# RBMRB

*Kumaran Baskaran*

*November 08, 2016*

## RBMRB

BMRB collects, annotates, archives, and disseminates (worldwide in the public domain) the important spectral and quantitative data derived from NMR spectroscopic investigations of biological macromolecules and metabolites. The goal is to empower scientists in their analysis of the structure, dynamics, and chemistry of biological systems and to support further development of the field of biomolecular NMR spectroscopy

RBMRB is a library to fetch NMR chemical shift data directly from BMRB into R environment as a data frame in R. This facilitates access to BMRB data for statistical analysis and data visualization. It is using the BMRB-API to fetch the data from BMRB database.

### Installation

RBMRB library has been developed and tested in R version 3.3.x. It requires the following R packages preinstalled

- **httr** to import data from BMRB web server(version 1.2.1 or later)
- **data.table** to format the imported data into a data frame in R (version 1.9.6 or later)
- **rjson** to deal with BMRB-API (version 0.2.15 or later)
- **ggplot2** to simulate spectra (version 2.1.0 or later)
- **plotly** for interactive graphics in simulated spectra (version 4.5.2 or later)

Users should make sure that the above packages have beed installed correctly with the required versions, before proceeding to RMBRM insallation.

Here is the instruction to install those packages. Open your R and use the following command

```r
installed.packages(c("httr","data.table","rjson","ggplot2","plotly"))
```

Once the necessary packages have been installed, proceed with RMRBM installation. The source file can be downloaded from GitHub

After downloading the source file, use the following command to install RBMRB library

```r
install.packages("~/Downloads/RBMRB_1.0.tar.gz",repos=NULL,type="source")
```

Note: provide the correct path to the downloaded file.

### Usage

RBMRB can be used in a similar way like any other library in R.

```r
library(RBMRB)
```

### Data access

BMRB data can be imported in two ways

- **Entry method** Chemical shift data from single or multiple entries
- **Atom method** Chemical shift data from all entries for a given atom

### Entry method

#### fetch_entry_chemical_shifts:

This function will fetch the 'Atom_chem_shift' loop from a NMR-STAR file for a given entry or a list of entries in CSV format. This function works on both macromolecules and metabolites data base. For metabilites entry ids should have right prefix (example 'bmse000034')

Examples:

```
df1<-fetch_entry_chemical_shifts(15060)
df2<-fetch_entry_chemical_shifts(c(17074,17076,17077))
df2<-fetch_entry_chemical_shifts(c('17074','17076','17077'))
df3<-fetch_entry_chemical_shifts(c('bmse000034','bmse000035','bmse000036'))
```

These data frames have the following columns

```
colnames(df1)
```

```
##  [1] "ID"                          "Assembly_atom_ID"
##  [3] "Entity_assembly_ID"          "Entity_ID"
##  [5] "Comp_index_ID"               "Seq_ID"
##  [7] "Comp_ID"                     "Atom_ID"
##  [9] "Atom_type"                   "Atom_isotope_number"
## [11] "Val"                         "Val_err"
## [13] "Assign_fig_of_merit"         "Ambiguity_code"
## [15] "Occupancy"                   "Resonance_ID"
## [17] "Auth_entity_assembly_ID"     "Auth_asym_ID"
## [19] "Auth_seq_ID"                 "Auth_comp_ID"
## [21] "Auth_atom_ID"                "Details"
## [23] "Entry_ID"                    "Assigned_chem_shift_list_ID"
```

### Atom method

#### fetch_atom_chemical_shifts:

This function will fetch the chemical shift data from all the entries for a given atom. The atom name should be in NMR-STAR atom nomenclature.

Examples:

```
df4<-fetch_atom_chemical_shifts('CG2')
df5<-fetch_atom_chemical_shifts('C9')
```

