

Data Analytics Engineering

For Accountants and Auditors

Stewart Li

2023-11-19

Table of contents

Preface	3
I Infrastructure	4
1 Local	6
2 ELT	20
II Data tools	21
3 Polars	25
4 Analysis	30
4.1 IO	30
4.2 Cleaning	31
4.3 Validate	31
4.4 Munging	32
4.5 EDA	33
4.6 Model	35
4.7 Report	36
References	38

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](#) and [financial statement preparation](#), 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 Windows Terminal, WSL2, and Linux distribution systems. Edit terminal theme/font, dotfiles of Bash/Tmux/Vim, and env variables. Install Git/GitBash and Docker/Podman if needed.

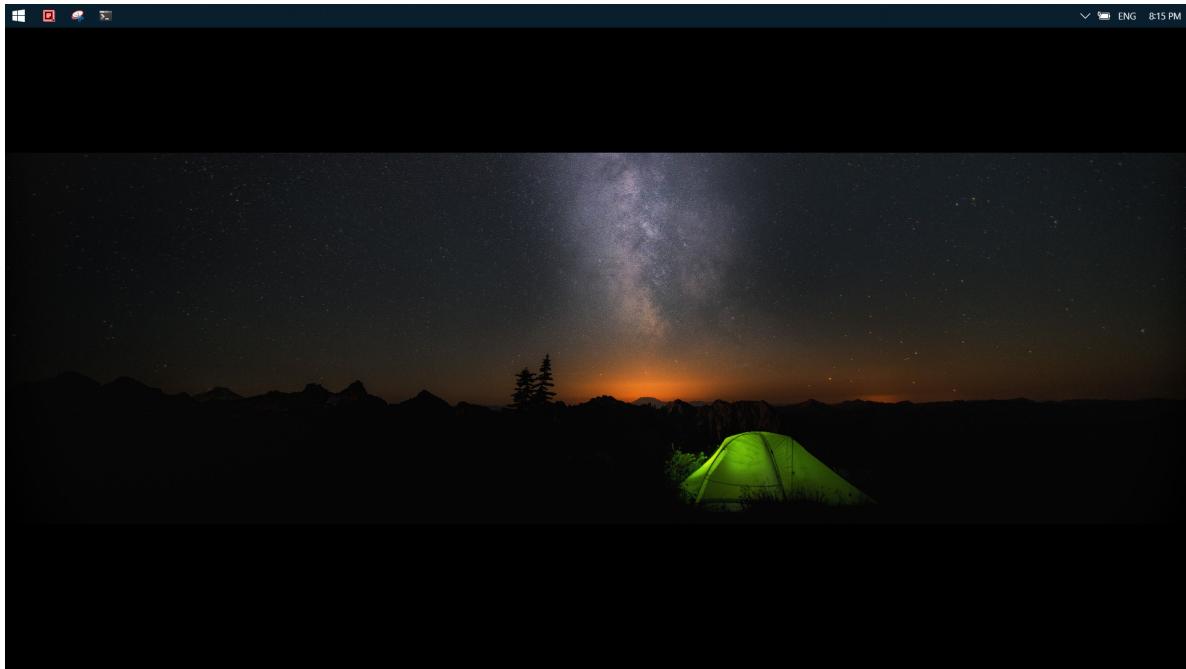


Figure 1.1: Desktop

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 manage `pass`, `rsync` files, `quarto` markdown, `sftp` to server, `ssh` into remote machine, 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))
Input Locale: en-US (English (United States))
Time Zone: (UTC+08:00) Kuala Lumpur, Singapore
Total Physical Memory: 8,077 MB
Available Physical Memory: 1,293 MB
Virtual Memory: Max Size: 14,733 MB
Virtual Memory: Available: 5,154 MB
Virtual Memory: In Use: 9,579 MB
Page File Location(s): C:\pagefile.sys
Domain: WORKGROUP
Logon Server: \DESKTOP-HCEU07A
Hotfix(s): 27 Hotfix(s) Installed.
[001]: KB5029932
[002]: KB5062430
[003]: KB5062525
[004]: KB5089481
[005]: KB5003791
[006]: KB5012170
[007]: KB5015684
[008]: KB5030211
[009]: KB5006753

```

Figure 1.2: CMD

```

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

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

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

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

```

Figure 1.3: PowerShell

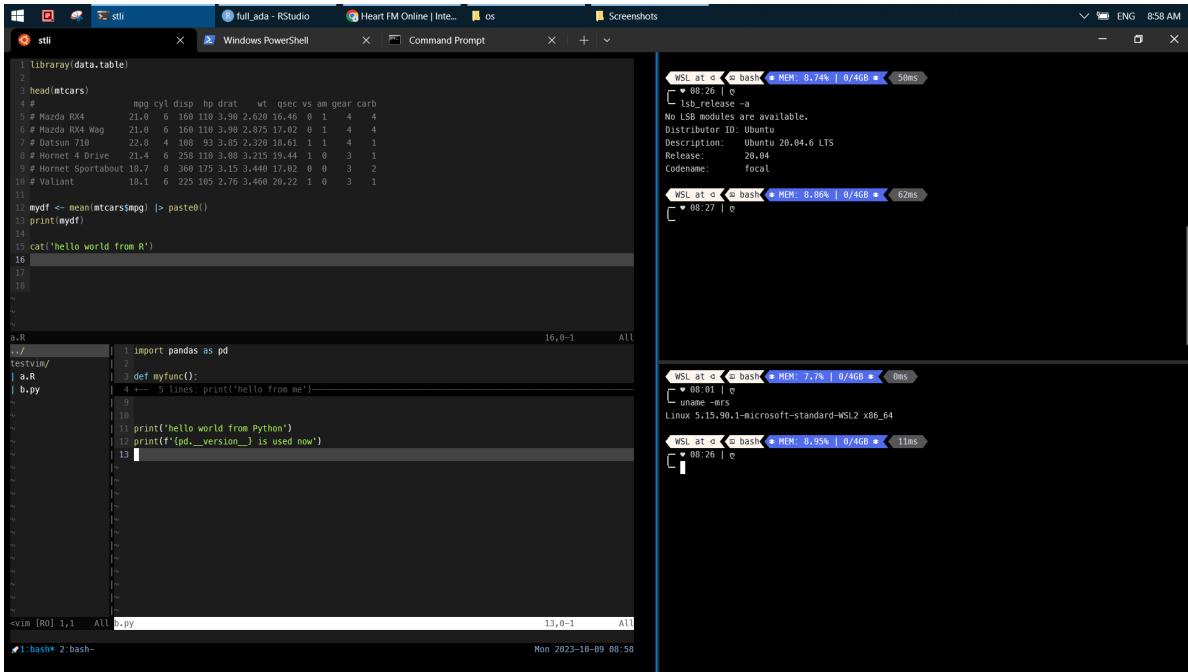


Figure 1.4: Ubuntu

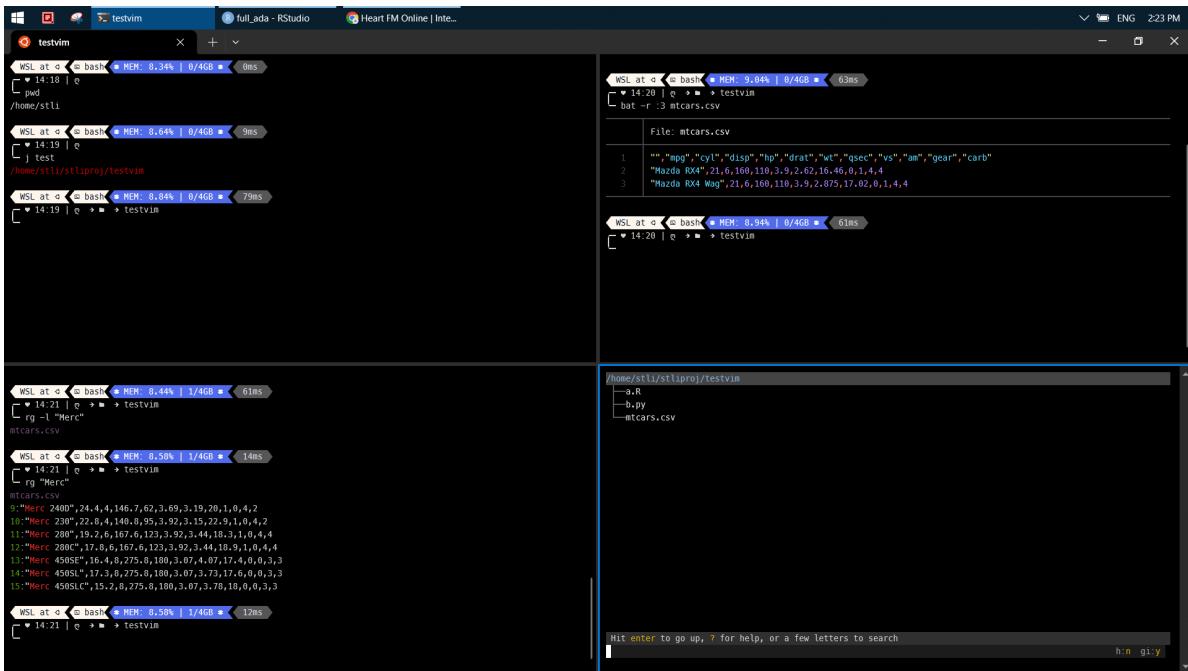


Figure 1.5: Terminal tools

