MetroScope Gen 3.5

Operator Guide and Technical Reference

Metro Research Center
Economic and Land Use Forecasting

Draft 4/24/2013 Jim Cser jim.cser@oregonmetro.gov

Metro | *Making a great place*

MetroScope Gen 3.5 Operator Guide

Draft 4/24/2013 Jim Cser, Metro Economic and Land Use Forecasting

Who this is for:

Anyone operating or installing MetroScope. Additional documentation can be found in "MetroScope Technical Reference" and "MetroScope Theory and Practice".

About "R"

MetroScope is written in the free and open source "R" mathematical scripting language, which can run on any operating system. MetroScope is compatible with the latest version of R (currently 2.15.1), but will most likely work with other versions. For downloads and more information, please go to http://www.r-project.org/ .

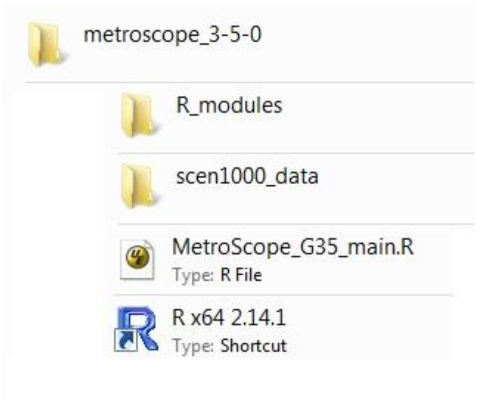
Installing MetroScope

The package folder contains all the directories and files necessary for MetroScope. To install, just copy the folder anywhere on your machine or network.

The top-level directory "MetroScope_3-5-0" is the "working directory", and all file paths in the scripts are relative to that.

Setting R Working Directory

When you install R, by default a desktop shortcut to the GUI is created-- this will be a big blue "R" with the version number. First, copy the shortcut to"MetroScope_3-5-0". Next, right click on the shortcut, and select Properties. In the dialog box, select the Shortcut tab, and type the full path name to "MetroScope_3-5-0" (if you installed it in C:, the path will be "C:/MetroScope_3-5-0". In the end, you should see something like this:



metroscope_3-5-0
R_modules
scen1000_data
MetroScope_G35_main.R
Type: R File
R x64 2.14.1
Type: Shortcut

Overview of R Control Script

The main script for loading and running MetroScope scenarios is “MetroScope G35_main.R”, which executes these general tasks:

```
Load Run Parameters
Load Global Constants
Load Input/Output Functions
Load Residential and Non-residential Module Functions

Loop through Model Years {
    Iterate Land Use Models {
        Iterate Non-Residential Model  {
            Run Non-Residential Demand Module
            Run Non-Residential Supply Module
            Calculate New Non-Residential Location Prices
         }
        Iterate Residential Model  {
            Run Residential Demand Module
            Run Residential Supply Module
            Calculate New Residential Location Prices
         }
    }
}
```

The complete R script can be found at the end of this document.

Editing R Control Scripts

Before running MetroScope, you must edit the header of the appropriate R script. You can either edit “MetroScope_G35_main.R” directly, but it is usually useful to make a new copy for each scenario, such as “Scen1000_main.R”   Don’t forget to save often!

User-defined scenario run parameters (shown here with sample values):

scenarioID <- 1000
The scenario ID number. Edit this is you want to run a different scenario (if a new one, remember to copy, paste and rename the data folder).

numModelYears <- 5
Number of 5-year increments calculated in the scenario.

numLandUseLoops <- 2
Number iterations between the res and non-res models. The Metro default is 2.

