Supplementary Information

Scaffold-Based Analytics: Enabling Hit-to-Lead Decisions by Visualizing Chemical Series Linked Across Large Datasets

Deepak Bandyopadhyay[†], Constantine Kreatsoulas[†] Pat G. Brady[†], Joseph Boyer[†], Zangdong He[†], Genaro Scavello Jr.[†], Tyler Peryea[‡], Ajit Jadhav[‡], Dac-Trung Nguyen[‡], Rajarshi Guha[‡]

[†] GlaxoSmithKline, 1250 S. Collegeville Rd, Collegeville, PA 19426

[‡] National Center for Advancing Translational Science, 9800 Medical Center Drive, Rockville, MD 20850

May 10, 2019

S1 NCATS R-group tool output files

Column Name	Description
ScaffoldID	Numeric scaffold identifier. Each scaffold occurs only once, and data columns are aggregated for all molecules containing the scaffold
Structure	Scaffold SMILES without R-groups attached
RgroupLabels	A comma separated list of R-group labels for all R-groups associated with the scaffold
ScaffoldScore	A quantitative assessment of the scaffold quality. See Section S2
Complexity	A number that captures increasing size and complexity of scaffolds. See Section S2.
Count	Number of molecules that share this scaffold

Table S1: A description of the fixed columns of the scaffold file generated by the NCATS R-group tool. Additional columns may be present which correspond to aggregated property columns. Thus for each property of the input molecules, we compute the mean and standard deviation of that property for all molecules containing the scaffold. These values are reported in columns labeled `X` and X_{sd}, where `X` is the property name.

Column Name	Description
ScaffoldID	Numeric scaffold identifier (corresponding to the *ScaffoldID* column in the scaffold file, Table S1)
MolID	Numeric or text molecule identifier (name). Each molecule is repeated once for each scaffold that it occurs in
Structure	Molecule structure in SMILES format
$R_1, \ldots, R_n$	R-group SMILES, with -atoms at attachment points. By default we limit to $n = 21$

Table S2: A description of the columns in the R-group decomposition file generated by the NCATS R-group tool.

S2 Scaffold metrics

The NCATS R-group tool is designed to fragment a collection of molecules. In addition to the fragmentation procedure it computes a series of scaffold metrics, described in Table S1. In this section we provide some details about the *ScaffoldScore* and *Complexity* metrics.

The *ScaffoldScore* is an empirical metric designed to summarize a scaffold (or more generally, a fragment) and the compounds containing the scaffold. Specifically, we define it as

$$S = -\log_{10}\left(\sqrt{N_{\text{core}} \times \frac{N_m}{N} \times \frac{1}{\sqrt{\sigma}} \times \frac{1}{R}}\right) \tag{1}$$

where N_{core} is the atom count of the scaffold, N_m is the size of the member set for the scaffold, N is the total number of molecules used as input, R is the number of R-groups identified for this scaffold and σ is a measure of how close the members are to the scaffold and is defined as

$$\sigma = \sum_{i=1}^{N_m} (A_i - N_{\text{core}})^2 \tag{2}$$

where A_i is the atom count of the i'th molecule in the scaffolds member set. In summary, the score for a scaffold is higher if it is larger, with fewer R-groups and with member molecules that are relatively close to the scaffold and cover a large fraction of the input set.

The *Complexity* metric is an implemention of the empirical complexity metric described by Barone and Channon [1]. *Complexity* can be used to prune away scaffolds that are too simple, by setting a cutoff such as 100.

S3 NCATS R-group tool input preprocessing

Here are the implementation details specific to summarizing screening datasets such as TCAMS and Kinase X successfully with the NCATS R-group tool:

- Convert ligand efficiency columns (LE/LLE) to LE_x10 (multiply by 10). Otherwise it will be rounded to one decimal place while summarizing (eg. 0.3 ± 0.1), which isn't useful. The conversion was done using a formula in Excel prior to converting the dataset from CSV to SDF. Summarized LE_x10 columns may be back-converted to LE by dividing by 10 in a Spotfire calculated column.

