Climate-resilient Reclamation (CRR) Shiny App

Peter Solymos / Analythium Solutions

2025-03-31

Introduction

The Climate-resilient Reclamation (CRR) Shiny App was originally developed by Ivan Bjelanovic to help visualize baseline and future ecosystem states in Alberta developed by the Canadian Forest Service (Natural Resources Canada).

This original version of the app was deployed to the Shinyapps.io platform and referenced raster layers hosted on a 3rd party geospatial server. This server became defunct and as a result the Shiny app stopped working.

The current project's goal is to revitalize the app and find sustainable and easy-to-manage hosting option for the app and raster layers:

1. Update existing app built by Ivan Bjelanovic to current standards and packages (including re-establishing broken links);
2. Add results from new Alberta simulations to be provided by ApexRMS (up to 64 scenarios);
3. Provide commented R code and instructions for updating the app with new layers.

This report outlines how the 3 delivery points were accomplished.

Project organization

The CRR Shiny App project is hosted on GitHb as a public repository: https://github.com/psolymos/crr-shiny. To get the source code, visit the link and click green "Code" button to download a zip file with the code in it. ALternatively, you can use `git` to clone the repository with `git clone https://github.com/psolymos/crr-shiny.git`.

The repository has the following folder:

- `app-v1`: Original (v1) Shiny app with updated references to static files.
- `app-v2`: V2 of the Shiny app with flexibility for adding new layers.
- `data`: Example raster files and R scripts for data processing.
- `docs`: Folder with static files to be served through GitHub Pages.

App deployment

The app needs 2 different deployments:

1. The Shiny app that is responsible for the user interface and the application logic;

2. and the file server that hosts the spatial files that are too big and numerous for being deployed alongside the Shiny app.

Shiny apps can be hosted in many ways. The easiest and free option is to register for a Shinyapps.io account at https://www.shinyapps.io/auth/oauth/signup. The free account's credentials can be added to RStudio Desktop. Then, the app can be deployed from RStudio within a user account.

Legacy app (v1)

The original version of the Shiny app was updated the following ways:

- R package dependencies revised and updated to reflect changes around the geospatial package ecosystem (retiring `sp`, `geos`, etc. in favor of `sf`, `terra` and others);
- Updated links to external file server (see below the *File server* section).

The source code of the updated version of the original app is in the `app-v1` folder of the GitHub repository. The current live deployment of this v1 of the app can be found at: https://analythium.shinyapps.io/crr-shiny-v1/.

Install dependencies (from inside the `app-v1` folder) using the `dependencies.json` file:

```
if (!requireNamespace("deps")) install.packages("deps")
deps::install()
```

To edit the v1 of the app, double click the Rproj file in the `app-v1` folder – this will open the project in RStudio Desktop.

To run the app, click on the "play" button in RStudio, call `shiny::runApp("app-v1")` from R or `R -q -e 'shiny::runApp("app-v1")'` from shell.

Updated Shiny app (v2)

The v1 of the Shiny app has the following updates compared to the v1:

- It is a stripped down version focusing on scenario comparisons;
- It is set up for easier maintenance explained later in this document;
- It has fewer dependencies.

The source code of the updated version of the original app is in the `app-v2` folder of the GitHub repository. The current live deployment of this v2 of the app can be found at: https://analythium.shinyapps.io/crr-shiny-v2/.

Install dependencies (from inside the `app-v2` folder) with:

```
if (!requireNamespace("deps")) install.packages("deps")
deps::install()
```

This command will use the `dependencies.json` file to install missing R packages.

To edit the v1 of the app, double click the Rproj file in the `app-v2` folder – this will open the project in RStudio Desktop.

To run the app, click on the "play" button in RStudio, call `shiny::runApp("app-v2")` from R or `R -q -e 'shiny::runApp("app-v2")'` from shell.

File server

The links in the Shiny apps point to the GitHub Pages based file server that is part of the `docs` folder. GitHub pages are available for free of charge for public GitHub repositories. It needs to be enabled in the Settings/Pages. The repository serves static files from the `docs` folder of the main Git branch. When new files are added to the repository, the files will be published on the `github.io` website.

