

BioSense Platform User Manual for RStudio Workbench

August 2022




**Centers for Disease
Control and Prevention**
Center for Surveillance, Epidemiology,
and Laboratory Services

Technical Assistance: support.syndromicsurveillance.org

The National Syndromic Surveillance Program (NSSP) promotes, and advances development of the cloud-based BioSense Platform, a secure integrated electronic health information system that hosts standardized analytic tools and facilitates collaborative processes. The BioSense Platform is a product of the Centers for Disease Control and Prevention (CDC).

Exit Notification and Disclaimer Policy

Links with this icon  indicate that you are leaving the CDC website.

- CDC cannot attest to the accuracy of a non-federal website.
- Linking to a non-federal website does not constitute an endorsement by CDC or any of its employees of the sponsors or information or products presented on the website.
- You will be subject to the destination website's privacy policy when you follow the link.
- CDC is not responsible for Section 508 compliance (accessibility) on other federal or private websites.

For more information on CDC's web notification policies, see [Website Disclaimers](#).

BioSense Platform User Manual for RStudio Workbench

formerly RStudio Server Pro

August 2022

Produced by

Division of Health Informatics and Surveillance
Center for Surveillance, Epidemiology, and Laboratory Services
Centers for Disease Control and Prevention

Revision History

Date	Revisions
August 2022	<ul style="list-style-type: none">■ Expands content and converts from Quick Start Guide to User Manual format■ Adds new, refreshed graphics to comply with current RStudio Workbench version (formerly RStudio Server Pro)■ Adds description of new Named License Policy and instructions for applying for a license■ Updates technical content:<ul style="list-style-type: none">○ Introduces Rnssp R Package for secure access to ESSENCE by using API Calls○ Expands Shared Project procedure○ Adds reference to R language version 4■ Adds new appendixes (B, C, D) to cover <i>System Library</i> for R Packages, Disk Space allotments, and best practices
November 2020	<ul style="list-style-type: none">■ Removes references to retired Adminer tool

Contents

1. Overview, 1

- Accessing RStudio, 1
- Logging in to RStudio, 2
- Changing your RStudio Password, 2

2. Basic Navigation, 3

- Overview, 3
 - Home Page: Top Menu, 3*
 - Home Page: Full View, 3*
 - Database Tables, 6*

3. Features, 9

- Project Sharing, 9
 - Steps for Creating a Shared Project in RStudio, 12*
- Multiple Versions of R, 14
- Multiple R Sessions, 14
- R Packages for Use in RStudio, 16
- Currently Installed R Packages for All Users, 17
- Requesting New Packages be Installed in the *System Library*, 18

4. How to Connect to Datamart, 19

- Securing Your Credentials, 20
 - The Rnssp Package, 20*
 - The Keyring Package, 21*

5. Using RStudio with ESSENCE APIs, 23

- Setting Up to Call the ESSENCE APIs, 23
- Setting up Rnssp, 24
- Setting up Keyring, 24
- Completing Setup to Use the ESSENCE APIs, 25
- Calling the ESSENCE APIs, 25
- Time Series Data Table (“timeseries” API), 26
- Time Series Graph from ESSENCE (“timeSeries graph” API), 27
- Table Builder Results (“tableBuilder” API), 28
- Data Details (dataDetails API), 30
- Summary Stats (summaryData API), 33
- Alert List Detection Table (regionSyndromeAlerts API and hospitalSyndromeAlerts), 34
- Tips and Tricks—ESSENCE API URL Length, 36
- Additional Resources, 36

Appendix A. Indexes Available on Datamart, 37

Appendix B. Installed R Packages for R Version 4, 39

Queries to Extract R Package Names in the *System Library* and Your Personal *User Library*, 39

Extract a List of R Packages Available to Your R Session, 39

Extract a List of R Packages Available in System Library, 39

Extract a List of R Packages Available in Your User Library, 39

Appendix C. Drive Space Allocation Rules for RStudio and SAS, 41

File Storage, 41

Drive Space, 41

Requesting More Storage Space, 42

Appendix D. Best Practices, 43

1. Overview

With the latest release of RStudio Server Pro, the RStudio Corporation has renamed it RStudio Workbench. The new name will be used in this manual to familiarize users with the terminology, but new features introduced in the Workbench integrated development environment (IDE) will NOT be addressed here. The NSSP team will test and evaluate the new features before adding them.

Numerous online resources are available to help you learn about and become comfortable with RStudio and the RStudio Workbench IDE. We suggest the RStudio Education website (education.rstudio.com) as a starting point. For more ideas, see typical search terms at [Additional Resources](#) in this document.

RStudio Workbench (RStudio) lets you access and analyze SQL data stored on the BioSense Platform. You can use RStudio to verify your data within the BioSense Platform archive (raw, processed, and exceptions tables), confirm information in your Master Facility Table (MFT), and extract data directly from ESSENCE using an ESSENCE application programming interface (API). RStudio also lets you access the BioSense Platform database that contains views into your site's data.

Accessing RStudio

RStudio and other analytical tools use your Access & Management Center (AMC) credentials to validate access.

To use RStudio Workbench, you must have an assigned license. These are assigned by the NSSP service team on a first-come, first-served basis from a limited pool of licenses. Only site administrators may request licenses for their users, so if you wish to use the BioSense Platform RStudio Workbench and do not currently have an assigned license, please discuss your planned usage with your site administrator. The site administrator can submit a service request to obtain a license for you.

You do NOT have an assigned license if you log in to RStudio and see this message:
*There are more concurrent users of RStudio Workbench than your license supports.
Please obtain an updated license to continue using the product.*

If no licenses are currently available, you will be assigned to the RStudio License Wait List. When a license becomes available, you will be contacted and asked if you want it. **Note:** Licenses that have not been used for ninety (90) days will be revoked and will be reassigned, if needed.

Once you have been assigned a license, RStudio Workbench can be launched by using the link on your AMC home page or by entering its direct URL in your browser. Its direct link is (rstudiopwd.syndromicsurveillance.org/rstudio2/auth-sign-in).

Note: You will be required to enter your AMC username and password to log in to RStudio Workbench.

If RStudio is not available on the user’s AMC home page, the site administrator should take the following steps:

1. Evaluate the user’s proposed need for RStudio. (Does this need warrant reassigning a license currently assigned to another user?)
2. Contact the NSSP support team (submit a ticket) to request a RStudio license:
<http://support.syndromicsurveillance.org>.
3. Open the user’s AMC profile and check the **Datamart (Site Level Access)** checkbox.

Logging in to RStudio

1. Go directly to RStudio at rstudiopwd.syndromicsurveillance.org/rstudio2/auth-sign-in or log in to the BioSense Platform AMC at amc.syndromicsurveillance.org.
2. If on the AMC home page, click the **RStudio link** to open the RStudio login page in a new browser tab (Figure 1).
3. Enter your BioSense Platform/AMC Username.
4. Enter your BioSense Platform/AMC Password, then click **Sign In**.

Your RStudio username and password are the same as the AMC/ESSENCE username and password. Enter these in the RStudio Login screen (Figure 1).

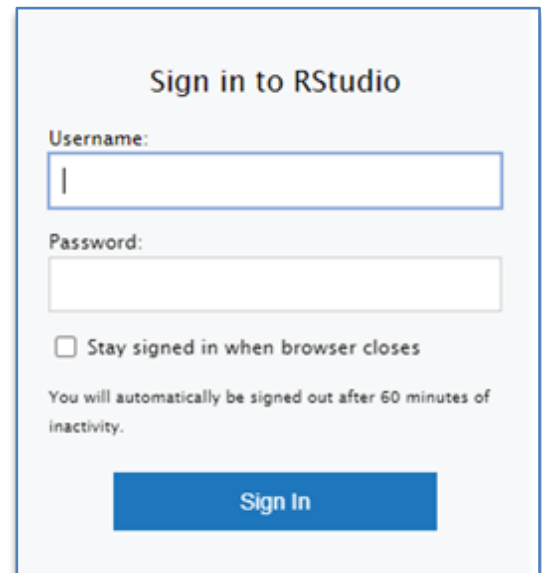


Figure 1. RStudio Login Screen

Changing your RStudio Password

AMC handles User ID and Password for access to RStudio (also ESSENCE and SAS) using Windows Active Directory. You cannot change your password for RStudio only. Your AMC password must be reset every 90 days, and you will receive emails from AMC prior to expiration. Follow the link in the email—or, you can reset your password any time at amc.syndromicsurveillance.org.

Note: If your password expires, you will not be able to log into AMC, RStudio, or other tools. Resetting your password in the AMC will restore all access after a short delay. Click the **Reset** link beside “Forgot Password?” on the AMC Login Screen and follow the directions you receive by email.

2. Basic Navigation

Overview

Numerous online resources are available to help you learn about and become comfortable with RStudio Workbench. We suggest the RStudio Education website for good starting points. Also, see typical search terms at [Additional Resources](#) in this document for more ideas.

Home Page: Top Menu

The home page consists of a top menu, a small toolbar, and 4 workspace panes. Figure 2 shows the top menu and toolbar with annotated icons.

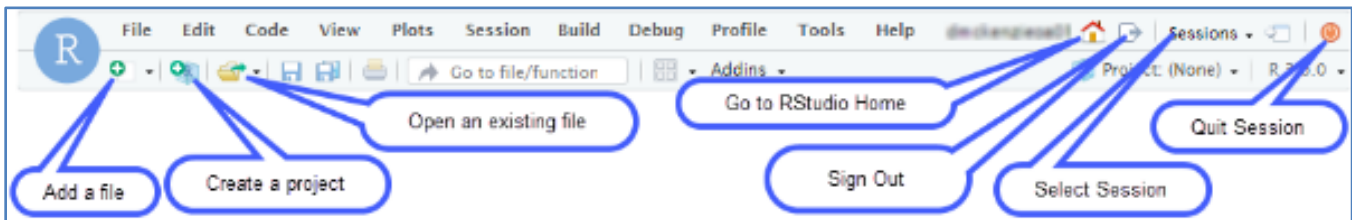


Figure 2. Top Menu and Toolbar for RStudio Home Page

Home Page: Full View

There are four workspace panes described in this document (e.g., “Pane 4 in the lower right” refers to the pane with the Files/Plots/Packages/Help/Viewer tabs as indicated in Figure 3).

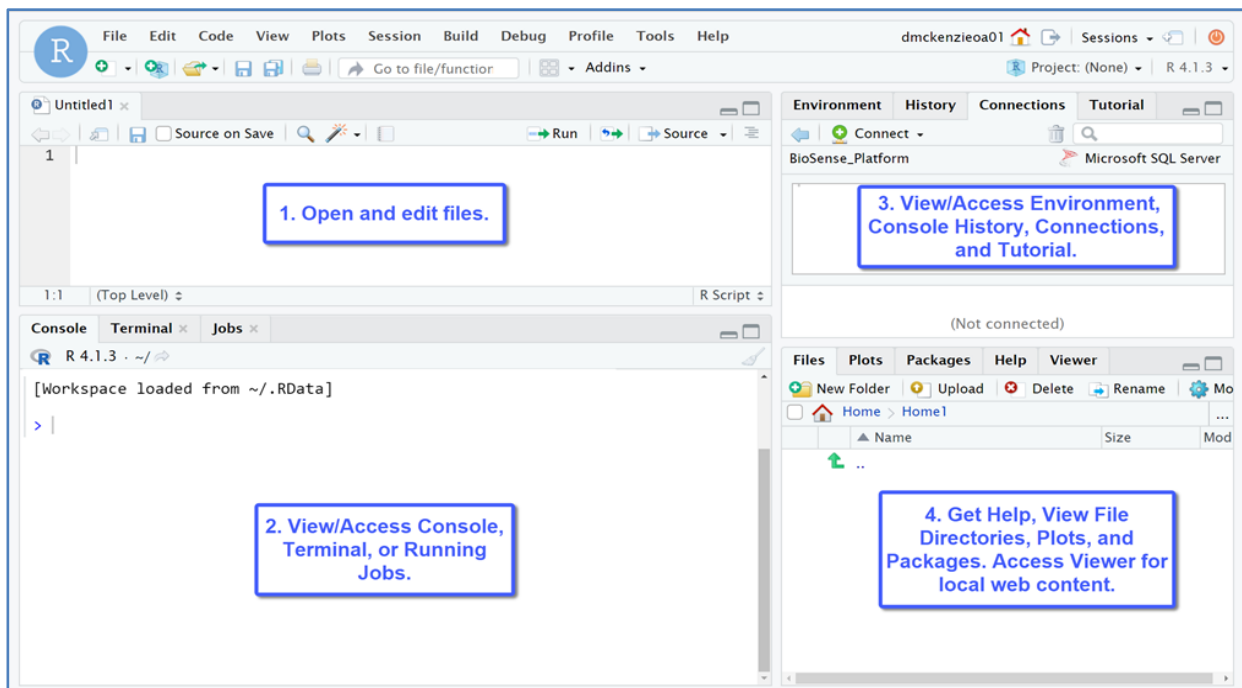


Figure 3. RStudio Navigation Guide

1. Pane 1 in the upper left part of the screen in Figure 3 is a workspace in which you can create a new file and edit it or edit an existing file (such as an R script, snippet, or macro). For example, to create a new R script, select **File** from the top menu, hover on **New File**, then click **R Script**. **Note:** The shortcut key combination, Ctrl+Alt+Shift+N can be used to directly open a new RStudio script file.

