

Data Analytics Engineering

For Accountants and Auditors

Stewart Li

2024-01-25

Table of contents

Preface	3
I Infrastructure	4
1 Local	6
2 ELT	20
3 Example 1	22
II Data tools	25
4 Polars	29
5 Analysis	34
5.1 IO	34
5.2 Cleaning	35
5.3 Validate	35
5.4 Munging	36
5.5 EDA	37
5.6 Model	39
5.7 Report	40
6 Example 2	42
6.1 Cleaning	42
6.2 Procedure	44
6.3 Enhanced	45
References	49

Preface

This book documents the data analytics engineering workflow, which contains two parts namely infrastructure and tools. It focuses on its implementation instead of its setup. macOS is left out as Windows OS is widely used in the business setting. Pick the preferred tools after considered your career path. For instance, data/dev ops, data/analytic/ML engineer, and data analyst/scientist. My goal is to have a better solution to do auditing/accounting job easily (powerful tools), accurately (reproducible process), and automatically (job scheduler). If you don't know what I am talking about, watch [data firm](#), [financial statement preparation](#), [insurance data analysis](#), and read the paper (Li, Fisher, and Falta 2020).

You might ask how it relates to you. Generally, CFO is charge of COA, Audit partner emphasize accounting treatments, and staffs do their job at the transactional level. You need much better tools to pan out at work. For instance,

1. New job requires the strong analytic mind. Excel or similar tools are not sufficient for pattern recognition.
2. A higher staff turnover is caused by pressure and boredom. You need to be efficient by automating repetitive work such as reconciliation.

Part I

Infrastructure

The knowledge of linux (Ubuntu LTS) terminal will be beneficial when you use remote AWS services. For instance,

1. `awscli`, `terraform`,
2. `docker`, `podman`, `k8`,

ELT seems better than ETL as you normally don't know the part of transformation upfront.

1 Local

My **OS** is Windows 11. Install Window manager `komorebi`, Windows Terminal `ws12`, and Linux distribution systems. Edit terminal theme/font, dotfiles of Bash/Tmux/Vim, and env variables. Install Git/GitBash and Docker/Podman if needed.

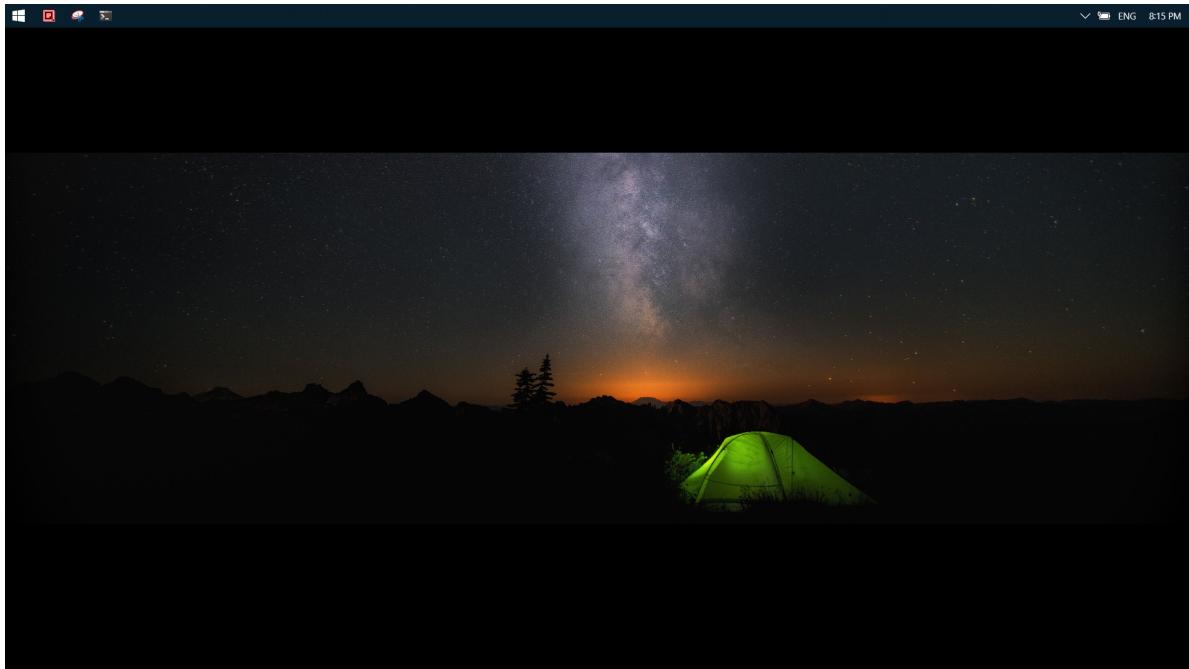


Figure 1.1: Desktop

Install **programming** languages R/Python/DuckDB/Rust/Go. R is a language designed to get shit done ([@hadleywickham](#)). Python is a glue language. Rust is a decent language for software engineering. I often live in terminal to `rofi` applications, manage `pass`, `rsync` files, `quarto` markdown, `sftp` to server, `ssh` into remote machines, and do a quick analysis for ad hoc tasks.

Editors like `nano` (Linux) and `notepad` (Windows) can be used for their simplicity. However, appropriate **IDE** helps you organize your project better. I choose Vim (Linux), RStudio (Windows), and VS Code (Both) based on the active development environment. Of course, RStudio can be launched in Linux as well.

```

Microsoft Windows [Version 10.0.19045.3440]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Stewart Li>systeminfo

Host Name: DESKTOP-HCEU07A
OS Name: Microsoft Windows 10 Home
OS Version: 10.0.19045 N/A Build 19045
OS Manufacturer: Microsoft Corporation
OS Configuration: Standalone Workstation
OS Build Type: Multiprocessor Free
Registered Owner: Stewart Li
Registered Organization: Microsoft
Product ID: K8S25-96013-32346-AA0EM
Original Install Date: 3/8/2021, 6:49:39 PM
System Boot Time: 10/9/2023, 12:31:39 PM
System Manufacturer: Dell Inc.
System Model: XPS 13 9360
System Type: x64-based PC
Processor(s): 1 Processor(s) Installed.
[001]: Intel® Family 6 Model 142 Stepping 9 GenuineIntel ~2701 MHz
BIOS Version: Dell Inc. 2.21.0, 6/2/2022
Windows Directory: C:\WINDOWS
System Directory: C:\WINDOWS\system32
Boot Device: \Device\harddiskVolume1
System Locale: en-US (English (United States))
UILocale: en-US (English (United States))
Time Zone: (UTC+08:00) Kuala Lumpur, Singapore
Total Physical Memory: 8,077 MB
Available Physical Memory: 1,293 MB
Virtual Memory: Max Size: 14,733 MB
Virtual Memory: Available: 5,154 MB
Virtual Memory: In Use: 9,579 MB
Page File Location(s): C:\pagefile.sys
Domain: WORKGROUP
Logon Server: \DESKTOP-HCEU07A
Hotfix(s): 27 Hotfix(s) Installed.
[001]: KB5029932
[002]: KB5062430
[003]: KB5062525
[004]: KB5089481
[005]: KB5003791
[006]: KB5012170
[007]: KB5015684
[008]: KB5030211
[009]: KB5006753

```

Figure 1.2: CMD

```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Stewart Li> Get-ComputerInfo

WindowsBuildLabEx : 19041.1.amd64fre.vb_release.191206-1406
WindowsCurrentVersion : 6.3
WindowsEditionId : Core
WindowsInstallationType : Client
WindowsInstallDateFromRegistry : 3/8/2021 10:49:39 AM
WindowsProductId : 00325-96013-32346-AA0EM
WindowsProductName : Windows 10 Home
WindowsRegisteredOrganization : Microsoft
WindowsRegisteredOwner : Stewart Li
WindowsSystemRoot : C:\WINDOWS
WindowsVersion : 2009
BiosCharacteristics : {7, 9, 11, 12...}
BiosTosVersion : (DELL - 1072009, 2.21.0, American Megatrends - 50008)
BiosNumber : 
BiosCaption : 
BiosCodeSet : 2.21.0
BiosCurrentLanguage : en[US]iso8859-1
BiosDescription : 2.21.0
BiosEmbeddedControllerMajorVersion : 255
BiosEmbeddedControllerMinorVersion : 255
BiosFirmwareType : Uefi
BiosIdentificationCode : 
BiosInstallableLanguages : 2
BiosInstallDate : 
BiosLanguageEdition : 
BiosListofLanguages : {en[US]iso8859-1, }
BiosManufacturer : Dell Inc.
BiosName : 
BiosOtherTargetOS : True
BiosPrimaryBios : 6/2/2022 8:00:00 AM
BiosReleaseDate : 1G73RC2
BiosSerialNumber : 2.21.0
BiosMBIOSIOSVersion : 
BiosSMBIOSMajorVersion : 3
BiosSMBIOSMinorVersion : 0
BiosMBIOSPresent : True
BiosSoftwareElementState : Running
BiosStatus : OK

```

Figure 1.3: PowerShell

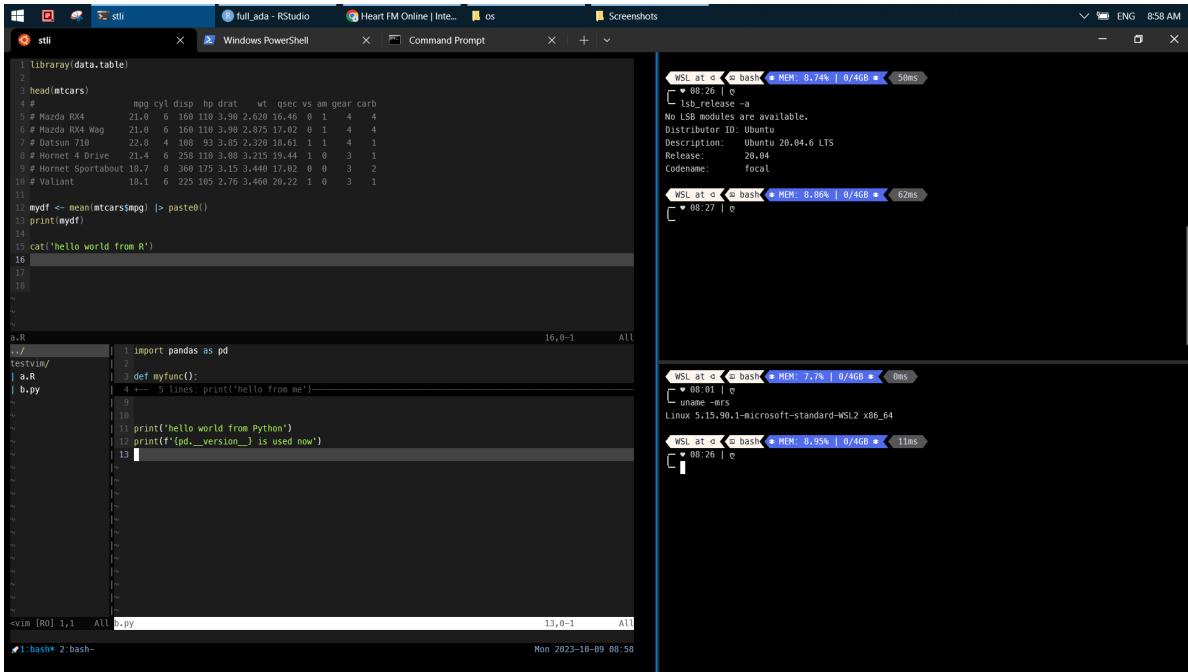


Figure 1.4: Ubuntu

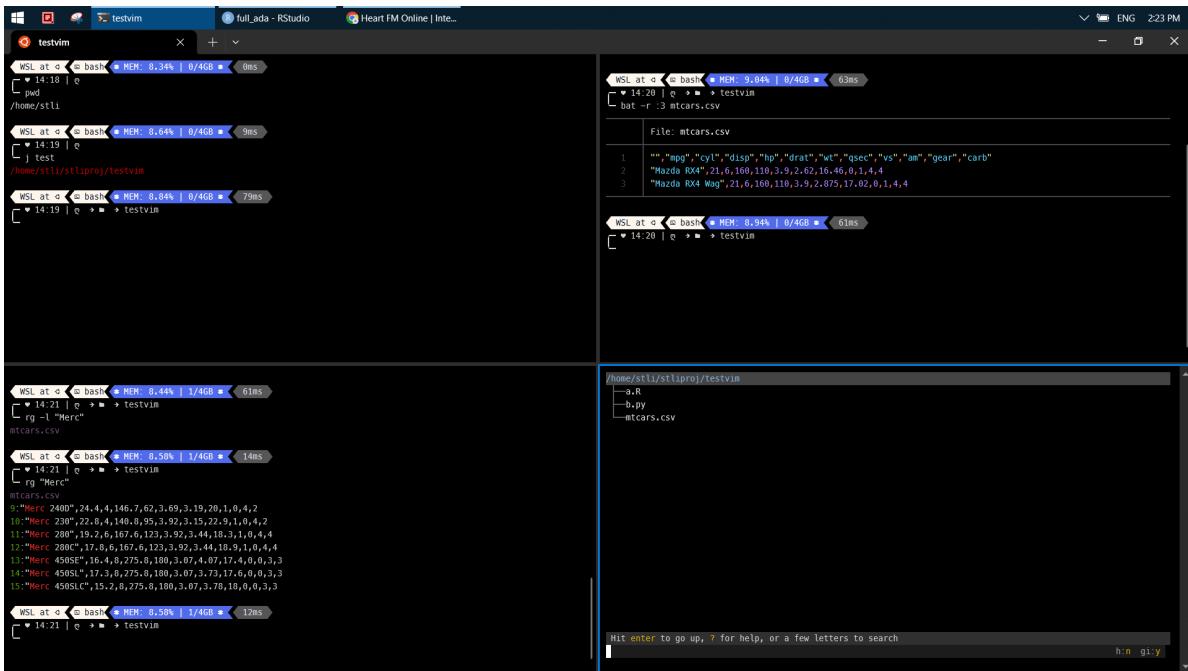


Figure 1.5: Terminal tools

```

WSL at < bash * MEM: 9.5% | 1/4GB * 8ms
└─ 15:23 | e ↵ ➤ testvim
  R

R version 4.3.1 (2023-06-16) -- "Beagle Scouts"
Copyright (C) 2023 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> write.csv(mtcars, 'mtcars.csv')
>

WSL at < bash * MEM: 10.0% | 1/4GB * 9ms
└─ 15:29 | e ↵ ➤ testvim
  python3
Python 3.8.10 (default, May 26 2023, 14:05:08)
[GCC 9.4.0] on linux
Type "help()", "copyright", "credits" or "license" for more information.
>>> import pandas as pd
>>> df=pd.read_csv("mtcars.csv")
>>> df.describe()
   mpg    cyl   disp     vs     am   gear   carb
count 32.000000 32.000000 32.000000 ... 32.000000 32.000000 32.000000
mean 20.090625 6.167500 230.721675 146.687500 ... 0.437500 4.062500 3.667500 2.8125
std  6.026948 1.785922 123.330694 ... 0.538016 0.498950 0.737800 1.6152
min  12.000000 4.000000 108.000000 ... 0.000000 0.000000 3.000000 1.0000
max 28.000000 8.000000 432.000000 95.500000 ... 8.000000 8.000000 8.000000 2.0000
50% 19.000000 6.000000 156.000000 123.000000 ... 0.000000 0.000000 4.000000 4.0000
75% 22.000000 8.000000 326.000000 180.000000 ... 1.000000 1.000000 4.000000 4.0000
max 33.900000 8.000000 472.000000 355.000000 ... 1.000000 1.000000 5.000000 8.0000

[8 rows x 11 columns]
>>>