E.g. for this setup, the file `<github-repository-url>/docs/index.html` will be published at `https://<github-pages-url/index.html`. The file `https://github.com/psolymos/crr-shiny/docs/v1/scenari` will be available at `https://psolymos.github.io/crr-shiny/docs/v1/scenarios/...tif`.

GitHub pages can be accessed from other websites because it allows cross origin resource sharing (via CORS headers). This makes this service ideal for hosting static assets. In our case, GeoTIFF files.

Collaborative code management

The repository is hosted under the `psolymos` personal GitHub account. The repository is public, which means that others can fork the project. Edits to forked projects can be merged to the upstream repository after submitting a pull request. But the changes do not have to be merged for others to develop the project and create independent Shinyapps.io deployments. However, submitting pull requests is good practice.

The owner of the repository (`psolymos`) can grant write access to other GitHub users. If that is desired, please let me know.

The ownership of the repository can also be transferred to another user or GitHub organization.

Adding new layers

The v2 of the Shiny app is developed with ease of maintenance in mind. Here we outline how to process new raster layers and add them to the app and the file server.

Raster layer processing

A new layer is expected to be a GeoTIFF file with a single band, that band should have positive integers as values, or missing. The cartographic reference system needs to be know, otherwise assumed to be Alberta 10-TM (Forest) (EPSG:3400). The raster cell resolution is 500 m by 500 m.

The cell values denote ecotype classes, such as:

Code	Classification	Upland	Color	ClassordNum
1	Poor-Xeric Grassland	1	#FFF0C6	1.0
2	Poor-Xeric Jack Pine	1	#FF9E56	2.0
3	Poor-Mesic Grassland	1	#FFE7A6	3.0
4	Poor-Mesic Pine	1	#FFA627	4.0
5	Poor-Mesic Black Spruce	1	#DBE628	5.0
6	Poor-Hygric Black Spruce	0	#A6AE1F	6.0
7	Poor-Hydric Black Spruce / Larch	0	#666B14	7.0
8	Poor-Hydric Shrub	0	#593E1F	8.0
9	Medium-Xeric Grassland	1	#FFFD77	9.0

Code	Classification	Upland	Color	ClassordNum
10	Medium-Xeric Aspen Mix	1	#FFFB00	10.0
11	Medium-Xeric Pine	1	#7D0C0C	11.0
12	Medium-Xeric Spruce	1	#698222	12.0
13	Medium-Mesic Grassland	1	#D5FF95	13.0
14	Medium-Mesic Aspen	1	#B1FF38	14.0
15	Medium-Mesic Aspen Mix	1	#9EFF0C	15.0
50	Medium-Mesic Aspen Boreal Mixedwood	1	#26FE00	15.5
16	Medium-Mesic Pine	1	#0B7401	16.0
17	Medium-Mesic Pine Mix	1	#0B7401	17.0
18	Medium-Mesic White Spruce	1	#085401	18.0
19	Medium-Hygric Grassland	0	#99FF89	19.0
20	Medium-Hygric Poplar Mix	0	#14C800	20.0
21	Medium-Hygric Spruce Mix	0	#3CBC2F	21.0
22	Medium-Hygric Black Spruce Mix	0	#39AE2C	22.0
23	Alpine	1	#ef476f	23.0
24	Barren	1	#DADAD0	24.0
25	Medium-Hydric Shrub (Poor Fen)	0	#9EB373	25.0
26	Medium-Hydric Black Spruce Fen (Poor Fen)	0	#75AC00	26.0
27	Rich-Mesic Grassland	1	#AFFFA3	27.0
28	Rich-Hygric Shrubland	0	#99FF89	28.0
29	Rich-Hygric Poplar	0	#00C89E	29.0
30	Rich-Hygric Lodgepole Pine	0	#00B991	30.0
31	Rich-Hygric Spruce	0	#0E995F	31.0
32	Rich-Hydric Grass Fen	0	#9FFFD7	32.0
33	Rich-Hydric Shrub Fen	0	#2DFFA7	33.0
34	Rich-Hydric Black Spruce	0	#0C5437	34.0
35	Shrub Swamp	0	#ADFFE7	35.0
36	Treed Swamp	0	#ADFFAF	36.0
37	Very Rich-Hydric Marsh	0	#AEEDFF	37.0
38	Alkali-Hydric	0	#8692FB	38.0
39	Lake	0	#3649F9	39.0
40	River	0	#071BD6	40.0
41	Agriculture	0	#AB9F95	41.0
42	Urban	0	#373737	42.0
43	Other Non-Fuel	0	#ACAC96	43.0