```

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

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

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

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

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

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

WSL at < bash * MEM: 10.0% | 1/4GB * 9ms
└─ 15:29 | e ↵ ➤ testvim
  python3
Python 3.8.10 (default, May 26 2023, 14:05:08)
[GCC 9.4.0] on linux
Type "help()", "copyright", "credits" or "license" for more information.
>>> import pandas as pd
>>> df=pd.read_csv("mtcars.csv")
>>> df.describe()
   mpg    cyl   disp     vs     am   gear   carb
count 32.000000 32.000000 32.000000 ... 32.000000 32.000000 32.000000
mean 20.090625 6.187500 230.721875 146.687500 ... 0.437500 4.062500 3.667500 2.8125
std 6.626948 1.785922 123.330694 56.562864 ... 0.538016 0.498951 0.737805 1.6152
min 12.000000 4.000000 71.0 52.0 0.0 1.0 1.0 1.0
max 26.000000 8.000000 429.000000 95.500000 ... 8.000000 8.000000 3.000000 8.000000
50% 19.000000 6.000000 156.000000 123.000000 ... 0.000000 0.000000 4.000000 4.000000
75% 22.000000 8.000000 326.000000 180.000000 ... 1.000000 1.000000 4.000000 4.000000
max 33.900000 8.000000 472.000000 355.000000 ... 1.000000 1.000000 5.000000 8.000000