```

Figure 1.6: R, Python, DuckDB

```

Library(data.table)
library(ggplot2)
# read(mtcars)
# # Mazda RX4
# # Mazda RX4 Wag
# # Datsun 710
# # Hornet 4 Drive
# # Hornet Sportabout
# # Valiant
# mydf <- mean(mtcars$mpg) |> paste0()
# print(mydf)
# cat('Hello world from R')
# head(mtcars)

```

```

WSL at < bash * MEM: 12.3% | 1/4GB * 9ms
└─ 15:32 | e ↵ ➤ testvim
  /duckdb
  — Loading resources from /home/stli/.duckdbrc
v0.8.1 6556a72232
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
⇒ SELECT * FROM read_csv_auto('stlipm/testvim/mtcars.csv') LIMIT 3;
   column00
      varchar
   mpg      double
   cyl      int64
   disp     double
   drat     double
   wt       double
   qsec     double
   vs       int64
   am       int64
   gear     int64
   carb     int64
Mazda RX4      21.0      6      160.0      110.0      3.90      2.620      16.46      0      1      4      4
Mazda RX4 Wag  21.0      6      160.0      110.0      3.90      2.875      17.02      0      1      4      4
Datsun 710     22.8      4      108.0      93.0      3.85      2.320      18.61      1      1      4      1
Hornet 4 Drive 21.4      6      250.0      110.0      3.08      3.215      19.44      1      0      3      1
Hornet Sportabout 18.7      8      360.0      175.0      3.15      3.440      17.02      0      0      3      2
Valiant        18.1      6      225.0      105.0      2.76      3.460      20.22      1      0      3      1

```

Figure 1.7: Vim - R

```
1 import pandas as pd
2
3 def myfunc():
4     print('hello from me')
5     print('hello from me')
6     print('hello from me')
7     print('hello from me')
8     print('hello from me')
9
10
11 print('Hello world from Python')
12 print(f'{pd.__version__} is used now')
13
```

b.py 13,0-1 All python3 "b.py" [finished] 2,17 All

Figure 1.8: Vim - Python

The screenshot shows a Windows taskbar at the top with icons for File Explorer, Task View, Start, and Task Switcher. Below the taskbar is a horizontal bar with the text "Neo-tree" and "stl". The main area contains three vertically stacked code editors:

- Neo-tree**: A file browser showing the directory structure of a Rust project. It includes files like `Cargo.toml` and `main.rs`.
- stl**: A terminal window showing the output of the command `stl`.
- Cargo.toml**: A code editor showing the contents of the `Cargo.toml` file, which specifies the package name as "testrust", version as "0.1.0", edition as "2021", and dependencies including `anyhow = "1.0.75"`. It also highlights the line `# See more keys and their definitions at https://doc.rust-lang.org/c...
- Cpp**: A code editor showing a Python script named `c.py` that imports `os` and defines a function `some()`. It also shows a command-line interface with a TODO item and a `fs::dir_tree` command.

The bottom of the screen shows the system tray with icons for battery, signal, and volume, along with the date and time: "Sat 2023-11-18 19:36".