- Convert integer columns that we are interested in summarizing to floating point format, i.e., append a ".0" to these integer values. Since Pubchem IDs are integers, there is logic built into the NCATS scaffold summarization code to ignore columns containing integers. However, percentage inhibition, IFI (percent), and many other measurements that are useful to aggregate are expressed as integers. We got around this by matching a pattern in the SDF file and replacing a PCT_INHIBITION column followed by an integer with the same number followed by ".0" using a Unix sed script. This technique helps work around assumptions made by the NCATS code that integers are compound IDs, and aggregate these columns.

- If the dataset contains Encoded Library Technology features or similar molecules, ensure that pendant R-atoms at building block attachment points are deleted rather than turned to Carbon. We implemented this conversion using a MOE SVL script "db_deleteatoms.svl" written by Barbara Sander at Chemical Computing group. The script will loop through a MOE database and delete all atoms named A in a database molecule entry, which are the R-atoms. The same thing can be accomplished in the reader's favorite cheminformatics package if they have to deal with molecules with pending R-atoms.

S4 Molecular frameworks input and preprocessing

The input for our implementation of frameworks is a comma separated text file with molecules encoded in a SMILES field. The code was modified by adding scripts to export the fuzzy clusters in a tabular format rather than prioritize them into mutually exclusive scaffolds as in [2]. This step produces a file similar to the R-group decomposition format described for the NCATS R-group tool in Section S1, including the following key columns: *framework ID, framework SMILES, molecule ID, SMILES* and *properties/activities*. There are no R-group columns simply because this is not a default computation in our frameworks code.

Bemis-Murcko-like and Recap fragments can be built from datasets within the GSK computational chemistry environment as described in [2]:

- Start with a two column space separated SMILES file, i.e., chemblntd_gsk2_spc.smi which contains molecule parent smiles (no salt forms) and IDs separated by a space.
- Run a Python script to generate framework descriptors for each molecule in this dataset; at GSK this script is called build_dd.py and uses JChem library functions, but the reader will find it easy to put together an equivalent script using their preferred cheminformatics toolkit. The descriptors output by build_dd.py include Detail Frameworks (chemblntd_gsk2_spc_df.db) and Recap (chemblntd_gsk2_spc_recap.db). These tab-delimited files contain one line per fragment found in any molecule, listing the fragment SMILES and the molecule ID, with all fragments in the first molecule before all in the second, and so on.
- Convert the two db files into files listing fragments shared by more than one molecule and not unique to a molecule. The resulting file is sorted so that all molecules containing the same fragment are on contiguous lines. Example Unix script for detailed frameworks:

```
cat chemblntd_gsk2_spc_df.db | sort -V -t \t -k 1
| awk '{print $2, $1}' | uniq -D -f 1
> chemblntd_gsk2_spc_frames_shared.txt
```

- As an optional step we did not implement, a scoring function may be

computed from the aggregate activity of a fragment and used to triage frameworks as was done for scaffolds from the NCATS R-group tool.

S5 Implementation Detail of linking Spotfire tables with different scaffold generation methods

Complete Linkage Clustering adds a Cluster Number to the primary data table. To get the Related Molecules, we simply add a duplicate copy of the primary data table and link it to the original via Scaffold ID. In other words, a Table Relation is entered into Spotfire so that $Main.CLink = Main(2).CLink$.

