RJafroc Documentation

Dev P. Chakraborty, PhD

2020-03-10

Contents

Chapter 1

Preface

- This book, an extended documentation of the **RJafroc** package, is undergoing extensive edits.
- It should not be used by the casual user until I give the go ahead.
- It bypasses the file size limits of **CRAN**, currently 5 MB, which severely limits the extent of the documentation that can be included with the CRAN version of the package.
- I welcome corrections and comments by the not-so-casual-user.
- Please use the GitHub website to raise issues and comments:
  - https://github.com/dpc10ster/RJafrocBook

Chapter 2

Introduction

- This is the book desribing the **RJafroc** packages.
- The name of the book is RJafrocBook
- Modality and treatment are used interchangeably.
- Reader is a generic radiologist, or a computer aided detection algorithm, or any algorithmic "reader"
- TBA

Chapter 3

ROC data format

3.1 Introduction

- The purpose of this vignette is to explain the data format of the input Excel file and to introduce the capabilities of the function `DfReadDataFile()`. Background on observer performance methods are in my book (Chakraborty, 2017).
- I will start with Receiver Operating Characteristic (ROC) data as this is by far the simplest paradigm.
- In the ROC paradigm the observer assigns a rating to each image. A rating is an ordered numeric label, and, in our convention, higher values represent greater certainty or **confidence level** for presence of disease. With human observers, a 5 (or 6) point rating scale is typically used, with 1 representing highest confidence for *absence* of disease and 5 (or 6) representing highest confidence for *presence* of disease. Intermediate values represent intermediate confidence levels for presence or absence of disease.
- Note that location information associated with the disease, if applicable, is not collected.
- There is no restriction to 5 or 6 ratings. With algorithmic observers, e.g., computer aided detection (CAD) algorithms, the rating could be a floating point number and have infinite precision. All that is required is that higher values correspond to greater confidence in presence of disease.

3.2 Note to existing users

- The Excel file format has recently undergone changes resulting in 4 extra `list` members in the final created `dataset` object (i.e., 12 members

instead of 8).
- Code should run on the old format Excel files as the 4 extra list members are simply ignored.
- Reasons for the change will become clearer in these vignettes
- Basically they are needed for generalization to other data collection paradigms instead of crossed, for example to the split-plot data acquisition paradigm, and for better data entry error control.

3.3 The Excel data format

- The Excel file has three worksheets.
- These are named
  - `Truth`,
  - `NL` (or `FP`),
  - `LL` (or `TP`).

3.4 Illustrative toy file

- *Toy files* are artificial small datasets intended to illustrate essential features of the data format.

- The examples shown in this vignette corresponds to Excel file `inst/extdata/toyFiles/ROC/rocCr.xlsx` in the project directory.
- To view these files one needs to `clone` the source files from `GitHub`.

3.5 The `Truth` worksheet

- The `Truth` worksheet contains 6 columns: `CaseID`, `LesionID`, `Weight`, `ReaderID`, `ModalityID` and `Paradigm`.
- For ROC data the first five columns contain as many rows as there are cases (images) in the dataset.
- `CaseID`: unique integers, one per case, representing the cases in the dataset.
- `LesionID`: integers 0 or 1, with each 0 representing a non-diseased case and each 1 representing a diseased case.
- In the current toy dataset, the non-diseased cases are labeled `1`, `2` and `3`, while the diseased cases are labeled `70`, `71`, `72`, `73` and `74`. The values do not have to be consecutive integers; they need not be ordered; the only requirement is that they be **unique**.
- `Weight`: Not used for ROC data, a floating point value, typically filled in with 0 or 1.

- `ReaderID`: a **comma-separated** listing of reader labels, each represented by a **unique string**, that have interpreted the case. In the example shown below each cell has the value `0, 1, 2, 3, 4` meaning that each of the readers, represented by the strings "0", "1", "2", "3" and "4", have interpreted all cases (hence the "crossed" design). **With reader names that could be confused with integers, each cell in this column has to be text formatted as otherwise Excel will not accept it.** [Try entering `0, 1, 2, 3, 4` in a numeric formatted Excel cell.]
- The reader names could just as well have been `Rdr0, Rdr1, Rdr2, Rdr3, Rdr4`. The only requirement is that they be unique strings.
- Look in in the `inst/extdata/toyFiles/ROC` directory for files `rocCrStrRdrsTrts.xlsx` and `rocCrStrRdrsNonUnique.xlsx` for examples of data files using longer strings for readers. The second file generates an error because the reader names are not unique.
- `ModalityID`: a comma-separated listing of modalities (one or more modalities), each represented by a **unique string**, that are applied to each case. In the example each cell has the value `"0", "1"`. **With treatment names that could be confused with integers, each cell has to be text formatted as otherwise Excel will not accept it.**
- The treatment names could just as well have been `Trt0, Trt1`. Again, the only requirement is that they be unique strings.
- `Paradigm`: this column contains two cells, `ROC` and `crossed`. It informs the software that this is an ROC dataset, and the design is crossed, meaning each reader has interpreted each case in each modality (in statistical terminology: modality and reader factors are "crossed").
- There are 5 diseased cases in the dataset (the number of 1's in the `LesionID` column of the `Truth` worksheet).
- There are 3 non-diseased cases in the dataset (the number of 0's in the `LesionID` column).
- There are 5 readers in the dataset (each cell in the `ReaderID` column contains the string `0, 1, 2, 3, 4`).
- There are 2 modalities in the dataset (each cell in the `ModalityID` column contains the string `0, 1`).

