Pacific Northwest
NATIONAL LABORATORY

Querying GCAM Outputs

8 June 2023

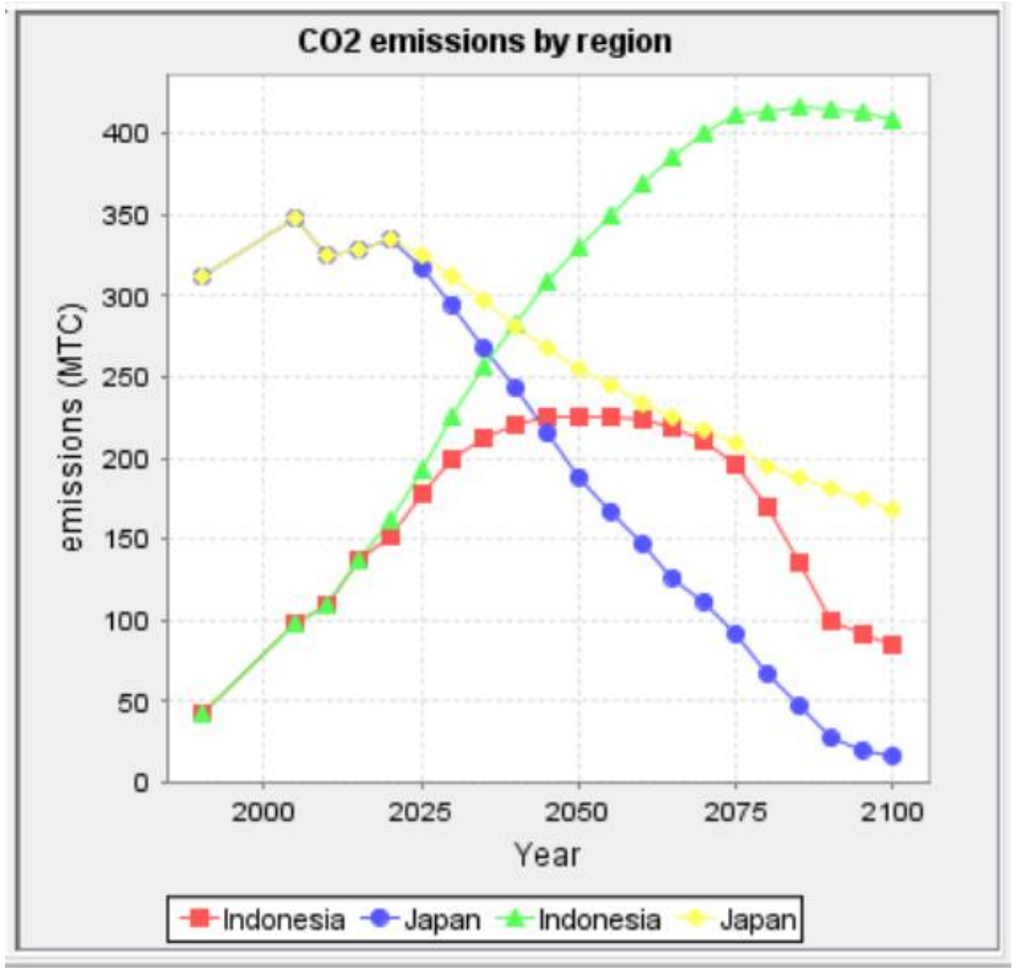
Maridee Weber

U.S. DEPARTMENT OF ENERGY BATTELLE

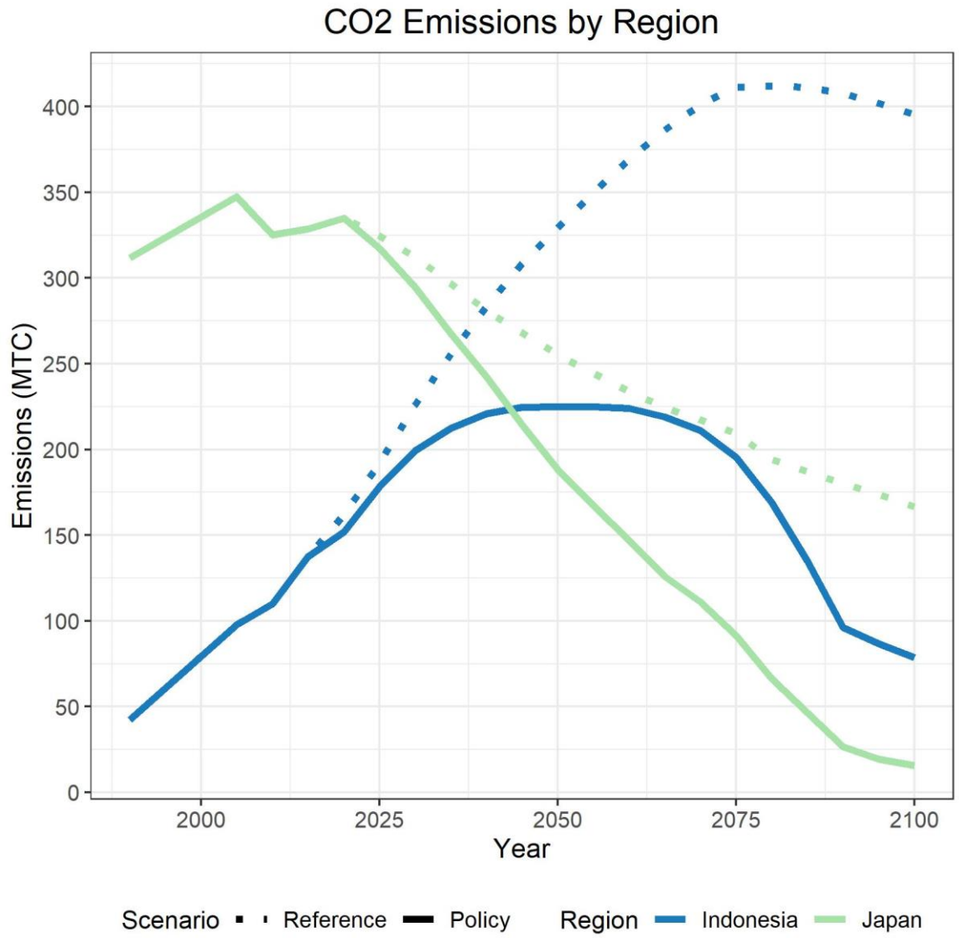
PNNL is operated by Battelle for the U.S. Department of Energy

Congrats, you ran GCAM! Now what?