Figure 1.9: Tmux - Nvim 1

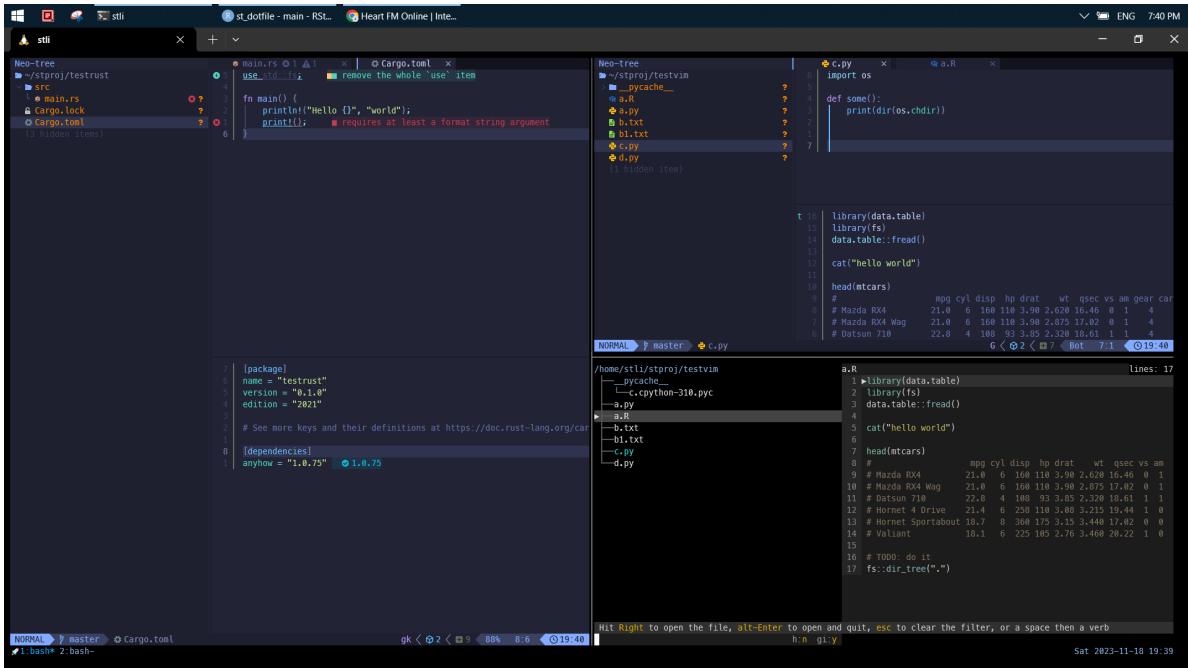


Figure 1.10: Tmux -Nvim 2

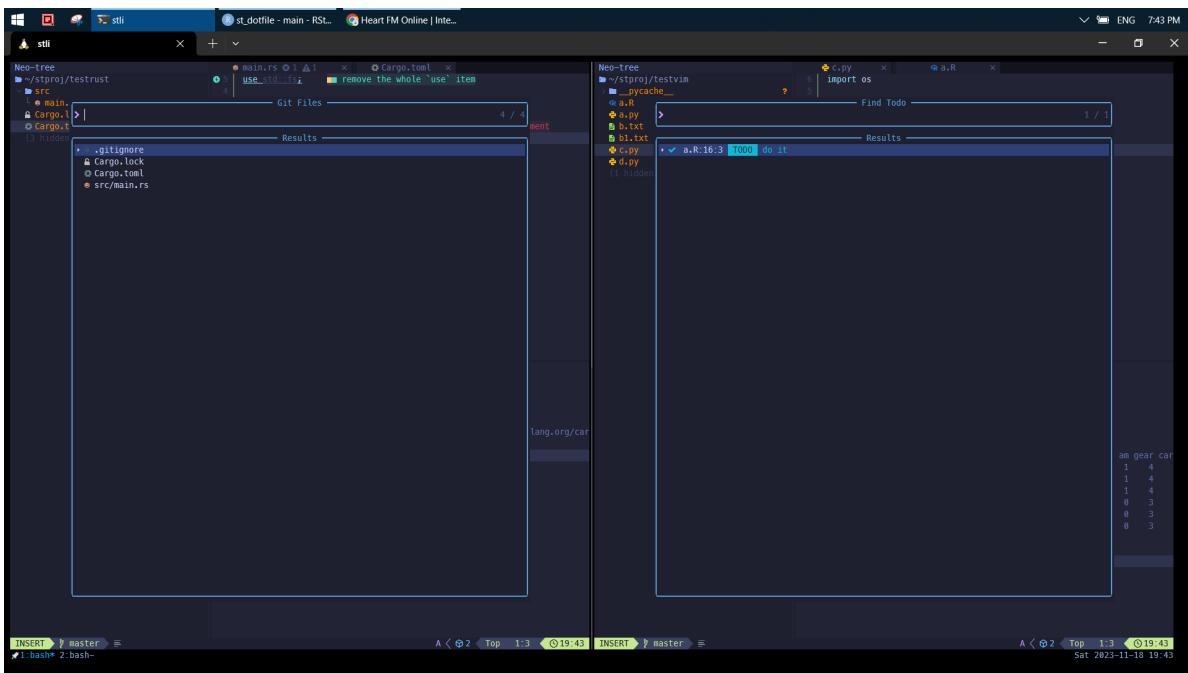


Figure 1.11: Tmux - Nvim 3

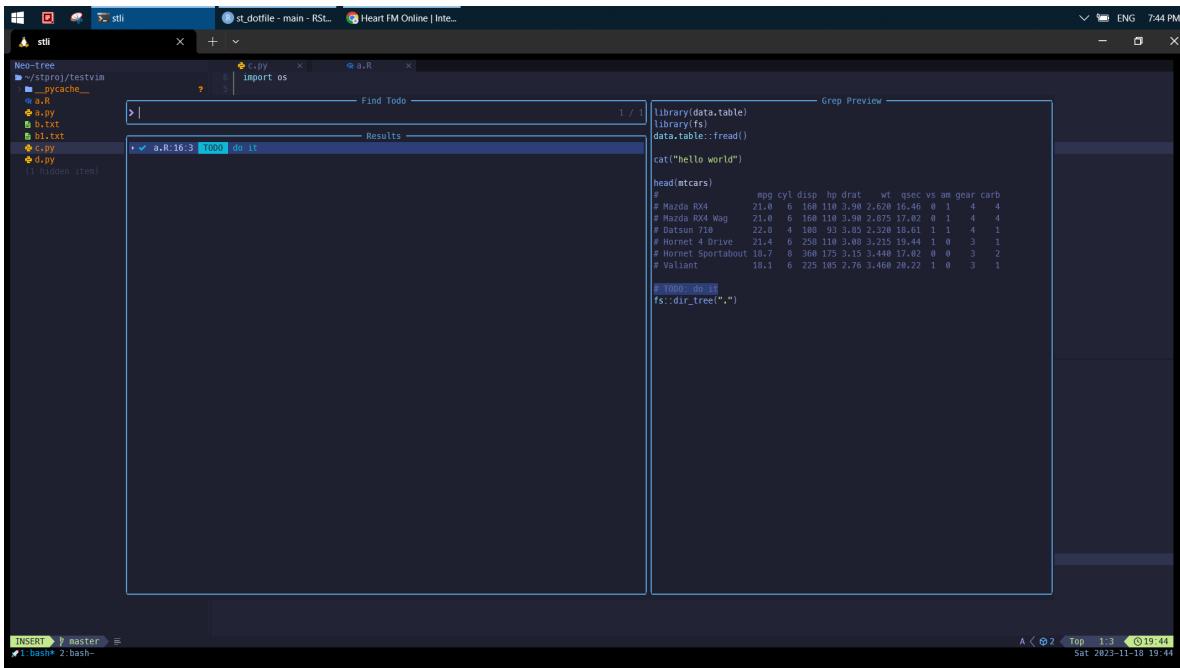


Figure 1.12: Tmux -Nvim 4

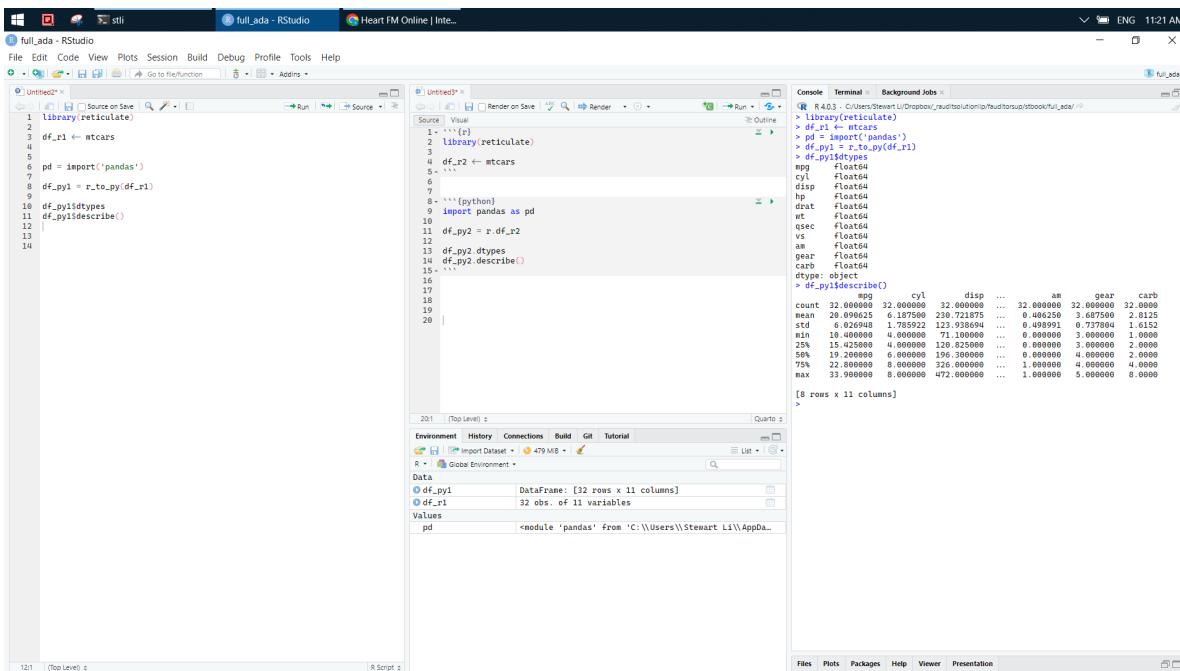


Figure 1.13: RStudio - R

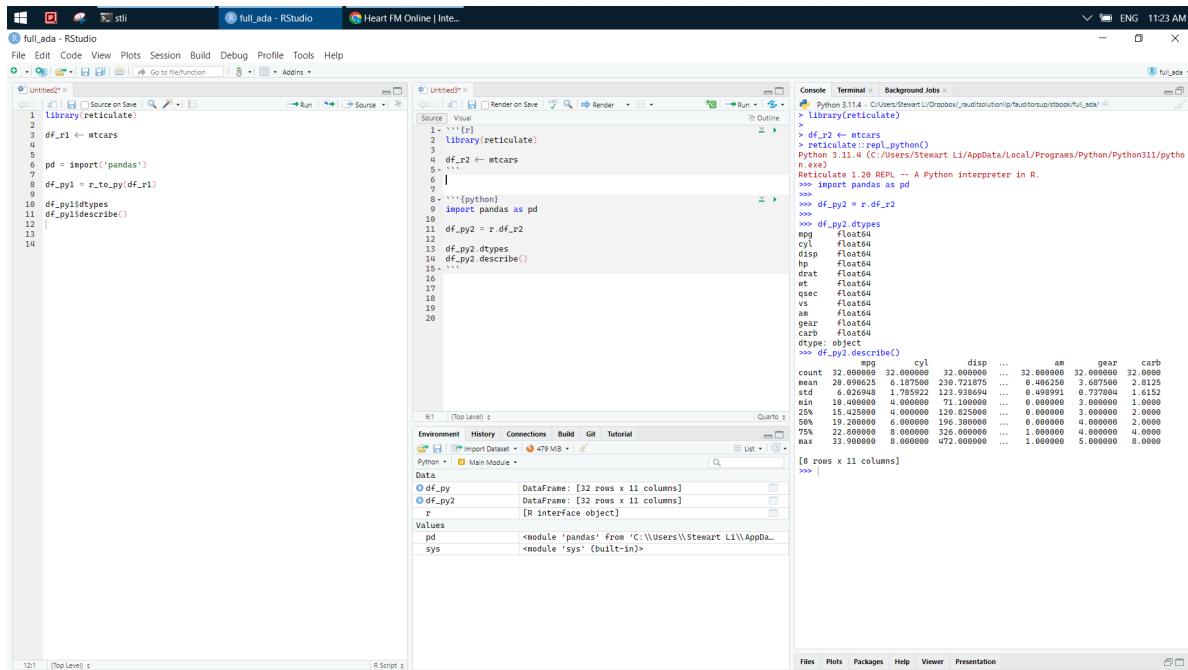


Figure 1.14: RStudio - Python

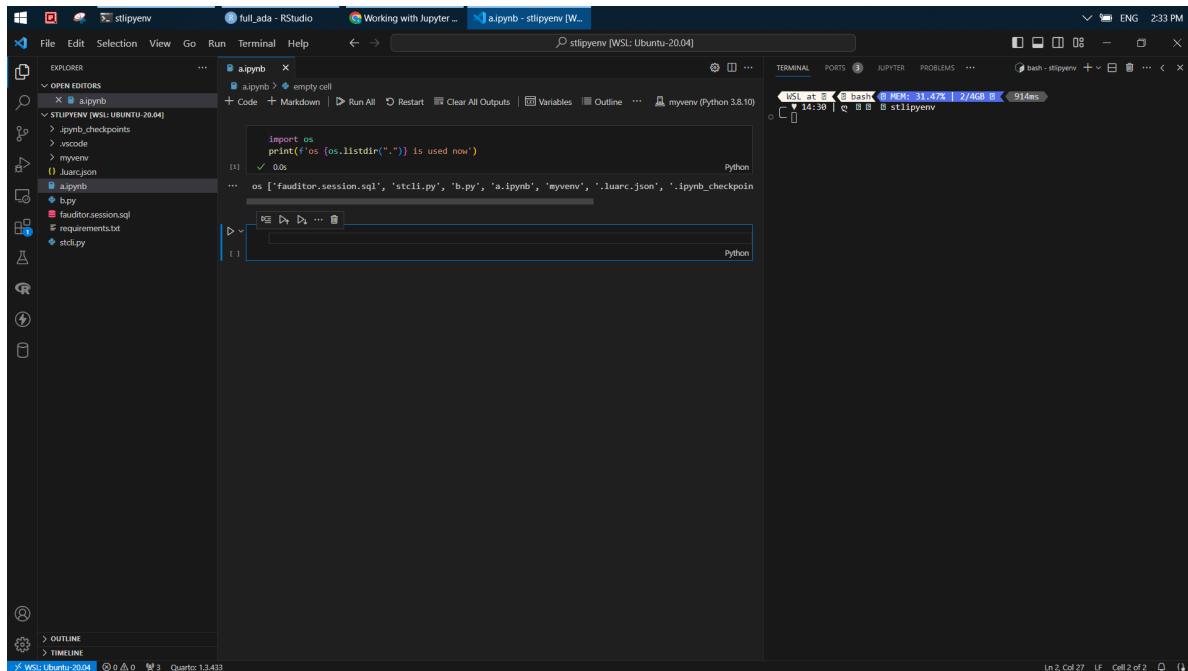


Figure 1.15: VS Code in Linux - Jupyter

A screenshot of the Visual Studio Code interface running on a Linux system. The title bar shows multiple tabs: 'full_ada - RStudio', 'Heart FM Online | Inte...', 'Interactive - b.py - stlipyenv', and 'stlipyenv [WSL: UBUNTU-20.04]'. The main area has a dark theme. On the left is the Explorer sidebar with a tree view of files and folders, including 'OPEN EDITORS' (b.py), 'GROUP 1' (b.py), 'GROUP 2' (Interactive - b.py), and 'STLIPYENV (WSL: UBUNTU-20.04)' (jupyter_checkpoints, .vscode, myenv, Juancion, a.ipynb, b.py, fauditor.session.sql, requirements.txt, stdlib). The center is the code editor with a file named 'b.py' containing the following code:

```
1 # %%\n2 import pandas as pd\n3\n4 print(f"pandas [{pd.__version__}] is used now")\n5\n6 # %%\n7 Run Cell | Run Above | Debug Cell
```

To the right is the 'Interactive' panel titled 'Interactive - b.py X'. It shows the message 'Connected to myenv (Python 3.8.10)'. Below that is a command history:

- ✓ import pandas as pd...
- ... pandas 2.0.2 is used now

At the bottom of the code editor, there's a status bar with 'Type "python" code here and press Shift+Enter to run' and 'Cell 3 of 3'.

Figure 1.16: VS Code in Linux - Interactive cell

A screenshot of the Visual Studio Code interface running on a Windows system. The title bar shows multiple tabs: 'full_ada - RStudio', 'Heart FM Online | Inte...', 'b.py - testvim - Visual ...', and 'testvim'. The main area has a dark theme. On the left is the Explorer sidebar with a tree view of files and folders, including 'OPEN EDITORS' (b.py), 'TESTVIM' (a.R, b.py), and 'b.py'. The center is the code editor with a file named 'b.py' containing the following code:

```
1 import os\n2\n3 print(f"os.listdir('.') is used now")\n4\n5 # %%\n6 import pandas as pd\n7 print(f"pandas [{pd.__version__}] is used now")\n8\n9 # %%\n10 Run Cell | Run Below | Debug Cell\n11 Run Cell | Run Above | Debug Cell
```

To the right is the Terminal panel titled 'TERMINAL' with the output:

```
Stewart: L1@DESKTOP-HCEU07A MINGW64 ~/Desktop/testvim\n$ python b.py\nos ['a.R', 'b.py'] is used now\npandas 2.0.3 is used now\n$
```

At the bottom of the code editor, there's a status bar with 'R (not attached) In 4, Col 1 Spaces:4 UTF-8 CR LF Python 3.11.4 64-bit Go Live Prettier'.

Figure 1.17: VS Code in Windows - Script

A screenshot of the Visual Studio Code interface on Windows. The title bar shows multiple tabs: 'full_ada - RStudio', 'Heart FM Online | Inte...', 'b.py - testvim - Visual...', and 'testvim'. The left sidebar has an 'EXPLORER' section with files 'a.R' and 'b.py'. The main area has an 'EDITOR' tab for 'b.py' containing the following code:

```
1 import os
2 print(f'os.listdir(".") is used now')
3
4 #%%
5 import pandas as pd
6 print(f'pandas ({pd.__version__}) is used now')
```

To the right is an 'Interactive' cell titled 'Interactive - b.py X' with the command 'import pandas as pd' and its output 'pandas 2.0.3 is used now'. At the bottom, a status bar shows 'R (not attached) Ln3, Col1 Spaces:4 UTF-8 CRLF Python 3.11.4 64-bit Go Live Prettier'.

Figure 1.18: VS Code in Windows - Interactive cell

A screenshot of the Visual Studio Code interface on Windows. The title bar shows multiple tabs: 'full_ada - RStudio', 'Heart FM Online | Inte...', 'a.R - testvim - Visual...', and 'R Graphics: Device 2 (ACTIVE)'. The left sidebar has an 'EXPLORER' section with files 'a.R' and 'b.py'. The main area has an 'EDITOR' tab for 'a.R' containing the following code:

```
> head(mtcars)
library(dplyr)
2
3 head(mtcars)
4 plot(mtcars)
```

To the right is a 'TERMINAL' tab showing the output of the R code, which includes the first few rows of the mtcars dataset and a scatter plot of mpg vs wt. Below the terminal is an 'R Graphics: Device 2 (ACTIVE)' window displaying a 10x10 correlation matrix of the mtcars dataset variables. The variables listed are mpg, cyl, disp, hp, drat, wt, qsec, vs, am, gear, carb.

Figure 1.19: VS Code in Windows - R

It is vital to create a proper **folder** structure along with config file as you are able to move quickly and organize your scripts better. I run a command line tool (written in R) from GitBash and PowerShell to do it.

The screenshot shows a Windows desktop environment with several open windows:

- RStudio:** A code editor window titled "full_ada - RStudio" containing the "full_ada.R" script. The script is an R script that handles folder creation and configuration based on command-line arguments. It includes functions for creating folders, writing configuration files, and managing project environments.
- GitBash:** A terminal window titled "MINGW64/c/Users/Stewart Li/Desktop/full_ada" showing the command "full_ada" being run. The output shows the script executing its logic to create a folder structure and write configuration files.
- File Explorer:** A file browser window titled "C:\Users\Stewart Li\Desktop" showing the contents of the "Desktop" folder. It lists three items: "desktop.ini" (376 B, modified Mar 8, 2021, 6:49 PM), ".DirIndex" (6505 kB, modified Mar 18, 2016, 12:17 PM), and "td" (empty folder).

Figure 1.20: CLI R - GitBash 1

Figure 1.21: CLI R - GitBash 2

The screenshot shows the RStudio interface with two main panes. The left pane displays the R script 'setup_term.R'. The right pane shows a terminal window with the command 'cd /Users/Stewart.Li/Desktop/newco -d' followed by a listing of the directory structure. The bottom navigation bar indicates the current tab is 'R Script'.

```
#!/usr/bin/env R --vanilla
# setup_term.R

1 #!/usr/bin/env C:/Program Files/R/R-4.0.3/bin/Rscript.exe
2
3
4 # Manual
5 # Run from GitBash
6 # Run logical functions:
7 # 1. If folders do not exist, create them, view dir tree, write config.yml
8 # 2. If folders do exist, list jobs in prod and create rproj. rproj name conflict is handled.
9 # 10. Create a newco folder
10 chmod +x ./setup_term.R
11 ./setup_term.R -h
12 ./setup_term.R newco -d
13 ./setup_term.R newco -e prod -r testmeil
14
15
16 # Load
17
18 # Helper
19# Main
20
21 scaffold <- function(f = "scaffold", seedir = TRUE, dev = "uat", rproj) {
22   cdesk, folder.level1, folder.level2, file.level1, file.level2) <- %> folders()
23   if(!dir.exists(paste(desk, f, sep = '/'))){
24     create_job(desk, f, dev, rproj)
25   }
26   create.folder(desk, f, folder.level1, folder.level2, file.level1, file.level2, seedir)
27   write.config(paste(desk, f, "tbox", sep = '/'), paste(desk, f, file.level2, sep = '/'))
28 }
29
30
31 # Parser
32 p <- arg_parser("Scaffold the folder structure in a new laptop")
33 p <- add_argument(p, "folders", help = "the path to the folder names", type = "character")
34 p <- add_argument(p, "rproj", help = "the path to the rproj file", type = "character")
35 p <- add_argument(p, "--env", help = "choose dev environment", type = "character", default = 'uat')
36 p <- add_argument(p, "--rproj", help = "create a Rproj", type = "character")
37 p <- add_argument(p, "tbox", help = "the path to the tbox file", type = "character")
38
39 args <- parse_args(p)
40 scaffold(f = args$folders, args$dev, args$rproj)
41
42
```

```
MINGW64/c/Users/Stewart.Li/Desktop/_rauditsolution1lp/fauditorsup/stbook/full_ada
$ ./setup_term.R newco -d
C:/Users/Stewart.Li/Desktop/newco
+- prod
  +- config.yml
  +- data
  +- proj
  +- res
+- README.qmd
+- sbin
+- stbox
+- tbox
+- tbd
+- uat
  +- proj

Stewart.Li@DESKTOP-HCEU07A MINGW64 ~\Dropbox/_rauditsolution1lp/fauditorsup/stbook/full_ada
$
```

Files Plots Packages Help Viewer Presentation
New Folder New Blank File < Delete Rename More
C:\Users\Stewart.Li\Desktop\newco\prod
Name Size Modified
config.yml 200 B Oct 12, 2023, 10:37 AM
data
proj
raw
res

Figure 1.22: CLI R - GitBash 3

The screenshot shows the RStudio interface with two panes. The left pane displays the `setup.Term.R` script, which is a series of R code for managing folder structures and environments. The right pane shows a terminal window titled "MINGW64/c/Users/Stewart Li/Desktop/full_ada" where the script is being run. The terminal output shows the script executing and creating a directory named "testmel". A file browser window is also visible at the bottom, showing files like "ignore", "R", and "testme1.Rproj".

```

#!/usr/bin/env R/R-4.0.3/bin/Rscript.exe
# Manual
# Run it from Gitbash
# Logical functions:
# 1. If folders do not exist, create them, view dir tree, write config.yml.
# 2. If folders do exist, list jobs in prod and create rproj. rproj name conflict is handled.
# Examples:
# 1. chmod +x ./supp/setup.Term.R
# 2. ./supp/setup.Term.R -h
# 3. ./supp/setup.Term.R newco -d
# 4. ./supp/setup.Term.R newco -e prod -r testmel
#
# Load
# Helper
# Main
scraftold <- function(f = "stliproj", seedir = TRUE, dev = "uat", rproj) {
  cdesk, folder_level1, folder_level2, file_level1, file_level2, %<-% folders()
  if(dir.exists(paste(desk, f, sep = '/'))){
    create.job(desk, f, dev, rproj)
  } else {
    create.folder(desk, f, folder_level1, folder_level2, file.level1, file.level2, seedir)
    write.config(paste(desk, f, "stbox", sep = '/'), paste(desk, f, file.level2, sep = '/'))
  }
}

# Parser
args <- parse.args(p)
scraftold = args$folder
args$folder = args$dev
args$dev = args$rproj

```

Figure 1.23: CLI R - GitBash 4

This screenshot is identical to Figure 1.23, showing the RStudio interface with the `setup.Term.R` script in the left pane and a terminal window in the right pane. The terminal shows the script running and creating a "testmel" directory. The file browser at the bottom is also the same.

```

#!/usr/bin/env R/R-4.0.3/bin/Rscript.exe
# Manual
# Run it from Gitbash
# Logical functions:
# 1. If folders do not exist, create them, view dir tree, write config.yml.
# 2. If folders do exist, list jobs in prod and create rproj. rproj name conflict is handled.
# Examples:
# 1. chmod +x ./supp/setup.Term.R
# 2. ./supp/setup.Term.R -h
# 3. ./supp/setup.Term.R newco -d
# 4. ./supp/setup.Term.R newco -e prod -r testmel
#
# Load
# Helper
# Main
scraftold <- function(f = "stliproj", seedir = TRUE, dev = "uat", rproj) {
  cdesk, folder_level1, folder_level2, file_level1, file_level2, %<-% folders()
  if(dir.exists(paste(desk, f, sep = '/'))){
    create.job(desk, f, dev, rproj)
  } else {
    create.folder(desk, f, folder_level1, folder_level2, file.level1, file.level2, seedir)
    write.config(paste(desk, f, "stbox", sep = '/'), paste(desk, f, file.level2, sep = '/'))
  }
}

# Parser
args <- parse.args(p)
scraftold = args$folder
args$folder = args$dev
args$dev = args$rproj

```

Figure 1.24: CLI R - GitBash 5

The screenshot shows the RStudio interface with an R script named `setup.R` open. The script contains code for creating a folder structure based on command-line arguments. A separate Windows PowerShell window is running in the background, showing the command `.\run_setup.bat newco -d` being executed. The PowerShell window also displays the resulting folder structure in the current directory.

```

# Manual
# Run it from Gitbash
# Two logical functions:
# 1. If folders do not exist, create them, view dir tree, write config.yml.
# 2. If folders do exist, list jobs in prod and create rproj. rproj name conflict is handled.
# Examples:
# 1. chmod +x ./supp/setup_term.R
# 2. ./supp/setup_term.R -h
# 3. ./supp/setup_term.R newco -d
# 4. ./supp/setup_term.R newco -e prod -r testml
# Load
# Helper
# Main
scffold <- function(f = "stlproj", seedir = TRUE, dev = "uat", rproj) {
  cdesk, folder_level1, folder_level2, file_level1, file_level2, sep = "%-%"
  if(dir.exists(paste(desk, f, sep = '/'))){
    create_job(desk, f, dev, rproj)
  } else {
    create_folder(desk, f, folder_level1, folder_level2, file_level1, file_level2, seedir)
    write_config(paste(desk, f, "stbox", sep = '/'), paste(desk, f, file_level2, sep = '/'))
  }
}

# Parser
p <- arg_parser("Scaffold the folder structure in a new laptop")
p <- add_argument(p, "folder", help = "provide a folder name", type = "character")
p <- add_argument(p, "dev", help = "choose the folder structure", flag = TRUE)
p <- add_argument(p, "--env", help = "choose dev environment", type = "character", default = "uat")
p <- add_argument(p, "--rproj", help = "create a Rproj", type = "character")
scffold(f = argv$folder, argv$dev, argv$env, argv$rproj)

args <- parse_args(p)
scffold(f = argv$folder, argv$dev, argv$env, argv$rproj)

