RJafroc Documentation

Dev P. Chakraborty, PhD

2020-03-13

Contents

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 1

Introduction

- This is the book desribing the **RJafroc** package.
- 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

1.1 References

Chapter 2

ROC DATA FORMAT

2.1 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 (Metz, 1978), 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.

2.2 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.

2.3 The Excel data format

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

2.4 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`.

2.5 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`).

2.6 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)
```

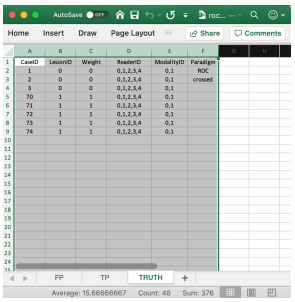


Figure 2.1: Truth worksheet for file rocCr.xlsx

```
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**.

2.7 The false positive (FP) ratings

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

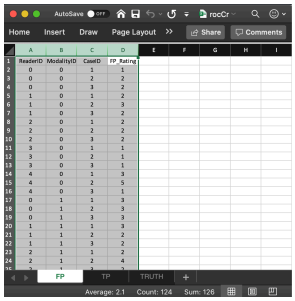


Figure 2.2: FP worksheet for file rocCr.xlsx

- 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.

2.8 The true positive (TP) ratings

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

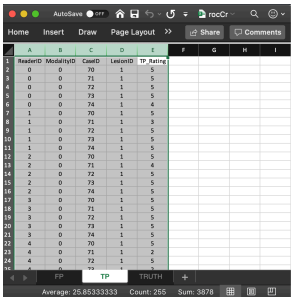


Figure 2.3: TP worksheet for file rocCr.xlsx

- 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.

2.9 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.

2.10 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.

2.11 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)
```

2.12 References

Chapter 3

FROC data format

3.1 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.

3.2 Introduction

- See my book Chakraborty (2017) for full details.
- In the Free-response Receiver Operating Characteristic (FROC) paradigm (Chakraborty, 1989) 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.

3.3 The Excel data format

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

3.4 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**.

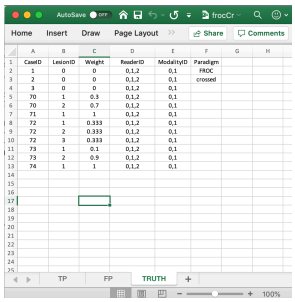


Figure 3.1: Truth worksheet for file inst/extdata/toyFiles/FROC/frocCr.xlsx

3.5 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.

3.6 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 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.

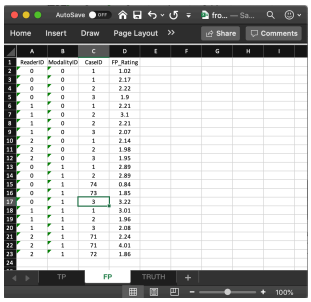


Figure 3.2: Fig. 2: FP/NL worksheet for file inst/extdata/toyFiles/FROC/frocCr.xlsx

- 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.

3.7 The true positive (TP) ratings

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

- This worksheet can only have diseased cases. The presence of a non-

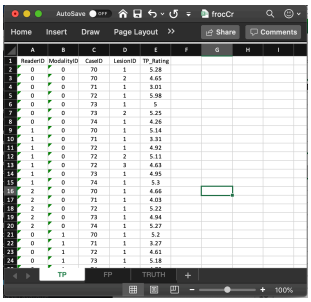


Figure 3.3: Fig. 3: TP/LL worksheet for file inst/extdata/toyFiles/FROC/frocCr.xlsx

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.

3.8 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.

3.8.1 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.

3.9 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.

3.10 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.

3.11 References

Chapter 4

ROC split plot data format

4.1 Introduction

- The purpose of this vignette is to explain the data format of the input Excel file for an ROC *split-plot* dataset.
- In a split-plot dataset each reader interprets a *different* sub-set of cases in all modalities, i.e., the cases interpreted by different readers have no overlap.
- Each sub-set of cases can have different numbers of non-diseased and diseased cases.
- The example below assumes the same numbers of non-diseased and diseased cases.
- The data format has been extended to `NewFormat` to allow such datasets.

4.2 The Excel data format

As before,the Excel file has three worsheets named `Truth`, `NL` or `FP` and `LL` or `TP`. The Excel file corresponding to the example that follows is `inst/extdata/toyFiles/ROC/rocSp.xlsx`.

4.3 The `Truth` worksheet

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

- The first five columns contain as many rows as there are cases in the dataset.

- `CaseID`: unique **integers**, one per case, representing the cases in the dataset.
- `LesionID`: integers 0, representing non-diseased cases and 1 representing the diseased cases.
- The `ReaderID` column is a listing of readers each represented by a **unique string**. Note that, unlike the crossed design, the `ReaderID` column has *single values*. **Each cell has to be text formatted.**
- The non-diseased cases interpreted by reader with `ReaderID` value `1` are labeled `6`, `7`, `8`, `9` and `10`, each with `LesionID` value `0`.
- The diseased cases interpreted by this reader are labeled `16`, `17`, `18`, `19` and `20`, each with `LesionID` value `1`.

- The second reader, with `ReaderID` value `4`, interprets five non-diseased cases labeled `21`, `22`, `23`, `24` and `25`, each with `LesionID` value `0`, and five diseased cases labeled `36`, `37`, `38`, `39` and `40`, each with `LesionID` value `1`.

- The third reader, with ReaderID value `5`, interprets five non-diseased cases labeled `46`, `47`, `48`, `49` and `50`, each with `LesionID` value `0` and five diseased cases labeled `51`, `52`, `53`, `54` and `55`, each with `LesionID` value `1`.

- `Weight`: floating point value 0 - this is not used for ROC data.

- `ModalityID`: a comma-separated listing of modalities, each represented by a **unique string**. In the example shown below each cell has the value `1, 2`. **Each cell has to be text formatted.**
- `Paradigm`: In the example shown in this vignette, the contents are `ROC` and `split-plot`.

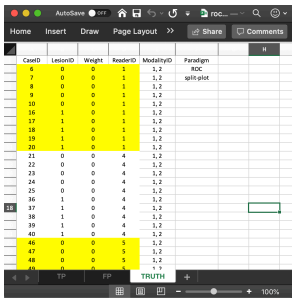


Figure 4.1: Fig. 1: Truth worksheet for file inst/extdata/toyFiles/ROC/rocSp.xlsx

4.4 The structure of the ROC split plot dataset

- The example shown in Fig. 1 corresponds to Excel file `inst/extdata/toyFiles/ROC/rocSp.xlsx` in the project directory.

