GaDGeT: GDGT calculations simplified - User manual

Table of Contents

1. Introduction

Welcome to the GaDGeT GDGT-data processing script User Guide! The "GaDGeT.R" script is the main part of the GaDGeT software, designed to facilitate the calculation of glycerol dialkyl glycerol tetraethers (GDGTs) fractional abundances and indices. The software calculates 107 published fractional abundance-types, environmental and climate proxies and indices, for five GDGT groups: branched glycerol dialkyl glycerol tetraethers (brGDGTs, 21 compounds), isoprenoid glycerol dialkyl glycerol tetraethers (isoGDGTs, 6 compounds), hydroxy glycerol dialkyl glycerol tetraethers (OHGDGTs, 4 compounds), glycerol monoalkyl glycerol tetraethers (GMGTs, 7 compounds), and isoprenoid glycerol dialkyl diethers (GDDs, 5 compounds).

The software aims to streamline data analysis and provide organized results within the GaDGeT framework. Researchers are free to use it under the CC BY 4.0 license, provided that the appropriate citation of "Schneider and Castañeda, (2024)" is given (specified in the GitHub repository). This software is exclusively based on the open-source statistical language R and is best used with the integrated development environment (IDE) "R-Studio".

2. Software Download

2.1 R and RStudio

Please find further information about R and the download links here: https://www.r-project.org/. To learn more about and downloading RStudio, follow this link: https://posit.co/products/open-source/rstudio/.

2.2 GaDGeT

You can download the software from the GitHub repository at the following link: GaDGeT GitHub Repository (https://github.com/brGDGTs/GaDGeT). Please download and locally store the software and review the README file for further information.

3. Data Preparation

Before running the software, ensure your data files (you can add files for as many datasets as you want, the software will walk through them) are stored in the "Input" folder within the GaDGeT working directory (directory structure in Table 1). The script automatically identifies and reads these files, creating a list of datasets crucial for subsequent processing steps within the GaDGeT software environment. To maintain data integrity, please refrain from modifying the header (column names) as provided in the template. The safest practice is to copy and paste all values into their respective columns in the template files without altering the headers in either an .xlsx or .csv file (both templates provided as “Test-data-csv.csv”, and “Test-data-excel.xlsx”). If specific compound groups or compounds were not measured or are below detection limits, leave the columns in the file and enter 'NA' in the respective rows (it is acceptable for entire columns to be filled with 'NA').

4. Running GaDGeT

Now it's time to initiate GaDGeT by double-clicking on the "GaDGeT.R" file located in the main directory. You can find comprehensive information about the script's structure, including what it calculates and its dependencies on function-call files, in the overview section within the script.

4.1 Preparing Your Workspace

Before diving into data analysis and calculations, it's crucial to set up your workspace correctly. To ensure a smooth process, highlight the entire script (e.g., Ctrl+A) and run the entire code (e.g., Ctrl+Enter). Here is a short explanation of what the script will be doing.

1. **Clean Workspace**: Start with a clean R workspace to prevent any interference from previous sessions.
2. **Clear Console Entries**: Remove any existing console entries to maintain a clean working environment.
3. **Close Plot Windows**: Close any open plot windows to avoid visual clutter.
4. **Set Working Directory**: Ensure that the script knows where to find your data by setting the working directory to the folder containing your data files within the GaDGeT environment.
5. **Install and Load Packages**: Install and load the necessary R packages (e.g., "stringr", "readxl", “readr”) to access essential functions.
6. **Load Custom Functions**: Import custom functions from separate function files from the “Functions” folder to extend the script's capabilities, all integrated into the GaDGeT suite.
7. **Loop through Input files**: The script loops through all input files and calculates all the available calculations for each input file.
8. **Write results**: The script writes all the results in separate subdirectories and files within the “Output” subdirectory (Table 1). More details in chapter 3.2 – 3.4.

4.2 Fractional Abundance Calculations

The script calculates fractional abundances (FAs) for various types of compounds within the GaDGeT framework:

- **brGDGTs**: Branched glycerol dialkyl glycerol tetraethers

- **isoGDGTs**: Isoprenoid glycerol dialkyl glycerol tetraethers
- **OHGDGTs**: Hydroxy glycerol dialkyl glycerol tetraethers
- **GMGTs**: Glycerol monoalkyl glycerol tetraethers

Multiple types of FAs are computed for each compound. These results are then organized and saved as separate CSV files within the GaDGeT "Output" directory, ensuring that your fractional abundance data is readily available for further analysis and visualization within the GaDGeT suite. The results are saved as CSV files in the GaDGeT "Output > SAMPLE > FAs" directory. A single .csv file is generated per GDGT-type. The file tagged with "FAs_brGDGTs_" contains FAs for the 5Me and 6Me isomers (15 compounds), while the file tagged "FAs_brGDGTs_7Me_" includes FAs for brGDGTs with the 7Me isomers (21 compounds) as well. Additionally, for the brGDGTs, there is a folder that calculates different "FA-groups" as described and proposed in Raberg et al. (2021).

