Seasonal Operability & Road Layout

Colin MacMichael & John Davis

RESOURCE INNOVATIONS

Topics

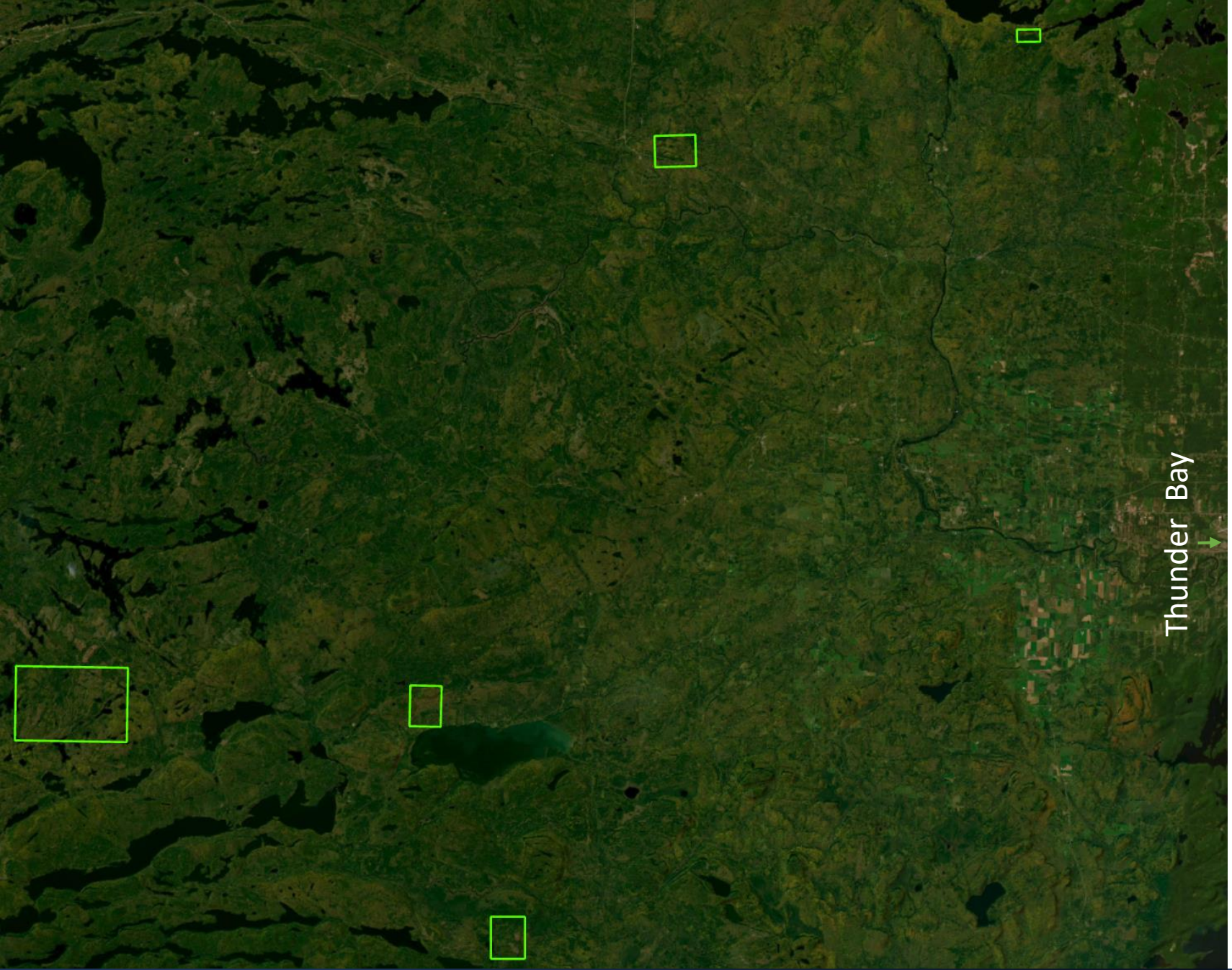
- Goals
- Steps
- Area
- Seasonal Operability
- Road Layout
- Future
- Considerations
- Questions

Goals

- Using LiDAR, eFRI and other spatial datasets
  - Refine the seasonal operability tool developed for KTTD round 2
  - Develop a road location optimization tool

Steps

- Familiarize with newer datasets
- Acquire field calibration points
- Update seasonal operability predictor to use new LiDAR derived data
- Create road layout optimization model
- Convert models to ArcPy Python Toolbox



Model Development Sites

Sustainable Forest License areas west of Thunder Bay, Ontario

These areas:

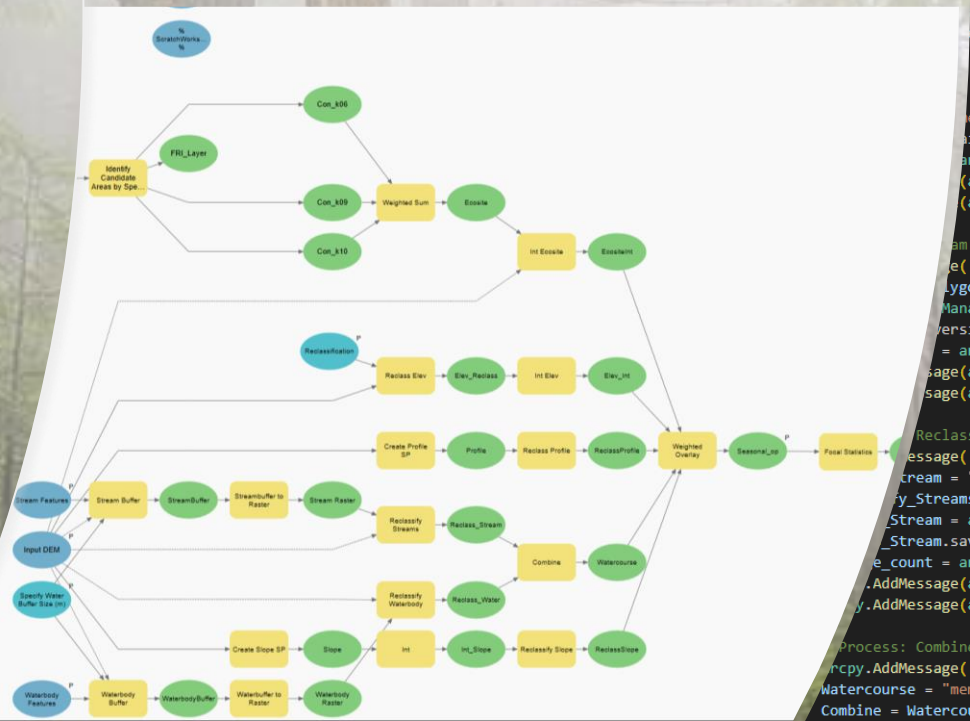
- Overlapped with all required datasets.
- Had access for ground truthing.
- Had existing resource road networks to benchmark outputs against.

Datasets

- Inputs
  - eFRI (ecosite numbers)(polygon)(Seasonal Operability)
  - LiDAR derived DEM (raster)
  - Ontario waterbodies and watercourses (polygon/polyline)
  - Workspace (geodatabase)
- Output
  - Seasonal operability raster (raster)
  - Road layout(polyline)

