Report Builder Guide

Github Version

Thomas Hoversen

03/12/2020

This guide explains the steps to using the Client Report Builder program, as well as step-by-step instructions for adding/removing reports and adding additional tables to individual reports.

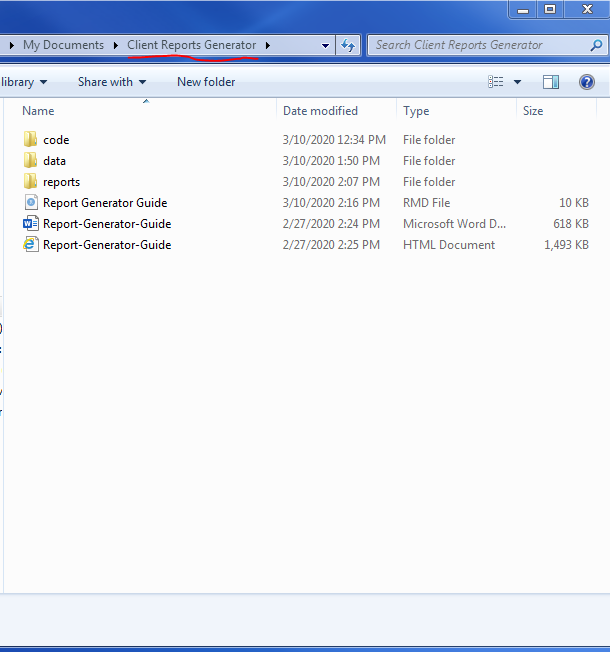
# How to Run the Program

**start**

### First, get the new data

Let’s use example data saved in C:\Users\thoversen\Documents\Client Reports Generator\data\company03102020. The actual Excel files this program processes are usually 15+ tabs and ~ 3000 - 7000 KB.

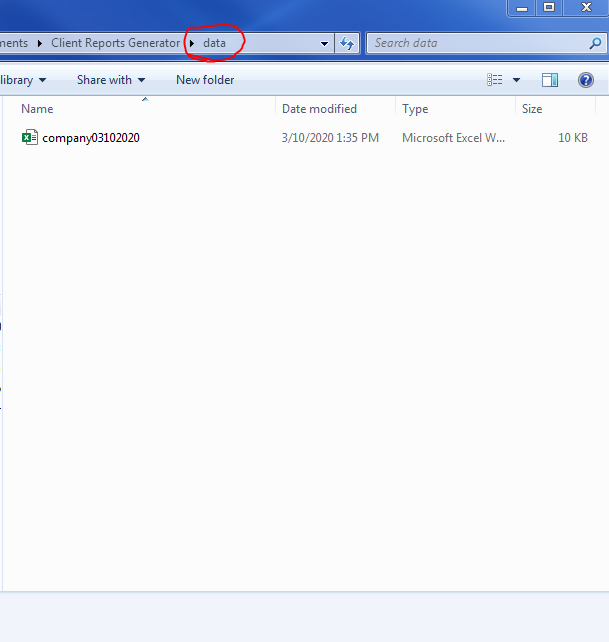
Open the Client Report Builder folder, saved here: C:\Users\thoversen\Documents\Client Reports Generator

This folder is the home-base for the program, so I’ll refer to it in this guide as the Main Directory, which looks like this: 

Notice there are three folders, “code”, “data”, and “reports”, as well as three versions of this User Guide. Please ignore the first version of this guide (it’s a .Rmd file that should not be altered). Please instead access the Microsoft Word and of html versions.