4.3 Index Calculations

The script calculates various indices for the compounds within the GaDGeT software environment, including:

- **brGDGTs**: Branched glycerol dialkyl glycerol tetraethers (52 indices)
- **isoGDGTs**: Isoprenoid glycerol dialkyl glycerol tetraethers (17 indices)
- **OHGDGTs**: Hydroxy glycerol dialkyl glycerol tetraethers (12 indices)
- **GMGTs**: Glycerol monoalkyl glycerol tetraethers (5 indices)
- **GDDs**: Isoprenoid glycerol dialkyl diethers (3 indices)

For details about the indices, please refer to the tables 1-6 in Schneider and Castañeda (2024). Datasets are combined with calculated indices, and the results are saved as CSV files in the GaDGeT "Output > SAMPLE > GDGT-INDICES" directory. A single .csv file is generated per GDGT-type.

4.4 Concentration Calculations

Concentration calculations are based on an internal standard with known amounts as provided by the user in the input file(s) ("IS_Area", "IS_Amount"). The script computes concentration factors, amounts, and concentrations for the compounds. Results are stored as CSV files, separately for amounts and concentrations, within the GaDGeT "Output" directory. These concentration values are crucial for understanding the relative abundance of compounds in your samples and can be further explored and visualized within the GaDGeT suite.

5. Troubleshooting

If you encounter any issues or errors while running the script within the GaDGeT environment, consider these troubleshooting steps:

- **Read console information:** Some processes, such as wrong or missing column names are directly printed to the console in RStudio.
- **Close Input/Output Files:** Ensure that none of the input or output files (e.g., when investigating a previous run) are open in another program (e.g., Excel or a text editor), as this can prevent the GaDGeT script from accessing or writing to the files properly.

- **Data Location**: Verify that your data files are correctly placed in the "Input" folder of the GaDGeT working directory. And make sure that the structure, and header of the spreadsheet template did not change.
- **Package Installation**: Ensure that the required R packages are installed and loaded correctly within the GaDGeT suite. Make sure that you use an updated R and Rstudio version (R v3.5 or higher).
- **Working Directory**: Double-check the working directory to make sure it points to the folder with your data files within the GaDGeT framework. If suspected that the working directory may be an issue, close the RStudio session and R, and double click the "GaDGeT.R"-script again, this should solve this problem.
- **Error Messages**: Examine any error messages in the R console for clues about the nature of the problem.
- **Restart RStudio and R:** If the problem persists, try closing and reopening RStudio and R, and restarting your computer. In some cases, this alone can resolve issues.
- **Reinstall R and RStudio:** As a last resort, if problems continue, consider reinstalling R and RStudio to ensure there are no underlying installation issues.
- **Contact the author**: as a final resort you may contact the author at tobiaschnei@gmail.com, or www.drtobiasschneider.com.

6. Conclusion

Congratulations! You've successfully used the GaDGeT Software to process your brGDGTs, isoGDGTs, OHGDGTs, GMGTs, and GDDs data. The script has organized your results in the GaDGeT "Output" directory, making it convenient for further analysis. If wished, you may use the "basic plot" script to plot a ternary diagram for example.

7. Script Description and User Instructions

The GaDGeT Script contains the main routines for calculating different kinds of fractional abundances, indices, amounts, and concentrations of GDGTs in geological samples.

Step by Step Instructions for using GaDGeT

1. Please avoid changing folder names or moving files from the "Functions" folder as this could disrupt the script's functionality.
2. Do not modify the files and their names in the "Functions" folder. These files contain essential functions that GaDGeT relies on for its calculations. If you would like to add different index-calculations to the software, please follow the instructions in Chapter 9.
3. Keep the basic folder structure in the main folder "GaDGeT": "Functions", "Input", "Output", "GaDGeT.R"
4. Provide your data in .xlsx or .csv format within the "Input" folder.
5. Use the provided template files (.xlsx; .csv) and do not alter the header names. Fill any empty cells with "NA". Do not delete any columns, this will interrupt the software run.
6. You can add multiple files to the "Input" directory (all in the exact same format), and GaDGeT will process them automatically ("loop through them") and will automatically provide separate output directories for each input file.
7. Start the script by double-clicking "GaDGeT.R" in the GaDGeT main folder.
8. Highlight all the code (e.g., Ctrl + A) and run it (e.g., Ctrl + Enter).

9. The subdirectories and files will be written into the “Output” folder (more information above)
10. That's it—congratulations on processing your data!

