

# Data Analytics Engineering

For Accountants and Auditors

Stewart Li

2024-03-08

# Table of contents

<b>Preface</b>	<b>3</b>
<b>I Infrastructure</b>	<b>4</b>
<b>1 Local</b>	<b>6</b>
<b>2 ELT</b>	<b>20</b>
<b>3 HTTP</b>	<b>22</b>
<b>4 FAudit</b>	<b>26</b>
<b>II Data tools</b>	<b>31</b>
<b>5 Polars</b>	<b>35</b>
<b>6 Analysis</b>	<b>40</b>
6.1 IO . . . . .	40
6.2 Cleaning . . . . .	41
6.3 Validate . . . . .	41
6.4 Munging . . . . .	42
6.5 EDA . . . . .	43
6.6 Model . . . . .	45
6.7 Report . . . . .	46
<b>7 Audit</b>	<b>48</b>
7.1 Cleaning . . . . .	48
7.2 Procedure . . . . .	50
7.3 Enhanced . . . . .	51
<b>References</b>	<b>55</b>

# 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.

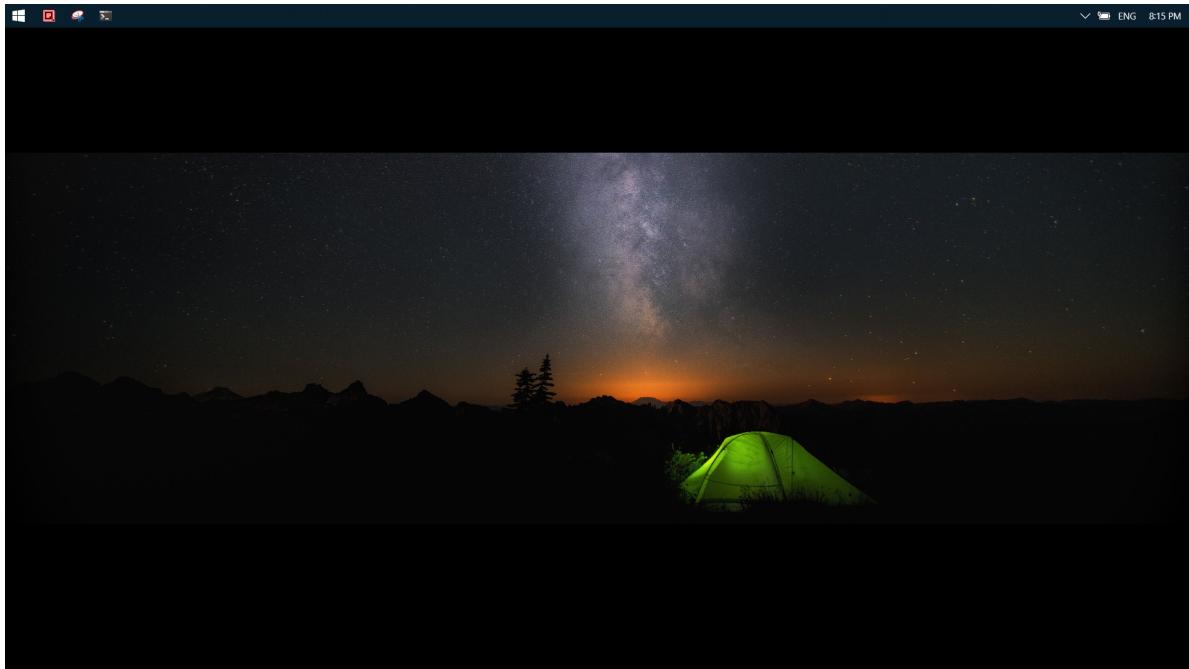


Figure 1.1: Desktop

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

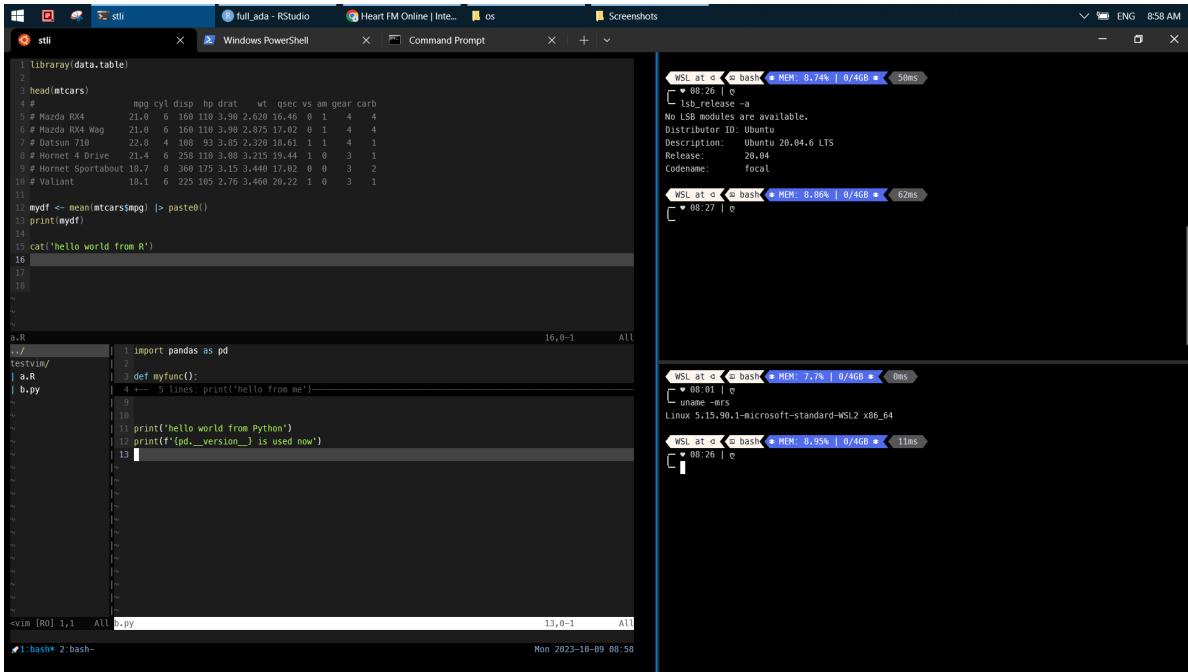


Figure 1.4: Ubuntu

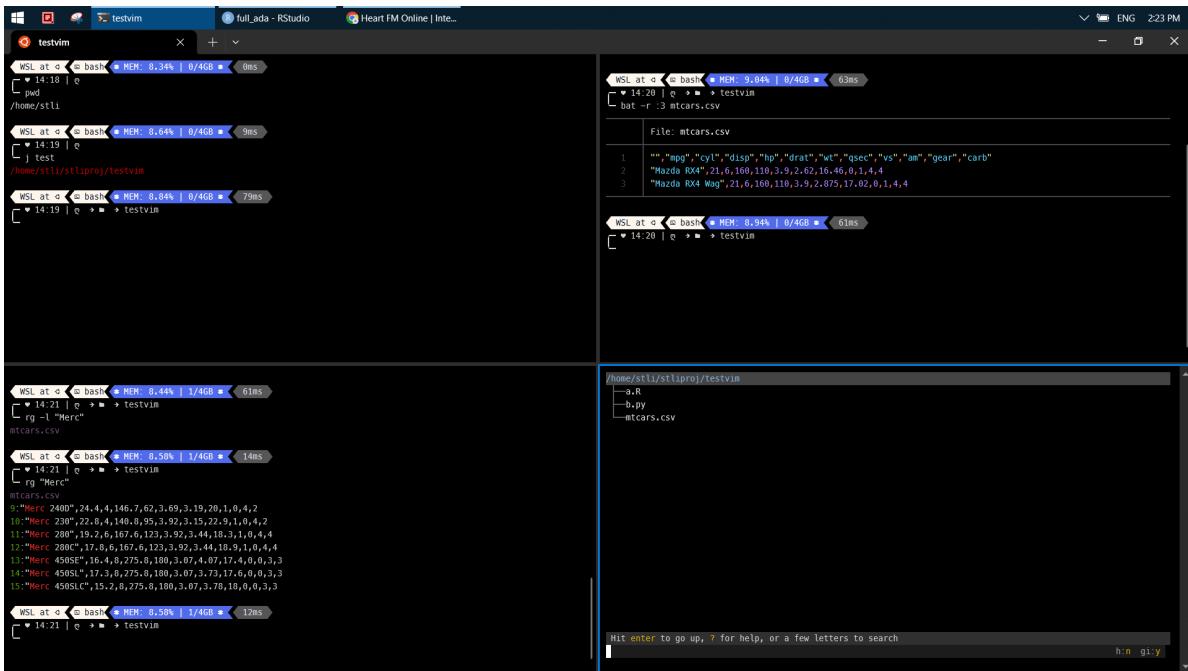


Figure 1.5: Terminal tools

WSL at < bash \* MEM: 9.5% | 1/4GB \* 8ms  
~ 15:25 | q ➔ testvnm

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,  
'help.start()' for an HTML browser interface to help.  
Type 'q()' to quit R.

```
> write.csv(mtcars, "mtcars.csv")
>
```

WSL at < bash \* MEM: 10.6% | 1/4GB \* 9ms  
~ 15:29 | q ➔ testvnm

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	hp	...	vs	am	gear	carb
count	32.000000	32.000000	32.000000	32.000000	...	32.000000	32.000000	32.000000	32.000000
mean	20.898625	6.187500	236.721875	146.597500	...	0.477500	0.406250	3.737500	2.8125
std	6.076942	1.785922	123.938694	66.592688	...	0.504016	0.499391	0.737804	1.6357
min	10.400000	4.000000	71.000000	57.000000	...	0.000000	0.000000	3.000000	1.0000
25%	15.425000	4.000000	120.825000	96.500000	...	0.000000	0.000000	3.000000	2.0000
50%	19.200000	6.000000	196.300000	123.000000	...	0.000000	0.000000	4.000000	2.0000
75%	22.800000	6.000000	326.000000	186.000000	...	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]

WSL at < bash \* MEM: 9.72% | 1/4GB \* 9ms  
~ 15:25 | q ➔ testvnm

ls

a.R b.py mtcars.csv

WSL at < bash \* MEM: 10.52% | 1/4GB \* 11ms  
~ 15:27 | q ➔ testvnm

WSL at < bash \* MEM: 12.3% | 1/4GB \* 9ms  
~ 15:32 | q ➔

/duckdb

-- Loading resources from /home/stli/.duckdbrc

v0.8.1 6536a77232

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("stripes/testvnm/mtcars.csv") LIMIT 3;
```

column0	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.9	2.62	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.9	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.32	18.61	1	1	4	1

Figure 1.6: R, Python, DuckDB

The screenshot shows a Windows desktop environment with several open windows. In the center is a RStudio interface displaying an R script named 'testvwm.R'. The script includes code to load the 'mtcars' dataset, calculate mean mpg, print the results, and then display the 'head' of the 'mtcars' dataset. To the right of the RStudio window is a terminal window titled 'full\_ada - RStudio' showing the output of the R script. The terminal output includes the R version information, copyright notice, platform details, and various help and documentation links. It also shows the 'head' command output for the 'mtcars' dataset, which contains columns: mpg, cyl, disp, hp, drat, wt, qsec, vs, am, gear, carb. The data includes rows for different car models like Mazda RX4, Mazda RX4 Wag, Datsun 710, Hornet 4 Drive, Hornet Sportabout, and Valiant.