An example input file is `Stralberg_baseline_vegeco4top_mod_int.tif`:

```
library(terra)
r1 <- rast("Stralberg/Stralberg_baseline_vegeco4top_mod_int.tif")
r1
```

```
## class       : SpatRaster
## dimensions  : 2468, 1390, 1  (nrow, ncol, nlyr)
## resolution  : 500, 500  (x, y)
## extent      : 170616.2, 865616.2, 5425532, 6659532  (xmin, xmax, ymin, ymax)
```

```
## coord. ref. : +proj=tmerc +lat_0=0 +lon_0=-115 +k=0.9992 +x_0=500000 +y_0=0 +datum=NAD83 +uni
## source      : Stralberg_baseline_vegeco4top_mod_int.tif
## name        : Stralberg_baseline_vegeco4top_mod_int
## min value   :                                       1
## max value   :                                      50
```

We can set the color table for the ecosite classes as:

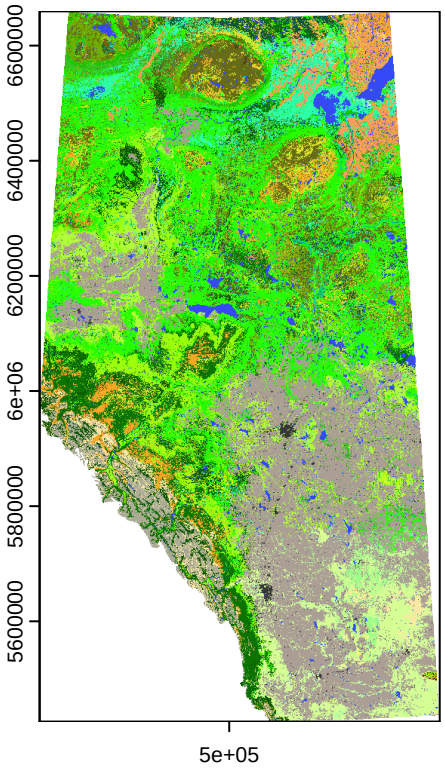
```
es <- read.csv("ecosite_classes.csv")
ct <- data.frame(value=es$Code, col=es$Color)
str(ct)
```

```
## 'data.frame':    44 obs. of  2 variables:
##  $ value: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ col  : chr  "#FFF0C6" "#FF9E56" "#FFE7A6" "#FFA627" ...
```

```
coltab(r1) <- ct
```

We can now plot the raster and get the assigned color mapping:

```
plot(r1)
```



Saving integer raster To save the integer version in a compressed format and without loss of information, use the following command:

```
writeRaster(r1, "file_int.tif", overwrite=TRUE, datatype="INT1U")
```

This will use the default LZW compression (lossless) and unsigned 8-bit integer. This can store values between 0 and 254 (2^8-1, minus one for NA storage), raster values are rounded if not already

integer. This will result in the smallest file size (<1Mb for a 500 m resolution raster for Alberta).

The integer version can be further processed and and summarized.

Save RGB The colorized (RGB: red-green-blue) version is useful to show the cells as the desired color without having to use a color table. This is important because Leaflet maps can't easily display categorical raster layers without extensive JavaScript coding.

The colorized version is also reprojected to web mercator that is the standard for web mapping:

- EPSG:4326 is used for feature geometries (i.e. roads, etc.),
- EPSG:3857 is used for base layers and similar (raster, image) objects.

We will use an existing TIF file as the template that has the desired EPSG3857 projection and matching resolution to our integer map:

```
rt <- rast("Stralberg/Stralberg_baseline_vegeco4top_mod_prj_on.tif")
rt
```

```
## class       : SpatRaster
## dimensions  : 2518, 1621, 4  (nrow, ncol, nlyr)
## resolution  : 861, 863  (x, y)
## extent      : -13463566, -12067885, 6252603, 8425637  (xmin, xmax, ymin, ymax)
## coord. ref. : WGS 84 / Pseudo-Mercator (EPSG:3857)
## source      : Stralberg_baseline_vegeco4top_mod_prj_on.tif
## names       : Stralbe~rj_on_1, Stralbe~rj_on_2, Stralbe~rj_on_3, Stralbe~rj_on_4
```