3.6 The structure of an ROC dataset

In the following code chunk the first statement retrieves the name of the data file, located in a hidden directory that one need not be concerned with. The second statement reads the file using the function `DfReadDataFile()` and saves it to object `x`. The third statement shows the structure of the dataset object `x`.

```
rocCr <- system.file("extdata", "toyFiles/ROC/rocCr.xlsx",
                         package = "RJafroc", mustWork = TRUE)
```

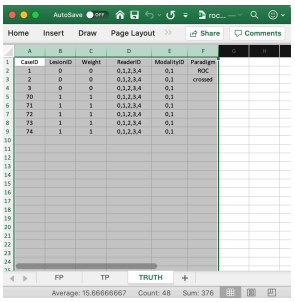

Figure 3.1: Truth worksheet for file rocCr.xlsx

```
x <- DfReadDataFile(rocCr, newExcelFileFormat = TRUE)
str(x)
#> List of 12
#>  $ NL           : num [1:2, 1:5, 1:8, 1] 1 3 2 3 2 2 1 2 3 2 ...
#>  $ LL           : num [1:2, 1:5, 1:5, 1] 5 5 5 5 5 5 5 5 5 5 ...
#>  $ lesionVector : int [1:5] 1 1 1 1 1
#>  $ lesionID     : num [1:5, 1] 1 1 1 1 1
#>  $ lesionWeight : num [1:5, 1] 1 1 1 1 1
#>  $ dataType     : chr "ROC"
#>  $ modalityID   : Named chr [1:2] "0" "1"
#>   ..- attr(*, "names")= chr [1:2] "0" "1"
#>  $ readerID     : Named chr [1:5] "0" "1" "2" "3" ...
#>   ..- attr(*, "names")= chr [1:5] "0" "1" "2" "3" ...
#>  $ design       : chr "CROSSED"
#>  $ normalCases  : int [1:3] 1 2 3
#>  $ abnormalCases: int [1:5] 70 71 72 73 74
#>  $ truthTableStr: num [1:2, 1:5, 1:8, 1:2] 1 1 1 1 1 1 1 1 1 1 ...
```

- In the above code chunk flag `newExcelFileFormat` is set to `TRUE` as otherwise columns D - F in the `Truth` worksheet are ignored and the dataset is assumed to be crossed, with `dataType` automatically determined from the contents of the FP and TP worksheets.
- Flag `newExcelFileFormat = FALSE` is for compatibility with older JAFROC format Excel files, which did not have these columns in the `Truth` worksheet. Its usage is deprecated.
- The dataset object `x` is a `list` variable with 12 members.
- The `x$NL` member, with dimension [2, 5, 8, 1], contains the ratings of normal cases. The extra values in the third dimension, filled with `NAs`, are needed for compatibility with FROC datasets, as unlike ROC, false positives are possible on diseased cases.
- The `x$LL`, with dimension [2, 5, 5, 1], contains the ratings of abnormal cases.

- The `x$lesionVector` member is a vector with 5 ones representing the 5 diseased cases in the dataset.
- The `x$lesionID` member is an array with 5 ones.
- The `x$lesionWeight` member is an array with 5 ones.
- The `lesionVector`, `lesionID` and `lesionWeight` members are not used for ROC datasets. They are there for compatibility with FROC datasets.
- The `dataType` member indicates that this is an `ROC` dataset.
- The `x$modalityID` member is a vector with two elements `"0"` and `"1"`, naming the two modalities.
- The `x$readerID` member is a vector with five elements `"0"`, `"1"`, `"2"`, `"3"` and `"4"`, naming the five readers.
- The `x$design` member is CROSSED; specifies the dataset design, which is "CROSSED".
- The `x$normalCases` member lists the integer names of the normal cases, 1, 2, 3.
- The `x$abnormalCases` member lists the integer names of the abnormal cases, 70, 71, 72, 73, 74.
- The `x$truthTableStr` member quantifies the structure of the dataset, as explained in **Chapter 00 Vignette #3-#5**.

3.7 The false positive (FP) ratings

These are found in the `FP` or `NL` worksheet, see below.

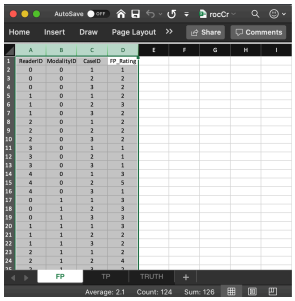

Figure 3.2: FP worksheet for file rocCr.xlsx

- It consists of 4 columns, each of length 30 (= # of modalities times number of readers times number of non-diseased cases).
- `ReaderID`: the reader labels: `0`, `1`, `2`, `3` and `4`. Each reader label occurs 6 times (= # of modalities times number of non-diseased cases).
- `ModalityID`: the modality or treatment labels: `0` and `1`. Each label occurs 15 times (= # of readers times number of non-diseased cases).
- `CaseID`: the case labels for non-diseased cases: `1`, `2` and `3`. Each label occurs 10 times (= # of modalities times # of readers).

