Introduction to the vegClass R Package

August 2024

Introduction to the vegClass R Package

Mark Castle, 2023 (revised August 21, 2024). Introduction to the vegClass R Package. Internal Rep. Fort Collins, CO: U. S. Department of Agriculture, Forest Service, Forest Management Service Center. 48p.

Table of Contents

1.0 Introduction

This user guide provides an overview of the vegClass R package. The vegClass package consists of a suite of R functions that are used to derive vegetation classifications and metrics defined by the US Forest Service (Vandendriesche 2013). Currently, the package only supports US Forest Service Regions 1, 3 and 8 rule sets for deriving vegetation classifications. The outputs that are produced from the functions in this package can be used as input for state and transition models and for other forest planning applications. The vegClass package is implemented through the R statistical programming language and environment (R) and depends on output produced from the Forest Vegetation Simulator (FVS) software. The contents of this guide describe the software you will need to run the vegClass R package, the workflow for producing output with the vegClass package, and documentation on the underlying R functions included in the vegClass package.

2.0 Installing Required Software

Three software components are required to use the vegClass R package: FVS software, R, and the vegClass R package itself. Instructions for installing each of these are provided in the sections below. Each software component should be installed in the order that they are presented in this guide.

Note: If you are an employee in the USDA Forest Service, then you can install RStudio Desktop IDE (optional) or R (when assuming installation Option 2 described below) from the software center.

2.1 FVS Software

FVS is a nationally supported growth and yield modelling framework that is maintained by the USDA Forest Service. The following link will take you to the webpage where the FVS software can be downloaded (instructions for installing and uninstalling FVS can be found here as well) https://www.fs.usda.gov/fvs/software/complete.php. Ensure that you install the latest version of the FVS software or a version that does not predate version 20210930. Users are encouraged to stay up to date with the latest version of the FVS software. Users also have the option to use the online configuration of FVS that is hosted by Virginia Tech University: https://charcoal2.cnre.vt.edu/FVSOnline/.

2.2 R

The vegClass R package can currently only be implemented through the R statistical programming language and environment. As such, a version of R needs to be installed on the user's PC. There are two options available for installing a version of R that can be used by the vegClass package.

Option 1 **(Recommended)**
The Forest Vegetation Simulator complete package installation (refer to FVS software section above) includes a version of R that is compatible with the vegClass package. This version of R is recommended since it comes included with several R packages that the vegClass package depends on. No further action is required if the FVS complete package is already installed on PC.

Install a version of R through the r-project: https://www.r-project.org/. Instructions for downloading R can be found in the getting started section of the r-project webpage. Please ensure that a version of R greater than or equal to 4.1.1 is downloaded and installed.

2.3 vegClassInstall Folder

The vegClassInstall folder can be downloaded through the following link: TO BE DETERMINED. Once the files have been downloaded, unzip the folder to a desired location on your PC. the vegClassInstall folder contains the following items which will be described in more depth further along in this guide:

Data_Combine_Example.R: R script that illustrates how to combine the FIA2FVS datasets for Arizona and New Mexico and create grouping codes corresponding to ERUs.

FVSOut.db: Example FVS output database that is used to illustrate the use of the vegClass package in vegClass_Example.R.

vegClass_X.X.X.tar.gz: Source vegClass R package that will be installed in Install.R. The X.X.X corresponds to the latest version number associated with the vegClass package.

vegClass_Example.R: R Script that presents an example of how to attach vegClass R package and call the main function to derive vegetation classification output. The installation of the vegClass R package will be described in section 3.0 of the guide.

helloWorld.R: Test R script that is used as an example in Section 4.2.

CustomVars_vegClass.xlsx: Template "order form" for specifying parameters for custom variable calculations. See the "!ReadMe!" sheet for specific instructions on how to fill it out.

2.4 RStudio Desktop IDE (Optional)

Although you can run vegClass directly through the RGui which comes included with the installation of R, it is recommended that you install the RStudio Desktop IDE as well. The RStudio Desktop IDE provides a more intuitive and flexible environment for interacting with R when compared to the default RGui. The following link will take you to the webpage for installing the RStudio IDE: https://www.rstudio.com/products/rstudio/download/.

Note: A desktop icon is not automatically created for RStudio once the application is installed. To easily access RStudio from your desktop, you will need to create a shortcut for the application or pin it to the Windows taskbar. For either option, you will need to navigate to the directory where RStudio was installed to. By default, RStudio is installed to the following directory: C:\Program Files\RStudio. In the RStudio folder, navigate to the bin folder. To pin RStudio to the taskbar, right click on the rstudio.exe file and select **Pin to taskbar** option (Figure

1). Alternatively, if you would like to create a shortcut, you can select the **Create shortcut** option.

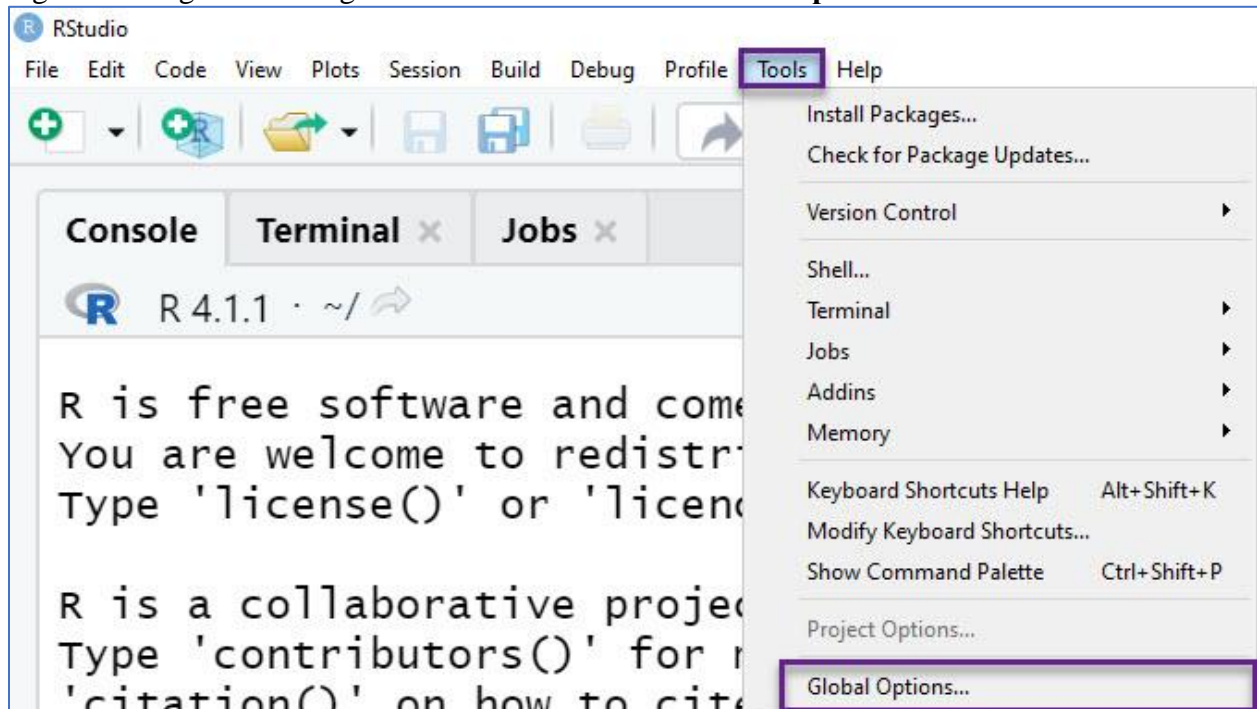Figure 1. Image illustrating how to access rstudio.exe in File Explorer.
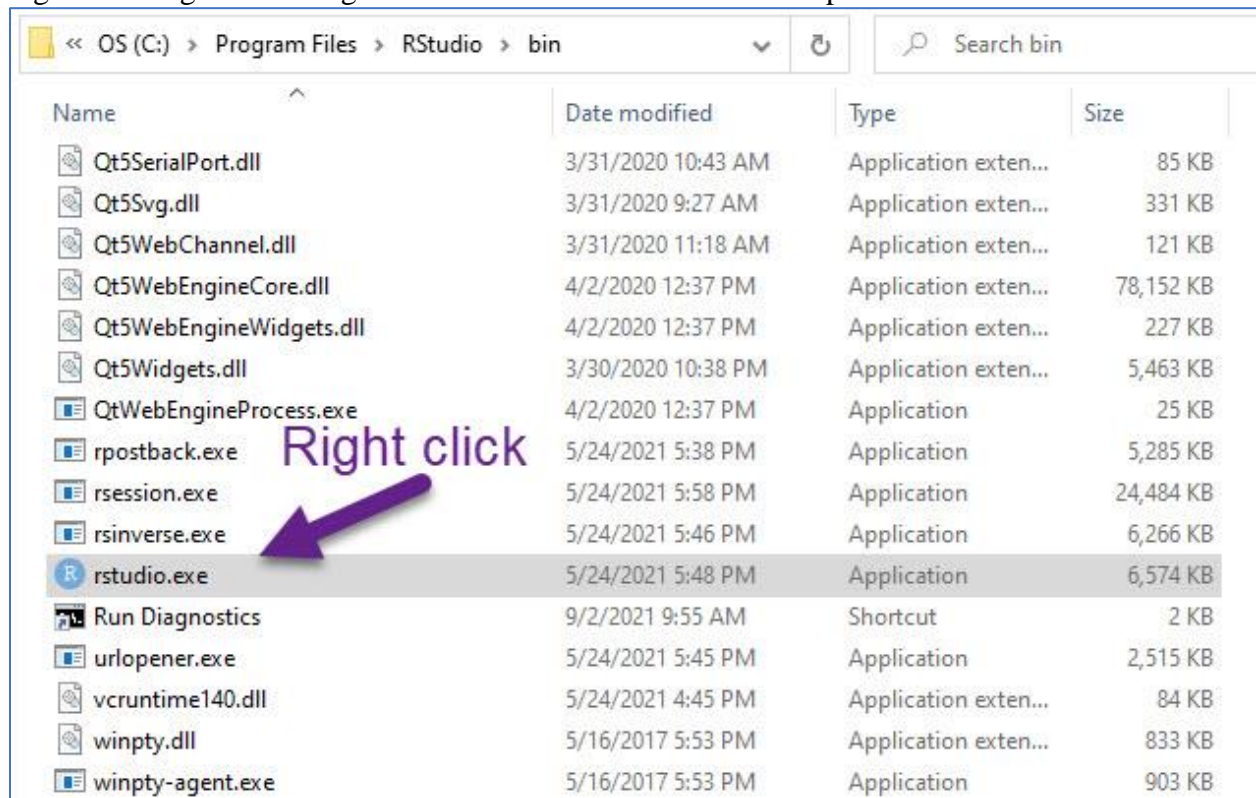
2.4.1 Linking RStudio Desktop IDE to R

If you are planning to use the vegClass package in RStudio, it is necessary to ensure that it is linked to the correct version of R. The following steps detail how to link RStudio to your desired version of R.
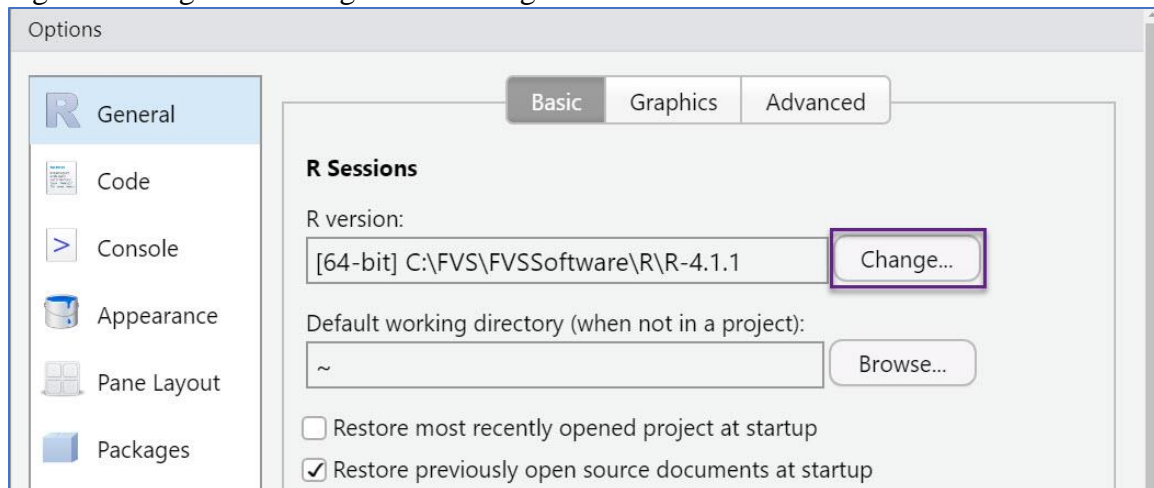
1. Navigate to the **Tools** menu in RStudio and select **Global Options…** from the dropdown list (Figure 2).

Figure 2. Image illustrating how to access **Tools** and **Global Options…** in RStudio.

2. Under R session, click **Change…** next to the **R version** field (Figure 3).

Figure 3. Image illustrating how to change version of R that RStudio uses.

3. Browse for the desired version of R. To do so, click the **Browse…** button under the **Choose a specific version of R** field (Figure 4). **Note: If the desired version of R is already active in the R version field, then skip the remaining steps in section 2.4.1.**

Figure 4. Image illustrating how to browse for version of R on PC in RStudio.
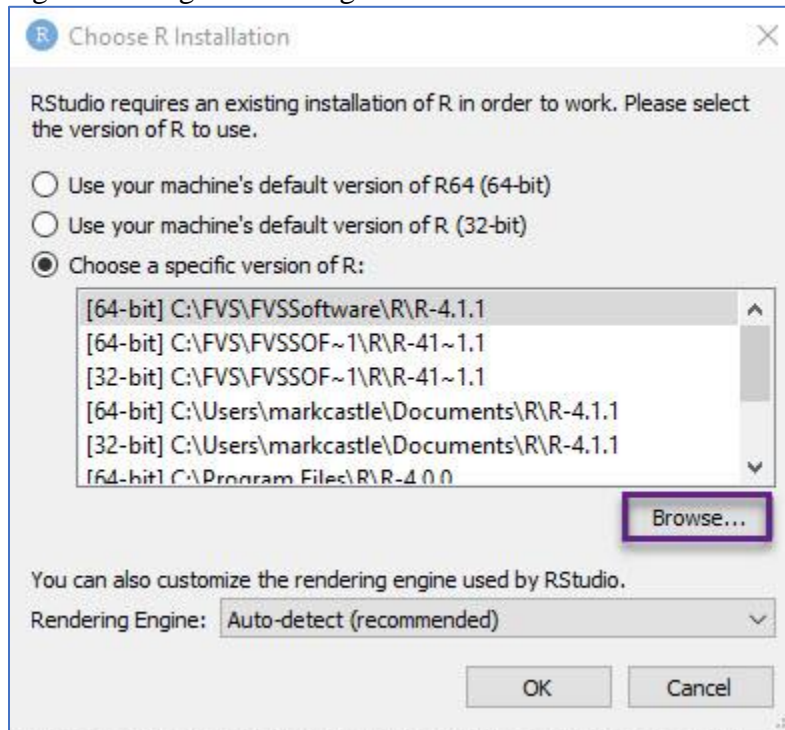

Note: It is recommended that you use the version of R that is distributed with the Forest Vegetation Simulator (FVS) software. By default, this version of R can be found in the following directory on your PC: C:\FVS\FVSSoftware\R\R-X.X.X. The three X values correspond to the version number associated with the installation of R (in this example 4.1.1). If you installed FVS to location other than the C drive, then browse for the version of R in the installation directory.

4. Once you have found the desired version of R, click the **Select Folder** button (Figure 5).

Figure 5. Image illustrating how to select R version folder.
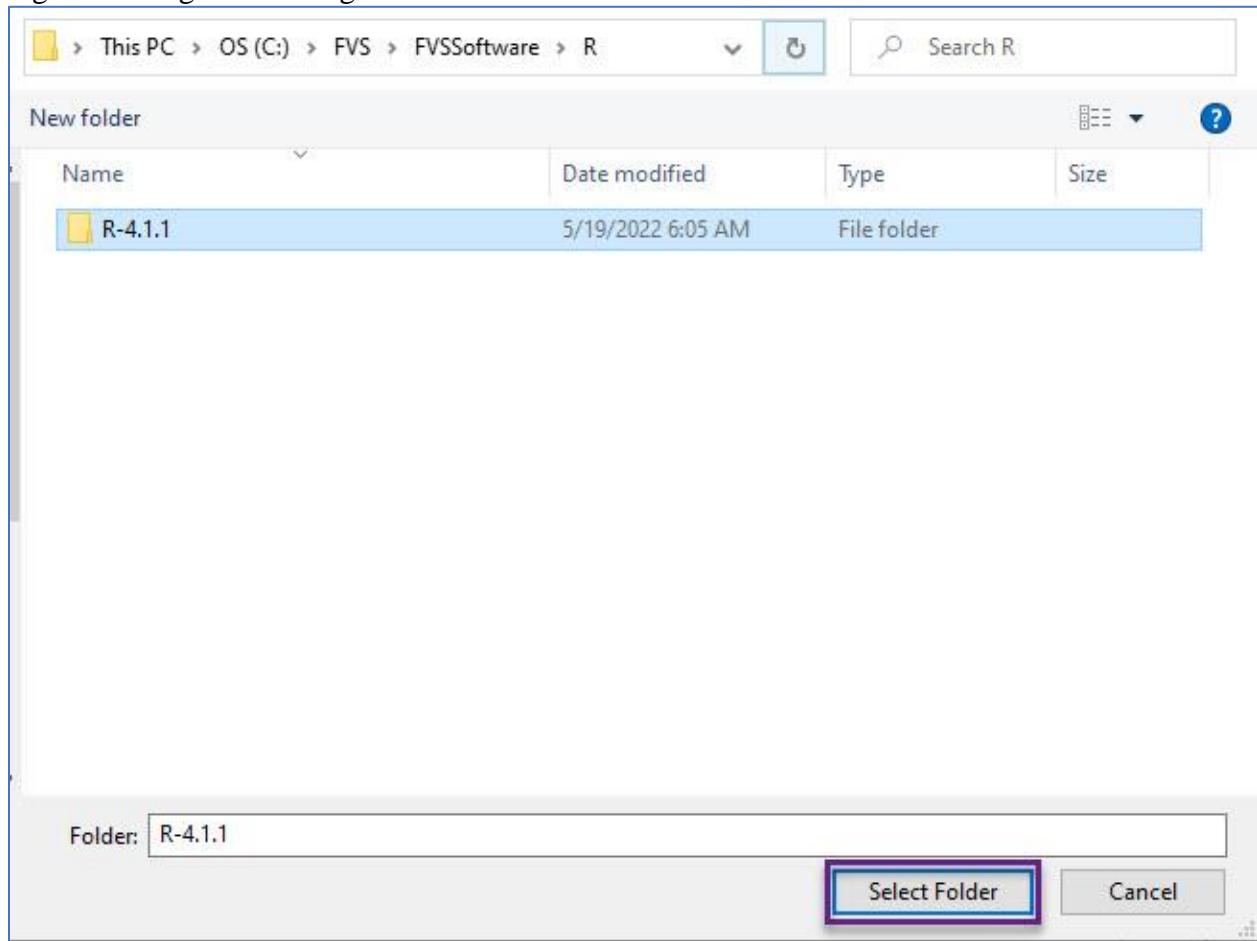
5. You will then be prompted to select either a 32- or 64-bit (Recommended) version of R. Once the version of R is selected, press the **OK** button in the **Choose Version** window (Figure 6).

