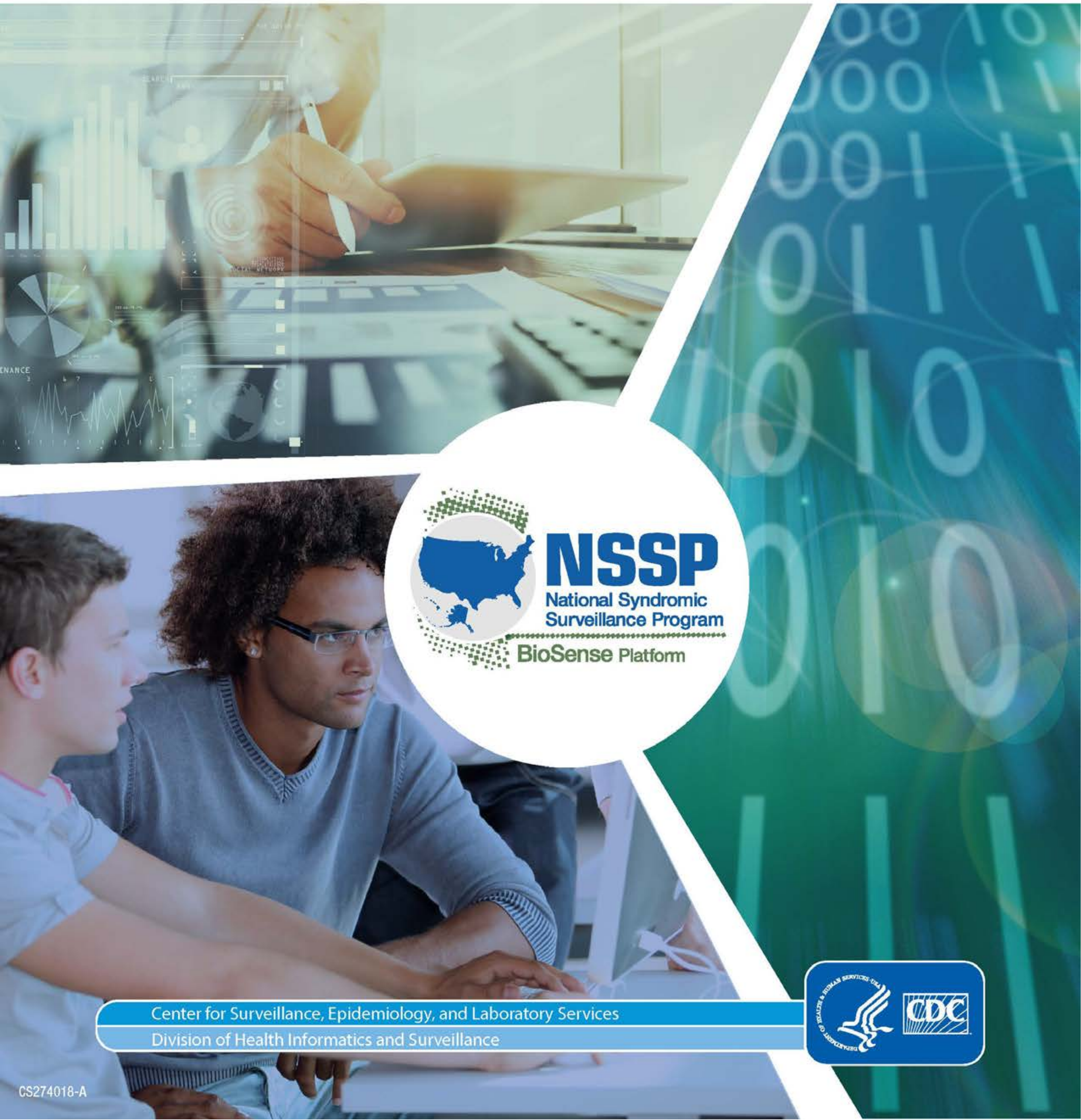
BioSense Platform Quick Start Guide to Using

