# RED Priority Bus Lanes Study

RED Lanes Toolkit User Guide (Report 5)

June 2020

CAMPO
NC Capital Area Metropolitan Planning Organization

RENAISSANCE PLANNING

WSP

# CONTENTS

# 1 REVIEW OF OBJECTIVES AND APPROACH

See RED Lanes Evaluation Methodology Report for details

# OBJECTIVES OF THE RED LANES TOOLKIT

For a given location, assign a value that reflects its suitability for RED Lanes, differentiated by travel demand, transportation system operations, and area design/context characteristics.

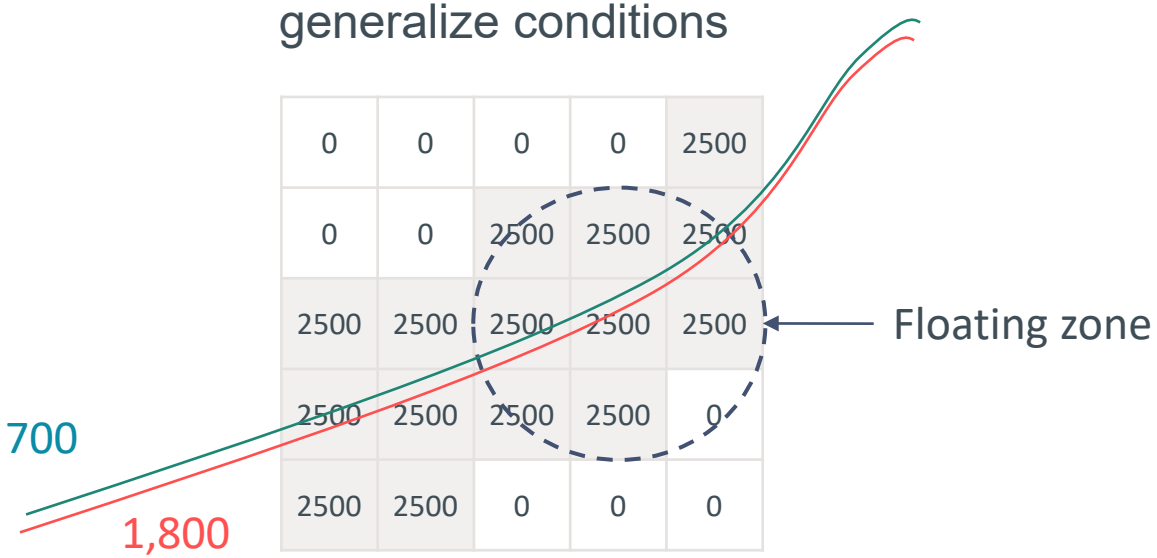
1. Major dimensions of RED Lanes suitability + enrichment elements for detailed differentiation and implementation guidance.
2. Analyze conditions on an “areawide” basis to address inconsistencies in the details of line geometries.
3. Create a consistent, predictable, and replicable process.
   - Facilitate testing of measures
   - Simplify updates to accommodate new/fresh data
   - Allow CAMPO and partner agencies to engage with and revise the RED Lanes Suitability process

# APPROACH – DIMENSIONS

1. Major dimensions of RED Lanes suitability.
    a. Details of data sources, scoring rubrics, processing concepts are available in the RED Lanes Evaluation Methodology Report
    b. Suitability dimensions
        a. Travel demand
        b. Transit operations
        c. Highway operations
        d. Context and design
    c. Enrichment variables
        a. Detailed differentiators – Feasibility and Communities of Concern
        b. Implementation Guidance – Nonmotorized propensity, TSP suitability, full-time suitability

# APPROACH – METHODS FOR MEASURING DIMENSIONS

2. Account for areawide conditions when measuring each dimension.
   a. Utilize spatial analysis to estimate typical conditions in a given area revealed by various linear datasets.
      - Since not all lines are digitized consistently, it is important to consider all lines within a small area to combine measures from diverse datasets.
      - Define "floating zones" as areas for which all available data points will be aggregated to generalize conditions




*The blue line and the red line represent the same facility but have inconsistent GIS representation.*

*The blue line shows 700 transit riders on route A; the red 1,800 riders on route B.*

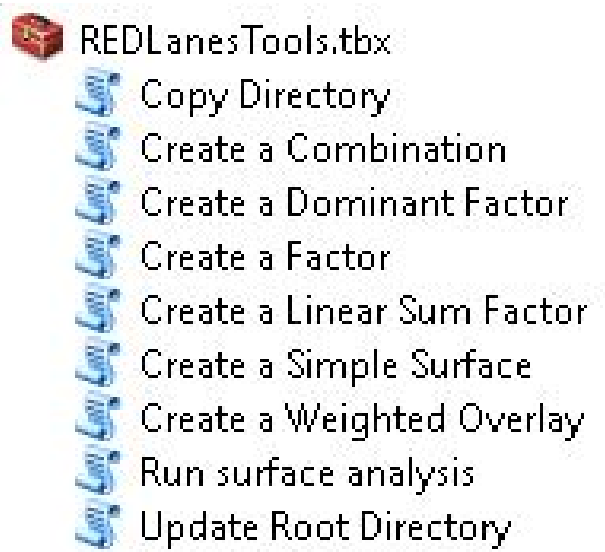
*The total ridership within the floating zone is… 2,500.*

# APPROACH – STREAMLINING PROCESSES

3. Create a consistent, predictable, and reliable process.
   a. Utilize standard geo-processing tools to develop measures.
      - ArcGIS's Spatial Analyst extension
   b. Develop scripted process to sequence geo-processing tasks and minimize the effort required to (re)run, modify, and update suitability estimates
      - Python (arcpy)
   - Provide a simple interface for ease of use
      - ArcMap geoprocessing script interfaces

# TOOLKIT OVERVIEW

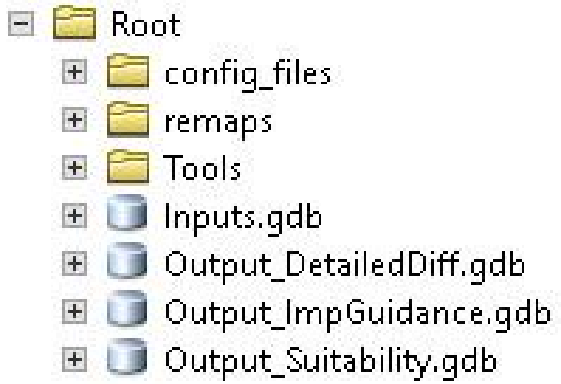
- The evaluation objectives are achieved through an ArcGIS-based Python toolkit
- The toolkit consists of several geoprocessing tools, most of which focus on developing configuration files (.json format) that guide spatial analysis procedures.
- Some tools are used for data transfer and version management.



# 2 DATA AND WORKSPACE PREPARATION

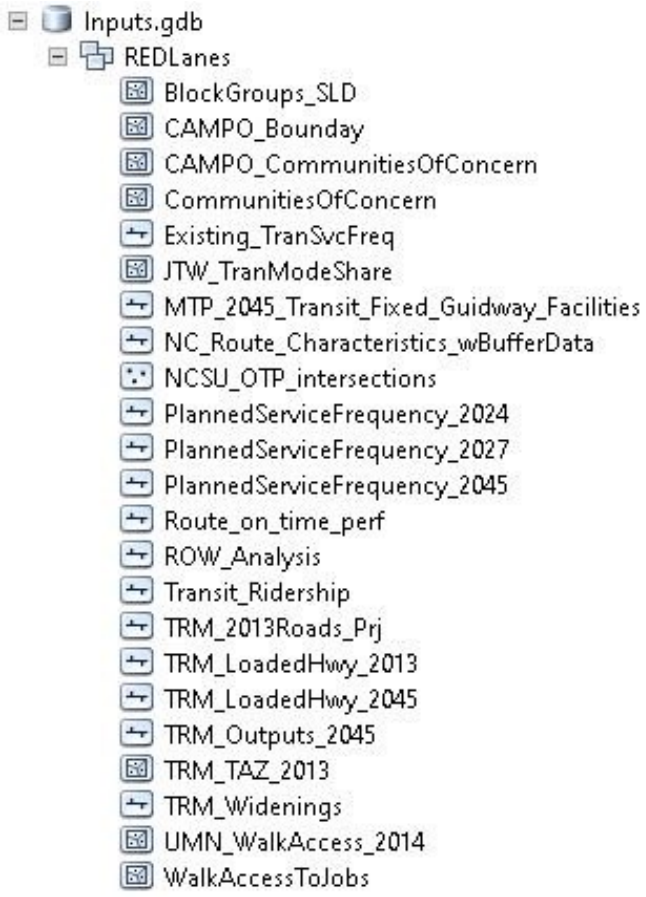
# WORKSPACE ORGANIZATION – EVERYTHING IN ONE ROOT

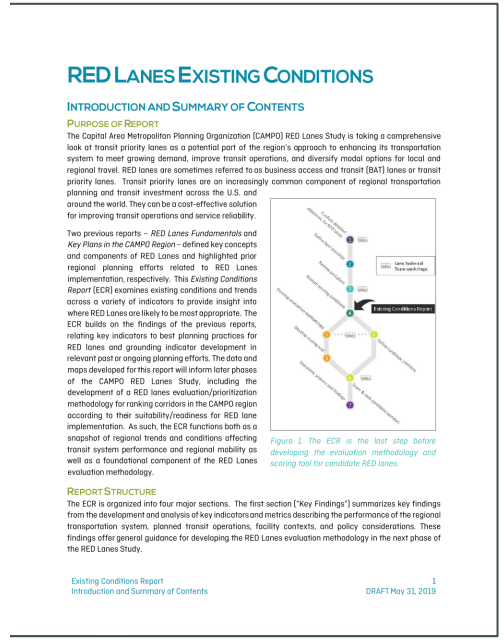
- {Root directory}
  - Configuration files
  - Inputs geodatabase
  - Output geodatabases
    - Suitability
    - Detailed Differentiators
    - Implementation Guidance
  - Remaps
    - Info table with remap files for loading raster classification details
  - Tools



# WORKSPACE ORGANIZATION – “INPUTS" GEODATABASE

- Inputs geodatabase
  - Contains a single feature dataset (“REDLanes”) using the NC State Plane coordinate system (WKID: 103122)
  - All input datasets for the RED Lanes toolkit have been imported to “REDLanes”, ensuing consistent projection.
  - “REDLanes” also includes a feature class of the CAMPO boundary. This is used to ensure consistent processing extents when running the “Run Surface Analysis” tool.
- Existing Conditions Report
  - Provides background information on raw data sources, analysis metrics, and steps taken to prepare the data to be used in the RED Lanes evaluation process.



RED LANES EXISTING CONDITIONS

INTRODUCTION AND SUMMARY OF CONTENTS

PURPOSE OF REPORT

The Capital Area Metropolitan Planning Organization (CAMPO) RED Lanes Study is taking a comprehensive look at transit priority lanes as a potential part of the region's approach to enhancing its transportation system to meet growing demand, improve transit operations, and diversify modal options for local and regional travel. RED lanes are sometimes referred to as business access and transit (BAT) lanes or transit priority lanes. Transit priority lanes are an increasingly common component of regional transportation planning and transit investment across the U.S. and around the world. They can be a cost-effective solution for improving transit operations and service reliability.

Two previous reports – *RED Lanes Fundamentals* and *Key Plans in the CAMPO Region* – defined key concepts and components of RED Lanes and highlighted prior regional planning efforts related to RED Lanes implementation, respectively. This *Existing Conditions Report* (ECR) examines existing conditions and trends across a variety of indicators to provide insight into where RED Lanes are likely to be most appropriate. The ECR builds on the findings of the previous reports, relating key indicators to best planning practices for RED lanes and grounding indicator development in relevant past or ongoing planning efforts. The data and maps developed for this report will inform later phases of the CAMPO RED Lanes Study, including the development of a RED lanes evaluation/prioritization methodology for ranking corridors in the CAMPO region according to their suitability/readiness for RED lane implementation. As such, the ECR functions both as a snapshot of regional trends and conditions affecting transit system performance and regional mobility as well as a foundational component of the RED Lanes evaluation methodology.

*Figure 1. The ECR is the last step before developing the evaluation methodology and scoring tool for candidate RED lanes.*

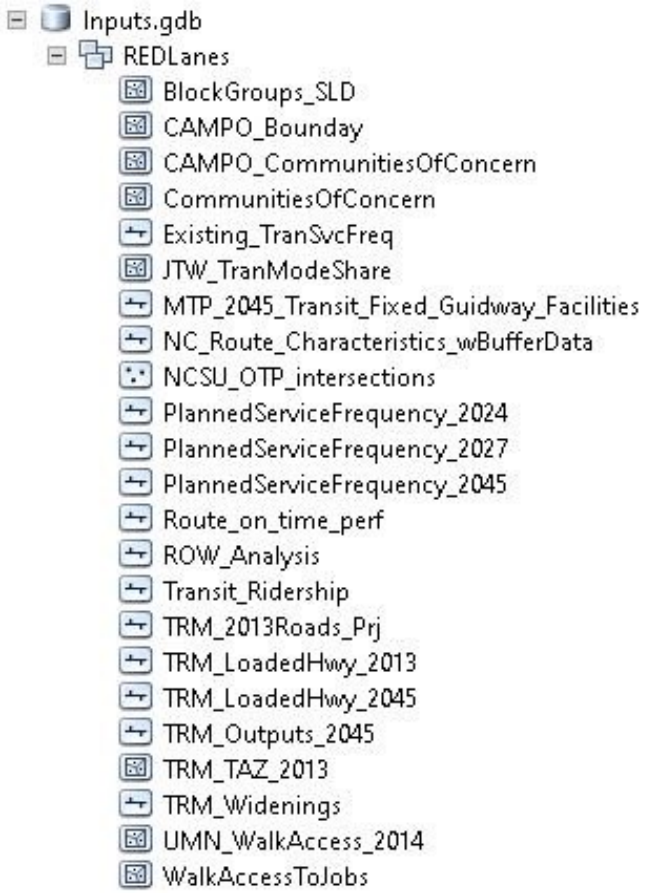
REPORT STRUCTURE

The ECR is organized into four major sections. The first section ("Key Findings") summarizes key findings from the development and analysis of key indicators and metrics describing the performance of the regional transportation system, planned transit operations, facility contexts, and policy considerations. These findings offer general guidance for developing the RED Lanes evaluation methodology in the next phase of the RED Lanes Study.

Existing Conditions Report
Introduction and Summary of Contents
1
DRAFT May 31, 2019

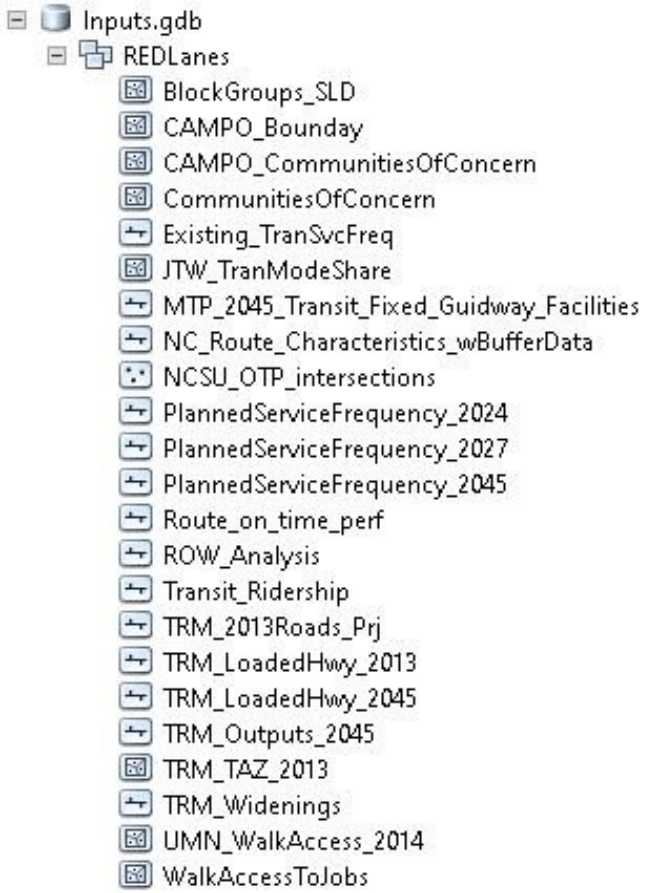
# INPUTS DETAILS

- Input geodatabase
  - BlockGroups_SLD
    - Source: EPA Smart Location Database extract
    - Use: intersection density (field=D3b)
  - CAMPO_Boundary:
    - Source: CAMPO
    - Use: set consistent processing extents for all surfaces
  - CAMPO_CommunitiesOfConcern
    - Source: CAMPO
    - Use: number of communities of concern served (field=overlap_count)
  - Existing_TranSvcFreq
    - Source: Wake Bus Plan GIS files
    - Use: existing number of buses per hour on each segment during peak (field=BusPerHrPk).
  - MTP_2045_Transit_Fixed_Guideway_Facilities
    - Source: CAMPO
    - Use: masking suitability results for corridors with fixed guideway ongoing studies



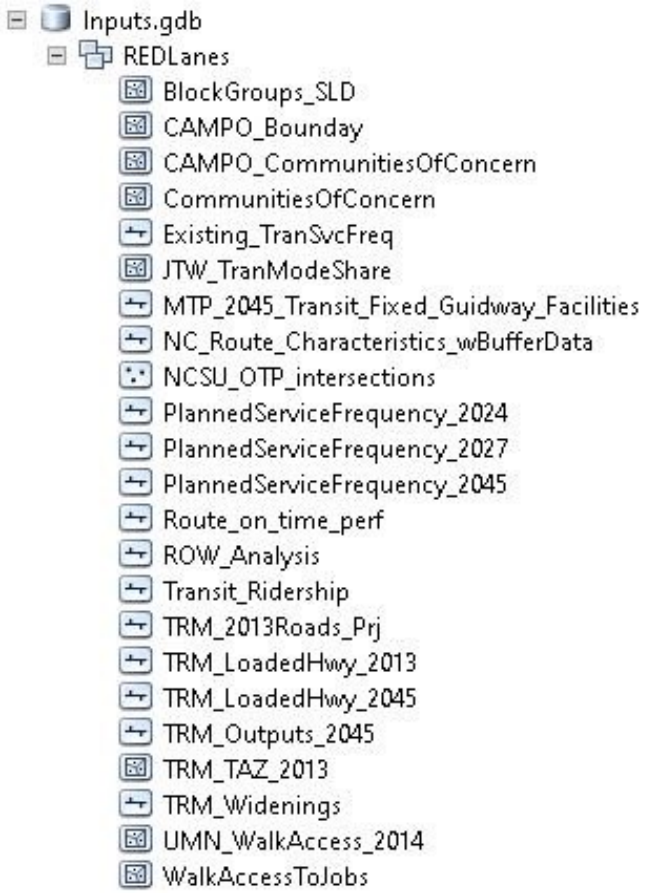
# INPUTS DETAILS (cont.)

- Input geodatabase
  - NCSU_OTP_intersections
    - Source: generated as part of the RED Lanes study based on input from NCSU Wolfline staff
    - Use: Highlight intersections that cause on-time performance issues for Wolfline buses.
  - PlannedServiceFrequency_{year}
    - Source: Wake Bus Plan GIS files, MTP
    - Use: number of buses per hour on each segment during peak in the named year (field=BusPerHrPk).
  - Route_on_time_perf
    - Source: generated as part of RED Lanes study based on transit agency route shape files and on-time performance tables.
    - Use: Route-level on-time performance rates (field=Pct_OnTime)
  - ROW Analysis
    - Source: generated as part of RED Lanes study based on NCDOT route characteristics shape file and Microsoft Building Footprints database.
    - Use: ROW analysis for feasibility ranking (field=bld_pr_mi)



# INPUTS DETAILS (cont.)

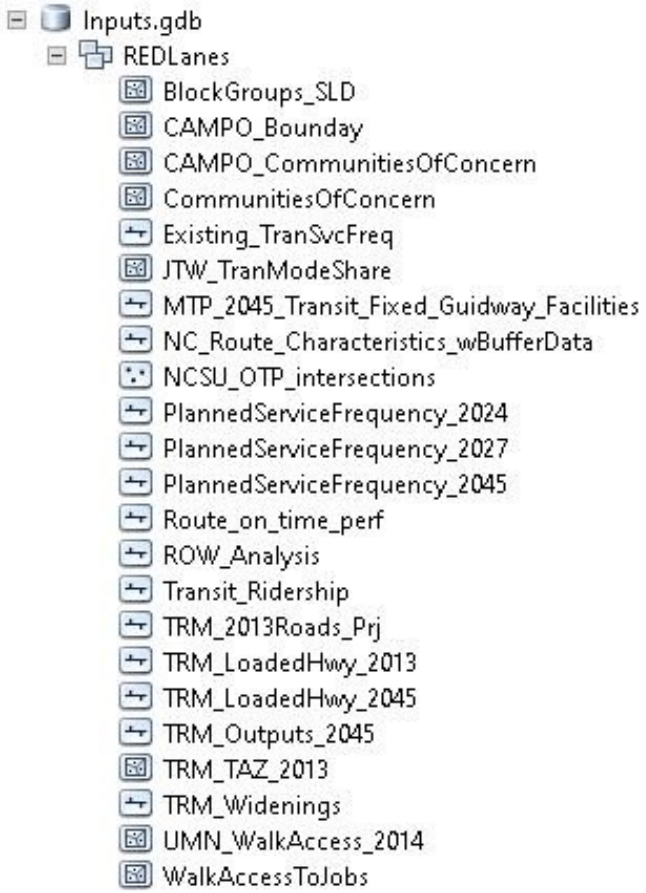
- Input geodatabase
  - Transit Ridership
    - Source: Triangle Regional Model
    - Use: Route-level peak and daily ridership forecasts in 2045 (fields=DAILY_RIDERS, PK_SHR_R)
  - TRM_2013Roads_prj
    - Source: Triangle Regional Model
    - Use: Number of lanes data for feasibility ranking (field=LANESDIR)
  - TRM_LoadedHwy_2045
    - Source: Triangle Regional Model (NCSU segment flag added manually as part of RED Lanes study based on input from NCSU Wolfline staff)
    - Use: Traffic volume (TOTDLYVOL), bus speed (MIN_PK_BUS_SPD), vehicle delay (MIN_PM_CFF_SPND), v/c ratio (MAX_PM_VC), segments that routinely pose on-time performance challenges for Wolfline routes (NCSU_OTP)
  - TRM_Outputs_2045
    - Source: Triangle Regional Model
    - Use: Peak-hour volume shares for full-time-suitability ranking (field=PM_SHARE)



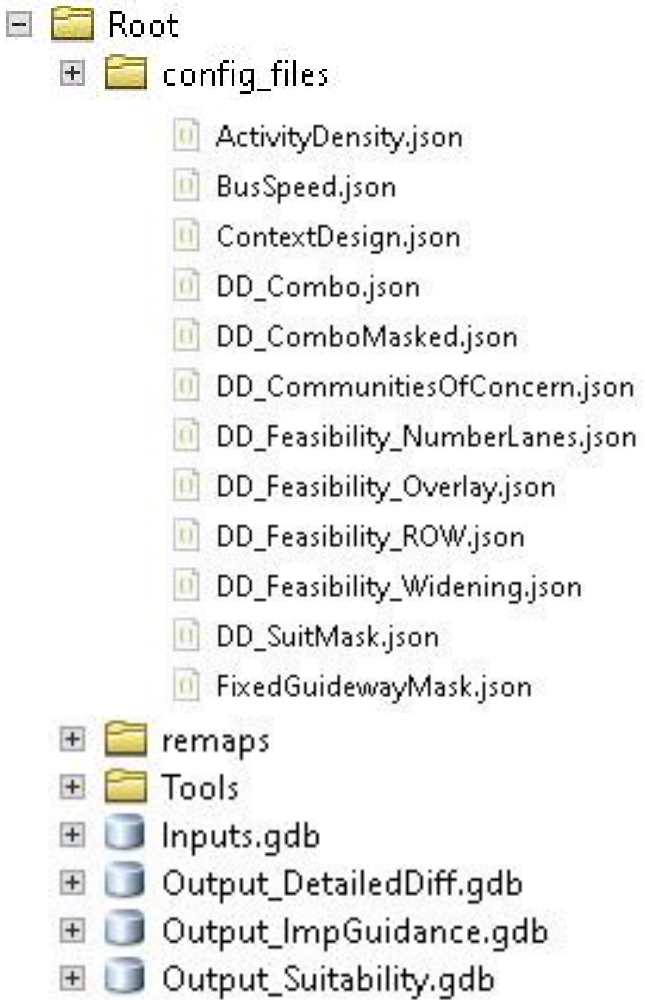

*Note: multiple extracts of TRM data were used throughout the development of RED Lanes evaluation process. It is likely the many feature classes listed here could be consolidated in a smaller number of extracts.*

# INPUTS DETAILS (cont.)

- Input geodatabase
  - TRM_TAZ_2013
    - Source: Triangle Regional Model
    - Use: Activity-unit density (field=AU_DENSITY)
  - UMN_WalkAccess_2014
    - Source: University of Minnesota Accessibility Observatory
    - Use: Walk access to jobs for nonmotorized propensity ranking (field=JT_LONG)



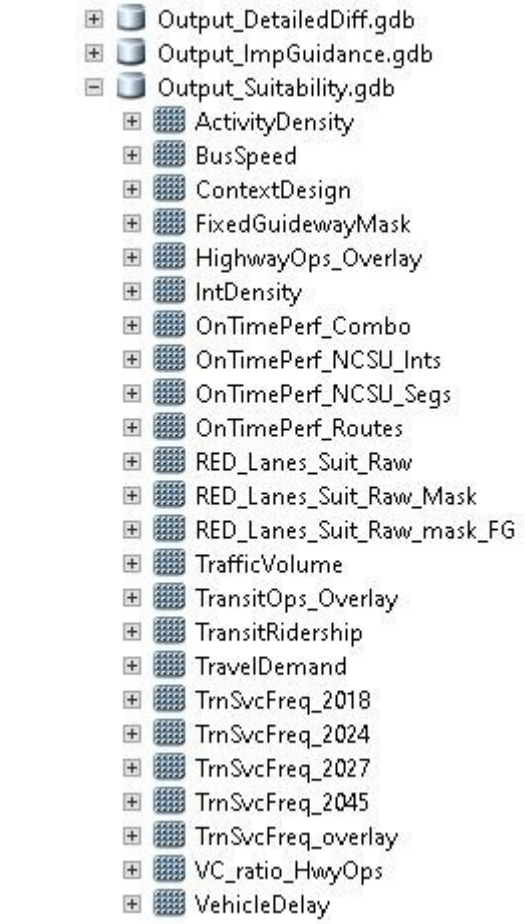
# WORKSPACE ORGANIZATION – CONFIGURATION FILES




- Configuration files store information about surface objects:
  - Where source data are stored (the inputs geodatabase, e.g.)
  - Dependencies on other surface objects (an overlay that depends on two factors, e.g.)
  - Processing parameters and reclassification specifications
- Use the "**Run Surface Analysis**" tool to create the resulting raster for the specified surface configuration (.json) file as well as all prerequisite files. (Warning! All existing files in the output geodatabase are deleted when this tool is run.)

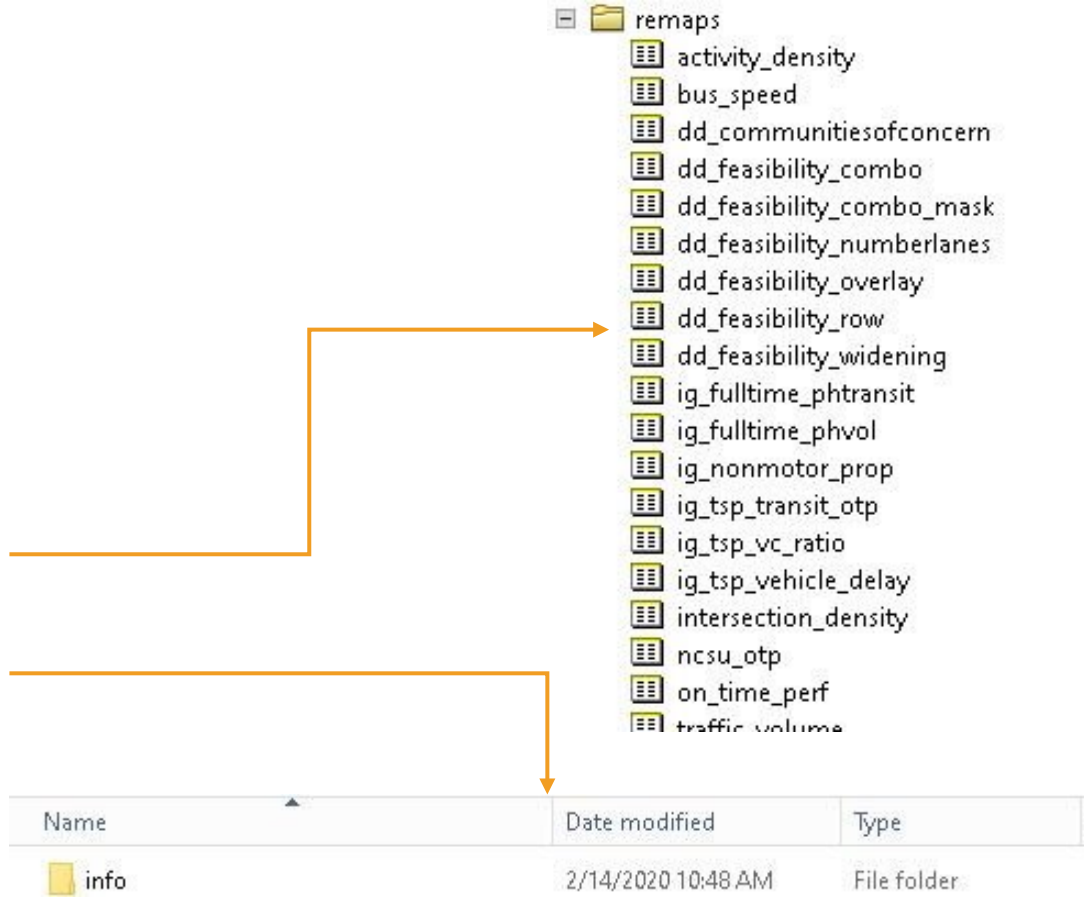
# WORKSPACE ORGANIZATION – OUTPUTS

- Output surfaces must be written to a geodatabase
- There are three separate output geodatabases for the RED Lanes evaluation process:
  - Output_Suitability: contains all rasters pertaining to RED Lanes Suitability (example to right)
  - Output_DetailedDiff: contains all rasters pertaining to the development of Detailed Differentiator measures
  - Output_ImpGuidance: contains all rasters pertaining to the development of Implementation Guidance measures



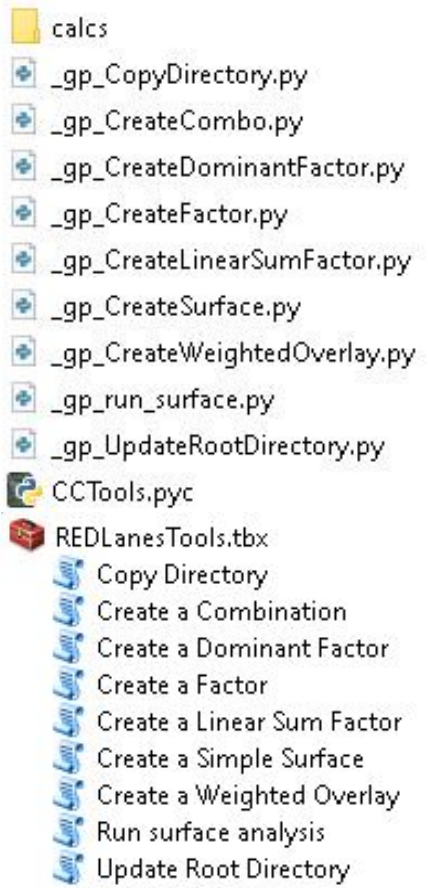
# WORKSPACE ORGANIZATION – REMAPS

- A key component of the evaluation process and each configuration file is the potential need to reclassify rasters. For example, continuous-value estimates of transit ridership by route are classified into 10 ordinal RED Lanes suitability scores.
- Reclassification details can be saved to/loaded from an ArcGIS INFO table. The remaps folder contains the INFO table and a collection of reclassification subtables.
  - In ArcCatalog, these appear as tables within the remaps folder.
  - In the file system, these appear as a folder called “info” with a collection of files inside it.
- These simplify the process of reviewing and updating configuration files and will be discussed further in the next section.



# WORKSPACE ORGANIZATION – TOOLS

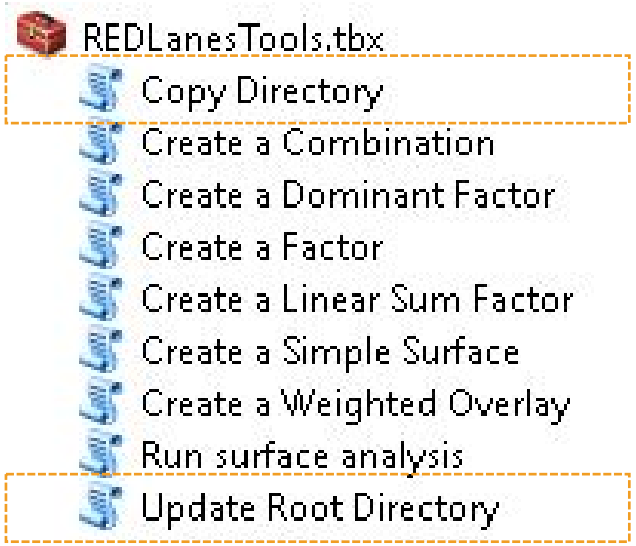
- The tools directory contains the RED Lanes toolbox and supporting resources, including
  - Calcs folder – contains calculation expressions for use in ArcGIS field calculation. These support input data preparation (processing native TRM fields to populate a user-added field, e.g.).
  - Python scripts – the scripts that power the toolbox. Users do not need to open, edit, or run these scripts directly and are discouraged from doing so.



# MANAGING AND SHARING WORKSPACES

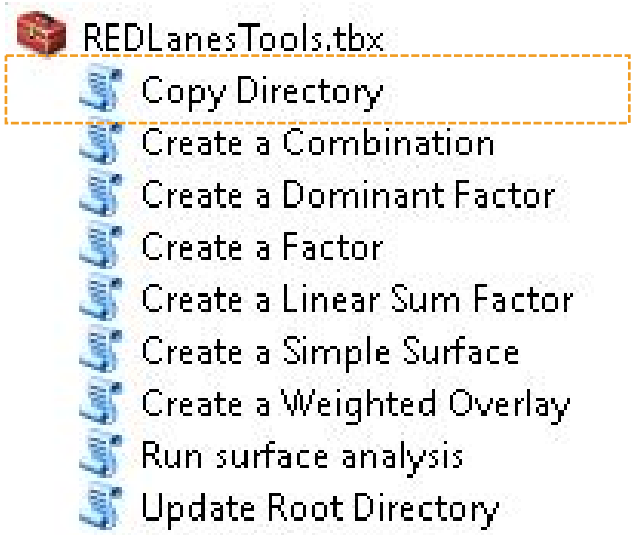
Config files contain full path references to input datasets and other config files. For this reason, moving and copying files to other root directories should be done using the RED Lanes toolbox:

- Use the **Copy Directory** tool to handle process versioning within the same root directory.
- Use the **Update Root Directory** tool when moving or replicating the process across different root directories.



# USING THE “COPY DIRECTORY” TOOL

- The process of setting up the entire set of surface configuration files can be onerous. To simplify setup for alternative versions/vintages/scenarios within the same root directory, use the “Copy Directory” tool.
- The tool copies configuration files and resets each json’s path.
- Optionally, a “reference workspace” can be reset as well. This can be the root directory or a subdirectory (like an alternative “inputs” geodatabase, e.g.).
  - Use this option if copying a configuration while linking inputs to a different input geodatabase.
  - If making a copy simply to test alternative analysis parameters (but not different input data), this option is not needed.



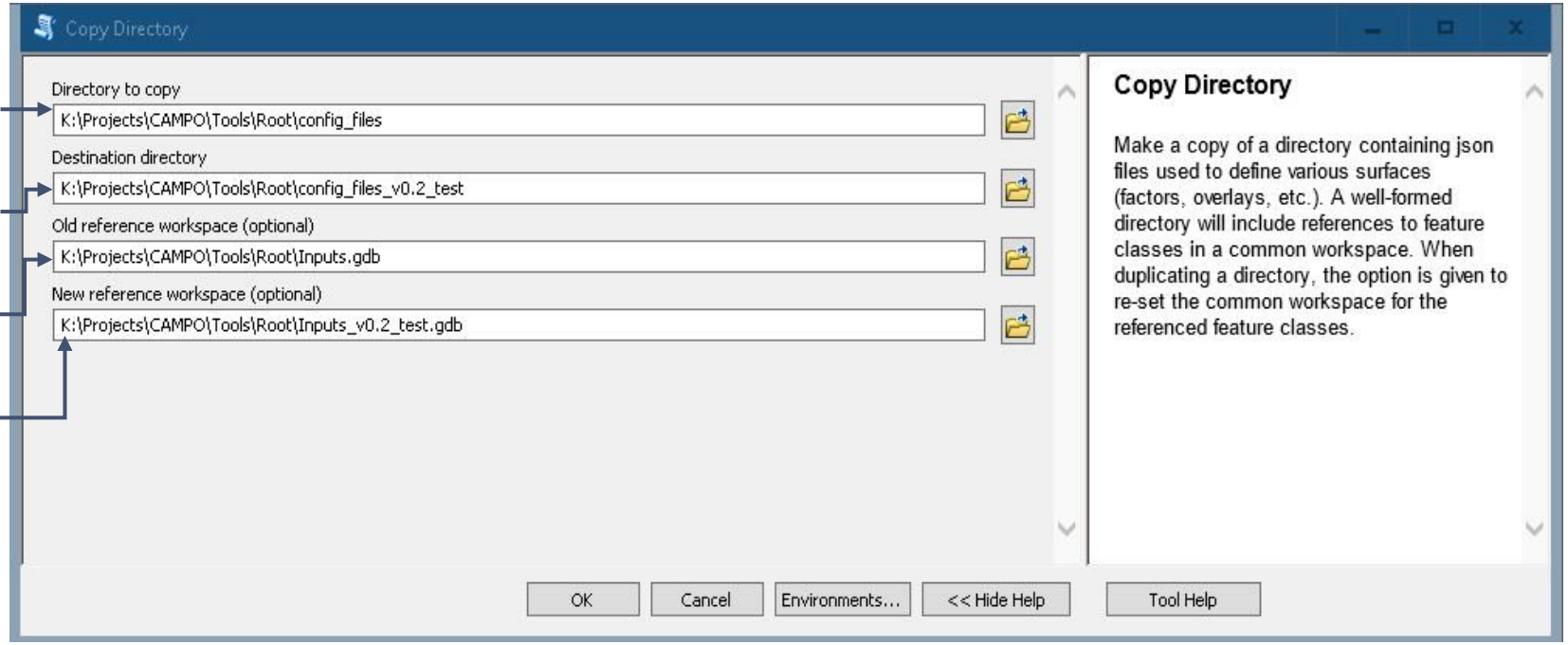
# USING THE “COPY DIRECTORY” TOOL

Directory to copy {config_dir}

Destination directory: {new_config_dir}

Old reference workspace: {input_gdb}

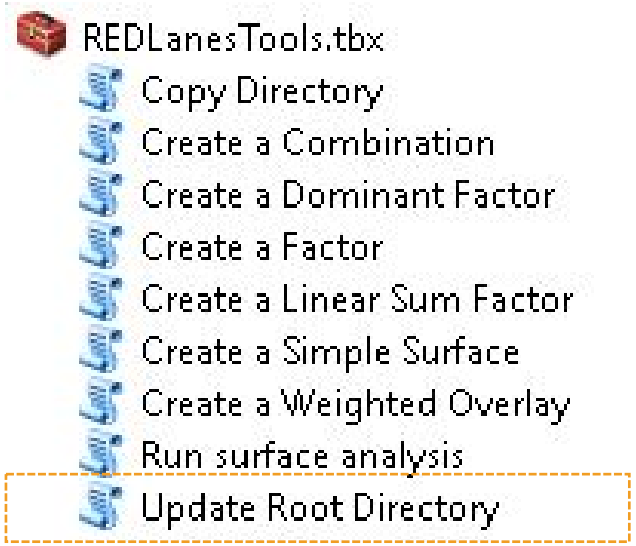
New reference workspace: {alt_input_gdb}




- *The “Copy Directory” tool will generate copies of config files in the “Directory to Copy” within the “Destination Directory.”*
- *In this example, the new files will need to refer to an alternative set of inputs (perhaps data updates or an alternative scenario), so the “Old Reference Workspace” and “New Reference Workspace” fields identify that previous references to “Inputs.gdb” should now point to “Inputs_v0.2_test.gdb.” If these fields are blank, the new config files will continue to reference input data from the origin “Inputs.gdb”*

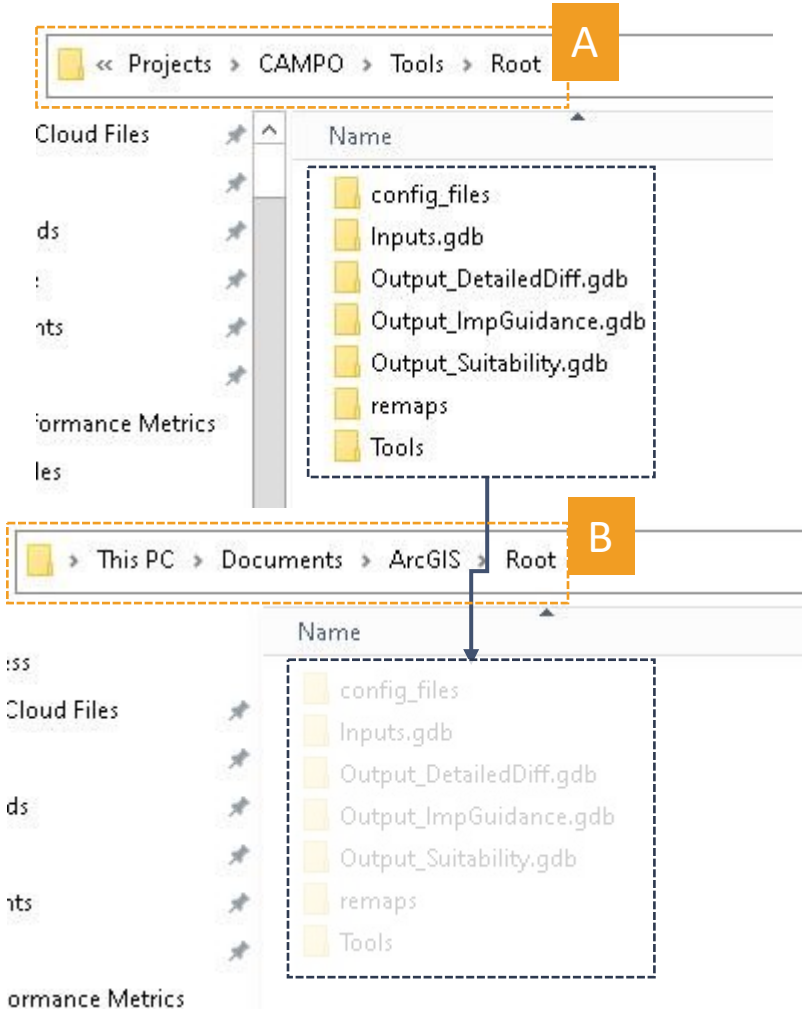
# USING THE “UPDATE ROOT DIRECTORY” TOOL

- Migrating data and configuration files to a new root directory (to a new server, e.g.) requires maintaining a consistent file structure and updating the path to the root directory. The “Update Root Directory” tool simplifies this process.
- Procedure:
  1. Copy the existing root directory and all sub-folders (including input data and configuration directories) to the new root directory.
  2. Copy the path of the old root directory as the “old root directory” input into the tool dialog.



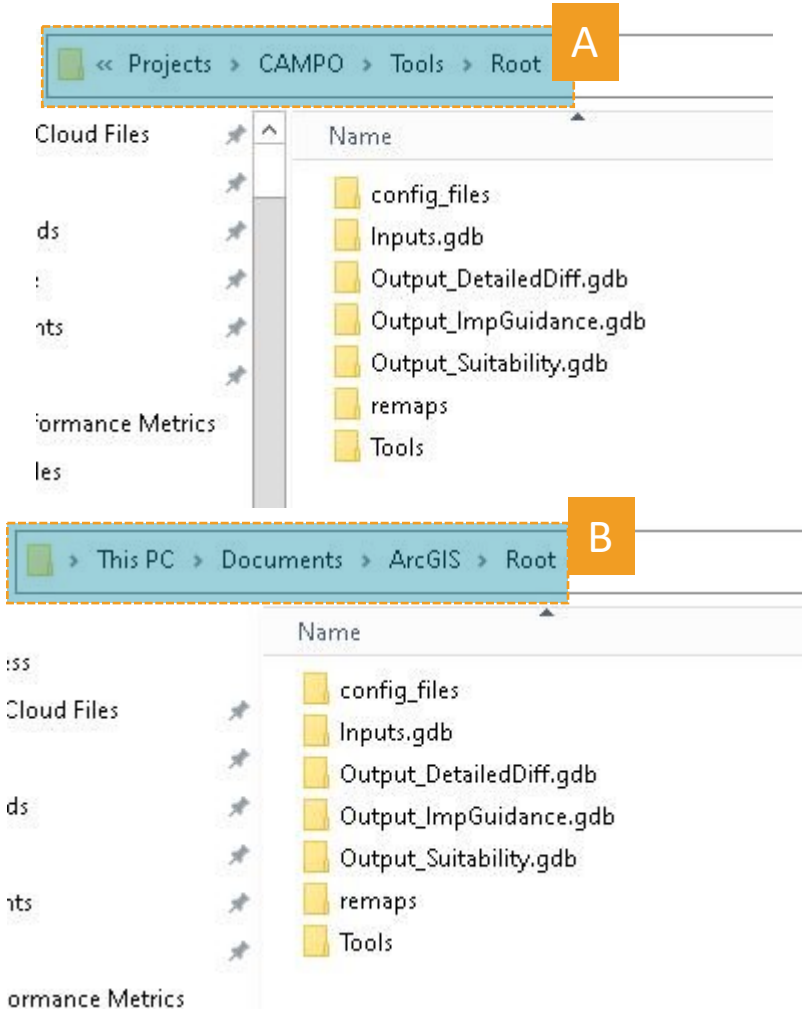
# USING THE “UPDATE ROOT DIRECTORY” TOOL

- Migrating data and configuration files across folders or servers requires maintaining a consistent file structure and updating the path to the “root directory” (see “Organization of Data” slide). The “update root directory” tool helps simplify this process.
- Procedure:
  1. Using the file system, copy the existing root directory (A) and all sub-folders (including input data and configuration directories) to the new root directory (B).