Seasonal Operability

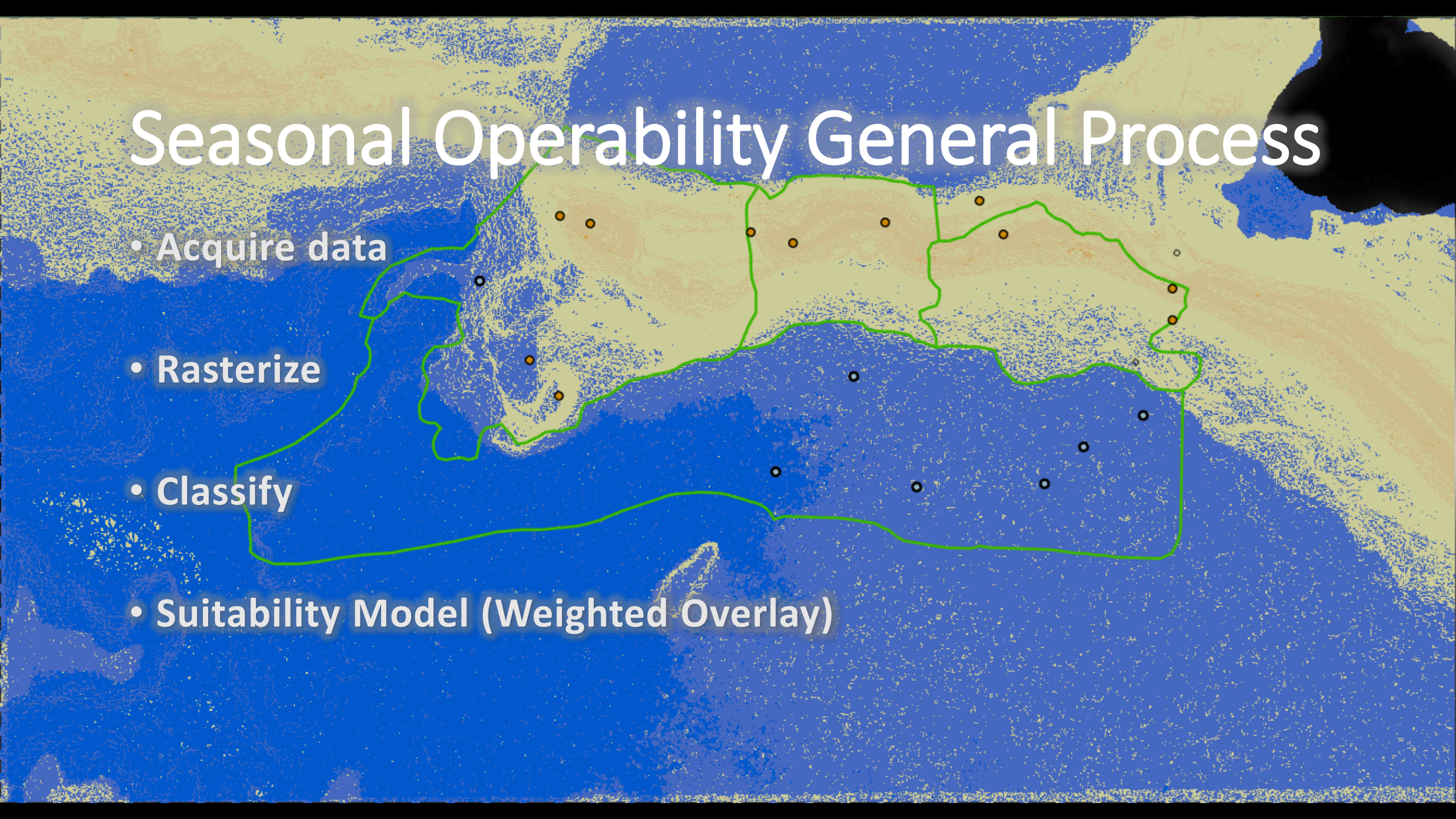
Choosing a time of year to harvest depending on the amount of ground moisture




```
arcpy.management.MakeFeatureLayer(input_FRI_feature, FRI_Layer, where_clause=where_clause)
arcpy.management.AddField(FRI_Layer, field_name="RasVal", field_type="SHORT")
arcpy.management.CalculateField(FRI_Layer, field="RasVal", expression="1")
.AddMessage('FRI Feature to Raster')
ite = fr"memory/Ecosite"
.conversion.FeatureToRaster(FRI_Layer, "RasVal", Ecosite, cell_size='3')
e_count = arcpy.GetMessageCount()
ddMessage(arcpy.GetMessage(0))
dMessage(arcpy.GetMessage(message_count - 1))

: Buffer Waterbody (Pairwise Buffer) (analysis)
essage('Buffering Waterbodies')
fer = "memory\\WB_Buffer"
sis.PairwiseBuffer(in_features=input_waterbody, out_feature_class=WbBody_Buffer, buffer_distance_or_field
t = arcpy.GetMessageCount()
sage(arcpy.GetMessage(0))
age(arcpy.GetMessage(message_count - 1))

erbody Buffer to Raster (Polygon to Raster) (conversion)
ge('Waterbody Buffer to Raster')
Polygon = "memory\\Stream_PolygonToRaster"
anager(extent=input_surface):
rsion.PolygonToRaster(in_features=WbBody_Buffer, value_field="BUFF_DIST", out_rasterdataset=Waterbody
arcpy.GetMessageCount()
(arcpy.GetMessage(0))
(arcpy.GetMessage(message_count - 1))

sify Waterbody (Reclassify) (sa)\
'Reclassify Waterbody Raster')
memory\\Reclass_Water"
ody = Reclass_Water
rcpy.sa.Reclassify(WaterbodyBufferPolygon, "VALUE", "1 1000000 NODATA;NODATA 3", "DATA")
rcpy.GetMessageCount()
arcpy.GetMessage(0))
arcpy.GetMessage(message_count - 1))
e(Reclassify_Waterbody)

Buffer (Pairwise Buffer) (analysis)
'Buffering Watercourses')
emory\\WCclipBuffer"
irwiseBuffer(in_features=input_watercourse, out_feature_class=StreamBuffer, buffer_distance_or_fiel
arcpy.GetMessageCount()
(arcpy.GetMessage(0))
(arcpy.GetMessage(message_count - 1))

am Buffer to Raster (Polygon to Raster) (conversion)
e('Watercourse Buffers to Raster')
lygon = "memory\\Stream_PolygonToRaster"
Manager(extent=input_surface):
ersion.PolygonToRaster(in_features=StreamBuffer, value_field="BUFF_DIST", out_rasterdataset=StreamBuff
= arcpy.GetMessageCount()
sage(arcpy.GetMessage(0))
sage(arcpy.GetMessage(message_count - 1))

Reclassify Streams (Reclassify) (sa)
essage('Reclassifying Watercourse Raster')
tream = "memory\\Reclass_Stream"
y_Streams = Reclass_Stream
Stream = arcpy.sa.Reclassify(StreamBufferPolygon, "VALUE", "1 1000000 4;NODATA 1", "DATA")
_Stream.save(Reclassify_Streams)
e_count = arcpy.GetMessageCount()
.AddMessage(arcpy.GetMessage(0))
.AddMessage(arcpy.GetMessage(message_count - 1))

Process: Combine (Combine) (sa)
rcpy.AddMessage('Combining Water Features')
Watercourse = "memory\\Watercourse"
Combine = Watercourse
```

Seasonal Operability General Process

- Acquire data
- Rasterize
- Classify
- Suitability Model (Weighted Overlay)