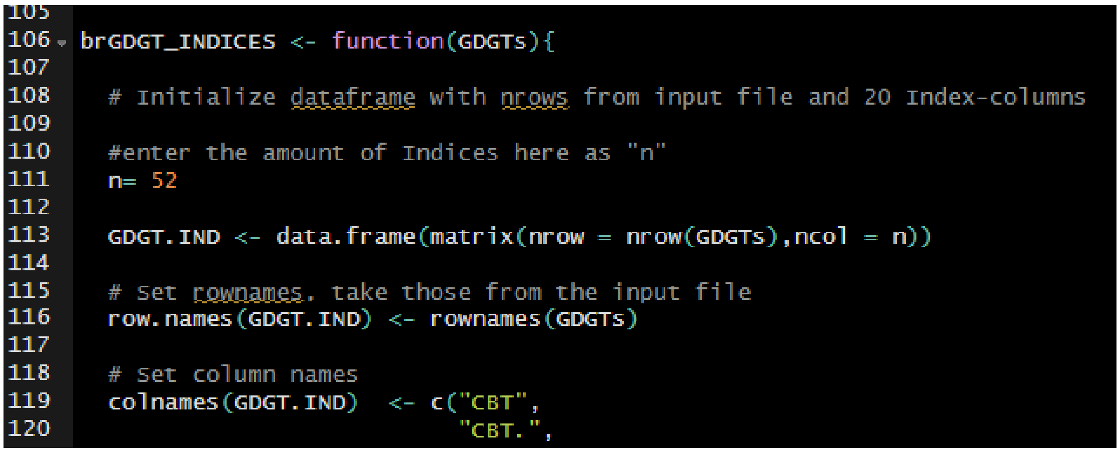
8. Adding Index calculations

There is another GDGT-ratio I want to calculate, can I add it to GaDGeT?

Yes! GaDGeT is designed to be flexible, allowing for the addition of new calculations as more laboratories measure and study GDGTs. As new proxies and calibrations are proposed, the ability to integrate additional equations is essential. Below, we demonstrate how to add new calculations to GaDGeT, using the (Isomer Ratio) IRpenta and IRhexa equations from Sinninghe Damsté et al. (2016) (equations 8 and 9 from their paper). Follow these steps to ensure a smooth integration:

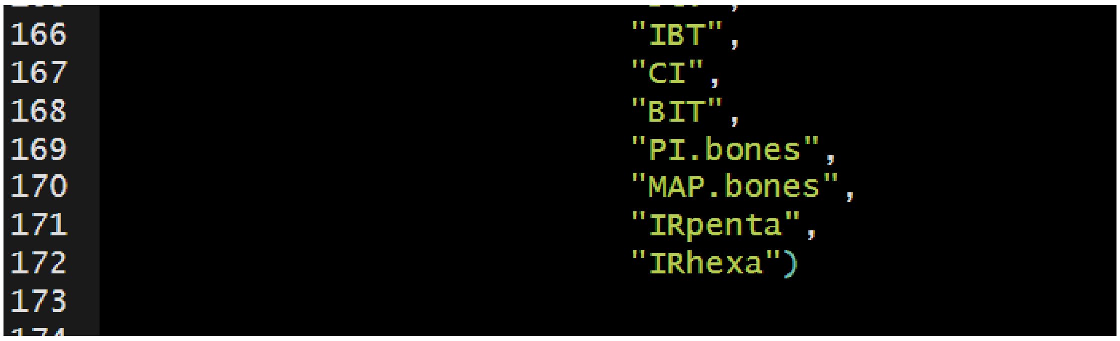
1. **Open the target Functions file**. Locate the "Functions" folder in the GaDGeT directory. In this folder, you’ll find files for each compound type (iso, branched, OH, etc.). Since the new equations are based on branched GDGTs, open the file named “brGDGT_INDEX-calculation_Functions.R” by double-clicking on it.

2. **Initialize the New Index:** Scroll down to the line that says “#enter the number of Indices here as "n"” (approximately at line 110). Update the line below it, changing n = 52 to n = 54, as you will be adding two more indices.