Model Interface



rgcam

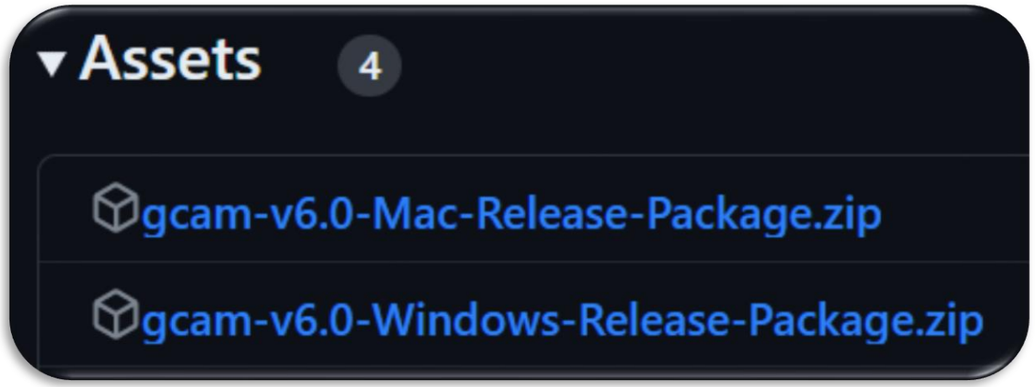


Model Interface

Model Interface (MI) is a program that comes with the Release Version of GCAM, and is used to view GCAM output.

To download MI: https://github.com/JGCRI/gcam-core/releases/tag/gcam-v6.0

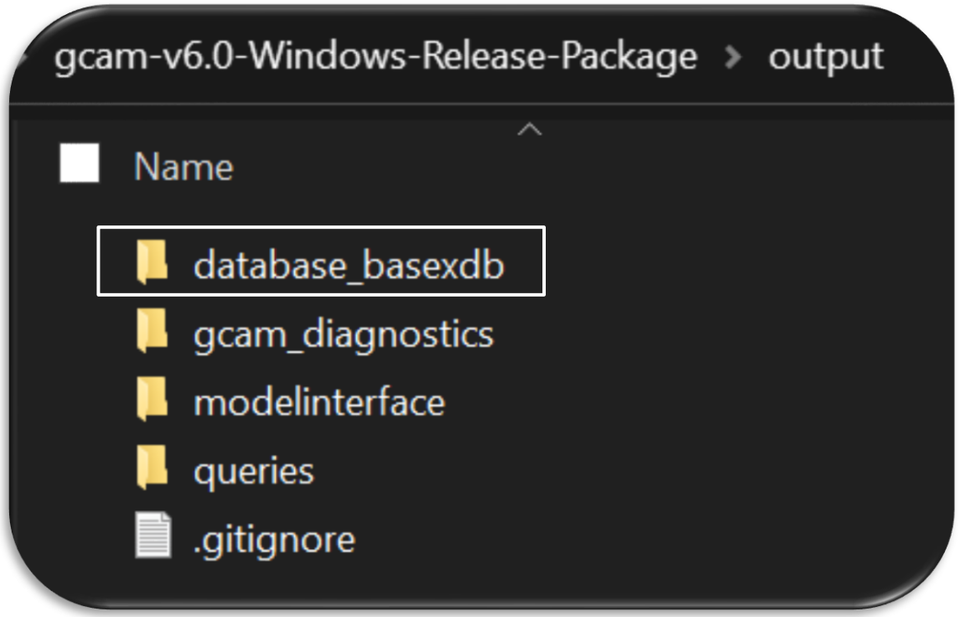
Download the .zip for your operating system, and extract all files.



Model Interface

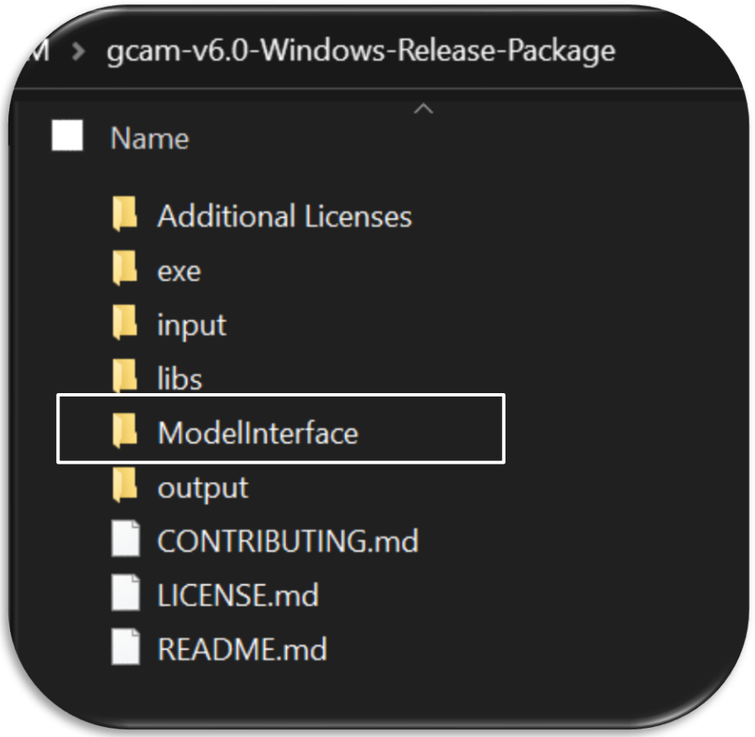
To view GCAM results in MI, we first need a database.