```
rocSp <- system.file("extdata", "toyFiles/ROC/rocSp.xlsx",
                        package = "RJafroc", mustWork = TRUE)
x <- DfReadDataFile(rocSp, newExcelFileFormat = TRUE)
str(x)
#> List of 12
#>  $ NL           : num [1:2, 1:3, 1:30, 1] 1 1 -Inf -Inf -Inf ...
#>  $ LL           : num [1:2, 1:3, 1:15, 1] 5 2.3 -Inf -Inf -Inf ...
#>  $ lesionVector : int [1:15] 1 1 1 1 1 1 1 1 1 1 ...
#>  $ lesionID     : num [1:15, 1] 1 1 1 1 1 1 1 1 1 1 ...
#>  $ lesionWeight : num [1:15, 1] 1 1 1 1 1 1 1 1 1 1 ...
#>  $ dataType     : chr "ROC"
#>  $ modalityID   : Named chr [1:2] "1" "2"
#>   ..- attr(*, "names")= chr [1:2] "1" "2"
#>  $ readerID     : Named chr [1:3] "1" "4" "5"
#>   ..- attr(*, "names")= chr [1:3] "1" "4" "5"
#>  $ design       : chr "SPLIT-PLOT"
#>  $ normalCases  : int [1:15] 6 7 8 9 10 21 22 23 24 25 ...
#>  $ abnormalCases: int [1:15] 16 17 18 19 20 36 37 38 39 40 ...
#>  $ truthTableStr: num [1:2, 1:3, 1:30, 1:2] 1 1 NA NA NA NA 1 1 NA NA ...
```

- `DfReadDataFile()` flag `newExcelFileFormat` **must** be set to `TRUE` for split plot data.
- The dataset object `x` is a `list` variable with 12 members.
- There are 15 diseased cases in the dataset (the number of 1's in the `LesionID` column of the `Truth` worksheet) and 15 non-diseased cases (the number of 0's in the `LesionID` column).
- `x$NL`, with dimension [2, 3, 30, 1], contains the ratings of normal cases. The extra values in the third dimension, filled with `NAs`, are needed for compatibility with FROC datasets.
- `x$LL`, with dimension [2, 3, 15, 1], contains the ratings of abnormal cases.
- The `x$lesionVector` member is a vector with 15 ones representing the 15 diseased cases in the dataset.
- The `x$lesionID` member is an array with 15 ones (this member is needed for compatibility with FROC datasets).
- The `x$lesionWeight` member is an array with 15 ones (this member is needed for compatibility with FROC datasets).
- The `dataType` member is ROC which specifies the data collection method ("ROC", "FROC", "LROC" or "ROI").
- The `x$modalityID` member is a vector with two elements `"1"` and `"2"`, naming the two modalities.

- The `x$readerID` member is a vector with three elements `"1"`, `"4"` and `"5"`, naming the three modalities.
- The `x$design` member is SPLIT-PLOT; specifies the dataset design, which can be either "CROSSED" or "SPLIT-PLOT".
- The `x$normalCases` member lists the names of the normal cases, 6, 7, 8, 9, 10, 21, 22, 23, 24, 25, 46, 47, 48, 49, 50.
- The `x$abnormalCases` member lists the names of the abnormal cases, 16, 17, 18, 19, 20, 36, 37, 38, 39, 40, 51, 52, 53, 54, 55.
- The `x$truthTableStr` member quantifies the structure of the dataset, as explained next. **It is used in the `DfReadDataFile()` function to check for data entry errors.**

4.5 The `truthTableStr` member

- This is a `2 x 3 x 30 x 2` array, i.e., I x J x K x (maximum number of lesions per case plus 1). The `plus 1` is needed to accommodate normal cases with `lesionID` = 0. [Zero is not a valid array subscript in `R`.]
- Each entry in this array is either `1`, meaning the corresponding interpretation exists, or `NA`, meaning the corresponding interpretation does not exist.
- For example, `x$truthTableStr[1,1,1,1]` is 1. This means that an interpretation exists for the first treatment (`modalityID` = 1), first reader (`readerID` = 1) and first (normal) case (`caseID` = 6 and `lesionID` = 0). This example corresponds to row 2 in the `TRUTH` worksheet.
- The following shows that the first reader interprets the first five normal cases in both modalities.

```
x$truthTableStr[,1,1:15,1]
#>      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14]
#> [1,]    1    1    1    1    1   NA   NA   NA   NA    NA    NA    NA    NA    NA
#> [2,]    1    1    1    1    1   NA   NA   NA   NA    NA    NA    NA    NA    NA
#>      [,15]
#> [1,]    NA
#> [2,]    NA
```

- In the following all elements are `NA` because normal cases correspond to lesionID = 1.

```
x$truthTableStr[,1,1:15,2]
#>      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14]
#> [1,]   NA   NA   NA   NA   NA   NA   NA   NA   NA    NA    NA    NA    NA    NA
#> [2,]   NA   NA   NA   NA   NA   NA   NA   NA   NA    NA    NA    NA    NA    NA
#>      [,15]
```

```
#> [1,]    NA
#> [2,]    NA
```

- The following shows that the second reader interprets the next group of five normal cases, indexed 6 through 10, in both modalities.

```
x$truthTableStr[,2,1:15,1]
#>      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14]
#> [1,]   NA   NA   NA   NA   NA    1    1    1    1     1    NA    NA    NA    NA
#> [2,]   NA   NA   NA   NA   NA    1    1    1    1     1    NA    NA    NA    NA
#>      [,15]
#> [1,]    NA
#> [2,]    NA
```

- The following shows that the third reader interprets the next group of five normal cases, indexed 11 through 15, in both modalities.

```
x$truthTableStr[,3,1:15,1]
#>      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14]
#> [1,]   NA   NA   NA   NA   NA   NA   NA   NA   NA    NA     1     1     1     1
#> [2,]   NA   NA   NA   NA   NA   NA   NA   NA   NA    NA     1     1     1     1
#>      [,15]
#> [1,]     1
#> [2,]     1
```

- The following shows that the first reader interprets the first group of five abnormal cases, indexed 16 through 20, in both modalities.

```
x$truthTableStr[,1,16:30,2]
#>      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14]
#> [1,]    1    1    1    1    1   NA   NA   NA   NA    NA    NA    NA    NA    NA
#> [2,]    1    1    1    1    1   NA   NA   NA   NA    NA    NA    NA    NA    NA
#>      [,15]
#> [1,]    NA
#> [2,]    NA
```

- In the following all elements are `NA` because abnormal cases correspond to `lesionID` = 2.

```
x$truthTableStr[,1,16:30,1]
#>      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14]
#> [1,]   NA   NA   NA   NA   NA   NA   NA   NA   NA    NA    NA    NA    NA    NA
```

```
#> [2,]   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA
#>      [,15]
#> [1,]    NA
#> [2,]    NA
```

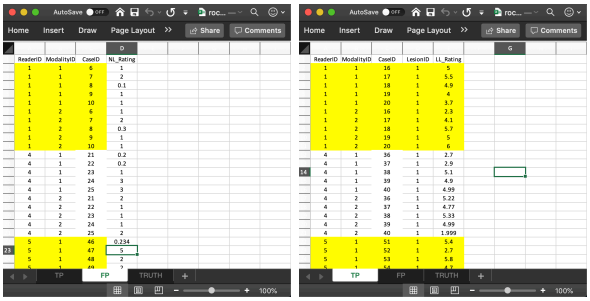


Figure 4.2: Fig. 2 FP/TP worksheets; LEFT=FP, (b) RIGHT=TP

4.6 The false positive (FP) ratings

- These are found in the `FP` or `NL` worksheet, see Fig. 2, left panel.
- This worksheet has the ratings of non-diseased cases.
- The common vertical length is 30 in this example (2 modalities times 3 readers times 5 non-diseased cases per reader).
- `ReaderID`: the reader labels: these must be from `1`, `4` or `5`, as declared in the `Truth` worksheet.
- `ModalityID`: the modality labels: `1` or `2`, as declared in the `Truth` worksheet.
- `CaseID`: the labels of non-diseased cases. Each `CaseID - ReaderID` combination must be consistent with that declared in the `Truth` worsheet.

- `NL_Rating`: the floating point ratings of non-diseased cases. Each row of this worksheet yields a rating corresponding to the values of `ReaderID`, `ModalityID` and `CaseID` for that row.

```
x$NL[,1,1:15,1]
#>      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14]
#> [1,]    1    2  0.1    1    1 -Inf -Inf -Inf -Inf  -Inf  -Inf  -Inf  -Inf  -Inf
#> [2,]    1    2  0.3    1    1 -Inf -Inf -Inf -Inf  -Inf  -Inf  -Inf  -Inf  -Inf
#>      [,15]
#> [1,]  -Inf
#> [2,]  -Inf
x$NL[,2,1:15,1]
```

```
#>      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14]
#> [1,] -Inf -Inf -Inf -Inf -Inf  0.2  0.2    1    3     3  -Inf  -Inf  -Inf  -Inf
#> [2,] -Inf -Inf -Inf -Inf -Inf  2.0  1.0    1    1     2  -Inf  -Inf  -Inf  -Inf
#>      [,15]
#> [1,]  -Inf
#> [2,]  -Inf
x$NL[,3,1:15,1]
#>      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14]
#> [1,] -Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf  -Inf 0.234     5     2     2
#> [2,] -Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf  -Inf 3.000     2     2     2
#>      [,15]
#> [1,]  2.00
#> [2,]  0.33
```

- The first line of the above code shows the ratings, in both modalities, of the first five non-diseased cases with `CaseID`s `6,7,8,9,10` (indexed 1, 2, 3, 4, 5 and appearing in the first five columns) interpreted by the first reader (`ReaderID` 1).
- The second line shows the ratings, in both modalities, of the next five non-diseased cases with `CaseID`s `21,22,23,24,25` (indexed 6, 7, 8, 9, 10and appearing in the next five columns) interpreted by the second reader (`ReaderID` 4).
- The third line shows the ratings, in both modalities, of the final five non-diseased cases with `CaseID`s `46,47,48,49,50` (indexed 11, 12, 13, 14, 15and appearing in the final five columns) interpreted by the third reader (`ReaderID` 5).
- Values such as `x$NL[,,16:30,1]`, which are there for compatibility with FROC data, are all filled with `-Inf`.

4.7 The true positive (TP) ratings

- These are found in the `TP` or `LL` worksheet, see Fig. 2, right panel.
- This worksheet has the ratings of diseased cases.
- The common vertical length is 30 in this example (2 modalities times 3 readers times 5 diseased cases per reader).
- `ReaderID`: the reader labels: these must be from `1`, `4` or `5`, as declared in the `Truth` worksheet.
- `ModalityID`: the modality labels: `1` or `2`, as declared in the `Truth` worksheet.
- `CaseID`: the labels of diseased cases. Each `CaseID` - `ReaderID` combination must be consistent with that declared in the `Truth` worsheet.

- `LL_Rating`: the floating point ratings of diseased cases. Each row of this worksheet yields a rating corresponding to the values of `ReaderID`, `ModalityID` and `CaseID` for that row.