Figure 6. Image illustrating how to select build of R to use in **Choose Version** window.
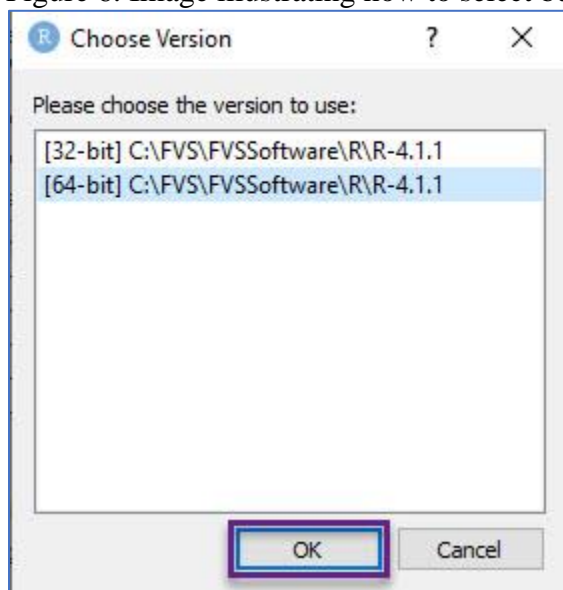
6. Press the **OK** button in the **Choose R Installation** window. You will then be prompted to close and reopen RStudio, so the change takes effect (Figure 7).

Figure 7. Image illustrating how to select version of R to use in **Choose R Installation** window.
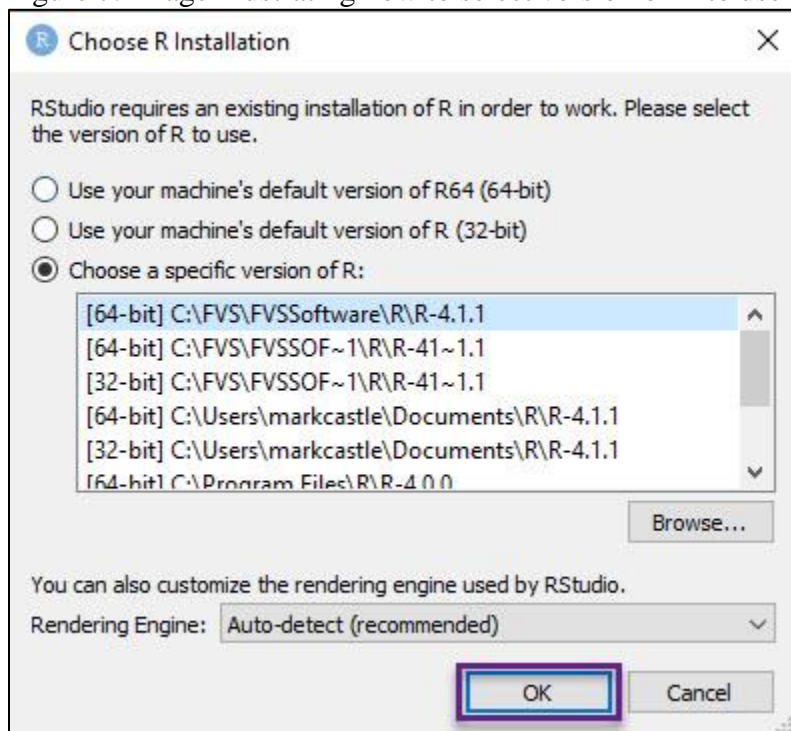
3.0 Installing vegClass R Package

This guide provides instructions for installing the vegClass R package in two environments: RGui (Section 3.1) and RStudio Desktop IDE (Section 3.2). The decision for what instructions to follow depend on the user's preference for the environment that the vegClass package will be implemented in. RGui is currently installed with the FVS package. RStudio Desktop IDE needs to be installed separately and USFS users will need CHD permissions to install it.

3.1 RGui

The RGui interface comes with the version of R that is included with the Forest Vegetation Simulator software. By default, the RGui installation can be found in the C:\FVS\FVSSoftware\R\R-X.X.X\bin directory. The three X values correspond to the version number associated with R (in this example 4.1.1). It is recommended to use the 64-bit version of RGui which can be found in C:\FVS\FVSSoftware\R\R-X.X.X\bin\x64. For ease of use, you can pin RGui to the taskbar or create a shortcut. To pin RGui to the taskbar, right click on the Rgui.exe file and select the **Pin to taskbar** option (Figure 8). Alternatively, if you would like to create a shortcut, you can select the **Create shortcut** option (Figure 8).

Figure 8. Imagine of Rgui.exe file in File Explorer.
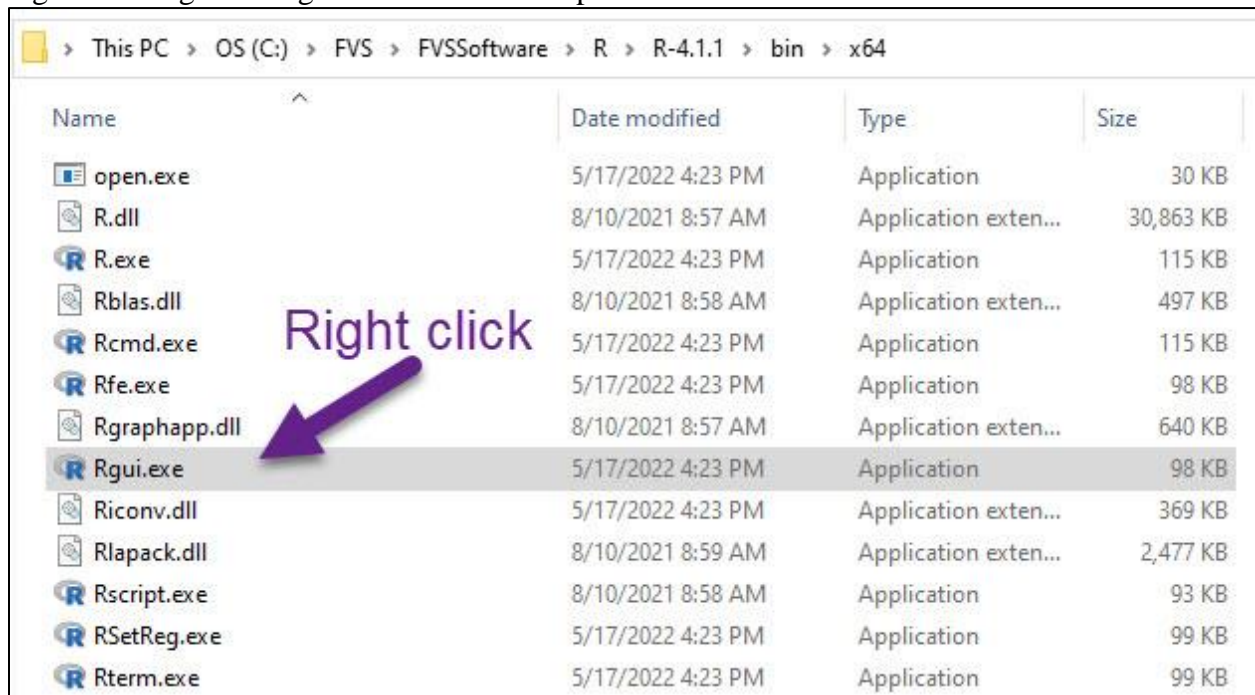
The following steps describe how to install the vegClass R package in the RGui interface.

1. Open the RGui interface.

2. If you are NOT using the version of R that is distributed with FVS and you don't have the RSQLite package (Muller et al. 2022) installed, then you will first need to install this

package. Steps 3 – 5 will outline how to do this. If you already have this package, skip to step 6

3. To install these packages, go to the **Packages** dropdown list and select the **Install package(s)…** option (Figure 9).

Figure 9. Image of **Install package(s)…** option in **Packages** menu of RGui.
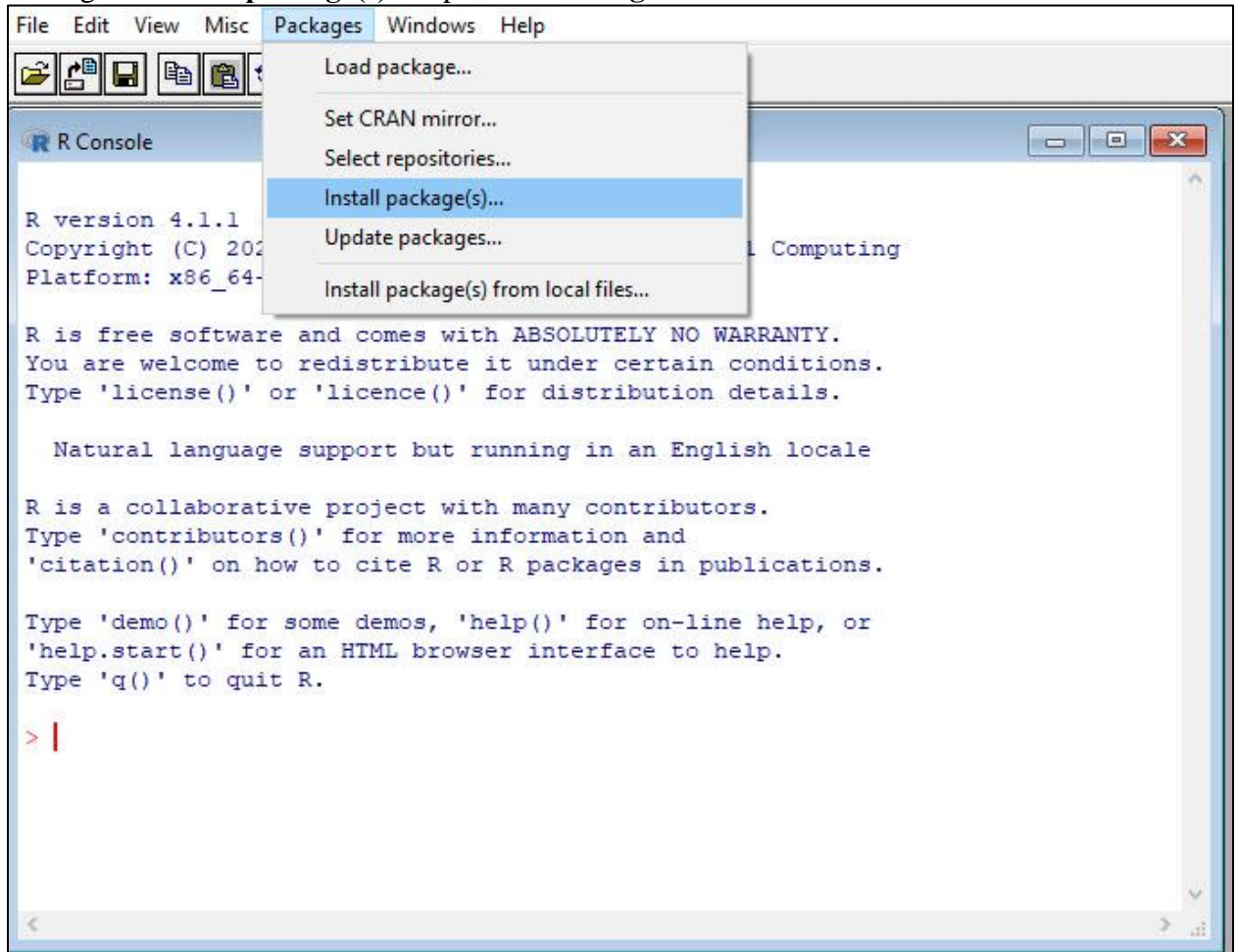
4. You will be prompted to select a CRAN mirror where these packages will be download from. It is recommended that you select a CRAN mirror that is hosted near your geographic region. Once you have selected a CRAN mirror, click the **OK** button.

5. From the **Packages** window, select the RSQLite package and then click the **OK** button. You can select multiple packages by pressing **Ctrl + mouse click**.

6. Now the vegClass package will need to be installed. You can install this by navigating to the Packages dropdown list and selecting the **Install package(s) from local file** option (Figure 10).

