Alex M. Chubaty; Ceres Barros; Ian M. S. Eddy; Eliot J. B. McIntire; David W. Andison

LandWeb Manual

v. 3.0.0

Contents

List of Figures

List of Tables

Overview

This manual is a live document which is automatically updated as changes are made to to underlying model code and documentation.

0.1 Background

0.2 The LandWeb Model

LandWeb is the first large scale, data-driven approach to simulating historic natural range of variation (H/NRV) (https://landweb.ca). In developing the model, analyses, as well as the infrastructure to host data, we strove to implement a single, reproducible workflow to facilitate running simulations, analyses, and model reuse and future expansion. This tight linkage between data and simulation model is made possible via its implementation using the `SpaDES` family of packages [5] within the R Statistical Language and Environment [**R-base**]. For more information about `SpaDES`, see https://spades.predictiveecology.org/.

The LandWeb model integrates two well-used models for forest stand succession and wildfire simulation, implemented in the `SpaDES` simulation platform as a collection of submodels (implement as `SpaDES` modules). Vegetation dynamics are modeled using the LandR Biomass suite of modules, which reimplement the LANDIS-II Biomass Succession model [10, 8] in R. Wildfire dynamics are modeled using an implementation of LandMine [1, 2]. Simulations were run for the entire LandWeb study area, which spans most of the western Canadian boreal forest. A summary of the results are presented using a web app, which can be run locally.

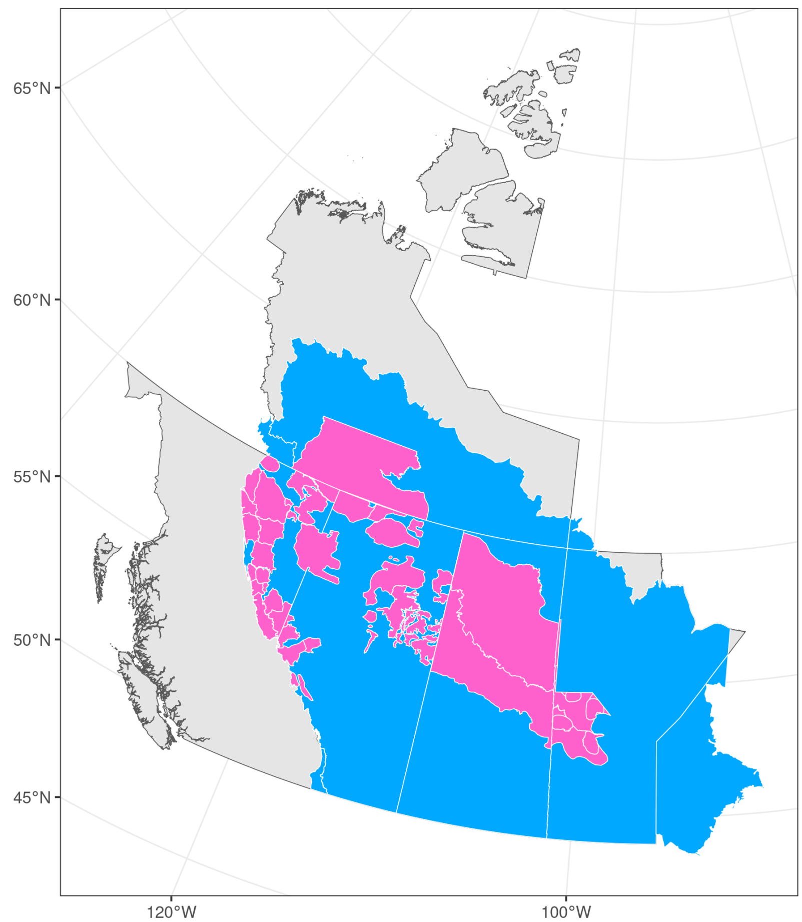

FIGURE 1: LandWeb study area (blue) with mountain and boreal caribou ranges highlighted (pink).

0.2.1 Data preparation

Input data were derived from multiple sources, including several publicly available as well as proprietary datasets.

Detailed descriptions of these sources are provided in the relevant sections of this manual.

0.2.1.1 Public data sources

- Land Cover Classification 2005 map (no longer available from Government of Canada's Open Data website);
- LANDIS-II species traits: https://github.com/dcyr/LANDIS-II_IA_generalUseFiles;
- LANDIS-II parameterization tables and data: https://github.com/LANDIS-II-Foundation/Extensions-Succession-Archive/master/biomass-succession-archive/trunk/tests/v6.0-2.0/;
- Canada biomass, stand volume, and species data [from 3]: http://tree.pfc.forestry.ca;
- National ecodistrict polygons: http://sis.agr.gc.ca/cansis/nsdb/ecostrat/district/ecodistrict_shp.zip;
- National ecoregion polygons: http://sis.agr.gc.ca/cansis/nsdb/ecostrat/region/ecoregion_shp.zip;
- National ecozone polygons: http://sis.agr.gc.ca/cansis/nsdb/ecostrat/zone/ecozone_shp.zip.

0.2.1.2 Proprietary data sources

All proprietary data used by for the model are stored in an access-controlled Google Drive location.

- biomass by species maps created by Pickell & Coops [7] resolution 100m x 100m from LandSat and kNN based on CASFRI;
- various reporting polygons used to summarize model results in the app.

To request access, please contact Alex Chubaty (achubaty@for-cast.ca[1]).

0.2.2 Vegetation dynamics

Vegetation growth and succession are modeled using a re-implementation of the LANDIS-II Biomass model, a widely used and well-documented dynamic vegetation succession model [10, 8]. Our re-implemented model

[1] mailto:achubaty@for-cast.ca

largely follows the original LANDIS-II source code (v 3.6.2; Scheller and Miranda [9]), but with some modifications with respect to species traits parameterization. This model simulates landscape-scale forest dynamics in a spatio-temporally explicit manner, using cohorts of tree species within each pixel. Multiple ecological processes are captured by the model, including vegetation growth, mortality, seed dispersal, and post-disturbance regeneration.

This submodel is described in further detail in Vegetation submodel.

0.2.3 Wildfire dynamics

Wildfire is simulated using a re-implementation of the fire submodel of Andison's [1, 2] `LandMine` model of landscape disturbance.

This submodel is described in further detail in [Wildfire submodel].

0.2.4 Summary maps and statistics

Summaries are derived from simulation outputs, and consist of maps showing the time since fire as well as histogram summaries of 1) number of large patches (i.e., patches above the number of hectares specified by the user) contained within the selected spatial area; and 2) the vegetation cover within the selected spatial area. Histograms are provided for each spatial area by polygon, age class, and species. Authorized users can additionally overlay current stand conditions onto these histograms. Simulation outputs are summarized for several publicly available reporting polygons (including Alberta Natural Ecoregions and Caribou Ranges).

These are described in further detail in [Model outputs].

0.2.5 LandWeb app

Using the web app is described in Web app.

0.3 Previous Manual Versions

If available, archived copies of previous manual versions are provided at the links below.

- LandR Manual v3.0.0[2] (*current*)

[2] archive/pdf/LandWeb-manual-v3.0.0.pdf

1 Getting started

1.1 Prerequisites

Minimum system requirements:

- Windows 10, macOS 10.13 High Sierra, or Ubuntu 20.04 LTS;
- 20 GB of storage space, plus additional storage for model outputs;
- 128 GB RAM to run the model over the full area (less for sub-areas);
- High-speed internet connection.

The following section provides details on installing prerequisite software for running LandWeb.

1.1.1 Docker

If you prefer to not use Docker, skip this subsection.

Due to idiosyncratic difficulties of installing multiple pieces of software and ensuring the correct versions are used throughout, we provide prebuilt Docker (https://www.docker.com/) images, which better provides a consistent and reproducible software environment for running the model.

Thus, using these images are preferred over 'bare-metal' installation.

Install Docker for your system following https://docs.docker.com/get-docker/.

