R at the Heart of VBZ Mobility Analytics

Samuel Wittwer

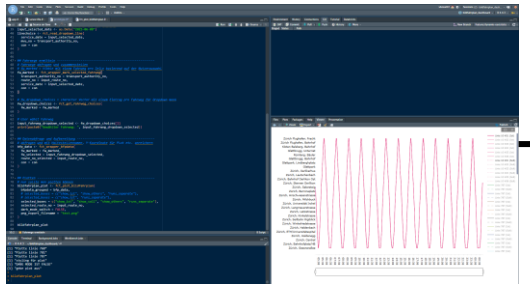
R Meetup Zurich
9.10.2025

VBZ Züri Linie
Umsteigen lohnt sich.

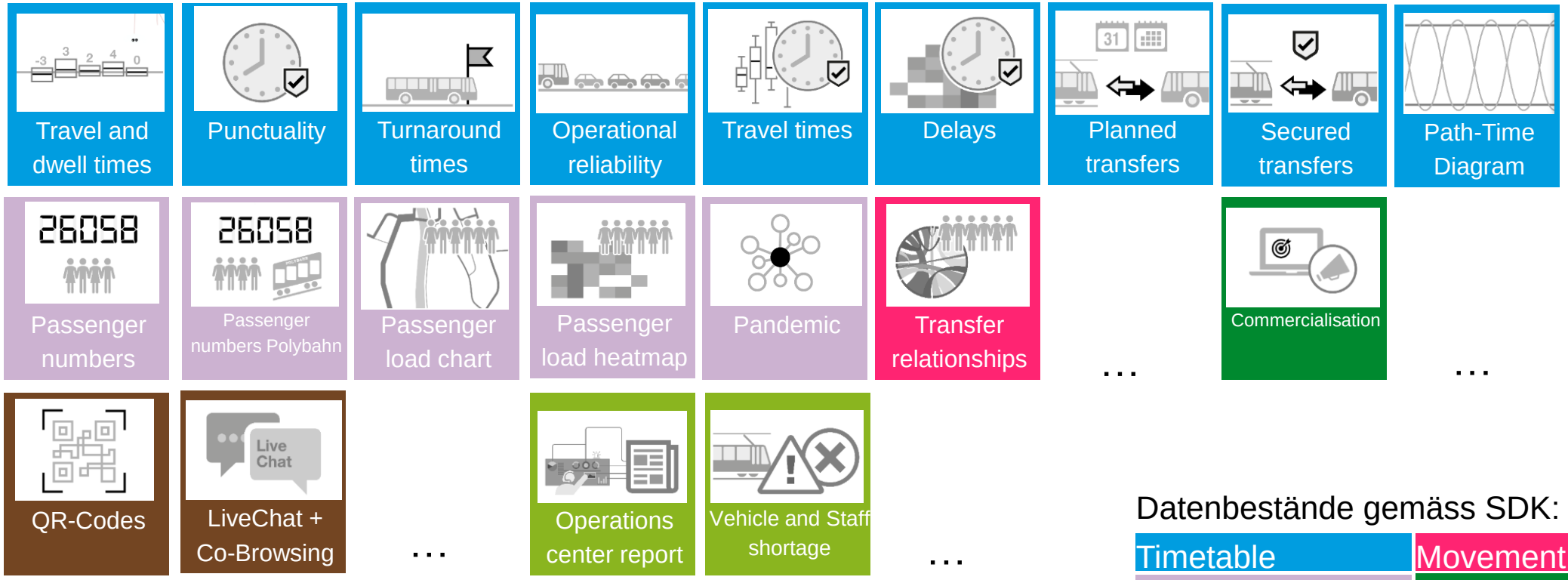
Ein Unternehmen der Stadt Zürich

Our Current Setup

- **Posit Workbench** for coding
  - **Renv** to manage individual packages per project
- **Gitlab** to host our repositories
- Posit Package Manager for our **R packages**
- **Azure Databricks** as our centralised data source
  - No direct connection between apps and other systems
- **Posit Connect** for automated deployment
  - Users navigate to individual apps from an Rmarkdown landing page