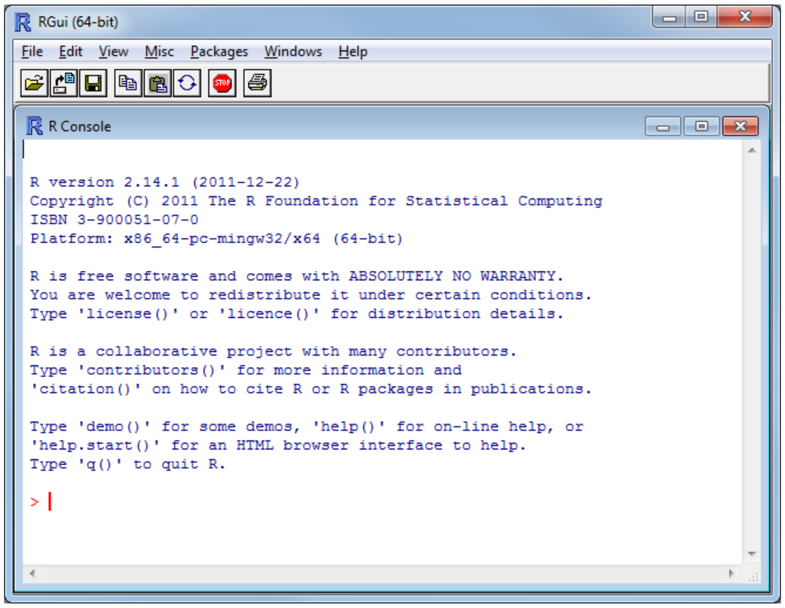
For both the non-residential and residential models, there is no explicit termination condition when supply is sufficiently close to demand, so the number of iterations must be set manually -- experiment with what works best. .

numNonresLoops <- 50
Number of iterations between the non-residential supply and demand modules. The Metro default is 50 .

numResLoops <- 25
Number of iterations between the residential supply and demand modules. Note: The residential demand module is the slowest part of the model, up to one minute per iteration. At Metro, we typically use 25 iterations as the default, but if we want to make sure the model converges as well as it can, we will use as many as 100.

Running MetroScope

Double-clicking on the R shortcut will bring up the R GUI (graphic user interface). You should see something like this:



```
R version 2.14.1 (2011-12-22)
Copyright (C) 2011 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
Platform: x86_64-pc-mingw32/x64 (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.

> |
```

To verify that the working directory is set correctly, at the prompt type “gwd()”

It is usually useful to set the GUI so that it displays each line of output as it is printed, rather than printing all the output at the end. Uncheck the menu item Misc > Buffered Output.

Call the script by selecting the menu item File > Source R code. A dialog box will appear with all the R files in the working folder, then select the one you want. When you click on Open, the screen will start scrolling with text, indicating that MetroScope is now running a scenario.

As the program runs, diagnostic information, such as iteration number and outputs from the various modules, will be output to both the screen and to a log file.

One useful diagnostic tool is “SumSq”, which is the sum of the squares of the difference between supply and demand for each zone, and is an indicator of the module convergence. While the SumSq should ideally decrease with each iteration, sometimes it will either increase or oscillate, due to a few zones which do not converge.

When the model is done running, you will see the “>” prompt again. Ignore any “ODBC” or “NAs introduced by coercion” errors.   An extensive troubleshooting guide is planned for future versions of this document.

When you are done with running MetroScope, close the R GUI. Answer “No” to “Save workspace image?”

MetroScope Results

Outputs from the run will be found in the folder “ScenX_data”, in subfolders “/outputs_year1”, “outputs/ year2”, etc. The output tables are in CSV format, and are listed and defined in the “MetroScope Technical Reference.”

There are many outputs to interpret, but the quickest way to “take the temperature” of the model is to look at residential and non-residential location prices.   The location price is an index of the match between supply and demand. If there is competition for real estate in a given zone, the unmet potential demand is reflected in a higher location price. During model calibration, these are set to an initial value close to 1, and will change over the course of the model run.

Large values (> 3 for residential, > 10 for non-residential), usually indicates that the land supply is starting to fall behind real estate demand. Extremely large values (100+) indicate that something is very wrong, and MetroScope cannot make supply and demand converge for that particular scenario.