These data frames have the following columns

```
colnames(df4)
```

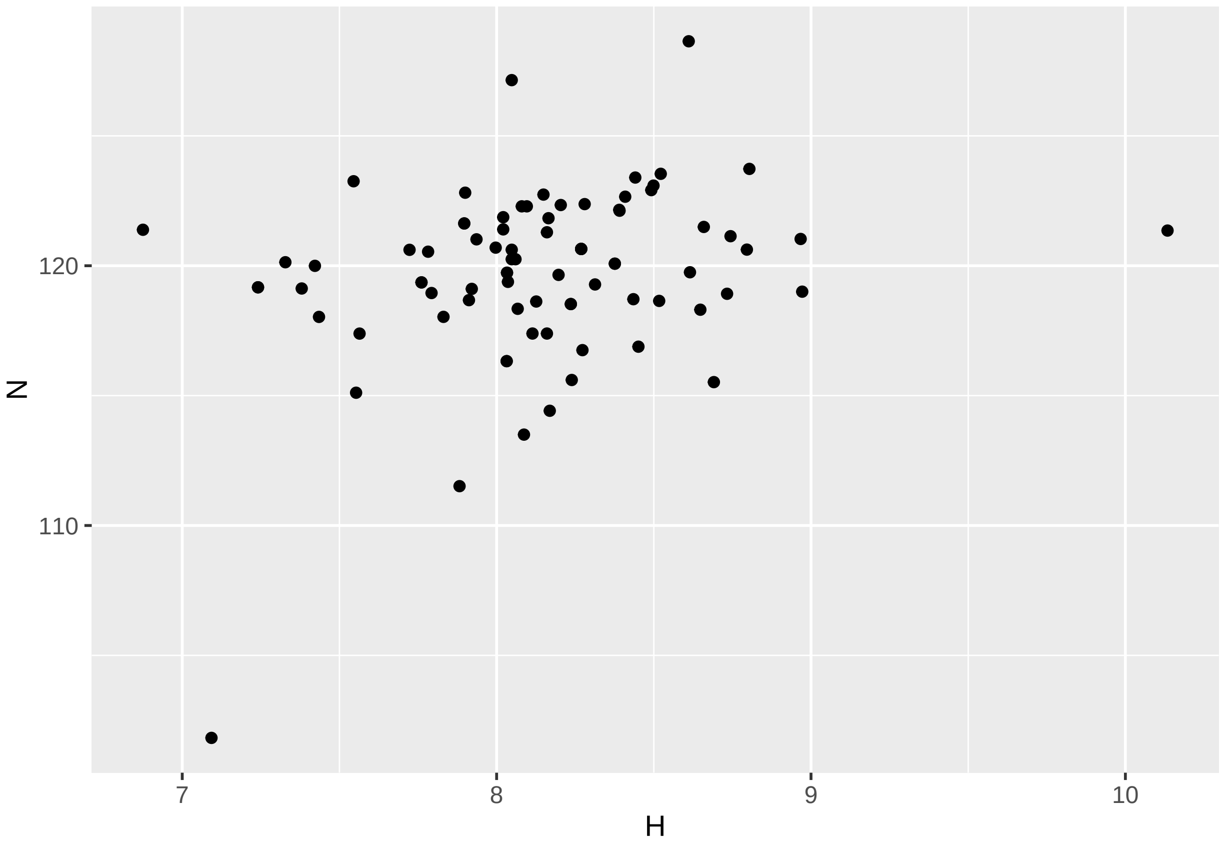
```
##  [1] "Entry_ID"                    "Entity_ID"
##  [3] "Comp_index_ID"               "Comp_ID"
##  [5] "Atom_ID"                     "Atom_type"
##  [7] "Val"                         "Val_err"
##  [9] "Ambiguity_code"              "Assigned_chem_shift_list_ID"
```