GSK Frameworks are similar to Clusters, except the one-to-many rather than one-to-one mapping of molecules to frameworks. To get the Related Molecules, we add an original and a duplicate copy of this mapping, and set Relations so that:

- $Main.Molecule\_ID = Frames(2).Molecule\_ID$
- $Frames(2).Framework\_ID = Frames.Framework\_ID$

NCATS R-group Tool adds an additional Annotation layer, i.e., the scaffold-level summaries in addition to the R-group decomposition table. In order to enable bidirectional navigation from scaffolds to molecules, we add both these tables and also a duplicate copy of the R-group decomposition table. Then Relations are set up as follows within Spotfire:

- $Main.Molecule\_ID = RGdecomp(2).Molecule\_ID$
- $RGdecomp(2).Scaffold\_ID = Scaffolds.Scaffold\_ID$
- $Scaffolds.Scaffold\_ID = RGdecomp.Scaffold\_ID$

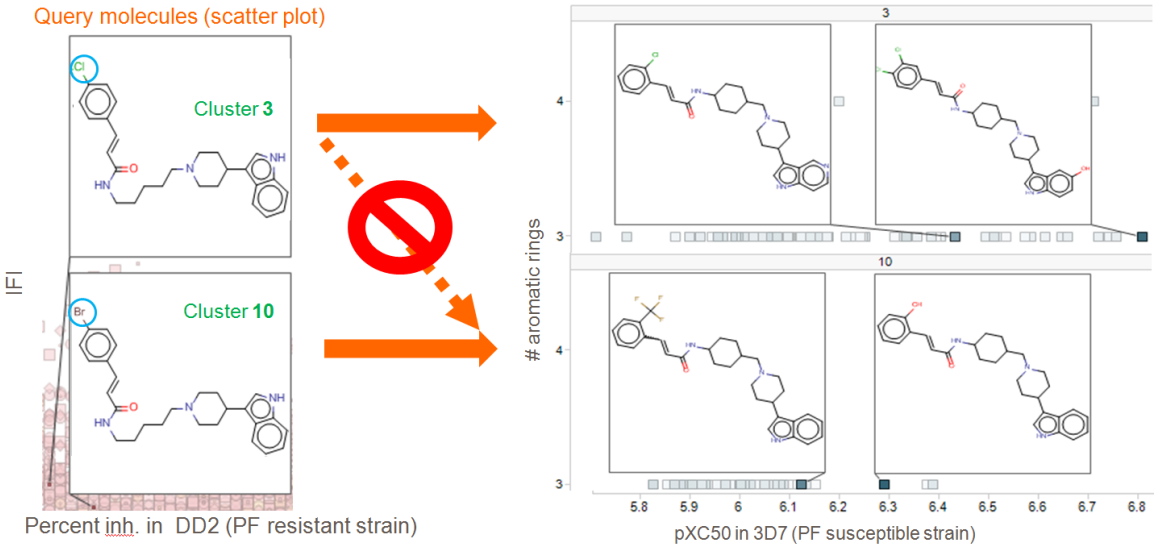

Figure 1: Illustrating one problem with clustering: bifurcation of related molecules. When two molecules of the same chemotype differing by a halogen are split across Complete Linkage Clusters, searches of cluster neighbors for one molecule do not find its analogs in the other cluster, i.e., the two related clusters are not linked.

S6 Qualitative Comparison of Scaffold-Generation Methods and Clustering

Complete-Linkage Clustering: As shown in Figure 1, the defining feature of a partitioning clustering is that every molecule maps to one and only one cluster. Thus if a chemotype is broken up among two or more clusters, using the cluster ID to map Related Molecules can retrieve only neighbors from the same cluster, ignoring the other cluster. This is not ideal for purposes of the visualization and navigation method presented here, as arbitrary neighbors would be excluded depending on how the clustering is defined. Thus we do not advocate the use of clustering, unless it is a fuzzy clustering where all meaningful class memberships a molecule might have are considered.

NCATS R-group tool: As opposed to the clustering method, if any two molecules share a common substructure that meets the standards required

of a scaffold by the NCATS method (e.g., being bordered by rings on each end), then those molecules will be found to contain that shared substructure as a scaffold and their activities will be used to compute aggregate properties for it.

