Installing R and RStudio

Installing R

Windows computer

On a Windows PC, go to `http://cran.r-project.org/bin/windows/base/` and click on `Download R x.x.x for Windows (32/64 bit)(x stands for the latest version number, currently it is 3.2.0)`. Rather than copying this link do a Google search for `R`-software, and you should be able to find it. The version number will probably have changed; get the latest version of the code. I highly recommend installing `RStudio` after installing `R`. Think of it as a user interface for `R` (`R` already has a "bare-bones" user interface; `RStudio` adds many powerful features which makes life a lot easier). Go to `www.rstudio.com/ide/` or simply do a search for `RStudio` and find the link. Download the software and follow the installation instructions.

OSX (MAC) machine

Go to `http://cran.r-project.org/bin/macosx/` and click on `R-x.x.x.pkg (x stands for the latest version, currently it is 3.2.0)`. Click on the downloaded link `R-latest.pkg` and follow standard MAC installation procedure. After the `R` installation is complete install `RStudio` for the Macintosh operating system.

Linux Users

For different distributions of Linux visit `http://cran.r-project.org/bin/linux/` and follow the instruction for your operating system.

Installing RStudio

After installing `R`, download the `RStudio` installer via `http://www.rstudio.com/products/rstudio/download/`. Windows, Mac OSX and Linux are supported. Install `RStudio` by double-clicking the downloaded installer. Locate the `RStudio` application and start it, Fig. 1.

Errors installing RStudio

If you encounter any error conditions with installing the software, *always obtain a screen shot of the error*, and send it to xuetong.zhai@gmail.com. To get a screen shot, used the `Grab` tool in the Mac (do a search for it using the magnifying glass icon). For one particular error installing RStudio that we are aware of, see last section, **Known Issues**, of this document.

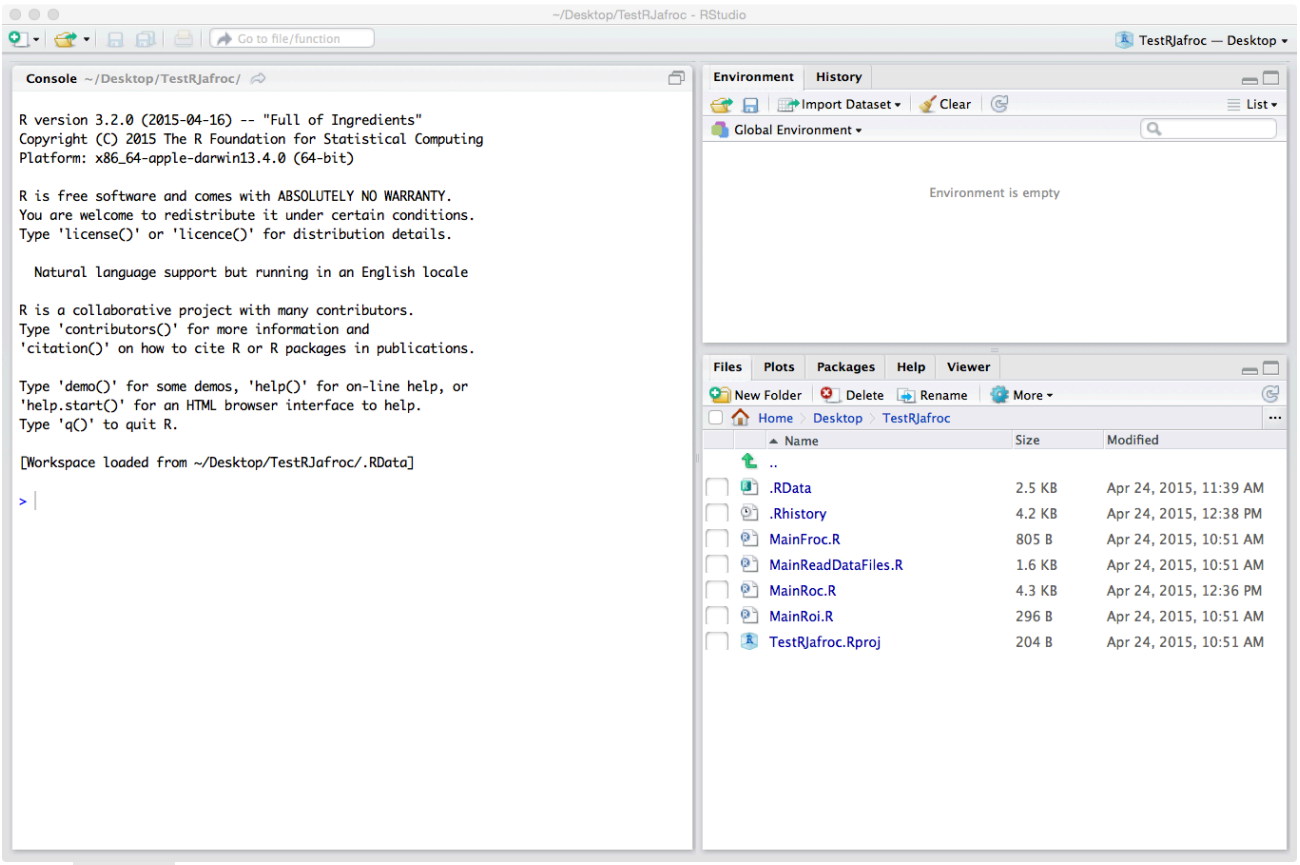


Fig. 1: `RStudio` startup screen.

Selecting the mirror server to download RJafroc

`CRAN` (The Comprehensive R Archive Network) has mirror servers worldwide. To allow us to track downloads, please use the `0-Cloud RStudio` mirror sever. This mirror automatically redirects you to servers worldwide and provides faster download speed and more reliability. Furthermore, download logs of this mirror are available to us. Therefore, we *strongly suggest* using the `0-Cloud RStudio` CRAN mirror. To change your CRAN mirror, select `Tools` → `Global Options` and click on the `Package` tab, Fig. .

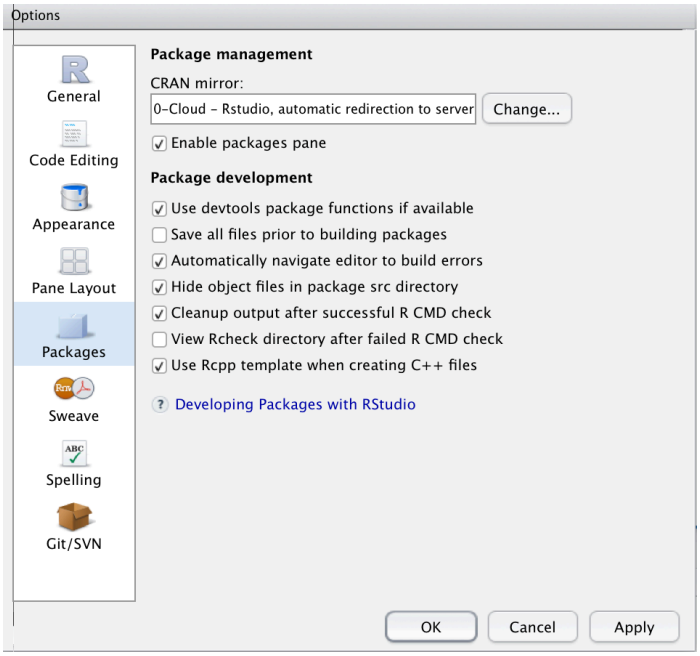


Fig. 2: Changing your CRAN mirror server

Click `Change…`: see Fig. 3.

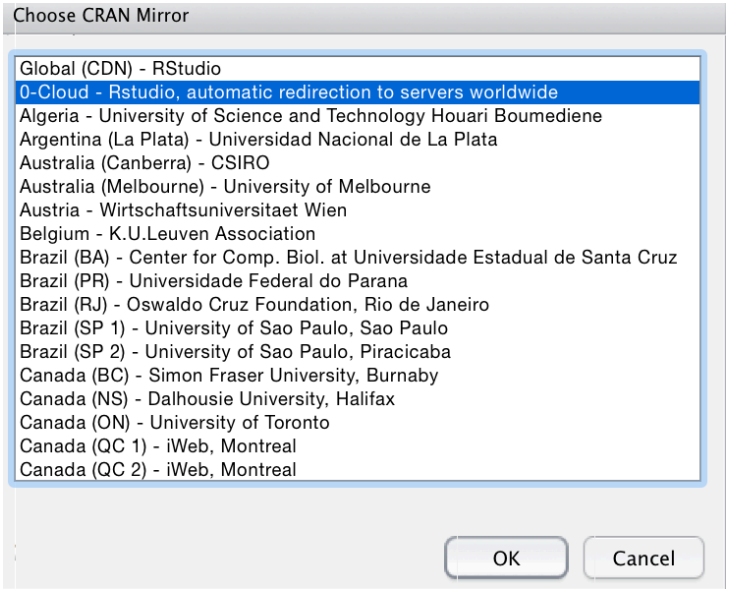


Fig. 3: CRAN mirrors list

Select `0-Cloud - Rstudio, automatic redirection to servers worldwide` and click `OK`. Now you are using the `RStudio` mirror as your `CRAN` server.

Introduction to the RStudio interface

In Fig. 1, notice the organization of the display into 3 major panels (if you see 4 panels, then the top-left window will be showing one or more `.R` files; close them one by one to get to 3). The left panel is labeled `Console` and the right consists of two panels: the top-right panel consists of two sub-panels labeled `Environment` and `History`, and the `Environment` sub-panel is in focus (notice the lighter background of the text `Environment` compared to `History`. The bottom-right panel consists of five sub-panels labeled `Files`, `Plots`, `Packages`, `Help` and `Viewer` and the

`Files` sub-panel is in focus. The actual number of menu items may have changed by the time you see this; simply ignore the extra ones for now. The `Console` panel simply shows the output screen one gets from starting `R`. Try it! Locate `R` using `Spotlight` (the magnifying glass symbol in the very top-right of the MAC display) and open it (i.e., click on it). This is what I see, 4. [On Windows go to `Program` and find the `R` executable file and open it.]