## Data manipulation

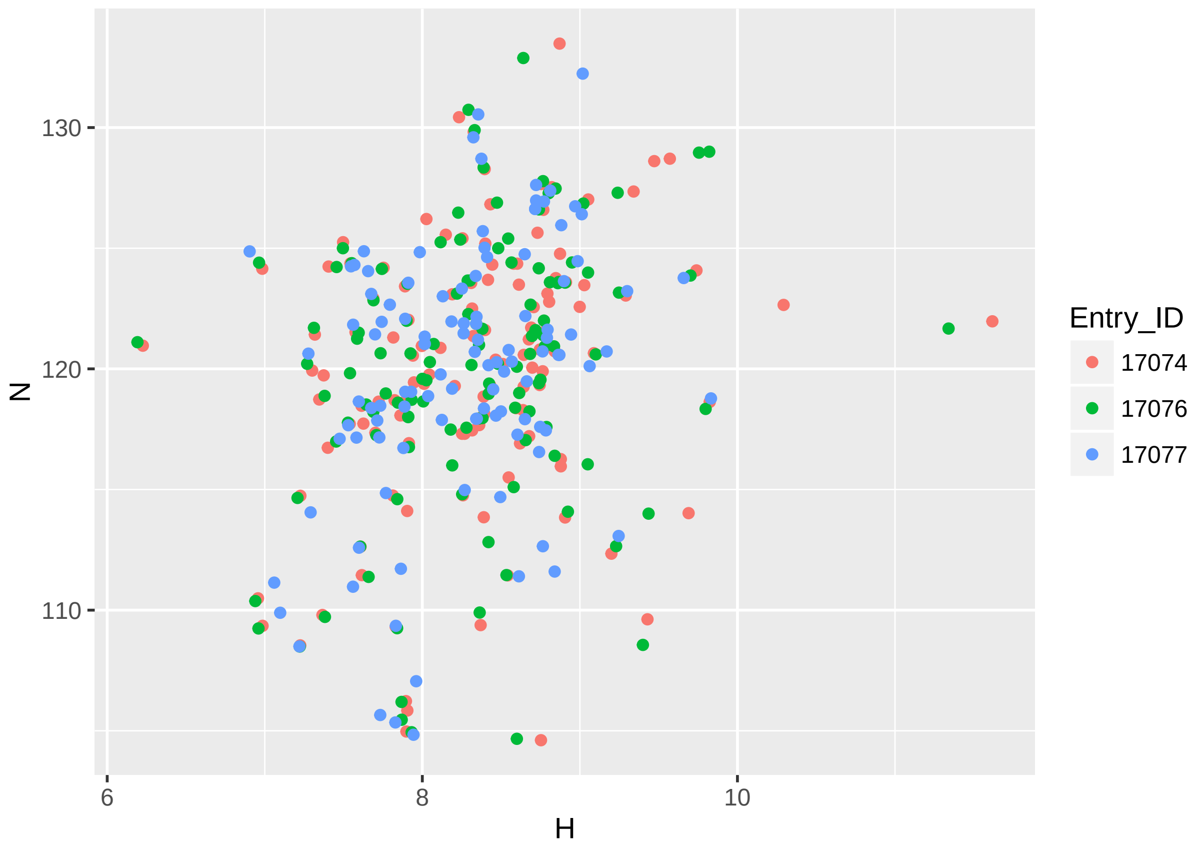
There are few data manipulation functions are availbale to facilitate plotting. ####convert_cs_to_n15hsqc: This function will reformat the chemical shift data frame into a data frame which is easy to plot the N15-HSQC spectrum from the data.