Next, pull the image from Docker Hub:

```
## get the image
docker pull achubaty/landweb-standalone:latest

## launch a new container based on thi image
docker run -d -it \
  -e GITHUB_PAT=$(cat ${HOME}/.Renviron | grep GITHUB_PAT | cut
  -d '=' -f 2) \
  -e PASSWORD='<mySecretPassword>' \
  --memory=128g \
  --cpus=32 \
  -p 127.0.0.1:8080:8787 \
  --name LandWeb \
  achubaty/landweb-standalone:latest
```

Once the container is running, open your web browser and go to `localhost:8080`.

Login to the Rstudio session as user `rstudio` and password `<mySecretPassword>` (change this password when launching container above).

Once finished, you can stop and destroy the container:

```
docker stop LandWeb
docker rm LandWeb
```

1.1.2 Development tools

1.1.2.1 Windows

1. Download `Rtools` version 4.2 from https://cran.r-project.org/bin/windows/Rtools/rtools42/rtools.html and install it *as administrator*. `Rtools` provides the necessary compilers etc. to build and install R packages from source on Windows.
   a. During installation, be sure to check the option to add `Rtools` to your `PATH`.
2. Download and install a proper text editor, *e.g.* `Notepad++` (https://notepad-plus-plus.org/downloads/).

1.1.2.2 macOS

1.1.2.2.1 Xcode command line tools

To build software, you will need the Xcode command line tools[1], which include various compilers and `git` version control software.

```
xcode-select --install
```

1.1.2.2.2 `homebrew` package manager

Next, install `homebrew` which provides a package manager for macOS. This will facilitate software updates and will handle various package dependency issues automatically.

```
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh
)"
```

1.1.2.3 Ubuntu Linux

```
sudo apt-get update

sudo apt-get -y install \
    build-essential \
    biber \
    ccache \
    cmake \
    curl \
    libarchive-dev \
    libcairo2-dev \
    libcurl4-openssl-dev \
```

[1] https://developer.apple.com/downloads/

```
    libgit2-dev \
    libglpk-dev \
    libgmp3-dev \
    libicu-dev \
    libjq-dev \
    libmagick++-dev \
    libnode-dev \
    libpng-dev \
    libprotobuf-dev \
    libprotoc-dev \
    libssh2-1-dev \
    libssl-dev \
    libxml2-dev \
    libxt-dev \
    make \
    p7zip-full p7zip-rar \
    pandoc pandoc-citeproc \
    protobuf-compiler \
    qpdf \
    screen \
    sysstat \
    texinfo texlive-base texlive-bibtex-extra \
    texlive-fonts-extra texlive-latex-extra texlive-xetex \
    wget \
    xauth \
    xfonts-base \
    xvfb \
    zlib1g-dev
```

1.1.3 Geospatial libraries

In order to work with geospatial data, recent versions of `GDAL`, `PROJ`, and `GEOS` geospatial libraries need to be available on your system.

1.1.3.1 Windows

No additional should be needed, as recent versions of R geospatial packages include pre-bundled versions of `GDAL`, `PROJ`, and `GEOS`.

1.1.3.2 macOS

Use `homebrew` to install the required geospatial software libraries:

```
brew install pkg-config
brew install gdal
# brew install geos
# brew install proj
brew install udunits
```

1.1.3.3 Ubuntu Linux

The default Ubuntu 20.04 LTS package repositories ship older versions of the geospatial libraries we will be using, so we will need to to add some additional repositories to get the latest versions.

```
## add GIS repository
sudo add-apt-repository ppa:ubuntugis-unstable/ppa
sudo apt-get update
```

Install additional system dependencies that serve as prerequisites for running the LandWeb model in R.

```
sudo apt-get -y install \
    gdal-bin \
    libgdal-dev \
    libgeos-dev \
    libproj-dev \
    libudunits2-dev \
    python3-gdal
```

Optionally, we install `mapshaper` geospatial library which is used to speed up polygon simplification.

```
## mapshaper installation
sudo apt-get remove -y libnode-dev

curl -sL https://deb.nodesource.com/setup_20.x | sudo -E bash -

sudo apt install nodejs
sudo npm install npm@latest -g
sudo npm install -g mapshaper
```

1.1.4 `git`, Git Kraken, and GitHub

`git` is the version control software used throughout this project, and is required to 'checkout' specific versions of the code as well as to make changes and 'push' these changes to the model code repository.

1. Install the latest version of `git` from https://git-scm.com/downloads or via your package manager.

Windows users should install *as administrator*. Use `nano` (instead of `vi`/`vim`) as the default text editor. For all other choices, use the recommended settings.

For macOS users, `git` is included with the Xcode command line tools.

2. Create a GitHub (https://github.com) account if you don't already have one, and configure a Personal Access Token (PAT).

A GitHub (https://github.com) account is required to assist with package installation and accessing model code.

Several packages used by LandWeb are only available on GitHub. Because we will be installing several of these, we want to ensure we can do so without GitHub rate-limiting our requests. Without a PAT, some packages may *temporarily* fail to install, but can be retried a little later (usually 1 hour).

a. Create a GitHub PAT following the instructions[2];
b. Be sure to uncheck all scopes.
c. Copy this token and save it in a text file in your home directory called `.Renviron`:

   Run the following in an R session to edit `~/.Renviron`:

```
install.packages("usethis")
usethis::edit_r_environ()
```

   Paste the following into the `.Renviron` file, using your copied token:

```
GITHUB_PAT=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

If creating `~/.Renviron` using a text editor, be sure to save it *without* a `.txt` extension. Windows users in particular should ensure they edit this file via Rstudio per above, or use a proper text editor, e.g., `Notepad++` (https://notepad-plus-plus.org/downloads/).

3. *Optional*. Install the latest version of `GitKraken` from https://www.gitkraken.com/download/.

The free version is sufficient to access the public repositories used in this project. However, the paid pro version is required to access private repositories.

1.1.5 R and Rstudio

1. Download and install R version 4.2.3.

Windows

[2] https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token

1. Download R from https://cran.r-project.org/bin/windows/base/R-4.2.3-win.exe;
2. Install R *as administrator*.

macOS

1. Install `rig` (https://github.com/r-lib/rig) to manage multiple R installations.

```
brew tap r-lib/rig
brew install --cask rig

## e.g., M1/M2 mac users, install the arm version
rig install 4.2-arm64

## start Rstudio using a specific R version:
rig rstudio 4.2-arm64
```

Ubuntu Linux

1. Add the CRAN `apt` repository to get the required version of R.

```
## add R repository
sudo sh -c 'echo "deb https://cran.rstudio.com/bin/linux/ubuntu
focal-cran40/" > \
    /etc/apt/sources.list.d/cran.list'
sudo apt-key adv --keyserver keyserver.ubuntu.com \
    --recv-keys E298A3A825C0D65DFD57CBB651716619E084DAB9

sudo apt-get update
```

2. Install R version 4.2.3

To install previous versions of R see https://github.com/achubaty/r-config/blob/master/using-multiple-R-versions-on-linux.Rmd or use `rig` (https://github.com/r-lib/rig).

2. Download and install the latest version of Rstudio from https://www.rstudio.com/products/rstudio/download/.

Windows users should install Rstudio *as administrator*.

3. (optional) On Linux, configure `ccache` to speed up R package re-installation and updates[3].