Numerous types of files can be created by selecting a type from the drop-down list (Figure 4). Expand the drop-down list by clicking the **green plus (+)** sign icon under File or click on the **arrow** beside the green plus. The drop-down list is automatically displayed when you select **File** then hover over New File.

2. Pane 2 in the lower left part of the screen shows the Console/Terminal/Jobs tabs.
 - You may type R commands into the **Console tab** to immediately execute and see output or logs from a program or command previously run. **Note:** The Console may be cleared by typing Ctrl+I (lower case L).
 - The **Terminal tab** provides access to the operating system shell from within RStudio.
 - The **Jobs tab** is used to run and manage background jobs.
3. Pane 3 in the upper right part of the screen shows the Environment, (Console) History, Connections, and Tutorial tabs.
 - The **Environment tab** lists all active objects, values, functions, or anything else you've created.
 - The **History tab** keeps a record of previous console commands.
 - The **Connections tab** makes it possible to easily connect to different data sources and explore the objects and data inside those connections.
 - The **Tutorial tab** provides access to various tutorials, such as Data basics, Filter observations, and Summarize Tables. **Note:** If the *learnr* package is not already loaded, you will be asked to load it when this tab is opened.

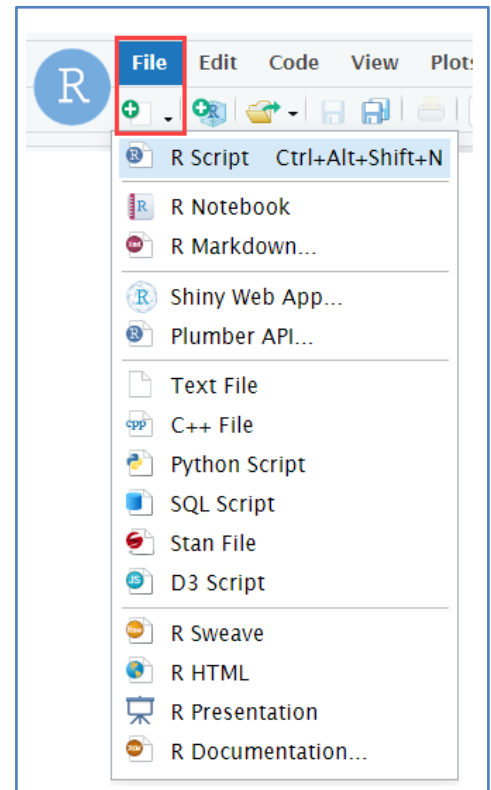


Figure 4. Create a New File (Choose Type)

4. Pane 4 is in the bottom right part of the screen and contains five tabs: **Files tab**, **Plots tab**, **Packages tab**, **Help tab**, and **Viewer tab** (Figure 5).

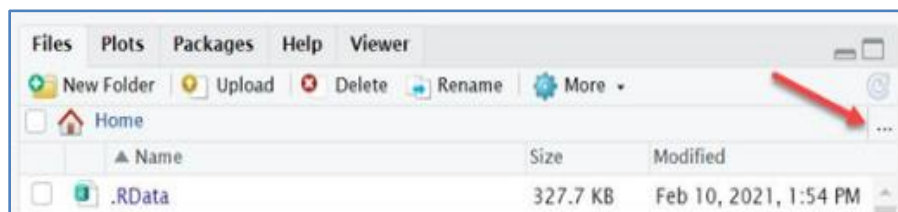



Figure 5. Pane 4 Files Tab (Click the three dots to open or change directory dialog.)




- The **Files tab** shows all files and folders available in your workspace. Default is your Home directory, but you can change to another directory by clicking the three dots shown on the far right in the line with the Home icon in Figure 5. Click on a filename to open a file. If it is a text type file, e.g., txt, R, sql, it will be opened in Pane 1 for viewing or editing. If it is a graphic file, RStudio will launch a browser tab, so you can view it there.
- The **Plots tab** displays any plots you generate.
- The **Packages tab** displays the list of packages installed in the *User Library* and *System Library*.
 - Any package with a check in the checkbox on the left is loaded in your active session; if a package does not have a checkmark, make it available by clicking the **checkbox**.
 - This tab has a dynamic search field to assist you with locating individual packages you might need. If you cannot locate the desired package there, search the Internet for it by name or view the [Comprehensive R Archive Network \(CRAN\) Contributed Packages](#)  web page for a list of available R packages. If you find the package in the CRAN Repository, you can install directly from the repository to your personal package library. Just click the **Install** button that is between the **Files tab** label in Pane 4 and the (Package) Name column header. As you type a package name in the Install dialog, a list of packages with the entered characters are displayed. You may select the desired package by clicking on it in the list.
 - You can also load Archive Packages (tar.gz) from other reliable sources. Your personal package library is in your Home directory (folder) under the “~/R/x86_64-redhat-linux-gnu-library” subdirectory and is displayed under the *User Library* sub-heading.
- The **Help tab** can be used to search for additional information on a package or function.
- The **Viewer tab** can be used to view local web content. **Note:** The Viewer pane can only be used for local web content. This content can be either static HTML files written to the session’s temporary directory (i.e., files with paths generated by the “tempfile” function) or a locally run web application to display generated HTML.

Changing the Layout of Panes

If you prefer a different Pane layout, this default layout can be changed.

Click on **View**, then **Pane**, and **Pane Layout...** for detailed control of the layout. Then **Apply** and **Save** your changes.

To temporarily change the layout, the icons in the upper right-hand corner in each pane can be used to minimize or expand the pane.

For new users becoming familiar with R or users wanting to increase their knowledge, the RStudio website can be a good resource. We suggest searching the internet for [RStudio Education](#)  or [RStudio Collections](#)  and [RStudio Webinars](#)  for links to web pages with more information.

Please remember: You do NOT need to download additional application software to use the web-based BioSense Platform RStudio Workbench.

Database Tables

This section describes the tables in the BioSense Platform’s Datamart database that you can access using RStudio. Replace the “XX” with your site’s short name, e.g., AK for Alaska, WY for Wyoming. If you have been granted access to the Datamart by your site administrator, you will have access to both production and staging versions of your data tables. *Production tables* (denoted by **PR**) contain production data sent to the BioSense Platform. *Staging tables* (denoted by **ST**) contain “test” data for use during the facility and feed onboarding processes.

For questions about onboarding, contact the NSSP Service Desk at support.syndromicsurveillance.org.

It will be useful to familiarize yourself with the following tables:

- **XX_MFT**: the MFT loaded to the BioSense Platform after the MFT clean-up process
- **XX_MFT_Except**: MFT records that were unable to be uploaded
- **XX_[ST|PR]_Raw**: the raw messages to be processed
- **XX_[ST|PR]_Processed**: the processed messages from the raw table
- **XX_[ST|PR]_Except** and related tables: the exceptions messages from the raw table
- **Filter_Reason**: Join the Filter_Reasons table to your **XX_[ST|PR]_Raw** table to understand why a record was filtered (i.e., not attempted to be processed)
- **Exceptions_Reason**: Join Except_Reasons table to your **XX_[ST|PR]_Except_Reason** table to understand why a record was exceptioned

Note: The nomenclature [ST|PR] denotes the option of choosing ST or PR, but not both.

Following are detailed descriptions of both production (PR) and staging (ST) tables.

Names and Descriptions of Available Tables

Facility Information

XX_MFT	Contains current Master Facility Table (MFT). <i>To update or modify your MFT, contact the NSSP Service Desk.</i>
XX_MFT_Except	Contains facility records that could <u>not</u> be loaded to the MFT.
XX_Operational_Crosswalk	Contains crosswalk used during production. Here, old or inactive facilities are mapped to new and potentially active facilities.
XX_Site_Contacts	Contains contact information of site administrators for your site.

Production-quality Data

XX_PR_Raw	Contains original messages you delivered to the platform and some metadata about those messages. If a message was filtered and designated invalid for syndromic surveillance, the Filter_Reason column will contain the code for why it was filtered.
XX_PR_Processed	Contains the processed messages received and the calculated values built from the elements. This table will only contain messages that met the minimum processing criteria. Incomplete or invalid messages will be sent to XX_PR_Except .

Names and Descriptions of Available Tables

XX_PR_Except	Contains the processed messages that did not meet minimum criteria for processing. The structure of the XX_PR_Except table and XX_PR_Processed table is identical. <i>To understand why a message appears in the XX_PR_Except table, you may want to join to the XX_PR_Except_Reason table.</i>
XX_PR_Except_Reason	Contains the message_IDs and any number of reasons for placing those records in the XX_PR_Except table. <i>To view the exception code descriptive values, join to Exceptions_Reason table.</i>

Staging Data (e.g., test data for feed and facility onboarding)

XX_ST_Raw	Contains original messages you delivered to the platform and some metadata about those messages. If a message was filtered and designated invalid for syndromic surveillance, the Filter_Reason column will contain the code for why it was filtered.
XX_ST_Processed	Contains the processed messages received and the calculated values built from the elements. This table will only contain messages that met the minimum processing criteria. Incomplete or invalid messages will be sent to the XX_ST_Except table.
XX_ST_Except	Contains the processed messages that did not meet minimum criteria for processing. The structure of the XX_ST_Except table and XX_ST_Processed table is identical. <i>To understand why a message appears in the XX_ST_Except table, join the XX_ST_Except and XX_ST_Except_Reason tables.</i>
XX_ST_Except_Reason	Contains the message_ID and all the reasons for placing the record in the XX_ST_Except table. <i>To view the exception code descriptive values, join to Except_Reasons table.</i>

Reference Tables

Filter_Reasons	Maps the Filter Reason Codes found in the Raw table to their descriptive text.
Except_Reasons	Maps the Exception Reason Codes found in the site's Except_Reason table to their descriptive text.

Figure 6 shows an Entity Relationship Diagram of available BioSense Platform archive tables. You may find this helpful as you join Datamart tables to perform advanced queries. **Note:** Production (PR) tables are shown in this diagram. The same relationships exist in the Staging (ST) tables.

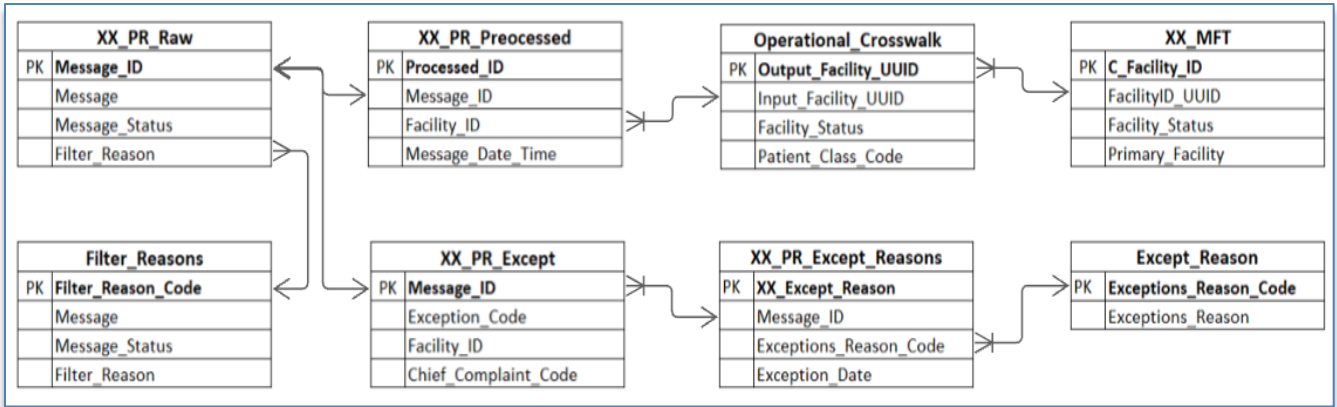



Figure 6. Entity Relationship Diagram for Production Tables

3. Features

Project Sharing

Project Sharing is a feature of RStudio Workbench that allows you to collaborate with other users and share data on the server. When you share a project, RStudio Workbench grants other users secure access to your project. If multiple users are active in the project at the same time, they can see the others' activities and work together in a shared editing tool.

Note: RStudio Support has a “How to” article about project sharing that will be of interest. Click here to view it: [Sharing Projects in RStudio Workbench](#) .