For more guidance on how to interpret the scenario outputs, please see “MetroScope Theory and Practice”.

```
#MetroScope_G35_main.R        3/5/2013 3:17:59 PM
#### debug values

scenarioID <- 1000    # scenario ID number

numModelYears <- 5
numLanduseLoops <- 2

numNonresLoops <- 50    # iterations between the supply and demand modules
numResLoops <- 25    # iterations between the supply and demand modules



######################
# global constants, file functions
source("R_modules/G35_globals.R")
source("R_modules/G35_IO_functions.R") #


# nonres functions
source("R_modules/G35_nonres_demand_module.R")  #
source("R_modules/G35_nonres_demand_calcNonresAccess.R") #
source("R_modules/G35_nonres_supply_module.R")
source("R_modules/G35_nonres_supply_calcNonresSupply.R")
source("R_modules/G35_nonres_calcNonresLocationPrice.R")

# res functions
source("R_modules/G35_res_demand_module.R")
source("R_modules/G35_res_demand_functions.R")
source("R_modules/G35_res_supply_module.R")
source("R_modules/G35_res_supply_functions.R")
source("R_modules/G35_res_calcResLocationPrice.R")


######################

runNonresModel <- function(){

    for (iLoops in 1:numNonresLoops){

        # calc nonres demand
        nonresDemandResults <- runNonresDemand(iLoops)

        # update nonres demand: sqftDemandEzRe     needed for supply module
        updateNonresDemand <-
updateNonresDemandData_yearN(nonresDemandResults)

        # calc nonres supply
        nonresSupplyResults <- runNonresSupply(iLoops)

        # calc new location price
        nonresDemandEzRe <- nonresDemandResults$sqftDemandEzRe
        nonresSupplyEzRe <- nonresSupplyResults$sqftVintageSupplyEzRe +
nonresSupplyResults$sqftNewSupplyEzRe +
nonresSupplyResults$sqftNewSupplyEzRe_UR
        nonresLocpriceResults <-
calcNonresLocationPrice(iLoops,nonresSupplyEzRe,nonresDemandEzRe)

        # update location price for next iteration
```

```
      writeTable(scenarioID,cbind(idxEz,nonresLocpriceResults$nonres_location
      price_new),"shared_nonres",paste("nonres_locationprice_year",currentYea
      r,sep=""))

        # status of current iteration
        print(paste("Iter",iLoops,"   SumSq
=",nonresLocpriceResults$totalSumSq))

        # output upon last iteration
        if (iLoops == numNonresLoops){

outputNonresData_yearN(nonresDemandResults,nonresSupplyResults,nonresLocprice
Results)
        }

    }

}

######################

runResModel <- function(){
    for (iLoops in 1:numResLoops){

        # calc res demand
        resDemandResults <- runResDemand(iLoops)

        # pass data to res supply module:   avgHouseSize, avgLotSize,
avgHedonic
        updateResDemandData_yearN(resDemandResults)

        # calc res supply
        resSupplyResults <- runResSupply(iLoops)

        # calc new location price
        resLocpriceResults <- calcResLocationPrice(iLoops,
resSupplyResults$res_supply_Rz, resDemandResults$res_demand_Rz)

        # update location price for next iteration

writeTable(scenarioID,cbind(idxRz,resLocpriceResults$res_locationprice_new),"
shared_res",paste("res_locationprice_year",currentYear,sep=""))

        # status of current iteration
        print(paste("Iter",iLoops,"   SumSq =",resLocpriceResults$totalSumSq,
date() ))

        # output upon last iteration
        if (iLoops == numResLoops){

outputResData_yearN(resDemandResults,resSupplyResults,resLocpriceResults)
         }

    }
}


######################
```

```
date1 <- date()
for (currentYear in 1:numModelYears){
    print(paste("###############   ",currentYear,"   ",date(),sep=""))
    for (iLanduseLoops in 1:numLanduseLoops){
    print(paste("###  current Year = ",currentYear,"   LU loop = 
",iLanduseLoops,"   ",date(),sep=""))

    runNonresModel()
    runResModel()

    }

}
print(date1)
print(date())
```