METIS Cheat Sheet

Webpage: https://jgcri.github.io/metis/
Github: https://github.com/JGCRI/metis
Cheat sheet

Install

Metis is an R package. The code in this cheat sheet are all meant to be run in R.

Install R: https://www.r-project.org/
Install R Studio: https://www.rstudio.com/
Then in R studio:

```
install.packages("devtools")
devtools::install_github("JGCRI/rgcam")
devtools::install_github("JGCRI/metis")
```

Note: The first time installation can take a while to get the required packages and data.

UBUNTU additional steps:

```
sudo add-apt-repository ppa:ubuntugis/ppa
sudo apt-get update
sudo apt-get install libudunits2-dev libgdal-dev
libgeos-dev libproj-dev libmagick++-dev
```

MAC OSX additional steps:

```
brew install pkg-config
brew install gdal
brew install imagemagick@6
```

metis.readgcam

metis.readgcam() reads data from a GCAM database and formats it for metis charts and maps

Extended Examples

KEY INPUTS

- gcamdatabase **OR** dataProjFile (try exampleGCAMproj)
- scenOrigNames (**Optional**) (Subset scenarios)
- regionsSelect (**Optional**) (Subset regions)
- paramsSelect (**Optional**) (Param list on **Page 3**)
- dirOutputs (**Optional**) (Default is working dir/outputs)

CODE

```
library(metis)

dataGCAM <- metis.readgcam (
    #gcamdatabase = "Path_to_GCAMdatabase",
    dataProjFile = metis::exampleGCAMproj)

df <- dataGCAM$data
dfParam <- dataGCAM$dataAggParam
dfClass1 <- dataGCAM$dataAggClass1
```

KEY OUTPUTS

- Function returns a list with data ("df" above) **AND**
- Data also saved in dirOutputs/readGCAM folder
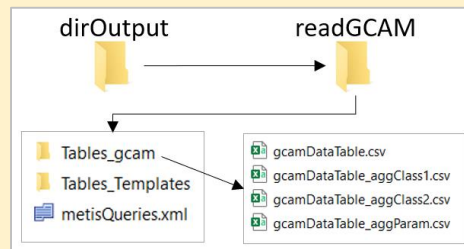
- gcamDataTable.csv has all data
- gcamDataTable_aggClass1.csv has data aggregated to class1 (same for class 2 and param)

Other Key Functions

metis.mapsProcess

Page 2

- All maps pre-loaded (GCAM regions, basins, states)
- Difference maps for multiple scenarios
- Animations for multiple years
- Easily customize scales to highlight data
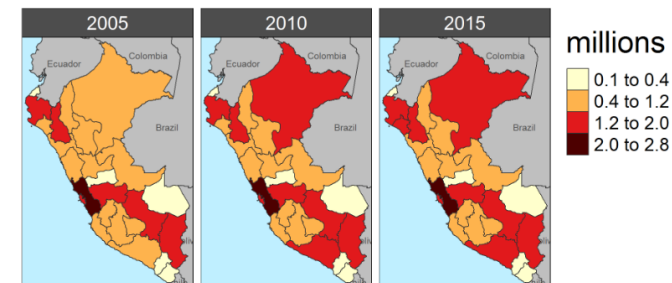
Colors, maps, params

Page 3

- List of metis color palettes
- List of metis maps
- List of available parameters

metis.chartsProcess

In progress..

- Easily process GCAM outputs
- Connect to a database Or .proj file
- Filter by scenario, region, year and params

metis.mapsProcess

Structure

KEY INPUTS

myFile.csv file **OR** **R Data Frame**

subRegion	value
TX	32
AZ	54

```
data = data.frame(
    subRegion = c("TX","AZ"),
    value = c(32,54))
```
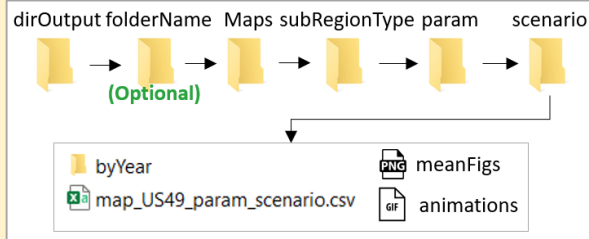
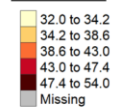**Optional Columns:** param, scenario, year, class, units

CODE

```
library(metis);
metis.mapsProcess (data)   # OR
metis.mapsProcess("path/To/myFile.csv")
```