After running GCAM, you will have a database in the **output** folder. It will be called “database_basexdb”, unless you changed the name.

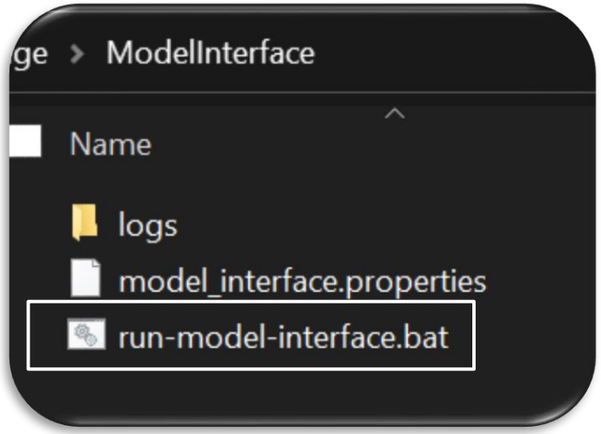


Model Interface

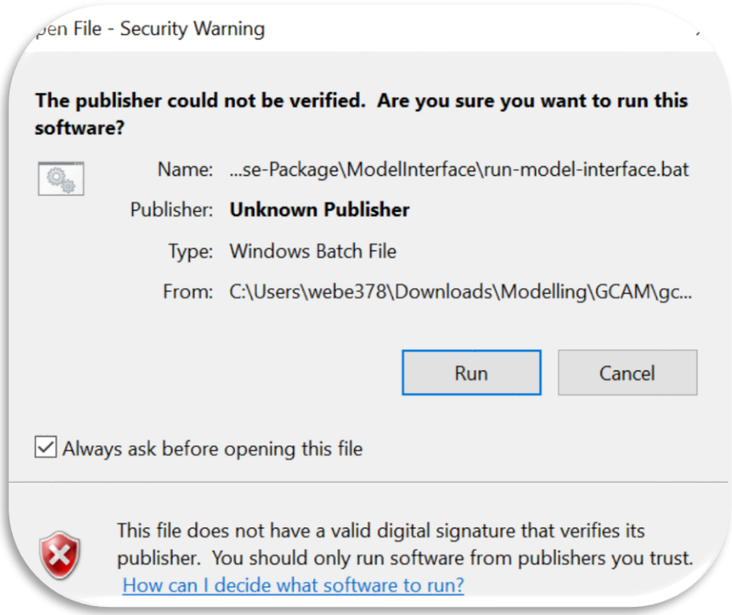
Model Interface lives in the **ModelInterface** folder.