```
## configure ccache for R package installation
mkdir -p ~/.ccache
mkdir -p ~/.R
{ echo 'VER='; \
  echo 'CCACHE=ccache'; \
  echo 'CC=$(CCACHE) gcc$(VER)'; \
  echo 'CXX=$(CCACHE) g++$(VER)'; \
  echo 'CXX11=$(CCACHE) g++$(VER)'; \
  echo 'CXX14=$(CCACHE) g++$(VER)'; \
  echo 'FC=$(CCACHE) gfortran$(VER)'; \
  echo 'F77=$(CCACHE) gfortran$(VER)'; } >>
  ~/.R/Makevars
{ echo 'max_size = 5.0G'; \
  echo 'sloppiness = include_file_ctime'; \
  echo 'hash_dir = false'; } >> ~/.ccache/ccache.conf
```

1.2 Getting the code

All modules are written in R and all model code was developed collaboratively using GitHub (https://github.com), with each module contained in its own repository. Code that is shared among modules was bundled into R packages, and hosted in on GitHub repositories. All package code is automatically and regularly tested using cross-platform continuous integration frameworks to ensure the code is reliable and free of errors.

[3] http://dirk.eddelbuettel.com/blog/2017/11/27/#011_faster_package_installation_one

```
mkdir -p ~/GitHub
cd ~/GitHub

## get development branch (app and deploy are private
submodules)
git clone --recurse-submodules \
  -j8  https://github.com/PredictiveEcology/LandWeb
```

Windows users should ensure the `GitHub/` directory is accessible at both `~/GitHub` and `~/Documents/GitHub` by creating a directory junction:

```
mklink /J C:\\Users\\username\\Documents\\GitHub
C:\\Users\\username\\GitHub
```

1.3 Project directory structure

Model code is organized by the following directories and summarized in the table below.

NOTE: it may be useful to store data in a different location, but to map this location back to the e.g., `cache/`, `inputs/`, and/or `outputs/` directories using symbolic links. See R's `?file.link` to set these up on your machine.

1.4 Updating the code

After having cloned the LandWeb code repository, users can keep up-to-date using their preferred graphical git tools (*e.g.*, GitKraken) or from the command line.

TABLE 1.1: LandWeb project directory structure

directory	description
R/	additional R helper scripts
cache/	all per-run and per-study area cache files stored here
docker/	Dockerfiles, scripts, and documentation
docs/	rendered model and app documentation
inputs/	all model data inputs stored here
m/	module code (git submodules)
manual/	raw files for generating documentation manual
outputs/	all per-run model outputs stored here
renv/	project package management directory

1.4.1 Using GitKraken

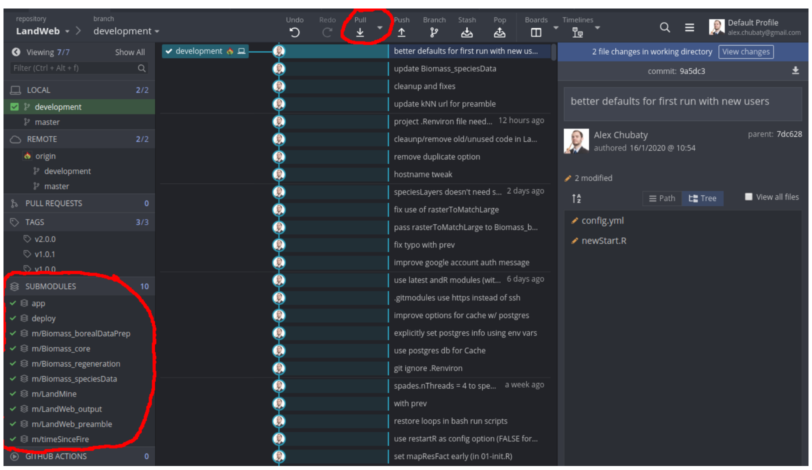

FIGURE 1.1: Screenshot showing showing code commits in Git Kraken. The submodules pane is highlighted on the bottom left.

1. Open the LandWeb repo, and after a few moments you will see the commit history update to reflect the latest changes on the server.
2. 'Pull' in the latest changes to this repo, noting that the status of the git submodules (left hand side) may change.

3. If any submodules have changed status, for each one, right-click and select 'Update '.

1.4.2 Using the command line

WARNING: experienced git users only!

```
git pull
git submodule update
```

1.5 Data requirements

In order to access and use the proprietary data in LandWeb simulations, you will need to be granted access to the shared Google Drive directory. During first-run of the model, all required data will be downloaded to the `inputs/` directory.

To request access, please contact Alex Chubaty (achubaty@for-cast.ca[4]).

1.6 Getting help

- https://github.com/PredictiveEcology/LandWeb/issues

[4] mailto:achubaty@for-cast.ca

2

Running LandWeb

1. Launch Rstudio and open the LandWeb Rstudio project (`LandWeb.Rproj`);
2. Open the file `00-global.R` and run each line in sequence, responding to any prompts as required.

Before you can run the model, you first need to install the packages required for the project by restoring from the project's snapshot file.

```
options(renv.config.mran.enabled = FALSE)
renv::restore()
```

2.1 Model setup and configuration

The default settings for study area, model version, and scenario are defined in `01a-globalvars.R`. These defaults are defined as 'dot-variables' (e.g., `.studyAreaName`) and can be set externally to the main script (e.g., if `.studyAreaName` is defined before running `01a-globalvars.R`, then the user-set value will be used, rather than the default defined in that script).

Advanced setup and model run customization is described in Advanced setup.

2.1.1 Select a study area

The model can be run over the entire study area, for certain individual provinces (currently only AB, SK, MB, NWT), or groups of predefined FMAs (see Fig. 1 and Table 1).

2.1.1.1 FMA boundaries

Currently, only a subset of the FMAs within the LandWeb study area are predefined to be run on their own (i.e., without needing to run the model over the entire study area; see Table 2).

TABLE 2.1: FMA polygon IDs (from Figure 1) and their corresponding FMA names.

ID	Name
1	Cranbrook
2	Fort Nelson
3	Mackenzie
4	Prince George
5	Fort St. John
6	Dawson Creek
7	Island Forests
8	Turtleford
9	Prince Albert
10	Kelvington
11	Island Forests
12	Northern Reconnaisance
13	Turnor East
14	Turnor West
15	Mistik
16	Mee-Toos
17	Kitsaki Zelensky
18	Nemeiben
19	Meadow Lake Fringe
20	Suggi Lowlands
21	Pasquia-Porcupine
22	Meadow Lake OSB
23	L M Wood Products
24	North West
25	Prince Albert FMA
26	Island Forests
27	Island Forests
28	Island Forests
29	Island Forests
30	Spiritwood
31	INTERLAKE
32	Mountain
33	SASKATCHEWAN RIVER

TABLE 2.1: FMA polygon IDs (from Figure 1) and their corresponding FMA names. *(continued)*

ID	Name
34	Cranbrook
35	Cranbrook
36	Cranbrook
37	Cranbrook
38	ALPAC Forest Products Incorporated
39	ANC Timber Ltd.
40	Blue Ridge Lumber Inc.
41	Canadian Forest Products Ltd.
42	Daishowa-Marubeni International Ltd. (East)
43	Daishowa-Marubeni International Ltd. (West)
44	West Fraser Mills Ltd. and Tolko Industries Ltd.
45	Manning Diversified Forest Products Ltd.
46	Millar Western Forest Products Ltd.
47	Spray Lake Sawmills (1980) Ltd.
48	Sundre Forest Products Inc.
49	Tolko Industries Ltd. (High Prairie)
50	Tolko Industries Ltd., Footner Forest Products Ltd. and La Crete Sawmills Ltd.
51	Tolko Industries Ltd., Vanderwell Contractors (1971) Ltd. and West Fraser Mills Ltd. (Slave Lake)
52	Vanderwell Contractors (1971) Ltd.
53	West Fraser Mills Ltd. (Edson)
54	West Fraser Mills Ltd. (Hinton)
55	West Fraser Mills Ltd. (Slave Lake)
56	Weyerhaeuser Company Limited (Grande Prairie)
57	Weyerhaeuser Company Limited (Pembina Timberland)
58	Fort Providence
59	Fort Resolution

2.1.1.2 Choosing a study area

The model can be run on any of several pre-defined study areas summarized in the table below. To select one of these predefined study areas, set `.studyAreaName` to use one of the following, corresponding to the polygon IDs in the map above.

TABLE 2.2: Model study areas with corresponding FMA polygon IDs (from Figure 1).

studyAreaName	ID	Description
ANC	39	ANC Timber Ltd.
Blueridge	40	Blueridge Lumber Inc.
DMI	42, 43	Mercer Peace River Pulp Ltd. (formerly DMI)
Edson	53	West Fraser Mills Ltd. (Edson)
FMANWT	59	Fort Resolution
FMANWT2	58	Fort Providence
LP_BC	5, 6	Lousiana Pacific (British Columbia)
LP_MB	32	Lousiana Pacific (Manitoba)
Manning	45	Manning Diversified Forest Products Ltd.
MillarWestern	46	Millar Western Forest Products Ltd.
Mistik	15	Mistik
MPR	42, 43	Mercer Peace River Pulp Ltd. (formerly DMI)
Sundre	48	Sundre Forest Products Inc.
Tolko_AB_N	50	Tolko Industries Ltd. (Alberta North)
Tolko_AB_S	44, 49, 51	Tolko Industries Ltd. (Alberta South)
Tolko_SK	22	Tolko (Saskatchewan)
Vanderwell	51, 52	Vanderwell Contractors (1971) Ltd.
WestFraser_N	44, 51, 55	West Fraser Mills Ltd. (Slave Lake)
WestFraser_S	53, 54	West Fraser Mills Ltd. (Edson + Hinton)
WeyCo_GP	56	Weyerhauser Company Ltd. (Grand Prairie)
WeyCo_PT	57	Weyerhauser Company Ltd. (Pembina Timberland)
WeyCo_SK	21	Weyerhauser Company Ltd. (Pasquia-Porcupine)

To run LandWeb over an entire province use one of `provAB`, `provMB`, `provNWT`, or `provSK`.

To run the entire LandWeb study area, use `LandWeb`.

2.1.2 Select a scenario

In version 2.0.0 of the LandWeb model, seed dispersal distances needed to be adjusted to ensure sufficient regeneration following fire. These adjustments cause the model to behave more like a state-transition model, rather than a process-based one.

Version 3.0.0 relaxes these parameter forcings to behave like the standard LANDIS-II model.

The dispersal scenario is principally set via `.version` but can be overridden is set via `.dispersalType`:

Dispersal Scenario	Description
`default` (v3)	default LANDIS-II dispersal
`aspen`	limit seed dispersal to deciduous only
`high` (v2)	high seed dispersal of all species (used for v2.0.0 runs)
`none`	no seed dispersal (all species)

Additionally, v2.0.0 of the model uses adjusted fire return intervals (FRI) and log-adjusted rates of spread (ROS). Version 3.0.0 uses the LandMine defaults.

The fire scenario is principally set via `.version` but can be overridden by specifying `.ROStype`:

Fire Scenario	Description
`default` (v3)	default rate of spread values
`burny`	Increases the flammability of non-forest types to facilitate fire spread in landscapes with discontinuous fuels.
`equal`	Set all rates of fire spread equal to each other (no vegetation differences)
`log` (v2)	Reduce the rates of spread but keep the magnitude of vegetation differences

To define a scenario to run, select one dispersal scenario and one fire scenario from the tables above. All LandWeb v2.0.0 runs from 2019 were run using `.dispersalType = "high"` and `.ROStype = "log"`. Using v3.0.0 of the model, `.dispersalType = "default"` and `.ROStype = "default"`.

2.1.3 Replication

To run multiple replicates of a given run, set `.rep` to an integer corresponding to the replicate id. All replicate runs use a different random seed, and this seed is saved as on output for reuse in the event that a replicate needs to be rerun. To rerun a replicate using a different seed, be sure to delete that run's `seed.rds` file. The seed used is also saved in human-readable `seed.txt` file.

2.2 Running the model

NOTE: The first time the model is run, it will automatically download additional data and install additional R packages, which can take some time to complete.

2.2.1 Interactive R session

When working in an R session, be sure to set the working directory to the LandWeb project directory. The first time running the model, open the file `00-global.R`, and step through each line to ensure any prompts etc. are answered correctly.

