GaDGeT: GDGT calculations simplified - User manual

Table of Contents

1. Introduction

Welcome to the GaDGeT 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 114 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 (Fig. 1). Researchers are free to use it under the CC BY 4.0 license, provided that the appropriate citation of "Schneider and Castañeda, (2025). GaDGeT: An open-source R-workflow for fast and flexible GDGT index calculations" 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 either the GaDGeT GitHub Repository (https://github.com/brGDGTs/GaDGeT) or from Zenodo (https://doi.org/10.5281/zenodo.15827945). Please download and locally store the software and review the README file for further information.

3. Input Files

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 2). GaDGeT processes data from input files formatted as .xlsx or .csv (Fig. 1). The software identifies and reads the data based on the column headers, so while the order of columns in your file can vary, the names of the columns **must remain unchanged (Table 1)**. Changing or modifying the column names will result in errors, and the script will not process your data correctly.

Key Guidelines:

- **File Format:** Ensure that your data files are either in .xlsx or .csv format. Both formats are supported, but the data structure must remain consistent across all files.
- **Header Names:** The header row should remain unmodified. Column names must exactly match the required headers listed below. Any deviation in spelling, case, or spacing will prevent the software from recognizing your data. Names and description are provided in Table 1.
- **Handling Missing Data:** If a specific GDGT group or compound was not measured or is below detection limits, leave the respective cells blank or fill them with NA. Ensure the column remains present even if no data is available for that compound.
- **Multiple Datasets:** You can include multiple datasets in your "Input" folder. GaDGeT will process each file sequentially, generating separate output folders for each dataset.

Important Notes:

- Ensure that every column in the template is included in your input file, even if certain columns contain no data (in which case, enter NA).
- The column headers are case-sensitive. Be sure to match the names exactly as provided in this manual.
- Multiple input files can be placed in the "Input" folder, and each will be processed individually by GaDGeT.

By following these guidelines, you will ensure that your data is processed accurately and efficiently by GaDGeT.

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 starting data analysis and calculations, it's important to correctly set up your workspace. To ensure everything runs smoothly, highlight the entire script (e.g., Ctrl+A) and execute it (e.g., Ctrl+Enter, or by clicking "Run" at the top of the script). Below is a brief overview of what the script will do:

1. **Clean Workspace**: The script clears any existing graphics, variables, and console messages, ensuring a fresh R workspace to avoid interference from previous sessions.
2. **Set Working Directory**: The script automatically identifies the working directory, but this can also be set manually. This step ensures that the script knows where to locate your data by setting the working directory to the folder containing your data files within the GaDGeT environment.
3. **Install and Load Packages**: The script installs and loads the required R packages (e.g., "stringr", "readxl", "readr") necessary to access essential functions.
4. **Load Custom Functions**: Custom functions from the "Functions" folder are automatically imported by the script, enhancing its capabilities within the GaDGeT suite.
5. **Loop through Input Files**: The script processes all input files by looping through them and performing the available calculations for each dataset.
6. **Write Results**: The script saves all results in organized subdirectories and files within the "Output" folder (as described in Table 2). Additional details are available 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. 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 (18 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

GaDGeT calculates various indices for the compounds, 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, provided by the user in the input file(s) under the columns "IS_Area" and "IS_Amount." In the "IS_Area" column, enter the HPLC-derived peak area, and in the "IS_Amount" column, provide the corresponding amount of the internal standard (e.g., C_{46}). The script computes concentration factors, amounts, and concentrations for the compounds. The results are saved as separate CSV files for amounts and concentrations within the GaDGeT "Output" directory. Note that these concentration values are intended for your own subsequent analyses (e.g., substance-flux calculations) and are not used for further calculations within GaDGeT itself, which relies on fractional abundances (FAs) or peak areas for its other computations.

5. 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 automatically process them ("loop through them") and 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!

6. 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.