Other Scaffold Generation Methods: Even though another scaffold generation method (represented here by molecular frameworks as implemented in [2]) differed in its implementation details and produced different numbers of scaffolds for the same molecule, it was roughly equivalent in a qualitative sense with regard to the insights obtained during Scaffold Walking. Due to substantial overlap between sets of scaffolds, ring systems responsible for activity of a molecule were generally revealed by either method. However, there were cases where the Frameworks revealed negative information about a fragment being not important for activity that is also useful for a drug discovery scientist. For example, in Figure 2 a substructure is highlighted that is on the aggregate inactive and could be removed or substituted. This insight is not available from SSSR-based scaffolding methods such as the NCATS R-group tool since they don't define or find that fragment as a scaffold.

To summarize, both multiple-scaffold decomposition methods considered in this study, i.e., NCATS R-group Tool and Frameworks give comparable insights when exploring the TCAMS dataset, with some differences stemming from individual substructures that are considered shared scaffolds or not by the individual methods.

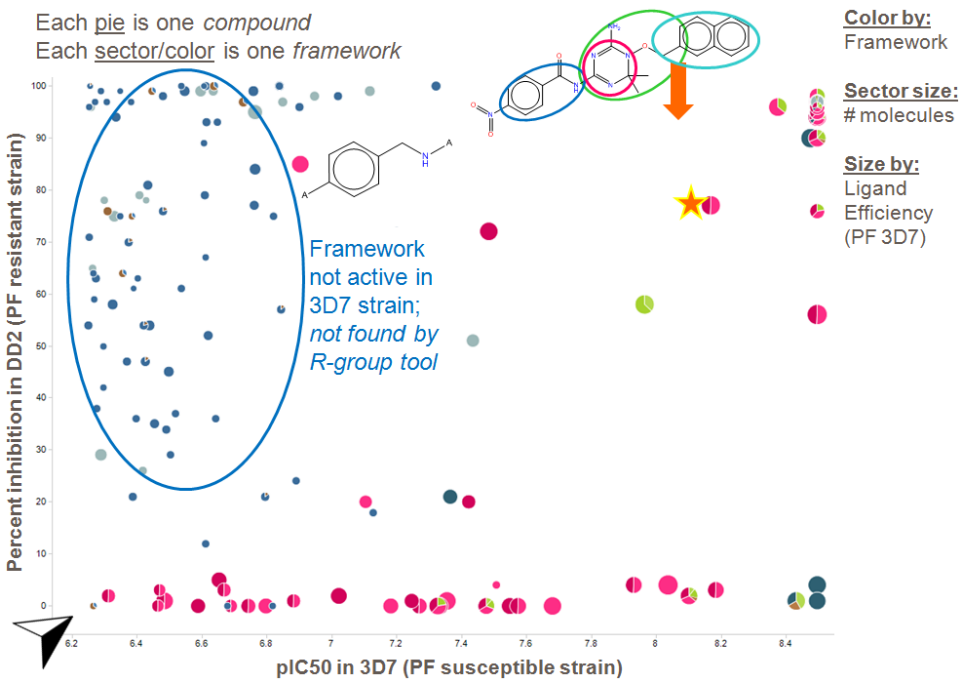

Figure 2: Using Frameworks with the Scaffold Pies visualization. One framework is highlighted that has no equivalent in the NCATS scaffolds, but is shown to reduce activity as related molecules containing it are less active than the parent molecule. The star symbol shows the location of the parent molecule in this Related Molecules plot, and the compass device at the origin shows the direction of favorable properties (+X and +Y axes).

S7 Spotfire alternate visualizations and usability tips

The following additional visualizations have been considered and found useful for NCATS scaffolds and R-groups in analyzing data for GSK projects. They are briefly mentioned below:

1. **Cross-tables** have been used to list scaffold IDs, compute statistics such as min, max and median activity for each, filter by maximum activity to remove wholly inactive scaffolds, and then prioritize the remaining by drilling down into individual properties
2. **Profile plots** are line graphs used as a visual drill-down mechanism to examine a suite of related assays across all compounds containing a scaffold.They typically have compounds on the X-axis, properties on the Y and are colored by assay name. Figure 6(b) in the main paper shows a profile plot on Kinase X data.
3. **Box plots** are used to visualize the distribution of values of an activity or property for separate scaffolds. Scaffold ID is typically on the X-axis and properties on the Y, allowing multiple scaffolds to be compared side-by-side at a deeper level than just their aggregate activities.