Shared Project Cautions

Users who share projects can inadvertently make their user credentials and site-specific information **visible to all other Shared Project users.**

Whenever you share a project on the NSSP RStudio Workbench server, all users with whom you share the project will have read-write-execute (rwx) access rights to the project folder just as you have. Thus, they can view all your files and execute any R computations normally available to you.

In other words, if you store login credentials in a connection string to query BioSense Platform data (instead of using the [Rnssp](#) or [keyring](#) R package) or save that information in a snippet or script in the project folder, the users you give access to will be able to use your credentials and retrieve other site-specific information that they may NOT have authorization or permission to access.

Be cautious. Only share projects with those who are trusted and who have received proper authorization to access your site's data.

Note: When using the NSSP RStudio Workbench, we recommend that you use the **Rnssp** package to store your credentials rather than including them in a snippet or script or use the less secure keyring package.

**ALWAYS USE THE Rnssp PACKAGE TO STORE YOUR CREDENTIALS
when using the NSSP RStudio Workbench to access BioSense Platform data.**

If an NSSP RStudio Workbench user does computations in the shared project directory, quits the R session, then chooses **Save** when prompted by the message below (Figure 7), the user makes all information (objects) saved there available to everyone with access to their shared project directory. This includes logins and site-specific information saved in code.

Specifically, the sensitive information that would be accessible to other shared project users will be in a file named `.RData` or in the folder named `.Ruserdata` (Figure 8).

For optimal performance and security, we generally advise that you click **Don't Save** when exiting a session.

Note: The Save Current Workspace prompt can be disabled in Global Options by selecting Tools → Global Options → General and changing “Save workspace to `.RData` (dot RData) on Exit” to “Never” in the drop-down list. (Don't forget to Save this change.)

Additionally, because saving your workspace can require significant disk space, as well as weaken security, we recommend that you do not save your workspace unless you will need to reload it.

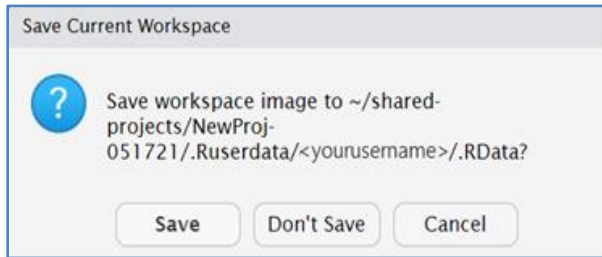


Figure 7. Clicking **Save** will make information available to everyone with access to your shared project directory.

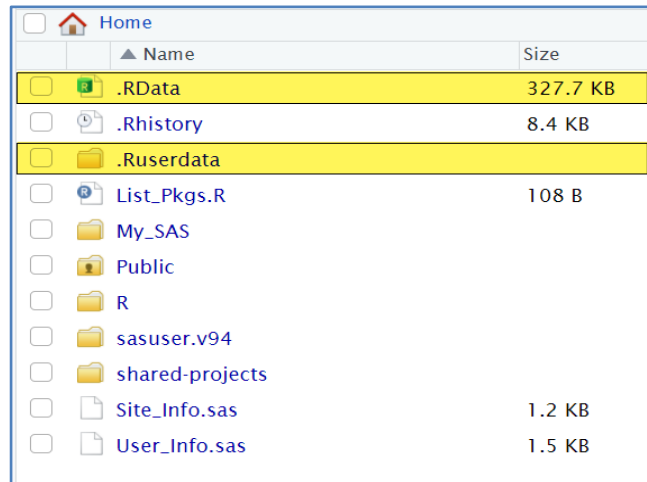


Figure 8. Storage Locations for Sensitive Information

To avoid the issue of exposing confidential information, *never* use the Shared Project Directory for work that is not intended to be shared. Instead, do your work in another session that's private. If you are in a Shared Project session, here's how to start a new (private) session:

1. Toward the middle of the RStudio top menu is an item called "Session." Click **Session**, and then click **New Session** on the drop-down menu (Figure 9).

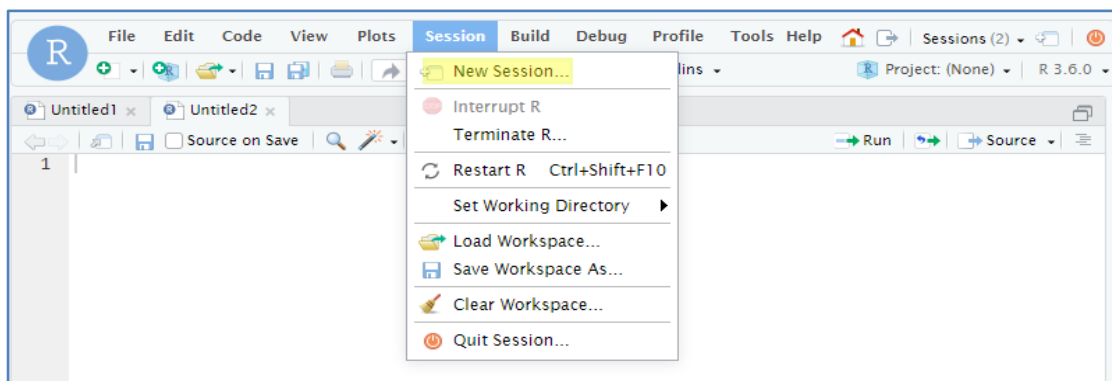


Figure 9. Start a New Session

2. Click the **Directory** radio button and then click **Start** (Figure 10). Note the tilde (~) in the directory field indicates your Home directory. This is the default directory setting. You may add a subdirectory or folder here, if you wish, by adding */folder_name* after the tilde, i.e., *~/folder_name*.

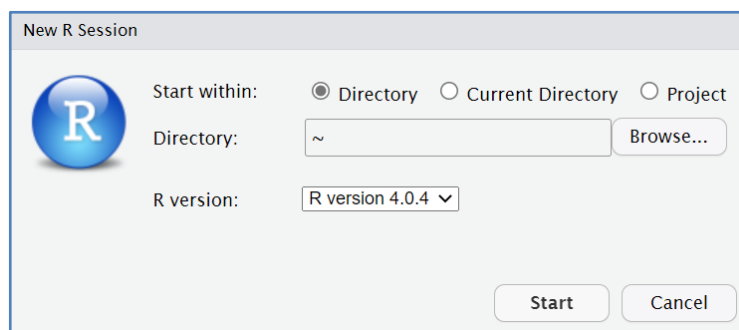


Figure 10. Opening a New Private Session

When you click **Start**, this will open a new initial session in your Home directory or subdirectory. **Only YOU have access to your Home directory and any of its subdirectories.** Here you can do work and store whatever you want without others seeing it, and, if needed, you can save your workspace here without risk of exposing sensitive data.

If you do not close either session, both Home and Shared Sessions will remain open. You can navigate between them by clicking the **Sessions down arrow** at the top right of the RStudio menu (Figure 11). Here, you can also see which sessions are open and navigate between them by clicking on the desired session.

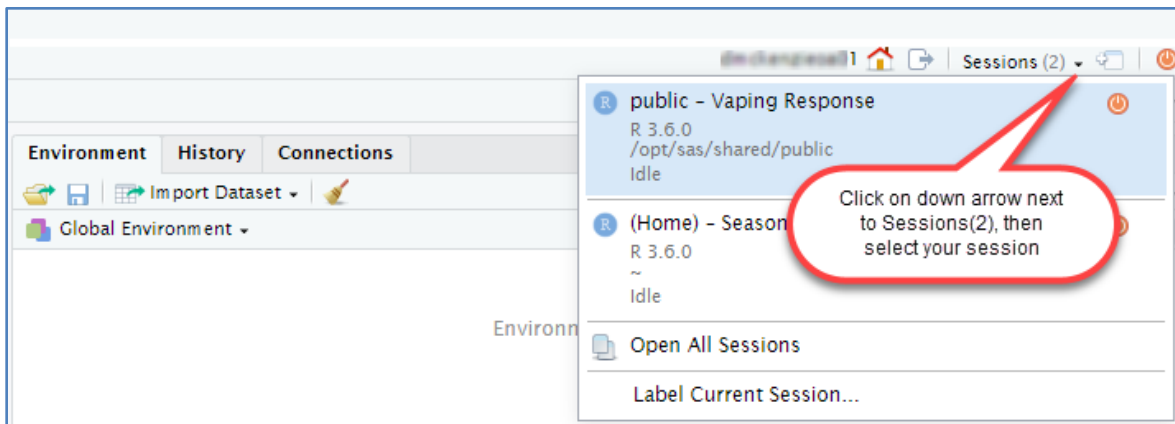


Figure 11. Navigating between Sessions

Steps for Creating a Shared Project in RStudio

1. Create a Project

Create your project by selecting **File, New Project**, or click on the **Create a project** icon (Figure 12).

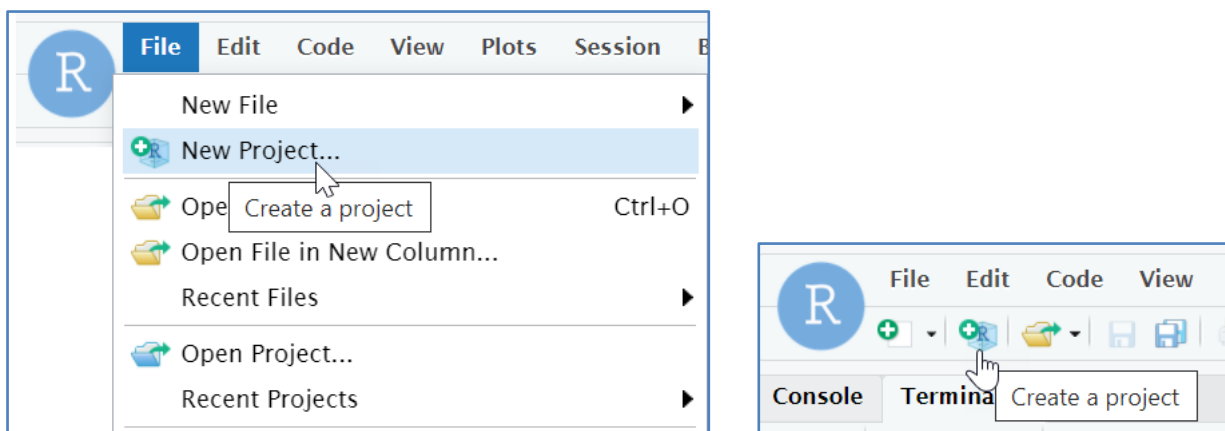


Figure 12. Create a new project by selecting **File, New Project** or by selecting the **Create a project** icon.

2. Select Directory or Version Control

After starting the process, the New Project Wizard will pop up and you will be given a choice to create your project either in a new directory or in an existing one (Figure 13).

Version Control is also available here from your Git Repository if you have you have set one up and wish to store your project files there. **Note:** Although the Subversion Repository is listed as a choice under Version Control, Subversion is NOT currently available on the NSSP RStudio Workbench (Figure 13).

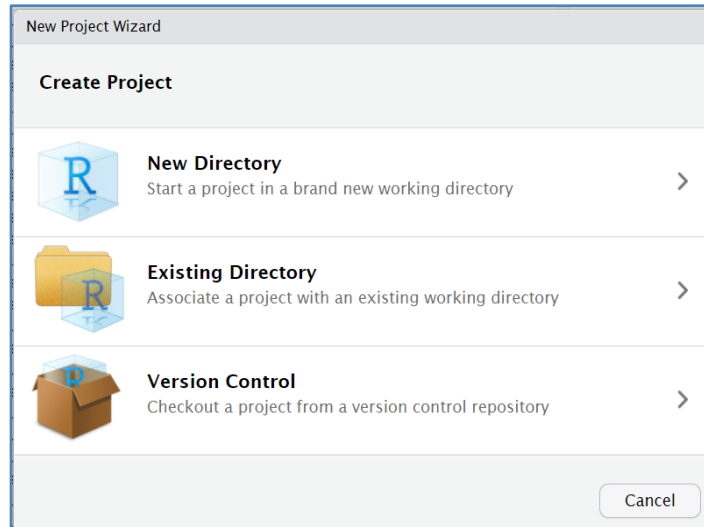


Figure 13. New Project Wizard

3. Create a Project Team to Share Your Project

Once you create a new project or open an existing one, you can share the project with other RStudio users by clicking on **File**, then **Share Project** (Figure 14, Left), to pop up the Project Options window. There you may choose those users you want to share the project with by selecting their AMC user ID and clicking the **Add** button (Figure 14, Right). Note that you can only share projects with users who have signed into RStudio Workbench previously.