run-model-interface.bat

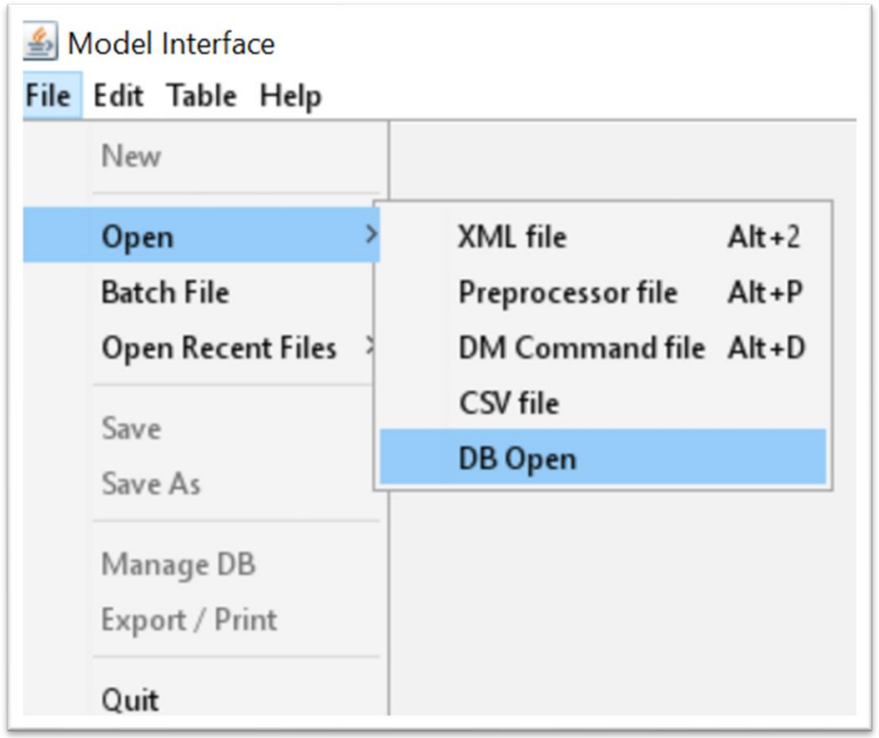


Run

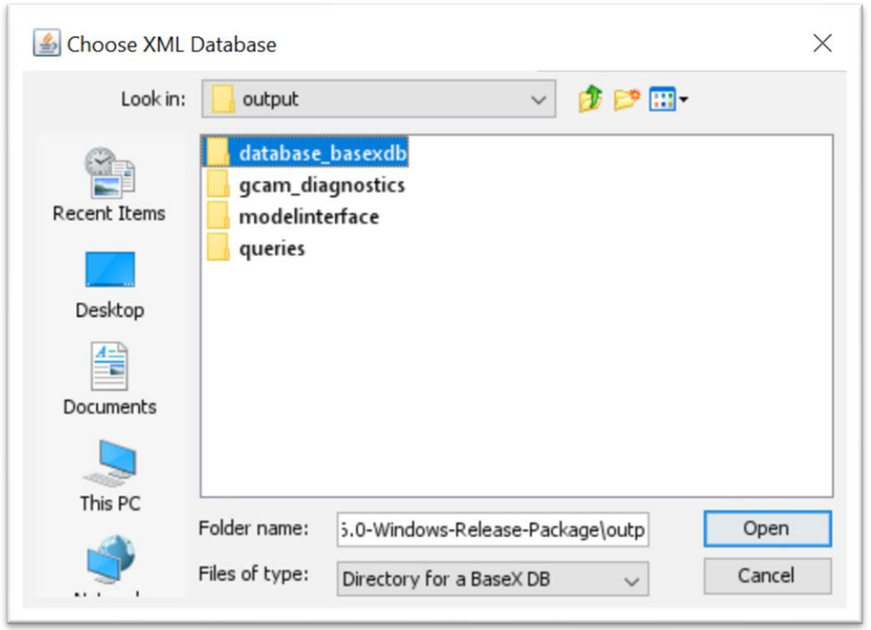


Model Interface

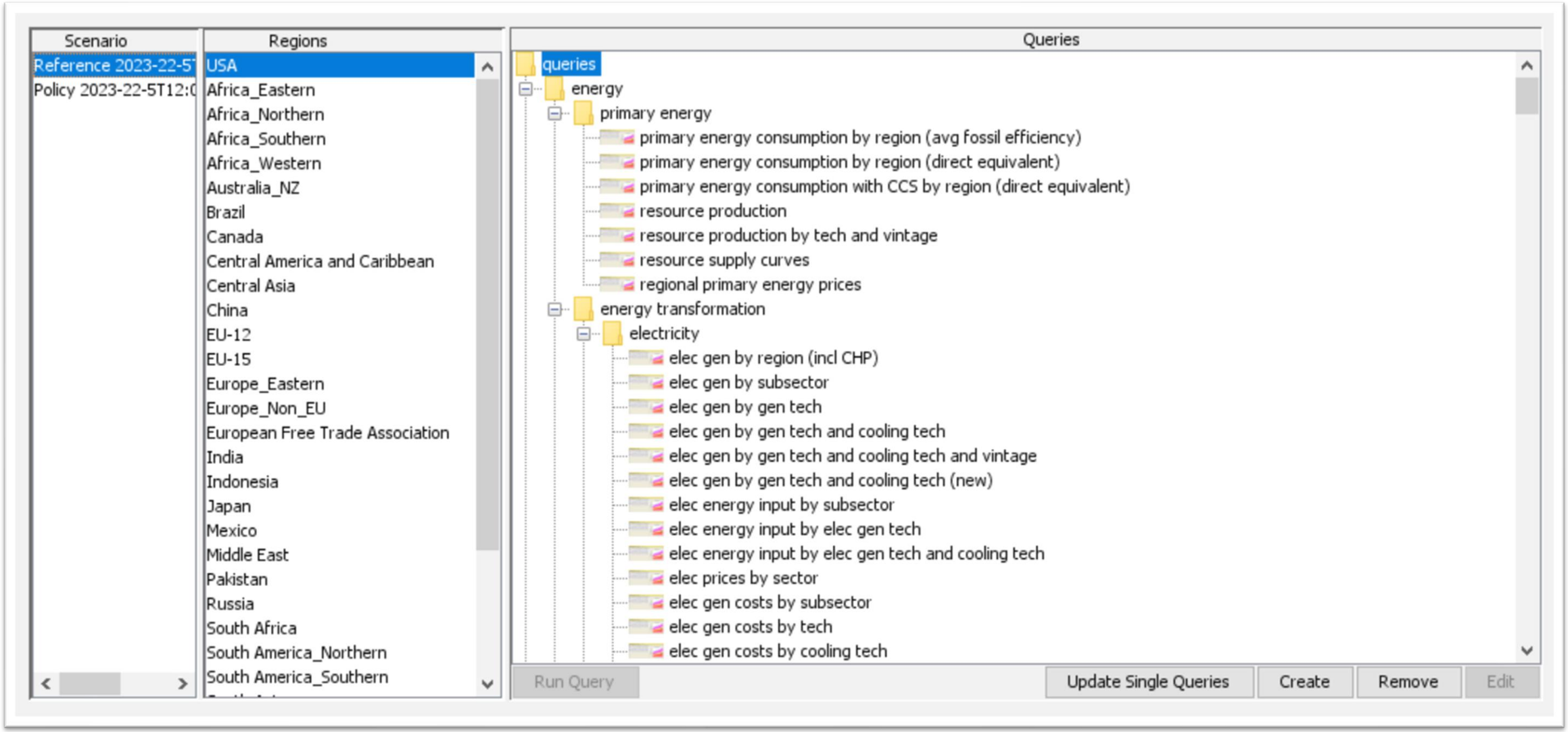
Once open, connect a database by going to File > Open > DB Open