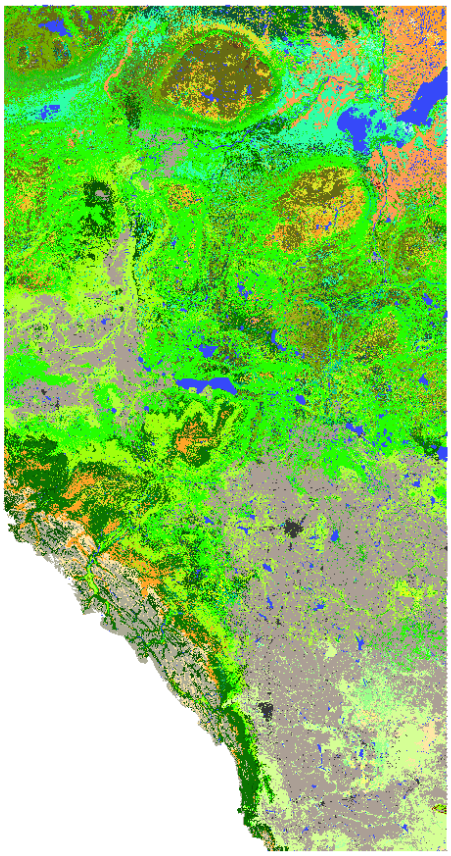
We need to use the nearest neighbor method:

```
r1w <- project(r1, rt, method = "near")
```

Now we can colorize the layer as:

```
r1c <- colorize(r1w, "rgb", alpha=FALSE)
```

```
plot(r1c)
```



Use unsigned 16-bit integer raster format for the RGB image to avoid potential issues with the missing data mask (alpha channel). Use the `DEFLATE` compression method for best results:

```
writeRaster(r1c, "file_rgb.tif", overwrite=TRUE, datatype="INT2U", gdal="COMPRESS=DEFLATE")
```

The output file is usually 1.5-2 times larger than integer version.

ApexRMS provided rasters The raster values provided by ApexRMS (`EcositeStateID` column) for the new scenarios need to be mapped to the ecosite classes using a lookup table:

Classification	EcositeStateID	Code
Poor-Xeric Grassland	1	1
Poor-Xeric Jack Pine	2	2
Poor-Xeric Jack Pine	100	2
Poor-Xeric Jack Pine	105	2
Poor-Xeric Jack Pine	106	2
Poor-Mesic Grassland	5	3
Poor-Mesic Pine	4	4
Poor-Mesic Pine	102	4
Poor-Mesic Pine	113	4
Poor-Mesic Pine	114	4
Poor-Mesic Black Spruce	4	5
Poor-Hygric Black Spruce	6	6

Classification	EcositeStateID	Code
Poor-Hygric Black Spruce	103	6
Poor-Hygric Black Spruce	125	6
Poor-Hygric Black Spruce	126	6
Poor-Hydric Black Spruce / Larch	7	7
Poor-Hydric Black Spruce / Larch	127	7
Poor-Hydric Black Spruce / Larch	128	7
Poor-Hydric Shrub	8	8
Medium-Xeric Grassland	9	9
Medium-Xeric Aspen Mix	10	10
Medium-Xeric Aspen Mix	109	10
Medium-Xeric Aspen Mix	110	10
Medium-Xeric Pine	11	11
Medium-Xeric Pine	101	11
Medium-Xeric Pine	107	11
Medium-Xeric Pine	108	11
Medium-Xeric Spruce	12	12
Medium-Xeric Spruce	111	12
Medium-Xeric Spruce	112	12
Medium-Mesic Grassland	13	13
Medium-Mesic Aspen	17	14
Medium-Mesic Aspen	115	14
Medium-Mesic Aspen	116	14
Medium-Mesic Aspen Mix	15	15
Medium-Mesic Aspen Mix	117	15
Medium-Mesic Aspen Mix	118	15
Medium-Mesic Pine	17	16
Medium-Mesic Pine Mix	17	17
Medium-Mesic White Spruce	18	18
Medium-Mesic White Spruce	104	18
Medium-Mesic White Spruce	119	18
Medium-Mesic White Spruce	120	18
Medium-Hygric Grassland	77	19
Medium-Hygric Poplar Mix	20	20
Medium-Hygric Poplar Mix	121	20
Medium-Hygric Poplar Mix	122	20
Medium-Hygric Spruce Mix	21	21
Medium-Hygric Spruce Mix	123	21
Medium-Hygric Spruce Mix	124	21
Medium-Hygric Black Spruce Mix	6	22
Alpine	43	23
Barren	43	24
Medium-Hydric Shrub (Poor Fen)	25	25
Medium-Hydric Black Spruce Fen (Poor Fen)	26	26
Medium-Hydric Black Spruce Fen (Poor Fen)	129	26
Medium-Hydric Black Spruce Fen (Poor Fen)	130	26
Rich-Mesic Grassland	77	27