[8 rows x 11 columns]
>>>

```

Figure 1.6: R, Python, DuckDB

```

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

```

Figure 1.7: Vim - R

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

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

Figure 1.8: Vim - Python

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

- Neo-tree:** A file browser showing the directory structure of a Rust project. It includes files like `Cargo.toml` and `main.rs` under the `src` folder.
- stl:** A terminal window showing the output of the command `stl`.
- Cpp:** A code editor showing a C++ file named `a.cpp` with imports and function definitions.

The bottom of the screen features a status bar with the following information:
NORMAL > master Cargo.toml gk < ⌂ 2 ⌂ 9 88% 8:6 ① 19:36 NORMAL > master Cpp
NORMAL > master Cpp gk < ⌂ 2 ⌂ 9 88% 8:6 ① 19:36 Bot 7-1 ⌂ 19:36 Sat 2023-11-18 19:36

Figure 1.9: Tmux - Nvim 1

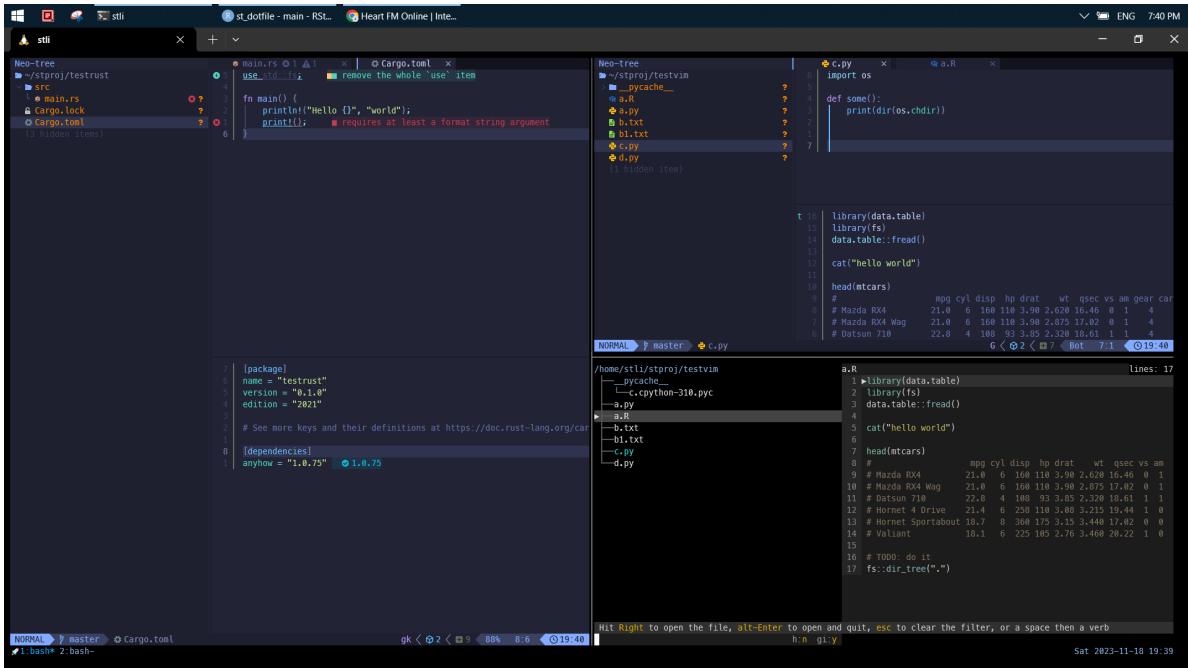