# RStudio Pro

November 2020



Center for Surveillance, Epidemiology, and Laboratory Services

Division of Health Informatics and Surveillance

CS274018-A

# CONTENTS

**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).

# Quick Start Guide to Using RStudio Pro

## 1. Overview

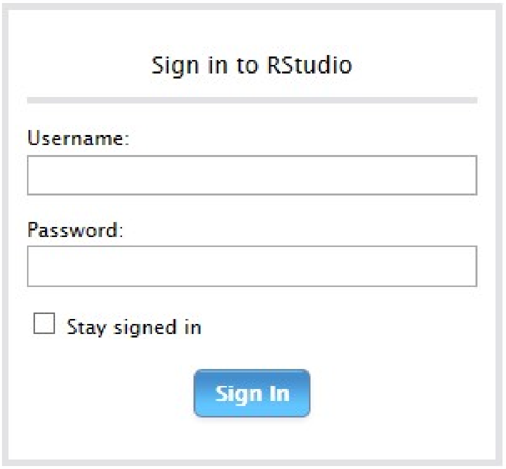
RStudio Pro ("RStudio") lets you access and analyze SQL data stored on the BioSense Platform. You can use RStudio to verify your data within the BioSense Platform archive (raw, processed, and exceptions tables) and confirm information in your Master Facility Table (MFT). RStudio lets you access a single database on the BioSense Platform that contains views into all of your site's data. When using RStudio, you can view and query multiple databases and data tables that contain data for your site.

### *Accessing RStudio*

You can request access to RStudio by having your site administrator create a ticket in the NSSP Service Desk (http://support.syndromicsurveillance.org). All requests for access must be approved by your site administrator. All platform users must have a production Access & Management Center (AMC) user account. The AMC application creates the user in the active directory that is required for RStudio and other analysis tool access. The site administrator must first create the AMC user account before access to RStudio can be provisioned.

### *Logging in to RStudio*

1. Go to the RStudio sign-in page: https://webquery.syndromicsurveillance.org/rstudio/
2. Enter the user ID.
3. Enter the temporary password.



The RStudio username and password are the same as the AMC/ESSENCE username and password.

The username you will use is biosense\[username].

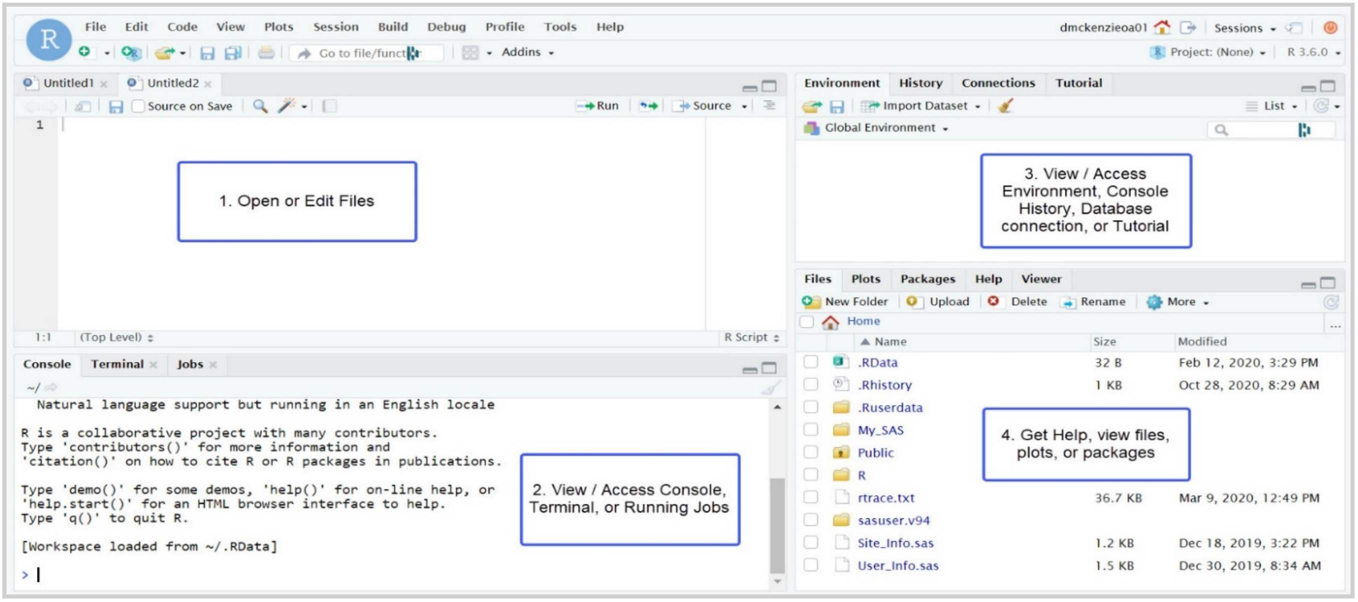
### *Changing your RStudio Password in Active Directory*

Your password needs to be reset every 90 days. Users will get emails from the AMC. Your password can be reset at https://amc.syndromicsurveillance.org/. If your password expires, you will not be able to log into RStudio or other tools and must reset your password in the AMC to restore access.

