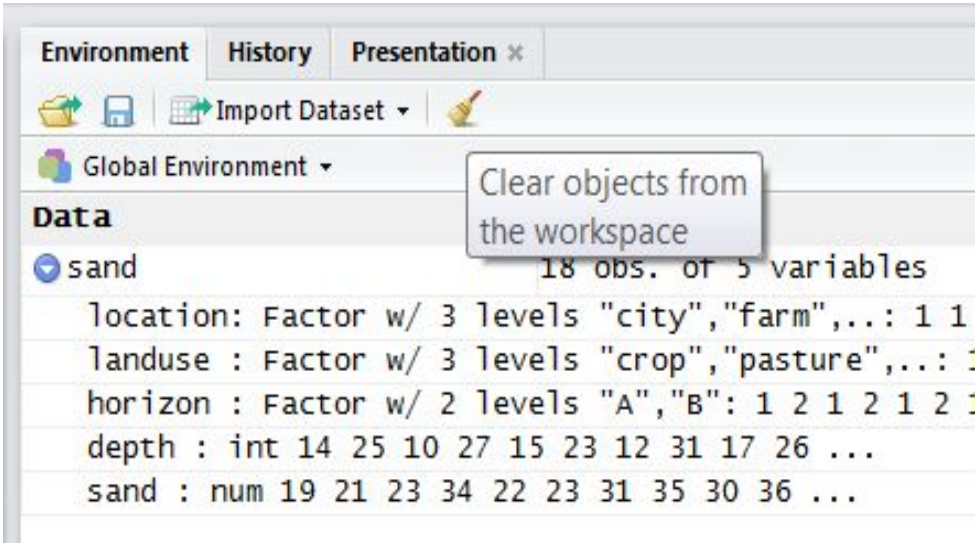
Statistics Documentation

1. Introduction

RStudio is an awesome tool that can help you do your work better and faster, also RStudio is a is a cross-platform integrated development environment (IDE) for R, a programming language for statistical computing and graphics.

2. Explaining the Choices

Because R Studio is such a powerful tool, built specifically for statistics, we decided to harvest its' power and use it to deliver meaningful statistics to our clients. It allows you to see a full list of objects you are currently working with, the interaction is natural (for example, by just double clicking one of the objects, you get a graphical representation of it, or you can easily click back and forth between plots, change the sizes of your plot without rerunning the code, and export or copy plots to include in other documents).

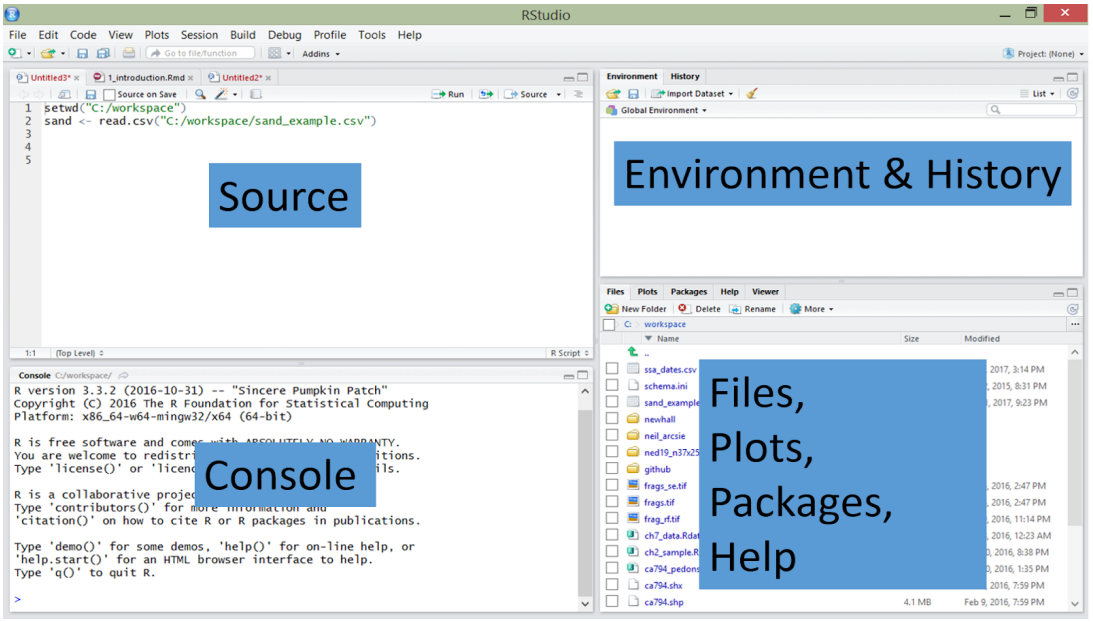


R Studio is also useful because of its' cross-platform interface, allowing you to get a seamless experience on both Windows and MacOS, reducing the differences between operating systems, so users don't get confused when switching between platforms.