```
x$LL[,1,1:15,1]
#>      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14]
#> [1,]  5.0  5.5  4.9    4  3.7 -Inf -Inf -Inf -Inf  -Inf  -Inf  -Inf  -Inf  -Inf
#> [2,]  2.3  4.1  5.7    5  6.0 -Inf -Inf -Inf -Inf  -Inf  -Inf  -Inf  -Inf  -Inf
#>      [,15]
#> [1,]  -Inf
#> [2,]  -Inf
x$LL[,2,1:15,1]
#>      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14]
#> [1,] -Inf -Inf -Inf -Inf -Inf 2.70 2.90 5.10 4.90 4.990  -Inf  -Inf  -Inf  -Inf
#> [2,] -Inf -Inf -Inf -Inf -Inf 5.22 4.77 5.33 4.99 1.999  -Inf  -Inf  -Inf  -Inf
#>      [,15]
#> [1,]  -Inf
#> [2,]  -Inf
x$LL[,3,1:15,1]
#>      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14]
#> [1,] -Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf  -Inf   5.4   2.7   5.8   4.7
#> [2,] -Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf  -Inf   5.4   2.7   5.8   4.7
#>      [,15]
#> [1,]     5
#> [2,]     5
```

- The first line of code shows the ratings, in both modalities, of the first five diseased cases with `CaseIDs` `16,17,18,19,20` (indexed 1, 2, 3, 4, 5and appearing in the first five columns) interpreted by the first reader (`ReaderID` 1).
- The second line shows the ratings, in both modalities, of the next five diseased cases with `CaseIDs` `36,37,38,39,40` (indexed 6, 7, 8, 9, 10and appearing in the next five columns) interpreted by the second reader (`ReaderID` 4).
- The third line shows the ratings, in both modalities, of the final five non-diseased cases with `CaseIDs` `51,52,53,54,55` (indexed 11, 12, 13, 14, 15and appearing in the final five columns) interpreted by the third reader (`ReaderID` 5).

4.8 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.

4.9 References

Chapter 5

FROC ROC DATA FORMAT SPLIT PLOT

5.1 Introduction

- The purpose of this vignette is to explain the data format of the input Excel file for an FROC *split-plot* dataset.
- In a split-plot dataset each reader interprets a sub-set of cases in all modalities.
- The cases interpreted by different readers have no overlap.
- It is assumed, for now, that each sub-set of cases has the same numbers of non-diseased and diseased cases.

5.2 The Excel data format

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

5.3 The `Truth` worksheet

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

- The first five columns contain as many rows as there are non-diseased cases (9) plus total number of lesions (27) in the dataset (each row with a non-zero `LesionID` corresponds to a lesion).

- `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 three non-diseased cases interpreted by reader with `ReaderID` value `0` are labeled `1`, `2`, `3`, while the diseased cases interpreted by this reader are labeled `70`, `71`, `72`, `73` and `74`, with `LesionID` values ranging from 1 to 3.

- The second reader, with `ReaderID` value `1`, interprets three non-diseased cases labeled `4`, `5` and `6`, each with `LesionID` value `0`, and five diseased cases labeled `80`, `81`, `82`, `83` and `84`, with `LesionID` values ranging from 1 to 3.

- The third reader, with `ReaderID` value `2`, interprets three non-diseased cases labeled `7`, `8` and `9`, each with `LesionID` value `0` and five diseased cases labeled `90`, `91`, `92`, `93` and `94`, with `LesionID` values ranging from 1 to 3.

- `Weight`: floating point value adding upto unity for diseased cases as required for FROC data.
- `ModalityID`: a comma-separated listing of modalities, each represented by a unique **integer**. In the example shown below 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 `split-plot`.

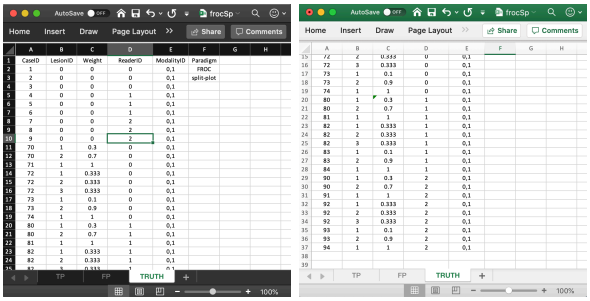


Figure 5.1: Two views of Truth worksheet for file frocSp.xlsx

5.4 The structure of the FROC split plot dataset

The example shown in Fig. 1 corresponds to Excel file `inst/extdata/toyFiles/FROC/frocSp.xlsx` in the project directory.

```
frocSp <- system.file("extdata", "toyFiles/FROC/frocSp.xlsx",
                      package = "RJafroc", mustWork = TRUE)
x <- DfReadDataFile(frocSp, newExcelFileFormat = TRUE)
str(x)
#> List of 12
#>  $ NL           : num [1:2, 1:3, 1:24, 1:3] 1.02 2.89 -Inf -Inf -Inf ...
#>  $ LL           : num [1:2, 1:3, 1:15, 1:3] 5.28 5.2 -Inf -Inf -Inf ...
#>  $ lesionVector : int [1:15] 2 1 3 2 1 2 1 3 2 1 ...
#>  $ lesionID     : num [1:15, 1:3] 1 1 1 1 1 1 1 1 1 1 ...
#>  $ lesionWeight : num [1:15, 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 "SPLIT-PLOT"
#>  $ normalCases  : int [1:9] 1 2 3 4 5 6 7 8 9
#>  $ abnormalCases: int [1:15] 70 71 72 73 74 80 81 82 83 84 ...
#>  $ truthTableStr: num [1:2, 1:3, 1:24, 1:4] 1 1 NA NA NA NA 1 1 NA NA ...
```

- Flag `newExcelFileFormat` **must** be set to `TRUE` for split plot data.
- The dataset object `x` is a `list` variable with 12 members.
- Note that the `dataType` member is FROC and the `design` member is SPLIT-PLOT.
- There are 15 diseased cases in the dataset (the number of 1's in the `LesionID` column of the `Truth` worksheet) and 9 non-diseased cases (the number of 0's in the `LesionID` column).
- The `x$lesionVector` member is a vector with 15 ones representing the 15 diseased cases in the dataset.
- The `x$lesionID` member is a 15 x 3 array labeling the lesions in the dataset.
- The `x$lesionWeight` member is a 15 x 3 array.

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