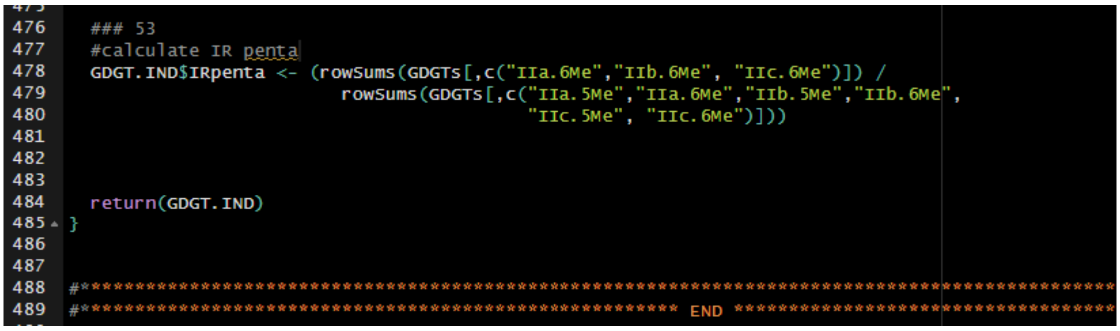
```
106  brGDGT_INDICES <- function(GDGTs){
107
108    # Initialize dataframe with nrows from input file and 20 Index-columns
109
110    #enter the amount of Indices here as "n"
111    n= 52
112
113    GDGT.IND <- data.frame(matrix(nrow = nrow(GDGTs),ncol = n))
114
115    # Set rownames, take those from the input file
116    row.names(GDGT.IND) <- rownames(GDGTs)
117
118    # Set column names
119    colnames(GDGT.IND)  <- c("CBT",
120                             "CBT.",
```

3. **Update the Column Names List:** Scroll to the last entry in the column names list (around line 170) where you see colnames(GDGT.IND) and the final entry ‘MAP.bones’. Add two more entries: "IRpenta" and "IRhexa". Be sure to include a comma after each entry and keep the closing parenthesis “)” at the end.



```
166                              "IBT",
167                              "CI",
168                              "BIT",
169                              "PI.bones",
170                              "MAP.bones",
171                              "IRpenta",
172                              "IRhexa")
173
```

4. **Add the New Calculations:** Scroll to the end of the file, just before return(GDGT.IND) (around line 478). Initialize the first calculation for IRpenta by adding this line of code:
   ```
   GDGT.IND$IRpenta <- (rowSums(GDGTs[,c("IIa.6Me","IIb.6Me", "IIc.6Me")]) /
                        rowSums(GDGTs[,c("IIa.5Me","IIa.6Me","IIb.5Me","IIb.6Me",
                        "IIc.5Me", "IIc.6Me")]))
   ```
   Since R is case-sensitive, ensure that the variable name is written exactly as initialized.



```
476      ### 53
477      #calculate IR penta
478      GDGT.IND$IRpenta <- (rowSums(GDGTs[,c("IIa.6Me","IIb.6Me", "IIc.6Me")]) /
479                             rowSums(GDGTs[,c("IIa.5Me","IIa.6Me","IIb.5Me","IIb.6Me",
480                                              "IIc.5Me", "IIc.6Me")]))
481
482
483
484      return(GDGT.IND)
485    }
486
487
488    #*****************************************************************************************************
489    #********************************************************** END ***************************************
```

5. **Repeat the Step for IRhexa:** Add the second calculation for IRhexa:
   ```
   GDGT.IND$IRhexa <- (rowSums(GDGTs[,c("IIIa.6Me","IIIb.6Me","IIIc.6Me")])/
                       rowSums(GDGTs[,c("IIIa.5Me","IIIa.6Me","IIIb.5Me","IIIb.6Me",
                       "IIIc.5Me", "IIIc.6Me")]))
   ```

6. **Save and Close:** After adding the equations, save the file and close RStudio.

7. **Run GaDGeT:** You can now run GaDGeT by double-clicking the GaDGeT.R file and following the standard procedure outlined earlier in the manual.

These steps are also applicable for adding other indices to different function scripts. The structure and process follow the same principle. Note that the position where you add the new function will be reflected in the output .csv file, ensuring that the order of the indices remains consistent.

9. Example Walkthrough

To help you get started, we've provided a sample dataset and a walkthrough to demonstrate the process:

1. **Download the Sample Dataset:** Sample Dataset
2. **Prepare the Data:** Open the sample dataset in Excel and familiarize yourself with the structure.
3. **Run the Script:** Follow the instructions above to process the sample data using GaDGeT.
4. **Check the Output:** After running the script, compare your output with the expected results provided in the repository.