```

Figure 1.25: CLI R - PowerShell 1

This screenshot is similar to Figure 1.25, showing the RStudio interface with the `setup.R` script and a Windows PowerShell window. In this version, the PowerShell window shows the command `PS C:\Users\Stewart Li\Desktop\newco>` and the resulting folder structure in the `newco` directory. The RStudio interface shows the script code again.

```

# Manual
# Run it from Gitbash
# Two logical functions:
# 1. If folders do not exist, create them, view dir tree, write config.yml.
# 2. If folders do exist, list jobs in prod and create rproj. rproj name conflict is handled.
# Examples:
# 1. chmod +x ./supp/setup_term.R
# 2. ./supp/setup_term.R -h
# 3. ./supp/setup_term.R newco -d
# 4. ./supp/setup_term.R newco -e prod -r testml
# Load
# Helper
# Main
scffold <- function(f = "stlproj", seedir = TRUE, dev = "uat", rproj) {
  cdesk, folder_level1, folder_level2, file_level1, file_level2, sep = "%-%"
  if(dir.exists(paste(desk, f, sep = '/'))){
    create_job(desk, f, dev, rproj)
  } else {
    create_folder(desk, f, folder_level1, folder_level2, file_level1, file_level2, seedir)
    write_config(paste(desk, f, "stbox", sep = '/'), paste(desk, f, file_level2, sep = '/'))
  }
}

# Parser
p <- arg_parser("Scaffold the folder structure in a new laptop")
p <- add_argument(p, "folder", help = "provide a folder name", type = "character")
p <- add_argument(p, "--dir", help = "show the folder structure", flag = TRUE)
p <- add_argument(p, "--env", help = "choose dev environment", type = "character", default = "uat")
p <- add_argument(p, "--rproj", help = "create a Rproj", type = "character")
scffold(f = argv$folder, argv$dev, argv$env, argv$rproj)

args <- parse_args(p)
scffold(f = argv$folder, argv$dev, argv$env, argv$rproj)

```

Figure 1.26: CLI R - PowerShell 2

2 ELT

Consider the following examples to establish a data pipeline.

1. A zip file lands in data lake (`s3/minio`) daily.
2. Execute scripts in the server (`ec2`) to download/unzip/select/upload files based on `mtime`. It produces a file (`csv`) to track work done at the agreed cut-off time (`cron`). AWS `lambda` is another option.
3. `snowflake` external stage (`s3`) is triggered by a file (`txt`) to kicks off `snowpipe` and ingest data to DB as `variant`. Similar storage are `databrick`, `dremio`, `clickhouse`. The preferred formats are `parquet`, `iceberg`, `ADBC`.
4. Move data between platforms via `airbyte`.
5. Validate and transform DB raw to DB mart through `dbt`.
6. Automatize the process by a task scheduler `prefect`, `airflow`, `dagster`.
7. Create a dashboard for DB mart via `metabase`, `superset`.

The screenshot shows a web-based data exploration environment. On the left, there's a sidebar titled "My databases" containing a tree view of database structures. The main area is titled "My Notebook" and contains a code editor with the following SQL query:

```
1 SELECT text, by, id, COUNT(*) AS n
2 FROM sample_data.hn.hacker_news GROUP BY ALL
3 ORDER BY n DESC
4 LIMIT 6;
```

Below the code editor, a message states "Query executed in 1.39 s. Row count: 6". A table displays the results:

text	by	id	n
In addition to being quite possibly the best free introduc...	sn9	31,213,459	1
>Especially seems unnecessary->p>An option that sh...	forgotpwd16	31,038,085	1
What you#x27;ve mentioned is objectively wrong and ...	fzeroracer	31,650,610	1
Oh i love talcscale, it is so performant and just works (m...	wjokinen	29,935,984	1
I'll be watching some YouTube with kids, or some ...	Borganicbits	32,765,091	1
I have the same issue(s) on my third set of earpods and...	arthurmorgan	32,765,097	1

Figure 2.1: DuckDB cloud

The screenshot shows a terminal window titled 'stli' running on a Windows operating system. The window displays the following command-line session:

```

d:\stli>true ~./testf > cd
d:\stli>true ~ > ./duckdb md:
-- Loading resources from /home/stli/.duckdbrc
v0.9.2 3c695d9ba9
Enter ".help" for usage hints.
==> show databases;
  database_name
  varchar
my_db
sample_data
  ==> SELECT text, by, id, COUNT(*) AS n
> FROM sample_data.hn.hacker_news GROUP BY ALL
> ORDER BY n DESC
> LIMIT 6;

```

Below the command history, a table is displayed with the following data:

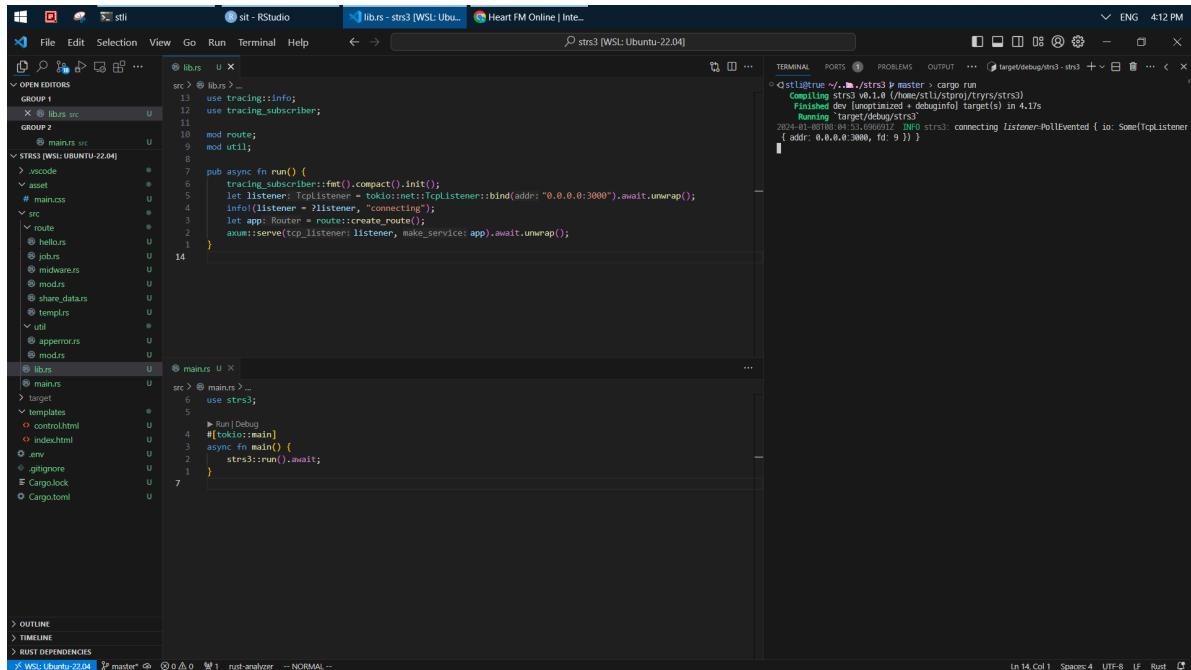
text	by	id	n
varchar	varchar	int64	int64
Roberts raise up.	rednerrus	32201944	1
Yep, I use this to get vimium-FF on Fennec. It's a little fiddly to setup but it's worth it if you're going for the cellphone model of ownership. Pay large for L.	re10	29798774	1
I do this, and it's very nice to tell who lost'f sold my email. psAlso I...	Shmueler2020	29798771	1
I have a learning disability related to some incredibly common cognitive issues t.	Shared444	32683683	1
You can theoretically send email to an IP address directly, like someone@[127.0...	DrewWebDesign	33807785	1
	csande17	33305117	1

The terminal window also shows the status bar indicating 'Thu 2024-01-25 17:08'.

Figure 2.2: DuckDB terminal

3 Example 1

It is very useful to create a micro service API internally.



```
git clone https://github.com/rust-lang/rustls.git
cd rustls
cargo build --release
cargo run
```

The screenshot shows a terminal window with the following output:

```
git clone https://github.com/rust-lang/rustls.git
cd rustls
cargo build --release
cargo run
```

The terminal shows the command being run and its output, indicating the application is running on port 3000.

Figure 3.1: Web server

```
httr2::request('localhost:3000/share') %>%
  httr2::req_perform() %>%
  httr2::res_body_string()
```