Figure 10. Image of **Install package(s) from local files…** option in **Packages** menu of RGui.
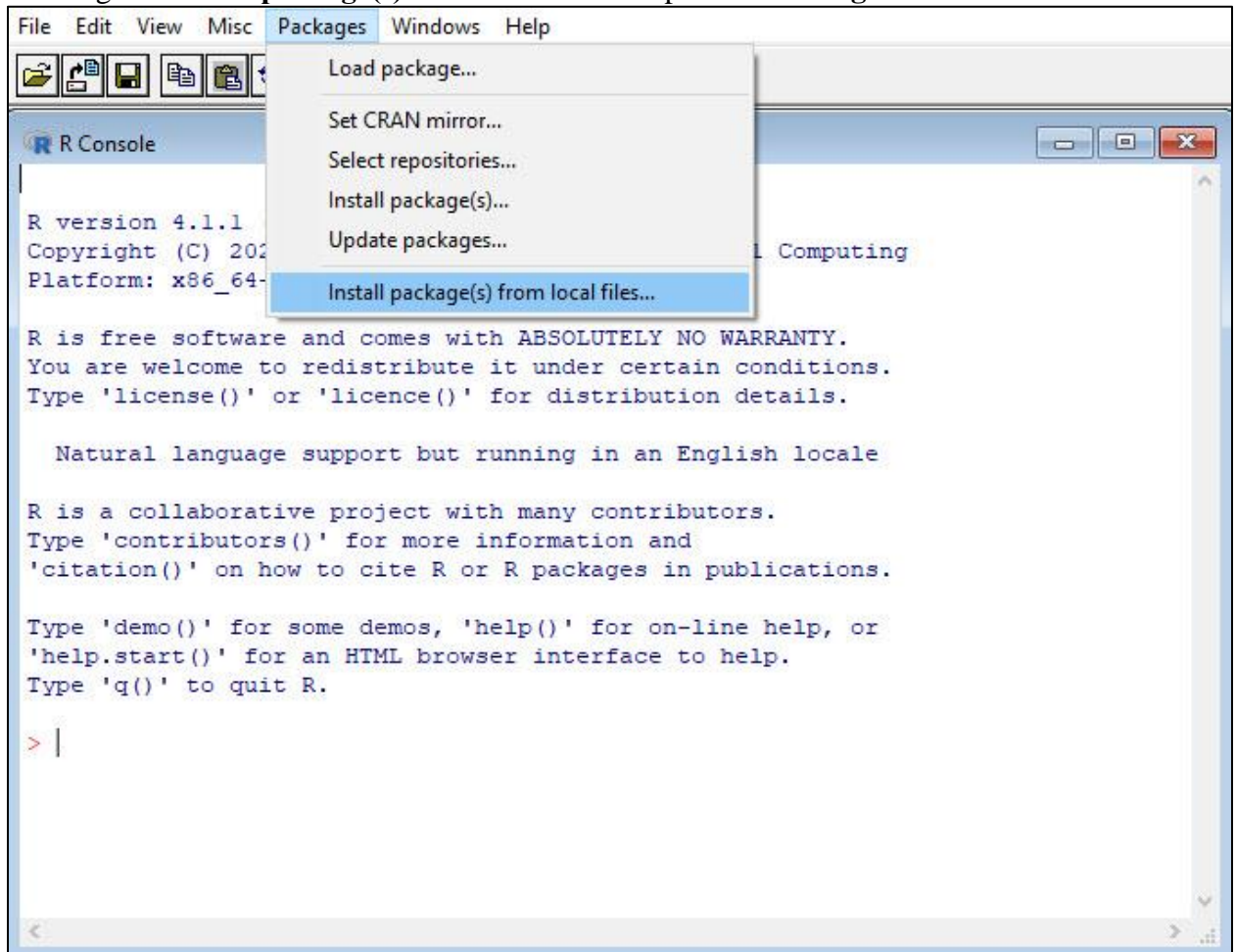
7. You will be prompted to navigate to the directory where vegClass_X.X.X.tar.gz is located. Navigate to this location, select the vegClass_X.X.X.tar.gz file, and then click the **Open** button (Figure 11).
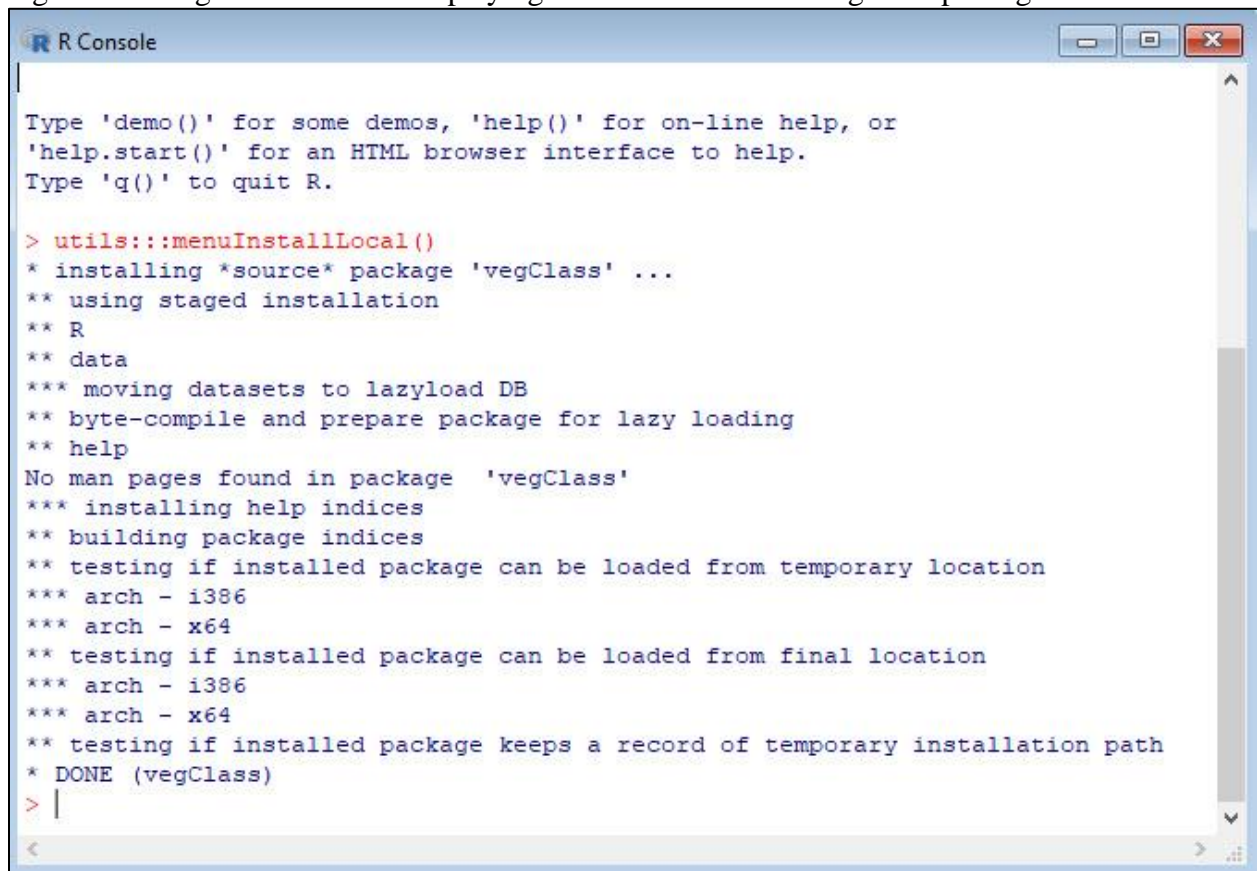
Figure 11. Image of vegClass_X.X.X.tar.gz file in **Select files** window.
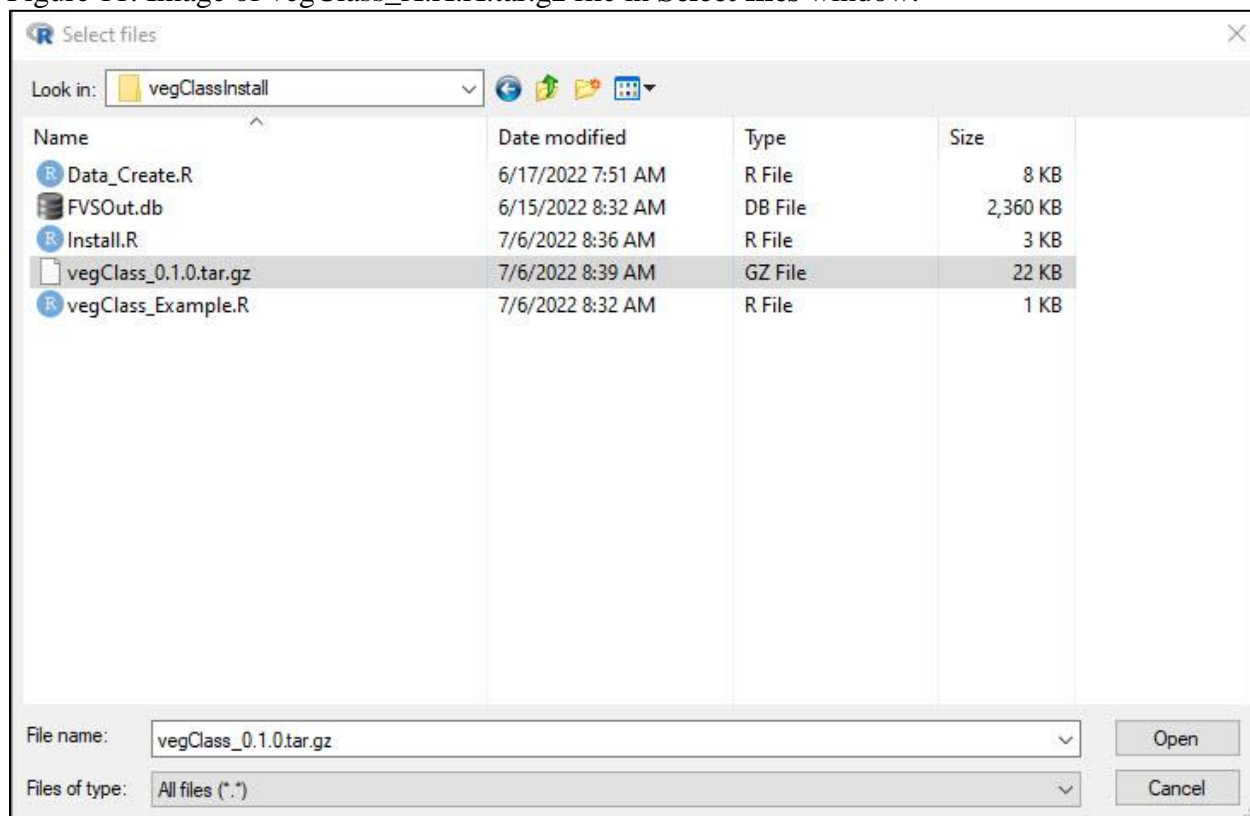
The information in the following screenshot should appear in the R console if the installation of the vegClass package is successful (Figure 12).

Figure 12. Image of R Console displaying installation status of vegClass package in RGui.

3.2 RStudio Desktop IDE

Once RStudio has been configured (refer to section 2.4), the vegClass R package will need to be installed so it can be accessed in R sessions. The follow steps describe how to install the vegClass R package.

1. Open RStudio and navigate to the **Packages** menu and then click the **Install** button (Figure 13).

Figure 13. Image of Install option in **Packages** menu of RStudio Desktop.
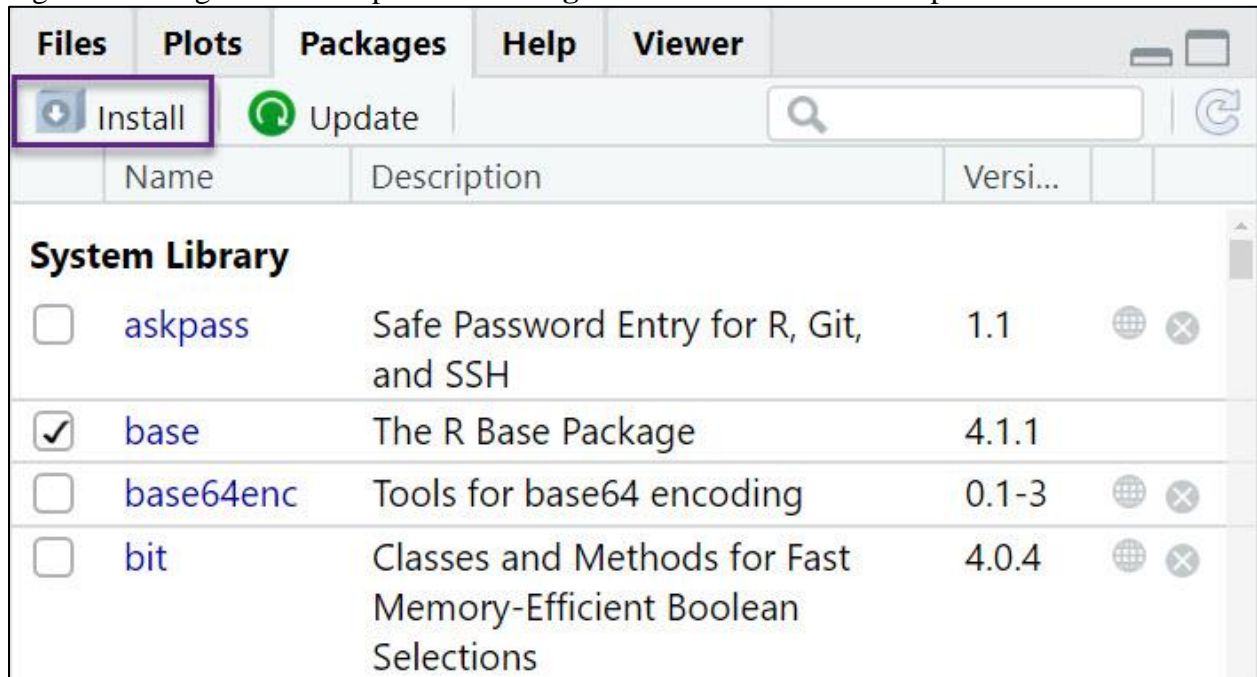
2. If you are NOT using the version of R that is distributed with FVS and you don't have the RSQLite package (Muller et al. 2022) installed, then you will first need to install this package. You can search for packages in the **Packages (separate multiple with space or column)** field and then click the install button (Figure 14).

Figure 14. Image of selected packages in **Install Packages** window of RStudio Desktop.
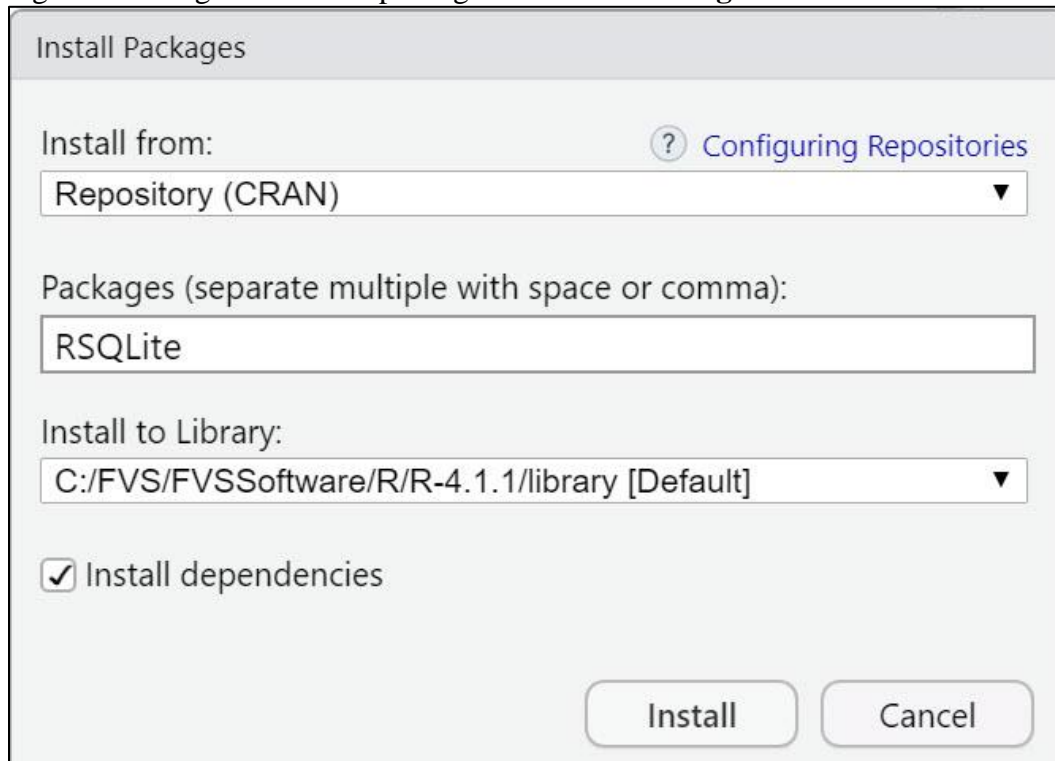
3. Now the vegClass package will need to be installed (vegClass_X.X.X.tar.gz). This package can be installed by selecting the **Package Archive File (.zip; .tar)** option from the **Install from:** dropdown list (Figure 15).

Figure 15. Image of **Package Archive File (.zip; .tar.gz)** option in **Install Packages** window of RStudio.
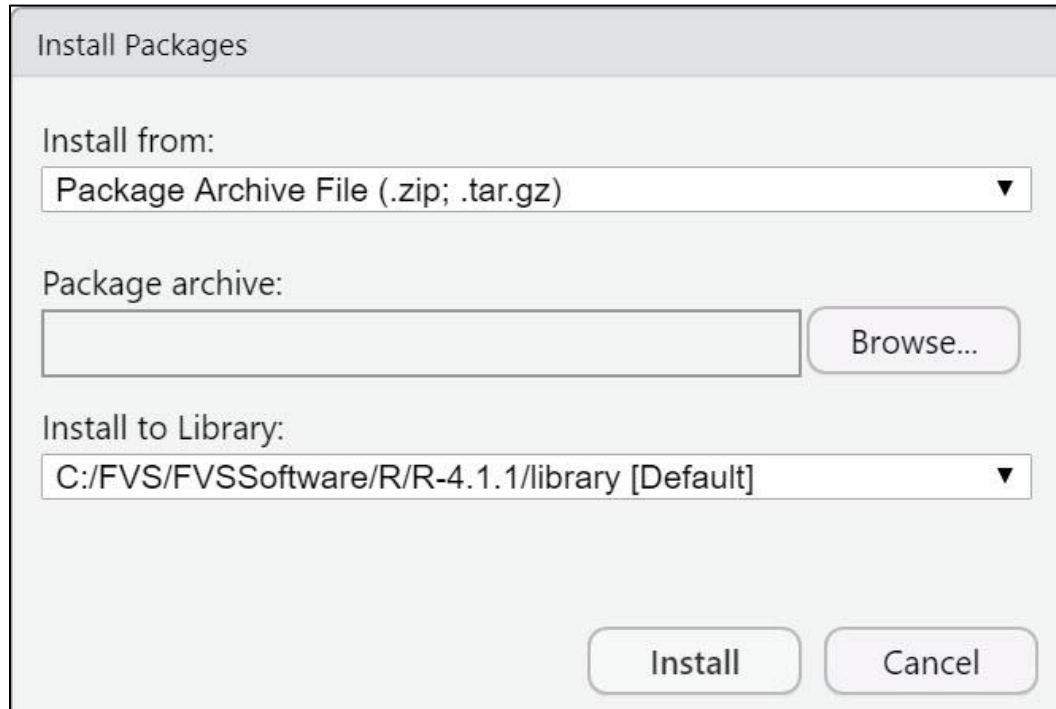
4. You will be prompted to navigate to the directory where vegClass_X.X.X.tar.gz is located. If you are not automatically prompted to browse for the installation file, then click the **Browse** button. Navigate to this location, select the vegClass_X.X.X.tar.gz file, and then click the **Open** button (Figure 16).

Figure 16. Image of vegClass_X.X.X.tar.gz file in **Select Package Archive** window.
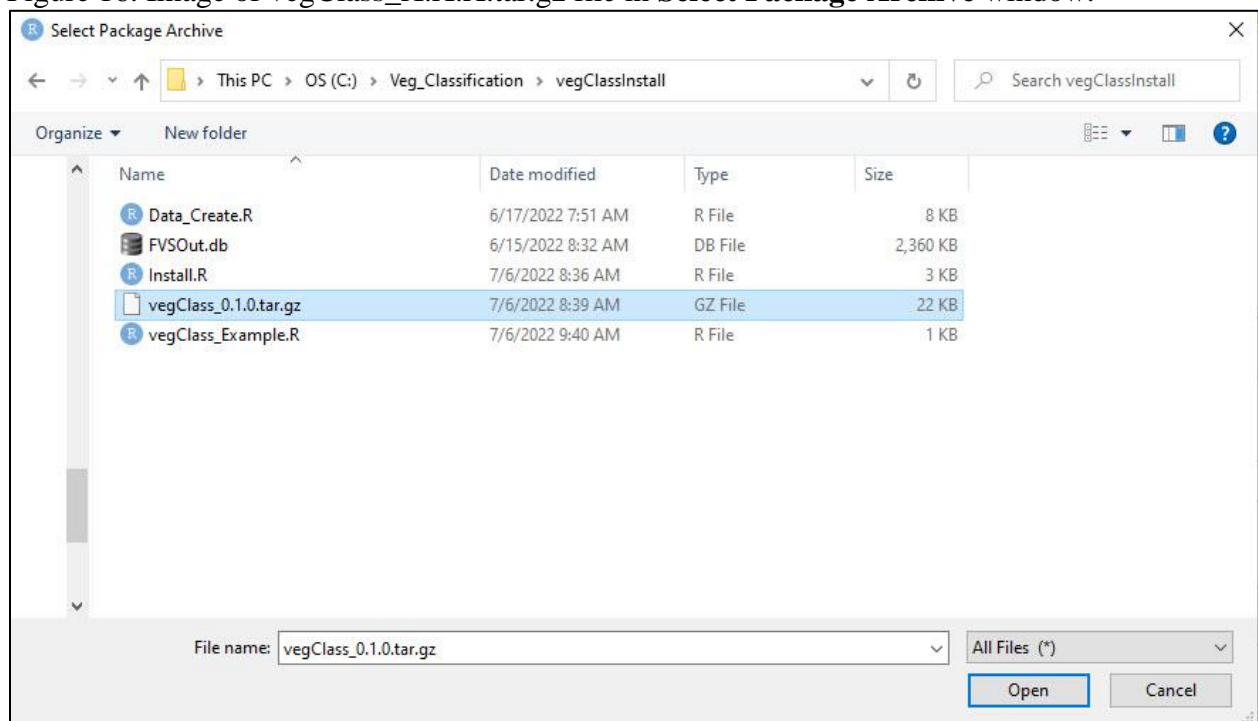
The information in Figure 17 should appear in the **Console** if the installation of the vegClass package is successful.