```
#>  [4,]    1    2 -Inf
#>  [5,]    1 -Inf -Inf
#>  [6,]    1    2 -Inf
#>  [7,]    1 -Inf -Inf
#>  [8,]    1    2    3
#>  [9,]    1    2 -Inf
#> [10,]    1 -Inf -Inf
#> [11,]    1    2 -Inf
#> [12,]    1 -Inf -Inf
#> [13,]    1    2    3
#> [14,]    1    2 -Inf
#> [15,]    1 -Inf -Inf
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
#>  [6,] 0.3000000 0.7000000      -Inf
#>  [7,] 1.0000000      -Inf      -Inf
#>  [8,] 0.3333333 0.3333333 0.3333333
#>  [9,] 0.1000000 0.9000000      -Inf
#> [10,] 1.0000000      -Inf      -Inf
#> [11,] 0.3000000 0.7000000      -Inf
#> [12,] 1.0000000      -Inf      -Inf
#> [13,] 0.3333333 0.3333333 0.3333333
#> [14,] 0.1000000 0.9000000      -Inf
#> [15,] 1.0000000      -Inf      -Inf
```

- The `x$truthTableStr` member is a `2 x 3 x 24 x 4` array, i.e., I x J x K x (maximum number of lesions per case plus 1). The `plus 1` is needed to accommodate normal cases with `lesionID` = 0.
- Each entry in this array is either `1`, meaning the corresponding interpretation exists, or `NA`, meaning the corresponding interpretation does not exist.
- For example, `x$truthTableStr[1,1,1,1]` is 1. This means that an interpretation exists for the first treatment (`modalityID` = 0), first reader (`readerID` = 0) and first (normal) case `caseID` = 1 and `lesionID` = 0. This example corresponds to row 2 in the `TRUTH` worksheet.
- `x$truthTableStr[1,1,4,1]` is NA, which means an interpretation does not exist for the first treatment, first reader and fourth (normal) case.
- However, `x$truthTableStr[1,2,4,1]` is 1, which means an interpretation does exist for the first treatment, second reader and fourth (normal) case. This example corresponds to row 5 in the `TRUTH` worksheet.

- Likewise, `x$truthTableStr[1,1,10,3]` is 1, which means an interpretation does exist for the first treatment, first reader, tenth (abnormal) case and `lesionID` = 2. This example corresponds to row 12 in the `TRUTH` worksheet.
- As an aside, in the FROC paradigm an interpretation need not yield a mark-rating pair. An interpretation means the reader was "exposed to" and had the opportunity to mark the corresponding treatment-reader-case-lesion combination.
- The reader should confirm that the contents of `x$truthTableStr` summarizes the structure of the data in the `TRUTH` worksheet.

5.5 The false positive (FP) ratings

These are found in the `FP` or `NL` worksheet, see Fig. 2.

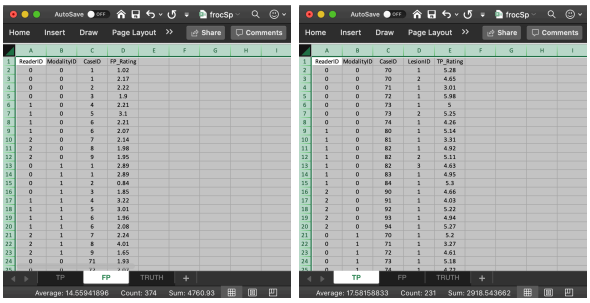


Figure 5.2: NL/FP worksheet, left, and LL/TP worksheet, right, for file frocSp.xlsx

- This worksheet has the ratings of non-diseased cases.
- The common vertical length is 30 in this example (2 modalities times 3 readers times 5 non-diseased cases per reader).
- `ReaderID`: the reader labels: these must be from `0`, `1` or `2`, as declared in the `Truth` worksheet.
- `ModalityID`: the modality labels: `0` or `1`, as declared in the `Truth` worksheet.
- `CaseID`: the labels of non-diseased cases. Each `CaseID`, `ModalityID`, `ReaderID` combination must be consistent with that declared in the `Truth` worsheet.

- `FP_Rating`: the floating point ratings of non-diseased cases. Each row of this worksheet yields a rating corresponding to the values of `ReaderID`, `ModalityID` and `CaseID` for that row. Each `CaseID`, `ModalityID`, `ReaderID` combination must be consistent with that declared in the `Truth` worsheet.

```
x$NL[,1,1:9,1]
#>      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
#> [1,] 1.02 2.22 1.90 -Inf -Inf -Inf -Inf -Inf -Inf
#> [2,] 2.89 0.84 1.85 -Inf -Inf -Inf -Inf -Inf -Inf
x$NL[,2,1:9,1]
#>      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
#> [1,] -Inf -Inf -Inf 2.21 3.10 2.21 -Inf -Inf -Inf
#> [2,] -Inf -Inf -Inf 3.22 3.01 1.96 -Inf -Inf -Inf
x$NL[,3,1:9,1]
#>      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
#> [1,] -Inf -Inf -Inf -Inf -Inf -Inf 2.14 1.98 1.95
#> [2,] -Inf -Inf -Inf -Inf -Inf -Inf 2.24 4.01 1.65
```

- The first line of the above code shows the ratings, in both modalities, of the first three non-diseased cases with `CaseID`s `1,3,3` (indexed 1, 2, 3 and appearing in the first three columns) interpreted by the first reader (`ReaderID` `0`).
- The second line shows the ratings, in both modalities, of the next three non-diseased cases with `CaseID`s `4,5,6` (indexed 4, 5, 6and appearing in the next three columns) interpreted by the second reader (`ReaderID` `1`).
- The third line shows the ratings, in both modalities, of the final three non-diseased cases with `CaseID`s `7,8,9` (indexed 7, 8, 9and appearing in the final three columns) interpreted by the third reader (`ReaderID` `2`).
- Values such as `x$NL[,,16:30,1]`, which are there for compatibility with FROC data, are all filled with `-Inf`.

5.6 The true positive (TP) ratings

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

- This worksheet has the ratings of diseased cases.
- The common vertical length is 30 in this example (2 modalities times 3 readers times 5 diseased cases per reader).
- `ReaderID`: the reader labels: these must be from `0`, `1` or `2`, as declared in the `Truth` worksheet.
- `ModalityID`: the modality labels: `0` or `1`, as declared in the `Truth` worksheet.
- `CaseID`: the labels of diseased cases. Each `CaseID`, `ModalityID`, `ReaderID` combination must be consistent with that declared in the `Truth` worsheet.

- `TP_Rating`: the floating point ratings of diseased cases. Each row of this worksheet yields a rating corresponding to the values of `ReaderID`, `ModalityID` and `CaseID` for that row. Each `CaseID`, `ModalityID`,

`ReaderID` combination must be consistent with that declared in the `Truth` worsheet.

```
x$LL[,1,1:15,1]
#>      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14]
#> [1,] 5.28 3.01 5.98 5.00 4.26 -Inf -Inf -Inf -Inf  -Inf  -Inf  -Inf  -Inf  -Inf
#> [2,] 5.20 3.27 4.61 5.18 4.72 -Inf -Inf -Inf -Inf  -Inf  -Inf  -Inf  -Inf  -Inf
#>      [,15]
#> [1,]  -Inf
#> [2,]  -Inf
x$LL[,2,1:15,1]
#>      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14]
#> [1,] -Inf -Inf -Inf -Inf -Inf 5.14 3.31 4.92 4.95  5.30  -Inf  -Inf  -Inf  -Inf
#> [2,] -Inf -Inf -Inf -Inf -Inf 4.77 3.19 5.20 5.39  5.01  -Inf  -Inf  -Inf  -Inf
#>      [,15]
#> [1,]  -Inf
#> [2,]  -Inf
x$LL[,3,1:15,1]
#>      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14]
#> [1,] -Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf  -Inf  4.66  4.03  5.22  4.94
#> [2,] -Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf -Inf  -Inf  4.87  1.94  -Inf  -Inf
#>      [,15]
#> [1,]  5.27
#> [2,]  4.78
```

- The first line of code shows the ratings, in both modalities, of the first five diseased cases with `CaseID`s `70,71,72,73,74` (indexed 1, 2, 3, 4, 5 and appearing in the first five columns) interpreted by the first reader (`ReaderID` `0`).
- The second line shows the ratings, in both modalities, of the next five diseased cases with `CaseID`s `80,81,82,83,84` (indexed 6, 7, 8, 9, 10 and appearing in the next five columns) interpreted by the second reader (`ReaderID` `1`).
- The third line shows the ratings, in both modalities, of the final five non-diseased cases with `CaseID`s `90,91,92,93,94` (indexed 11, 12, 13, 14, 15 and appearing in the final five columns) interpreted by the third reader (`ReaderID` `2`).

5.7 Summary

- TBA

5.8 References

Chapter 6

QUICK START DBM1

6.1 Introduction

- This vignette is intended for those seeking a quick transition from Windows **JAFROC** to `RJafroc`.
- Described first is the structure of an `RJafroc` dataset followed by how to read a *JAFROC* format Excel file to create an `RJafroc` dataset.

6.2 An ROC dataset

Dataset `dataset03` corresponding to the Franken ROC data (Franken et al., 1992) is predefined. The following code shows the structure of this dataset.