- The label of a diseased case cannot occur in the FP worksheet. If it does the software generates an error.
- `FP_Rating`: the floating point ratings of non-diseased cases. Each row of this worksheet contains a rating corresponding to the values of `ReaderID`, `ModalityID` and `CaseID` for that row.

3.8 The true positive (TP) ratings

These are found in the `TP` or `LL` worksheet, see below.

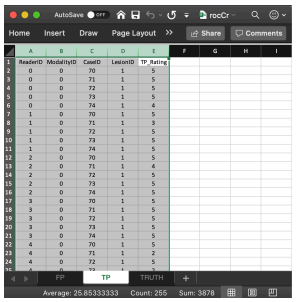

Figure 3.3: TP worksheet for file rocCr.xlsx

- It consists of 5 columns, each of length 50 (= # of modalities times number of readers times number of diseased cases).
- `ReaderID`: the reader labels: `0`, `1`, `2`, `3` and `4`. Each reader label occurs 10 times (= # of modalities times number of diseased cases).
- `ModalityID`: the modality or treatment labels: `0` and `1`. Each label occurs 25 times (= # of readers times number of diseased cases).
- `LesionID`: For an ROC dataset this column contains fifty 1's (each diseased case has one lesion).
- `CaseID`: the case labels for non-diseased cases: `70`, `71`, `72`, `73` and `74`. Each label occurs 10 times (= # of modalities times # of readers). The label of a non-diseased case cannot occur in the TP worksheet.
- `TP_Rating`: the floating point ratings of diseased cases. Each row of this worksheet contains a rating corresponding to the values of `ReaderID`, `ModalityID`, `LesionID` and `CaseID` for that row.

3.9 Correspondence between `NL` member of dataset and the `FP` worksheet

- The list member `x$NL` is an array with `dim = c(2,5,8,1)`.

  - The first dimension (2) comes from the number of modalities.
  - The second dimension (5) comes from the number of readers.
  - The third dimension (8) comes from the **total** number of cases.
  - The fourth dimension is alway 1 for an ROC dataset.

- The value of `x$NL[1,5,2,1]`, i.e., 5, corresponds to row 15 of the FP table, i.e., to `ModalityID` = 0, `ReaderID` = 4 and `CaseID` = 2.
- The value of `x$NL[2,3,2,1]`, i.e., 4, corresponds to row 24 of the FP table, i.e., to `ModalityID` 1, `ReaderID` 2 and `CaseID` 2.
- All values for case index > 3 are `-Inf`. For example the value of `x$NL[2,3,4,1]` is `-Inf`. This is because there are only 3 non-diseased cases. The extra length is needed for compatibility with FROC datasets.

3.10 Correspondence between LL member of dataset and the TP worksheet

- The list member `x$LL` is an array with `dim = c(2,5,5,1)`.
  - The first dimension (2) comes from the number of modalities.
  - The second dimension (5) comes from the number of readers.
  - The third dimension (5) comes from the number of diseased cases.
  - The fourth dimension is alway 1 for an ROC dataset.
- The value of `x$LL[1,1,5,1]`, i.e., 4, corresponds to row 6 of the TP table, i.e., to `ModalityID` = 0, `ReaderID` = 0 and `CaseID` = 74.
- The value of `x$LL[1,2,2,1]`, i.e., 3, corresponds to row 8 of the TP table, i.e., to `ModalityID` = 0, `ReaderID` = 1 and `CaseID` = 71.
- There are no -Inf values in `x$LL`: `any(x$LL == -Inf)` = FALSE.

3.11 Correspondence using the which function

- Converting from **names** to **subscripts** (indicating position in an array) can be confusing.
- The following example uses the `which` function to help out.
- The first line says that the `abnormalCase` named 70 corresponds to subscript 1 in the LL array case dimension.
- The second line prints the NL rating for `modalityID` = 0, `readerID` = 1 and `normalCases` = 1.
- The third line prints the LL rating for `modalityID` = 0, `readerID` = 1 and `abnormalCases` = 70.
- The last line shows what happens if one enters an invalid value for name; the result is a `numeric(0)`.
- Note that in each of these examples, the last dimension is 1 because we are dealing with an ROC dataset.

- The reader is encouraged to examine the correspondence between the NL and LL ratings and the Excel file using this method.

```
which(x$abnormalCases == 70)
#> [1] 1
x$NL[which(x$modalityID == "0"),which(x$readerID == "1"),which(x$normalCases == 1),1]
#> [1] 2
x$LL[which(x$modalityID == "0"),which(x$readerID == "1"),which(x$abnormalCases == 70),
#> [1] 5
x$LL[which(x$modalityID == "a"),which(x$readerID == "1"),which(x$abnormalCases == 70),
#> numeric(0)
```

3.12 References

Chapter 4

FROC data format

4.1 Purpose

- Explain the data format of the input Excel file for FROC datasets.
- Explain the format of the FROC dataset.
- Explain the lesion distribution array returned by `UtilLesionDistr()`.
- Explain the lesion weights array returned by `UtilLesionWeightsDistr()`.
- Details on the FROC paradigm are in my book.