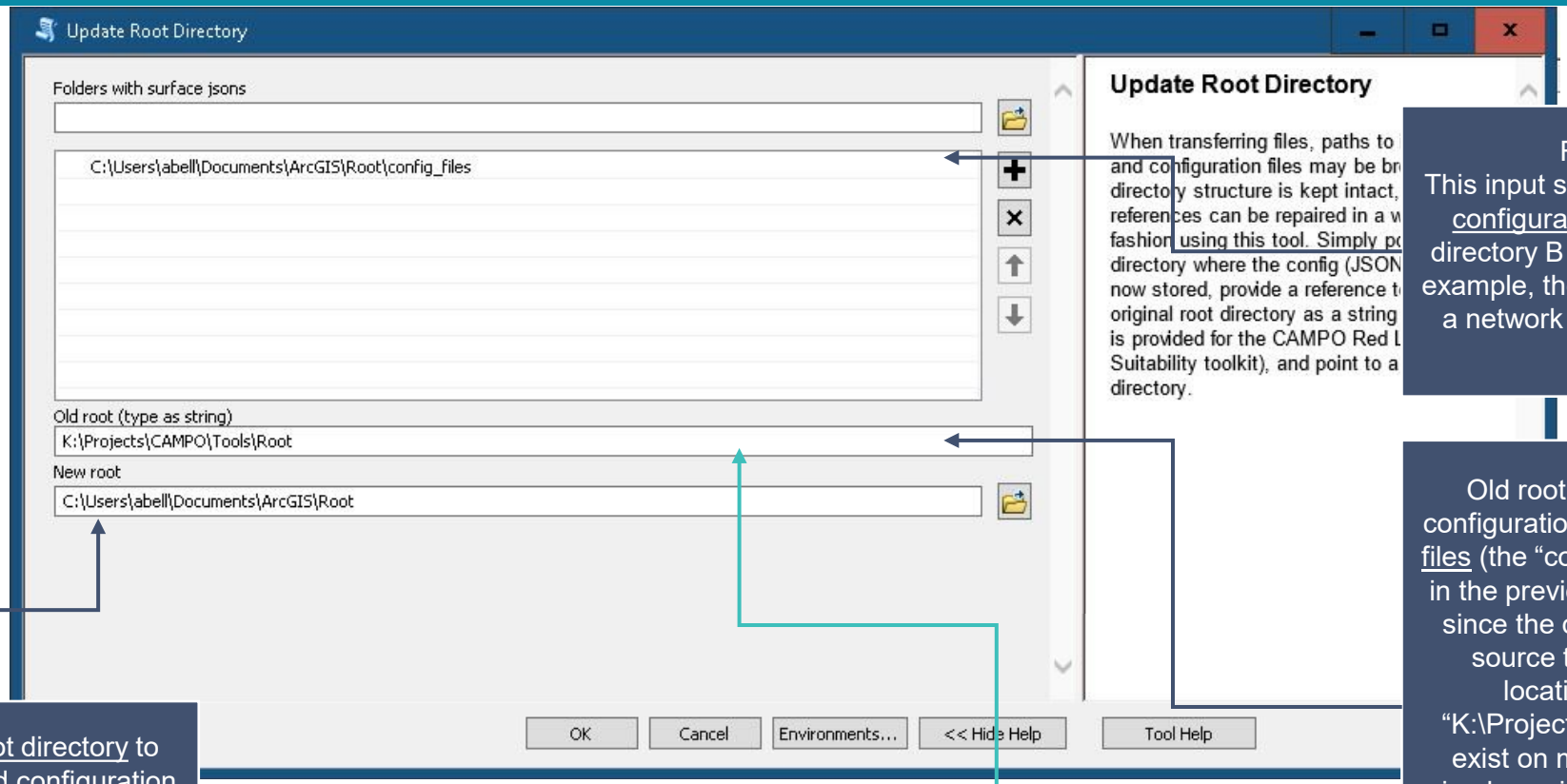


# USING THE “UPDATE ROOT DIRECTORY” TOOL

- Migrating data and configuration files across folders or servers requires maintaining a consistent file structure and updating the path to the “root directory” (see “Organization of Data” slide). The “update root directory” tool helps simplify this process.
- Procedure:
  2. Copy the path of the old root directory (A) and paste it as the “old root directory” input into the tool dialog (see next slide).
  3. Copy the path of the new root directory (B) and paste it as the “new root directory” input into the tool dialog (see next slide).



# USING THE “UPDATE ROOT DIRECTORY” TOOL




Folders with surface jsons:
This input should point to any folders with copied configuration files (the “config_files” folder in directory B shown in the previous slides). In this example, the root directory has been copied from a network drive (K:\Projects…) to a local drive (C:\Users…).

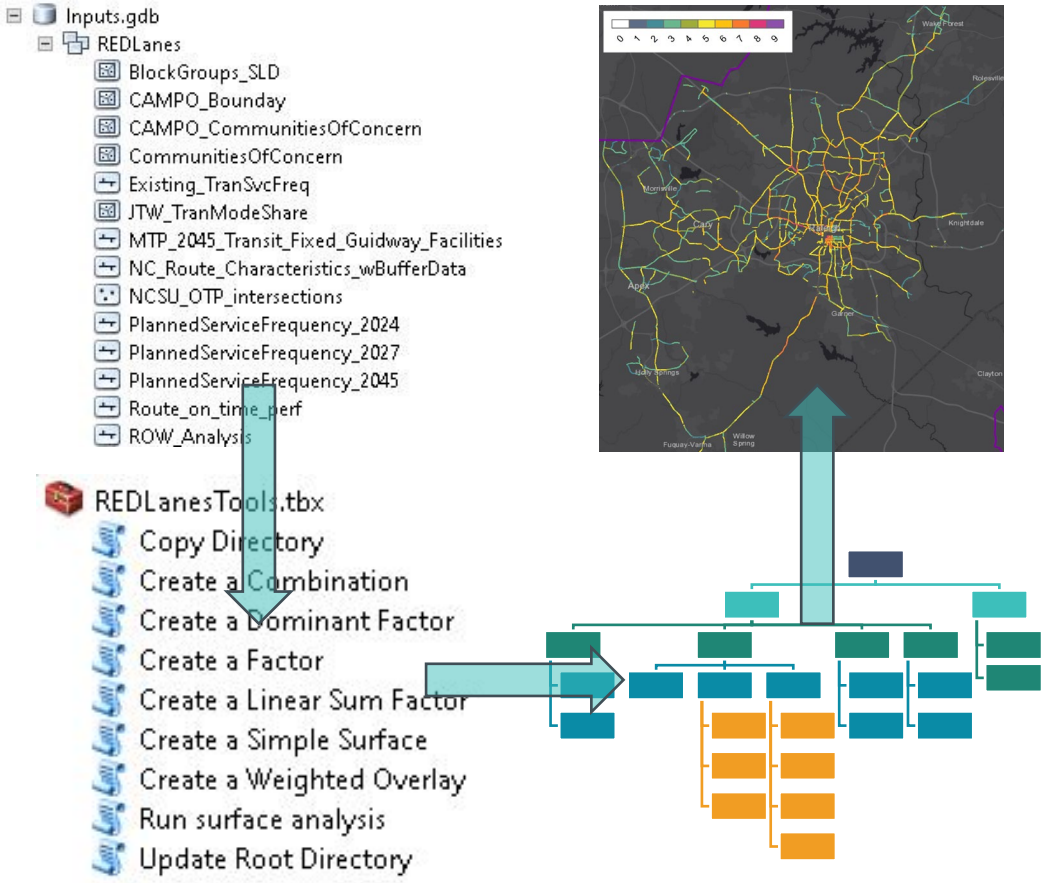
Old root: The original root directory for the configuration files in the folder(s) specified above files (the “config_files” folder in directory A shown in the previous slides). The input takes a string, since the data may be shared from an original source that is not accessible from its new location. For example, the directory “K:\Projects\CAMPO\Tools\Root” is unlikely to exist on most computers, so the user cannot simply navigate to that folder. Nevertheless that directory is still referenced by the copied json configuration files and needs to be searched for and revised to match the “new root” input (next field).

New root: The new root directory to which the input data and configuration files have been copied. References in the configuration files will be updated to point to this new location.

“K:\Projects\CAMPO\Tools\Root” is the default value in the tool interface since this is the original directory of the RED Lanes suitability config files and input data.

# 3 GEOPROCESSING TOOLKIT INTERFACES

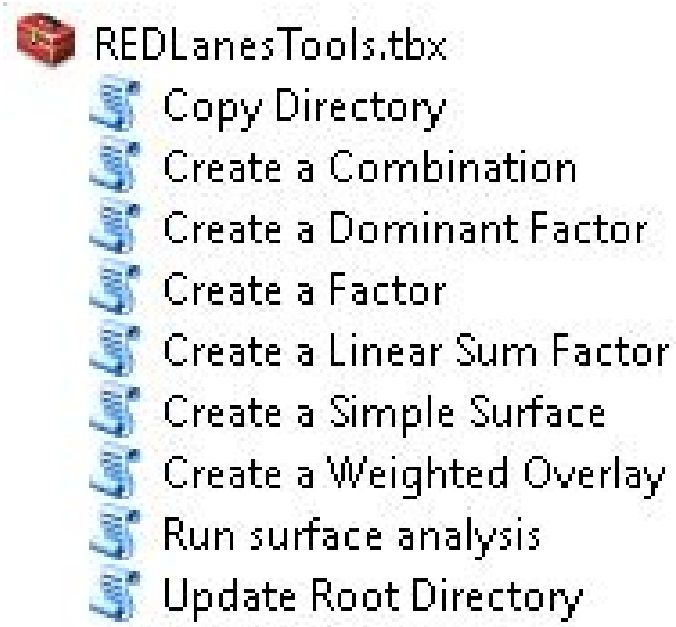
# SECTION OVERVIEW




*This section explains the tools in the RED Lanes toolbox and provides a walkthrough using the tool interfaces to configure, run, and manage all aspects of the RED Lanes evaluation process.*

- Organization of data inputs, configuration information, and outputs simplify the process (see "Data and Workspace Preparation" section above).
- ArcGIS Toolbox designed to facilitate creation and management of hierarchically-related metrics (surfaces).
- See "RED Lanes Evaluation Methodology" document for explanation of measures and general approach.

# RED LANES TOOLBOX



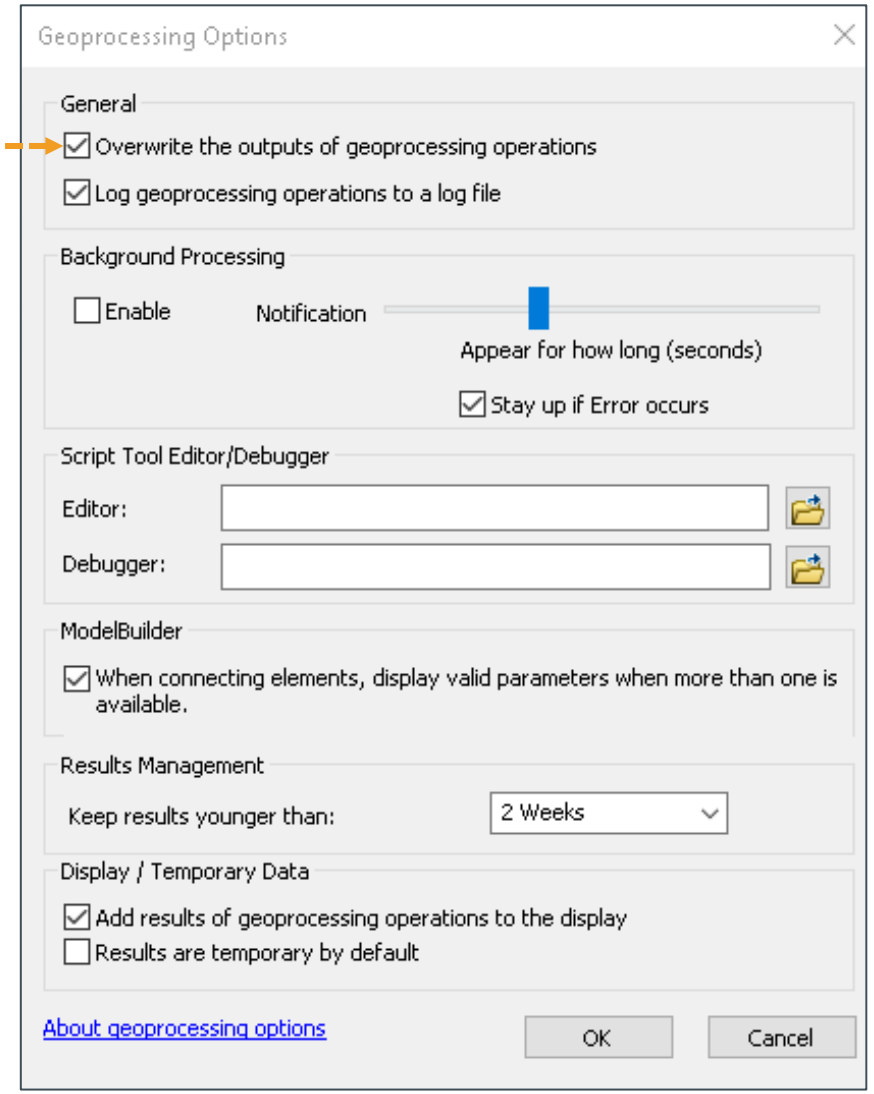

TOOLS

- Create surface object configuration files (.json format)
  - Simple surface
  - Factor
  - Dominant Factor
  - Linear Sum Factor
  - Weighted Overlay
  - Combination
- Copy a directory of configuration files*
- Update the root directory when moving an entire set of configuration files and input data to a new location*
- "Run surface analysis" – using a specified configuration file, create a raster output based on the chosen surface and all prerequisite surfaces

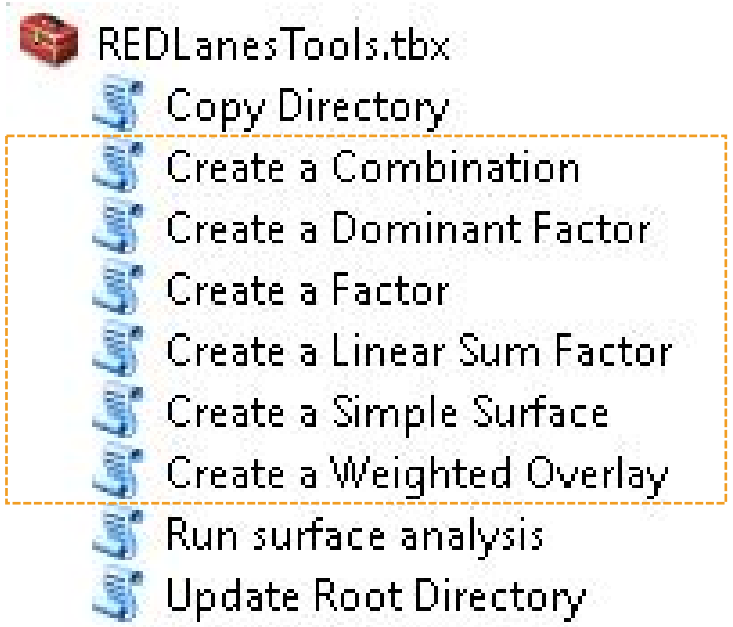
**See "Managing and Sharing Workspaces" in the previous section for more information on the use of each tool.*

# RED LANES TOOLBOX – TIPS

- You may need to run ArcGIS as Administrator or work on a local drive rather than a network drive since many of the tools require read/write permissions.
- The tools that create surface object configuration files work best when the option to overwrite geoprocessing outputs is enabled.
  - In ArcMap, the “Geoprocessing Options” dialog can be found in the main window’s menu bar under “Geoprocessing” >> “Geoprocessing Options…”



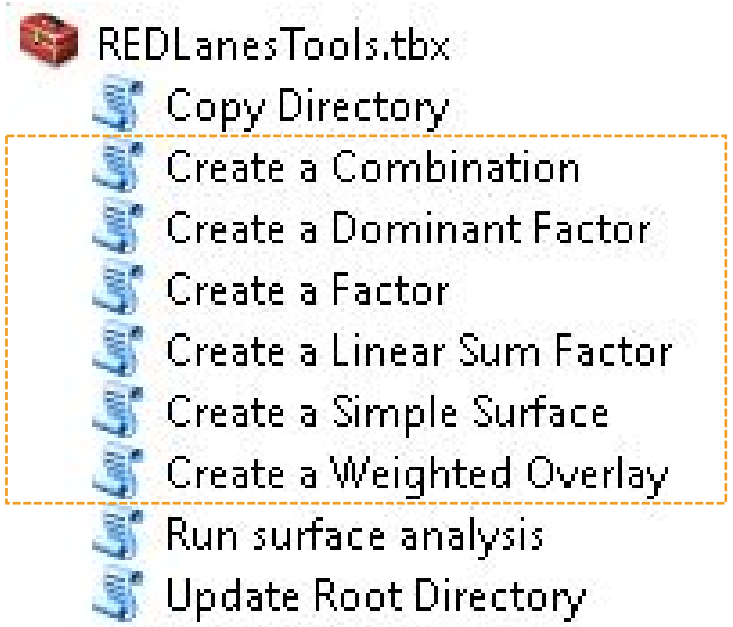
# RED LANES TOOLBOX – SURFACE TYPES




TOOLS

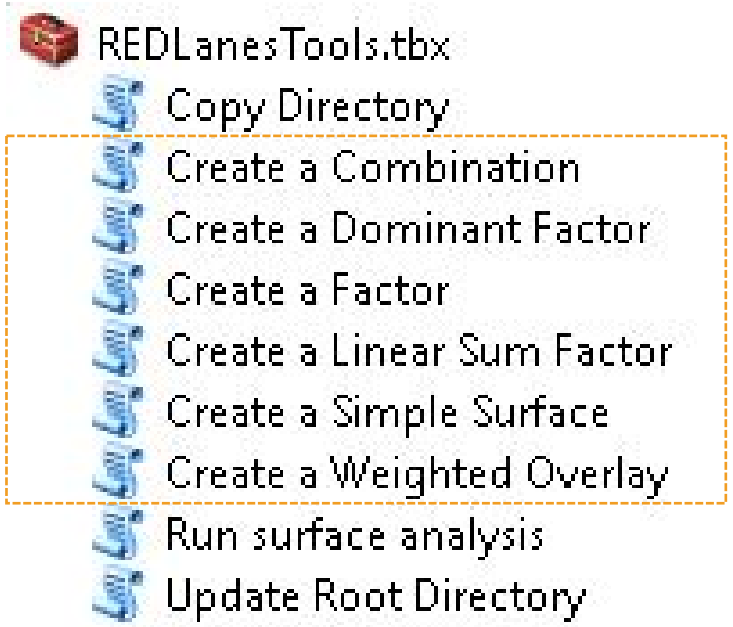
- Different “surface” types:
  - **Simple surface** – Uses an existing raster
  - **Factor** – simple rasterization of vector data
    - **Dominant Factor** – Uses grouping and weight fields to generate a raster containing the indices of the dominant group
    - **Linear Sum Factor** – Simple summation of attribute values of linear features.
  - **Weighted overlay** – weighted averaging of overlapping surface values.
  - **Combination** – combine overlapping surface values to calculate a new value.
- See “PYTHON TOOLKIT DOCUMENTATION” section for details of each surface type.

# COMMON ELEMENTS OF SURFACE CREATION




- Details of objects are stored in .json files ("JSON file" field in script tool dialogs) for easy updates and processing
- Description field offers an opportunity to give the surface object a brief description that might be easier to understand than the .json name itself
- "Remap groups" can be specified to automate reclassification of resulting rasters as needed.
  - See RED Lanes Evaluation Methodology Report for threshold details
  - See "PYTHON TOOLKIT DOCUMENTATION" section for illustrations of raster reclassification

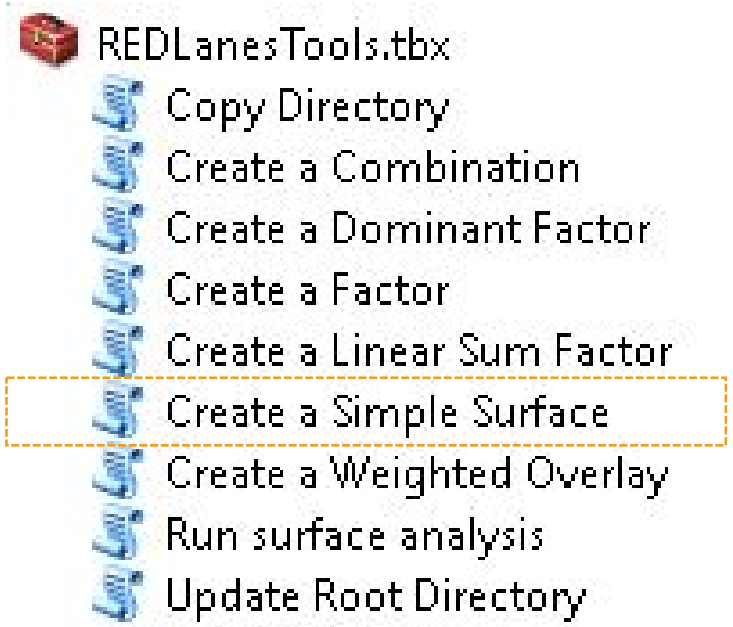
# COMMON ELEMENTS OF SURFACE CREATION




- "No data value" specifies how to reclassify any parts of the resulting raster that are missing data (see "PYTHON TOOLKIT DOCUMENTATION" section).
  - For many factors, the No Data Value will be set to 0 or 1, indicating that if no data are present in the resulting raster, there is no suitability or very low suitability.
  - For adjustments, the No Data Value will generally be set to 0 (zero), indicating that no adjustment should be made in areas where no data are present in the resulting raster
- "Keep unmapped values" specifies what to do with values that fall outside the ranges specified in the remap groups. (Note: It is rare to leave any unclassified values, so usually this option has no bearing on the output raster.)
  - If True, unmapped values will be retained during reclassification
  - If False, unmapped values will be converted to "NO DATA" during reclassification and reclassified based on the No Data Value.

# SIMPLE SURFACE INPUTS

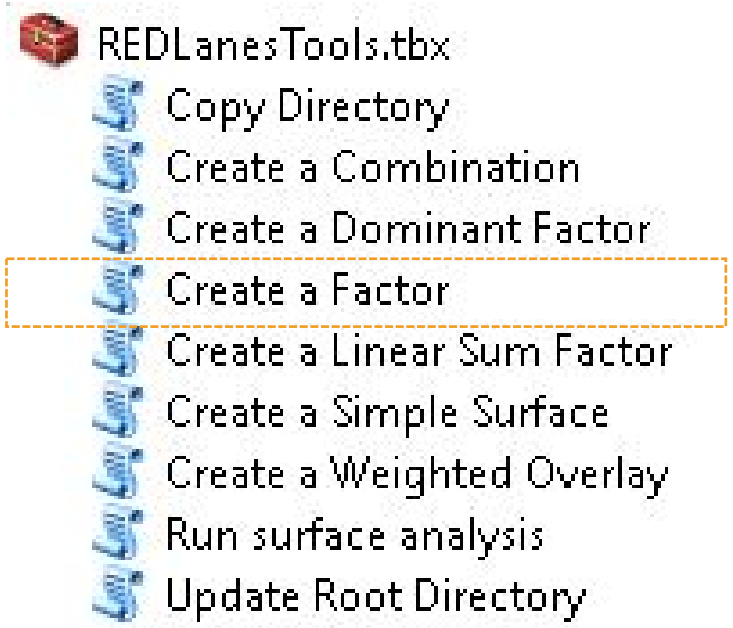
*Simple surfaces record the location of existing raster data for use in downstream analyses (see "PYTHON TOOLKIT DOCUMENTATION" section).*




- Raster – the path to an existing raster dataset

# FACTOR INPUTS

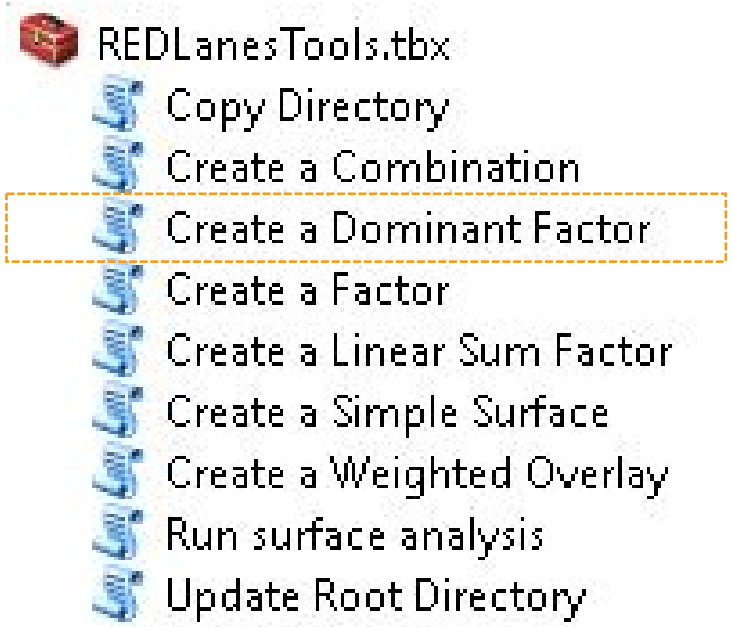
*Factors convert vector data to raster data (see "PYTHON TOOLKIT DOCUMENTATION" section)*




- Reference feature class – the vector features to convert to a raster dataset
- Weight field – the field in the reference feature class to reference to "weight" the resulting raster dataset (optional depending on "Analysis method")
- Where clause – sets criteria for which features in the reference feature class to utilize or ignore when converting to a raster dataset
- Analysis method – the measure (sum, mean, count, e.g.) to report in the resulting raster dataset
- Cell size – the size of the cells in the resulting raster dataset (in units equal to the linear units used by the reference feature class's spatial reference system)
- Neighborhood size – the radius of the floating zone used to analyze the features in the reference feature class to convert to a raster dataset (in units equal to the linear units used by the reference feature class's spatial reference system)
- Output units – for certain analysis methods, it is possible to specify what units the resulting raster values will be in. Remap values should reflect the chosen output units.

# DOMINANT FACTOR INPUTS

*Dominant Factors use grouping and weight fields to generate a raster containing the indices of the dominant (or least dominant) group (see "PYTHON TOOLKIT DOCUMENTATION" section)*

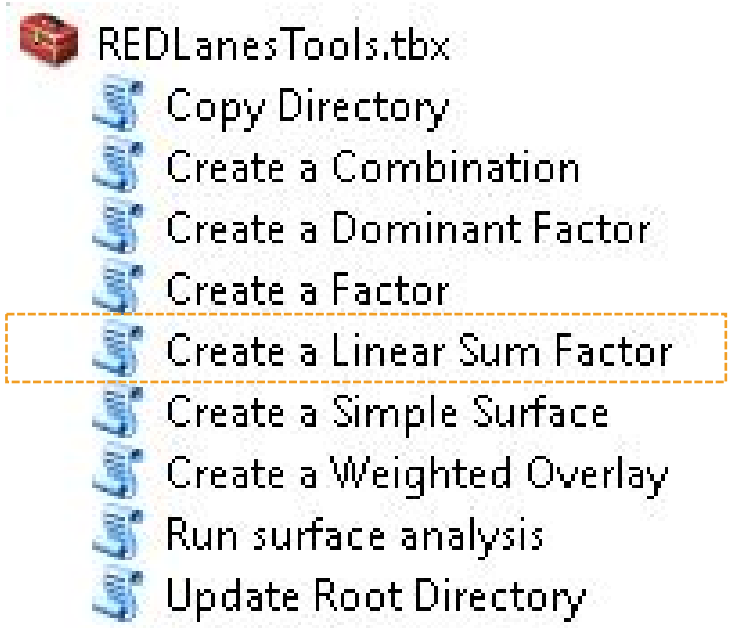



*Input fields match those of "Factor" except as noted below.*

- Value field – the field in the reference feature class by which to weight features in the rasterization process. Feature weights are summarized for each distinct value in the "group field" and the group with the highest weighted total is identified by its index in the output raster dataset.
- Group field – A field that groups features into distinct categories. When analyzed, the dominant factor will generate a raster with the index value of the "group" with the greatest sum of feature values (provided in the "Value field" in each cell.
- Inverse – If checked, return the raster index of the group with the lowest total feature values in each cell rather than the highest value. If multiple groups are missing (meaning more than one "lowest" group exists), the first index among lowest groups is returned in the raster.

# LINEAR SUM FACTOR INPUTS

*Linear Sum Factors provide simple summation of attribute values of linear features instead of length-weighted sums. (see "PYTHON TOOLKIT DOCUMENTATION" section)*

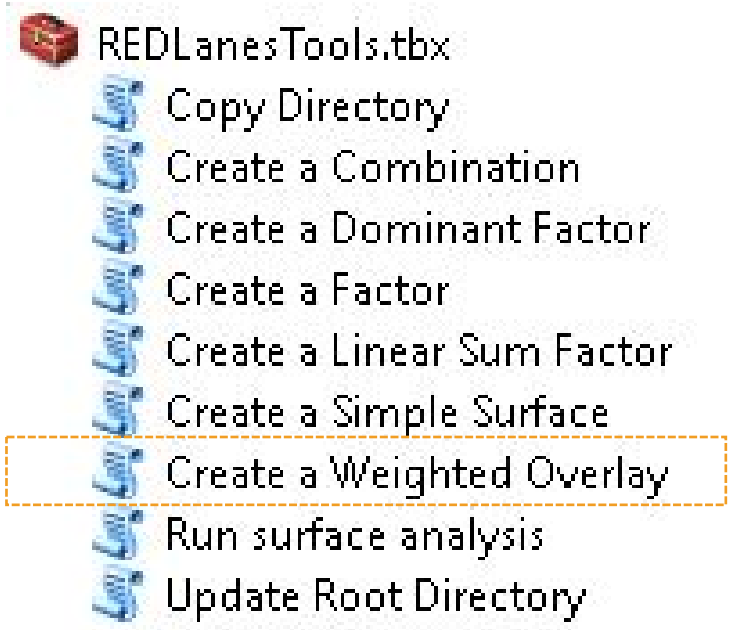



*Input fields match those of "Factor" except as noted below.*

- ID field – the field in the reference feature class that uniquely identifies each line. This field is required to incorporate a reliable count of line features in the neighborhood.

# WEIGHTED OVERLAY INPUTS

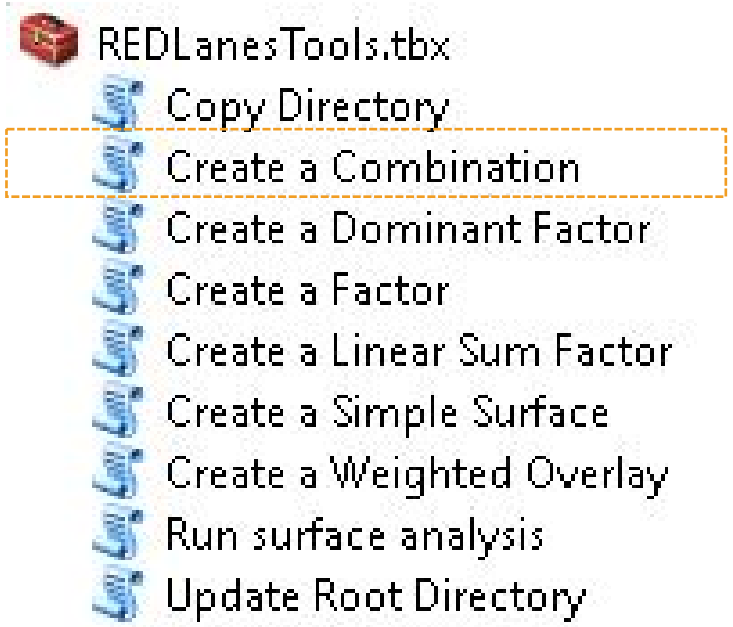
*Overlays create a new surface by overlaying two or more existing surface objects (see "PYTHON TOOLKIT DOCUMENTATION" section)*




- Input surface json files – list of the json files defining the surfaces to be overlaid to create the resulting raster.
- Weights – the relative weight of each input surface in the resulting raster. The list of weights parallels the list of input json files, so attention must be paid to the order of items in each list.
  - Best practice: the sum of the weights should add to 100.
- Results mapped from/to/by – these parameters define the evaluation scale of the resulting raster to be produced by the overlay. Generally, for the RED Lanes Suitability toolkit, the default values should be used:
  - From: 0
  - To: 10
  - By: 1

# COMBINATION INPUTS

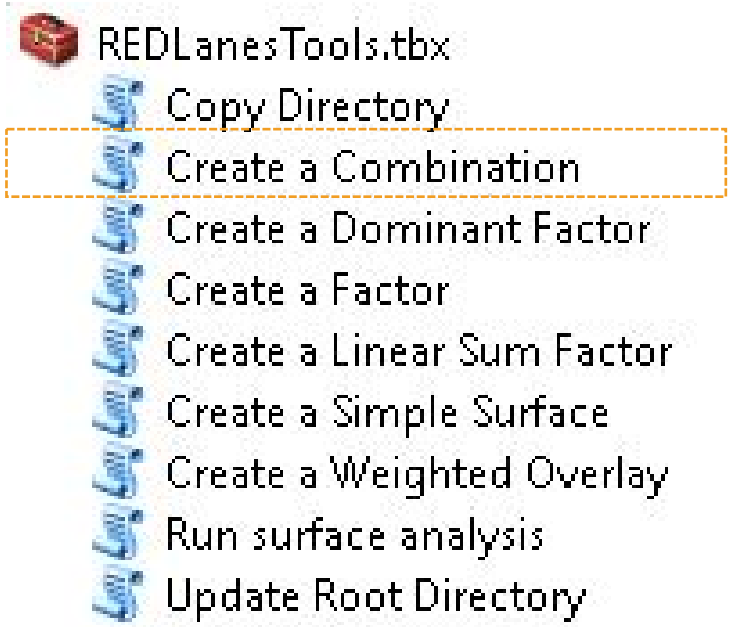
*Combinations create a new surface by combining a base surface with one or more adjustment surfaces (see "PYTHON TOOLKIT DOCUMENTATION" section)*




- Base surface – the combination will modify the data in this surface's output raster based on the values in the adjustment surface rasters, combination type, and processing parameters.
- Adjustment surfaces – The raster data to combine with the base surface to produce modified values. Multiple adjustment surfaces can be listed.

# COMBINATION INPUTS

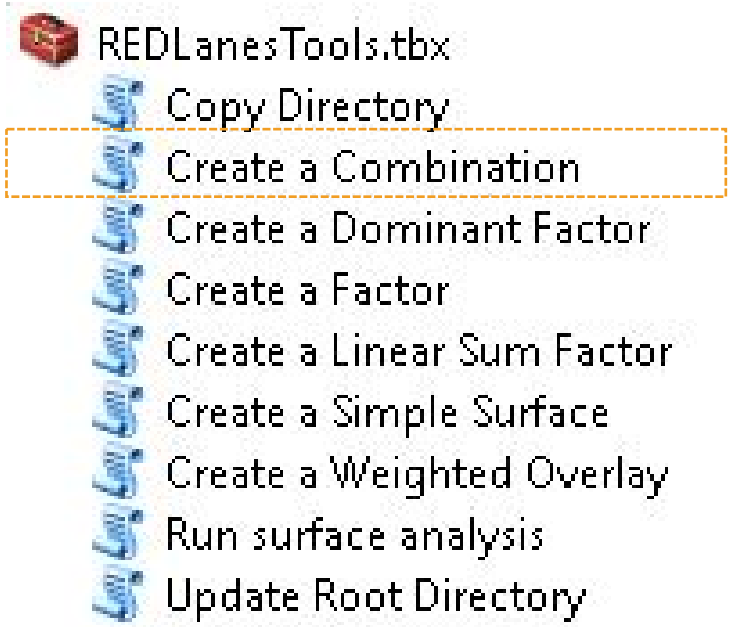
*Combinations create a new surface by combining a base surface with one or more adjustment surfaces (see "PYTHON TOOLKIT DOCUMENTATION" section)*




- Apply limits/apply above value/apply below value – if the "apply limits" option is selected, only certain values in the base surface will be modified – those above the "apply above value" and those below the "apply below value." All other base surface values will be retained without modification. Not applicable for "lookup" combos.
- Combo type – the modification logic depends on the combination type:
  - Calculation: perform simple mathematical operations to modify the values in the base surface based on values in the adjustment surface(s)
  - Conditional: modify values in the base surface where certain conditions apply in the adjustment surface(s)
  - Lookup: modify values in the base surface based on specific combinations of values with adjustment surfaces as specified in a lookup table.

# COMBINATION INPUTS

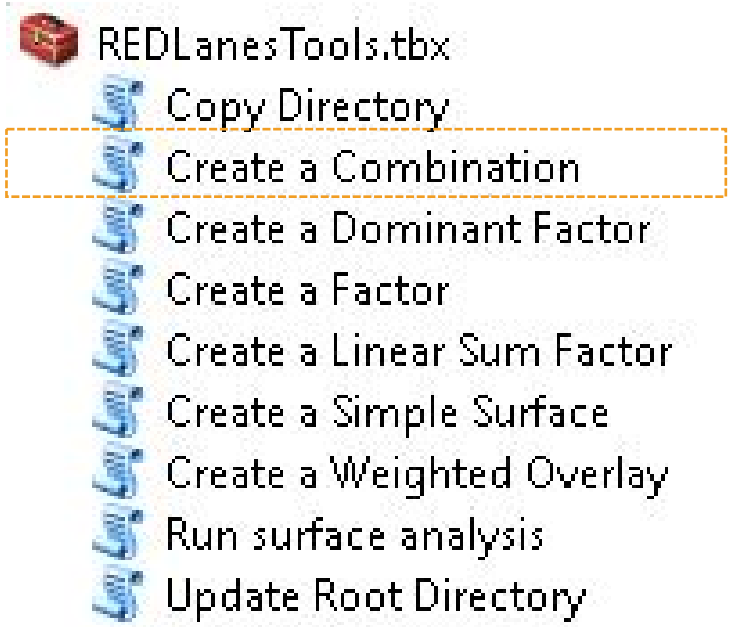
*Combinations create a new surface by combining a base surface with one or more adjustment surfaces (see "PYTHON TOOLKIT DOCUMENTATION" section)*




- Adjustment surface params – specifications for how to modify the values in the base surface based on the adjustment surface(s). The list of params parallels the list of adjustment factors, so attention must be paid to the order of items in each list. The format of the parameters to enter depend on combo type:
  - Conditional: Comma-separated list as follows: {conditional evaluation}, {value if true}, {value if false}
    - Example: "==1, 801, Base"
    - Interpretation: If the adjustment surface value is equal to 1, alter the base value to be 801, otherwise use the base value
  - Calculation: Comma-separated list as follows: {primary arithmetic operation}, {adjustment factor modification}
    - Example: "+, /3.0"
    - Interpretation: Increase the base surface value by the value in the adjustment surface divided by 3
  - Lookup: The column name in the lookup table that corresponds to the values in the adjustment surface

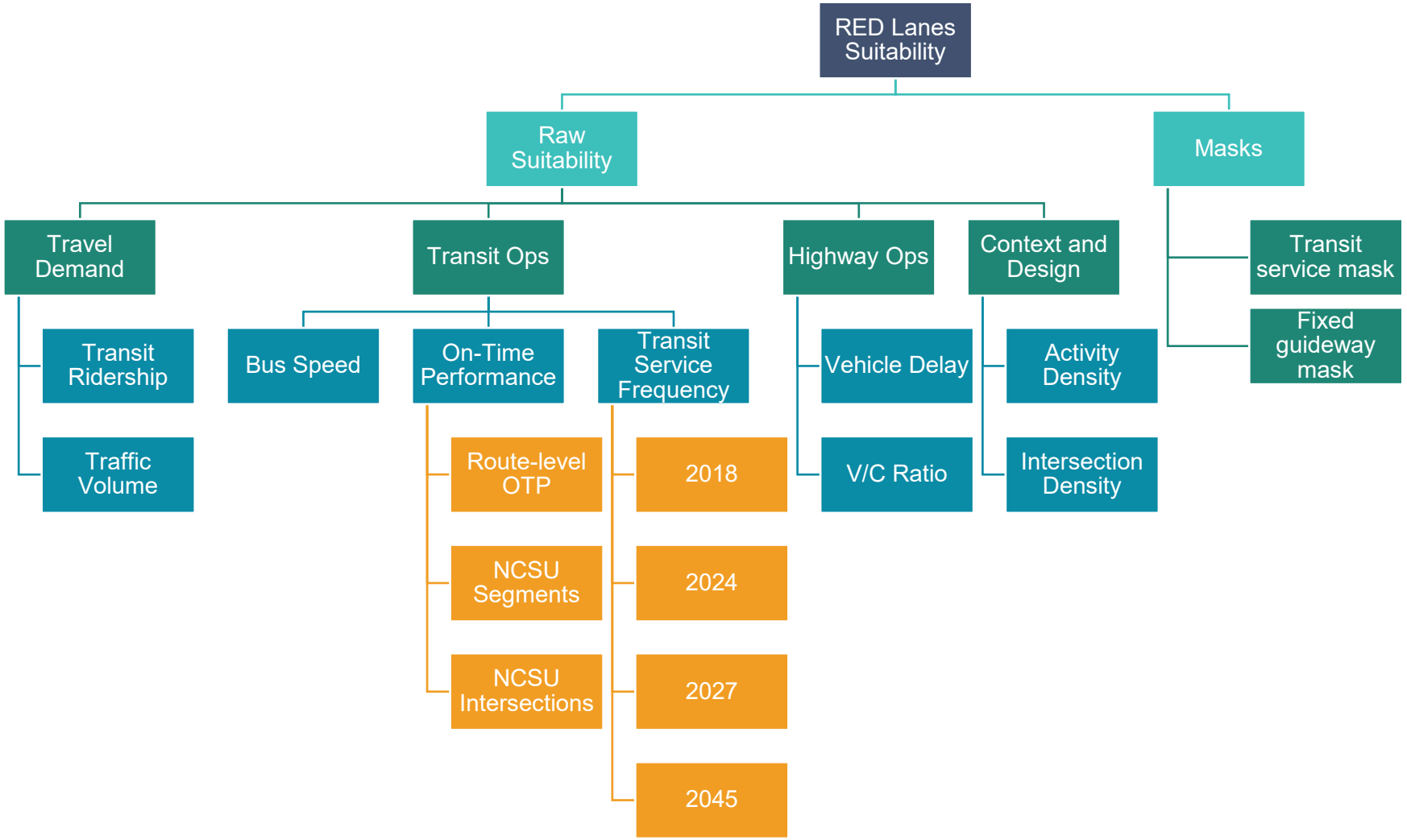
# COMBINATION INPUTS

*Combinations create a new surface by combining a base surface with one or more adjustment surfaces (see "PYTHON TOOLKIT DOCUMENTATION" section)*

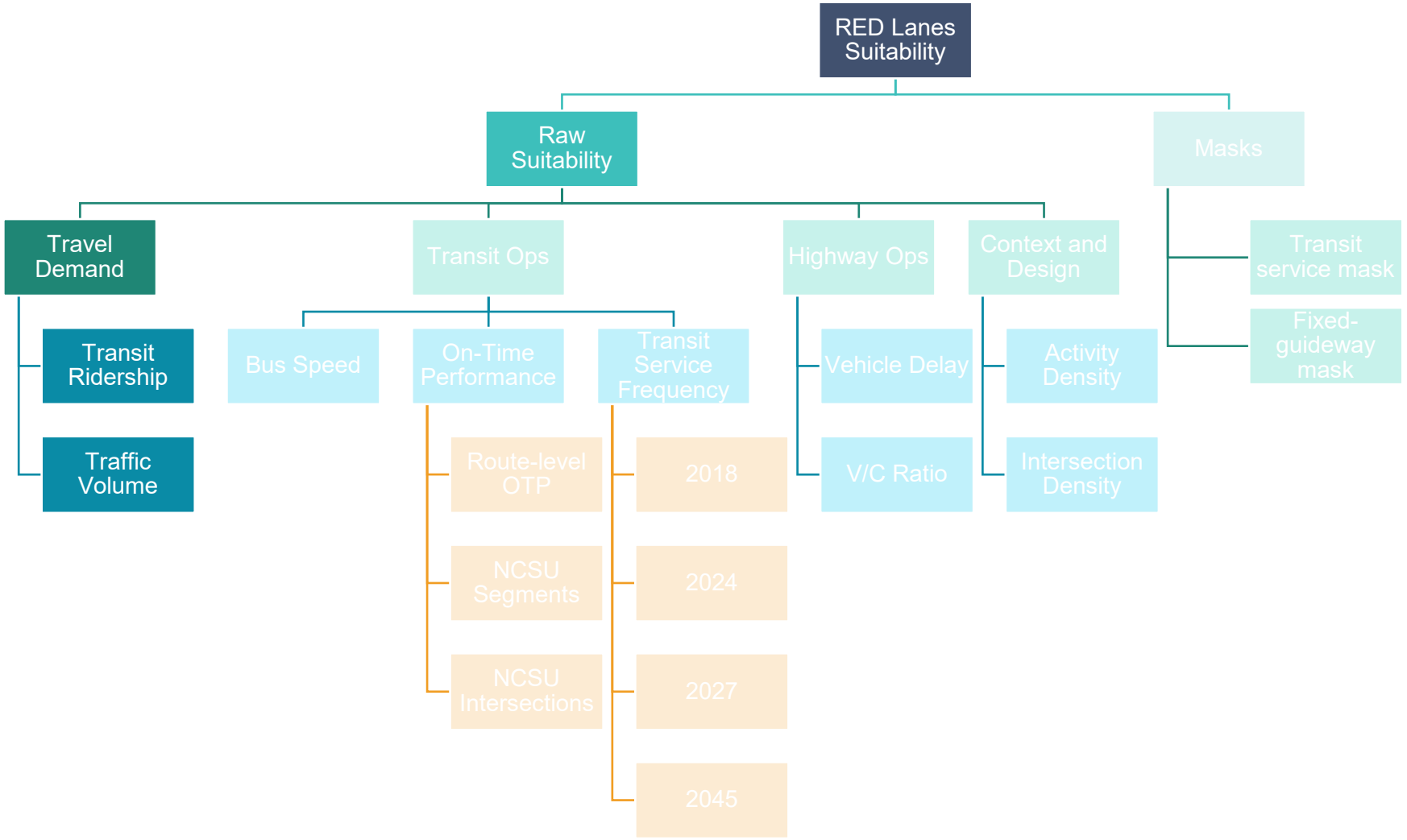



- Apply calculation bounds/Calculation lower bound/Calculation upper bound – if the "apply calculation bounds" option is selected, the results of the calculation will be capped based on the "calculation lower bound" and "calculation upper bound" values. Applicable for "calculation" combinations only.
- Lookup table\Base value column\New value column – The table that defines what values will be yielded by specific combinations of base and adjustment values. The "base value column" refers to values in the base surface. The new value column defines resulting values. Adjustment factor values are looked up from columns as specified in the Adjustment Surfaces Params input. Applicable for "lookup" combos only.