Although R comes with many common statistical functions and models, most of our work requires aditional packages witch R allows us to include to our project.

RStudio is designed to make it easy to write your scripts. For exemple as soon as you create a new script, the windos within your RStudio session adjust automatically so you can view both the script and the results you are getting from the console when you run your syntax.

R Studio is not only powerful, it is also free to use, making it perfect for delivering statistics, while also saving money, leaving room for profit. It is also a full-featured text editor, meaning it includes syntax highlighting, parenthesis and bracket matching, find/replace with regular expressions and so on.



3. Methods Implemented

In the following text we will enumerate the columns which will be used in statistics from each table. Columns will either be used alone or in combinations to provide more relevant statistics.

In table Patients: Sex, Age, U/R, Gastritis, Tumor Location, Stage of Disease, Patological Anatomy, IHC, Degree of Differention, Distance, Metastasis, Lymphadenopathy, Tabacosis, Ethanol Consumption, Diagnostic Imagistic, Anemia, Trombocytopenia, Trombocytosis, Leucocytosis, Immune Deficiency, Sepsis, BPOC, TBC Aftermath, Respiratory Failure, AOMI, HTA, Aortic Autorematosis, Ischaemic Heart Failure, SInus Tachycardia, Cardiac Insufficiency, AVC Aftermath, DZ, Obesity, Nutrition Disorders, Candida, Chronic Hypertrophic Rhinitis, Epidermoid Operated Cyst, Brachial Operated Cyst, Facial Nerve Paralysis, Acute Lymphadenitis (of face, neck and head), Toxic Chronic Hepatitis, Viral C Hepatitis, Hepatic Statosis, Lung Cancer, Lip Carcinoma, Pelvis Operated and Chemo-treated Tumor, Arm Meningioma, Mucositis, Oedema (lateral wall), Renal Failure, Treatment, Surgery, Radiotherapy.

In table Predictions: The number of pacients having each disease, the most applied treatments and their success rate, on each disease the rate of relapseness (in percentages), top 5 most used treatments, the state of each disease (in percentages, and states can be: light, medium and advanced), based on the range of age a statistic on how frequent a disease is, and a more advanced top 5 most applied treatments, this time applied on the stages of the disease (example: treatment 2 applied on cancer in light stage has a 95% success rate, but on an advanced stage it has 10% success rate and so on).

In table Doctors: Sex, Field, Range of Age, Range of Working Years, Number of Patients, Successful Cases, Doctor of the Month (award for the doctor with the most successful cases in that month).