```
source("00-global.R")
```

When authenticating with Google Drive, be sure to check the box to allow access to files.

2.2.2 Commandline interface

In addition to running the model in an interactive R session, we provide a command line interface to run replicates of the model for the study areas defined above (i.e., batch mode).

For example, to run replicate number 7 of the model at $250m$ resolution using FRI multiple of 1 for Alberta FMU L11, use:

```
cd ~/GitHub/LandWeb

## ./run_fmu.sh <FMU> <FRI> <RES> <REP>
./run_fmu.sh L11 1 250 7
```

FMU command line runs do not use modified dispersal nor fire scenarios, and thus only require the study area, replicate, and FRI multiple to be defined.

To run replicate number 7 of the model for the entire province of Alberta, use:

```
cd ~/GitHub/LandWeb

## ./run_fma.sh <FMU> <REP>
./run_fma.sh provAB 7
# ./run_fma_win.sh provAB 7 ## if on Windows!
```

FMA command line runs use `highDispersal_logROS` scenarios at $250m$ resolution, and thus only require the study area and replicate to be defined.

To run the entire LandWeb study area, only a replicate number needs to be passed. For example:

```
## ./run_landweb.sh <REP>
./run_landweb.sh 7
```

LandWeb command line run set the dispersal and fire scenarios as above, and the pixel resolution (`.pixelSize`) at $250m$, and thus only require the study area and replicate to be defined.

2.3 Post-processing analyses

After having run several reps of the model on a given study area, results are combined in subsequent post-processing analyses to generate the following outputs for each set of reporting polygons within the study area:

- boxplots of leading vegetation cover;
- histograms of leading vegetation cover;
- histograms of large patches.

To run processing, use `.mode = "postprocess"` and be sure to set the number of replicates run in the config.

2.4 Advanced setup

2.4.1 Customizing model run configuration

See SpaDES.config/R/config.landweb.R[1].

(TODO)

2.4.2 Cache backend

Simulation caching is provided by the `reproducible` and `SpaDES.core` packages, and is enabled by default.

The default cache uses a SQLite database backend and stores cache files in `cache/`. However, other database backends can also be used, and advanced users running multiple parallel simulations may wish to set up and use a PostgreSQL database for this cache.

See https://github.com/PredictiveEcology/SpaDES/wiki/Using-alternate-

[1] https://github.com/PredictiveEcology/SpaDES.config/blob/development/R/config.landweb.R

database-backends-for-Cache and ensure the following options are added to your user-specific config in `02a-user-config.R`:

```
reproducible.cacheSaveFormat = "qs",
reproducible.conn = SpaDES.config::dbConnCache("postgresql"),
```

2.4.3 Speeding up disk-based operations

Caching and other disk-based file operations benefit from using an solid state drive (SSD) instead of a conventional spinning hard drive. Advanced users can move their `cache/` directory to an SSD mountpoint and create a symlink to this location in the project directory. Likewise, users can configure a scratch path for temporary raster file operations to point to an SSD location.

2.5 Additional Resources

Resources for (re)learning R and spatial data:

- https://rspatial.org

Study areas

Lorem ipsum ... (TODO)

3

LandR LandWeb_preamble *Module*

3.0.0.1 Authors:

Eliot J B McIntire eliot.mcintire@nrcan-rncan.gc.ca[1] [aut, cre], Alex M. Chubaty achubaty@for-cast.ca[2] [aut], Ceres Barros cbarros@mail.ubc.ca[3] [aut]

3.1 Module Overview

3.1.1 Module summary

Set up study areas and parameters for LandWeb simulations.

3.2 Parameters

Provide a summary of user-visible parameters.

[1] mailto:eliot.mcintire@nrcan-rncan.gc.ca
[2] mailto:achubaty@for-cast.ca
[3] mailto:cbarros@mail.ubc.ca

paramName	paramClass	default	min	max	paramDesc
bufferDist	numeric	25000	20000	1e+05	Study area buffer distance (m) used to make studyArea.
bufferDistLarge	numeric	50000	20000	1e+05	Study area buffer distance (m) used to make studyAreaLarge.
forceResprout	logical	FALSE	NA	NA	‘TRUE‘ forces all species to resprout, setting ‘resproutage_min‘ to zero, ‘resproutage_max‘ to 400, and ‘resproutProb‘ to 1.0.
friMultiple	numeric	1	0.5	2	Multiplication factor for adjusting fire return intervals.
dispersalType	character	default	NA	NA	One of ’aspen’, ’high’, ’none’, or ’default’.
minFRI	numeric	40	0	200	The value of fire return interval below which, pixels will

3.3 Data dependencies

3.3.1 Input data

Description of the module inputs.

objectName	objectClass	desc	sourceURL
canProvs	SpatialPolygonsDataFrame	Canadian provincial boundaries shapefile	NA

3.3.2 Output data

Description of the module outputs.

objectName	objectClass	desc
CC TSF	RasterLayer	Time since fire (aka age) map derived from Current Conditions data.
fireReturnInterval	RasterLayer	fire return interval raster
LandTypeCC	RasterLayer	Land Cover Classification map derived from Current Conditions data.
ml	map	‘map‘ object containing study areas, reporting polygons, etc. for post-processing.
LCC	RasterLayer	A key output from this module: it is the result of LandR::overlayLCCs on LCC2005 and LandTypeCC
nonTreePixels	integer	NA
rasterToMatch	RasterLayer	NA
rasterToMatchLarge	RasterLayer	NA
rasterToMatchReporting	RasterLayer	NA
ROSTable	data.table	A data.table with 3 columns, ’age’, ’leading’, and ’ros’. The values under the ’age’ column can be ’mature’, ’immature’, ’young’ and compound versions of these, e.g., ’immature_young’ which can be used when 2 or more age classes share same ’ros’. ’leading’ should be vegetation type. ’ros’ gives the rate of spread values for each age and type.
rstFlammable	RasterLayer	NA
speciesParams	list	list of updated species trait values to be used to updated ‘speciesTable‘ to create ‘species‘.
speciesTable	data.table	a table of invariant species traits with the

3.4 Links to other modules

Originally developed for use with the LandR Biomass suite of modules, with LandMine fire model. ## References

Vegetation submodel

The LandR ecosystem of SpaDES modules has a variety of data and/or calibration modules that are used to obtain and pre-process input data, as well as estimate input parameters required by the core forest landscape simulation module *Biomass_core*. These modules are presented in the subsequent chapters.

4

LandR Biomass_speciesData *Module*

This documentation is work in progress. Please report any discrepancies or omissions at https://github.com/PredictiveEcology/Biomass_speciesData/issues.

4.0.0.1 Authors:

Eliot J B McIntire eliot.mcintire@nrcan-rncan.gc.ca[1] [aut, cre], Alex M. Chubaty achubaty@for-cast.ca[2] [aut], Ceres Barros cbarros@mail.ubc.ca[3] [aut]

4.1 Module Overview

4.1.1 Module summary

This module downloads and pre-process species % cover data layers to be passed to other LandR data modules (e.g., *Biomass_borealDataPrep*) or to the LandR forest simulation module *Biomass_core*.

4.1.2 Module inputs and parameters at a glance

Below are the full list of input objects (Table 4.1) and parameters (Table 4.2) that *Biomass_speciesData* expects. Of these, the only input that **must** be provided (i.e., *Biomass_speciesData* does not have a default for) is `studyAreaLarge`.

Raw data layers downloaded by the module are saved in `dataPath(sim)`,

[1] mailto:eliot.mcintire@nrcan-rncan.gc.ca
[2] mailto:achubaty@for-cast.ca
[3] mailto:cbarros@mail.ubc.ca

which can be controlled via `options(reproducible.destinationPath = ...)`.

4.1.3 Events

Biomass_speciesData only runs two events:

- Module "initiation" (`init` event), during which all species % cover layers are downloaded and processed.
- Plotting of the processed species cover layers (`initPlot` event).

4.1.4 Module outputs

The module produces the following outputs (Table 4.3):

and automatically saves the processed species cover layers in the output path defined in `getPaths(sim)$outputPath`.

4.1.5 Links to other modules

Intended to be used with other LandR data modules (e.g., *Biomass_borealDataPrep*) that require species cover data and the LandR forest simulation *Biomass_core* module. You can see all *potential* module linkages within the LandR ecosystem here[4]. Select *Biomass_speciesData* from the drop-down menu to see linkages.

4.1.6 Getting help

- https://github.com/PredictiveEcology/Biomass_speciesData/issues

[4] https://rpubs.com/PredictiveEcology/LandR_Module_Ecosystem

TABLE 4.1: List of *Biomass_speciesData* input objects and their description.

objectName	desc
rasterToMatchLarge	a raster of 'studyAreaLarge' in the same resolution and projection the simulation's. Defaults to the using the Canadian Forestry Service, National Forest Inventory, kNN-derived stand biomass map.
rawBiomassMap	total biomass raster layer in study area. Only used to create 'rasterToMatchLarge' if necessary. Defaults to the Canadian Forestry Service, National Forest Inventory, kNN-derived total aboveground biomass map from 2001 (in tonnes/ha), unless 'dataYear' != 2001. See https://open.canada.ca/data/en/dataset/ec9e2659-1c29-4ddb-87a2-6aced147a990 for metadata.
sppColorVect	A named vector of colors to use for plotting. The names must be in sim$sppEquiv[[sim$sppEquivCol]], and should also contain a color for 'Mixed'
sppEquiv	table of species equivalencies. See 'LandR::sppEquivalencies_CA'.
sppNameVector	an optional vector of species names to be pulled from 'sppEquiv'. Species names must match 'P(sim)$sppEquivCol' column in 'sppEquiv'. If not provided, then species will be taken from the entire 'P(sim)$sppEquivCol' column in 'sppEquiv'. See 'LandR::sppEquivalencies_CA'.
studyAreaLarge	Polygon to use as the parametrisation study area. Must be provided by the user. Note that 'studyAreaLarge' is only used for parameter estimation, and can be larger than the actual study area used for LandR simulations (e.g, larger than 'studyArea' in LandR Biomass_core).
studyAreaReporting	multipolygon (typically smaller/unbuffered than 'studyAreaLarge' and 'studyArea' in LandR Biomass_core) to use for

TABLE 4.2: List of *Biomass_speciesData* parameters and their description.

paramName	paramDesc
coverThresh	The minimum % cover a species needs to have (per pixel) in the study area to be considered present
dataYear	Passed to ‘paste0(’prepSpeciesLayers_’, types)‘ function to fetch data from that year (if applicable). Defaults to 2001 as the default kNN year.
sppEquivCol	The column in ‘sim$sppEquiv‘ data.table to group species by and use as a naming convention. If different species in, e.g., the kNN data have the same name in the chosen column, their data are merged into one species by summing their % cover in each raster cell.
types	The possible data sources. These must correspond to a function named paste0(’prepSpeciesLayers_’, types). Defaults to ’KNN’ to get the Canadian Forestry Service, National Forest Inventory, kNN-derived species cover maps from year ’dataYear’, using the ‘LandR::prepSpeciesLayers_KNN‘ function (see https://open.canada.ca/data/en/dataset/ec9e2659-1c29-4ddb-87a2-6aced147a990 for details on these data). Other currently available options are ’ONFRI’, ’CASFRI’, ’Pickell’ and ’ForestInventory’, which attempt to get proprietary data - the user must be granted access first. A custom function can be used to retrieve any data, just as long as it is accessible by the module (e.g., in the global environment) and is named as paste0(’prepSpeciesLayers_’, types).
vegLeadingProportion	a number that defines whether a species is leading for a given pixel. Only used for plotting.
.plotInitialTime	This describes the simulation time

TABLE 4.3: List of *Biomass_speciesData* output objects and their description.

objectName	desc
speciesLayers	biomass percentage raster layers by species in Canada species map
treed	Table with one logical column for each species, indicating whether there were non-zero cover values in each pixel.
numTreed	a named vector with number of pixels with non-zero cover values for each species
nonZeroCover	A single value indicating how many pixels have non-zero cover

4.2 Module manual

4.2.1 Detailed description

This module accesses and processes species percent cover (% cover) data for the parametrisation and initialization of LandR *Biomass_core*. This module ensures 1) all data use the same geospatial geometries and 2) that these are correctly re-projected to studyAreaLarge, and 3) attempts to sequentially fill-in and replace the lowest quality data with higher quality data when several data sources are used. It's primary output is a `RasterStack` of species % cover, with each layer corresponding to a species.