# DETAILED IMPLEMENTATION STEPS – RED LANES SUITABILITY



# SUITABILITY – TRAVEL DEMAND

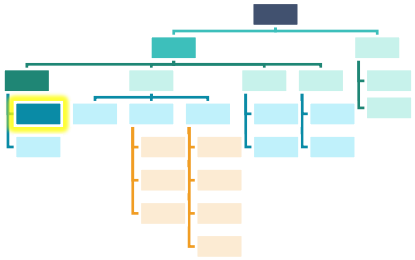


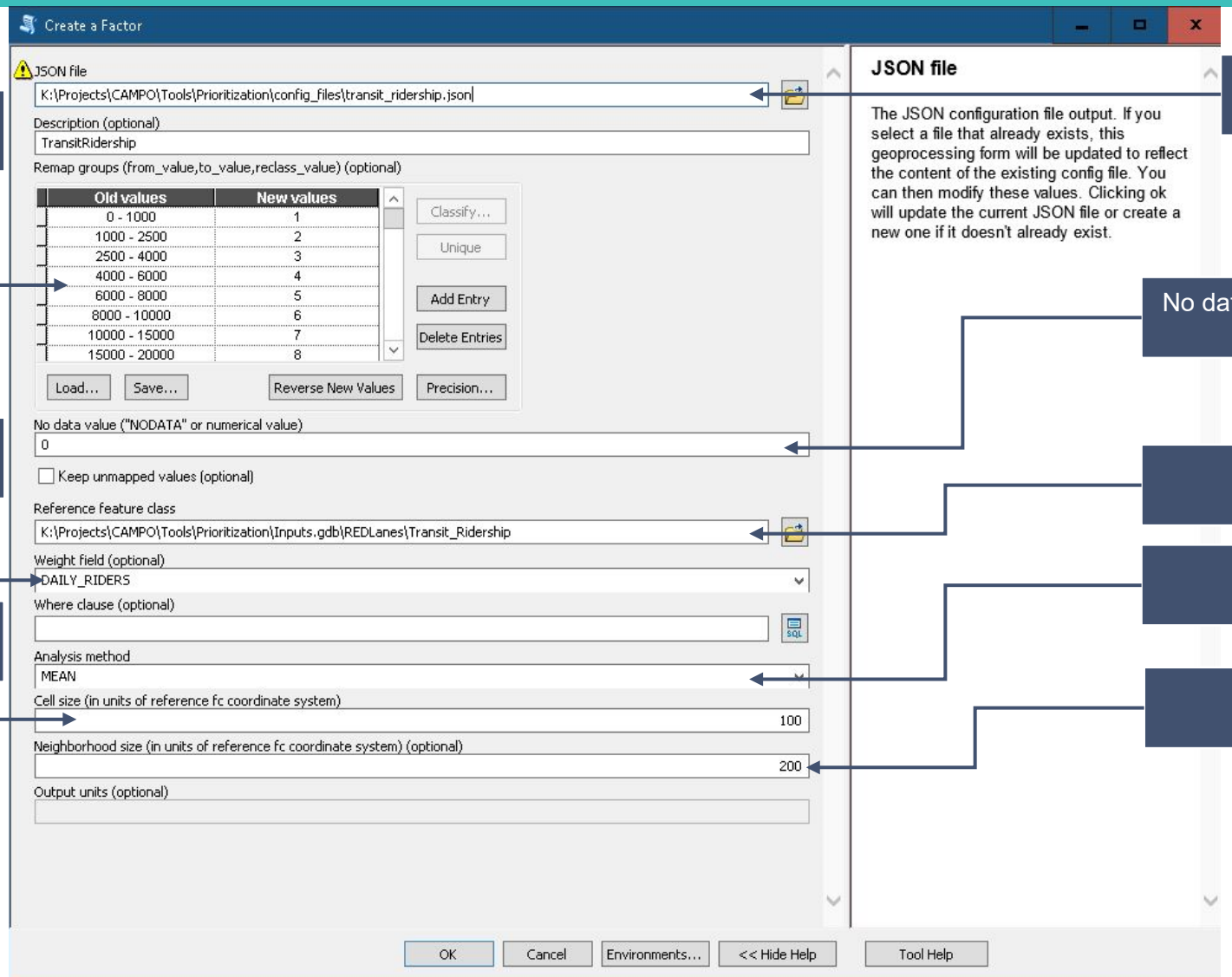
# SUITABILITY – TRAVEL DEMAND – TRANSIT RIDERSHIP

Remap groups: load from {remaps_dir}/transit_ridership

Weight field: DAILY_RIDERS

Cell size: 100 (feet)






JSON file: {config_dir}\transit_ridership.json

No data value: 0 (No suitability due to ridership if no ridership data in neighborhood)

Reference feature class: {inputs_gdb}\Transit_Ridership

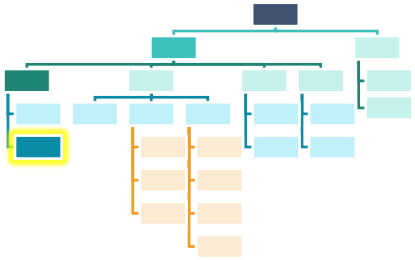
Analysis method: MEAN

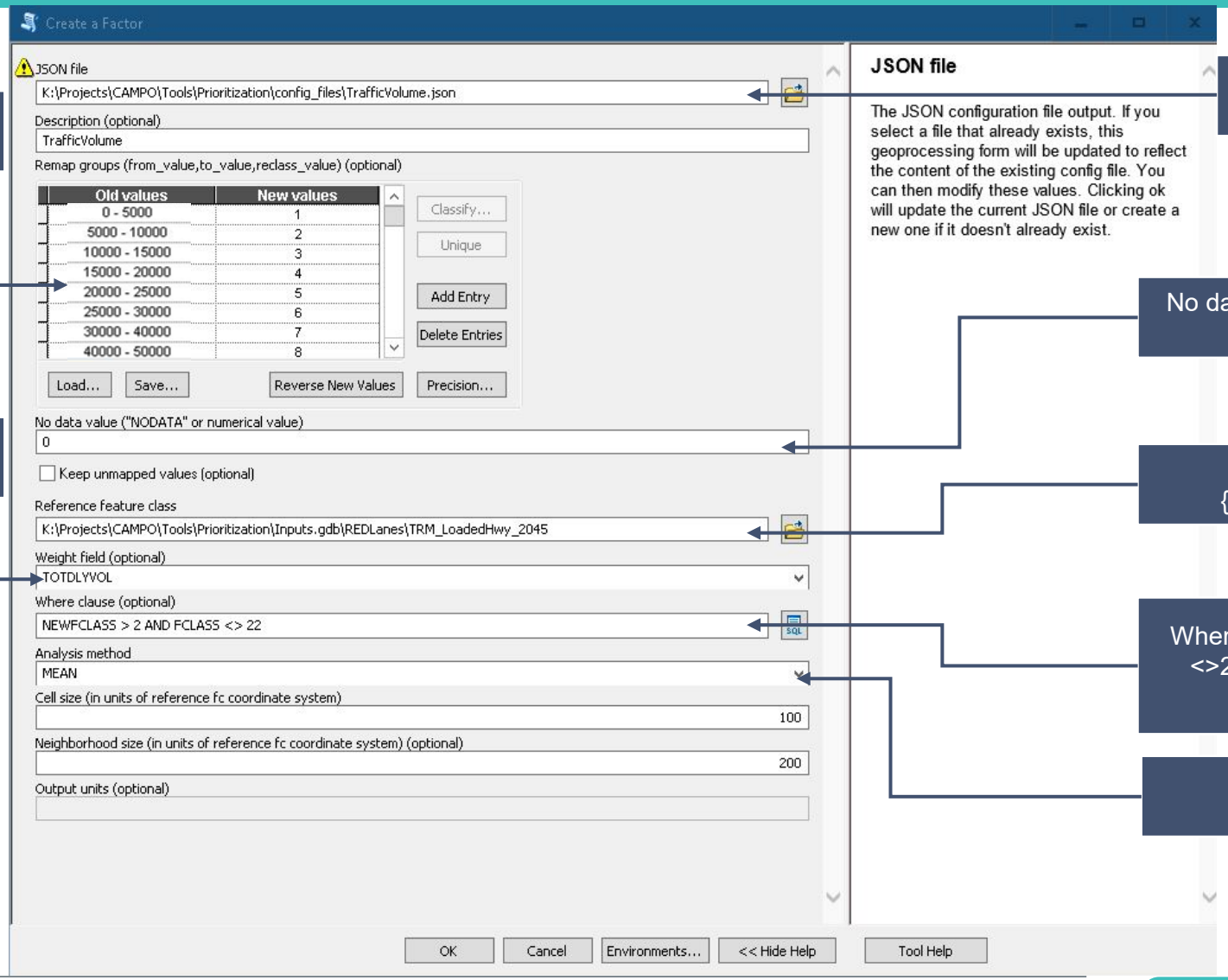
Neighborhood size: 200 (feet)

# SUITABILITY – TRAVEL DEMAND – TRAFFIC VOLUME

Remap groups: load from {remaps_dir}/TrafficVolume

Weight field: TOTDLYVOL






JSON file: {config_dir}\TrafficVolume.json

No data value: 0 (No suitability due to volume if no volume data in neighborhood)

Reference feature class: {inputs_gdb}\TRM_LoadedHwy_2045

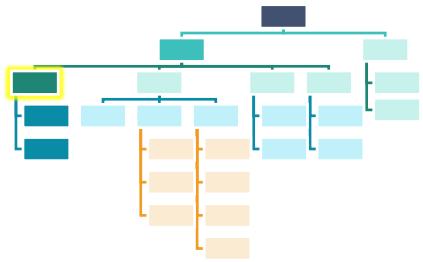
Where clause: NEWFCLASS>2 AND FCLASS <>22 (only include non-limited access/non-tollway facilities)

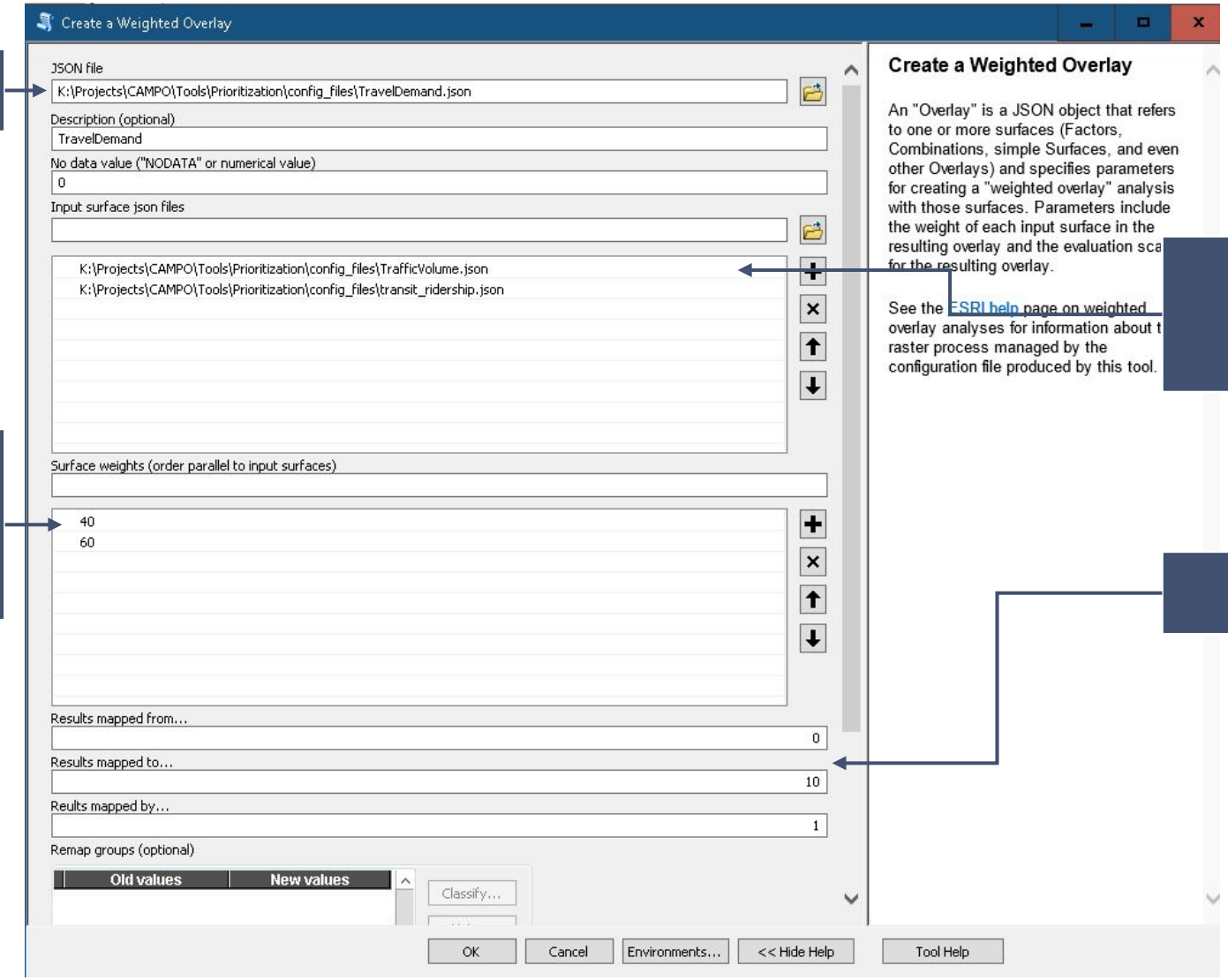
Analysis method: MEAN

# SUITABILITY – TRAVEL DEMAND – OVERLAY

JSON file:
{config_dir}\TravelDemand.json

Weights:
Traffic volume: 40
Transit ridership: 60

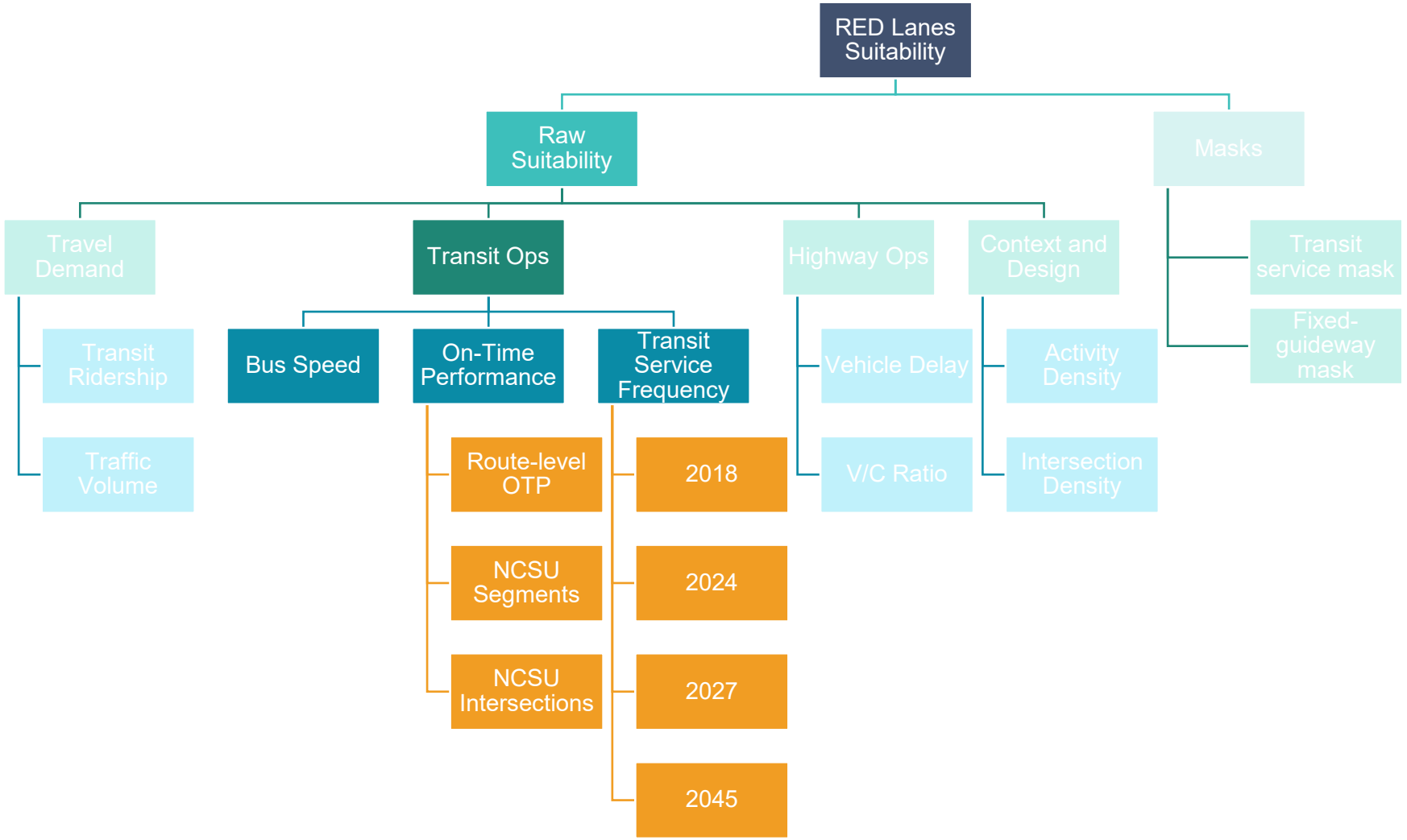





Input surface json files:
{config_dir}\TrafficVolume.json
{config_dir}\transit_ridership.json

Results mapped from/to/by:
Defaults (0/10/1)

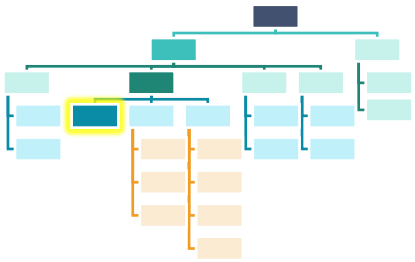
# SUITABILITY – TRANSIT OPERATIONS

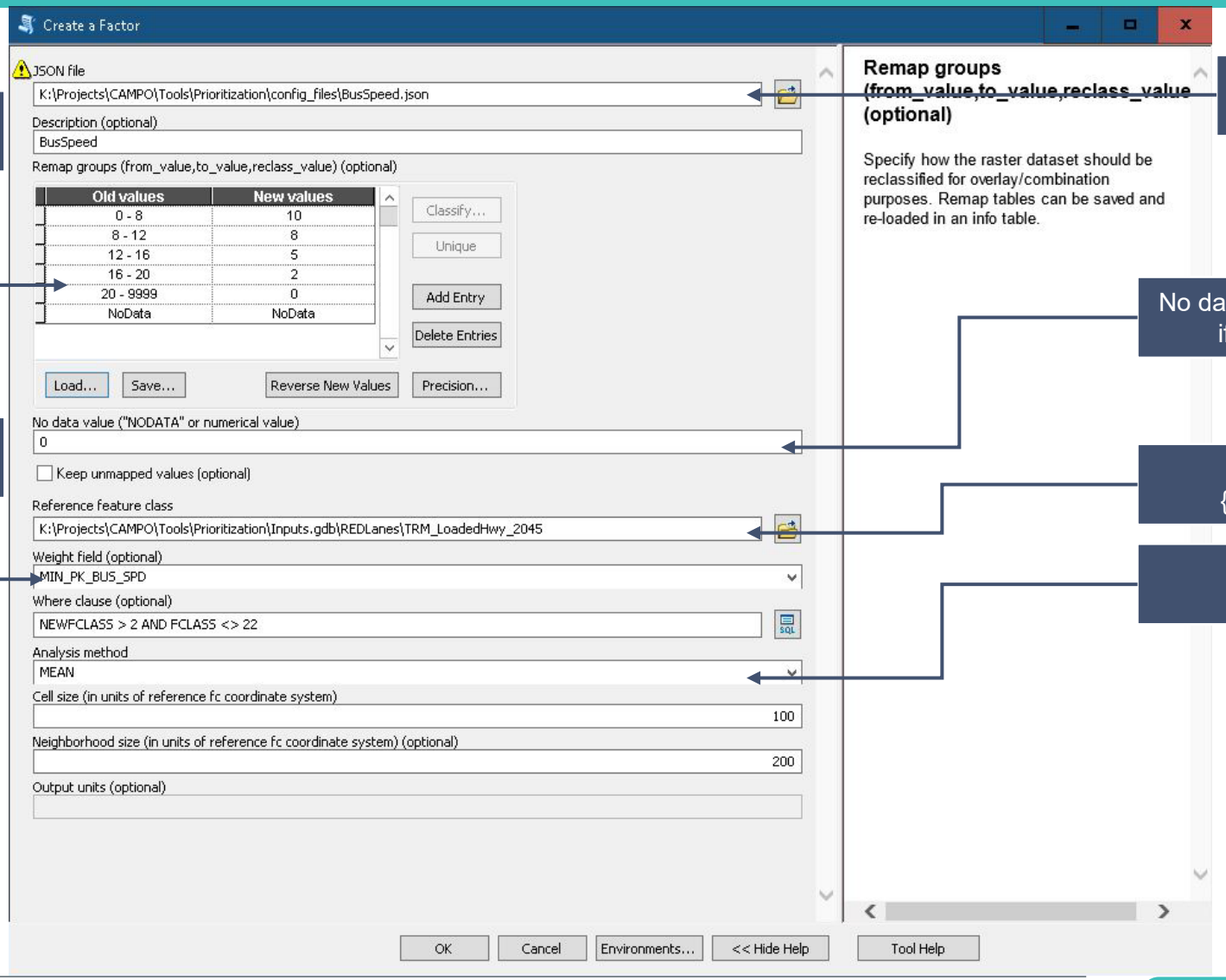


# SUITABILITY – TRANSIT OPS – BUS SPEED

Remap groups: load from {remaps_dir}/BusSpeed

Weight field: MIN_PK_BUS_SPD






JSON file: {config_dir}\BusSpeed.json

No data value: 0 (No suitability due to bus speed if no bus speed data in neighborhood)

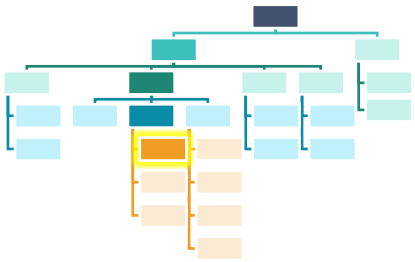
Reference feature class: {inputs_gdb}\TRM_LoadedHwy_2045

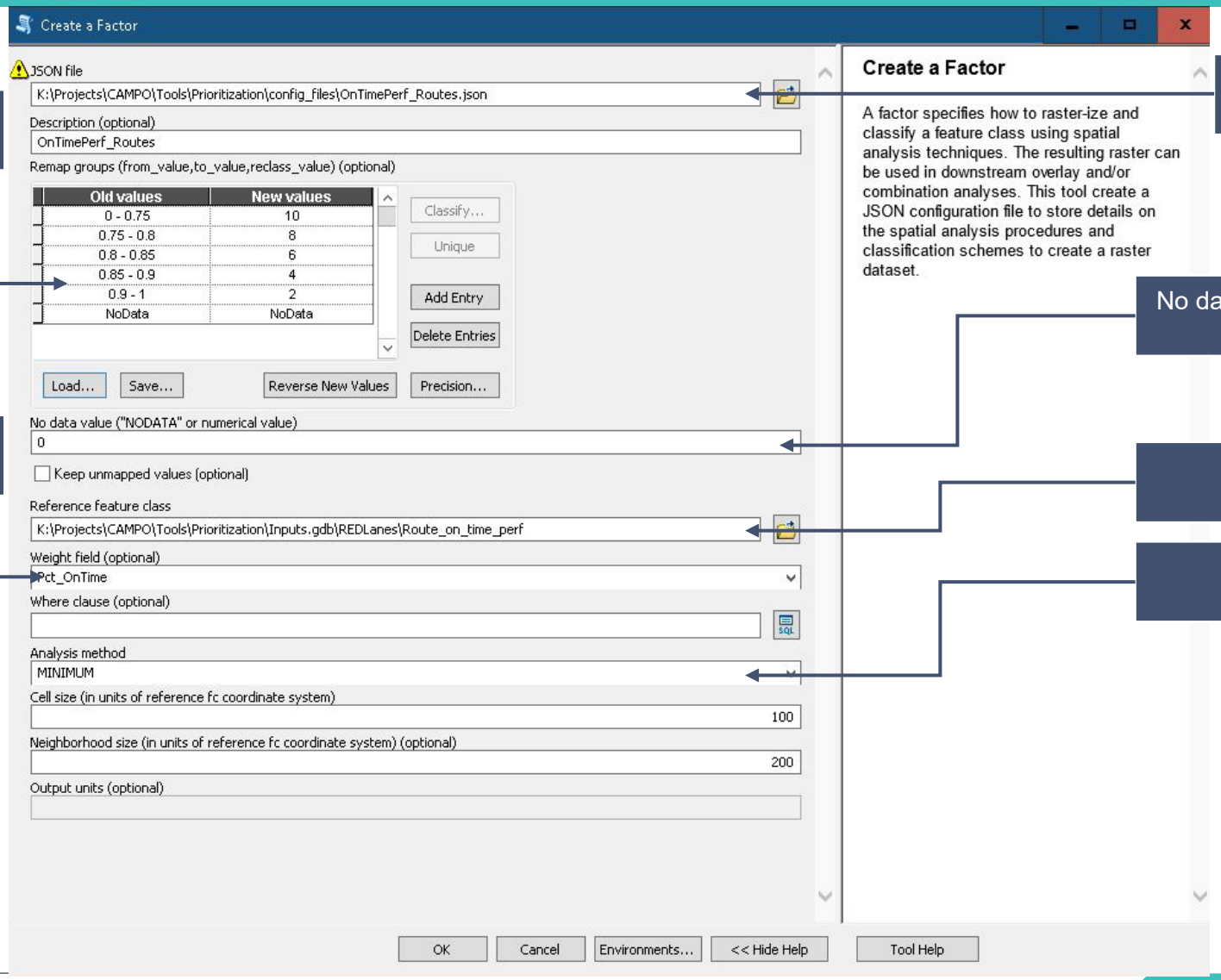
Analysis method: MEAN

# SUITABILITY – TRANSIT OPS – ON TIME PERFORMANCE – ROUTES

Remap groups: load from {remaps_dir}/on_time_perf

Weight field: Pct_OnTime






JSON file: {config_dir}\OnTimePerf_Routes.json

No data value: 0 (No suitability due to route OTP if no OTP data in neighborhood)

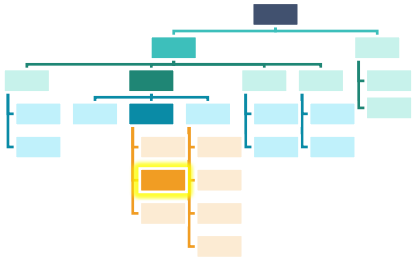
Reference feature class: {inputs_gdb}\Route_on_time_oerf

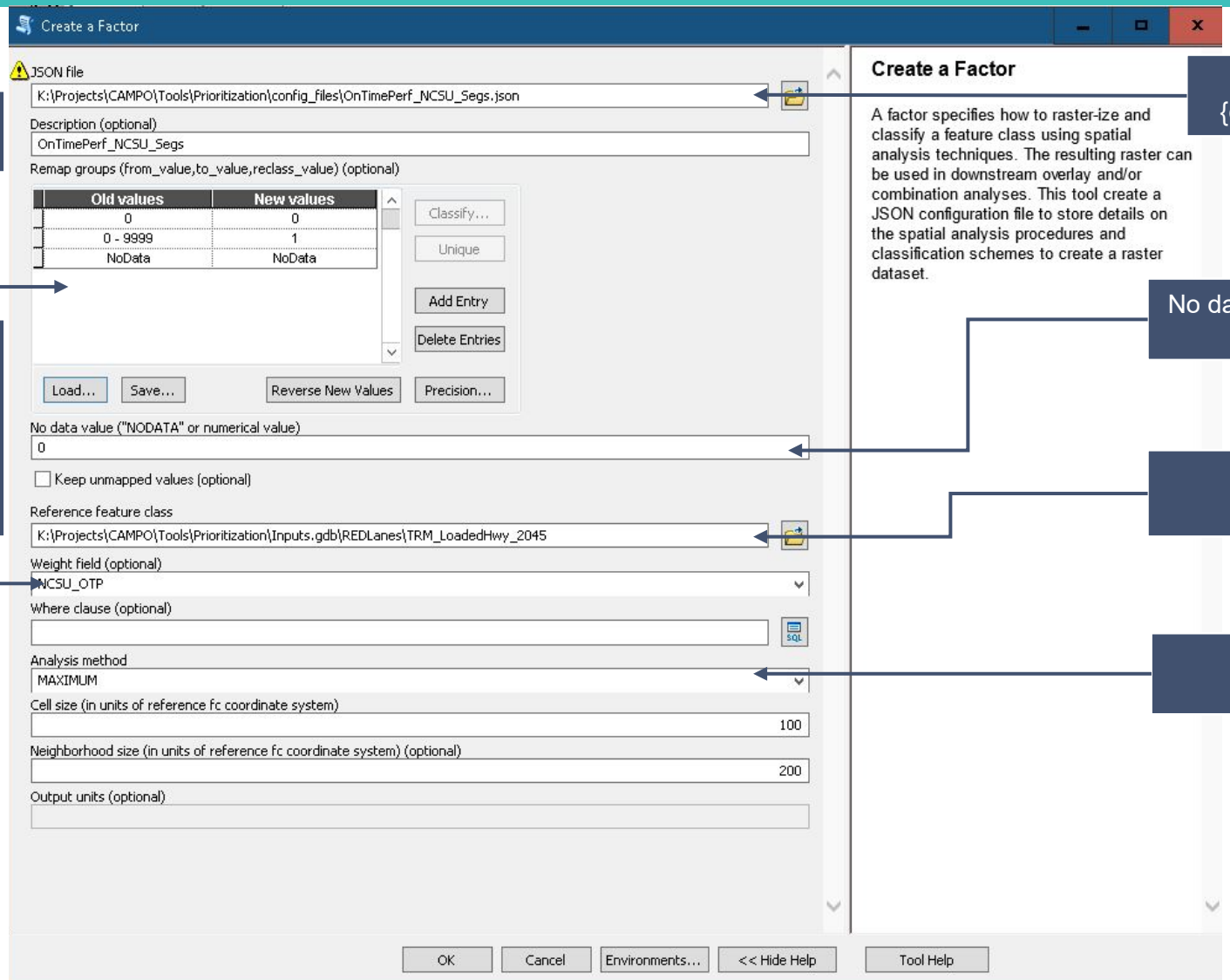
Analysis method: MINIMUM

# SUITABILITY – TRANSIT OPS – ON TIME PERFORMANCE – NCSU SEGMENTS

Remap groups: load from {remaps_dir}/ncsu_otp

Weight field: NCSU_OTP
(this is not a native TRM field, but a user-added field to flag segments identified by NCSU Wolfline staff as presenting reliability challenges)






JSON file: {config_dir}\OnTimePerf_NCSU_Segs.json

No data value: 0 (No suitability due to no flagged NCSU segments in neighborhood)

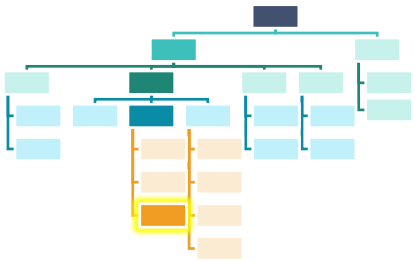
Reference feature class: {inputs_gdb}\TRM_LoadedHwy_2045

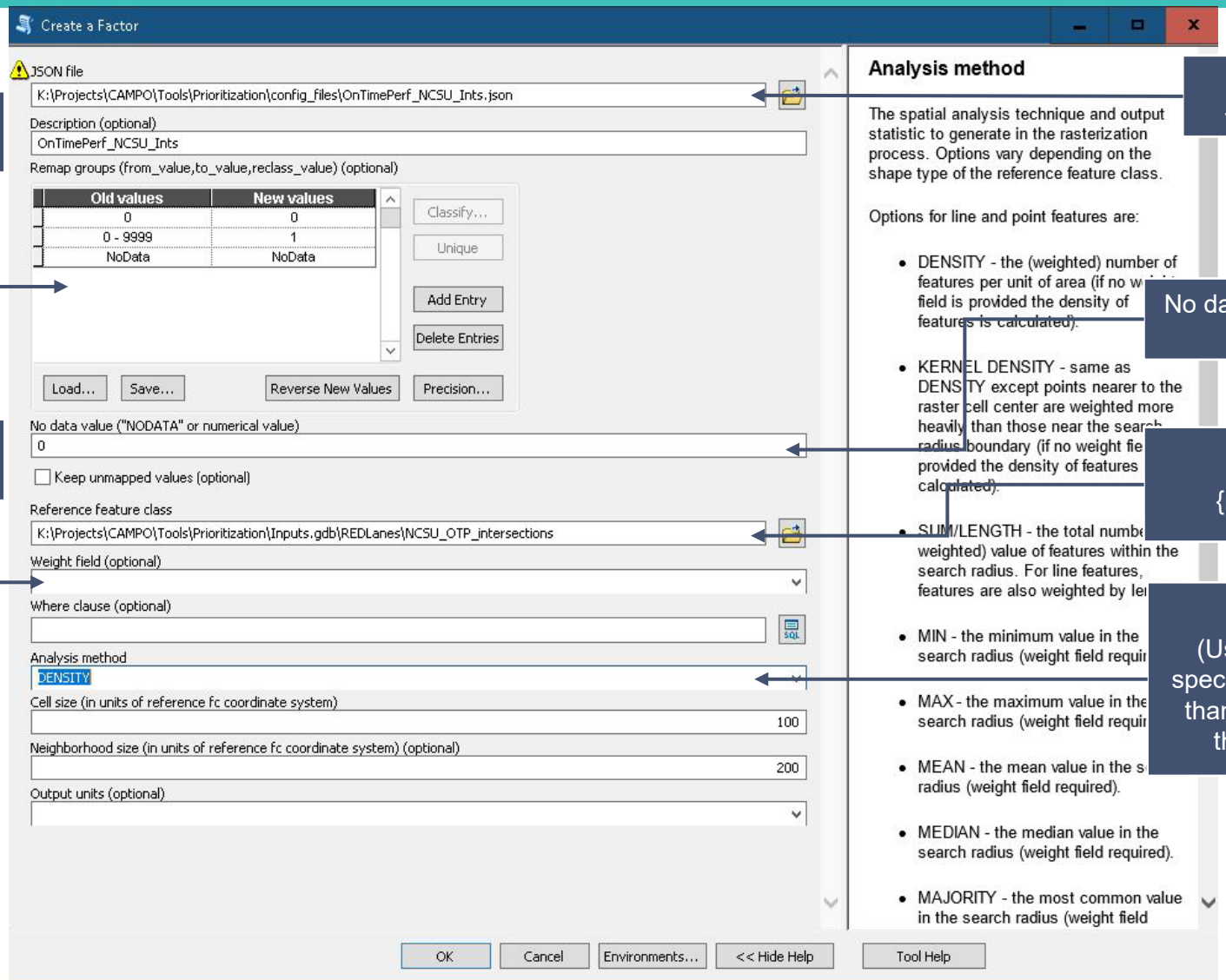
Analysis method: MAXIMUM

# SUITABILITY – TRANSIT OPS – ON TIME PERFORMANCE – NCSU INTERSECTIONS

Remap groups: load from {remaps_dir}/ncsu_otp

Weight field: (None)






JSON file: {config_dir}\OnTimePerf_NCSU_Ints.json

No data value: 0 (No suitability due to no flagged NCSU intersections in neighborhood)

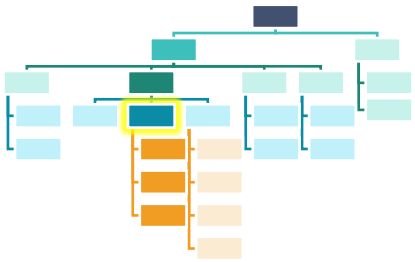
Reference feature class: {inputs_gdb}\NCSU_OTP_intersections

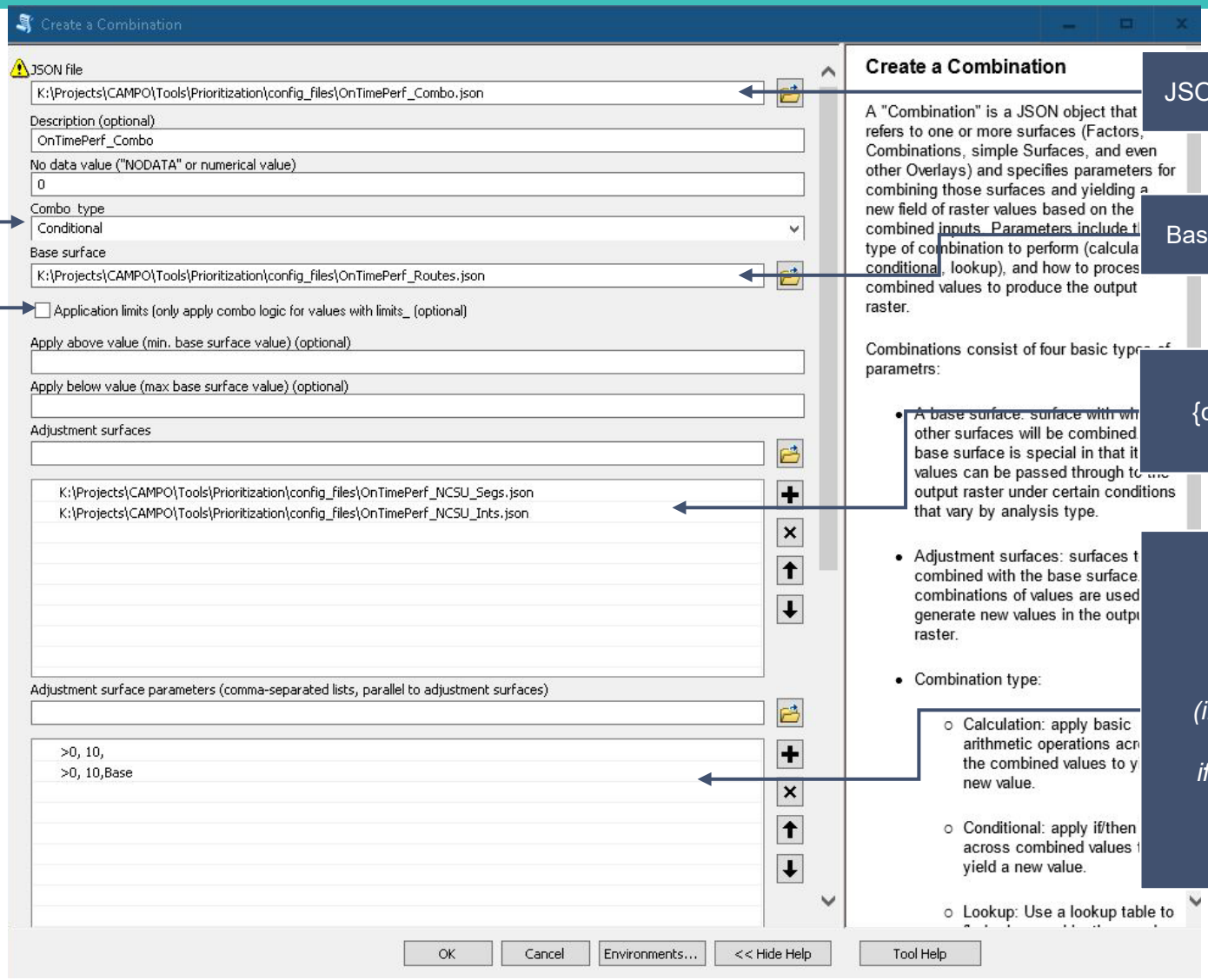
Analysis method: DENSITY
(Using this method means we don't have to specify a weight field – feature densities greater than zero are all we need to flag intersections that present transit reliability challenges)

# SUITABILITY – TRANSIT OPS – ON TIME PERFORMANCE COMBO

Combo type: Conditional

Apply limits: False (unchecked)






JSON file: {config_dir}\OnTimePerf_Combo.json

Base factor: {config_dir}\OnTimePerf_Routes.json

Adjustment surfaces:
{config_dir}\OnTimePerf_NCSU_Segs.json
{config_dir}\OnTimePerf_NCSU.json

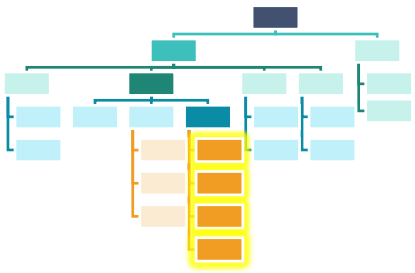
Adjustment surface parameters
>0, 10,
>0, 10, Base

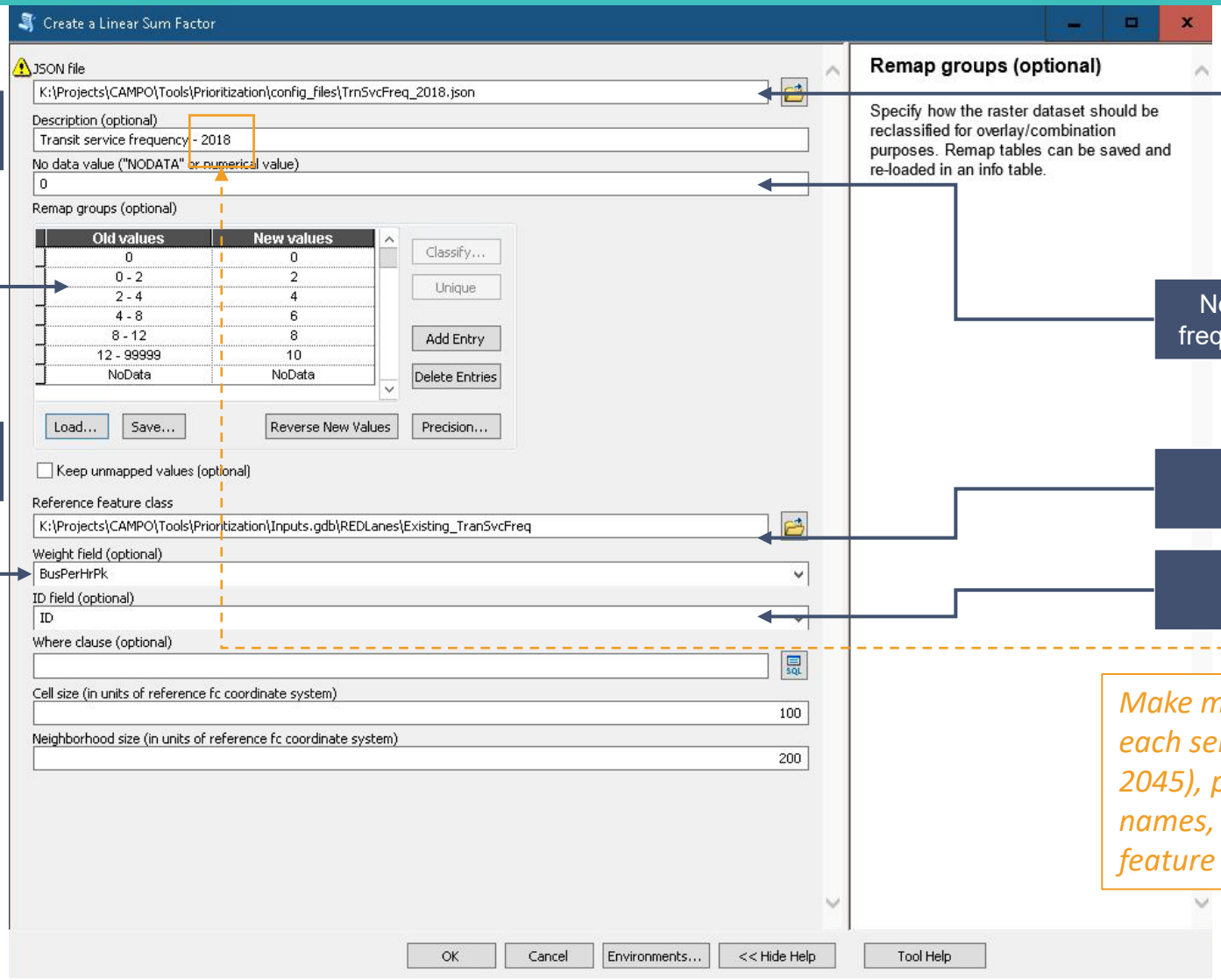
*(if OnTimePerf_NCSU_segs > 0, return 10, else....*
*if OnTimePerf_NCSU_Intss > 0, return 10, else....*
*Return the value in the base surface)*

# SUITABILITY – TRANSIT OPS – TRANSIT SERVICE FREQUENCY - YEARS

Remap groups: load from {remaps_dir}/transit_service_freq

Weight field: BusPerHrPk






JSON file: {config_dir}\TrnSvcFreq_2018.json

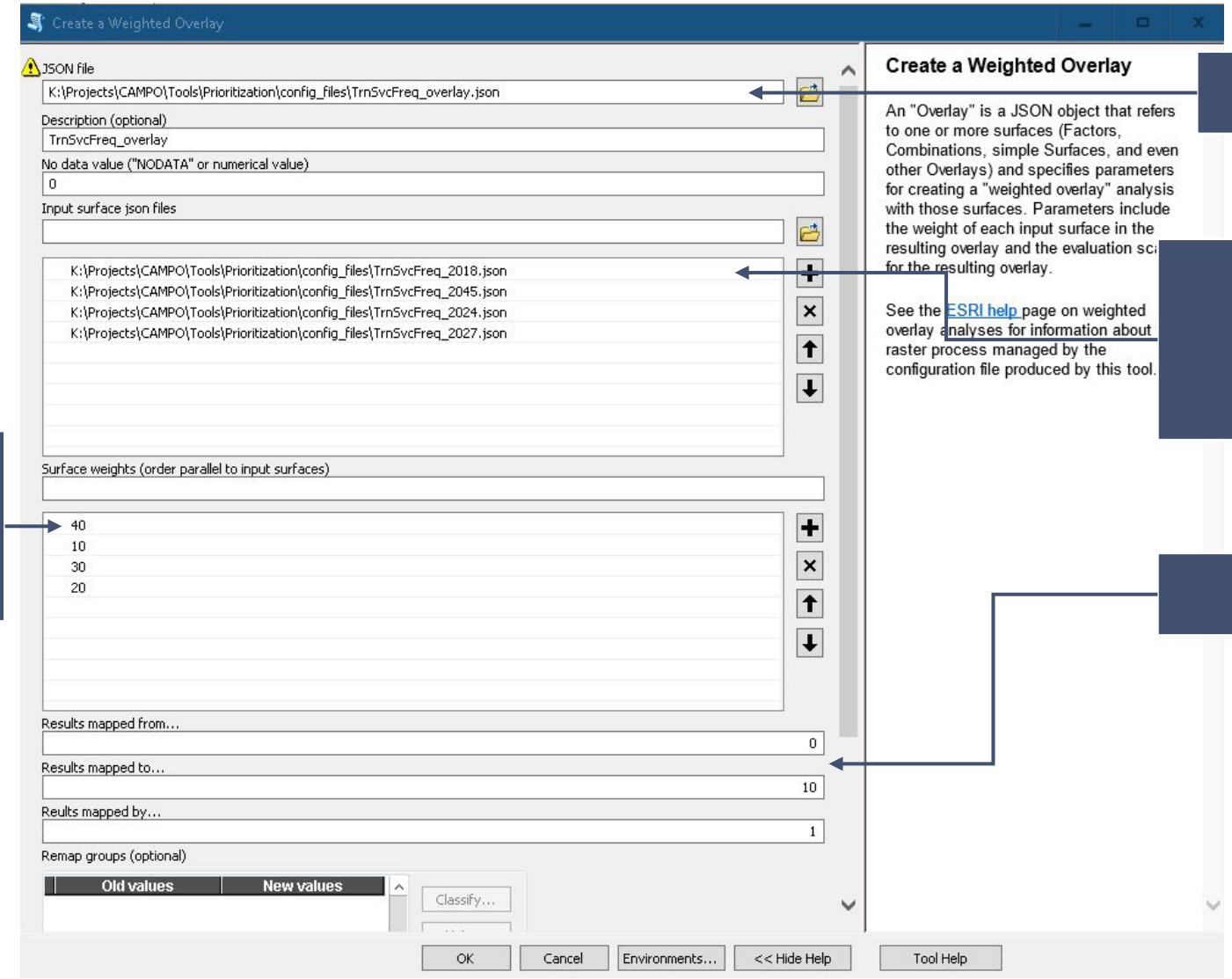
No data value: 0 (No suitability due to transit frequency if no frequency data in neighborhood)

Reference feature class: {inputs_gdb}\Existing_TranSvcFreq

ID Field: ID

*Make multiple config files – one for each service year (2018, 2024, 2027, 2045), providing appropriate file names, descriptions, and reference feature classes to the tool.*

# SUITABILITY – TRANSIT OPS – TRANSIT SERVICE FREQUENCY OVERLAY

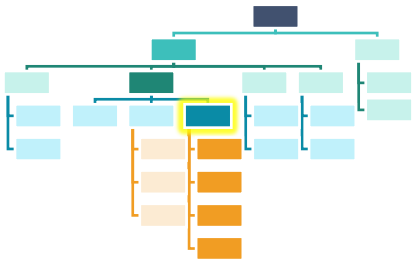



JSON file:
{config_dir}\TrnSvcFreq_overlay.json

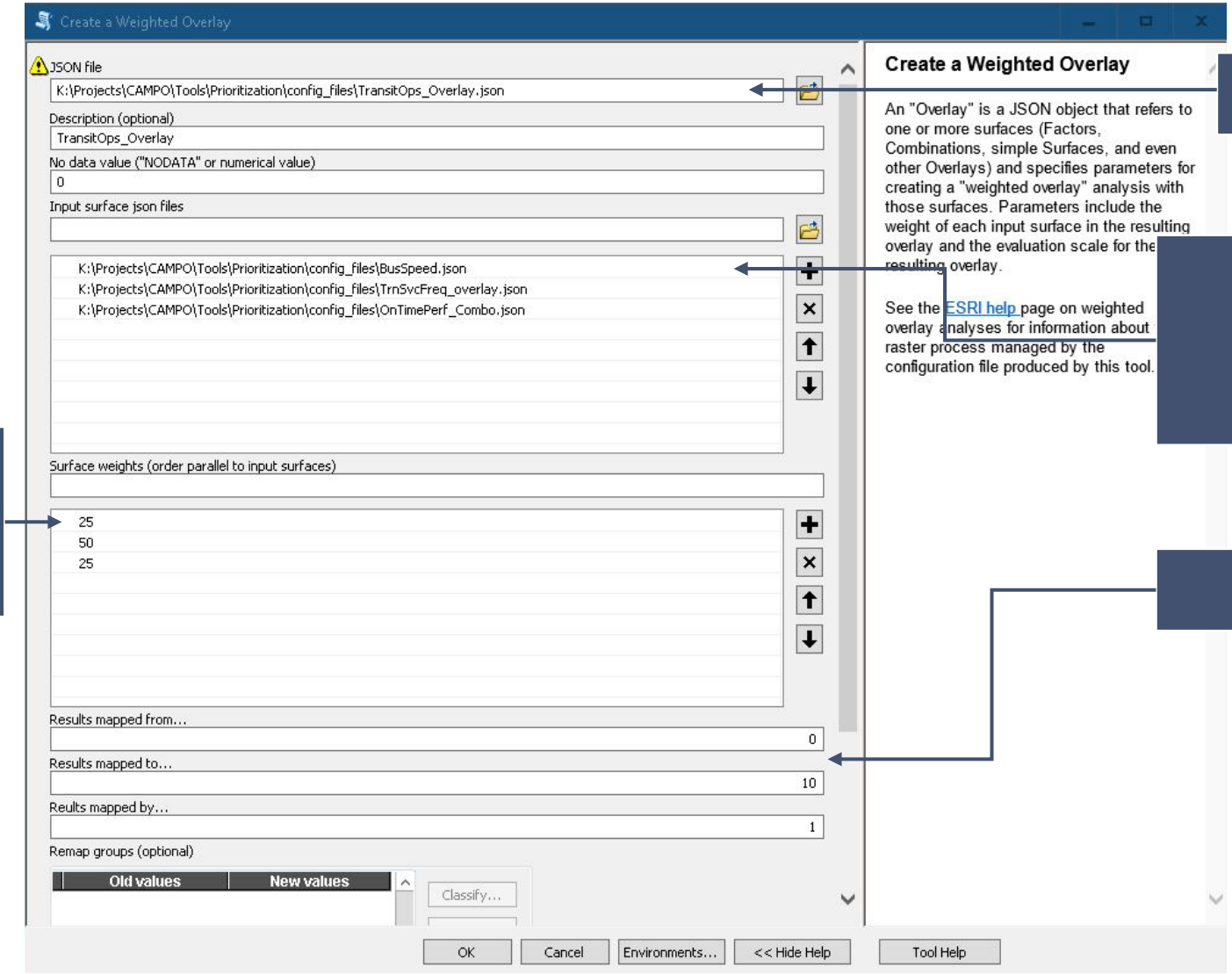
Input surface json files:
{config_dir}\TrnSvcFreq_2018.json
{config_dir}\TrnSvcFreq_2045.json
{config_dir}\TrnSvcFreq_2024.json
{config_dir}\TrnSvcFreq_2027.json