7. Troubleshooting

If you encounter any issues or errors while running the GaDGeT script, here are some common troubleshooting steps to resolve them:

- **Read Console Messages**: Check the RStudio console for error messages. These often provide helpful information about missing or incorrect column names, missing packages, or other issues.
- **Check Input/Output Files**: Ensure that no input or output files are open in another program (e.g., Excel or a text editor). Files that are open elsewhere cannot be accessed or written to by the GaDGeT script.
- **Verify Data Location and Structure**: Double-check that your data files are placed in the "Input" folder within the GaDGeT working directory. Ensure that the file structure and header names (Table 1) are correct and haven't been altered.
- **Ensure Correct Package Installation**: Confirm that all required R packages (e.g., "stringr", "readxl", "readr") are installed and loaded properly. If necessary, reinstall missing or outdated packages by using install.packages("package_name").
- **Working Directory Issues**: If the script is not finding your data, verify that the working directory is correctly set to the folder where your data is located. You can manually set the working directory by using setwd("path/to/your/folder") or by re-running the script from a fresh RStudio session.
- **Restart RStudio and R**: If problems persist, try closing and reopening RStudio. Sometimes, restarting the session can resolve conflicts or temporary issues related to the working directory or package loading.
- **Reinstall R and RStudio**: If none of the above steps work, consider reinstalling R and RStudio to ensure that there are no deeper installation issues affecting the script's performance.

- **Contact the Author**: If you continue experiencing issues, feel free to contact the author for support at tobiaschnei@gmail.com or via www.drtobiasschneider.com.

8. Adding Index Calculations

Can I add a new GDGT ratio or index to GaDGeT?

Yes! GaDGeT is designed to be flexible, allowing users to add new calculations as research evolves and new proxies and calibrations are proposed. Below is a step-by-step guide on how to integrate new calculations into GaDGeT. In this example, we will demonstrate how to add 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 within the GaDGeT directory. Inside this folder, you will find files for different compound types (e.g., isoGDGTs, branched GDGTs, OHGDGTs). 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 to the line that says #enter the number of Indices here as "n" (approximately at line 110). Update the number of indices by changing n = 52 to n = 54 to account for the two additional indices you are about to add.

```
105
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 the new column names "IRpenta" and "IRhexa" to the list, making sure to include a comma after each new entry and keeping 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
```

5. **For IRhexa, add:**

   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:** Double-click the GaDGeT.R file to rerun the software, following the standard procedure. The new indices will now be included in the output.

Notes:

- You can use the same steps to add other indices to different function scripts.
- The position of the new indices in the script will determine their order in the output CSV file, maintaining consistency.

9. Figures and Tables

Table 1. Required column headers.

Column Header	Description
Label	Unique identifier or name for each sample.
DEPTH	Depth from which the sample was taken (usually in cm), important for stratigraphic or sediment core samples.
AGE	Age of the sample, either measured or estimated, typically expressed in years or calibrated years before present (cal BP).
WEIGHT	Weight of the sample, typically provided in grams, necessary for concentration calculations, keep track of the units.
IS_AREA	HPLC-derived peak area for the internal standard, used for normalizing GDGT concentrations.
IS_AMOUNT	The amount of internal standard (e.g., C46) added to each sample, typically in ng or µg.
GDGT.0	Peak area for GDGT-0, one of the isoprenoid glycerol dialkyl glycerol tetraethers.
GDGT.1	Peak area for GDGT-1, an isoprenoid GDGT containing one cyclopentyl moiety.
GDGT.2	Peak area for GDGT-2, an isoprenoid GDGT containing two cyclopentyl moieties.
GDGT.3	Peak area for GDGT-3, an isoprenoid GDGT containing three cyclopentyl moieties.
GDGT.4	Peak area for GDGT-4, a tetraether compound with four cyclopentyl moieties, also referred to as Crenarchaeol.
GDGT.4.	Peak area for GDGT.4.; the isomer of GDGT.4
OH-GDGT.0	Peak area for hydroxylated GDGT-0 (OH-GDGT), where a hydroxyl group replaces one of the alkyl groups.
OH-GDGT.1	Peak area for hydroxylated GDGT-1, containing one cyclopentyl moiety.
2OH-GDGT.0	Peak area for 2-hydroxy GDGT-0, containing two hydroxyl groups.
OH-GDGT.2	Peak area for hydroxylated GDGT-2, containing two cyclopentyl moieties.
IIIa.5Me	Peak area for branched GDGT IIIa, containing 5-methyl isomers.
IIIa.6Me	Peak area for branched GDGT IIIa, containing 6-methyl isomers.
IIIa.7Me	Peak area for branched GDGT IIIa, containing 7-methyl isomers.
IIIb.5Me	Peak area for branched GDGT IIIb, containing 5-methyl isomers.
IIIb.6Me	Peak area for branched GDGT IIIb, containing 6-methyl isomers.
IIIb.7Me	Peak area for branched GDGT IIIb, containing 7-methyl isomers.
IIIc.5Me	Peak area for branched GDGT IIIc, containing 5-methyl isomers.
IIIc.6Me	Peak area for branched GDGT IIIc, containing 6-methyl isomers.
IIa.5Me	Peak area for branched GDGT IIa, containing 5-methyl isomers.
IIa.6Me	Peak area for branched GDGT IIa, containing 6-methyl isomers.
IIa.7Me	Peak area for branched GDGT IIa, containing 7-methyl isomers.
IIb.5Me	Peak area for branched GDGT IIb, containing 5-methyl isomers.
IIb.6Me	Peak area for branched GDGT IIb, containing 6-methyl isomers.
IIc.5Me	Peak area for branched GDGT IIc, containing 5-methyl isomers.
IIc.6Me	Peak area for branched GDGT IIc, containing 6-methyl isomers.
Ia	Peak area for branched GDGT Ia.
Ib	Peak area for branched GDGT Ib.
Ic	Peak area for branched GDGT Ic.
H1048	Peak area for glycerol monoalkyl glycerol tetraether GMGT1.
H1034a	Peak area for GMGT2a.
H1034b	Peak area for GMGT2b.
H1034c	Peak area for GMGT2c.
H1020a	Peak area for GMGT3a.
H1020b	Peak area for GMGT3b.
H1020c	Peak area for GMGT3c.
isoGDD0	Peak area for isoprenoid glycerol dialkyl diethers GDD0.
isoGDD1	Peak area for isoprenoid GDD1.
isoGDD2	Peak area for isoprenoid GDD2.
isoGDD3	Peak area for isoprenoid GDD3.
isoGDDCren	Peak area for isoprenoid GDDCren.