A few Spotfire techniques were employed to make the users' experience with our Spotfire files more friendly. These have all been tested in TIBCO Spotfire 7.0.2, the version installed at GSK when this work was being concluded.

- **Structure Visualization:** GSK has a plugin from ChemAxon to visualize structures in tables, labels and tooltips in addition to a dedicated Structure Viewer window. Tooltips were widely employed as they are interactive and allow us to add multiple structure columns, typically a scaffold and the full structure. Labels were used to mark interesting compounds, and structures of cores, R-groups and full molecules depicted in tables helped us create SAR tables directly in Spotfire.
- **Filtering between related tables:** In addition to creating Table Relations among the Molecules, Scaffolds and Related Molecules tables, we often want to display only scaffolds matching selected molecules

of interest, or show only molecules that contain a selected set of scaffolds. This can be achieved by setting the "Filtering on Related Data Tables" to either Include Filtered Rows or Exclude Filtered Out Rows.

- **Tagging:** Used to mark scaffolds selected by the user as being of interest, fulfilling various criteria verified by drilling down into the data behind those molecules.

S8 R code for statistical comparison of scaffold generation methods

The code below expects two files, DDframes.txt (from framework clustering, Method A in the results) and RGD.txt (NCATS R-group tool decomposition, Method B). They both have a column named Compound_ID. DDframes.txt has a column StrucUniqueID which is computed for example using the DenseRank function in Spotfire, assigning a unique numerical ID to each fragment having different Canonical SMILES. RGD.txt has a column SCAFFOLD_ID that is already numerically distinct for each separate scaffold. The code can be generalized for any pair of methods as long as the map from numerical compound IDs to scaffold IDs exists in the input data.

```
#Note: To do the calculations comparing 2 fragmentation methods
#took about 10 minutes of computer time.

#Set working directory to location of the data files
setwd("C:\\Work\\Consulting\\MDR\\Other MDR Issues\\CSC\\FragmentOntologies")
####################
rm(list = ls()) #Erase anything in R's working memory

DDData <- read.table("DDframes.txt", header = T, sep = "\t")
RGData <- read.table("RGD.txt", header = T, sep = "\t")

#Create a list of compounds shared between two datasets.
#Scores will not make sense if we include non-common compounds

DDCompounds <- unique(DDData$COMPOUND_ID)
RGCompounds <- unique(RGData$COMPOUND_ID)

Compounds <- intersect(DDCompounds, RGCompounds)
# Compounds <- as.numeric(as.character(intersect(DDCompounds, RGCompounds)))
N <- length(Compounds)

#Calculate PI's and common proportions for each compound

#Proportions by compound

PropByCompound <- data.frame(CompoundID = rep(NA, N),
     FragA = rep(NA, N),
     FragB = rep(NA, N),
                              Ca = rep(NA, N),
```

```
                                  Cb = rep(NA, N),
                                  IntAB = rep(NA, N),
                                  UnionAB = rep(NA, N),
                                  CommonProp = rep(NA, N),
                                  PIa = rep(NA, N),
                                  PIb = rep(NA, N),
                                  PIaU = rep(NA, N),
                                  PIbU = rep(NA, N),
                                 FragEffA = rep(NA, N),