KEY OUTPUTS

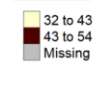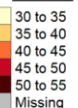- Maps saved in the working directory as follows:

dirOutput folderName Maps subRegionType param scenario

(Optional)

byYear
map_US49_param_scenario.csv

meanFigs (PNG)
animations (GIF)

| **FreeScale**	**Kmeans**	**Pretty**
32.0 to 34.2	32 to 43	30 to 35
34.2 to 38.6	43 to 54	35 to 40
38.6 to 43.0	Missing	40 to 45
43.0 to 47.4		45 to 50
47.4 to 54.0		50 to 55
Missing		Missing
Each map own scale	Same scale across years and classes	

List of Maps and Color Palettes on **Page 3**

Pre-loaded Maps (Automatically find maps for data if available)

US49

```
data = data.frame(subRegion = c("TX", "AZ"),
value = c(32, 54), year=c(2010,2010))
metis.mapsProcess(polygonTable = data)
```
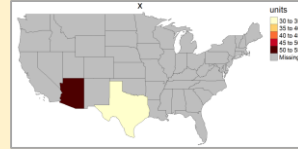
Countries and cropToBoundary

```
data = data.frame(subRegion = c("India","China"), value = c(32,54))
metis.mapsProcess(polygonTable = data, cropToBoundary=T)
```

GCAM Basins

```
data = data.frame( subRegion = c("La_Plata","Amazon"),
value = c(32,54)
metis.mapsProcess(polygonTable = data , cropToBoundary=T)
```
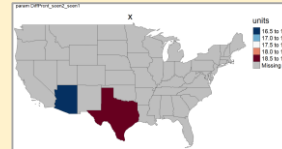

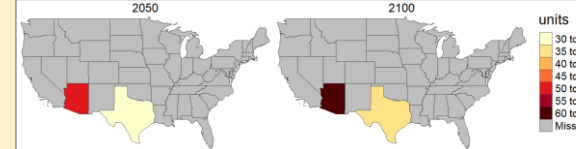## Multiple Scenarios, Years and Classes

Multi-scenario Diff plots

```
data = data.frame(subRegion = c("TX","TX", "AZ", "AZ"),
scenario = c("scen1","scen2","scen1","scen2"),
value = c(32, 38, 54, 63))
metis.mapsProcess(polygonTable = data, scenRef="scen1")
```

- DiffAbs_scen2_scen1
- DiffPrcnt_scen2_scen1
- scen1
- scen2

Multi-Year Animation/Mean

```
data = data.frame(subRegion = c("TX","TX", "AZ", "AZ"),
year = c("2050","2100","2050","2100"), value = c(32, 38, 54, 63))
metis.mapsProcess(polygonTable = data,
folderName="multiyear")
```

Multi-Class

```
data = data.frame(subRegion = c("TX","TX", "AZ", "AZ"),
class = c("class1","class2","class1","class2"),
value = c(32, 38, 54, 63))
metis.mapsProcess(polygonTable = data)
```
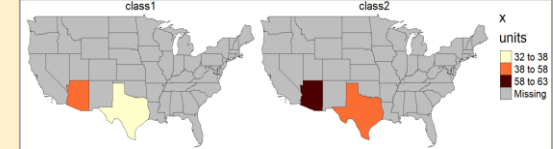

Customize Scales, Colors, Background

Set scale ranges

```
data = data.frame(subRegion = c("TX","TX", "AZ", "AZ"),
scenario = c("scen1","scen2","scen1","scen2"),
value = c(32, 38, 54, 63))
metis.mapsProcess(polygonTable = data,
scaleRange = c(30,50), scaleRangeDiffPrcnt = c(10,30))
```

Change Palettes

```
data = data.frame(subRegion = c("TX","TX", "AZ", "AZ"),
scenario = c("scen1","scen2","scen1","scen2"),
value = c(32, 38, 54, 63))
metis.mapsProcess(polygonTable = data, scenRef= "scen1",
classPalette = "pal_wet", classPaletteDiff = "pal_green")
```
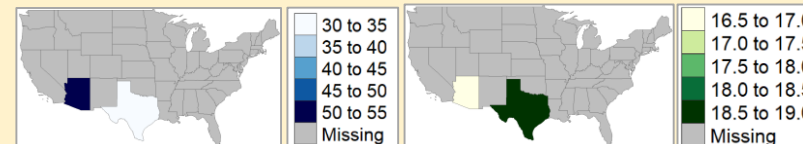
Extended Boundary

```
data = data.frame(
subRegion = c("India","China"), value = c(32,54))
metis.mapsProcess(polygonTable = data,
extension = T)
```
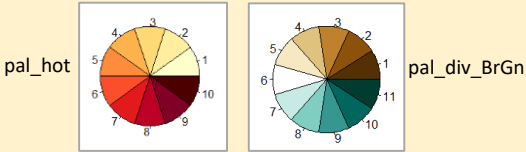
Colors, maps, params

