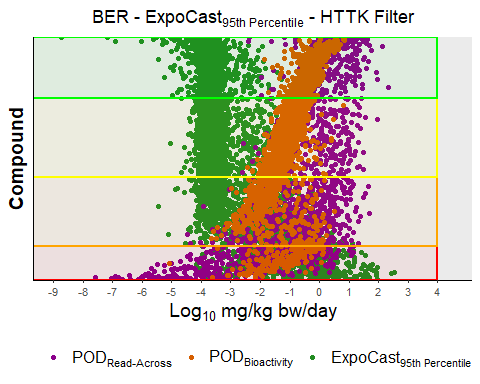
## Calculate BERs

In order to prioritize by risk, the points of depature based on *in vitro* bioactivity are compared against ExpoCast 95th percentile exposure predictions. ExpoCast data were downloaded from CompTox database on October 9, 2020.

##Load in exposure data  
expoCast <- read.xlsx(xlsxFile="ExpoCast\_Seem3\_20201009.xlsx", colNames=TRUE)  
  
#Define plot function  
  
berPlotFunction <- function(inExposureData, titleLab, exposurePercentileLabel, exposurePercentileBreak, inPlotOrder, Bin1Top, Bin1Bottom, Bin2Top, Bin3Top, Bin4Top) {  
 ggplot() +  
 geom\_point(data = inExposureData, aes(y=inExposureData$"CASRN", x=value, color=Group), size=1.5) +  
 ggtitle(titleLab) +  
 ylab("Compound") +   
 xlab(expression(Log["10"]~"mg/kg"~"bw/day")) +  
 scale\_x\_continuous(breaks=seq( floor(min(inExposureData$value[!is.na(inExposureData$value)])), ceiling(max(inExposureData$value[!is.na(inExposureData$value)])) ,1)) +  
 scale\_y\_discrete(limits=inPlotOrder) +  
 coord\_cartesian(xlim = c(floor(min(inExposureData$value[!is.na(inExposureData$value)])),ceiling(max(inExposureData$value[!is.na(inExposureData$value)]))+0.5)) +  
 theme\_bw() +  
 theme(panel.border = element\_blank(),   
 panel.grid.major.y = element\_line(),  
 panel.grid.minor.y = element\_blank(),  
 panel.grid.minor.x = element\_blank(),  
 axis.text.y=element\_blank(),  
 plot.title = element\_text(hjust = 0.5),  
 axis.ticks.y=element\_blank(),  
 axis.line = element\_line(colour = "black"),  
 axis.text = element\_text(size=8),  
 axis.title = element\_text(size=14, face="bold")) +  
 scale\_colour\_manual(  
 breaks=c("Log.POD.Bioactivity", "Log.POD.ReadAcross", exposurePercentileBreak),  
 values=c("Log.POD.Bioactivity" = "#D55E00", "Log.POD.ReadAcross" = "darkmagenta", "forestgreen"),  
 labels=c(expression(POD["Read-Across"]), expression(POD[Bioactivity]), exposurePercentileLabel)) +  
 theme(legend.position="bottom", legend.title=element\_blank(), legend.text=element\_text(size=12)) +  
 annotate("rect",   
 xmin = floor(min(inExposureData$value[!is.na(inExposureData$value)])-1),   
 xmax = ceiling(max(inExposureData$value[!is.na(inExposureData$value)])),   
 ymin = Bin1Bottom,   
 ymax = Bin1Top+0.5,   
 fill = "red",   
 alpha = 0.05,   
 size = 1,   
 color = "red") +  
 annotate("rect",   
 xmin = floor(min(inExposureData$value[!is.na(inExposureData$value)])-1),   
 xmax = ceiling(max(inExposureData$value[!is.na(inExposureData$value)])),   
 ymin = Bin1Top+0.6,   
 ymax = Bin2Top+0.5,   
 fill = "orange",   
 alpha = 0.05,   
 size = 1,   
 color = "orange") +  
 annotate("rect",   
 xmin = floor(min(inExposureData$value[!is.na(inExposureData$value)])-1),   
 xmax = ceiling(max(inExposureData$value[!is.na(inExposureData$value)])),   
 ymin = Bin2Top+0.6,   
 ymax = Bin3Top+0.5,   
 fill = "yellow",   
 alpha = 0.05,   
 size = 1,   
 color = "yellow") +  
 annotate("rect",   
 xmin = floor(min(inExposureData$value[!is.na(inExposureData$value)])-1),   
 xmax = ceiling(max(inExposureData$value[!is.na(inExposureData$value)])),   
 ymin = Bin3Top+0.6,   
 ymax = Bin4Top+0.5,   
 fill = "green",   
 alpha = 0.05,   
 size = 1,   
 color = "green")  
}  
  