```
library(data.table)
# Load the mtcars dataset
# Calculate mean mpg
mydf <- mean(mtcars$mpg) > paste0()
print(mydf)
cat('hello world from R')
>
> head(mtcars)
mpg cyl disp hp drat wt qsec vs am gear carb
Mazda RX4 21.0 6 160 110 3.90 2.620 16.46 0 1 4 4
Mazda RX4 Wag 21.0 6 160 110 3.90 2.675 17.02 0 1 4 4
Datsun 710 22.8 4 108 93 3.85 2.320 18.61 1 1 4 1
Hornet 4 Drive 21.4 6 258 110 3.08 3.215 19.44 1 0 3 1
Hornet Sportabout 18.7 8 360 175 3.15 3.440 17.02 0 0 3 2
Valiant 18.1 6 225 105 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 vertical resize +2 13,0-1 All python3 "b.py" [finished] 2,17 All

Figure 1.8: Vim - Python

The screenshot shows a Windows desktop with several open windows. The main focus is a terminal window titled 'stli' which contains a file browser interface. The left pane shows a tree view of a project structure:

- Neo-tree
- ~/stproj/testrust
- src
- main.rs
- Cargo.lock
- Cargo.toml

The right pane shows the contents of 'Cargo.toml':

```
[package]
name = "testrust"
version = "0.1.0"
edition = "2021"

# See more keys and their definitions at https://doc.rust-lang.org/cargo/reference/manifest.html

[dependencies]
anyhow = "1.0.75" 1.0.75
```

Below the file browser is another terminal window titled 'stli' showing the command:

```
git < ② < ⑨ 88% 8:6 ⑩ 19:36 NORMAL > master > Cargo.toml
```

To the right of the file browser is a 'Neotree' window showing a file tree:

- ~/stproj/testvim
- \_\_pycache\_\_
- a.R
- a.py
- b.txt
- b1.txt
- c.py
- d.py

Below the Neotree window is another terminal window titled 'stli' showing the command:

```
t ⑧ library(data.table)
library(fs)
data.table::fread()
cat("Hello world")
head(mtcars)
#          mpg cyl disp  wt  qsec vs am gear car
# Mazda RX4   21.0   6 108 3.98 2.628 16.46 0  1  4
# Mazda RX4 Wag 21.0   6 108 3.98 2.875 17.02 0  1  4
# Datsun 710  22.8   4 108 3.85 2.320 18.61 0  1  4
# Hornet 4 Drive 21.4   6 123 3.08 3.215 19.44 1  0  3
# Hornet Sportabout 18.7   8 160 3.44 3.615 17.02 0  0  3
# Valiant 18.1   6 225 105 2.76 3.460 20.22 1  0  3
```

At the bottom of the screen is a status bar:

```
:TODO do it
17 | $:dir_tree(`..`)
```

At the very bottom of the screen is a footer:

```
NORMAL > master > Cargo.toml
gk < ② < ⑨ 88% 8:6 ⑩ 19:36 NORMAL > master > c.py
1| bash* 2: bash-
6 < ② < ⑦ ⑧ Ret 7:1 ⑩ 19:36
Set 2023-11-18 19:36
```

Figure 1.9: Tmux - Nvim 1

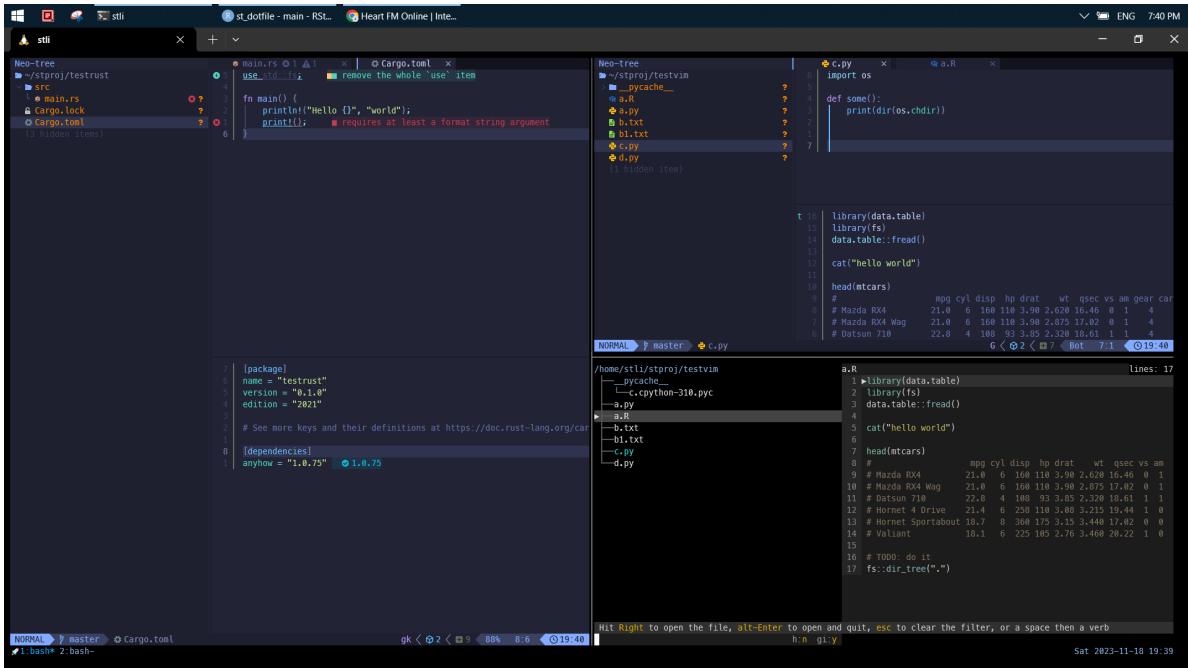


Figure 1.10: Tmux -Nvim 2

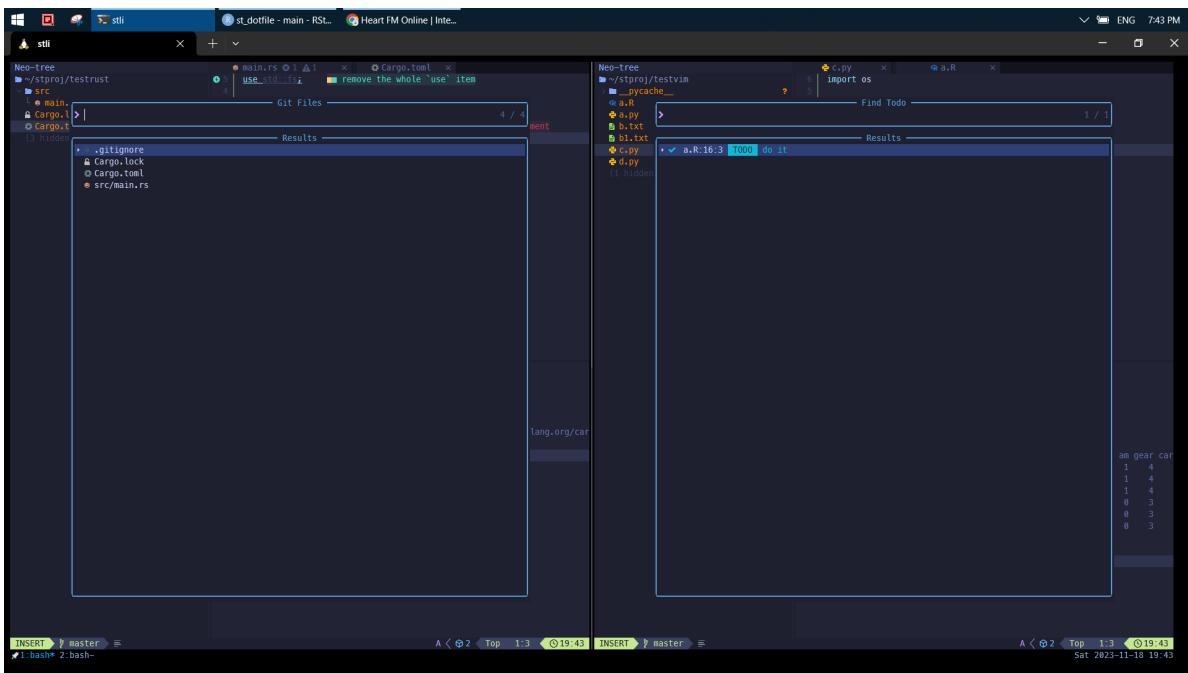


Figure 1.11: Tmux - Nvim 3

This screenshot shows a Tmux session with four panes. The top-left pane displays a file tree and a search results window for 'Find Todo'. The top-right pane shows R code and its output, including the `mtcars` dataset and a 'Hello world' message. The bottom-left pane shows a terminal window with the command 'ls' and the output '1.bash 2.bash-'. The bottom-right pane shows system status information.

```

library(data.table)
library(tidyverse)
data.table::fread()

cat("Hello world")

head(mtcars)

# Mazda RX4      21.0   6 160 108 3.98 2.875 16.46 0 1 4 4
# Mazda RX4 Wag 21.0   6 160 108 3.98 2.075 17.82 0 1 4 4
# Datsun 710     22.8   4 108 93 3.85 2.328 18.61 1 1 4 1
# Hornet 4 Drive 21.4   6 258 110 3.08 3.215 19.44 1 0 3 1
# Hornet Sportabout 18.7 8 360 175 3.15 3.440 17.02 0 0 3 2
# Valiant        18.1   6 225 105 3.08 2.76 3.460 20.22 1 0 3 1

# 100% done
fs::dir_tree(".")

Source : Visual
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20 |
```

Figure 1.12: Tmux -Nvim 4

This screenshot shows an RStudio interface. The left pane contains an R script with code for reading the 'mtcars' dataset into Python using the 'reticulate' package. The right pane shows a data preview of the 'mtcars' dataset, including columns like mpg, cyl, disp, am, gear, and carb. The bottom pane shows the RStudio environment and file menu.

```

library(reticulate)
df_r1 <- mtcars
pd = import('pandas')
df_py1 = r_to_py(df_r1)
df_py1.dtypes
df_py1.describe()

```

	mpg	cyl	disp	am	gear	carb
count	32.000000	32.000000	32.000000	32.000000	32.000000	32.000000
min	20.09025	6.187500	236.72175	0	0.46250	2.8125
q1	22.10000	8.000000	160.00000	0	1.00000	3.00000
median	22.80000	8.000000	167.60000	0	1.00000	3.00000
q3	24.30000	8.000000	167.00000	1	1.00000	4.00000
max	33.90000	8.000000	472.00000	1	5.00000	8.00000