### Second, copy the new data, open the data folder in the Main Directory, and paste and save it there

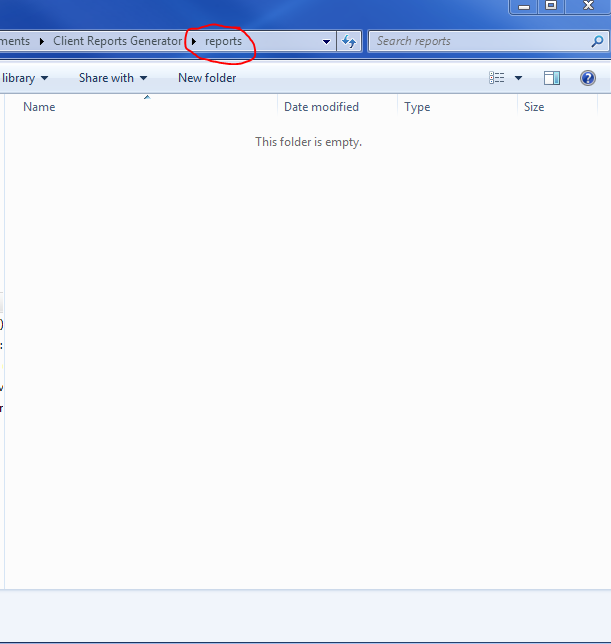
Open the “data” folder in the Main Directory. Copy the new data file, paste it in the data folder, and rename it with the naming convention companymmddyy or yyyy. Our data is saved in the data folder as company03102020. Once your data is correctly saved, return to the Main Directory.



our data is saved as “company03102020”

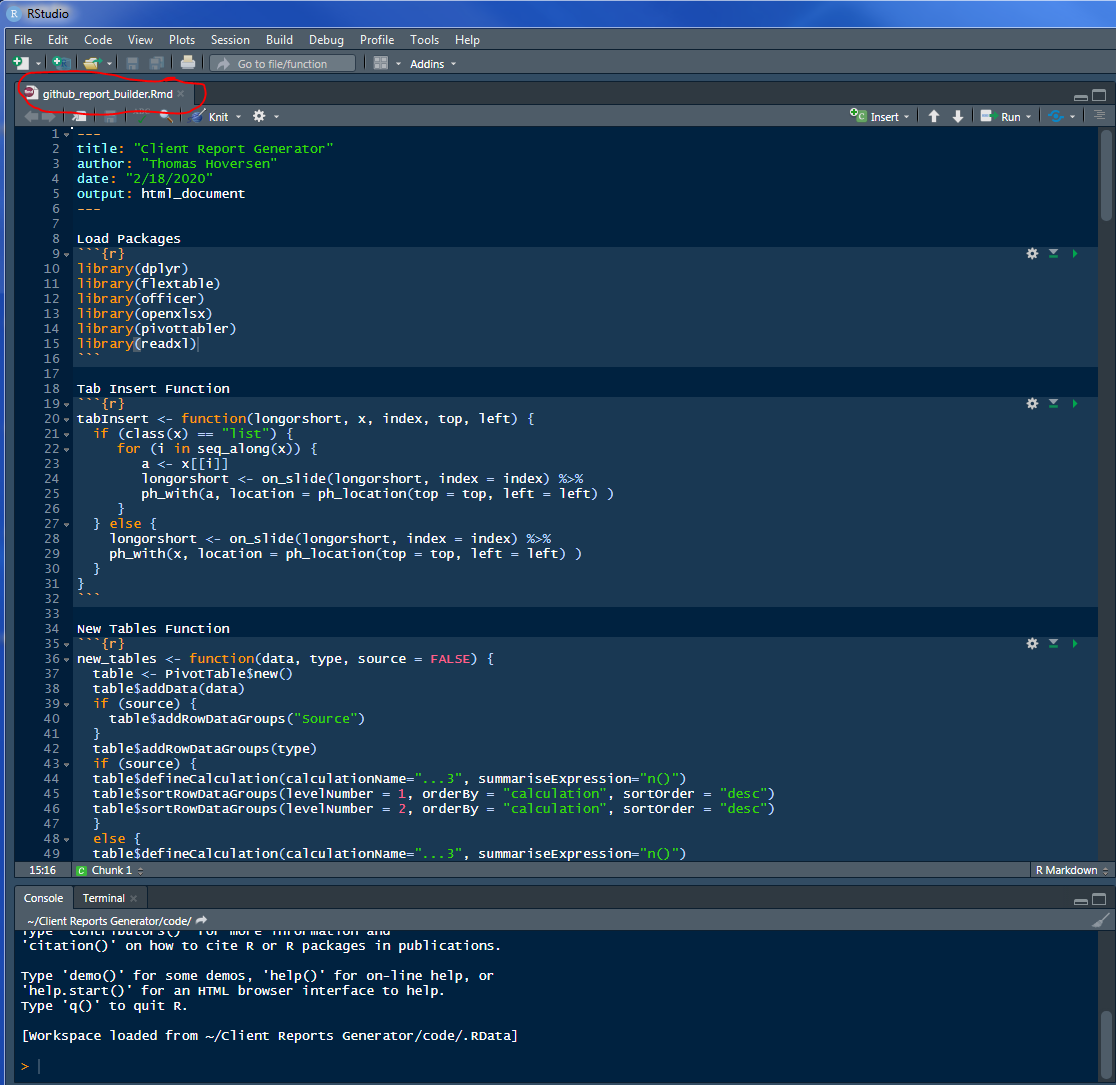
### Third, open the reports folder and clear any old reports