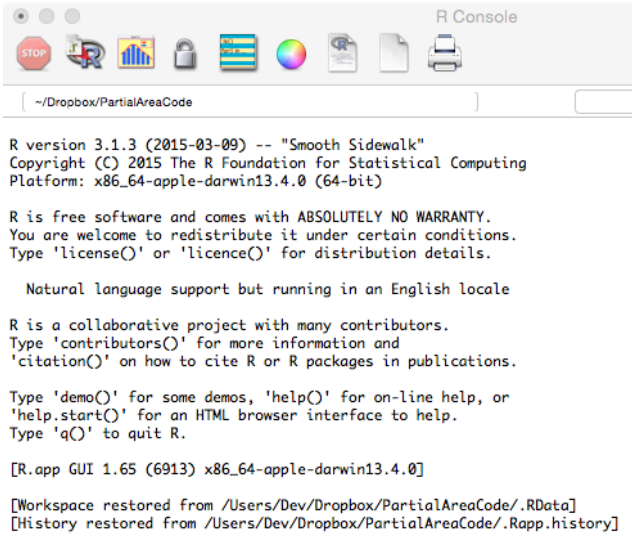


Fig. 4: Opening screen when one runs `R` directly.

Excepting for the last few lines, the two screens (the left panel of Fig. 1 and Fig.) are identical. Now quit `R` - from now on we will always use `RStudio` and never open `R` directly.

Installing RJafroc

Locate the `Packages` tab on the lower-right panel of Fig. 1 and click on it. You should see Fig. 2 (I am just showing the lower right panel).

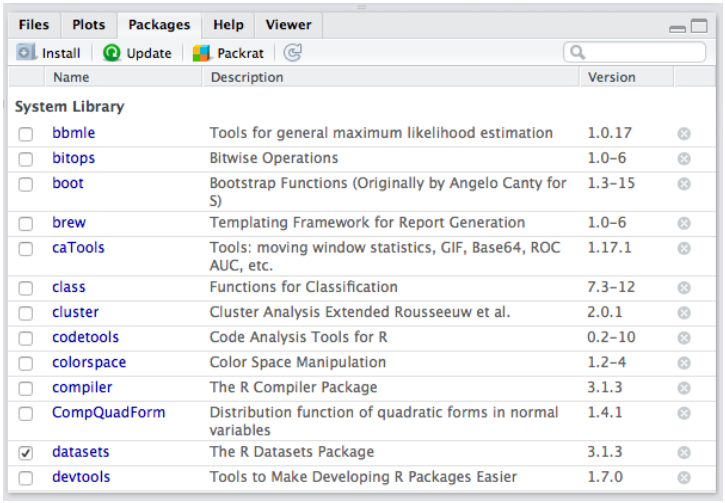


Fig. 2: Installing a new package.

Click on the button that says `Install`. You should see Fig. 3.

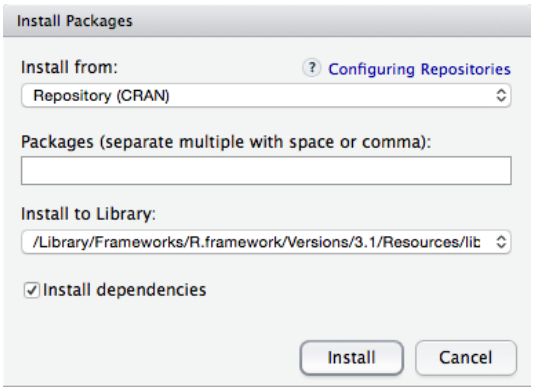


Fig. 3: Installing a new package, continued.

Start typing in `RJafroc`; before you get too far a pop-up menu should appear allowing you to choose the right package, see Fig. 4.

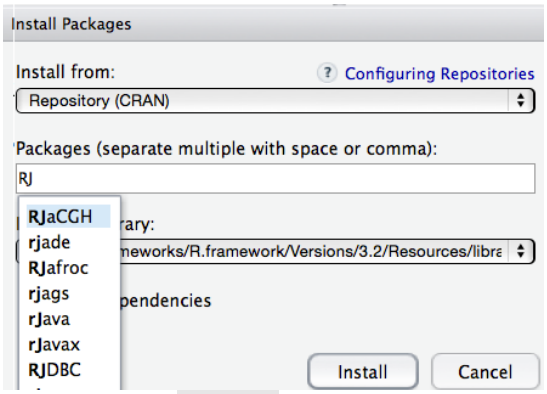


Fig. 4: Installing `RJafroc` package, continued.

Select `RJafroc` and click on `Install`. The `console` window should show some activity and look somewhat like Fig. 5:

Fig. 5: Completion of installation of `RJafroc`.

A simple example using R/RStudio

This section is intended as a gentle introduction to the software. The screen-shots that follow are for my iMAC installation (my operating system is `OS X Yosemite 10.10.2`). If your screen shots look different, don't worry. These are due to differences in platforms (Windows, OSX, Linux, etc.) and version numbers. With a little faith and inquisitiveness you should be able to follow the brief tutorial below.

Download `TestRJafroc.zip` from the JAFROC website (http://www.devchakraborty.com) and unzip the files to your desktop (website downloads require a password created account, if you already don't have one). This should create a folder named `TestRJafroc` on your desktop, Fig. 6.

Fig. 6: Unzip the downloaded file to your desktop creating this folder.

Open this folder, Fig. 7.

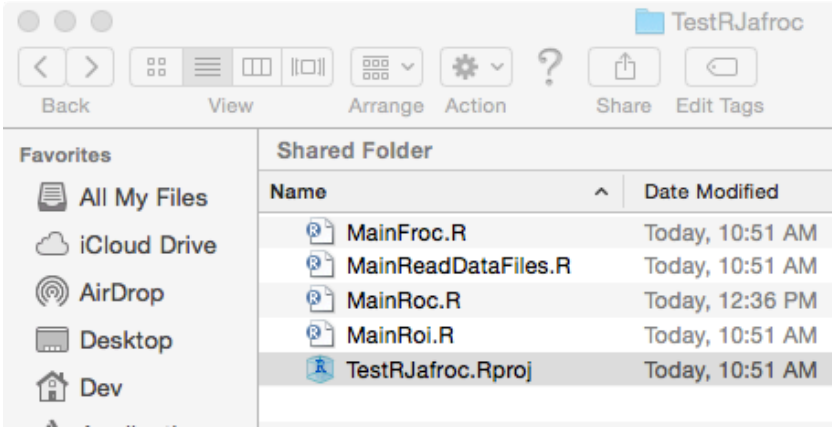


Fig. 7: Contents of the created folder (there could be additional files).

Open the file `TestRJafroc.Rproj` (this is a project file maintained by `RStudio` that keeps track of the files in the project, the startup directory, etc.). You are now in `RStudio` and you initial startup screen may not look exactly like Fig. 1, but don't worry.

Running a R script file

At this point `RJafroc` is installed in your machine (as disk files) but not yet ready for use in `RStudio`. In the lower right panel click on the `Files` button, Fig. 8, where I am showing only the relevant window.

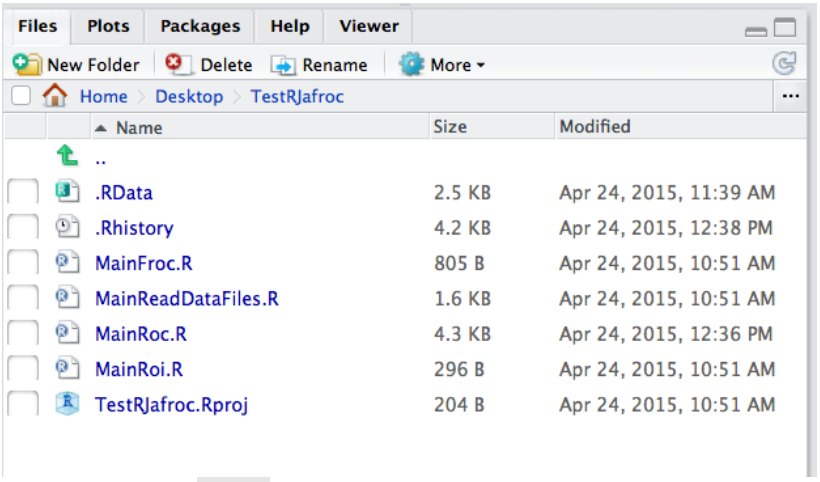


Fig. 8: Contents of `Files` window.

A word on my convention: *any file whose name begins with Main contains directly executable code and if it does not, then it is a function (think subroutine) intended to be called by some other code*. Open the file `MainRoc.R`, Fig. 9.