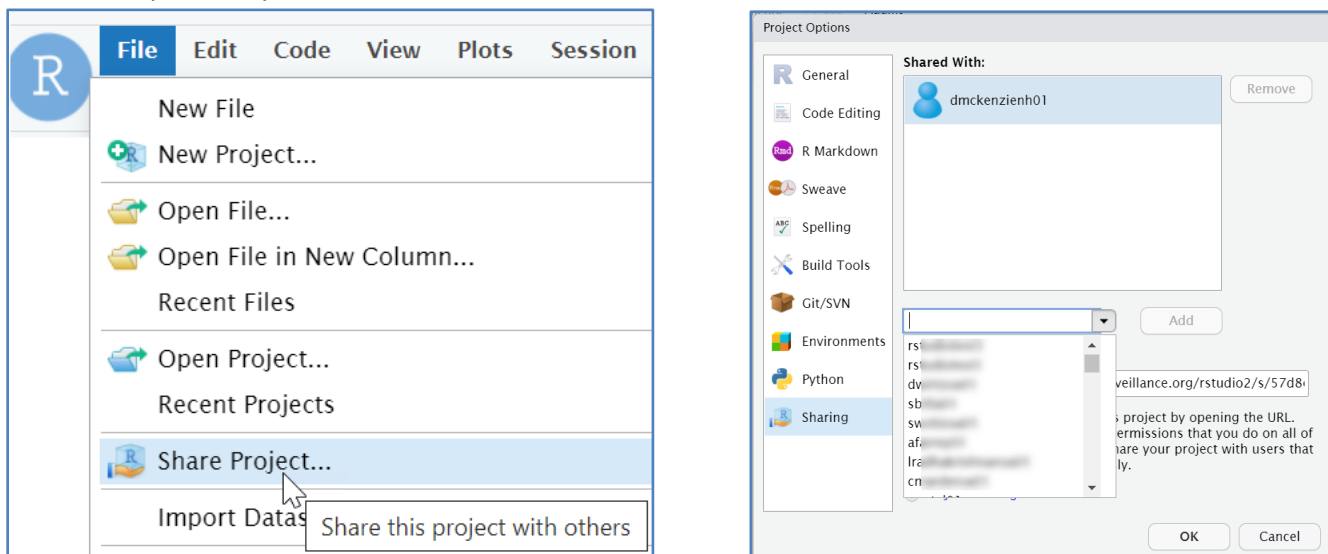


Figure 14. Share your project with other users by selecting **File**, then **Share Project** (project must be open) [left]. Then select **Sharing** from the navigation pane and choose **User IDs** from the Project Options, Sharing window [right].

Note: The URL in the Project URL field (Figure 15) may be used by your project team members to access the project from their workstations. Please share this link with them.

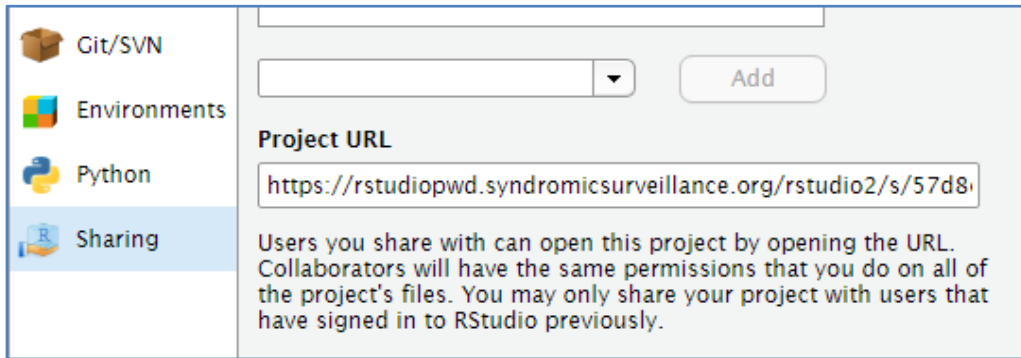


Figure 15. Project URL for Sharing Access

 **Your project team will have the same permissions on all the project's files that you do.**

Multiple Versions of R

The BioSense Platform RStudio Workbench supports a single version of the R language for all users. However, RStudio allows users to run other versions of R from their Home folder.

Users wanting to run other versions of R are encouraged to review RStudio's Multiple R Versions article (referenced at the right) because those versions will need to be self-supported, i.e., the NSSP support team cannot assist you with using other versions of R.

Multiple R Versions

To learn more about running multiple R versions, click here:

[Using multiple versions of R with RStudio Workbench/RStudio Server Pro](#) 


Multiple R Sessions

RStudio lets you open multiple concurrent sessions, which can be useful when you want to:

- Run multiple analyses in parallel,
- Keep multiple sessions open indefinitely, or
- Participate in shared projects while doing other work.

Multiple R Sessions

To learn more about running multiple R sessions, click here:

[Multiple R Sessions in RStudio Workbench / RStudio Server Pro](#) 

The hyperlink at the right accesses RStudio's article.

You can start a new session by clicking the **Session** button at the middle of the top menu (Figure 16) and selecting **New Session** from the drop-down menu.

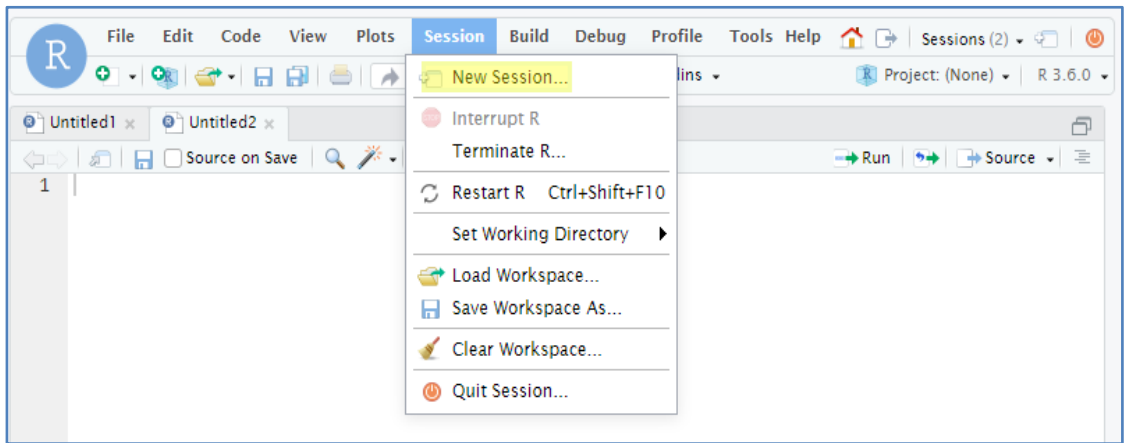


Figure 16. Start a New Session

To see all active sessions, click on the **Sessions (n)** button at the right end of the top toolbar (see Figure 17). This will display a drop-down list showing your open sessions. Here, you can give your sessions names to help keep them organized.

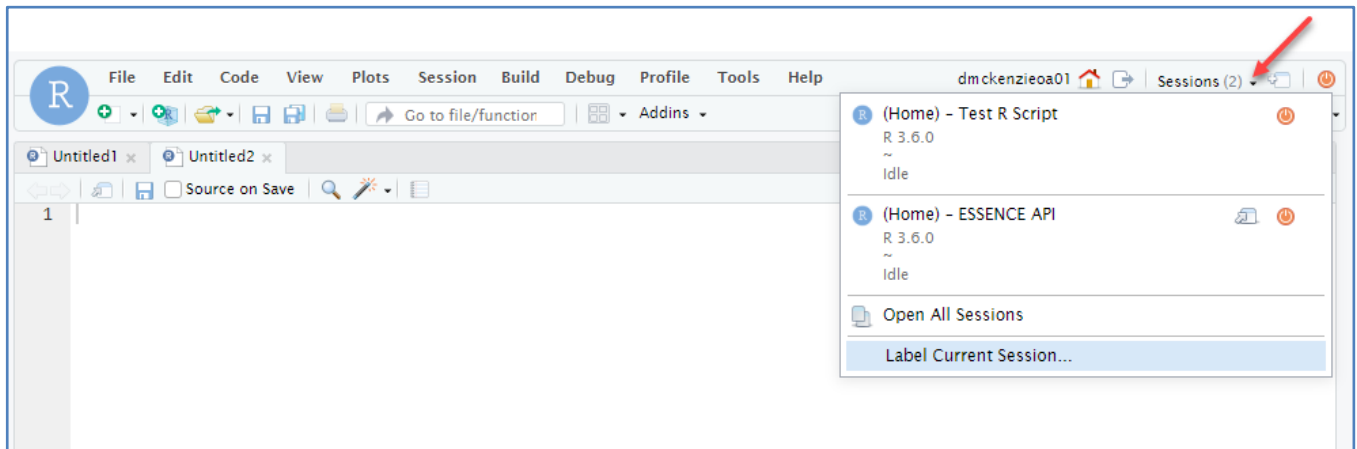


Figure 17. Select Session and Label Sessions

R Packages for Use in RStudio

There are numerous RStudio packages that you may find to assist you with data analysis. Many of these R packages have been preloaded in RStudio Workbench, and new packages are being added periodically.

The extensive list of R packages in the *System Library* is available for immediate use on the RStudio Workbench server. Scripts for obtaining the current list are noted in [Appendix B](#).

In addition, after launching RStudio, you can see the complete list of packages in the *System Library* with short descriptions on pane number 4 in the lower right (the Files/Plots/Packages/Help/Viewer pane). Any packages installed in your personal *User Library* will be listed before the *System Library*. Just click on the **Packages tab** (Pane 4) and scroll through the list.

In case you don't have the exact package name, a search field is available on the **Packages tab** (Pane 4). This search field uses a dynamic search function and can find your search string anywhere in the package name or description field (Figure 18).

Should you be unable to find the package you want to use in the *System Library*, you may install additional R-compatible packages from the [CRAN Repository](#) [↗](#) or other repositories (tar.gz files) by using the instructions noted previously in the [Home Page Full View](#) description of the **Packages tab** (Pane 4) or by running the following code (change *PackageName* to the case-sensitive name of the package you want to install):

```
install.packages("PackageName") [QUOTE MARKS REQUIRED]
```

To install multiple packages, you must use the vector command `c()`,
e.g., `install.packages(c("Package1", "Package2"))`

The default repository for packages is CRAN (<https://cran.r-project.org>). You may also load directly from a package archive file (*PackageName*.tar.gz) after you upload the file to your Home(~) directory or a subdirectory.

After successful installation, the new package will appear in your *User Library*. This personal library is automatically created when you load your first user package to RStudio. Personally loaded packages will be displayed in the **Packages tab** (Pane 4) under *User Library*, listed before the *System Library*, and can be selected there.

Note: Your packages will be physically stored in the directory below:

```
./opt/sas/shared/homes/<yourusername>/R/x86_64-redhat-linux-gnu-library/4.0
```

Once you install a new package in your *User Library*, you assume responsibility for keeping it up to date.

If you have not modified the package, you can check that it is current and automatically update it, if needed, by clicking on the **Update** button on the **Packages tab** in pane 4. (Note that this will only check and update the packages in your *User Library*.)

When you no longer need a package in your *User Library*, you may delete it by clicking on the small "x" located to the far right of that package name. Removing a package from your *User Library* will not affect a *System Library* copy if it exists.

Note: Even when a package has been installed and selected for inclusion in a script by checkmarking it on the **Packages tab**, it is a good practice to add the following code at the beginning of your programs to call that package:

```
library(PackageName) [DO NOT USE QUOTE MARKS AROUND PackageName]
```

CAVEAT: If you choose to load a package to your *User Library* with the same name as a package in the *System Library* or one that is subsequently added to the *System Library*, the package you loaded (i.e., residing in your *User Library*) will always be chosen when the “library(PackageName)” statement is executed in your R scripts. Keep this in mind—especially if you choose to customize a copy of an existing package but NOT rename it. It is the user’s responsibility to administer the personal *User Library*, including keeping it up to date.

Currently Installed R Packages for All Users

Instructions to obtain a complete listing of currently installed packages in the *System Library* can be found in [Appendix B](#).

You may also use the dynamic search function in the **Packages tab** (Pane 4) as shown in Figure 18 to search for specific packages. The search begins as soon as you enter the first character and searches both the package name and description for the character string you enter.

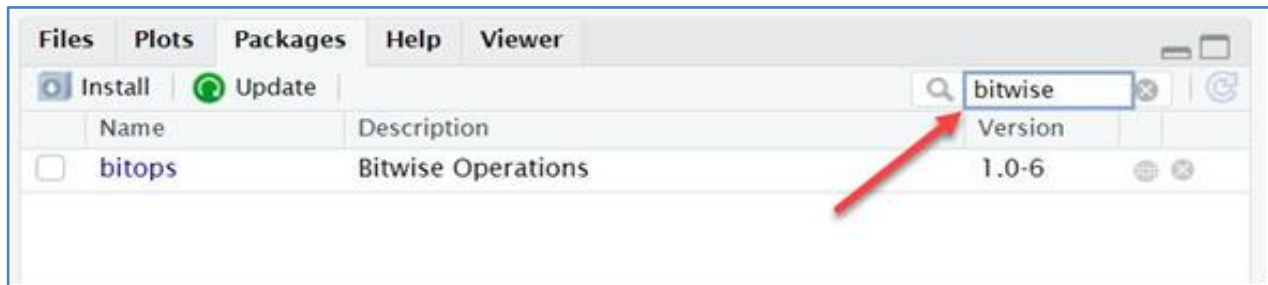


Figure 18. Dynamic Search for R Package by Name or String in the Name or Description

As shown in Figure 19, you may verify that your *User Library* packages are up to date by clicking the green **Update** button on the **Packages** tab (Pane 4). (This only checks the packages in your *User Library*. The *System Library* is updated monthly when the Monthly Server Patches are applied.)