Weights:
2018: 40
2045: 10
2024: 30
2027: 20

Results mapped from/to/by:
Defaults (0/10/1)



# SUITABILITY – TRANSIT OPS – OVERLAY

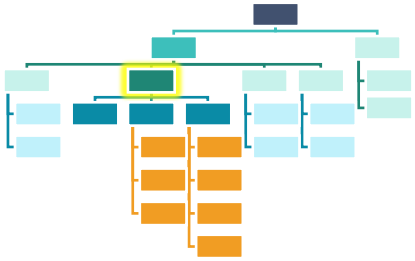



JSON file:
{config_dir}\TransitOps_Overlay.json

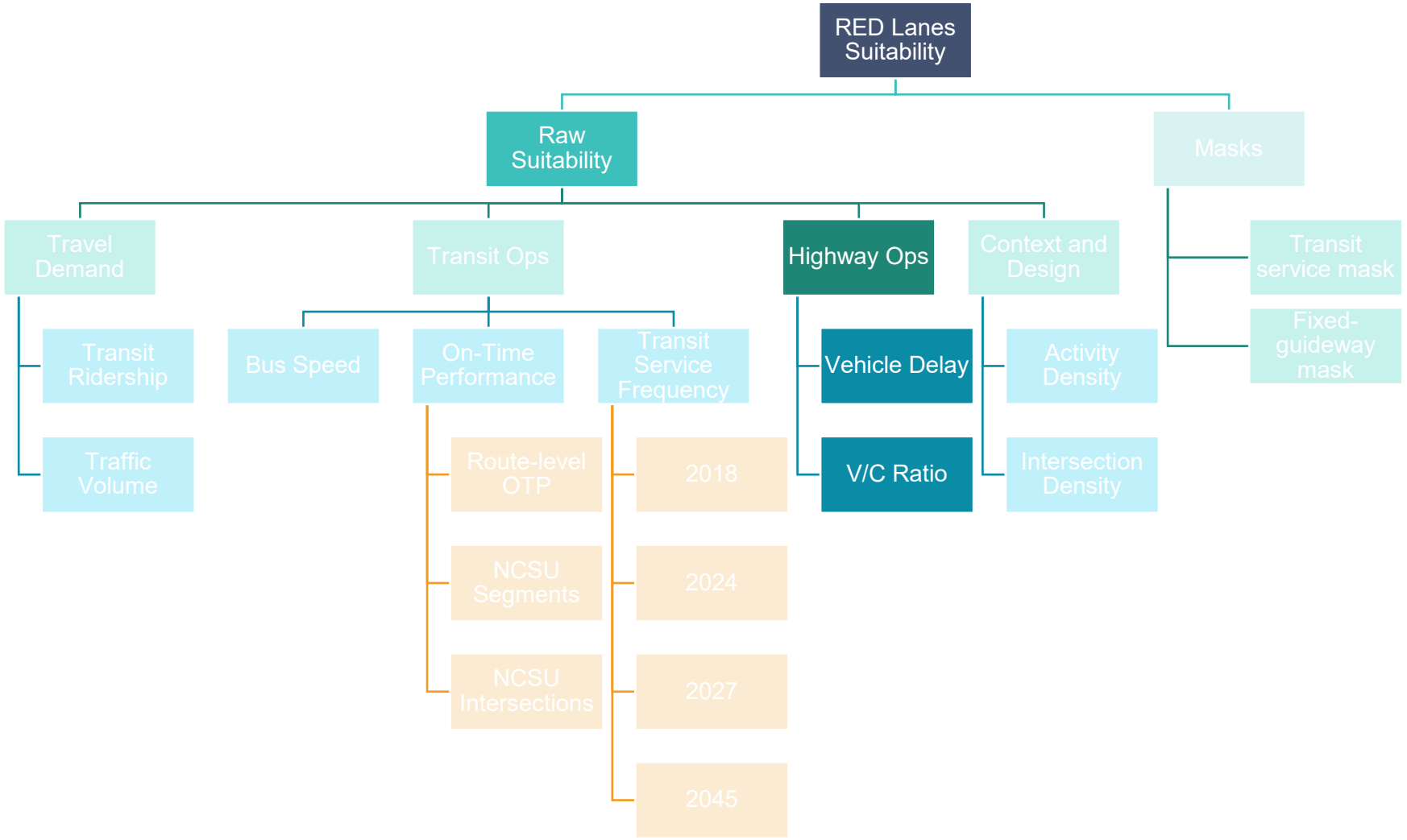
Input surface json files:
{config_dir}\BusSpeed.json
{config_dir}\TrnSvcFreq_overlay.json
{config_dir}\OnTimePerf_Combo.json

Weights:
Bus speed: 25
Transit service frequency: 50
On-time performance: 25

Results mapped from/to/by:
Defaults (0/10/1)



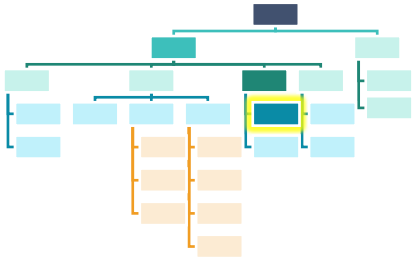
# SUITABILITY – HIGHWAY OPERATIONS

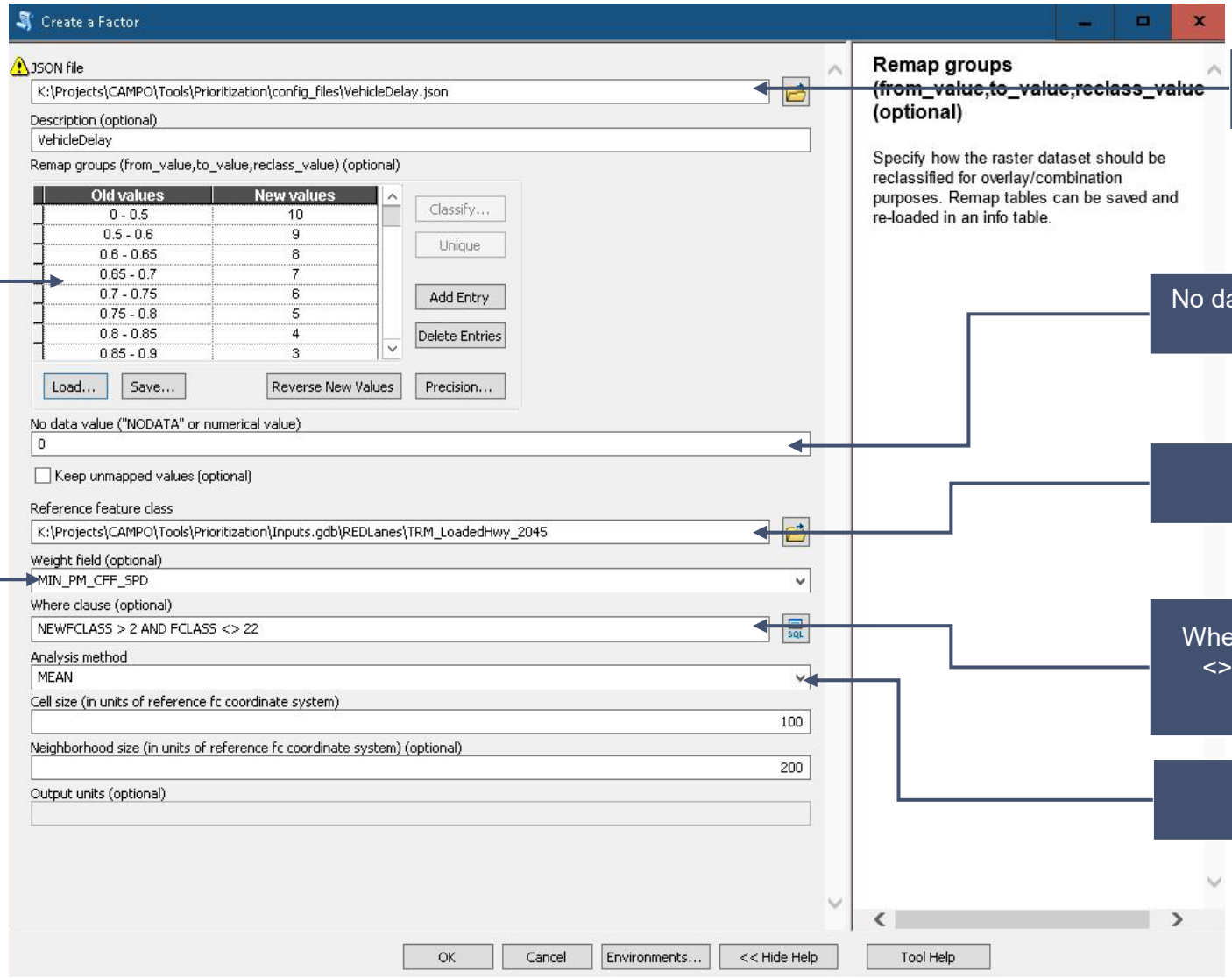


# SUITABILITY – HIGHWAY OPS – VEHICLE DELAY

Remap groups: load from {remaps_dir}/vehicle_delay

Weight field: MIN_PM_CFF_SPD






JSON file: {config_dir}\VehicleDelay.json

No data value: 0 (No suitability due to delay if no delay data in neighborhood)

Reference feature class: {inputs_gdb}\TRM_LoadedHwy_2045

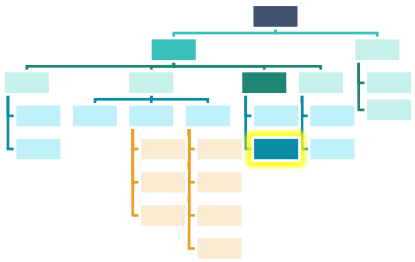
Where clause: NEWFCLASS>2 AND FCLASS <>22 (only include non-limited access/non-tollway facilities)

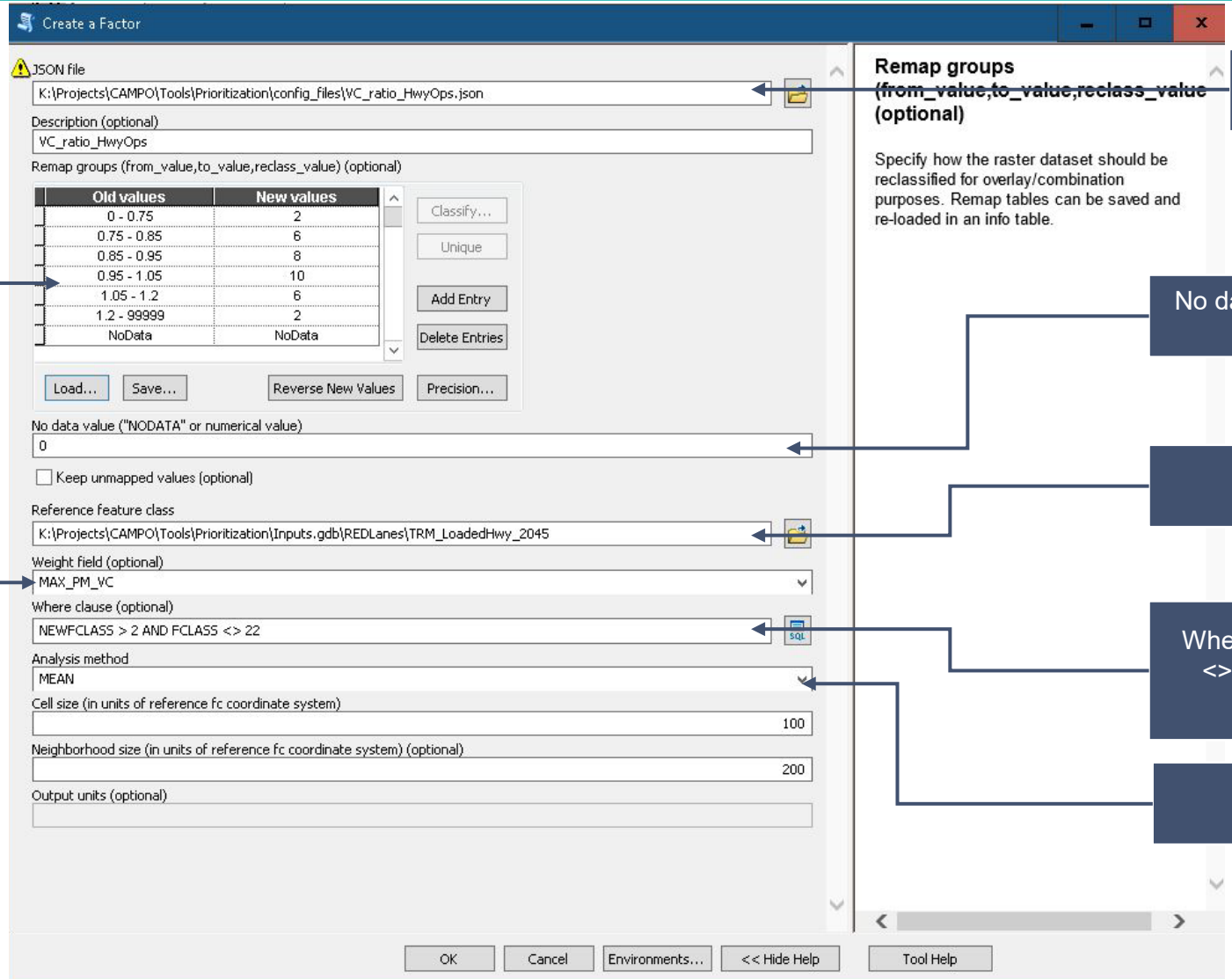
Analysis method: MEAN

# SUITABILITY – HIGHWAY OPS – V/C RATIO

Remap groups: load from {remaps_dir}/vc_ratio_hwy_ops

Weight field: MAX_PM_VC






JSON file: {config_dir}\VC_ratio_HwyOps.json

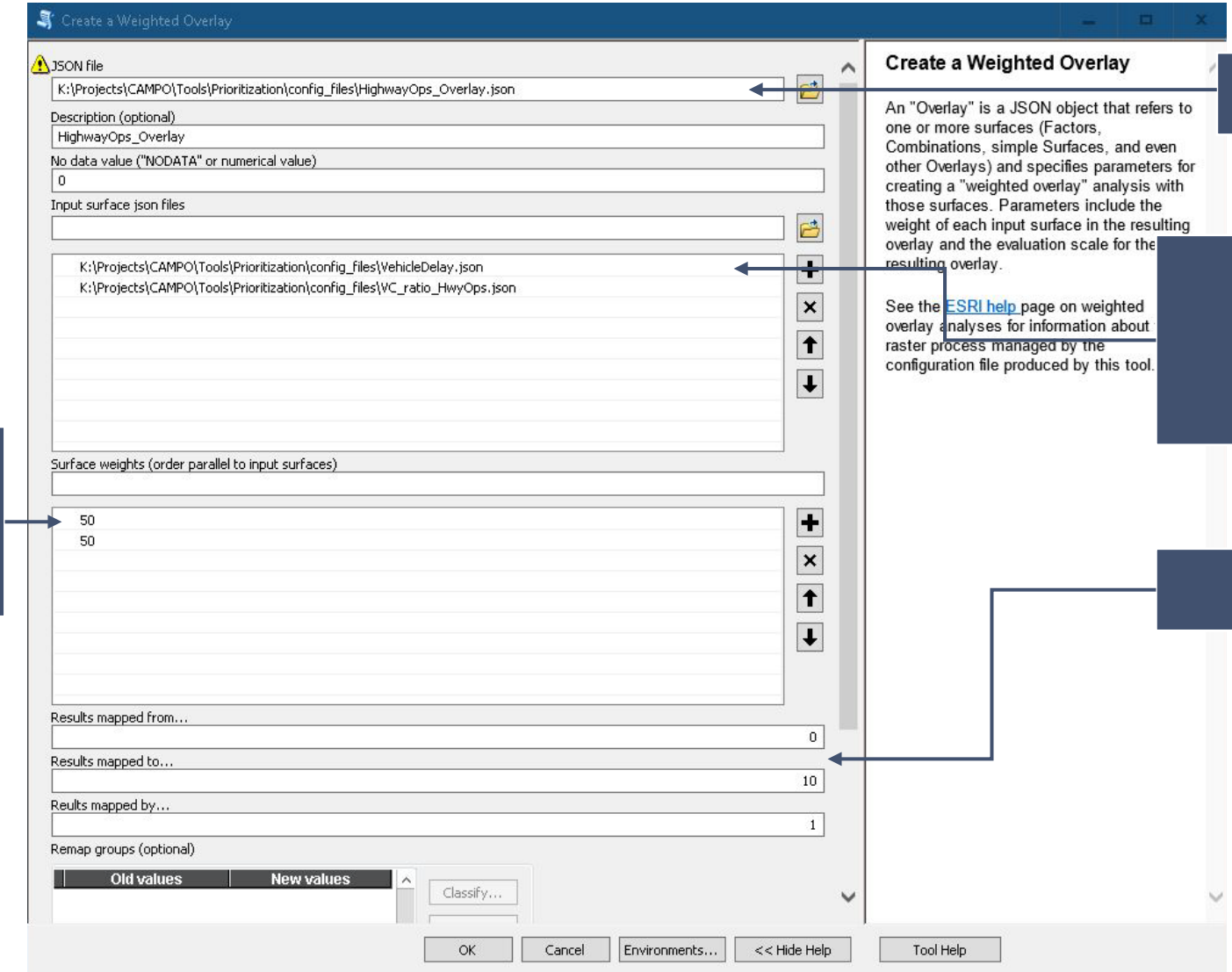
No data value: 0 (No suitability due to v/c ratio if no v/c ratio data in neighborhood)

Reference feature class: {inputs_gdb}\TRM_LoadedHwy_2045

Where clause: NEWFCLASS>2 AND FCLASS <>22 (only include non-limited access/non-tollway facilities)

Analysis method: MEAN

# SUITABILITY – HIGHWAY OPS – OVERLAY

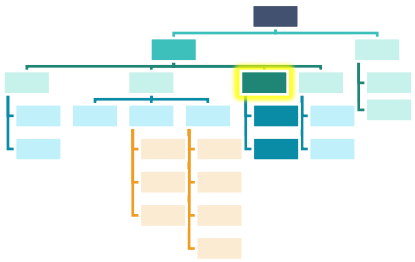



JSON file:
{config_dir}\HighwayOps_Overlay.json

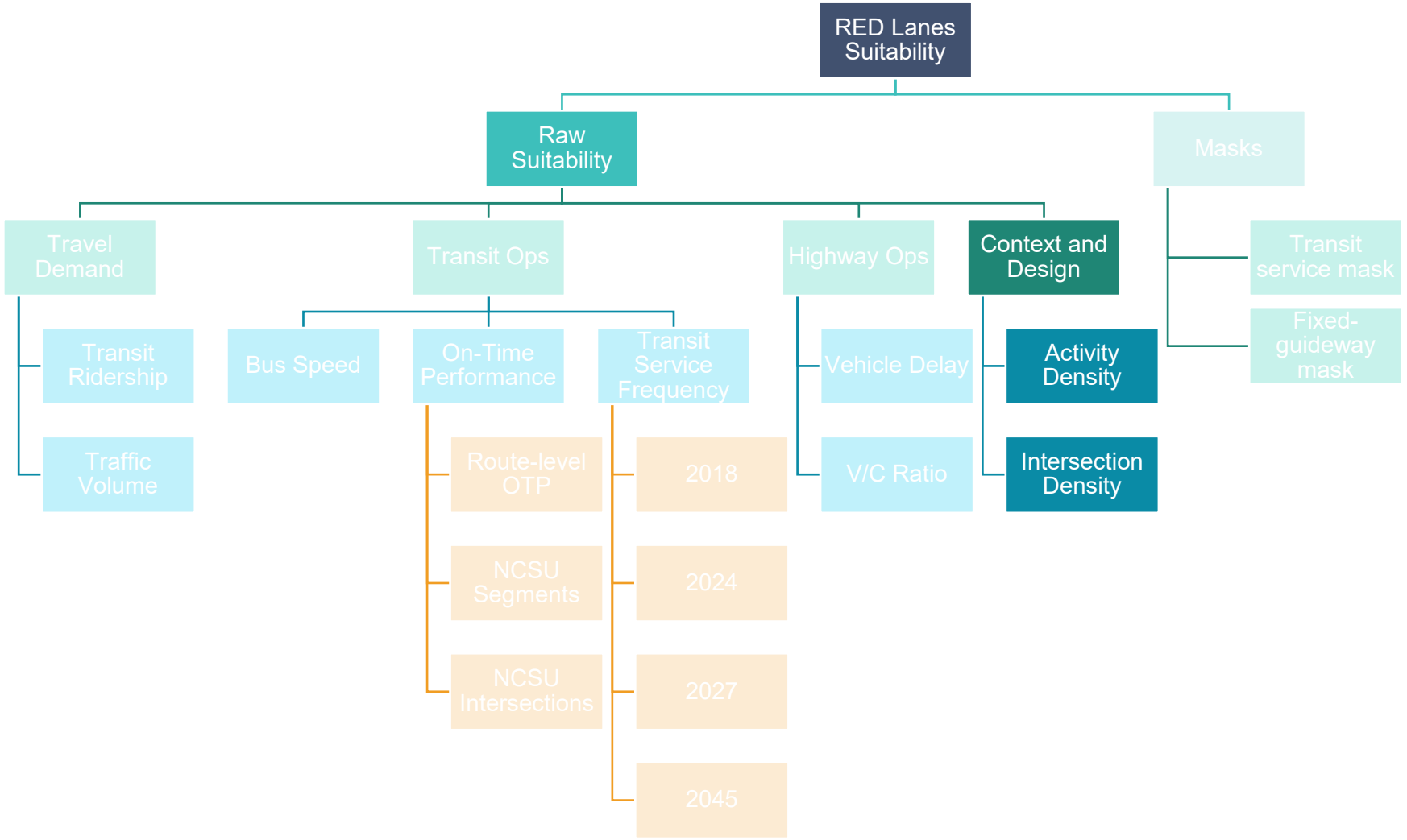
Input surface json files:
{config_dir}\VehicleDelay.json
{config_dir}\VC_ratio_HwyOps.json

Weights:
Vehicle delay: 50
VC ratio: 50

Results mapped from/to/by:
Defaults (0/10/1)



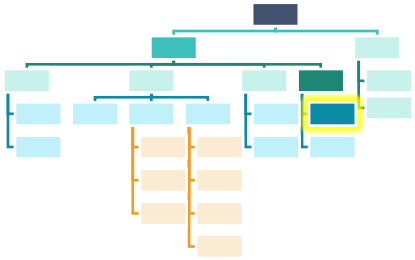
# SUITABILITY – CONTEXT AND DESIGN

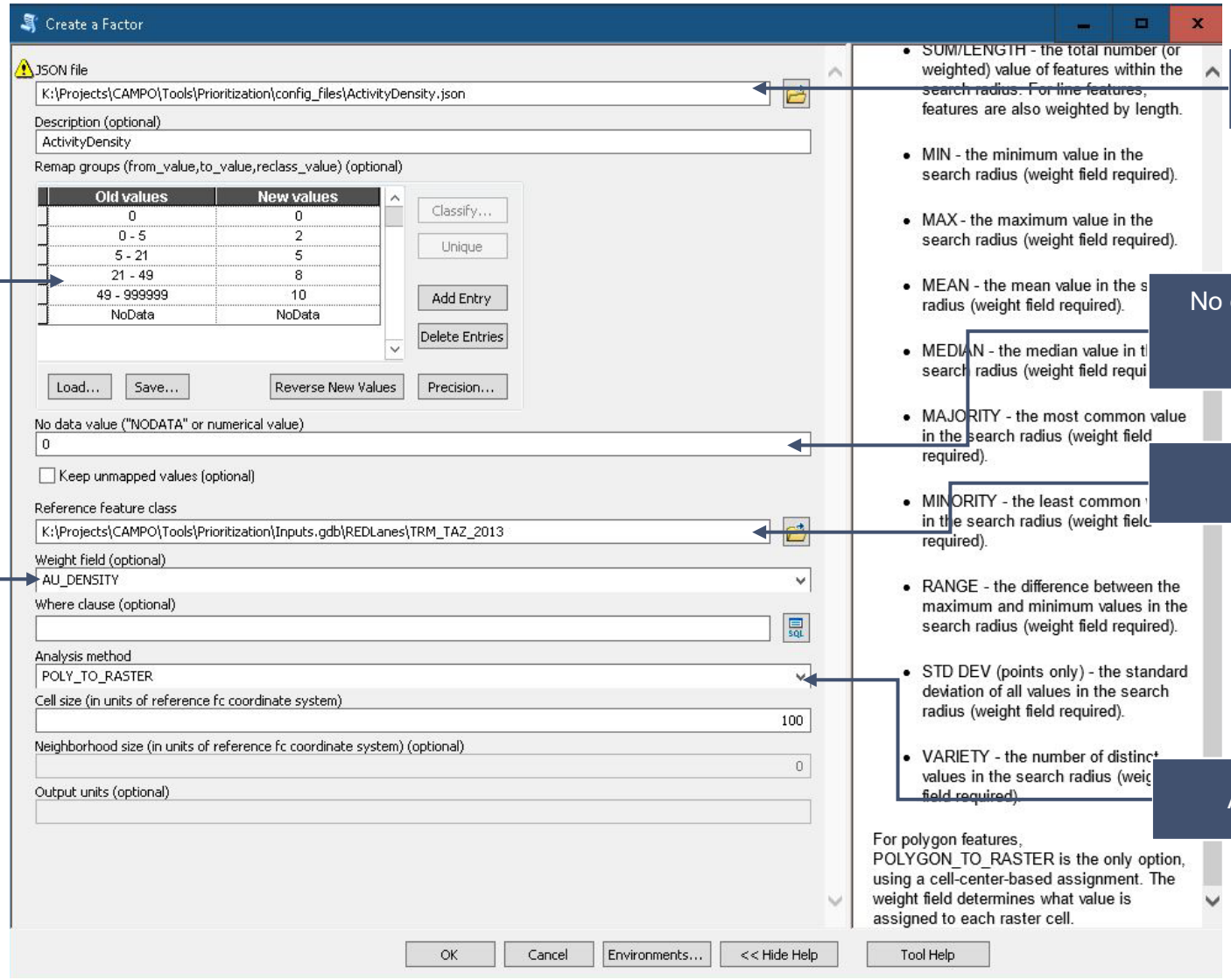


# SUITABILITY – CONTEXT & DESIGN – ACTIVITY DENSITY

Remap groups: load from {remaps_dir}/activity_density

Weight field: AU_DENSITY






JSON file: {config_dir}\ActivityDensity.json

No data value: 0 (No suitability due to activity density if no activity density data in neighborhood)

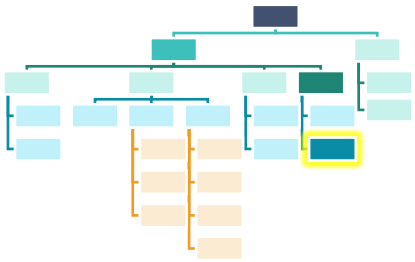
Reference feature class: {inputs_gdb}\TRM_TAZ_2013

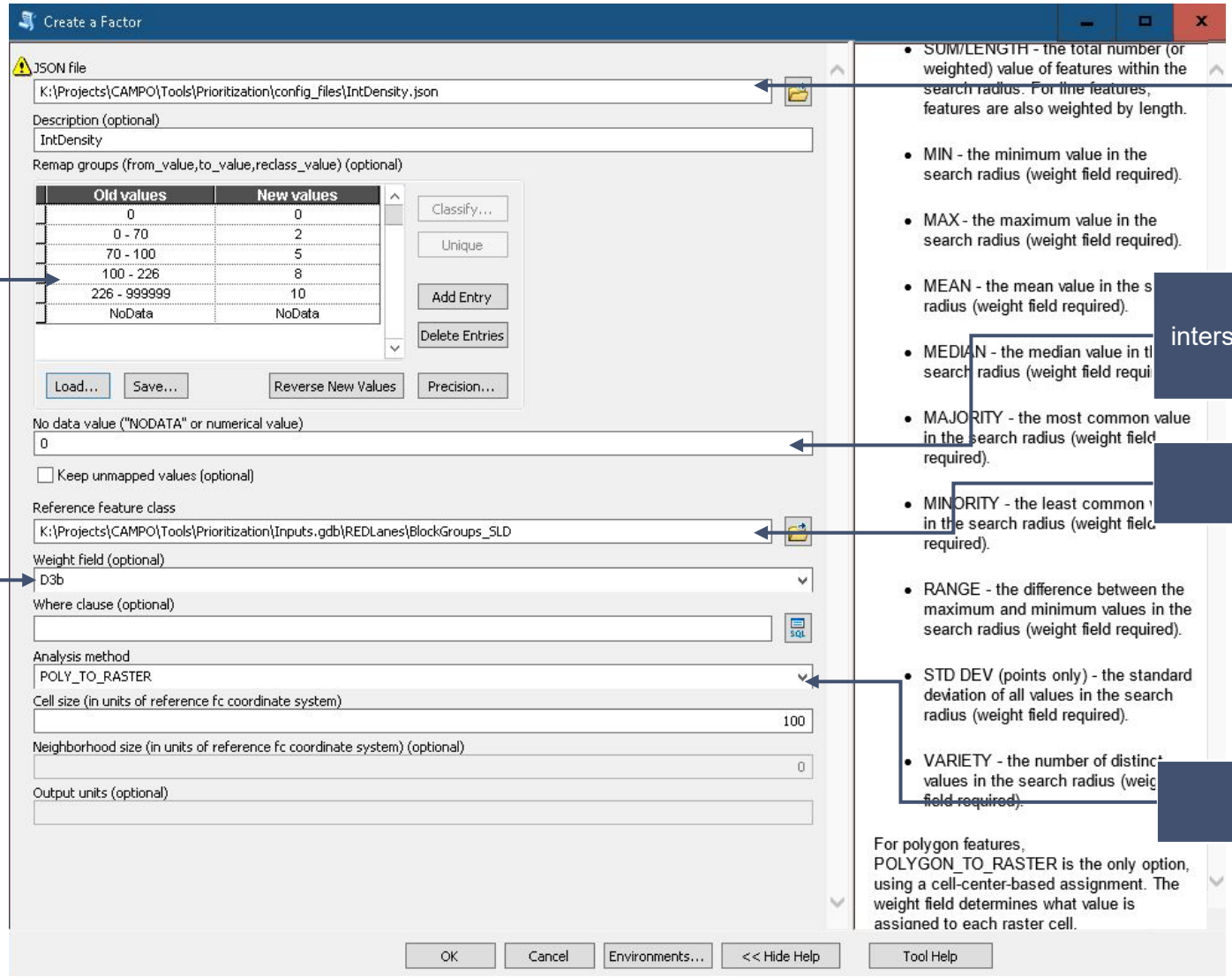
Analysis method: POLY_TO_RASTER

# SUITABILITY – CONTEXT & DESIGN – INTERSECTION DENSITY

Remap groups: load from {remaps_dir}/intersection_density

Weight field: D3b





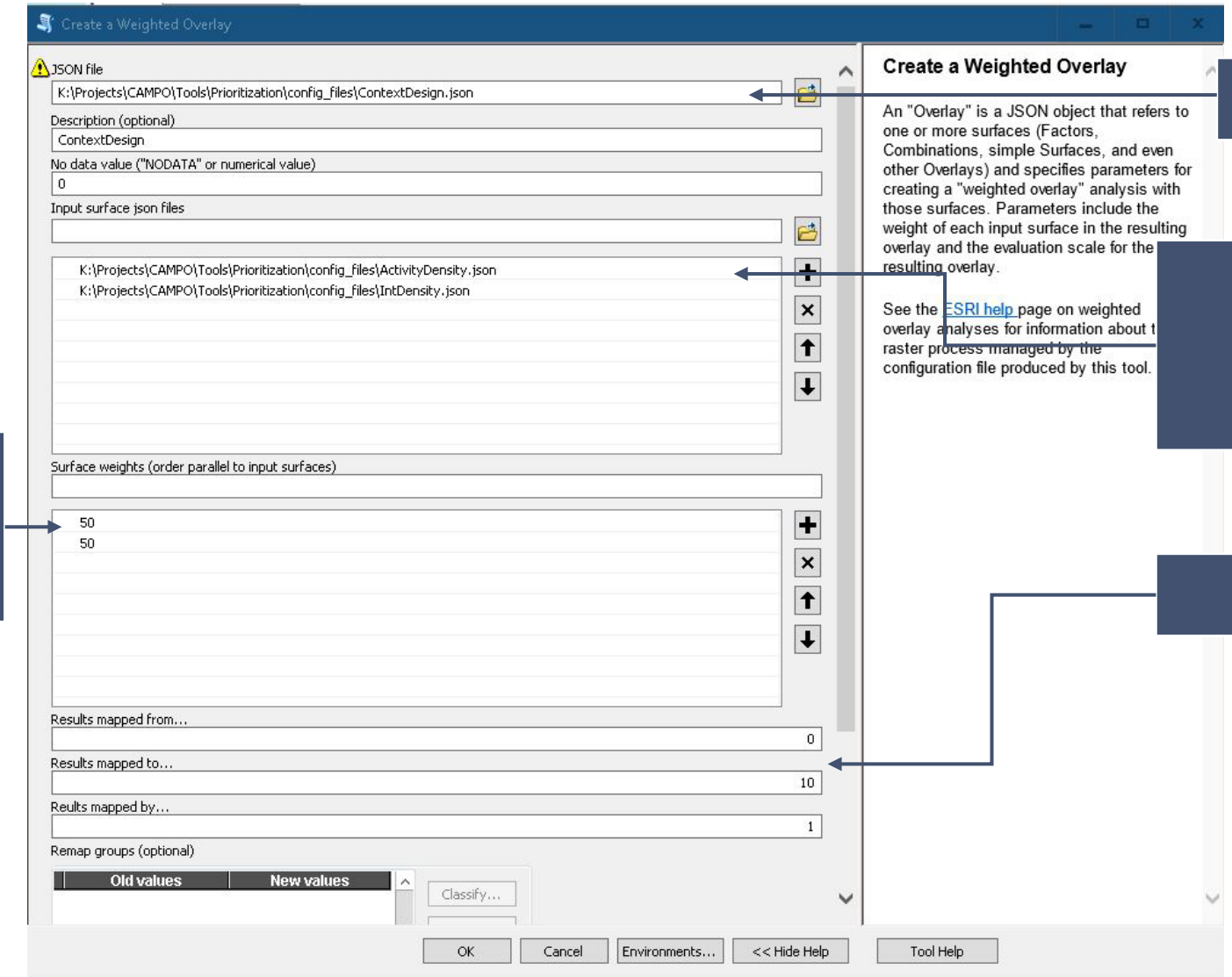

JSON file: {config_dir}\IntDensity.json

No data value: 0 (No suitability due to intersection density if no intersection density data in neighborhood)

Reference feature class: {inputs_gdb}\BlockGroups_SLD

Analysis method: POLY_TO_RASTER

# SUITABILITY – CONTEXT & DESIGN – OVERLAY

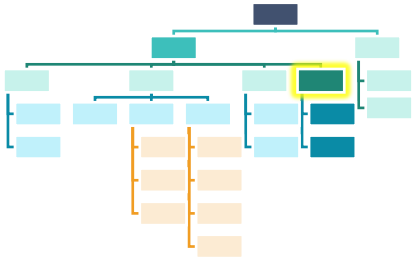



JSON file: {config_dir}\ContextDesign.json

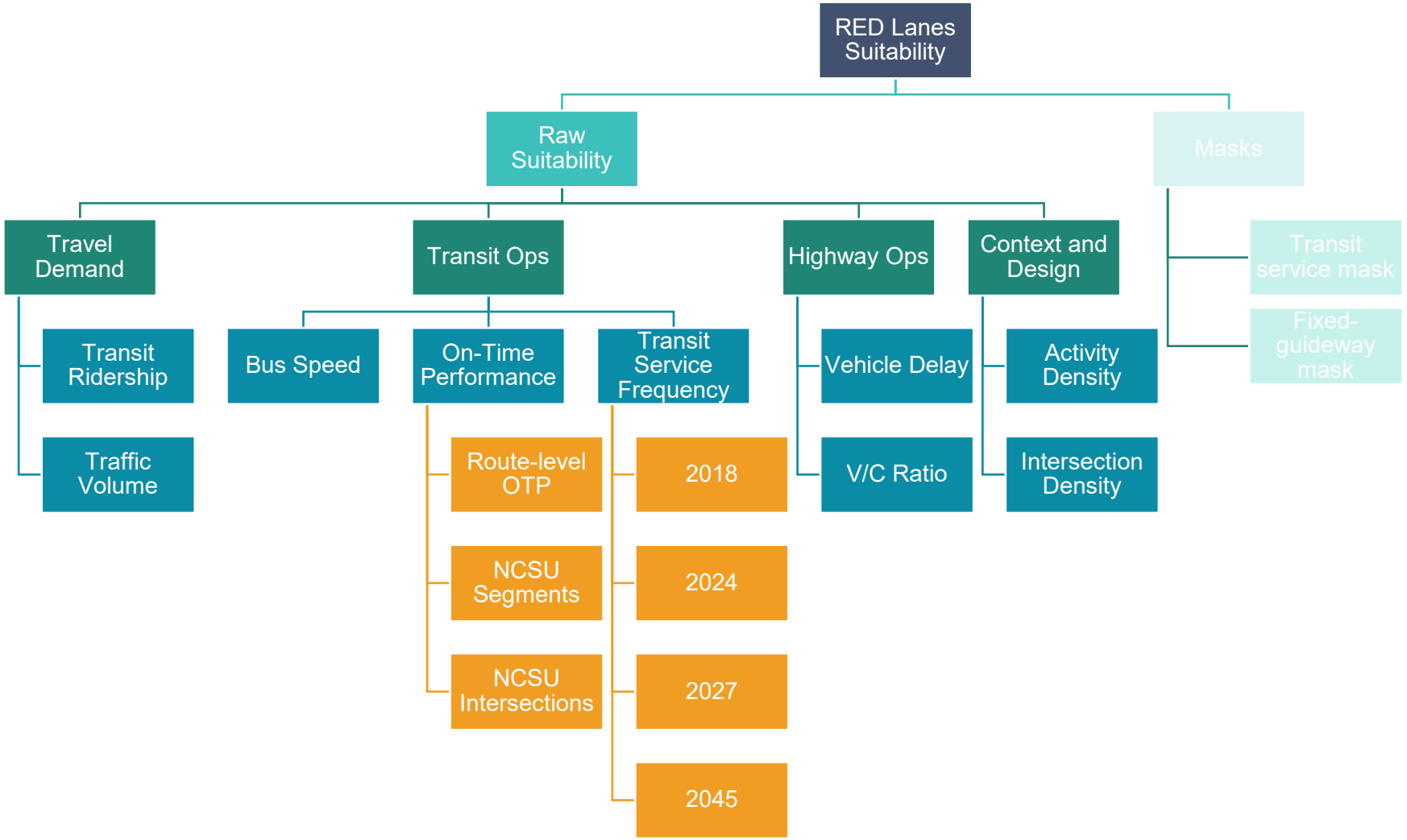
Input surface json files:
{config_dir}\ActivityDensity.json
{config_dir}\IntDensity.json

Weights:
Activity Density: 50
Intersection Density: 50

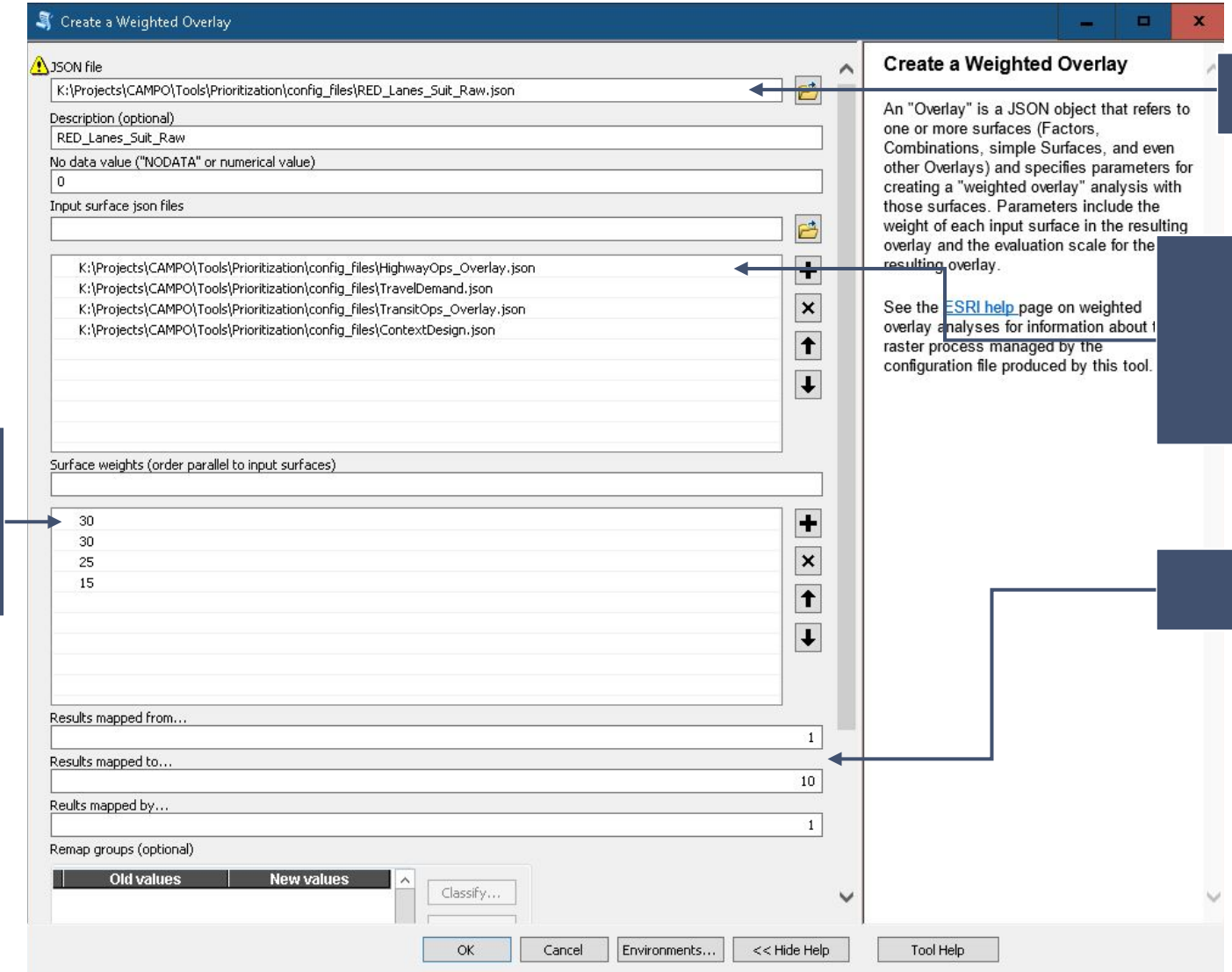
Results mapped from/to/by:
Defaults (0/10/1)



# SUITABILITY – RAW SUITABILITY



# SUITABILITY – RAW SUITABILITY – OVERLAY

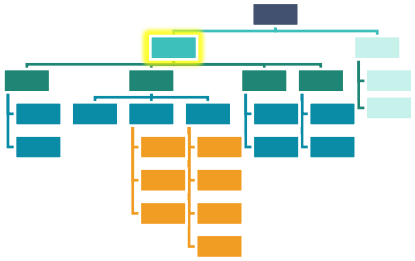



JSON file:
{config_dir}\RED_Lanes_Suit_Raw.json

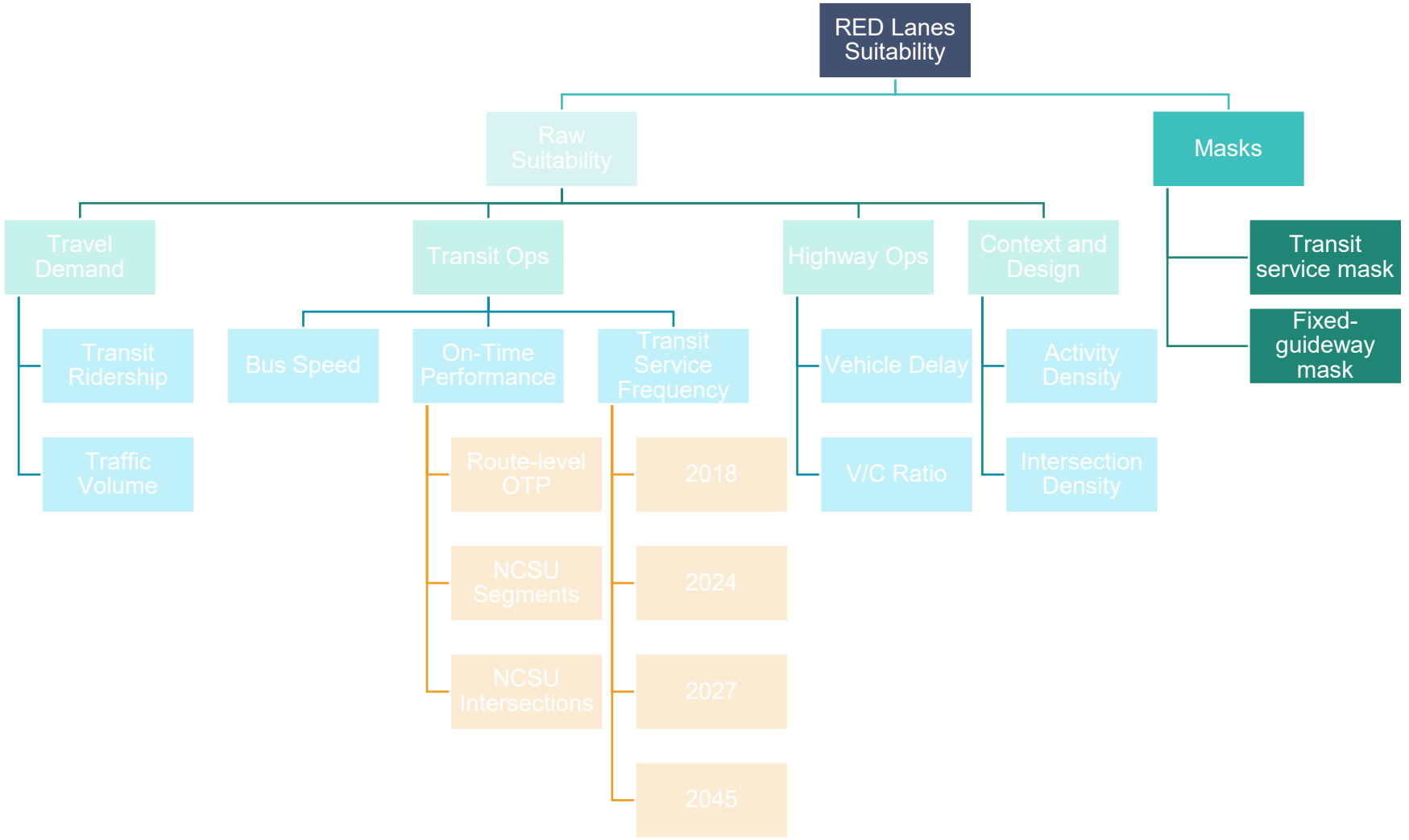
Input surface json files:
{config_dir}\HighwayOps_Overlay.json
{config_dir}\TravelDemand.json
{config_dir}\TransitOps_Overlay.json
{config_dir}\ContextDesign.json

Weights:
Highway Operations: 30
Travel Demand: 30
Transit Operations: 25
Context Design: 15

Results mapped from/to/by:
Defaults (0/10/1)