4.2 Introduction

- In the Free-response Receiver Operating Characteristic (FROC) paradigm the observer searches each case for signs of **localized disease** and marks and rates localized regions that are sufficiently suspicious for disease presence.
- FROC data consists of **mark-rating pairs**, where each mark is a localized-region that was considered sufficiently suspicious for presence of a localized lesion and the rating is the corresponding confidence level.
- By adopting a proximity criterion, each mark is classified by the investigator as a lesion localization (`LL`) - if it is close to a real lesion - or a non-lesion localization (`NL`) otherwise.
- The observer assigns a rating to each region. The rating, as in the ROC paradigm, can be an integer or quasi-continuous (e.g., 0 – 100), or a floating point value, as long as higher numbers represent greater confidence in presence of a lesion at the indicated region.

4.3 The Excel data format

The Excel file has three worsheets. These are named `Truth`, `NL` or `FP` and `LL` or `TP`.

4.4 The `Truth` worksheet

The `Truth` worksheet contains 6 columns: `CaseID`, `LesionID`, `Weight`, `ReaderID`, `ModalityID` and `Paradigm`.

- Since a diseased case may have more than one lesion, the first five columns contain **at least** as many rows as there are cases (images) in the dataset.
- `CaseID`: unique **integers**, one per case, representing the cases in the dataset.
- `LesionID`: integers 0, 1, 2, etc., with each 0 representing a non-diseased case, 1 representing the *first* lesion on a diseased case, 2 representing the second lesion on a diseased case, if present, and so on.
- The non-diseased cases are labeled `1`, `2` and `3`, while the diseased cases are labeled `70`, `71`, `72`, `73` and `74`.
- There are 3 non-diseased cases in the dataset (the number of 0's in the `LesionID` column).
- There are 5 diseased cases in the dataset (the number of 1's in the `LesionID` column of the `Truth` worksheet).
- There are 3 readers in the dataset (each cell in the `ReaderID` column contains `0, 1, 2`).
- There are 2 modalities in the dataset (each cell in the `ModalityID` column contains `0, 1`).
- `Weight`: floating point; 0, for each non-diseased case, or values for each diseased case that add up to unity.

- Diseased case `70` has two lesions, with `LesionID`s 1 and 2, and weights 0.3 and 0.7. Diseased case `71` has one lesion, with `LesionID` = 1, and `Weight` = 1. Diseased case `72` has three lesions, with `LesionID`s 1, 2 and 3 and weights 1/3 each. Diseased case `73` has two lesions, with `LesionID`s 1, and 2 and weights 0.1 and 0.9. Diseased case `74` has one lesion, with `LesionID` = 1 and `Weight` = 1.
- `ReaderID`: a comma-separated listing of readers, each represented by a unique **integer**, that have interpreted the case. In the example shown below each cell has the value `0, 1, 2`. **Each cell has to be text formatted. Otherwise Excel will not accept it.**
- `ModalityID`: a comma-separated listing of modalities (or treatments), each represented by a unique **integer**, that apply to each case. In the example each cell has the value `0, 1`. **Each cell has to be text formatted.**

- `Paradigm`: In the example shown below, the contents are `FROC` and `crossed`. It informs the software that this is an `FROC` dataset and the design is "crossed", as in **Vignette #1**.

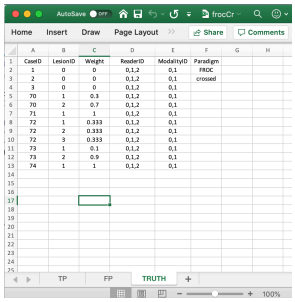

Figure 4.1: Fig. 1: Truth worksheet for file inst/extdata/toyFiles/FROC/frocCr.xlsx

4.5 The structure of an FROC dataset

The example shown above corresponds to Excel file `inst/extdata/toyFiles/FROC/frocCr.xlsx` in the project directory.

```
frocCr <- system.file("extdata", "toyFiles/FROC/frocCr.xlsx",
                      package = "RJafroc", mustWork = TRUE)
x <- DfReadDataFile(frocCr, newExcelFileFormat = TRUE)
str(x)
#> List of 12
#>  $ NL           : num [1:2, 1:3, 1:8, 1:2] 1.02 2.89 2.21 3.01 2.14 ...
#>  $ LL           : num [1:2, 1:3, 1:5, 1:3] 5.28 5.2 5.14 4.77 4.66 4.87 3.01 3.27 3.31 3.19 ..
#>  $ lesionVector : int [1:5] 2 1 3 2 1
#>  $ lesionID     : num [1:5, 1:3] 1 1 1 1 1 ...
#>  $ lesionWeight : num [1:5, 1:3] 0.3 1 0.333 0.1 1 ...
#>  $ dataType     : chr "FROC"
#>  $ modalityID   : Named chr [1:2] "0" "1"
#>   ..- attr(*, "names")= chr [1:2] "0" "1"
#>  $ readerID     : Named chr [1:3] "0" "1" "2"
#>   ..- attr(*, "names")= chr [1:3] "0" "1" "2"
#>  $ design       : chr "CROSSED"
#>  $ normalCases  : int [1:3] 1 2 3
#>  $ abnormalCases: int [1:5] 70 71 72 73 74
#>  $ truthTableStr: num [1:2, 1:3, 1:8, 1:4] 1 1 1 1 1 1 1 1 1 1 ...
```