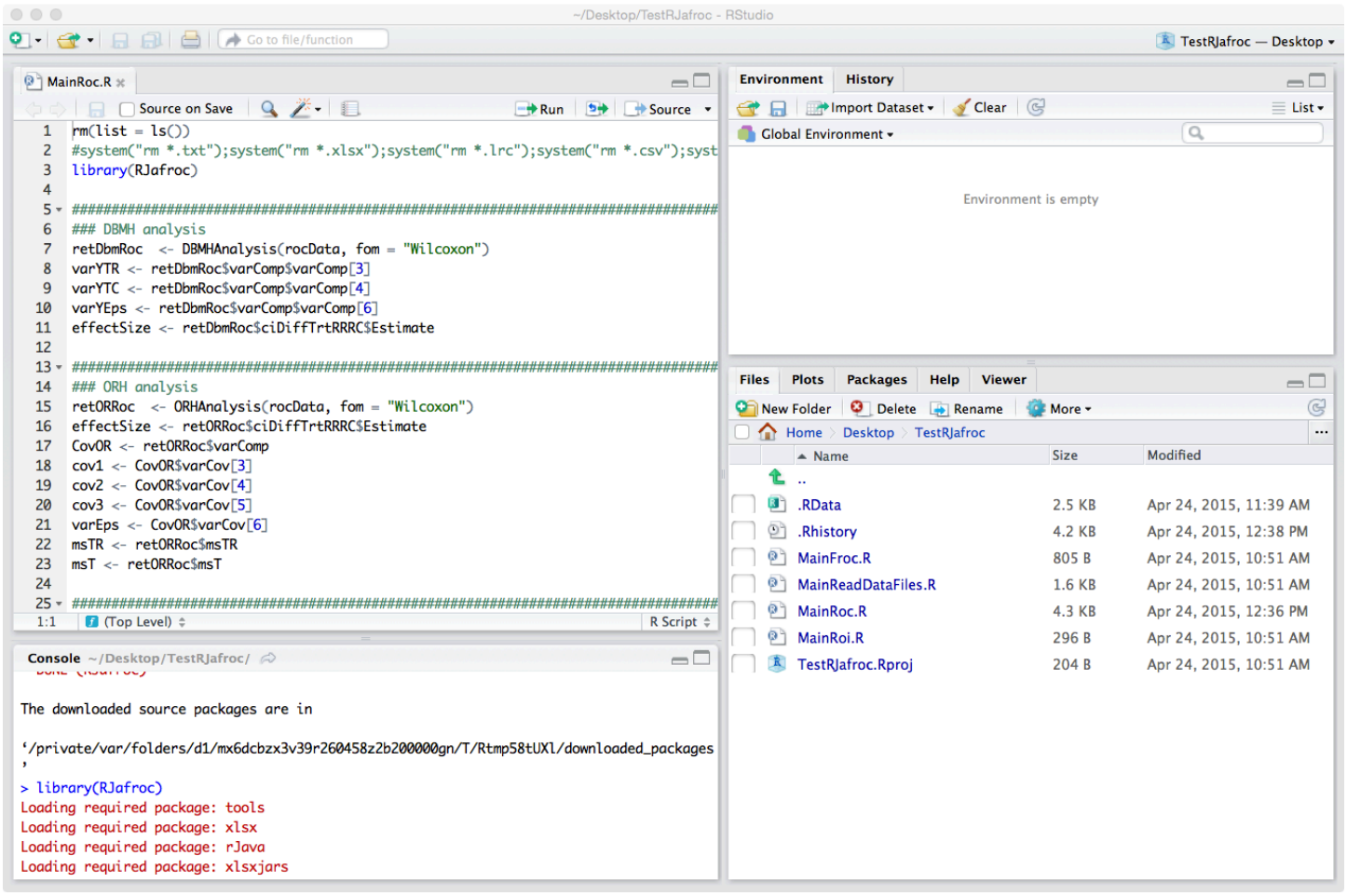


Fig. 9: `RStudio` window with all four panels showing.

Notice that the left `Console` window has split horizontally into two windows: the `Console` panel now occupies the bottom-left `RStudio` window and the file we just opened, `MainRoc.R`, occupies the top-left `RStudio` window; the latter panel organizes your source code files (also called *script* files, these are all files with the `.R` extension) into one window for convenient viewing and editing; we will refer to it as the "*source-code*" window. Whenever you create new code for this project you should save it to the directory indicated on the bottom-right panel. Since the code may be difficult to read in the screen-shot shown in Fig. 9, it is replicated below. To distinguish it easily from regular text, all code will look like the one shown below (i.e., using the same font type and size, background shading and box). A listing of the file `MainRoc.R` follows (Listing 1):

Begin Listing 1

```
rm(list = ls()) # delete all existing variables
#system("rm *.txt");system("rm *.xlsx");system("rm *.lrc");system("rm *.csv");system("rm *.imrmc")
library(RJafroc)

#################################################################################
### DBMH analysis
retDbmRoc  <- DBMHAnalysis(rocData, fom = "Wilcoxon")
varYTR <- retDbmRoc$varComp$varComp[3]
varYTC <- retDbmRoc$varComp$varComp[4]
varYEps <- retDbmRoc$varComp$varComp[6]
effectSize <- retDbmRoc$ciDiffTrtRRRC$Estimate

#################################################################################
### ORH analysis
```

```
retORRoc  <- ORHAnalysis(rocData, fom = "Wilcoxon")
effectSize <- retORRoc$ciDiffTrtRRRC$Estimate
CovOR <- retORRoc$varComp
cov1 <- CovOR$varCov[3]
cov2 <- CovOR$varCov[4]
cov3 <- CovOR$varCov[5]
varEps <- CovOR$varCov[6]
msTR <- retORRoc$msTR
msT <- retORRoc$msT

################################################################################
### sample size
### sample size using DBMH variance components
for (J in 6:10) {
  ret <- SampleSizeGivenJ(J, varYTR, varYTC, varYEps, effectSize = effectSize)
  cat("# of rdrs = ", J, "estimated # of cases = ", ret$K, ", predicted power = ", ret$power, "\n")
}

### sample size using ORH variance components
KStar <- length(rocData$NL[1,1,,1])
for (J in 6:10) {
  ret <- SampleSizeGivenJ(J, cov1 = cov1, cov2 = cov2, cov3 = cov3, varEps = varEps, msTR = msTR,
KStar=KStar, effectSize = effectSize)
  cat("# of rdrs = ", J, "estimated # of cases = ", ret$K, ", predicted power = ", ret$power, "\n")
}

SampleSizeGivenJ(J = 6, cov1 = cov1, cov2 = cov2, cov3 = cov3, varEps = varEps, msTR = msTR,
KStar=KStar, effectSize = effectSize)

K <- 251
PowerGivenJK(6, K, varYTR, varYTC, varYEps, alpha = 0.05,
             effectSize = effectSize, randomOption = "ALL")

PowerTable(data = rocData, alpha = 0.05, effectSize = effectSize, desiredPower = 0.8, randomOption
= "ALL")

################################################################################
### output reports
rep1 <- OutputReport(data = rocData, dataDscrpt = "ROC Data", method = "ORH", fom = "Wilcoxon",
             covEstMethod = "Jackknife", showWarnings = FALSE)
rep2 <- OutputReport(dataset = rocData, method = "DBMH", fom = "Wilcoxon", dataDscrpt =
"MyROCData", showWarnings = FALSE)
rep3 <- OutputReport(dataset = rocData, method = "DBMH", fom = "Wilcoxon", reportFile =
"MyROCDataAnalysis.txt", showWarnings = FALSE)
## rep4 <- OutputReport(dataset = frocData, method = "ORH", fom = "SongA2", showWarnings = FALSE)
## very long computation
## rep5 <- OutputReport(dataset = frocData, method = "DBMH", fom = "Wilcoxon", showWarnings =
FALSE) ## error
rep6 <- OutputReport(dataset = frocData, method = "ORH", showWarnings = FALSE) # default fom is
wJAFROC
rep7 <- OutputReport(dataset = frocData, method = "DBMH", fom = "HrAuc", showWarnings = FALSE)
rep8 <- OutputReport(dataset = roiData, method = "ORH", fom = "ROI", showWarnings = FALSE)
rep9 <- OutputReport("rocData.xlsx", format = "JAFROC", method = "DBMH", fom = "Wilcoxon",
dataDscrpt = "MyROC2Data", showWarnings = FALSE)

################################################################################
### save file in alternate formats
SaveDataFile(dataset = rocData, fileName = "rocData2.xlsx", format = "JAFROC")
SaveDataFile(dataset = rocData, fileName = "rocData2.csv", format = "MRMC")
SaveDataFile(dataset = rocData, fileName = "rocData2.lrc", format = "MRMC", dataDscrpt =
"ExampleROCdata")
SaveDataFile(dataset = rocData, fileName = "rocData2.txt", format = "MRMC", dataDscrpt =
"ExampleROCdata2")
SaveDataFile(dataset = rocData, fileName = "rocData.imrmc", format = "iMRMC", dataDscrpt =
"ExampleROCdata3")

################################################################################
### plots
```

```
plotM <- c(1:2)
plotR <- c(1:5)
plotROC <- EmpiricalOpCharac(data = rocData, trts = plotM, rdrs = plotR, opChType = "ROC")

plotMAvg <- list(1, 2)
plotRAvg <- list(c(1:5),c(1:5))
plotRocAvg <- EmpiricalOpCharac(dataset = rocData, trts = plotMAvg, rdrs = plotRAvg, opChType =
"ROC")

##############################################################################
### ROI paradigm
retORRoi  <- ORHAnalysis(roiData, fom = "ROI")
OutputReport(dataset = roiData, method = "DBMH", fom = "ROI", showWarnings = "FALSE")
OutputReport(dataset = roiData, method = "ORH", fom = "ROI", showWarnings = "FALSE")
```

End Listing 1

Explanation of the code

First, notice that the existence of helpful line numbers, generated by `RStudio`, not shown in the above listing. The hash symbol (`#`) is used in `R` to insert comments: the compiler[1] ignores anything appearing on a line after the hash symbol. This is a convenient way to insert explanations of what we are doing and/or why we are doing it this way. For example, the comment on line 1 says (in effect) that we are deleting any and all existing variables so as to start with a "clean slate". This command is used sparingly (for obvious reasons you would never have it inside a function). You can use the `Help` system to find out exactly what the code `rm(list = ls())` does. Blank lines of code, which are ignored by the compiler, are inserted for improved readability. Multiple commands, separated by semi-colons (`;`) can be put on one line. `R` does not have a continuation symbol for long lines - Line 43-44 shows how to split up long lines across several lines without using any special continuation symbol (as long as the first line is not a *complete* `R` command, the next line is read, and so on, until it all makes sense to `R` or it "bombs out" with an error). Also, `R` commands are case sensitive: as one example, the command `Rm(list = ls())` will not work [try it! type `Rm(list = ls())` into the `Console` window (bottom-left panel in Fig. 9) and press the `return/enter` key]:

```
> Rm(list = ls())
Error: could not find function "Rm"
```

Line 2 is commented out; you will see later that it is used to clear out all files created by the program (obviously not the `*.R` files). A convenient way to run the program one line at a time is to position the cursor anywhere inside the text in line 1 and click on the `Run` button (upper right side of the source code window – do not click on the `Source` button right next to it!). This will cause line 1 to execute, and the cursor will automatically move to line 2. Click on `Run` again. Notice how the commands are being echoed to the `Console` window. The cursor should have moved to line 3. Click on `Run` again. This time there is some activity in the `Console` window, Fig. 10.