##Merge exposure data with summaryResults for comparison  
percentile95expoCast <- expoCast[expoCast$INPUT %in% summaryResults$CASRN, c("INPUT", "EXPOCAST\_95th\_PERCENTILE\_EXPOSURE\_PREDICTION\_MG/KG-BW/DAY")]  
Group <- rep("ExpoCast\_95th\_Percentile", nrow(percentile95expoCast))  
percentile95expoCast <- cbind(percentile95expoCast, Group)  
colnames(percentile95expoCast)[colnames(percentile95expoCast) %in% "EXPOCAST\_95th\_PERCENTILE\_EXPOSURE\_PREDICTION\_MG/KG-BW/DAY"] <- "value"  
colnames(percentile95expoCast)[colnames(percentile95expoCast) %in% "INPUT"] <- "CASRN"  
percentile95expoCast$value <- log10(as.numeric(percentile95expoCast$value))  
  
percentile95expoCast <- as.data.frame(rbind(summaryResults[summaryResults$CASRN %in% unique(percentile95expoCast$CASRN),], percentile95expoCast[,c("CASRN", "Group","value")]))  
  
##Determine plot order - unfiltered  
percentile95CasrnForComparisonBER <- unique(percentile95expoCast$CASRN)  
percentile95Ber <- rep(NA, length(percentile95CasrnForComparisonBER))  
  
for (i in 1:length(percentile95CasrnForComparisonBER)) {  
 percentile95Ber[i] <- as.numeric(percentile95expoCast[percentile95expoCast$CASRN == percentile95CasrnForComparisonBER[i] & grepl("Log.POD", percentile95expoCast$Group), "value"]) - as.numeric(percentile95expoCast[percentile95expoCast$CASRN == percentile95CasrnForComparisonBER[i] & percentile95expoCast$Group == "ExpoCast\_95th\_Percentile", "value"])  
}  
  
percentile95CasrnForComparisonBER <- percentile95CasrnForComparisonBER[which(!is.na(percentile95Ber))]  
percentile95Ber <- percentile95Ber[which(!is.na(percentile95Ber))]  
  
percentile95PlotOrderExpoCast <- percentile95CasrnForComparisonBER[order(percentile95Ber)]  
  
percentile95expoCast <- as.data.table(percentile95expoCast)  
  
percentile95Bin1Top = table(percentile95Ber < 0)["TRUE"]  
percentile95Bin1Bottom = 0  
percentile95Bin2Top = table(percentile95Ber < 2)["TRUE"]  
percentile95Bin3Top = table(percentile95Ber < 3)["TRUE"]  
percentile95Bin4Top = length(percentile95Ber)  
  
percentile95expoCast <- percentile95expoCast[rev(order(percentile95expoCast$Group))]  
  
#Plot unfiltered  
berPlotFunction(percentile95expoCast, expression(BER~"-"~ExpoCast["95th Percentile"]~"-"~No~HTTK~Filter), expression(ExpoCast["95th Percentile"]), "ExpoCast\_95th\_Percentile", percentile95PlotOrderExpoCast, percentile95Bin1Top, percentile95Bin1Bottom, percentile95Bin2Top, percentile95Bin3Top, percentile95Bin4Top)