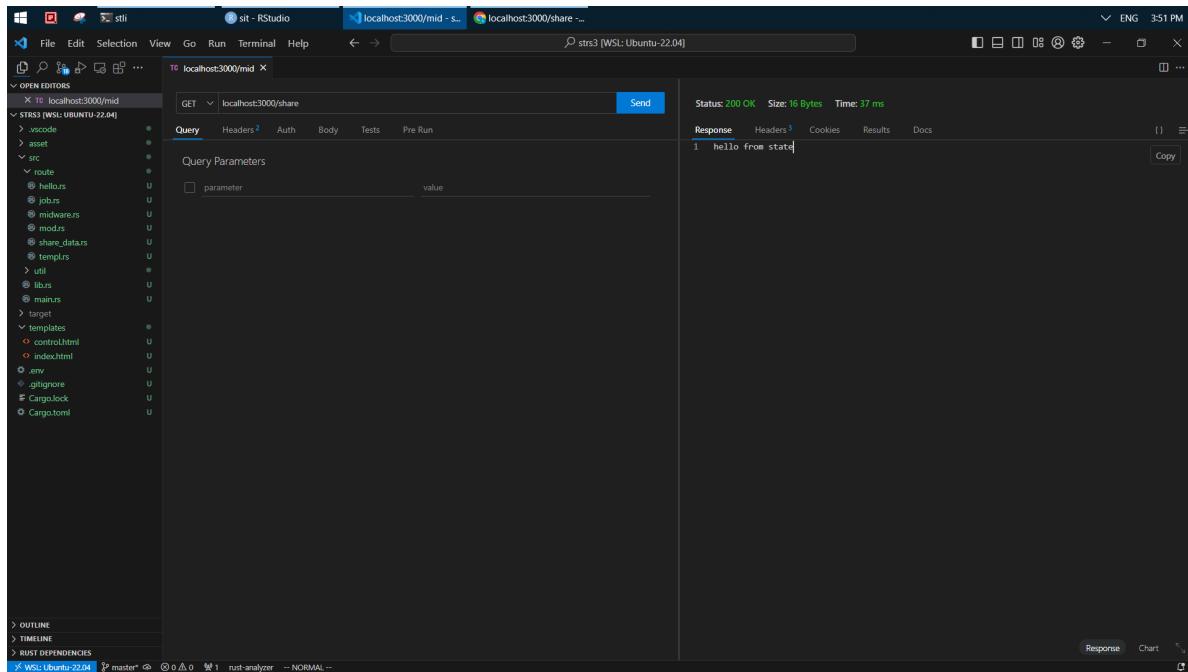


Figure 3.2: Get

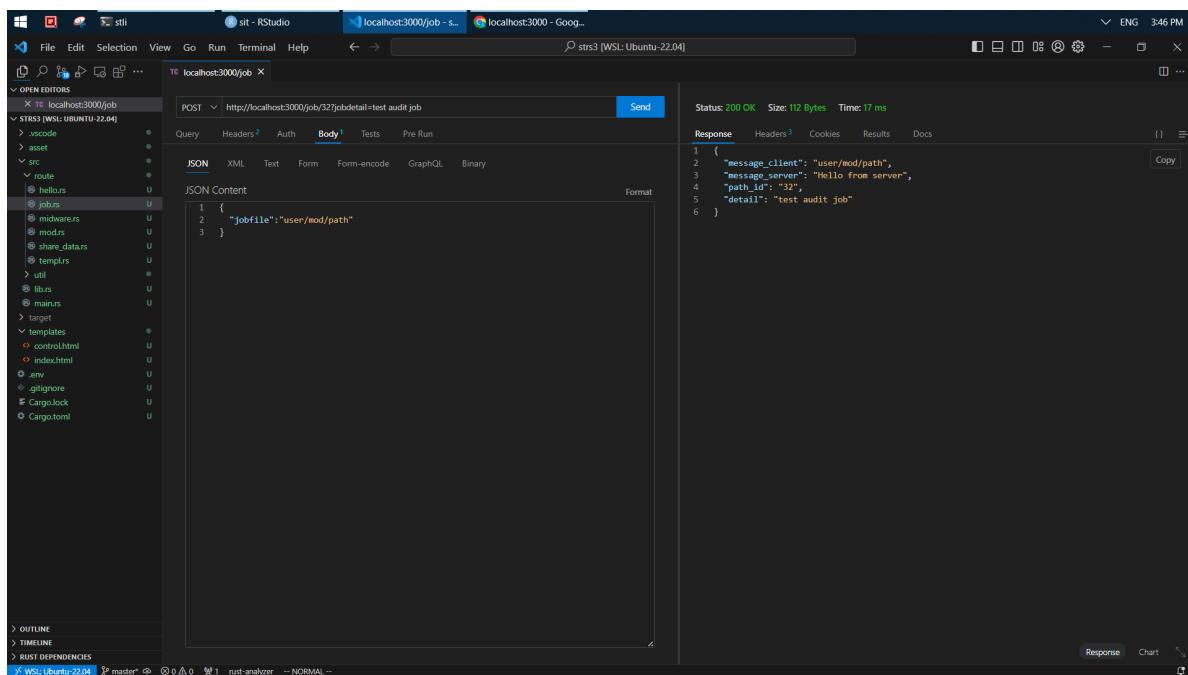


Figure 3.3: Post

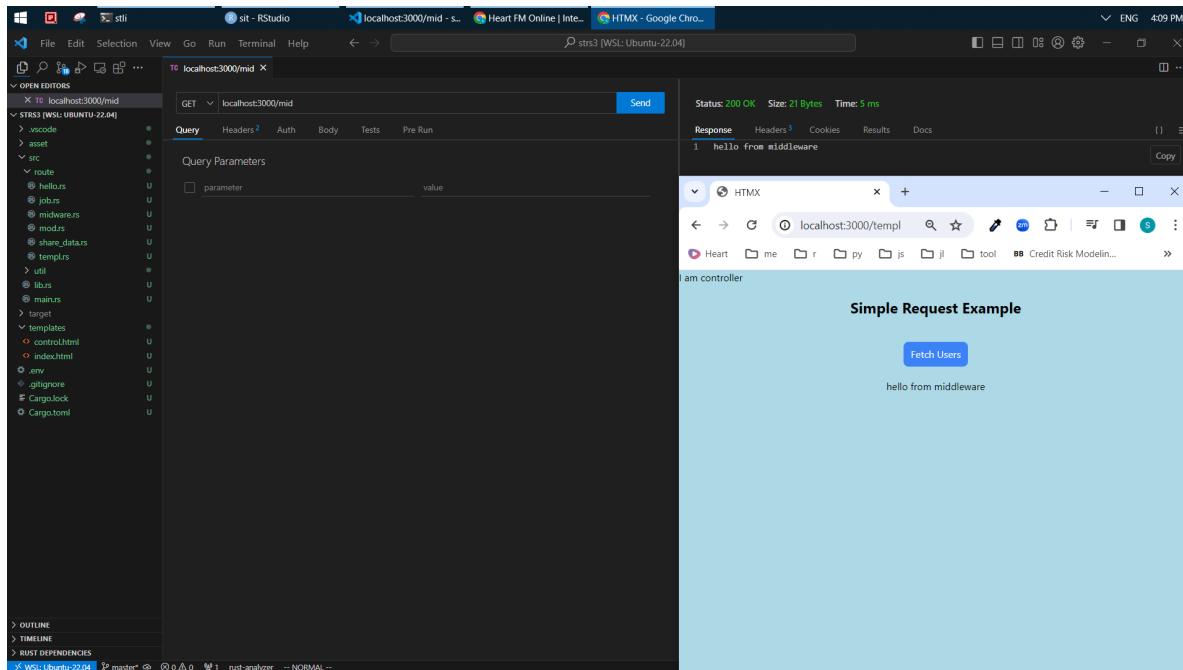


Figure 3.4: Template

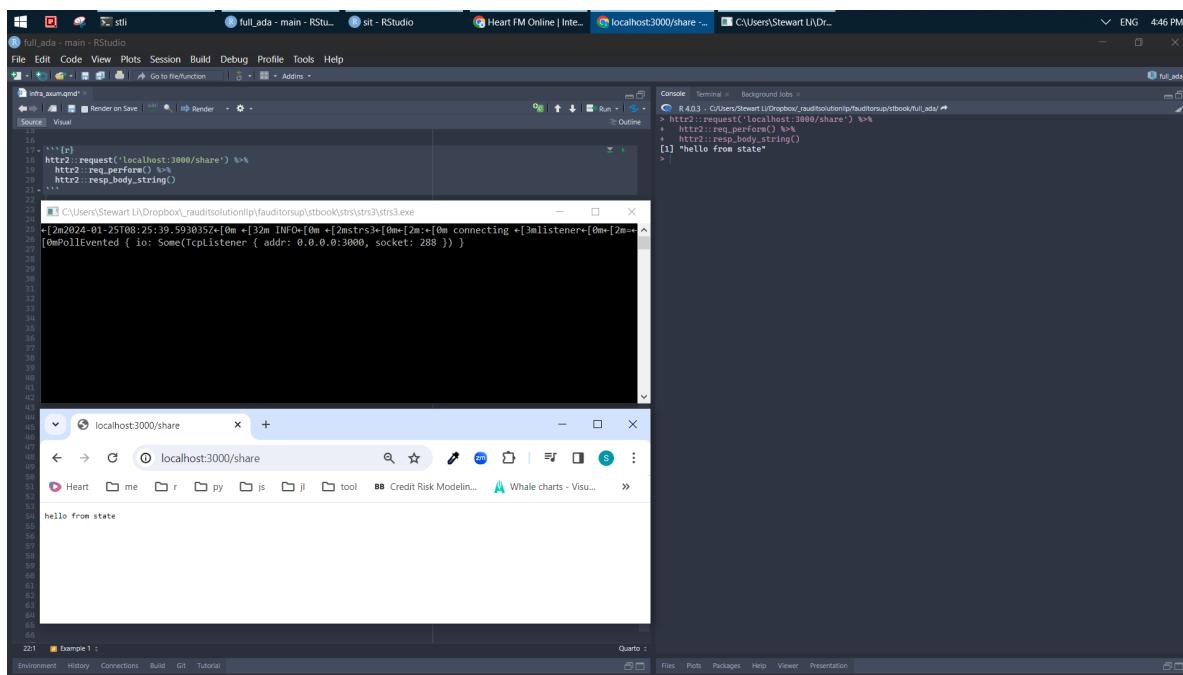


Figure 3.5: R client

Part II

Data tools

SQL, R, Python, Julia, Rust, and JavaScript can be used interchangeably to perform data work at most of the time. Choose programming languages and relevant packages based on your needs and personal preference.

Assess your IO scenario after considered the followings.

How big is data?

1. Memory:

- `datatable`, `collapse`, `duckdb`, `polars`,
- `ibis`, `DataFusion`, `deltalake`,

2. Hard disk:

- `arrow`,

3. Cluster:

- `spark`, `dask`,

Where data lives?

1. DB:

- `DBI`, `odbc`, `SQLAlchemy`, `connectorx`, `sqlx`,

2. SFTP:

- `RCurl`, `paramiko`,

3. Blob:

- `pins`, `aws.s3`, `s3fs`, `boto3`,

In what form? The preferred file types are `txt`, `csv`, `parquet`, `feather`.

1. Excel:

- `tidyxl`, `unpivotr`, `openxlsx`, `openpyxl`,

2. Word:

- `officer`, `docx`,

3. PPT:

- `officer`, `python-pptx`,

4. PDF:

- `pdftools`, `PDFminer`, `PyPDF2`, `pdfplumber`,

5. SAS:

- `haven`,

6. Image:

- `magick`, `tesseract`, `pillow`, `cv2`,

7. Geo:

- `sf`, `countrycode`,

8. API:

- `httr2`, `request`, `reqwest`,

- `jsonlite`, `yaml`, `toml`,

9. Website:

- `html`, `xml`, `rvest`, `bs4`,

- `v8`, `chromote`, `selenium`, `playwright`,

In what data structure and type?

1. Data type:
 - numeric, string, bool, factor, date,
2. Data collection:
 - list, vector, data.frame (cell/0 row/1 column),
3. Verb:
 - count/sort/select/filter/mutate/summarize/pivot/join,

Analysis work is to produce meaningful insight via slice dice. Classify a set of tools based on the following analytics steps. To reduce repetitive work, you can create functions, OOP, box, package, and cli.

1. Interact with DB:
 - dbplyr, dbplot, dbcooper,
2. Data cleaning:
 - base, tidyverse, pandas,
 - janitor, glue, tidylog,
 - waldo, diffobj, compareDF,
3. Data validation:
 - pointblank, validate, pandera, greate expectation, pydantic,
4. Data visualization¹:
 - grid, patchwork, ggfx, ggtext, showtext,
 - ragg, scales, formattable, sparkline,
 - gghighlight, ggforce,
 - imager, imagerExtra, ggimage, ggpibr,
 - igraph, ggraph, tidygraph, networkD3, visNetwork,
 - DiagrammeR, UpSetR, tmap,
5. Table:
 - gt, gtExtras, gtsummary, modelsummary,
 - flextable, kableExtra,
6. EDA:
 - skimr, naniar, visdat, inspectdf,
7. Stats:
 - corrplot, tidylo, widyr, broom,
8. Report:
 - quarto, whisker, target, jinja2,
9. API deploy:
 - vetiver, plumber, fastapi,
10. Dashboard:
 - shiny, htmltools, htmlwidgets, crosstalk, leaflet,
 - bslib, thematic, sass,
 - DT, reactable, reactablefmtr,
 - plotly, echarts4r, bokeh,

¹ggplot2 (Wickham 2016)

- `dash`, `streamlit`,
- 11. WASM:
 - `webr`, `pyodide`, `wasm_bindgen`,
- 12. GUI:
 - [PyAutoGUI](#),
 - `Tkinter`, [PyQt5](#),

Consider other utility tools when necessary.

1. Environment:

- `rvenv`, `venv`,

2. Helper:

- `cli`, `crayon`,

- `clipr`, `withr`, `callr`, `pingr`, `curl`,

3. Email:

- `blastula`, `emayili`, `smtplib`, `pywin32`,

4. Unzip:

- `archive`, `zipfile`,

5. FFI:

- `rlang`, `vctrs`, `lobstr`, `S7`,

- `cpp11`, `Rcpp`, `extindr`, `pyo3`, `bindgen`,

4 Polars

Command line tools allow you to do those repetitive data work easily. The following three examples are.

1. argparse and duckdb.
2. click and polars.
3. clap and polars.

The screenshot shows a Windows desktop environment. In the foreground, a code editor window titled 'stcli.py' is open, displaying Python code for a command-line interface using argparse and duckdb. The code defines functions for filtering and summarizing data from a DuckDB database. In the background, a terminal window titled 'stclienv [WSL: Ubuntu-20.04]' is running a DuckDB query on a 'finsample' dataset. The terminal output shows the results of the query, which includes columns: segment, country, product, cogs, profit, and date timestamp. The data shows three rows for different segments and countries, with their respective values for each column.

```
stcli.py
1 import argparse
2 import duckdb
3
4 def st_filter(db, tbl, ex="cogs > 200000"):
5     conn = duckdb.connect(db)
6     df = conn.execute(f"select * from {tbl}").df()
7     df_res = duckdb.filter(df, ex)
8     conn.close()
9     print(df_res)
10
11
12 def st_summarize(db, tbl):
13     conn = duckdb.connect(db)
14     df = conn.execute(f"select * from {tbl}").df()
15     df_res = duckdb.sql("""
16         """select segment, country, count(*) as n from df group by all order by n desc"""
17     )
18     conn.close()
19     print(df_res)
20
21
22 if __name__ == "__main__":
23     parser = argparse.ArgumentParser()
24     # namespace db contains multiple args
25     parser.add_argument("db", type=str, nargs="+")
26     # optional function
27     parser.add_argument(
28         "--filter",
29         dest="do",
30         action="store_const",
31         const=st_filter,
32         default=st_summarize,
33     )
34     args = parser.parse_args()
35     # st.filter(*args[0])
36     args.do(*args.db)
37
38
39
40 # python3 stcli.py ~/finsample.duckdb finsample
41 # python3 stcli.py ~/finsample.duckdb finsample --filter
42
```