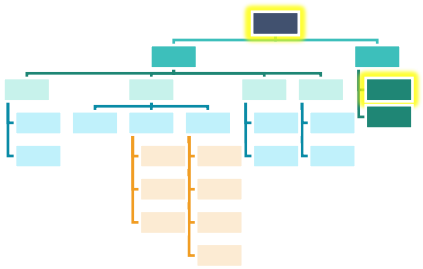
# SUITABILITY – MASKING RESULTS

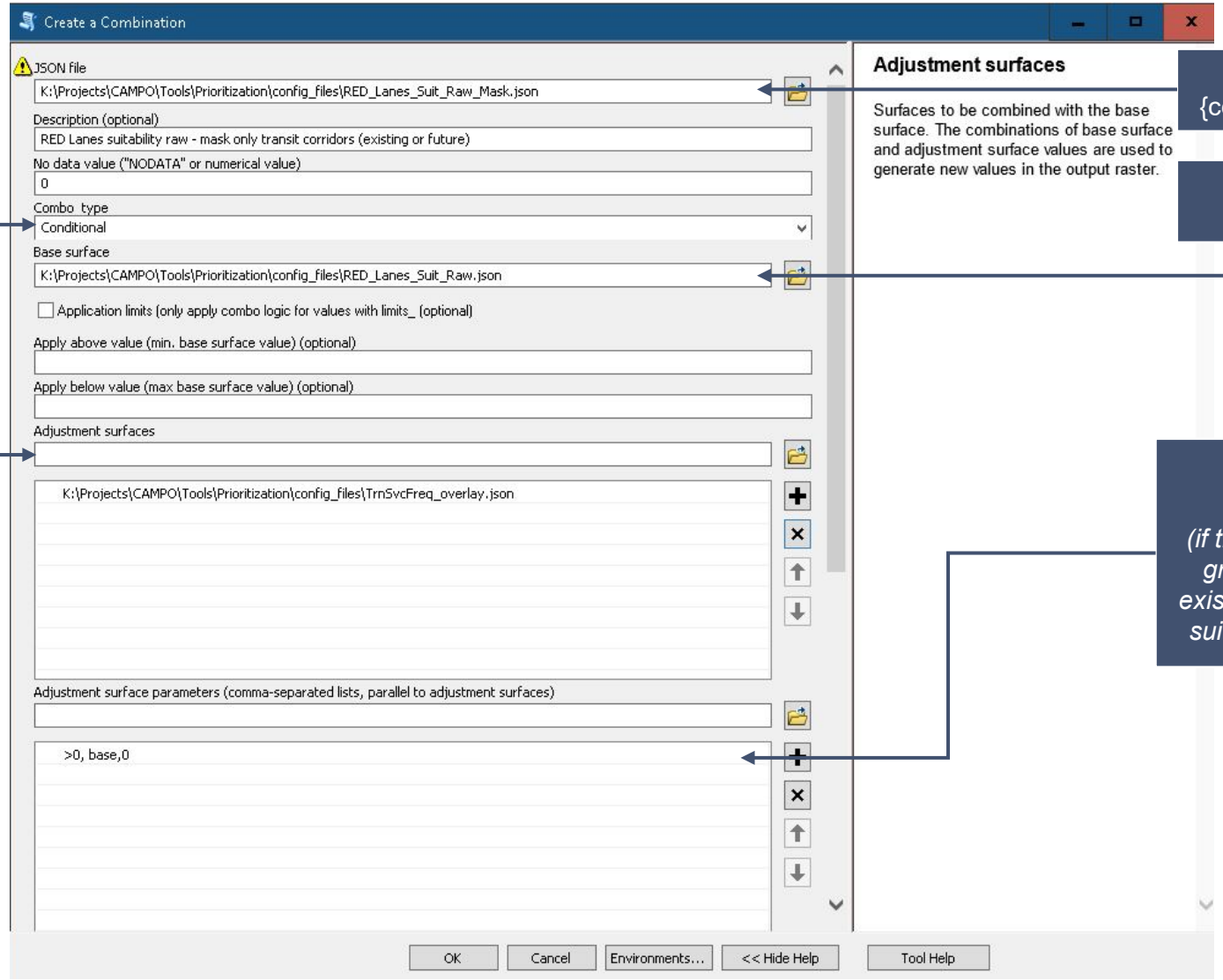


# SUITABILITY – MASKS – SUITABILITY COMBO W/ TRANSIT MASK

Combo type: Conditional

Adjustment surfaces:
{config_dir}\TrnSvcFreq_overlay






JSON file:
{config_dir}\RED_Lanes_Suit_Raw_Mask.json

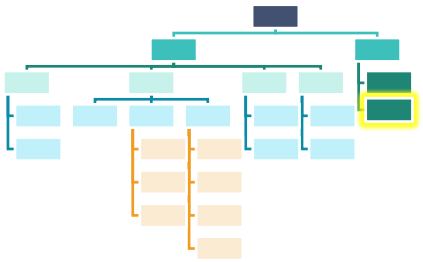
Base surface:
{config_dir}\RED_Lanes_Suit_Raw.json

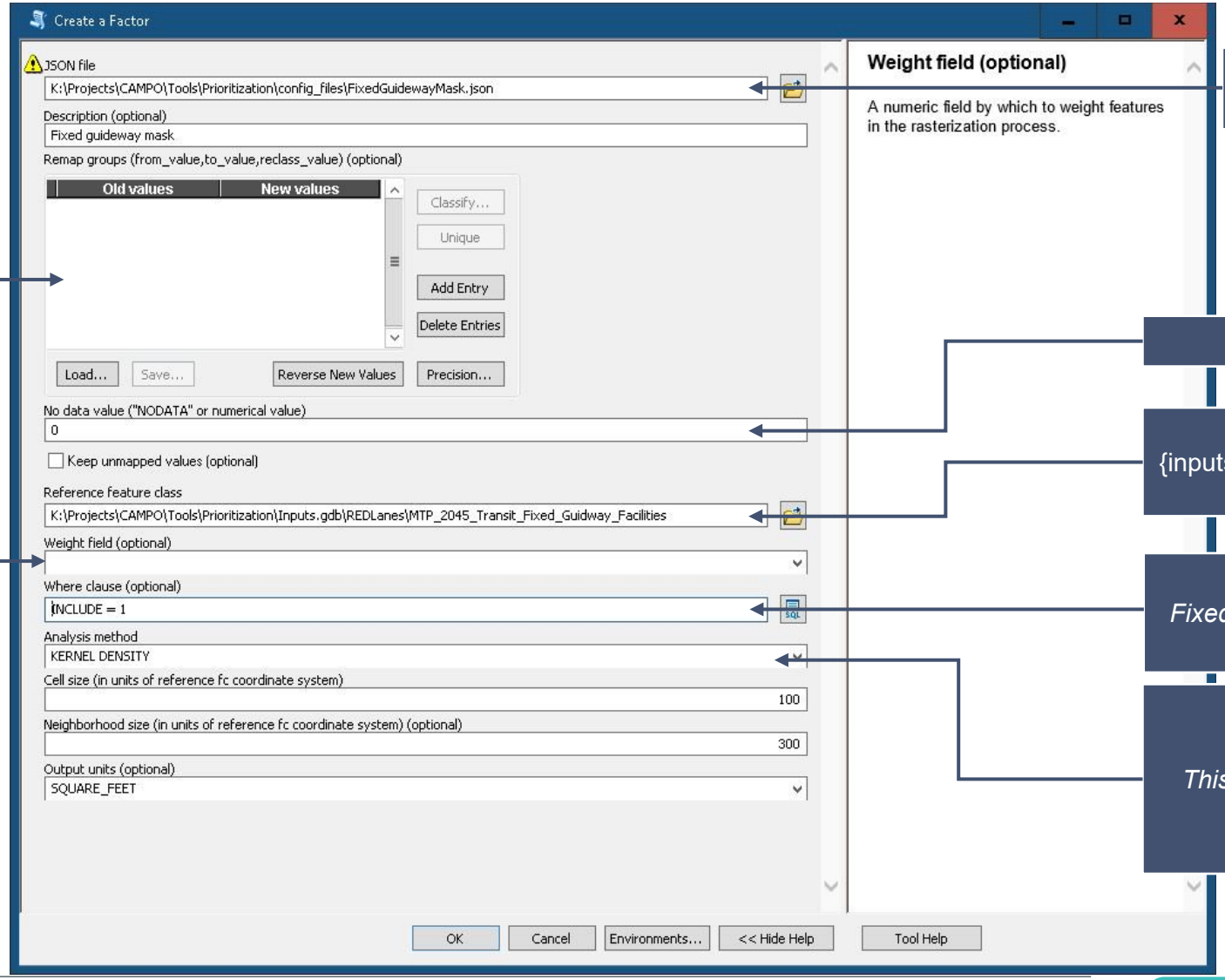
Adjustment surface parameters:
>0,base,0
*(if the transit service frequency overlay score is greater than zero [i.e., there is at least some existing or planned transit service], keep the raw suitability score, else set the cell value to zero)*

# SUITABILITY – MASKS – CREATING A FIXED GUIDEWAY MASK

Remap groups: (none)

Weight field: (none)






JSON file: {config_dir}\FixedGuidewayMask.json

No data value: 0

Reference feature class: {inputs_gdb}\MTP_2045_Transit_Fixed_Guidewa y_Facilitiies

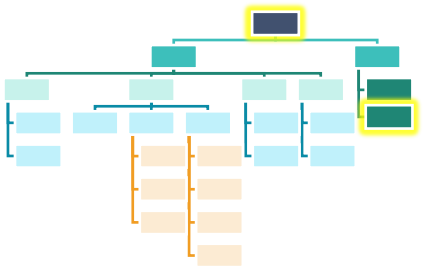
Where clause: INCLUDE=1
*Fixed guideway facilities that should be masked are flagged in this field.*

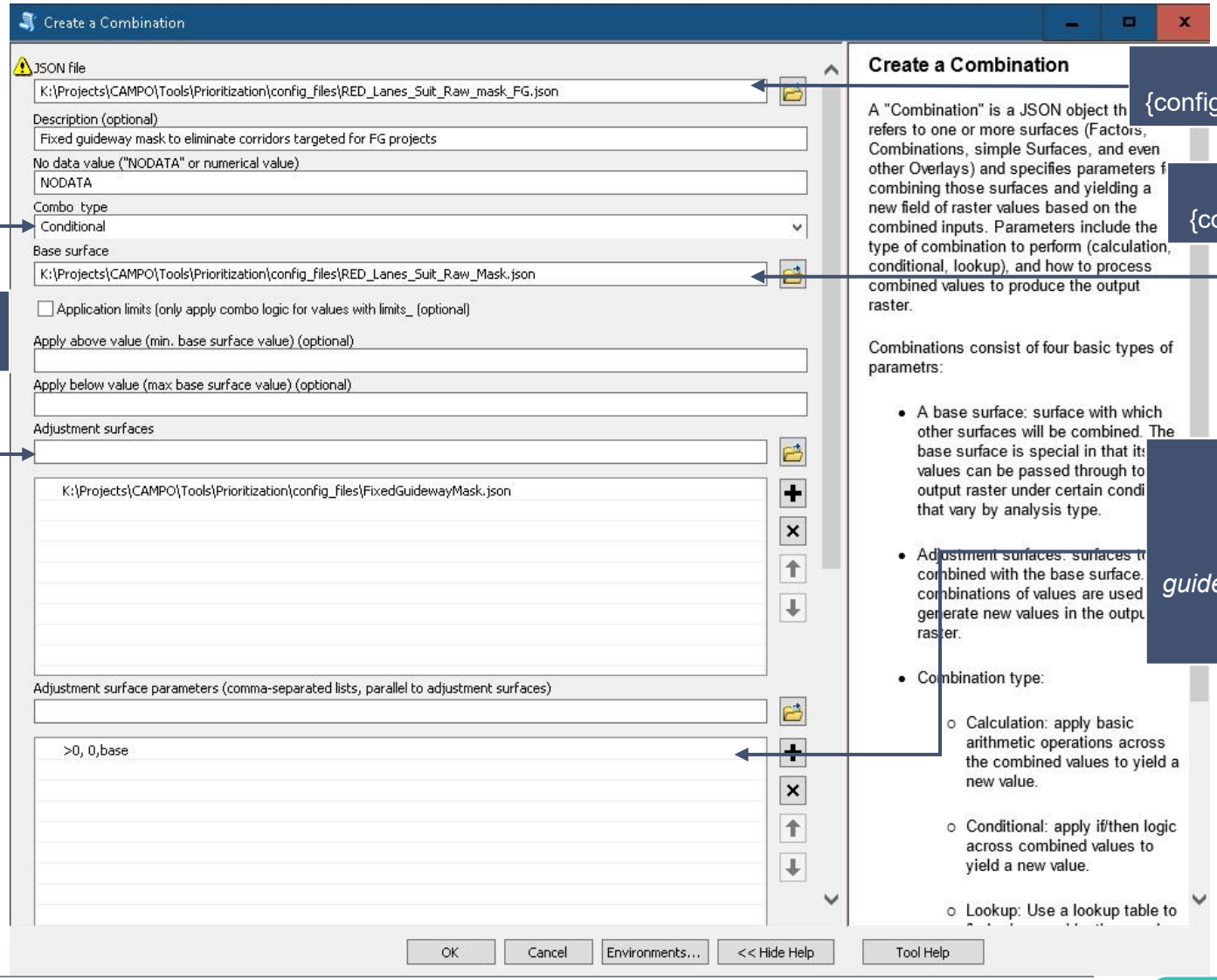
Analysis method: KERNEL DENSITY
*This allows us to use no weight field, and any value > 0 can be used downstream*

# SUITABILITY – MASKS – ADDING THE FIXED GUIDEWAY MASK

Combo type: Conditional

Adjustment surfaces:
{config_dir}\FixedGuidewaMask.json



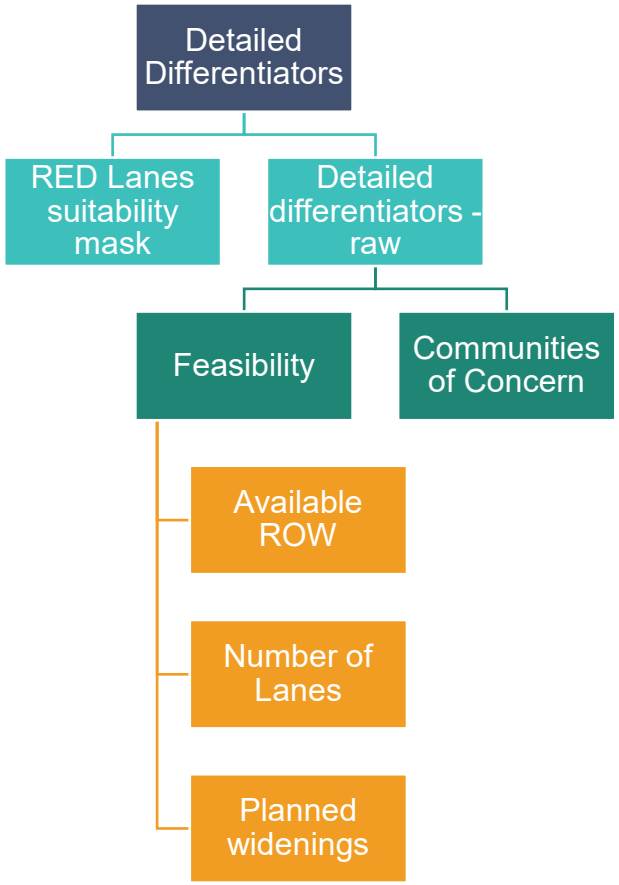



JSON file:
{config_dir}\RED_Lanes_Suit_Raw_mask_FG.json

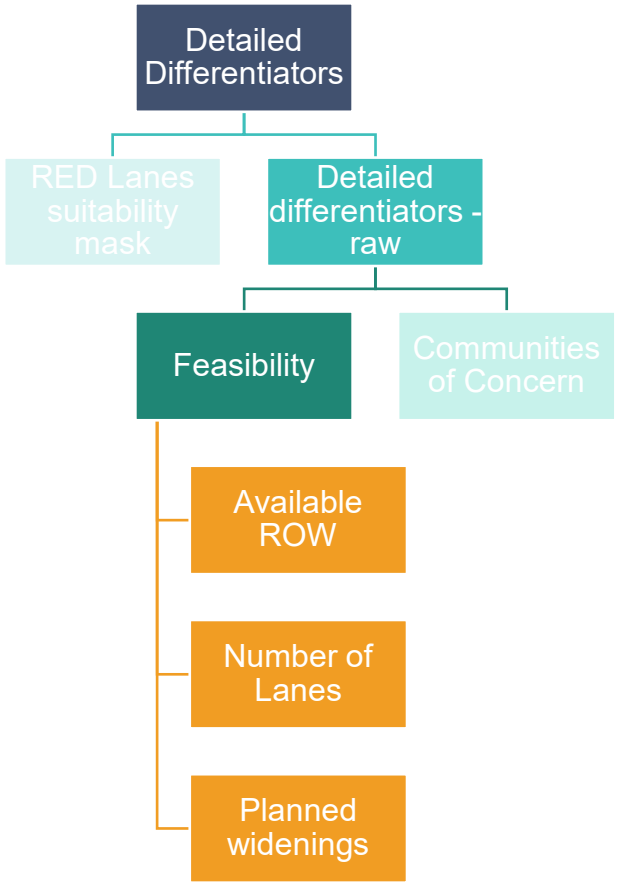
Base surface:
{config_dir}\RED_Lanes_Suit_Raw_Mask.json

Adjustment surface parameters:
>0,0,base
*(if the cell overlaps with a planned fixed guideway project, set the value to zero, otherwise retain the base value)*

# DETAILED DIFFERENTIATORS



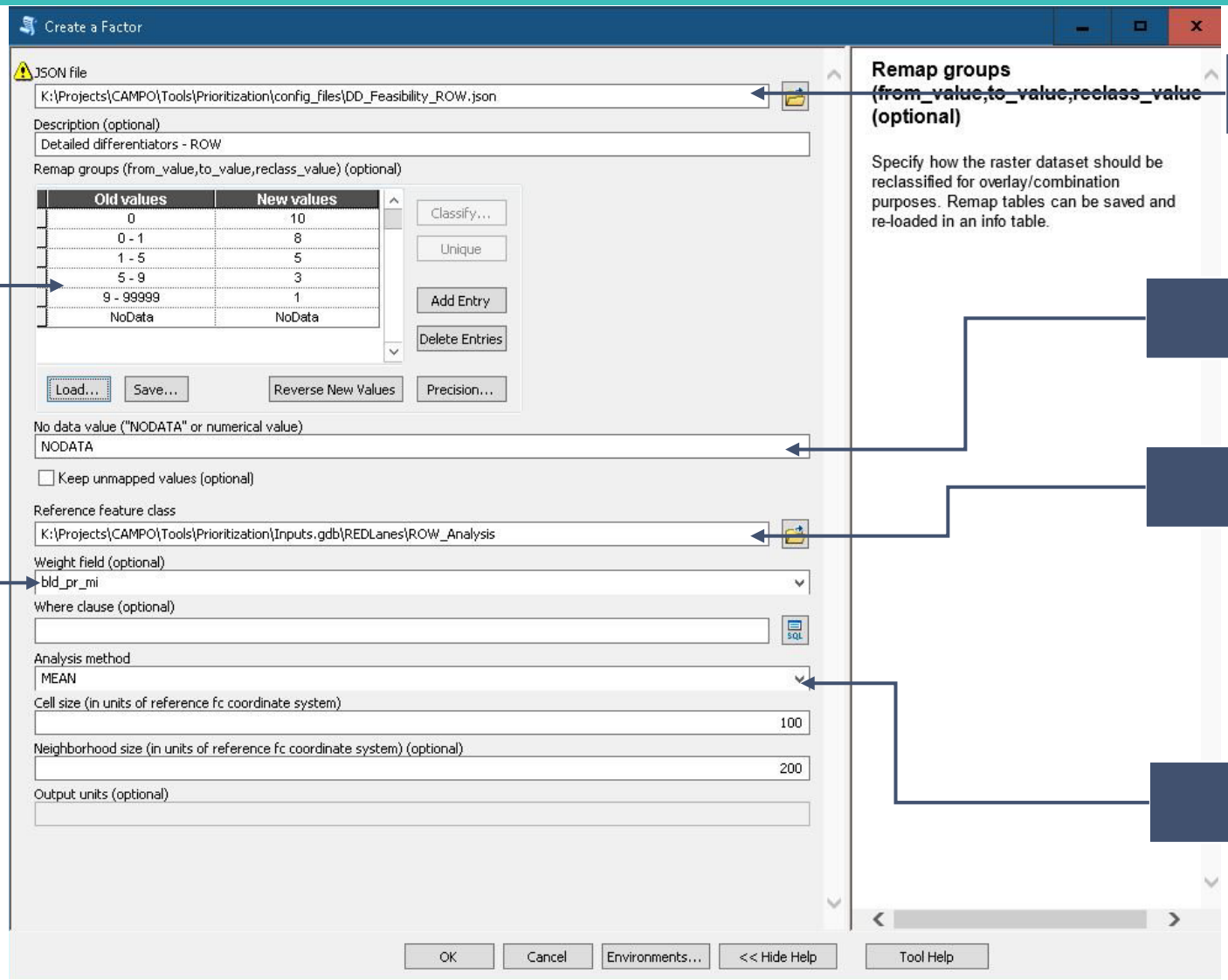
# DETAILED DIFFERENTIATORS - FEASIBILITY



# DETAILED DIFFERENTIATORS – FEASIBILITY – AVAILABLE ROW

Remap groups: load from {remaps_dir}/DD_Feasibility_ROW

Weight field: bld_pr_mi



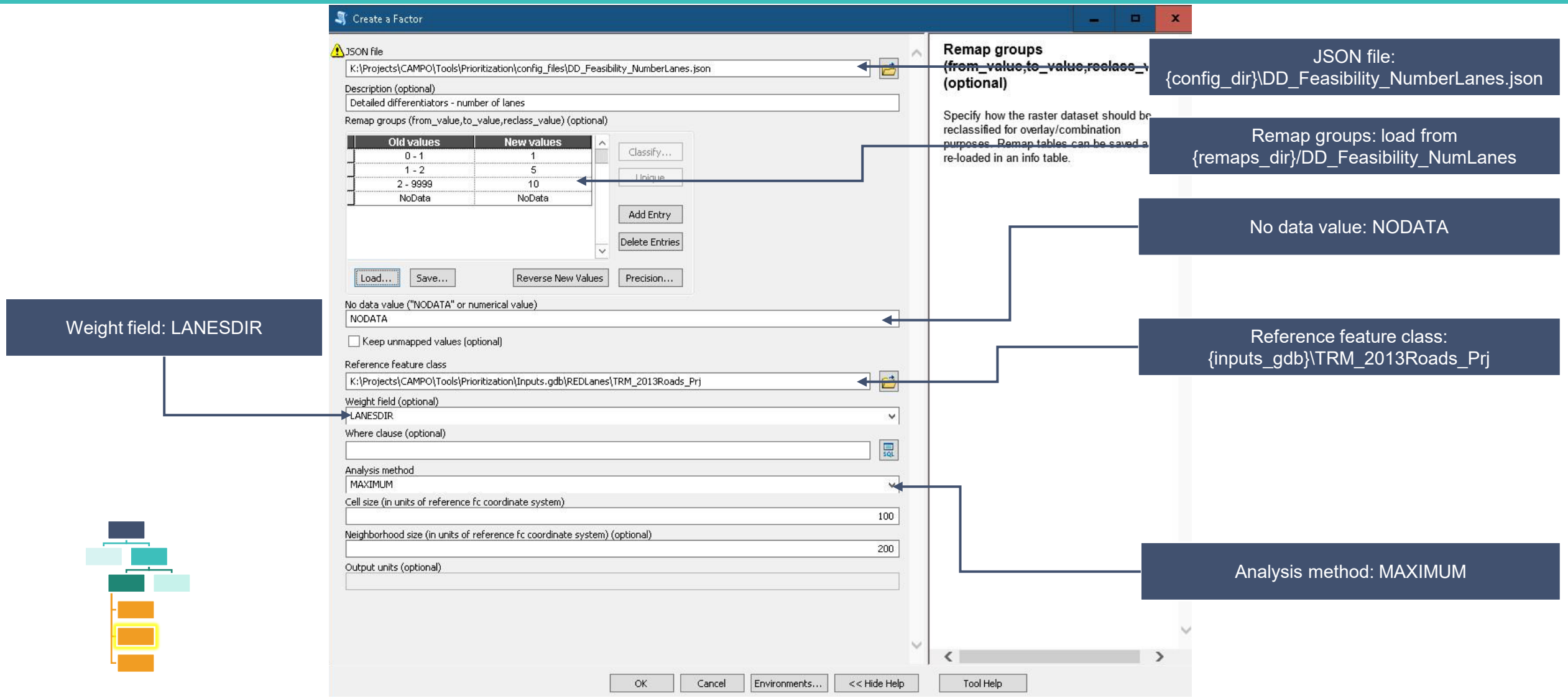

JSON file: {config_dir}\DD_Feasibility_ROW.json

No data value: NODATA

Reference feature class: {inputs_gdb}\ROW_Analysis

Analysis method: MEAN

# DETAILED DIFFERENTIATORS – FEASIBILITY – WIDENING




JSON file: {config_dir}\DD_Feasibility_NumberLanes.json

Remap groups: load from {remaps_dir}/DD_Feasibility_NumLanes

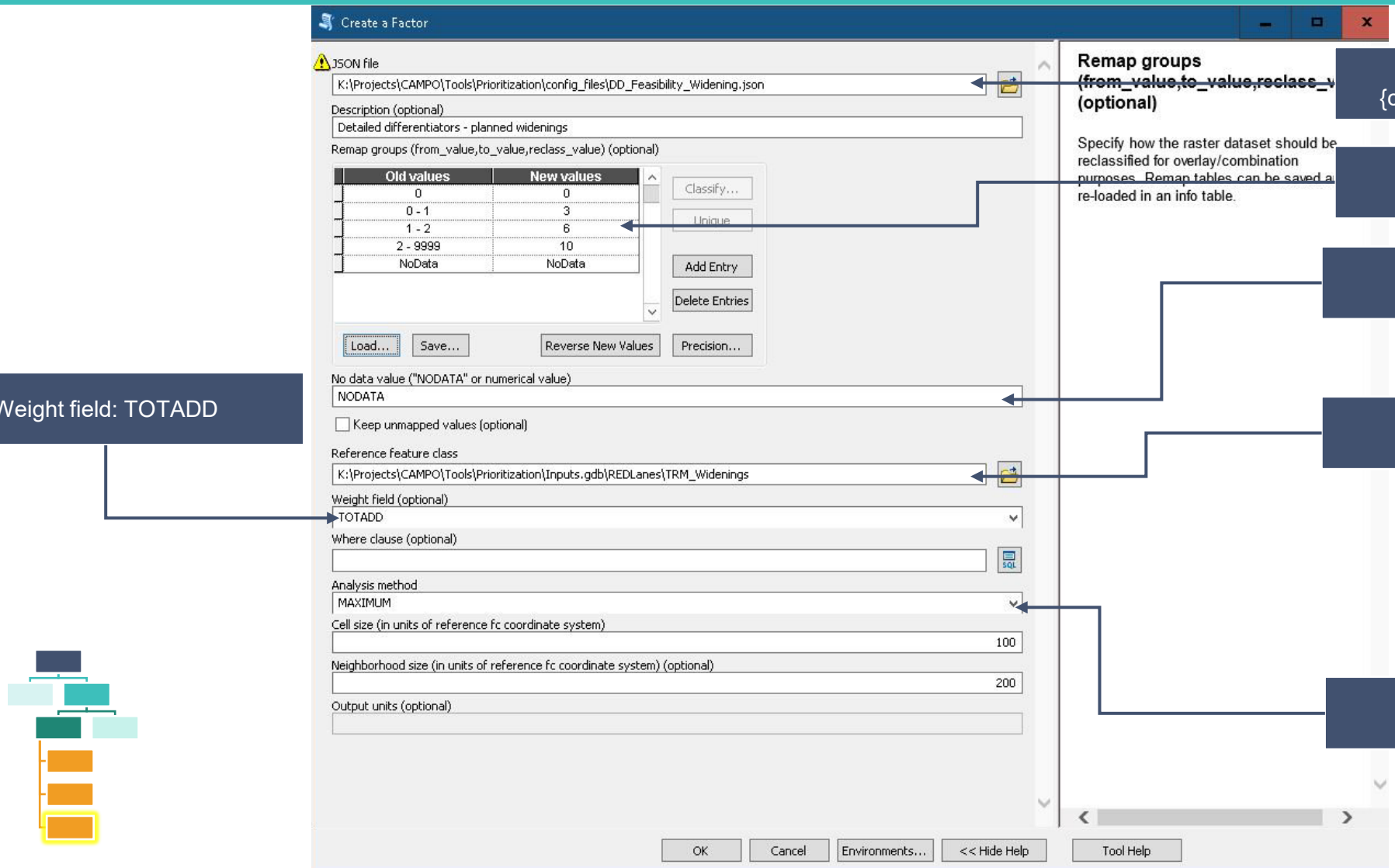
No data value: NODATA

Weight field: LANESDIR

Reference feature class: {inputs_gdb}\TRM_2013Roads_Prj

Analysis method: MAXIMUM

# DETAILED DIFFERENTIATORS – FEASIBILITY – WIDENING




JSON file: {config_dir}\DD_Feasibility_Widening.json

Remap groups: load from {remaps_dir}/DD_Feasibility_Widening

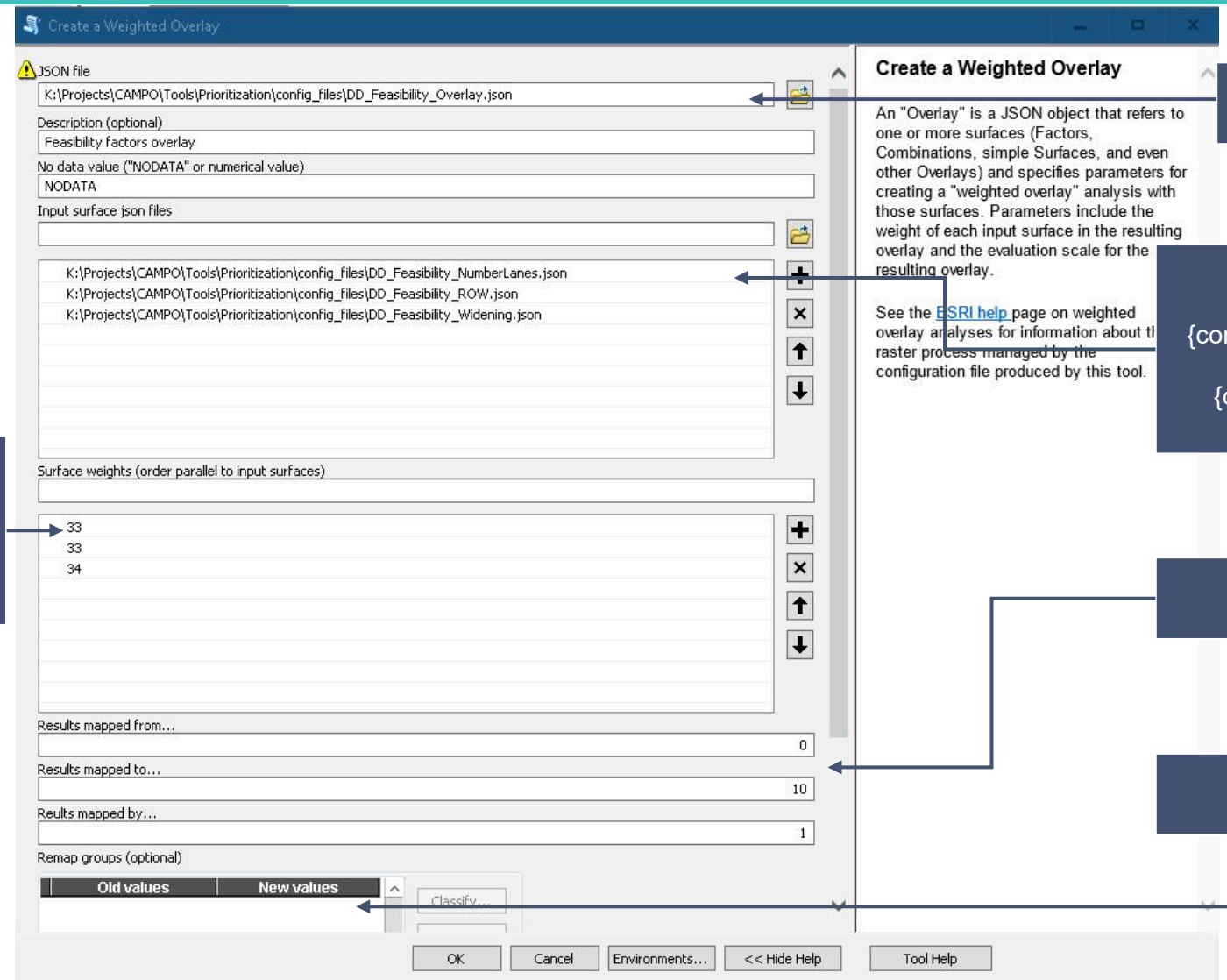
No data value: NODATA

Weight field: TOTADD

Reference feature class: {inputs_gdb}\TRM_Widenings

Analysis method: MAXIMUM

# DETAILED DIFFERENTIATORS – FEASIBILITY – OVERLAY




JSON file:
{config_dir}\DD_Feasibility_Overlay.json

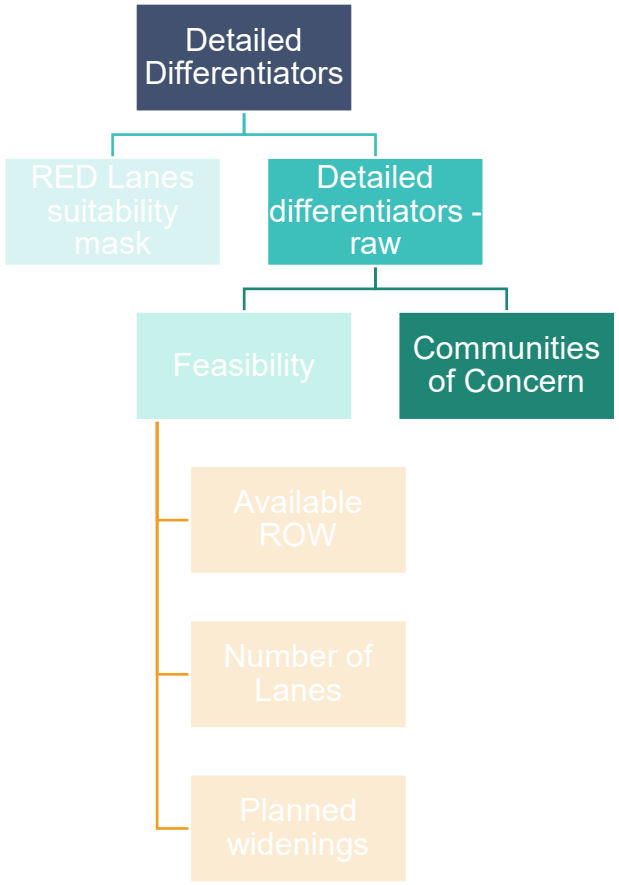
Input surface json files:
{config_dir}\DD_Feasibility_NumberLanes.json
{config_dir}\DD_Feasibility_ROW.json
{config_dir}\DD_Feasibility_Widening.json

Weights:
Number of Lanes: 33
Available ROW: 33
Widenings: 34

Results mapped from/to/by:
Defaults (0/10/1)

Remap groups: load from
{remaps_dir}/DD_Feasibility_Overlay

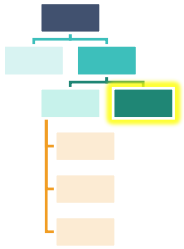
# DETAILED DIFFERENTIATORS – COMMUNITIES OF CONCERN

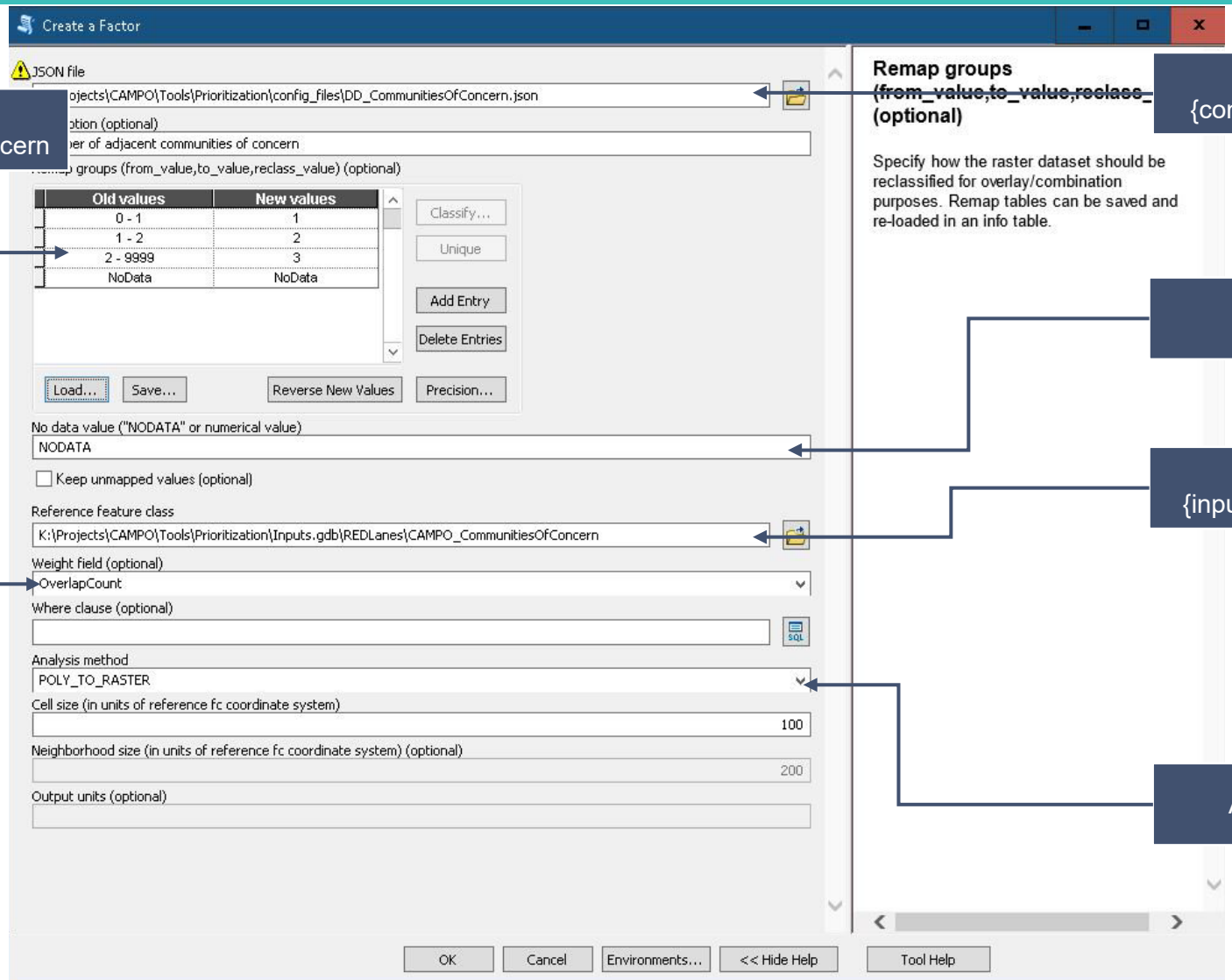


# DETAILED DIFFERENTIATORS – COMMUNITIES OF CONCERN

Remap groups: load from {remaps_dir}/DD_CommunitiesOfConcern

Weight field: OverlapCount





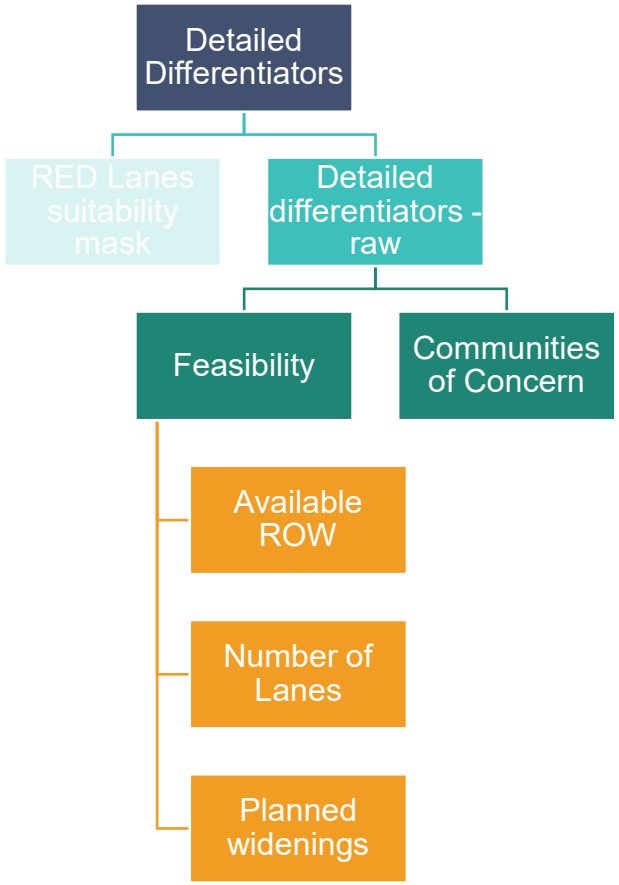

JSON file: {config_dir}\DD_CommunitiesOfConcern.json

No data value: NODATA

Reference feature class: {inputs_gdb}\CAMPO_CommunitiesofConcern

Analysis method: POLY_TO_RASTER

# DETAILED DIFFERENTIATORS – RAW COMBO

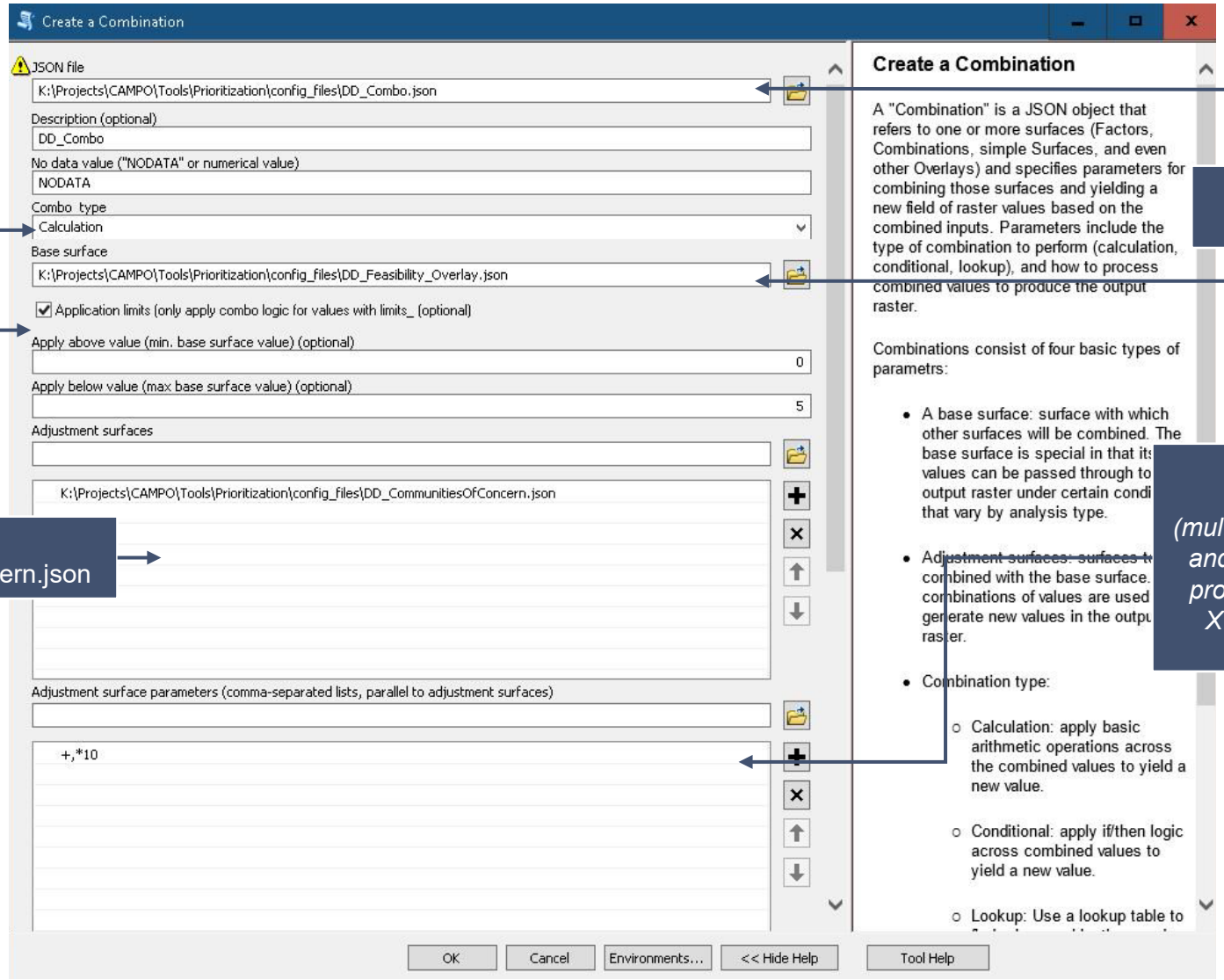


# DETAILED DIFFERENTIATORS – RAW COMBO

Combo type: Calculation

Application limits:
Apply above: 0
Apply Below: 5

Adjustment surfaces:
{config_dir}\DD_CommunitiesOfConcern.json

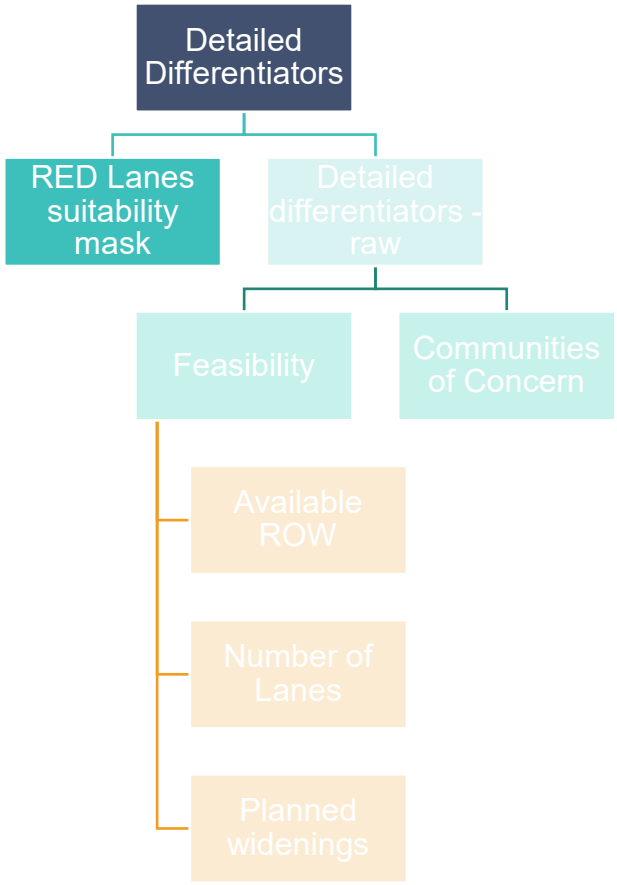



JSON file: {config_dir}\DD_Combo.json

Base surface:
{config_dir}\DD_Feasibility_Overlay.json

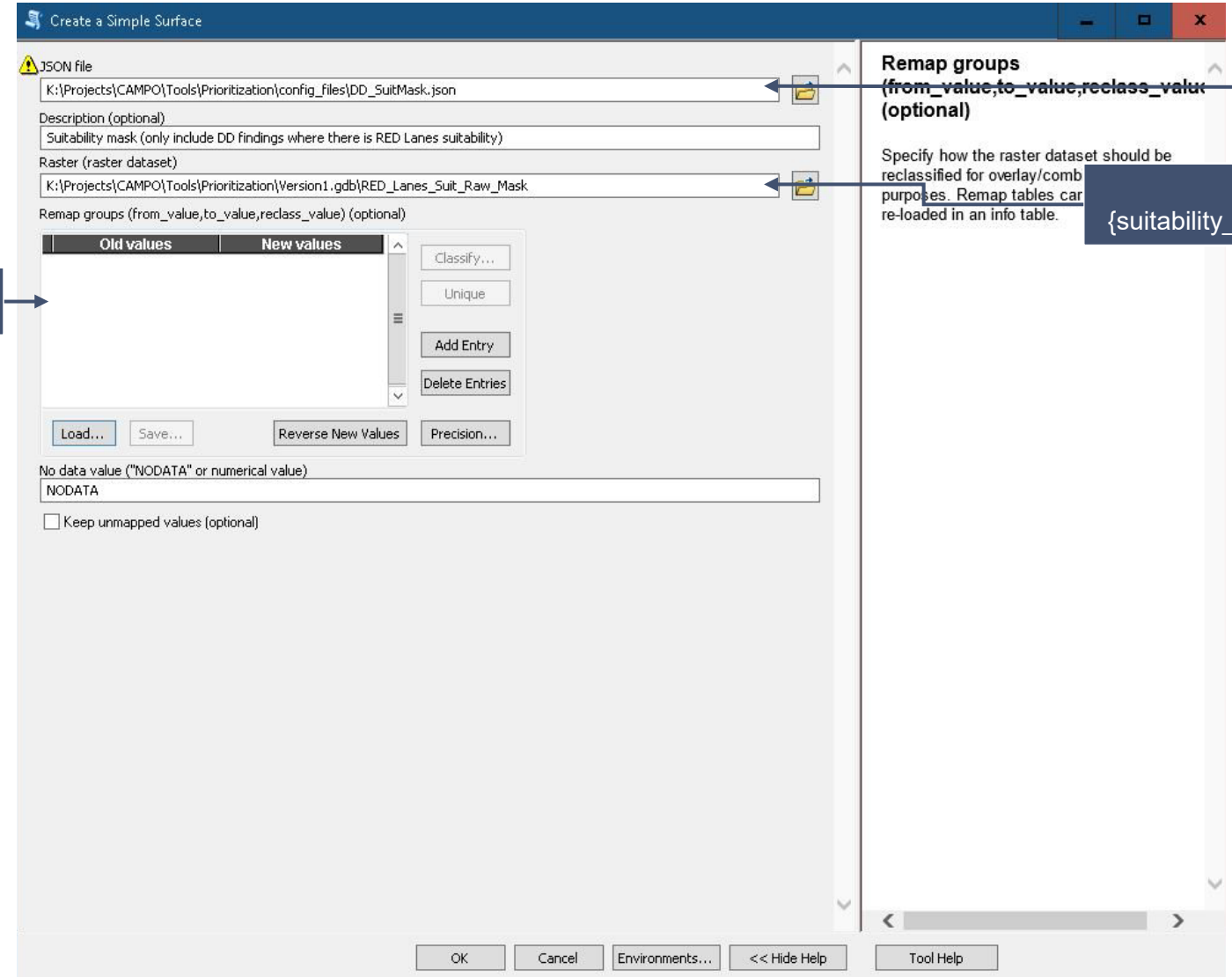
Adjustment surface parameters:
+,*10
*(multiply the communities of concern value by 10 and add it to the feasibility overlay value – this produces a raster with two-digit output values, XY, where X is the CofC score and Y is the feasibility score)*