##Determine plot order - filtered  
percentile95CasrnForComparisonBERFiltered <- unique(percentile95expoCast$CASRN)  
percentile95CasrnForComparisonBERFiltered <- percentile95CasrnForComparisonBERFiltered[percentile95CasrnForComparisonBERFiltered %in% filteredDSLHTTK]  
percentile95BerFiltered <- rep(NA, length(percentile95CasrnForComparisonBERFiltered))  
  
for (i in 1:length(percentile95CasrnForComparisonBERFiltered)) {  
 percentile95BerFiltered[i] <- as.numeric(percentile95expoCast[percentile95expoCast$CASRN == percentile95CasrnForComparisonBERFiltered[i] & grepl("Log.POD", percentile95expoCast$Group), "value"]) - as.numeric(percentile95expoCast[percentile95expoCast$CASRN == percentile95CasrnForComparisonBERFiltered[i] & percentile95expoCast$Group == "ExpoCast\_95th\_Percentile", "value"])  
}  
  
percentile95CasrnForComparisonBERFiltered <- percentile95CasrnForComparisonBERFiltered[which(!is.na(percentile95BerFiltered))]  
percentile95BerFiltered <- percentile95BerFiltered[which(!is.na(percentile95BerFiltered))]  
  
percentile95PlotOrderExpoCastFiltered <- percentile95CasrnForComparisonBERFiltered[order(percentile95BerFiltered)]  
  
percentile95Bin1TopFiltered = table(percentile95BerFiltered < 0)["TRUE"]  
percentile95Bin1BottomFiltered = 0  
percentile95Bin2TopFiltered = table(percentile95BerFiltered < 2)["TRUE"]  
percentile95Bin3TopFiltered = table(percentile95BerFiltered < 3)["TRUE"]  
percentile95Bin4TopFiltered = length(percentile95BerFiltered)  
  
#Plot filtered  
berPlotFunction(percentile95expoCast, expression(BER~"-"~ExpoCast["95th Percentile"]~"-"~HTTK~Filter), expression(ExpoCast["95th Percentile"]), "ExpoCast\_95th\_Percentile", percentile95PlotOrderExpoCastFiltered, percentile95Bin1TopFiltered, percentile95Bin1BottomFiltered, percentile95Bin2TopFiltered, percentile95Bin3TopFiltered, percentile95Bin4TopFiltered)

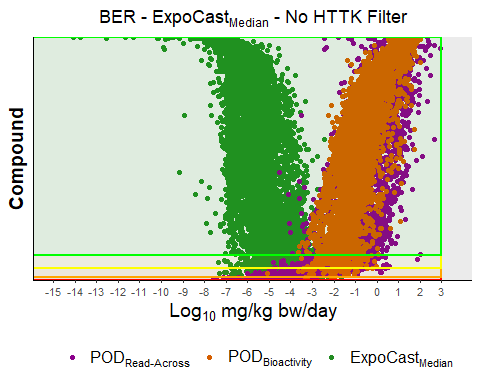


The BER results above show that there are 953 compounds with a log10 BER < 0 and 1877 compounds with a log10 BER < 2. Furthermore, there are 2077 compounds with a log10 BER < 3 that may be considered more on a case-by-case basis, and 2135 compounds with a log10 BER > 3.

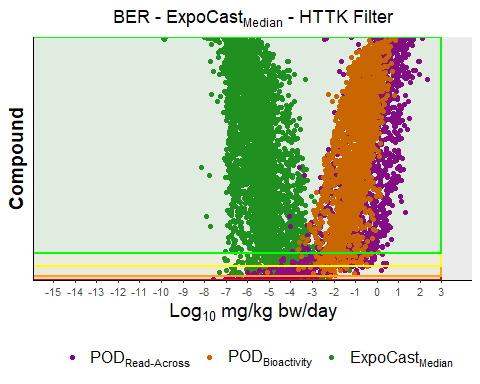
After applying the HTTK filter, the BER results above show that there are 505 compounds with a log10 BER < 0 and 1054 compounds with a log10 BER < 2. Furthermore, there are 1200 compounds with a log10 BER < 3 that may be considered more on a case-by-case basis, and 921 compounds with a log10 BER > 3.

Below, the points of depature based on *in vitro* bioactivity are compared against ExpoCast median exposure predictions. ExpoCast data downloaded form CompTox database on October 9, 2020.