- This follows the general description in **Vignette #1**. The differences are described below.
- The `x$dataType` member indicates that this is an `FROC` dataset.
- The `x$lesionVector` member is a vector whose contents reflect the number of lesions in each diseased case, i.e., 2, 1, 3, 2, 1 in the current example.
- The `x$lesionID` member indicates the labeling of the lesions in each diseased case.

```
x$lesionID
#>      [,1] [,2] [,3]
#> [1,]    1    2 -Inf
#> [2,]    1 -Inf -Inf
#> [3,]    1    2    3
#> [4,]    1    2 -Inf
#> [5,]    1 -Inf -Inf
```

- This shows that the lesions on the first diseased case are labeled 1 and 2. The `-Inf` is a filler used to denote a missing value. The second diseased case has one lesion labeled 1. The third diseased case has three lesions labeled 1, 2 and 3, etc.
- The `lesionWeight` member is the clinical importance of each lesion. Lacking specific clinical reasons, the lesions should be equally weighted; this is *not* true for this toy dataset.

```
x$lesionWeight
#>           [,1]      [,2]      [,3]
#> [1,] 0.3000000 0.7000000      -Inf
#> [2,] 1.0000000      -Inf      -Inf
#> [3,] 0.3333333 0.3333333 0.3333333
#> [4,] 0.1000000 0.9000000      -Inf
#> [5,] 1.0000000      -Inf      -Inf
```

- The first diseased case has two lesions, the first has weight 0.3 and the second has weight 0.7. The second diseased case has one lesion with weight 1.The third diseased case has three equally weighted lesions, each with weight 1/3. Etc.

4.6 The false positive (FP) ratings

These are found in the `FP` or `NL` worksheet, see below.

- It consists of 4 columns, of equal length. **The common length is unpredictable.** It could be zero if the dataset has no NL marks (a distinct

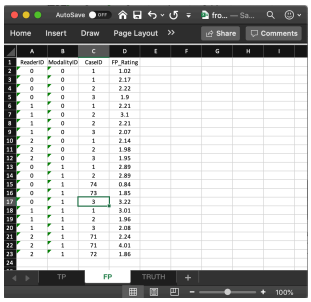

Figure 4.2: Fig. 2: FP/NL worksheet for file inst/extdata/toyFiles/FROC/frocCr.xlsx

possibility if the lesions are very easy to find and the modality and/or observer has high performance). All one knows is that the common length is an integer greater than or equal to zero.

- In the example dataset, the common length is 22.
- `ReaderID`: the reader labels: these must be `0`, `1`, or `2`, as declared in the `Truth` worksheet.
- `ModalityID`: the modality labels: must be `0` or `1`, as declared in the `Truth` worksheet.
- `CaseID`: the labels of cases with `NL` marks. In the FROC paradigm, `NL` events can occur on non-diseased **and** diseased cases.
- `FP_Rating`: the floating point ratings of `NL` marks. Each row of this worksheet yields a rating corresponding to the values of `ReaderID`, `ModalityID` and `CaseID` for that row.
- For `ModalityID` 0, `ReaderID` 0 and `CaseID` 1 (the first non-diseased case declared in the `Truth` worksheet), there is a single `NL` mark that was rated 1.02, corresponding to row 2 of the `FP` worksheet.
- Diseased cases with `NL` marks are also declared in the `FP` worksheet. Some examples are seen at rows 15, 16 and 21-23 of the `FP` worksheet.
- Rows 21 and 22 show that `caseID` = 71 got two `NL` marks, rated 2.24, 4.01.
- That this is the *only* case with two marks determines the length of the fourth dimension of the `x$NL` list member, 2 in the current example. Absent this case, the length would have been one.
- In general, the case with the most `NL` marks determines the length of the fourth dimension of the `x$NL` list member.
- The reader should convince oneself that the ratings in `x$NL` reflect the contents of the `FP` worksheet.

4.7 The true positive (TP) ratings

These are found in the `TP` or `LL` worksheet, see below.