[1] A compiler is software the converts the more-or-less English-like instructions that make some sense to us to sequences of binary instructions that the computer understands, but which would be gibberish to us. When we execute a program, the binary instructions are executed, one after one, by the computer.

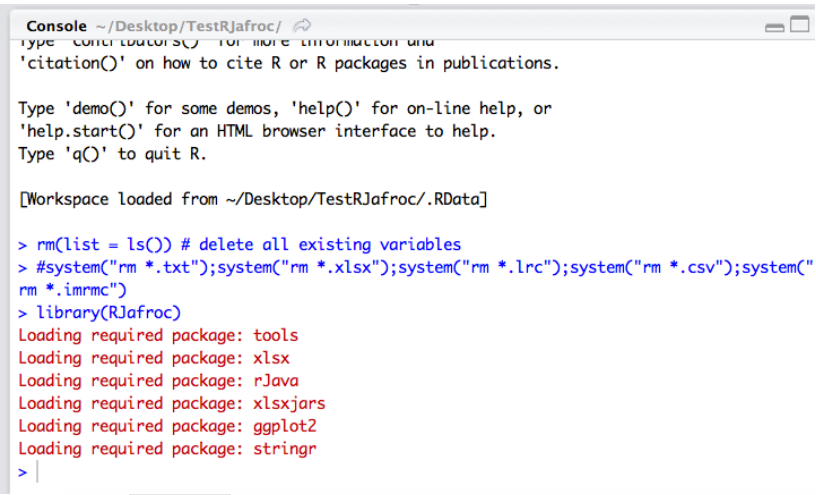


Fig. 10: The `RJafroc` package and its dependencies have been compiled into memory.

At this point the `RJafroc` package has been loaded from disk files and compiled, and all functions in it are ready for use. Click on the `Help` button in the lower right panel, and start typing in `RJafroc` in the text input area, upon which you should see Fig. 11.

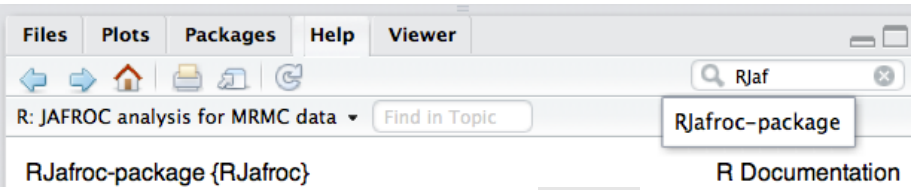


Fig. 11: Getting to the main documentation page for `RJafroc`.

Select `RJafroc-package`, the main documentation for this code. The lower right panel will look like Fig. 12.

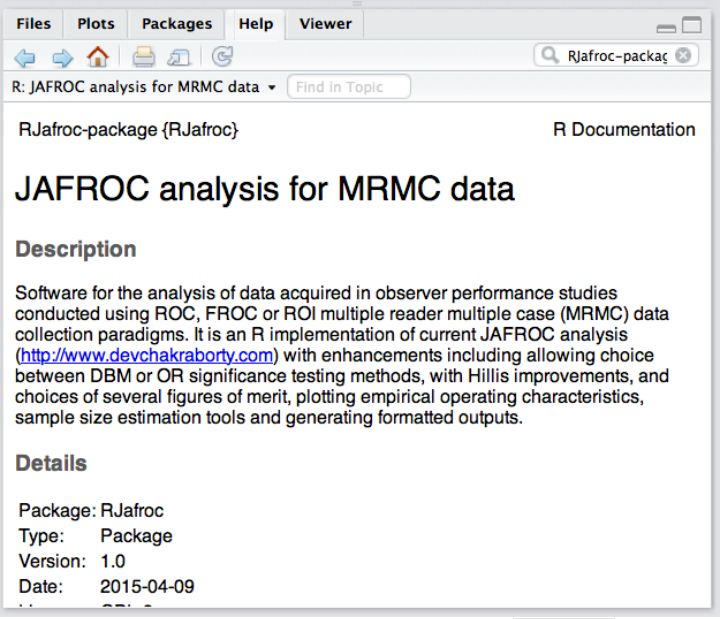


Fig. 12: The main on-line documentation page for `RJafroc`.

You can choose to read this scrollable document. If you scroll to the bottom you will see Fig. 13.

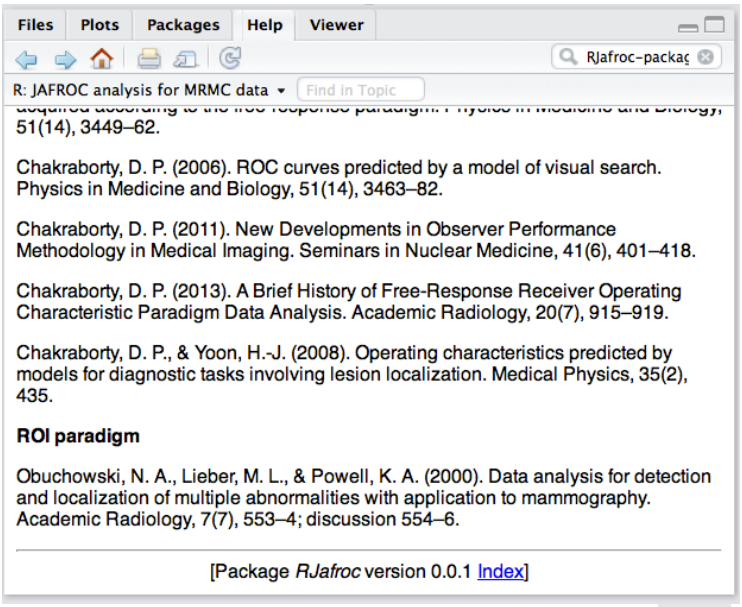


Fig. 13: Bottom of the window shown in Fig. 12; note the link to `index`.

Notice the link to `index`, clicking on which shows you Fig. 14 (I pulled a little trick to expand the lower right window to occupy more vertical real estate: I clicked on the rectangle symbol on the upper right corner of the window; clicking on it again will reduce the window to half height).

The RJafroc pdf documentation file

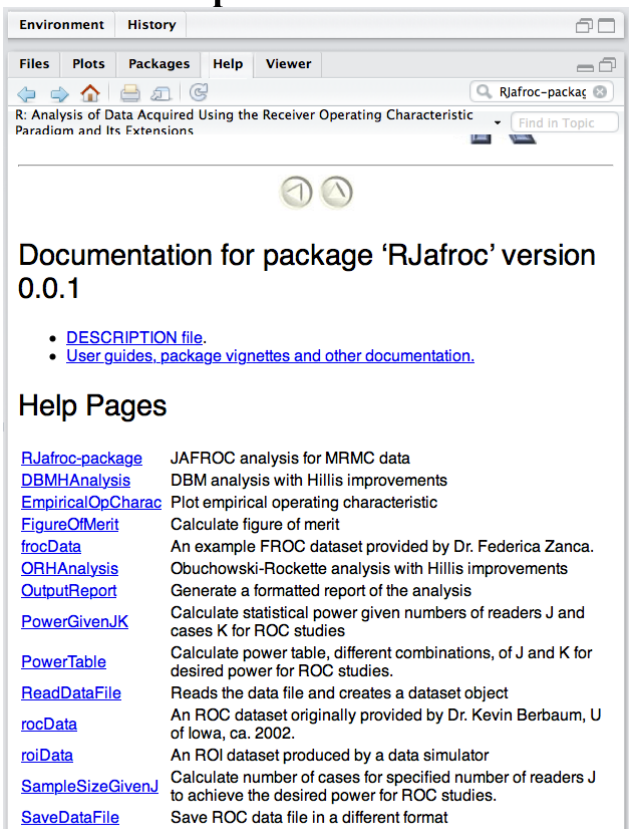


Fig. 14: Showing all the functions available in the package.

This shows you all the available functions in the package. To download the full documentation for this package, click on the link that says `User guides, package vignettes and other documentation`. This will bring up Fig. 15.

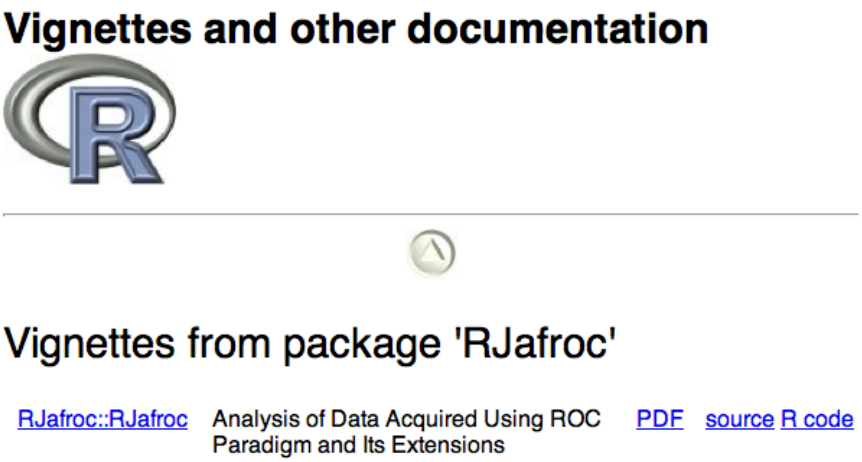


Fig. 15: Getting to the full documentation.