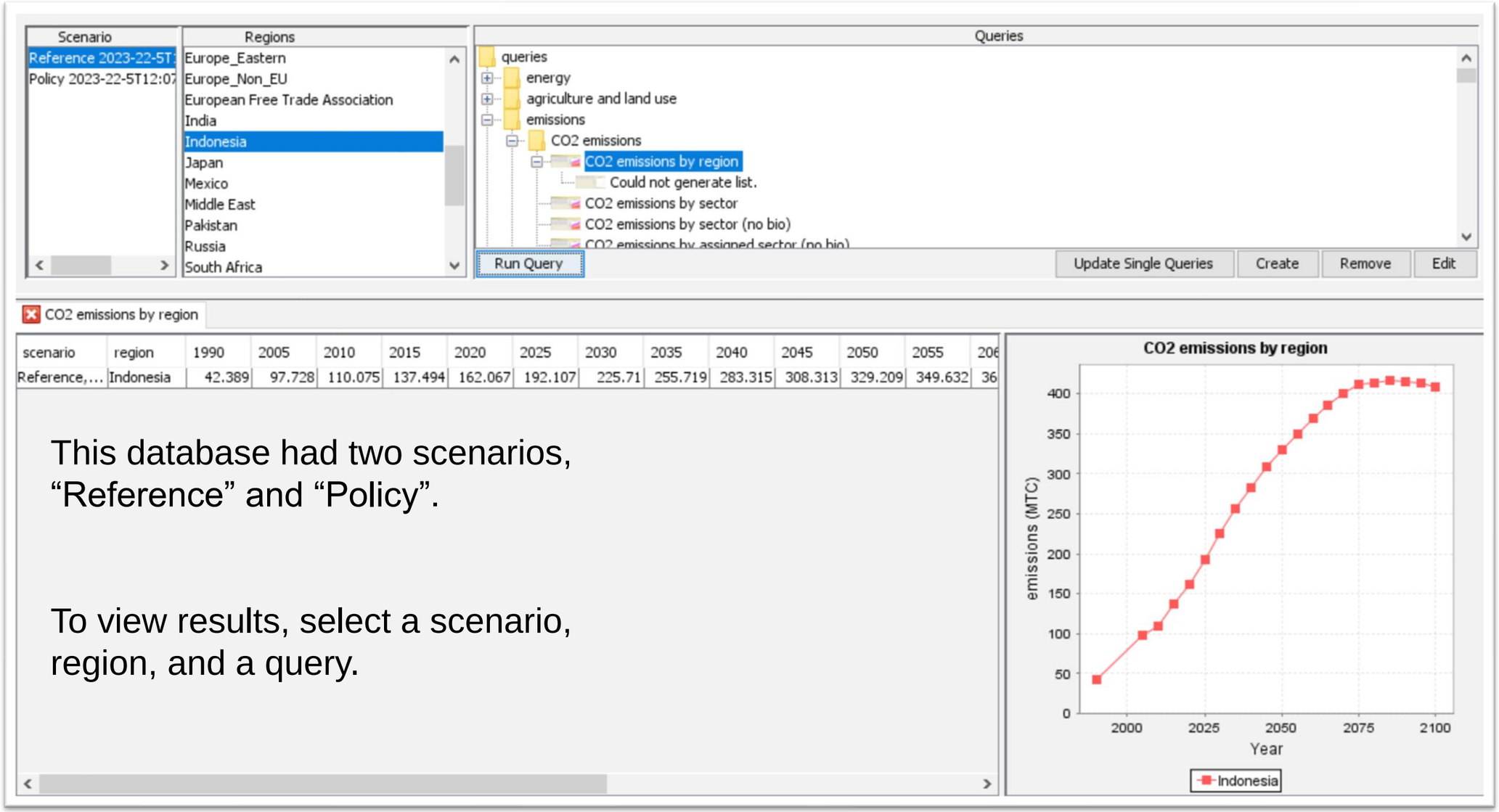
Then, select your database and click Open