Figure 17. Image of **Console** output following installation of vegClass R package in RStudio Desktop.
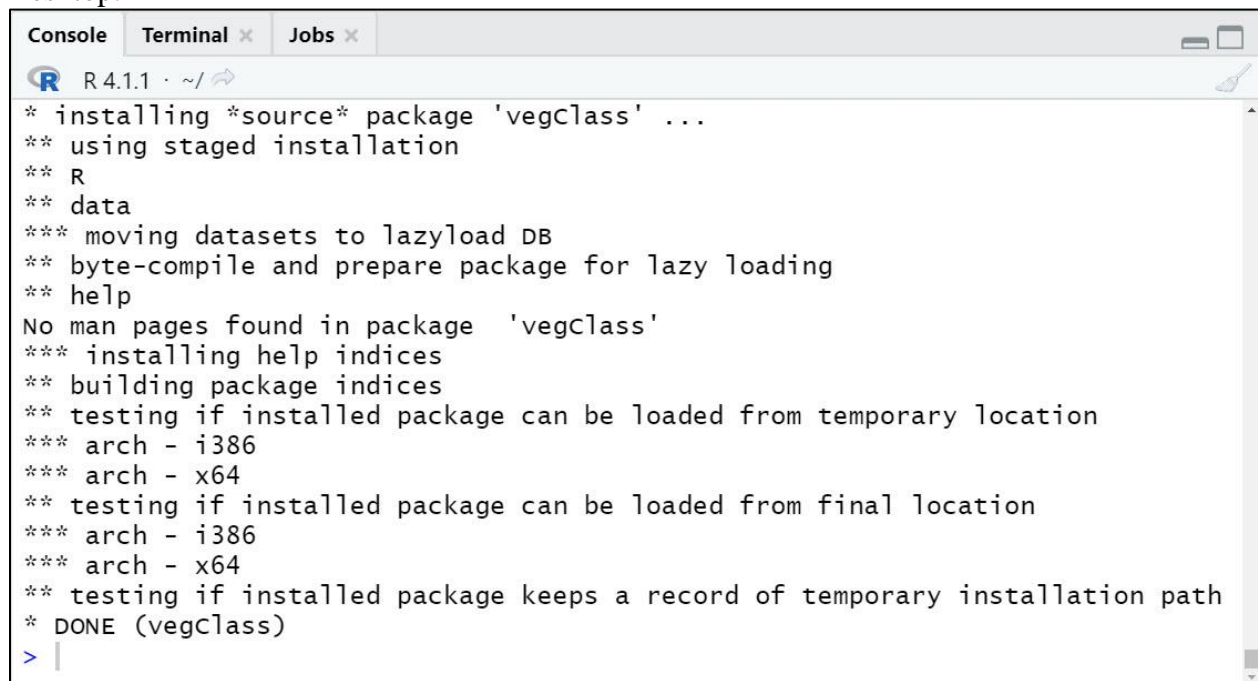
4.0 Workflow for Deriving Vegetation Classifications

4.1 General Processing Sequence

Deriving vegetation classifications with the vegClass package is a two-step process. First, the user will parameterize and run FVS simulations to derive stand and tree level output. The output derived in FVS simulations is stored in SQLite databases. The vegClass package is then implemented in R to derive vegetation classifications using the output SQLite database (.db) produced by FVS. The vegetation classifications will be sent to a comma separated value file (.csv). Figure 18 provides a conceptual diagram of the general processing sequence used for producing vegetation classification information.

Figure 18. Conceptual illustration of general processing sequence for producing vegetation classification output.
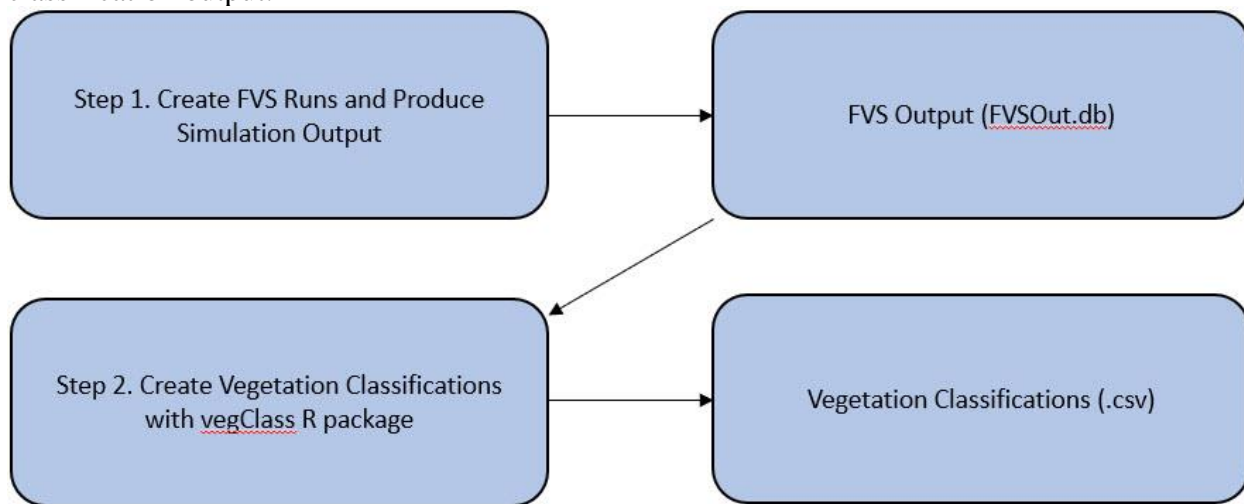
4.2 The Basics of Using R

Users will invoke the vegClass package through the R statistical programming language and environment. R is an interpreted, script-based language that does not depend on compiled code. The functions in the vegClass package are to be used within R scripts (.R files) that be can be created in either the RGui and RStudio Desktop applications. The sections below will provide basic details on how to create scripts and run code using RGui and RStudio Desktop.

4.2.1 Creating New Scripts in RGui

New R scripts can be created in RGui through the following steps:

1) Navigate to the **File** menu in the top left corner of RGui and selecting the **New Script** option from the dropdown list.

2) Once **New Script** is selected, an untitled script will be created displayed in the **Editor** window of RGui. Users can provide a name for the script and save it to a desired location on their computer by selecting the **File** menu and clicking the **Save as** button.

Two windows should now be seen in RGui, the **Editor** window and the **R Console** window.

Users can arrange the position of these windows by dragging them with their mouse or by selecting a window arrangement options (Cascade, Tile Horizontally, Tile Vertically, or Arrange Icons) from the **Windows** menu (Figure 19). The **Editor** window is where you specify the commands and instructions that will be executed in the R script. Code in scripts is run from within the **Editor** window. When line(s) of code are run, output will be displayed in the **R Console** window.

Figure 19. Figure illustrating how to select window arrangement in RGui.
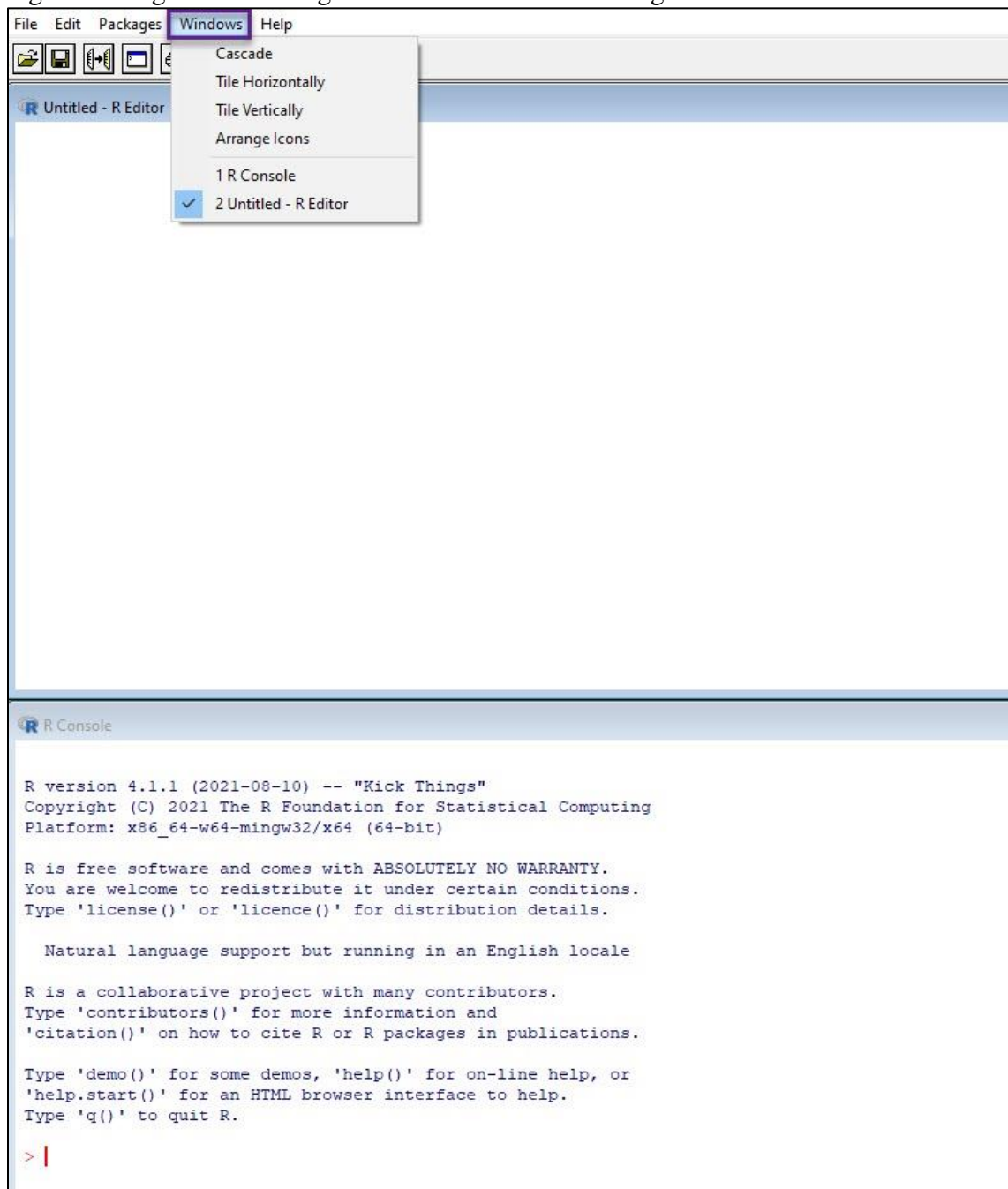
4.2.2 Opening Existing Script in RGui and Running Code

This section of the guide details how existing R scripts can be opened on a computer. The steps below show how to open the helloWorld.R file that comes with the vegClass installation.

1) Navigate to the **File** menu in RGui and select the **Open Script…** option. RGui will then prompt you to select an R file to open in the **Open script** window.

2) Navigate to the helloWorld.R file, select the file in the **Open script** window and then click the **Open** button (Figure 20).

3) If the arrangement of the **Editor** and **R Console** windows are not to your liking, you can arrange them as described in section 4.2.1.

Figure 20. Example of how to open helloWorld.R script from RGui **Open script** window.
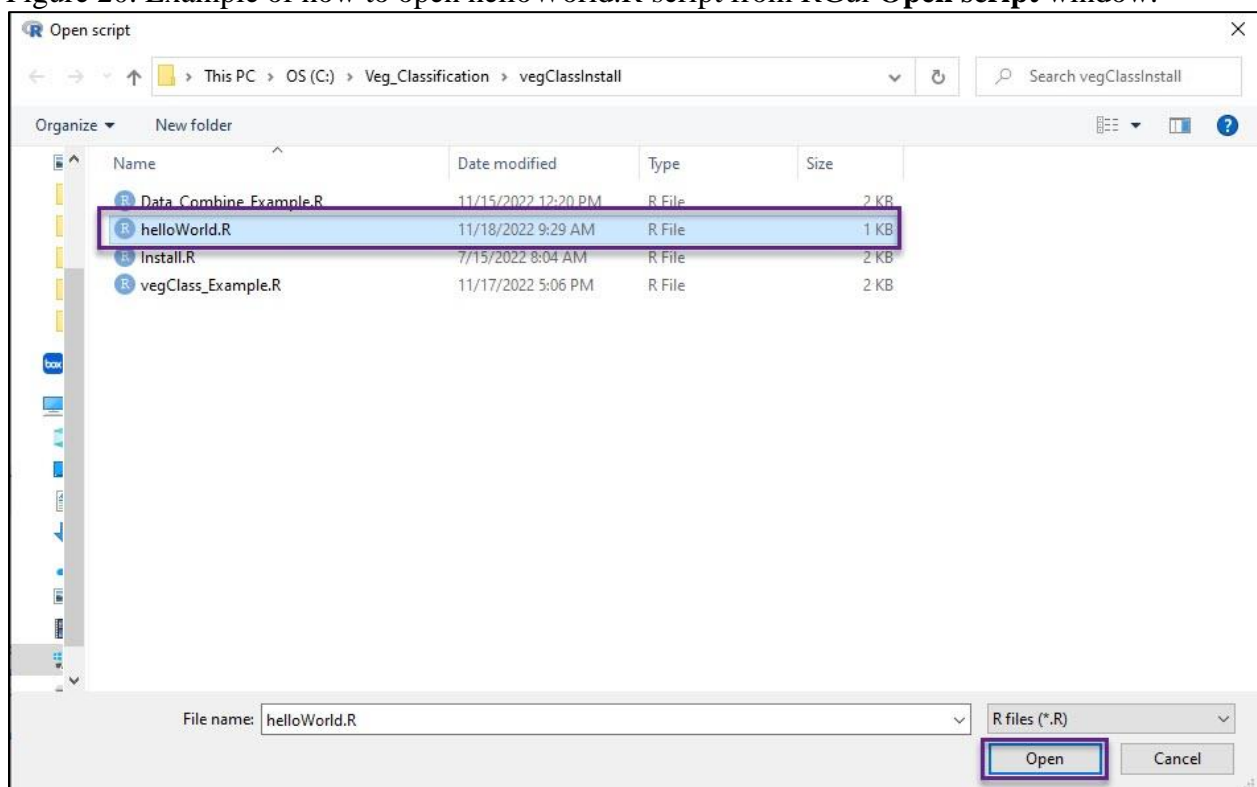
The helloWord.R script should now be shown in the **Editor** window in RGui. Lines of code that start with # are comments and will not be interpreted by R. The purpose of comments is to provide a description of the intent and context of the code in the script for users. The lines with cat("Hello world", "\n) and cat("Welcome to R.") are code that will print messages to the **R Console** window when run. There are numerous ways to run code in R scripts. In general, there are three ways code can be run from within RGui.

1) Individual lines of code can be run by clicking the desired line of code and pressing the **Ctrl + R** buttons on the keyboard. Alternatively, a user can press the **Run line or selection** button (Figure 21) after clicking the desired line of code.

2) Multiple lines of code can be run by highlighting the desired lines of code with a mouse and pressing the **Crtl + R** buttons on the keyboard. Alternatively, a user can press the **Run line or selection** button (Figure 21) after selecting the desired lines of code.

3) An entire script can be run by pressing **Crtl + A** followed by the the the **Ctrl + R** buttons on the keyboard. Alternatively, a user can press the **Run line or selection** button (Figure 21) after selecting all lines of code with **Ctrl + A** buttons.