Classification	EcositeStateID	Code
Rich-Hygric Shrubland	NA	28
Rich-Hygric Poplar	20	29
Rich-Hygric Lodgepole Pine	34	30
Rich-Hygric Lodgepole Pine	131	30
Rich-Hygric Lodgepole Pine	132	30
Rich-Hygric Spruce	21	31
Rich-Hydric Grass Fen	32	32
Rich-Hydric Shrub Fen	32	33
Rich-Hydric Black Spruce	34	34
Shrub Swamp	35	35
Treed Swamp	NA	36
Very Rich-Hydric Marsh	NA	37
Alkali-Hydric	NA	38
Lake	39	39
River	NA	40
Agriculture	41	41
Urban	42	42
Other Non-Fuel	43	43
Medium-Mesic Aspen Boreal Mixedwood	17	50
Other Non-Fuel	52	43
Other Non-Fuel	53	43
Other Non-Fuel	54	43
Other Non-Fuel	55	43
Other Non-Fuel	56	43
Other Non-Fuel	57	43
Other Non-Fuel	66	43
Other Non-Fuel	99	43

Here is an example raster file:

```
r2 <- rast("STSim_Outputs/sc.it1.ts100.tif")
r2
```

```
## class       : SpatRaster
## dimensions  : 1958, 1306, 1  (nrow, ncol, nlyr)
## resolution  : 500, 500  (x, y)
## extent      : 177116.2, 830116.2, 5680032, 6659032  (xmin, xmax, ymin, ymax)
## coord. ref. : NAD83 / Alberta 10-TM (Forest) (EPSG:3400)
## source      : sc.it1.ts100.tif
## name        : sc.it1.ts100
```

```
summary(r2)
```

```
##   sc.it1.ts100
##  Min.   :  1.00
##  1st Qu.: 15.00
##  Median : 26.00
```

```
##  Mean   : 48.11
##  3rd Qu.:113.00
##  Max.   :132.00
##  NA's   :29223
```

We can apply the ecosite classification as follows:

```
vo <- values(r2) # old values
summary(vo)
```

```
##   sc.it1.ts100
##  Min.   :  1.0
##  1st Qu.: 15.0
##  Median : 26.0
##  Mean   : 48.2
##  3rd Qu.:113.0
##  Max.   :132.0
##  NA's   :743435
```

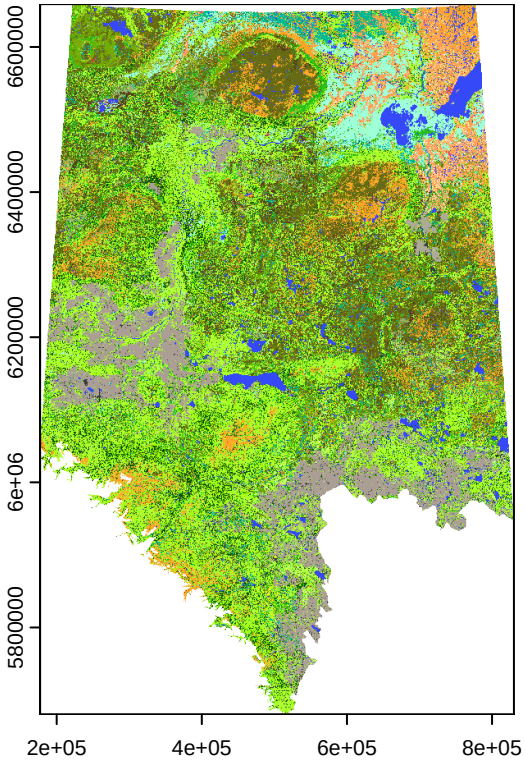
```
vn <- vo          # new values
lt <- read.csv("ecosite_classes_lookup.csv")
vn[,1] <- ifelse(is.na(vo[,1]), NA, lt$Code[match(vo[,1], lt$EcositeStateID)])
summary(vn)
```

```
##   sc.it1.ts100
##  Min.   : 1.0
##  1st Qu.: 8.0
##  Median :15.0
##  Mean   :18.1
##  3rd Qu.:26.0
##  Max.   :43.0
##  NA's   :743435
```

```
r3 <- r2          # create a copy
values(r3) <- vn # set values
```