Click on `PDF`, upon which you should see Fig. 16.

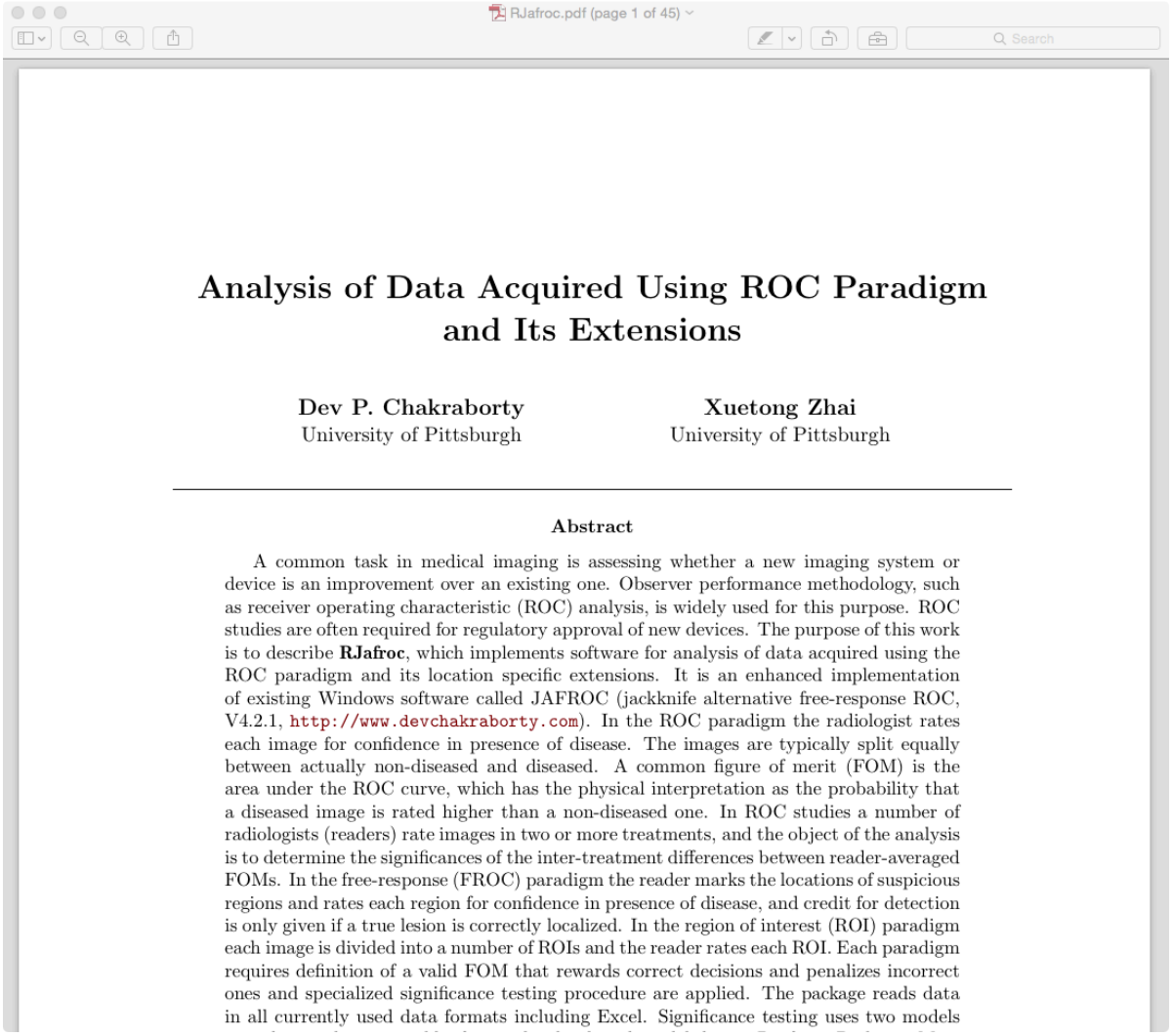


Fig. 16: Page 1 of 45 of the full documentation for the package.

You should save this as a `pdf` file for easy reading and making your own comments, Fig. 17.

Fig. 17: The main documentation saved as a `pdf` file.

If I open this file I see Fig. 18.

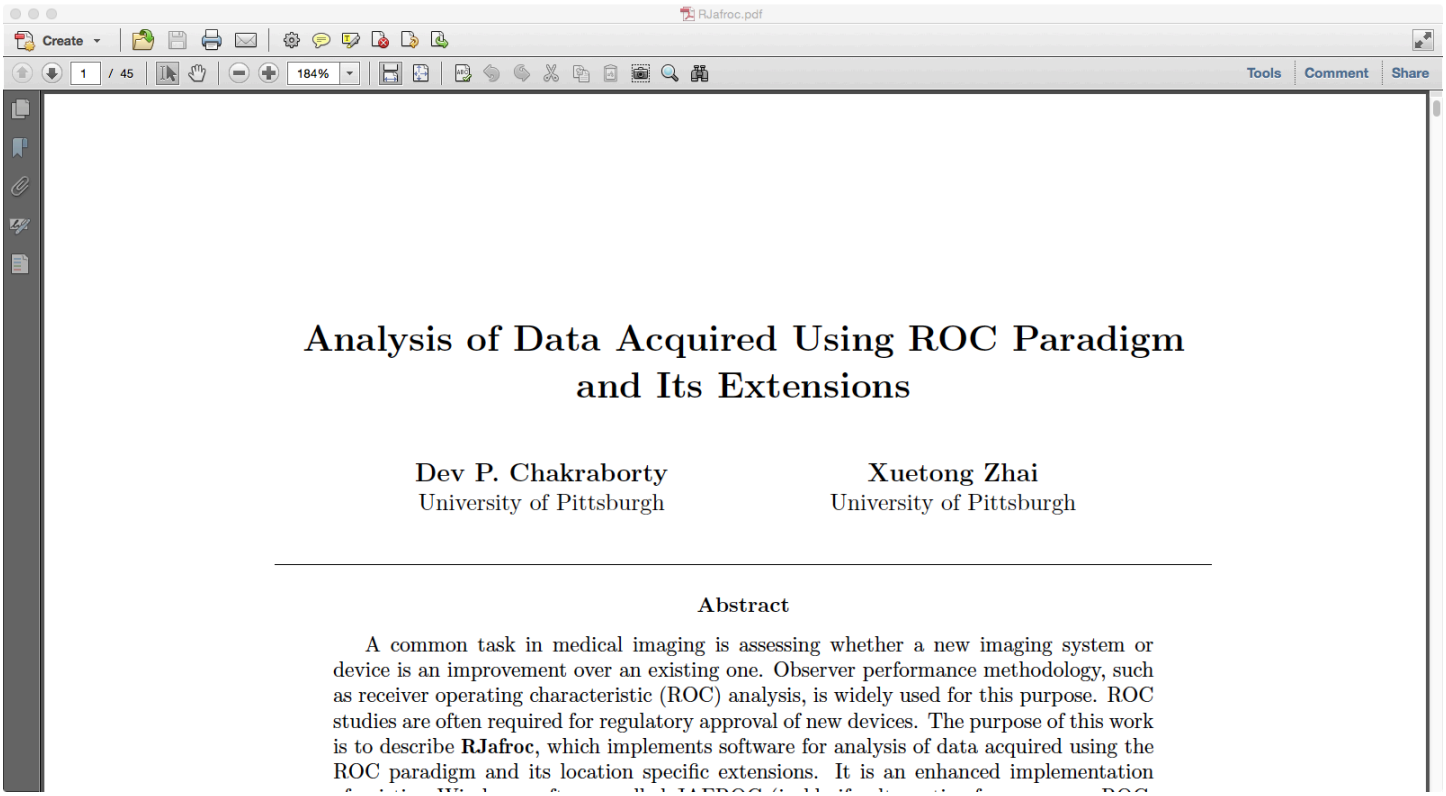


Fig. 18: Main documentation, opened in Acrobat.

DBMH analysis of a pre-loaded ROC dataset

To complete the example, we return to the source-code window, where we should be on line 5 (`RStudio` skipped blank line 4). Click on `Run` twice bring the cursor to line 7. This line calls the function `DBMHAnalysis` with first argument equal to `rocData`, which is a built in dataset, and second argument figure of merit `fom = "Wilcoxon"` and assigns the result to `retDbmRoc` (in case you had not guessed it, DBMH stands for Dorfman-Berbaum-Metz-Hillis).

```
retDbmRoc  <- DBMHAnalysis(rocData, fom = "Wilcoxon")
```

Click on `Run`, there is a brief delay and you see the `>` symbol in the `Console` window, meaning `RStudio` is ready for more commands. To print the results of the analysis highlight `retDbmRoc` in the top left window (line 7) and click on `Run`. There is a lot of output in the `Console` window, and if you scroll to the top you will see Fig. 19 (I pulled that vertical real estate trick again).

```
Source

Console ~/Desktop/TestRJafroc/
> retDbmRoc  <- DBMHAnalysis(rocData, fom = "Wilcoxon")
> retDbmRoc
$fomArray
          Rdr - 0   Rdr - 1   Rdr - 2   Rdr - 3   Rdr - 4
Trt - 0 0.9196457 0.8587762 0.9038647 0.9731079 0.8297907
Trt - 1 0.9478261 0.9053140 0.9217391 0.9993559 0.9299517

$anovaY
  Source          SS   DF         MS
1      T   0.5467634    1 0.54676344
2      R   1.7493072    4 0.43732680
3      C  44.8462969  113 0.39686988
4     TR   0.2512700    4 0.06281749
5     TC  11.2828335  113 0.09984808
6     RC  29.1544793  452 0.06450106
7    TRC  18.0671646  452 0.03997160
8  Total 105.8981150 1139         NA

$anovaYi
  Source  DF          0          1
1      R   4 0.35141967 0.14872462
2      C 113 0.33629430 0.16042367
3     RC 452 0.06043607 0.04403659

$varComp
               varComp
Var(R)     0.0015349993
Var(C)     0.0272492343
Var(T*R)   0.0002004025
Var(T*C)   0.0119752962
Var(R*C)   0.0122647286
Var(Error) 0.0399716032

$fRRRC
[1] 4.456319

$ddfRRRC
[1] 15.25967
```

Fig. 19: Partial results of DBMH analysis in scrollable window.