Figure 1.10: Tmux -Nvim 2

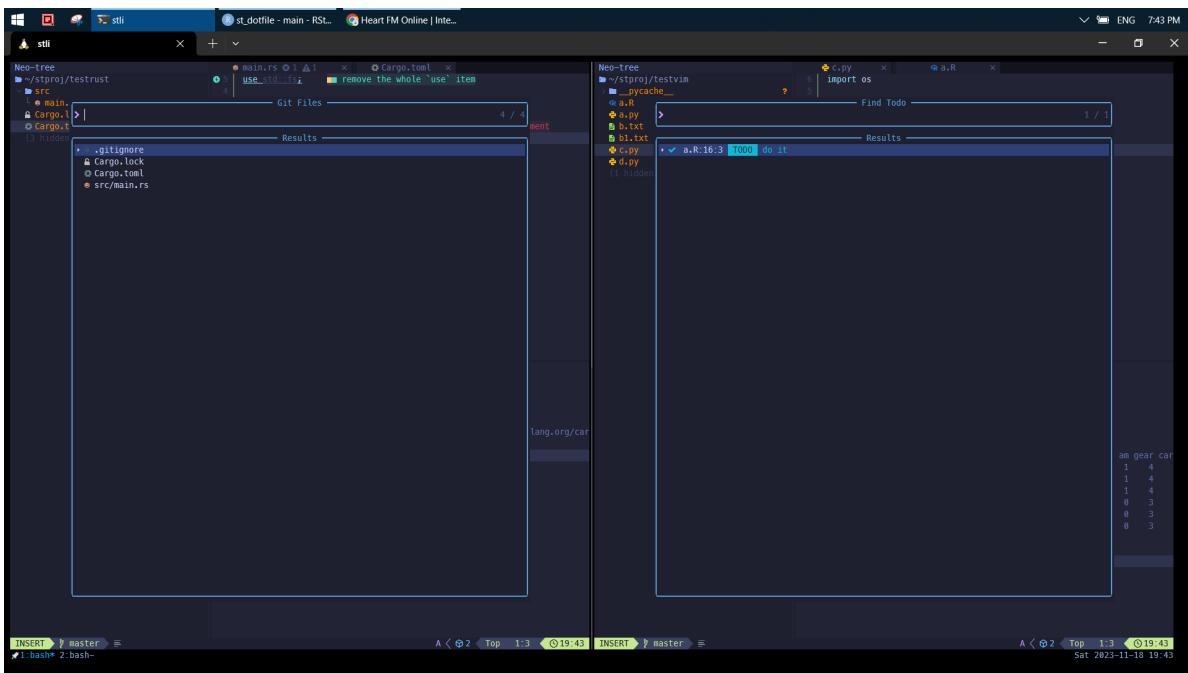


Figure 1.11: Tmux - Nvim 3

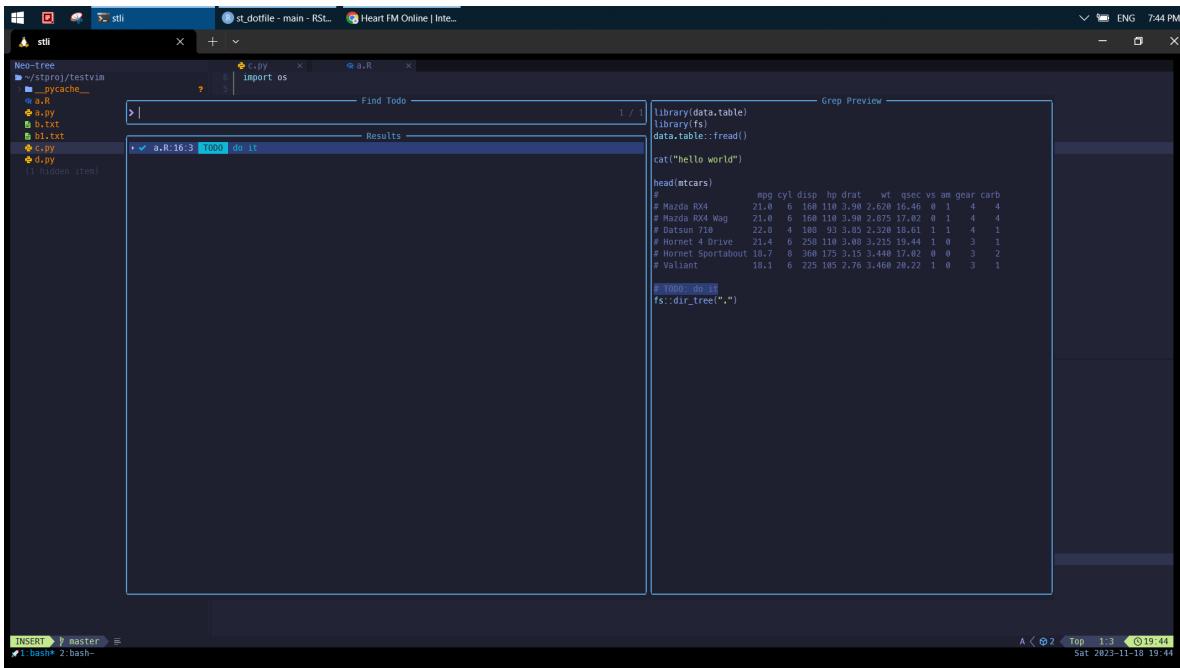


Figure 1.12: Tmux -Nvim 4

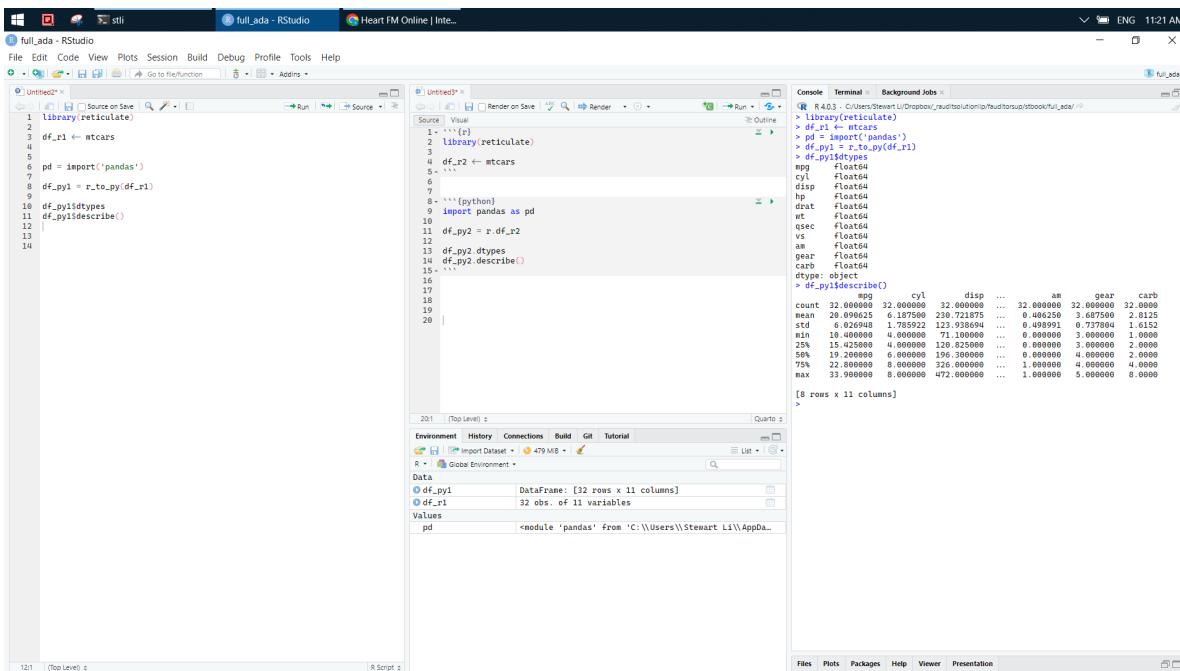


Figure 1.13: RStudio - R

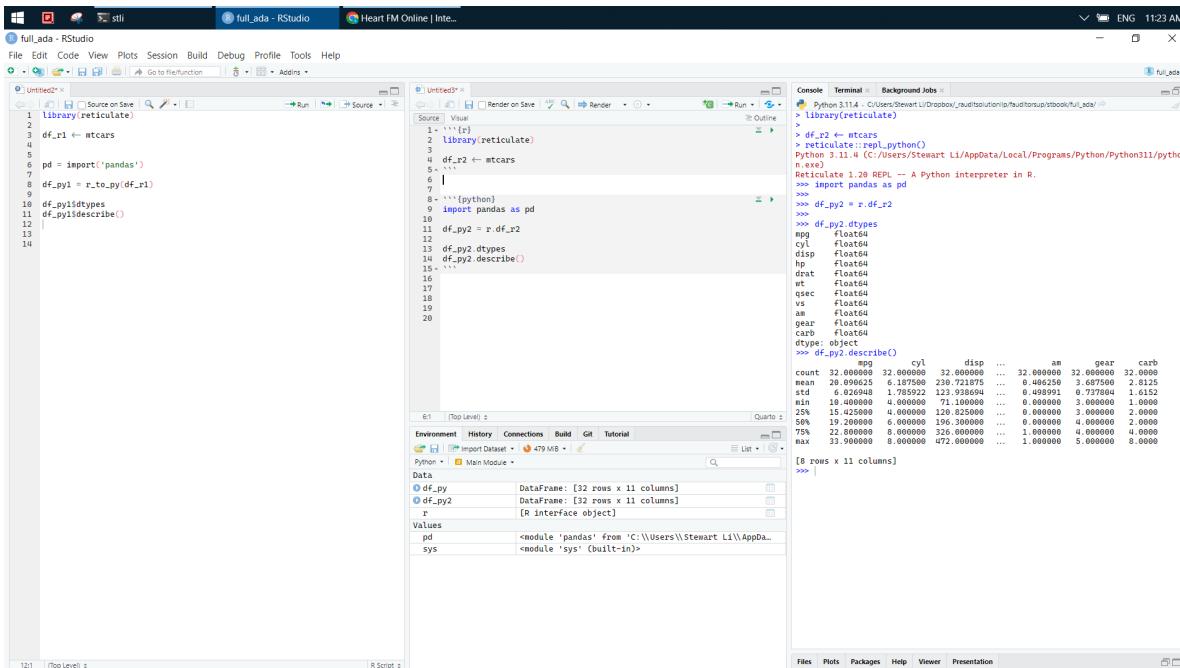


Figure 1.14: RStudio - Python

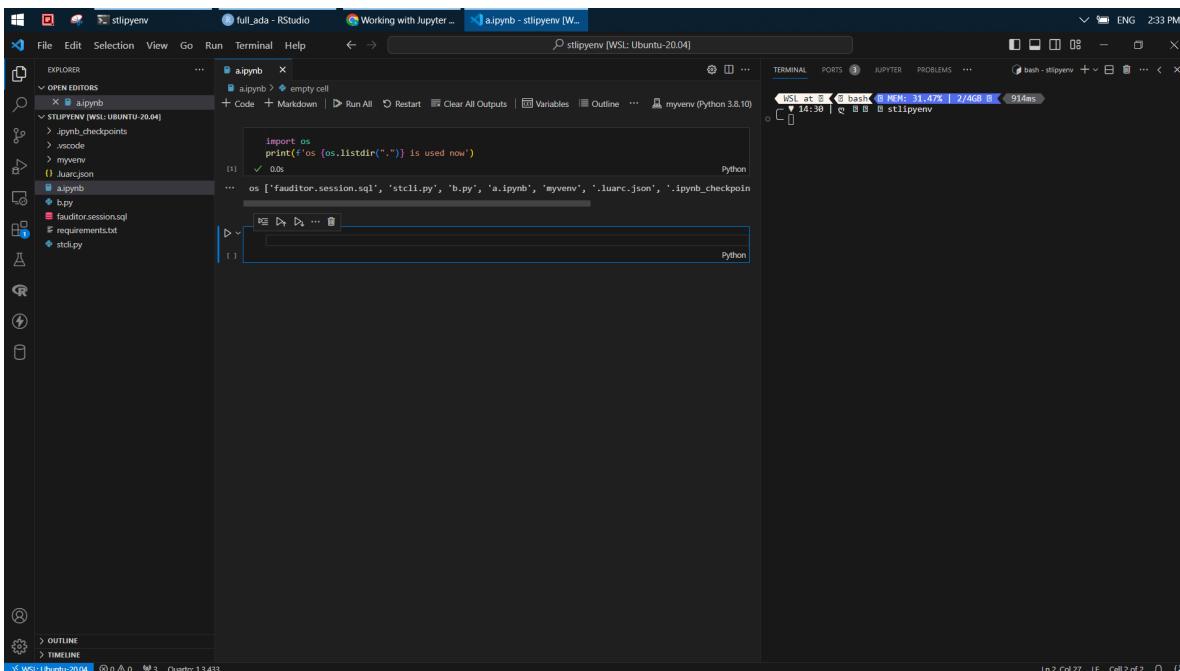


Figure 1.15: VS Code in Linux - Jupyter

A screenshot of the Visual Studio Code interface running on a Linux system. The title bar shows multiple tabs: 'full_ada - RStudio', 'Heart FM Online | Inte...', 'Interactive - b.py - stlipyenv', and 'stlipyenv [WSL: UBUNTU-20.04]'. The main area displays an 'EXPLORER' sidebar with files like 'b.py', 'a.json', and 'requirements.txt'. The 'Interactive' tab is active, showing a Python code editor with the following script:

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

The output pane below shows the results of running the script:

```
Connected to myenv (Python 3.8.10)\n✓ import pandas as pd...\n... pandas 2.0.2 is used now\n\n1+1\n✓ 0.05\n...\n2
```

At the bottom, a terminal window shows the command 'python b.py' being run.

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 displays an 'EXPLORER' sidebar with files like 'a.R', 'b.py', and 'r'. The 'TERMINAL' tab is active, showing the command 'python b.py' being run in a terminal window:

```
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\n$
```

Figure 1.17: VS Code in Windows - Script

A screenshot of the Visual Studio Code interface on Windows. The title bar shows multiple tabs: 'full_ada - RStudio', 'Heart FM Online | Inte...', 'b.py - testvim - Visual...', and 'testvim'. The left sidebar has an 'EXPLORER' view with files 'a.R' and 'b.py' listed under 'TESTVIM'. The main area has two code editors: one with 'b.py' containing Python code and another with 'pandas' imports. To the right is an 'Interactive' cell window titled 'Interactive - b.py X'. It contains a command-line interface with the output of running the code. At the bottom, there's a status bar with 'R (not attached) Ln3, Col1 Spaces:4 UTF-8 CRLF Python 3.11.4 64-bit Go Live Prettier'.

Figure 1.18: VS Code in Windows - Interactive cell

A screenshot of the Visual Studio Code interface on Windows. The title bar shows multiple tabs: 'full_ada - RStudio', 'Heart FM Online | Inte...', 'a.R - testvim - Visual...', and 'R Graphics: Device 2 (ACTIVE)'. The left sidebar has an 'EXPLORER' view with files 'a.R' and 'b.py' listed under 'TESTVIM'. The main area shows an R session with code like 'library(dplyr)', 'head(mtcars)', and 'plot(mtcars)'. Below the session is a 'TERMINAL' tab showing the output of 'head(mtcars)' which lists car models and their attributes. To the right is an 'R Graphics: Device 2 (ACTIVE)' window displaying a correlation matrix plot for the mtcars dataset, showing a grid of scatter plots for each variable against every other variable.

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.

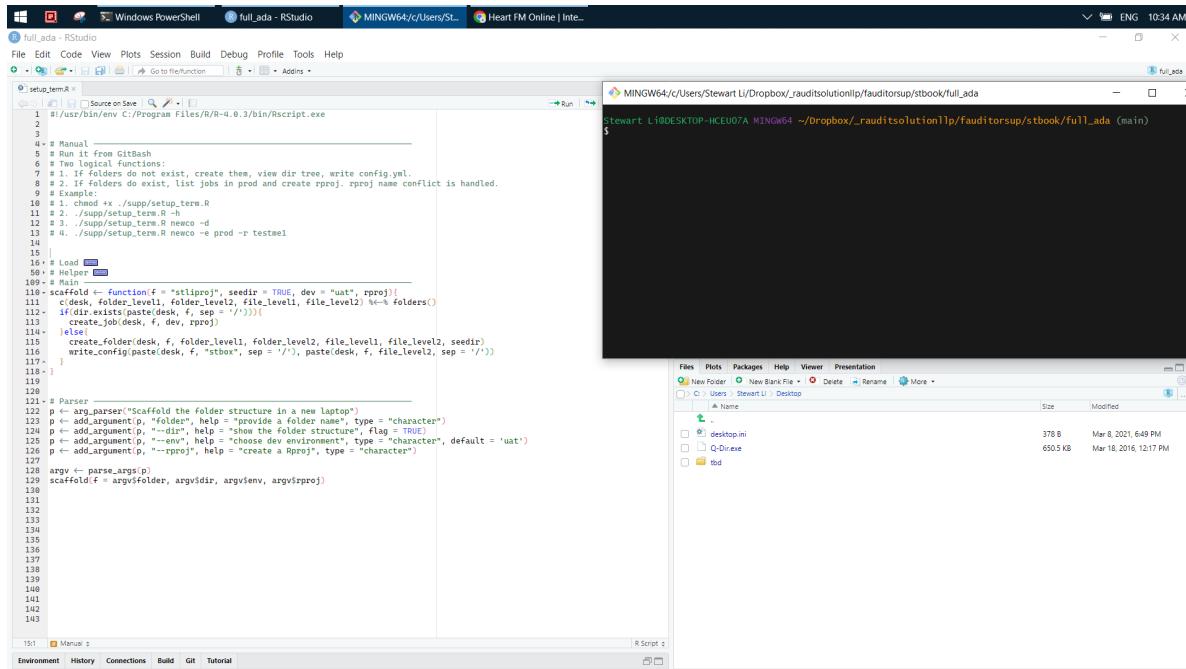


Figure 1.20: CLI R - GitBash 1

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`. It also includes command-line argument parsing for options like `-h` (help), `-d` (directory), `-e` (environment), and `-r` (R project). Below the code editor, the RStudio environment tab shows tabs for Environment, History, Connections, Build, Git, and Tutorial.

Windows PowerShell full_ada - RStudio MINGW64/c/Users/Stewart Li/Desktop/rauditst... Google Chrome - Full Ada - RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to function Addins

setup_term.R

Source on Save

1 #!/usr/bin/env C:/Program Files/R/R-4.0.3/bin/Rscript.exe

2

3 # Manual

4 # Run this from GitBash

5 # Two logical functions:

6 # 1. If folders do not exist, create them, view dir tree, write config.yml

7 # 2. If folders do exist, list jobs in prod and create rproj. rproj name conflict is handled.

8

9 # Functions

10 # 1. chmod +x ./setup_term.R

11 # 2. ./setup_term.R -h

12 # 3. ./setup_term.R newco -e prod

13 # 4. ./setup_term.R newco -e prod -r testmvel

14

15 |

16 | Load [R](#)

17 |

18 # Helper [functions](#)

19# [set_if_not_exist](#) = function(f = "stlproj", seedir = TRUE, dev = "uat", rproj){

20# c(desk, folder.level1, folder.level2, file.level1, file.level2) <- %> folders()

21# if(dir.exists(paste(desk, f, sep = '/'))){

22# create_job(desk, f, dev, rproj)

23# }

24# create.folder(desk, f, folder.level1, folder.level2, file.level1, file.level2, seedir)

25# write_config(paste(desk, f, "tbox", sep = '/'), paste(desk, f, file.level2, sep = '/'))

26# }

27|

28|

29# Parser

30# [arg_parser](#)"Scaffold the folder structure in a new laptop"

31# p <- add_argument(p, "folder", help = "path to provide a folder name", type = "character")

32# p <- add_argument(p, "dev", help = "choose the dev environment", type = "character")

33# p <- add_argument(p, "rproj", help = "choose a Rproj", type = "character")

34# p <- add_argument(p, "--env", help = "choose dev environment", type = "character", default = 'uat')

35# p <- add_argument(p, "--rproj", help = "create a Rproj", type = "character")

36#

37#

38 args <- parse_args(p)

39 scaffold(f = args\$folder, args\$dir, args\$env, args\$rproj)

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273

274

275

276

277

278

279

280

281

282

283

284

285

286

287

288

289

290

291

292

293

294

295

296

297

298

299

300

301

302

303

304

305

306

307

308

309

310

311

312

313

314

315

316

317

318

319

320

321

322

323

324

325

326

327

328

329

330

331

332

333

334

335

336

337

338

339

340

341

342

343

344

345

346

347

348

349

350

351

352

353

354

355

356

357

358

359

360

361

362

363

364

365

366

367

368

369

370

371

372

373

374

375

376

377

378

379

380

381

382

383

384

385

386

387

388

389

390

391

392

393

394

395

396

397

398

399

400

401

402

403

404

405

406

407

408

409

410

411

412

413

414

415

416

417

418

419

420

421

422

423

424

425

426

427

428

429

430

431

432

433

434

435

436

437

438

439

440

441

442

443

444

445

446

447

448

449

450

451

452

453

454

455

456

457

458

459

460

461

462

463

464

465

466

467

468

469

470

471

472

473

474

475

476

477

478

479

480

481

482

483

484

485

486

487

488

489

490

491

492

493

494

495

496

497

498

499

500

501

502

503

504

505

506

507

508

509

510

511

512

513

514

515

516

517

518

519

520

521

522

523

524

525

526

527

528

529

530

531

532

533

534

535

536

537

538

539

540

541

542

543

544

545

546

547

548

549

550

551

552

553

554

555

556

557

558

559

560

561

562

563

564

565

566

567

568

569

570

571

572

573

574

575

576

577

578

579

580

581

582

583

584

585

586

587

588

589

590

591

592

593

594

595

596

597

598

599

600

601

602

603

604

605

606

607

608

609

610

611

612

613

614

615

616

617

618

619

620

621

622

623

624

625

626

627

628

629

630

631

632

633

634

635

636

637

638

639

640

641

642

643

644

645

646

647

648

649

650

651

652

653

654

655

656

657

658

659

660

661

662

663

664

665

666

667

668

669

670

671

672

673

674

675

676

677

678

679

680

681

682

683

684

685

686

687

688

689

690

691

692

693

694

695

696

697

698

699

700

701

702

703

704

705

706

707

708

709

710

711

712

713

714

715

716

717

718

719

720

721

722

723

724

725

726

727

728

729

730

731

732

733

734

735

736

737

738

739

740

741

742

743

744

745

746

747

748

749

750

751

752

753

754

755

756

757

758

759

760

761

762

763

764

765

766

767

768

769

770

771

772

773

774

775

776

777

778

779

780

781

782

783

784

785

786

787

788

789

790

791

792

793

794

795

796

797

798

799

800

801

802

803

804

805

806

807

808

809

810

811

812

813

814

815

816

817

818

819

820

821

822

823

824

825

826

827

828

829

830

831

832

833

834

835

836

837

838

839

840

841

842

843

844

845

846

847

848

849

850

851

852

853

854

855

856

857

858

859

860

861

862

863

864

865

866

867

868

869

870

871

872

873

874

875

876

877

878

879

880

881

882

883

884

885

886

887

888

889

890

891

892

893

894

895

896

897

898

899

900

901

902

903

904

905

906

907

908

909

910

911

912

913

914

915

916

917

918

919

920

921

922

923

924

925

926

927

928

929

930

931

932

933

934

935

936

937

938

939

940

941

942

943

944

945

946

947

948

949

950

951

952

953

954

955

956

957

958

959

960

961

962

963

964

965

966

967

968

969

970

971

972

973

974

975

976

977

978

979

980

981

982

983

984

985

986

987

988

989

990

991

992

993

994

995

996

997

998

999

1000

1001

1002

1003

1004

1005

1006

1007

1008

1009

1010

1011

1012

1013

1014

1015

1016

1017

1018

1019

1020

1021

1022

1023

1024

1025

1026

1027

1028

1029

1030

1031

1032

1033

1034

1035

1036

1037

1038

1039

1040

1041

1042

1043

1044

1045

1046

1047

1048

1049

1050

1051

1052

1053

1054

1055

1056

1057

1058

1059

1060

1061

1062

1063

1064

1065

1066

1067

1068

1069

1070

1071

1072

1073

1074

1075

1076

1077

1078

1079

1080

1081

1082

1083

1084

1085

1086

1087

1088

1089

1090

1091

1092

1093

1094

1095

1096

1097

1098

1099

1100

1101

1102

1103

1104

1105

1106

1107

1108

1109

1110

1111

1112

1113

1114

1115

1116

1117

1118

1119

1120

1121

1122

1123

1124

1125

1126

1127

1128

1129

1130

1131

1132

1133

1134

1135

1136

1137

1138

1139

1140

1141

1142

1143

1144

1145

1146

1147

1148

1149

1150

1151

1152

1153

1154

1155

1156

1157

1158

1159

1160

1161

1162

1163

1164

1165

1166

1167

1168

1169

1170

1171

1172

1173

1174

1175

1176

1177

1178

1179

1180

1181

1182

1183

1184

1185

1186

1187

1188

1189

1190

1191

1192

1193

1194

1195

1196

1197

1198

1199

1200

1201

1202

1203

1204

1205

1206

1207

1208

1209

1210

1211

1212

1213

1214

1215

1216

1217

1218

1219

1220

1221

1222

1223

1224

1225

1226

1227

1228

1229

1230

1231

1232

1233

1234

1235

1236

1237

1238

1239

1240

1241

1242

1243

1244

1245

1246

1247

1248

1249

1250

1251

1252

1253

1254

1255

1256

1257

1258

1259

1260

1261

1262

1263

1264

1265

1266

1267

1268

1269

1270

1271

1272

1273

1274

1275

1276

1277

1278

1279

1280

1281

1282

1283

1284

1285

1286

1287

1288

1289

1290

1291

1292

1293

1294

1295

1296

1297

1298

1299

1300

1301

1302

1303

1304

1305

1306

1307

1308

1309

1310

1311

1312

1313

1314

1315

1316

1317

1318

1319

1320

1321

1322

1323

1324

1325

1326

1327

1328

1329

1330

1331

1332

1333

1334

1335

1336

1337

1338

1339

1340

1341

1342

1343

1344

1345

1346

1347

1348

1349

1350

1351

1352

1353

1354

1355

1356

1357

1358

1359

1360

1361

1362

1363

1364

1365

1366

1367

1368

1369

1370

1371

1372

1373

1374

1375

1376

1377

1378

1379

1380

1381

1382

1383

1384

1385

1386

1387

1388

1389

1390

1391

1392

1393

1394

1395

1396

1397

1398

1399

1400

1401

1402

1403

1404

1405

1406

1407

1408

1409

1410

1411

1412

1413

1414

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 directory named "testmel". A file browser window is also visible at the bottom, showing files like "ignore", "R", and "testme1.Rproj".