Figure 1.13: RStudio - R

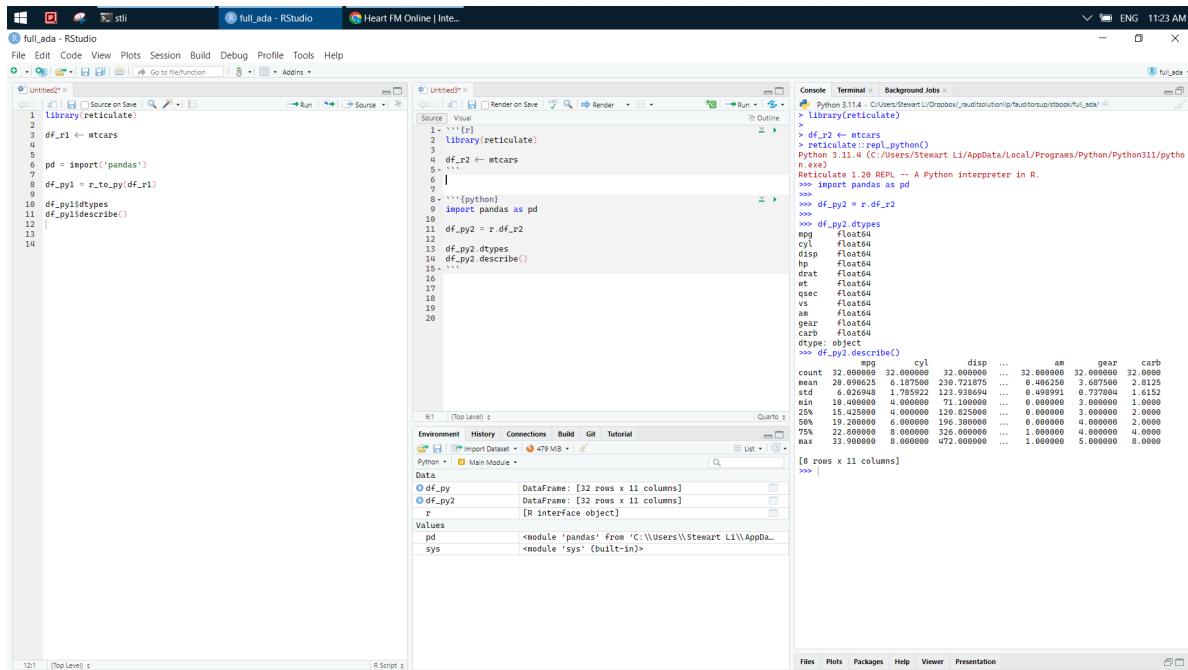


Figure 1.14: RStudio - Python

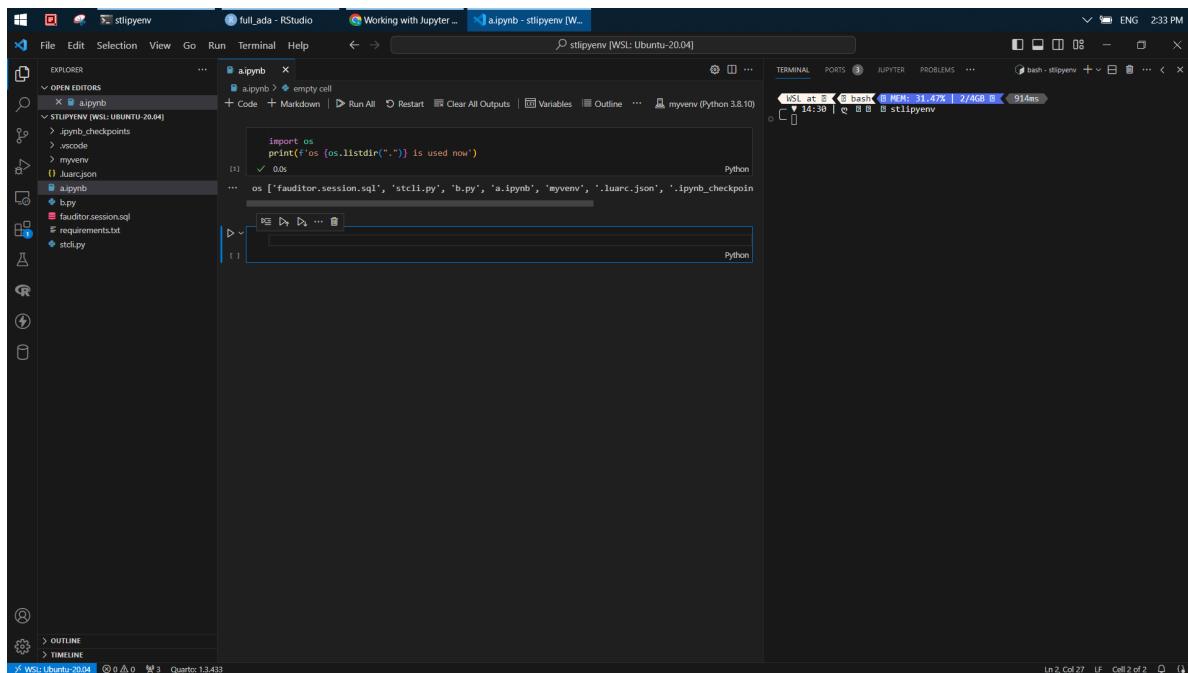


Figure 1.15: VS Code in Linux - Jupyter

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' showing the output of the script execution:

```
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 main area has an 'EXPLORER' sidebar on the left with files 'a.R' and 'b.py' listed. The central editor window displays the following Python 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 of the editor is an 'Interactive' cell window titled 'Interactive - b.py'. It contains the command 'import pandas as pd ..' and the output 'pandas 2.0.3 is used now'. At the bottom of the screen, there is a status bar with the text 'Type "python" code here and press Shift+Enter to run'.

Figure 1.18: VS Code in Windows - Interactive cell

A screenshot of the Visual Studio Code interface on Windows, similar to Figure 1.18 but with an R session. The title bar shows tabs for 'full\_ada - RStudio', 'Heart FM Online | Inte...', 'a.R - testvim - Visual...', and 'R Graphics: Device 2 (ACTIVE)'. The 'TERMINAL' tab is active, displaying R code and its output:

```
> head(mtcars)
#> #>   mpg cyl disp hp drat wt qsec vs am gear carb
#> #>   Mazda RX4    21.0   6 160 110 3.98 2.875 17.02 0 1 4 4
#> #>   Mazda RX4 Wag 21.0   6 160 110 3.98 2.875 17.02 0 1 4 4
#> #>   Datsun 710   22.8   4 108 223 4.32 1.90 16.46 1 0 3 1
#> #>   Hornet 4 Drive 21.4   4 123 108 3.08 1.83 15.44 1 0 3 1
#> #>   Hornet Sportabout 18.7   8 258 110 3.08 1.41 22.80 0 0 3 2
#> #>   Valiant   18.1   6 225 105 2.76 3.460 20.22 1 0 3 1
#> > plot(mtcars)
```

The 'OUTPUT' tab shows a correlation matrix plot for the mtcars dataset, featuring a grid of scatter plots where each cell shows the correlation between two variables. The variables are listed along the axes: 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/StewartLi/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\StewartLi\Desktop" showing the directory structure. It contains a ".desktop.ini" file (376 B, modified Mar 8, 2021, 6:49 PM), a "QDir.exe" file (6505 kB, modified Mar 18, 2016, 12:17 PM), and a "td" folder.

Figure 1.20: CLI R - GitBash 1

The screenshot shows a Windows desktop with several open windows. In the foreground, there's an RStudio interface with a code editor containing R script code. The code is related to setting up a folder structure for a project, involving functions like `create_folder`, `write_config`, and `scaffold`. The RStudio interface includes tabs for Environment, History, Connections, Build, Git, and Tutorial. Below the RStudio window is a taskbar with icons for File Explorer, Task View, and other system tools. In the background, there's a terminal window titled "MINGW64/c/Users/Stewart Li/Desktop/raudit solution/lp/fauditorsup/stbook/full\_ada" showing command-line usage for a script named `setup_term.R`. The terminal also displays the output of running the script with various flags.

```
#!/usr/bin/env R --R-4.0.3/bin/Rscript.exe
# Manual
# Run this from GitHub
# 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.
# 3. If folders do exist, write config.yml
# 4. If folders do exist, list jobs in prod and create rproj
# Load
# Helper
# Helper
# scaffold = function(f = "stlproj", seedir = TRUE, dev = "uat", rproj) {
#   c(desk, folder.level1, folder.level2, file.level1, file.level2) <-> folders<-
#   if(dir.exists(paste(desk, f, sep = '/'))){
#     create_job(desk, f, dev, rproj)
#   }
#   create.folder(desk, f, folder.level1, folder.level2, file.level1, file.level2, seedir)
#   write_config(paste(desk, f, "tbox", sep = '/'), paste(desk, f, file.level2, sep = '/'))
# }
# Parser
# arg_parser("Scaffold the folder structure in a new laptop")
# p <- add_argument(p, "folder", help = "the folder name", type = "character")
# p <- add_argument(p, "dev", help = "the dev environment", type = "character")
# p <- add_argument(p, "--seedir", help = "set the seedir", 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")
# parse_args(p)
# scaffold(f = args$folder, arg$dir, arg$env, arg$rproj)
# Environment History Connections Build Git Tutorial
```

```
MINGW64/c/Users/Stewart Li/Desktop/raudit solution/lp/fauditorsup/stbook/full_ada (main)
$ ./setup/setup_term.R -h
usage: setup_term.R [-h] [--help] [--dir] [-o OPTS] [-e ENV] [-r RPROJ]
[-f FOLDER]

Scaffold the folder structure in a new laptop

positional arguments:
  folder      provide a folder name

flags:
  -h, --help  show this help message and exit
  -d, --dir   show the folder structure

optional arguments:
  -o, --opts  RProj file containing argument values
  -e, --env   choose dev environment [default: uat]
  -r, --rproj  create a Rproj

$ MINGW64/c/Users/Stewart Li/Desktop/raudit solution/lp/fauditorsup/stbook/full_ada (main)
$
```

Figure 1.21: CLI R - GitBash 2

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 folder named "testmel". A file browser window is also visible at the bottom, showing files like "ignore", "R", and "testmel.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.folder(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 folder named "testmel". The file browser window 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.folder(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 setting up a folder structure, handling command-line arguments, and creating R projects. To the right of the RStudio window is a Windows PowerShell window showing the command `.\run_setup.bat newco -d` being run, which generates a folder structure. Below the PowerShell window is a file explorer showing the created directory `newco` containing subfolders like `data`, `proj`, `raw`, and `res`.