Open the “reports” folder in the Main Directory. If the folder contains Powerpoint reports from the last time the program ran then double check they have been turned in and then delete them. Once the reports folder is empty, return to the Main Directory.

The empty reports folder looks like this: 

### Fourth, open the code folder and then open github\_report\_builder.Rmd

Once back in the Main Directory, now that the old reports are cleared and the new data is saved in the data folder as companymmddyyyy, open the “code” folder and then double click github\_report\_builder.Rmd (please ignore copies and other folders).Double-clicking github\_report\_builder.Rmd opens RStudio. RStudio looks something like this:

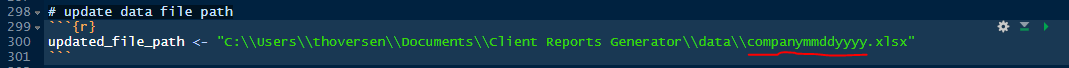


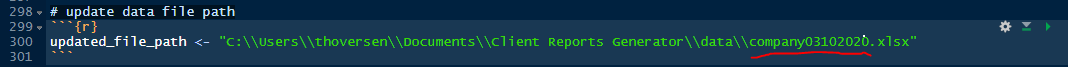
This is RStudio

The file is organized in five parts:

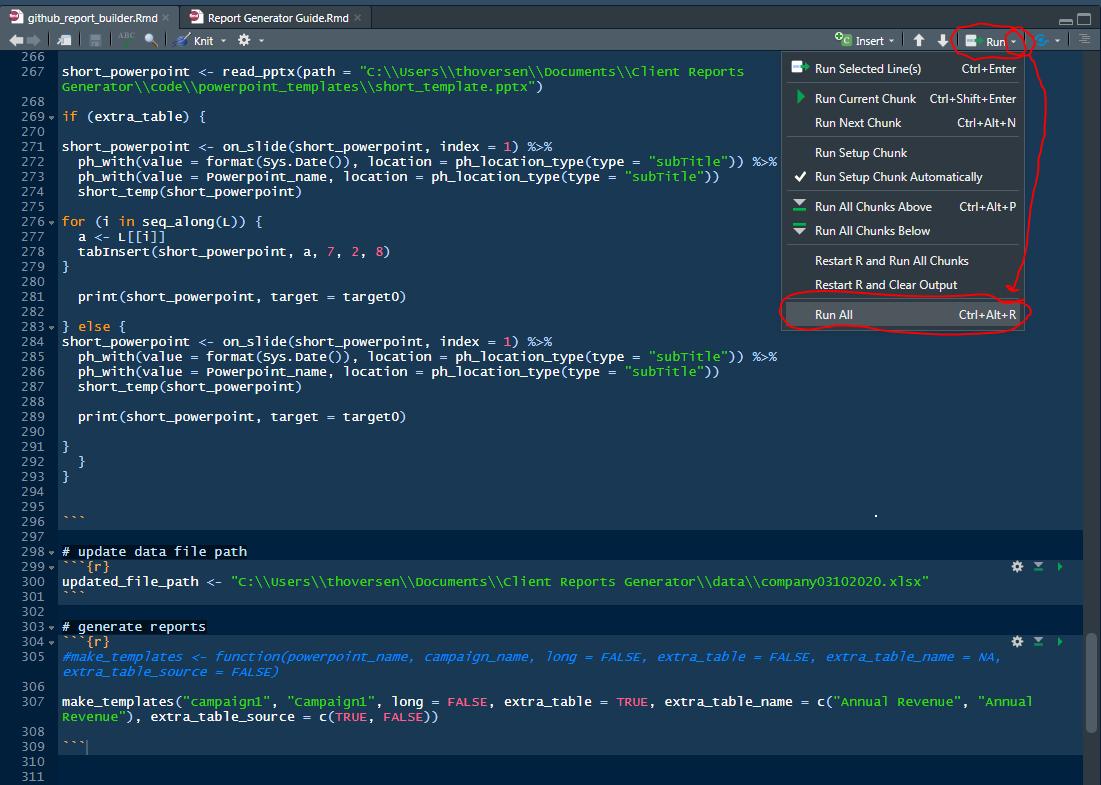
1. Loading external function packages,
2. the tabInsert() function,
3. the new\_tables() function,
4. the make\_templates() function,
5. the updated\_file\_path section, and
6. the generate reports section.

Now that github\_report\_builder.Rmd is open, scroll down to the updated\_file\_path section and update the file path. FYI, if you used the data directory’s naming convention, you only need to update the date:

From companymmddyyyy: 

To company03102020: 

Note: please do not delete .xlsx at the end of the file path name.

Lastly, click the white down-facing arrow on the Run button, then click the Run All option: 