DBMH variance components

The next 4 lines extract the 3 pseudovalue variance components and observed effect size, which are required for sample size estimation, and assign them to `varYTR`, `varYTC`, `varYEps` and `effectSize`. You can run them one line at a time and the results will show up in the upper-right panel under `Environment`, Fig. 20.

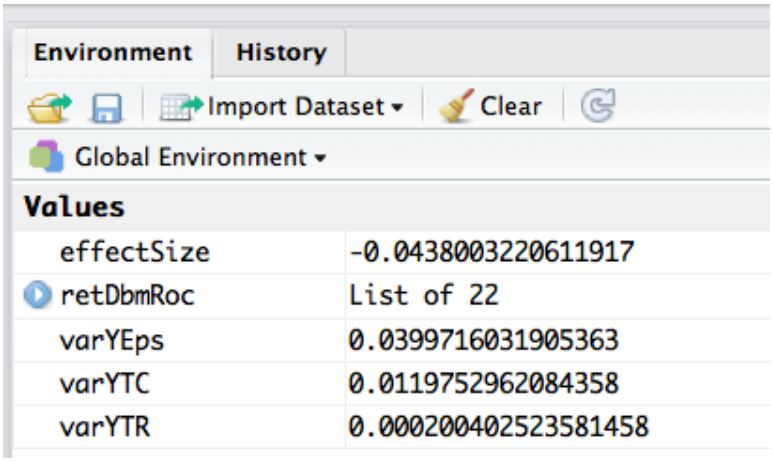


Fig. 20: Results of analysis so far.

Lines 15 – 23 analyze the same ROC dataset using the ORH method (in case you had not guessed, ORH stands for Obuchowski-Rockette-Hillis).

Notice the use of the `<-` symbol for assignment of the right hand side to the variable name appearing on the left hand side (scalar variables do not have to be declared before use; R is quite forgiving in this regard; it does not care about data types like integers, floats, doubles, etc. But arrays do need to be declared or "dimensioned" before hand). Do not use the `=` sign for assignment (the program may work "some" of the time, which is worse than never working at all). The `=` sign is used to *pass parameters to functions*, as in line 7: `fom = "Wilcoxon"`. Here R is quite restrictive. The function must have declared `fom` as a parameter and one of the allowed values is `"Wilcoxon"`, quotes and all, and it is case sensitive. To find out the allowed values click on `DBMHAnalysis` in the main documentation page of the package, Fig. 14, which should bring up Fig. 21.

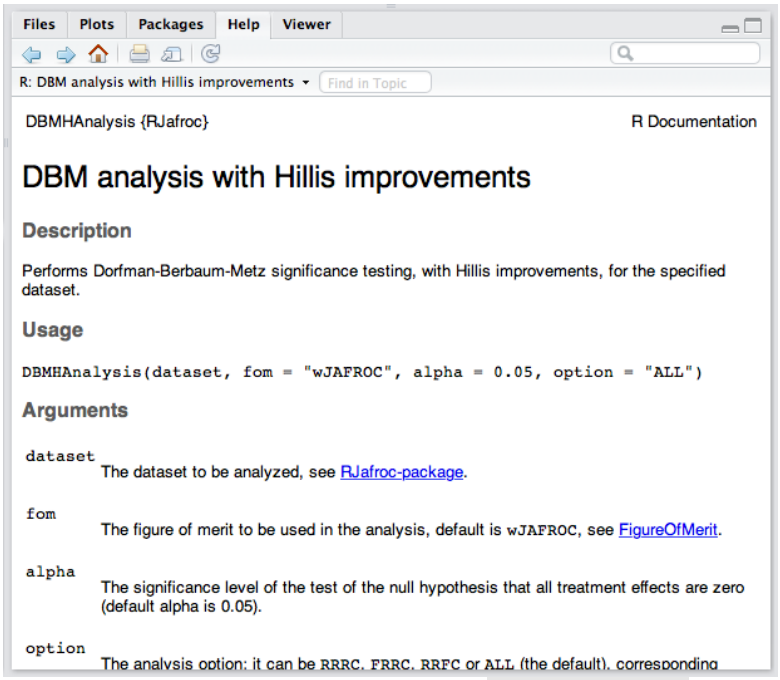
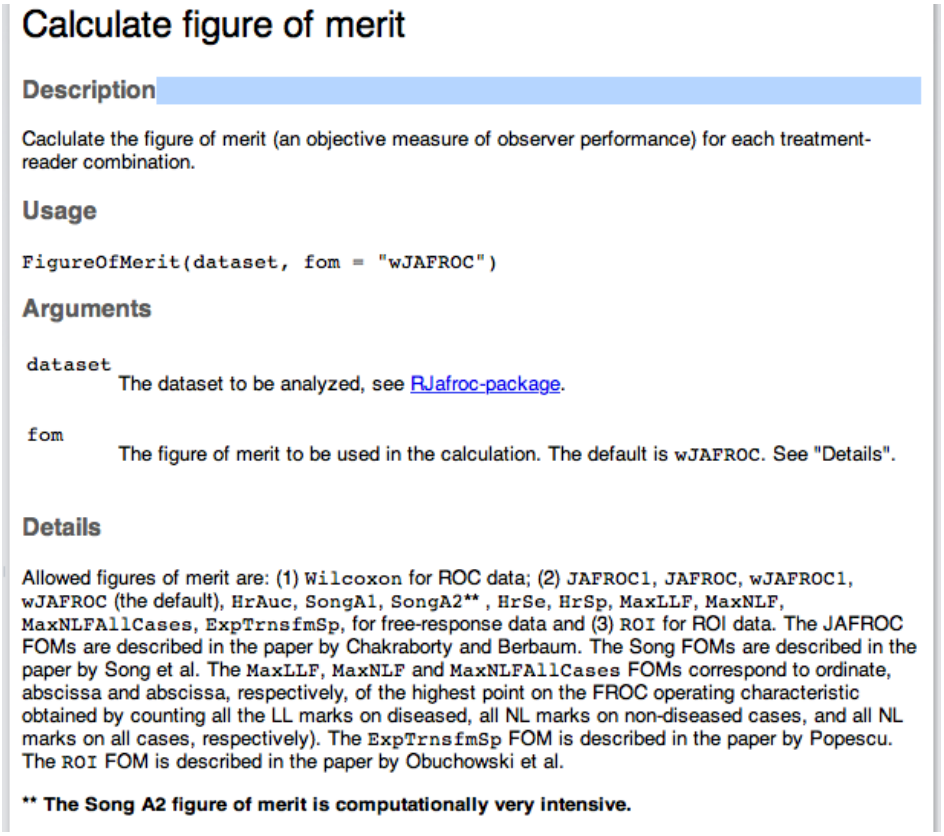


Fig. 21: Online documentation for the function `DBMHAnalysis()`.

Figure of merit

As you can see under `Usage`, the first argument of the function is a `dataset`, the parameter `fom` (for figure of merit) has the default value `"wJAFROC"`, for weighted JAFROC, which is why we had to explicitly specify `"Wilcoxon"` as the figure of merit. To verify that this is one of the allowed choices, click on the link `FigureOfMerit`, Fig. 22.



Calculate figure of merit

Description

Caclulate the figure of merit (an objective measure of observer performance) for each treatment-reader combination.

Usage

```
FigureOfMerit(dataset, fom = "wJAFROC")
```

Arguments

`dataset` The dataset to be analyzed, see RJafroc-package.

`fom` The figure of merit to be used in the calculation. The default is `wJAFROC`. See "Details".

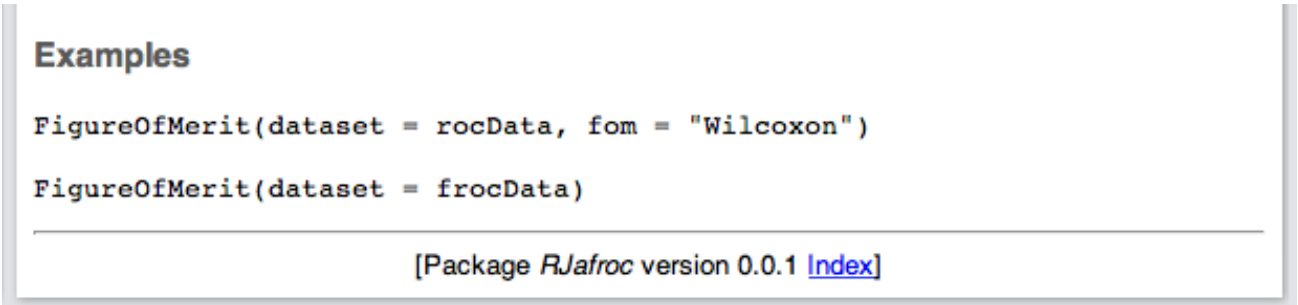
Details

Allowed figures of merit are: (1) `Wilcoxon` for ROC data; (2) `JAFROC1`, `JAFROC`, `wJAFROC1`, `wJAFROC` (the default), `HrAuc`, `SongA1`, `SongA2**` , `HrSe`, `HrSp`, `MaxLLF`, `MaxNLF`, `MaxNLFAllCases`, `ExpTrnsfmSp`, for free-response data and (3) `ROI` for ROI data. The JAFROC FOMs are described in the paper by Chakraborty and Berbaum. The Song FOMs are described in the paper by Song et al. The `MaxLLF`, `MaxNLF` and `MaxNLFAllCases` FOMs correspond to ordinate, abscissa and abscissa, respectively, of the highest point on the FROC operating characteristic obtained by counting all the LL marks on diseased, all NL marks on non-diseased cases, and all NL marks on all cases, respectively). The `ExpTrnsfmSp` FOM is described in the paper by Popescu. The `ROI` FOM is described in the paper by Obuchowski et al.