```

# 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 <- 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 the resulting folder structure in Windows PowerShell and file explorer. The command run is `PS C:\Users\Stewart Li\Desktop\newco> .\run_setup.bat newco -d`. The file explorer shows the `newco` directory with subfolders `data`, `proj`, `raw`, and `res`.

```

# 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 <- 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
&gt;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&#x27;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&#x27;s a little fiddly to setup but it&#x27;s worth it if you&#x27;re going for the cellphone model of ownership. Pay large for L.	re10	29798774	1
I do this, and it&#x27;s very nice to tell who lost&#x27;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 HTTP

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

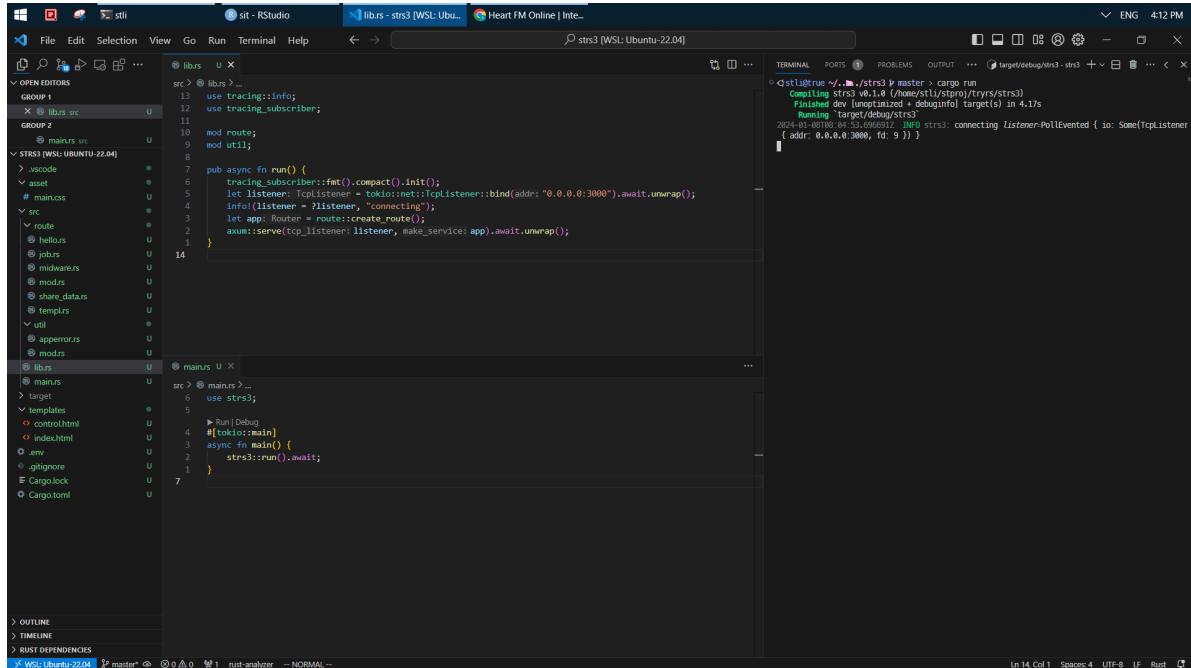


Figure 3.1: Web server

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

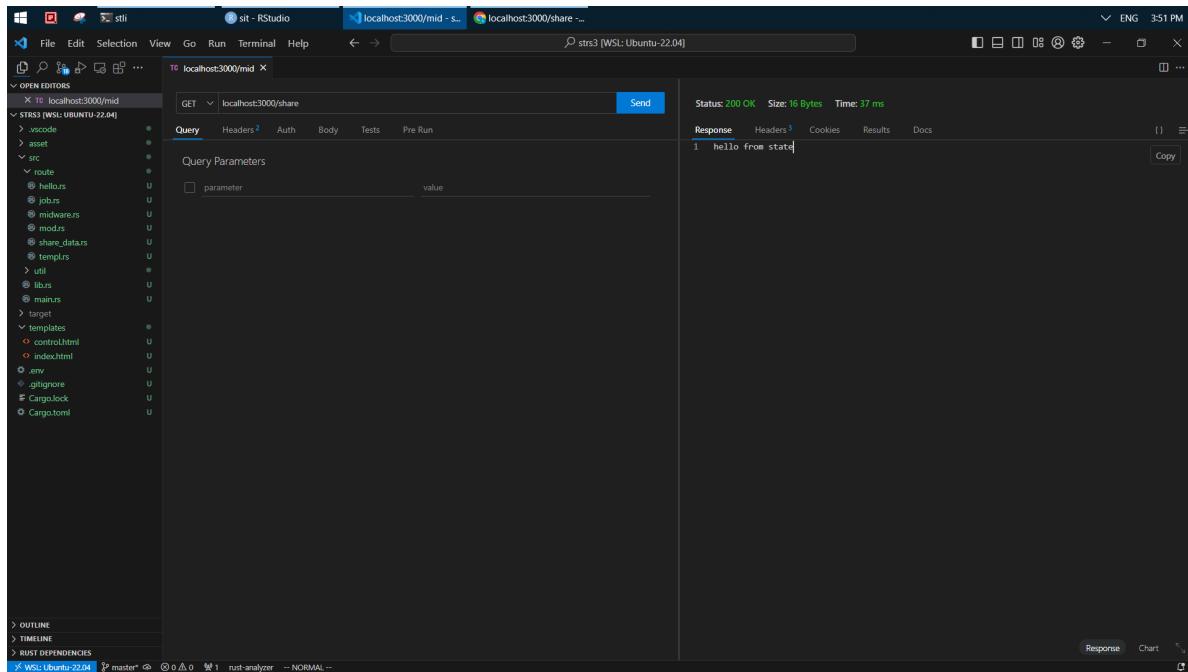


Figure 3.2: Get

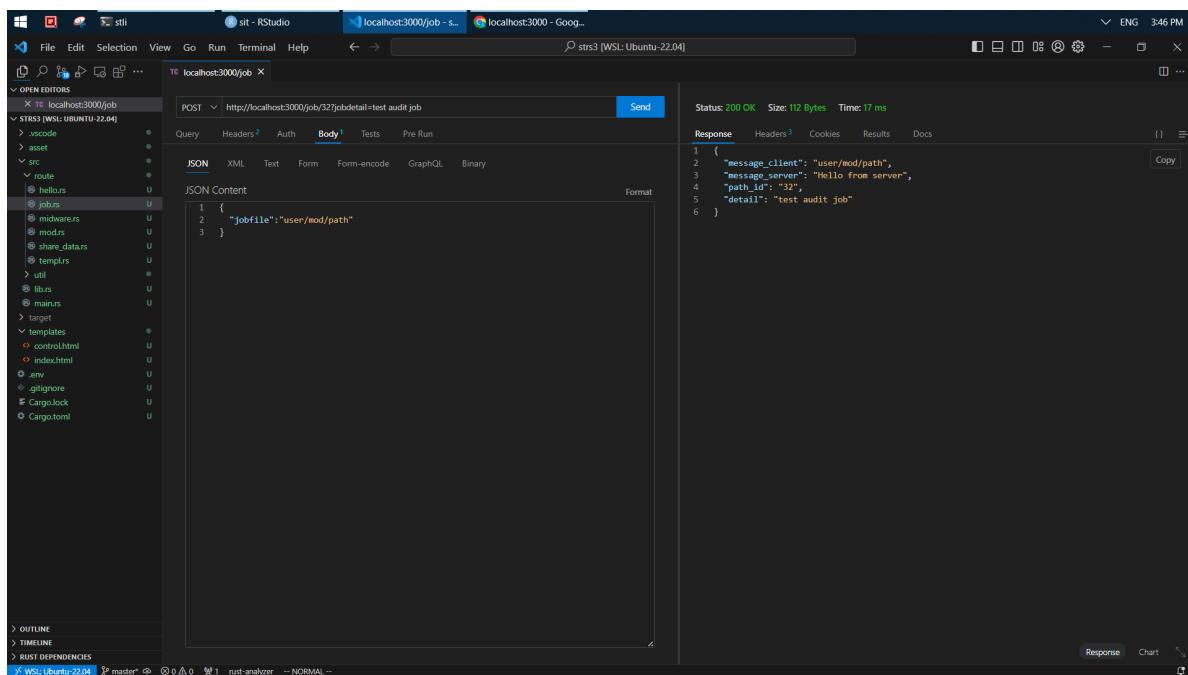


Figure 3.3: Post

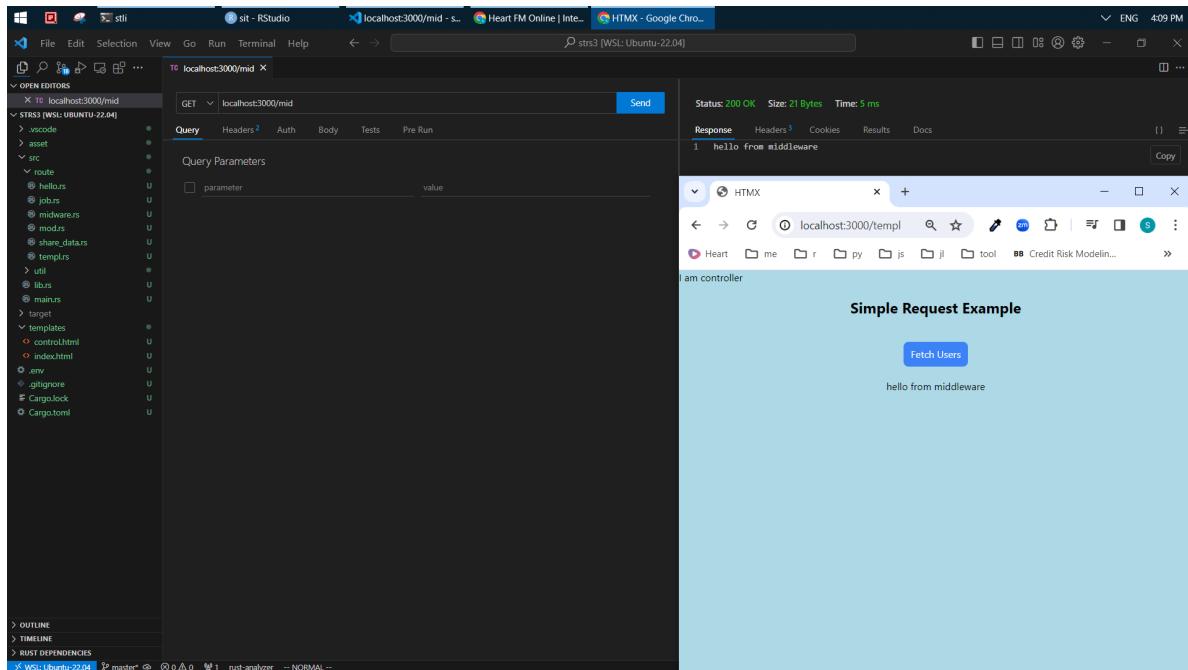


Figure 3.4: Template

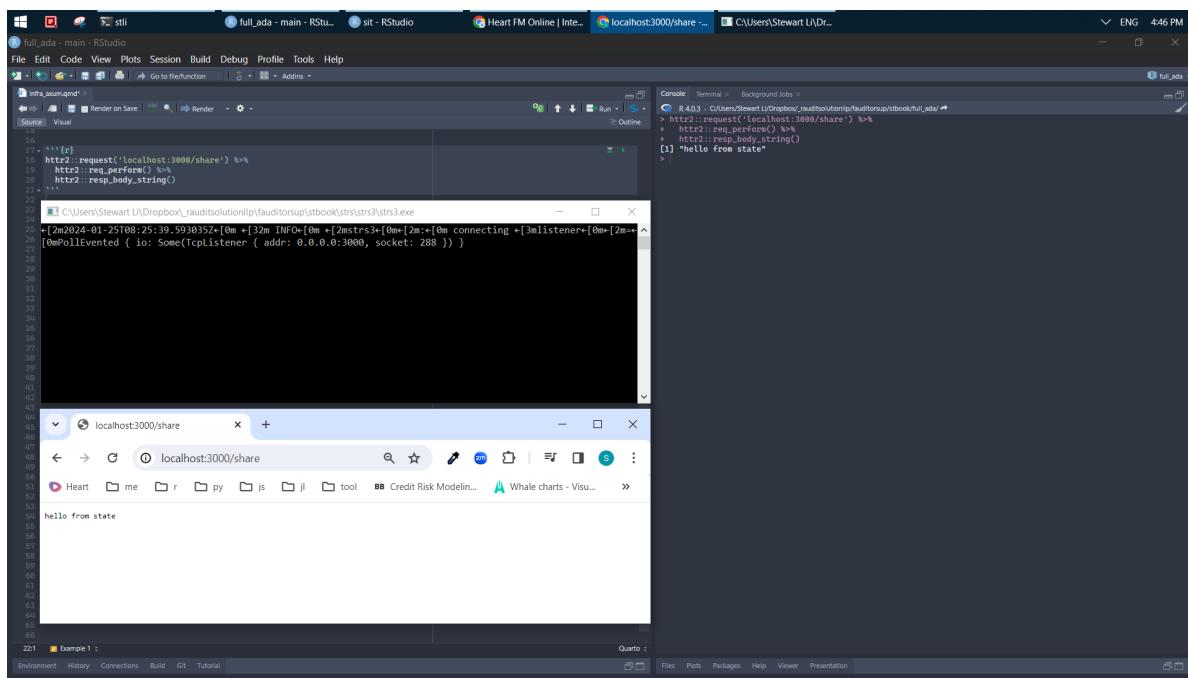


Figure 3.5: R client

The screenshot shows a Windows desktop environment with several open windows. In the foreground, a terminal window titled 'sts - RStudio' displays Rust code. The code includes imports for `std::io` and `Box`, defines a struct `mainus` with fields `stock` and `datas`, and implements a `main` function. The `main` function uses `tokio::run` to handle an asynchronous loop. Inside the loop, it reads command-line arguments, processes them, and then performs a series of operations involving `StringTreeNodes` and `Value` objects, ultimately printing the result to the console.