We can now apply the previous color table to this new raster:

```
coltab(r3) <- ct
plot(r3)
```



Saving the integer version, projecting to web mercator and colorization, finally saving the RGB version is the same as shown before.

Adding the layer to the file server

The layer can now be added to the `docs` folder so it is served via GitHub pages. The `docs` folder has the following high level structure:

- `testing`: folder for testing related files,
- `v1`: files served for the v1 of the Shiny app,
- `v2`: this will be used for organizing future layers for the v2 of the Shiny app.

The `v1` folder currently contains the following files:

```
docs/v1
|-- base
|   |-- BaselineUpland_prj_on.tif
|   |-- DEP_baselayer_on.tif
|   |-- PLC30_OS_100_prj_on.tif
|   +-- Stralberg_baseline_vegeco4top_mod_prj_on.tif
+-- scenarios
    |-- climate-driven
    |   |-- Stralberg_20112040_CSIRO_prj_on.tif
    |   |-- Stralberg_20112040_CanESM2_prj_on.tif
    |   |-- Stralberg_20112040_HadGEM2_prj_on.tif
    |   |-- Stralberg_20412070_CSIRO_prj_on.tif
    |   |-- Stralberg_20412070_CanESM2_prj_on.tif
    |   |-- Stralberg_20412070_HadGEM2_prj_on.tif
```

```
    |   |-- Stralberg_20712100_CSIRO_prj_on.tif
    |   |-- Stralberg_20712100_CanESM2_prj_on.tif
    |   +-- Stralberg_20712100_HadGEM2_prj_on.tif
    |-- fire-mediated-constrained
    |   |-- ab_veg_fm_c_20112040_CSIRO_prj_on.tif
    |   |-- ab_veg_fm_c_20112040_CanESM2_prj_on.tif
    |   |-- ab_veg_fm_c_20112040_HadGEM2_prj_on.tif
    |   |-- ab_veg_fm_c_20412070_CSIRO_prj_on.tif
    |   |-- ab_veg_fm_c_20412070_CanESM2_prj_on.tif
    |   |-- ab_veg_fm_c_20412070_HadGEM2_prj_on.tif
    |   |-- ab_veg_fm_c_20712100_CSIRO_prj_on.tif
    |   |-- ab_veg_fm_c_20712100_CanESM2_prj_on.tif
    |   |-- ab_veg_fm_c_20712100_HadGEM2_prj_on.tif
    +-- fire-mediated-unconstrained
        |-- ab_veg_fm_uc_20112040_CSIRO_prj_on.tif
        |-- ab_veg_fm_uc_20112040_CanESM2_prj_on.tif
        |-- ab_veg_fm_uc_20112040_HadGEM2_prj_on.tif
        |-- ab_veg_fm_uc_20412070_CSIRO_prj_on.tif
        |-- ab_veg_fm_uc_20412070_CanESM2_prj_on.tif
        |-- ab_veg_fm_uc_20412070_HadGEM2_prj_on.tif
        |-- ab_veg_fm_uc_20712100_CSIRO_prj_on.tif
        |-- ab_veg_fm_uc_20712100_CanESM2_prj_on.tif
        +-- ab_veg_fm_uc_20712100_HadGEM2_prj_on.tif
```

The `base` subfolder is for baseline rasters, while the `scenarios` subfolder has the different runs inside.

New GeoTIFF files (EPSG:3857 and RGB format) should be organized within the `v2` folder similarly to how it was done for `v1`. The integer version and other supporting files can also be added as needed.