Figure 4.1: CLI - argparse 1

Figure 4.2: CLI - argparse 2

Figure 4.3: CLI - argparse 3

```

stclick > ./stclick.py ...
59 @stat.command()
60 @click.pass_context
61 @click.argument('col', type=str)
62 @click.argument('n', type=int)
63 def topn(ctx, col, n):
64     res = ctx.obj.sort(pl.col(f'{col}'), descending=True).limit(n)
65     click.echo(res)

66 @main.group()
67 > def calc():
68     ...

69     @click.command()
70     @click.pass_context
71     @click.argument("output", type=click.File("w"), default="-", required=False)
72     @click.argument(["-highlight"], type=click.Choice(["red", "green"]),
73                   help="highlight data based on the provided threshold",
74                 )
75     @click.argument(["-n"], type=int,
76                   help="highlight [red|green] highlight data based on the provided threshold",
77                 )
78     @click.argument(["-highlight"], type=click.Choice(["red", "green"]),
79                   help="highlight data based on the provided threshold",
80                 )
81     @click.argument(["-n"], type=int,
82                   help="highlight [red|green] highlight data based on the provided threshold",
83                 )
84     def cond(ctx, output, highlight):
85         ...
86         choose your data
87         ...
88         res = ctx.obj.with_columns(
89             new(
90                 pl.when(pl.col("hp") > 200
91                         .then(pl.col("mpg").filter(pl.col("hp") > 200).sum())
92                         .otherwise(pl.col("mpg").filter(pl.col("hp") < 200).sum())
93                     )
94             .round(2)
95             .cast(pl.Utf8)
96         )
97         if highlight == "red":
98             res = res.with_columns(pl.col("new").map(lambda x: f"\033[91m{x} + \x1b[0m"))
99         else:
100             res = res.with_columns(pl.col("new").map(lambda x: f"\033[96m{x} + \x1b[0m"))
101
102         # click.echo(output)
103         click.echo(res)

104     if __name__ == "__main__":
105         main()
106
107
108
WSL at E: ~ basic [MEM: 27.4% | 2/4GB B] 191ms
python3 ./stclick/stclick.py --help
Usage: stclick.py [OPTIONS] INPUT COMMAND [ARGS]...
    data validation
        Options:
            --version Show the version and exit.
            --help Show this message and exit.

    Commands:
        about tool introduction
        calc conditional calculation
        stat data stats
WSL at E: ~ basic [MEM: 27.4% | 2/4GB B] 247ms
python3 ./stclick/stclick.py --version
data validation, version 0.01
WSL at E: ~ basic [MEM: 27.4% | 2/4GB B] 219ms
python3 ./stclick/stclick.py "/home/stli/stliproj/testvim/mtcars.csv" calc --help
data source: /home/stli/stliproj/testvim/mtcars.csv
Usage: stclick.py INPUT COMMAND [OPTIONS] [ARGS]...
    conditional calculation
        Options:
            --help Show this message and exit.

    Commands:
        condc choose your data
WSL at E: ~ basic [MEM: 28.3% | 2/4GB B] 196ms
python3 ./stclick/stclick.py "/home/stli/stliproj/testvim/mtcars.csv" calc condc --help
data source: /home/stli/stliproj/testvim/mtcars.csv
Usage: stclick.py INPUT calc condc [OPTIONS] [OUTPUT]
    choose your data
        Options:
            --highlight [red|green] highlight data based on the provided threshold
            --help Show this message and exit.

WSL at E: ~ basic [MEM: 27.4% | 2/4GB B] 196ms
python3 ./stclick/stclick.py "/home/stli/stliproj/testvim/mtcars.csv" stat topm disp 3
data source: /home/stli/stliproj/testvim/mtcars.csv
shape: (3, 12)
      mpg   cyl  disp   vs   am   gear   carb
0   18.0   6   160   0   0   1   4
1   18.0   6   160   1   1   1   4
2   14.3   8   235   1   1   2   3
3   22.8   4   108   1   1   1   1
4   19.2   6   160   1   0   1   4
5   17.8   6   160   1   0   1   4
6   18.7   8   167.6  0   1   1   5
7   18.0   8   167.6  1   1   1   5
8   14.3   8   235   1   0   1   4
9   22.8   4   108   1   1   1   1
10  19.2   6   160   1   0   1   4
11  15.2   8   160   1   0   1   4
12  18.0   8   167.6  0   1   1   5
13  15.2   8   160   1   0   1   4
14  18.0   8   167.6  0   1   1   5
15  16.4   8   160   1   0   1   4
16  20.8   4   108   1   1   1   1
17  15.2   8   160   1   0   1   4
18  18.0   8   167.6  0   1   1   5
19  18.0   8   167.6  1   1   1   5
20  14.3   8   235   1   0   1   4
21  22.8   4   108   1   1   1   1
22  19.2   6   160   1   0   1   4
23  15.2   8   160   1   0   1   4
24  18.0   8   167.6  0   1   1   5
25  15.2   8   160   1   0   1   4
26  18.0   8   167.6  0   1   1   5
27  16.4   8   160   1   0   1   4
28  20.8   4   108   1   1   1   1
29  15.2   8   160   1   0   1   4
30  18.0   8   167.6  0   1   1   5
31  15.2   8   160   1   0   1   4
32  18.0   8   167.6  0   1   1   5
33  16.4   8   160   1   0   1   4
34  20.8   4   108   1   1   1   1
35  15.2   8   160   1   0   1   4
36  18.0   8   167.6  0   1   1   5
37  15.2   8   160   1   0   1   4
38  18.0   8   167.6  0   1   1   5
39  16.4   8   160   1   0   1   4
40  20.8   4   108   1   1   1   1
41  15.2   8   160   1   0   1   4
42  18.0   8   167.6  0   1   1   5
43  15.2   8   160   1   0   1   4
44  18.0   8   167.6  0   1   1   5
45  16.4   8   160   1   0   1   4
46  20.8   4   108   1   1   1   1
47  15.2   8   160   1   0   1   4
48  18.0   8   167.6  0   1   1   5
49  15.2   8   160   1   0   1   4
50  18.0   8   167.6  0   1   1   5
51  16.4   8   160   1   0   1   4
52  20.8   4   108   1   1   1   1
53  15.2   8   160   1   0   1   4
54  18.0   8   167.6  0   1   1   5
55  15.2   8   160   1   0   1   4
56  18.0   8   167.6  0   1   1   5
57  16.4   8   160   1   0   1   4
58  20.8   4   108   1   1   1   1
59  15.2   8   160   1   0   1   4
60  18.0   8   167.6  0   1   1   5
61  15.2   8   160   1   0   1   4
62  18.0   8   167.6  0   1   1   5
63  16.4   8   160   1   0   1   4
64  20.8   4   108   1   1   1   1
65  15.2   8   160   1   0   1   4
66  18.0   8   167.6  0   1   1   5
67  15.2   8   160   1   0   1   4
68  18.0   8   167.6  0   1   1   5
69  16.4   8   160   1   0   1   4
70  20.8   4   108   1   1   1   1
71  15.2   8   160   1   0   1   4
72  18.0   8   167.6  0   1   1   5
73  15.2   8   160   1   0   1   4
74  18.0   8   167.6  0   1   1   5
75  16.4   8   160   1   0   1   4
76  20.8   4   108   1   1   1   1
77  15.2   8   160   1   0   1   4
78  18.0   8   167.6  0   1   1   5
79  15.2   8   160   1   0   1   4
80  18.0   8   167.6  0   1   1   5
81  16.4   8   160   1   0   1   4
82  20.8   4   108   1   1   1   1
83  15.2   8   160   1   0   1   4
84  18.0   8   167.6  0   1   1   5
85  15.2   8   160   1   0   1   4
86  18.0   8   167.6  0   1   1   5
87  16.4   8   160   1   0   1   4
88  20.8   4   108   1   1   1   1
89  15.2   8   160   1   0   1   4
90  18.0   8   167.6  0   1   1   5
91  15.2   8   160   1   0   1   4
92  18.0   8   167.6  0   1   1   5
93  16.4   8   160   1   0   1   4
94  20.8   4   108   1   1   1   1
95  15.2   8   160   1   0   1   4
96  18.0   8   167.6  0   1   1   5
97  15.2   8   160   1   0   1   4
98  18.0   8   167.6  0   1   1   5
99  16.4   8   160   1   0   1   4
100 18.0   8   167.6  0   1   1   5
101 15.2   8   160   1   0   1   4
102 18.0   8   167.6  0   1   1   5
103 16.4   8   160   1   0   1   4
104 20.8   4   108   1   1   1   1
105 15.2   8   160   1   0   1   4
106 18.0   8   167.6  0   1   1   5
107 15.2   8   160   1   0   1   4
108 18.0   8   167.6  0   1   1   5
WSL at E: ~ basic [MEM: 27.4% | 2/4GB B] 248ms
python3 ./stclick/stclick.py "/home/stli/stliproj/testvim/mtcars.csv" stat desc
data source: /home/stli/stliproj/testvim/mtcars.csv
shape: (9, 6)
      mpg   cyl  disp   vs   am   gear
0   18.0   6   160   0   0   1
1   18.0   6   160   1   1   1
2   14.3   8   235   1   1   2
3   22.8   4   108   1   1   1
4   19.2   6   160   1   0   1
5   17.8   6   160   1   0   1
6   18.7   8   167.6  0   1   1
7   18.0   8   167.6  1   1   1
8   14.3   8   235   1   0   1
9   22.8   4   108   1   1   1
10  19.2   6   160   1   0   1
11  15.2   8   160   1   0   1
12  18.0   8   167.6  0   1   1
13  15.2   8   160   1   0   1
14  18.0   8   167.6  0   1   1
15  16.4   8   160   1   0   1
16  20.8   4   108   1   1   1
17  15.2   8   160   1   0   1
18  18.0   8   167.6  0   1   1
19  15.2   8   160   1   0   1
20  18.0   8   167.6  0   1   1
21  16.4   8   160   1   0   1
22  20.8   4   108   1   1   1
23  15.2   8   160   1   0   1
24  18.0   8   167.6  0   1   1
25  15.2   8   160   1   0   1
26  18.0   8   167.6  0   1   1
27  16.4   8   160   1   0   1
28  20.8   4   108   1   1   1
29  15.2   8   160   1   0   1
30  18.0   8   167.6  0   1   1
31  15.2   8   160   1   0   1
32  18.0   8   167.6  0   1   1
33  16.4   8   160   1   0   1
34  20.8   4   108   1   1   1
35  15.2   8   160   1   0   1
36  18.0   8   167.6  0   1   1
37  15.2   8   160   1   0   1
38  18.0   8   167.6  0   1   1
39  16.4   8   160   1   0   1
40  20.8   4   108   1   1   1
41  15.2   8   160   1   0   1
42  18.0   8   167.6  0   1   1
43  15.2   8   160   1   0   1
44  18.0   8   167.6  0   1   1
45  16.4   8   160   1   0   1
46  20.8   4   108   1   1   1
47  15.2   8   160   1   0   1
48  18.0   8   167.6  0   1   1
49  15.2   8   160   1   0   1
50  18.0   8   167.6  0   1   1
51  16.4   8   160   1   0   1
52  20.8   4   108   1   1   1
53  15.2   8   160   1   0   1
54  18.0   8   167.6  0   1   1
55  15.2   8   160   1   0   1
56  18.0   8   167.6  0   1   1
57  16.4   8   160   1   0   1
58  20.8   4   108   1   1   1
59  15.2   8   160   1   0   1
60  18.0   8   167.6  0   1   1
61  15.2   8   160   1   0   1
62  18.0   8   167.6  0   1   1
63  16.4   8   160   1   0   1
64  20.8   4   108   1   1   1
65  15.2   8   160   1   0   1
66  18.0   8   167.6  0   1   1
67  15.2   8   160   1   0   1
68  18.0   8   167.6  0   1   1
69  16.4   8   160   1   0   1
70  20.8   4   108   1   1   1
71  15.2   8   160   1   0   1
72  18.0   8   167.6  0   1   1
73  15.2   8   160   1   0   1
74  18.0   8   167.6  0   1   1
75  16.4   8   160   1   0   1
76  20.8   4   108   1   1   1
77  15.2   8   160   1   0   1
78  18.0   8   167.6  0   1   1
79  15.2   8   160   1   0   1
80  18.0   8   167.6  0   1   1
81  16.4   8   160   1   0   1
82  20.8   4   108   1   1   1
83  15.2   8   160   1   0   1
84  18.0   8   167.6  0   1   1
85  15.2   8   160   1   0   1
86  18.0   8   167.6  0   1   1
87  16.4   8   160   1   0   1
88  20.8   4   108   1   1   1
89  15.2   8   160   1   0   1
90  18.0   8   167.6  0   1   1
91  15.2   8   160   1   0   1
92  18.0   8   167.6  0   1   1
93  16.4   8   160   1   0   1
94  20.8   4   108   1   1   1
95  15.2   8   160   1   0   1
96  18.0   8   167.6  0   1   1
97  15.2   8   160   1   0   1
98  18.0   8   167.6  0   1   1
99  16.4   8   160   1   0   1
100 18.0   8   167.6  0   1   1
101 15.2   8   160   1   0   1
102 18.0   8   167.6  0   1   1
103 16.4   8   160   1   0   1
104 20.8   4   108   1   1   1
105 15.2   8   160   1   0   1
106 18.0   8   167.6  0   1   1
107 15.2   8   160   1   0   1
108 18.0   8   167.6  0   1   1

```

Figure 4.4: CLI - click 1

```

stclick > ./stclick.py ...
59 @stat.command()
60 @click.pass_context
61 @click.argument('col', type=str)
62 @click.argument('n', type=int)
63 def topn(ctx, col, n):
64     res = ctx.obj.sort(pl.col(f'{col}'), descending=True).limit(n)
65     click.echo(res)

66 @main.group()
67 > def calc():
68     ...

69     @click.command()
70     @click.pass_context
71     @click.argument("output", type=click.File("w"), default="-", required=False)
72     @click.argument(["-highlight"], type=click.Choice(["red", "green"]),
73                   help="highlight data based on the provided threshold",
74                 )
75     @click.argument(["-n"], type=int,
76                   help="highlight [red|green] highlight data based on the provided threshold",
77                 )
78     @click.argument(["-highlight"], type=click.Choice(["red", "green"]),
79                   help="highlight data based on the provided threshold",
80                 )
81     @click.argument(["-n"], type=int,
82                   help="highlight [red|green] highlight data based on the provided threshold",
83                 )
84     def cond(ctx, output, highlight):
85         ...
86         choose your data
87         ...
88         res = ctx.obj.with_columns(
89             new(
90                 pl.when(pl.col("hp") > 200
91                         .then(pl.col("mpg").filter(pl.col("hp") > 200).sum())
92                         .otherwise(pl.col("mpg").filter(pl.col("hp") < 200).sum())
93                     )
94             .round(2)
95             .cast(pl.Utf8)
96         )
97         if highlight == "red":
98             res = res.with_columns(pl.col("new").map(lambda x: f"\033[91m{x} + \x1b[0m"))
99         else:
100             res = res.with_columns(pl.col("new").map(lambda x: f"\033[96m{x} + \x1b[0m"))
101
102         # click.echo(output)
103         click.echo(res)

104     if __name__ == "__main__":
105         main()
106
107
108
WSL at E: ~ basic [MEM: 27.4% | 2/4GB B] 191ms
python3 ./stclick/stclick.py --help
Usage: stclick.py [OPTIONS] INPUT COMMAND [ARGS]...
    data validation
        Options:
            --version Show the version and exit.
            --help Show this message and exit.

    Commands:
        about tool introduction
        calc conditional calculation
        stat data stats
WSL at E: ~ basic [MEM: 27.4% | 2/4GB B] 247ms
python3 ./stclick/stclick.py --version
data validation, version 0.01
WSL at E: ~ basic [MEM: 27.4% | 2/4GB B] 219ms
python3 ./stclick/stclick.py "/home/stli/stliproj/testvim/mtcars.csv" calc --help
data source: /home/stli/stliproj/testvim/mtcars.csv
Usage: stclick.py INPUT COMMAND [OPTIONS] [ARGS]...
    conditional calculation
        Options:
            --highlight [red|green] highlight data based on the provided threshold
            --help Show this message and exit.

WSL at E: ~ basic [MEM: 28.3% | 2/4GB B] 196ms
python3 ./stclick/stclick.py "/home/stli/stliproj/testvim/mtcars.csv" calc condc --help
data source: /home/stli/stliproj/testvim/mtcars.csv
Usage: stclick.py INPUT calc condc [OPTIONS] [OUTPUT]
    choose your data
        Options:
            --highlight [red|green] highlight data based on the provided threshold
            --help Show this message and exit.

WSL at E: ~ basic [MEM: 27.4% | 2/4GB B] 196ms
python3 ./stclick/stclick.py "/home/stli/stliproj/testvim/mtcars.csv" stat topm disp 3
data source: /home/stli/stliproj/testvim/mtcars.csv
shape: (3, 12)
      mpg   cyl  disp   vs   am   gear   carb
0   18.0   6   160   0   0   1   4
1   18.0   6   160   1   1   1   4
2   14.3   8   235   1   1   2   3
3   22.8   4   108   1   1   1   1
4   19.2   6   160   1   0   1   4
5   17.8   6   160   1   0   1   4
6   18.7   8   167.6  0   1   1   5
7   18.0   8   167.6  1   1   1   5
8   14.3   8   235   1   0   1   4
9   22.8   4   108   1   1   1   1
10  19.2   6   160   1   0   1   4
11  15.2   8   160   1   0   1   4
12  18.0   8   167.6  0   1   1   5
13  15.2   8   160   1   0   1   4
14  18.0   8   167.6  0   1   1   5
15  16.4   8   160   1   0   1   4
16  20.8   4   108   1   1   1   1
17  15.2   8   160   1   0   1   4
18  18.0   8   167.6  0   1   1   5
19  15.2   8   160   1   0   1   4
20  18.0   8   167.6  0   1   1   5
21  16.4   8   160   1   0   1   4
22  20.8   4   108   1   1   1   1
23  15.2   8   160   1   0   1   4
24  18.0   8   167.6  0   1   1   5
25  15.2   8   160   1   0   1   4
26  18.0   8   167.6  0   1   1   5
27  16.4   8   160   1   0   1   4
28  20.8   4   108   1   1   1   1
29  15.2   8   160   1   0   1   4
30  18.0   8   167.6  0   1   1   5
31  15.2   8   160   1   0   1   4
32  18.0   8   167.6  0   1   1   5
33  16.4   8   160   1   0   1   4
34  20.8   4   108   1   1   1   1
35  15.2   8   160   1   0   1   4
36  18.0   8   167.6  0   1   1   5
37  15.2   8   160   1   0   1   4
38  18.0   8   167.6  0   1   1   5
39  16.4   8   160   1   0   1   4
40  20.8   4   108   1   1   1   1
41  15.2   8   160   1   0   1   4
42  18.0   8   167.6  0   1   1   5
43  15.2   8   160   1   0   1   4
44  18.0   8   167.6  0   1   1   5
45  16.4   8   160   1   0   1   4
46  20.8   4   108   1   1   1   1
47  15.2   8   160   1   0   1   4
48  18.0   8   167.6  0   1   1   5
49  15.2   8   160   1   0   1   4
50  18.0   8   167.6  0   1   1   5
51  16.4   8   160   1   0   1   4
52  20.8   4   108   1   1   1   1
53  15.2   8   160   1   0   1   4
54  18.0   8   167.6  0   1   1   5
55  15.2   8   160   1   0   1   4
56  18.0   8   167.6  0   1   1   5
57  16.4   8   160   1   0   1   4
58  20.8   4   108   1   1   1   1
59  15.2   8   160   1   0   1   4
60  18.0   8   167.6  0   1   1   5
61  15.2   8   160   1   0   1   4
62  18.0   8   167.6  0   1   1   5
63  16.4   8   160   1   0   1   4
64  20.8   4   108   1   1   1   1
65  15.2   8   160   1   0   1   4
66  18.0   8   167.6  0   1   1   5
67  15.2   8   160   1   0   1   4
68  18.0   8   167.6  0   1   1   5
69  16.4   8   160   1   0   1   4
70  20.8   4   108   1   1   1   1
71  15.2   8   160   1   0   1   4
72  18.0   8   167.6  0   1   1   5
73  15.2   8   160   1   0   1   4
74  18.0   8   167.6  0   1   1   5
75  16.4   8   160   1   0   1   4
76  20.8   4   108   1   1   1   1
77  15.2   8   160   1   0   1   4
78  18.0   8   167.6  0   1   1   5
79  15.2   8   160   1   0   1   4
80  18.0   8   167.6  0   1   1   5
81  16.4   8   160   1   0   1   4
82  20.8   4   108   1   1   1   1
83  15.2   8   160   1   0   1   4
84  18.0   8   167.6  0   1   1   5
85  15.2   8   160   1   0   1   4
86  18.0   8   167.6  0   1   1   5
87  16.4   8   160   1   0   1   4
88  20.8   4   108   1   1   1   1
89  15.2   8   160   1   0   1   4
90  18.0   8   167.6  0   1   1   5
91  15.2   8   160   1   0   1   4
92  18.0   8   167.6  0   1   1   5
93  16.4   8   160   1   0   1   4
94  20.8   4   108   1  
```

```

stli@stli:~/stliproj/testv1m$ python3 ./stclick/stclick.py "/home/stli/stliproj/testv1m/mtcars.csv" calc condc
data source: /home/stli/stliproj/testv1m/mtcars.csv
shape: (32, 13)
  mpg cyl disp  am gear carb new
  ...
str   f64 164 f64  i64 164 i64 str
Mazda RX4 21.0 6 160.0 - 1 4 4 549.0
Mazda RX4 Wag 21.0 6 160.0 - 1 4 4 549.0
Datsun 710 22.8 4 108.0 - 1 4 1 549.0
Hornet 4 Drive 21.4 6 258.0 - 0 3 1 549.0
...
Ford Pantera L 15.8 8 351.0 - 1 5 4 93.9
Ferrari Dino 19.7 6 145.0 - 1 5 6 549.0
Maserati Bora 15.0 8 301.0 - 1 5 8 93.9
Volvo 142E 21.4 4 121.0 - 1 4 2 549.0

stli@stli:~/stliproj/testv1m$ grep V
Volvo

stli@stli:~/stliproj/testv1m$ python3 ./stclick/stclick.py "/home/stli/stliproj/testv1m/mtcars.csv" calc condc | grep V
Volvo

stli@stli:~/stliproj/testv1m$ python3 ./stclick/stclick.py "/home/stli/stliproj/testv1m/mtcars.csv" calc condc
data source: /home/stli/stliproj/testv1m/mtcars.csv
shape: (32, 13)
  mpg cyl disp  am gear carb new
  ...
str   f64 164 f64  i64 164 i64 str
Mazda RX4 21.0 6 160.0 - 1 4 4 549.0
Mazda RX4 Wag 21.0 6 160.0 - 1 4 4 549.0
Datsun 710 22.8 4 108.0 - 1 4 1 549.0
Hornet 4 Drive 21.4 6 258.0 - 0 3 1 549.0
...
Ford Pantera L 15.8 8 351.0 - 1 5 4 93.9
Ferrari Dino 19.7 6 145.0 - 1 5 6 549.0
Maserati Bora 15.0 8 301.0 - 1 5 8 93.9
Volvo 142E 21.4 4 121.0 - 1 4 2 549.0