Model Interface



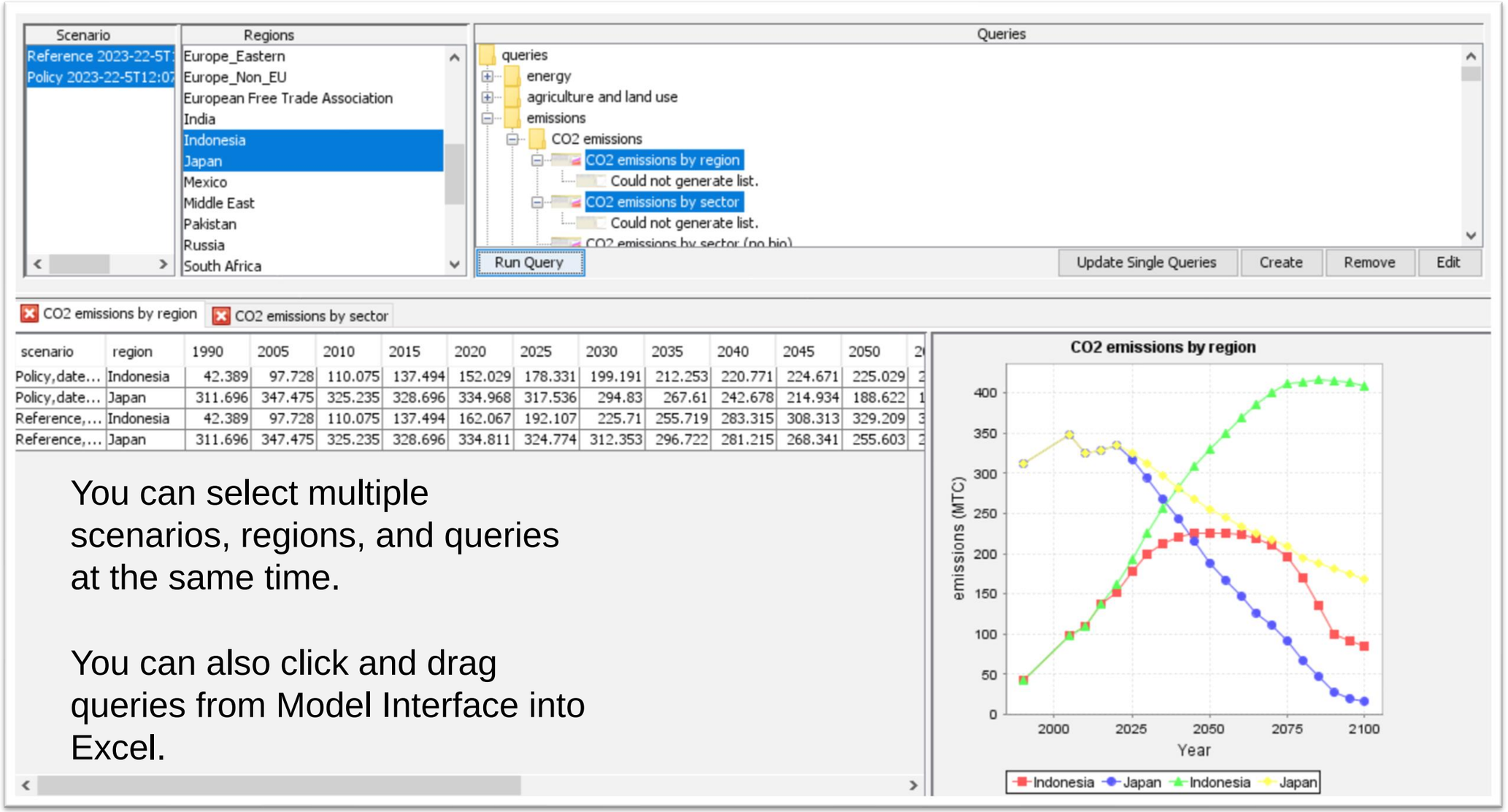
Model Interface



This database had two scenarios, “Reference” and “Policy”.

To view results, select a scenario, region, and a query.

Model Interface



You can select multiple scenarios, regions, and queries at the same time.

You can also click and drag queries from Model Interface into Excel.

rgcam

rgcam is an R package that makes querying and visualizing GCAM output easy.

- Connecting a database
- Querying GCAM outputs
- Visualizing

To install rgcam, open an R session and run:

```
install_github('JGCRI/rgcam', build_vignettes=TRUE)
```

Additional rgcam information: https://github.com/JGCRI/rgcam

rgcam – Connecting a database

Running the `localDBConn` command **connects your database** to rgcam and lists the scenarios available in that database.

```
17  # 1) Connecting a database
18  # localDBConn(`path to your database`, `name of database`)
19  db1 <- localDBConn('./gcam-v6.0-Windows-Release-Package/output', 'database_basexdb')
```

```
> # 1) Connecting a database
> # localDBConn(`path to your database`, `name of database`)
> db1 <- localDBConn('./gcam-v6.0-Windows-Release-Package/output', 'database_basexdb')
Database scenarios:  Reference, Policy
```

rgcam – Querying GCAM output

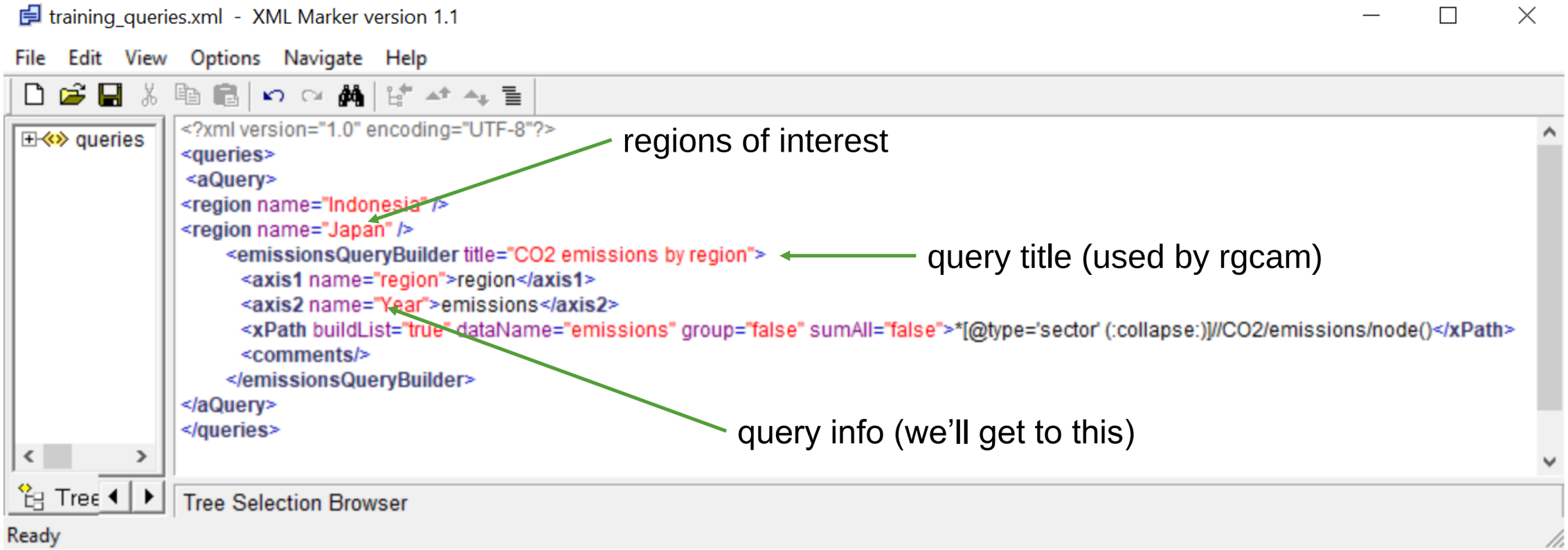
Running the `addScenario` command **extracts the query data** for a scenario in a database.

```
24  # 2) Querying GCAM output
25  # addScenario(database object, `name of file to store rgcam data`, `name of scenario`, `path to/name of query XML`)
26  prj <- addScenario(db1, 'Training.dat', 'Reference', '../GCAM Training/rgcam/training_queries.xml', clobber = TRUE)
27  prj <- addScenario(db1, 'Training.dat', 'Policy', '../GCAM Training/rgcam/training_queries.xml', clobber = TRUE)
```

```
> # 2) Querying GCAM output
> # addScenario(database object, `name of file to store rgcam data`, `name of scenario`, `path to/name of query XML`)
> prj <- addScenario(db1, 'Training.dat', 'Reference', '../GCAM Training/rgcam/training_queries.xml', clobber = TRUE)
Scenario Reference does not exist in this project.  Creating.
> prj <- addScenario(db1, 'Training.dat', 'Policy', '../GCAM Training/rgcam/training_queries.xml', clobber = TRUE)
Scenario Policy does not exist in this project.  Creating.
```