```
src/main.rs
1 use mainus;
2
3 fn main() {
4     use serde_json::Value;
5     use send_to_trees::FormatCharacters, StringTreeNode, TreeFormatting;
6
7     let stock: String = std::env::args().skip(1).collect::();
8     let datas: Value = manifest_from_server(stock).await.unwrap();
9
10
11     let mut tree: TreeNode<String> = StringTreeNode::new();
12     datas.as_array().unwrap().first().unwrap()["symbol"].Value
13         .as_str().Option<&str>
14             .unwrap() &str
15                 .to_string(),
16
17     tree.push(
18         datas.as_array().unwrap().first().unwrap()["entityRegistrantName"].Value
19             .as_str().Option<&str>
20                 .unwrap() &str
21                     .to_string(),
22
23     tree.push(
24         datas.as_array().unwrap().first().unwrap()["auditorName"].Value
25             .as_str().Option<&str>
26                 .unwrap() &str
27                     .to_string(),
28
29     tree.push(
30         datas.as_array().unwrap().first().unwrap()["netIncomeloss"].Value
31             .as_i64().Option<i64>
32                 .unwrap() i64
33                     .to_string(),
34
35     println!(
36         "{}",
37         tree.to_string_with_format(TreeFormatting::dir_tree(FormatCharacters::box_chars()))
38             .unwrap()
39     );
40 }
41
42 } fn main
43
44 async fn manifest_from_server(ticker: String) -> Result<Value, Box<dyn std::error::Error>> {
45     let url: String =
46         "https://financialmodelingprep.com/api/v3/financial-statement-full-as-reported/".to_owned();
47     let client = Client::new();
48     let response = client.get(url).header("User-Agent", "Mozilla/5.0").send().await;
49
50     if response.is_err() {
51         return Err("Failed to fetch financial statement from the API.".into());
52     }
53
54     let body = response.unwrap().text().await;
55
56     if body.is_err() {
57         return Err("Failed to parse the JSON response from the API.".into());
58     }
59
60     let value: Value = serde_json::from_str(&body.unwrap()).unwrap();
61
62     Ok(value)
63 }
```

Figure 3.6: Request CLI 1

Figure 3.7: Request CLI 2

4 FAudit

Create a command line tool to organize the workflow including folder structure and relevant config files.

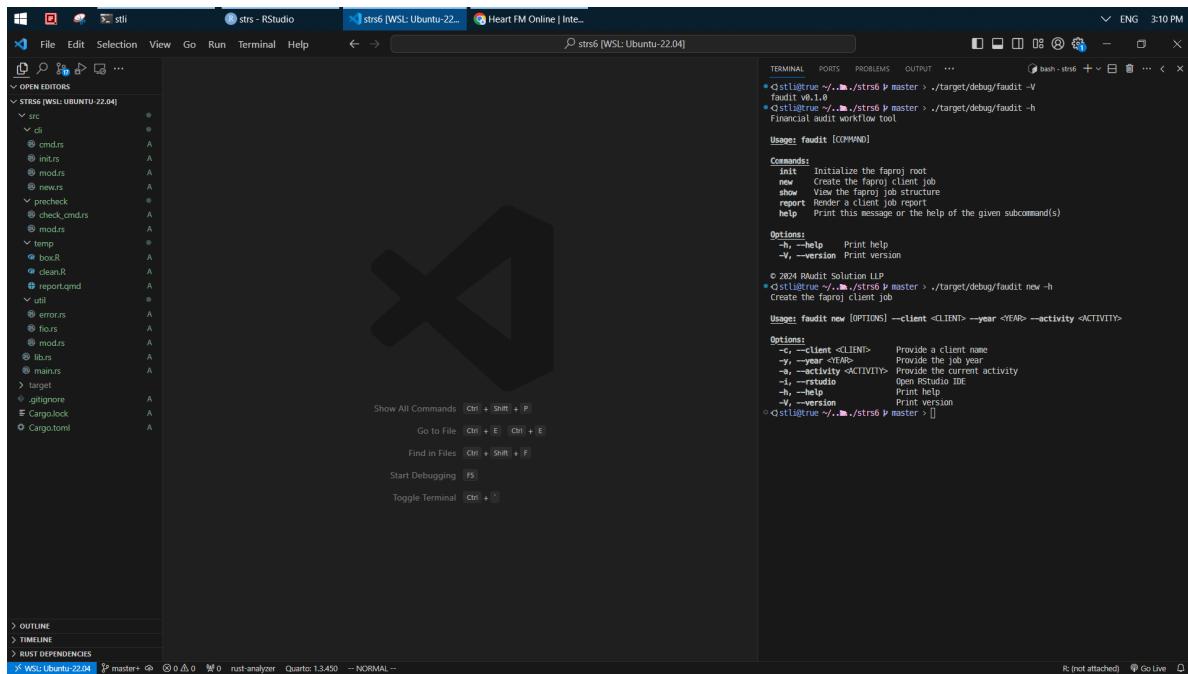


Figure 4.1: faudit help

The screenshot shows a Windows terminal window with several tabs open. The tabs include 'sts - RStudio', 'sts - RStudio', 'box.R - sts6 [WSL: Ubuntu-22.04]', 'Heart FM Online | Inte...', and 'sts6 [WSL: Ubuntu-22.04]'. The main area of the terminal is a code editor displaying R script code. The code includes functions for checking R packages and installing missing ones. The right side of the terminal has tabs for 'TERMINAL', 'PORTS', 'PROBLEMS', 'OUTPUT', and 'COMMAND'. The bottom of the screen shows the Windows taskbar with icons for File Explorer, Task View, Start, Task Manager, and others.

```
faproj > box > boxR > check_r_pkg
1  #' @description "Welcome test"
2  #' @param ...
3  #' @return ...
4  hello <- function(...) {
5    cat(cli$col_red(~` Welcome to FAudit\n`))
6  }
7  #' @description "Check if the required R packages are installed"
8  #' @export
9  check_pkgs <- function() {
10    deps <- c(
11      ...DBI", "data.table", "tidyverse", "pointblank", "gt", "shiny", "quarto", "renv", "target",
12      "devtools", "usethis", "fusen", "pak", "cli"
13    )
14    needed <- setdiff(deps, rownames(installed.packages()))
15    if (length(needed) == 0) {
16      cat(~`you are good now`)
17    } else {
18      answer <- askYesNo(~`Do you want to install all?`)
19      if (is.na(answer) && answer) {
20        install.packages(needed)
21      } else {
22        cat(~`you need to install missing packages`)
23      }
24    }
25  }
26  # config.yml
27  faproj > box > ! config.yml
28  default:
29    rbox: /home/stili/stproj/trrys/strs6/faproj/box
30    duckdb: ''
31    psql: ''
32    pin: ''
33    fpp: ''
34    dev:
35      psql: ''
36      duckdb: ''
37      pin: ''
38    rbox: /home/stili/stproj/trrys/strs6/faproj/box
39    fpp: ''
40
41  # config.json
42  faproj > ! configuration
43  1
```

Figure 4.2: faudit init

The screenshot shows a Windows desktop environment with several open windows. In the foreground, the Visual Studio Code (VS Code) application is running. The left sidebar displays a file tree for a project named 'faproj'. The main editor area has four tabs open:

- `faproj`: A JSON configuration file containing activity logs for a job in Shanghai.
- `clean.R`: An R script that performs cleanup operations like removing old files and checking dependencies.
- `report.qmd`: A QMD report template for data cleaning.
- `config.json`: Another JSON configuration file.

A terminal window titled 'strs6 [WSL: Ubuntu-22.04]' is visible at the bottom, showing command-line interactions related to the project's build process. The status bar at the bottom provides information about the current file and workspace.

Figure 4.3: faudit new 1

```

> config <- config_get(file = file.path(Sys.getenv("USER_FA_DIR"), "/box/config.yml"))
> options(box_path = config$box)
> getoption("box.path")
[1] "/home/stli/stproj/trysrstrs6/faproj/box"
> boxShellInfo()
> boxShowActivity()
> boxShowActivity()
activity                         date
1 start 2024-02-29T07:33:49.587883866
1 /home/stli/stproj/trysrstrs6/faproj/job/shanghai/2023 shanghai_2023
> []

```

Figure 4.4: faudit new 2

mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb	
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1

Figure 4.5: faudit report

```

faudit new -c shanghai -y 2023 -a "open ide" -i
/home/stli/stproj/tryrs/strs6/faproj/job/shanghai/2023 as it exists
your cwd is /home/stli/stproj/tryrs/strs6/faproj/job/shanghai/2023
faudit new -c shanghai -y 2023 -a "open ide"

```

Figure 4.6: faudit new 3

```

faudit new -c beijing -y 2023 -a start
/home/stli/stproj/myaudit/faproj/job/beijing/2023 as it exists
your cwd is /home/stli/stproj/myaudit/faproj/job/beijing/2023
faudit new -c beijing -y 2023 -a start

```

Figure 4.7: faudit new 4

The screenshot shows a terminal window titled 'strs6 [WSL: Ubuntu-22.04]' with the command 'faudit show' running. The output indicates that the USER\_FA\_DIR environment variable is set to '/shanghai/2023'. The terminal also shows the directory structure of the project, which includes 'src', 'target', and 'Cargo.toml' files.

```
ENG 5:11 PM
strs6 [WSL: Ubuntu-22.04]
File Edit Selection View Go Run Terminal Help
strs6 [WSL: Ubuntu-22.04] rust std::process::Com...
strs6 [WSL: Ubuntu-22.04]
TERMINAL PORTS PROBLEMS OUTPUT DEBUG CONSOLE SQL CONSOLE
bash - str6 + ... < >
faudit show
USER_FA_DIR is set
└─ shanghai
   └─ 2023
2 directories, 0 files
faudit true ./strs6 master > ./target/debug/faudit show list
USER_FA_DIR is set
./shanghai
./shanghai/2023
faudit true ./strs6 master > []
```

Figure 4.8: faudit show

## **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 visualization<sup>1</sup>:
  - 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,

---

<sup>1</sup>ggplot2 (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`,

# 5 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. On the left, there is a code editor window titled 'stcli.py' containing Python code. The code uses the argparse module to parse command-line arguments and the duckdb module to interact with a DuckDB database. It includes functions for filtering and summarizing data from a 'finsample' table. On the right, there is a terminal window titled 'stclienv [WSL: Ubuntu-20.04]' showing the output of running the script. The terminal shows the command './duckdb finsample.duckdb', loading resources, and then executing a query to print the 'finsample' table. The table has columns: segment, country, product, cogs, profit, and date. The data shows three rows for Government and Midmarket segments across Canada, Germany, and France. The terminal also shows the command 'stcli.py -f'.