### Examples

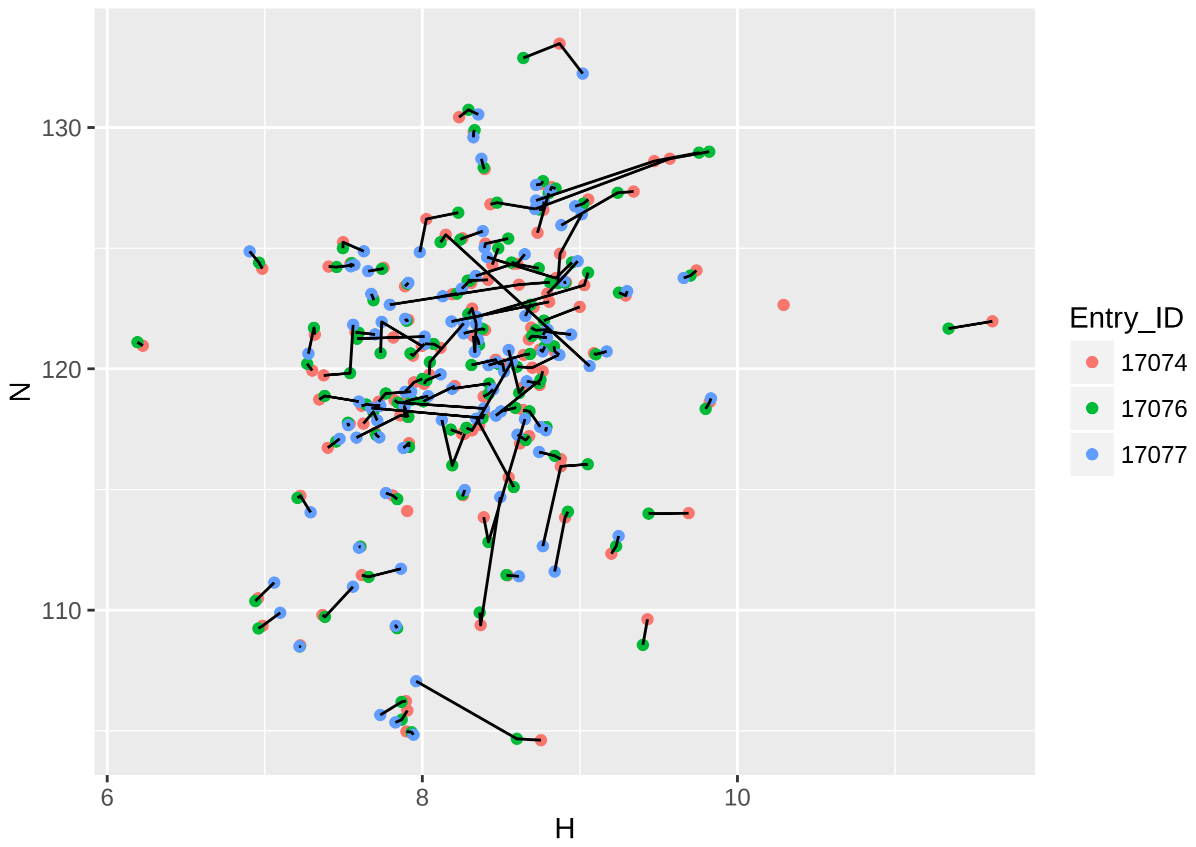
```
n15hsqc1<-convert_cs_to_n15hsqc(df1)
n15hsqc2<-convert_cs_to_n15hsqc(df2)
library(ggplot2)
plt1<-ggplot(n15hsqc1)+geom_point(aes(x=H,y=N))
plt1
```



```
plt2<-ggplot(n15hsqc2)+geom_point(aes(x=H,y=N,color=Entry_ID))
plt2
```



```
plt3<-ggplot(n15hsqc2)+geom_point(aes(x=H,y=N,color=Entry_ID))+geom_line(aes(x=H,y=N,group=Comp_index_II
plt3
```