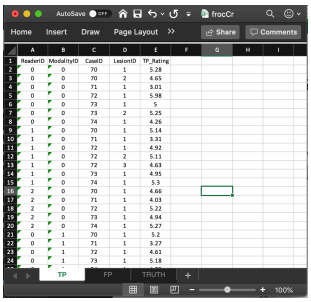

Figure 4.3: Fig. 3: TP/LL worksheet for file inst/extdata/toyFiles/FROC/frocCr.xlsx

- This worksheet can only have diseased cases. The presence of a non-diseased case in this worksheet will generate an error.
- The common vertical length, 31 in this example, is a-priori unpredictable. Given the structure of the `Truth` worsheet for this dataset, the maximum length would be 9 times 2 times 3, assuming every lesion is marked for each modality, reader and diseased case. The 9 comes from the total number of non-zero entries in the `LesionID` column of the `Truth` worksheet.
- The fact that the length is smaller than the maximum length means that there are combinations of modality, reader and diseased cases on which some lesions were not marked.
- As an example, the first lesion in `CaseID` equal to `70` was marked (and rated 5.28) in `ModalityID 0` and `ReaderID 0`.
- The length of the fourth dimension of the `x$LL` list member, 3 in the present example, is determined by the diseased case with the most lesions in the `Truth` worksheet.
- The reader should convince oneself that the ratings in `x$LL` reflect the contents of the `TP` worksheet.

4.8 On the distribution of numbers of lesions in abnormal cases

- Consider a much larger dataset, `dataset11`, with structure as shown below:

```
x <- dataset11
str(x)
#> List of 12
#>  $ NL           : num [1:4, 1:5, 1:158, 1:4] -Inf -Inf -Inf -Inf -Inf ...
#>  $ LL           : num [1:4, 1:5, 1:115, 1:20] -Inf -Inf -Inf -Inf -Inf ...
#>  $ lesionVector : int [1:115] 6 4 7 1 3 3 3 8 11 2 ...
#>  $ lesionID     : num [1:115, 1:20] 1 1 1 1 1 1 1 1 1 1 ...
#>  $ lesionWeight : num [1:115, 1:20] 0.167 0.25 0.143 1 0.333 ...
#>  $ dataType     : chr "FROC"
#>  $ modalityID   : Named chr [1:4] "1" "2" "3" "4"
#>   ..- attr(*, "names")= chr [1:4] "1" "2" "3" "4"
#>  $ readerID     : Named chr [1:5] "1" "2" "3" "4" ...
#>   ..- attr(*, "names")= chr [1:5] "1" "2" "3" "4" ...
#>  $ design       : chr "CROSSED"
#>  $ normalCases  : int [1:43] 6 9 14 27 62 66 70 71 83 91 ...
#>  $ abnormalCases: int [1:115] 1 2 3 5 7 8 10 11 13 17 ...
#>  $ truthTableStr: num [1:4, 1:5, 1:158, 1:21] 1 1 1 1 1 1 1 1 1 1 ...
```

- Focus for now in the 115 abnormal cases.
- The numbers of lesions in these cases is contained in `x$lesionVector`.

```
x$lesionVector
#>   [1]  6  4  7  1  3  3  3  8 11  2  4  6  2 16  5  2  8  3  4  7 11  1  4  3  4
#>  [26]  4  7  3  2  5  2  2  7  6  6  4 10 20 12  6  4  7 12  5  1  1  5  1  2  8
#>  [51]  3  1  2  2  3  2  8 16 10  1  2  2  6  3  2  2  4  6 10 11  1  2  6  2  4
#>  [76]  5  2  9  6  6  8  3  8  7  1  1  6  3  2  1  9  8  8  2  2 12  1  1  1  1
#> [101]  1  3  1  2  2  1  1  1  1  3  1  1  1  2  1
```

- For example, the first abnormal case contains 6 lesions, the second contains 4 lesions, the third contains 7 lesions, etc. and the last abnormal case contains 1 lesion.
- To get an idea of the distribution of the numbers of lesions per abnormal cases, one could interrogate this vector as shown below using the `which()` function:

```
for (el in 1:max(x$lesionVector)) cat(
  "abnormal cases with", el, "lesions = ",
  length(which(x$lesionVector == el)), "\n")
#> abnormal cases with 1 lesions =  25
#> abnormal cases with 2 lesions =  23
#> abnormal cases with 3 lesions =  13
#> abnormal cases with 4 lesions =  10
#> abnormal cases with 5 lesions =  5
#> abnormal cases with 6 lesions =  11
```

```
#> abnormal cases with 7 lesions =  6
#> abnormal cases with 8 lesions =  8
#> abnormal cases with 9 lesions =  2
#> abnormal cases with 10 lesions =  3
#> abnormal cases with 11 lesions =  3
#> abnormal cases with 12 lesions =  3
#> abnormal cases with 13 lesions =  0
#> abnormal cases with 14 lesions =  0
#> abnormal cases with 15 lesions =  0
#> abnormal cases with 16 lesions =  2
#> abnormal cases with 17 lesions =  0
#> abnormal cases with 18 lesions =  0
#> abnormal cases with 19 lesions =  0
#> abnormal cases with 20 lesions =  1
```

- This tells us that 25 cases contain 1 lesion
- Likewise, 23 cases contain 2 lesions
- Etc.