```

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

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

```

Figure 1.23: CLI R - GitBash 4

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

```

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

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

```

Figure 1.24: CLI R - GitBash 5

The screenshot shows the RStudio interface with an R script named `setup.R` open. The script contains code for 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`.

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, greate expectation, pydantic,

4. Data visualization¹:

- grid, patchwork, ggfx, ggtext, showtext,

- ragg, scales, formattable, sparkline,

- gghighlight, ggforce,

- imager, imagerExtra, ggimage, ggpibr,

- igraph, ggraph, tidygraph, networkD3, visNetwork,

- DiagrammeR, UpSetR, tmap,

5. Table:

- gt, gtExtras, gtsummary, modelsummary,

- flextable, kableExtra,

6. EDA:

- skimr, naniar, visdat, inspectdf,

7. Stats:

- corrplot, tidylo, widyr, broom,

8. Report:

- quarto, whisker, target, jinja2,

9. API deploy:

- vetiver, plumber, fastapi,

10. Dashboard:

- shiny, htmltools, htmlwidgets, crosstalk, leaflet,

- bslib, thematic, sass,

- DT, reactable, reactablefmtr,

- plotly, echarts4r, bokeh,

¹ggplot2 (Wickham 2016)

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

Consider other utility tools when necessary.

1. Environment:

- `rvenv`, `venv`,

2. Helper:

- `cli`, `crayon`,

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

3. Email:

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

4. Unzip:

- `archive`, `zipfile`,

5. FFI:

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

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

3 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 defines functions for filtering and summarizing data from a table named "finsample". The terminal window on the right shows the execution of the script and displays the resulting data frame. The data frame has columns: segment, country, product, cogs, profit, and date timestamp. It contains three rows of data for segments Government, Government, and Midmarket across countries Canada, Germany, and France, with products Carretera and various numerical values for cogs and profit.

```
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
```

Figure 3.1: CLI - argparse 1

```

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

```

Running in Ubuntu-20.04 (WSL 2)

WSL:Ubuntu-20.04 0 0 0 4 Quarto:1.3.433

In 42, Col 1 Spaces:4 UFF-8 ⓘ Python 3.8.10 (myvenv:venv) ⓘ

Figure 3.2: CLI - argparse 2

```

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

```

Running in Ubuntu-20.04 (WSL 2)

WSL:Ubuntu-20.04 0 0 0 4 Quarto:1.3.433

In 42, Col 1 Spaces:4 UFF-8 ⓘ Python 3.8.10 (myvenv:venv) ⓘ

Figure 3.3: CLI - argparse 3

```

stclick > ./stclick.py ...
59 @click.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.argument("output", type=click.File("w"), default="-", required=False)
79     @click.argument("highlight", type=click.Choice(["red", "green"]),
80                    help="highlight data based on the provided threshold",
81                )
82     def cond(ctx, output, highlight):
83         ...

84     @click.command()
85     @click.pass_context
86     @click.argument("output", type=click.File("w"), default="-", required=False)
87     @click.argument("highlight", type=click.Choice(["red", "green"]),
88                    help="highlight data based on the provided threshold",
89                )
90     def disp(ctx, output, highlight):
91         ...

92     @click.command()
93     @click.pass_context
94     @click.argument("output", type=click.File("w"), default="-", required=False)
95     @click.argument("highlight", type=click.Choice(["red", "green"]),
96                    help="highlight data based on the provided threshold",
97                )
98     def desc(ctx, output, highlight):
99         ...

100    @click.command()
101    @click.pass_context
102    @click.argument("output", type=click.File("w"), default="-", required=False)
103    @click.argument("highlight", type=click.Choice(["red", "green"]),
104                    help="highlight data based on the provided threshold",
105                )
106    def about(ctx, output, highlight):
107        ...

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

WSL at E: ~ basic [MEM: 27.4% | 2/4GB B] 191ms
python3 ./stclick/stclick.py --help
Usage: stclick.py [OPTIONS] RESV [COMMAND [ARGS]...]
      data validation
      Options:
        --version Show the version and exit.
        --help Show this message and exit.

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

Commands:
  condc choose your data
WSL at E: ~ basic [MEM: 28.3% | 2/4GB B] 196ms
python3 ./stclick/stclick.py "/home/stli/stliproj/testvim/mtcars.csv" condc --help
data source: /home/stli/stliproj/testvim/mtcars.csv
Usage: stclick.py INPUT COMMAND [OPTIONS] [ARGS]...
      choose your data
      Options:
        --highlight [red|green] highlight data based on the provided threshold
        --help Show this message and exit.
WSL at E: ~ basic [MEM: 27.4% | 2/4GB B] 196ms
python3 ./stclick/stclick.py "/home/stli/stliproj/testvim/mtcars.csv" stat topm disp 3
data source: /home/stli/stliproj/testvim/mtcars.csv
shape: (3, 12)
      mpg cyl disp  vs am gear carb
      -   -   -   -   -   -   -   -
      -   f64 164 f64 164 164 164
      Cadillac Fleetwood 10.4 8 472.0 0 0 3 4
      Lincoln Continental 10.4 8 460.0 0 0 3 4
      Chrysler Imperial 14.7 8 440.0 0 0 3 4
WSL at E: ~ basic [MEM: 27.5% | 2/4GB B] 169ms
python3 ./stclick/stclick.py "/home/stli/stliproj/testvim/mtcars.csv" stat desc
data source: /home/stli/stliproj/testvim/mtcars.csv
shape: (9, 6)
      describe mpmo disp drat wt qsec
      -   -   -   -   -   -
      -   f64 f64 f64 f64 f64
      count 32.0 32.0 32.0 32.0 32.0
      null_count 0.0 0.0 0.0 0.0 0.0
      mean 23.990625 230.721875 3.995625 3.99775 17.84875
      std 6.029548 113.93654 1.49779 0.578657 1.000000
      min 10.4 71.0 2.76 1.513 14.5
      max 33.9 472.0 4.93 5.424 22.9
      median 15.5 216.0 3.905 3.025 17.71
      25% 12.0 105.0 3.88 2.62 15.9
      75% 22.8 350.0 3.92 3.73 18.9
WSL at E: ~ basic [MEM: 27.3% | 2/4GB B] 249ms
python3 ./stclick/stclick.py "/home/stli/stliproj/testvim/mtcars.csv" stat desc
data source: /home/stli/stliproj/testvim/mtcars.csv
shape: (9, 6)
      describe mpmo disp drat wt qsec
      -   -   -   -   -   -
      -   f64 f64 f64 f64 f64
      count 32.0 32.0 32.0 32.0 32.0
      null_count 0.0 0.0 0.0 0.0 0.0
      mean 23.990625 230.721875 3.995625 3.99775 17.84875
      std 6.029548 113.93654 1.49779 0.578657 1.000000
      min 10.4 71.0 2.76 1.513 14.5
      max 33.9 472.0 4.93 5.424 22.9
      median 15.5 216.0 3.905 3.025 17.71
      25% 12.0 105.0 3.88 2.62 15.9
      75% 22.8 350.0 3.92 3.73 18.9

```

Figure 3.4: CLI - click 1