Currently, the module can access the Canadian Forest Inventory forest attributes kNN dataset [the default; Beaudoin et al. [4]], the Common Attribute Schema for Forest Resource Inventories [CASFRI; Cosco [6]] dataset, the Ontario Forest Resource Inventory (ONFRI), a dataset specific to Alberta compiled by Paul Pickell, and other Alberta forest inventory datasets. However, **only the NFI kNN data are freely available** – access to the other datasets must be granted by module developers and data owners, and a Google account is required. Nevertheless, the module is flexible enough that any user can use it to process additional datasets, provided that an adequate R function is passed to the module (see `types` parameter details in Parameters)

When multiple data sources are used, the module will use replace lower quality data with higher quality data following the order specified by the parameter `types` (see Parameters).

When multiple species of a given data source are to be grouped, % cover is summed across species of the same group within each pixel. Please see the `sppEquiv` input in Input objects for information on how species groups are defined.

The module can also exclude species % cover layers if they don't have a minimum % cover value in at least one pixel. This means that the user should still inspect in how many pixels the species is deemed present, as it is possible that some data have only a few pixels with high % cover for a given species. In this case, the user may choose to exclude these species *a posteriori*. The summary plot automatically shown by *Biomass_speciesData* can help diagnose whether certain species are present in very few pixels (see Fig. 4.1).

4.2.2 Initialization, inputs and parameters

Biomass_speciesData initializes itself and prepares all inputs provided that it has internet access to download the raw data layers (or that these layers have been previously downloaded and stored in the folder specified by `options("reproducible.destinationPath")`).

The module defaults to processing cover data fo all species listed in the `Boreal` column of the default `sppEquiv` input `data.table` object, for which there are available % cover layers in the kNN dataset (Table 4.4; see `?LandR::sppEquivalencies_CA` for more information):

4.2.2.1 Input objects

Biomass_speciesData requires the following input data layers

Of the inputs in Table 4.5, the following are particularly important and deserve special attention:

- `studyAreaLarge` – the polygon defining the area for which species cover data area desired. It can be larger (but never smaller) that the study area used in the simulation of forest dynamics (i.e., `studyArea` object in *Biomass_core*).
- `sppEquiv` – a table of correspondences between different species naming conventions. This table is used across several LandR modules, including

TABLE 4.4: List of species cover data downloaded by default by *Biomass_speciesData*.

Species	Generic name
Abies balsamea	Balsam Fir
Abies lasiocarpa	Fir
Acer negundo	Boxelder maple
Acer pensylvanicum	Striped maple
Acer saccharinum	Silver maple
Acer saccharum	Sugar maple
Acer spicatum	Mountain maple
Acer spp.	Maple
Alnus spp	Alder
Betula alleghaniensis	Swamp birch
Betula papyrifera	Paper birch
Betula populifolia	Gray birch
Betula spp.	Birch
Fagus grandifolia	American beech
Fraxinus americana	American ash
Fraxinus nigra	Black ash
Fraxinus spp.	Ash
Larix laricina	Tamarack
Larix lyallii	Alpine larch
Larix occidentalis	Western larch
Larix spp.	Larch
Picea engelmannii x glauca	Engelmann's spruce
Picea engelmannii x glauca	Engelmann's spruce
Picea engelmannii	Engelmann's spruce
Picea glauca	White.Spruce
Picea mariana	Black.Spruce
Picea spp.	Spruce
Pinus albicaulis	Whitebark pine
Pinus banksiana	Jack pine
Pinus contorta	Lodgepole pine
Pinus monticola	Western white pine
Pinus resinosa	Red pine
Pinus spp.	Pine
Populus balsamifera	Balsam poplar
Populus balsamifera v. balsamifera	Balsam poplar
Populus trichocarpa	Black cottonwood
Populus grandidentata	White poplar
Populus spp.	Poplar
Populus tremuloides	Trembling poplar
Tsuga canadensis	Eastern hemlock
Tsuga spp.	Hemlock

TABLE 4.5: List of *Biomass_speciesData* input objects and their description.

objectName	objectClass	desc	sourceURL
rasterToMatchLarge	RasterLayer	a raster of ‘studyAreaLarge‘ in the same resolution and projection the simulation’s. Defaults to the using the Canadian Forestry Service, National Forest Inventory, kNN-derived stand biomass map.	
rawBiomassMap	RasterLayer	total biomass raster layer in study area. Only used to create ‘rasterToMatch-Large‘ if necessary. Defaults to the Canadian Forestry Service, National Forest Inventory, kNN-derived total aboveground biomass map from 2001 (in tonnes/ha), unless ’dataYear’ != 2001. See https://open.canada.ca/data/en/dataset/ec9e2659-1c29-4ddb-87a2-6aced147a990 for metadata.	
sppColorVect	character	A named vector of colors to use for plotting. The names must be in	NA

TABLE 4.6: Example of species merging for simulation. Here the user wants to model *Abies balsamea*, *A. lasiocarpa* and *Pinus contorta* as separate species, but all *Picea spp.* as a genus-level group. For this, all six species are identified in the ́KNNćolumn, so that their % cover layers can be obtained, but in the ́Borealćolumn (which defines the naming convention used in the simulation in this example) all *Picea spp.* have the same name. *Biomass_speciesData* will merge their % cover data into a single layer by summing their cover per pixel.

Species	KNN	Boreal	Modelled as
Abies balsamea	Abie_Bal	Abie_Bal	*Abies balsamea*
Abies lasiocarpa	Abie_Las	Abie_Las	*Abies lasiocarpa*
Picea engelmannii x glauca	Pice_Eng_Gla	Pice_Spp	*Picea spp.*
Picea engelmannii x glauca	Pice_Eng_Gla	Pice_Spp	*Picea spp.*
Picea engelmannii	Pice_Eng	Pice_Spp	*Picea spp.*
Picea glauca	Pice_Gla	Pice_Spp	*Picea spp.*
Picea mariana	Pice_Mar	Pice_Spp	*Picea spp.*
Pinus contorta	Pinu_Con	Pinu_Con	*Pinus contorta*

Biomass_core. It is particularly important here because it will determine whether and how species (and their cover layers) are merged, if this is desired by the user. For instance, if the user wishes to simulate a generic *Picea spp.* that includes, *Picea glauca*, *Picea mariana* and *Picea engelmannii*, they will need to provide these three species names in the data column (e.g., `KNN` if obtaining forest attribute kNN data layers from the Canadian Forest Inventory), but the same name (e.g., "Pice_Spp") in the coumn chosen for the naming convention used throughout the simulation (the `sppEquivCol` parameter); see Table 4.6 for an example).

4.2.2.2 Parameters

Table 4.7 lists all parameters used in *Biomass_speciesData* and their detailed information.

TABLE 4.7: List of *Biomass_speciesData* parameters and their description.

paramName	paramClass	default	min	max	paramDesc
coverThresh	integer	10	NA	NA	The minimum % cover a species needs to have (per pixel) in the study area to be considered present
dataYear	numeric	2001	NA	NA	Passed to 'paste0('prepSpeciesLayers_', types)' function to fetch data from that year (if applicable). Defaults to 2001 as the default kNN year.
sppEquivCol	character	Boreal	NA	NA	The column in 'sim$sppEquiv' data.table to group species by and use as a naming convention. If different species in, e.g., the kNN data have the same name in the chosen column,

Of the parameters listed in Table 4.7, the following are particularly important:

- `coverThresh` – integer. Defines a minimum % cover value (from 0-100) that the species must have in at least one pixel to be considered present in the study area, otherwise it is excluded from the final stack of species layers. Note that this will affect what species have data for an eventual simulation and the user will need to adjust simulation parameters (e.g., species in trait tables will need to match the species in the cover layers) accordingly.
- `types` – character. Which % cover data sources are to be used (see Detailed description). Several data sources can be passed, in which case the module will overlay the lower quality layers with higher quality ones **following the order of data sources specified by `types`** – i.e., if `types == c("KNN", "CASFRI", "ForestInventory")`, *KNN* is assumed to be the lowest quality data set and *ForestInventory* the highest: values in *KNN* layers are replaced with overlapping values from *CASFRI* layers and values from *KNN* and *CASFRI* layers are replaced with overlapping values of *ForestInventory* layers.

4.2.3 Simulation flow

The general flow of *Biomass_speciesData* processes is:

1. Download (if necessary) of and spatial processing of species cover layers from the first data source listed in the `types` parameter. Spatial processing consists in sub-setting the data to the area defined by `studyAreaLarge` and ensuring that the spatial projection and resolution match those of `rasterToMatchLarge`. After spatial processing, species layers that have no pixels with values >= to the `coverThresh` parameter are excluded.
2. If more than one data source is listed in `types`, the second set of species cover layers is downloaded and processed as above.
3. The second set of layers is assumed to be the highest quality dataset and used to replaced overlapping pixel values on the first (including for species whose layers may have been initially excluded after applying the `coverThresh` filter).
4. Steps 2 and 3 are repeated for remaining data sources listed in `types`.

5. Final layers are saved to disk and plotted. A summary of number of pixels with forest cover are calculated (`treed`and `numTreed` output objects; see Module outputs).

4.3 Usage example

4.3.1 Load `SpaDES` and other packages.