4.8.1 Definition of `lesDistr` array

- Let us ask what is the fraction of (abnormal) cases with 1 lesion, 2 lesions etc.

```
for (el in 1:max(x$lesionVector)) cat("fraction of abnormal cases with", el, "lesions =
                                      length(which(x$lesionVector == el))/leng
#> fraction of abnormal cases with 1 lesions =  0.2173913
#> fraction of abnormal cases with 2 lesions =  0.2
#> fraction of abnormal cases with 3 lesions =  0.1130435
#> fraction of abnormal cases with 4 lesions =  0.08695652
#> fraction of abnormal cases with 5 lesions =  0.04347826
#> fraction of abnormal cases with 6 lesions =  0.09565217
#> fraction of abnormal cases with 7 lesions =  0.05217391
#> fraction of abnormal cases with 8 lesions =  0.06956522
#> fraction of abnormal cases with 9 lesions =  0.0173913
#> fraction of abnormal cases with 10 lesions =  0.02608696
#> fraction of abnormal cases with 11 lesions =  0.02608696
#> fraction of abnormal cases with 12 lesions =  0.02608696
#> fraction of abnormal cases with 13 lesions =  0
#> fraction of abnormal cases with 14 lesions =  0
#> fraction of abnormal cases with 15 lesions =  0
#> fraction of abnormal cases with 16 lesions =  0.0173913
#> fraction of abnormal cases with 17 lesions =  0
#> fraction of abnormal cases with 18 lesions =  0
```

```
#> fraction of abnormal cases with 19 lesions =  0
#> fraction of abnormal cases with 20 lesions =  0.008695652
```

- This tells us that fraction 0.217 of (abnormal) cases contain 1 lesion
- And fraction 0.2 of (abnormal) cases contain 2 lesions
- Etc.
- This information is contained the the `lesDistr` array
- It is coded in the `Utility` function `UtilLesionDistr()`

```
lesDistr <- UtilLesionDistr(x)
lesDistr
#>       [,1]        [,2]
#>  [1,]    1 0.217391304
#>  [2,]    2 0.200000000
#>  [3,]    3 0.113043478
#>  [4,]    4 0.086956522
#>  [5,]    5 0.043478261
#>  [6,]    6 0.095652174
#>  [7,]    7 0.052173913
#>  [8,]    8 0.069565217
#>  [9,]    9 0.017391304
#> [10,]   10 0.026086957
#> [11,]   11 0.026086957
#> [12,]   12 0.026086957
#> [13,]   16 0.017391304
#> [14,]   20 0.008695652
```

- The `UtilLesionDistr()` function returns an array with two columns and number of rows equal to the number of distinct values of lesions per case.
- The first column contains the number of distinct values of lesions per case, 14 in the current example.
- The second column contains the fraction of diseased cases with the number of lesions indicated in the first column.
- The second column must sum to unity

```
sum(UtilLesionDistr(x)[,2])
#> [1] 1
```

- The lesion distribution array will come in handy when it comes to predicting the operating characteristics from using the Radiological Search Model (RSM), as detailed in Chapter 17 of my book.