Figure 21. Location of **Run line or selection** button in RGui.
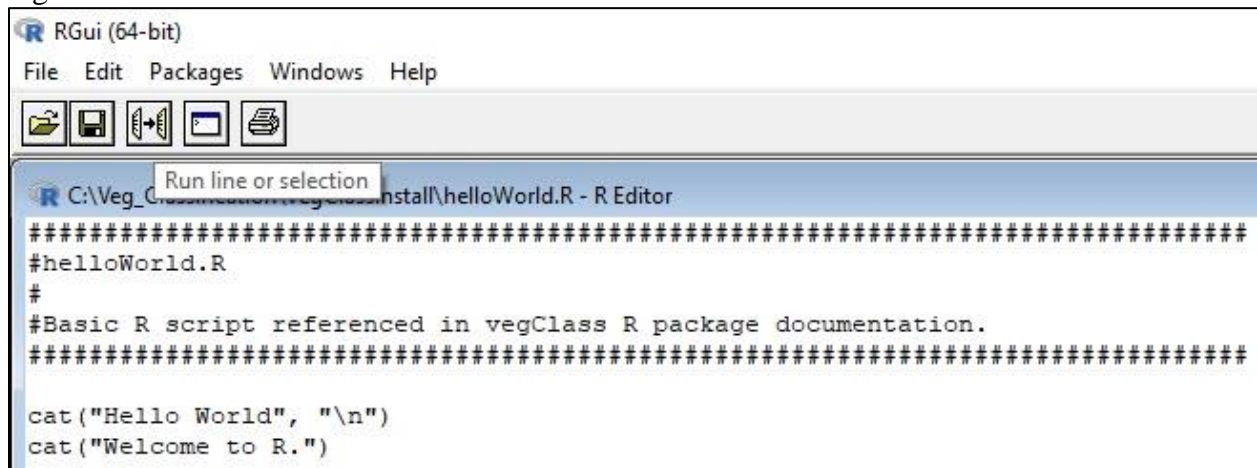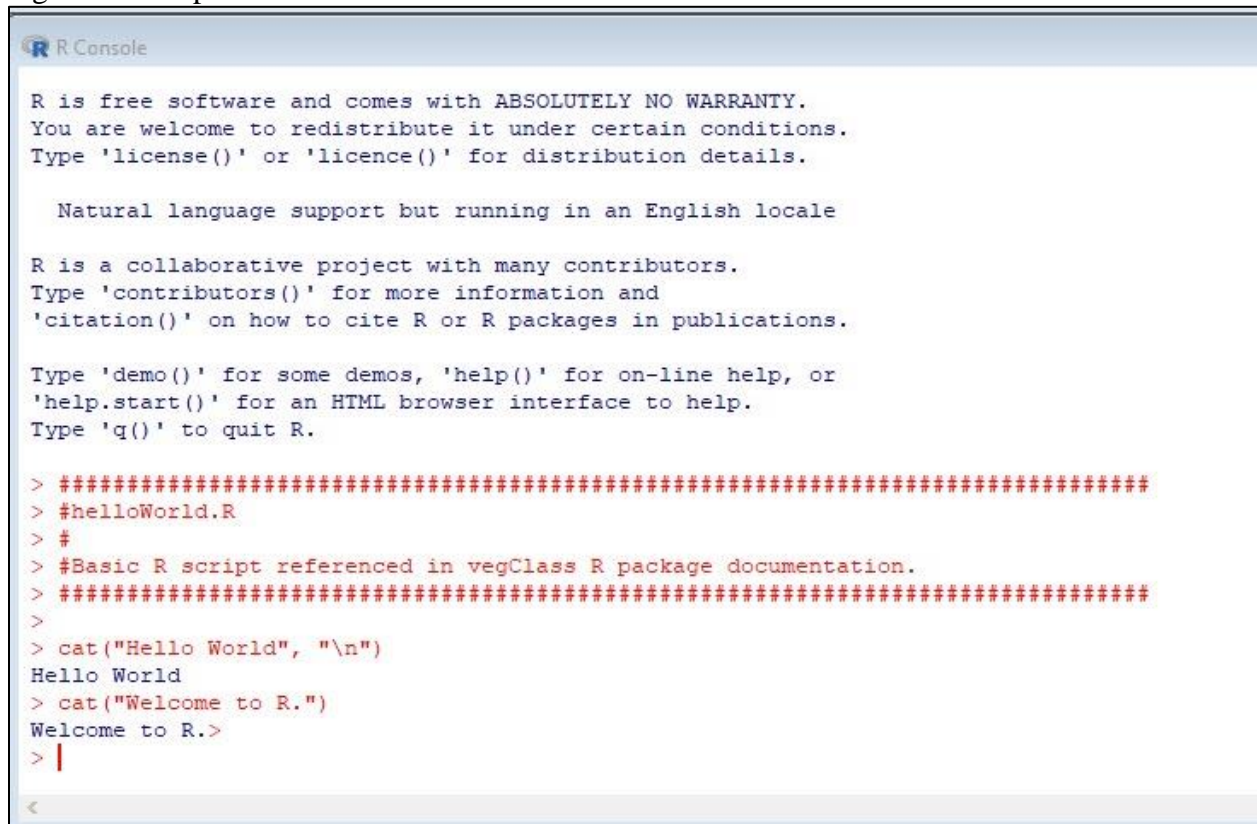
Figure 22 displays the information in the **R Console** window after running the lines of code in the helloWorld.R script using the third option described above. The **R Console** window shows the line(s) of code that are run followed by the output they create.

Figure 22. Output from helloWorld.R in **R Console** window of RGui.

4.2.3 Creating New Scripts in RStudio

New R scripts can be created in RStudio through the following steps:

1) Navigate to the **File** menu in the top left corner of RStudio, select the **New File** option and then **R Script** from the dropdown list (Figure 23).

Figure 23. Steps to open new R script in RStudio.
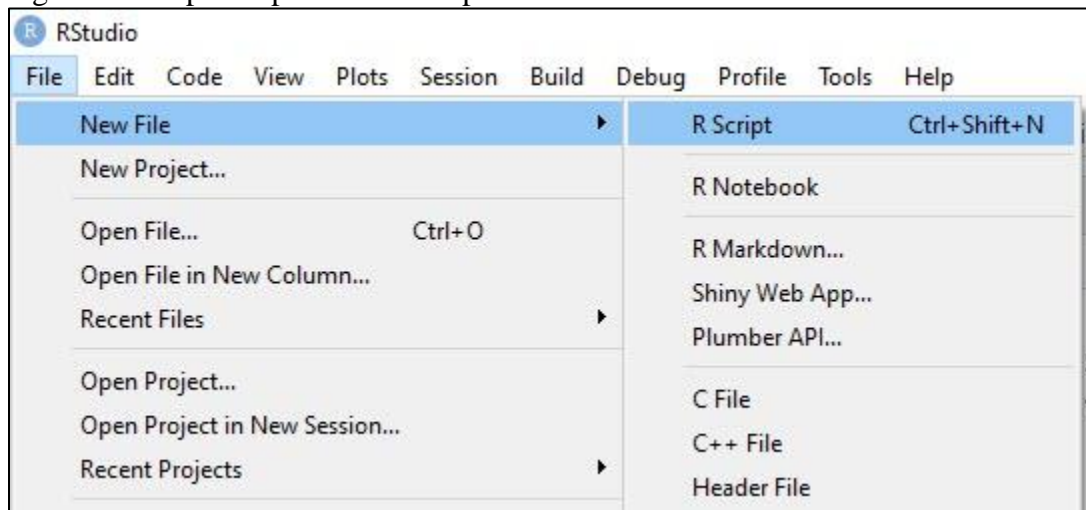
2) Once **R Script** is selected, an untitled script will be created displayed in the **Console** pane of RStudio. Users can provide a name for the script and save it to a desired location on their computer by selecting the **File** menu and clicking the **Save as** button.

Four panes should now be seen in RStudio: **Source** (editor), **Console**, **Environment**, and **Output** (
). More information about each of these panes can be found in the Posit documentation for RStudio: https://docs.posit.co/ide/user/ide/guide/ui/ui-panes.html. The **Source**, **Console**, and **Output** panes will likely be the most important for users of the vegClass R package. The **Source** pane is where you specify the commands and instructions that will be executed in the R script. Code in scripts is run from within the **Source** pane. When line(s) of code are run, output will be displayed in the **Console** pane. The **Output** pane is where you can keep track of R Packages, files, and plots. Users can change the arrangement of panes in RStudio by Selecting the **Tools** menu and **Global Options…** from the dropdown list. The **Pane Layout** option from the **Option** windows can be used to customize the arrangement of panes in RStudio (Figure 25).

Figure 24. **Source**, **Console**, **Environment**, and **Output** panes in RStudio.
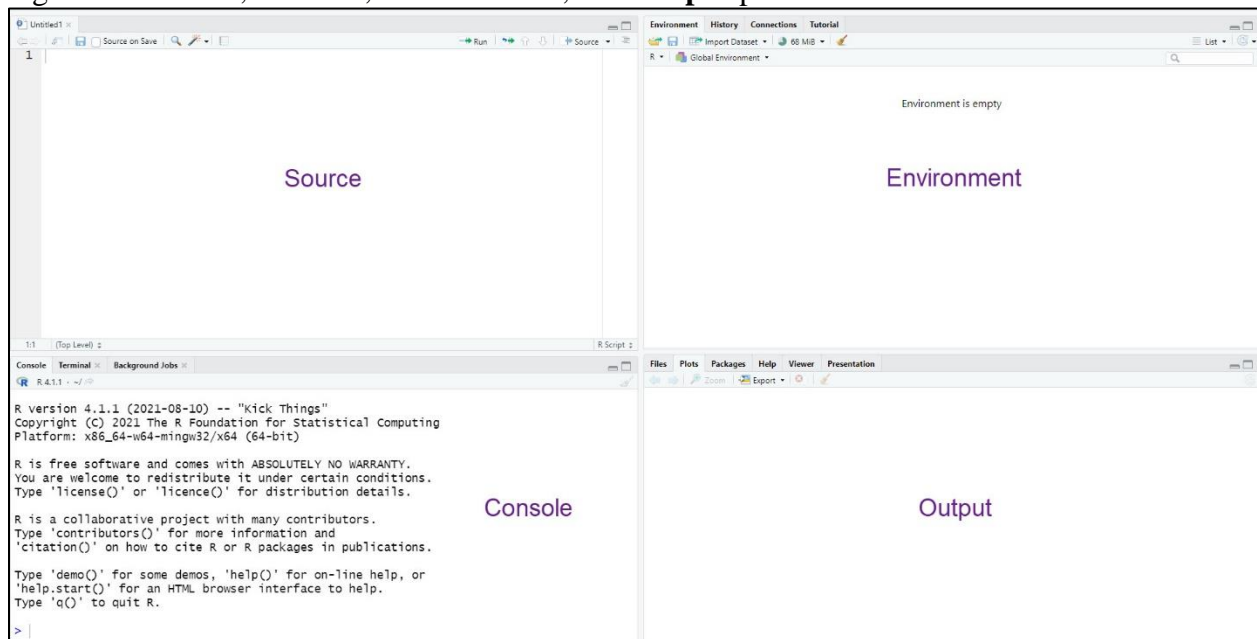
Figure 25. Pane Layout option in **Global Options…** within RStudio.
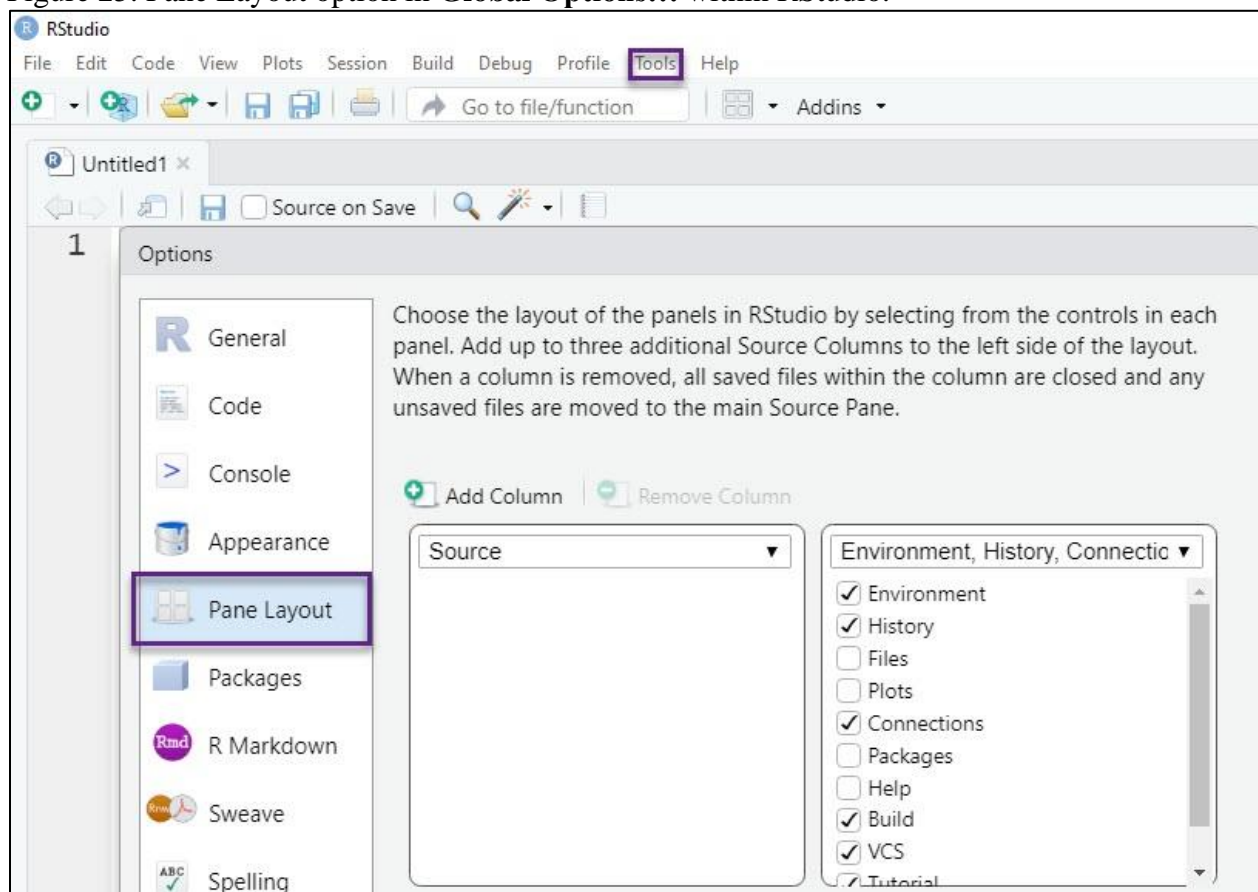
4.2.4 Opening Existing Script in RStudio and Running Code

This section of the guide details how existing R scripts can be opened on a computer. The steps below show how to open the helloWorld.R file that comes with the vegClass installation.

1) Navigate to the **File** menu in RStudio and select the **Open File…** option. RStudio will then prompt you to select an R file to open in the **Open File** window.

2) Navigate to the helloWorld.R file, select the file in the **Open File** window and then click the **Open** button (Figure 26).

3) If the arrangement of the **Source** and **Console** panes are not to your liking, you can arrange them as described in section 4.2.3.

Figure 26. Example of how to open helloWorld.R script from RStudio **Open File** window.
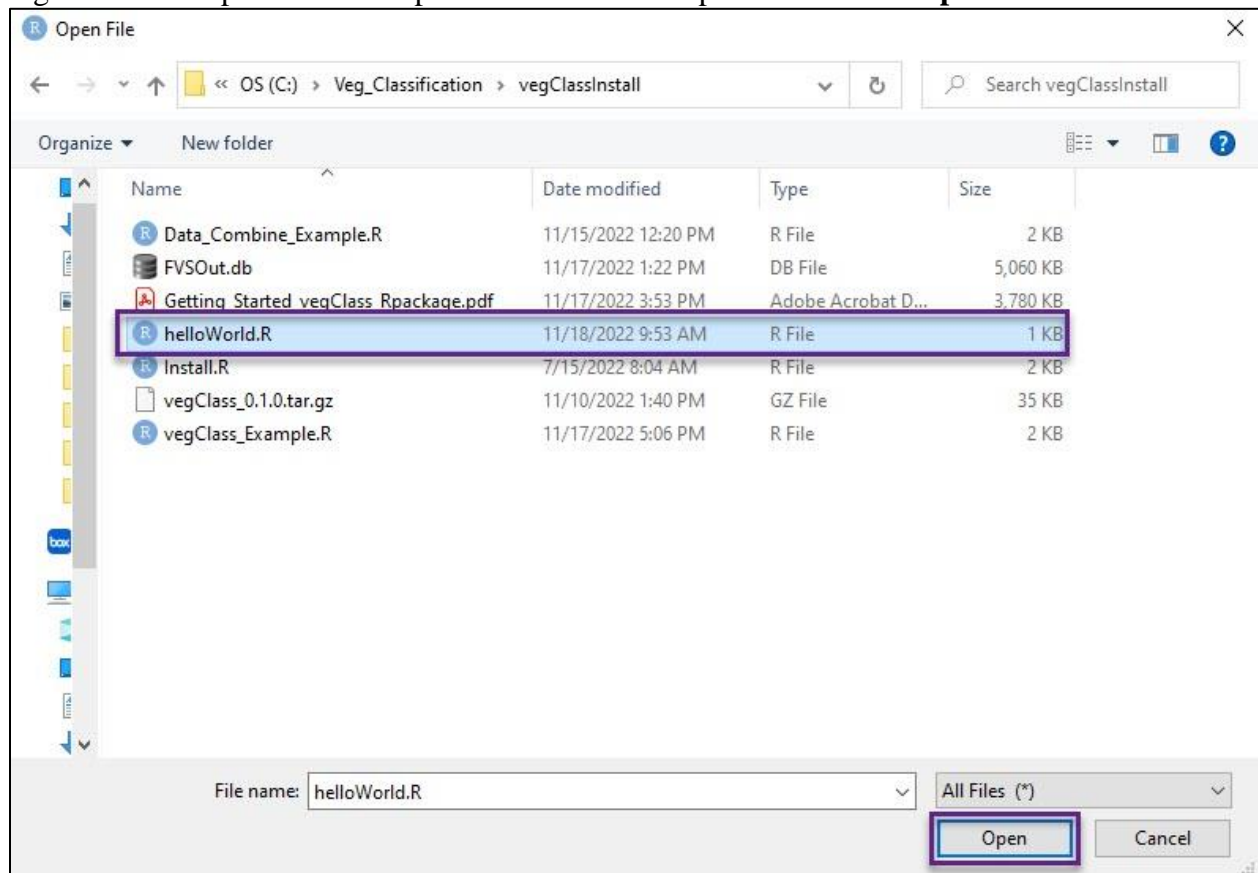
The helloWord.R script should now be shown in the **Source** pane in RStudio. Lines of code that start with # are comments and will not be interpreted by R. The purpose of comments is to provide a description of the intent and context of the code in the script for users. The lines with cat("Hello world", "\n) and cat("Welcome to R.") are code that will print messages to the **Console** pane when run. There are numerous ways to run code in R scripts. In general, there are three ways code can be run from within RStudio.