Geoprocessing

Seasonal Operability

Parameters Environments

* FRI Layer

* Watercourse Feature

* Waterbody Feature

Buffer Distance

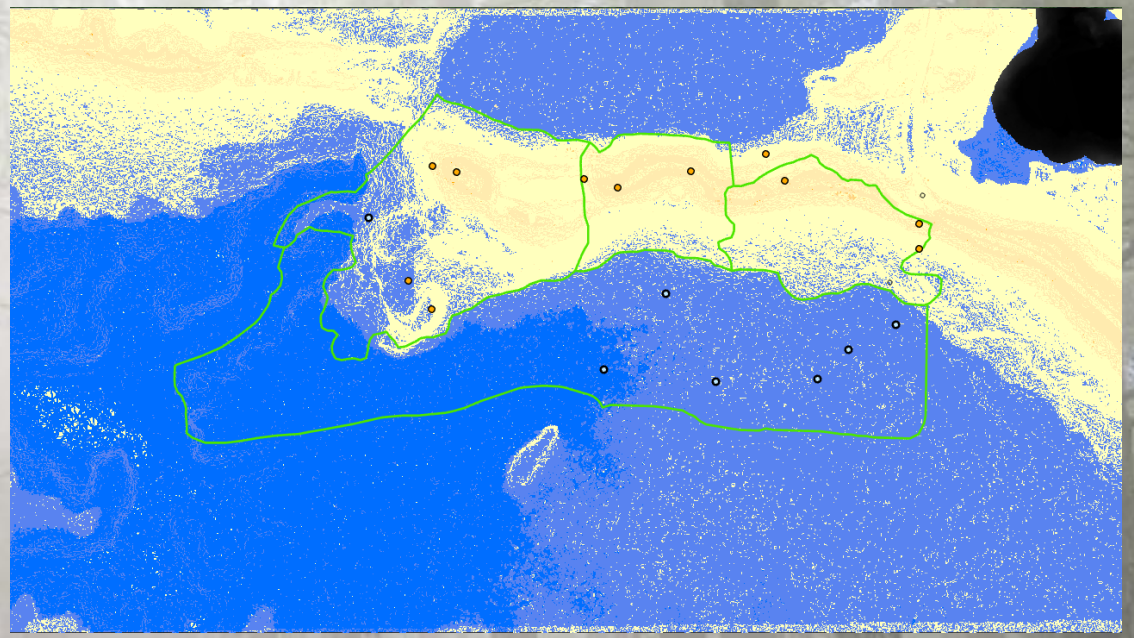
30 Meters

* Input Surface

* Workspace

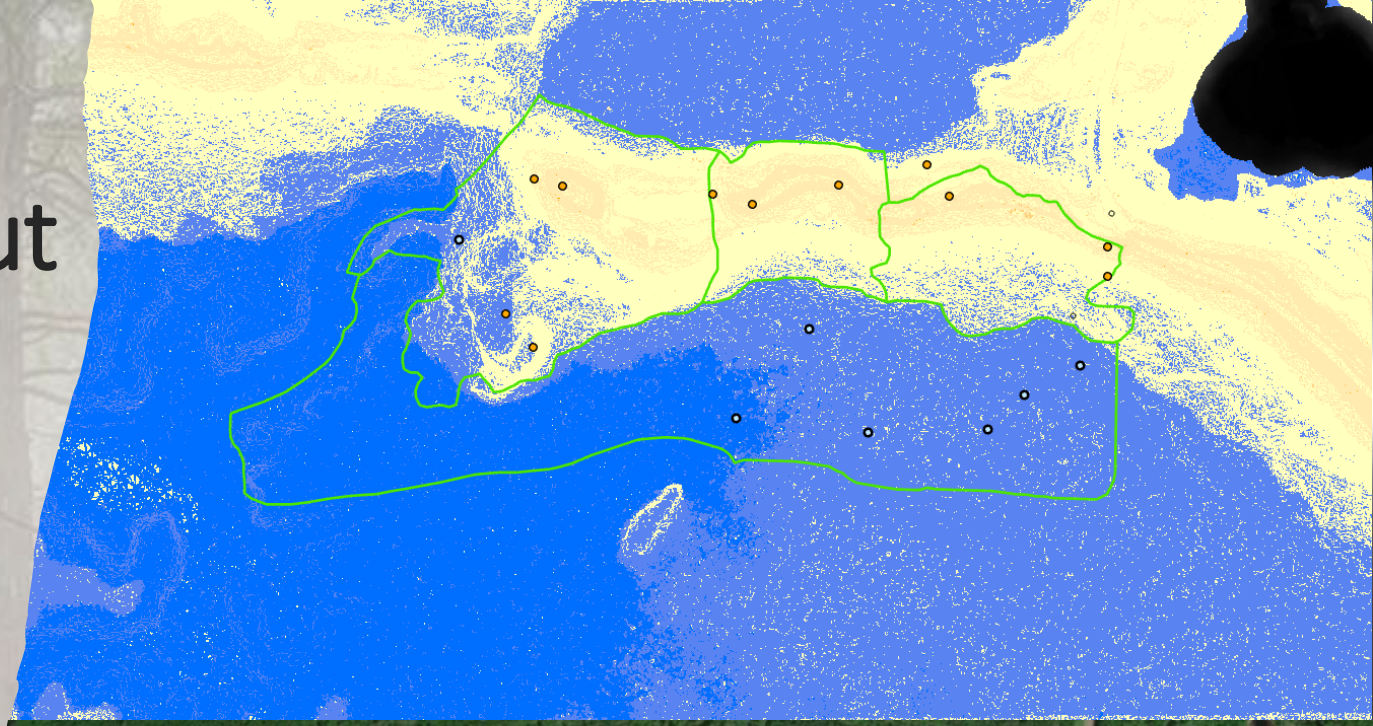
Seasonal Operability Output

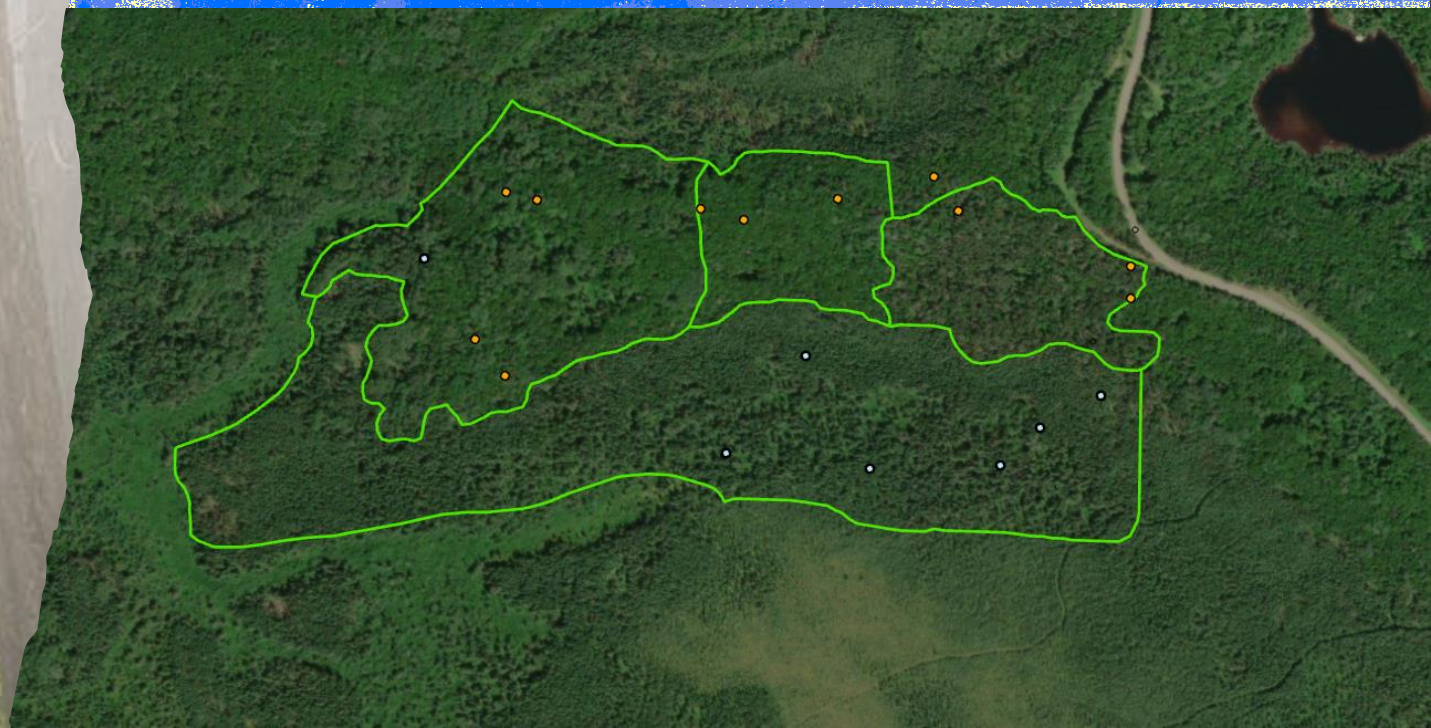
SeasonalOp



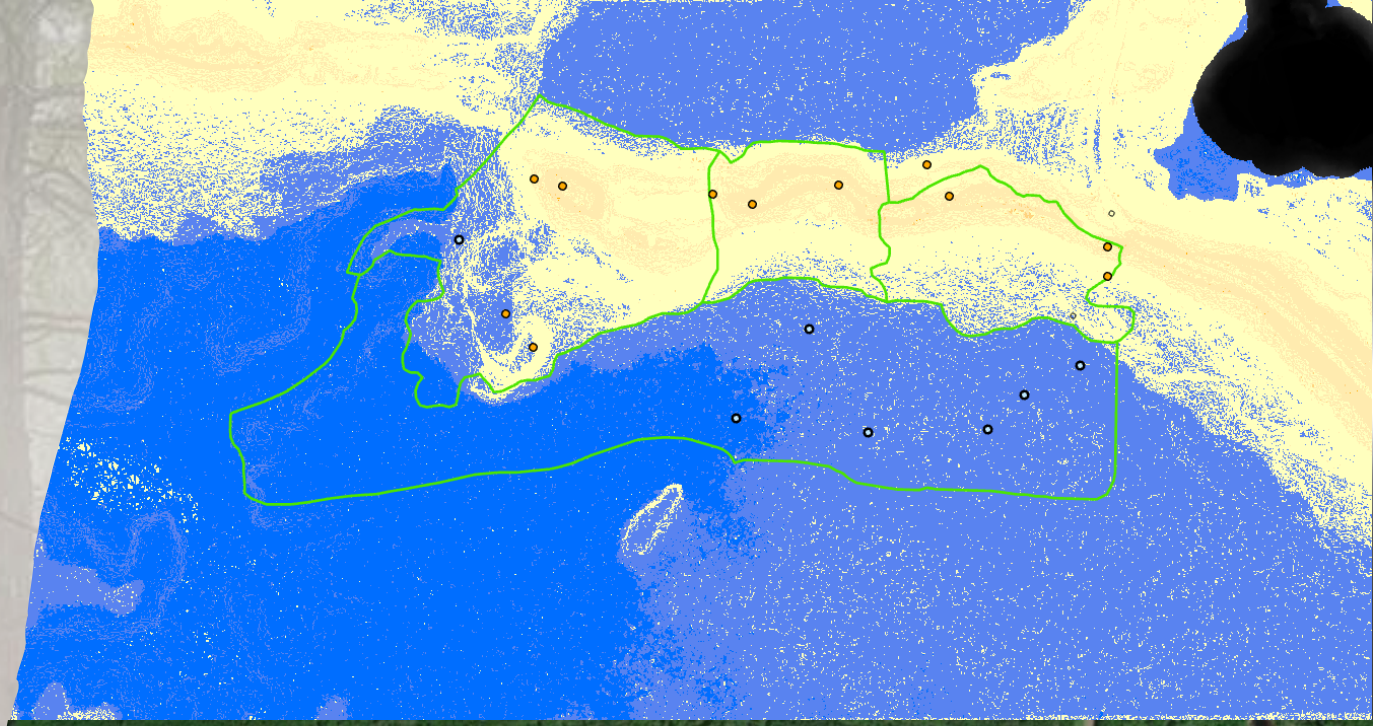
Seasonal Operability Output

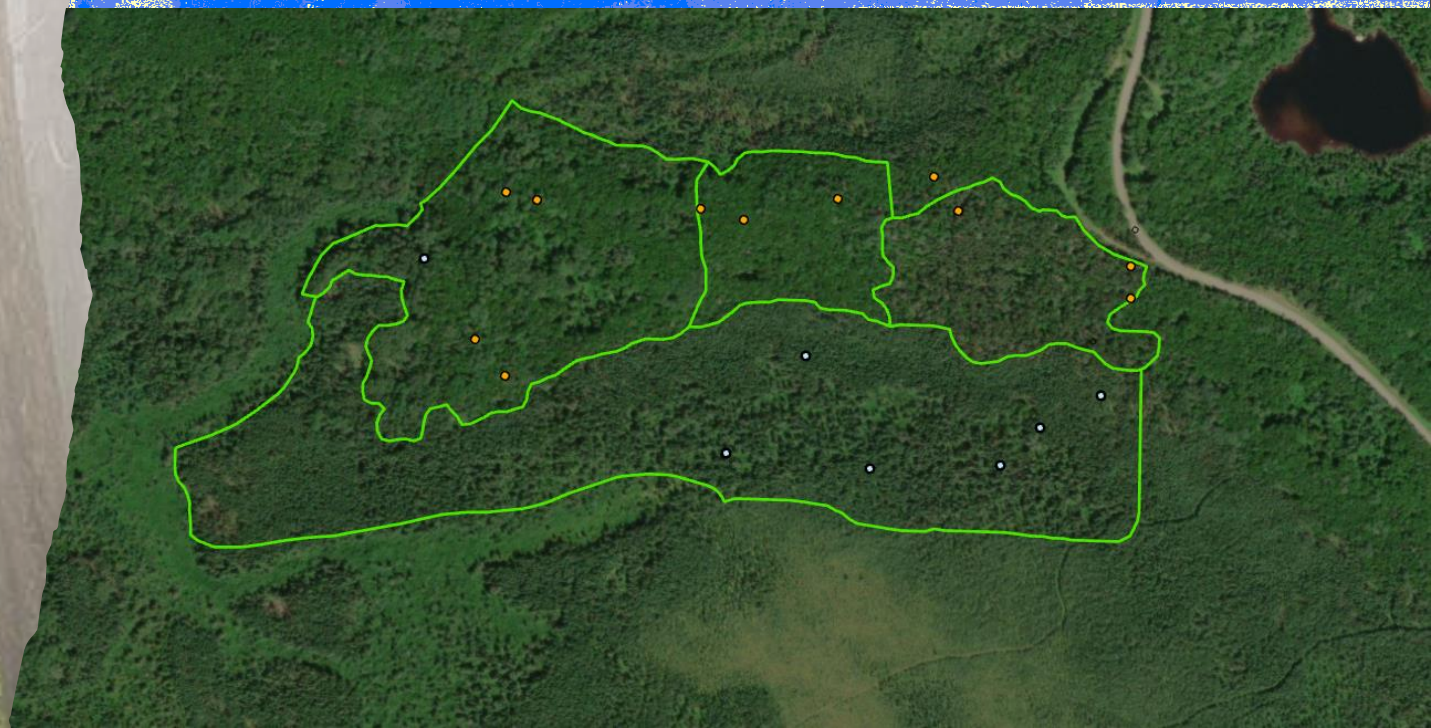
- Low lying areas have poor drainage and a tendency to hold water. These areas often have organic soil properties, making them difficult to operate on when ground isn't frozen.