```

stclick > ./stclick.py ...
59 @click.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 cond(ctx, output, highlight):
84         ...

84     @click.command()
85     @click.pass_context
86     @click.argument("output", type=click.File("w"), default="-", required=False)
87     @click.argument("highlight", type=click.Choice(["red", "green"]),
88                    help="highlight data based on the provided threshold",
89                )
89     def disp(ctx, output, highlight):
90         ...

92     @click.command()
93     @click.pass_context
94     @click.argument("output", type=click.File("w"), default="-", required=False)
95     @click.argument("highlight", type=click.Choice(["red", "green"]),
96                    help="highlight data based on the provided threshold",
97                )
98     def desc(ctx, output, highlight):
99         ...

100    @click.command()
101    @click.pass_context
102    @click.argument("output", type=click.File("w"), default="-", required=False)
103    @click.argument("highlight", type=click.Choice(["red", "green"]),
104                    help="highlight data based on the provided threshold",
105                )
106    def about(ctx, output, highlight):
107        ...

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

WSL at E: ~ basic [MEM: 27.4% | 2/4GB B] 169ms
python3 ./stclick/stclick.py --help
Usage: stclick.py [OPTIONS] RESV [COMMAND [ARGS]...]
      data validation
      Options:
        --version Show the version and exit.
        --help Show this message and exit.

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

Commands:
  condc choose your data
WSL at E: ~ basic [MEM: 28.3% | 2/4GB B] 196ms
python3 ./stclick/stclick.py "/home/stli/stliproj/testvim/mtcars.csv" condc --help
data source: /home/stli/stliproj/testvim/mtcars.csv
Usage: stclick.py INPUT COMMAND [OPTIONS] [ARGS]...
      choose your data
      Options:
        --highlight [red|green] highlight data based on the provided threshold
        --help Show this message and exit.
WSL at E: ~ basic [MEM: 27.4% | 2/4GB B] 196ms
python3 ./stclick/stclick.py "/home/stli/stliproj/testvim/mtcars.csv" stat topm disp 3
data source: /home/stli/stliproj/testvim/mtcars.csv
shape: (3, 12)
      mpg cyl disp  vs am gear carb
      -   -   -   -   -   -   -   -
      -   f64 164 f64 164 164 164
      Cadillac Fleetwood 10.4 8 472.0 0 0 3 4
      Lincoln Continental 10.4 8 460.0 0 0 3 4
      Chrysler Imperial 14.7 8 440.0 0 0 3 4
WSL at E: ~ basic [MEM: 27.5% | 2/4GB B] 169ms
python3 ./stclick/stclick.py "/home/stli/stliproj/testvim/mtcars.csv" stat desc
data source: /home/stli/stliproj/testvim/mtcars.csv
shape: (9, 6)
      describe mpmo disp drat wt qsec
      -   -   -   -   -   -
      -   f64 f64 f64 f64 f64
      count 32.0 32.0 32.0 32.0 32.0
      null_count 0.0 0.0 0.0 0.0 0.0
      mean 23.990625 230.721875 3.995625 3.99775 17.84875
      std 6.029548 113.93654 1.49779 0.578657 1.000000
      min 10.4 71.0 2.76 1.513 14.5
      max 33.9 472.0 4.93 5.424 22.9
      median 15.5 216.0 3.905 3.025 17.71
      25% 12.0 105.0 3.88 2.62 15.9
      75% 22.8 350.0 3.92 3.73 18.9
WSL at E: ~ basic [MEM: 27.3% | 2/4GB B] 249ms
python3 ./stclick/stclick.py "/home/stli/stliproj/testvim/mtcars.csv" stat desc
data source: /home/stli/stliproj/testvim/mtcars.csv
shape: (9, 6)
      describe mpmo disp drat wt qsec
      -   -   -   -   -   -
      -   f64 f64 f64 f64 f64
      count 32.0 32.0 32.0 32.0 32.0
      null_count 0.0 0.0 0.0 0.0 0.0
      mean 23.990625 230.721875 3.995625 3.99775 17.84875
      std 6.029548 113.93654 1.49779 0.578657 1.000000
      min 10.4 71.0 2.76 1.513 14.5
      max 33.9 472.0 4.93 5.424 22.9
      median 15.5 216.0 3.905 3.025 17.71
      25% 12.0 105.0 3.88 2.62 15.9
      75% 22.8 350.0 3.92 3.73 18.9

```

Figure 3.5: CLI - click 2

```

WSL at stli [M: 27.0% | 2/4GB B] 249ms
python3 ./stclick/stclick.py "/home/stli/stliproj/testv1m/mtcars.csv" calc condc
data source: /home/stli/stliproj/testv1m/mtcars.csv
shape: (32, 13)
  mpg cyl disp  am gear carb new
  ...
str   f64 164 f64  i64 164 i64 str
Mazda RX4 21.0 6 160.0 - 1 4 4 549.0
Mazda RX4 Wag 21.0 6 160.0 - 1 4 4 549.0
Datsun 710 22.8 4 108.0 - 1 4 1 549.0
Hornet 4 Drive 21.4 6 258.0 - 0 3 1 549.0
...
Ford Pantera L 15.8 8 351.0 - 1 5 4 93.9
Ferrari Dino 19.7 6 145.0 - 1 5 6 549.0
Maserati Bora 15.0 8 301.0 - 1 5 8 93.9
Volvo 142E 21.4 4 121.0 - 1 4 2 549.0

WSL at stli [M: 26.9% | 1/4GB B] 189ms
python3 ./stclick/stclick.py "/home/stli/stliproj/testv1m/mtcars.csv" calc condc | grep Volvo
Volvo 142E 21.4 4 121.0 - 1 4 2 549.0

WSL at stli [M: 27.0% | 2/4GB B] 174ms
python3 ./stclick/stclick.py "/home/stli/stliproj/testv1m/mtcars.csv" calc condc | grep V
o

```

Figure 3.6: CLI - click 3

```

WSL at stli [M: 33.7% | 2/4GB B] 281ms
stpolars "/home/stli/stliproj/testv1m/mtcars.csv" about
data source: /home/stli/stliproj/testv1m/mtcars.csv
shape: (32, 13)

WSL at stli [M: 33.0% | 2/4GB B] 214ms
stpolars "/home/stli/stliproj/testv1m/mtcars.csv" calc condc
data source: /home/stli/stliproj/testv1m/mtcars.csv
shape: (32, 13)
  mpg cyl disp  am gear carb new
  ...
str   f64 164 f64  i64 164 i64 str
Mazda RX4 21.0 6 160.0 - 1 4 4 549.0
Mazda RX4 Wag 21.0 6 160.0 - 1 4 4 549.0
Datsun 710 22.8 4 108.0 - 1 4 1 549.0
Hornet 4 Drive 21.4 6 258.0 - 0 3 1 549.0
...
Ford Pantera L 15.8 8 351.0 - 1 5 4 93.9
Ferrari Dino 19.7 6 145.0 - 1 5 6 549.0
Maserati Bora 15.0 8 301.0 - 1 5 8 93.9
Volvo 142E 21.4 4 121.0 - 1 4 2 549.0