# DETAILED DIFFERENTIATORS – MASKING



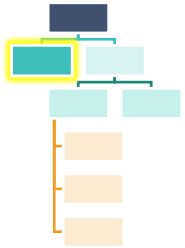
# DETAILED DIFFERENTIATORS – SUITABILITY MASK

Remap groups: (none)




JSON file:
{config_dir}\DD_SuitMask.json

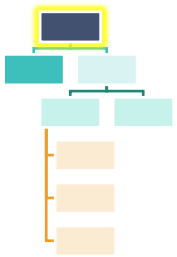
Raster:
{suitability_outputs_gdb}\RED_Lanes_Suit_Raw_Mask

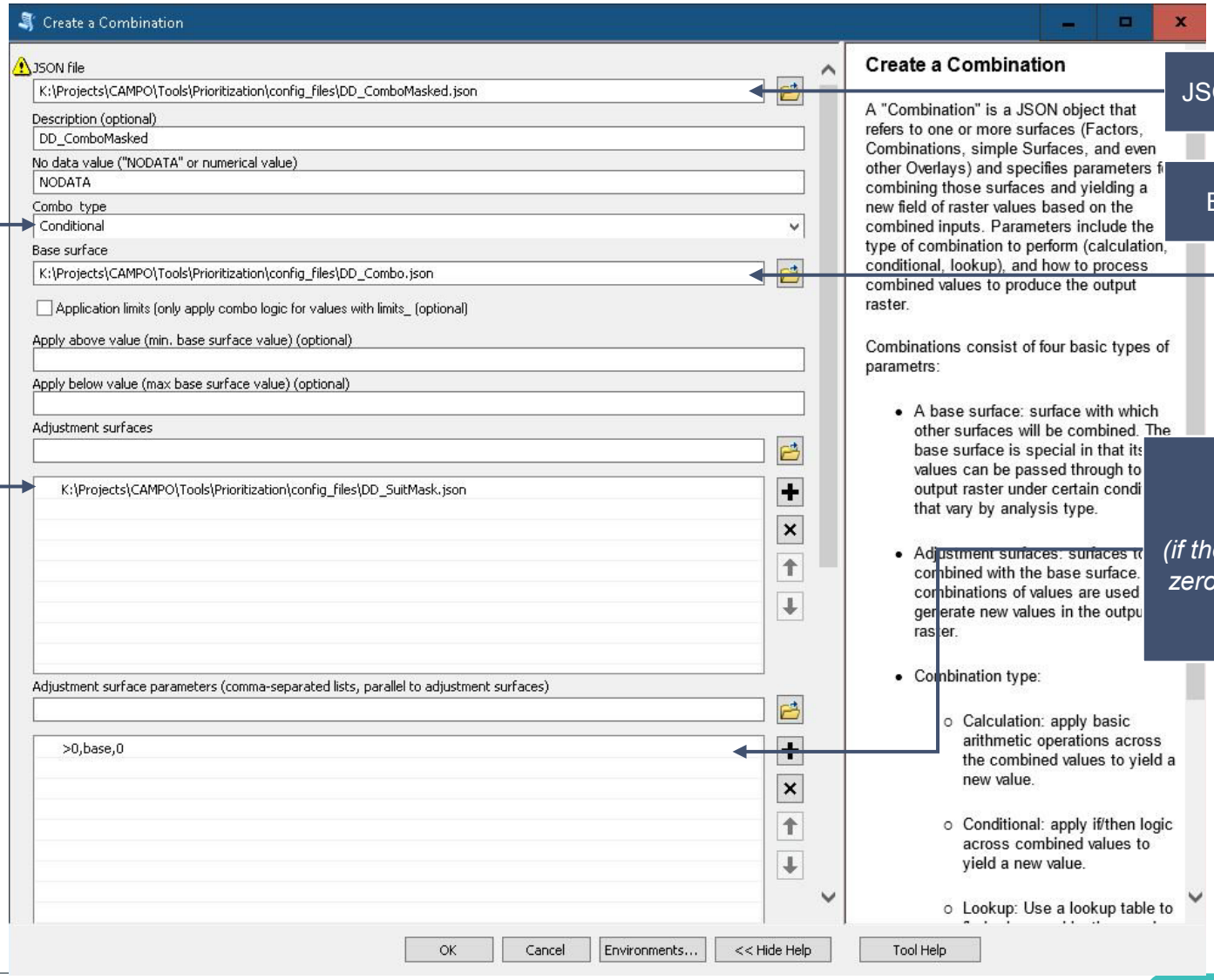


# DETAILED DIFFERENTIATORS – ADDING THE SUITABILITY MASK

Combo type: Conditional

Adjustment surfaces: {config_dir}\DD_SuitMask.json



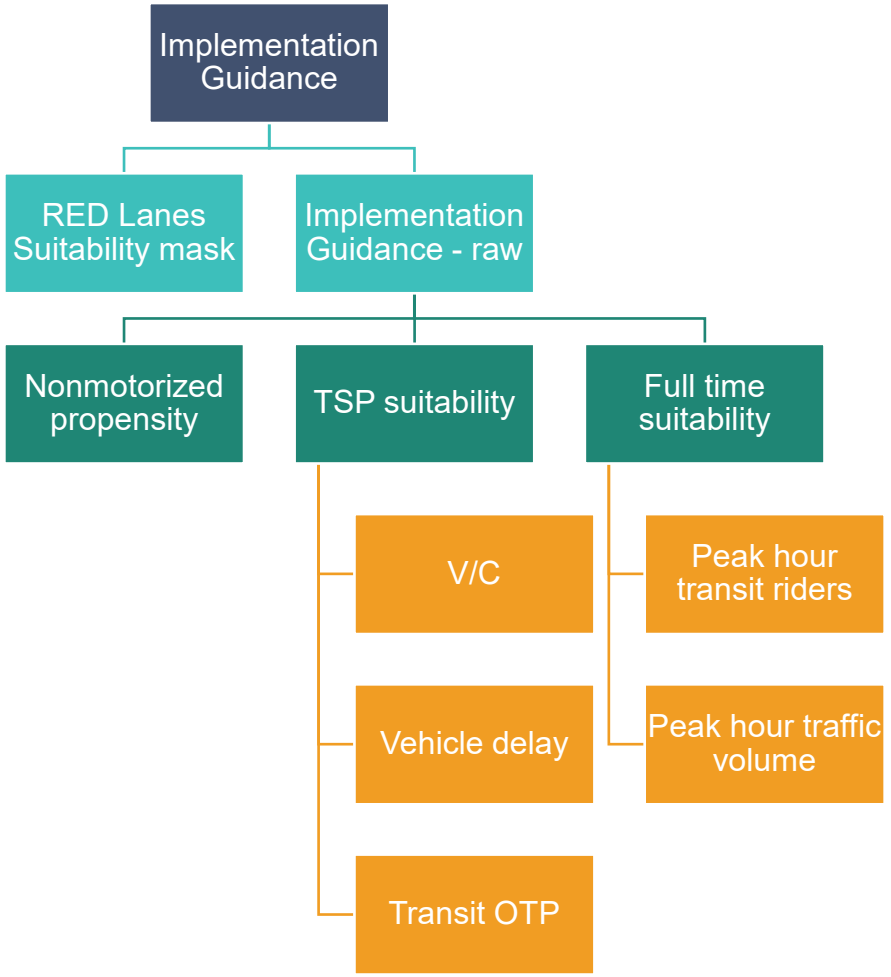



JSON file: {config_dir}\DD_ComboMasked.json

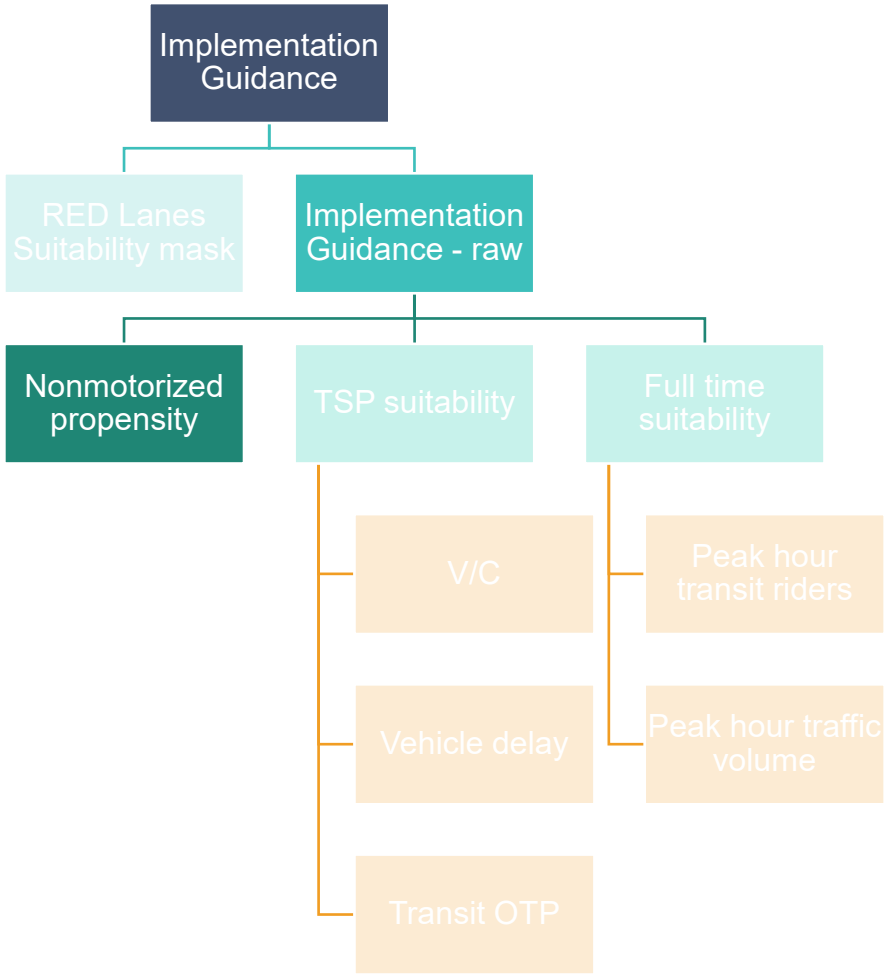
Base surface: {config_dir}\DD_Combo.json

Adjustment surface parameters: >0,base,0
*(if the RED Lanes suitability score is greater than zero keep the DD combo score, else set the cell value to zero)*

# IMPLEMENTATION GUIDANCE



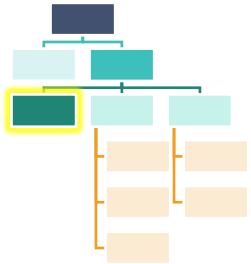
# IMPLEMENTATION GUIDANCE – NONMOTORIZED PROPENSITY

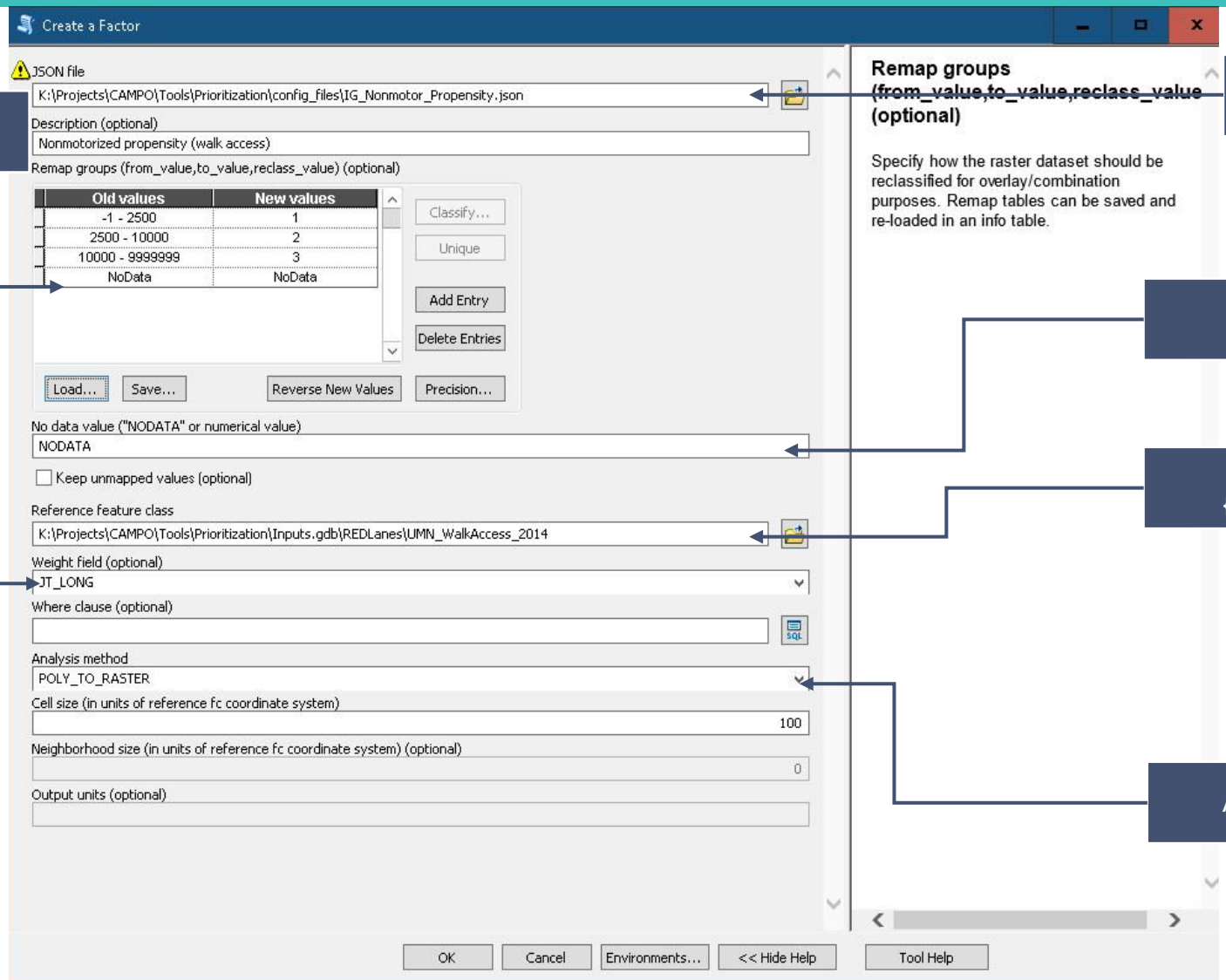


# IMPLEMENTATION GUIDANCE – NONMOTORIZED PROPENSITY

Remap groups: load from {remaps_dir}/IG_Nonmotor_Prop

Weight field:JT_LONG





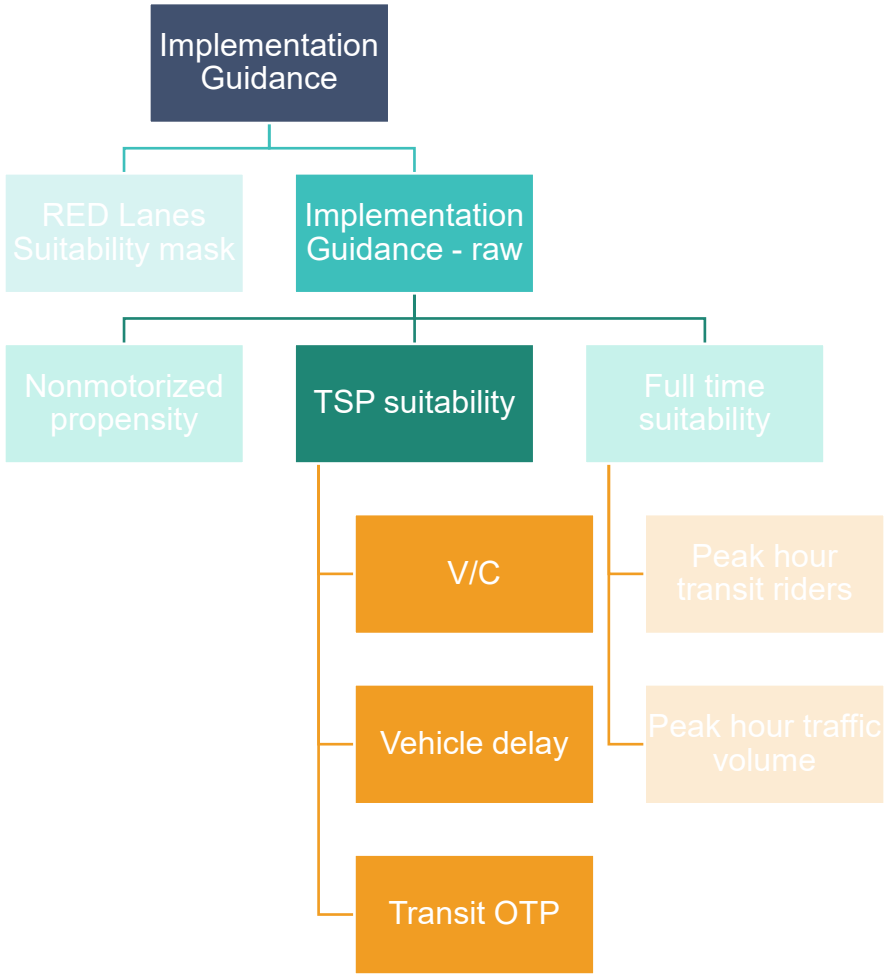

JSON file: {config_dir}\IG_Nonmotor_Propensity.json

No data value: NODATA

Reference feature class: {inputs_gdb}\UMN_WalkAccess_2014

Analysis method: POLY_TO_RASTER

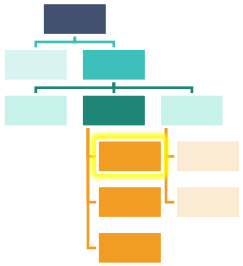
# IMPLEMENTATION GUIDANCE – TSP SUITABILITY

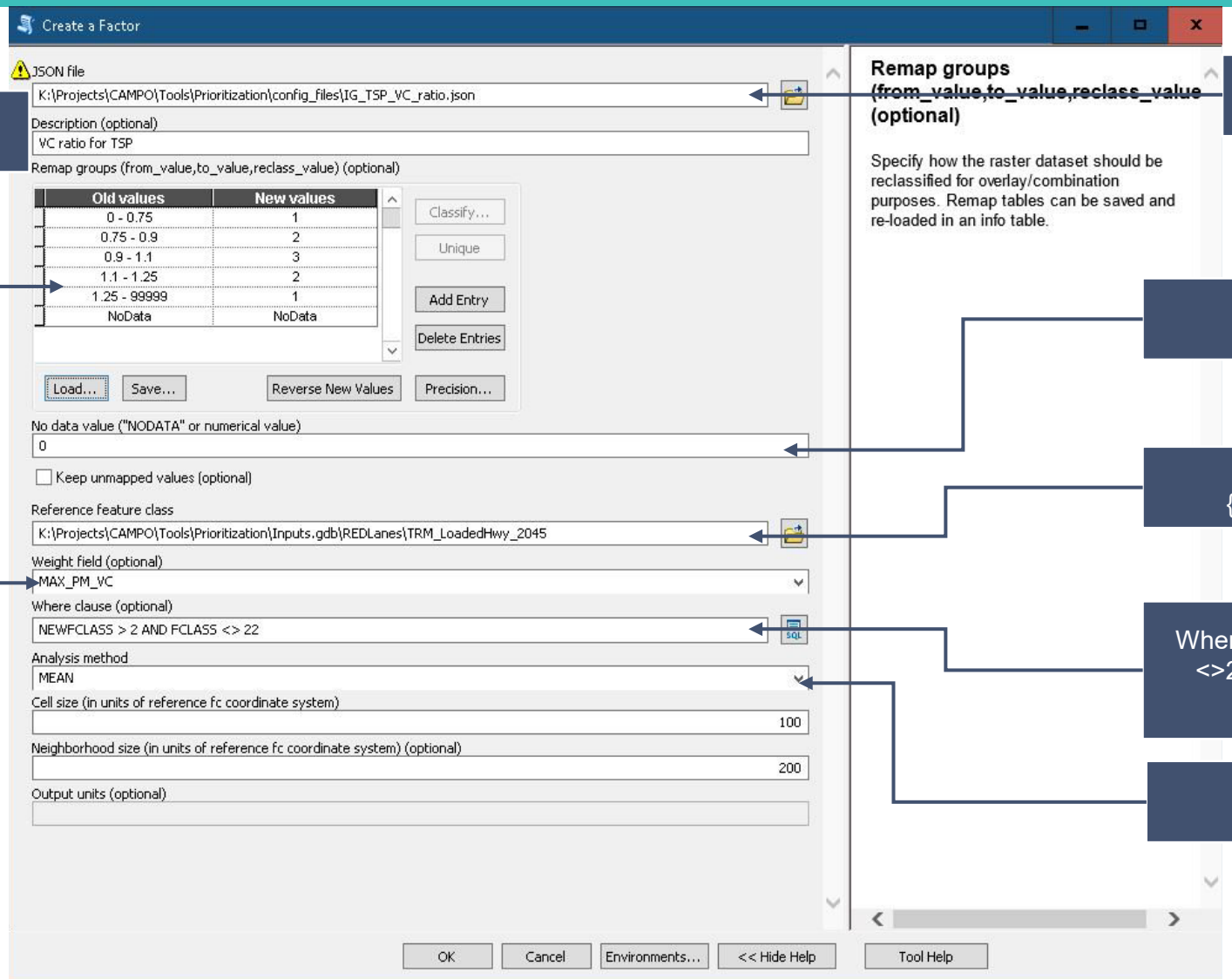


# IMPLEMENTATION GUIDANCE – TSP SUITABILITY – V/C RATIO

Remap groups: load from {remaps_dir}/IG_TSP_VC_ratio

Weight field: MAX_PM_VC






JSON file: {config_dir}\IG_TSP_VC_ratio.json

No data value: 0

Reference feature class: {inputs_gdb}\TRM_LoadedHwy_2045

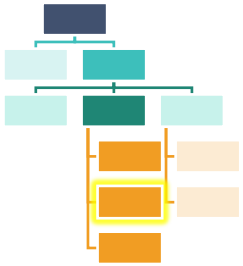
Where clause: NEWFCLASS>2 AND FCLASS <>22 (only include non-limited access/non-tollway facilities)

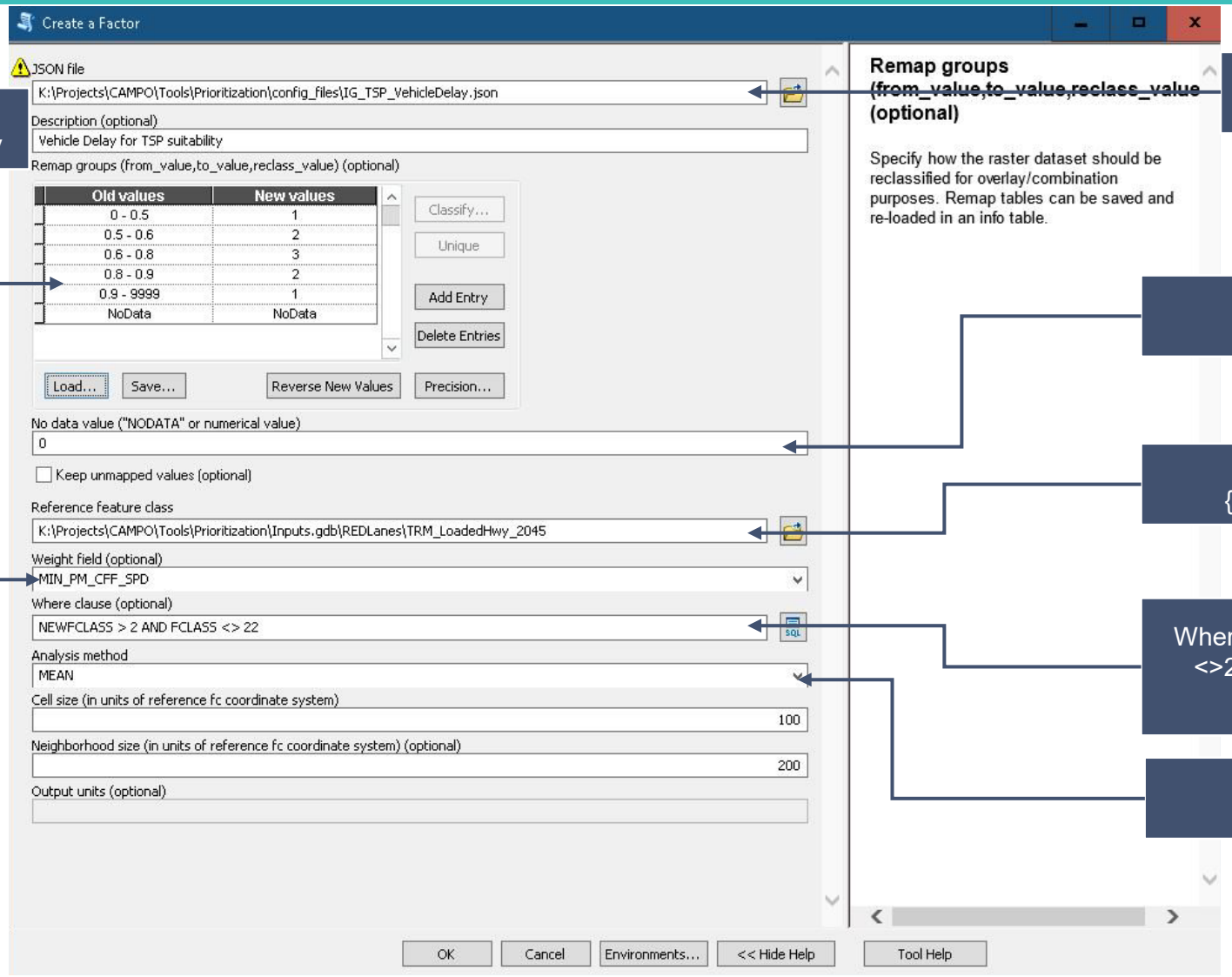
Analysis method: MEAN

# IMPLEMENTATION GUIDANCE – TSP SUITABILITY – VEHICLE DELAY

Remap groups: load from {remaps_dir}/IG_TSP_Vehicle_Delay

Weight field: MIN_PM_CFF_SPD






JSON file: {config_dir}\IG_TSP_VehicleDelay.json

No data value: 0

Reference feature class: {inputs_gdb}\TRM_LoadedHwy_2045

Where clause: NEWFCLASS>2 AND FCLASS <>22 (only include non-limited access/non-tollway facilities)

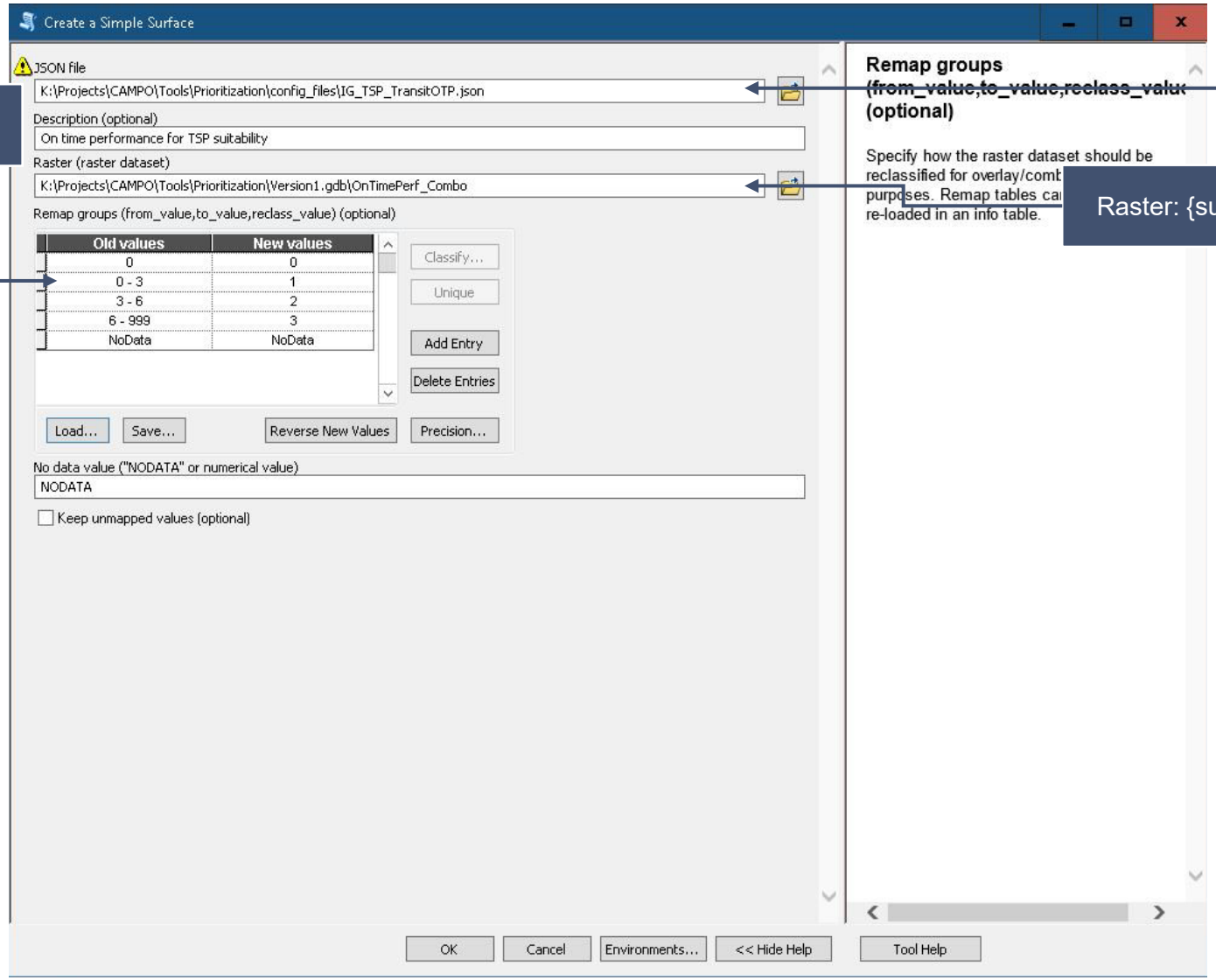
Analysis method: MEAN

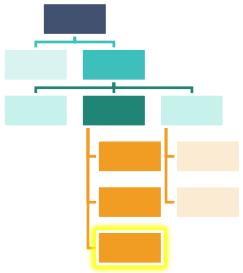
# IMPLEMENTATION GUIDANCE – TSP SUITABILITY – TRANSIT OTP

Remap groups: load from {remaps_dir}/IG_TSP_Transit_OTP

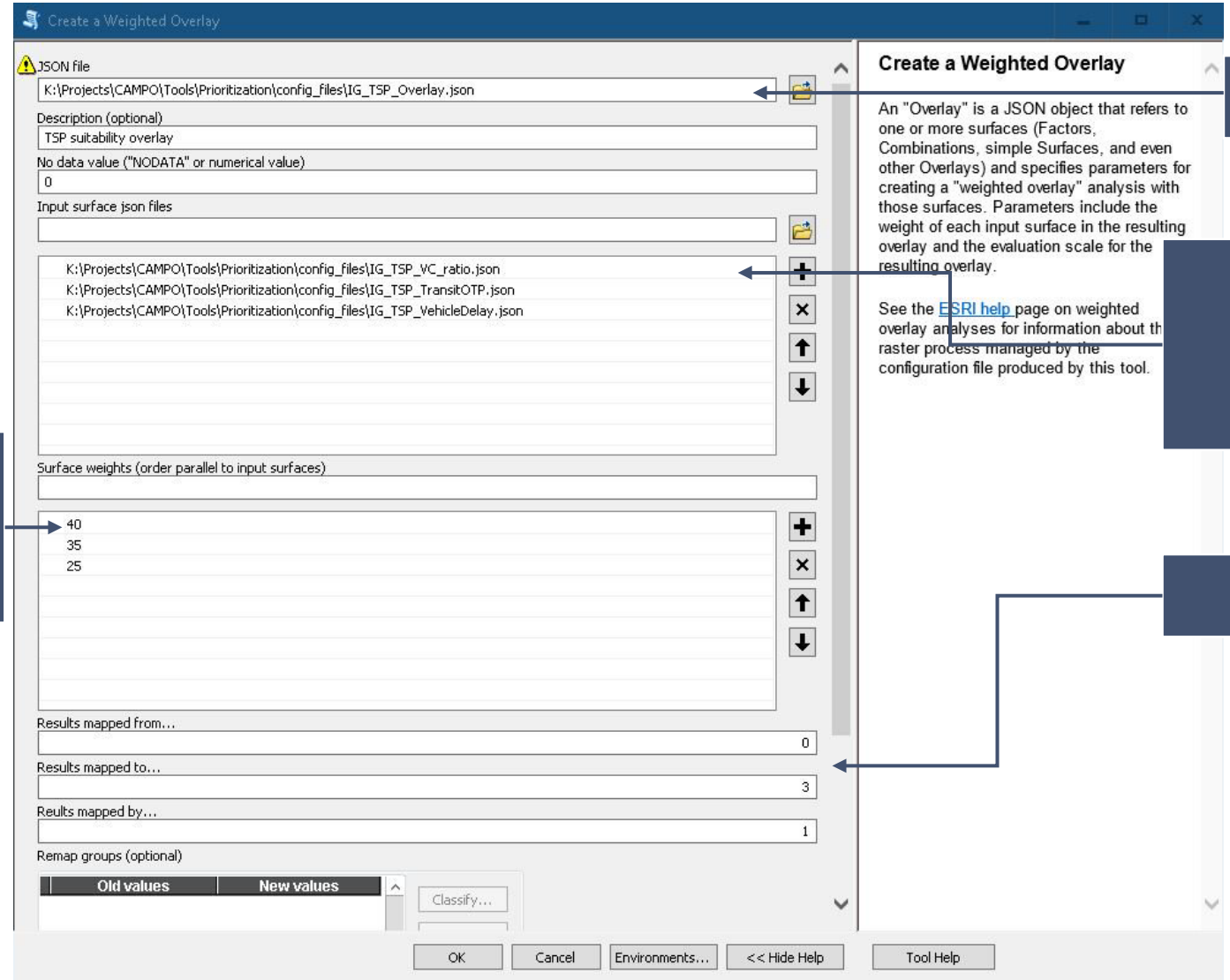
JSON file: {config_dir}\IG_TSP_TransitOTP.json

Raster: {suitability_outputs_gdb}\TransitOps_Overlay



# IMPLEMENTATION GUIDANCE – TSP SUITABILITY – OVERLAY

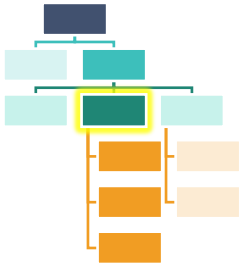



JSON file:
{config_dir}\IG_TSP_Overlay.json

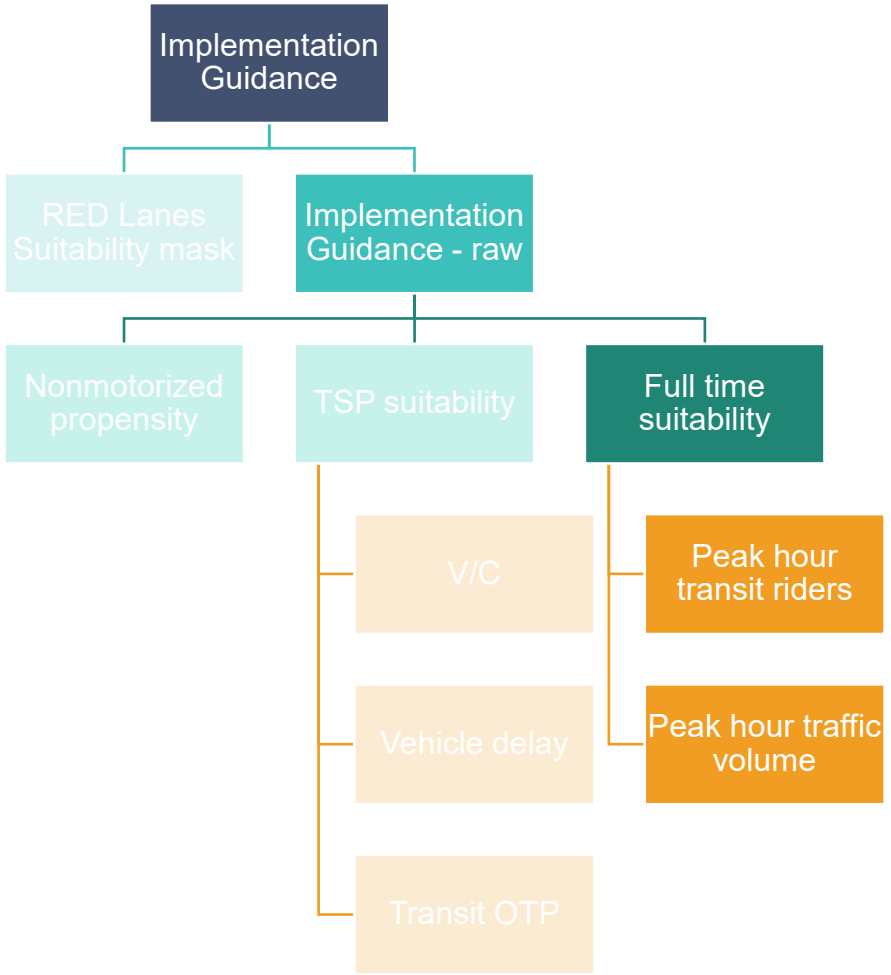
Input surface json files:
{config_dir}\IG_TSP_VC_Ratio.json
{config_dir}\IG_TSP_TransitOTP.json
{config_dir}\IG_TSP_VehicleDelay.json

Weights:
VC Ratio: 40
Transit OTP: 35
Vehicle Delay: 25

Results mapped from/to/by:
Defaults (0/3/1)



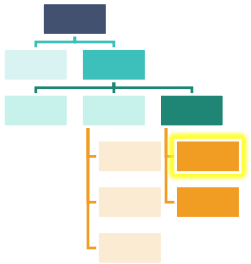
# IMPLEMENTATION GUIDANCE – FULL TIME SUITABILITY

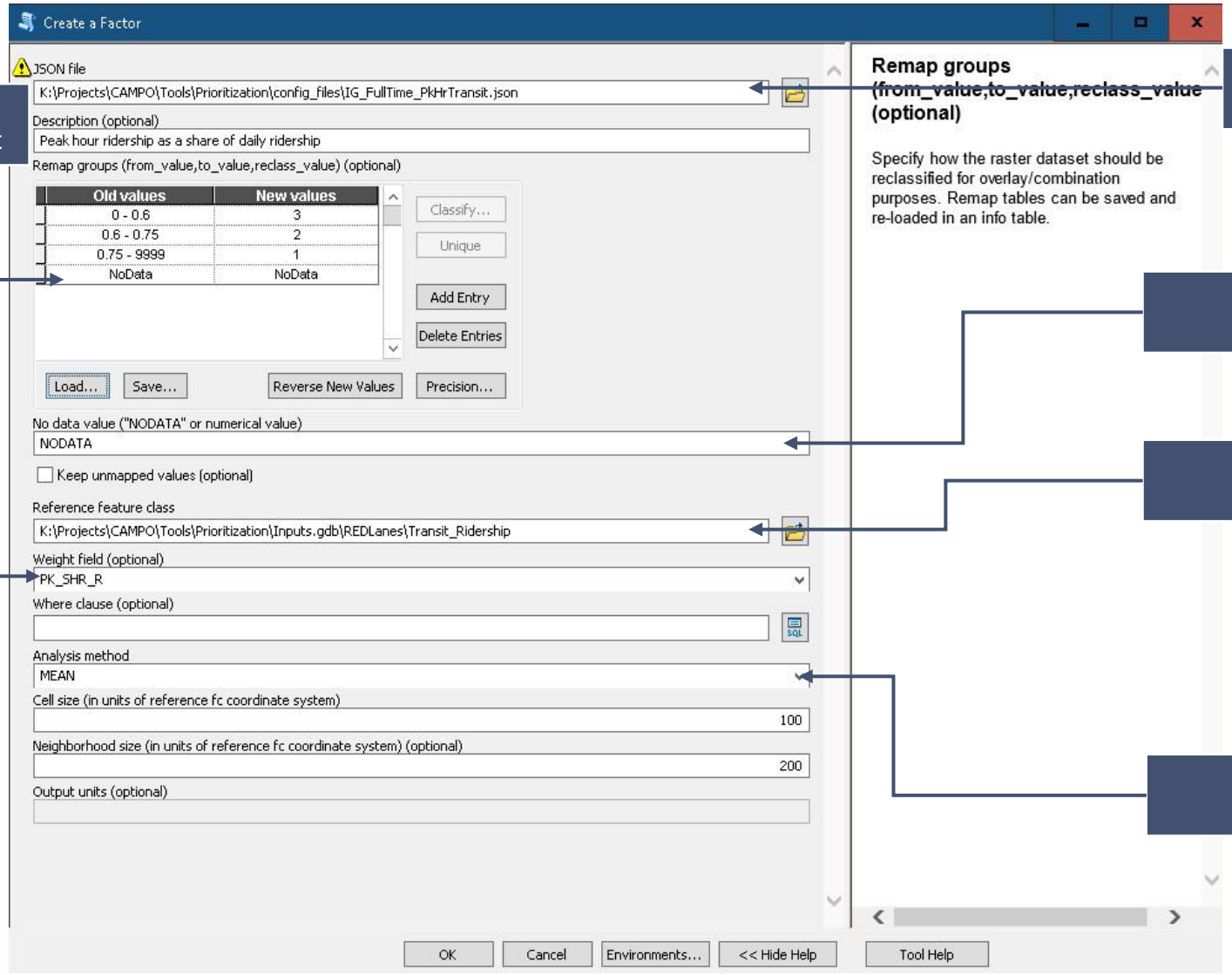


# IMPLEMENTATION GUIDANCE – FULL TIME SUITABILITY – PEAK RIDERSHIP

Remap groups: load from {remaps_dir}/IG_FullTime_PHTransit

Weight field: PK_SHR_R






JSON file: {config_dir}\IG_FullTime_PkHrTransit.json

No data value: NODATA

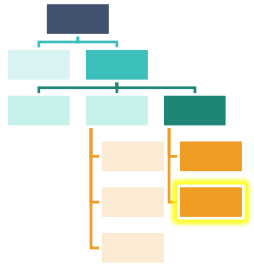
Reference feature class: {inputs_gdb}\Transit_Ridership