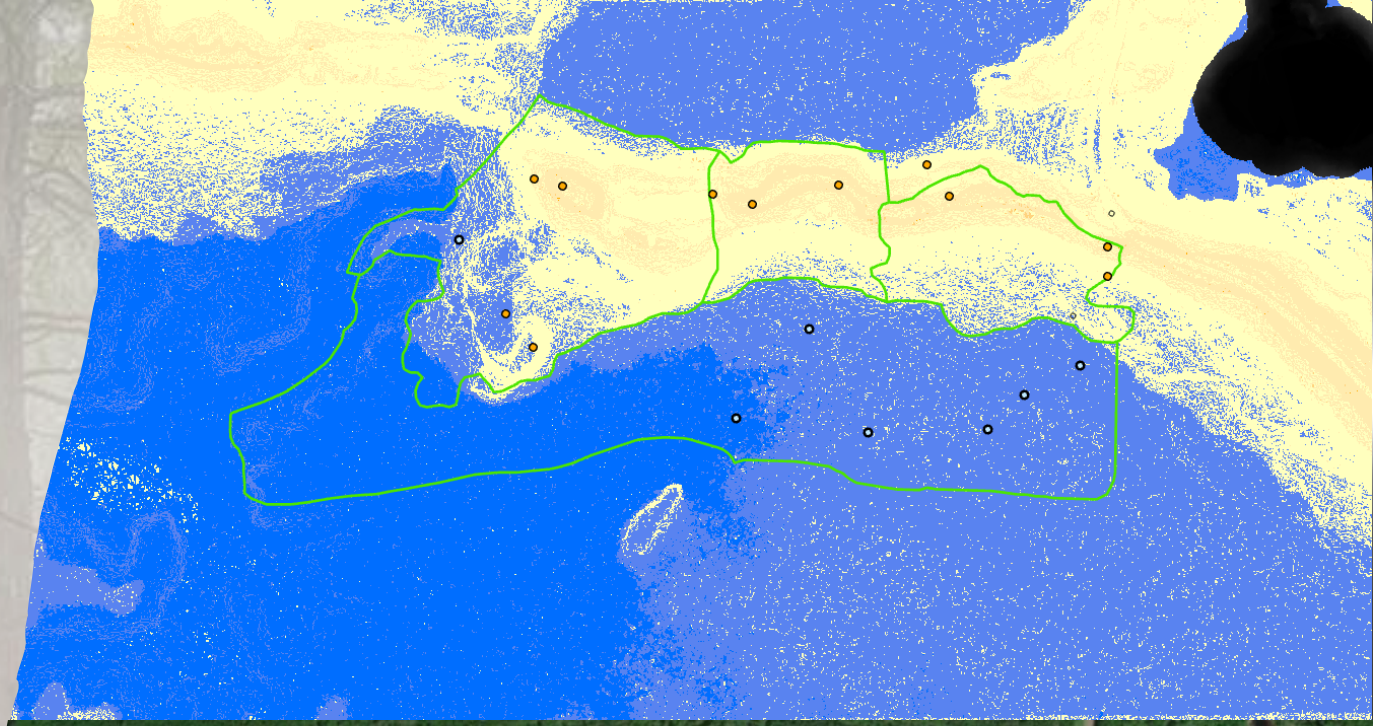


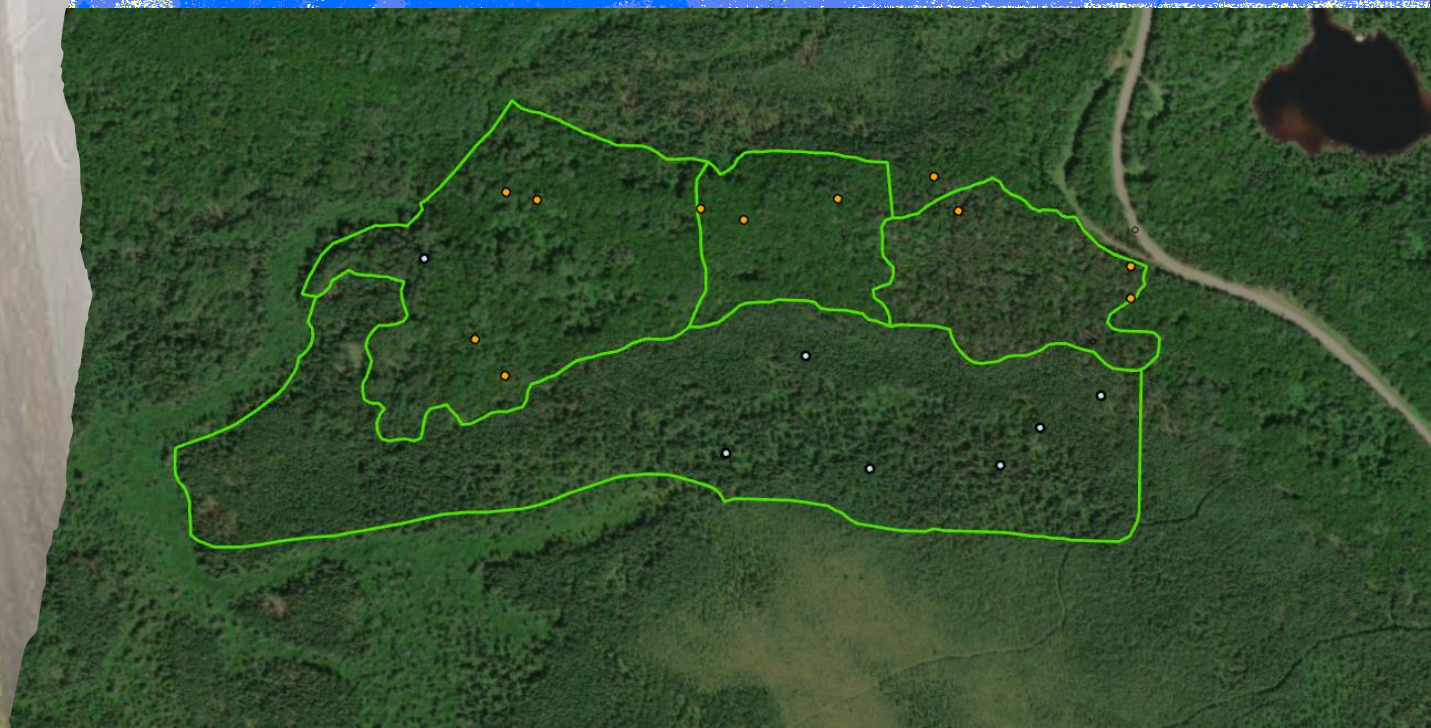
- Mid slopes are often well drained, and while water can hold up in some areas (drainage areas, convex slopes, etc.), they are more likely to have mineral soil properties and be operable throughout the year.