1) Individual lines of code can be run by clicking the desired line of code and pressing the **Ctrl** + **Enter** buttons on the keyboard. Alternatively, a user can press the **Run** button (Figure 27) after clicking the desired line of code.

2) Multiple lines of code can be run by highlighting the desired lines of code with a mouse and pressing the **Crtl** + **Enter** buttons on the keyboard. Alternatively, a user can press the **Run** button (Figure 27) after selecting the desired lines of code.

3) An entire script can be run by pressing **Crtl** + **A** followed by the **Ctrl** + **Enter** buttons on the keyboard. Alternatively, a user can press the **Run** button (Figure 27) after selecting all lines of code with **Ctrl** + **A** buttons.

27

Figure 27. Location of **Run** button in RStudio.
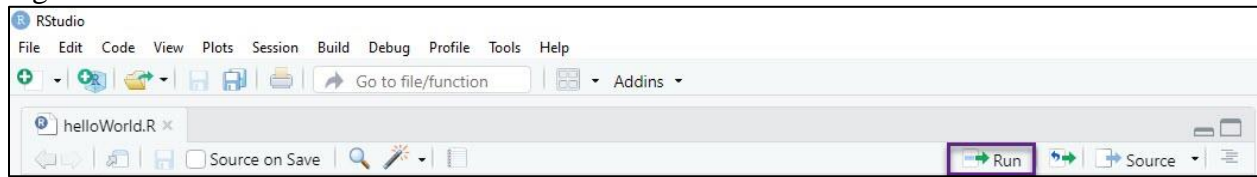
Figure 28 displays the information in the **Console** pane after running the lines of code in the helloWorld.R script using the third option described above. The **Console** pane shows the line(s) of code that are run followed by the output they create.

Figure 28. Output from helloWorld.R in **Console** pane of RStudio.
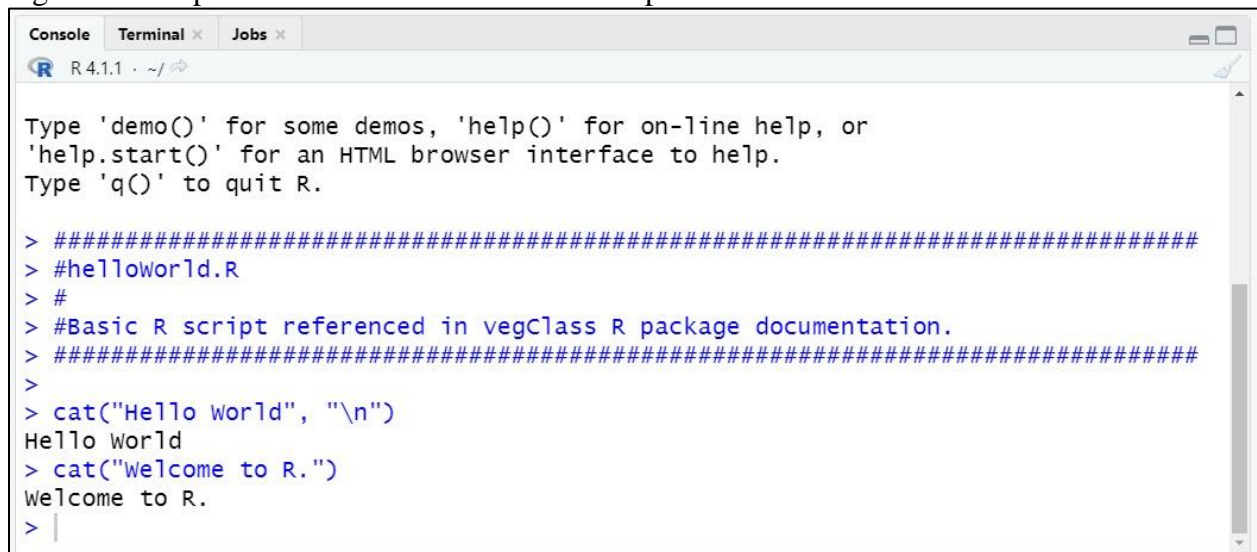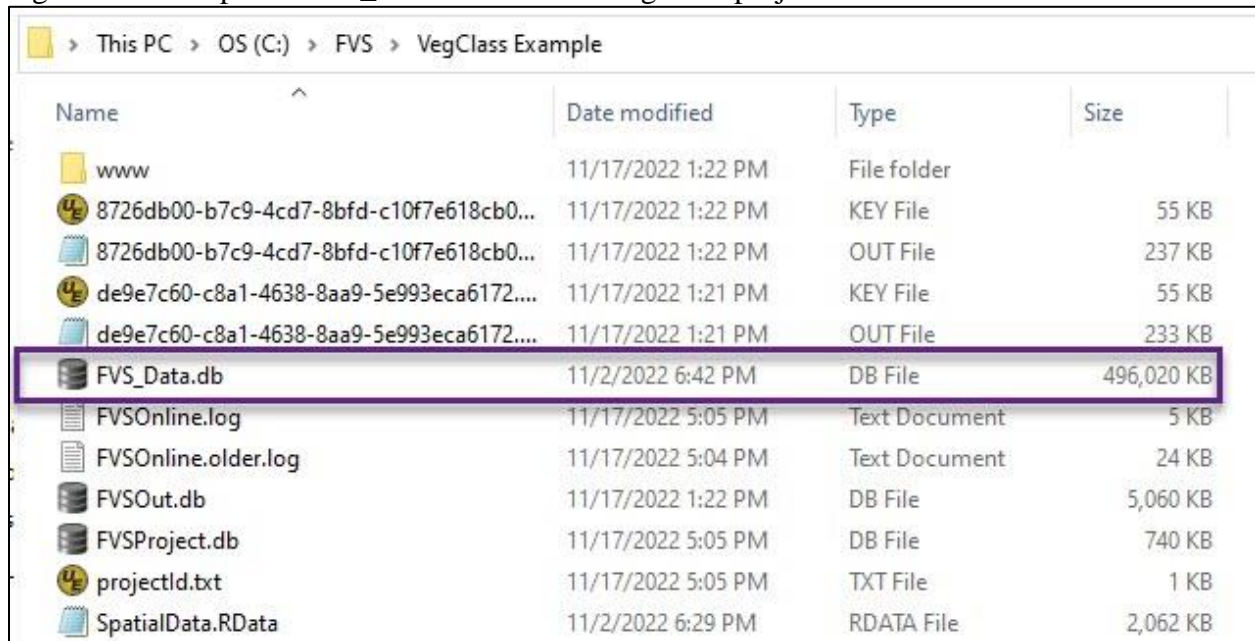
4.3 Producing Simulation Output with FVS

4.3.1 FVS-ready data

Data used in FVS simulations is commonly stored in an input database. Several types of input database file types are available to users when using the FVS interface: Microsoft Access databases (.accdb, .mdb), Excel workbooks (.xlsx), or SQLite3 databases (.db).
Any standard FVS input database must have an FVS_StandInit or FVS_PlotInit database table in addition to an FVS_TreeInit database table. Section 3.0 of the DBS user guide describes the information that can be included in these database tables and the required formatting: https://www.fs.usda.gov/fmsc/ftp/fvs/docs/gtr/DBSUserGuide.pdf. It is worth noting that the FVS interface will make a copy of the users database when it is uploaded into the FVS system and convert the database to a SQLite3 database(.db) that will be stored in the users FVS project folder. The copied database is entitled FVS_Data.db and supplies the data that will be used in simulations associated with the FVS project. Figure 29 shows an example of an FVS_Data.db stored in a project folder entitled VegClass Example.

Figure 29. Example of FVS_Data.db stored in VegClass project folder.

4.3.2 FVS-ready FIA data

FVS-ready Forest Inventory and Analysis (FIA) data is available for users. With this data, users can run FIA plots, conditions, and subplots as stands in FVS simulations. A thorough discuss of FVS-ready FIA data can be found in the Quick-Start Guide to the Forest Inventory and Analysis Data in the Forest Vegetation Simulator guide:
https://www.fs.usda.gov/fvs/documents/FIA_Data_Quick_Start_Guide_20200914.pdf.

FVS-ready FIA can be downloaded on a state-by-state basis from the FIA Datamart: https://apps.fs.usda.gov/fia/datamart/datamart.html. FVS-ready FIA data is only available in SQLite version of the state databases. Users can acquire FVS-ready FIA data from the Datamart by clicking the SQLite button under **Data Type** and selecting a state(s) from the **Select State/s** dropdown list. Figure 30 shows the selection of the New Mexico and Arizona SQLite databases. Once databases are selected, they can be individually downloaded by clicking the SQLite_FIADB_XX.zip files under **Data Available for Download** (Figure 30).

Figure 30. Example of Arizona and New Mexico FIA database selection from FIA Datamart.
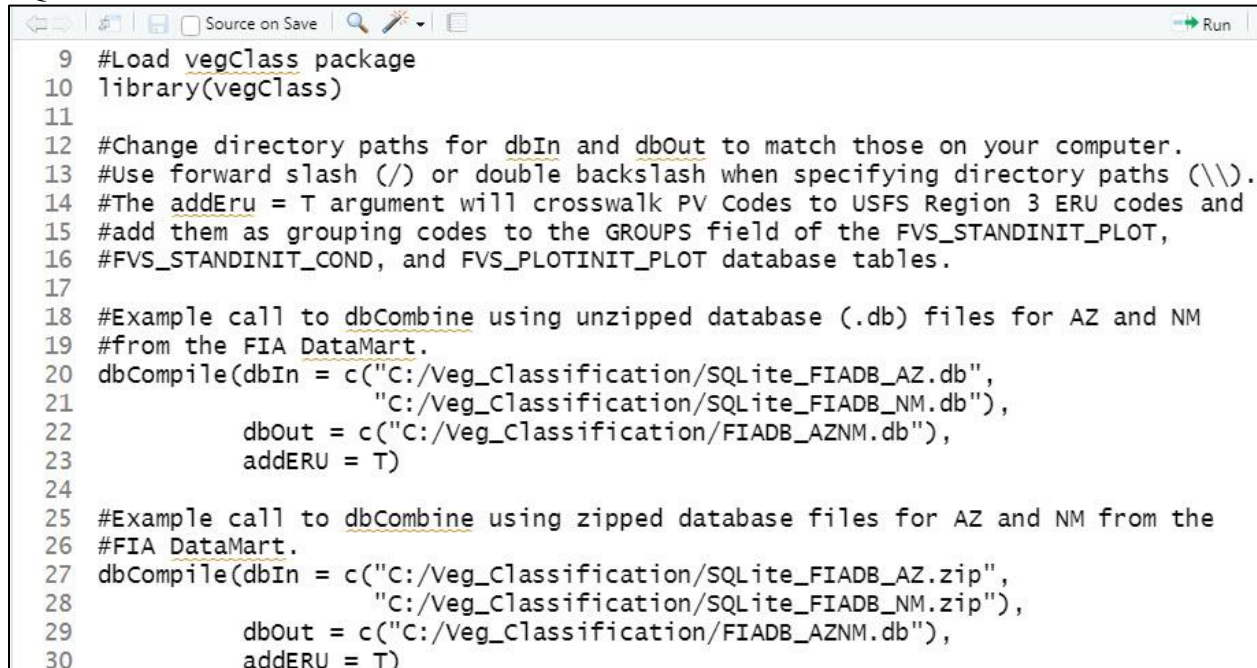
The state FIA databases come with all the tables described in the FIADB databases documentation in addition to the FVS database tables: https://www.fia.fs.usda.gov/library/database-documentation/index.php#FIADB. As such, the state databases are quite large (gigabytes in size) and contain a lot of information that the FVS system does not use. The large size of the databases can be cumbersome in the FVS interface and users may be interested in having a database that only contains the FVS-ready FIA database tables. The dbCompile function (6.0 Functions Available to User in vegClass R package) in the vegClass package can be used to combine FVS-ready FIA data from multiple states and create a database that contains the FVS-ready databases tables and/or a subset of the FIA database tables. The dbCompile function can also be used on a single FIA database to create an FVS-ready database which only incudes the FVS database tables. The Data_Combine_Example.R script (Figure 31) shows an example of how the FVS-ready FIA data for Arizona and New Mexico can be combined into a single database that only includes the FVS-ready databases tables. The Data_Combine_Example.R script can be used as a template for users to create their own custom FVS-ready FIA database. The following steps can be used to create a custom database:

1. Change the directory paths and file names in dbIn argument to those specific to your computer and FVS-ready FIA databases being used.

2. Change the directory path and file name in dbOut argument to one specific to your computer

3. If you are a user from USFS Region 3, you can add ERU grouping codes to your FVS-ready FIA database tables by setting the addERU argument to TRUE (T).

30

4. Run the code in the script. To invoke the dbCompile function, the library(vegClass) command must be run and all lines code from dbCompile to the closing parentheses in the dbCompile function call must be selected and run.

Figure 31. Example script illustrating how to combine FVS-ready database tables into a single SQLite database for Arizona and New Mexico data.

```
 9  #Load vegClass package
10  library(vegClass)
11
12  #Change directory paths for dbIn and dbOut to match those on your computer.
13  #Use forward slash (/) or double backslash when specifying directory paths (\\).
14  #The addEru = T argument will crosswalk PV Codes to USFS Region 3 ERU codes and
15  #add them as grouping codes to the GROUPS field of the FVS_STANDINIT_PLOT,
16  #FVS_STANDINIT_COND, and FVS_PLOTINIT_PLOT database tables.
17
18  #Example call to dbCombine using unzipped database (.db) files for AZ and NM
19  #from the FIA DataMart.
20  dbCompile(dbIn = c("C:/Veg_Classification/SQLite_FIADB_AZ.db",
21                     "C:/Veg_Classification/SQLite_FIADB_NM.db"),
22            dbOut = c("C:/Veg_Classification/FIADB_AZNM.db"),
23            addERU = T)
24
25  #Example call to dbCombine using zipped database files for AZ and NM from the
26  #FIA DataMart.
27  dbCompile(dbIn = c("C:/Veg_Classification/SQLite_FIADB_AZ.zip",
28                     "C:/Veg_Classification/SQLite_FIADB_NM.zip"),
29            dbOut = c("C:/Veg_Classification/FIADB_AZNM.db"),
30            addERU = T)
```

4.3.3 Setting up FVS Simulations

This section of the guide details how to setup projects in FVS and describes how output from FVS simulations is stored. It is assumed that users understand how to construct and run FVS simulations. Resources for learning more about the FVS software can be found on the Forest Vegetation Simulator training webpage: https://www.fs.usda.gov/fvs/training/index.shtml and user guides webpage: https://www.fs.usda.gov/fvs/documents/guides.shtml. A step-by-step FVS training guide (FVSTrainingGuide.pdf) can also be found in the Training Guides folder that comes installed with the FVS software (C:\FVS\Training Guides).

After FVS is installed, users should create a new FVS project. A project is simply a workspace for creating FVS runs that are linked to a single input database. The steps for creating a project are described in the *Creating an FVS project* section of the FVS training guide. Users can create as many projects as they would like for their work. Once a project has been created, the user should upload the inventory database that they want to produce vegetation classification information for. The steps for uploading an inventory database are presented in the *Replacing the training database* section of the FVS training guide.

Once a FVS project has been created and an FVS-ready input database has been uploaded, the user can begin to create FVS simulations and produce simulation output. There are no restrictions on what keywords and components can be included in simulations. However, certain output must be generated in simulations to produce vegetation classification output with the vegClass R package. The FVS_Cases and FVS_Treelist (western FVS variants) or

FVS_Treelist_East (eastern FVS variants) tables output database tables are required. By default, the FVS_Cases table is generated for every FVS simulation that is run. The FVS_TreeList and FVS_TreeList_East table can be created by selecting the **Tree lists** options in the **Select Outputs** tab (Figure 32). Users can also include both the TREELIST (Figure 33) and TREELIDB (Figure 34) keywords from the **Keywords** tab in a simulation to produce the FVS_Treelist. These keywords can be specified from the component windows in the FVS interface or included in a keyword component file (.kcp). Further information on the TREELIST and TREELIDB keywords can be found in the FVS keyword reference guide: https://www.fs.usda.gov/fmsc/ftp/fvs/docs/gtr/keyword.pdf and DBS user guide: https://www.fs.usda.gov/fmsc/ftp/fvs/docs/gtr/DBSUserGuide.pdf respectively.

Figure 32. Selecting **Tree lists** output in **Select Outputs** tab in FVS interface.

Figure 33. Specifying TREELIST keyword from **Keywords** tab in FVS interface.

Figure 34. Selecting TREELIDB keyword from **Keywords** tab in FVS interface.

Simulation output is stored in an SQLite database entitled FVSOut.db that is specific to the project the user is working from. By default, FVSOut.db files are stored in project folders that are in the FVS folder created from the FVS installation. Project folders share the same name as the project being assumed in the FVS interface. The screenshot in Figure 35 shows an example FVSOut.db file that is stored in the Project_1, project folder.

Figure 35. Image of output FVS database (FVSOut.db) in Project_1 project folder.

When using the local configuration of FVS, users can make a copy of the FVSOut.db database by right clicking, selecting the **Copy** option, and then use the **Paste** option to store the database in a desired directory. In both the local and online configuration of FVS, the FVSOut.db can be downloaded and saved to a desired directory from within in the FVS interface as shown in Figure 36.