VBZ Analytics Products



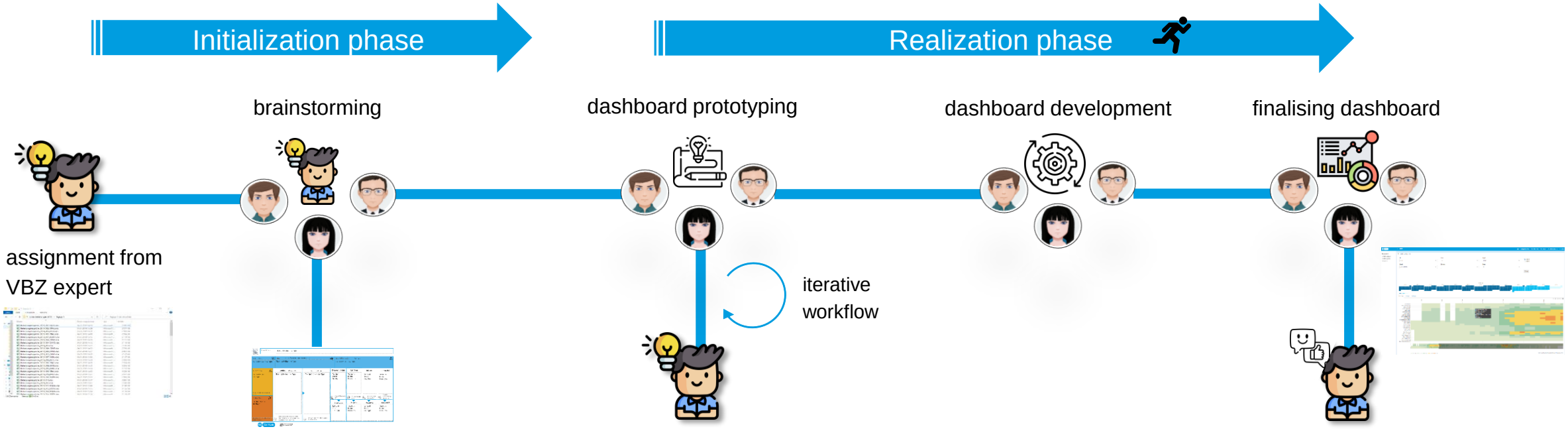
Datenbestände gemäss SDK:

- Timetable
- Movement data
- Passenger counts
- Vehicle
- Events data
- Passenger info