```
stcli.py
1 import argparse
2 import duckdb
3
4
5 def st_filter(db, tbl, ex="cogs > 200000"):
6     conn = duckdb.connect(db)
7     df = conn.execute(f"select * from {tbl}").df()
8     df_res = duckdb.filter(df, ex)
9     conn.close()
10    print(df_res)
11
12
13 def st_summarize(db, tbl):
14     conn = duckdb.connect(db)
15     df = conn.execute(f"select * from {tbl}").df()
16     df_res = duckdb.sql(
17         """select segment, country, count(*) as n from df group by all order by n desc"""
18     )
19     conn.close()
20     print(df_res)
21
22
23 if __name__ == "__main__":
24     parser = argparse.ArgumentParser()
25     # namespace db contains multiple args
26     parser.add_argument("db", type=str, nargs="+")
27     # optional function
28     parser.add_argument(
29         "--filter",
30         dest="do",
31         action="store_const",
32         const=st_filter,
33         default=st_summarize,
34     )
35     args = parser.parse_args()
36     # st_filter(*args[0])
37     args.do(*args.db)
38
39
40 # python3 stcli.py ~/finsample.duckdb finsample
41 # python3 stcli.py ~/finsample.duckdb finsample --filter
42
WSL:at: 25% bash 1/4GB 1m 22s 746ms
./duckdb finsample.duckdb
-- Loading resources from /home/stli/.duckdbrc
v0.8.1 6536a7732
Enter "help" for usage hints.
> .table
finsample
> from finsample limit 3;
+-----+-----+-----+-----+-----+
| segment | country | product | cogs | profit |
+-----+-----+-----+-----+-----+
| Government | Canada | Carretera | 36185.0 | 16185.0 |
| Government | Germany | Carretera | 13210.0 | 13210.0 |
| Midmarket | France | Carretera | 21780.0 | 10890.0 |
+-----+-----+-----+-----+-----+
3 rows
13 columns (6 shown)
> .q
WSL:at: 26.64% bash 1/4GB 1m 15s 919ms
0 88.45% Q
In 42 Col 1 Spaces:4 UTF-8 LF Python 3.8.10 (myenv)env
```

Figure 5.1: CLI - argparse 1

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

- File Explorer**: Shows a folder structure.
- Terminal**: Shows command-line interactions:
  - ls -l: Lists files in the current directory.
  - cd stcli/proj/stlipyenv: Changes directory to a virtual environment.
  - python3 stcli.py ~/finsample.duckdb finsample: Runs a Python script to process a DuckDB database.
- Code Editor**: An IDE window titled "stcli.py" containing Python code. The code connects to a DuckDB database, filters data, and prints results. It also includes a main function that parses command-line arguments for a database file and a filter option.
- Terminal (Background)**: Shows a table output from a DuckDB query. The table has columns: segment, country, product, cogs, profit, and date timestamp. Data rows include Government, Canada, Carretera, 16185.0, 16185.0, 2014-01-01 00:00:00, and Government, Germany, Carretera, 13220.0, 13220.0, 2014-01-01 00:00:00, among others.
- Terminal (Bottom)**: Shows another table output from a Python script. This table has columns: segment, country, and n. Data rows include Government, Canada, 60, and Government, Germany, 60, among others.
- System Tray**: Shows battery status, signal strength, and other system icons.

The bottom status bar indicates "Running in Ubuntu-20.04 (WSL 2)".

Figure 5.2: CLI - argparse 2

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

- RStudio**: A window titled "full\_ada - RStudio" containing R code.
- stlpy**: A window titled "stlpy > ...". It contains Python code for filtering and summarizing data from a DuckDB database.
- Terminal**: A window titled "stlpyenv [WSL: Ubuntu-20.04]" showing the output of running the Python script.

The terminal output shows the following command and its results:

```
python3 stlcli.py ~/finsample.duckdb finsample --filter
```

segment	country	profit	date
varchar	varchar	double	timestamp
Government	Germany	136170.0	2014-12-01 00:00:00
Enterprise	USA	13310.0	2014-08-01 00:00:00
Small Business	Mexico	47900.0	2014-08-01 00:00:00
Government	Germany	90540.0	2014-06-01 00:00:00
Government	Canada	155250.0	2013-11-01 00:00:00
Government	Germany	150000.0	2013-11-01 00:00:00
Enterprise	France	9820.0	2014-02-01 00:00:00
Government	Germany	90540.0	2014-06-01 00:00:00
Enterprise	United States of A...	14105.0	2014-08-01 00:00:00
Small Business	Canada	100850.0	2014-02-01 00:00:00
	.	.	.
Small Business	Canada	7104.0	2014-03-01 00:00:00
Enterprise	United States of A...	-3550.0	2014-05-01 00:00:00
Enterprise	Germany	-38064.25	2014-08-01 00:00:00
Government	Mexico	186912.5	2014-05-01 00:00:00
Government	United States of A...	20700.0	2013-11-01 00:00:00
Small Business	Mexico	88662.5	2013-11-01 00:00:00
Enterprise	United States of A...	12870.0	2013-11-01 00:00:00
Enterprise	Canada	-33522.5	2013-12-01 00:00:00
Small Business	France	-40617.5	2014-03-01 00:00:00

Below the table, the terminal shows:

```
202 rows (28 shown) 13 columns (4 shown)
```

At the bottom of the terminal window, there is another prompt:

```
WSL at 8 bash [MEM: 26.58% | 1/4GB 8 726ms]
```

Figure 5.3: 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     def main_group():
76         ...

77     @click.command()
78     @click.pass_context
79     @click.argument("output", type=click.File("w"), default="-", required=False)
80     @click.argument("highlight", type=click.Choice(["red", "green"]),
81                    help="highlight data based on the provided threshold",
82                )
83     def condc(ctx, output, highlight):
84         ...

85     choose your data
86     ...
87     res = ctx.obj.with_columns(
88         new(
89             pl.when(pl.col("hp") > 200
90                   .then(pl.col("mpg").filter(pl.col("hp") > 200).sum())
91                   .otherwise(pl.col("mpg").filter(pl.col("hp") < 200).sum())
92             )
93         ).round(2)
94         .cast(pl.Utf8)
95     )
96     if highlight == "red":
97         res = res.with_columns(pl.col("new").map(lambda x: f"\033[91m{x} + \033[0m"))
98     else:
99         res = res.with_columns(pl.col("new").map(lambda x: f"\033[96m{x} + \033[0m"))
100    # click.echo(output)
101    click.echo(res)

102 if __name__ == "__main__":
103     main()

104 if __name__ == "__main__":
105     main()

106 if __name__ == "__main__":
107     main()

108 if __name__ == "__main__":
109     main()

110 if __name__ == "__main__":
111     main()

112 if __name__ == "__main__":
113     main()

114 if __name__ == "__main__":
115     main()

116 if __name__ == "__main__":
117     main()

118 if __name__ == "__main__":
119     main()

120 if __name__ == "__main__":
121     main()

122 if __name__ == "__main__":
123     main()

124 if __name__ == "__main__":
125     main()

126 if __name__ == "__main__":
127     main()

128 if __name__ == "__main__":
129     main()

130 if __name__ == "__main__":
131     main()

132 if __name__ == "__main__":
133     main()

134 if __name__ == "__main__":
135     main()

136 if __name__ == "__main__":
137     main()

138 if __name__ == "__main__":
139     main()

140 if __name__ == "__main__":
141     main()

142 if __name__ == "__main__":
143     main()

144 if __name__ == "__main__":
145     main()

146 if __name__ == "__main__":
147     main()

148 if __name__ == "__main__":
149     main()

150 if __name__ == "__main__":
151     main()

152 if __name__ == "__main__":
153     main()

154 if __name__ == "__main__":
155     main()

156 if __name__ == "__main__":
157     main()

158 if __name__ == "__main__":
159     main()

160 if __name__ == "__main__":
161     main()

162 if __name__ == "__main__":
163     main()

164 if __name__ == "__main__":
165     main()

166 if __name__ == "__main__":
167     main()

168 if __name__ == "__main__":
169     main()

170 if __name__ == "__main__":
171     main()

172 if __name__ == "__main__":
173     main()

174 if __name__ == "__main__":
175     main()

176 if __name__ == "__main__":
177     main()

178 if __name__ == "__main__":
179     main()

180 if __name__ == "__main__":
181     main()

182 if __name__ == "__main__":
183     main()

184 if __name__ == "__main__":
185     main()

186 if __name__ == "__main__":
187     main()

188 if __name__ == "__main__":
189     main()

190 if __name__ == "__main__":
191     main()

192 if __name__ == "__main__":
193     main()

194 if __name__ == "__main__":
195     main()

196 if __name__ == "__main__":
197     main()

198 if __name__ == "__main__":
199     main()

200 if __name__ == "__main__":
201     main()

202 if __name__ == "__main__":
203     main()

204 if __name__ == "__main__":
205     main()

206 if __name__ == "__main__":
207     main()

208 if __name__ == "__main__":
209     main()

210 if __name__ == "__main__":
211     main()

212 if __name__ == "__main__":
213     main()

214 if __name__ == "__main__":
215     main()

216 if __name__ == "__main__":
217     main()

218 if __name__ == "__main__":
219     main()

220 if __name__ == "__main__":
221     main()

222 if __name__ == "__main__":
223     main()

224 if __name__ == "__main__":
225     main()

226 if __name__ == "__main__":
227     main()

228 if __name__ == "__main__":
229     main()

230 if __name__ == "__main__":
231     main()

232 if __name__ == "__main__":
233     main()

234 if __name__ == "__main__":
235     main()

236 if __name__ == "__main__":
237     main()

238 if __name__ == "__main__":
239     main()

240 if __name__ == "__main__":
241     main()

242 if __name__ == "__main__":
243     main()

244 if __name__ == "__main__":
245     main()

246 if __name__ == "__main__":
247     main()

248 if __name__ == "__main__":
249     main()

250 if __name__ == "__main__":
251     main()

252 if __name__ == "__main__":
253     main()

254 if __name__ == "__main__":
255     main()

256 if __name__ == "__main__":
257     main()

258 if __name__ == "__main__":
259     main()

260 if __name__ == "__main__":
261     main()

262 if __name__ == "__main__":
263     main()

264 if __name__ == "__main__":
265     main()

266 if __name__ == "__main__":
267     main()

268 if __name__ == "__main__":
269     main()

270 if __name__ == "__main__":
271     main()

272 if __name__ == "__main__":
273     main()

274 if __name__ == "__main__":
275     main()

276 if __name__ == "__main__":
277     main()

278 if __name__ == "__main__":
279     main()

280 if __name__ == "__main__":
281     main()

282 if __name__ == "__main__":
283     main()

284 if __name__ == "__main__":
285     main()

286 if __name__ == "__main__":
287     main()

288 if __name__ == "__main__":
289     main()

290 if __name__ == "__main__":
291     main()

292 if __name__ == "__main__":
293     main()

294 if __name__ == "__main__":
295     main()

296 if __name__ == "__main__":
297     main()

298 if __name__ == "__main__":
299     main()

299

```

Figure 5.4: 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     def main_group():
76         ...

77     @click.command()
78     @click.pass_context
79     @click.argument("output", type=click.File("w"), default="-", required=False)
80     @click.argument("highlight", type=click.Choice(["red", "green"]),
81                    help="highlight data based on the provided threshold",
82                )
83     def condc(ctx, output, highlight):
84         ...