Extended Examples

Selected Color Palettes

- pal_hot
- pal_wet
- pal_green
- pal_spectral
- pal_metis
- pal_div_RdBl
- pal_div_RdBlu
- pal_div_BrGn
- pal_16

```
library(metis);  ?metis.colors() # See all palettes
metis.colors("pal_hot")
```

pal_hot pal_div_BrGn

Selected Maps List

- mapCountries
- mapStates
- mapUS49
- mapUS49County
- mapUS49HUC2
- mapGCAMReg32
- mapGCAMBasinsUS49
- mapIntersectGCAMBasin32Reg
- mapIntersectGCAMBasinCountry
- mapHydroShed1
- mapHydroShed2
- mapHydroShed3
- mapUS49HUC4
- mapGCAMBasins
- mapGCAMLand

```
library(metis);  head(mapGCAMReg32@data)
metis::metis.map(mapUS49, labels=T)
```
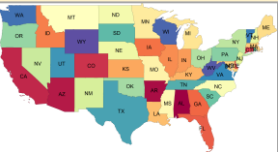

metis.readGCAM paramsSelect list

Pick individual parameters or the param-set name (energy, electricity, transport, water, socioecon, ag, livestock, land, emissions)

energy

- energyPrimaryByFuelEJ
- energyPrimaryRefLiqProdEJ
- energyFinalConsumBySecEJ
- energyFinalByFuelBySectorEJ
- energyFinalSubsecByFuelTranspEJ
- energyFinalSubsecByFuelBuildEJ
- energyFinalSubsecByFuelIndusEJ
- energyFinalSubsecBySectorBuildEJ
- energyFinalConsumByIntlShpAvEJ
- energyPrimaryByFuelMTOE
- energyPrimaryRefLiqProdMTOE
- energyFinalConsumBySecMTOE
- energyFinalbyFuelMTOE
- energyFinalSubsecByFuelTranspMTOE
- energyFinalSubsecByFuelBuildMTOE
- energyFinalSubsecByFuelIndusMTOE
- energyFinalSubsecBySectorBuildMTOE
- energyFinalConsumByIntlShpAvMTOE
- energyPrimaryByFuelTWh
- energyPrimaryRefLiqProdTWh
- energyFinalConsumBySecTWh
- energyFinalbyFuelTWh
- energyFinalSubsecByFuelTranspTWh
- energyFinalSubsecByFuelBuildTWh
- energyFinalSubsecByFuelIndusTWh
- energyFinalSubsecBySectorBuildTWh
- energyFinalConsumByIntlShpAvTWh

electricity

- elecByTechTWh
- elecCapByFuel
- elecFinalBySecTWh
- elecFinalByFuelTWh
- elecNewCapCost
- elecNewCapGW
- elecAnnualRetPrematureCost
- elecAnnualRetPrematureGW
- elecCumCapCost
- elecCumCapGW
- elecCumRetPrematureCost
- elecCumRetPrematureGW

transport

- transportPassengerVMTByMode
- transportFreightVMTByMode
- transportPassengerVMTByFuel
- transportFreightVMTByFuel

water

- watConsumBySec
- watWithdrawBySec
- watWithdrawByCrop
- watBioPhysCons
- watIrrWithdrawBasin
- watIrrConsBasin
- watSupRunoffBasin

socioecon

- gdpPerCapita
- gdp
- gdpGrowthRate
- pop

ag

- agProdbyIrrRfd
- agProdBiomass
- agProdForest
- agProdByCrop

livestock

- livestock_MeatDairybyTechMixed
- livestock_MeatDairybyTechPastoral
- livestock_MeatDairybyTechImports
- livestock_MeatDairybySubsector

land

- landIrrRfd
- landIrrCrop
- landRfdCrop
- landAlloc
- landAllocByCrop

emissions

- emissNonCO2BySectorGWPAR5
- emissNonCO2BySectorGTPAR5
- emissNonCO2BySectorOrigUnits
- emissLUC
- emissCO2BySectorNoBio
- emissNonCO2ByResProdGWPAR5
- emissMethaneBySourceGWPAR5
- emissByGasGWPAR5FFI
- emissByGasGWPAR5LUC
- emissBySectorGWPAR5FFI
- emissBySectorGWPAR5LUC
- emissNonCO2ByResProdGTPAR5
- emissMethaneBySourceGTPAR5
- emissByGasGTPAR5FFI
- emissByGasGTPAR5LUC
- emissBySectorGTPAR5FFI
- emissBySectorGTPAR5LUC

```
library(metis)
df1 <- metis.readgcam(dataProjFile=metis::exampleGCAMproj, paramsSelect="energy", saveData = F)
df2 <- metis.readgcam(dataProjFile=metis::exampleGCAMproj, paramsSelect="elecByTechTWh",
                      saveData = F)
head(df1$data); head(df2$data)
```