WARNING: If you have modified a package in your *User Library*, it will be overwritten when you click the **Update** button.

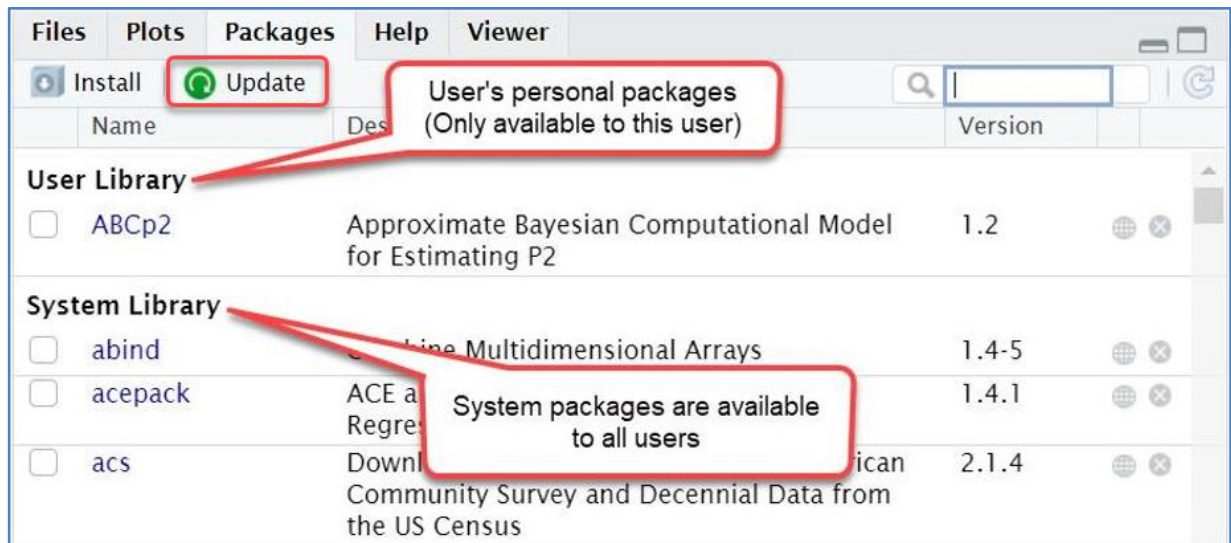


Figure 19. User and System Package Libraries

Requesting New Packages be Installed in the *System Library*

Since users may install packages for use in their personal environment (*User Library* as shown in Figure 19 above), adding new packages to the RStudio Workbench *System Library* is not normally needed.

However, if you believe that a package will be needed or prove useful to multiple users, please enter a ticket at support.syndromicsurveillance.org with the information below:

- Describe the package and its functions.
- Explain how it will be used (include a business case or the type of analysis or study it would be used for).
- Identify typical users who would benefit from having this package available.

The NSSP team will review all requests and provide feedback on disposition.

Remember: If you need the package right away, you may install it in your *User Library* until it has been processed and added to the *System Library*.

4. How to Connect to Datamart

Data for a given site are in tables in the BioSense Platform Datamart. The platform for this service is a Microsoft SQL Server.

RStudio developed the R Database Interface (DBI) package to provide common access to several database management systems (DBMS). In addition, the BioSense Platform RStudio configuration uses the Open Database Connectivity (ODBC) interface by Microsoft that allows applications to access data using SQL.

You can use the DBI package, the ODBC driver, and the preconfigured data source names to interact with the BioSense Platform Microsoft SQL Server DBMS.

RStudio uses a dbConnect method found in the DBI package to connect to the Datamart and access your tables. The following code shows how a database connection can be assigned to an object named "con":

```
library(DBI)
con <- dbConnect(odbc::odbc(), "BioSense_Platform")
```

Both connection methods use your Active Directory AMC account credentials (User ID/Password) to access the Datamart server and do not require manual input for authentication. Your database connection is accessible through the **Connections tab** in Pane 3 on the upper right of the RStudio screen (Figure 20).

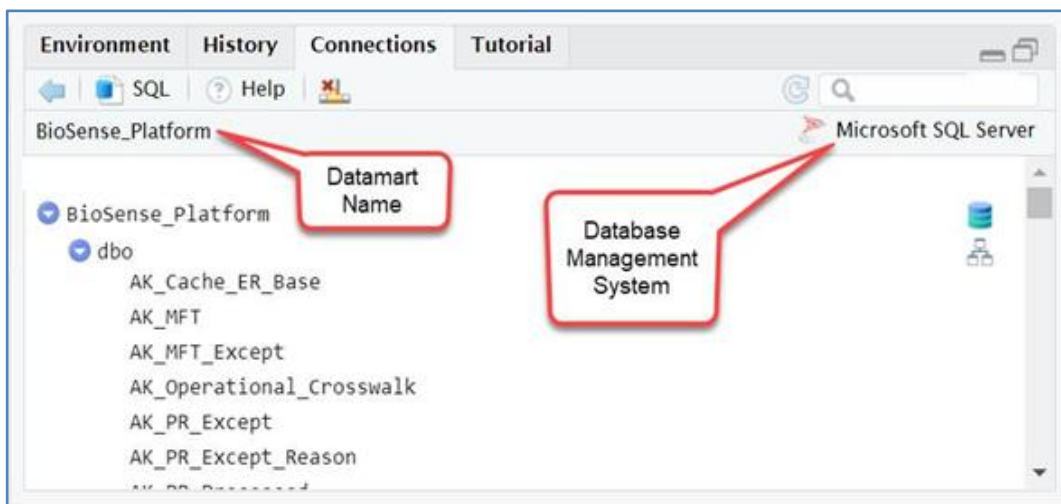


Figure 20. Connection Tab Showing Connection to the BioSense Platform Datamart



Securing Your Credentials

As with any programming language, do not hard code your credentials (User Name/Password) in your scripts. There are two packages that allow you to store your ESSENCE login credentials:

1. Documentation for Rnssp: <https://knowledgerepository.syndromicsurveillance.org/updated-how-use-rstudio-essence-api-guide>.
2. Documentation for keyring: <https://github.com/r-lib/keyring> .

When using the BioSense Platform’s RStudio Workbench, please use the Rnssp package to secure your credentials. It is more secure than keyring, and the Rnssp package has the advantage of:

- Not having to re-input your credentials each time you start a new session, and
- Not exposing your credentials while in a session.

If you are using a locally installed version of RStudio, you may choose to use the keyring package or, better yet, you may install the current version of Rnssp from GitHub by running `devtools::install_github("cdcgov/Rnssp")` in the console. The Rnssp GitHub repository can be accessed at <https://github.com/CDCgov/Rnssp> , with additional documentation and vignettes located at <https://cdcgov.github.io/Rnssp/> .

We describe both approaches here, however, the [NSSP ESSENCE API Guide](#) (in the Update section, click on the “here” link), which can be found in the [NSSP Surveillance Knowledge Repository](#), contains an in-depth description of these two options.

Securing Your Credentials—Required Packages

Insert these statements in your R script to use the examples in Section 5:

```
library(tidyverse)
```

```
library(httr)
```


```
library(jsonlite)
```

```
# Select Rnssp or keyring package
```

```
library(Rnssp) OR library(keyring)
```

```
# Please use the Rnssp package when accessing ESSENCE data for better security
```

The Rnssp Package

NSSP released the Rnssp package in June 2021, and it has been loaded to the BioSense Platform RStudio Workbench *System Library*. You may use `library(Rnssp)` in your R script or load it from the [Packages tab](#) (Pane 4). (For detailed information, visit the NSSP Community of Practice Knowledge Repository at [How to Use Rstudio with ESSENCE APIs](#) , dated 06/17/2021.)

To use Rnssp to save your credentials for use with the ESSENCE APIs, you will need to run the code snippet below. When executed and you supply your credentials, it creates an encrypted file named *myProfile.rds* in your Home directory.

```
# Run this code to create your encrypted myProfile.rds file containing your AMC Username and Password.
```

```
library(Rnssp)

myProfile <- Credentials$new(
  username = askme("Enter your username: "),
  password = askme()
)
# saveRDS allows you to later use another object name when reloading your credentials
saveRDS(myProfile, file = "~/myProfile.rds")
```

```
# REMEMBER: You should only run this code to set up your credentials file; DO NOT run this code each time you start # a session. You will have to re-run this code each time you update your AMC password.
```

When you have a script that requires your credentials, use the code below to fetch your username and password from the myProfile.rds file. Once read and assigned to an object, the three `get_api` methods may be used to retrieve data from the ESSENCE API Calls.

```
# When you create code to use an ESSENCE API, insert:
```

```
library(Rnssp)
```

```
# Having used saveRDS, you can now use another object name when reloading your credentials
```

```
# E.g., you could use myNewProfile instead of myProfile, if you preferred.
```

```
myProfile <- readRDS("~/myProfile.rds")
```

```
# Once the above code executes, the myProfile command will provide access to three new methods:
```

```
# $get_api_response(): Retrieves requested information specified in the API URL from ESSENCE
```

```
# $get_api_data(): Extracts the content (data) from the API response and parses it into an R data frame
```

```
# $get_api_tsgraph(): Retrieves an ESSENCE timeseries graph and saves it as a PNG to a temporary directory
```

The Keyring Package

As with any programming language, avoid hard coding your credentials in your scripts. In context, when using the ESSENCE APIs to retrieve data, you must supply your user ID and password. For this situation, the Rnssp package provides the preferred solution; however, on local instances of RStudio, you may choose to use the keyring package. To reemphasize, **Rnssp can also be used locally and is more secure.**

We provide keyring information only for convenience. Here is a brief description and instructions for using the keyring package to store and pass your credentials to ESSENCE.

A key in your keyring has four main attributes:

1. Service—The key's unique identifier; in examples used below, "ESSENCE" is the service.
2. Keyring—The key's "parent" keyring. If not specified, the default keyring is used.
3. Username—Typically, this is the AMC/ESSENCE/RStudio username with which you logged in.
4. Password—Typically, this is the AMC/ESSENCE/RStudio password with which you logged in.

Keyring retrieves data by passing the keyring name and service name. If no keyring name is passed, the default keyring is used. The `keyring::key_set()` function is used to create a key; when executed, a prompt will appear asking for the password that should be used with the username submitted:

```
keyring::key_set(service = "ESSENCE",  
                username = "yourusername")
```

Retrieve credentials: The `key_list()` command is used to retrieve the username:

```
keyring::key_list("ESSENCE")
```

To extract only the username to pass it inside the connection string, use:

```
keyring::key_list("ESSENCE")[1,2]
```

To extract the password, use `key_get()`:

```
keyring::key_get("ESSENCE", "yourusername") or better yet,  
keyring::key_get("ESSENCE", keyring::key_list("ESSENCE")[1,2])
```



CAVEAT: If a user runs the above code separately, keyring will display the unencrypted password in the console. Rnssp does not have this security flaw, hence, the preferred method of storing credentials is using the Rnssp package. Please set up Rnssp and use it on the BioSense Platform RStudio Workbench.

PROGRAMMING NOTE about the `keyring` calls: The [1,2] following the `key_list` statements above is used to extract your username from the object that `key_list` returns. Our `key_list` call returns one row with two fields. They are `service` and `username` because we set this up when we executed `key_set` earlier:

```
key_set(service = "ESSENCE", username = "yourusername")
```

So, for our use, `service` is ESSENCE and `username` is your AMC User Name; when we add the suffix [1,2], we specify row 1, column 2, i.e., `username`.

These functions can be used to supply AMC credentials without storing them in plain text or an environment variable. For example, when pulling data using the ESSENCE APIs, you would use this or similar code where "url" is the API call object. Here's how you would call the ESSENCE API (specified in the url object) and pass your credentials:

```
api_response <- GET(url, authenticate(key_list("ESSENCE")[1,2],  
                                   key_get("ESSENCE", key_list("ESSENCE")[1,2])))
```

5. Using RStudio with ESSENCE APIs

RStudio Workbench lets you access ESSENCE data via application programming interfaces (APIs). An API is a structured and consistent way for one machine to exchange information with another machine.