33

Figure 36. Image showing button used to download output FVS database in **Downloads** tab of FVS interface.
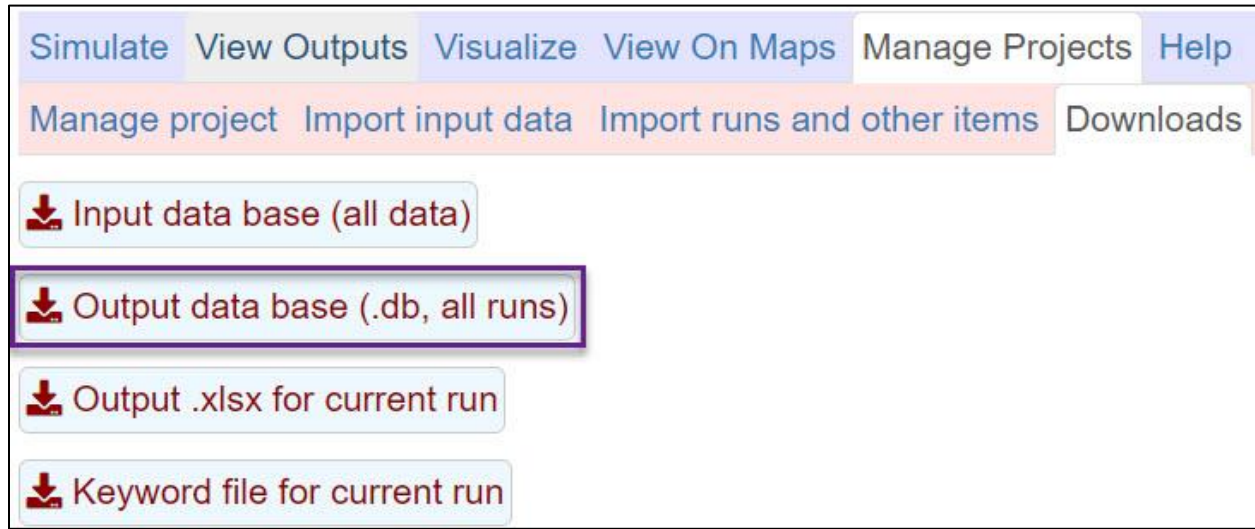
Information from the FVS_Compute, FVS_Potfire/FVS_Potfire_East, FVS_Fuels, and FVS_Carbon database tables can be joined to the default information produced by the vegClass R package. By default, the FVS_Compute database table will be produced if a simulation run from the FVS interface includes the use of the Event Monitor to report a user defined variable(s). If a user is running simulations with rFVS or FVS through the command line, then the COMPUTDB keyword should be used to create the FVS_Compute table in the output database. More information on the COMPUTBD keyword can be found in the DBS user guide: https://www.fs.usda.gov/fmsc/ftp/fvs/docs/gtr/DBSUserGuide.pdf. Users can request the FVS_Carbon, FVS_Fuels, and FVS_Potfire/FVS_Potfire_East database tables be produced in the output database by selecting the **Carbon and fuels** and **Fire and mortality** outputs in the **Select Outputs** tab (Figure 37). It is worth noting that selecting these options will have additional database tables be produced if a fire is scheduled in a simulation. Keywords can be directly specified to have only the FVS_Carbon, FVS_Fuels, and FVS_Potfire/FVS_Potfire_East table be produced in any circumstance. The POTFIRE and POTFIRDB keywords together can be used to produce the FVS_Potfire/FVS_Potfire_East database table, the CARBREDB can be used to produce the FVS_Carbon database table, and the FUELOUT and FUELSOUT keywords together can be used to produce the FVS_Fuels database table. Like the TREELIST and TREEIDB keywords, the POTIFRE, POTFIDB, CARBREDB, FUELOUT, and FUELSOUT keywords can be specified through components windows in the FVS interface or through kcp files. More information on the POTFIRE and FUELOUT keywords can be found in The Fire and Fuels Extension to the Forest Vegetation Simulator guide: https://www.fs.usda.gov/fmsc/ftp/fvs/docs/gtr/FFEguide.pdf and further information on the POTFIRDB, CARBREDB, and FUELSOUT keywords can be found in the DBS user guide: https://www.fs.usda.gov/fmsc/ftp/fvs/docs/gtr/DBSUserGuide.pdf.

Figure 37. Selecting **Carbon and fuels** and **Fire and mortality** outputs in **Select Outputs** tab.in FVS interface.

4.4 Using the vegClass Package

Once the desired number of FVS simulations have been run (refer to section 4.3.3), the vegClass R package will be used to derive vegetation classification output. Users should invoke the vegClass package through a R script (.R file). First time users should refer to the example script, vegClass_Example.R, located in the vegClassInstall folder. The vegClass_Example.R file can be opened in either RStudio or RGui applications. Instructions for opening preexisting scripts can be found in section 4.2.1 (RGui) and 4.2.3 (RStudio). The components in the vegClass_Example.R script are described below.

All the lines that start with the # symbol are comments for the user and are not interpreted by R. The first line of code library(vegClass) is used to attach the vegClass package to the current R session (Figure 38). This line of code needs to be included in any R script that a user creates when the invocation of the vegClass package is desired.

Vegetation classification output is created by calling function main (Figure 38). There are several arguments used in this function that should be considered, and each are described below (further discussion of the main function and all argument is provided in 6.0 Functions Available to User in vegClass R package). The **input** argument should be a directory path and file name to an output FVS SQLite database. The main function will use the information in this database to produce vegetation classifications and other stand attributes. The **output** argument is a directory path and file name to a comma separated values file (.csv). The information produced by the main function is sent to this file. The **runTitles** argument is a character vector of FVS run titles that will be processed by the main function. The run titles specified in this argument correspond to run titles that exist in the FVS project and output database you are working with. Individual and multiple runs can be processed by a single call to the main function. The **region** argument is an integer value corresponding to a USFS region associated with the runs being processed in the main function. This argument is currently a dummy variable and does not affect the default variables returned from the main function. The value specified in the region argument will apply to all runs specified in the **runTitles** argument or all runs being processed if the **allRuns** argument is set to TRUE (T). The **addCompute**, **addPotFire**, **addFuels**, and **addCarbon** arguments can be used to add additional information to the standard output that is produced by the main function (Table 1). If these arguments are set to TRUE (T), then variables from the FVS_Compute, FVS_Potfire, FVS_Fuels, and FVS_Carbon tables will be joined to the output. If the FVS_Compute, FVS_Potfire, FVS_Fuels, and FVS_Carbon, tables do not exist in the

database specified in the input argument when the corresponding **addCompute**, **addPotFire**, **addFuels**, and **addCarbon** arguments are TRUE (T), then the main function will stop with an error message. The **addVolume** argument specifies whether calculations of total cubic foot volume (merchantable cubic foot volume for eastern FVS variants), merchantable cubic foot volume (sawlog cubic foot volume for eastern FVS variants), and merchantable board foot volume (sawlog board foot volume for eastern FVS variants) should be included in the file specified in the **output** argument. The **vol1DBH**, **vol2DBH**, and **vol3DBH**, arguments determine what tree sizes based on DBH should be included in the volume calculations when **addVolume** is TRUE (T). The **startYear** and **endYear** arguments specify the first and last years that output from the main function should be produced in the file specified in the **output** argument. The **startCycle** and **endCycle** arguments specify the first and last cycle numbers that output from the main function should be produced in the file specified in the **output** argument. The **setInidices** argument creates CaseID indices in FVS database tables found in **input** argument, significantly increase the speed of SQL queries executed in the main function. The **modstandID** argument appends an underscore before each Stand ID sent to the output argument, forcing Microsoft Excel to recognize that Stand ID is a character and avoids the problem of long character strings of numbers (i.e. Stand IDs in FIA data: 0004201904090101990050) being truncated and converted to numbers. The **invDB** argument specifies the location of the inventory database used for the FVS runs specified in the runTitles argument, or all runs being processed if the allRuns argument is set to TRUE (T). This is for obtaining the "PV_CODE" value associated with each stand being processed. The **invStandTbl** argument specifies stand data table found within the SQLite database (*.db extension) specified in the invDB argument. Finally, the suctomVars argument specifies the location of the Microsoft Excel file that contains the specifications for custom variable calculations.

In the example below, all information from 2022 and onward will be reported in the FVSOut.csv specified in the **output** argument, and the custom variable XLXS file location is specified.

Figure 38. Image of vegClass_Example R script in RStudio.

```r
17
18  #Attach vegClass package
19  library(vegClass)
20
21  #Example call to main function in vegClass package
22  main(input = "C:/FVS/VegClass Example/FVSOut.db",
23       output = "C:/Veg_Classification/vegClassOut.csv",
24       runTitles = c("MCD Run"),
25       region = 3,
26       addCompute = T,
27       addPotFire = T,
28       addFuels = T,
29       addCarbon = T,
30       addVolume = T,
31       vol1DBH = 5,
32       vol2DBH = 5,
33       vol3DBH = 9,
34       startYear = 2022,
35       customVars = "C:/Veg_Classification/CustomVars_vegClass.xlsx")
36
37  #You can call the main function as many times as you want in an R script. You
38  #can simply copy and paste the code above and change the function arguments as
39  #needed.
```

4.4.1 Requesting Custom Variables

In addition to the suite of variables that are output from the vegClass R package, custom variables for 11 different attributes can be calculated by species/species group, and by custom diameter and height ranges. Included in the **vegClassInstall folder** (see Section 2.3) is a **CustomVars_vegClass.xlsx** file. This template provides instructions on how to fill it out on the "!ReadMe!" sheet. Once filled out, make sure to specify its location in the **customVars** argument of the main function.

4.4.2 Running vegClass_Example.R
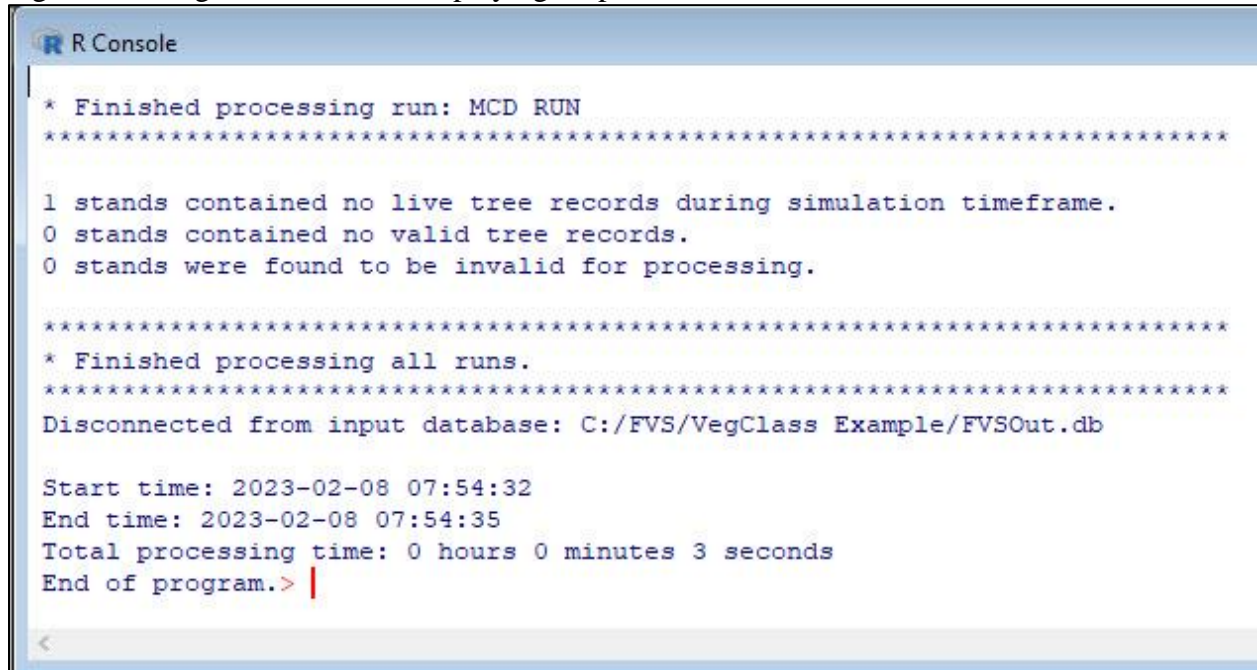
The following steps can be used to run the vegClass_Example.R script in either the RGui or RStudio application.

1. Change the directory path for the input and output arguments to ones relevant on your computer. Either forward slashes (/) or double back slashes (\\) can be used to specify directory paths.

2. Select the desired lines of code to run. To invoke main function, the library(vegClass) command must be run and all lines code from main to the closing parentheses in the main function call must be selected and run. Instructions on how to run code are provided in Sections 4.2.2 (RGui) and 4.2.4 (RStudio).

3. Wait for the main function to finish running. Do NOT close RGui/RStudio when the main function is running, doing so will lead to incomplete output. When the main function is called, messages will be printed to the **R Console** window (RGui; Figure 39) or **Console** pane (RStudio; Figure 40). These messages report the runs and stands that have been processed by the main function and primarily serve as mechanism to inform the user that the main function is running. "End of program." will be printed in the **R**

Console window (RGui; Figure 39) or **Console** pane (RStudio; Figure 40) once the main function has finished processing.

Figure 39. Image of **R Console** displaying output from main function in RGui.
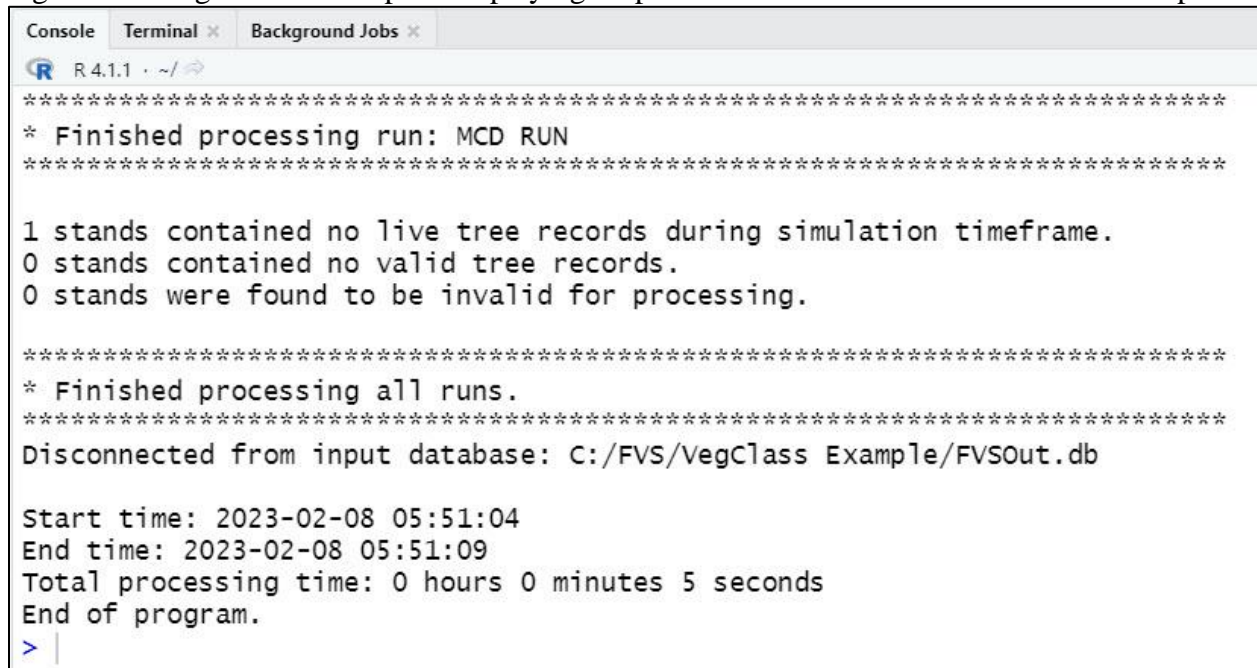
Figure 40. Image of **Console** pane displaying output from main function in RStudio Desktop.

5.0 Standard Output Produced by vegClass R Package.

This section of the guide provides a description of the information that is produced by the main function in the vegClass R package. Table 1 lists the name and definition of each variable that can be produced from the main function. Some variables are always calculated and reported from the main function and others are only reported when a specific value is assigned in the region argument of the main function. Currently, region specific variables and rulesets are only available to USFS regions 1, 3, and 8.

Table 1. Vegetation classification variables reported by vegClass package.