** The Song A2 figure of merit is computationally very intensive.

Fig. 22: Allowed figures of merit.

As you can seem for ROC data the only choice is `Wilcoxon` (with the quotes, we were sloppy creating this documentation). If you scroll to the bottom of this page, you should see Fig. 23.



Examples

```
FigureOfMerit(dataset = rocData, fom = "Wilcoxon")

FigureOfMerit(dataset = frocData)
```

[Package *RJafroc* version 0.0.1 Index]

Fig. 23: Showing examples of usage of `FigureOfMerit()` function.

You can literally copy and paste these examples into the `Console` window to see what happens. Try it:

```
> FigureOfMerit(dataset = rocData, fom = "Wilcoxon")
          Rdr - 0   Rdr - 1   Rdr - 2   Rdr - 3   Rdr - 4
Trt - 0 0.9196457 0.8587762 0.9038647 0.9731079 0.8297907
Trt - 1 0.9478261 0.9053140 0.9217391 0.9993559 0.9299517
```

The construct `dataset =` is not strictly needed, since `rocData` is the first argument. But in the following it is needed:

```
> FigureOfMerit(fom = "Wilcoxon", dataset = rocData)
          Rdr - 0   Rdr - 1   Rdr - 2   Rdr - 3   Rdr - 4
Trt - 0 0.9196457 0.8587762 0.9038647 0.9731079 0.8297907
Trt - 1 0.9478261 0.9053140 0.9217391 0.9993559 0.9299517
```

ORH analysis of the ROC dataset

To continue with the rest of the example, look at the top-left window, where the cursor should be on line 13. Another way of executing a bunch of lines is to highlight them and then clicking on `Run`. Highlight lines 13 - 23; see Fig. 24.

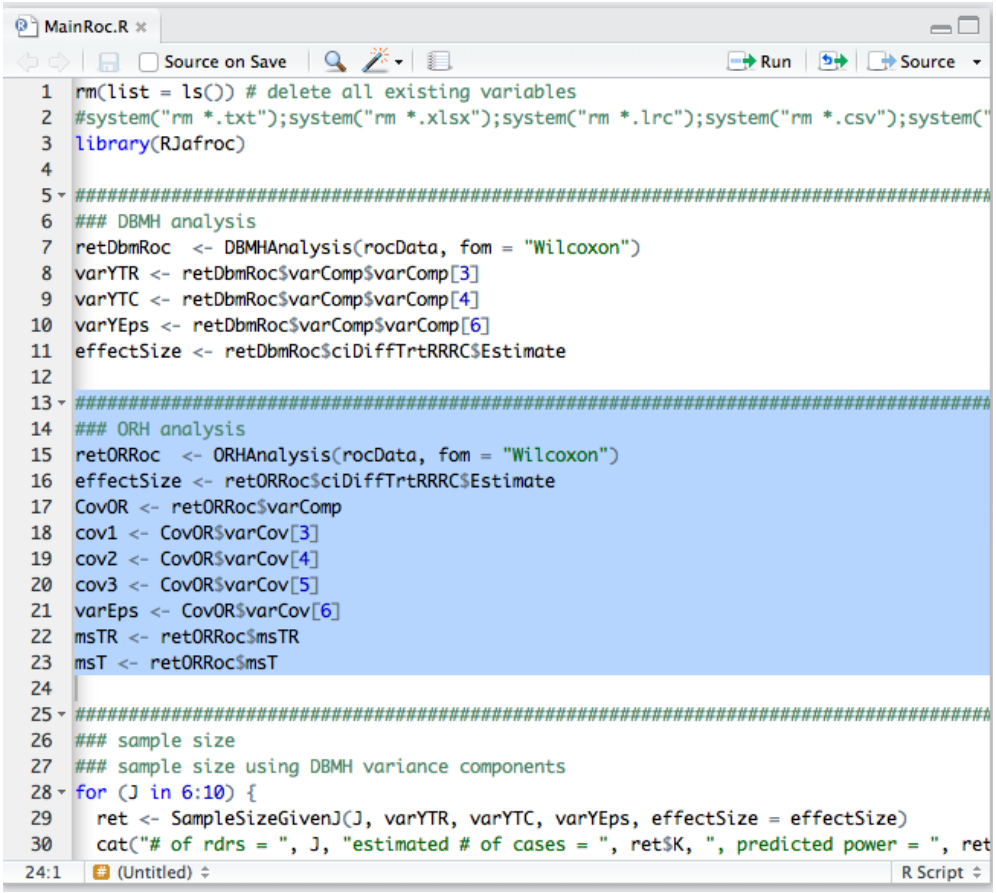


Fig. 24: Running a selection of lines.

Now click on `Run`. Notice how the lines are echoed to the `Console` window and how the `Environment` panel fills up with more values. This time we analyzed the data using the Obuchowski-Rockette-Hillis (ORH) method. Again, you can print the results of the analysis by highlighting `retORRoc` and clicking on `Run`. See Tables 4 and 5 in `RJafroc.pdf` for an explanation of the output. Briefly, the output can be understood if one makes use of the following abbreviations, often used in combination: `Y` = pseudovalue based, `Trt` = treatment, `Rdr` = reader, `RRRC` = random reader random case, `FRRC` = fixed reader random case, `RRFC` = random reader fixed case, `ci` = $1-\alpha$ confidence interval, `fomArray` = $\widehat{\theta}_{ij}$, `f` = value of observed F-statistic, `p` = p-value for rejecting the null hypothesis, `DiffTrt` = reader-averaged FOM differences between pairs of modalities, `ddf` = denominator degrees of freedom for F-test (the numerator degrees of freedom is always *I*-1), `AvgRdrEachTrt` = the FOM is averaged over all readers, separately for each treatment, `varComp` = the DBM pseudovalue variance components defined in connection with Eqn. 12. For the dataset shown, the

reader-averaged difference between the two modalities is not significant for `RRRC` (p = 0.0517), but is significant if either reader (p = 0.019) or case (p = 0.042) is regarded as a fixed factor.

Sample size functions

Now "block-run" lines 28 - 31, illustrating sample size estimation using `SampleSizeGivenJ()` (which means the number of cases to achieve default power 0.8 at default alpha 0.05 for `J` readers - sample size is generic for numbers of cases and readers, so if number of readers is specified it must return number of cases). This example uses the pseudovalue based DBM variance components. The result is:

```
> for (J in 6:10) {
+   ret <- SampleSizeGivenJ(J, varYTR, varYTC, varYEps, effectSize = effectSize)
+   cat("# of rdrs = ", J, "estimated # of cases = ", ret$K, ", predicted power = ", ret$power,
"\n")
+ }
# of rdrs =  6 estimated # of cases =  251 , predicted power =  0.8005403
# of rdrs =  7 estimated # of cases =  211 , predicted power =  0.8007873
# of rdrs =  8 estimated # of cases =  188 , predicted power =  0.800656
# of rdrs =  9 estimated # of cases =  173 , predicted power =  0.8005128
# of rdrs =  10 estimated # of cases =  163 , predicted power =  0.8015625
>
```

Now "block-run" lines 34 - 38, illustrating sample size estimation using the ORH variance components. The result is:

```
> KStar <- length(rocData$NL[1,1,,1])
> for (J in 6:10) {
+   ret <- SampleSizeGivenJ(J, cov1 = cov1, cov2 = cov2, cov3 = cov3, varEps = varEps, msTR = msTR,
KStar=KStar, effectSize = effectSize)
+   cat("# of rdrs = ", J, "estimated # of cases = ", ret$K, ", predicted power = ", ret$power,
"\n")
+ }
# of rdrs =  6 estimated # of cases =  251 , predicted power =  0.8005403
# of rdrs =  7 estimated # of cases =  211 , predicted power =  0.8007873
# of rdrs =  8 estimated # of cases =  188 , predicted power =  0.800656
# of rdrs =  9 estimated # of cases =  173 , predicted power =  0.8005128
# of rdrs =  10 estimated # of cases =  163 , predicted power =  0.8015625
```

`KStar` is the number of cases in the pilot dataset represented by the object `rocData`. To see the structure of the dataset, type `str(rocData)` at the command prompt (did you notice how `RStudio` automatically created the closing parenthesis?).

```
> str(rocData)
List of 8
 $ NL          : num [1:2, 1:5, 1:114, 1] 1 3 2 3 2 2 1 2 3 2 ...
 $ LL          : num [1:2, 1:5, 1:45, 1] 5 5 5 5 5 5 5 5 5 5 ...
 $ lesionNum   : int [1:45] 1 1 1 1 1 1 1 1 1 1 ...
 $ lesionID    : num [1:45, 1] 1 1 1 1 1 1 1 1 1 1 ...
 $ lesionWeight: num [1:45, 1] 1 1 1 1 1 1 1 1 1 1 ...
 $ dataType    : chr "ROC"
 $ modalityID  : chr [1:2] "0" "1"
 $ readerID    : chr [1:5] "0" "1" "2" "3" ..
```

This is actually the Van Dyke dataset used frequently by Hillis et al to illustrate advances in ROC analysis. The dataset is coded as a FROC dataset (ROC is a special case of FROC).