The ESSENCE APIs are ideal for generating reports published at regular intervals. For example, if you create reports for public health officials, try using the APIs to share trends by syndrome or to compare weekly visits for opioid overdose. You can also use APIs to create infographics for nontechnical audiences.

Information about APIs can be found in ESSENCE under “More” → “User Guide” → “API Documentation.” You can write API URL syntax on your own after reading the documentation; or, you can let ESSENCE create the API URL for you by clicking the **API URL** button on an ESSENCE page after you complete a query that provides the results you are interested in.

At the time of this writing, ESSENCE offered six APIs (follow hyperlinks to examples):

- [Time Series Data Table](#),
- [Time Series Graph from ESSENCE](#),
- [Table Builder Results](#),
- [Data Details \(line level\)](#),
- [Summary Stats](#) for the number of unique facilities or regions in your query results, and
- [Alert List Detection Table](#).

Setting Up to Call the ESSENCE APIs

To extract data from ESSENCE, you must pass your authentication information from your RStudio session to ESSENCE so that it knows what data you have permission to see.

Avoid including your username and password in your code! Instead, use the `Rnssp` or the `keyring` package as described in [Section 4](#).

Setting up Rnssp

If you are using the BioSense Platform RStudio Workbench and have run the code to create your myProfile.rds credential file, no additional setup is required.

If you have not yet created your myProfile.rds file, please load the Rnssp package and create your myProfile.rds file by using this code snippet:

```
# Create your myProfile.rds credential file first
library(Rnssp)
myProfile <- Credentials$new(
  username = askme("Enter your username: "),
  password = askme() )
saveRDS(myProfile, file = "~/myProfile.rds")
```

Note: This code must be re-run whenever you change your AMC password.

Setting up Keyring

If you are running a local instance of RStudio and wish to use the keyring package, start by executing the key_set command as shown here. (**Note:** This code must be run each time you open a new RStudio session.)

```
# library_set should only be run once per session to load your username and
# password into the keyring. REPLACE "yourusername" with your AMC Username

# Uncomment the following line if you have not installed the keyring package
# install.packages("keyring")

library(keyring)
key_set(service = "essence", username = "yourusername")
```

When you run this code (with your AMC username entered in the quotes), a pop-up will be displayed in RStudio asking you to enter your AMC password. (The operating system will store your username and password for the remainder of the session without rerunning this code again.)

Note: After submitting your password, the following warning will be returned:

Warning message:

In default_backend_auto() :

Selecting 'env' backend. Secrets are stored in environment variables

THIS IS EXPECTED AND DOES NOT INDICATE A PROBLEM.

Completing Setup to Use the ESSENCE APIs

Certain libraries (packages) must be available before calling an API in your R code. By default, most of these packages will be available in the *System Library*. If not, they can be installed by running the `install.packages()` command in the console. For example:

```
install.packages("tidyverse")
```

Below are the package names you'll need to get started (Figure 21):

You may wish to create a short code chunk or a snippet to include in your R scripts to quickly load these packages.

```
library(tidyverse)
library(httr)
library(jsonlite)
library(Rnssp) or library(keyring)
# LOAD THESE PACKAGES or COPY TO YOUR SCRIPT BEFORE RUNNING CODE EXAMPLES BELOW!
```

Figure 21. Package Names Needed for ESSENCE APIs

Calling the ESSENCE APIs


Note: Except for the “summary stats” API, the following examples use ESSENCE limited details data sources.

In each example that follows, you will see a common pattern emerge:



1. Define the URL as an object in your RStudio session.
2. Pass your credentials using the **Rnssp** or **keyring** package.
3. Tell RStudio to “get” data from the URL object.
4. Extract the contents of the object into an R data frame for further analysis or different presentation.

Note: Most users use ESSENCE for interactive querying and building myESSENCE dashboards in the user interface. So, we recommend use of ESSENCE2 for pulling data via APIs.

Programming Note: Coupling ESSENCE API calls with [R Markdown](#)  will allow you to code an easily reproducible workflow that will integrate report text with code to read in and manipulate data, then produce analyses and visualizations in such a way that the results can be handed off to colleagues without having to document manual actions (point/click, etc.).

Time Series Data Table (“timeseries” API)

In the code that follows, the first object created, “myProfile,” contains your credentials that were extracted from your myProfile.rds file.

The second object is “url” for the timeseries API with parameters of interest for examining the national injury syndrome trend.

The third object, “api_response,” is created and receives the results from the API call after passing ESSENCE your credentials.

The next two objects, “api_response_json” and “api_data,” are for processing the JSON-formatted data into an R data frame.

The glimpse function (from dplyr package) provides a quick sense of every variable in the data frame (Figure 22):

```
# This example uses the Rnssp package to secure your credentials
# Your MyProfile.rds file must have already been created
# (See instructions to create myProfile.rds file above)
library(tidyverse)
library(httr)
library(jsonlite)
library(Rnssp)
myProfile <- readRDS("~/myProfile.rds")
url <-
"https://essence2.syndromicsurveillance.org/nssp_essence/api/timeSeries?endDate=29Mar2022&
medicalGrouping=injury&percentParam=noPercent&geographySystem=hospitaldhsregion&dataso
urce=va_hospdreg&detector=probrepswitch&startDate=30Dec2021&timeResolution=daily&medica
lGroupingSystem=essencesyndromes&userId=455&aqtTarget=TimeSeries"
api_response <- myProfile$get_api_response(url)
api_response_json <- content(api_response, as = "text")
api_data <- fromJSON(api_response_json) %>%extract2("timeSeriesData")
# Use glimpse to view some of the data and information about the dataset
glimpse(api_data)
Rows: 61
Columns: 8
$ date   <chr> "2021-10-18", "2021-10-19", "2021-10-20", "2021-10-21", "2021-10-22", "2021-1...
$ count  <dbl> 46968, 45703, 44508, 44369, 44522, 41986, 41343, 46603, 44121, 44215, 44204, ...
$ expected <chr> "44826.929", "44807.893", "44731.964", "44702.286", "44701.107",
"44712.857",...
$ levels <chr> "0.146", "0.264", "0.445", "0.518", "0.542", "0.9", "0.977", "0.187", "0.572" ...
$ colorID <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1,...
$ color  <chr> "blue", "blue", "blue", "blue", "blue", "blue", "blue", "blue", "blue", "blue...
$ altText <chr> "Data: Date: 18Oct21, Level: 0.146, Count: 46968, Expected: 44826.9, Switch: ...
$ details <chr>
"/nssp_essence/api/dataDetails?medicalGrouping=injury&percentParam=noPercent&...
>
```

Figure 22. Data for Time Series Code

Time Series Graph from ESSENCE (“timeSeries graph” API)

This example shows how to retrieve the ESSENCE graph itself instead of the underlying data for the graph, as shown previously. Here, we use the same national injury syndrome trend as before, but notice the URL now includes “api/timeSeries/graph?” (Figure 23).

You can add title and axis labels to the graph by adding other parameters to the URL, e.g., “&graphTitle=Injury%20Syndrome&xAxisLabel=Date&yAxisLabel=Count” (Figures 23 and 24).

Note: Title Label (**&graphTitle=National%20-%20Injury%20Syndrome%20Daily%20Counts**) and Axis Labels (**&xAxisLabel=Date&yAxisLabel=Count**) are italicized black in the URL below for emphasis.

The “write_disk” command in the script saves the graph to your Home directory.

```
# This example uses the keyring package to secure your credentials
# Be sure to run the code at Setting Up Keyring to load your credentials at the
# beginning of each new session.
library(tidyverse)
library(httr)
library(jsonlite)
library(keyring)
url <-
  "https://essence.syndromicsurveillance.org/nssp_essence/api/timeSeries/graph?endDate=29Jan20
  22&medicalGrouping=injury&percentParam=noPercent&geographySystem=hospitaldhhsregion&da
  tasource=va_hospdreg&detector=probrepswitch&startDate=18Dec2021&timeResolution=daily&me
  dicalGroupingSystem=essencesyndromes&userId=455&aqtTarget=TimeSeries&graphTitle=National
  %20-%20Injury%20Syndrome%20Daily%20Counts&xAxisLabel=Date&yAxisLabel=Count"
api_response <- GET(url, authenticate(key_list("essence")[1,2],
  key_get("essence", key_list("essence")[1,2])),
  write_disk("test.png", overwrite=TRUE))
```

Figure 23. TimeSeries Graph Retrieval Code

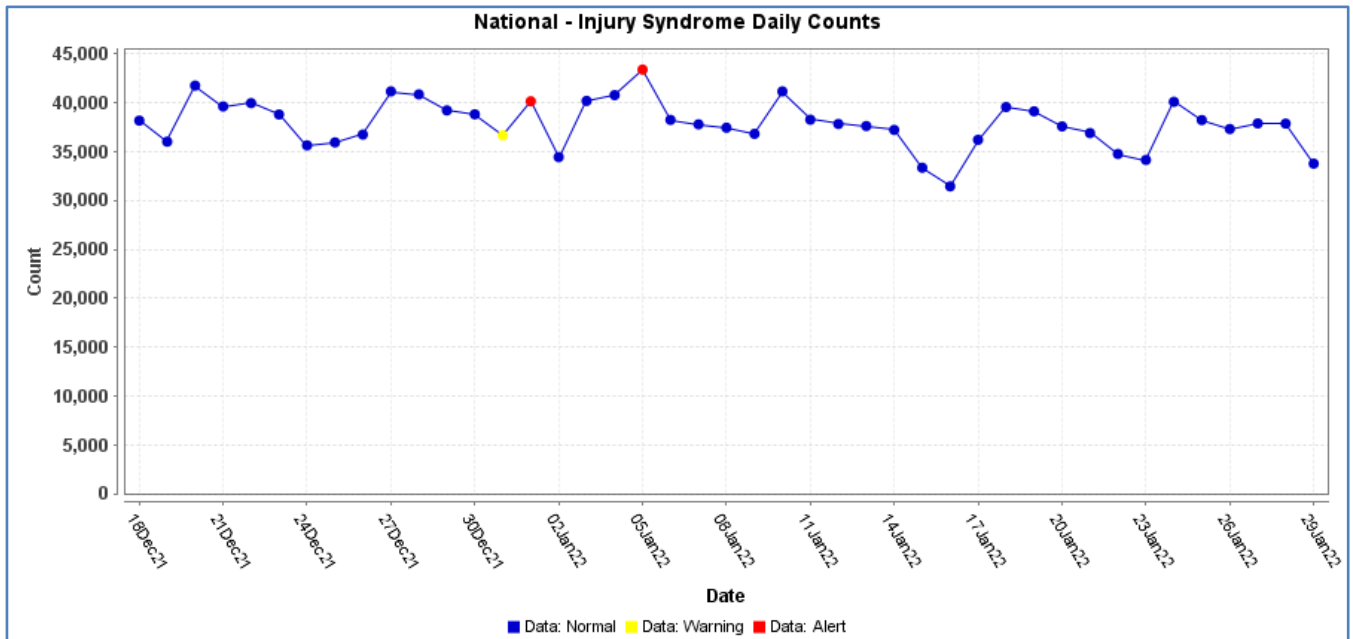


Figure 24. Visualization of National Injury Syndrome Data

Table Builder Results (“tableBuilder” API)

The table builder API lets you use ESSENCE for the heavy lifting of summarizing your query and presenting the results in tabular format. You can use embedded parameters to define rows, nested rows, and column variables for output.

If the query is supported in the ESSENCE query manager, you should be able to use table builder to summarize the output. Using the ESSENCE table builder API is usually more efficient than manipulating large amounts of line-level data with RStudio Workbench.

In the following example (Figure 25), we will use the Opioid Overdose Chief Complaint Discharge Diagnosis (CCDD) category to create a table of counts per month by age group for U.S. Department of Health and Human Services regions.