Variable Name	Description
RUNTITLE	Name of the FVS run title
CASEID	Case ID associated with STANDID
STANDID	Plot/Stand/Point Identification Number
VARIANT	FVS variant code associated with FVS run title (RUNTITLE).
REGION	USFS region number.
YEAR	FVS projection year
CY	FVS projection cycle
CAN_COV	Canopy cover corrected for overlap (including seedlings and stems)
BA	Basal area per acre (including seedlings and stems)
TPA	Trees per acre (including seedlings and stems)
QMD	Quadratic mean diameter (including seedlings and stems)
ZSDI	Zeide SDI (including seedlings and stems)
RSDI	Reineke SDI (including seedlings and stems)
BA_STM	Basal area per acre (trees ≥ 1" diameter)
TPA_STM	Trees per acre (trees ≥ 1" diameter)
QMD_STM	Quadratic mean diameter (trees ≥ 1" diameter)
ZSDI_STM	Zeide SDI (trees ≥ 1" diameter)
RSDI_STM	Reineke SDI (trees ≥ 1" diameter)
BA_WT_DIA	Basal area weighted diameter (including seedlings and stems)
BA_WT_HT	Basal area weighted height (including seedlings and stems)
QMD_TOP20	Quadratic-Mean-Diameter: Top 20% by diameter (exclude seedlings unless percent canopy cover less than 10%)
DOM_TYPE	Dominance type (R3 ruleset)
DCC1	Primary attribute in DOM_TYPE (R3 ruleset)
XDCC1	Canopy cover corrected for overlap represented by DCC1 (R3 ruleset)
DCC2	Secondary attribute in DOM_TYPE (R3 ruleset)
XDCC2	Canopy cover corrected for overlap represented by DCC2 (R3 ruleset)
CAN_SIZCL	Canopy cover dominant size class: mid-scale mapping (R3 ruleset)
CAN_SZTMB	Canopy cover dominant size class: timberland types (R3 ruleset)
CAN_SZWDL	Canopy cover dominant size class: woodland types (R3 ruleset)
BA_STORY	Canopy Layers/Stories: R3 ruleset; basal area per 8" sliding diameter range (R3 ruleset)
VEGTYPE	STSIM potential vegetation type & cover type grouping (R1 ruleset)
COVERTYPE_R1	Cover Type (R1 ruleset)

VERTICAL STRUCTURE	Number of Canopy layers (R1 ruleset)
SIZECLASS_NTG	Basal area weighted diameter size class (R1 ruleset)
STRCLSSTR	Size & density structure class strata (R1 ruleset)
SSSIZE	Trees per acre weighted average height for advance regeneration class (R8 ruleset)
SSTPA	Trees per acre for advance regeneration class (R8 ruleset)
NMSIZE	Basal area weighted diameter for non-merchantable size class (R8 ruleset)
NMBA	Basal area of non-merchantable size class (R8 ruleset)
PWSIZE	Basal area weighted diameter for pulpwood size class (R8 ruleset)
PWBA	Basal area of pulpwood size class (R8 ruleset)
STSIZE	Basal area weighted diameter for saw timber size class (R8 ruleset)
STBA	Basal area of saw timber size class (R8 ruleset)
SSDOMSPP	Dominance type of advance regeneration size class (R8 ruleset)
NMDOMSPP	Dominance type of non-merchantable size class (R8 ruleset)
PWDOMSPP	Dominance type of pulpwood size class (R8 ruleset)
STDOMSPP	Dominance type of saw timber size class (R8 ruleset)
VEGCLASS	Density and size class combination (R8 ruleset)
VOL1	Total cubic foot volume (western variants) / merchantable cubic foot volume (eastern variants). This variable is only included in output if addVolume argument in main function is set to TRUE (T).
VOL2	Merchantable cubic foot volume (western variants) / sawlog cubic foot volume (eastern variants). This variable is only included in output if addVolume argument in main function is set to TRUE (T).
VOL3	Merchantable board foot volume (western variants) / sawlog board foot volume (eastern variants). This variable is only included in output if addVolume argument in main function is set to TRUE (T).
DEADVOL1	Total cubic foot volume (western variants) / merchantable cubic foot volume (eastern variants) that died in cycle. This variable is only included in output if addVolume argument in main function is set to TRUE (T).
DEADVOL2	Merchantable cubic foot volume (western variants) / sawlog cubic foot volume (eastern variants) that died in cycle. This variable is only included in output if addVolume argument in main function is set to TRUE (T).
DEADVOL3	Merchantable board foot volume (western variants) / sawlog board foot volume (eastern variants) that died in cycle. This variable is only included in output if addVolume argument in main function is set to TRUE (T).

6.0 Functions Available to User in vegClass R package

This section provides documentation for the functions available to the user in the vegClass package. Users will primarily be calling the following the functions from the vegClass package: dbCompile and main. The other functions listed in this section are called from within either the

main or dbCompile functions. All function descriptions from the vegClass package are presented if users wish to examine the source code or test individual functions. The source code for all functions can be found in the ForestVegetationSimulator-Utilities GitHub repository: https://github.com/USDAForestService/ForestVegetationSimulator-Utilities/tree/main/packages/vegClass/R.

dbCompile

Description

This function is used to read in database tables from input SQLite databases and write the database tables from each of these to a single output SQLite database. SQLite databases (.db) are the only compatible input database type that can be processed in this function. The primary purpose of this function is to combine input FVS databases into a single database or to extract FVS database tables and/or a subset of database tables from a larger database such as those on the FIA datamart.

Source Code

dbCompile function source code can be found here: https://github.com/USDAForestService/ForestVegetationSimulator-Utilities/blob/main/packages/vegClass/R/dataTools.R

Arguments

dbIn: Character vector of directory paths and file names for SQLite databases to process. Files can either be an SQLite database (.db) or zipped folder (.zip) which contains an SQLite database(s).

Examples of valid dbIn formats:
"C:/FIA2FVS_Databases/SQLite_FIADB_AZ/FIADB_AZ.db",
"C:\\FIA2FVS_Databases\\SQLite_FIADB_AZ\\FIADB_AZ.db"

"C:/FIA2FVS_Databases/SQLite_FIADB_AZ/ FIADB_AZ.zip",
"C:\\FIA2FVS_Databases\\SQLite_FIADB_AZ\\ FIADB_AZ.zip "

NOTE: The contents of any zipped file specified in dbIn will be unzipped to a temporary folder called xxxvegClassdbCompileUnzipxxx in your working directory. The dbCompile function will delete this folder and all contents within before the function completes processing. If the function fails to delete this folder, a message will be displayed in the **R Console** (Rgui) or **Console** (Rstudio) indicating that this folder was not deleted.

dbOut: Character string corresponding to SQLite database to write out to.

Examples of valid dbOut formats:

"C:/FIA2FVS_Databases/FVS_Data.db",
"C:\\FIA2FVS_Databases\\FVS_Data.db"

dbTables: Character vector of database tables to process from argument dbIn. By default, this argument contains the following values:
"FVS_STANDINIT"
"FVS_TREEINIT"
"FVS_STANDINIT_PLOT"
"FVS_STANDINIT_COND"
"FVS_PLOTINIT_PLOT"
"FVS_TREEINIT_PLOT"
"FVS_TREEINIT_COND"

NOTE: The values specified in the dbTables argument do not have to be FVS related database tables. They can be any database table found within the databases specified in the dbIn argument.

buildGaak: Logical variable used to determine if FVS_GROUPADDFILESANDKEYWORDS will be written to dbOut. If TRUE (T), this table will be written to dbOut. By default, this argument is set to TRUE (T).

gaakType: Integer value from 1 - 3 used to determine what kind of GAAK table will be written to dbOut if buildGaak is TRUE.
1: GAAK table with All_Stands and All_Plots grouping codes.

2: GAAK table with All_FIA_Conditions, All_FIA_Plots, All_FIA_Subplots grouping codes.

3: GAAK table with All_Stands, All_Plots, All_FIA_Conditions, All_FIA_Plots, All_FIA_Subplots grouping codes. For more information refer to fvsGaak function.

addEru: Logical variable used to determine if ERU should be added as a field in FVS_STANDINIT, FVS_PLOTINIT, FVS_STANDINIT_PLOT, FVS_STANDINIT_COND, and FVS_PLOTINIT_PLOT tables. In addition, ERU code will be added as a grouping code in the GROUPS field of these tables if addERU is TRUE (T). By default, this argument is set to TRUE (T).

deleteInput: Logical variable used to determine if values in dbIn should be deleted after dbCompile has been called. Be careful with this argument. The primary purpose of this argument is to conserve hard disk space for users who do not want the input databases specified in dbIn after the database in dbOut is created. By default, this argument is set to FALSE (F).

readChunks: Logical variable used to determine if data from database table should be read in chunks. In general, processing time of dbCompile increases but less RAM is used in R session if this argument is TRUE (T). By default, this argument is set to FALSE (F).

rowsToRead: Integer value corresponding to number of rows to read from a database table if readChunks is TRUE (T). By default, this argument is set to 5000.

Value

Message indicating that database has been created.

Examples

```
#Example of how to combine FIA data from AZ and NM and produce database with just
#FVS tables.
dbCompile(dbIn = c("C:/Veg_Classification/SQLite_FIADB_AZ/FIADB_AZ.db",
                   "C:/Veg_Classification/SQLite_FIADB_NM/FIADB_NM.db"),
          dbOut = "C:/Veg_Classification/FIADB_AZNM.db",
          dbTables = c("FVS_STANDINIT_PLOT",
                       "FVS_STANDINIT_COND",
                       "FVS_PLOTINIT_PLOT",
                       "FVS_TREEINIT_PLOT",
                       "FVS_TREEINIT_COND"),
          buildGaak = T,
          gaakType = 2)
```

main

Description

This function reads data from the FVS_Treelist/FVS_Treelist_East in an FVS output database to determine vegetation classifications and other attributes. The vegetation classifications and attributes are sent to a comma separated values file (.csv) specified in the call to the main function. Information from the FVS_Compute, FVS_Potfire/FVS_Potfire_East, FVS_Fuels, and FVS_Carbon database tables can be included in the output comma separated values file.

Source Code

main function source code can be found here:
https://github.com/USDAForestService/ForestVegetationSimulator-Utilities/blob/main/packages/vegClass/R/main.R

Arguments

Input: Character string for directory path and file name to an FVS output SQLite database (.db). Directory path and file name must be surrounded with double quotes "" and double back slashes or single forward slashes need to be used for

specifying directory paths. Input database must contain the following tables: FVS_TreeList and FVS_Cases or main function will terminate with an error. By default, this argument is set to NULL.

Examples of valid formatting for input argument:

"C:/FVS/R3_Work/FVSOut.db"
"C:\\FVS\\R3_Work\\FVSOut.db"

output: Character string for directory path and file name to output .csv. Directory path and file name must be surrounded with double quotes "" and double back slashes or single forward slashes need to be used for specifying directory paths. By default, this argument is set to NULL.

Examples of valid formatting for output argument:
"C:/FVS/R3_Work/FVSOut.csv"
"C:\\FVS\\R3_Work\\FVSOut.csv"

overwriteOut: Logical variable used to determine if output file should be overwritten. If value is TRUE (T), any information existing in output will be overwritten with new information. If value is FALSE (F) and the file in output argument exists, then main function will stop with an error message. The default value of this argument is FALSE (F).

runTitles: Character vector of FVS run titles that will be processed from database defined in input argument. Each run title in argument must be surrounded by double or single quotes. The values specified for runTitles argument are case insensitive but need to be spelled correctly. If any of the values in runTitles are spelled incorrectly, the execution of main function will terminate with error message. By default, this argument is set to NULL. If runTitles is left as NULL and allRuns argument is not TRUE (T), execution of main function will terminate with an error. By default, this argument is set to NULL.

Example of how to specify single run title:
runTitles = "Run 1"

Example of how to specify multiple run titles:
runTitles = c("Run 1", "Run 2",...) Note the use of the c(...) when specifying multiple run titles

allRuns: Logical variable that is used to determine if all runs in argument input should processed. If value is TRUE (T), then all runs will be processed and any runs specified in argument runTitles will be ignored. By default, this variable is set to FALSE (F).

startYear: Integer value corresponding to the year that data should start being reported in output argument. Data with years prior to this value will not be included in the output argument. By default,
this argument is set to NA.

endYear: Integer value corresponding to the last year that data should be reported in output argument. Data associated with years after this value will not be included in the output argument. By default, this value is set to NA. If this argument is left as NA, then all information after value in startYear argument will be sent to output argument.

startCycle: Integer value corresponding to the cycle that data should start being reported in output argument. Data with cycles prior to this value will not be included in the output argument. By default, this value is set to NA.

endCycle: Integer value corresponding to the last cycle that data should be reported in output argument. Data associated with cycles after this value will not be included in the output argument. By default, this value is set to NA. If this argument is left as NA, then all information after startCycle argument will be sent to output argument.

               NOTE: If both startYear and startCycle arguments do not have a value specified (they are left as NA), main function will stop with an error message. One of these arguments must be used. If non-NA values are specified for both startYear and startCycle, then the main function will default to using cycles for determining what information gets sent to output argument.

region: Integer variable corresponding to USFS region number. Valid values are 1, 3, and 8. This variable is used to determine what region-specific vegetation classifications and other attributes are reported and what rulesets are used to calculate these. By default, this argument is set to NA.

               NOTE: the value specified in the region argument will apply to all runs specified in the runTitles argument or all runs being processed if the allRuns argument is set to TRUE (T).

addCompute: Logical variable used to indicate if information in FVS_Compute table should be included in file specified in output argument. If the FVS_Compute table does not exist in the database specified in input argument and addCompute is TRUE (T), main function will stop with an error message. If the FVS_Compute table does exist in the database specified input argument but there is no information in the FVS_Compute table for a given run, then that run will have NA values reported in the file specified output argument for all variables found in the FVS_Compute table. By default, this argument is set to FALSE (F).

addPotFire: Logical variable used to indicate if information in FVS_Potfire or FVS_Potfire_East table should be included in the file specified output argument. If the FVS_Potfire table does not exist in the database specified in input argument and addPotFire is TRUE (T), main function will stop with an error message. If the FVS_Potfire or FVS_Potfire_East table do exist in the database specified in input argument but there is no information in the FVS_Potfire or FVS_Potfire_East table for a given run, then that run will have NA values reported in the file specified output for all variables extracted from the FVS_Potfire/FVS_Potfire_East table. By default, this argument is set to FALSE (F).

addFuels: Logical variable used to indicate if information in FVS_Fuels table should be included in file specified output argument. If the FVS_Fuels table does not exist in database specified in input argument and addFuels is TRUE (T), main function will stop with an error message. If the FVS_Fuels table does exist in database specified in input argument but there is no information in the FVS_Fuels table for a given run, then that run will have NA values reported in the file specified in output argument for all variables found in the FVS_Fuels table. By default, this argument is set to FALSE (F).

addCarbon: Logical variable used to indicate if information in FVS_Carbon table should be included in file specified in output argument. If the FVS_Carbon table does not exist in the database specified in input argument and addCarbon is TRUE (T), main function will stop with an error message. If the FVS_Carbon table does exist in the database specified in input argument but there is no information in the FVS_Carbon table for a given run, then that run will have NA values reported in the file specified in output argument for all variables found in the FVS_Carbon table. By default, this argument is set to FALSE (F).

addVolume: Logical variable used to indicate if 6 measures of volume should be calculated and reported. If the value of this argument is set to TRUE (T), then the following measures of volume will be calculated:

VOL1: Total cubic foot volume (western variants) / merchantable cubic foot volume (eastern variants).

VOL2: Merchantable cubic foot volume (western variants) / sawlog cubic foot volume (eastern variants).

VOL3: Board foot volume (western variants) / sawlog board foot volume (eastern variants).

DEADVOL1: Total cubic foot volume (western variants) / merchantable cubic foot volume (eastern variants) that died in that cycle.

DEADVOL2: Merchantable cubic foot volume (western variants) / sawlog cubic foot volume (eastern variants) that died in that cycle.

DEADVOL3: Board foot volume (western variants) / sawlog board foot volume (eastern variants) that died in that cycle.

By default, this argument is set to FALSE (F).

vol1DBH: Minimum DBH of tree records included in calculation of VOL1 and DEADVOL1 when addVolume is TRUE. By default, this argument is set to 0.

vol2DBH: Minimum DBH of tree records included in calculation of VOL2 and DEADVOL2 when addVolume is TRUE. By default, this argument is set to 5.

vol3DBH: Minimum DBH of tree records included in calculation of VOL3 and DEADVOL3 when addVolume is TRUE. By default, this argument is set to 9.

setIndices: Logical variable, where if TRUE (T), CaseID indices will be created in FVS database tables found in the input argument. These indices significantly increase the speed of the SQL queries executed in the main function. This argument SHOULD NOT be set to TRUE when users are processing an output database produced by the FVS graphical user interface (local or online configuration of FVS). This argument should only be set to TRUE (T) when a user produces an output database through rFVS or FVS run through the command line. If this argument is set to TRUE (T), the created CaseID indices will be removed from the database specified in the input argument before the main function returns. By default, this argument is set to FALSE (F).

modstandID: Logical variable, where if TRUE (T), an underscore will be appended before each Stand ID sent to output argument. The addition of the underscore forces Microsoft Excel to recognize that the Stand ID is a character and avoids the problem of long character strings of numbers (i.e. Stand IDs in FIA data: 00042019040901011990050) being truncated and converted to numbers. By default, this argument is set to TRUE (T).

InvDB: Character string corresponding to the full directory location of the inventory database that was used for the FVS runs specified in the runTitles argument, or all runs being processed if the allRuns argument is set to TRUE (T). Used to obtain the PV_CODE value from the stand data table for each stand. By default, this argument is set to NULL.

NOTE: Currently only required if the region argument is set to 1. If this argument is populated, then the below invStandTbl argument needs to be populated as well.

InvStandTbl: Character string corresponding to name of the stand data table
in InvDB that contains the PV_CODE values for the stands in the
runs specified in the runTitles argument, or all runs being
processed if the allRuns argument is set to TRUE (T). By default,
this argument is set to NULL.

NOTE: Currently only required if the region argument is set to 1,
and if the above InvDB argument is populated.

customVars: Character string corresponding to the full directory location of
the CustomVars_vegClass.XLSX file with the requested custom
variable specifications. By default, this argument is set to NULL.

Value

0 value invisibly returned.

Examples

```
#This example will produce vegetation classification output for MCD Run and
#MEW Run from the FVS output database: "C:/FVS/VegClass Example/FVSOut.db".
#The vegetation classification output will be sent to the vegClassOut.csv
#specified in the output argument. Information from the FVS_Compute,
#FVS_PotFire, FVS_Fuels, and FVS_Carbon table will be joined to the
#vegetation classification output. Volume variables (VOL1 - VOL3 and
#DEADVOL1 - DEADVOL3) will also be calculated and sent to vegClassOut.csv.
#Output from the year 2022 and onward will be available in the
#vegClassOut.csv as specified by the startYear argument.

main(input = "C:/FVS/VegClass Example/FVSOut.db",
     output = "C:/Veg_Classification/vegClassOut.csv",
     runTitles = c("MCD Run", "Mew Run"),
     region = 3,
     addCompute = T,
     addPotFire = T,
     addFuels = T,
     addCarbon = T,
     addVolume = T,
     vol1DBH = 5,
     vol2DBH = 5,
     vol3DBH = 9,
     startYear = 2023)
```

7.0 References

Müller K, Wickham H, James DA, Falcon S (2022). *RSQLite: SQLite Interface for R*.
https://rsqlite.r-dbi.org, https://github.com/r-dbi/RSQLite.

Vandendriesche 2013. A Compendium of NFS Regional Vegetation Classification Algorithms. Internal Rep. Fort Collins, CO: U. S. Department of Agriculture, Forest Service, Forest Management Service Center. 75p.