rgcam – Querying GCAM output

Running the `addScenario` command **extracts the query data** for a scenario in a database.



rgcam – Querying GCAM output

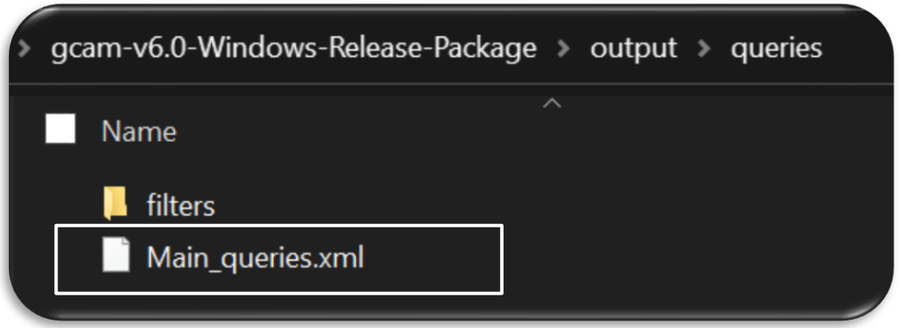
Running the `addScenario` command **extracts the query data** for a scenario in a database.

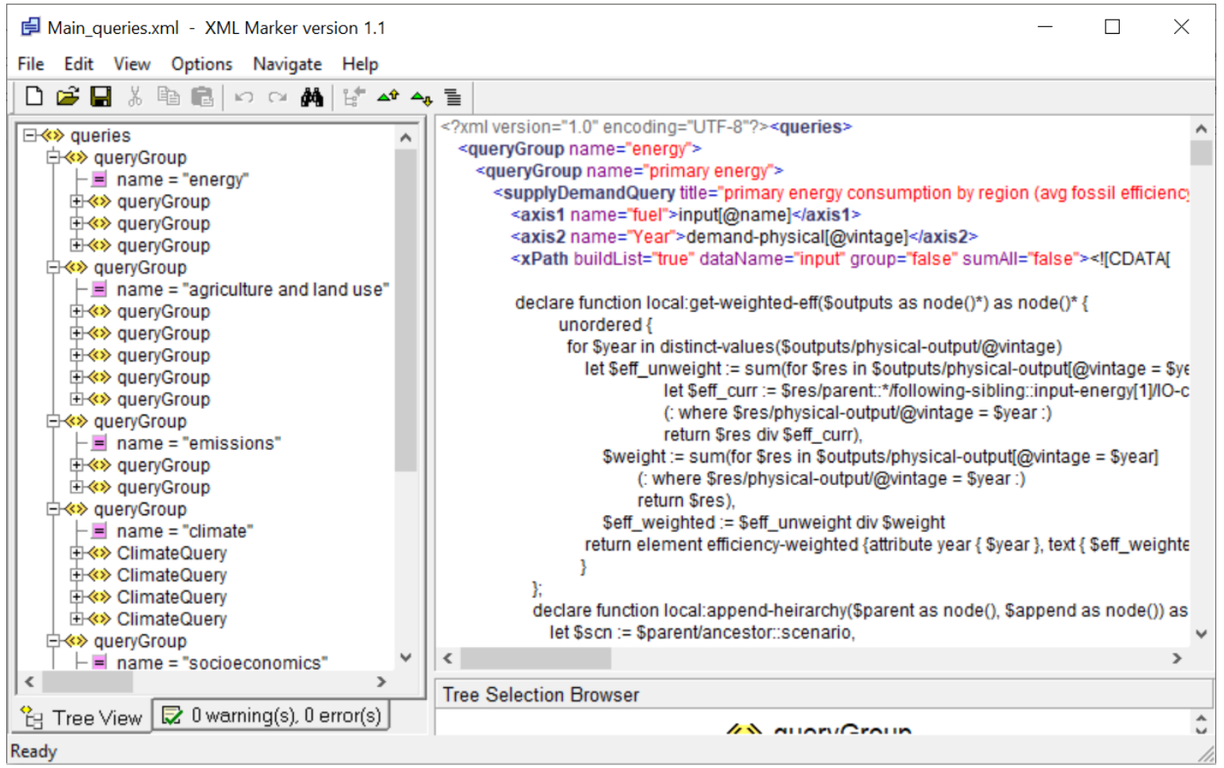
rgcam – Querying GCAM output

To query in rgcam, we need an XML that provides the query info.

We can utilize the Main_queries.xml to get this info.