Table 2. 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.

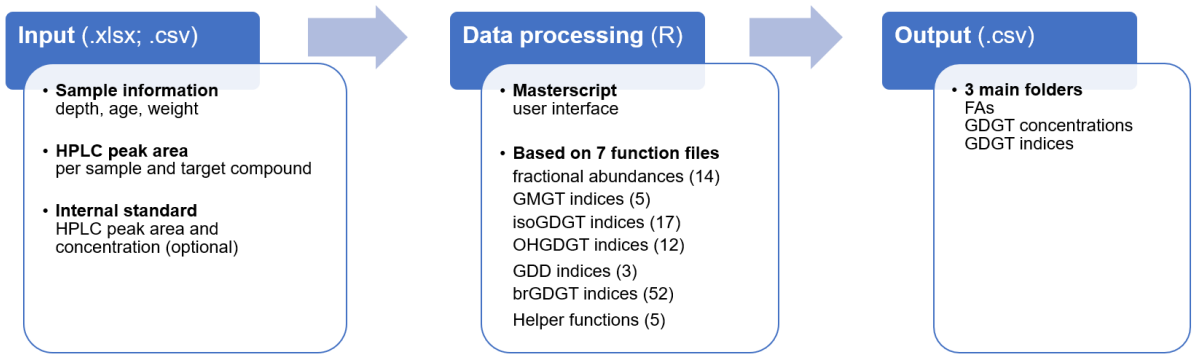

Fig. 1 Workflow diagram illustrating the data processing pipeline of the GaDGeT software for GDGT analysis. The process starts with the input of sample information in an .xlsx; .csv format, which includes details like depth, age, weight, HPLC peak areas, and optional internal standard concentrations. 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.

10. R markdown file

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

GaDGeT: GDGT Calculations Simplified – User Manual

Tobias Schneider and Isla Castañeda

July 2025

Contents

1 Overview

GaDGeT automates GDGT calculations based on raw HPLC peak area data provided by the user. It processes HPLC peak area data from .csv or .xlsx files in the `Input` directory and outputs CSV files with the calculated results into the `Output` directory. It computes compound concentrations (if an internal standard is provided), fractional abundances using various approaches, and a wide range of published indices (Tables 1-7, Schneider and Castañeda, 2025).

- **Software version:** v1.0.0 butterfly

- **Author:** Tobias Schneider
  Date: December 5, 2020
  Last Modification: July 07, 2025
  Contact: tobiaschnei@gmail.com, www.drtobiasschneider.com

- **References:**

  - Schneider, T., & Castañeda, I.S. (2025). GaDGeT: An open-source R-workflow for fast and flexible GDGT index calculations. SoftwareX. DOI: xxxx/yyyy
  - Schneider, T., & Castañeda, I.S. (2025). GaDGeT: An open-source R-workflow for fast and flexible GDGT index calculations. Zenodo. DOI: https://doi.org/10.5281/zenodo.15827945

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 GaDGeT Script

`GaDGeT.R` is the "master" script that needs to be run by the user. In order to keep it simple and clear, all the GDGT and FA calculations are implemented in the function files. The `GaDGeT.R` script sources the implemented functions from the function files in the "Functions" folder. You may access the files and check the index calculations per function file.