Voila! The report builder will build every report specified in the generate reports section. After the program finishes running, you will still need to adjust the formatting of the tables in the reports and add any additional information.

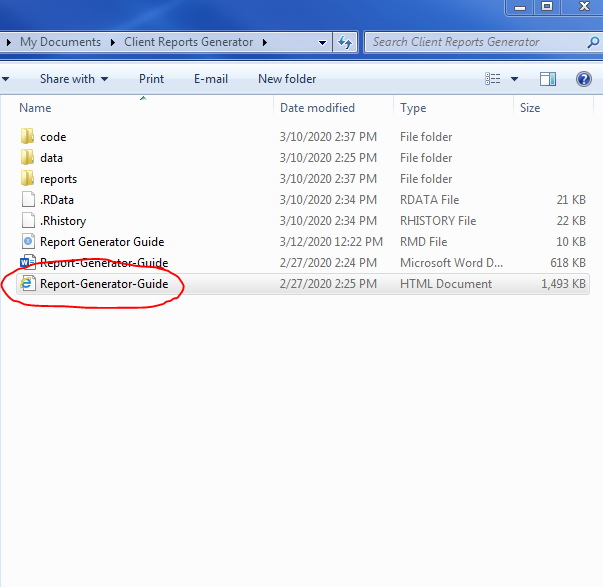
**end**

**Github Annotation**

Some additional information (specifics omitted per my NDA) must be added to each report by hand. I left this part manual so the user will confirm every report’s tables match what is expected from other data. That is slightly more work than it has to be, but my decision to do this ensures the reports are manually audited for correctness and completeness before going to the client. In a typical week this program generates ~ 15 reports, with ~ 13 tables each. This job was done manually, by hand, before I proposed and completed this project. By hand this task took ~ 4 hours of work in Excel and Powerpoint; if you are fast then it takes may be ~ 3 hours manually. It now takes ~ 20 minutes, or ~ 30 minutes at most. The bulk of this time is spent adjusting the formatting in the Powerpoint presentations and adding additional information. This program handles reports for my past employer’s largest client and has been in use since around November, 2019. There have been several iterations between its conception and its current version. Sections have also been changed to omit any client or company information. My primary focus, besides writing code that make reports that meet the client’s quality expectations, was to create a program that my co-workers (with no coding experience) can rely on and use with ease when I’m not around.