```

Figure 4.6: CLI - click 3

```

stli@stli:~/stliproj/testv1m$ stpolars "/home/stli/stliproj/testv1m/mtcars.csv" about
data source: /home/stli/stliproj/testv1m/mtcars.csv
shape: (32, 13)

stli@stli:~/stliproj/testv1m$ stpolars "/home/stli/stliproj/testv1m/mtcars.csv" calc condc
data source: /home/stli/stliproj/testv1m/mtcars.csv
shape: (32, 13)
  mpg cyl disp  am gear carb new
  ...
str   f64 164 f64  i64 164 i64 str
Mazda RX4 21.0 6 160.0 - 1 4 4 549.0
Mazda RX4 Wag 21.0 6 160.0 - 1 4 4 549.0
Datsun 710 22.8 4 108.0 - 1 4 1 549.0
Hornet 4 Drive 21.4 6 258.0 - 0 3 1 549.0
...
Ford Pantera L 15.8 8 351.0 - 1 5 4 93.9
Ferrari Dino 19.7 6 145.0 - 1 5 6 549.0
Maserati Bora 15.0 8 301.0 - 1 5 8 93.9
Volvo 142E 21.4 4 121.0 - 1 4 2 549.0

stli@stli:~/stliproj/testv1m$ python3 ./stclick/stclick.py "/home/stli/stliproj/testv1m/mtcars.csv" calc condc
data source: /home/stli/stliproj/testv1m/mtcars.csv
shape: (32, 13)
  mpg cyl disp  am gear carb new
  ...
str   f64 164 f64  i64 164 i64 str
Mazda RX4 21.0 6 160.0 - 1 4 4 549.0
Mazda RX4 Wag 21.0 6 160.0 - 1 4 4 549.0
Datsun 710 22.8 4 108.0 - 1 4 1 549.0
Hornet 4 Drive 21.4 6 258.0 - 0 3 1 549.0
...
Ford Pantera L 15.8 8 351.0 - 1 5 4 93.9
Ferrari Dino 19.7 6 145.0 - 1 5 6 549.0
Maserati Bora 15.0 8 301.0 - 1 5 8 93.9
Volvo 142E 21.4 4 121.0 - 1 4 2 549.0

```

Figure 4.7: CLI - click 4

```

fauditor# select * from client;
name | year | joblist | auditor | status
-----+-----+-----+-----+-----+
clientA | 2023 | clientA_2023 | stewartli | f
(1 row)

fauditor#

```

Figure 4.8: CLI - clap 1

```

cargo run -- -n clientA -y 2023 -a stewartli init
cargo run -- -n clientA -y 2023 -a stewartli new -p /mnt/c/Users/Stewart_Li/Desktop/mpbc/

```

Figure 4.9: CLI - clap 2

5 Analysis

Factored Accounts Receivable - The biggest challenge of Factoring is to predict if and when invoices will be paid. The factor provides funds against this future payment to the business by buying their invoice. The factor then collects the payment and charges their interest rate. If the invoice isn't paid, the factor loses their advanced funds. Try using this data set for predicting when payments will be made. Get the data [here](#).

5.1 IO

```
df_raw <- read_csv(here::here('data/factor_ar.csv')) %>%
  janitor::clean_names()

glimpse(df_raw)
```

`data.table` is the fastest IO tool if your data can fit in the memory.

```
library(data.table)

# read in
data.table::fread("grep -v '770' ./data/factor_ar.csv")[, .N, by = countryCode]

# write out
df_dt <- as.data.table(df_raw)

df_dt[, 
       fwrite(data.table(.SD),
              paste0("C:/Users/Stewart Li/Desktop/res/",
                     paste0(country_code, ".csv"))), by = country_code]

# read in
data.table(
  country_code.csv = Sys.glob("C:/Users/Stewart Li/Desktop/res/*.csv")
)[, fread(country_code.csv), by = country_code.csv]
```

Get to know your data. For instance, any missing value, counting variables, and others.

```
# no NA
sapply(df_raw, function(x) {sum(is.na(x)) / nrow(df_raw)}) %>%
  enframe() %>%
  mutate(value = formattable::percent(value))

naniar::gg_miss_var(df_raw)
naniar::vis_miss(df_raw)

# no duplicate
df_raw %>% count(invoice_number, sort = TRUE)

# overview of data
skimr::skim(df_raw)
```

5.2 Cleaning

After having a basic understanding about data, do the followings to clean it up.

1. cast data types.
2. 30 days credit term is allowed. drop it subsequently (constant).
3. drop column (paperless_date).
4. rename and rearrange columns.

```
df_clean <- df_raw %>%
  mutate(across(contains("date"), lubridate::mdy),
        across(c(country_code, invoice_number), as.character)) %>%
  mutate(credit = as.numeric(due_date - invoice_date)) %>%
  select(c(country_code, customer_id, paperless_bill, disputed,
           invoice_number, invoice_amount, invoice_date, due_date, settled_date,
           settle = days_to_settle, late = days_late))

setdiff(colnames(df_raw), colnames(df_clean))
```

5.3 Validate

Validate data if it is received from other team members.

```

# data type
df_clean %>%
  select(contains("date")) %>%
  pointblank::col_is_date(columns = everything())

# cross checking
df_clean %>%
  mutate(settle1 = as.numeric(settled_date - invoice_date),
         late1 = as.numeric(settled_date - due_date),
         late1 = if_else(late1 < 0, 0, late1)) %>%
  summarise(late_sum = sum(late1) - sum(late),
            settle_sum = sum(settle1) - sum(settle))

```

5.4 Munging

Ask reasonable questions via slice dice.

```

# window operation: lag, first, nth,
df_clean %>%
  arrange(invoice_date) %>%
  group_by(country_code) %>%
  mutate(increase = invoice_amount - dplyr::lag(invoice_amount, default = 0),
         indicator = ifelse(increase > 0, 1, 0)) %>%
  ungroup() %>%
  mutate(settle_grp = (settle %% 10) * 10)

df_clean %>%
  group_by(country_code) %>%
  arrange(invoice_date) %>%
  summarise(n = n(),
            sales = sum(invoice_amount),
            first_disputed_late = first(late[disputed == 'Yes']),
            first_disputed_inv_date = first(invoice_date[disputed == 'Yes']),
            largest_late = max(late[disputed == 'Yes']),
            largest_inv_amt = invoice_amount[late == max(late)],
            .groups = 'drop')

```

Cut late into four categories based on the firm's credit policy.

```

sort(unique(df_clean$late))

df_late <- df_clean %>%
  dplyr::filter(late != 0) %>%
  mutate(reminder = case_when(late > 0 & late <= 10 ~ "1st email",
                               late > 10 & late <= 20 ~ "2nd email",
                               late > 20 & late <= 30 ~ "legal case",
                               TRUE ~ "bad debt"))

# anomaly by country
df_late %>%
  ggplot(aes(late, disputed, color = country_code)) +
  geom_boxplot() +
  theme_light()

# summary table
df_late %>%
  group_by(reminder, disputed) %>%
  summarise(across(late, tibble::lst(sum, min, max, sd)),
            .groups = 'drop') %>%
  gt::gt()

# clients without dispute do not pay.
df_late %>%
  dplyr::filter(disputed == 'No', reminder %in% c('legal case', 'bad debt'))

```

5.5 EDA

Focus on a handful of variables after dropped others.

```

df <- df_clean %>%
  select(-c(contains('date'), invoice_number))

# freq table
with(df, table(disputed, country_code) %>% addmargins())
tapply(df$invoice_amount, list(df$disputed, df$country_code), median)

# descriptive stats
df %>%

```

Figure 5.1: Data munging

```
select(where(is.numeric)) %>%
summary()

# normal distribution
df %>%
  ggplot(aes(invoice_amount, fill = disputed)) +
  geom_histogram(bins = 10, position = 'dodge') +
  geom_vline(xintercept = median(df$invoice_amount), color = 'red',
             size = 3, linetype = "dashed") +
  theme_light()

# correlation
df %>%
  select(where(is.numeric)) %>%
  cor() %>%
  corrplot::corrplot(method = 'color', order = 'FPC', type = 'lower', diag = FALSE)

df %>%
  select(where(is.numeric)) %>%
```

```
corrr::correlate() %>%
corrr::rearrange() %>%
corrr::shave() %>%
corrr::fashion()
```

5.6 Model

Read more about logistic regression [here](#), [here](#), and [here](#).

```
# easy stats plot
df %>%
  mutate(prob = ifelse(disputed == "Yes", 1, 0)) %>%
  ggplot(aes(late, prob)) +
  geom_point(alpha = .2) +
  geom_smooth(method = "glm", method.args = list(family = "binomial")) +
  theme_light()

# model comparison
df_mod <- df %>%
  mutate(disputed = as.factor(disputed))

mod1 <- glm(disputed ~ late, family = "binomial", data = df_mod)
mod2 <- glm(disputed ~ late + settle + invoice_amount,
             family = "binomial", data = df_mod)

summary(mod1)
anova(mod1, mod2, test = "Chisq")

# model diagnostic
df_mod_res <- broom::augment(mod1, df_mod) %>%
  mutate(pred = ifelse(.fitted > .5, "Yes", "No") %>% as.factor())

# confusion matrix
df_mod_res %>%
  yardstick::conf_mat(disputed, pred) %>%
  autoplot()

# plot pred
df_mod_res %>%
```

```

mutate(res = disputed == pred) %>%
ggplot(aes(invoice_amount, settle, color = res)) +
geom_point() +
theme_light()

df_mod_res %>%
ggplot(aes(invoice_amount, settle, color = disputed)) +
geom_point() +
facet_wrap(~pred) +
theme_light()

```

5.7 Report

```

library(patchwork)
library(ggtext)
library(showtext)

p1 <- df %>%
ggplot(aes(invoice_amount, settle, color = disputed)) +
geom_point() +
scale_color_manual(labels = c("Agreed", 'Disputed'),
values = c("#9AC2BB", '#E99184')) +
guides(color = guide_legend(title.position = "top", title = ""))
labs(x = "", y = "Settlement days") +
theme_light() +
theme(
  legend.position = c(.95, .98),
  legend.background = element_rect(color = "transparent", fill = 'transparent'),
  legend.box.background = element_rect(color = "transparent", fill = "transparent"),
  legend.key = element_rect(colour = "transparent", fill = "transparent")
)

p2 <- df %>%
group_by(if_late = late == 0) %>%
ggplot(aes(invoice_amount, settle, color = disputed)) +
geom_point(show.legend = FALSE) +
scale_color_manual(labels = c("Agreed", 'Disputed'),
values = c("#9AC2BB", '#E99184')) +
facet_wrap(~if_late) +

```

```

  labs(caption = "@RAudit Solution | **Stewart Li**<br>(Data source: Kaggle)",
       x = "Invoice amount",
       y = "Settlement days") +
  theme_light() +
  theme(
    axis.title.y = element_text(margin = margin(b = 1, unit = "in")),
    strip.text = element_text(color = '#2D4248'),
    strip.background = element_blank(),
    plot.caption = element_markdown(lineheight = 1.2)
  )
)

p1 / p2 +
  plot_annotation(
    title = "The <span style = 'color:#E99184;'>Analysis</span> of cash collection",
    subtitle = 'Focus on those slow settlement without dispute',
    tag_levels = 'A'
  ) &
  theme(plot.tag = element_text(size = 8),
        plot.title = element_markdown())

```

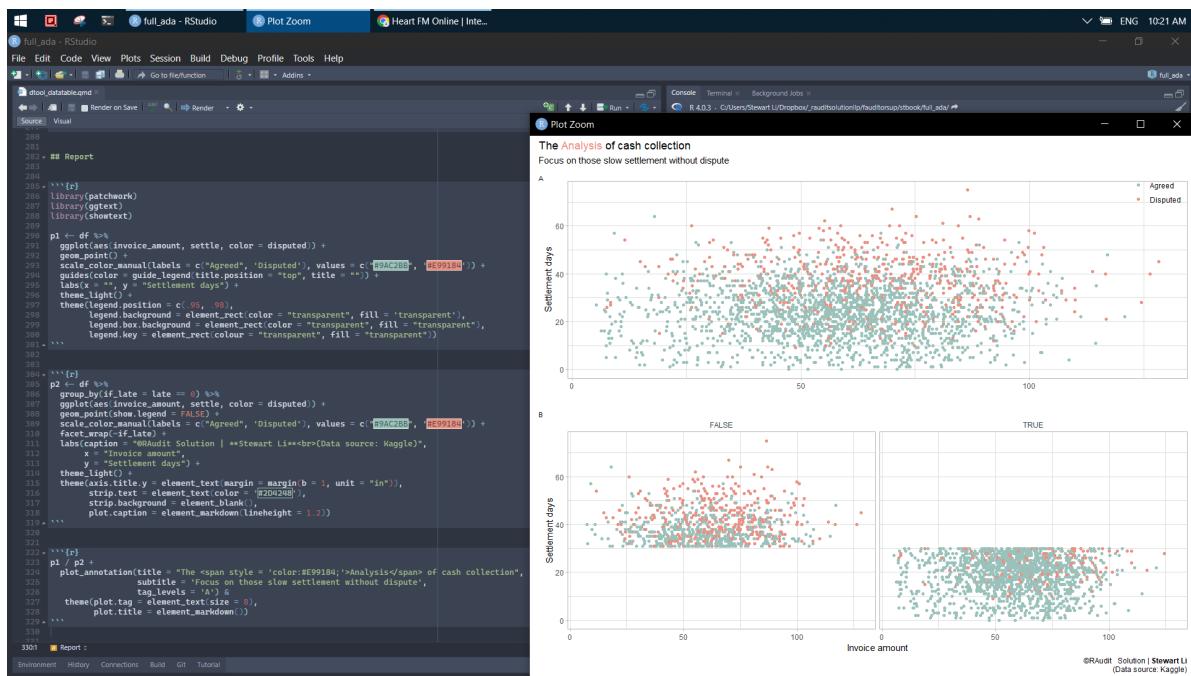


Figure 5.2: Combined plot

6 Example 2

[To my understanding] Audit includes **tools and work** stipulated by Standards. Audit Data Analytics (ADA) replaces excel-related tools with R/Python to improve efficiency/effectiveness. It does not necessarily reduce audit work required by ISCA. The following example is to audit expense claim based on data from payroll, hr, and finance departments, which demonstrates ADA is a vital move for auditors from all possible perspectives.