10. Figures and Tables

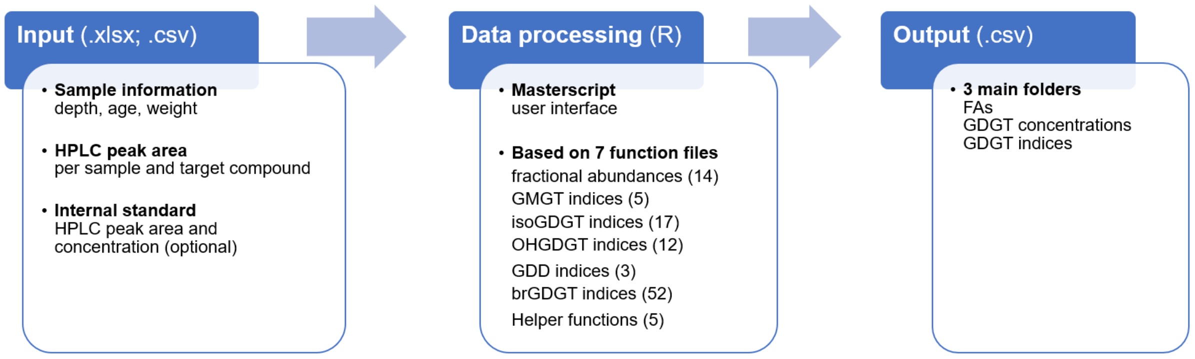


Fig. 1 Workflow diagram illustrating the data processing pipeline of the GaDGeT software for GDGT analysis. The process starts with the input of sample information in an .xlsx; .csv format, which includes details like depth, age, weight, HPLC peak areas, and optional internal standard concentrations. The data is then processed in the R environment, utilizing a master script and seven supporting function files to compute fractional abundances and various GDGT indices. The output is generated in a .csv format, organized into three primary folders: FAs, GDGT concentrations, and GDGT indices.

Table 1. Overview of the folder structure.

Folder Name	Contents	Description
GaDGeT (Main Folder)	- GaDGeT.R (Script)	Main folder containing the GaDGeT script to execute calculations.
	- Functions (Folder)	Contains essential function files, such as FA and Index calculations. For adding other index calculations, refer to the user manual's step-by-step guide.
	- Input (Folder)	Where data files (.xlsx or .csv) should be placed. Ensure the specific structure is followed, and use the header names as provided in the templates, e.g., “Test-data-csv.csv”
	- Output (Folder)	Where results and subdirectories will be written.
Output (Subdirectory)	- SampleName (Subdir)	Subdirectory for a specific dataset. Each dataset will have its own directory.
SampleName (Subdir)	- FAs (Subdir)	Subdirectory containing fractional abundance (FA) results.
	- GDGT-CONCENTRATIONS (Subdir)	Subdirectory containing concentration results.
	- GDGT-INDICES (Subdir)	Subdirectory containing index calculation results.
FAs (Subdir)	- brGDGTs (Subdir)	Subdirectory for various brGDGTs FA results.
	- .csv (File)	The FAs are contained in a .csv file for each GDGT type.
brGDGTs (Subdir)	- multiple .csv (Files)	Subdirectory containing FA calculations for brGDGTs as described in Raberg et al. (2021).
GDGT-CONCENTRATIONS (Subdir)	- …AMOUNT….csv (File)	Contains amount (absolute value relative to the internal standard area size) results.
	- … CONC….csv (File)	Contains concentration (per dry weight) results, in separate files if inputs were provided.
GDGT-INDICES (Subdir)	- …BR….csv (File)	Contains index files for brGDGTs.
	- …GMGT….csv (File)	Contains index files for GMGTs.
	- …iso….csv (File)	Contains index files for isoGDGTs.
	- …OH….csv (File)	Contains index files for OHGDGTs.
	- …GDD….csv (File)	Contains index files for GDDs.

10. R markdown file

You can find detailed explanations of the different code chunks and their functions in the accompanying RMarkdown file. However, to run the GaDGeT script, simply highlight all the code in the GaDGeT.R file and execute it. The script will automatically handle the rest of the process.

GaDGeT: GDGT Calculations Simplified

Tobias Schneider and Isla Castañeda

September 2024

Contents

1 Overview

This script calculates fractional abundances and GDGT indices from brGDGT and other GDGT data. It processes data from .csv or .xlsx files in the `Input` directory and outputs CSV files with the calculated results into the `Output` directory.

- **Author:** Tobias Schneider
  Date: December 5, 2020
  Last Modification: September 09, 2024
  Contact: tobiaschnei@gmail.com, www.drtobiasschneider.com

For more details, refer to the corresponding article:

- **Reference:** Schneider, T., & Castaneda, I.S. (2024). “GaDGeT – GDGT calculations simplified: an adaptable R-toolbox for rapid GDGT index calculations.” Organic Geochemistry. DOI: xxxx/yyyy

2 Requirements

- **R Version:** 3.5 or above
- **Packages:** `stringr`, `readxl`, `readr`

3 Input Data

The user needs to provide either a .csv or .xlsx Excel file containing the different GDGT peaks in the `Input` directory. If calculations for amounts and concentrations are required, the extracted dry sample weight (EXTRACTEDSAMPLEWEIGHT), and the area and amount added of the internal standard (IS) must also be provided. The script will not discriminate between units, the user is required to keep track of the units (e.g., mg, g, ug, kg).

- Follow the structure of the example file provided.
- Ensure the sheet is named "GDGTs" when using ".xlsx" files.
- Do not change the header names; the script extracts all necessary info from them.
- You can add multiple files to the "Input" directory; the script will automatically process all files in this directory and output it in separate output directories (folders).