                                 FragEffB = rep(NA, N))

for (index in 1:N){

  PropByCompound$CompoundID[index] <- Compounds[index]

  MethodABelongsTo <- DDData[DDData$COMPOUND_ID == Compounds[index],
                             "StrucUniqueID"]
  MethodBBelongsTo <- RGData[RGData$COMPOUND_ID == Compounds[index],
                             "SCAFFOLD_ID"]

  PropByCompound$FragA[index] <- length(unique(MethodABelongsTo))
  PropByCompound$FragB[index] <- length(unique(MethodBBelongsTo))

  MethodACompoundCluster <- unique(DDData[DDData$StrucUniqueID %in% MethodABelongsTo,
                                   "COMPOUND_ID"])
  MethodBCompoundCluster <- unique(RGData[RGData$SCAFFOLD_ID %in% MethodBBelongsTo,
                                   "COMPOUND_ID"])

  PropByCompound$Ca[index] <- length(MethodACompoundCluster)
  PropByCompound$Cb[index] <- length(MethodBCompoundCluster)
  PropByCompound$IntAB[index] <- length(intersect(MethodACompoundCluster,
                                                  MethodBCompoundCluster))
  PropByCompound$UnionAB[index] <- length(union(MethodACompoundCluster,
                                                  MethodBCompoundCluster))
  PropByCompound$CommonProp[index] <- PropByCompound$IntAB[index]/
                                      PropByCompound$UnionAB[index]
  PropByCompound$PIa[index] <- PropByCompound$Ca[index]/PropByCompound$UnionAB[index]
  PropByCompound$PIb[index] <- PropByCompound$Cb[index]/PropByCompound$UnionAB[index]
  PropByCompound$PIaU[index] <- 1 - PropByCompound$Cb[index]/PropByCompound$UnionAB[index]
  PropByCompound$PIbU[index] <- 1 - PropByCompound$Ca[index]/PropByCompound$UnionAB[index]
  PropByCompound$FragEffA[index] <- PropByCompound$Ca[index]/PropByCompound$FragA[index]
  PropByCompound$FragEffB[index] <- PropByCompound$Cb[index]/PropByCompound$FragB[index]

}
```

```
#Create output -- averages, quantiles, and histograms

ACP <- mean(PropByCompound$CommonProp, na.rm = T)
APIaU <- mean(PropByCompound$PIaU, na.rm = T)
APIbU <- mean(PropByCompound$PIbU, na.rm = T)
AFragEffA <- mean(PropByCompound$FragEffA, na.rm = T)
AFragEffB <- mean(PropByCompound$FragEffB, na.rm = T)

CP95 <- quantile(PropByCompound$CommonProp, c(0.05,0.25,0.5,0.75, 0.95))
PIaU95 <- quantile(PropByCompound$PIaU, na.rm = T, c(0.05,0.25,0.5,0.75, 0.95))
PIbU95 <- quantile(PropByCompound$PIbU, na.rm = T, c(0.05,0.25,0.5,0.75, 0.95))
FragEffA95 <- quantile(PropByCompound$FragEffA, na.rm = T, c(0.05,0.25,0.5,0.75, 0.95))
FragEffB95 <- quantile(PropByCompound$FragEffB, na.rm = T, c(0.05,0.25,0.5,0.75, 0.95))

ACP
APIaU
APIbU
AFragEffA
AFragEffB

CP95
PIaU95
PIbU95
AFragEffA95
AFragEffB95

write.table(PropByCompound,"PropByCompound.txt",sep="\t",row.name=F,col.name=T)
##############################################################################################

################## Plot ####################
attach(PropByCompound)

hist(FragA,main="FragA")
hist(FragB,main="FragB")
hist(CommonProp,main="CommonProp")

plot(CommonProp~UnionAB)
plot(CommonProp~IntAB)
plot(Ca~Cb)

plot(UnionAB, IntAB)
plot(FragA + FragB, CommonProp)

detach(PropByCompound)
```

References

[1] René Barone and Michel Chanon. A new and simple approach to chemical complexity. application to the synthesis of natural products. *J. Chem. Inf. Comput. Sci.*, 41(2):269–272, 2001.

[2] G. Harper, G. S. Bravi, S. D. Pickett, J. Hussain, and D. V. Green. The reduced graph descriptor in virtual screening and data-driven clustering of high-throughput screening data. *J Chem Inf Comput Sci*, 44(6):2145–2156, 2004.