Note: Output formats for table builder results can be CSV or JSON. The CSV option will pull in data that more closely resembles what you would expect based on what is displayed in the ESSENCE interface, whereas the JSON option will pull data in a long, pivoted format.

```

# This example uses the Rnssp package to secure your credentials.
# Your MyProfile.rds file must have already been created (see instructions in Setting up Rnssp).
# This example extracts data in a CSV format.

library(tidyverse)
library(httr)
library(Rnssp)
myProfile <- readRDS("~/myProfile.rds")

url <-
"https://essence2.syndromicsurveillance.org/nssp_essence/api/tableBuilder/csv?endDate=31Jan2022&cc
ddCategory=cdc%20opioid%20overdose%20v2&percentParam=noPercent&geographySystem=hospitaldh
hsregion&datasource=va_hospdreg&detector=nodetector&startDate=1Jan2022&ageNCHS=11-
14&ageNCHS=15-24&ageNCHS=25-34&ageNCHS=35-44&ageNCHS=45-54&ageNCHS=55-
64&ageNCHS=65-74&ageNCHS=75-84&ageNCHS=85-
1000&ageNCHS=unknown&timeResolution=quarterly&hasBeenE=1&medicalGroupingSystem=essencesyn
dromes&userId=455&aqtTarget=TableBuilder&rowFields=timeResolution&rowFields=geographyhospitald
hsregion&columnField=ageNCHS"

api_response <- myProfile$get_api_response(url)
api_data <- content(api_response, as = "text") %>% read_csv()

> glimpse(api_data)
Rows: 11
Columns: 12
$ timeResolution <chr> "2022-1", "2022-1", "2022-1", "2022-1", "2022-1", "2022..."
$ geographyhospitaldhsregion <chr> "OTHER_REGION", "Region 1", "Region 10", "Region 2",
"R..."
$ `11-14` <dbl> 0, 1, 2, 0, 1, 5, 0, 6, 1, 3, 4
$ `15-24` <dbl> 0, 80, 119, 89, 112, 521, 219, 143, 50, 85, 173
$ `25-34` <dbl> 0, 402, 288, 347, 427, 1651, 749, 338, 70, 162, 298
$ `35-44` <dbl> 0, 411, 231, 306, 377, 1497, 635, 293, 48, 129, 226
$ `45-54` <dbl> 0, 286, 121, 254, 237, 893, 529, 170, 19, 76, 155
$ `55-64` <dbl> 0, 237, 184, 270, 229, 887, 575, 153, 29, 77, 194
$ `65-74` <dbl> 0, 94, 165, 124, 113, 560, 277, 109, 20, 83, 100
$ `75-84` <dbl> 0, 23, 58, 17, 21, 270, 50, 29, 5, 34, 47
$ `85+` <dbl> 0, 11, 11, 3, 7, 90, 8, 18, 2, 6, 18
$ Unknown <dbl> 0, 27, 10, 19, 27, 58, 20, 5, 3, 1, 4

```

Figure 25. Example of CSV Output Format

Tip for Creating an API: Instead of having ESSENCE create an API via the **API URLs** button, start by familiarizing yourself with the [ESSENCE API](#) structure. Look at examples where ESSENCE has created the API URLs. Then follow the same structure to add your own parameters.

Table Size Limit: The ESSENCE user interface is limited to creating tables of up to 30,000 cells, which should suffice in most situations. The table builder API, however, *does not limit* the output table size.

Data Details (dataDetails API)

Sometimes you may need line-level data, and this example describes how to extract those data from ESSENCE.



CAVEAT: Extracting line-level data can tie up limited resources and create large data sets. So, we recommend testing your query on a small amount of data (e.g., use a short time frame such as a day or two first to get some sense of how much data your query may retrieve).

If your query will create a large data set, consider multiple calls using shorter time ranges and create several less resource intensive data sets. These can then be combined to create your final large data set.

Remember, extracting large data sets ties up resources and can impede others' work.

By using the dataDetails API, you can specify the variables to include and whether you want a data set with raw or reference values. Data sets with reference values take longer to stream into RStudio because ESSENCE must create the reference values.

You can reduce the file size by specifying only the variables you need and by adding additional parameters to the URL, for example, “&field=age&field=ChiefComplaintParsed.” Reference the current [NSSP Data Dictionary](#) and see the [ESSENCE API and Data Details](#) and [ESSENCE API Query Parameters](#) tabs. (**Note:** The NSSP Data Dictionary will open as an Excel file.)

The [ESSENCE API and Data Details](#) tab referenced above contains an ESSENCE Data Details Display Name Reference table that cross references the *Data Details Web Display Name* to the *API URL Name*.

Here are a couple of sample scripts that you can try with your data. You can choose CSV or JSON output formats for data details (Figures 26 and 27).

```

# This example uses the Rnssp package to secure your credentials.
# Your MyProfile.rds file must have already been created (see instructions in Setting up Rnssp).
library(tidyverse)
library(httr)
library(Rnssp)
myProfile <- readRDS("~/myProfile.rds")
url <-
"https://essence2.syndromicsurveillance.org/nssp_essence/api/dataDetails/csv?medicalGrouping=injur
y&geography=region%20i&percentParam=noPercent&geographySystem=hospitaldhhsregion&datasour
ce=va_hospdreg&detector=probrepswitch&timeResolution=daily&medicalGroupingSystem=essencesyn
dromes&userId=455&aqtTarget=dataDetails&startDate=30Dec2021&endDate=30Dec2021"
api_response <- myProfile$get_api_response(url)
api_data <- content(api_response, as = "text") %>% read_csv()
# Use glimpse to view some of the data and information about the dataset
glimpse(api_data)

Rows: 2,260
Columns: 21
$ Date           <chr> "12/30/2021", "12/30/2021", "12/30/2021", "12/30/2021", ...
$ Category_flat  <chr> ";Injury;", ";Injury;", ";Injury;", ";Injury;", ";Injury...
$ SubCategory_flat <chr> ";Fall;", ";Fall;", ";Fall;", ";Fall;", ";Fall;", ";Moto...
$ Patient_Class  <chr> "E", "E", "E", "E", "E", "E", "E", "E", "E", "E", "E", "...
$ HospitalDHHSRegion <chr> "Region I", "Region I", "Region I", "Region I", "Region ...
$ dhhsregion     <chr> "Region I", "Region I", "Region I", "Region I", "Region ...
$ AgeGroup       <chr> "Unknown", "Unknown", "Unknown", "Unknown", "Unknown", "...
$ Sex            <chr> "F", "F", "F", "M", "M", "M", "F", "F", "F", "M", "M", "...
$ DispositionCategory <chr> "DISCHARGED", "DISCHARGED", "DISCHARGED", "DISCHARGED", ...
$ AdmissionTypeCategory <chr> "E", "E", "E", "E", "E", "NR", "NR", "NR", "NR", "NR", "...
$ HasBeenE       <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
$ HasBeenI       <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
$ HasBeenO       <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
$ DDAvailable    <dbl> 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, ...
$ DDInformative  <dbl> 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, ...
$ CCAvailable    <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
$ CCInformative  <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
$ FirstDateTimeAdded <dttm> 2021-12-30 23:22:14, 2021-12-31 05:21:10, 2021-12-30 17...
$ HasBeenAdmitted <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
$ CRace_CEth_Combined_Broad <chr> "White and non-Hispanic", "White and non-Hispanic", "Whi...
$ CRace_CEth_Combined_Narrow <chr> "White and non-Hispanic", "White and non-Hispanic", "Whi...

```

Figure 26. Example of CSV Output for Line-level Details

```

# This example uses the Rnssp package to secure your credentials.
# Your MyProfile.rds file must have already been created (see instructions in Setting up Rnssp).
library(tidyverse)
library(httr)
httr::set_config(config(ssl_verifypeer = 0L))
library(jsonlite)
library(Rnssp)
myProfile <- readRDS("~/myProfile.rds")

url <-
"https://essence.syndromicsurveillance.org/nssp_essence/api/dataDetails?medicalGrouping=injury&ge
ography=region%20i&percentParam=noPercent&geographySystem=hospitaldhhsregion&datasource=va
_hospdreg&detector=probrepswitch&timeResolution=daily&medicalGroupingSystem=essencesyndrom
es&userId=455&aqtTarget=TimeSeries&startDate=12Mar2022&endDate=12Mar2022"
api_response <- myProfile$get_api_response(url)
api_response_json <- content(api_response, as = "text")
api_data <- fromJSON(api_response_json)
api_data <- api_data$dataDetails
glimpse(api_data)

Rows: 2,677
Columns: 21
$ Date          <chr> "12/27/2021", "12/27/2021", "12/27/2021", "12/27/2021", "12/27/2021", ...
$ Category_flat <chr> ";Injury;", ";Injury;", ";Injury;", ";Injury;", ";Injury;", ";Injury;"...
$ SubCategory_flat <chr> ";Fall;", ";CutOrPierce;ToolsOrMachinery;AlcoholUse;", ";Fall;AlcoholU...
$ Patient_Class <chr> "E", "E", "E", "E", "E", "E", "E", "E", "E", "E", "E", "E", "E", ...
$ HospitalDHHSRegion <chr> "Region I", "Region I", "Region I", "Region I", "Region I", "Region I"...
$ dhhsregion <chr> "Region I", "Region I", "OTHER_REGION", "Region I", "Region I", "Regio...
$ AgeGroup <chr> "Unknown", "Unknown", "Unknown", "Unknown", "Unknown", "Unknown",
"Unk...
$ Sex <chr> "M", "F", "M", "M", "F", "F", "F", "M", "M", "M", "M", "F", "F", "M", ...
$ DispositionCategory <chr> "DISCHARGED", "DISCHARGED", "none", "DISCHARGED", "none", "none",
"DIS...
$ AdmissionTypeCategory <chr> "E", "U", "NR", "NR", "NR", "NR", "NR", "NR", "NR", "NR", "E", "U", "NR", "E...
$ HasBeenE <chr> "1", "1", "1", "1", "1", "1", "1", "1", "1", "1", "1", "1", "1", ...
$ HasBeenI <chr> "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", ...
$ HasBeenO <chr> "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", ...
$ DDAvailable <chr> "1", "1", "0", "1", "0", "0", "1", "1", "1", "1", "1", "1", "1", ...
$ DDInformative <chr> "1", "1", "0", "1", "0", "0", "1", "1", "1", "1", "1", "1", "1", ...
$ CCAvailable <chr> "1", "1", "1", "1", "1", "1", "1", "1", "1", "1", "1", "1", "1", ...
$ CCInformative <chr> "1", "1", "1", "1", "1", "1", "1", "1", "1", "1", "1", "1", "1", ...
$ FirstDateTimeAdded <chr> "2021-12-27 17:22:12.82", "2021-12-28 12:23:34.267", "2021-12-28
11:10...
$ HasBeenAdmitted <chr> "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", "0", ...
$ CRace_CEth_Combined_Broad <chr> "Multiracial/Other and non-Hispanic", "Multiracial/Other and non-
Hispa...
$ CRace_CEth_Combined_Narrow <chr> "Multiracial/Other and non-Hispanic", "Multiracial/Other and non-
Hispa...

```

Figure 27. Example of JSON Output for Line-level Details

Summary Stats (summaryData API)

The Summary Stats ESSENCE API counts regions or counties and facilities in your query by the time resolution you define (daily, weekly, monthly, quarterly, or yearly). One difference between this API and others is that it is only available on full details data sources (these are the only data sources that expose this level of information). The API in Figure 28 is particularly useful for understanding the number of hospitals with results for your query.

```
# This example uses the Rnssp package to secure your credentials
# Your MyProfile.rds file must have already been created
# (See instructions to create myProfile.rds file above)
library(tidyverse)
library(httr)
library(jsonlite)
library(Rnssp)
myProfile <- readRDS("~/myProfile.rds")
url <-
  "https://essence2.syndromicsurveillance.org/nssp_essence/api/summaryData?endDate=28Feb20
  22&medicalGrouping=injury&geography=region%20i&percentParam=noPercent&geographySyste
  m=hospitaldhsregion&datasource=va_hosp&detector=probrepswitch&startDate=1Feb2022&tim
  eResolution=daily&medicalGroupingSystem=essencesyndromes&userId=455&aqtTarget=TimeSeri
  es"
api_response <- myProfile$get_api_response(url)
api_response_json <- content(api_response, as = "text")
api_data <- fromJSON(api_response_json)
api_data <- api_data$summaryData
glimpse(api_data)

Rows: 28
Columns: 5
$ date      <chr> "01Feb22", "02Feb22", "03Feb22", "04Feb22", "05Feb22", "06Feb22", "07F...
$ HospitalState <dbl> 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6...
$ State      <dbl> 20, 19, 26, 20, 19, 23, 25, 27, 22, 27, 27, 23, 27, 24, 29, 21, 19, 22...
$ Region     <dbl> 105, 98, 111, 101, 111, 119, 121, 124, 103, 110, 123, 118, 126, 109, 1...
$ Hospital   <dbl> 209, 208, 202, 203, 206, 203, 216, 205, 216, 208, 209, 202, 199, 212, ...
```

Figure 28. Example of Summary Stats

Alert List Detection Table (regionSyndromeAlerts API and hospitalSyndromeAlerts)

The Alert List API provides programmatic access to the Alert List table in the ESSENCE user interface. The results in this table are updated a few times each day and are run by patient region (“county” in ESSENCE) and by hospital. Figure 29 shows the results by patient region, and Figure 30 shows results by hospital.