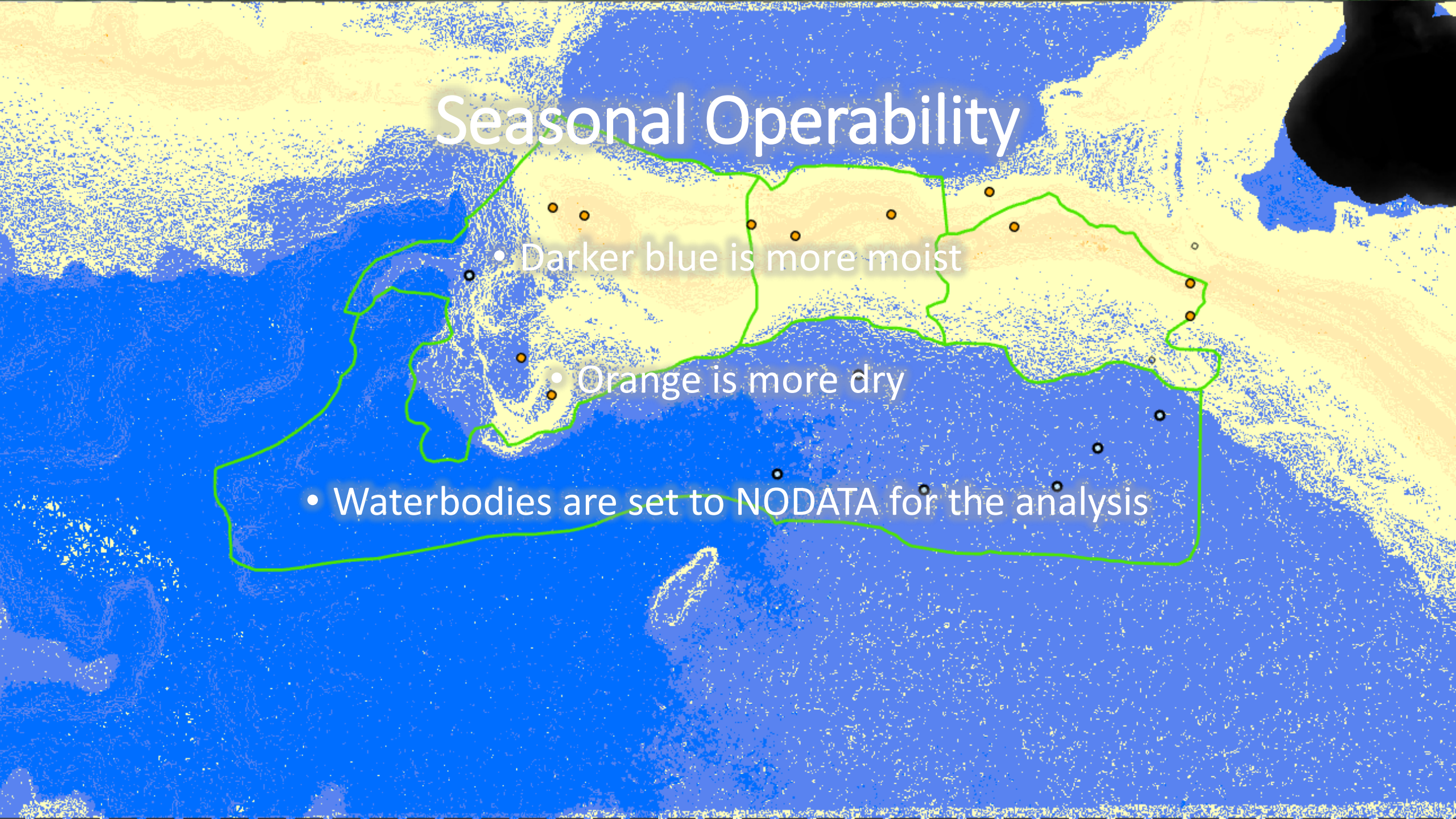


- Upper slopes and top of hills are generally well drained and have mineral soil properties. These areas are generally well drained and operable all year round.





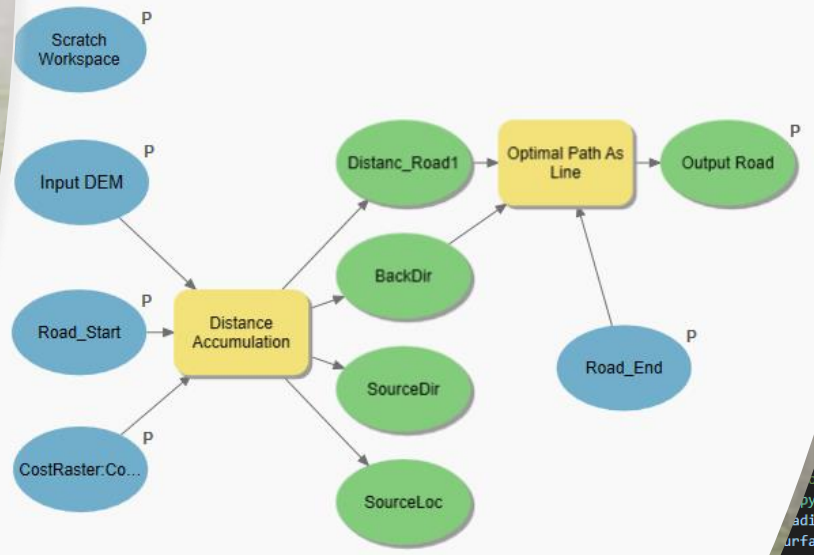
Seasonal Operability



- Darker blue is more moist
- Orange is more dry
- Waterbodies are set to NODATA for the analysis

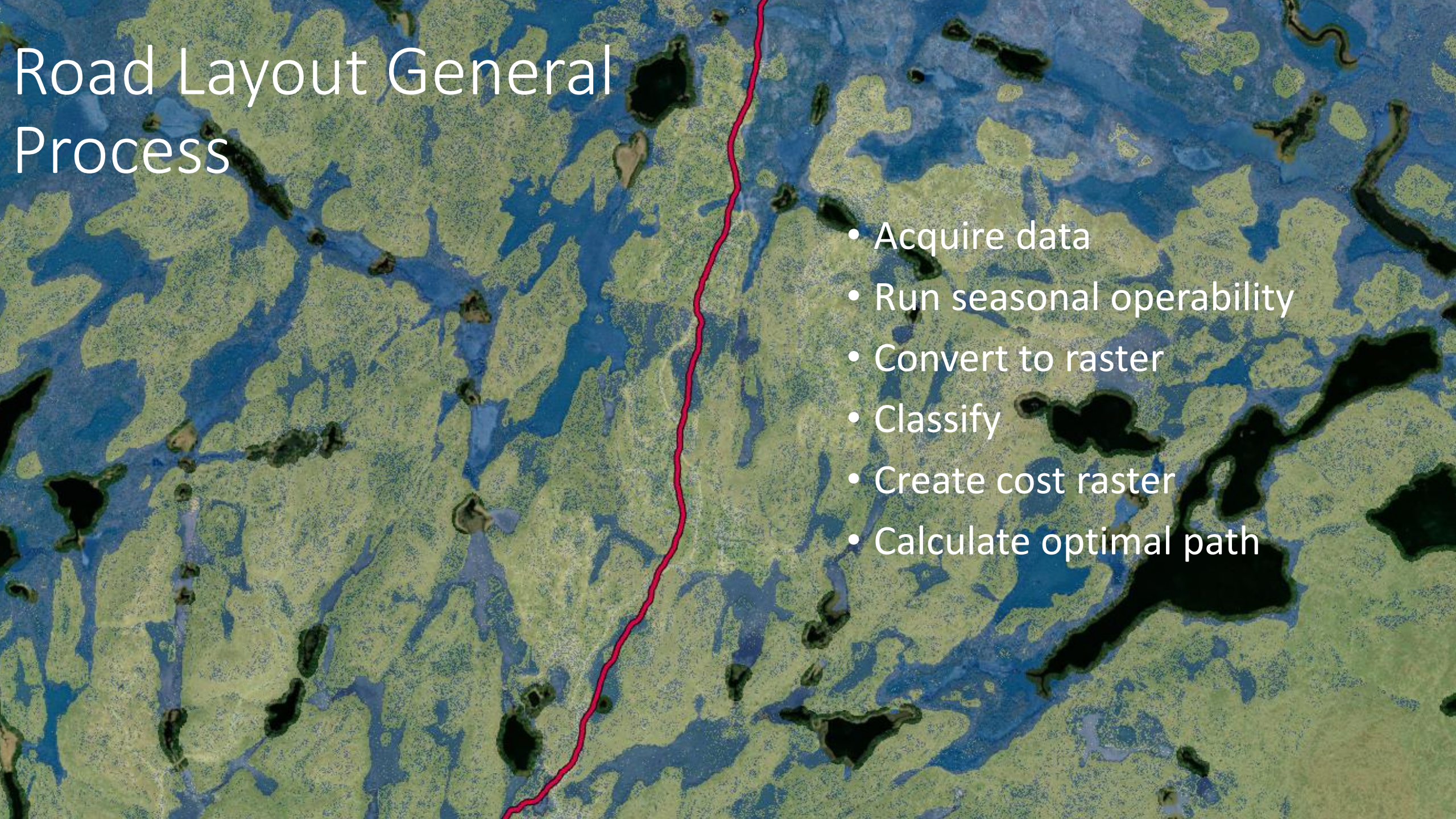
Road Layout

- This tool aims to automate more of the office planning portion of forest access road construction




```
Surface_Parameters = Plan
Plan = arcpy.sa.SurfaceParameters(input_surface,
                                  "MEAN_CURVATURE", "QUADRATIC", "1 Meters", "FIXED_NEIGHBORHOOD",
                                   "Meter", "DEGREE", "GEODESIC_AZIMUTHS", "NORTH_POLE_ASPECT", "")

n.save(Surface_Parameters)

ocess: Reclassfy Plan (Reclass) (sa)
.AddMessage('Reclass Plan')
sPlan = "memory\\Reclass_Plan1"
ify_Plan = ReclassPlan
lan = arcpy.sa.Reclassify(Plan,"Value", RemapRange([[-50,-0.5,5],[-0.5,-0.0001,3],
                                                    [-0.0001,0.0001,9],[0.0001,0.5,1],[0.5,50,5]]))
ount = arcpy.GetMessageCount()
essage(arcpy.GetMessage(0))
essage(arcpy.GetMessage(message_count - 1))
.save(Reclassify_Plan)

nt Plan (Int) (sa)
sage('Reclass Plan Int')
emory\\Int_Reclass_1"
an_Int
py.sa.Int(ReclassPlan)
Int_Plan)
= arcpy.GetMessageCount()
ge(arcpy.GetMessage(0))
e(arcpy.GetMessage(message_count - 1))

ace Profile (Surface Parameters) (sa)
e('Profile Curvature')
y\\Profile"
= Profile
.sa.SurfaceParameters(input_surface,
                      "PROFILE_CURVATURE", "QUADRATIC", "1 Meters", "FIXED_NEIGHBORHOOD",
                       "Meter", "DEGREE", "GEODESIC_AZIMUTHS", "NORTH_POLE_ASPECT", "")
arcpy.GetMessageCount()
(arcpy.GetMessage(0))
(arcpy.GetMessage(message_count - 1))
face_Profile)

assify Profile (Reclass) (sa)
e('Reclass Profile')
emory\\Reclass_Prof1"
file = ReclassPro
rcpy.sa.Reclassify(Profile, "VALUE",RemapRange([[-10000,-0.500000,5],[-0.500000,-0.000100,3],
                                                [-0.000100,0.000100,9],[0.000100,0.500000,1],[0.500000,10000,5]]))
ve(Reclassify_Profile)
= arcpy.GetMessageCount()
sage(arcpy.GetMessage(0))
sage(arcpy.GetMessage(message_count - 1))

Int Pro (Int) (sa)
ssage('Reclass Profile Int')
"memory\\Int_ProfileR1"
Pro_Int
arcpy.sa.Int(ReclassPro)
count = arcpy.GetMessageCount()
dMessage(arcpy.GetMessage(0))
dMessage(arcpy.GetMessage(message_count - 1))
t.save(Int_Pro)

ocess: Surface Slope (Surface Parameters) (sa)
y.AddMessage('Slope')
adient = "memory\\Slope"
rface_Slope = Gradient
radient = arcpy.sa.SurfaceParameters(input_surface, "SLOPE",
                                     "QUADRATIC", "1 Meters", "FIXED_NEIGHBORHOOD",
                                     "Meter", "PERCENT_RISE", "GEODESIC_AZIMUTHS", "NORTH_POLE_ASPECT", "")
message_count = arcpy.GetMessageCount()
```