## 2. Basic Navigation

### *Overview*

The RStudio home pane is shown below (figure 1). Numerous online resources are available to help you learn about and become comfortable with the RStudio tool. RStudio's website includes many resources to explore: https://www.rstudio.com/online-learning/#R



```
~/
 Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from ~/.RData]

>
```

| Name | Size | Modified |
|---|---|---|
| .RData | 32 B | Feb 12, 2020, 3:29 PM |
| .Rhistory | 1 KB | Oct 28, 2020, 8:29 AM |
| .Ruserdata | | |
| My_SAS | | |
| Public | | |
| R | | |
| rtrace.txt | 36.7 KB | Mar 9, 2020, 12:49 PM |
| sasuser.v94 | | |
| Site_Info.sas | 1.2 KB | Dec 18, 2019, 3:22 PM |
| User_Info.sas | 1.5 KB | Dec 30, 2019, 8:34 AM |

*Figure 1. RStudio Navigation Guide*

1. The upper left part of the screen is a space in which you can open the files you create or edit (such as an R script). To create a new R script, select File, click "New File," then click "R Script"—or, click the icon of a document with a plus sign (located directly below the top menu).
2. The lower left part of the screen shows the **console** window. You may type commands into the console window and see output or logs from programs previously run.
3. The upper right part of the screen is the **environment** (or **workspace**). The environment lists all active objects, values, functions, or anything else you've created. The history tab within the environment window keeps a record of previous commands.
4. The bottom right part of the screen contains different tabs:
   - **Files** tab shows all files and folders available in your workspace.
   - **Plots** tab displays any plots you generate.
   - **Packages** tab contains a limited list of packages available for installation. Any package with a check next to it has been installed; if a package does not contain a check, those package functions are not usable.
   - **Help** tab can be used to search for additional information on a package or function.

For new users becoming familiar with R, the RStudio Essentials webinar will be particularly helpful: https://www.rstudio.com/resources/webinars/rstudio-essentials-webinar-series-part-1/ (starting at 5:44).
***Note.*** You do NOT need to download additional software to use the BioSense Platform web-based RStudio. This presentation will guide you through the different windows in RStudio and will also help you begin to write code.

## *Database Tables*

This section describes the tables you can access by using RStudio. Replace the "XX" with your site's abbreviation. If you have been granted access by your site administrator, you will have access to both production and staging versions of your data tables. *Production tables* contain production-quality data sent to the BioSense Platform. *Staging tables* contain staging, or "test," data that you may use during the facility and feed onboarding process. For questions about onboarding, contact the NSSP Service Desk at http://support.syndromicsurveillance.org.

You may want 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_Raw**: the raw messages that will be processed against the new data flow
- **XX_ST_Processed**: the processed messages from the raw table
- **XX_ST_Except** and related tables: the exceptions messages from the raw table
- **Filter_Reason**: join to XX_ST_Raw to understand why a record was filtered (i.e., not attempted to be processed)
- **Exceptions_Reason**: join to XX_ST_Except_Reason to understand why a record was exceptioned

**Names and Descriptions of Tables Available**

| | |
|---|---|
| **Facility Information** | |
| **XX_MFT** | Contains current Master Facility Table (MFT).<br>*To update or modify your MFT, contact the NSSP Service Desk.* |
| **XX_MFT_Except** | Contains facility records that could not 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 message you delivered to the platform and some metadata about that message. 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 message 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. |
| **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.<br>*To understand why a message appears in the XX_PR_Except table, you may want to join to the XX_PR_Except_Reason table.* |
| **XX_PR_Except_Reasons** | Contains the message_ID and any number of reasons for placing that record in the XX_PR_Except table.<br>*To view the exception code descriptive values, join to Exceptions_Reason table.* |

## Names and Descriptions of Tables Available *(continued)*

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

| | |
|---|---|
| **XX_ST_Raw** | Contains the original message you delivered to the platform and some metadata about that message. 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 message 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_ST_Except. |
| **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. *To understand why a message appears in the XX_ST_Except table, you may want to join to the XX_ST_Except_Reason table.* |
| **XX_ST_Except_Reasons** | Contains the message_ID and all the reasons for placing the record in the XX_ST_Except table. *To view the exception code descriptive values, join to Except_Reason table.* |

### Reference Tables

| | |
|---|---|
| **Filter_Reasons** | Maps the Filter Reason Code found in the raw table to its descriptive text. |
| **Exceptions_Reasons** | Maps the Exception Reason Code found in the Exceptions_Reasons table to its descriptive text. |

Figure 2 shows an Entity Relationship Diagram of available BioSense Platform archive tables. You may find this helpful as you join tables to perform advanced queries.