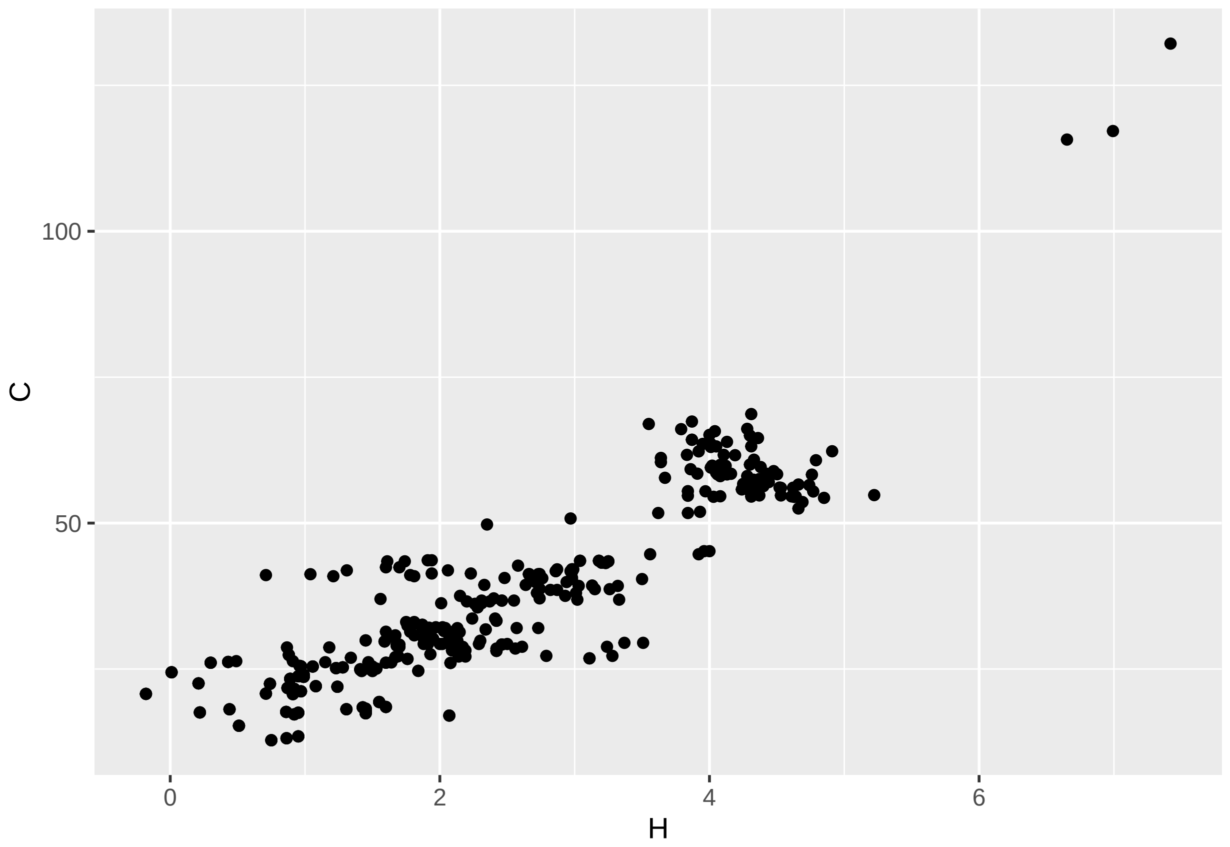


**convert__cs__to__c13hsqc:**

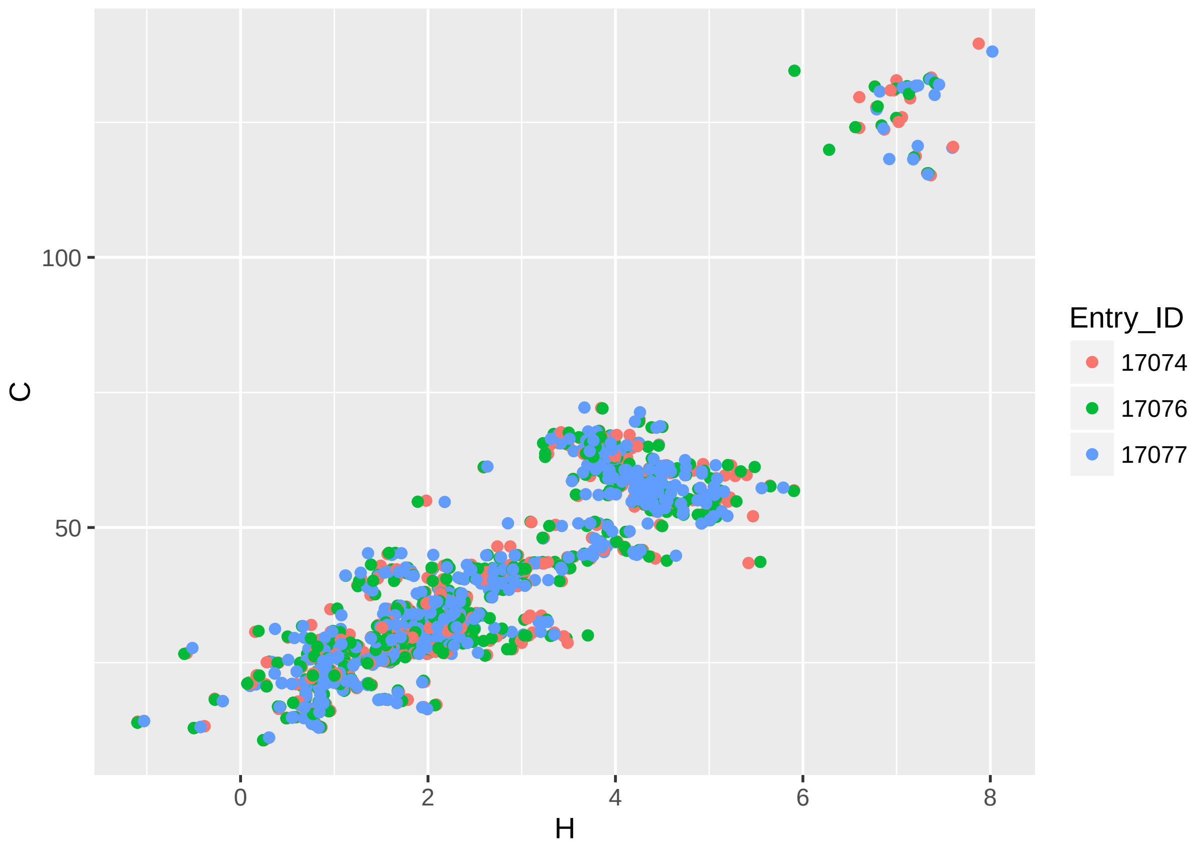
This function will reformat the chemical shift data frame into a data frame which is easy to plot the C13-HSQC spectrum from the data.