VBZ Züri Linie

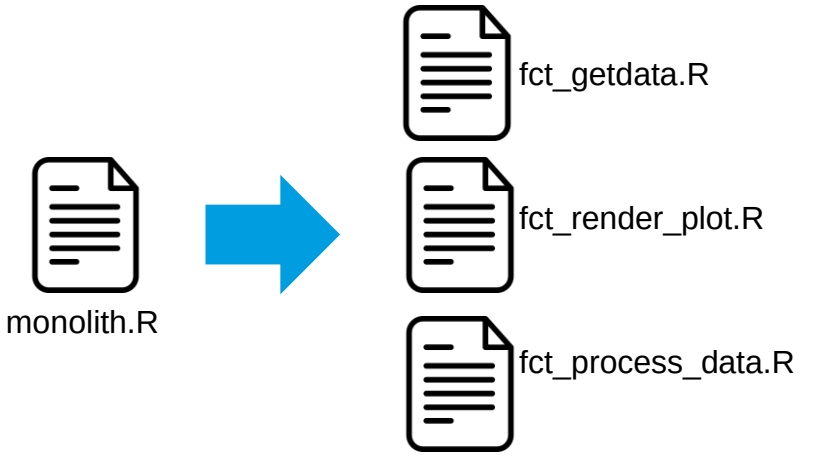
Ein Unternehmen der Stadt Zürich

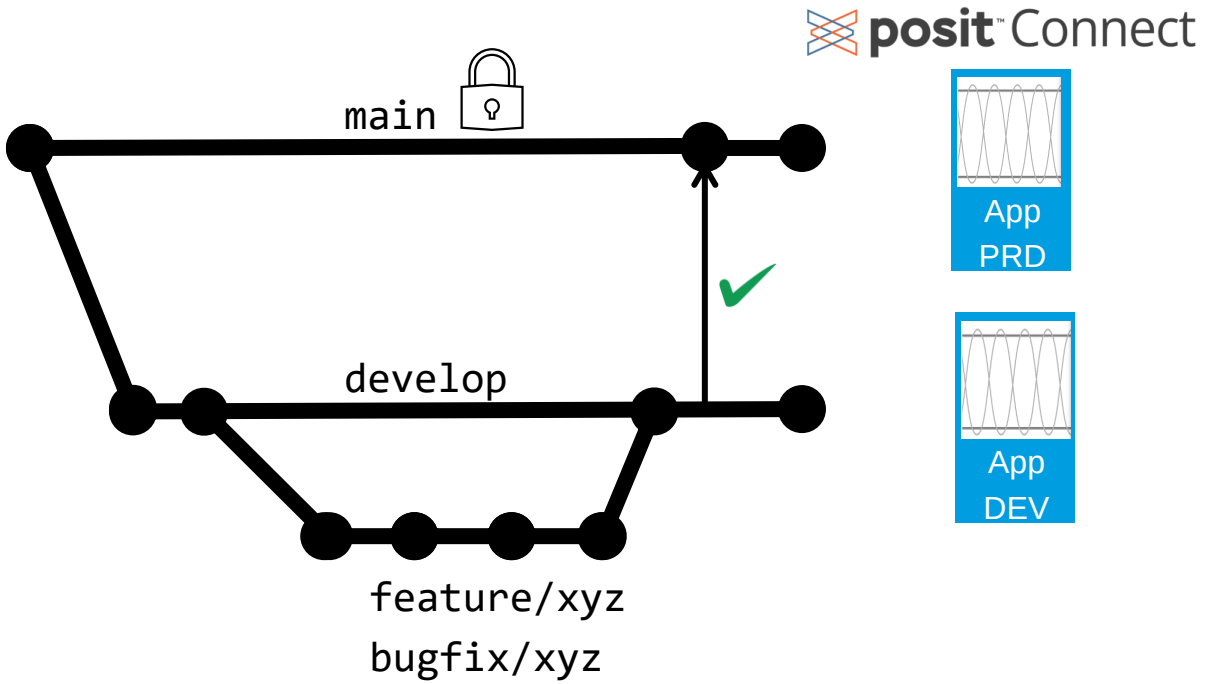
Dashboard development in collaboration with users



Our git workflow

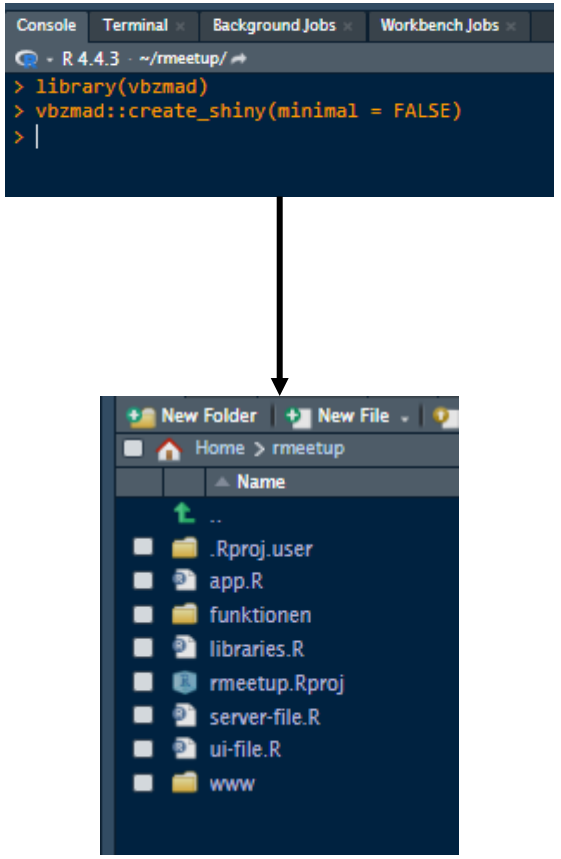
- Managing our code and the increasing number of dashboards has become more challenging
- Our measures:
  - Modularise our code (one function = one file)
  - Describe each function with a complete **Roxygen2** block
  - Adapted a "**git-flow**" branching strategy with code approval
  - Reduce code duplications through custom **R packages**
  - Started implementing unit tests for packages with **testthat**