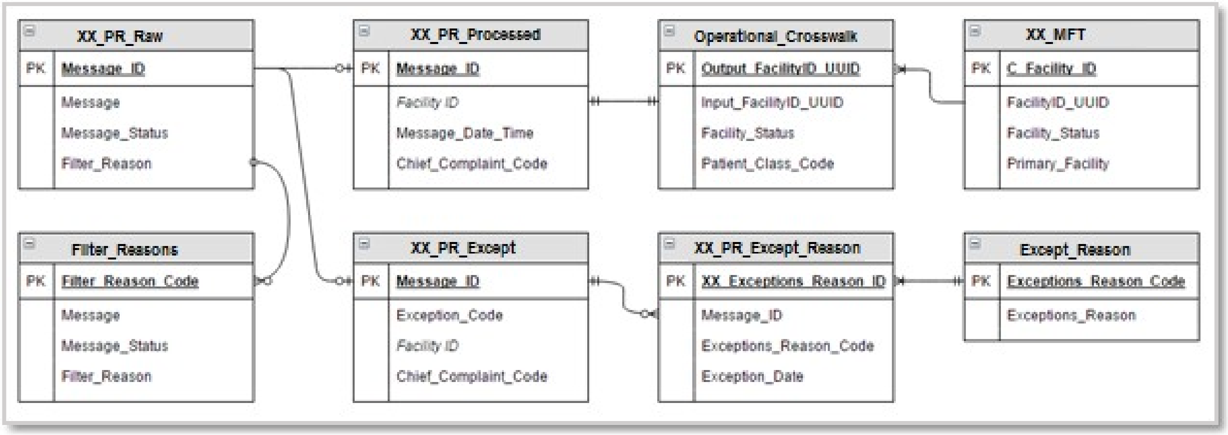


*Figure 2. Entity Relationship Diagram of Archive Tables*

# 3. Features

To see RStudio Pro features, click here: https://www.rstudio.com/products/rstudio-server-pro/.

## *Project Sharing*

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

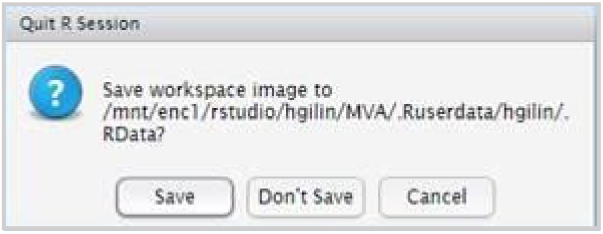
***Note.*** To share projects, your system must meet the prerequisites described here. To learn more about project sharing, click here: https://support.rstudio.com/hc/en-us/articles/211659737-Sharing-Projects-in-RStudioServer-Pro.

**Shared Project Cautions:**

Users who share projects can inadvertently make their user logins and jurisdiction-protected health information ***visible to all other Shared Project users.***

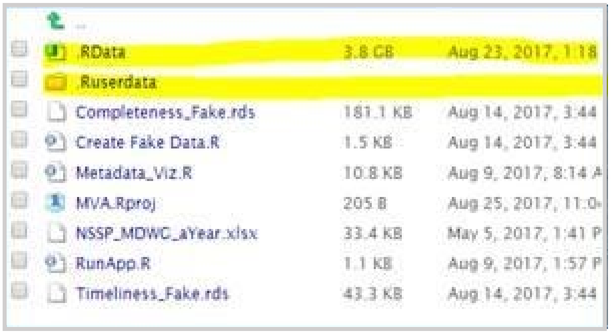
Whenever a user accesses the Shared Project on the NSSP RStudio server, that user can do any R computations normally available. In other words, the user is able to use and store logins in a connection string and query BioSense Platform data to retrieve the jurisdiction's protected health information.

**If an NSSP RStudio server user does computations in the shared project directory, quits the R session, AND chooses "Save" when prompted by the message below** (figure 3)**, the user makes all information (objects) saved available to everyone with access to the shared project directory. This includes logins and jurisdiction-protected health information.** The sensitive information that would have been accessible to other shared project users will be in a file named .RData or in a folder named .Ruserdata (figure 4).