4 Running the Script

If open, close R and RStudio. Then start the script by double-clicking the `GaDGeT.R` file. Do not change folder names or move any files from the `Functions` folder.

5 Script Structure

5.1 Workspace Preparation

5.1.1 Clear the workspace, console, and close all graphics

```r
rm(list = ls(all = TRUE))
cat("\014")  # Clear console
graphics.off()  # Close all graphics windows
```

5.1.2 Set working directory

```r
workingdir <- getwd() # Use default working directory
#uncomment below and add the workingdir mannually
#workingdir<-"C:/Users/..."

setwd(workingdir)
```

5.1.3 Load required packages

```r
packs <- c("stringr", "readxl", "readr")

# Install missing packages
install.packages(setdiff(packs, installed.packages()[, "Package"]))

# Load the packages
invisible(lapply(packs, library, character.only = TRUE))
```

5.1.4 Load custom functions

```r
# List of function files to source
function_files <- c("GDGT_FA-calculation_Functions.R",
                    "brGDGT_INDEX-calculation_Functions.R",
                    "isoGDGT_INDEX-calculation_Functions.R",
                    "OHGDGT_INDEX-calculation_Functions.R",
                    "GMGT_INDEX-calculation_Functions.R",
                    "GDD_INDEX-calculation_Functions.R",
                    "Helper_Functions.R")

# Source all files in the list
invisible(lapply(function_files, function(file) source(file.path("Functions", file))))
```

5.2 Data Preparation

5.2.1 Get list of Excel files in the 'Input' directory

```r
# Get list of Excel files in the 'Input' directory
GDGT.files <- list.files(path = paste0(workingdir, "/Input/"), pattern = "\\.(xlsx|csv)$")

if (length(GDGT.files) == 0) {
  stop("No input files found in the 'Input' directory.
       Please add input files according to the template.")
}
```

5.2.2 Read and process files and data

```r
# Initialize list for data compilation
files_info <- read_and_process_files(GDGT.files, workingdir)# read in datafiles
                                                            # based on the helper function

data.sets <- files_info$data_sets
data.sets.names <- files_info$data_sets_names #lists all available datasets
```

5.3 Main Processing Loop {GaDGeT} Starts here

```r
# Set global precision for numeric outputs
options(digits = 15)

# build a loop to browse through all files and calculate all the FAs for all Excel sheets

for(f in 1:length(data.sets.names)){

# Choose the file according to the list provided above
data.sets.name <- data.sets.names[f]
```

5.3.1 Prepare data

```r
# extract data from list, convert it into numeric matrix for calculations
GDGT.temp <- matrix(unlist(data.sets[[f]]),ncol = ncol(data.sets[[f]]), byrow = F)
GDGT.temp <- mapply(GDGT.temp, FUN = as.numeric)
GDGT.temp <- matrix(GDGT.temp,ncol=ncol(data.sets[[f]]))

# label columns and rows of new matrix
rownames(GDGT.temp) <- unlist(data.sets[[f]][,1])
colnames(GDGT.temp) <- colnames(data.sets[[f]])
```

5.3.2 Separate compounds

```r
# === Select Relevant Data Columns ===

# Define the sets of compounds to extract
brGDGTs_cols <- c("IIIa.5Me", "IIIa.6Me","IIIa.7Me", "IIIb.5Me", "IIIb.6Me","IIIb.7Me",
                  "IIIc.5Me", "IIIc.6Me","IIIc.7Me", "IIa.5Me", "IIa.6Me","IIa.7Me",
                  "IIb.5Me", "IIb.6Me","IIb.7Me", "IIc.5Me", "IIc.6Me","IIc.7Me",
                  "Ia", "Ib", "Ic")

GDGTs_cols   <- c("GDGT.0", "GDGT.1", "OH-GDGT.0", "GDGT.2", "OH-GDGT.1", "2OH-GDGT.0",
                "GDGT.3", "OH-GDGT.2", "GDGT.4","GDGT.4.")

GMGTs_cols   <- c("H1048", "H1034a", "H1034b","H1034c", "H1020a", "H1020b", "H1020c")

GDDs_cols    <- c("isoGDD0", "isoGDD1","isoGDD2","isoGDD3", "isoGDDCren")

IS_cols      <- c("Label", "DEPTH", "AGE", "EXTRACTEDSAMPLEWEIGHT", "IS_AREA",
                  "IS_AMOUNT")


# column check, are all required columns available? -If not, then remind the user
# to strictly using the headers as provided in the template.

if (!all(c(brGDGTs_cols,GDGTs_cols,GMGTs_cols,GDDs_cols, IS_cols)
         %in% colnames(GDGT.temp))) {
```

```r
  stop("The input file does not contain the required columns. Please use the column
       header names as provided in the template.")
}


# === Extract relevant data, filling NAs with 0 ===

brGDGTs <- GDGT.temp[, brGDGTs_cols, drop = FALSE]
GDGTs <- GDGT.temp[, GDGTs_cols, drop = FALSE]
GMGTs <- GDGT.temp[, GMGTs_cols, drop = FALSE]
GDDs <- GDGT.temp[, GDDs_cols, drop = FALSE]
IS <- GDGT.temp[, IS_cols, drop = FALSE]

brGDGTs[is.na(brGDGTs)] <- 0
GDGTs[is.na(GDGTs)] <- 0
GMGTs[is.na(GMGTs)] <- 0
GDDs[is.na(GDDs)] <- 0
IS[is.na(IS)] <- 0

#compile the compounds for concentration calculation later on
GDGTs.conc <- cbind(GDGTs,brGDGTs)
```