Block run lines 40 - 42; the result is:

```
> PowerGivenJK(6, K, varYTR, varYTC, varYEps, alpha = 0.05,
+              effectSize = effectSize, randomOption = "ALL")
```

```
[1] 0.8005403
```

`PowerGivenJK` means give me the statistical power for supplied values of `J` (# of readers, first argument) and `K` (# of cases, 2nd argument). Now run line 44:

```
> PowerTable(data = rocData, alpha = 0.05, effectSize = effectSize, desiredPower = 0.8,
randomOption = "ALL")
  numReaders numCases             power
1          2    >2000              <NA>
2          3    >2000              <NA>
3          4     1089 0.800037793175997
4          5      344 0.800514102941281
5          6      251 0.800540260369563
6          7      211  0.80078731736164
7          8      188 0.800655958321171
8          9      173 0.800512762837402
9         10      163 0.801562494495545
```

Creating report files

Block-run lines 48 - 57: this time there is lots of activity and several report files, created by the function `OutputReport()`, are visible in the `Files` window (now you can see the utility of line 2, which if run uncommented will erase all these files). These are text files that closely follow the format of `JAFROC` output that most of you are familiar with: Fig. 25.

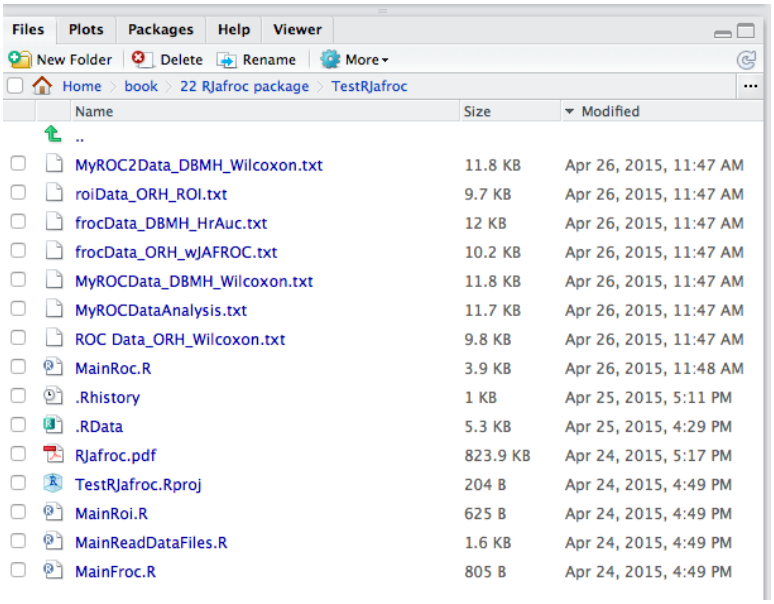


Fig. 25: Output files created by running lines 48 - 57 (sorted conveniently by field `Modified`).

You may have notice that I commented out line 52 and 53. Line 52 applies the Song A2 figure of merit to the pre-loaded FROC dataset. This takes very long to compute. Line 53 will generate an error because with FROC data one cannot use the Wilcoxon figure of merit. Line 61-65 saves ROC data in alternate data formats.

Creating plots

Lines 69-75 create empirical ROC plots and reader averaged ROC plots, respectively. Block-run these lines. To view them highlight `plotRoc` and/or `plotRocAvg` and click on `Run`, Fig. 26. Use the arrow buttons to switch between plots.

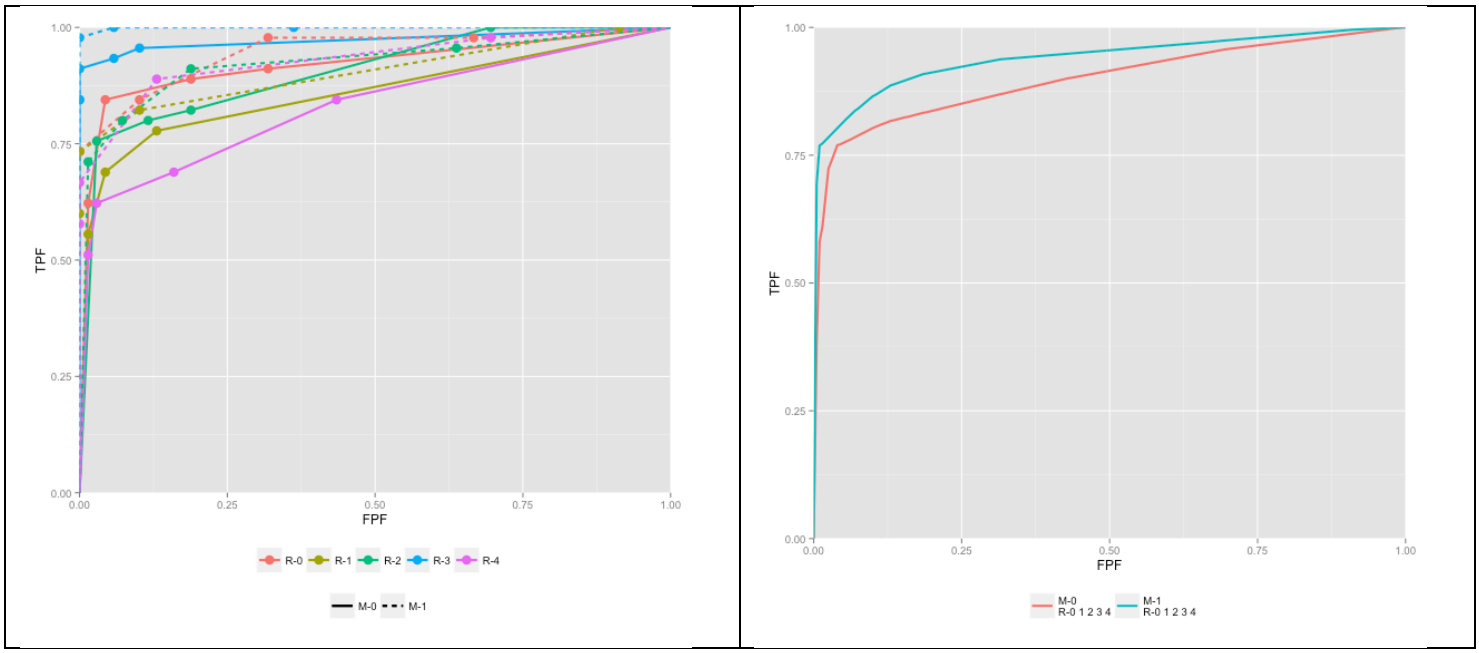


Fig. 26: Individual modality-reader empirical ROC plots, left panel, and reader empirical averaged ROC plots, right panel. The observed difference was not significant using random reader random case analysis. Use the arrows to switch between plots.

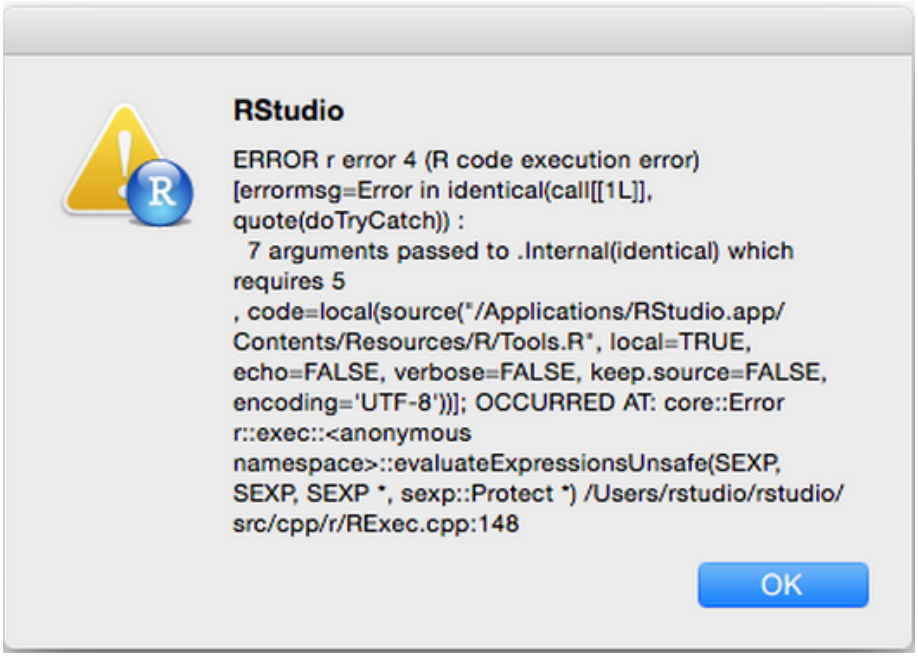
To run everything in this file all in one fell swoop, just click on `Source`.

I think I have written enough. With this introduction, the full `pdf` documentation file, and running the examples already in the folder `RJafroc`, you should be able to play around with different abilities of this package.

Let me have your feedback. Please make comments brief and to the point. For those new to this field, please *carefully* read the `pdf` documentation file.

Known issues

1. A user has told us about a problem installing `RStudio`; the `R` installation went fine, but on attempting to install `RStudio` the following error screen was seen:



The solution to this, found by a Google search by Xuetong Zhai is to locate and remove all previous versions of the `R` framework form `/Library/Frameworks/R.frameworks/`, see screen shot below:

jjallaire

Hi there,

This error typically occurs when you have one version of R that is launched which then connects to libraries from an older version of R. In this case the newer version of R is calling the identical function with 6 arguments and the (older) version of the base library receiving the call expects only 5. Once you make sure you've got only one version of R active you should have no problems.

J.J.

Sean

Locating and removing all previous versions of the R framework from /Library/Frameworks/R.framework/ indeed fixed the problem completely.

Thank you very much!

Dev P. Chakraborty, Ph.D.
Xuetong Zhai, M.S.
Last updated: 5/3/15 11:55 AM