Save workspace image to /mnt/enc1/rstudio/hgilin/MVA/.Ruserdata/hgilin/.RData?

*Figure 3. Clicking "Save" makes information available to everyone with access to the shared project directory.*



| | | |
|---|---|---|
| .. | | |
| .RData | 3.8 GB | Aug 23, 2017, 1:18 |
| .Ruserdata | | |
| Completeness_Fake.rds | 181.1 KB | Aug 14, 2017, 3:44 |
| Create Fake Data.R | 1.5 KB | Aug 14, 2017, 3:44 |
| Metadata_Viz.R | 10.8 KB | Aug 9, 2017, 8:14 A |
| MVA.Rproj | 205 B | Aug 25, 2017, 11:0 |
| NSSP_MDWG_aYear.xlsx | 33.4 KB | May 5, 2017, 1:41 P |
| RunApp.R | 1.1 KB | Aug 9, 2017, 1:57 P |
| Timeliness_Fake.rds | 43.3 KB | Aug 14, 2017, 3:44 |

*Figure 4. Storage location for sensitive information.*

**To avoid this issue**, *never* use the Shared Project Directory for work that will *not* be shared. Instead, do your work in another session area that's private. Here's how:

1. Toward the middle of the RStudio menu is a tab called "Session." Click "Session," and then click "New Session" on the drop-down menu (figure 5).

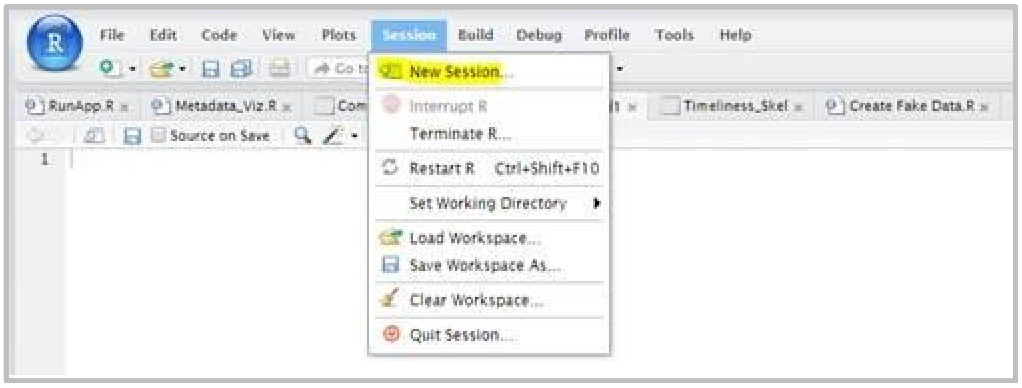
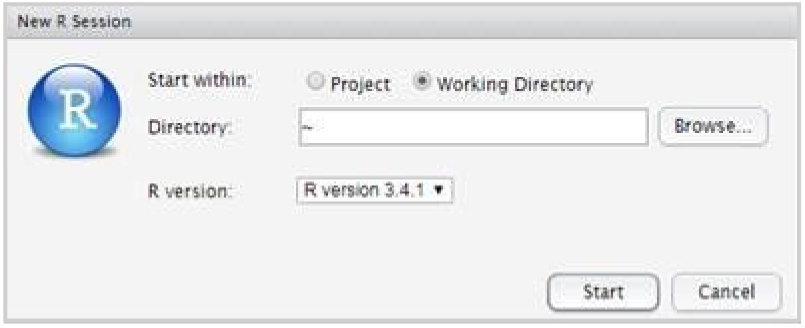


*Figure 3. New Session.*

2. Click "Working Directory," and then click "Start" (figure 6).


New R Session

Start within: Project [✓ Working Directory]

Directory: ~

R version: R version 3.4.1

*Figure 6. Opening a new private session.*

This will open a blank new session in your Home directory, which will likely be empty if you haven't done work there already. **Only you have access to your Home directory**. So, you can do work and store whatever you want without others seeing it. You can also save your workspace here without risk of exposing sensitive data.

If you do not close either session, Home and Shared Sessions will remain open. You can navigate between both by clicking the "Sessions" button at the top right of the RStudio menu. Here, you can see which sessions are open and navigate between them by clicking on either.

