STi | Sonoma Technology

RShiny, Big Data, and AWS: A Tidy Solution Using Arrow

By Cari Gostic

CascadiaR 2023

Project Overview

- Task
  - Create an RShiny dashboard to visualize air quality data
- Data
  - EPA air monitoring network
    - 400+ sites nationwide, hourly data, 60+ pollutants
    - > 30 million rows
- Available resources:
  - Professional Shinyapps.io license
  - AWS S3 storage

App Characteristics

Network-wide summary metrics

- Pre-calculated aggregate data

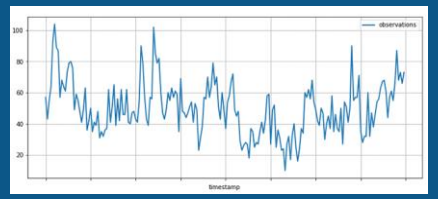
Annual Trend

Measurement Completeness

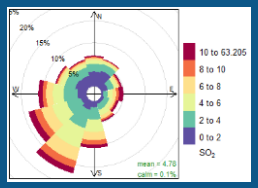
Data exploration

- Raw hourly data over range of years
- One site, 1-2 pollutants visualized

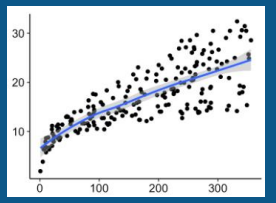
Timeseries



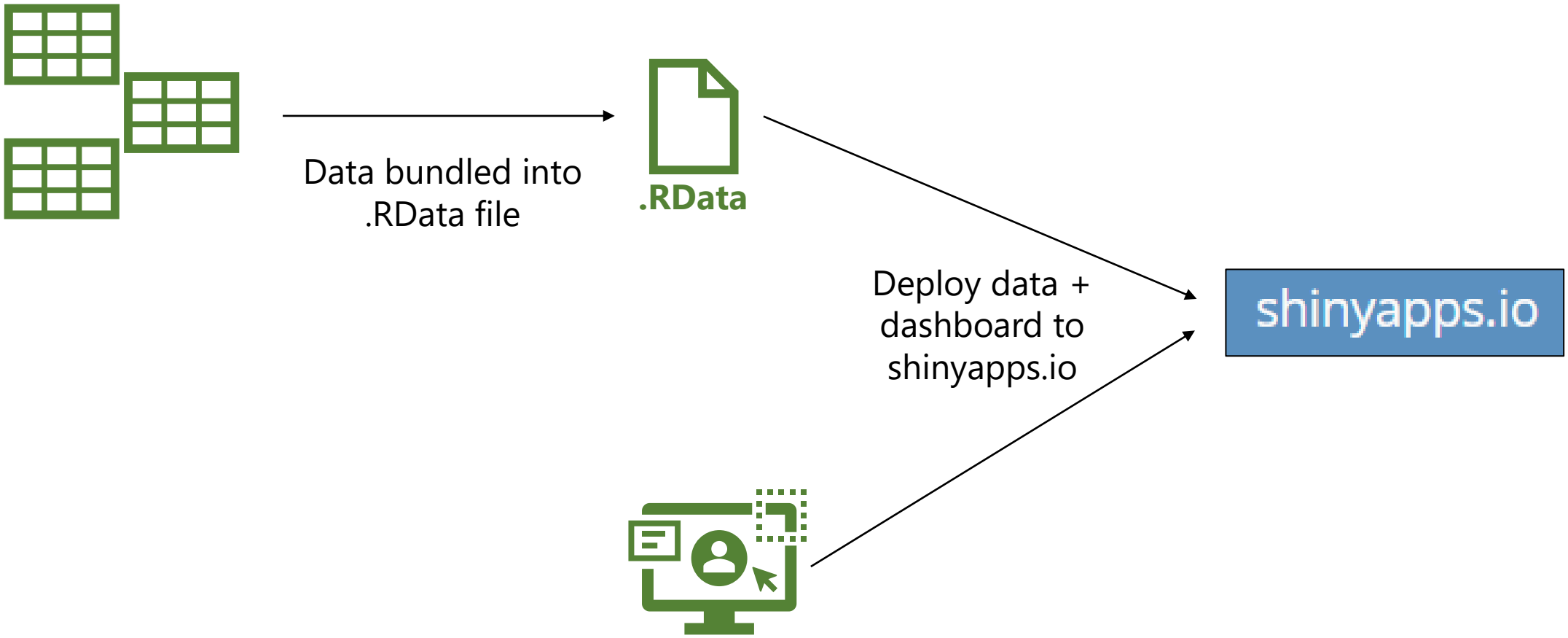
Pollution Rose



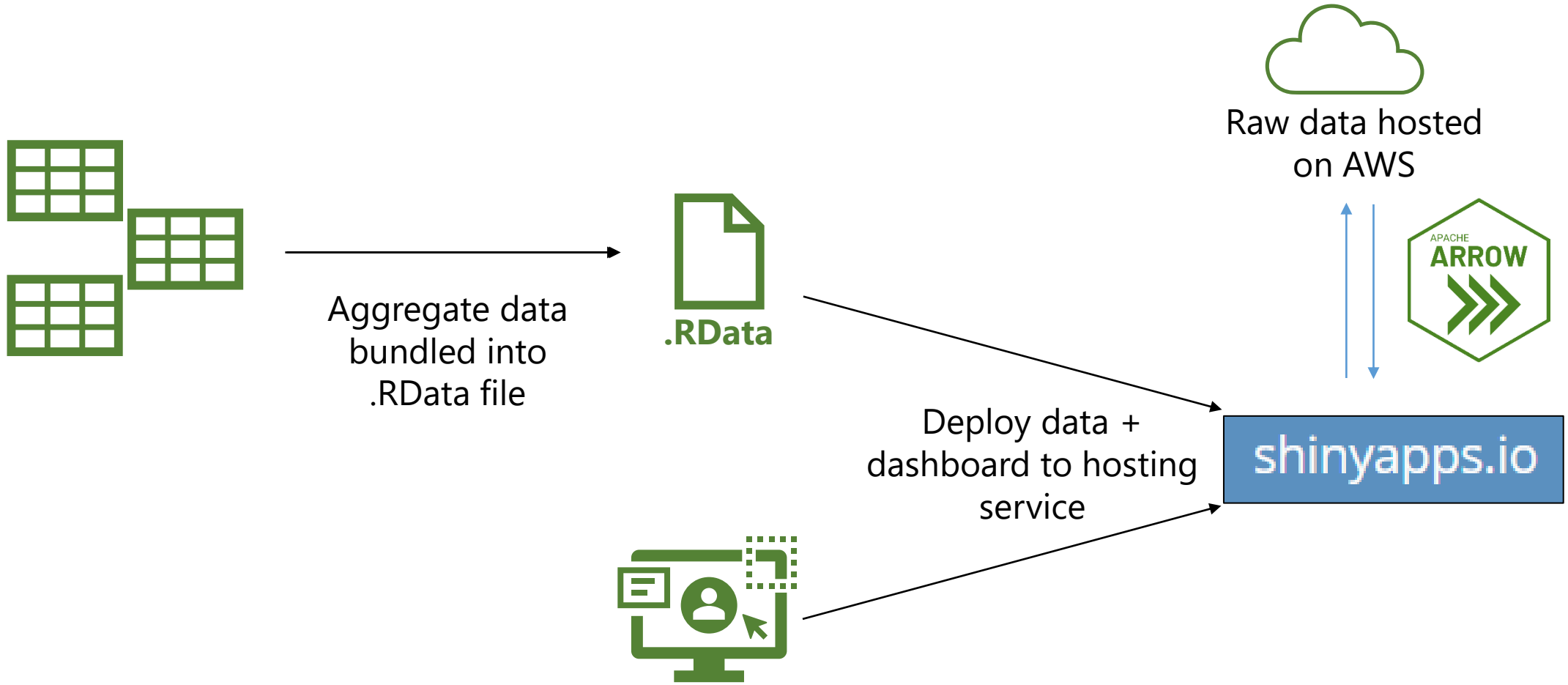
Scatter Plot



“Classic” RShiny Setup



End Product: RShiny + AWS + Arrow

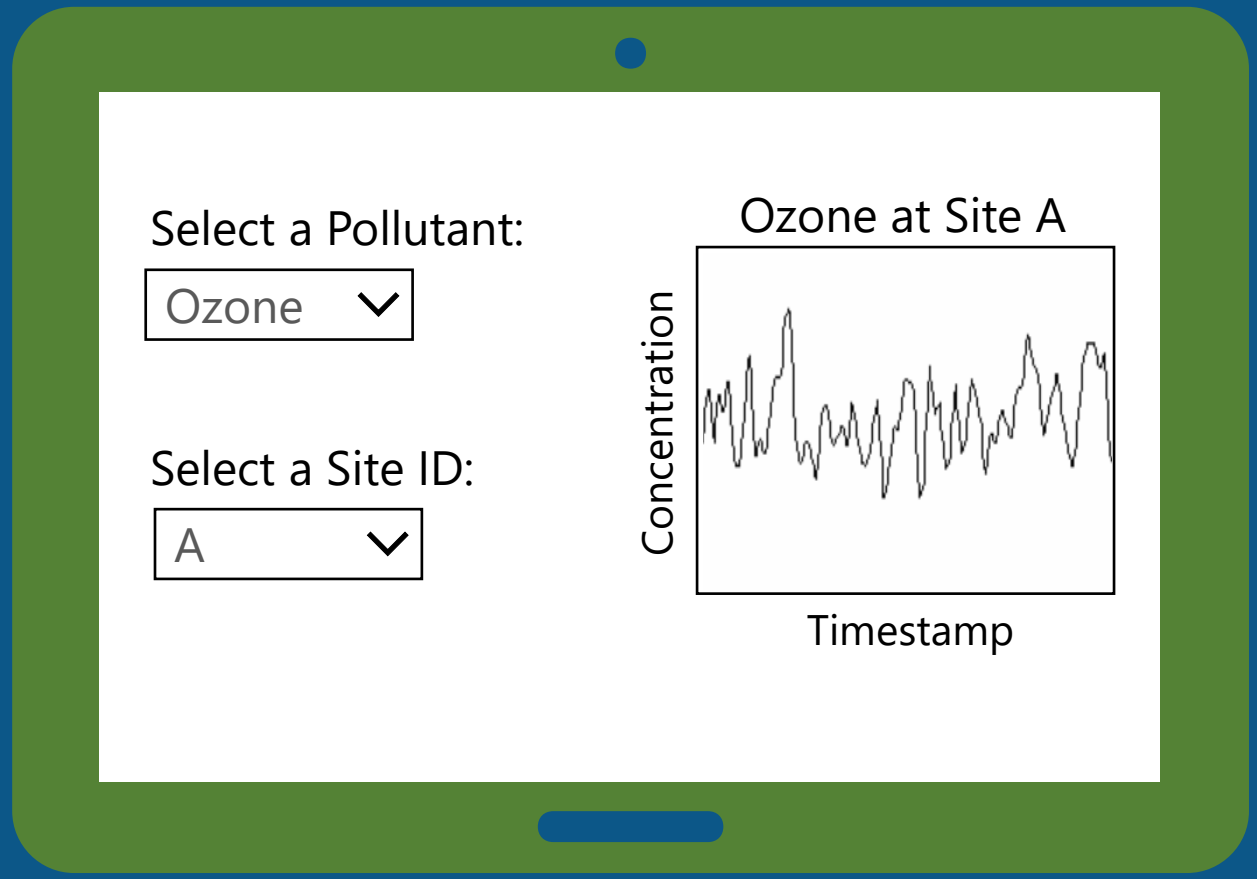


What's So Special About Arrow?

- Time efficient
  - Fast querying of larger-than-memory data
- Memory efficient
  - Perform filters, column selections, joins, aggregations before loading into R
- Well-developed, tidyverse-compatible R package
  - No external software needed
  - Integrates into dplyr pipeline
- AWS compatible
- No license required

Example: “Classic” to AWS + Arrow

Dashboard:
Timeseries of user-selected pollutant at user-selected site



Example: “Classic” to AWS + Arrow

Data:
Hourly data with pollutant concentrations for multiple measurement sites

timeseries_data <-

timestamp	site_id	pollutant	concentration
2022-01-01 00:00	A	Ozone	22
2022-01-01 00:00	B	Ozone	21
2022-01-01 01:00	A	Ozone	26
2022-01-01 01:00	B	Ozone	28
2022-01-01 02:00	A	Ozone	22
2022-01-01 02:00	B	Ozone	21
2022-01-01 03:00	A	Ozone	26
2022-01-01 03:00	B	Ozone	28
2022-01-01 04:00	A	Ozone	22
…	…	…	…