85     choose your data
86     ...
87     res = ctx.obj.with_columns(
88         new(
89             pl.when(pl.col("hp") > 200
90                   .then(pl.col("mpg").filter(pl.col("hp") > 200).sum())
91                   .otherwise(pl.col("mpg").filter(pl.col("hp") < 200).sum())
92             )
93         ).round(2)
94         .cast(pl.Utf8)
95     )
96     if highlight == "red":
97         res = res.with_columns(pl.col("new").map(lambda x: f"\033[91m{x} + \033[0m"))
98     else:
99         res = res.with_columns(pl.col("new").map(lambda x: f"\033[96m{x} + \033[0m"))
100    # click.echo(output)
101    click.echo(res)

102 if __name__ == "__main__":
103     main()

104 if __name__ == "__main__":
105     main()

106 if __name__ == "__main__":
107     main()

108 if __name__ == "__main__":
109     main()

110 if __name__ == "__main__":
111     main()

112 if __name__ == "__main__":
113     main()

114 if __name__ == "__main__":
115     main()

116 if __name__ == "__main__":
117     main()

118 if __name__ == "__main__":
119     main()

120 if __name__ == "__main__":
121     main()

122 if __name__ == "__main__":
123     main()

124 if __name__ == "__main__":
125     main()

126 if __name__ == "__main__":
127     main()

128 if __name__ == "__main__":
129     main()

130 if __name__ == "__main__":
131     main()

132 if __name__ == "__main__":
133     main()

134 if __name__ == "__main__":
135     main()

136 if __name__ == "__main__":
137     main()

138 if __name__ == "__main__":
139     main()

139

```

Figure 5.5: CLI - click 2

```

stli@stli:~/stliproj/testv1$ python3 ./stclick/stclick.py "/home/stli/stliproj/testv1/mtcars.csv" calc condc
data source: /home/stli/stliproj/testv1/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/testv1$ grep V
Volvo

stli@stli:~/stliproj/testv1$ python3 ./stclick/stclick.py "/home/stli/stliproj/testv1/mtcars.csv" calc condc | grep V
Volvo

stli@stli:~/stliproj/testv1$ python3 ./stclick/stclick.py "/home/stli/stliproj/testv1/mtcars.csv" calc condc
data source: /home/stli/stliproj/testv1/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 5.6: CLI - click 3

```

stli@stli:~/stliproj/testv1$ stpolars "/home/stli/stliproj/testv1/mtcars.csv" about
data source: /home/stli/stliproj/testv1/mtcars.csv
shape: (32, 13)

stli@stli:~/stliproj/testv1$ python3 ./stclick/stclick.py "/home/stli/stliproj/testv1/mtcars.csv" calc condc
data source: /home/stli/stliproj/testv1/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/testv1$ python3 ./stclick/stclick.py "/home/stli/stliproj/testv1/mtcars.csv" calc condc | grep V
Volvo

stli@stli:~/stliproj/testv1$ python3 ./stclick/stclick.py "/home/stli/stliproj/testv1/mtcars.csv" calc condc
data source: /home/stli/stliproj/testv1/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 5.7: CLI - click 4

File Edit Selection View Go Run Terminal Help

client.json

```
clientA
  "clientList": [{"name": "clientA"}, {"year": 2023}, {"joblist": "clientA_2023"}, {"auditor": "stewartli"}, {"status": false}]]
```

rootproj.toml

```
[JobList]
joblist = ["ClientA_2023"]
```

Terminal

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

fauditor=#

```

Compiling home v0.5.5  
Compiling hdkf v0.12.3  
Compiling rustix v0.38.20  
Compiling md-5 v0.10.6  
Compiling tokio v0.26.0  
Compiling tokio-stream v0.1.14  
Compiling sqlx-core v0.7.2  
Compiling tempfile v1.8.0  
Compiling byteorder v1.0.4  
Compiling serde\_spans v0.6.3  
Compiling tom\_l\_datetime v0.6.3  
Compiling serde\_yaml v0.9.25  
Compiling tokio-util v0.6.0  
Compiling sqlx-postgres v0.7.2  
Compiling sqlx-macros-core v0.7.2  
Compiling tom\_l v0.8.2  
Compiling byteorder v1.0.4  
Compiling sqlx-macros v0.7.2  
Compiling sqlx v0.7.2  
Compiling testoml v0.1.0 (/home/stilt/stiltpro/trust/testoml)
initializing + (debugging) target(s) in 1m 27s
running 'target/debug/testoml' as clientA\_2023 + stewartli init'
> Initializing in [/home/stilt/stiltpro/trust/testoml]

WSL at E bash master 16:45 6/10/24 1m 27s 63ms

Figure 5.8: CLI - clap 1

The screenshot shows a Windows desktop with several open windows. At the top, there's a taskbar with icons for File Explorer, Task View, and Start. Below the taskbar, the desktop background is a dark blue gradient.

**RStudio Session:** The main window is titled "awp.R - testtomi [WSL]". It displays code in the "awp.R" file, which includes imports from tidyverse and a function definition. The "TERMINAL" tab is active, showing a WSL terminal session for "testtomi". The terminal output shows the execution of cargo commands like "cargo run -- -n clientA -y 2023" and "cargo run -- -n clientA -y 2023 -a stewartl init". Other visible text in the terminal includes "cargo run -- -n clientA -y 2023 -a stewartl new -p '/mnt/c/Users/Stewart/LI/Desktop/mpbc/'", "Running dev [unoptimized + debuginfo] target(s) in B-12s", and "Running 'target/debug/testtoml' -a stewartl".

**Terminal Session:** A separate terminal window titled "awp.R - testtomi [WSL: Ubuntu-20.04]" is also visible, showing a similar command-line interface with "cargo" and "stewartl" commands.

**File Explorer:** A sidebar titled "EXPLORER" shows the file structure of the "awp.R" project, including "src", "tests", and "supp" directories, along with various R files and CSV files.

**Bottom Navigation:** The bottom of the screen features a navigation bar with icons for "OUTLINE", "TIMELINE", and "RUST DEPENDENCIES".

Figure 5.9: CLI - clap 2

# 6 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](#).

## 6.1 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)
```

## 6.2 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))
```

## 6.3 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))

```

## 6.4 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'))

```

## 6.5 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 %>%

```

full\_ada - RStudio Heart FM Online | Int... ENG 95% AM

File Edit Code View Plots Session Build Debug Profile Tools Help

File Edit Code View Plots Session Build Debug Profile Tools Help Go to Refunction Addins

Source Visual

133 `##[r]`

134 `df_clean %>%`

135 `group_by(country_code) %>%`

136 `arrange(first_disputed)`

137 `summarise(n = n(),`

138 `sales = sum(invoice_amount),`

139 `late = mean(late),`

140 `first_disputed_inv_date = first(invdate[date.disputed == 'Yes']),`

141 `largest_late = max(late[date.disputed == 'Yes']),`

142 `largest_inv_amt = invoice_amount[late == max(late)], .groups = 'drop'`

143 `##`

144 `##`

145 `Cut late into four categories based on the Firm's credit policy.`

146 `##`

147 `##`

148 `##`

149 `##[r]`

150 `sort(unique(df_clean$late))`

151 `##`

152 `df_late <- df_clean %>%`

153 `dplyr::filter(late > 0 & late <= 10 ~ "1st email",`

154 `mutate(reminder = case_when(late > 0 & late <= 10 ~ "1st email",`

155 `late > 10 & late <= 28 ~ "2nd email",`

156 `late > 28 & late <= 38 ~ "legal case",`

157 `TRUE ~ "bad debt")`

158 `##`

159 `# anomaly by country`

160 `df_late %>%`

161 `ggplot(aes(late, disputed, color = country_code)) +`

162 `geom_boxplot() +`

163 `theme_light()`

164 `##`

165 `# summary table`

166 `df_late %>%`

167 `group_by(reminder, disputed) %>%`

168 `summarise(across(late, tibble::lst(sum, min, max, sd)), .groups = 'drop') %>%`

169 `gt() %>%`

170 `gt::gt`

171 `# clients without dispute do not pay.`

172 `df_late %>%`

173 `dplyr::filter(disputed == 'No', reminder %in% c('legal case', 'bad debt'))`

174 `##`

175 `##`

176 `## EDA`

177 `##`

178 `##`

179 `## Focus on a handful of variables after dropped others.`

180 `##`

181 `##`

182 `##`

183 `##[r]`

184 `Merging :`

185 `##`

Console Terminal Background Jobs

R 4.0.3 C:\Users\Stuart\Dropbox\rauthdsolutions\heartfmdata\ada\ada.R

1 161 29023 01-01-05 41 9

2 406 561 39023 31 2012-01-13 45 86.4

3 778 586 273881 24 2012-01-18 34 75.2

4 818 347 39292 13 2012-01-09 34 87

5 997 396 16388 6 2012-01-07 32 51.8

# add table & 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30

[32] 31 32 33 34 35 37 45

df\_late <- df\_clean %>%

dplyr::filter(late > 0) %>%

mutate(case\_when = case\_when(late > 0 & late <= 10 ~ "1st email",

late > 10 & late <= 28 ~ "2nd email",

late > 28 & late <= 38 ~ "legal case",

TRUE ~ "bad debt")

# anomaly by country

df\_late %>%

group\_by(crossdate, disputed, color = country\_code) +

geom\_boxplot() +

theme\_light()

# EDA

df\_late %>%

group\_by(reminder, disputed) %>%

summarise(across(late, tibble::lst(sum, min, max, sd)), .groups = 'drop') %>%

df\_late %>%

dplyr::filter(disputed == 'No', reminder %in% c('legal case', 'bad debt'))

A tibble: 13 x 12

country\_code customer\_id paperless\_bill disputed invoice\_number invoice\_amount invoice\_date

1 897 1488-QQZE Paper No 265518254 27.6 2012-06-05

2 897 6688-XNRO Paper No 981596189 45.8 2012-06-07

3 778 7228-LEMP Paper No 165784645 27.6 2012-06-19

4 897 6688-XNRO Electronic No 230855977 33.8 2013-01-12

5 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

6 486 2621-XCLZ Paper No 5724623284 89.6 2012-09-23

7 818 1646-LTRKX Paper No 585997997 67.6 2012-09-23

8 818 1646-LTRKX Paper No 592697811 77.6 2012-01-01

9 897 8698-EEDP Paper No 621956346 71.3 2012-05-16

10 897 6688-XNRO Electronic No 6254565089 86.6 2013-01-18

11 897 6688-XNRO Electronic No 6254565089 86.6 2013-01-18

12 818 1646-LTRKX Paper No 9385395392 54.4 2012-03-08

13 897 8698-EEDP Paper No 964751843 71.6 2012-05-25

14 897 1488-QQZE Paper No 265518254 27.6 2012-06-05

15 897 6688-XNRO Paper No 981596189 45.8 2012-06-07

16 897 6688-XNRO Electronic No 230855977 33.8 2013-01-12

17 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

18 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

19 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

20 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

21 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

22 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

23 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

24 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

25 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

26 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

27 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

28 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

29 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

30 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

31 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

32 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

33 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

34 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

35 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

36 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

37 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

38 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

39 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

40 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

41 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

42 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

43 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

44 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

45 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

46 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

47 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

48 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

49 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

50 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

51 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

52 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

53 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

54 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

55 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

56 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

57 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

58 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

59 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

60 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

61 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

62 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

63 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

64 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

65 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

66 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

67 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

68 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

69 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

70 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

71 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

72 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

73 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

74 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

75 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

76 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

77 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

78 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

79 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

80 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

81 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

82 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

83 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

84 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

85 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

86 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

87 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

88 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

89 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

90 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

91 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

92 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

93 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

94 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

95 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

96 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

97 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

98 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

99 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

100 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

101 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

102 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

103 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

104 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

105 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

106 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

107 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

108 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

109 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

110 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

111 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

112 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

113 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

114 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

115 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

116 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

117 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

118 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

119 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

120 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

121 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

122 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

123 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

124 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

125 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

126 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

127 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

128 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

129 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

130 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

131 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

132 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

133 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

134 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

135 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

136 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

137 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

138 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

139 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

140 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

141 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

142 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

143 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

144 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

145 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

146 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

147 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

148 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

149 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

150 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

151 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

152 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

153 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

154 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

155 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

156 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

157 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

158 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

159 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

160 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

161 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

162 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

163 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

164 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

165 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

166 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

167 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

168 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

169 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

170 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

171 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

172 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

173 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

174 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

175 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

176 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

177 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

178 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

179 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

180 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

181 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

182 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

183 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

184 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

185 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

186 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

187 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

188 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

189 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

190 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

191 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

192 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

193 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

194 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

195 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

196 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

197 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

198 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

199 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

200 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

201 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

202 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

203 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

204 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

205 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

206 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

207 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

208 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

209 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

210 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

211 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

212 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

213 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

214 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

215 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

216 897 6688-XNRO Electronic No 6349329313 34.8 2013-01-12

217 897 6688-XNRO Electronic No 6349329313 34.8 2013-0

Figure 6.1: 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()
```