Compared to excel-related tools, it could be easily used to test audit assertions (e.g., occurrence, existence, completeness, cut-off, valuation, classification) after reconciled in terms of P2P, O2C, Payroll, R2R, GL.

1. benefit: version control `diff`, lightweight `size`, powerful `1m` rows, automation `script`.
2. pattern recognition: spot deviation and inconsistency.

It also addresses common mistakes throughout the audit process. For instance,

1. version control: which version of PBC data is the latest?
2. reproducible: my result is different from yours after rerun.
3. report: check if number in working papers tally to those in financial statement.
4. automation: roll out audit work next year by copy+paste.

6.1 Cleaning

```
exp_claim_raw <- readxl::read_excel("isca_cpe_2023/1. Anomalies in Payroll data.xlsx",
                                      sheet = 1,
                                      range = "A1:G33") %>%
janitor::clean_names()

hr_data_raw <- readxl::read_excel("isca_cpe_2023/1. Anomalies in Payroll data.xlsx",
                                    sheet = 2) %>%
janitor::clean_names()

pay_data_raw <- readxl::read_excel("isca_cpe_2023/1. Anomalies in Payroll data.xlsx",
                                    sheet = 3,
                                    skip = 2, range = "A3:D25") %>%
janitor::clean_names()
```

```

stli@true:~/.../testf> bat a.R
File: a.R
1 cat(cli::bg_red("Hello world - audit data analytics in R\n"))
2
3 library(data.table)
4
5 # Part 1 - IO
6 df <- fread(cmd = "grep -v Merc 'C:/Users/Stewart Li/Desktop/tbd/a.csv'", select = 2:13, colClasses = list(character = c("car"), numeric = 3:13))
7
8 dim(df)
9 glimpse(df)
10 df[sample(1:nrow(df), 3, replace = TRUE)]
11
12 # Part 2 - Count
13 df[, .N., .by = .(am, gear)]
14 df[, sum(mpg > mean(mpg)), .by = .(am, gear)]
15
16 # Part 3 - Summarize
17 df[, .SD(), by = .(am, gear)]
18 df[, lapply(.SD, max), .by = .(am, gear)]
19
20 with(df, tapply(mpg, list(am, gear), max, default = 0))
21 with(df, by(mpg, c(gear), summary))
22 aggregate(mpg ~ am * gear, data = df, FUN = median, subset = df$hp > 150)
23
24 # Part 4 - Reshape
25 melt(df, id = c("am", "gear"), measure = c("mpg", "hp"))
26 dcast(df, am ~ gear ~ carb, value.var = "mpg", fun.aggregate = list(min, mean, max), fill = 0)
27
28 # Part 5 - Filter
29 df[between(mpg, 25, 30)]
30 df[mpg > 20 & hp < 100, ]
31
32 # Part 6 - Mutate
33 df[, created_date := seq(as.Date("2021-12-1"), by = "month", length.out = nrow(df))]
34 df[, :=(new_mpg = (mpg %/% 2) * 2, new_mpg_hp = mpg/hp)]
35 df[, c("var1", "var2") := tstrsplit(displ, ".", fixed = TRUE, fill = 0)][]
36 df[, (mpg, new_mpg_if) = fcase(mpg < 15, "small", mpg > 30, "large", default = "medium")]
37
38 # Part 7 - Reshape
39 melt(df, id = c("am", "gear"), measure = c("mpg", "hp"))
40 dcast(df, am ~ gear ~ carb, value.var = "mpg", fun.aggregate = list(min, mean, max), fill = 0)
41
42 # Part 8 - Filter
43 df[between(mpg, 25, 30)]
44 df[mpg > 20 & hp < 100, ]
45
46 # Part 9 - Mutate
47 df[, created_date := seq(as.Date("2021-12-1"), by = "month", length.out = nrow(df))]

stli@true:~/.../testf> bat a1.R
File: a1.R
1 cat(cli::bg_red("Hello world - audit data analytics in Python\n"))
2
3 library(data.table)
4
5 # Part 1 - IO
6 df <- fread(cmd = "grep -v Merc 'C:/Users/Stewart Li/Desktop/tbd/a.csv'", select = 2:13, colClasses = list(character = c("car"), numeric = 3:13))
7
8 dim(df)
9 glimpse(df)
10 df[sample(1:nrow(df), 3, replace = TRUE)]
11
12 # Part 2 - Count
13 df[, .N., .by = .(am, gear)]
14 df[, sum(mpg > mean(mpg)), .by = .(am, gear)]
15
16 # Part 3 - Summarize
17 df[, .SD(), by = .(am, gear)]
18 df[, lapply(.SD, max), .by = .(am, gear)]
19
20 with(df, tapply(mpg, list(am, gear), max, default = 0))
21 with(df, by(mpg, c(gear), summary))
22 aggregate(mpg ~ am * gear, data = df, FUN = median, subset = df$hp > 150)
23
24 # Part 4 - Reshape
25 melt(df, id = c("am", "gear"), measure = c("mpg", "hp"))
26 dcast(df, am ~ gear ~ carb, value.var = "mpg", fun.aggregate = list(min, mean, max), fill = 0)
27
28 # Part 5 - Filter
29 df[between(mpg, 25, 30)]
30 df[mpg > 20 & hp < 100, ]
31
32 # Part 6 - Mutate
33 df[, created_date := seq(as.Date("2021-12-1"), by = "month", length.out = nrow(df)))
34 df[, :=(new_mpg = (mpg %/% 2) * 2, new_mpg_hp = mpg/hp)]
35 df[, c("var1", "var2") := tstrsplit(displ, '.', fixed = TRUE, fill = 0)][]
36 df[, (mpg, new_mpg_if) = fcase(mpg < 15, "small", mpg > 30, "large", default = "medium")]

stli@true:~/.../testf>

```

Thu 2024-01-25 15:10

Figure 6.1: Diff 1

```

stli@true:~/.../testf> bat a.R
File: a.R
1 cat(cli::bg_red("Hello world - audit data analytics in R\n"))
2
3 library(data.table)
4
5 # Part 1 - IO
6 df <- fread(cmd = "grep -v Merc 'C:/Users/Stewart Li/Desktop/tbd/a.csv'", select = 2:13, colClasses = list(character = c("car"), numeric = 3:13))
7
8 dim(df)
9 glimpse(df)
10 df[sample(1:nrow(df), 3, replace = TRUE)]■ Trailing whitespace is superfluous.
11
12 # Part 2 - Count
13 df[, .N., .by = .(am, gear)]
14 df[, sum(mpg > mean(mpg)), .by = .(am, gear)]
15
16 # Part 3 - Summarize
17 df[, .SD(), by = .(am, gear)]
18 df[, lapply(.SD, max), .by = .(am, gear)]
19
20 with(df, tapply(mpg, list(am, gear), max, default = 0))
21 with(df, by(mpg, c(gear), summary))
22 aggregate(mpg ~ am * gear, data = df, FUN = median, subset = df$hp > 150)
23
24 # Part 4 - Reshape
25 melt(df, id = c("am", "gear"), measure = c("mpg", "hp"))
26 dcast(df, am ~ gear ~ carb, value.var = "mpg", fun.aggregate = list(min, mean, max), fill = 0)■ Lines should not be here
27
28 # Part 5 - Filter
29 df[between(mpg, 25, 30)]
30 df[mpg > 20 & hp < 100, ]
31
32 # Part 6 - Mutate
33 df[, created_date := seq(as.Date("2021-12-1"), by = "month", length.out = nrow(df))]■ Lines should not be here
34 df[, :=(new_mpg = (mpg %/% 2) * 2, new_mpg_hp = mpg/hp)]■ Put spaces around all infix operators.
35 df[, c("var1", "var2") := tstrsplit(displ, '.', fixed = TRUE, fill = 0)][]■ Trailing whitespace is superfluous.
36 df[, (mpg, new_mpg_if) = fcase(mpg < 15, "small", mpg > 30, "large", default = "medium")]■ Trailing whitespace is superfluous.
37
38 # Part 7 - Reshape
39 melt(df, id = c("am", "gear"), measure = c("mpg", "hp"))

stli@true:~/.../testf> bat a1.R
File: a1.R
1 cat(cli::bg_red("Hello world - audit data analytics in Python\n"))
2
3 library(data.table)
4
5 # Part 1 - IO
6 df <- fread(cmd = "grep -v Merc 'C:/Users/Stewart Li/Desktop/tbd/a.csv'", select = 2:13, colClasses = list(character = c("car"), numeric = 3:13))
7
8 dim(df)
9 glimpse(df)
10 df[sample(1:nrow(df), 3, replace = TRUE)]■ Trailing whitespace is superfluous.
11
12 # Part 2 - Count
13 df[, .N., .by = .(am, gear)]
14 df[, sum(mpg > mean(mpg)), .by = .(am, gear)]
15
16 # Part 3 - Summarize
17 df[, .SD(), by = .(am, gear)]
18 df[, lapply(.SD, max), .by = .(am, gear)]
19
20 with(df, tapply(mpg, list(am, gear), max, default = 0))
21 with(df, by(mpg, c(gear), summary))
22 aggregate(mpg ~ am * gear, data = df, FUN = median, subset = df$hp > 150)
23
24 # Part 4 - Reshape
25 melt(df, id = c("am", "gear"), measure = c("mpg", "hp"))
26 dcast(df, am ~ gear ~ carb, value.var = "mpg", fun.aggregate = list(min, mean, max), fill = 0)■ Lines should not be here
27
28 # Part 5 - Filter
29 df[between(mpg, 25, 30)]
30 df[mpg > 20 & hp < 100, ]
31
32 # Part 6 - Mutate
33 df[, created_date := seq(as.Date("2021-12-1"), by = "month", length.out = nrow(df))]■ Lines should not be here
34 df[, :=(new_mpg = (mpg %/% 2) * 2, new_mpg_hp = mpg/hp)]■ Put spaces around all infix operators.
35 df[, c("var1", "var2") := tstrsplit(displ, '.', fixed = TRUE, fill = 0)][]■ Trailing whitespace is superfluous.
36 df[, (mpg, new_mpg_if) = fcase(mpg < 15, "small", mpg > 30, "large", default = "medium")]■ Trailing whitespace is superfluous.
37
38 # Part 7 - Reshape
39 melt(df, id = c("am", "gear"), measure = c("mpg", "hp"))

stli@true:~/.../testf>

```

Thu 2024-01-25 15:20

Figure 6.2: Diff 2

```

df_comb <- exp_claim_raw %>%
  full_join(hr_data_raw, by = c('staff_id' = 'staff_id')) %>%
  left_join(pay_data_raw, by = c('staff_id' = 'staff_id'))

df_clean <- df_comb %>%
  mutate(across(contains("date"), lubridate::dmy)) %>%
  mutate(on_leave = lubridate::dmy(on_leave)) %>%
  mutate(staff_name = coalesce(staff_name, name.x))

# check if amount is correct
sum(df_clean$amount_s.x, na.rm = TRUE)

df_clean %>%
  distinct(staff_id, amount_s.y) %>%
  summarise(app_c = sum(amount_s.y, na.rm = TRUE))

sheets <- list("comb" = df_comb, "clean" = df_clean)
writexl::write_xlsx(sheets, here::here(paste0('audit_sit/audit_payroll', Sys.Date(), '.xlsx'))
openxlsx::openXL(here::here("audit_sit/audit_payroll2023-12-22.xlsx"))

df_clean <- readxl::read_excel(here::here("audit_sit/audit_payroll2023-12-22.xlsx")) %>%
  mutate(across(c(contains("date"), on_leave), lubridate::dmy))

```

6.2 Procedure

```

# cross check payroll amount against finance amount
df_clean %>%
  group_by(staff_id, staff_name) %>%
  summarise(amt_exp = sum(amount_s.x),
            amt_paid = sum(amount_s.y) / n(),
            amt_diff = amt_exp - amt_paid,
            .groups = 'drop')

# compare date to ensure no claim happens before incurred or after resigned
df_clean %>%
  dplyr::filter(claim_date > expense_date)

```

```

df_clean %>%
  dplyr::filter(claim_date > last_date | claim_date == on_leave)

# identify multiple claims for the same expense
df_clean %>%
  group_by(staff_id, staff_name, purpose, amount_s.x) %>%
  dplyr::filter(n() > 1)

# ensure staff name and their bank account number updated timely
df_clean %>%
  dplyr::filter(!is.na(edits_to_hr_data),
               bank_account_no.x == bank_account_no.y)

df_clean %>%
  dplyr::filter(name.x != name.y)

# produces audit working paper
library(pointblank)

ag <- df_clean %>%
  create_agent(label = "A very *simple* example.", tbl_name = "payroll") %>%
  col_vals_between(columns = claim_date, left = vars(expense_date), right = vars(last_date))
interrogate()

ag

```

6.3 Enhanced

```

df_clean %>%
  count(staff_name, sort = TRUE)

df_clean %>%
  dplyr::filter(grepl("\\d+", purpose)) %>%
  mutate(purpose = gsub("\\d+", "", purpose)) %>%
  mutate(across(where(is.character), ~na_if(., "AB99"))) %>%
  mutate(staff_id = replace_na(staff_id, 0))

```

Figure 6.3: Audit Procedure 1

Figure 6.4: Audit Procedure 2

```

df_clean %>%
  select(contains("date"), purpose) %>%
  mutate(if_taxi = case_when(str_detect(purpose, "Taxi") ~ "taxi",
                             TRUE ~ "other"),
         total_date = lubridate::floor_date(claim_date, "week"),
         first_date = first(total_date)) %>%
  slice_max(order_by = claim_date, n = 3)

df_clean %>%
  dplyr::filter(!is.na(amount_s.x)) %>%
  mutate(new = (amount_s.x %% 100) * 100) %>%
  group_by(new, amount_s.x > 300) %>%
  summarise(new1 = mean(amount_s.x), .groups = 'drop')

df_clean %>%
  dplyr::filter(!is.na(staff_name)) %>%
  group_nest(staff_id, staff_name) %>%
  mutate(new = map(data, ~pluck(.x, 4))) %>%
  mutate(new1 = map(new, ~paste(.x, collapse = '|'))) %>%
  select(-data, -new) %>%
  unnest(new1)

df_clean %>%
  dplyr::filter(!is.na(staff_name)) %>%
  select(staff_id, staff_name, purpose) %>%
  summarise(new1 = paste(purpose, collapse = '|'), .by = c(staff_id, staff_name))

df_clean %>%
  select(staff_id, staff_name, division, purpose, amount_s.x) %>%
  dplyr::filter(!is.na(purpose)) %>%
  separate(purpose, into = c("type", "info"),
            extra = 'merge', remove = FALSE, fill = 'right') %>%
  group_by(division, type) %>%
  summarise(n = n(),
            amt_type = sum(amount_s.x), .groups = 'drop') %>%
  arrange(-amt_type)

library(lubridate)

df_clean %>%

```

```
pivot_longer(cols = where(is.Date),
             names_to = 'activity_date',
             values_to = 'detail_date',
             names_pattern = "(.*).",
             names_transform = list(activity_date = toupper)))
```

References

- Li, Stewart, Richard Fisher, and Michael Falta. 2020. “The Effectiveness of Artificial Neural Networks Applied to Analytical Procedures Using High Level Data: A Simulation Analysis.” *Meditari Accountancy Research* 29 (6): 1425–50. <https://doi.org/10.1108/medar-06-2020-0920>.
- Wickham, Hadley. 2016. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. <https://ggplot2.tidyverse.org>.