**end**

# A "#" makes a comment, which is R code the computer doesn't read.  
# This is another comment.  
  
# These instructions may look better in the html version of this guide, saved in the Main Directory.



Consider reading the following sections in Report-Generator-Guide.html

# How to Add/ Remove Reports

**start**

The make\_templates() function takes the following parameters: make\_templates(powerpoint\_name, campaign\_name, long = FALSE, extra\_table = FALSE, extra\_table\_name = NA, extra\_table\_source = FALSE). To add a standard report you only need to use the first two arguments, and the optional third argument if you’d like a longer report (most Powerpoint reports are not longer than eight slides, but the third parameter provides more space if you need it).

# To add a report, type make\_templates() and type what you want the report to be saved as in the "reports" folder in the "powerpoint\_name" parameter. It must be in quoatation marks. Then, in the second parameter, aka the "campaign\_name" parameter, type the name of the Excel tab/ sheet that contains the desired report's data. This must also be in quotation marks.  
  
# Examples:  
  
# This is an example of how to add a report containing the data in the Campaign1 tab of company03102020.  
# Note: this will not work as you expect if you forget to update the file path, and it simply won't run if the Excel tab/ sheet is misspelled or doesn't exist.  
  
make\_templates("campaign1\_Report", "Campaign1") # make\_template("powerpoint\_name", "campaign\_name"). Notice the file name states what campaign the report is for (Campaign1).   
  
# The default report length is 8 slides. If you want a longer report with 11 slides, then write "long = TRUE" in the third argument (default is long = FALSE).  
  
make\_templates("campaign1", "Campaign1", long = TRUE) # make\_template(powerpoint\_name, campaign\_name, long = TRUE).

This is sufficient to add the majority of client reports. Standard tables and general formatting are already hard-coded into the reports. Instructions for adding additional tables are in the next section. To remove a report, simply delete the line containing the report you’re removing, or put a # in front of it to make it a comment.

# By adding a "#" in front of the make\_templates() function, the report will not run. This is an alternative way to remove reports without deleting the line of code.  
  
# make\_templates("campaign1", "Campaign1", long = TRUE)

**end**

# How to Add/ Remove Additional Tables to Reports

**start**

Right now, by default, every report in this github-friendly-version of the program gets 6 tables. Those are:

1. Industry
2. Job Title
3. Employee Size
4. Industry pivoted by Source
5. Job Title pivoted by Source
6. Employee Size pivoted by Source

Sometimes I needed to add additional tables to certain reports if they have custom questions. Other times you may just want to show more information. To show how to do this, let’s add additional tables to the Campaign1 report as an example.

**NOTE:** The rest of this section looks like a lot of code, but each code chunk is simply a copy of the previous code chunk with the next consecutive parameter filled in with a brief explanation. Please read this section carefully to avoid confusion. This section should be sufficient to implement the additional tables functionality with any report and any type of table for the foreseeable future. **While I acknowledge the rest of this guide is code heavy, this section can and should serve as a clear, step-by-step template for adding/ removing custom tables to reports.**

# This is the example report we will add custom tables to:  
make\_templates("campaign1", "Campaign1")

Let’s make a custom table for the “Annual Revenue” column in Campaign1 report’s Excel file. In order for the Report Builder to run, you must exactly copy the name of the column of the additional table.

Normally you only need to use the first two parameters to make a report, with the option of the third parameter when you need a longer report (long = TRUE). The rest of the parameters, by default, do nothing. You need to change the default settings for these parameters in order to add additional tables. **Note: Separate each parameter with a comma.**

In this example we will add two additional tables to Campaign1’s report: an “Annual Revenue” table and an “Annual Revenue” pivoted by Source table.