Road Layout General Process



- Acquire data
- Run seasonal operability
- Convert to raster
- Classify
- Create cost raster
- Calculate optimal path

Geoprocessing

Road Layout

Parameters Environments

* Road Start

* Road End

* Watercourse Feature

* Waterbody Feature

Buffer Distance

30 Meters

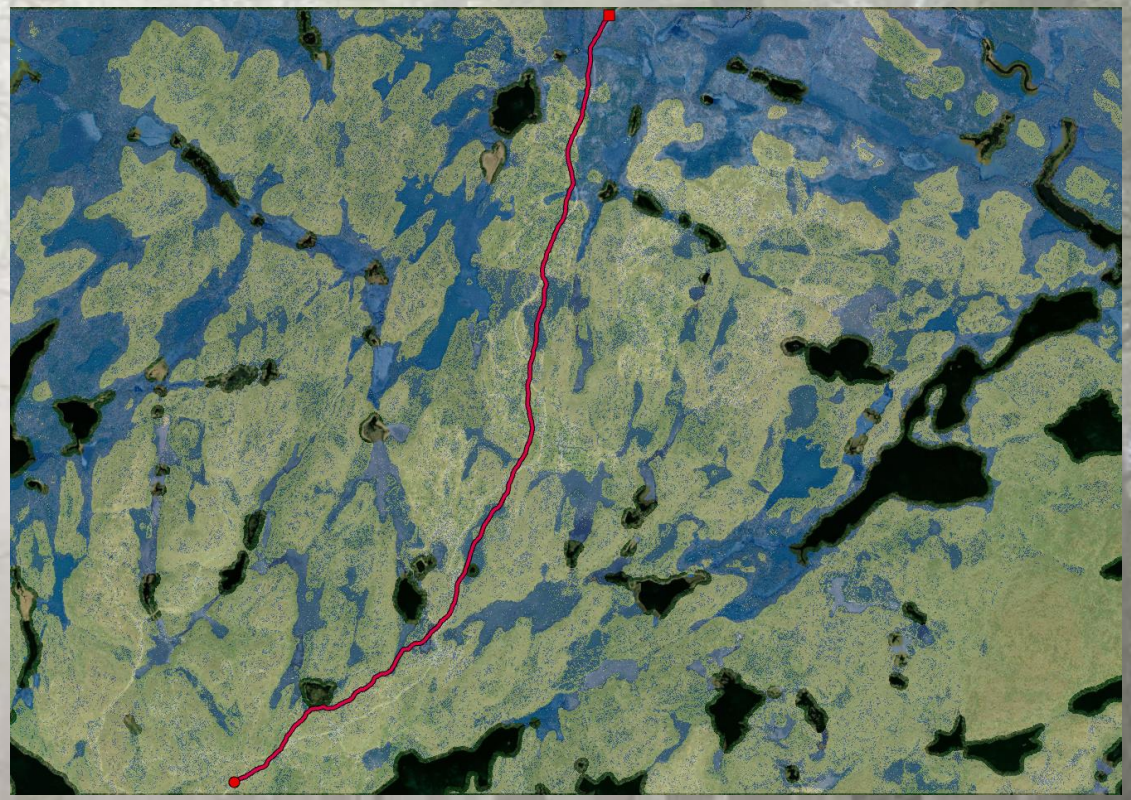
* Input Surface

* Workspace

* Seasonal Operability

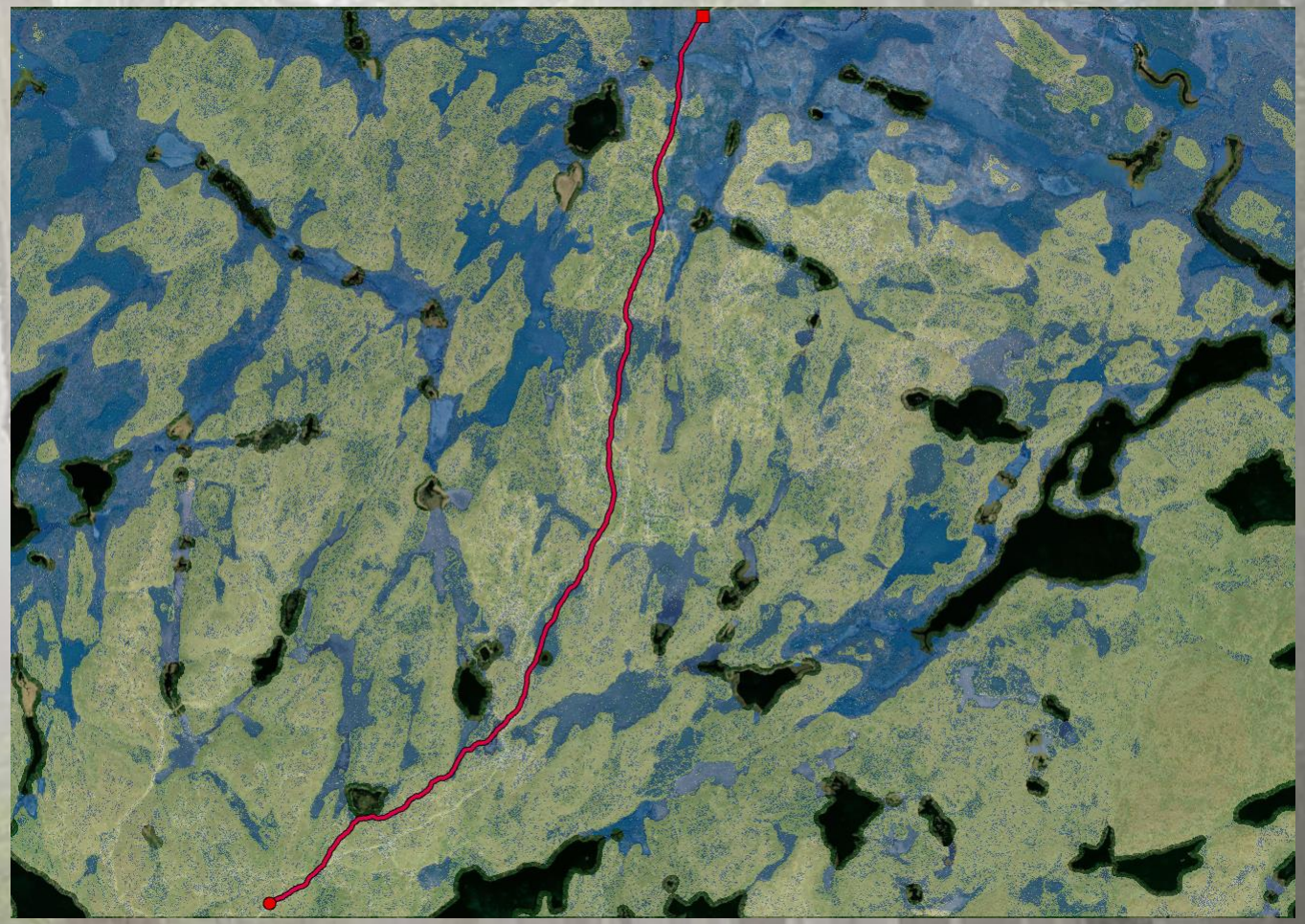
Output Road

RoadLayout



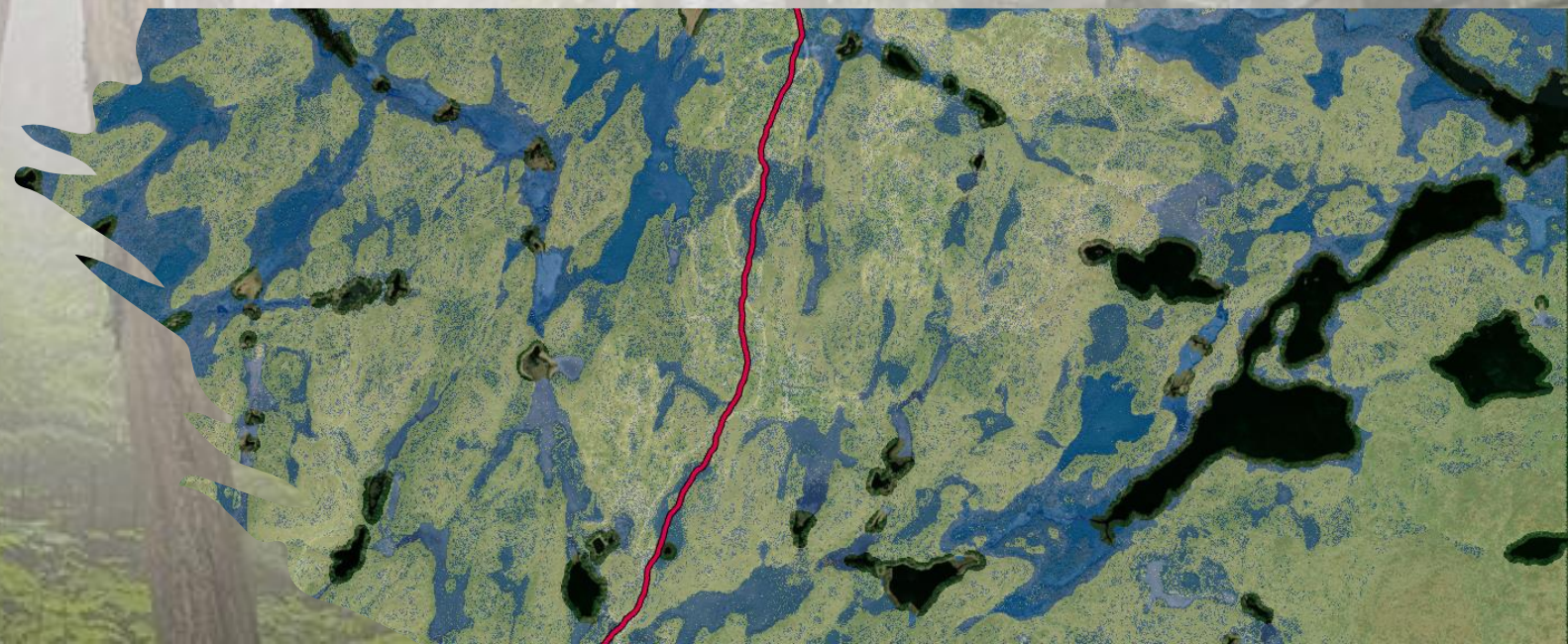
Road Layout

- Road takes most efficient route between two points, according to the inputs provided.