Note: Alert data is only available for the recent past. Please adjust dates in your API URLs to more current dates.

```
# This example uses the Rnssp package to secure your credentials.
# Your MyProfile.rds file must have already been created.
# (See instructions to create myProfile.rds file above.)
library(tidyverse)
library(httr)
library(jsonlite)
library(Rnssp)
myProfile <- readRDS("~/myProfile.rds")
#
# Change end_Date and start_Date to more current dates before running.
url <-
"https://essence2.syndromicsurveillance.org/nssp_essence/api/alerts/regionSyndromeAlerts?end
_date=16Feb2022&start_date=10Feb2022"
api_response <- myProfile$get_api_response(url)
api_response_json <- content(api_response, as = "text")
api_data <- fromJSON(api_response_json)
api_data <- api_data$regionSyndromeAlerts
glimpse(api_data)

Rows: 47,091
Columns: 12
 $ date          <chr> "2022-02-11", "2022-02-12", "2022-02-11", "2022-02-15", "2022-02...
 $ datasource    <chr> "va_er", "va_er", "va_er", "va_er", "va_er", "va_er", "va_er", "...
 $ age          <chr> "00-04", "00-04", "00-04", "00-04", "00-04", "00-04", "00-04", "...
 $ sex          <chr> "all", "all", "all", "all", "all", "all", "all", "all", "...
 $ detector      <chr> "probrepswitch", "probrepswitch", "probrepswitch", "probrepswitc...
 $ level        <dbl> 0.0166510624, 0.0151498470, 0.0214355047, 0.0246069293, 0.028779...
 $ count        <int> 4, 3, 4, 3, 4, 1, 6, 3, 1, 1, 2, 3, 72, 3, 6, 3, 6, 1, 1, 1, 2, ...
 $ expected     <dbl> 1.3928571, 1.5384615, 1.2857143, 1.5000000, 1.7142857, 0.2857143...
 $ region       <chr> "XX_Coweti", "XX_Coweti", "XX_Coweti", "XX_Coweti", "XX_Coweti",...
 $ syndrome     <chr> "GI", "GI", "Injury", "Injury", "Injury", "Rash", "Injury", "Inj...
 $ timeResolution <chr> "daily", "daily", "daily", "daily", "daily", "daily", "daily", "...
 $ `observed/expected` <dbl> 2.871795, 1.950000, 3.111111, 2.000000, 2.333333, 3.500000, 4.30...
 >
```

Figure 29. Alert List by **Region** (JSON output)

Note: In the hospital example, we have removed two variables—hospital name and hospital region—to avoid exposing this information in the examples, but you don’t need to do that in practice.

```
# This example uses the Rnssp package to secure your credentials.
# Your MyProfile.rds file must have already been created (see instructions in Setting up Rnssp).
library(tidyverse)
library(httr)
library(jsonlite)
library(Rnssp)
myProfile <- readRDS("~/myProfile.rds")
# NOTE: Alert data is only available for the recent past. Please adjust dates in your API URL to more
current dates.
url <-
"https://essence2.syndromicsurveillance.org/nssp_essence/api/alerts/hospitalSyndromeAlerts?end
_date=28Mar2022&start_date=28Mar2022"
api_response <- myProfile$get_api_response(url)
api_response_json <- content(api_response, as = "text")
api_data <- fromJSON(api_response_json)
api_data <- api_data$hospitalSyndromeAlerts %>% select(-hospitalName, -regionOfHospital)

glimpse(api_data)

Rows: 1,395
Columns: 11
$ date      <chr> "2022-03-28", "2022-03-28", "2022-03-28", "2022-03-28", "2022-03-28", "2022-
03-28"...
$ datasource <chr> "va_hosp", "va_hosp", "va_hosp", "va_hosp", "va_hosp", "va_hosp",
"va_hosp", "va_ho...
$ age       <chr> "65-1000", "18-44", "65-1000", "45-64", "05-17", "65-1000", "45-64", "all", "45-
64"...
$ sex       <chr> "all", "all", "all", "all", "all", "all", "all", "all", "all", "all", "all", "all", ...
$ detector  <chr> "probrepswitch", "probrepswitch", "probrepswitch", "probrepswitch",
"probrepswitch"...
$ level     <dbl> 0.0498047983, 0.0112446255, 0.0249981408, 0.0034725649, 0.0205652741,
0.0463442776, ...
$ count     <int> 1, 1, 1, 2, 2, 2, 4, 3, 3, 1, 2, 2, 2, 3, 3, 3, 7, 5, 1, 1, 1, 2, 2, 1, 2, 2, 1, 1...
$ expected  <dbl> 0.07142857, 0.28571429, 0.07142857, 0.21428571, 0.35714286, 0.50000000,
0.53571429, ...
$ syndrome  <chr> "ILI", "Fever", "Fever", "GI", "GI", "Resp", "Resp", "Resp", "Resp", "GI", "Rash",
...
$ hospital  <chr> "11429", "11462", "11469", "11564", "11564", "11669", "11669", "11785",
"11788", "1...
$ timeResolution <chr> "daily", "daily", "daily", "daily", "daily", "daily", "daily", "daily", "daily", "d...
>
```

Figure 30. Alert List by **Hospital**

Tips and Tricks—ESSENCE API URL Length

Sometimes the length of the API URL string may become too long to assign to the “url” object. If this occurs, the URL may be split into two or more strings to create separate objects that then can be pasted together to reform the complete URL.

The code chunk shown in Figure 31 begins with one long URL broken into two pieces that are assigned to individual URL objects, “url1” and “url2.” To create the object “url,” paste the pieces together.

This final URL can be passed to ESSENCE along with your credentials to return your results.

```
## DON'T FORGET TO LOAD REQUIRED PACKAGES (see instructions in Setting up Rnssp).  
#First part of URL  
url1 <- "https://essence2.syndromicsurveillance.org/nssp_essence/api/a_long_URL_very_long" #  
[use_your_imagination_here]  
  
#Second part of URL  
url2 <- "_URL-still_going_even_longer_here/even_more_and_more_characters_in_this_URL "  
# Note the underscore at the beginning of url2  
  
#Join the two parts into one longer string using “paste”  
url<- paste(url1, url2)  
# The url object can now be used in a GET call.  
# We just print it here to display the resulting very long url.  
print(url)  
The url object now contains  
"https://essence2.syndromicsurveillance.org/nssp_essence/api/very_very_very_long_URL_ver_lon  
g_use_your_imagination_here-still_going_even_longer_here"
```

Figure 31. Tip for Handling Lengthy URLs

Additional Resources

The preceding examples will help you start pulling data into your RStudio environment. If you are unfamiliar with R and RStudio, we suggest using your favorite search engine to look for terms such as these:

- RStudio Tutorial
- R Graphics Cookbook
- R Markdown
- Text Mining with R
- Tidyverse
- Forecasting with R

Appendix A. Indexes Available on Datamart

Indexes are used to expedite database queries. Normally, when a table with no indexes is queried, the entire table must be scanned to locate the correct record, and this takes time. When an index is added, the query only needs to search a subset of the data. This can significantly increase efficiency, so we add indexes to several of the most used tables to help make your queries more effective.

We periodically analyze table usage and other database statistics to determine where indexes will be most useful. Therefore, the lists of indexed tables and columns we presented in previous versions of this manual are no longer valid, though we can provide current lists upon request.

As of publication of this user manual, your site's Raw, Processed, Exceptions, and Exceptions_Reason tables have indexes.

If you find your queries processing more slowly than expected, please consider using an index created for that table.

To receive information on the current indexes available, contact NSSP's technical support team by opening a ticket at the Service Desk (support.syndromicsurveillance.org) and requesting the latest list of indexes for your site's datamart.

This page intentionally left blank.

Appendix B. Installed R Packages for R Version 4

The list of installed R Packages is evolving as more packages are developed and made available to the user community. Instead of providing a table of packages currently in the *System Library*, here are some queries you can use to create your own list of the R Packages for R 4.0 that will be up to date.

Queries to Extract R Package Names in the *System Library* and Your Personal *User Library*

Run the desired script in the Console window to extract package names from the *System Library* and your *User Library* and write the results to a CSV file. Note that the `lib.loc` parameter in the `installed.packages()` method is used to select the System or User's Library. If omitted, all installed packages are listed.

- *Extract a List of R Packages Available to Your R Session*

After you run this R script, use Excel or another program to open the `r_package_list_All.csv` file.

```
installed <- as.data.frame(installed.packages())
write.csv(installed, 'r_package_list_All.csv')
```

The output file (`r_package_list_All.csv`) will be written to your Home directory.

- *Extract a List of R Packages Available in System Library*

After you run this R script, use Excel or other program to open the `r_package_list_System.csv` file.

```
installed <- as.data.frame(installed.packages(lib.loc = .Library))
write.csv(installed, 'r_package_list_System.csv')
```

The output file (`r_package_list_System.csv`) will be written to your Home directory.

- *Extract a List of R Packages Available in Your User Library*

After you run this R script, use Excel or other program to open the `r_package_list_User.csv` file. Note that the third statement “`quit()`” will close your RStudio session.

```
installed <- as.data.frame(installed.packages(lib.loc = Sys.getenv('R_LIBS_USER')))
write.csv(installed, 'r_package_list_User.csv')
quit()
```

The output file (`r_package_list_User.csv`) will be written to your Home directory.

This page intentionally left blank.

Appendix C. Drive Space Allocation Rules for RStudio and SAS

File Storage

Both RStudio and SAS datasets are saved to your Home folder or shared Site folder on the same file server. This file server is a shared resource that everyone can use to store datasets and programming files, so rules for storing RStudio and SAS files, as well as drive space limitations, can impact both RStudio and SAS.

Drive Space

Each user is allocated 500 gigabytes (GB) of drive space on the file server. This allocation can be used to store both RStudio- and SAS-related files in the user's Home folder and their shared site folder.

All files a user creates count as part of the user's allocated storage, including:

- Files created *manually* by writing programs; saving macros, programs, and snippets; or uploading files; and
- Files created *programmatically* by pulling data from other sources, such as ESSENCE API results.

When users exceed the 500 GB maximum, they will receive a warning message and will have 7 days to reduce their total storage below 500 GB. During this grace period, the users will continue to receive warning messages but will not be prevented from creating or extending files because this space allocation maximum has a buffer. A user may temporarily use up to 600 GB if work in progress requires more data or workspace than expected. However, 600 GB is a hard limit and, if exceeded, further attempts to write to the disk will fail with an error message, even if the grace period has not yet expired.

If a user exceeding the 500 GB limit allows the 7-day grace period to expire without reducing storage to under 500 GB, the user will *not be permitted* to create new files or extend existing files until space usage is adequately reduced.

Effectively, there is no limit to the size of your site's shared folder. If each user who contributes files to the shared site folder remains under the individual drive space allocation, this folder can be added to as needed. This means that individual users do not need to know about the space usage of other users in their site before writing data or saving files to the shared site folder.

NSSP considers personal and shared storage needs when allocating drive space. Here are examples of how drive space allocation works:

- *Personal Drive Space Allocation*—Say you currently have 300 GB of files stored in your Home folder and a 500 GB quota for storage, but you need to add an additional 250 GB of data. Hoping you can avoid hitting your quota, you copy the 250 GB of data to your site's shared folder. Because your total usage is now 550 GB and exceeds your 500 GB quota, the system will send you a warning. With only 50 GB remaining for your maximum allowed storage (600 GB hard limit), if you continue to write data to your site's shared folder or to your Home folder, you will likely exceed this hard limit of 600 GB. You cannot write more data once you reach 600 GB.

- *Shared Drive Space Allocation*—You and two others are using the same shared site folder, and everyone has fewer than 100 GB of data stored in their individual Home folders. If each runs a program that saves 300 GB of data to the shared site folder, each will have used 400 GB of the 500 GB allocated. That is under the individual limit, but our shared site folder would total 900 GB (3 X 300 GB) of data. This is not a quota violation.

Requesting More Storage Space

If you require more than 500 GB of storage on a continuing basis, we will need to know why the additional space is needed (for example “I manage a 750-gigabyte data file that is shared with a dozen different sites”) and how much space is necessary (“I need 1.5 terabytes so I can store both the new and old copy of the file while it is being re-created, but my ongoing usage will be 750 gigabytes”).

Submit a request for additional storage to the BioSense Platform support team by opening a ticket at support.syndromicsurveillance.org.

Appendix D. Best Practices

- Drive space is limited. Only save datasets that are essential. You are assigned a half a terabyte of disk space for storing datasets and other files. When your allocated space is exhausted, the job that is running will return an error and fail. To recover storage space, delete previously created or extracted datasets.
- Run programs with large datasets in the background or at times when others are unlikely to run analyses.
- Your site's shared folder named XX_folder (XX represents your site short_name or abbreviation, e.g., Alabama would use the AL_folder and Wyoming would use the WY_folder) is available to all your site's users. There is, in addition, a Public folder that is available to users in any site. Keep this in mind when saving a dataset or other files to a folder that isn't your personal folder.
- There is a large memory pool allocated to the RStudio server and all active users share this memory; so, when working with a large dataset, try adjusting your code to limit dataset size or, possibly, running the program in the background to prevent out-of-memory errors. You can also try running your program when fewer people are using the system (late at night or on weekends if maintenance is not scheduled).