```
if (!require(Require)) {
    install.packages("Require")
    library(Require)
}

Require(c("PredictiveEcology/SpaDES.install", "SpaDES",
"PredictiveEcology/SpaDES.core@development",
    "PredictiveEcology/LandR"), install_githubArgs =
    list(dependencies = TRUE))
```

4.3.2 Get module, necessary packages and set up folder directories

```
tempDir <- tempdir()
paths <- list(inputPath = normPath(file.path(tempDir,
"inputs")),
    cachePath = normPath(file.path(tempDir, "cache")),
    modulePath = normPath(file.path(tempDir,
        "modules")), outputPath = normPath(file.path(tempDir,
        "outputs")))

getModule("PredictiveEcology/Biomass_speciesData", modulePath =
paths$modulePath,
    overwrite = TRUE)
```

```
## make sure all necessary packages are installed:
makeSureAllPackagesInstalled(paths$modulePath)
```

4.3.3 Setup simulation

For this demonstration we are using all default parameter values, except `coverThresh` , which is lowered to 5%. The species layers (the major output of interest) are saved automatically, so there is no need to tell `spades` what to save using the `outputs` argument (see `?SpaDES.core::outputs`).

We pass the global parameter `.plotInitialTime = 1` in the `simInitAndSpades` function to activate plotting.

```
# User may want to set some options -- see
# ?reproducibleOptions -- e.g., often the path to the
# 'inputs' folder will be set outside of project by user:
# options(reproducible.inputPaths =
# 'E:/Data/LandR_related/') # to re-use datasets across
# projects
studyAreaLarge <- Cache(randomStudyArea, size = 1e+07, cacheRepo
= paths$cachePath)  # cache this so it creates a random one only
once on a machine

# Pick the species you want to work with -- here we use the
# naming convention in 'Boreal' column of
# LandR::sppEquivalencies_CA (default)
speciesNameConvention <- "Boreal"
speciesToUse <- c("Pice_Gla", "Popu_Tre", "Pinu_Con")

sppEquiv <-
LandR::sppEquivalencies_CA[get(speciesNameConvention) %in%
    speciesToUse]
# Assign a colour convention for graphics for each species
sppColorVect <- LandR::sppColors(sppEquiv,
speciesNameConvention,
    newVals = "Mixed", palette = "Set1")
```

```
## Usage example
modules <- list("Biomass_speciesData")
objects <- list(studyAreaLarge = studyAreaLarge, sppEquiv =
sppEquiv,
    sppColorVect = sppColorVect)
params <- list(Biomass_speciesData = list(coverThresh = 5L))
```

4.3.4 Run module

Note that because this is a data module (i.e., only attempts to prepare data for the simulation) we are not iterating it and so both the start and end times are set to 1 here.

```
opts <- options(reproducible.useCache = TRUE,
reproducible.inputPaths = paths$inputPath)

mySimOut <- simInitAndSpades(times = list(start = 1, end = 1),
    modules = modules, parameters = params, objects = objects,
    paths = paths, .plotInitialTime = 1)
options(opts)
```

Here are some of outputs of *Biomass_speciesData* (dominant species) in a randomly generated study area within Canada.

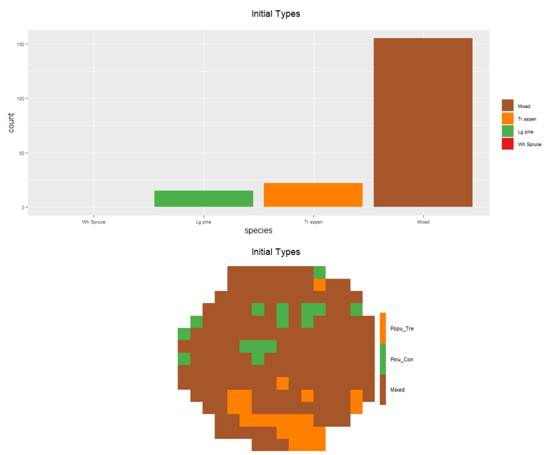

FIGURE 4.1: *Biomass_speciesData* automatically generates a plot of species dominance and number of presences in the study area when '.plotInitialTime=1' is passed as an argument.

4.4 References

[4] A Beaudoin et al. *Species composition, forest properties and land cover types across Canada's forests at 250m resolution for 2001 and 2011*. 2017. DOI: 10.23687/EC9E2659-1C29-4DDB-87A2-6ACED147A990[5]. URL: http://open.canada.ca/data/en/dataset/ec9e2659-1c29-4ddb-87a2-6aced147a990.

[6] John Cosco. *COMMON ATTRIBUTE SCHEMA (CAS) FOR FOREST INVENTORIES ACROSS CANADA*. Feb. 2011, p. 117.

[5] https://doi.org/10.23687/EC9E2659-1C29-4DDB-87A2-6ACED147A990

Web app

As of spring 2023, fRI Research no longer hosts the LandWeb App. However, with appropriate access to the output data, the app may be run in a local `shiny` instance.

Previously available from https://landweb.ca.

5

Modifying LandWeb

5.1 Example 1: adding new reporting polygons

'Reporting polygons' refer to the polygons by which LandWeb results are summarized within the study area. These are created in the `LandWeb_preamble` module, and added to `sim$ml` (an object of class `map`, from the `map` package). Reporting polygons are not used during simulation, but rather during post-processing in the `LandWeb_summaries` module.

Adding a new reporting polygons is straightforward - one simply needs to edit the `LandWeb_preamble` code to 1) create the object and 2) add it to the `map` (*i.e.*, `sim$ml`) object. For reporting polygons retrieved from public or Google Drive URLs, both steps may be combined in a single `mapAdd()` call.

Downstream use requires each set of reporting polygons to have a `Name` (and `shinyLabel`) field (column). These fields are typically derived from a single field in the raw source, and are thus identical in value. Additionally, the helper function `joinreportingPolygons()` should be used following a geospatial intersection operation (*e.g.*, using `postProcess()`) to ensure that post-intersection labels like `Names.1` and `shinyLabel.1` are corrected.

5.1.1 National-scale polygons

When adding reporting polygons derived from *national-scale* data sources, the changes need to be made in `LandWeb_preamble.R` *as well as* each of the study-area-specific files in `LandWeb_preamble/R/` so that LandWeb runs using any study area can make use the new reporting polygons.

```
## LandWeb_preamble.R
```

```
## create the national-scale object and add to map object in a
single call
ml <- mapAdd(map = ml, layerName = "National Ecoregions",
             useSAcrs = TRUE, poly = TRUE, overwrite = TRUE,
             url =
             "https://sis.agr.gc.ca/cansis/nsdb/ecostrat/region/ecoregion_shp.zip"
             ,
             columnNameForLabels = "REGION_NAM", isStudyArea =
             FALSE, filename2 = NULL)

## ensure there is a 'Name' field in the new object
ml[["National Ecoregions"]][["Name"]] <- ml[["National
Ecoregions"]][["REGION_NAM"]]
```

Next, we need to modify each of the regional files to make use of these new national polygons, but intersected and cropped to the study area:

```
## LandWeb_preamble/R/provSK.R

## 1. create the reporting polygons by intesecting the National
Ecoregions with SK admin boundary
SK.natler <- postProcess(ml[["National Ecoregions"]],
                         studyArea = SK, useSAcrs = TRUE,
                         filename2 = file.path(dataDir,
                         "SK_NATLER.shp")) |>
  joinReportingPolygons(SK)


## 2. add to the map object
ml <- mapAdd(SK.natler, ml, layerName = "SK NATLER", useSAcrs =
TRUE, poly = TRUE,
             analysisGroupReportingPolygon = "SK NATLER",
             columnNameForLabels = "REGION_NAM", filename2 =
             NULL)
```

5.1.2 Regional or study-area-specific polygons

If adding regional or study-area-specific reporting polygons (*e.g.*, for an FMA's active/passive landbase), then the new reporting polygons should be added to the corresponding study area code file found in `LandWeb_preamble/R`.