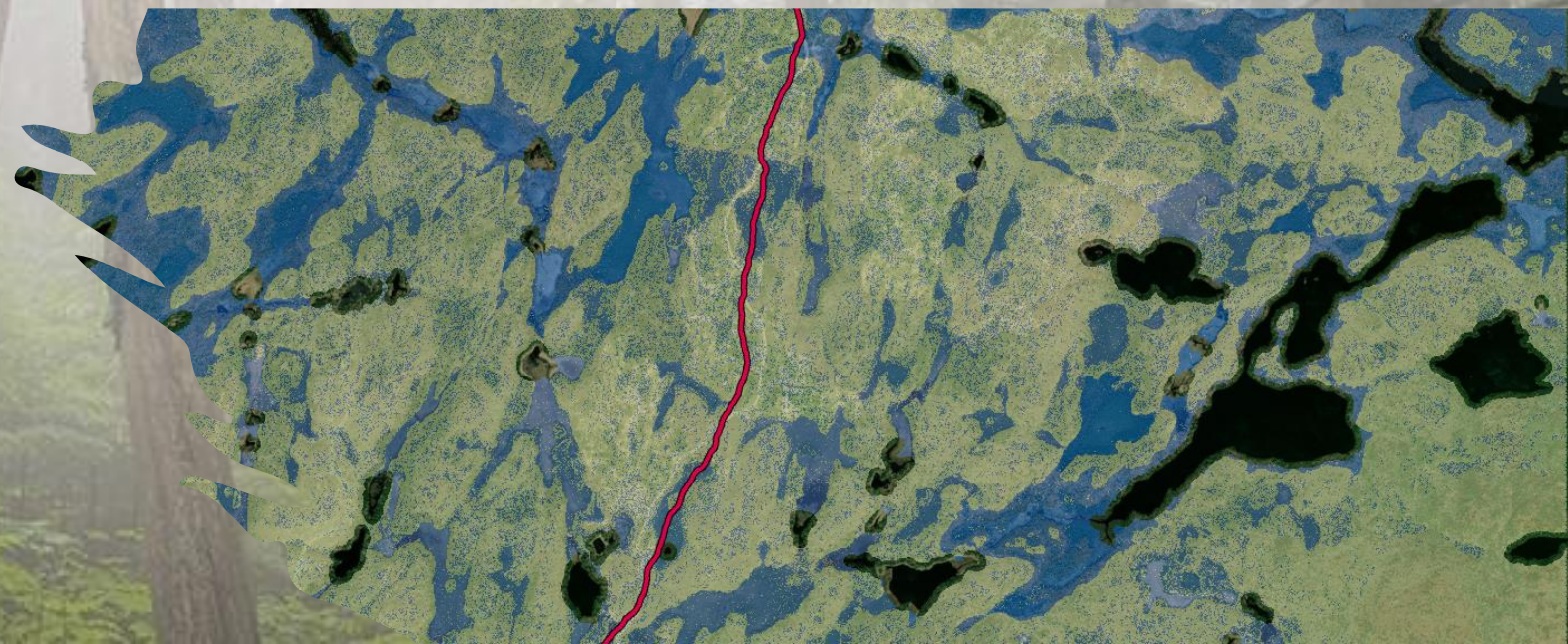
Road Layout Output

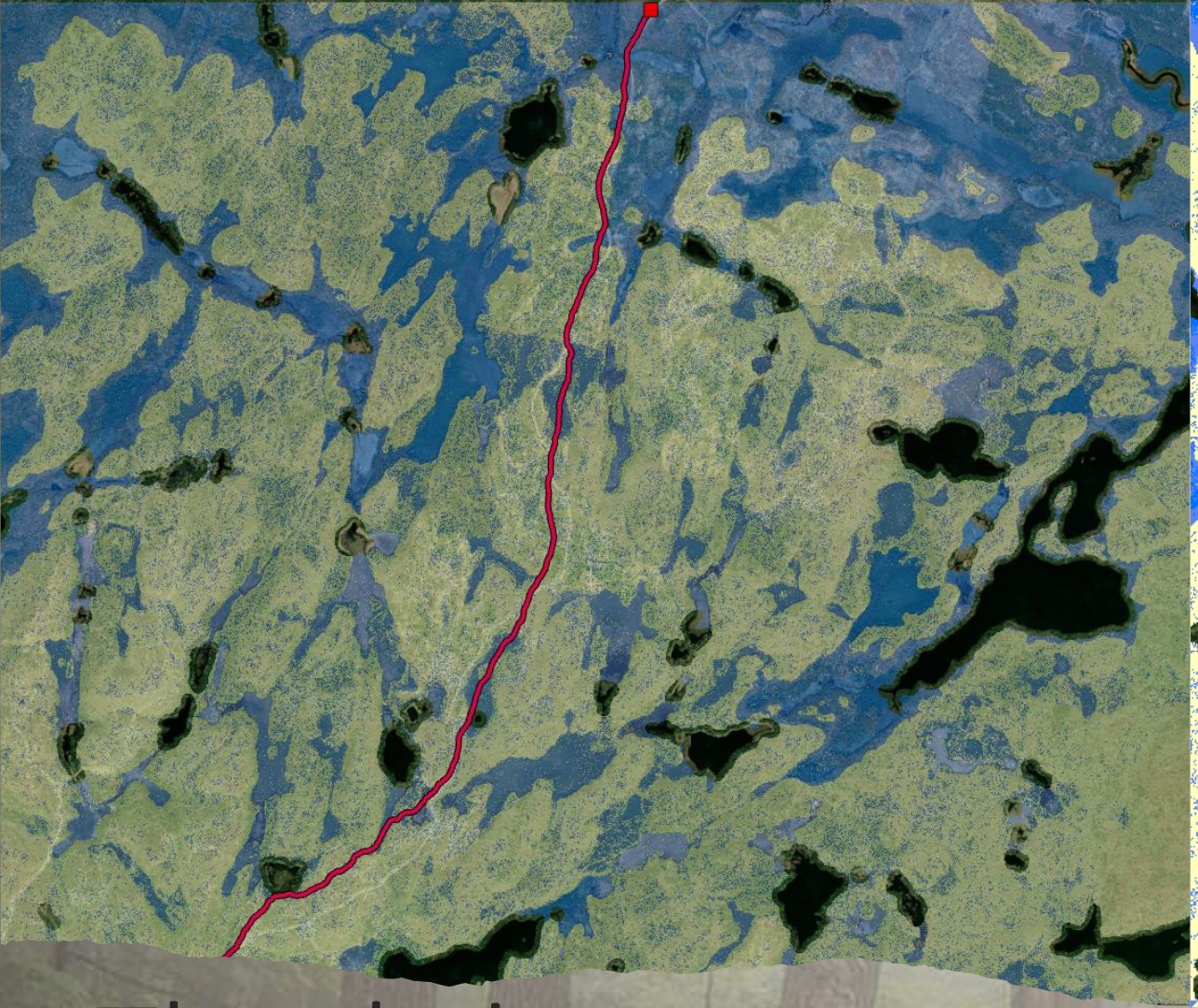
- The process still requires ground truthing, but our tool is designed to save time and energy.