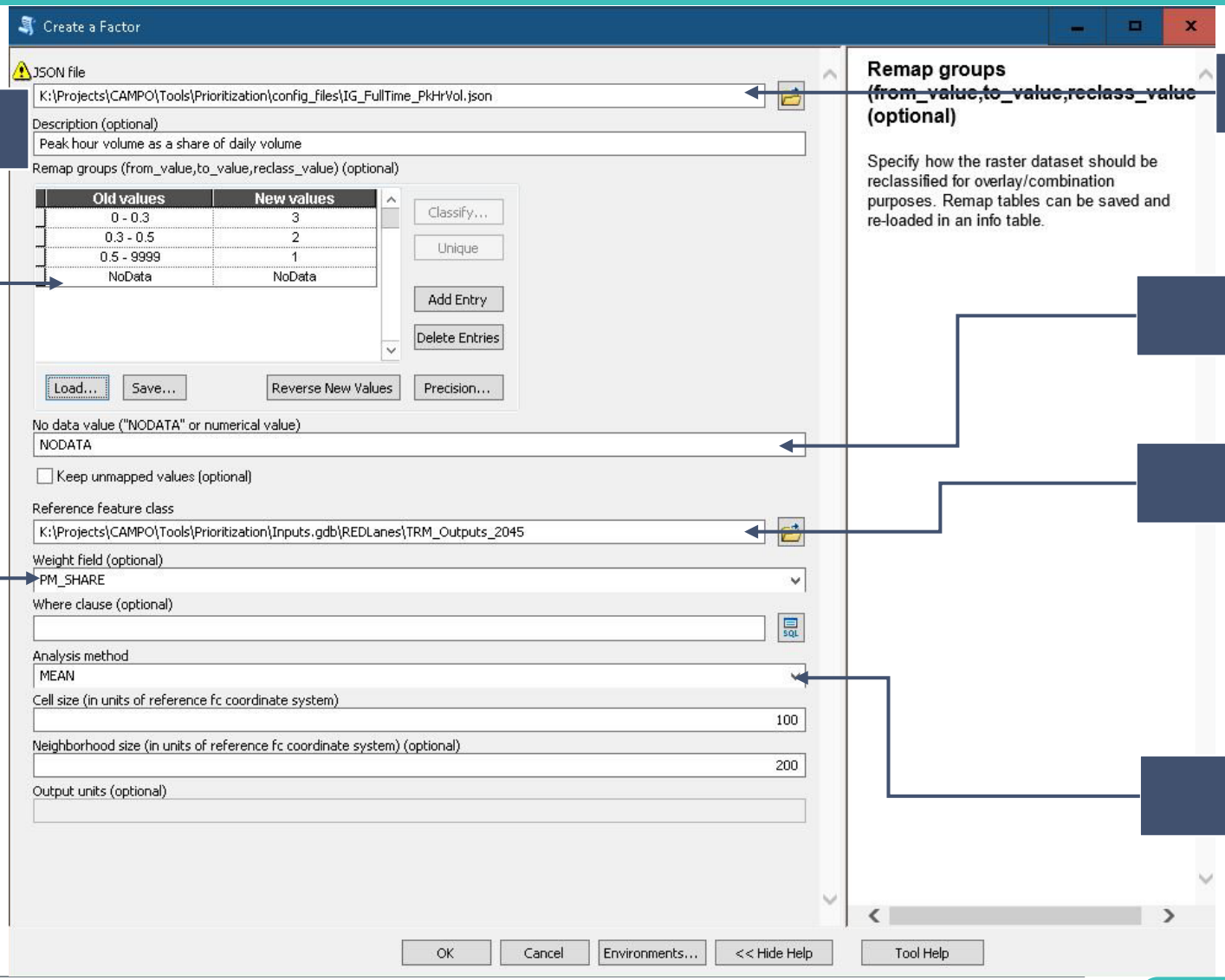
Analysis method: MEAN

# IMPLEMENTATION GUIDANCE – FULL TIME SUITABILITY – PEAK VOLUME

Remap groups: load from {remaps_dir}/IG_FullTime_PHVol

Weight field: PM_SHARE





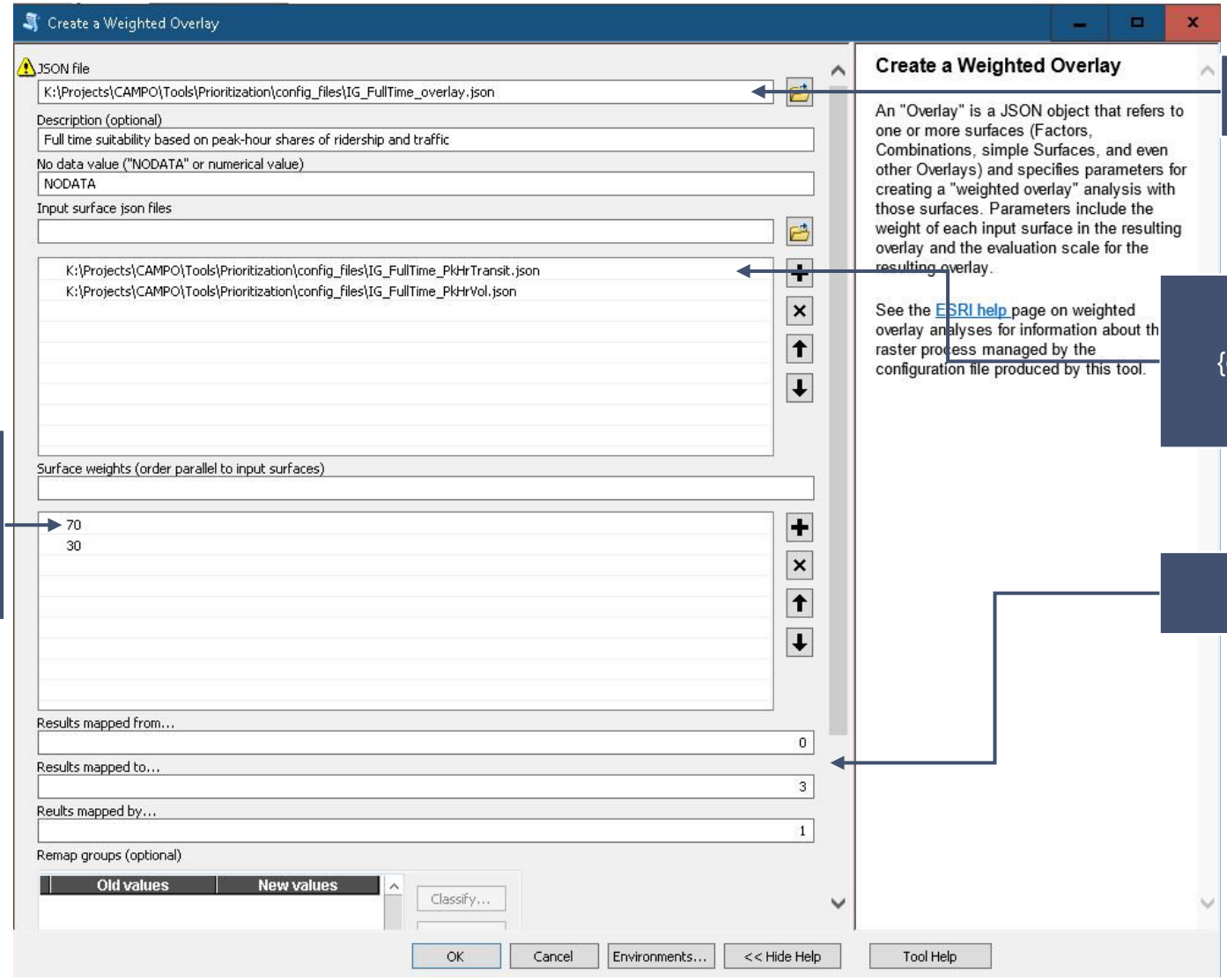

JSON file: {config_dir}\IG_FullTime_PkHrVol.json

No data value: NODATA

Reference feature class: {inputs_gdb}\TRM_Outputs_2045

Analysis method: MEAN

# IMPLEMENTATION GUIDANCE – FULL TIME SUITABILITY – OVERLAY

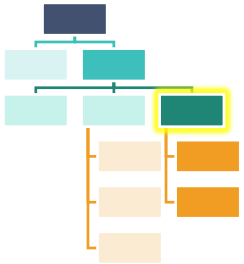



JSON file:
{config_dir}\IG_FullTime_Overlay.json

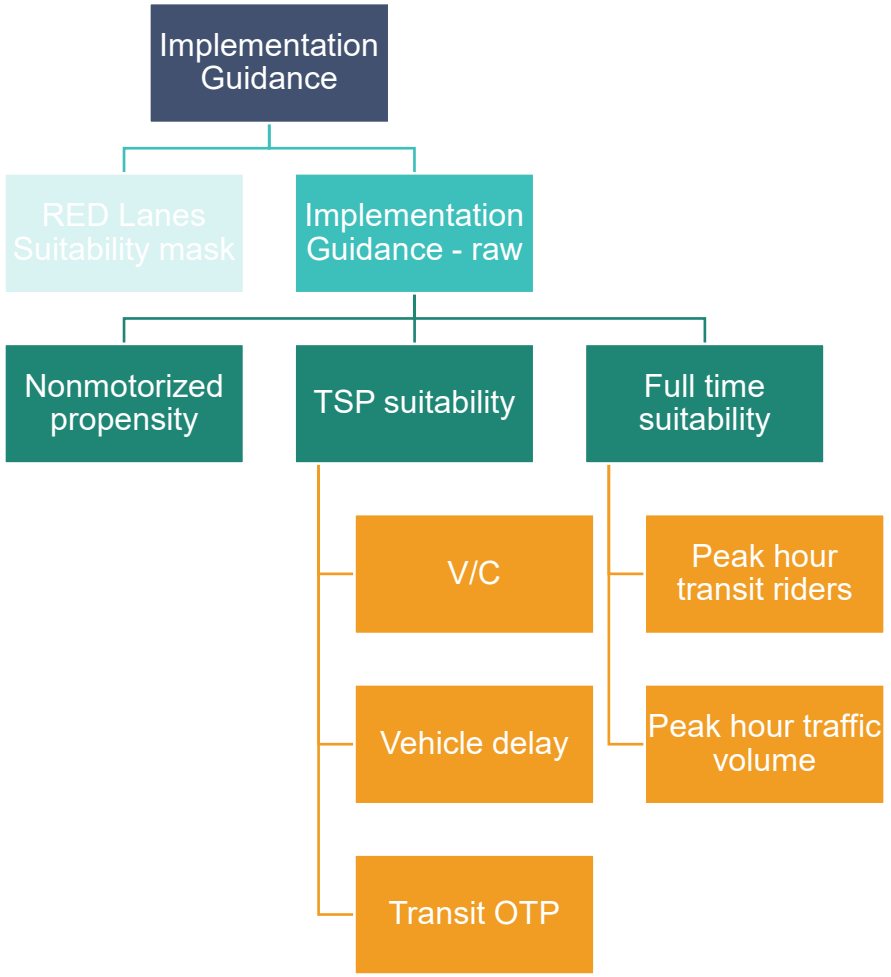
Input surface json files:
{config_dir}\IG_FullTime_PkHrTransit.json
{config_dir}\IG_FullTime_PkHrVol.json

Weights:
Peak Hour Ridership Share: 70
Peak Hour Volume Share: 30

Results mapped from/to/by:
Defaults (0/3/1)



# IMPLEMENTATION GUIDANCE – RAW COMBO



# IMPLEMENTATION GUIDANCE – RAW COMBO

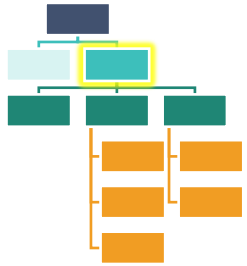
Combo type: Calculation

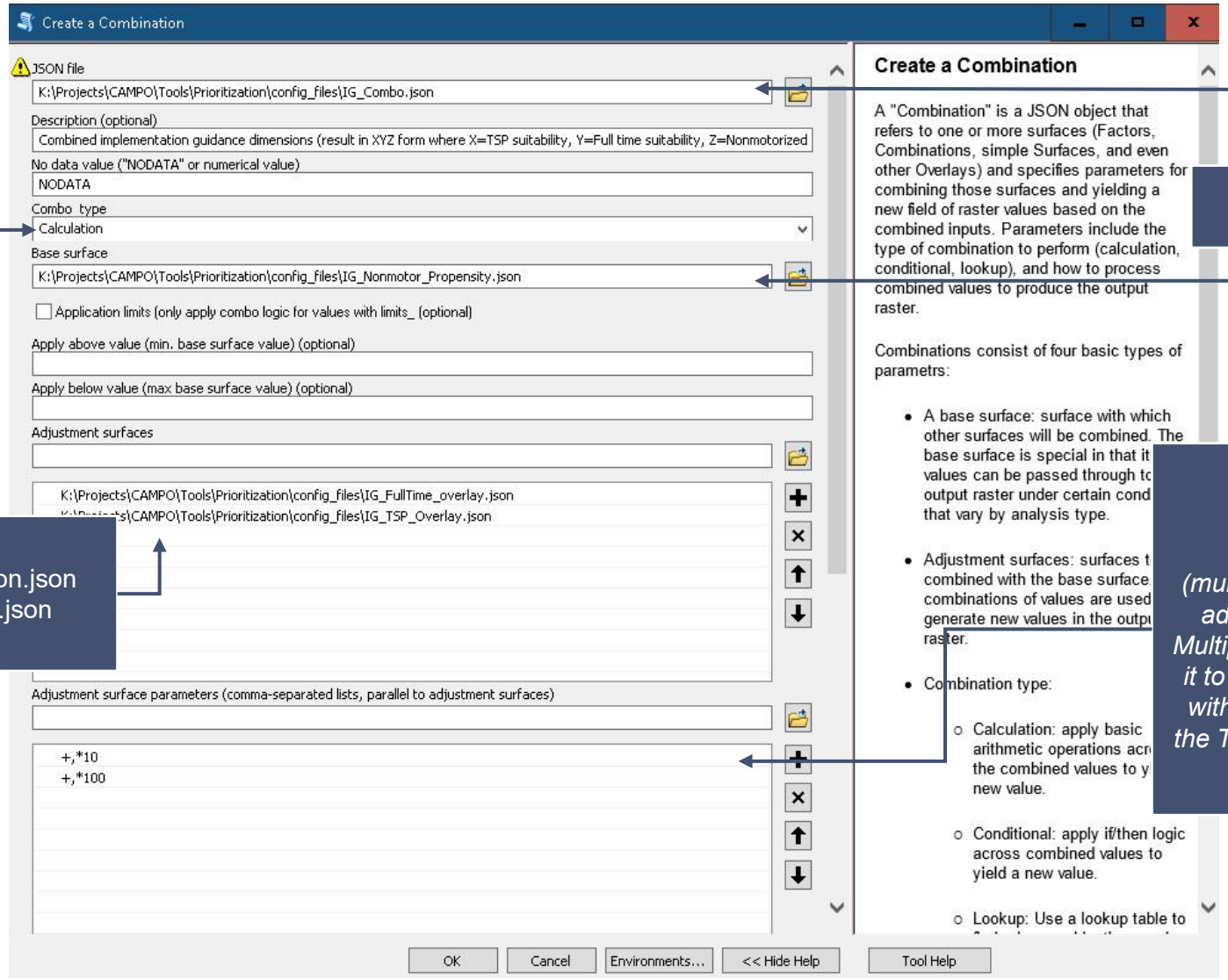
JSON file: {config_dir}\IG_Combo.json

Base surface: {config_dir}\IG_Nonmotor_Propensity.json

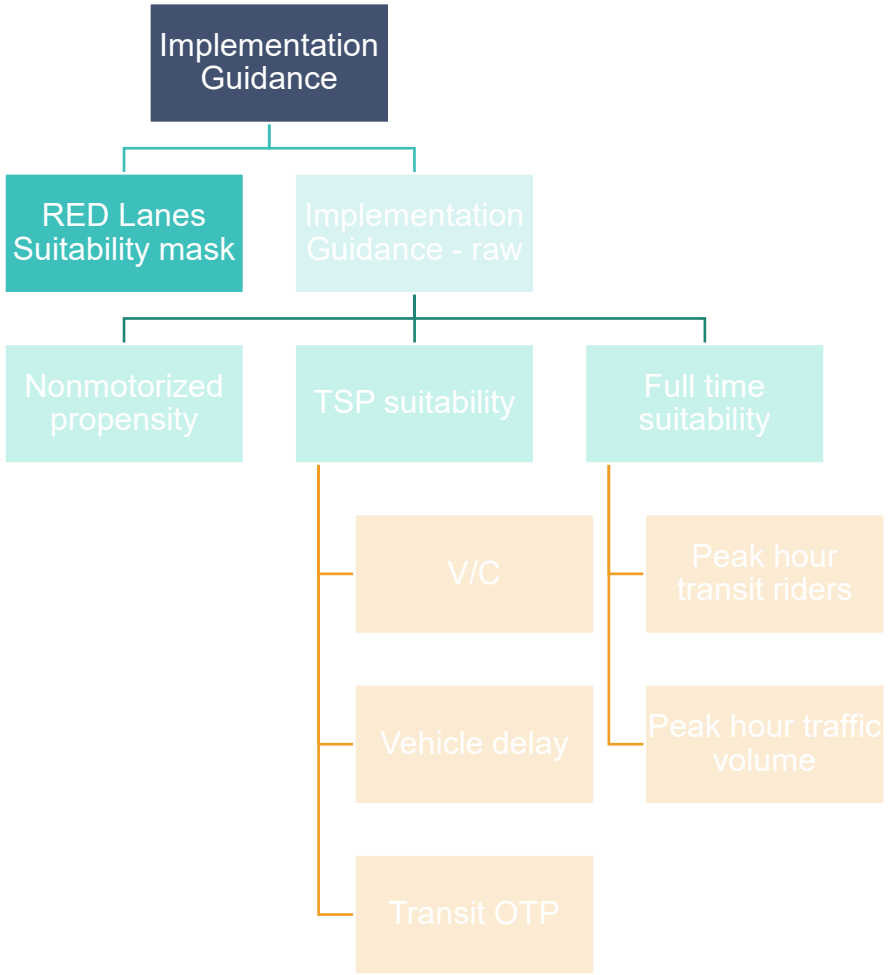
Adjustment surfaces:
{config_dir}\IG_FullTime_Overlay.json.json
{config_dir}\IG_TSP_Overlay.json.json

Adjustment surface parameters:
+,*10
+,*100
*(multiply the full-time suitability value by 10 and add it to the nonmotorized propensity value; Multiple the TSP suitability value by 100 and add it to the previous value – this produces a raster with three-digit output values, XYZ, where X is the TSP score and Y is the full-time score, and Z is the nonmotorized propensity score)*

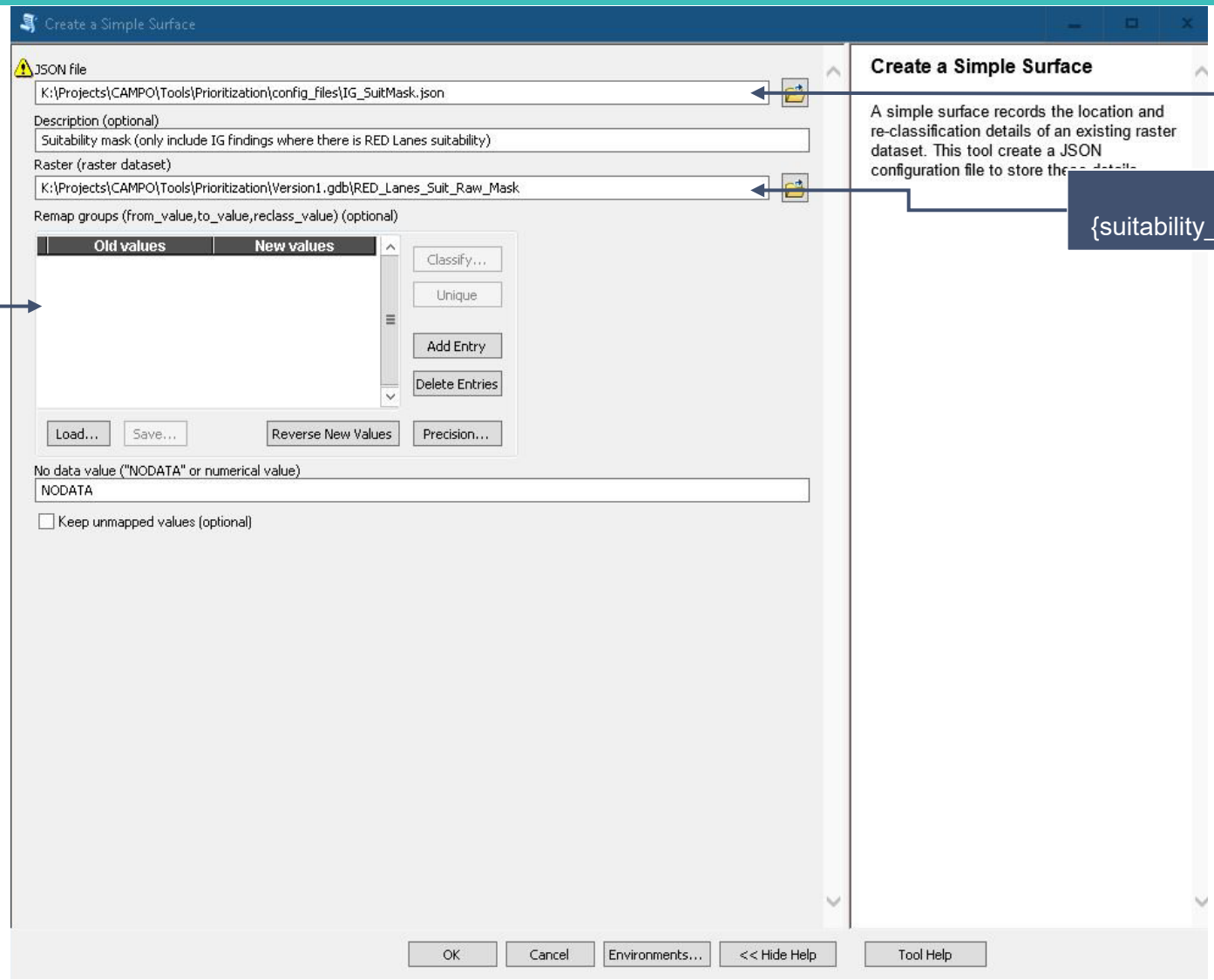




# IMPLEMENTATION GUIDANCE – MASKING



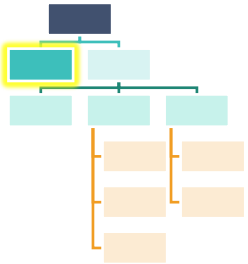
# IMPLEMENTATION GUIDANCE – SUITABILITY MASK




JSON file: {config_dir}\IG_SuitMask.json

Raster: {suitability_outputs_gdb}\RED_Lanes_Suit_Raw_Mask

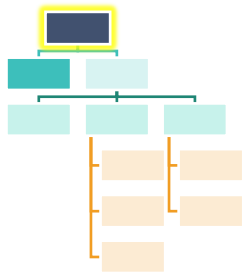
Remap groups: (none)

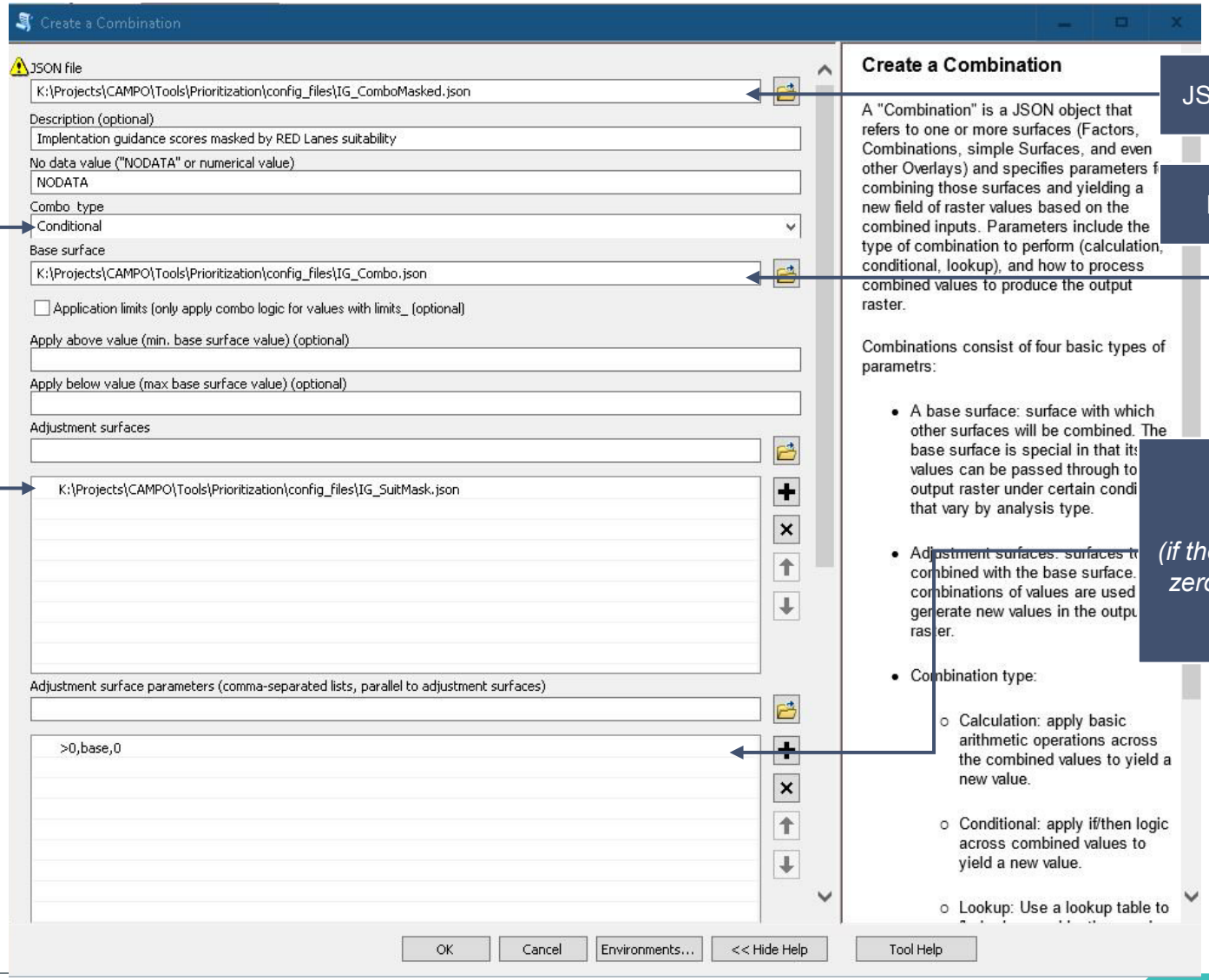


# IMPLEMENTATION GUIDANCE – ADDING THE SUITABILITY MASK

Combo type: Conditional

Adjustment surfaces: {config_dir}\IG_SuitMask.json





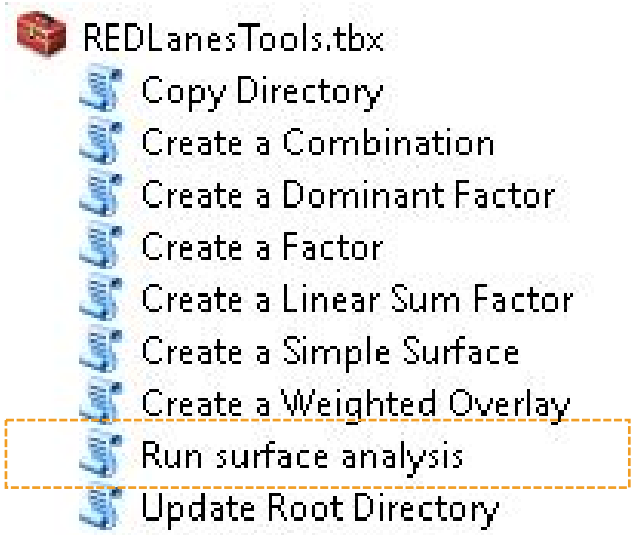

JSON file: {config_dir}\IG_ComboMasked.json

Base surface: {config_dir}\IG_Combo.json

Adjustment surface parameters:
>0,base,0
*(if the RED Lanes suitability score is greater than zero keep the IG combo score, else set the cell value to zero)*

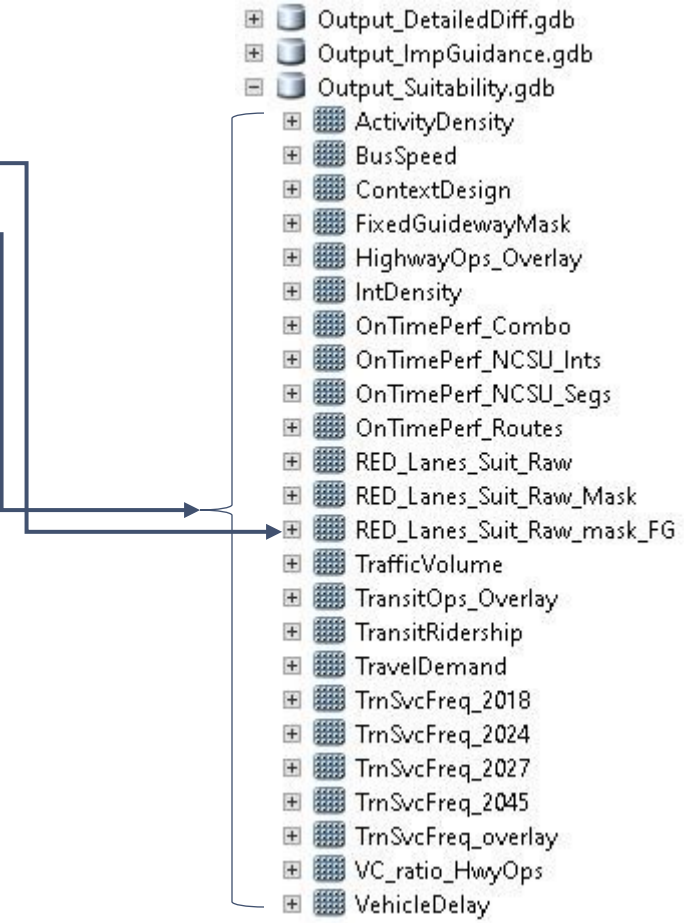
# USING THE “RUN SURFACE ANLAYSIS” TOOL

- Use the “**Run Surface Analysis**” tool to create the resulting raster for a specified surface configuration (.json) file as well as all prerequisite files. Warning! All existing files in the output geodatabase are deleted when this tool is run.
- **Three** runs of the “Run Surface Analysis” tool are made for the RED Lanes evaluation process:
    1. “RED_Lanes_Suit_Raw_Mask_FG.json” - Calculates raw suitability and applies the transit service and fixed-guideway masks.
    2. “DD_ComboMasked.json” – Calculates and combines detailed differentiator variables and applies the transit service mask.
    3. “IG_ComboMasked.json” – Calculates and combines implementation guidance variables and applies the transit service mask.



# USING THE “RUN SURFACE ANLAYSIS” TOOL

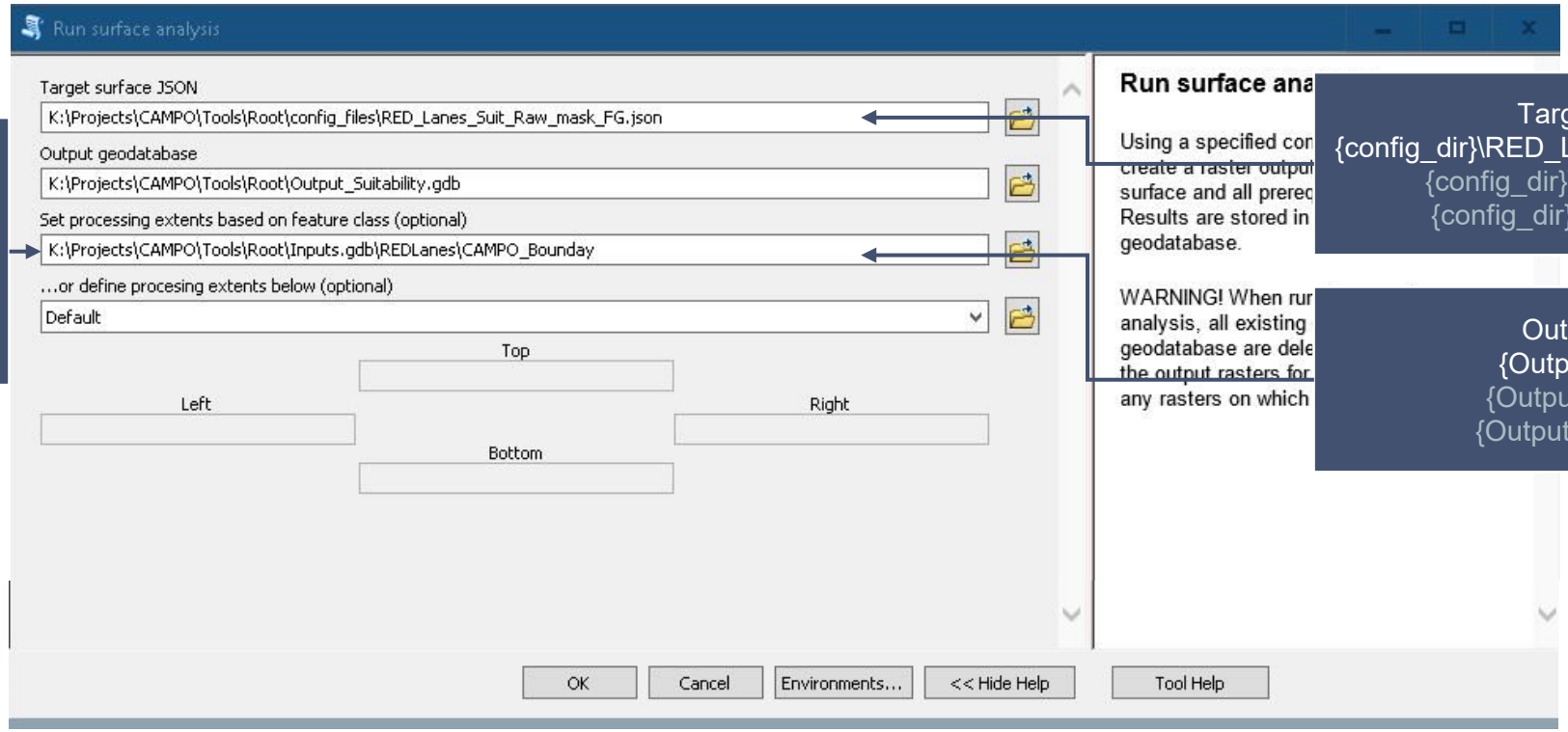
- Select a surface json file as the “final” output.  This surface’s resulting raster will be produced in the output workspace.
- All pre-requisite surfaces will be analyzed and resulting rasters produced in the output workspace.
- Spatial analyst extension must be installed and licensed for this tool to run successfully.
- Optional: set processing extents to define a consistent frame of reference for all surfaces to be produced
  - This is recommended as different input datasets have different default processing extents.
  - The CAMPO boundary polygon in the Inputs geodatabase is provided for precisely this application.



# USING THE “RUN SURFACE ANLAYSIS” TOOL

Set processing extents:
{input_gdb}\CAMPO_Boundary

*(Extents can all be set manually in the field below, but this is more complex than simply pointing to the boundary file.)*




Target surface JSON:
{config_dir}\RED_Lanes_Suit_Raw_mask_FG.json
{config_dir}\DD_ComboMasked.json
{config_dir}\IG_ComboMasked.json

Output geodatabase:
{Outputs_Suitability.gdb}
{Outputs_DetailedDiff.gdb}
{Outputs_ImpGuidance.gdb}

Warning! All existing files in the output geodatabase are deleted when this tool is run.

# AUTOMATED SEGMENTATION USING R

- The RED Lanes evaluation process outputs are in raster format (100-foot grid cells).
- To generate segment-level scores from the raster datasets, a spatial analytics script has been developed in R.
  - R provides spatial analysis capabilities and conveniences that ArcGIS either does not offer or requires additional licenses beyond Spatial Analyst.
  - The script is simple to run in R Studio.

# AUTOMATED SEGMENTATION USING R

- **To download R for the first time**, visit http://archive.linux.duke.edu/cran/.
  - For Windows, select "Download R for Windows", then "install R for the first time", then "Download R for Windows". Once the installer is downloaded, open it and complete the setup wizard, keeping all defaults.
  - For Mac, select "Download R for (Mac) OS X", then the download link for the .pkg file for the latest release of R. Once the installer is downloaded, open it and complete the setup wizard, keeping all defaults.

- **Once R is installed,** visit https://rstudio.com/products/rstudio/download/ to download RStudio.
  - Select "Download" beneath RStudio Desktop, then under "All Installers" on the next page, select the download link for your OS. Once the installer is downloaded, open it and complete the setup wizard, keeping all defaults.

# AUTOMATED SEGMENTATION USING R

- Once R and RStudio are installed, open each of the provided files in RStudio. They are numbered according to the order in which they should be completed (00 through 06)
- Go to the "00_Dependencies" script. At the top right of Script window (top left panel), click the "Source" button. Your first time running these scripts on a new machine, you will be prompted to allow package installs – follow the prompts on the screen to complete any necessary installs.
- Once "Complete" is printed in the RStudio console (bottom left panel), continue to the "01_Intersect" script. Click "Source"; this time, you will be prompted for a few function inputs – enter them according to the on-screen instructions.
- Once "Complete" is printed in the console, move onto "02_Clip". Again, Click "Source", and again follow the prompts. Once "Complete" is printed, continue to the next script.
- Continue the above pattern of Sourcing the script, providing inputs, waiting for "Complete", and moving to the next script until the final script "06_Enrich" is completed. At this point, segmentation is finished, and the final output will be at the write directory you specified as an input to "06_Enrich".

# AUTOMATED SEGMENTATION USING R

- Notes on processing the scripts:
  - The NCDOT street routes should be saved as ".shp". The suitability, detailed differentiators, and implementation guidance rasters should be saved as ".tif"
  - When a read or write directory is requested as a function input, we recommend using "copy as path" functionality (shift-right click on the folder, then select "copy as path") for inputting the directory path. The scripts are designed to work best with paths input using this method.
  - We **highly recommend** writing all outputs to the same directory. We also recommend placing the NCDOT street routes shapefile, suitability raster, and detailed differentiators/implementation guidance rasters in this directory for the same reason. **If you do this, you can enter the same path every time a directory is requested as an input, read or write!**
  - The read/write directories **cannot** be geodatabases. R does not support writing to geodatabases; if you'd like your outputs in a geodatabase, please do this manually upon completion of the entire process.
  - Outputs of each script will automatically be saved with a file name matching that of the script it produces for ease of process

# MANUAL QA/QC

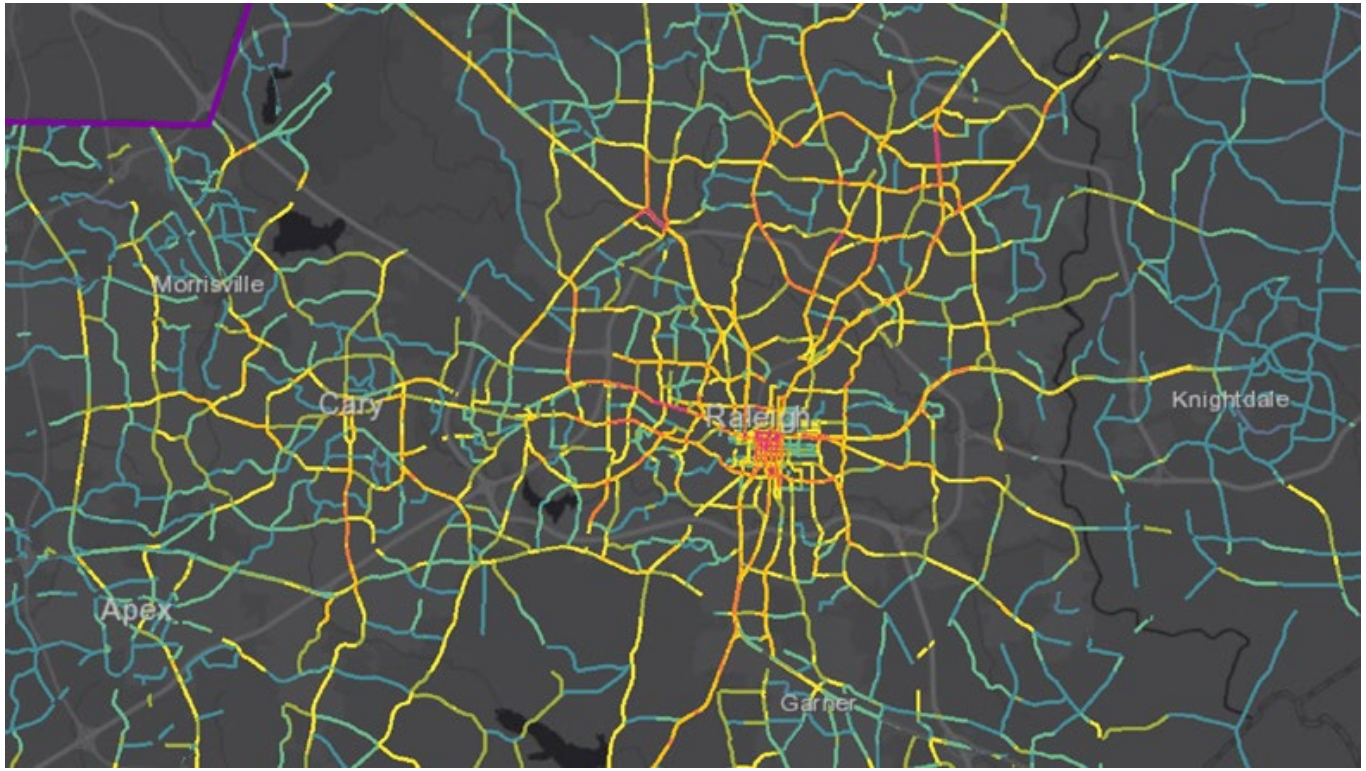
- Compare segment outputs to raw suitability rasters
- Manually code and overwrite features/attributes for missing segments.

# 4 SPATIAL ANALYST CONCEPTS

# SPATIAL ANALYST - OVERVIEW

From ArcGIS.com…

*"The ArcGIS Spatial Analyst extension provides a rich set of spatial analysis and modeling tools for both raster (cell-based) and feature (vector) data."*



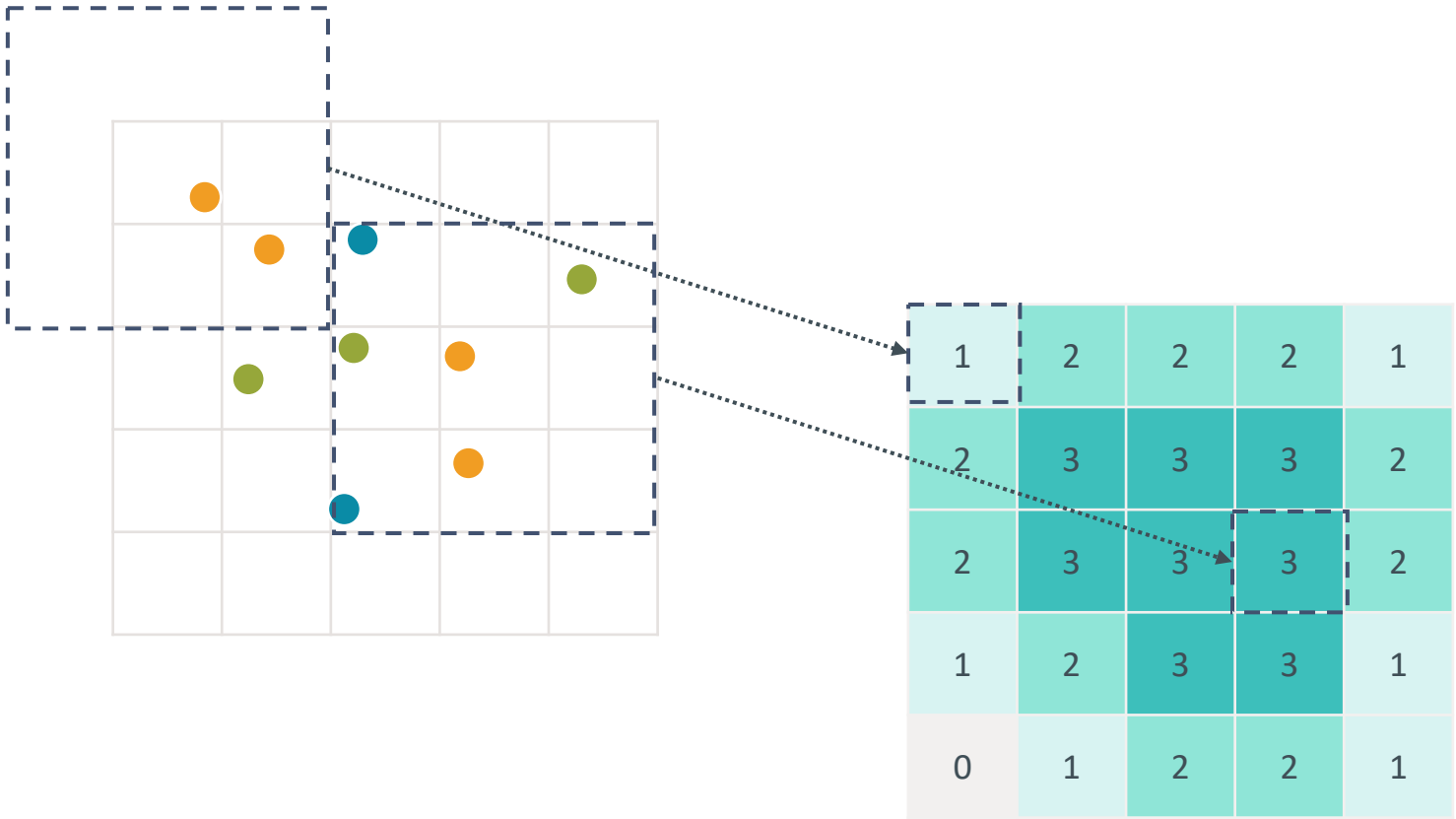
# SPATIAL ANALYST - OVERVIEW

Tools for analyzing spatial patterns based on raster and vector data

- Many useful capabilities for operationalizing the concepts and measures identified for RED Lanes suitability
  - Create raster datasets from vector data (points, lines, polygons)
  - Process raster datasets
    - Weighted overlay analysis
    - Combine and calculate values in the same place or in the vicinity

# USEFUL CAPABILITIES – CREATE RASTERS FROM VECTOR DATA

Translate vector data (points, lines, polygons) into rasters




Example: how many distinct colors are there within the “floating zone”?

- Vector data = features in a feature class or shape file *(dots in illustration)*
- Raster = network of equally-sized cells *(grids in illustration)*
- Floating zone or “neighborhood” in arcpy terminology = area of specified size and shape *(dashed outlines in illustration)*
  - Circle
  - Square
  - Annulus (doughnut)
  - Wedge

# USEFUL CAPABILITIES – OVERLAY RASTERS

Get the weighted average of cells representing the same location.

- Raster 1 and raster 2 define the same area using cells of the same size
- Raster 1 is assigned a weight of 30%; raster 2 is assigned a weight of 70%
- The weighted overlay yields raster 3
- Consider the outlined cell in each raster as an example:
  - (2 * 0.3) + (1 * 0.7) = (0.6 + 0.7) = 1.3
  - The overlay analysis will return the nearest integer, so the value in raster 3 is 1

Raster 1

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 2 | 4 | 4 |
| 1 | 2 | 3 | 3 | 4 |
| 2 | 2 | 3 | 3 | 4 |
| 1 | 2 | 3 | 3 | 4 |
| 1 | 1 | 2 | 3 | 3 |

Raster 2

| | | | | |
|---|---|---|---|---|
| 2 | 3 | 4 | 5 | 5 |
| 2 | 2 | 3 | 4 | 5 |
| 1 | 1 | 3 | 4 | 4 |
| 1 | 1 | 2 | 3 | 4 |
| 1 | 1 | 2 | 3 | 3 |

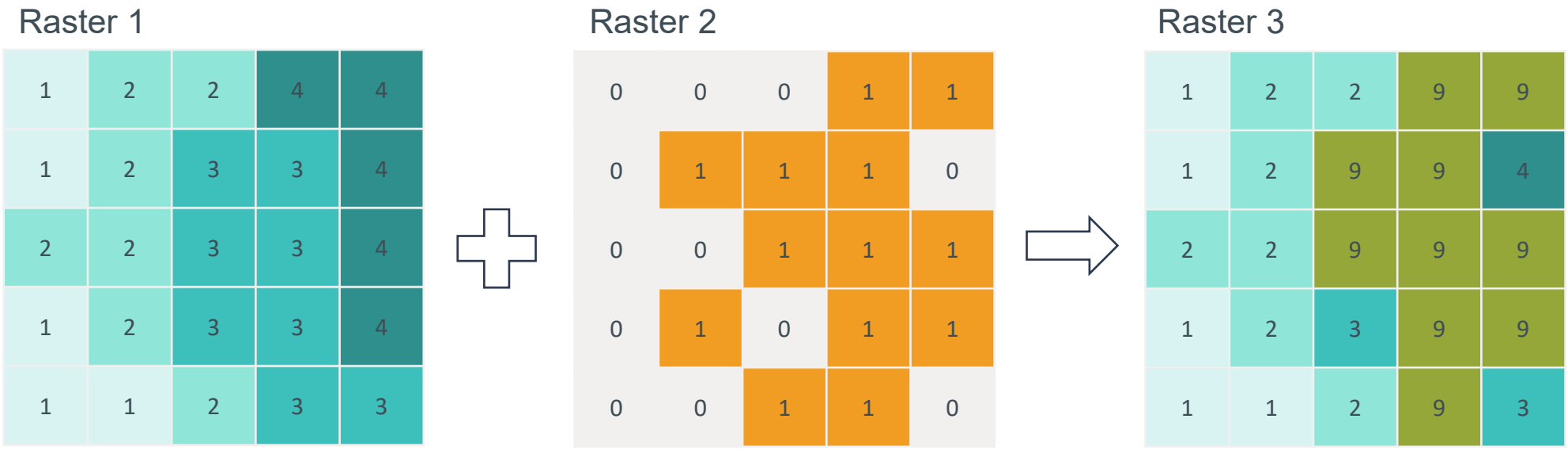
Raster 3

| | | | | |
|---|---|---|---|---|
| 2 | 3 | 3 | 5 | 5 |
| 2 | 2 | 3 | 4 | 5 |
| 1 | 1 | 3 | 4 | 4 |
| 1 | 1 | 2 | 3 | 4 |
| 1 | 1 | 2 | 3 | 3 |

# USEFUL CAPABILITIES – COMBINE RASTERS

Calculate a new value based on the values in two raster datasets

- Raster 1 and raster 2 define the same area using cells of the same size
- If the value in raster 1 is 3 or greater and the value in raster 2 is 1, calculate a new value of 9
- Otherwise, retain the value from raster 1
- The combination yields raster 3



# ADDITIONAL NOTES ON SPATIAL ANALYST

STRENGTHS

- Faster, easier to implement, and simpler to construct “surfaces” representing all locations within a study area than vector-based or network-based analysis methods.
- Account for areawide typical conditions using consistent cell size and neighborhood size (floating zone) definitions

LIMITATIONS

- Can be unpredictable when working with source data in inconsistent spatial reference systems
  - Best practice: ensure that all input data are projected into the same spatial reference
- Account for areawide conditions without regard for barriers, such as waterways or major highways
- Most geo-processors yield integer rasters (floating point rasters can be created but can be unwieldy in terms of designing a process around these)
  - Potential loss of precision for any given step (review “raster overlay” example illustration above)

# 5 PYTHON TOOLKIT DOCUMENTATION

# HIERARCHICAL OVERLAY AND COMBINATION SCRIPTS

- Object-oriented approach
  - Defines “objects” that define how to develop and process raster data sets for analysis in ArcGIS’s Spatial Analyst extension
  - Objects have “attributes”: information stored in the object
  - Objects have “methods”: functions that facilitate or automate a variety of workflows
- Utilizes ArcGIS’s (v. 10.2.1) arcpy library to automate geo-processing steps
  - Also numpy (v. 1.13.3), which is installed alongside arcpy
  - Also a couple of standard Python libraries
    - json
    - copy
    - Ast
    - OrderedDict (from collections)
  - Scripts developed in Python v. 2.7.12

# SCRIPT OBJECTS

- Surface (the basic building block)
- Sub-classes of Surface
  - Factor
    - DominantFactor
    - LinearSum
  - Overlay
  - Combination
    - ConditionalCombination
    - CalculationCombination
    - LookupCombination
- HierarchyManager

# SURFACE CLASS

# SURFACE CLASS

Primary purpose: store meta-data about a raster dataset and how to re-classify values to facilitate geo-processing

- Attributes:
  - workspace, raster, and path: Where is the raster dataset that is the focus of the Surface object?
  - no_data_value: How to treat “NO DATA” (missing) values in the raster dataset when processing
  - remap_groups, remap: How to reclassify values (arcpy.sa.RemapRange object)
  - Status flags (processed, reprocess):
    - Has this surface already been processed?
    - Does it need to be re-processed (based on user actions)?
- Methods:
  - setRaster, setWorkspace: Set raster location (workspace or file)
  - addRemapGroup, removeRemapGroup, updateRemapGroup: Manage reclassification preferences

# SURFACE CLASS – RECLASSIFICATION PROTOCOLS

| | | | | |
|---|---|---|---|---|
| 31 | 31 | 48 | 46 | 55 |
| 21 | 32 | 33 | 45 | 46 |
| 20 | 21 | 33 | 33 | 45 |
| 10 | 22 | 24 | 32 | 31 |
| ND | 18 | 24 | 33 | 34 |

REMAP:

| Raster value | Reclass value |
|---|---|
| 10-19 | 1 |
| 20-29 | 2 |
| 30-39 | 3 |
| 40-49 | 4 |
| 50-59 | 5 |
| 60-99 | 6 |

| | | | | |
|---|---|---|---|---|
| 3 | 3 | 4 | 4 | 5 |
| 2 | 3 | 3 | 4 | 4 |
| 2 | 2 | 3 | 3 | 4 |
| 1 | 2 | 2 | 3 | 3 |
| 1 | 1 | 2 | 3 | 3 |

NO DATA (ND) = 1

# FACTOR CLASS

# FACTOR CLASS (SUBCLASS OF SURFACE)

Primary purpose: create a Surface object from vector data

- Major attributes:
  - reference_fc: What feature class (vector data) will be used to create the surface?
  - field: What field in the feature class above will be used to create the surface? (optional)
  - analysis_method: What analysis method will be used to create the surface (options depend on feature class type – point, line, or polygon)?
    - Density, kernel density, sum/length, min, max, mean, median, majority, minority, range, standard deviation, variety
  - units: Units to use for raster processing
  - cell_size: Output cell size
  - neighborhood_size: Search radius for the floating zone
  - sr: Spatial reference system
  - where_clause: Where clause (defining criteria for features in the feature class used to create the surface)
- Also inherits attributes from the Surface class

# FACTOR CLASS (SUBCLASS OF SURFACE)

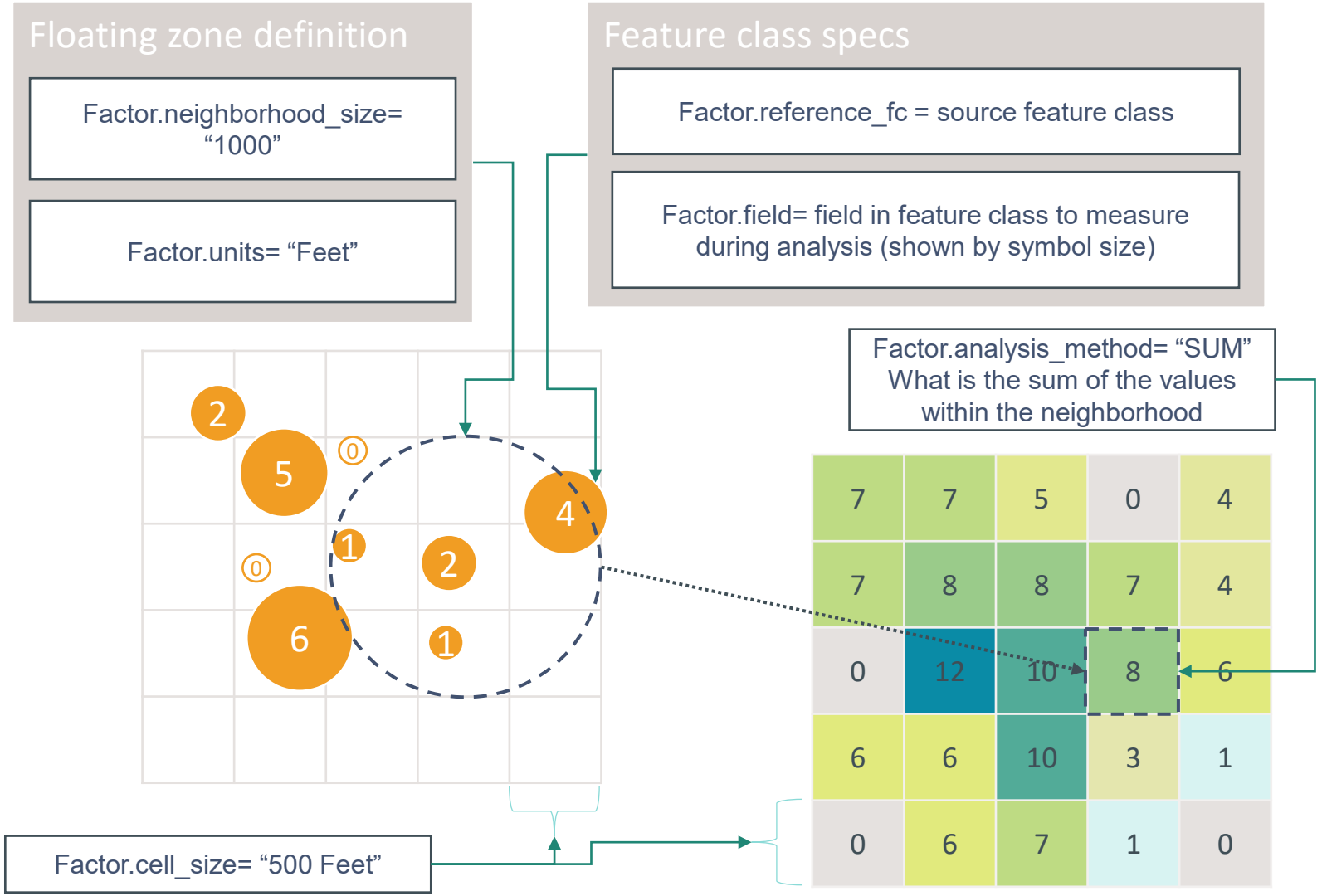
Primary purpose: create a Surface object from vector data

- Major methods:
  - rasterize: based on the attributes, create a raster from the vector data in the referenced feature class
- Also inherits methods of the Surface class

# FACTOR CLASS (SUBCLASS OF SURFACE) – RASTERIZE METHOD

The attributes of the Factor class define how a raster (Surface) will be developed from vector data (reference_fc). Resulting raster values are based on applying the chosen analysis method to evaluate features within the floating zone (neighborhood_size, units), weighted by the chosen field in the reference feature class. The resulting raster will have square cells of the size specified by the cell_size attribute.