##Merge exposure data with summaryResults for comparison  
medianexpoCast <- expoCast[expoCast$INPUT %in% summaryResults$CASRN, c("INPUT", "EXPOCAST\_MEDIAN\_EXPOSURE\_PREDICTION\_MG/KG-BW/DAY")]  
Group <- rep("ExpoCast\_Median", nrow(medianexpoCast))  
medianexpoCast <- cbind(medianexpoCast, Group)  
colnames(medianexpoCast)[colnames(medianexpoCast) %in% "EXPOCAST\_MEDIAN\_EXPOSURE\_PREDICTION\_MG/KG-BW/DAY"] <- "value"  
colnames(medianexpoCast)[colnames(medianexpoCast) %in% "INPUT"] <- "CASRN"  
medianexpoCast$value <- log10(as.numeric(medianexpoCast$value))  
  
medianexpoCast <- as.data.frame(rbind(summaryResults[summaryResults$CASRN %in% unique(medianexpoCast$CASRN),], medianexpoCast[,c("CASRN", "Group","value")]))  
  
##Determine plot order - unfiltered  
medianCasrnForComparisonBER <- unique(medianexpoCast$CASRN)  
medianBer <- rep(NA, length(medianCasrnForComparisonBER))  
  
for (i in 1:length(medianCasrnForComparisonBER)) {  
 medianBer[i] <- as.numeric(medianexpoCast[medianexpoCast$CASRN == medianCasrnForComparisonBER[i] & grepl("Log.POD", medianexpoCast$Group), "value"]) - as.numeric(medianexpoCast[medianexpoCast$CASRN == medianCasrnForComparisonBER[i] & medianexpoCast$Group == "ExpoCast\_Median", "value"])  
}  
  
medianCasrnForComparisonBER <- medianCasrnForComparisonBER[which(!is.na(medianBer))]  
medianBer <- medianBer[which(!is.na(medianBer))]  
  
medianPlotOrderExpoCast <- medianCasrnForComparisonBER[order(medianBer)]  
  
medianexpoCast <- as.data.table(medianexpoCast)  
  
medianBin1Top = table(medianBer < 0)["TRUE"]  
medianBin1Bottom = 0  
medianBin2Top = table(medianBer < 2)["TRUE"]  
medianBin3Top = table(medianBer < 3)["TRUE"]  
medianBin4Top = length(medianBer)  
  
medianexpoCast <- medianexpoCast[rev(order(medianexpoCast$Group))]  
  
#Plot unfiltered  
berPlotFunction(medianexpoCast, expression(BER~"-"~ExpoCast["Median"]~"-"~No~HTTK~Filter), expression(ExpoCast["Median"]), "ExpoCast\_Median", medianPlotOrderExpoCast, medianBin1Top, medianBin1Bottom, medianBin2Top, medianBin3Top, medianBin4Top)



##Determine plot order - filtered  
medianCasrnForComparisonBERFiltered <- unique(medianexpoCast$CASRN)  
medianCasrnForComparisonBERFiltered <- medianCasrnForComparisonBERFiltered[medianCasrnForComparisonBERFiltered %in% filteredDSLHTTK]  
medianBerFiltered <- rep(NA, length(medianCasrnForComparisonBERFiltered))  
  
for (i in 1:length(medianCasrnForComparisonBERFiltered)) {  
 medianBerFiltered[i] <- as.numeric(medianexpoCast[medianexpoCast$CASRN == medianCasrnForComparisonBERFiltered[i] & grepl("Log.POD", medianexpoCast$Group), "value"]) - as.numeric(medianexpoCast[medianexpoCast$CASRN == medianCasrnForComparisonBERFiltered[i] & medianexpoCast$Group == "ExpoCast\_Median", "value"])  
}  
  
medianCasrnForComparisonBERFiltered <- medianCasrnForComparisonBERFiltered[which(!is.na(medianBerFiltered))]  
medianBerFiltered <- medianBerFiltered[which(!is.na(medianBerFiltered))]  
  
medianPlotOrderExpoCastFiltered <- medianCasrnForComparisonBERFiltered[order(medianBerFiltered)]  
  
medianBin1TopFiltered = table(medianBerFiltered < 0)["TRUE"]  
medianBin1BottomFiltered = 0  
medianBin2TopFiltered = table(medianBerFiltered < 2)["TRUE"]  
medianBin3TopFiltered = table(medianBerFiltered < 3)["TRUE"]  
medianBin4TopFiltered = length(medianBerFiltered)  
  