WSL at stli [M: 32.5% | 2/4GB B] 214ms

```

Figure 3.7: CLI - click 4

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

- File Explorer**: Shows a project structure for "rootproj.toml" under "TESTOMI (WSL: UBUNTU-20.04)".
- Terminal 1**: Running on "WSL: Ubuntu-20.04". It displays a PostgreSQL query:fauditor=# select * from client;
name | year | joblist | auditor | status
-----+-----+-----+-----+-----
(0 rows)

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

fauditor=#
- Terminal 2**: Running on "WSL: Ubuntu-20.04". It shows the output of a "cargo build" command:Compiling hame v0.5.5
Compiling hdkf v0.12.3
Compiling rustix v0.38.20
Compiling md-5 v0.10.6
Compiling tokio v0.24.0
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/still/stillpro/trust/testoml)
testoml initialized + (debugging) target(s) in 1m 27s
Running 'target/debug/testoml' as clientA_2023 + stewartli init'
Initializing in [/home/still/stillpro/trust/testoml]
- Terminal 3**: Running on "WSL: Ubuntu-20.04". It shows rust-analyzer diagnostics:WSL at E bash 8.0.0 M 6.1.2025 V 14.6.8 1m 27s 630ms E master w/ 34
16:45 8 8 8 8 testoml
R (not attached) Ln 6, Col 1 Spaces: 4 UF-8 LF TML no schema selected

Figure 3.8: 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, including "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", "cargo run -- -n clientA -y 2023 -a stewartl new p", and "cargo run -- -n clientA -y 2023 -a stewartl new p". The terminal also shows the creation of a "myenv" environment and the activation of "myvenv/bin/activate".

Terminal Session: A separate terminal window titled "awp.R - testtomi [WSL]" is also visible, showing similar cargo command outputs.

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 like "client.R", "client.rs", "config.rs", and "main.rs".

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

Figure 3.9: CLI - clap 2

4 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](#).

4.1 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)
```

4.2 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))
```

4.3 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))

```

4.4 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'))

```

4.5 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 %>%

```

The screenshot shows an RStudio session titled "full_ada - RStudio". The Editor tab contains R code for data munging, including grouping by country code, summarizing sales, and creating a new column for first disputed late. The Console tab displays a large data frame named "df_clean" with columns like "country_code", "n_sales", "first_disputed_late", and "largest_inv_amt". The data frame has 10 rows and 10 columns. The bottom of the console shows a summary of the data frame.

```

132 +
133 +   * [r]
134 +   df_clean %>%
135 +     group_by(country_code) %>%
136 +     arrange(invdate) %>%
137 +     summarise(
138 +       sales = sum(invoice_amount),
139 +       first_disputed_late = first(invdate[disputed == 'Yes']),
140 +       first_disputed_d1 = first(invdate[disputed == 'Yes']),
141 +       largest_late = max(late[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 +   * [r]
148 +   sort(unique(df_clean$late))
149 +
150 +   df_clean <- df_clean %>%
151 +     dplyr::filter(late != 0) %>%
152 +     mutate(reminder = case_when(late > 0 & late <= 10 ~ "1st email",
153 +                                 late > 10 & late <= 20 ~ "2nd email",
154 +                                 late > 20 & late <= 30 ~ "legal case",
155 +                                 TRUE ~ "bad debt"))
156 +
157 +   # anomaly by country
158 +   df_clean <- df_clean %>%
159 +     ggplot(aes(late, disputed, color = country_code)) +
160 +     geom_boxplot() +
161 +     theme_light()
162 +
163 +   # summary table
164 +   df_clean <- df_clean %>%
165 +     group_by(reminder, disputed) %>%
166 +     summarise(across(late, tibble::lst(sum, min, max, sd)), .groups = 'drop') %>%
167 +     gt::gt()
168 +
169 +   # clients without dispute do not pay.
170 +   df_clean %>%
171 +     dplyr::filter(disputed == 'No', reminder %in% c("legal case", "bad debt"))
172 +
173 +   ## EDA
174 +
175 +
176 +   # Focus on a handful of variables after dropped others.
177 +
178 +   * [r]
179 +   sort(unique(df_clean$late))
180 +
181 +   * [r]
182 +   df_clean %>%
183 +     select(where(is.numeric)) %>%
184 +
185 +     # normal distribution
186 +     df %>%
187 +       ggplot(aes(invoice_amount, fill = disputed)) +
188 +       geom_histogram(bins = 10, position = 'dodge') +
189 +       geom_vline(xintercept = median(df$invoice_amount), color = 'red',
190 +                  size = 3, linetype = "dashed") +
191 +       theme_light()
192 +
193 +     # correlation
194 +     df %>%
195 +       select(where(is.numeric)) %>%
196 +       cor() %>%
197 +       corrplot::corrplot(method = 'color', order = 'FPC', type = 'lower', diag = FALSE)
198 +
199 +     df %>%
200 +       select(where(is.numeric)) %>%

```

Figure 4.1: 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()
```

4.6 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()

```

4.7 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())

```

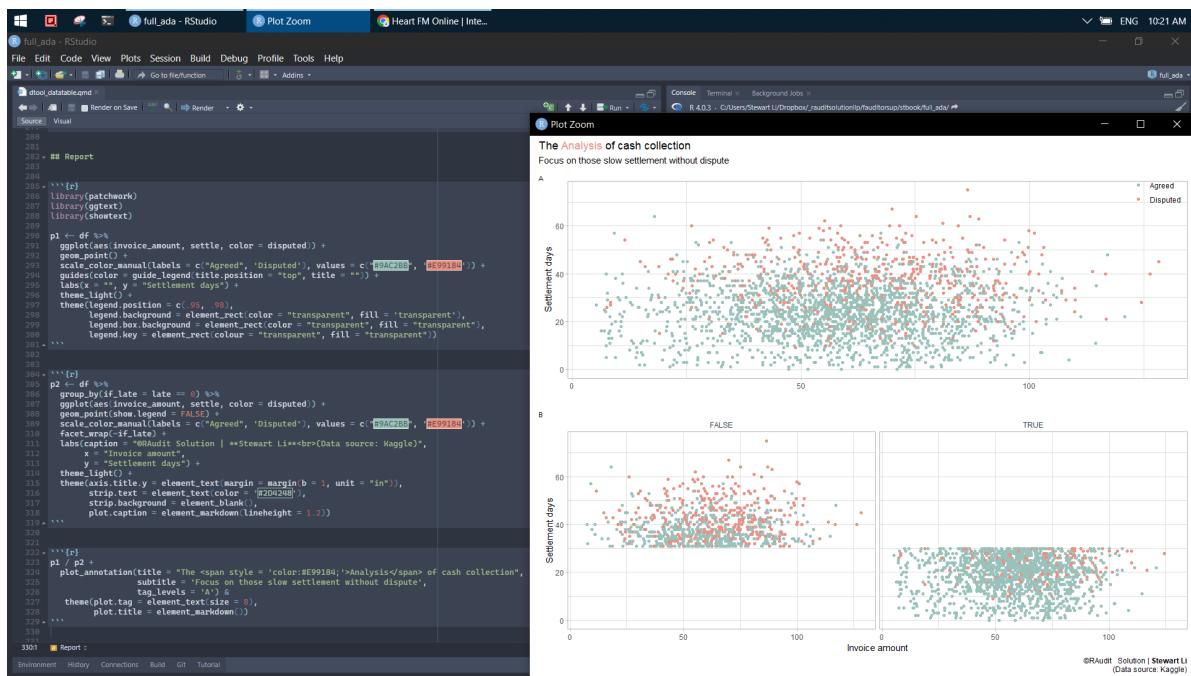


Figure 4.2: Combined plot

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>.