Example: “Classic” to AWS + Arrow

“Classic” RShiny

```
save(timeseries_data, file = '.RData')
```

AWS + Arrow

```
library(arrow)
library(aws.s3)

write_dataset(timeseries_data,
              's3//:<bucket name>', # S3 URI
              ...)
```

Example: “Classic” to AWS + Arrow

“Classic” RShiny

```
# Subset full dataframe to subset needed for plot
plot_data <- reactive({

  timeseries_data %>%
    filter(site_id == input$chooseSite,
           pollutant == input$choosePollutant) %>%
    select(timestamp, concentration)

})
# Plot data subset
output$timeseries <- renderPlot({
  ggplot(plot_data(), aes(timestamp, concentration)) +
    geom_line() +
    ...
})
```

AWS + Arrow

```
library(arrow)
library(aws.s3)

plot_data <- reactive({
  # Connect to AWS bucket
  # Increase connection and request timeouts
  bucket <- s3_bucket('s3://<bucket_name>',
                      connect_timeout = 30,
                      request_timeout = 30)
  # Open dataset created in previous step
  ds <- open_dataset(bucket)
  # Query dataset and collect subset of data
  ds %>%
    filter(site_id == input$chooseSite,
           pollutant == input$choosePollutant) %>%
    select(timestamp, concentration) %>%
    collect() # Initiates computations
})

# Plot data subset
output$timeseries <- renderPlot({
  ggplot(plot_data(), aes(timestamp, concentration)) +
  geom_line() +
  ...
})
```

Example: “Classic” to AWS + Arrow

“Classic” RShiny

```
# Subset full dataframe to subset needed for plot
plot_data <- reactive({

  timeseries_data %>%
    filter(site_id == input$chooseSite,
           pollutant == input$choosePollutant) %>%
    select(timestamp, concentration)

})
# Plot data subset
output$timeseries <- renderPlot({
  ggplot(plot_data(), aes(timestamp, concentration)) +
    geom_line() +
    ...
})
```

AWS + Arrow

```
library(arrow)
library(aws.s3)

plot_data <- reactive({
  # Connect to AWS bucket
  # Increase connection and request timeouts
  bucket <- s3_bucket('s3://<bucket_name>',
                      connect_timeout = 30,
                      request_timeout = 30)
  # Open dataset created in previous step
  ds <- open_dataset(bucket)
  # Query dataset and collect subset of data
  ds %>%
    filter(site_id == input$chooseSite,
           pollutant == input$choosePollutant) %>%
    select(timestamp, concentration) %>%
    collect() # Initiates computations
})

# Plot data subset
output$timeseries <- renderPlot({
  ggplot(plot_data(), aes(timestamp, concentration)) +
  geom_line() +
  ...
})
```

Example: “Classic” to AWS + Arrow

“Classic” RShiny

```
# Subset full dataframe to subset needed for plot
plot_data <- reactive({

  timeseries_data %>%
    filter(site_id == input$chooseSite,
           pollutant == input$choosePollutant) %>%
    select(timestamp, concentration)

})
# Plot data subset
output$timeseries <- renderPlot({
  ggplot(plot_data(), aes(timestamp, concentration)) +
    geom_line() +
    ...
})
```

AWS + Arrow

```
library(arrow)
library(aws.s3)

plot_data <- reactive({
  # Connect to AWS bucket
  # Increase connection and request timeouts
  bucket <- s3_bucket('s3://<bucket_name>',
                      connect_timeout = 30,
                      request_timeout = 30)
  # Open dataset created in previous step
  ds <- open_dataset(bucket)
  # Query dataset and collect subset of data
  ds %>%
    filter(site_id == input$chooseSite,
           pollutant == input$choosePollutant) %>%
    select(timestamp, concentration) %>%
    collect() # Initiates computations
})

# Plot data subset
output$timeseries <- renderPlot({
  ggplot(plot_data(), aes(timestamp, concentration)) +
  geom_line() +
  ...
})
```

Summary

- Arrow is fast, memory-efficient, and free!
- Arrow facilitates real-time querying from cloud storage
- Arrow is useful for loading subsets of a larger data set.

STi | Sonoma Technology

Thank You!

Cari Gostic

Air Quality Data Scientist

cgostic@sonomatech.com

Resources

- https://arrow.apache.org/docs/r/
- https://ursalabs.org/arrow-r-nightly/articles/dataset.html
- https://blog.djnavarro.net/posts/2021-11-19_starting-apache-arrow-in-r/