#Plot filtered  
berPlotFunction(medianexpoCast, expression(BER~"-"~ExpoCast["Median"]~"-"~HTTK~Filter), expression(ExpoCast["Median"]), "ExpoCast\_Median", medianPlotOrderExpoCastFiltered, medianBin1TopFiltered, medianBin1BottomFiltered, medianBin2TopFiltered, medianBin3TopFiltered, medianBin4TopFiltered)



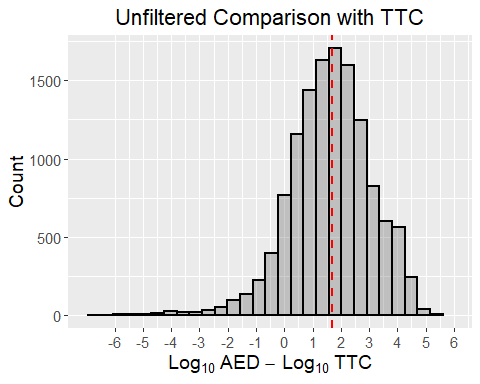
The BER results above show that there are 89 compounds with a log10 BER < 0 and 249 compounds with a log10 BER < 2. Furthermore, there are 390 compounds with a log10 BER < 3 that may be considered more on a case-by-case basis, and 6314 compounds with a log10 BER > 3.

After applying the HTTK filter, the BER results above show that there are 55 compounds with a log10 BER < 0 and 149 compounds with a log10 BER < 2. Furthermore, there are 206 compounds with a log10 BER < 3 that may be considered more on a case-by-case basis, and 3270 compounds with a log10 BER > 3.

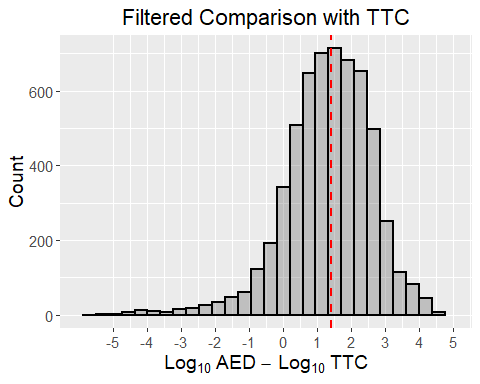
rownames(expoCast) <- expoCast$INPUT  
  
summaryTable[medianCasrnForComparisonBER, "Log10BERMedian"] <- medianBer  
summaryTable[percentile95CasrnForComparisonBER, "Log10BER95th"] <- percentile95Ber  
  
summaryTable[expoCast$INPUT,"ExpoCast95thPercentile"] <- expoCast[expoCast$INPUT,"EXPOCAST\_95th\_PERCENTILE\_EXPOSURE\_PREDICTION\_MG/KG-BW/DAY"]  
  
summaryTable[expoCast$INPUT,"ExpoCastMedian"] <- expoCast[expoCast$INPUT,"EXPOCAST\_MEDIAN\_EXPOSURE\_PREDICTION\_MG/KG-BW/DAY"]

## Compare to Threshold of Toxicological Concern

ttc <- read.xlsx("DSL\_TTC.xlsx")  
  
rownames(ttc) <- ttc$CASRN  
  
overlappingCASRN <- ttc$CASRN[ttc$CASRN %in% summaryTable[,"CASRN"]]  
  
summaryTable[overlappingCASRN, "TTC(mg/kg-bw/day)"] <- ttc[overlappingCASRN,"TTC\_mg/kg"]  
summaryTable[overlappingCASRN, "CramerClass"] <- ttc[overlappingCASRN,"Cramer\_Rules"]  
summaryTable[overlappingCASRN,"ExposureAboveTTC"] <- as.numeric(summaryTable[overlappingCASRN,"ExpoCast95thPercentile"])>as.numeric(summaryTable[overlappingCASRN,"TTC(mg/kg-bw/day)"])  
  
