

# Code & Data Handling in Empirical Research Projects

Could We Do Better?

Tobias Witter

HUB, TRR 266

Februar 15, 2022

## Mental juggling with research projects

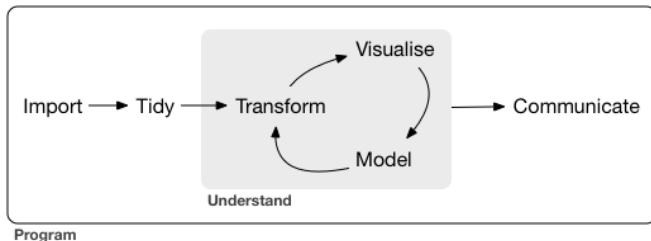
A while ago, every time I looked into old project folders, or project folders of others, it felt like...



## Part 1: An integrated view on empirical research projects

# Empirical researcher, data scientist or programmer?

Model of the tools needed in a typical data science project  
([Wickham and Grolemund 2017](#)):



# Empirical research projects

1. Retrieve/collect raw data
2. Import raw data
3. Tidy raw data
4. Transform (raw) data
5. Visualize transformed data [tables, figures, statistics]
6. Model [explore, describe, causally test for relationships between variables]
7. Generate research “products”
8. Communicate

# Software developers as new best buddies?

## Python File Structure Tree


```
Project_Name/  
├── .git/  
├── .vscode/  
├── data/  
├── debug/  
├── docs/  
├── etc/  
├── include/  
│   └── Project_Name/  
│       └── public_functions.py  
├── lib/  
├── src/  
│   ├── assets/  
│   │   ├── fonts/  
│   │   ├── images/  
│   │   ├── sounds/  
│   │   └── videos/  
│   ├── utils/  
│   ├── functions_code.py  
│   ├── main.py  
│   └── private_funtions.py  
├── tests/  
│   ├── alpha_version/  
│   └── beta_version/  
├── Python.gitignore  
└── ReadMe.md
```


## The TRR 266 template:

 code/R

 data

 doc

 media

 output

 .gitignore

 LICENSE

 Makefile

 README.md

## Part 2: Code & data







# Code & data organization

- ▶ Product-oriented organization
  - ▶ Create software and data “products”
  - ▶ Products: functions, output
  - ▶ Publish your code and data, get a DOI?
  - ▶ License your stuff!
- ▶ Version control systems
  - ▶ Git and GitHub (free account for researchers)
  - ▶ Using clouds like Nextcloud (desktop application with local drive?)
  - ▶ Advantages of GitHub over clouds?



# Code organization

An example from the TRR 266 template repository:






 main ▾ <a href="#">treat</a> / <a href="#">code</a> / <a href="#">R</a> /		
..		
5	 do_analysis.R	one file for all analyses
4	 prepare_data.R	code to tidy and transform data
3	 pull_wrds_data.R	code to automatically download WRDS data
1	 read_config.R	code to check/read passwords (PW kept in a local folder!)
2	 theme_trr.R	functions (e.g. plot and table design)

# Code organization

- ▶ Consistent naming of code and variables!
  - ▶ you\_could\_use\_snake\_case
  - ▶ CamelCasesAnAlternative
- ▶ Code automation?
  - ▶ No need to run line-by-line, but script-by-script
  - ▶ Having a master file
  - ▶ Input/output pre-defined
  - ▶ Final product: Set of tables/figures, presentation or paper
- ▶ Functional vs object-oriented programming
  - ▶ Functional: A function takes a defined input to create a defined output
  - ▶ Repeated evaluation, efficient, re-using code
- ▶ Code testing: Writing tests that the code must pass

# Data organization

An example from the TRR 266 template repository:

 main ▾		<b>treat / data /</b> <i>...separate: treat / raw_data /</i>
..		
	external	<b>prepared data coming from external sources (not raw data!)</b>
	generated	<b>data your code generates</b>
	pulled	<b>data you downloaded from databases, e.g. WRDS pulls</b>
	data_readme.md	<b>info on datasets and their sources</b>

# Data handling

- ▶ Consistent naming of data files!
  - ▶ 'raw\_data\_wrds\_analyst\_forecasts.csv'
  - ▶ 'dataset\_final.csv'
- ▶ Automated data retrieval
  - ▶ Retrieve data using code
  - ▶ WRDS automated code for Stata, R, Python, SAS
- ▶ Product-orientation
  - ▶ Define data products upfront
  - ▶ Often several version ('\_v08.csv', '\_20220211.xlsx')
- ▶ Generate tidy data (next slide)

# Tidy data

Have tidy data! ([Wickham 2014](#)) ...but what's tidy data?

1. Each variable forms a column.
2. Each observation forms a row.
3. Each type of observational unit forms a table.

## Part 3: Looking at examples

## Example 1: A project folder

```
['Old versions']  
['Resources']  
['Raw data']  
definition_of_variables.xlsx  
data_snippet_A.xlsx  
R_script_1_masterfile.R  
R_script_2_load_and_class_variables.R  
R_script_3_prepare_data.R  
R_script_4_check_data_for_missing_values.R  
R_script_5_check_data_for_missing_values_old_version.R  
R_script_6_data_cleaning.R  
R_script_7_running_analyses_A.R  
R_script_8_running_analyses_B.R  
R_script_9_additional_analyses.R  
raw_data_af.xlsx  
raw_data_step_1.xlsx  
raw_data_step_2.xlsx  
raw_data_step_3.xlsx  
raw_data_step_4.xlsx  
final_dataset.xlsx
```