Standardising our apps

- Template app, server, ui and helper files are now all bundled in our vbzmad package.
- Creating a new app becomes a one-liner!
- app.R
  - sources all necessary helper functions
  - fetches additional CSS style sheets for individual colour palettes depending on environmental variables
- server-file.R and ui-file.R contain a basic app in our desired layout and utilise bootstrap 5 (library **bslib**)
- => only app.R needs to be updated (also via our package)



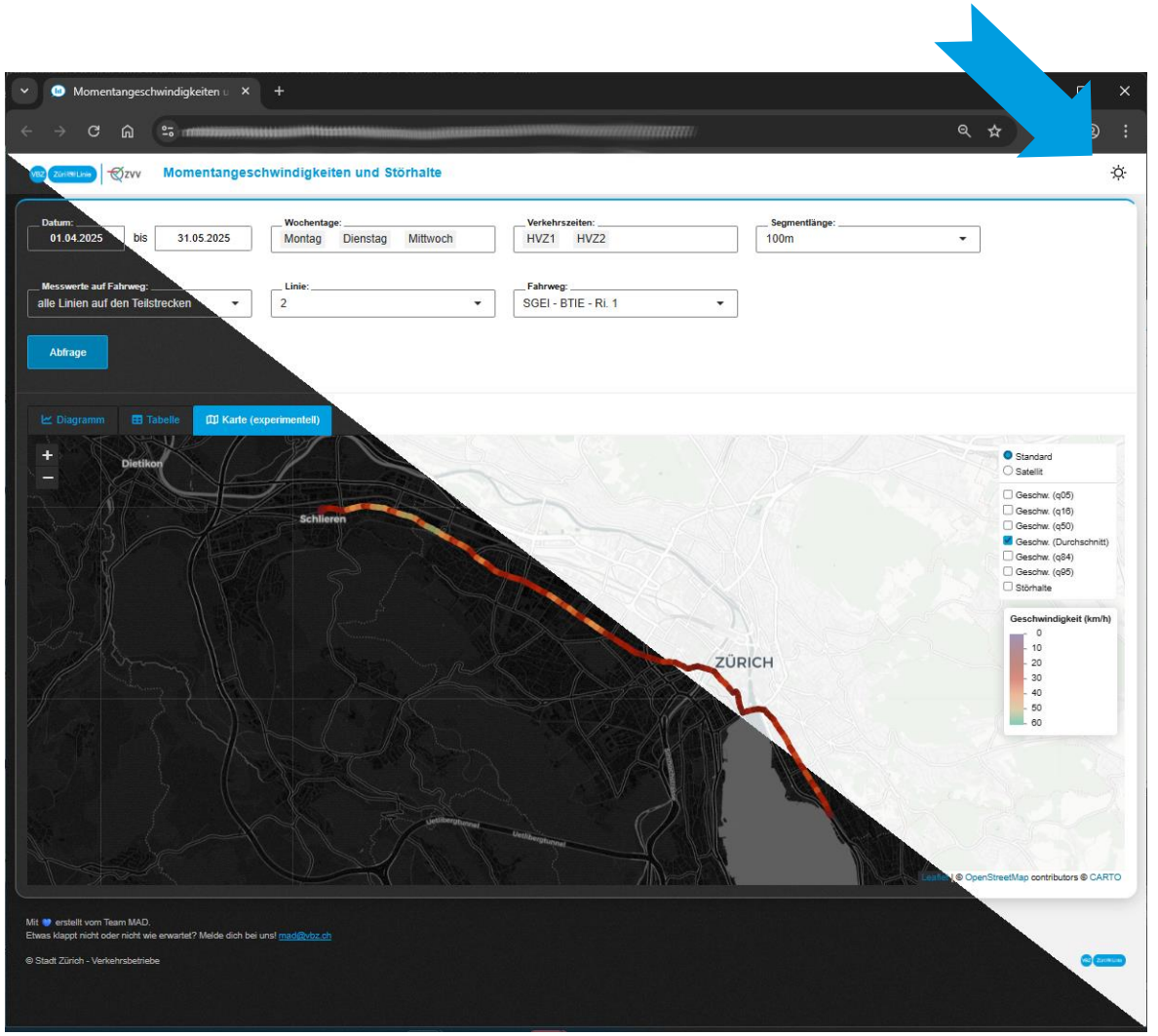
Bootstap 5 with bslib: Easy dark mode (finally!)