ttcDifference <- log10(as.numeric(summaryTable[,"AED(mg/kg-bw/day)"]))-log10(as.numeric(summaryTable[,"TTC(mg/kg-bw/day)"]))  
ttcDifference <- ttcDifference[!is.na(ttcDifference)]  
  
ttcDifference <- data.frame(label=1:length(ttcDifference), value=as.numeric(ttcDifference))  
  
suppressWarnings(  
 print(  
 ggplot(ttcDifference, aes(x=as.numeric(value))) +   
 geom\_histogram(aes(y=..count..), alpha=0.3, color="black", size=1) +  
 geom\_vline(aes(xintercept=median(ttcDifference$value)), color="red", linetype="dashed", size=1) +  
 ggtitle("Unfiltered Comparison with TTC") +  
 ylab("Count") +  
 scale\_x\_continuous(name=expression(Log["10"]~AED~-~Log["10"]~TTC), limits=c(floor(min(ttcDifference$value)), ceiling(max(ttcDifference$value))), breaks=-seq(floor(min(ttcDifference$value)), ceiling(max(ttcDifference$value)),1)) +  
 theme(text = element\_text(size=14),  
 plot.title = element\_text(hjust = 0.5)  
 )  
 )  
)

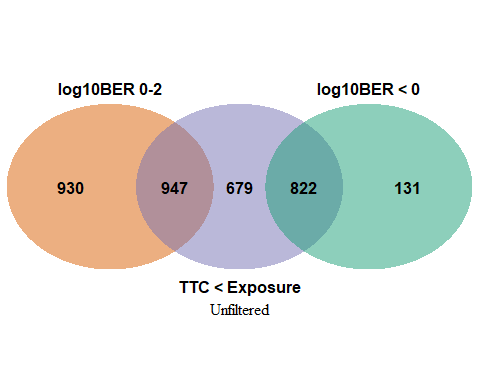


ttcDifferenceFiltered <- log10(as.numeric(summaryTable[summaryTable[,"CASRN"] %in% filteredDSLHTTK,"AED(mg/kg-bw/day)"]))-log10(as.numeric(summaryTable[summaryTable[,"CASRN"] %in% filteredDSLHTTK,"TTC(mg/kg-bw/day)"]))  
ttcDifferenceFiltered <- ttcDifferenceFiltered[!is.na(ttcDifferenceFiltered)]  
  
ttcDifferenceFiltered <- data.frame(label=1:length(ttcDifferenceFiltered), value=as.numeric(ttcDifferenceFiltered))  
  
suppressWarnings(  
 print(  
 ggplot(ttcDifferenceFiltered, aes(x=as.numeric(value))) +   
 geom\_histogram(aes(y=..count..), alpha=0.3, color="black", size=1) +  
 geom\_vline(aes(xintercept=median(ttcDifferenceFiltered$value)), color="red", linetype="dashed", size=1) +  
 ggtitle("Filtered Comparison with TTC") +  
 ylab("Count") +  
 scale\_x\_continuous(name=expression(Log["10"]~AED~-~Log["10"]~TTC), limits=c(floor(min(ttcDifferenceFiltered$value)), ceiling(max(ttcDifferenceFiltered$value))), breaks=-seq(floor(min(ttcDifferenceFiltered$value)), ceiling(max(ttcDifferenceFiltered$value)),1)) +  
 theme(text = element\_text(size=14),  
 plot.title = element\_text(hjust = 0.5)  
 )  
 )  
)

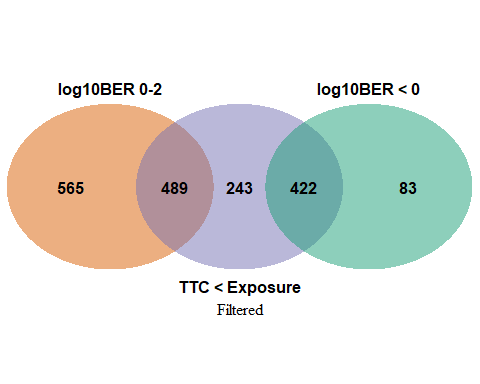


The filtered log10Difference in AED - TTC is 1.393832 which is 24.7646414 on arithmetic scale. The TTC was lower than AED for 87.6724137931034% of compounds.