The creation of the resulting raster is facilitated by the rasterize method.



# DOMINANT FACTOR CLASS

# DOMINANT FACTOR CLASS (SUBCLASS OF FACTOR)

Primary purpose: create a Surface object by creating a temporary series of Factor objects and choosing the largest (or smallest) among them

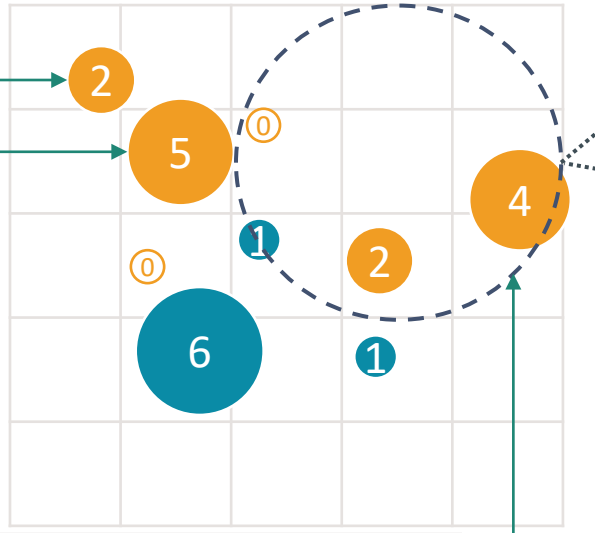
- Major attributes:
  - value_field: Groups features within the reference_fc attribute (inherited from Factor class) into discrete categories
  - weight_field: What field in reference_fc will be used to create each temporary Factor object?
- Major methods:
  - dominantValue: Use the attributes named above to determine which group of features is the most (or least) prevalent
- Also inherits attributes and methods from the Factor and Surface classes
- Example of use: “dominant land use” – what land use category (value_field) is most common within the floating zone area based on the total building area (weight_field) in the floating zone for each distinct use code?

# DOMINANT FACTOR (SUBCLASS OF FACTOR)

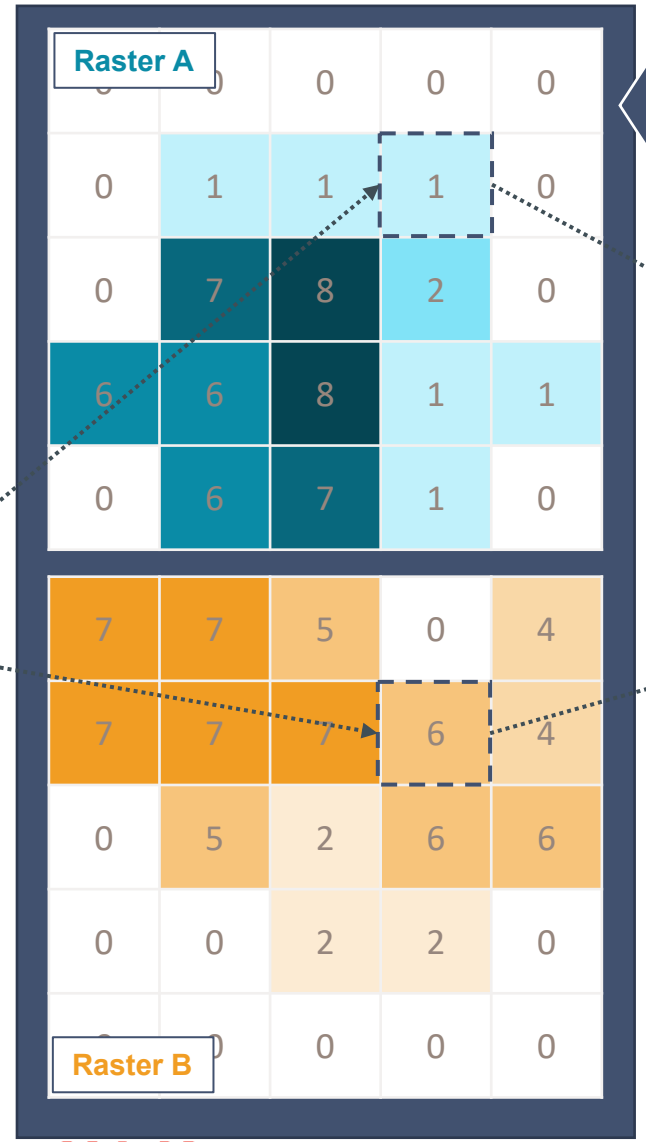
## Feature class specs (reference_fc inherited from Factor class)

DominantFactor.value_field= field in feature class for grouping features during analysis (shown by symbol color)

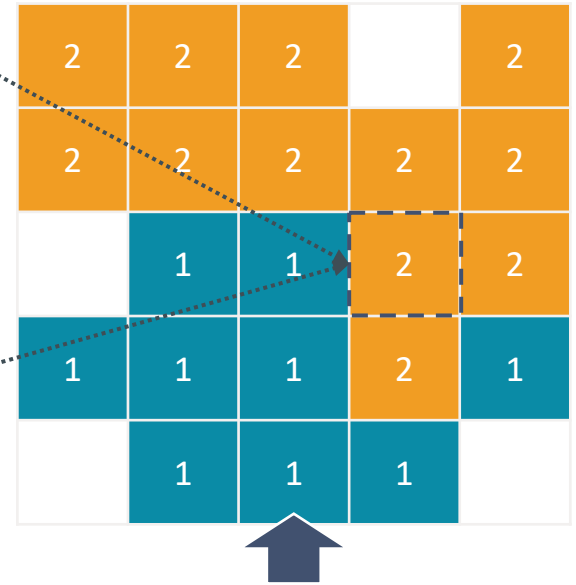
DominantFactor.weight_field= field in feature class to measure during analysis (shown by symbol size)




Floating zone definition (inherited from Factor class)




Temporary rasters measure intensity of values by each group. Here, raster A represents the intensity of the blue dots and raster B represents the intensity of the orange dots (groupings based on the value_field). Each rasters' values are weighted by the values in the weight_field for each feature




The resulting raster returns the integer of the index for the raster that has the highest value. In the outlined cell, the value in raster B (6) is higher than the value in raster A (1). The index for raster B is 2, since it was the second raster analyzed. Thus the resulting value is 2.

# LINEAR SUM CLASS

# LINEAR SUM CLASS (SUBCLASS OF FACTOR)

Primary purpose: create a Surface object by creating three temporary rasters for calculating a simple sum of values associated with polyline features in the floating zone area

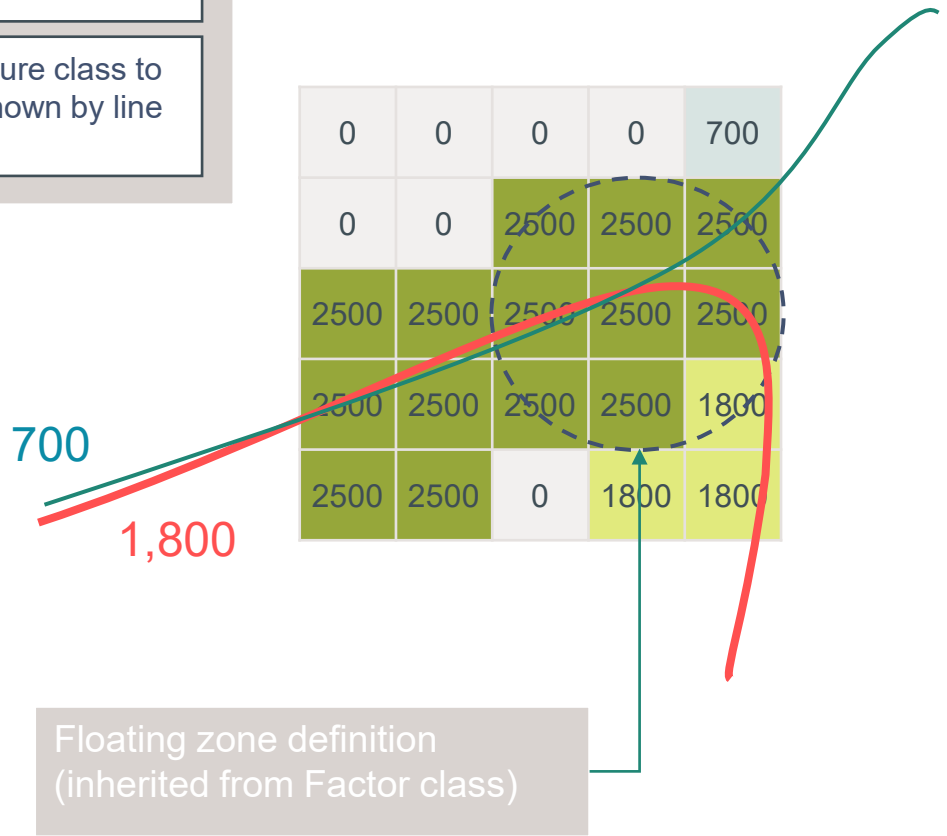
- Major attributes:
  - weight_field: The polyline values to summarize.
  - line_id_field: A field that uniquely identifies each polyline feature. This is required for counting features for the summarization calculation.
- Major methods:
  - rasterize: Creates three rasters of linear statistics: variety of line id's (count of features), cumulative length of line features, weighted length of line features (weighted by weight field); these are then used in an expression to obtain a simple sum of weight_field values (weighted length/(cumulative length/count)).
- Also inherits attributes and methods from the Factor and Surface classes
- The "SUM/LENGTH" analysis method in the factor class produces a weighted sum (value field * feature length) for linear features, reflecting the behavior of SpatialAnalyst's Linear Statistics tool. The LinearSum class accounts for line length and number of line features to provide a simple sum of linear values in a neighborhood.

# LINEAR SUM CLASS (SUBCLASS OF FACTOR)

Feature class specs (reference_fc inherited from Factor class)

LinearSum.weight_field= field in feature class for to be summed (shown by bandwidth)

LinearSum.line_id_field= field in feature class to uniquely identify each line feature(shown by line color)




The LinearSum class allows the values on the blue and red lines to be summed together.

The "SUM/LENGTH" method of the Factor class weights line values by line length, resulting in summary values that may be difficult to interpret. The LinearSum class automates a series of *LinearStatistics* geoprocessing runs to calculate a simple sum of line values.

# OVERLAY CLASS

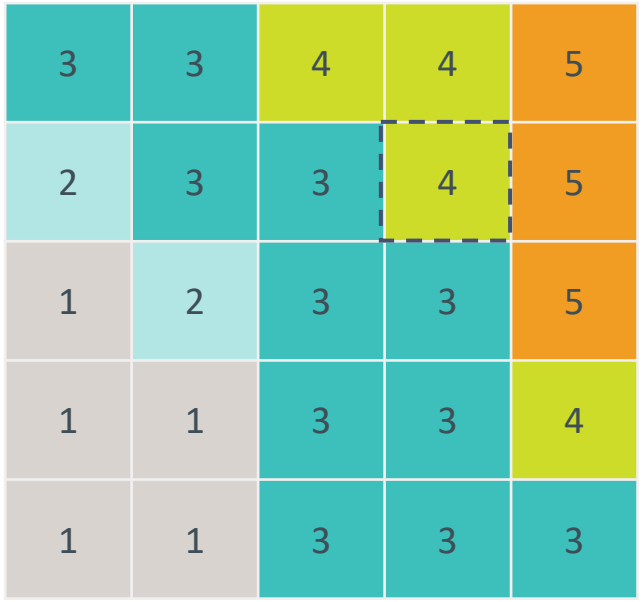
# OVERLAY CLASS (SUBCLASS OF SURFACE)

Primary purpose: create a new Surface object by overlaying two or more existing Surface objects

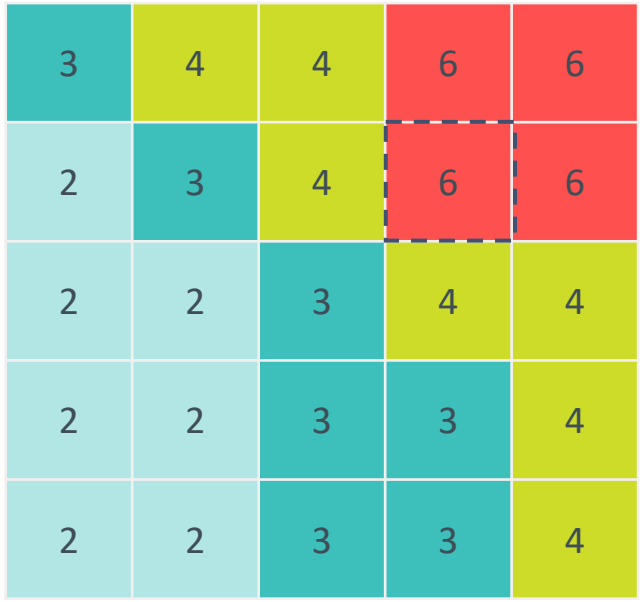
- Major attributes:
  - surfaces: What Surface objects (usually Factor objects) will be used in the overlay?
  - surface_weights: What weight should be assigned to each surface object listed in surfaces?
    - Python dictionary ({surface_object.name: weight})
  - evaluation_scale: What range of resulting values will be produced by the overlay analysis?
    - Python list ([from_value, to_value, by_value])
- Major methods:
  - addSurface/dropSurface: Manage which surfaces will be included in the overlay analysis
  - updateSurfaceWeights: Manage how surfaces will be weighted in the overlay analysis
  - overlaySurfaces: Run the weighted overlay analysis
- Also inherits attributes and methods from the Surface class

# OVERLAY CLASS (SUBCLASS OF SURFACE)

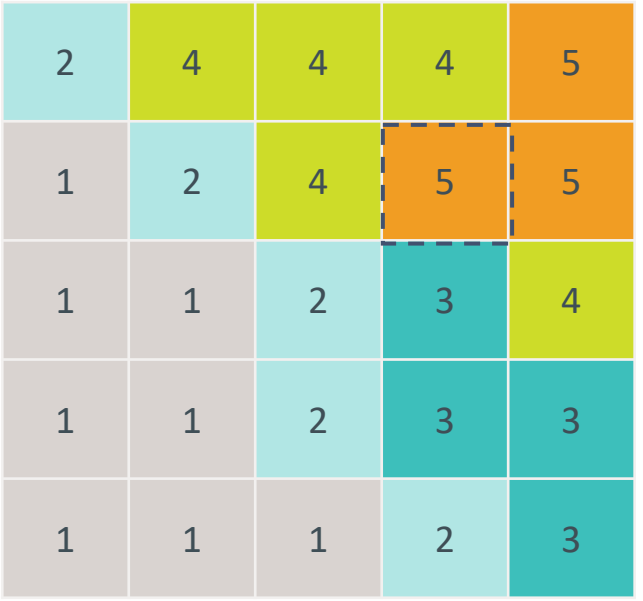
Raster A (34)




Raster B (33)




Raster C (33)




Overlay.surfaces = Input rasters that will be overlaid and analyzed. For this illustration, three surfaces will be analyzed.
Overlay.surface_weights = a dictionary containing the surface names and weights for use in the weighted overlay analysis. For this illustration, all surfaces are effectively weighted equally (see parenthetical values, which must sum to 100).

# OVERLAY CLASS (SUBCLASS OF SURFACE)

Raster A (34)

| | | | | |
|---|---|---|---|---|
| 3 | 3 | 4 | 4 | 5 |
| 2 | 3 | 3 | 4 | 5 |
| 1 | 2 | 3 | 3 | 5 |
| 1 | 1 | 3 | 3 | 4 |
| 1 | 1 | 3 | 3 | 3 |

Raster B (33)

| | | | | |
|---|---|---|---|---|
| 3 | 4 | 4 | 6 | 6 |
| 2 | 3 | 4 | 6 | 6 |
| 2 | 2 | 3 | 4 | 4 |
| 2 | 2 | 3 | 3 | 4 |
| 2 | 2 | 3 | 3 | 4 |

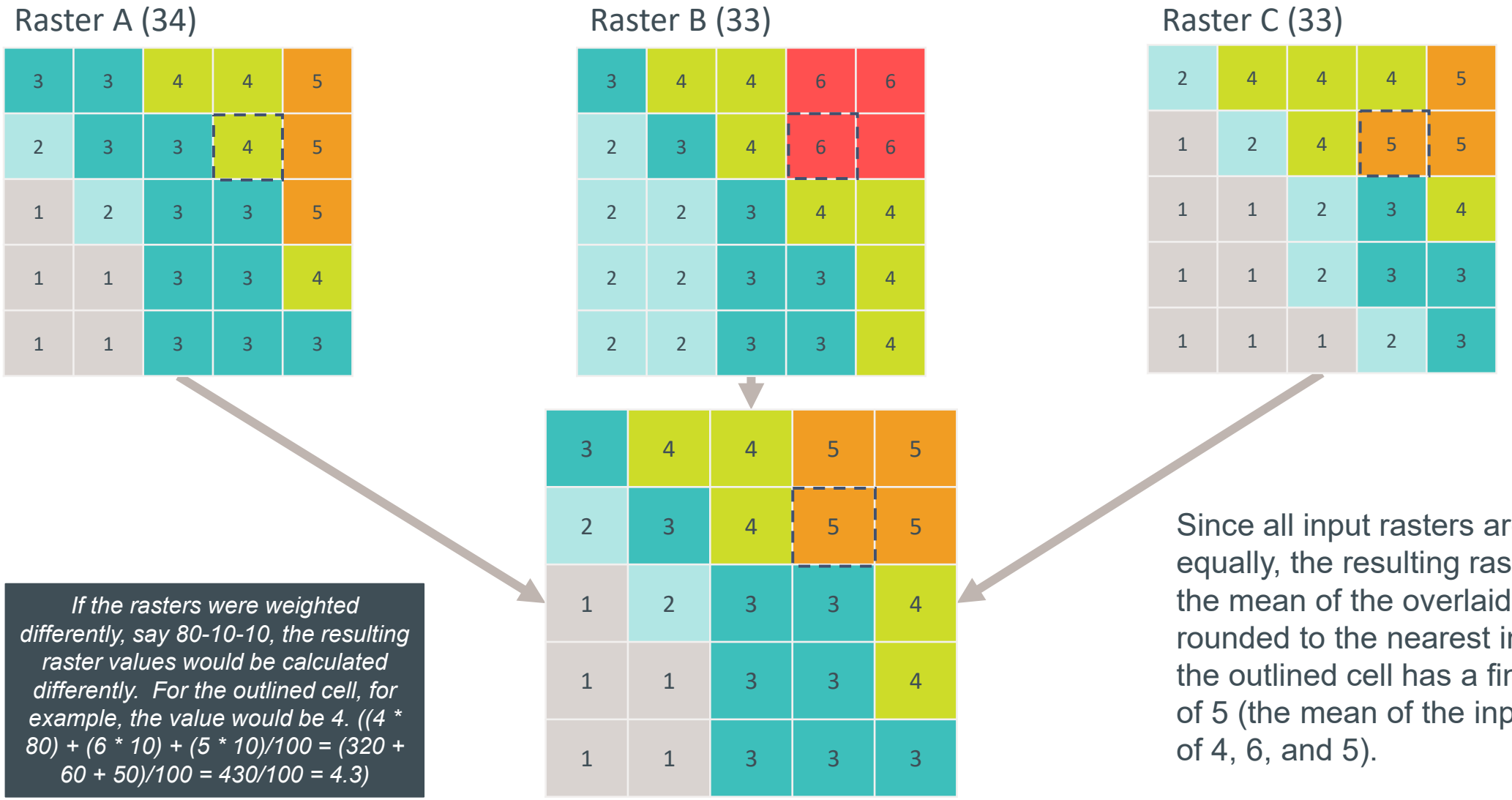
Raster C (33)

| | | | | |
|---|---|---|---|---|
| 2 | 4 | 4 | 4 | 5 |
| 1 | 2 | 4 | 5 | 5 |
| 1 | 1 | 2 | 3 | 4 |
| 1 | 1 | 2 | 3 | 3 |
| 1 | 1 | 1 | 2 | 3 |

| | | | | |
|---|---|---|---|---|
| 3,3,2 | 3,4,4 | 4,4,4 | 4,6,4 | 5,6,5 |
| 2,2,1 | 3,3,2 | 3,4,4 | 4,6,5 | 5,6,5 |
| 1,2,1 | 2,2,1 | 3,3,2 | 3,4,3 | 5,4,4 |
| 1,2,1 | 1,2,1 | 3,3,2 | 3,3,3 | 4,4,3 |
| 1,2,1 | 1,2,1 | 3,3,1 | 3,3,2 | 3,4,3 |

The three surfaces are overlaid and their values combined. Corresponding cells in the input rasters represent the same location using different measures. Thus the outlined cell is a single location, with values of 4, 6, and 5 in rasters A, B, and C respectively.

# OVERLAY CLASS (SUBCLASS OF SURFACE)




*If the rasters were weighted differently, say 80-10-10, the resulting raster values would be calculated differently. For the outlined cell, for example, the value would be 4. ((4 * 80) + (6 * 10) + (5 * 10)/100 = (320 + 60 + 50)/100 = 430/100 = 4.3)*

Since all input rasters are weighted equally, the resulting raster is effectively the mean of the overlaid input rasters, rounded to the nearest integer. Thus, the outlined cell has a final output value of 5 (the mean of the input raster values of 4, 6, and 5).

# COMBINATION CLASS

# COMBINATION CLASS (SUBCLASS OF SURFACE)

Primary purpose: create a new Surface object by combining two or more existing Surface objects

- Major attributes:
  - base_surface: The Surface object that will be modified based on the combination
  - adjustment_surfaces: The Surface object(s) that will be combined with the base_surface to return new values
    - Python dictionary ({surface_object.name: *parameters*})
  - adj_above_vaule/adj_below_value: The values in the base_surface raster above or below which adjustments from combinations will apply. Values outside of these bounds will retain their original value in the base_surface raster.
- Major methods:
  - combineSurfaces: execute the combination, returning resulting values based on the type of combination desired (see next slides on subclasses) according the specified parameters.
- Also inherits attributes and methods from the Surface class

# COMBINATION CLASS (SUBCLASS OF SURFACE)

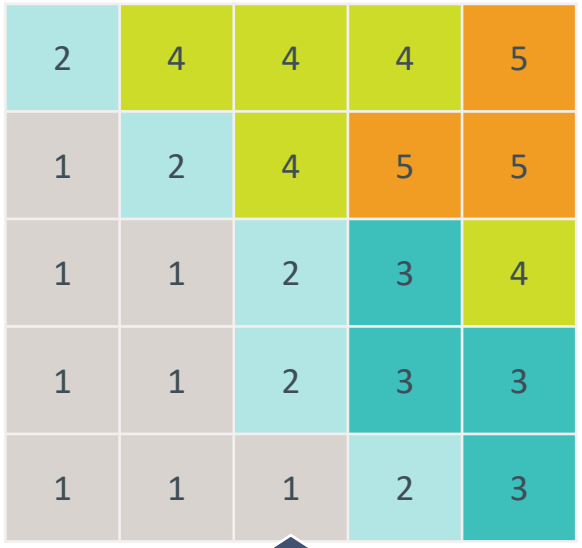
- Subclasses of Combination:
  - ConditionalCombination
  - CalculationCombination
  - LookupCombination

# CONDITIONAL COMBINATION (SUBCLASS OF COMBINATION)

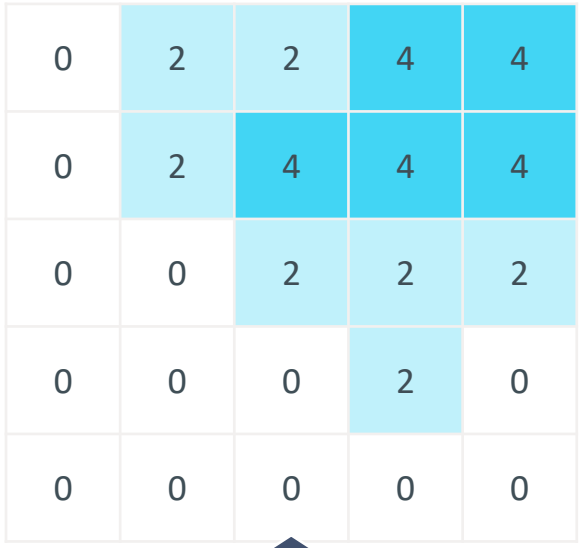
Primary purpose: create a new Surface object by combining two or more existing Surface objects, based on if-then style conditions

- Major methods:
  - addAdjustmentSurface: Update the adjustment_surfaces attribute (from Combination class), specifying the following parameters
    - adj_surface_obj: the adjustment surface object to be added
    - comparison: the comparison operation ("==3", ">3", "<=3", etc.) to use when applying the conditional logic in combining Surface objects
    - val_if_true: the value to return if the comparison returns a value of "TRUE"
    - val_if_false: the value to return if the comparison returns a value of "FALSE"
      - Use "base" to revert to the value in the base_surface raster when false
- Also inherits attributes and methods from the Combination class

# CONDITIONAL COMBINATION (SUBCLASS OF COMBINATION)

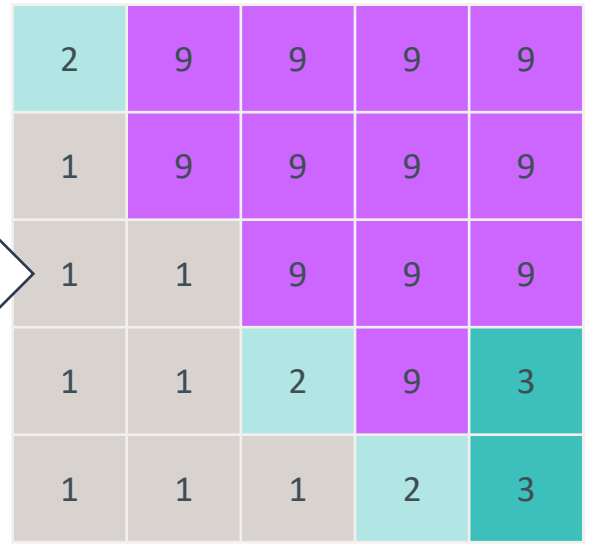



base_surface = The input raster that will be adjusted by any adjustment_surfaces




adjustment_surface = The input raster that will be used to modify the base_surface

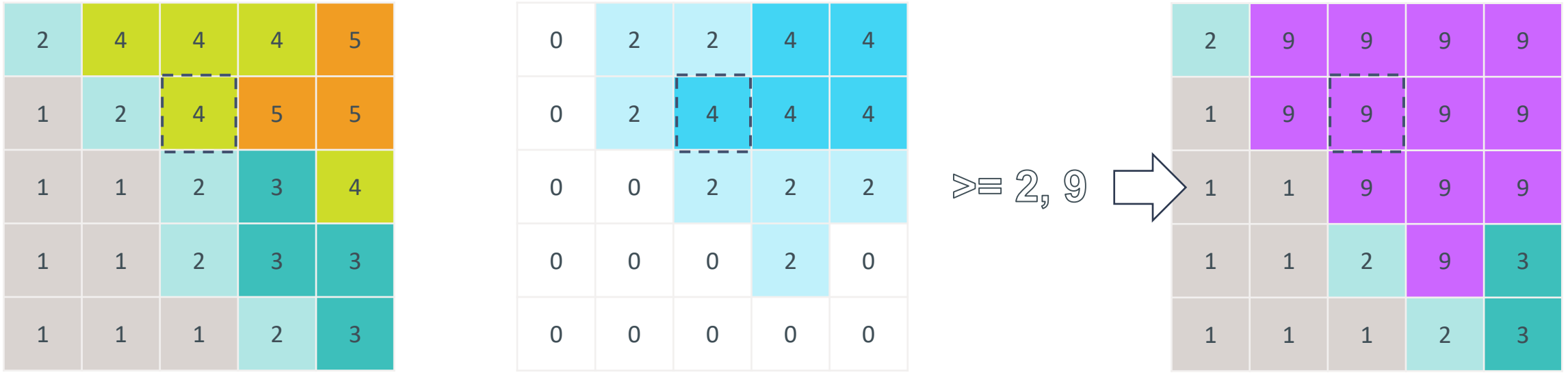
>= 2, 9




comparison = The condition to check for in the adjustment_surfaces

val_if_true = The new value to be assigned if the condition is true

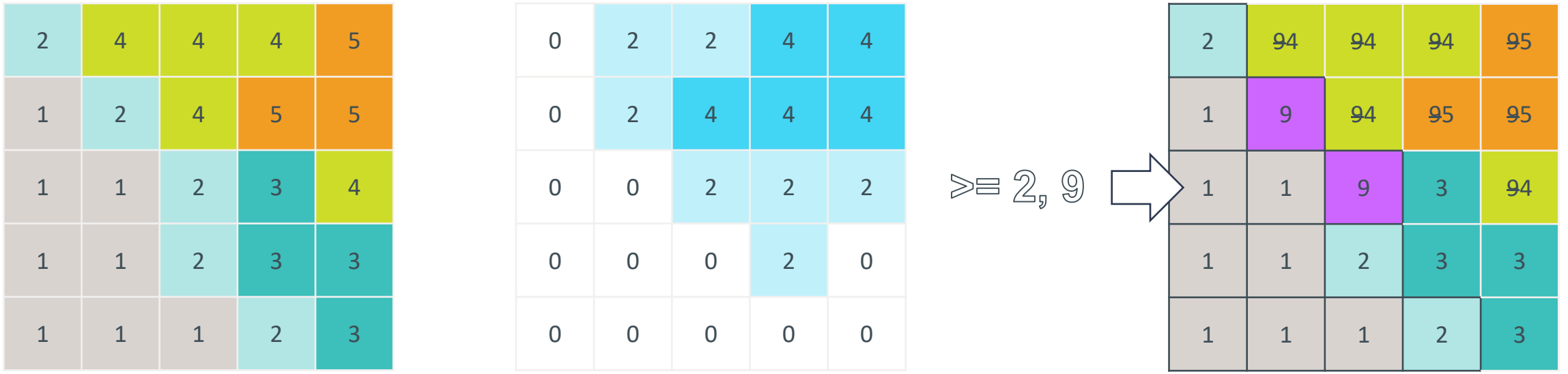
# CONDITIONAL COMBINATION (SUBCLASS OF COMBINATION)




Applying the condition…

- For the outlined cell,
  - the base_surface value is 4
  - The adjustment_surface value is 4
  - The condition is then applied using the comparison and the val_if_true
    - The conditions is TRUE (adjustment surface value of 4 is >= 2), so the resulting value is 9
- For all cells with adjustment surface values <2, the base surface value is retained, since there is no val_if_false attribute assigned

# CONDITIONAL COMBINATION (SUBCLASS OF COMBINATION)




Applying limits…

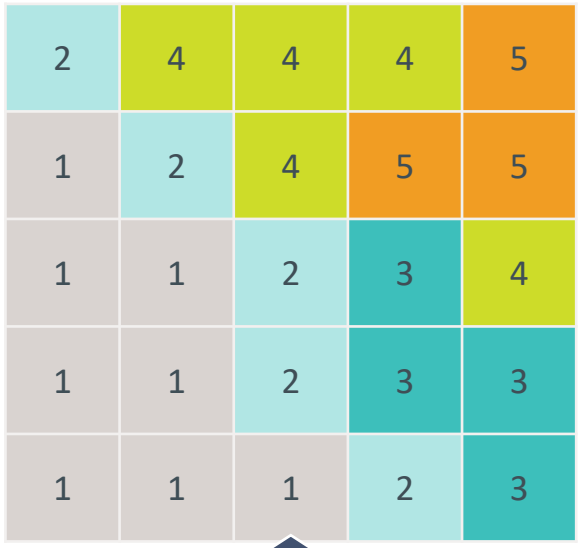
- Limits (adj_above_val/adj_below_val) affect which cells will be subject to the conditional logic
  - If adj_below_val = 3, only cells having a base surface of 2 or lower will be subject to the condition (dark borders)
  - All other base surface values are passed through without the application of the condition

# CALCULATION COMBINATION (SUBCLASS OF COMBINATION)

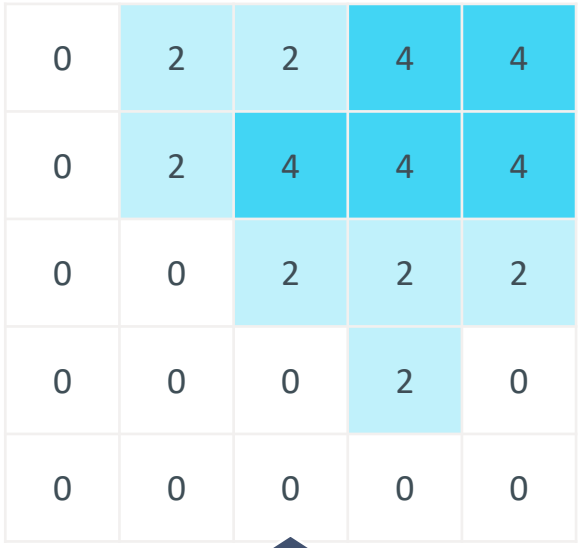
Primary purpose: create a new Surface object by combining two or more existing Surface objects, based on a mathematical expression

- Major attributes:
  - adj_lbound/adj_ubound: The lower/upper bound to apply to the calculation result
- Major methods:
  - addAdjustmentSurface: Update the adjustment_surfaces attribute (from Combination class), specifying the following parameters
    - adj_surface_obj: the adjustment surface object to be added
    - operator: the mathematical operator (“+”, “-”, “*”, “/” etc.) to use when applying the mathematical logic in combining Surface objects
    - surface_expr: any additional mathematical logic that should follow after the operator
- Also inherits attributes and methods from the Combination class

# CALCULATION COMBINATION (SUBCLASS OF COMBINATION)




base_surface = The input raster that will be adjusted by any adjustment_surfaces

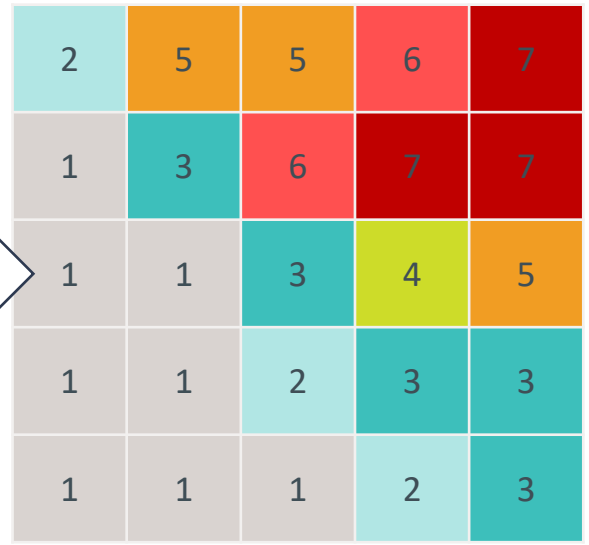



adjustment_surface = The input raster that will be used to modify the base_surface

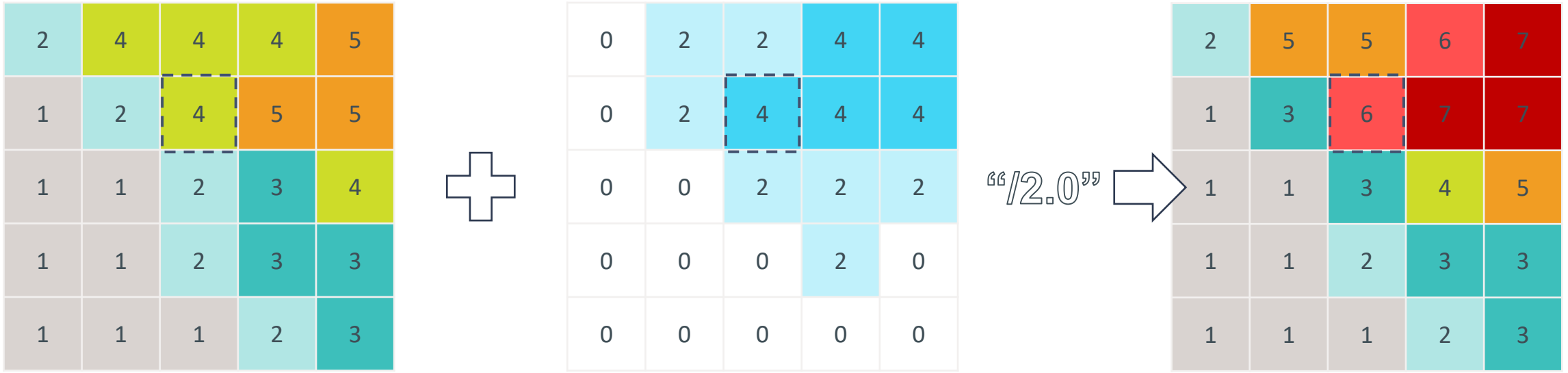
operator = The mathematical operation to be applied to this adjustment_surface

"/2.0"

surface_expr= Any additional mathematical logic to be applied to this adjustment_surface



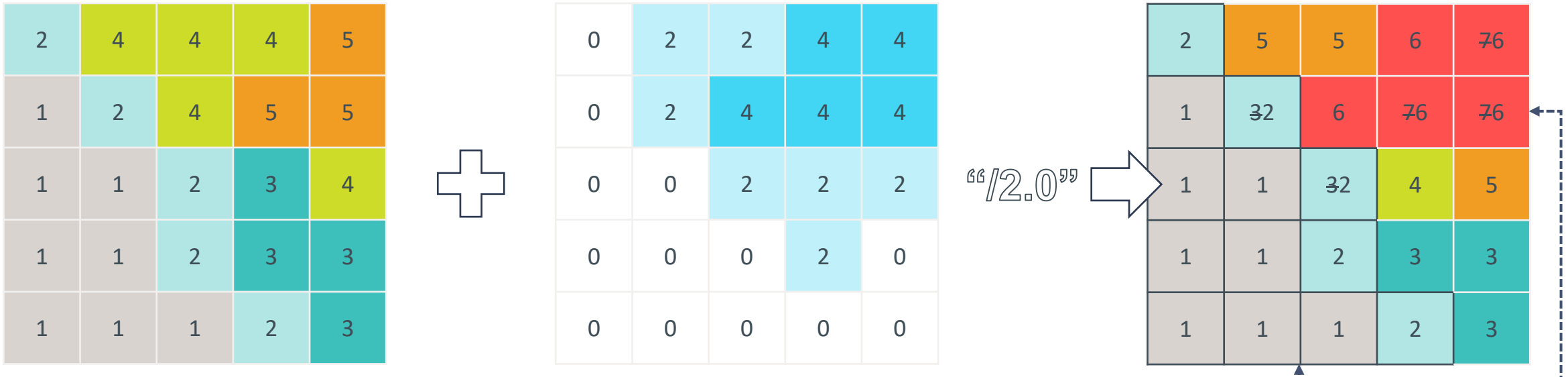
# CALCULATION COMBINATION (SUBCLASS OF COMBINATION)




Applying the calculation…

- For the outlined cell,
  - the base_surface value is 4
  - The adjustment_surface value is 4
  - The calculation is then applied using the operator and the surface_expr
    - Expression = "{base surface value} {operator} ({adjustment surface value} {surface_expr})"
    - Expression = 4 + (4/2) = 4 + 2 = 6

# CALCULATION COMBINATION (SUBCLASS OF COMBINATION)




Applying bounds and limits…

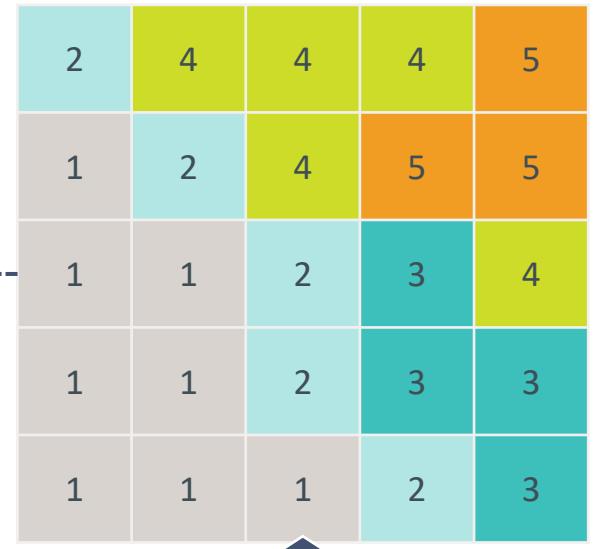
- Limits (adj_above_val/adj_below_val) affect which cells will be subject to the calculation
  - If adj_above_val = 2, only cells having a base surface of 3 or higher will be subject to the calculation (dark borders)
- Bounds (ubound/lbound) control the output of the calculation
  - If ubound = 6, any resulting value greater than 6 will be capped at 6 (red areas)

# LOOKUP COMBINATION (SUBCLASS OF COMBINATION)

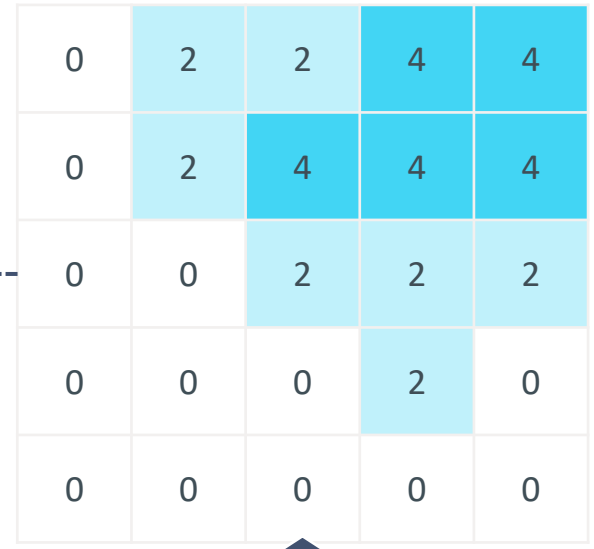
Primary purpose: create a new Surface object by combining two or more existing Surface objects, based on a table of combined values

- Major attributes:
  - lookup_table: The table defining how combinations of values will be reclassified
  - base_surface: The Surface object that serves as the “base” for the reclass. Any combination of values not addressed in the lookup table will be assigned their “base” value.
  - adjustment_surfaces: List of additional Surface objects that will be combined with the base surface. Resulting combinations of values will be reclassed according to the data in the lookup_table.
- Major methods:
  - addAdjustmentSurface: Update the adjustment_surfaces attribute (from Combination class), specifying the following parameters
    - lookup_column: the column in the lookup_table that corresponds to this surface, ensuring that value combinations are looked up properly
- Also inherits attributes and methods from the Combination class

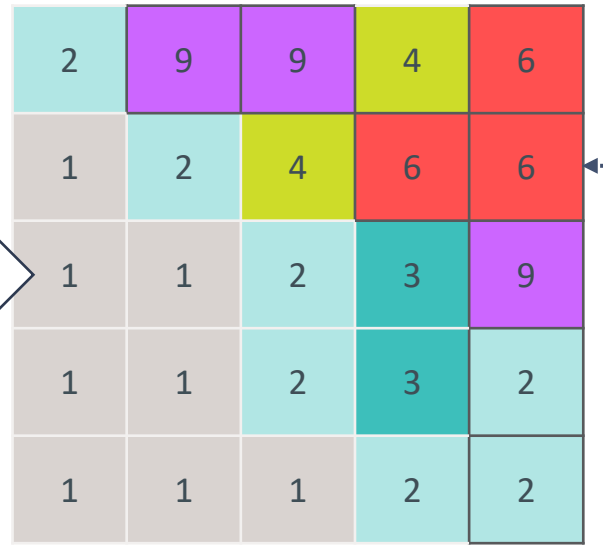
# LOOKUP COMBINATION (SUBCLASS OF COMBINATION)




base_surface




adjustment_surface




| Base value | Adj value | Reclass value |
|---|---|---|
| 3 | 0 | 2 |
| 4 | 2 | 9 |
| 5 | 2 | 9 |
| 5 | 4 | 6 |

Lookup table – for combinations of values in the base and adjustment tables, assign the new raster a new value. For unspecified combinations, retain the base value