## Example 1: A project folder (ctd.)

.['Old versions']  
.[Resources]  
.[Raw data]  
definition\_of\_variables.xlsx  
data\_snippet\_A.xlsx  
R\_script\_1\_masterfile.R  
R\_script\_2\_load\_and\_....R  
R\_script\_3\_prepare\_data.R  
R\_script\_4\_check\_data....R  
R\_script\_5\_check\_data\_....R  
R\_script\_6\_data\_cleaning.R  
R\_script\_7\_running\_analyses\_A.R  
R\_script\_8\_running\_analyses\_B.R  
R\_script\_9\_additional\_analyses.R  
raw\_data\_af.xlsx  
raw\_data\_step\_1.xlsx  
raw\_data\_step\_2.xlsx  
raw\_data\_step\_3.xlsx  
raw\_data\_step\_4.xlsx  
final\_dataset.xlsx

.[raw\_data]  
... raw\_data.xlsx  
... definition\_of\_variables.xlsx  
.[data]  
... main\_dataset.xlsx  
... definition\_of\_variables.xlsx  
.[code]  
... prepare\_data.R  
... do\_analyses.R  
.[output] <includes your 'products'>  
... set\_of\_table\_and\_figures.docx  
masterfile.R

(+ use if-else-clauses to check if a certain step was already carried out)



## Example 2: A data folder

```
.['MainTestsData']  
['MainTestsData_v2']  
data_A_yearly_v1.xlsx  
data_A_yearly_v1.csv  
data_B_yearly_v2.xlsx  
data_B_yearly_v2.csv  
data_C_yearly_final.xlsx  
data_C_yearly_final.csv  
data_C_yearly_final.txt  
data_C_yearly_final2.txt  
data_D_compuSTAT_2005_20220210.txt  
data_E_analysis_1.txt  
data_E_analysis_1.xlsx  
data_F_analysis_2.txt  
data_F_analysis_2.xlsx
```

## Example 2: A data folder (ctd.)

['MainTestsData']

['MainTestsData\_v2']

data\_A\_yearly\_v1.xlsx

data\_A\_yearly\_v1.csv

data\_B\_yearly\_v2.xlsx

data\_B\_yearly\_v2.csv

data\_C\_yearly\_final.xlsx

data\_C\_yearly\_final.csv

data\_C\_yearly\_final.txt

data\_C\_yearly\_final2.txt

data\_D\_compustat\_2005\_20220210.txt

data\_E\_analysis\_1.txt

data\_E\_analysis\_1.xlsx

data\_F\_analysis\_2.txt

data\_F\_analysis\_2.xlsx

[data]

... main\_dataset.csv

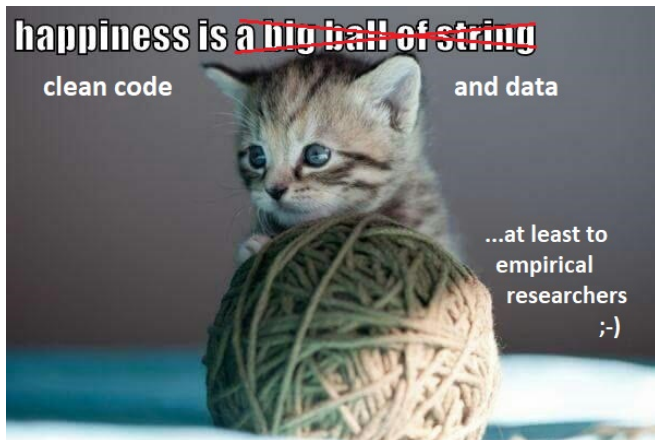
... component\_compustat.csv

... component\_ibes.csv

... component\_capitaliq.csv

... definition\_of\_variables.csv

Thanks for listening!



# Resources

- ▶ The Python File Structure Tree was taken from [AlexDCode \(2020\)](#).
- ▶ The TRR 266 Template for Reproducible Empirical Accounting Research is available from [TRR 266 \(2021\)](#).
- ▶ Free data science resources:  
<https://github.com/alastairrushworth/free-data-science>
- ▶ Read about tidy data here: [Wickham \(2014\)](#)
- ▶ Licensing of code and data: C02 Open Science office hours, [Creative Commons \(n.d.\)](#)
- ▶ Do's and don'ts from a survey of software developers ([datree.io 2019](#))

# Goals and characteristics of research templates

Reasons to follow a standard structure ([AlexDCode 2020](#)):

1. You avoid confusion
2. It is as simple as possible
3. You keep your code clean, neat, structured, and clutter free
4. The file structure system is modular
5. Each folder has an explanation
6. More documentation in the folder itself
7. Hierarchical tree file organization system
8. Standard for small to medium size projects
9. ...

# References

- AlexDCode. 2020. "Software Development Project Structure: A Template for Different Programming Languages."  
<https://github.com/AlexDCode/Software-Development-Project-Structure>.
- Creative Commons. n.d. "Share Your Work: The Creative Commons License Generator."  
<https://creativecommons.org/share-your-work/>.
- datree.io. 2019. "Top GitHub Best Practices Guide for Developers [Expanded Dec 2019]."  
<https://www.datree.io/resources/github-best-practices>.
- TRR 266. 2021. "The TRR 266 Template for Reproducible Empirical Accounting Research."  
<https://github.com/trr266/treat>.
- Wickham, Hadley. 2014. "Tidy Data." *Journal of Statistical Software* 59: 1--23.  
<https://doi.org/10.18637/jss.v059.i10>.
- Wickham, Hadley, and Garrett Golemund. 2017. *R for Data Science*. O'Reilly.  
<https://doi.org/10.1002/9781107121301>.