#Examine how the BERs overlap with chemicals where exposure is above TTC  
percentile95BERNegative <- summaryTable[as.numeric(summaryTable[,"Log10BER95th"])<0,"CASRN"]  
percentile95BERNegative <- as.character(percentile95BERNegative[!is.na(percentile95BERNegative)])  
  
percentile95BER0To2 <- summaryTable[as.numeric(summaryTable[,"Log10BER95th"])>=0 & as.numeric(summaryTable[,"Log10BER95th"])<2,"CASRN"]  
percentile95BER0To2 <- as.character(percentile95BER0To2[!is.na(percentile95BER0To2)])  
  
ttcBelow95th <- summaryTable[as.numeric(summaryTable[,"TTC(mg/kg-bw/day)"])<as.numeric(summaryTable[,"ExpoCast95thPercentile"]),"CASRN"]  
ttcBelow95th <- as.character(ttcBelow95th[!is.na(ttcBelow95th)])  
  
ttcVennPlot <- venn.diagram(  
 x = list(percentile95BERNegative, percentile95BER0To2, ttcBelow95th),  
 category.names = c("log10BER < 0" , "log10BER 0-2", "TTC < Exposure"),  
 filename = NULL,  
 output=TRUE,  
 main="Unfiltered",  
 main.pos=c(0.5,0.2),  
  
   
 # Output features  
 imagetype="png" ,  
 height = 480 ,   
 width = 480 ,   
 resolution = 300,  
 compression = "lzw",  
   
 # Circles  
 lwd = 2,  
 lty = 'blank',  
 fill = brewer.pal(3, "Dark2"),  
   
 # Numbers  
 cex = 1,  
 fontface = "bold",  
 fontfamily = "sans",  
   
 # Set names  
 cat.cex = 1,  
 cat.fontface = "bold",  
 cat.default.pos = "outer",  
 cat.pos = c(0, 180, 0),  
 cat.dist = c(0.055, 0.055, 0.055),  
 cat.fontfamily = "sans",  
 rotation = 1  
)  
  
grid.draw(ttcVennPlot)



#Repeat after applying HTTK filter  
percentile95BERNegative <- percentile95BERNegative[percentile95BERNegative %in% filteredDSLHTTK]  
percentile95BER0To2 <- percentile95BER0To2[percentile95BER0To2 %in% filteredDSLHTTK]  
ttcBelow95th <- ttcBelow95th[ttcBelow95th %in% filteredDSLHTTK]  
  
ttcVennPlot <- venn.diagram(  
 x = list(percentile95BERNegative, percentile95BER0To2, ttcBelow95th),  
 category.names = c("log10BER < 0" , "log10BER 0-2", "TTC < Exposure"),  
 filename = NULL,  
 output=TRUE,  
 main="Filtered",  
 main.pos=c(0.5,0.2),  
  
   
 # Output features  
 imagetype="png" ,  
 height = 480 ,   
 width = 480 ,   
 resolution = 300,  
 compression = "lzw",  
   
 # Circles  
 lwd = 2,  
 lty = 'blank',  
 fill = brewer.pal(3, "Dark2"),  
   
 # Numbers  
 cex = 1,  
 fontface = "bold",  
 fontfamily = "sans",  
   
 # Set names  
 cat.cex = 1,  
 cat.fontface = "bold",  
 cat.default.pos = "outer",  
 cat.pos = c(0, 180, 0),  
 cat.dist = c(0.055, 0.055, 0.055),  
 cat.fontfamily = "sans",  
 rotation = 1  
)  
  
grid.draw(ttcVennPlot)



#Get the partition information for reporting  
ttcVennPartitions <- get.venn.partitions(list(percentile95BERNegative, percentile95BER0To2, ttcBelow95th))

After filtering, there were 911 chemicals with a BER < 2 and an exposure above TTC out of 1154 chemicals where exposure was above TTC.

write.xlsx(summaryTable, "summaryTable.xlsx")