***Note.*** **RStudio projects stored in your Home directory cannot be shared.** This is an unfortunate technical limitation of RStudio and the current network file-sharing protocols. However, **NSSP provides a location on the WEBQUERY server to support shared projects.**

***Steps for Creating a Shared Project in RStudio***

You can create your shared RStudio projects in subdirectories of **/mnt/enc1/rstudio** on the WEBQUERY server. First, you'll need to set up a subdirectory that matches your login name by following these steps:

1. Click the three dots (figure 7).
2. Navigate from RStudio to /mnt/enc1/rstudio/ (figure 7).

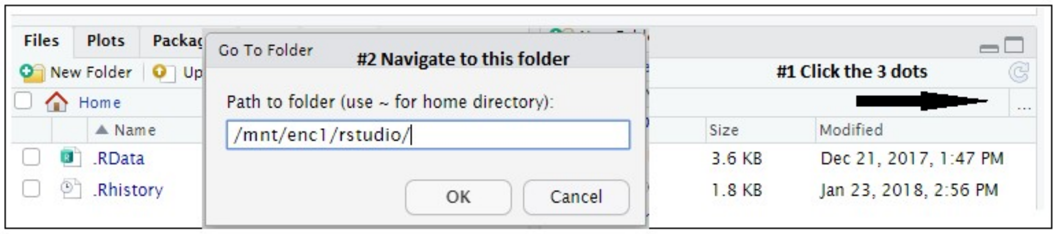


*Figure 7. Create a subdirectory*

3. Click "New Folder" to create a new <*yourusername*> in this location (figure 8):

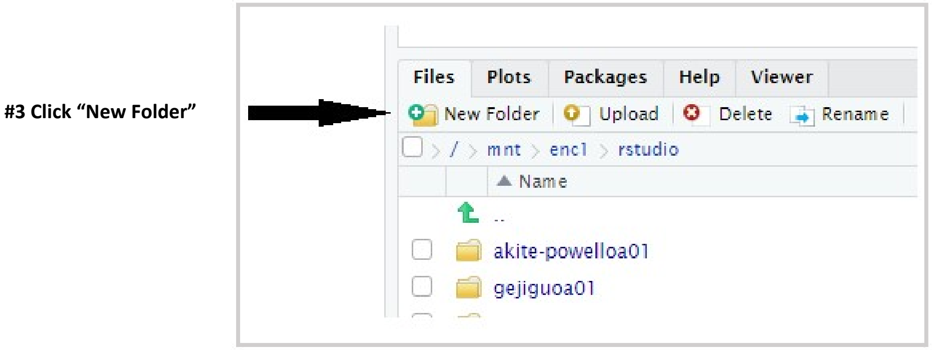


Figure 8. Add new folder to subdirectory.

4. Click "Refresh" to see the pane that contains your new Shared Projects folder.

***Steps for Creating Specific Shared Projects Inside the Subdirectory***

For multiple shared projects, you'll want to set up project-specific folders inside the subdirectory. Here's how:

1. Navigate from RStudio to /mnt/enc1/rstudio/<*yourusername*>
2. Create shared projects and project folders as needed, and
3. Copy necessary programs and data files into the project folder.

Keep in mind that the /mnt/enc1/rstudio directory on the WEBQUERY server is *local* storage. Local storage:

- Is readable and writable by all users, and
- Supports all requirements for RStudio shared projects.

Whenever you create a personal subdirectory in this location, be aware that:

- You control the subdirectory and all of its contents.
- By default, other users can read these files but cannot change the content. ▪ You can change permissions on the files to prevent direct access by other users.

## *Multiple Versions of R*

RStudio lets you select multiple versions of R from within the integrated development environment, or IDE, which can be useful for the following tasks:

- Manage R upgrades
- Test code on a variety of R versions and distributions
- Support projects that depend on various versions of R

**Multiple R Versions**

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

https://support.rstudio.com/hc/en-us/articles/212364537-MultipleVersions-of-R-in-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
- Participate in one or more shared projects

**Multiple R Sessions**

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

https://support.rstudio.com/hc/en-us/articles/211789298-Multiple-RSessions-in-RStudio-Server-Pro

## *User-installed RStudio Pro Packages*

In addition to packages installed and available for immediate use,
you may install additional R-compatible packages by running the following code (altering PackageName to the case-sensitive name of your project):

```
install.packages(“PackageName”)
```

***Note.*** Even when a package has been installed, running the following code at the beginning of your programs to call a package is a good practice:

```
library(PackageName)
```