5.3.3 Create storage folders and directories in output

```r
# === Create Output Directories ===

base_dir <- paste0(workingdir, "/Output/", data.sets.name)
create_dir(base_dir)

# Directories for outputs
DirFA.br <- paste0(base_dir, "/FAs/brGDGTs/")
DirFA <- paste0(base_dir, "/FAs/")
DirIND <- paste0(base_dir, "/GDGT-INDICES/")
DirCONC <- paste0(base_dir, "/GDGT-CONCENTRATIONS/")

# Create directories if they do not exist
create_dir(DirFA.br)
create_dir(DirFA)
create_dir(DirIND)
create_dir(DirCONC)
```

5.4 Fractional Abundances

5.4.1 brGDGTs

The functions are stored in the "Functions" folder. Different fractional abundances according to Raberg et al. (2021) are being calculated

```r
# calculate the FA following 1. brGDGT_FA
brGDGT.FA            <- brGDGT_FA(brGDGTs = brGDGTs)
```

```
# calculate the FA following 2. brGDGT_MI_FA
brGDGT.MI.FA          <- brGDGT_MI_FA(brGDGTs = brGDGTs)

# calculate the FA following 3. brGDGT_METH_5MeP_FA
brGDGT.METH.5Mep.FA   <- brGDGT_METH_5Mep_FA(brGDGTs = brGDGTs)

# calculate the FA following 4. brGDGT_METH_6MeP_FA
brGDGT.METH.6Mep.FA   <- brGDGT_METH_6Mep_FA(brGDGTs = brGDGTs)

# calculate the FA following 4. brGDGT_METH_5Me_FA
brGDGT.METH.5Me.FA    <- brGDGT_METH_5Mep_FA(brGDGTs = brGDGTs)

# calculate the FA following 5. brGDGT_METH_6Me_FA
brGDGT.METH.6Me.FA    <- brGDGT_METH_6Me_FA(brGDGTs = brGDGTs)

# calculate the FA following 7. brGDGT_METH_FA
brGDGT.METH.FA        <- brGDGT_METH_FA(brGDGTs = brGDGTs)

# calculate the FA following 8. brGDGT_CYCL_FA
brGDGT.CYCL.FA        <- brGDGT_CYCL_FA(brGDGTs = brGDGTs)

# calculate the FA following 9. brGDGT_CYCL_5Me_FA
brGDGT.CYCL.5Me.FA    <- brGDGT_CYCL_5Me_FA(brGDGTs = brGDGTs)

# calculate the FA following 10. brGDGT_CYCL_6Me_FA
brGDGT.CYCL.6Me.FA    <- brGDGT_CYCL_6Me_FA(brGDGTs = brGDGTs)
```

5.4.2 isoGDGTs

The functions are stored in the “Functions” folder.

```
# calculate the FA following 11
isoGDGTs.FA            <- isoGDGT_FA(isoGDGTs = GDGTs)
```

5.4.3 OHGDGTs

The functions are stored in the “Functions” folder.

```
# calculate the FA following 12.
OHGDGTs.FA          <- OHGDGT_FA(OHGDGTs = GDGTs)
```

5.4.4 GMGTs

The functions are stored in the “Functions” folder.

```
# calculate the FA following 13.
GMGTs.FA         <- GMGT_FA(GMGTs = GMGTs)
```