```
str(dataset03)
#> List of 12
#>  $ NL           : num [1:2, 1:4, 1:100, 1] 3 3 4 3 3 3 4 1 1 3 ...
#>  $ LL           : num [1:2, 1:4, 1:67, 1] 5 5 4 4 5 4 4 5 2 2 ...
#>  $ lesionVector : num [1:67] 1 1 1 1 1 1 1 1 1 1 ...
#>  $ lesionID     : num [1:67, 1] 1 1 1 1 1 1 1 1 1 1 ...
#>  $ lesionWeight : num [1:67, 1] 1 1 1 1 1 1 1 1 1 1 ...
#>  $ dataType     : chr "ROC"
#>  $ modalityID   : Named chr [1:2] "TREAT1" "TREAT2"
#>   ..- attr(*, "names")= chr [1:2] "TREAT1" "TREAT2"
#>  $ readerID     : Named chr [1:4] "READER_1" "READER_2" "READER_3" "READER_4"
#>   ..- attr(*, "names")= chr [1:4] "READER_1" "READER_2" "READER_3" "READER_4"
#>  $ design       : chr "CROSSED"
#>  $ normalCases  : int [1:33] 1 2 3 4 5 6 7 8 9 10 ...
```

```
#>  $ abnormalCases: int [1:67] 34 35 36 37 38 39 40 41 42 43 ...
#>  $ truthTableStr: num [1:2, 1:4, 1:100, 1:2] 1 1 1 1 1 1 1 1 1 1 ...
```

- It is a list with 8 members. The false positive ratings are contained in `{NL}`, an array with dimensions `[1:2,1:4,1:100,1]`. The first index corresponds to treatments, and since the dataset has 2 treatments, the corresponding dimension is 2. The second index corresponds to readers, and since the dataset has 4 readers, the corresponding dimension is 4. The third index corresponds to the total number of cases. Since the dataset has 100 cases, the corresponding dimension is 100. But, as you can see from the code below, the entries in this array for cases 34 through 100 are `-Inf`: i.e., `all(dataset03$NL[1,1,34:100,1] == -Inf)` = TRUE.
- This is because in the ROC paradigm false positive are not possible on diseased cases. So the actual FP ratings are contained in the first 33 elements of the array. How did I know that there are 33 non-diseased cases? This can be understood in several ways.
- `LL` is an array with dimensions `[1:2,1:4,1:67,1]`. This implies 67 diseased cases, and by subtraction from 100, there must be 33 non-diseased cases.
- The list member `lesionVector` is a vector with length 67, implying 33 non-diseased cases.
- The list members `lesionID` and `lesionWeight` are arrays with dimensions `[1:67,1]` containing ones. Again, these imply 67 diseased cases.
- The fields `lesionVector`, `lesionID` and `lesionWeight`, while not needed for ROC data, are needed for the FROC paradigm.

The `dataType` list member is the character string `"ROC"`, characterizing the ROC dataset.

```
dataset03$dataType
#> [1] "ROC"
```

The `modalityID` list member is a character string with two entries, `"TREAT1"` and `"TREAT2"`, corresponding to the two modalities.

```
dataset03$modalityID
#>    TREAT1    TREAT2
#> "TREAT1" "TREAT2"
```

The `readerID` list member is a character string with four entries, `"READER_1"`, `"READER_2"`, `"READER_3"` and `"READER_4"` corresponding to the four readers.

```
dataset03$readerID
#>   READER_1   READER_2   READER_3   READER_4
#> "READER_1" "READER_2" "READER_3" "READER_4"
```

Here are the actual ratings for cases 1:34.

```
dataset03$NL[1,1,1:33,1]
#>  [1] 3 1 2 2 2 2 2 4 1 1 4 2 1 2 4 2 1 2 1 2 4 2 3 2 2 2 4 3 2 2 2 5 3
```

- This says that for treatment 1 and reader 1, (non-diseased) case 1 was rated 3, case 2 was rated 1, cases 3-7 were rated 2, case 8 was rated 4, etc.
- As another example, for treatment 2 and reader 3, the FP ratings are:

```
dataset03$NL[2,3,1:33,1]
#>  [1] 3 1 2 2 2 2 4 4 2 3 2 2 1 3 2 4 2 3 2 2 2 2 2 2 4 2 2 1 2 2 2 2 4 2
```

6.3 Creating a dataset from a JAFROC format file

There is a file `RocData.xlsx` that is part of the package installation. Since it is a system file one must get its name as follows.

```
fileName <- "RocData.xlsx"
sysFileName <- system.file(paste0("extdata/",fileName), package = "RJafroc", mustWork = TRUE)
```

Next, one uses `DfReadDataFile()` as follows, assuming it is a JAFROC format file.

```
ds <- DfReadDataFile(sysFileName, newExcelFileFormat = FALSE)
str(ds)
#> List of 12
#>  $ 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 ...
#>  $ lesionVector : 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   : 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 [1:69] 1 2 3 4 5 6 7 8 9 10 ...
#>  $ abnormalCases: int [1:45] 70 71 72 73 74 75 76 77 78 79 ...
#>  $ truthTableStr: num [1:2, 1:5, 1:114, 1:2] 1 1 1 1 1 1 1 1 1 1 ...
```

Analysis is illustrated for `dataset03`, but one could have used the newly created dataset `ds`.

6.4 Analyzing the ROC dataset

This illustrates the `StSignificanceTesting()` function. The significance testing method is specified as `"DBMH"` and the figure of merit `FOM` is specified as "Wilcoxon".

```
ret <- StSignificanceTesting(dataset03, FOM = "Wilcoxon", method = "DBMH")
print(ret)
#> $fomArray
#>            RdrREADER_1 RdrREADER_2 RdrREADER_3 RdrREADER_4
#> TrtTREAT1   0.8534600   0.8649932   0.8573044   0.8152420
#> TrtTREAT2   0.8496156   0.8435097   0.8401176   0.8143374
#>
#> $anovaY
#>       Source           SS  DF          MS
#> 1     Row1_T   0.02356541   1 0.023565410
#> 2     Row2_R   0.20521800   3 0.068406000
#> 3     Row3_C  52.52839868  99 0.530589886
#> 4    Row4_TR   0.01506079   3 0.005020264
#> 5    Row5_TC   6.41004881  99 0.064747968
#> 6    Row6_RC  39.24295381 297 0.132131158
#> 7   Row7_TRC  22.66007764 297 0.076296558
#> 8 Row8_Total 121.08532315 799          NA
#>
#> $anovaYi
#>   Source  DF  TrtTREAT1  TrtTREAT2
#> 1      R   3 0.04926635 0.02415991
#> 2      C  99 0.29396753 0.30137032
#> 3     RC 297 0.10504787 0.10337984
#>
#> $varComp
#>            varR       varC          varTR        varTC     varRC     varErr
#> 1 3.775568e-05 0.05125091 -0.0007127629 -0.002887147 0.0279173 0.07629656
#>
```

```
#> $FTestStatsRRRC
#>      fRRRC ndfRRRC ddfRRRC     pRRRC
#> 1 4.694058       1       3 0.1188379
#>
#> $ciDiffTrtRRRC
#>               TrtDiff   Estimate      StdErr DF        t     PrGTt      CILower
#> 1 TrtTREAT1-TrtTREAT2 0.01085482 0.005010122  3 2.166577 0.1188379 -0.005089627
#>      CIUpper
#> 1 0.02679926
#>
#> $ciAvgRdrEachTrtRRRC
#>   Treatment      Area     StdErr        DF   CILower   CIUpper
#> 1 TrtTREAT1 0.8477499 0.02440215  70.12179 0.7990828 0.8964170
#> 2 TrtTREAT2 0.8368951 0.02356642 253.64403 0.7904843 0.8833058
#>
#> $FTestStatsFRRC
#>      fFRRC ndfFRRC ddfFRRC    pFRRC
#> 1 0.363956       1      99 0.547697
#>
#> $ciDiffTrtFRRC
#>             Treatment   Estimate     StdErr DF         t    PrGTt     CILower
#> 1 TrtTREAT1-TrtTREAT2 0.01085482 0.01799277 99 0.6032876 0.547697 -0.02484675
#>      CIUpper
#> 1 0.04655638
#>
#> $ciAvgRdrEachTrtFRRC
#>   Treatment      Area     StdErr DF   CILower   CIUpper
#> 1 TrtTREAT1 0.8477499 0.02710939 99 0.7939590 0.9015408
#> 2 TrtTREAT2 0.8368951 0.02744860 99 0.7824311 0.8913591
#>
#> $msAnovaEachRdrFRRC
#>   Source DF  RdrREADER_1 RdrREADER_2 RdrREADER_3  RdrREADER_4
#> 1      T  1 0.0007389761  0.02307702  0.01476929 4.091217e-05
#> 2      C 99 0.2038747746  0.22344191  0.21424677 2.854199e-01
#> 3     TC 99 0.0915587344  0.08027926  0.06122898 6.057067e-02
#>
#> $ciDiffTrtEachRdrFRRC
#>        Reader           Treatment     Estimate     StdErr DF          t
#> 1 RdrREADER_1 TrtTREAT1-TrtTREAT2 0.0038444143 0.04279223 99 0.08983908
#> 2 RdrREADER_2 TrtTREAT1-TrtTREAT2 0.0214834916 0.04006975 99 0.53615233
#> 3 RdrREADER_3 TrtTREAT1-TrtTREAT2 0.0171867933 0.03499399 99 0.49113552
#> 4 RdrREADER_4 TrtTREAT1-TrtTREAT2 0.0009045681 0.03480536 99 0.02598933
#>       PrGTt     CILower    CIUpper
#> 1 0.9285966 -0.08106465 0.08875348
#> 2 0.5930559 -0.05802359 0.10099057
```