## 6.6 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()

```

## 6.7 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())

```

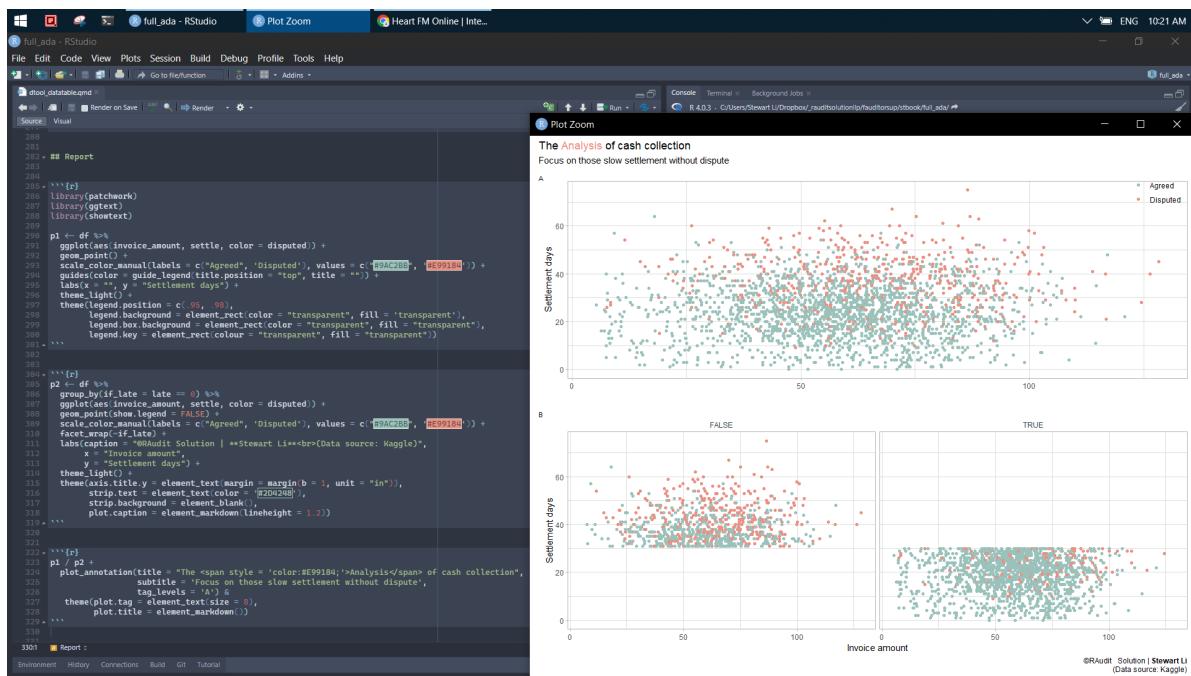


Figure 6.2: Combined plot

# 7 Audit

[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 `lm` 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.

## 7.1 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()
```

The screenshot shows two side-by-side RStudio sessions comparing R and Python code for data analysis.

**Left Session (R):**

```
stli@true ~/.../testf > bat a.R
File: a.R
1 cat(cle::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 = gear]
14 df[, sum(mpg > mean(mpg)), .(am, gear)]
15
16 # Part 3 - Summarize
17 df[, .SDby, by = (am, gear)]
18 df[, tapply(.SD, max), .(am, gear)]
19
20 with(df, tapply(mpg, list(am, gear), max, default = 0))
21 with(df, by(gear, c(gear), summary))
22 aggregate(mpg ~ am * gear, data = df, FUN = median, subset = df$hpa > 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_if := ifelse(mpg < 15, "small", mpg > 30, "large", default = "medium")]
37
38 stli@true ~/.../testf >
```

**Right Session (Python):**

```
stli@true ~/.../testf > bat a1.R
File: a1.R
1 cat(cle::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 = gear]
14 df[, sum(mpg > mean(mpg)), .(am, gear)]
15
16 # Part 3 - Summarize
17 df[, .SDby, by = (am, gear)]
18 df[, tapply(.SD, max), .(am, gear)]
19
20 with(df, tapply(mpg, list(am, gear), max, default = 0))
21 with(df, by(gear, c(gear), summary))
22 aggregate(mpg ~ am * gear, data = df, FUN = median, subset = df$hpa > 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_if := ifcase(mpg < 15, "small", mpg > 30, "large", default = "medium")]
37
38 stli@true ~/.../testf >
```

Figure 7.1: Diff 1

The screenshot shows a Windows desktop with several open windows. In the center is a large RStudio session titled 'stli' containing R code. The code is color-coded for syntax highlighting. A browser window titled 'Heart FM Online | Int...' is visible at the top right. The taskbar at the bottom shows icons for File Explorer, Task View, and other applications.

```
32 cat(cli::bg_red("Hello world - audit data analytics in R\n"))
33
34 library(data.table)
35
36 # Part 1 - Read
37 df <- fread(cmd = "grep -v Merc 'C:/Users/Stewart Li/Desktop/tbd/a.csv'", 
38             select = 2:13,
39             colClasses = list(character = c("car"), numeric = 3:13))
40
41 dim(df)
42 glimpse(df)
43 df[sample(1:nrow(df), 3, replace = TRUE)]■ Trailing whitespace is superfluous.
44
45 # Part 2 - Count
46 df[, .N, .by = gear]
47 df[, summpg > mean(mpq), .by = gear]
48
49
50 # Part 3 - Summarize
51 df[, .SD[, by = .by, gear]]
52 df[, lapply(.SD, max), .by = gear]
53
54 with(df, tapply(mpq, list(am, gear), max, default = 0))
55 with(df, by(mpq, c(gear), summary))
56 aggregate(mpq ~ am + gear, data = df, FUN = median, subset = df$hp > 150)
57
58 # Part 4 - Reshape
59 melt(df, id = c("am", "gear"), measure = c("mpq", "hp"))
60 dcast(df, am + gear ~ carb, value.var = "mpq", fun.aggregate = list(min, mean, max), fill = 0)■ Lines should not be
61
62 # Part 5 - Filter
63 df[between(mpq, 25, 30)]
64 df[mpg > 20 & hp < 100, ]
65
66 # Part 6 - Mutate
67 df$created_date := seq(as.Date("2021-12-1"), by = "month", length.out = nrow(df))■ Lines should not be
68 df[[1]] = paste0(mpq %/% 2) * 2, new_mpg_hp = mpg/hp)■ Put spaces around all infix operators.
69 df[, c("var1", "var2") := lstrsplit(displ, "_", fixed = TRUE, fill = 0)]■ Trailing whitespace is superfluous.
70 df[, (mpg, new_mpg_if = case_when(mpg < 15 ~ "small", mpg > 30 ~ "large", default = "medium"))]■ Lines should not be
71
72 cat(cli::bg_red("Hello world - audit data analytics in Python\n"))
73
74 library(data.table)
75
76 # Part 1 - Read
77 df <- fread(cmd = "grep -v Merc 'C:/Users/Stewart Li/Desktop/tbd/a.csv'", 
78             select = 2:13,
79             colClasses = list(character = c("car"), numeric = 3:13))
80
81 dim(df)
82 glimpse(df)
83 df[sample(1:nrow(df), 3, replace = TRUE)]■ Trailing whitespace is superfluous.
84
85 # Part 2 - Count
86 df[, .N, .by = gear]
87 df[, sum(mpq + mean(mpq)), .by = gear]
88
89 # Part 3 - Summarize
90 df[, .SD[, by = .by, gear]]
91 df[, lapply(.SD, max), .by = gear]
92
93 with(df, tapply(mpq, list(am, gear), max, default = 0))
94 with(df, by(mpq, c(gear), summary))
95 aggregate(mpq ~ am + gear, data = df, FUN = median, subset = df$hp > 150)
96
97 # Part 4 - Reshape
98 melt(df, id = c("am", "gear"), measure = c("mpq", "hp"))
99 dcast(df, am + gear ~ carb, value.var = "mpq", fun.aggregate = list(min, mean, max), fill = 0)■ Lines should not be
100
101 # Part 5 - Filter
102 df[between(mpq, 25, 30)]
103 df[mpg > 20 & hp < 100, ]
104
105 # Part 6 - Mutate
106 df$created_date := seq(as.Date("2021-12-1"), by = "month", length.out = nrow(df))■ Lines should not be
107 df[[1]] = paste0(mpq %/% 2) * 2, new_mpg_hp = mpg/hp)■ Put spaces around all infix operators.
108 df[, c("var1", "var2") := lstrsplit(displ, "_", fixed = TRUE, fill = 0)]■ Trailing whitespace is superfluous.
109 df[, (mpg, new_mpg_if = case_when(mpg < 15 ~ "small", mpg > 30 ~ "large", default = "medium"))]■ Lines should not be
```

Figure 7.2: 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))

```

## 7.2 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

```

## 7.3 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))

```

```

audit.R
# cross check payroll amount against finance amount
df_clean %>%
  group_by(staff_id, staff_name) %>%
  summarise(exp = sum(amount_s.x),
            ant_id = sum(amount_s.y) / n(),
            ant_diff = exp - ant_id,
            .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 = c(claim_date, left = vars(expense_date), right = vars(last_date))) %>%
  mutate()
```
# creates agent
create_agent(label = "A very +simple+ example.", tbl_name = "payroll") %>%
  col_vals_between(columns = c(claim_date, left = vars(expense_date), right = vars(last_date))) %>%
  mutate()
```
# produces audit working paper
library(pointblank)
```

```

Figure 7.3: Audit Procedure 1

```

audit.R
# cross check payroll amount against finance amount
df_clean %>%
  group_by(staff_id, staff_name, purpose, amount_s.x) %>%
  summarise(exp = sum(amount_s.x),
            ant_id = sum(amount_s.y) / n(),
            ant_diff = exp - ant_id,
            .groups = 'drop')
```
# 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 = c(claim_date, left = vars(expense_date), right = vars(last_date))) %>%
  mutate()
```
# creates agent
create_agent(label = "A very +simple+ example.", tbl_name = "payroll") %>%
  col_vals_between(columns = c(claim_date, left = vars(expense_date), right = vars(last_date))) %>%
  mutate()
```
# produces audit working paper
library(pointblank)
```

```

Figure 7.4: 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>.