Adding the new layer to the Shiny app

The Shiny app pulls the information about the file server from a csv file. It is in `app-v2/data/files_list.csv`:

```
files_list <- read.csv("../app-v2/data/files_list.csv")
files_list$time_period <- as.character(files_list$time_period)
head(files_list)
```

```
##                                                                path version
## 1                                  v1/base/BaselineUpland_prj_on.tif      v1
## 2                                       v1/base/DEP_baselayer_on.tif      v1
## 3                                     v1/base/PLC30_OS_100_prj_on.tif      v1
## 4                 v1/base/Stralberg_baseline_vegeco4top_mod_prj_on.tif      v1
## 5 v1/scenarios/climate-driven/Stralberg_20112040_CanESM2_prj_on.tif      v1
## 6   v1/scenarios/climate-driven/Stralberg_20112040_CSIRO_prj_on.tif      v1
##                          dir                                        file
## 1                       base                     BaselineUpland_prj_on.tif
## 2                       base                          DEP_baselayer_on.tif
```

```
## 3                     base                      PLC30_OS_100_prj_on.tif
## 4                     base Stralberg_baseline_vegeco4top_mod_prj_on.tif
## 5 scenarios/climate-driven       Stralberg_20112040_CanESM2_prj_on.tif
## 6 scenarios/climate-driven         Stralberg_20112040_CSIRO_prj_on.tif
##                                file_name file_ext pred_type pred_type_label
## 1                   BaselineUpland_prj_on      tif  baseline            <NA>
## 2                       DEP_baselayer_on      tif  baseline            <NA>
## 3                     PLC30_OS_100_prj_on      tif  baseline            <NA>
## 4 Stralberg_baseline_vegeco4top_mod_prj_on      tif  baseline            <NA>
## 5       Stralberg_20112040_CanESM2_prj_on      tif Stralberg  Climate driven
## 6         Stralberg_20112040_CSIRO_prj_on      tif Stralberg  Climate driven
##   time_period time_period_label climate_model file_type
## 1        <NA>              <NA>          <NA>      mask
## 2        <NA>              <NA>          <NA>       rgb
## 3        <NA>              <NA>          <NA>      mask
## 4        <NA>              <NA>          <NA>       rgb
## 5    20112040         2011-2040       CanESM2       rgb
## 6    20112040         2011-2040         CSIRO       rgb
```

This file captures the important aspects of the baseline and scenario related raster files. The v2 app has functions that can retrieve the following info:

```
Files$get_pred_types()
#   Climate driven   Fire mediated, constrained Fire mediated, unconstrained
#      "Stralberg"                "ab_veg_fm_c"                "ab_veg_fm_uc"

Files$get_time_periods()
#  2011-2040  2041-2070  2071-2100
# "20112040" "20412070" "20712100"

Files$get_climate_models()
#   CanESM2     CSIRO   HadGEM2
# "CanESM2"   "CSIRO" "HadGEM2"
```

These are used to construct the right-hand-side menu in the app.

Inside the Shiny server function, the following expressions return the path required for the GeoTIFF URLs:

```
Files$get_path_for_baseline("BaselineUpland_prj_on")
# [1] "v1/base/BaselineUpland_prj_on.tif"

Files$get_path_for_scenario("Stralberg", "20112040", "CanESM2")
# [1] "v1/scenarios/climate-driven/Stralberg_20112040_CanESM2_prj_on.tif"
```

If the files list CSV file reflects the file server status, the v2 Shiny app will be able to display any new layers.

Note about Leaflet limitations

Leaflet is a web mapping library that is used inside the Shiny app. It uses resampling for displaying rasters at lower (coarser) zoom levels. This behavior cannot be turned off. The benefit of the trade-off is that the GeoTIFF layers can be displayed entirely on the client side. I.e. the Shiny app does not have to send into to the browser. Also, the browser can cache the layers, so loading the 2nd time should take less time.

An alternative option is to load the integer version of the TIF from the file server and display the color mapped or colorized version from within the Shiny R session. This is an extremely resource intensive step and requires patience from the user. Alberta-sized rasters can be handled this way, and the Shiny session has to send the whole raster image to the client-side HTML, so on top of the server side compute requirements, it is also heavy on the client side.