5 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 nor move any files from the `Functions` folder.

6 Script Structure

Here we provide a step by step explanation of the script underpinned by code snippets.

6.1 Workspace Preparation

Here we make sure to start with a 'tabula rasa' environment and set the path to the main "GaDGeT"-directory.

6.1.1 Clear the workspace, console, and close all graphics

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

6.1.2 Set working directory

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

setwd(workingdir)
```

6.1.3 Load required packages

Here we install/load the necessary r-packages that are used for the GaDGeT script.

```
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))
```

6.1.4 Load custom functions

Here we load/source all the necessary function files that contain all the GDGT and FA calculations.

```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))))
```

6.2 Data Preparation

6.2.1 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.")
}
```

6.2.2 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
```

6.3 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]
```

6.3.1 Prepare data

```
# 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]])
```

6.3.2 Separate compounds

```
# === 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", "IIa.5Me", "IIa.6Me",
                  "IIa.7Me", "IIb.5Me", "IIb.6Me", "IIc.5Me", "IIc.6Me",
                  "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", "WEIGHT", "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))) {
  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
```

```r
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)
```

6.3.3 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)
```

6.4 Fractional Abundances

6.4.1 brGDGTs

The functions are stored in the "Functions" folder. The functions are only called here –> the calculations can be found in the Functions folder. Here we calculate different FAs of brGDGTs, following the groups as suggested in Raberg et al. (2021), as well as caluclating FAs based on the 15 compounds data (5Me and 6Me), and also one containing 7Me. 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 1a. brGDGT.7Me_FA
brGDGT.7Me.FA        <- brGDGT.7Me_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)
```

6.4.2 isoGDGTs

The functions are stored in the "Functions" folder. Here, we calculate FAs for isoGDGTs based on all the included compounds.

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

6.4.3 OHGDGTs

The functions are stored in the "Functions" folder. Here, we calculate FAs for OHGDGTs based on all the included compounds.

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

6.4.4 GMGTs

The functions are stored in the "Functions" folder. Here, we calculate FAs for GMGTs based on all the included compounds.

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

6.5 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)
```

6.6 Index calculations

6.6.1 Data preparation

Here we prepare the data for the Indices functions. Note that we replace NAs with 0. It is possible that you receive values for indices containing compounds that were not measured, please do not interpret these.

```
# cut out and shape the GDGTs and the brGDGT Fractional Abundances for Index calculation

# rename the 7Me fractional abundances to avoid confusion for the software.
colnames(brGDGT.7Me.FA) <- paste0("7Me.",colnames(brGDGT.7Me.FA), sep="")

GDGTs                  <-   cbind(IS, GDGTs, GDDs, GMGTs,
                                  apply(brGDGT.FA[,-1],2,as.double),
                                  apply(brGDGT.7Me.FA[,-1],2,as.double))
GDGTs[is.na(GDGTs)]    <-   0
GDGTs[GDGTs=="NaN"]    <-   0
GDGTs                  <-   data.frame(GDGTs)
```

6.6.2 Index calculation

All the calculation functions are stored in the functions folder. Here we simply calculate the Indices, ratios, and calibrations per GDGT group and store them as data frames for later use.

```
# 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)
```

6.6.3 Print the different indices files

Here we order the data and bring it in shape to print them as .csv files to the "GDGT-Indices" output directory.

```
# 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"))
})
```

6.7 Compound concentration calculations

Here we calculate the concentration per GDGT compound based on the added internal standard with known amount. First, we calculate a ratio based on the Internal Standard (IS) and HPLC-output area size IS/Area to further convert the area of the GDGTs into contained amounts (IS.factor, also contained in your output file). In a next step, we divide it by the extracted sample weight (the in-weight to the ASE-cell) in order to calculate concentrations. Note: if you split your total lipid extract (TLE) prior measurements, you need to recalculate this to 100%.

6.7.1 Amount and concentration calculations

```
# 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[,"WEIGHT"]


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

6.8 Save session information

In case there are issues with a run of the script, you may provide this file to the author of the script when reaching out. Note: Make sure to use the } bracket, as this is the "closing" of the Main processing loop.

```
# 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.