- In the UI, all that is needed is an input_dark_mode()

```
# darkmode toggler in navbar
nav_item(input_dark_mode(
  id = "darkmode_toggle",
  mode = NULL
))
```

- Observe it in the server and track with a reactiveVal()

```
# dark_mode als TRUE/FALSE value um diesen einfacher zu nutzen
dark_mode <- reactiveVal(FALSE)
# Beobachten und setzen
observe({
  dark_mode(input$darkmode_toggle == "dark")
})
```

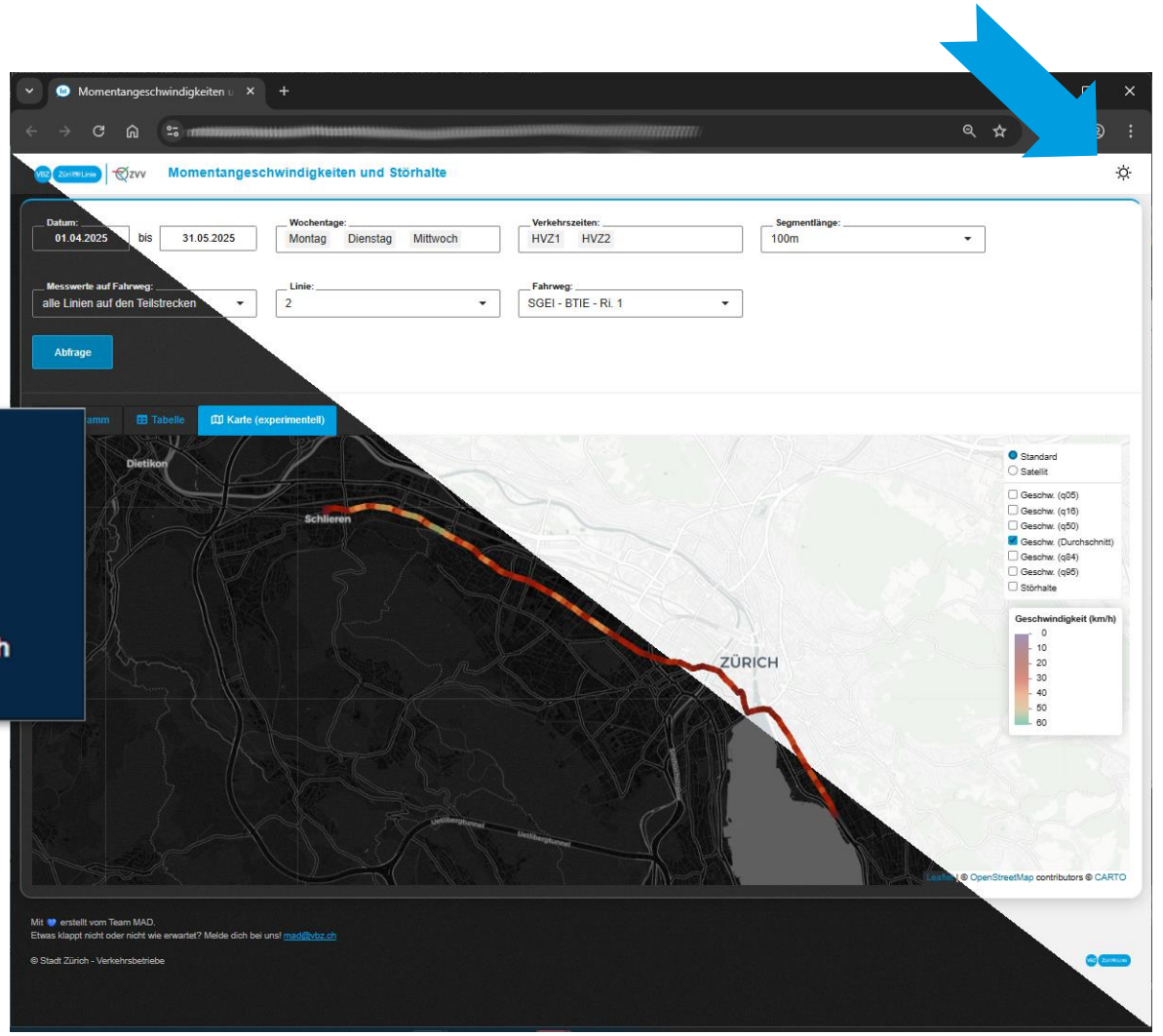


Bootstap 5 with bslib: Easy dark mode (finally!)