```
#> 3 0.6244176 -0.05224888 0.08662247
#> 4 0.9793182 -0.06815683 0.06996596
#>
#> $FTestStatsRRFC
#>      fRRFC ndfRRFC ddfRRFC     pRRFC
#> 1 4.694058       1       3 0.1188379
#>
#> $ciDiffTrtRRFC
#>             Treatment   Estimate      StdErr DF        t     PrGTt      CILower
#> 1 TrtTREAT1-TrtTREAT2 0.01085482 0.005010122  3 2.166577 0.1188379 -0.005089627
#>      CIUpper
#> 1 0.02679926
#>
#> $ciAvgRdrEachTrtRRFC
#>   Treatment      Area     StdErr DF   CILower   CIUpper
#> 1 TrtTREAT1 0.8477499 0.01109801  3 0.8124311 0.8830687
#> 2 TrtTREAT2 0.8368951 0.00777173  3 0.8121620 0.8616282
```

6.5 Explanation of the output

The function returns a long unwieldy list. Let us consider them one by one. The function `UtilOutputReport()`, which can generate an Excel file report, making it much easier to visualize the results, is described in another vignette.

6.5.1 FOMs

- `fomArray` contains the `[1:2,1:4]` FOM values.

```
ret$fomArray
#>           RdrREADER_1 RdrREADER_2 RdrREADER_3 RdrREADER_4
#> TrtTREAT1   0.8534600   0.8649932   0.8573044   0.8152420
#> TrtTREAT2   0.8496156   0.8435097   0.8401176   0.8143374
```

This shows the 2 x 4 array of FOM values.

6.5.2 Pseudovalue ANOVA table

- `anovaY`, where the Y denotes that these are pseudovalue based, is the ANOVA table.

```
ret$anovaY
#>       Source           SS  DF          MS
#> 1     Row1_T   0.02356541   1 0.023565410
#> 2     Row2_R   0.20521800   3 0.068406000
#> 3     Row3_C  52.52839868  99 0.530589886
#> 4    Row4_TR   0.01506079   3 0.005020264
#> 5    Row5_TC   6.41004881  99 0.064747968
#> 6    Row6_RC  39.24295381 297 0.132131158
#> 7   Row7_TRC  22.66007764 297 0.076296558
#> 8 Row8_Total 121.08532315 799          NA
```

6.5.3 Pseudovalue ANOVA table, each treatment

- `anovaYi` is the ANOVA table for individual treatments.

```
ret$anovaYi
#>   Source  DF  TrtTREAT1  TrtTREAT2
#> 1      R   3 0.04926635 0.02415991
#> 2      C  99 0.29396753 0.30137032
#> 3     RC 297 0.10504787 0.10337984
```

The `0` and `1` headers come from the treatment names.

6.5.4 Pseudovalue Variance Components

- `varComp` is the variance components (needed for sample size estimation).

```
ret$varComp
#>            varR       varC          varTR        varTC     varRC     varErr
#> 1 3.775568e-05 0.05125091 -0.0007127629 -0.002887147 0.0279173 0.07629656
```

6.5.5 Random-reader random-case (RRRC) analysis

- `ret$FTestStatsRRRC$fRRRC` is the F-statistic for testing the NH that the treatments have identical FOMs. RRRC means random-reader random-case generalization.

```
ret$FTestStatsRRRC$fRRRC
#> [1] 4.694058
```

6.5.5.1 F-statistic and p-value for RRRC analysis

- `ret$FTestStatsRRRC$ddfRRRC` is the denominator degrees of freedom of the F-statistic.

```
ret$FTestStatsRRRC$ddfRRRC
#> [1] 3
```

- `ret$FTestStatsRRRC$pRRRC` is the p-value of the test.

```
ret$FTestStatsRRRC$pRRRC
#> [1] 0.1188379
```

6.5.5.2 Confidence Intervals for RRRC analysis

- `ciDiffTrtRRRC` is the 95% confidence interval of reader-averaged differences between treatments.

```
ret$ciDiffTrtRRRC
#>               TrtDiff   Estimate      StdErr DF        t     PrGTt      CILower
#> 1 TrtTREAT1-TrtTREAT2 0.01085482 0.005010122  3 2.166577 0.1188379 -0.005089627
#>      CIUpper
#> 1 0.02679926
```

- `ciAvgRdrEachTrtRRRC` is the 95% confidence interval of reader-averaged FOMs for each treatments.

```
ret$ciAvgRdrEachTrtRRRC
#>   Treatment      Area     StdErr        DF   CILower   CIUpper
#> 1 TrtTREAT1 0.8477499 0.02440215  70.12179 0.7990828 0.8964170
#> 2 TrtTREAT2 0.8368951 0.02356642 253.64403 0.7904843 0.8833058
```

6.5.6 Fixed-reader random-case (FRRC) analysis

6.5.6.1 F-statistic and p-value for FRRC analysis

- `ret$FTestStatsFRRC$fFRRC` is the F-statistic for fixed-reader random-case analysis.

```
ret$FTestStatsFRRC$fFRRC
#> [1] 0.363956
```

- `ret$FTestStatsFRRC$ndfFRRC` is the numerator degrees of freedom of the F-statistic, always one less than the number of treatments.

```
ret$FTestStatsFRRC$ndfFRRC
#> [1] 1
```

- `ret$FTestStatsFRRC$ddfFRRC` is the denominator degreesof freedom of the F-statistic, for fixed-reader random-case analysis.

```
ret$FTestStatsFRRC$ddfFRRC
#> [1] 99
```

- `ret$FTestStatsFRRC$pFRRC` is the p-value for fixed-reader random-case analysis.

```
ret$FTestStatsFRRC$pFRRC
#> [1] 0.547697
```

6.5.6.2 Confidence Intervals for FRRC analysis

- `ciDiffTrtFRRC` is the 95% CI of reader-average differences between treatments for fixed-reader random-case analysis

```
ret$ciDiffTrtFRRC
#>              Treatment   Estimate     StdErr DF         t    PrGTt     CILower
#> 1 TrtTREAT1-TrtTREAT2 0.01085482 0.01799277 99 0.6032876 0.547697 -0.02484675
#>      CIUpper
#> 1 0.04655638
```

- `ret$ciAvgRdrEachTrtFRRC` is the 95% CI of reader-average FOMs of each treatment for fixed-reader random-case analysis

```
ret$ciAvgRdrEachTrtFRRC
#>   Treatment      Area     StdErr DF   CILower   CIUpper
#> 1 TrtTREAT1 0.8477499 0.02710939 99 0.7939590 0.9015408
#> 2 TrtTREAT2 0.8368951 0.02744860 99 0.7824311 0.8913591
```

6.5.6.3 ANOVA for FRRC analysis

- `ret$msAnovaEachRdrFRRC` is the mean-squares ANOVA for each reader

```
ret$msAnovaEachRdrFRRC
#>   Source DF  RdrREADER_1 RdrREADER_2 RdrREADER_3  RdrREADER_4
#> 1      T  1 0.0007389761  0.02307702  0.01476929 4.091217e-05
#> 2      C 99 0.2038747746  0.22344191  0.21424677 2.854199e-01
#> 3     TC 99 0.0915587344  0.08027926  0.06122898 6.057067e-02
```

6.5.6.4 Confidence Intervals for FRRC analysis

- `ciDiffTrtFRRC` is the CI for reader-averaged treatment differences, for fixed-reader random-case analysis