5.5 Store csv output files

```
# Define a list of datasets and corresponding filenames. This will be handed
# to the export_data_to_csv function in the helper functions file.
data_sets <- list(
  brGDGT.FA = brGDGT.FA,
  brGDGT.7Me.FA = brGDGT.7Me.FA,
  brGDGT.MI.FA = brGDGT.MI.FA,
  brGDGT.METH.5Mep.FA = brGDGT.METH.5Mep.FA,
  brGDGT.METH.6Mep.FA = brGDGT.METH.6Mep.FA,
  brGDGT.METH.5Me.FA = brGDGT.METH.5Me.FA,
  brGDGT.METH.6Me.FA = brGDGT.METH.6Me.FA,
  brGDGT.METH.FA = brGDGT.METH.FA,
  brGDGT.CYCL.FA = brGDGT.CYCL.FA,
  brGDGT.CYCL.5Me.FA = brGDGT.CYCL.5Me.FA,
  brGDGT.CYCL.6Me.FA = brGDGT.CYCL.6Me.FA,
  isoGDGTs.FA = isoGDGTs.FA,
  OHGDGTs.FA = OHGDGTs.FA,
  GMGTs.FA = GMGTs.FA
)

output_directory <- list(
  DirFA.br = DirFA.br,
  DirFA = DirFA
)

# write csv files into output directory using helper function.
export_data_to_csv(data_sets, output_directory, data.sets.name)
```

5.6 Index calculations

5.6.1 data preparation

```
#cut out and shape the GDGTs ad the brGDGT Fractional Abundances

GDGTs                  <-   cbind(IS, GDGTs, GDDs, GMGTs,
                                  apply(brGDGT.FA[,-1],2,as.double))
GDGTs[is.na(GDGTs)]    <-   0
GDGTs[GDGTs=="NaN"]    <-   0
GDGTs                  <-   data.frame(GDGTs)
```

5.6.2 Index calculation

All the calculation functions are stored in the functions folder.

```
# Calculate the different indices based on the custom functions
brGDGT.IND   <- brGDGT_INDICES(GDGTs = GDGTs)
isoGDGT.IND  <- isoGDGT_INDICES(GDGTs = GDGTs)
OHGDGT.IND   <- OHGDGT_INDICES(GDGTs = GDGTs)
GMGT.IND    <- GMGT_INDICES(GDGTs = GDGTs)
GDD.IND    <- GDD_INDICES(GDGTs = GDGTs)
```

5.6.3 Print the different indices files

```r
# Define a list of data frames and corresponding file suffixes
indices.print <- list(
  list(data = brGDGT.IND, suffix = "BR_INDICES"),
  list(data = isoGDGT.IND, suffix = "ISO_INDICES"),
  list(data = OHGDGT.IND, suffix = "OH_INDICES"),
  list(data = GMGT.IND, suffix = "GMGT_INDICES"),
  list(data = GDD.IND, suffix = "GDD_INDICES")
)

# Iterate over the list to prepare and write CSV files
lapply(indices.print, function(ind) {
  # Prepare the print file by combining the relevant columns
  ind_print <- cbind(
    Label = rownames(ind$data),
    mid.depth = GDGTs$cum.depth,
    Age = GDGTs$Age,
    ind$data
  )

  # Write the CSV file into the Output directory
  write.csv(ind_print, row.names = FALSE,
            file = paste0(DirIND, "/", data.sets.name, "_", ind$suffix, "_",
                          Sys.Date(), ".csv"))
})
```

5.7 Compound concentration calculations

here we calculate the concentration per GDGT compound based on the added internal standard with known amount.

5.7.1 Amount and concentration calculations

```r
# calculate the concentration factor (concentration/area) of the Internal Standard (IS)
IS.factor <- IS[,"IS_AMOUNT"] / IS[,"IS_AREA"]

#set all inf values to NA
IS.factor[is.infinite(IS.factor)] <- NA

#Calculate the amount per substance and sample based on the Internal Standard (IS)
# and add the sample information
GDGTs.amount <- GDGTs.conc*IS.factor

#Calculate the concentration per dry sample mass (the in-weight for the extraction)
GDGTs.conc <- GDGTs.amount/IS[,"EXTRACTEDSAMPLEWEIGHT"]


# Prepare and save amounts and concentrations
GDGTs.amount.IS <- cbind(rownames(IS), IS[, 2:6], IS.factor, GDGTs.amount)
```

```r
colnames(GDGTs.amount.IS)[1] <- "Label"
write.csv(GDGTs.amount.IS, file = paste0(DirCONC, data.sets.name, "_AMOUNT_",
                                         Sys.Date(), ".csv"), row.names = FALSE)

GDGTs.conc.IS <- cbind(rownames(IS), IS[, 2:6], IS.factor, GDGTs.conc)
colnames(GDGTs.conc.IS)[1] <- "Label"
write.csv(GDGTs.conc.IS, file = paste0(DirCONC, data.sets.name, "_CONC_",
                                       Sys.Date(), ".csv"), row.names = FALSE)
```

5.8 Save session information

Make sure to use the } bracket, as this is the "closing" of the Main processing loop.

```r
# Save session info for reproducibility, no need to change anything

save_session_info("Output", data.sets.name)

}
```

Now you should be able to find all your processed data in the "output" directory.