## *Currently Installed R Packages for All Users*

| | | | | | |
|---|---|---|---|---|---|
| abind | dplyr | irlba | pkgmaker | RODBC | udunits2 |
| acepack | DT | ISOcodes | plogr | rpart | units |
| acs | epitools | iterators | plotly | rprojroot | utf8 |
| assertthat | evaluate | itertools | plyr | Rserve | utils |
| backports | dplyr | janeaustenr | png | RSQLite | uuid |
| base | feather | jpeg | polyclip | rstudioapi | V8 |
| base64enc | forcats | jsonlite | polyCub | rvest | viridis |
| BH | foreach | keras | praise | rvg | viridisLite |
| bindr | foreign | KernSmooth | prettyunits | scales | WDI |
| bindrcpp | formatR | knitr | processx | selectr | whisker |
| bit | Formula | labeling | progress | shiny | widyr |
| bit64 | gdtools | lattice | proto | slam | withr |
| bitops | geosphere | latticeExtra | psych | SnowballC | XML |
| blob | GGally | lazyeval | purrr | sourcetools | xml2 |
| boot | ggforce | leaflet | quantmod | sp | xtable |
| broom | ggiraph | lubridate | R.methodsS3 | spatial | xts |
| callr | ggmap | magrittr | R.oo | spatstat | yaml |
| caTools | ggplot2 | mailR | R.utils | spatstat.data | zeallot |
| cellranger | ggraph | mapproj | R6 | spatstat.utils | zip |
| checkmate | ggrepel | maps | rappdirs | splines | zoo |
| choroplethr | ggthemes | maptools | raster | sqldf | |
| choroplethrMaps | ggvis | markdown | RColorBrewer | stats | |
| chron | git2r | MASS | Rcpp | stats4 | |
| class | glue | Matrix | RCurl | stopwords | |
| cli | goftest | memoise | readr | stringi | |
| cluster | graphics | methods | readxl | stringr | |
| codetools | grDevices | mgcv | registry | surveillance | |
| colorspace | grid | mime | rematch | survival | |
| compiler | gridBase | MMWRweek | reprex | tcltk | |
| config | gridExtra | mnormt | reshape | tensor | |
| crayon | gsubfn | modelr | reshape2 | tensorflow | |
| crosstalk | gtable | munsell | reticulate | testthat | |
| curl | haven | networkD3 | rex | tfruns | |
| data.table | hexbin | nlme | rgdal | tibble | |
| dataframes2xls | highr | NLP | rgeos | tidyr | |
| datasets | Hmisc | NMF | RgoogleMaps | tidyselect | |
| DBI | hms | nnet | rio | tidytext | |
| dbplyr | htmlTable | odbc | rJava | tidyverse | |
| debugme | htmltools | officer | rjson | tinytex | |
| deldir | htmlwidgets | openssl | RJSONIO | tm | |
| devtools | httpuv | openxlsx | rlang | tokenizers | |
| dichromat | httr | parallel | rmarkdown | tools | |
| digest | hunspell | pillar | RMySQL | TTR | |
| doParallel | igraph | pkgconfig | rngtools | tweenr | |

## 4. How to Connect to Data

Data for a given site is located in tables on the DataMart server on the BioSense Platform database. RStudio uses an odbcConnect string to connect to the DataMart and to access these tables. The following code may be used to connect to new BioSense Platform data in the DataMart:

```
library(RODBC)
odbcConnect("BioSense_Platform", "BIOSENSE\\USERNAME", "PASSWORD")
```

Remember to use your username and password in the connection string. ***Note.*** The username and password are the same for the AMC, RStudio, and ESSENCE.

# Appendix—Introduction to RStudio SQL Query Examples

**Query Limitations**

A good practice is to limit query results to 100,000 records or less. If downloading more than 1 year of data, it is recommended to download in smaller increments.

**Memory Limitations**

Tools on the BioSense Platform are shared resources. Please keep the following in mind:

- Queries will continue to run after a user closes a window.
- Queries running longer than 72 hours may be terminated if the performance of other resources is degraded.