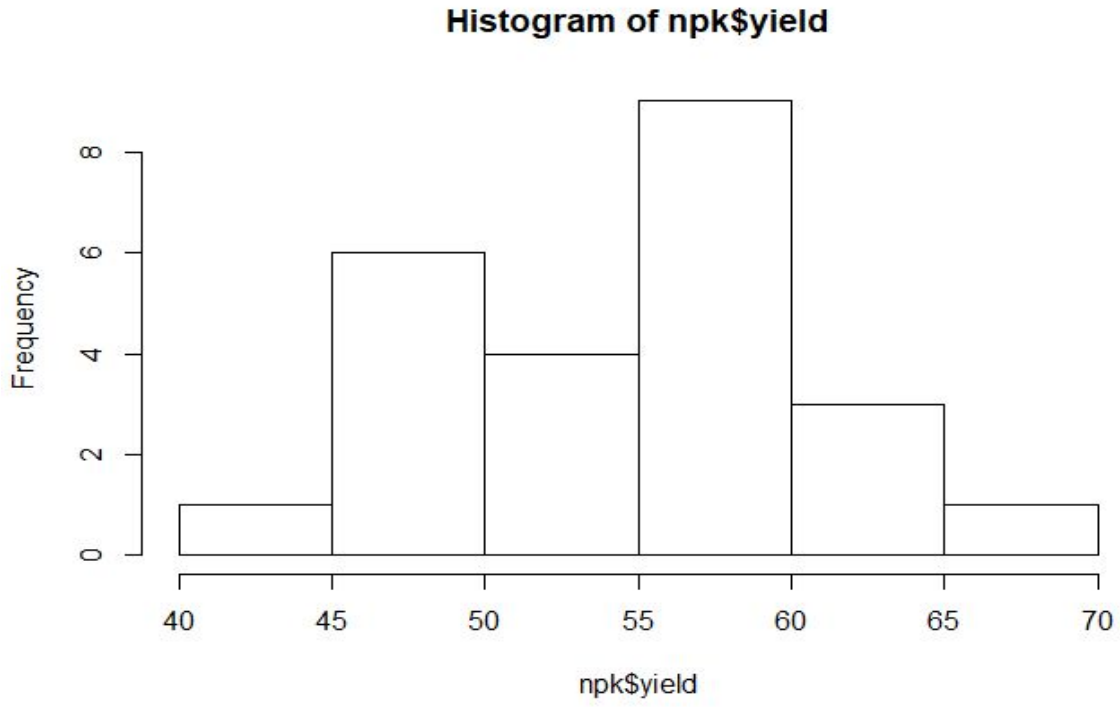
Unit Testing is the first level of software testing and it's performed prior to Integration Testing. The purpose of this level is to check if the code produces the wanted results, and if it does, in what percentage. Some benefits of unit testing include making the code more reusable, it increases confidence in changing/maintaining code, it makes debugging easier and development faster and it makes the code more reliable.

Code coverage, in short, is a measurement to see how many of your lines/blocks/arcs of code are being executed while automated tests are running. Code coverage is collected by using specialized tools/instruments that add tracing calls and run fully automated tests on the code.

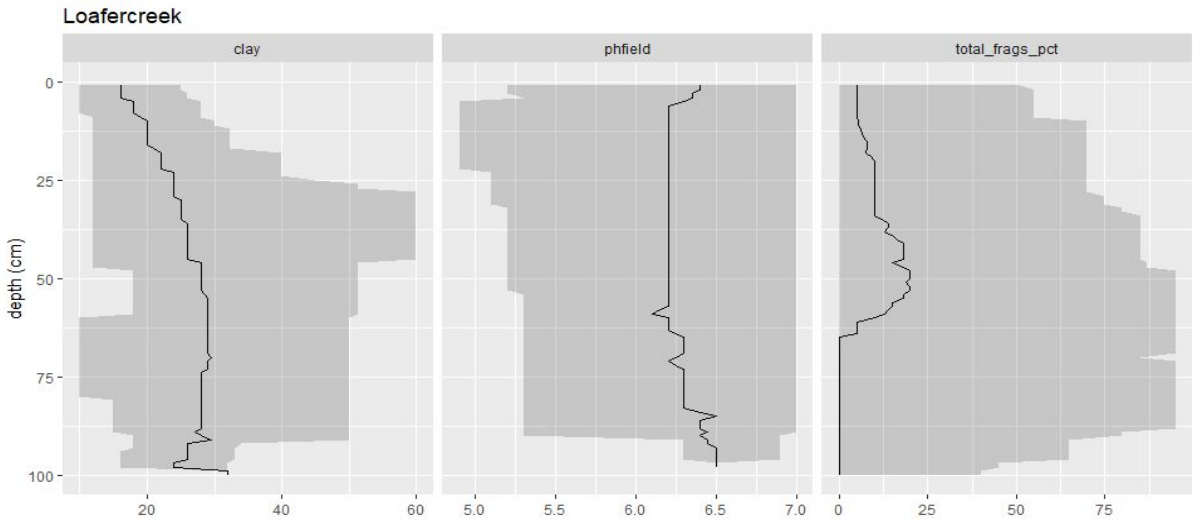
Connections to other teams' databases will be used in order to ensure consistency of information, especially with the Web Module. In order to make a connection with the main database, we will use the Web Module's API to get the database as a JSON file, which we will later convert into a matrix using a script, for ease of use with out R code.