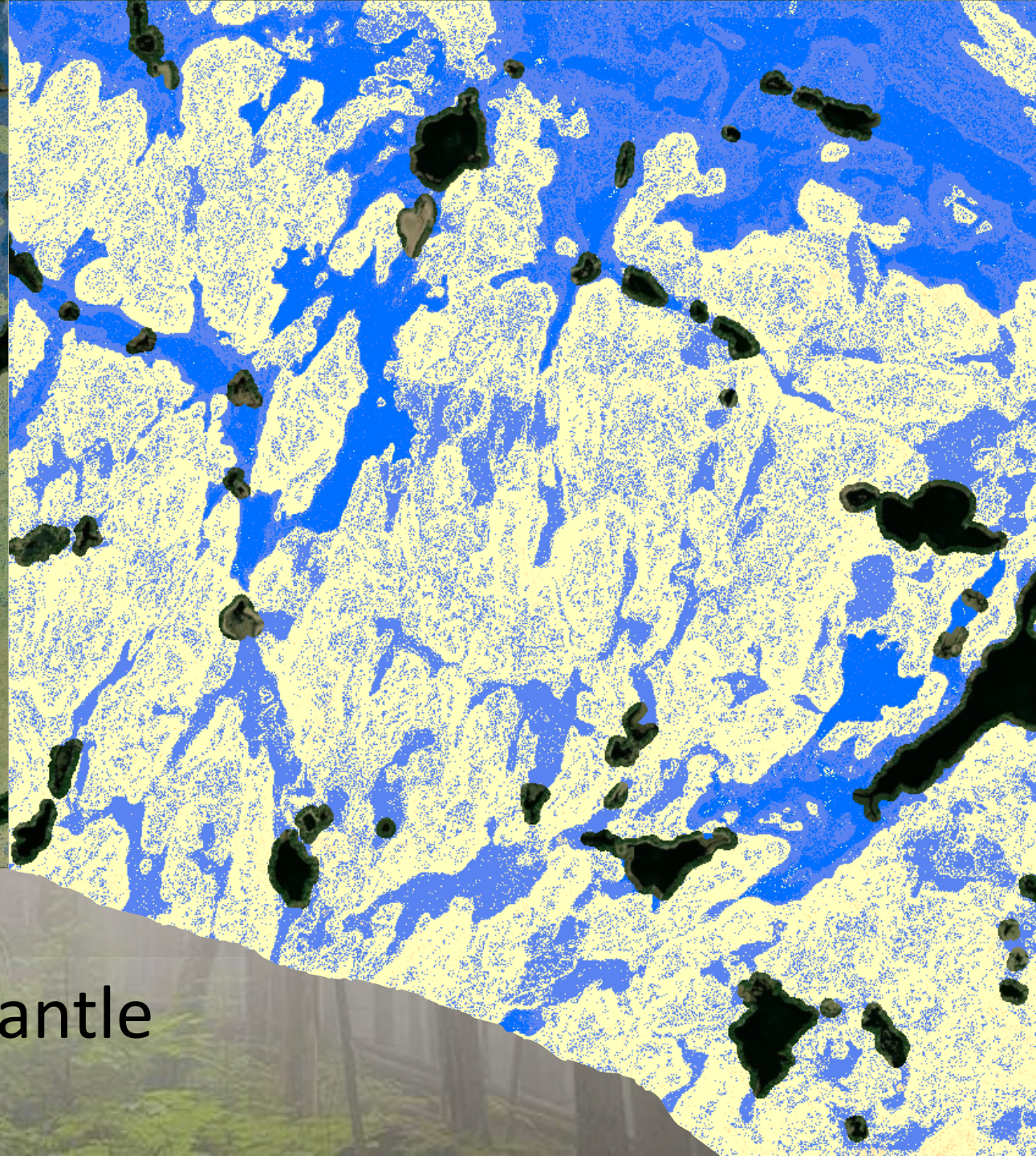


Future Considerations

- Additional parameters could be added to improve functionality and autonomy.



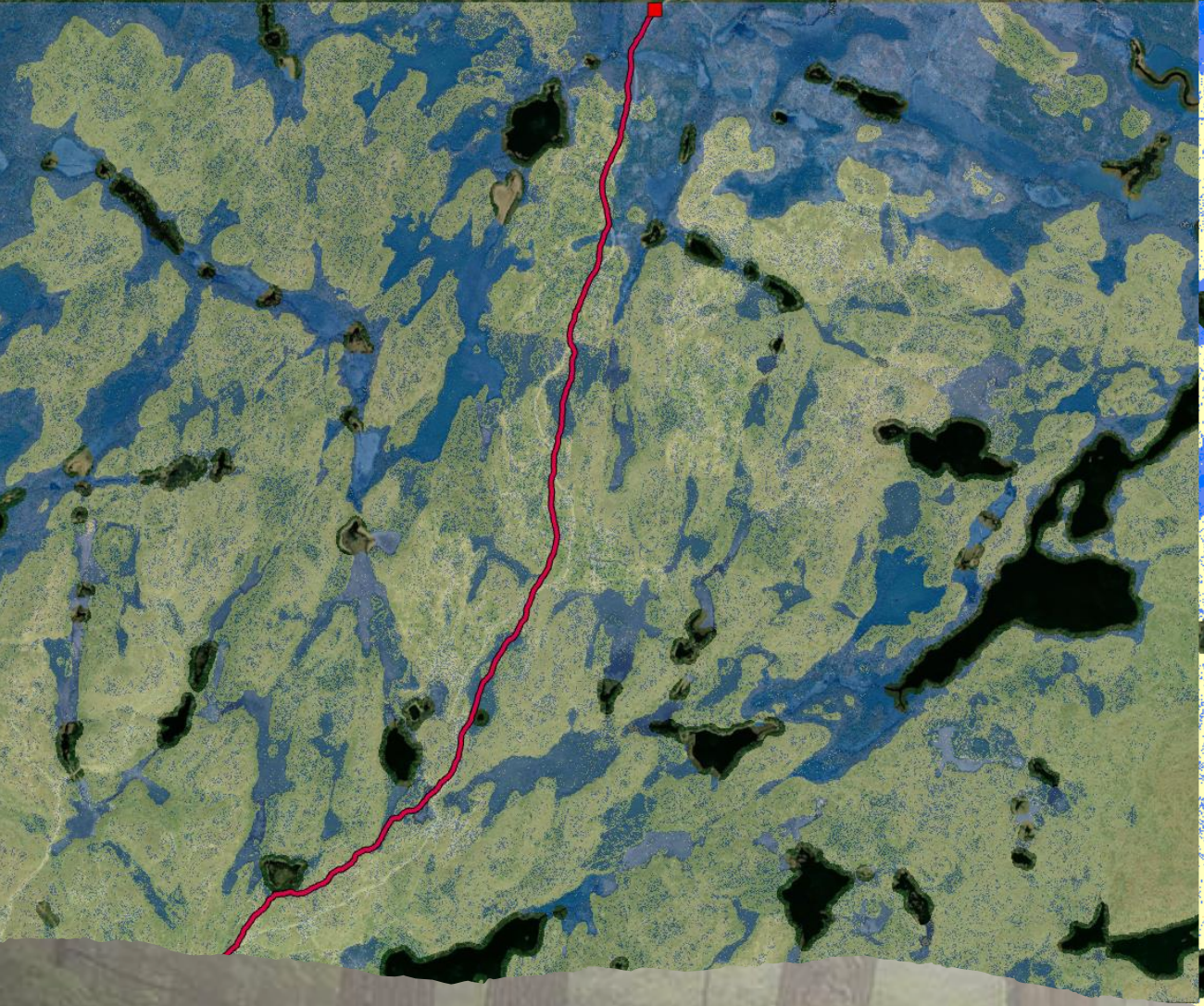


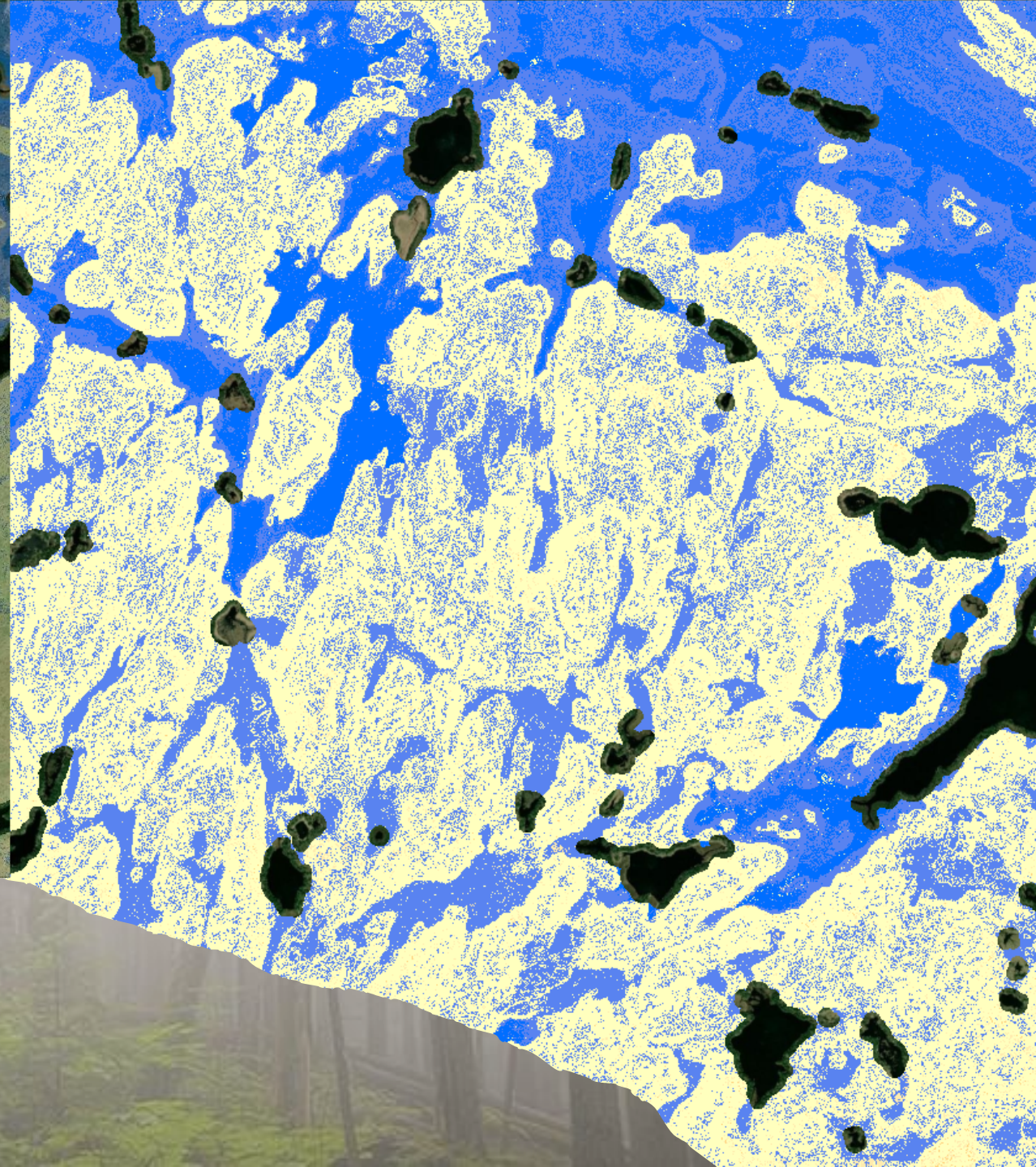


Thanks!

Jennifer Frechette at GreenMantle

Forestry Futures Trust





Questions?