```
ret$ciDiffTrtFRRC
#>             Treatment   Estimate     StdErr DF         t    PrGTt     CILower
#> 1 TrtTREAT1-TrtTREAT2 0.01085482 0.01799277 99 0.6032876 0.547697 -0.02484675
#>      CIUpper
#> 1 0.04655638
```

6.5.7 Random-reader fixed-case (RRFC) analysis

6.5.7.1 F-statistic and p-value for RRFC analysis

- `ret$FTestStatsRRFC$fRRFC` is the F-statistic for for random-reader fixed-case analysis

```
ret$FTestStatsRRFC$fRRFC
#> [1] 4.694058
```

- `ret$FTestStatsRRFC$ddfRRFC` is the ddf for for random-reader fixed-case analysis

```
ret$FTestStatsRRFC$ddfRRFC
#> [1] 3
```

- `ret$FTestStatsRRFC$pRRFC` is the p-value for for random-reader fixed-case analysis

```
ret$FTestStatsRRFC$pRRFC
#> [1] 0.1188379
```

6.5.7.2 Confidence Intervals for RRFC analysis

- `ciDiffTrtRRFC` is the CI for reader-averaged inter-treatment FOM differences for random-reader fixed-case analysis

```
ret$ciDiffTrtRRFC
#>              Treatment   Estimate      StdErr DF        t     PrGTt      CILower
#> 1 TrtTREAT1-TrtTREAT2 0.01085482 0.005010122  3 2.166577 0.1188379 -0.005089627
#>      CIUpper
#> 1 0.02679926
```

- `ciAvgRdrEachTrtRRFC` is the CI for treatment FOMs for each reader for random-reader fixed-case analysis

```
ret$ciAvgRdrEachTrtRRFC
#>   Treatment      Area     StdErr DF   CILower   CIUpper
#> 1 TrtTREAT1 0.8477499 0.01109801  3 0.8124311 0.8830687
#> 2 TrtTREAT2 0.8368951 0.007777173  3 0.8121620 0.8616282
```

6.6 ORH significance testing

Simply change `method = "DBMH"` to `method = "ORH"`.

```
ret <- StSignificanceTesting(dataset03, FOM = "Wilcoxon", method = "ORH")
str(ret)
#> List of 14
#>  $ fomArray             : num [1:2, 1:4] 0.853 0.85 0.865 0.844 0.857 ...
#>   ..- attr(*, "dimnames")=List of 2
#>   .. ..$ : chr [1:2] "TrtTREAT1" "TrtTREAT2"
#>   .. ..$ : chr [1:4] "RdrREADER_1" "RdrREADER_2" "RdrREADER_3" "RdrREADER_4"
#>  $ meanSquares          :'data.frame':    1 obs. of  3 variables:
#>   ..$ msT : num 0.000236
#>   ..$ msR : num 0.000684
#>   ..$ msTR: num 5.02e-05
#>  $ varComp              :'data.frame':    1 obs. of  6 variables:
#>   ..$ varR : num 2.33e-05
#>   ..$ varTR: num -0.000684
#>   ..$ cov1 : num 0.000792
#>   ..$ cov2 : num 0.000484
#>   ..$ cov3 : num 0.000513
#>   ..$ var  : num 0.00153
#>  $ FTestStatsRRRC       :'data.frame':    1 obs. of  4 variables:
#>   ..$ fRRRC  : num 4.69
```

```
#>   ..$ ndfRRRC: num 1
#>   ..$ ddfRRRC: num 3
#>   ..$ pRRRC  : num 0.119
#>  $ ciDiffTrtRRRC       :'data.frame':    1 obs. of  8 variables:
#>   ..$ Treatment: chr "TrtTREAT1-TrtTREAT2"
#>   ..$ Estimate : num 0.0109
#>   ..$ StdErr   : num 0.00501
#>   ..$ DF       : num 3
#>   ..$ t        : num 2.17
#>   ..$ PrGTt    : num 0.119
#>   ..$ CILower  : num -0.00509
#>   ..$ CIUpper  : num 0.0268
#>  $ ciAvgRdrEachTrtRRRC :'data.frame':    2 obs. of  6 variables:
#>   ..$ Treatment: Factor w/ 2 levels "TrtTREAT1","TrtTREAT2": 1 2
#>   ..$ Area     : num [1:2] 0.848 0.837
#>   ..$ StdErr   : num [1:2] 0.0244 0.0236
#>   ..$ DF       : num [1:2] 70.1 253.6
#>   ..$ CILower  : num [1:2] 0.799 0.79
#>   ..$ CIUpper  : num [1:2] 0.896 0.883
#>  $ FTestStatsFRRC      :'data.frame':    1 obs. of  4 variables:
#>   ..$ fFRRC  : num 0.364
#>   ..$ ndfFRRC: num 1
#>   ..$ ddfFRRC: num Inf
#>   ..$ pFRRC  : num 0.546
#>  $ ciDiffTrtFRRC       :'data.frame':    1 obs. of  8 variables:
#>   ..$ Treatment: chr "TrtTREAT1-TrtTREAT2"
#>   ..$ Estimate : num 0.0109
#>   ..$ StdErr   : num 0.018
#>   ..$ DF       : num Inf
#>   ..$ t        : num 0.603
#>   ..$ PrGTt    : num 0.546
#>   ..$ CILower  : num -0.0244
#>   ..$ CIUpper  : num 0.0461
#>  $ ciAvgRdrEachTrtFRRC :'data.frame':    2 obs. of  6 variables:
#>   ..$ Treatment: Factor w/ 2 levels "TrtTREAT1","TrtTREAT2": 1 2
#>   ..$ Area     : num [1:2] 0.848 0.837
#>   ..$ StdErr   : num [1:2] 0.0271 0.0274
#>   ..$ DF       : num [1:2] Inf Inf
#>   ..$ CILower  : num [1:2] 0.795 0.783
#>   ..$ CIUpper  : num [1:2] 0.901 0.891
#>  $ ciDiffTrtEachRdrFRRC:'data.frame':    4 obs. of  9 variables:
#>   ..$ Reader   : Factor w/ 4 levels "RdrREADER_1",..: 1 2 3 4
#>   ..$ Treatment: Factor w/ 1 level "TrtTREAT1-TrtTREAT2": 1 1 1 1
#>   ..$ Estimate : num [1:4] 0.003844 0.021483 0.017187 0.000905
#>   ..$ StdErr   : num [1:4] 0.0428 0.0401 0.035 0.0348
```

```
#>   ..$ DF       : num [1:4] Inf Inf Inf Inf
#>   ..$ t        : num [1:4] 0.0898 0.5362 0.4911 0.026
#>   ..$ PrGTt    : num [1:4] 0.928 0.592 0.623 0.979
#>   ..$ CILower  : num [1:4] -0.08 -0.0571 -0.0514 -0.0673
#>   ..$ CIUpper  : num [1:4] 0.0877 0.1 0.0858 0.0691
#>  $ varCovEachRdr       :'data.frame':    4 obs. of  3 variables:
#>   ..$ Reader: Factor w/ 4 levels "RdrREADER_1",..: 1 2 3 4
#>   ..$ Var   : num [1:4] 0.00148 0.00152 0.00138 0.00173
#>   ..$ Cov1  : num [1:4] 0.000562 0.000716 0.000765 0.001124
#>  $ FTestStatsRRFC      :'data.frame':    1 obs. of  4 variables:
#>   ..$ fRRFC  : num 4.69
#>   ..$ ndfRRFC: num 1
#>   ..$ ddfRRFC: num 3
#>   ..$ pRRFC  : num 0.119
#>  $ ciDiffTrtRRFC       :'data.frame':    1 obs. of  8 variables:
#>   ..$ Treatment: chr "TrtTREAT1-TrtTREAT2"
#>   ..$ Estimate : num 0.0109
#>   ..$ StdErr   : num 0.00501
#>   ..$ DF       : num 3
#>   ..$ t        : num 2.17
#>   ..$ PrGTt    : num 0.119
#>   ..$ CILower  : num -0.00509
#>   ..$ CIUpper  : num 0.0268
#>  $ ciAvgRdrEachTrtRRFC :'data.frame':    2 obs. of  6 variables:
#>   ..$ Treatment: Factor w/ 2 levels "TrtTREAT1","TrtTREAT2": 1 2
#>   ..$ Area     : num [1:2] 0.848 0.837
#>   ..$ StdErr   : num [1:2] 0.0111 0.00777
#>   ..$ DF       : num [1:2] 3 3
#>   ..$ CILower  : num [1:2] 0.812 0.812
#>   ..$ CIUpper  : num [1:2] 0.883 0.862
```

6.7 References

Chapter 7

QUICK START DBM2

7.1 Introduction

This vignette illustrates significance testing using the DBMH method. But, instead of the unwieldy output in *QuickStartDBMH.html*, it generates an Excel output file containing the following worksheets:

- `Summary`
- `FOMs`
- `RRRC`
- `FRRC`
- `RRFC`
- `ANOVA`

7.2 Generating the Excel output file

This illustrates the `UtilOutputReport()` function. The significance testing `method` is "DBMH", the default, and the figure of merit `FOM` is "Wilcoxon". Note `ReportFileExt` = "xlsx"‘ telling the function to create an Excel output file. The Excel output is created in a temporary file.