Main_queries.xml can be found in **output/queries** folder.





rgcam – Querying GCAM output

1) Copy + paste the query you want from Main_queries.xml into your XML used in rgcam (training_queries.xml)
2) Add regions
3) Add `<aQuery>` before, and `</aQuery>` after.

Main_queries.xml

```
<queryGroup name="emissions">
  <queryGroup name="CO2 emissions">
    <emissionsQueryBuilder title="CO2 emissions by region">
      <axis1 name="region">region</axis1>
      <axis2 name="Year">emissions</axis2>
      <xPath buildList="true" dataName="emissions" group="fa
      <comments/>
    </emissionsQueryBuilder>
```

training_queries.xml

```
<queries>
 <aQuery>
<region name="Indonesia" />
<region name="Japan" />
    <emissionsQueryBuilder title="CO2 emissions by region">
      <axis1 name="region">region</axis1>
      <axis2 name="Year">emissions</axis2>
      <xPath buildList="true" dataName="emissions" group="fa
      <comments/>
    </emissionsQueryBuilder>
</aQuery>
</queries>
```

add your next query here, before `</queries>`

rgcam – Querying GCAM output

Running the `getQuery` command **assigns query data** to an object.

```
32  ## Retrieve queries for all scenarios in the database,
33  # getQuery(project object, `query title from XML`)
34  co2_emissions <- getQuery(prj, "CO2 emissions by region")
```

training_queries.xml had one query in it, so we retrieved one table.

We now have a table called `co2_emissions` that contains data from two scenarios, and two regions.

training_rgcam.R | co2_emissions

Filter

	Units	scenario	region	year	value
	All	All	All	[...]	All
1	MTC	Reference	Indonesia	2020	162.0674
2	MTC	Reference	Indonesia	2025	192.1072
3	MTC	Reference	Japan	2020	334.7475
4	MTC	Reference	Japan	2025	324.6737
5	MTC	Policy	Indonesia	2020	152.0291
6	MTC	Policy	Indonesia	2025	178.3314
7	MTC	Policy	Japan	2020	334.9050
8	MTC	Policy	Japan	2025	317.4384

rgcam - Visualizing

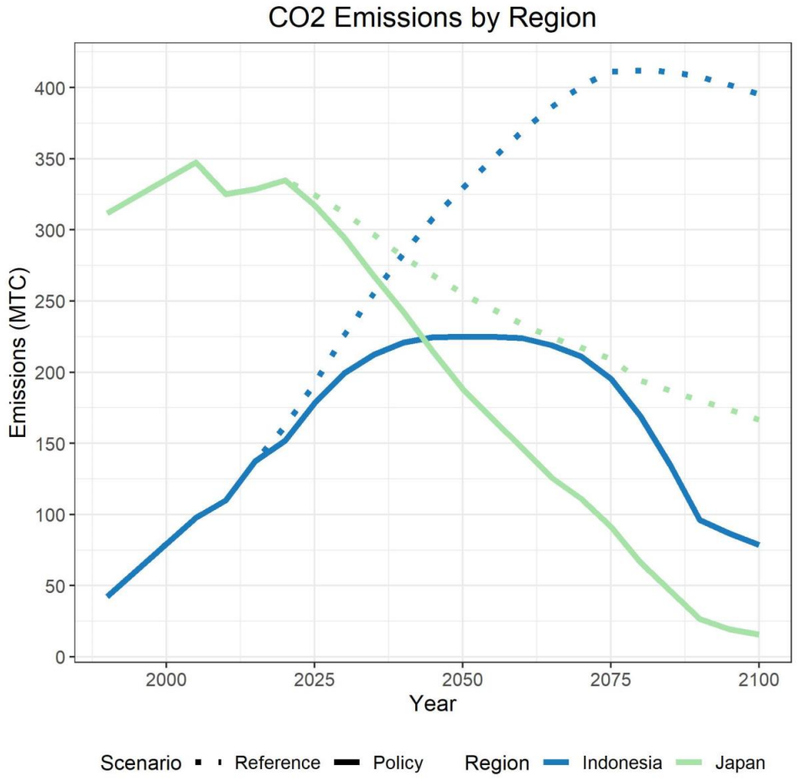
Use your data visualization tools of choice, and add new databases, scenarios, and queries as needed.

```
# ======================================== CO2 Emissions
country_color <- c("Indonesia" = "#1C7BBC",
                   "Japan" = "#A7E3A8")

scenario_line <- c("Reference" = "dotted",
                   "Policy" = "solid")

co2_emissions %>%
  filter(year > 1975) %>%
  rename("MTC" = value,
         "Year" = year,
         "Scenario" = scenario,
         "Region" = region) %>%
  ggplot(aes(Year, MTC)) +
  geom_line(size = 1.5, aes(color = Region, linetype = Scenario)) +
  scale_color_manual(values = country_color) +
  scale_linetype_manual(values = scenario_line) +
  scale_x_continuous(name = "Year", breaks = c(2000, 2025, 2050, 2075, 2100)) +
  scale_y_continuous(name = "Emissions (MTC)", breaks = c(0, 50, 100, 150, 200, 250, 300, 350, 400)) +
  theme_bw() +
  theme(legend.position = "bottom",
        plot.title = element_text(hjust = 0.5, size = 15),
        axis.text = element_text(size = 10),
        axis.title = element_text(size = 12),
        legend.text = element_text(size = 10)) +
  labs(title = "CO2 Emissions by Region")

ggsave("co2_emissions.jpg", height = 6, width = 6)
```



Resources

Model Interface: https://github.com/JGCRI/gcam-core/releases/tag/gcam-v6.0

rgcam: https://github.com/JGCRI/rgcam

R: https://www.r-project.org/

RStudio: https://posit.co/download/rstudio-desktop/

ggplot2: https://ggplot2.tidyverse.org/