First (optional), set long = TRUE because we may need extra room for the additional tables.

make\_templates("campaign1", "Campaign1", long = TRUE) # notice this code is a copy of the previous code chunk with the next consecutive parameter filled in.

Next, write extra\_table = TRUE to tell the program this report has extra tables.

make\_templates("campaign1", "Campaign1", long = TRUE, extra\_table = TRUE) # notice the fourth parameter of make\_templates() is now set to TRUE (default: extra\_table = FALSE)

Third, write “COLUMN NAME OF TABLE” in the extra\_table\_name section, which in our case is “Annual Revenue”.

make\_templates("campaign1", "Campaign1", long = TRUE, extra\_table = TRUE, "Annual Revenue") # notice you should only write the column name in quotation marks

This is sufficient to get a normal pivot table for the custom table.

To make the “Annual Revenue” table pivot by source, write TRUE in the consecutive extra\_table\_source parameter.

make\_templates("campaign1", "Campaign1", long = TRUE, extra\_table = TRUE, "Annual Revenue", TRUE) # I wrote TRUE, meaning this table will be pivoted by Source. By default it is FALSE.

However, we want to pivot “Annual Revenue” both with source **and** without source. Therefore, we simply edit the fifth parameter, extra\_table\_name, from “Annual Revenue” to c(“Annual Revenue”, “Annual Revenue”). We write it twice inside of c() to add multiple tables. This applies for every extra table you add. There is no limit to how many tables you can add. Also, the program does not prevent you from adding (albeit frivolously) tables that are already hard-coded in the report. Simply fill in the extra\_table\_name parameter with c("FIRST TABLE NAME", "SECOND TABLE NAME", "THIRD TABLE NAME", ... "NTH TABLE NAME"), where each element inside of c() represents an additional table (column name) you’re adding to the report.

# We want to pivot this table normally and by vendor, so we need to write the column name twice.  
make\_templates("campaign1", "Campaign1", long = TRUE, extra\_table = TRUE, c("Annual Revenue", "Annual Revenue"), TRUE)  
  
# notice the extra\_table\_vendor parameter (the sixth, and last, parameter) only accounts for one table, where TRUE means pivot the first custom table by source. Due to the inner workings of R, this code would recycle the TRUE value and apply it to every element of the extra\_table\_name parameter. This is wrong.

To solve the issue identified in the above comment, use c() in the extra\_table\_source too. Write TRUE or FALSE for as many additional tables you add (Note: TRUE and FALSE must **not** be in quoatation marks. **Also:** The order of TRUEs and FALSEs corresponds to the order of c(“tables”, …) in the extra\_table\_name parameter. In other words, write c(FIRST TABLE TRUE/FALSE, SECOND TABLE TRUE/FALSE, THIRD TABLE TRUE/FALSE, ... NTH TABLE TRUE/FALSE) in the extra\_table\_vendor section.

# we write "c(TRUE, FALSE)" in the extra\_table\_vendor section because we want to make two tables for "Annual Revenue". The first custom table will be pivoted by Source (the TRUE), and the second will not be (the FALSE).  
make\_templates("campaign1", "Campaign1", long = TRUE, extra\_table = TRUE, c("Annual Revenue", "Annual Revenue"), c(TRUE, FALSE))

That is it! You can add as many or as few tables as you wish, so long as they exist as a column in the data tab you’re using. To remove one of these custom tables, simply delete it from the extra\_table\_name parameter, and delete its corresponding TRUE/FALSE value from the extra\_table\_vendor parameter. If you want to remove all of the custom tables, please remember to set the fourth argument, extra\_table, to FALSE (…, extra\_table = FALSE), or remove it altogether to invoke its default (FALSE). If this section seems confusing, try re-reading it again. Adding custom tables only **requires** invoking three new parameters in the make\_templates() function. After doing it once or twice, the steps should become intuitive. If you’re unsure, follow the above template step-by-step, replacing the example inputs with your your own.

**end**