- Defined a styler function to pipe a plotly object into
  - Sets colours based on predefined palettes for light and dark

```
fig <- fig %>%
  vbzmad::plotly_styler(
    type = "minimal",
    time_series = FALSE,
    rangeslider = FALSE,
    rangeselector = FALSE,
    dark_mode = dark_mode_switch
  ) %>%
  layout(
```

- The plot now updates correctly when dark mode is toggled



Unit testing with testthat

- We want to make sure our functions behave as expected
- Applying "Test-driven development": Write tests first
- Once the tests are there, you write the function until every test passes, adding more tests as you go if necessary
- Tests can be run within Workbench or directly on gitlab

test-sql_date.R 1.71 KiB

```r
test_that("Input formatting", {
  # Korrekt formatiertes Datum als DATE wird gegeben
  well_formed_date <- as.Date('2024-01-01')
  expect_equal(vbzmad::sql_date(well_formed_date),
               "( make_date(2024,1,1) )")

  # Leerer oder falscher Datentyp als Input
  expect_error(vbzmad::sql_date(NA))
  expect_error(vbzmad::sql_date('20210105'))
  expect_error(vbzmad::sql_date('2021-01-05'))
  expect_no_error((vbzmad::sql_date(as.Date('2021-01-05'))))

})
```

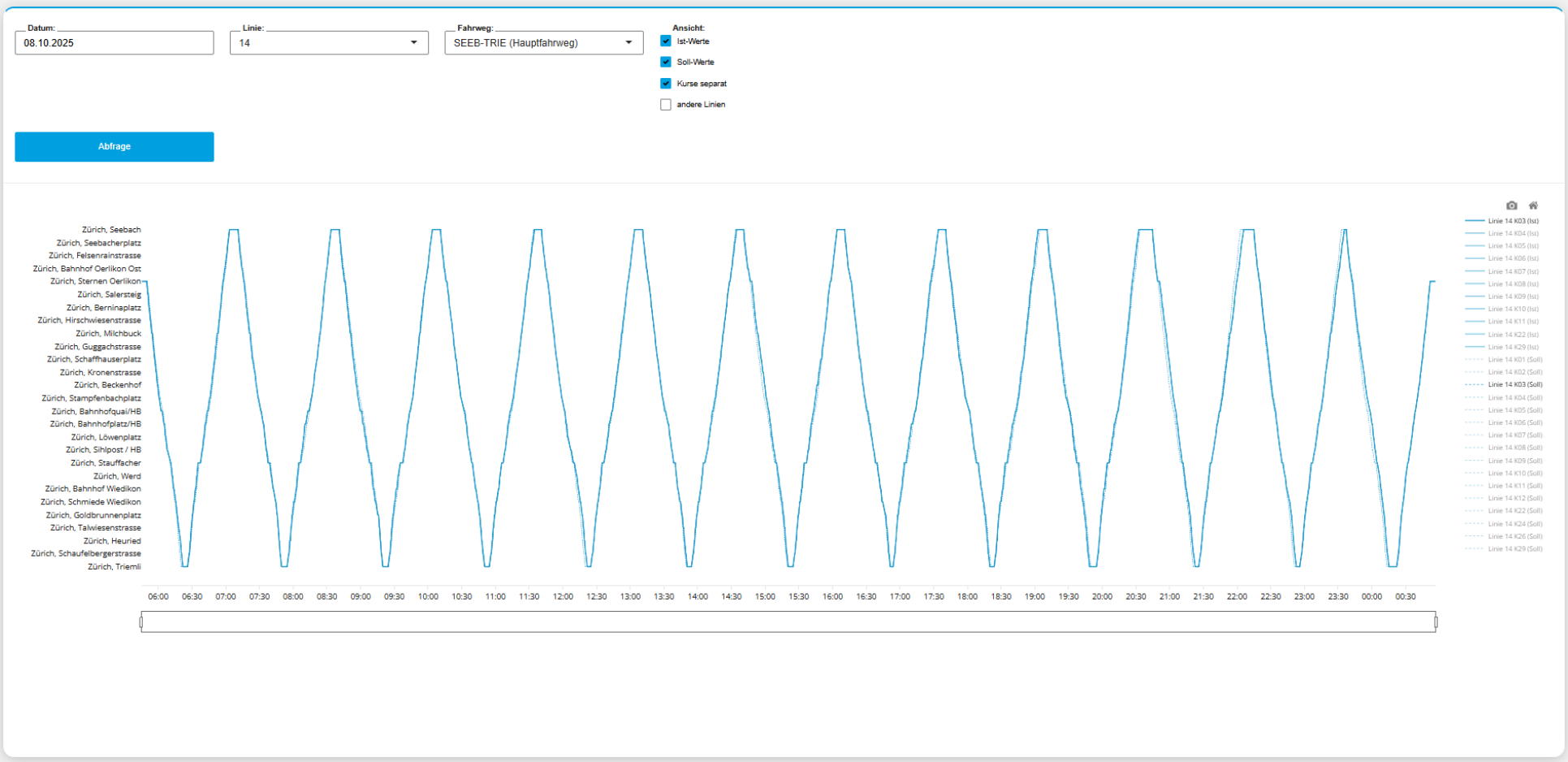
Environment | History | Connections | Build | Git | Tutorial | Databricks