4. Photo Examples of Graphics and Statistics

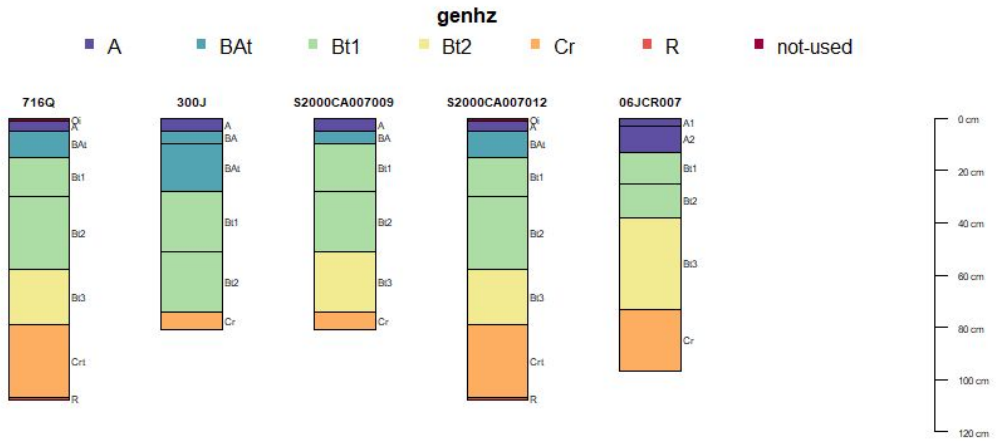
Draw Graphics based on Statistics :



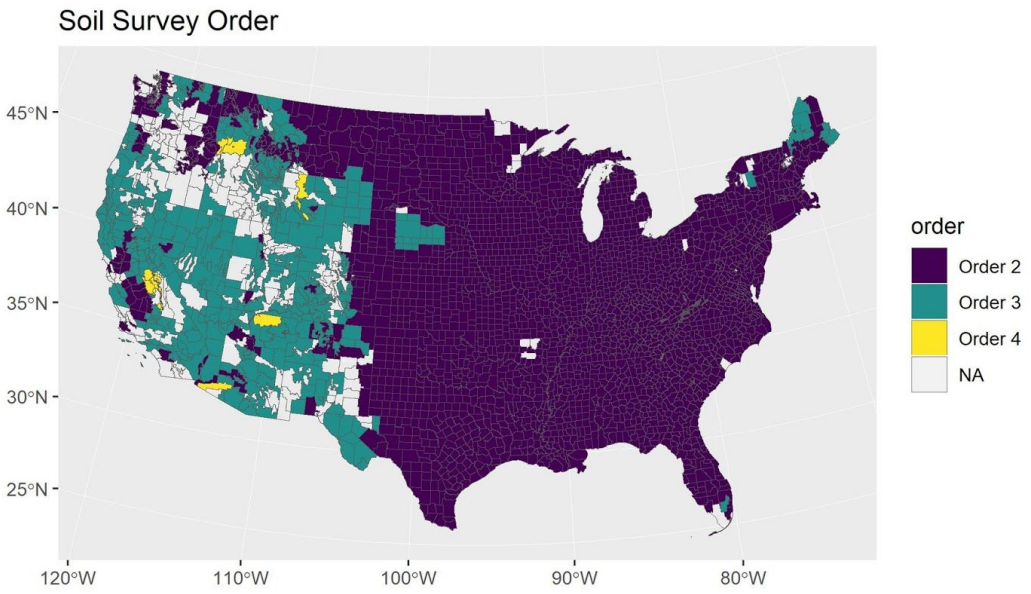
Draw Depth Plots :



Draw Solid Profiles :



Create Maps :



5. Conclusions

Because of the explanations and reasons listed above, we believe that R Studio is perfect for our statistics task. The sheer power of the tool, combined with the fact that it is both open-source and free to use, make it the perfect candidate tool for this job.

Resources:

https://www.r-bloggers.com/top-6-reasons-you-need-to-be-using-rstudio/

https://libguides.library.kent.edu/statconsulting/r?fbclid=IwAR0tGoTq5XV-nwd0ud7uZga6uGpUbLGRTBt0DP3eBuAWdz5oHJCzrlZFRNw

https://www.quora.com/What-is-RStudio-used-for?fbclid=IwAR0LoJE3fwjGyjd0JpjTXLql4LfJHjxhj6yhX1sThD9Ys67ImCgS1kqxR8w

http://softwaretestingfundamentals.com/unit-testing/

https://stackoverflow.com/questions/195008/what-is-code-coverage-and-how-do-you-measure-it