```
## LandWeb_preamble/R/WestFraser.R

## 1. create the reporting polygons by downloading and cleaning
up shapefile
wf_br.lbstatus <- Cache(
  prepInputs,
  url = "https://drive.google.com/file/d/1A7N_EIbO2wMBI_YTmU2Z-
  bQwqC9sY_EC/",
  destinationPath = dataDir,
  targetFile = "BRL_Landbase.shp", alsoExtract = "similar",
  fun = "sf::st_read", studyArea = wf_br, useSAcrs = TRUE
)
wf_br.lbstatus <- wf_br.lbstatus[st_is_valid(wf_br.lbstatus), ]
## remove invalid geometries
wf_br.lbstatus <- wf_br.lbstatus[!st_is_empty(wf_br.lbstatus),
] ## remove empty polygons
wf_br.lbstatus <- Cache({
  mutate(wf_br.lbstatus, Name = LBC_LBStat, geometry = geometry,
  .keep = "used") |>
    group_by(Name) |>
    summarise(geometry = sf::st_union(geometry)) |>
    ungroup() |>
    mutate(shinyLabel = Name, .before = geometry) |>
    joinReportingPolygons(wf_br)
})

## 2. add to the map object
ml <- mapAdd(wf_br.lbstatus, ml, layerName = "West Fraser Blue
Ridge LBstatus", useSAcrs = TRUE, poly = TRUE,
             analysisGroupReportingPolygon = "West Fraser Blue
             Ridge LBstatus",
             columnNameForLabels = "Name", filename2 = NULL)
```

5.2 Example 2: adding a new study area

Here we will create a custom study area for an area in northwestern Alberta. We want to use the latest FMU map from Spring 2022 (https://www.alberta.ca/forest-management-agreements).

1. **Externally from LandWeb**, create a shapefile (or similar) for the new study area, dissolving any internal polygon boundaries.

```
nwab <- sf::st_read("AB_FMAs_2022-03-25/Forest
Management Unit.shp") |>
  subset(FMU_CODE %in% c("F10", "F11", "F14", "F15",
  "F20", "F23", "F26", "F51", "M01",
                         "P08", "P14", "P19", "P20",
                         "P21", "P22", "P52", "P55",
                         "S10", "S11", "S14")) |>
  sf::st_union()
```

2. **Externally from LandWeb**, upload the shapefile to Google Drive and record the file's Google Drive ID.

```
shpfile <- "inputs/NW_AB.shp"
sf::st_write(nwab, shpfile)

result <- googledrive::drive_put(shpfile, googledrive
::as_id("1LsYuuYICkcpElAkEABFM5zJXf5tTyMLG"))
fid <- result$id
```

3. Determine the name to use for running LandWeb with your new study area, and modify `LandWeb_preamble.R` to use this new

study area by name. Add a new case to the `if-else` block in `LandWeb_preamble.R`:

```
} else if (grepl("customABNW", P(sim)$.studyAreaName))
{
  ml <- customABNW(ml, P(sim)$.studyAreaName, dataDir,
sim$canProvs, P(sim)$bufferDist, asStudyArea = TRUE)
```

4. Add this new study area to the `map` object by wrapping the necessary components in a new function defined in a new file at `LandWeb_preamble/R`. It's simplest to use an existing function/file in `LandWeb_preamble/R` as a template, making modifications as appropriate for your new study area.
   a. copy `LandWeb_preamble/R/provAB.R` to `LandWeb_preamble/R/NWAB.R`;
   b. edit `LandWeb_preamble/R/NWAB.R`, to crop `AB` to the new study area (`NWAB`) and subsquently using `NWAB` in lieu of `AB` throughout;
   c. remove any unnecessary elements (*e.g.*, planning units and planning regions).
5. You should now be able to run the LandWeb model setting '.studyAreaName = "customABNW" and it will use the new study area.

5.3 Example 3: updating the LTHFC map

Using an updated version of a data source requires only basic code modifications provided the following conditions are met:

- spatial data attributes remain the same (*e.g.*, field names are the same);
- a new URL is provided to the new data source;
- the new file name in different from the previous version (*e.g.*, it has a version number or date).

Here we show how to modify the `LandWeb_preamble` module to use a hypothetical new version of the longs-term historic fire cycle (LTHFC) map.

1. looking at `LandWeb_preamble.R` we see that the current version of the file is `landweb_ltfc_v8.shp`.
2. create a new spatial layer and save as `landweb_ltfc_v9.shp`.
3. zip the new shapefile (with auxiliary files) upload this zip file to Google Drive as `landweb_ltfc_v9.zip`, making note of the Google Drive share URL.
4. modify the **two** `mapAdd()` calls in `LandWeb_preamble.R` that reference the previous version (v8) of the LTHFC map to use the new (v9) Google Drive URL.

5.4 Contributing changes

- via pull request[1] against `development` branch on GitHub

[1] https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/creating-a-pull-request

References

Bibliography

[1] David W Andison. "Managing for landscape patterns in the sub-boreal forests of British Columbia". PhD Thesis. Vancouver, BC: University of British Columbia, 1996. URL: https://open.library.ubc.ca/cIRcle/collections/ubctheses/831/items/1.0075275.

[2] David W. Andison. "Temporal patterns of age-class distributions on foothills landscapes in Alberta". In: *Ecography* 21.5 (1998), pp. 543–550. ISSN: 09067590. DOI: 10.1111/j.1600-0587.1998.tb00446.x[2].

[3] A Beaudoin et al. "Mapping attributes of Canada's forests at moderate resolution through kNN and MODIS imagery". In: *Canadian Journal of Forest Research* 44 (2014), pp. 521–532. DOI: 10.1139/cjfr-2013-0401[3].

[4] A Beaudoin et al. *Species composition, forest properties and land cover types across Canada's forests at 250m resolution for 2001 and 2011*. 2017. DOI: 10.23687/EC9E2659-1C29-4DDB-87A2-6ACED147A990[4]. URL: http://open.canada.ca/data/en/dataset/ec9e2659-1c29-4ddb-87a2-6aced147a990.

[5] Alex M. Chubaty and Eliot J. B. McIntire. *SpaDES: Develop and Run Spatially Explicit Discrete Event Simulation Models*. 2019. URL: https://CRAN.R-project.org/package=SpaDES.

[6] John Cosco. *COMMON ATTRIBUTE SCHEMA (CAS) FOR FOREST INVENTORIES ACROSS CANADA*. Feb. 2011, p. 117.

[7] Paul D Pickell and Nicholas C Coops. *Development of historical forest attribute layers using Landsat time series and kNN imputation for the western Canadian boreal forest*. Tech. rep. University of British Columbia, Dec. 2016, pp. 1–27.

[2] https://doi.org/10.1111/j.1600-0587.1998.tb00446.x
[3] https://doi.org/10.1139/cjfr-2013-0401
[4] https://doi.org/10.23687/EC9E2659-1C29-4DDB-87A2-6ACED147A990

[8] Robert M Scheller et al. "Design, development, and application of LANDIS-II, a spatial landscape simulation model with flexible temporal and spatial resolution". In: *Ecological Modelling* 201 (2007), pp. 409–419. ISSN: 03043800. DOI: 10.1016/j.ecolmodel.2006.10.009[5].

[9] Robert M. Scheller and Brian R. Miranda. *LANDIS-II Biomass Succession v3.2 Extension – User Guide*. 2015.

[10] Robert M. Scheller and David J. Mladenoff. "A forest growth and biomass module for a landscape simulation model, LANDIS: design, validation, and application". In: *Ecological Modelling* 180.1 (Dec. 2004), pp. 211–229. DOI: 10.1016/j.ecolmodel.2004.01.022[6]. URL: https://linkinghub.elsevier.com/retrieve/pii/S0304380004003837.

[5] https://doi.org/10.1016/j.ecolmodel.2006.10.009
[6] https://doi.org/10.1016/j.ecolmodel.2004.01.022