Install | Test | Check | More

```
==> devtools::test()

ℹ Testing vbzmad
✓ | F W  S  OK | Context
✓ |          5 | annual_timetable_bounds
✓ |         19 | annual_timetable
✓ |          5 | colors_by_name
✓ |         21 | create_shiny [2.2s]
✓ |         12 | db_con [2.3s]
✓ |          2 | db_write
✓ |         26 | leaflet_provider
✓ |          1 | lookup_traffic_period_short_name
✓ |         11 | lookup_traffic_period [1.6s]
✓ |          3 | max_date_table
✓ |          3 | min_date_table
✓ |          6 | min_max_date_table
✓ |         22 | plotly_styler
✓ |          5 | qualitative_colors
✓ |          5 | quantiles
✓ |          1 | set_modebar_buttons
✓ |         11 | sql_date
✓ |         10 | timetable_day_code [2.5s]
✓ |         10 | update_or_create_quarto_dashboard [2.2s]
✓ |         10 | update_or_create_quarto_report [1.9s]
✓ |          2 | update_or_create_quarto
✓ |          8 | update_or_create_shiny
✓ |          7 | update_shiny [4.9s]
✓ |         12 | vbz_loader

══ Results ═══════════════════════
Duration: 24.0 s

[ FAIL 0 | WARN 0 | SKIP 0 | PASS 217 ]
```

Shiny app: Speed Monitoring and Operational Disruptions

Shiny app: Path-Time Diagram



Challenges

- Steadily increasing demand for analytics products
  - Vehicle range and battery usage along lines
  - Optimise vehicle deployment and minimise charging time
  - Monitor vehicles for alerts to reduce reaction time if maintenance or intervention is required

- Increasing dataset sizes affect performance
  - Implementing costly computations and preprossecing of data on databricks (R and Python)

- One app performs only one specific analysis
  - Frequent context switching and wait times for users

The future of R at VBZ

- R is here to stay
  - BUT: Constant financial pressure due to high licensing cost
  - However, large potential savings by automating menial data crunching tasks and replacing costly commercial products with custom-made solutions

- Our Goal: Move away from "one app = one analysis" to a more integrated approach
  - Functions from each app will need to be packaged
  - Our Shiny apps need to undergo a further adaptation into a more modular form
    - Shiny modules with golem

Thank you very much!

samuel.wittwer@vbz.ch

VBZ Züri Linie
Umsteigen lohnt sich.

Ein Unternehmen der Stadt Zürich