**Examples**

```
c13hsqc1<-convert_cs_to_c13hsqc(df1)
c13hsqc2<-convert_cs_to_c13hsqc(df2)
library(ggplot2)
plt1<-ggplot(c13hsqc1)+geom_point(aes(x=H,y=C))
plt1
```



```
plt2<-ggplot(c13hsqc2)+geom_point(aes(x=H,y=C,color=Entry_ID))
plt2
```

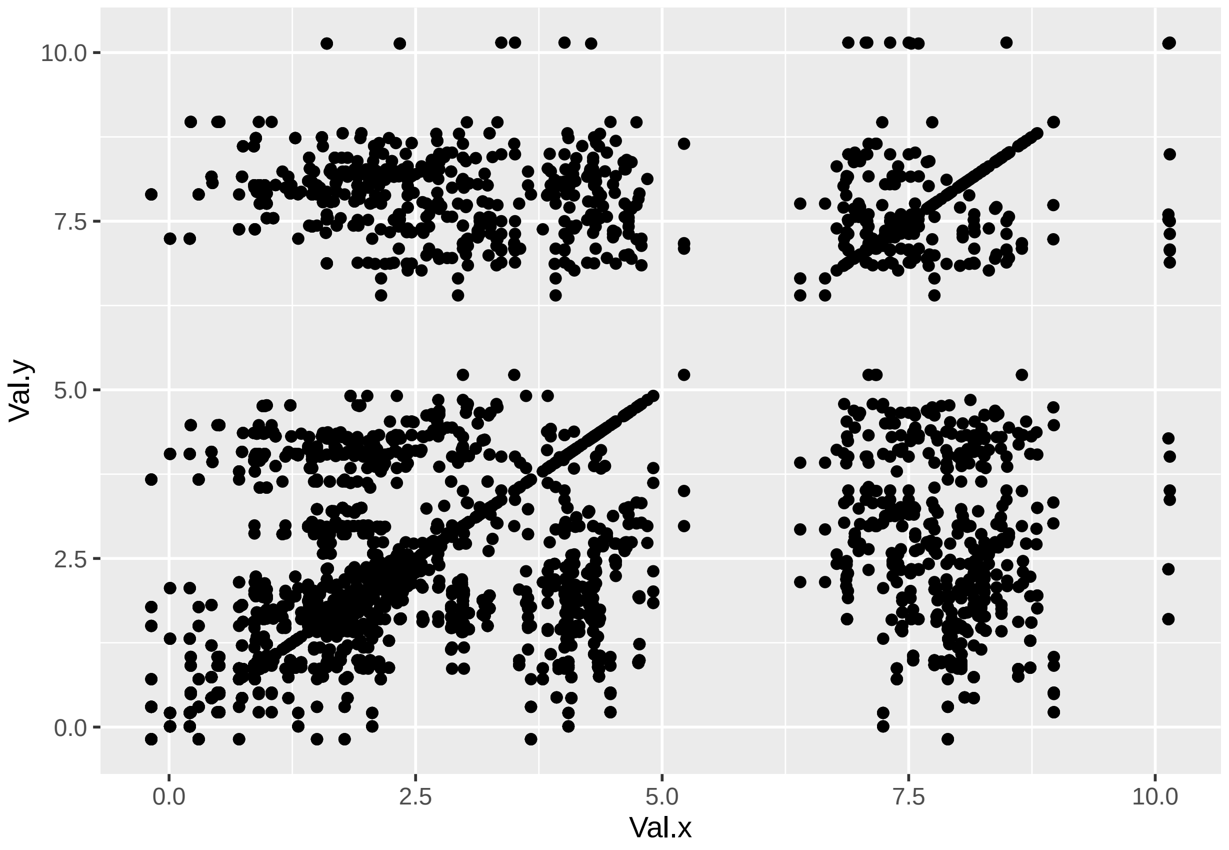


**convert_cs_to_tocsy:**

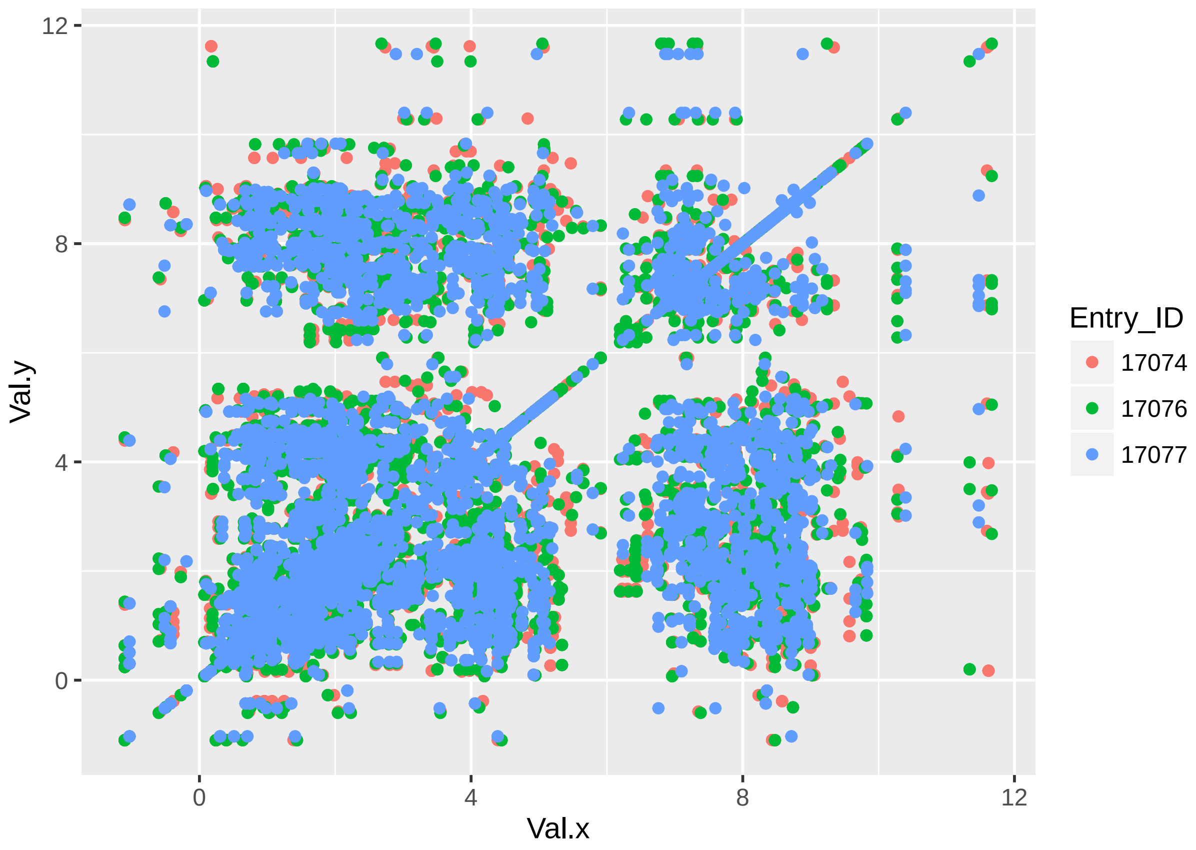
This function will reformat the chemical shift data frame into a data frame which is easy to plot the TOCSY spectrum from the data. Note : Since both dimensions have protein chemical shifts, the columns are named as Val.x and Val.y

**Examples**

```
tocsy1<-convert_cs_to_tocsy(df1)
tocsy2<-convert_cs_to_tocsy(df2)
library(ggplot2)
plt1<-ggplot(tocsy1)+geom_point(aes(x=Val.x,y=Val.y))
plt1
```



```
plt2<-ggplot(tocsy2)+geom_point(aes(x=Val.x,y=Val.y,color=Entry_ID))
plt2
```



**filter_residue:**

This function will filter the data frame and remove all non standard amino acids. The data frame should contain the amino acid information in the Comp_ID column. ####Examples

```
df6<-fetch_atom_chemical_shifts('CG2')
df7<-filter_residue(df6)
```

## Data visualization