## Calls RODBC Package (required to connect to database)

```
library(RODBC)
```

## Builds Connection String to Database

```
con <- odbcConnect("Biosense_Platform","BIOSENSE\\USERNAME","PASSWORD")

#You should also close any open connections when you are done connecting to the server by using the following code:  close(con)
```

## List Tables to Which User Has Access

In general, users have access to the following tables (replace XX with your site abbreviation):

- XX_MFT
- XX_Operational_Crosswalk
- XX_PR_Except
- XX_PR_Except_Reason
- XX_PR_Processed
- XX_PR_Raw
- XX_Site_Contacts
- XX_ST_Except
- XX_ST_Except_Reason
- XX_ST_Processed
- XX_ST_Raw
- Except_Reasons
- Filter_Reasons

## List Accessible Tables

This code will create an object df.tables that contains a list of the tables to which a user has access:

```
#Call the RODBC Package
library(RODBC)

#Build the connection to your database
con <- odbcConnect("Biosense_Platform","BIOSENSE\\USERNAME","PASSWORD")

#Create list of accessible tables as object df.tables
df.tables<-subset(sqlTables(con), TABLE_SCHEM=='dbo')

# Closes connection
close(con)
```

## Query Database Table

It is important to note that R uses SQL language to query SQL tables. You may run any SQL query in R by altering the code to use a query of your choice.

```
#Call the RODBC Package
library(RODBC)

#Build the connection to your database
con <- odbcConnect("Biosense_Platform","BIOSENSE\\USERNAME","PASSWORD")

#Develop the query to be run
#In this example, we create an object called “query” that is a query to pull 100 records from the
XX_PR_Processed table with Arrived_Date between 1/1/17 and 1/2/17

query <- paste("select top 100 * from [BioSense_Platform].dbo.XX_PR_Processed
          WHERE Arrived_Date_Time between '2017-01-01' and '2017-01-02';",
          sep="")

#Run the query using the connection string and put the results in an object called “table”
table <- sqlQuery(con, query)

# Closes connection
close(con)
```

## MS SQL Query Examples

These queries can be copied and pasted into the query section of the previous example to produce the same results using RStudio.

## Collapse Records in Table into Visits

```
Unique_row_counts <-table%>% group_by(variable)%>%summarise(n=n())
```

Or

```
library(sqldf)
Unique_row_counts<-sqldf('select "variable", count(*) from "table" group by variable')
```

## Output Object to .csv File

```
write.csv(table, file="FileName.csv", row.names = FALSE)
```

## Indexes Available on DataMart

Indexes are used to expedite database queries. Normally, a table that contains no indexes must search the entire table in a linear fashion. When an index is added, the query will instead only search a subset of the data, which increases efficiency. We have added indexes to the columns listed below to help make your analysis more efficient.

| DataMart Indexes | | | | |
|---|---|---|---|---|
| **Column Name** | **Production** | | **Staging** | |
| | **Processed** | **Raw** | **Processed** | **Raw** |
| **Arrived_Date_Time** | Yes | Yes | Yes | Yes |
| **Arrived_Date** | Yes | Yes | Yes | Yes |
| **Create_Raw_Date_Time** | No | Yes | No | Yes |
| **Feed_Name** | Yes | Yes | Yes | Yes |
| **Message_ID** | Yes | Yes | Yes | Yes |
| **Message_Status** | No | Yes | No | Yes |
| **Update_Processed** | No | Yes | No | Yes |
| **C_Visit_Date_Time** | Yes | No | Yes | No |
| **C_Visit_Date** | Yes | No | Yes | No |
| **C_BioSense_Facility_ID** | Yes | No | Yes | No |
| **C_BioSense_ID** | Yes | No | Yes | No |
| **C_Facility_ID** | Yes | No | Yes | No |
| **Message_Date_Time** | Yes | No | Yes | No |
| **Message_Date** | Yes | No | Yes | No |
| **C_Unique_Patient_ID** | Yes | No | Yes | No |
| **Legacy_Flag** | Yes | No | Yes | No |
| **Create_Processed_Date_Time** | Yes | No | Yes | No |
| **Processed_ID** | Yes | No | Yes | No |
| **Site_ID** | Yes | No | Yes | No |
| **Update_Essence** | Yes | No | Yes | No |
| **Legacy_Row_Number** | No | No | Yes | No |

## Using Arrived_Date_Time vs Arrived_Date

**Arrived_Date**—This index is derived from the Arrived_Date_Time data using *only* the date portion. Users should see improved performance for queries using Arrived_Date compared with functionally identical queries using Arrived_Date_Time.

For example, this query:

`sum(XX_PR_RAW!Arrived_Date == ‘2016-03-11’)` is almost **twice as efficient** as this one:

```
sum(XX_PR_RAW!Arrived_Date_Time >= ‘2016-03-11’ & XX_PR_Raw!Arrived_Date_Time <
‘201603-12’)
```