4.9 Definition of `lesWghtDistr` array

- This is returned by `UtilLesionWeightsDistr()`.
- This contains the same number of rows as `lesDistr`.
- The number of columns is one plus the number of rows as `lesDistr`.
- The first column contains the number of distinct values of lesions per case, 14 in the current example.
- The second column contains the weights of cases with number of lesions per case corresponding to row 1.
- The third column contains the weights of cases with number of lesions per case corresponding to row 2.
- Etc.
- Missing values are filled with `-Inf`.

```
lesWghtDistr <- UtilLesionWeightsDistr(x)
cat("dim(lesDistr) =", dim(lesDistr),"\n")
#> dim(lesDistr) = 14 2
cat("dim(lesWghtDistr) =", dim(lesWghtDistr),"\n")
#> dim(lesWghtDistr) = 14 21
cat("lesWghtDistr = \n\n")
#> lesWghtDistr =
lesWghtDistr
#>       [,1]       [,2]       [,3]       [,4]       [,5]       [,6]       [,7]
#>  [1,]    1 1.00000000       -Inf       -Inf       -Inf       -Inf       -Inf
#>  [2,]    2 0.50000000 0.50000000       -Inf       -Inf       -Inf       -Inf
#>  [3,]    3 0.33333333 0.33333333 0.33333333       -Inf       -Inf       -Inf
#>  [4,]    4 0.25000000 0.25000000 0.25000000 0.25000000       -Inf       -Inf
#>  [5,]    5 0.20000000 0.20000000 0.20000000 0.20000000 0.20000000       -Inf
#>  [6,]    6 0.16666667 0.16666667 0.16666667 0.16666667 0.16666667 0.16666667
#>  [7,]    7 0.14285714 0.14285714 0.14285714 0.14285714 0.14285714 0.14285714
#>  [8,]    8 0.12500000 0.12500000 0.12500000 0.12500000 0.12500000 0.12500000
#>  [9,]    9 0.11111111 0.11111111 0.11111111 0.11111111 0.11111111 0.11111111
#> [10,]   10 0.10000000 0.10000000 0.10000000 0.10000000 0.10000000 0.10000000
#> [11,]   11 0.09090909 0.09090909 0.09090909 0.09090909 0.09090909 0.09090909
#> [12,]   12 0.08333333 0.08333333 0.08333333 0.08333333 0.08333333 0.08333333
#> [13,]   16 0.06250000 0.06250000 0.06250000 0.06250000 0.06250000 0.06250000
#> [14,]   20 0.05000000 0.05000000 0.05000000 0.05000000 0.05000000 0.05000000
#>             [,8]       [,9]       [,10]      [,11]      [,12]      [,13]  [,14]
#>  [1,]       -Inf       -Inf       -Inf       -Inf       -Inf       -Inf   -Inf
#>  [2,]       -Inf       -Inf       -Inf       -Inf       -Inf       -Inf   -Inf
#>  [3,]       -Inf       -Inf       -Inf       -Inf       -Inf       -Inf   -Inf
#>  [4,]       -Inf       -Inf       -Inf       -Inf       -Inf       -Inf   -Inf
#>  [5,]       -Inf       -Inf       -Inf       -Inf       -Inf       -Inf   -Inf
#>  [6,]       -Inf       -Inf       -Inf       -Inf       -Inf       -Inf   -Inf
#>  [7,] 0.14285714       -Inf       -Inf       -Inf       -Inf       -Inf   -Inf
```

```
#>  [8,] 0.12500000 0.12500000       -Inf       -Inf       -Inf       -Inf   -Inf
#>  [9,] 0.11111111 0.11111111 0.11111111       -Inf       -Inf       -Inf   -Inf
#> [10,] 0.10000000 0.10000000 0.10000000 0.10000000       -Inf       -Inf   -Inf
#> [11,] 0.09090909 0.09090909 0.09090909 0.09090909 0.09090909       -Inf   -Inf
#> [12,] 0.08333333 0.08333333 0.08333333 0.08333333 0.08333333 0.08333333   -Inf
#> [13,] 0.06250000 0.06250000 0.06250000 0.06250000 0.06250000 0.06250000 0.0625
#> [14,] 0.05000000 0.05000000 0.05000000 0.05000000 0.05000000 0.05000000 0.0500
#>        [,15]  [,16]  [,17] [,18] [,19] [,20] [,21]
#>  [1,]   -Inf   -Inf   -Inf  -Inf  -Inf  -Inf  -Inf
#>  [2,]   -Inf   -Inf   -Inf  -Inf  -Inf  -Inf  -Inf
#>  [3,]   -Inf   -Inf   -Inf  -Inf  -Inf  -Inf  -Inf
#>  [4,]   -Inf   -Inf   -Inf  -Inf  -Inf  -Inf  -Inf
#>  [5,]   -Inf   -Inf   -Inf  -Inf  -Inf  -Inf  -Inf
#>  [6,]   -Inf   -Inf   -Inf  -Inf  -Inf  -Inf  -Inf
#>  [7,]   -Inf   -Inf   -Inf  -Inf  -Inf  -Inf  -Inf
#>  [8,]   -Inf   -Inf   -Inf  -Inf  -Inf  -Inf  -Inf
#>  [9,]   -Inf   -Inf   -Inf  -Inf  -Inf  -Inf  -Inf
#> [10,]   -Inf   -Inf   -Inf  -Inf  -Inf  -Inf  -Inf
#> [11,]   -Inf   -Inf   -Inf  -Inf  -Inf  -Inf  -Inf
#> [12,]   -Inf   -Inf   -Inf  -Inf  -Inf  -Inf  -Inf
#> [13,] 0.0625 0.0625 0.0625  -Inf  -Inf  -Inf  -Inf
#> [14,] 0.0500 0.0500 0.0500  0.05  0.05  0.05  0.05
```

- Row 3 corresponds to 3 lesions per case and the weights are 1/3, 1/3 and 1/3.
- Row 13 corresponds to 16 lesions per case and the weights are 0.06250000, 0.06250000, ..., repeated 13 times.
- Note that the number of rows is less than the maximum number of lesions per case (20).
- This is because some configurations of lesions per case (e.g., cases with 13 lesions per case) do not occur in this dataset.

4.10 Summary

- The FROC dataset has far less regularity in structure as compared to an ROC dataset.
- The length of the first dimension of either `x$NL` or `x$LL` list members is the total number of modalities, 2 in the current example.
- The length of the second dimension of either `x$NL` or `x$LL` list members is the total number of readers, 3 in the current example.
- The length of the third dimension of `x$NL` is the total number of cases, 8 in the current example. The first three positions account for `NL` marks on non-diseased cases and the remaining 5 positions account for `NL` marks on diseased cases.

- The length of the third dimension of `x$LL` is the total number of diseased cases, 5 in the current example.
- The length of the fourth dimension of `x$NL` is determined by the case (diseased or non-diseased) with the most `NL` marks, 2 in the current example.
- The length of the fourth dimension of `x$LL` is determined by the diseased case with the most lesions, 3 in the current example.

4.11 References

Bibliography

Chakraborty, D. P. (2017). *Observer Performance Methods for Diagnostic Imaging - Foundations, Modeling, and Applications with R-Based Examples*. CRC Press, Boca Raton, FL.