```
ret <- UtilOutputReport(dataset03, FOM = "Wilcoxon", overWrite = TRUE, ReportFileExt = "xlsx")
#>
#> Output file name is:      /var/folders/d1/mx6dcbzx3v39r260458z2b200000gn/T//RtmpbsPYtf/RJafroc
```

7.3 ORH significance testing

Simply change `method = "DBMH"` (the default) to `method = "ORH"`.

```
ret <- UtilOutputReport(dataset03, FOM = "Wilcoxon", method = "ORH", overWrite = TRUE,
#>
#> Output file name is:      /var/folders/d1/mx6dcbzx3v39r260458z2b200000gn/T//RtmpbsP
```

Chapter 8

BACKGROUND ON THE F-DISTRIBUTION

8.1 Introduction

Since it plays an important role in sample size estimation, it is helpful to examine the behavior of the F-distribution. In the following `ndf` = numerator degrees of freedom, `ddf` = denominator degrees of freedom and `ncp` = non-centrality parameter (i.e., the Δ appearing in Eqn. (11.6) of (Chakraborty, 2017)).

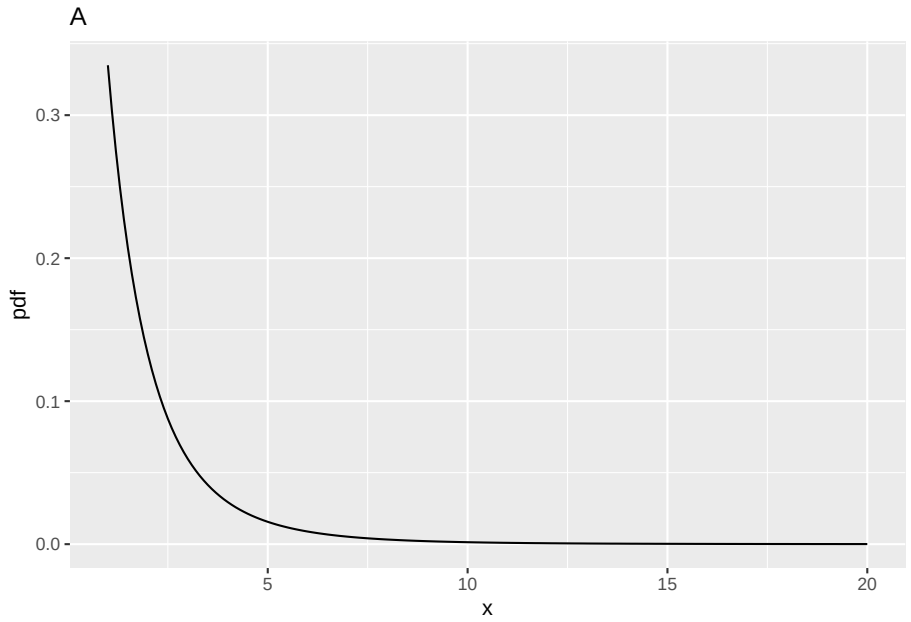
The use of three `R` functions is demonstrated.

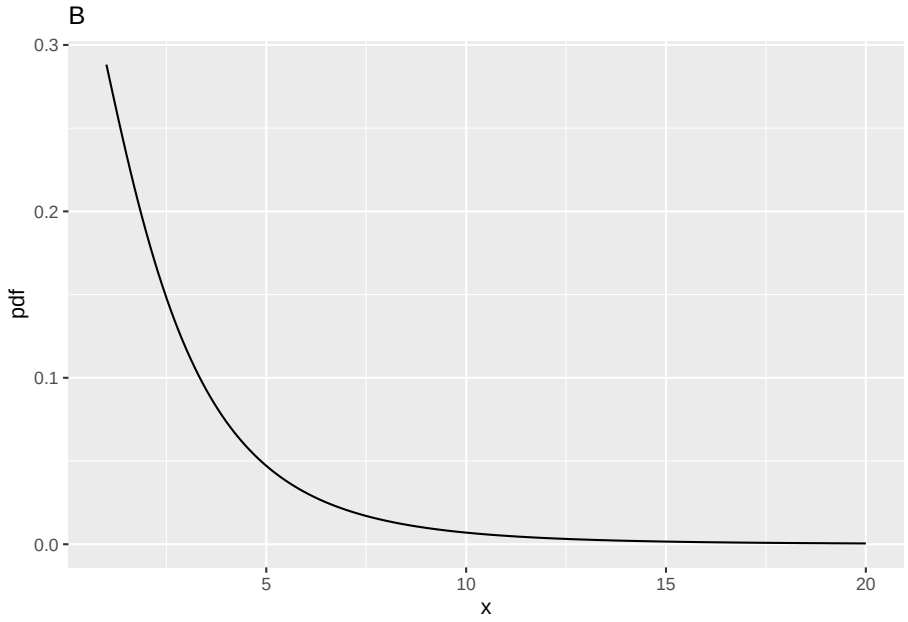
- `qf(p,ndf,ddf)` is the *quantile* function of the F-distribution for specified values of `p`, `ndf` and `ddf`, i.e., the value `x` such that fraction `p` of the area under the F-distribution lies to the right of `x`. Since `ncp` is not included as a parameter, the default value, i.e., zero, is used. This is called the *central* F-distribution.
- `df(x,ndf,ddf,ncp)` is the probability density function (*pdf*) of the F-distribution, as a function of `x`, for specified values of `ndf`, `ddf` and `ncp`.
- `pf(x,ndf,ddf,ncp)` is the probability (or cumulative) distribution function of the F-distribution for specified values of `ndf`, `ddf` and `ncp`.

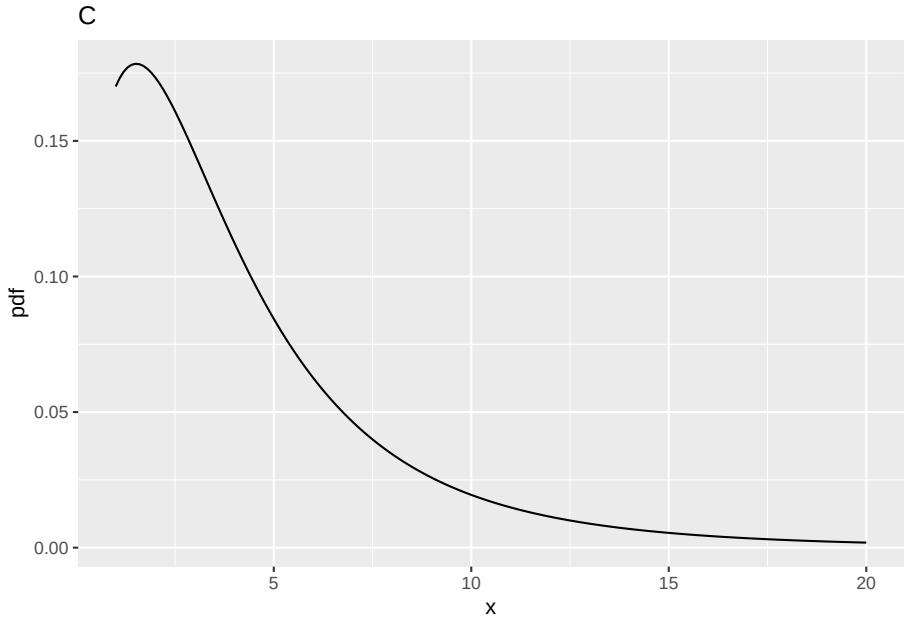
8.2 Effect of `ncp` for `ndf` = 2 and `ddf` = 10

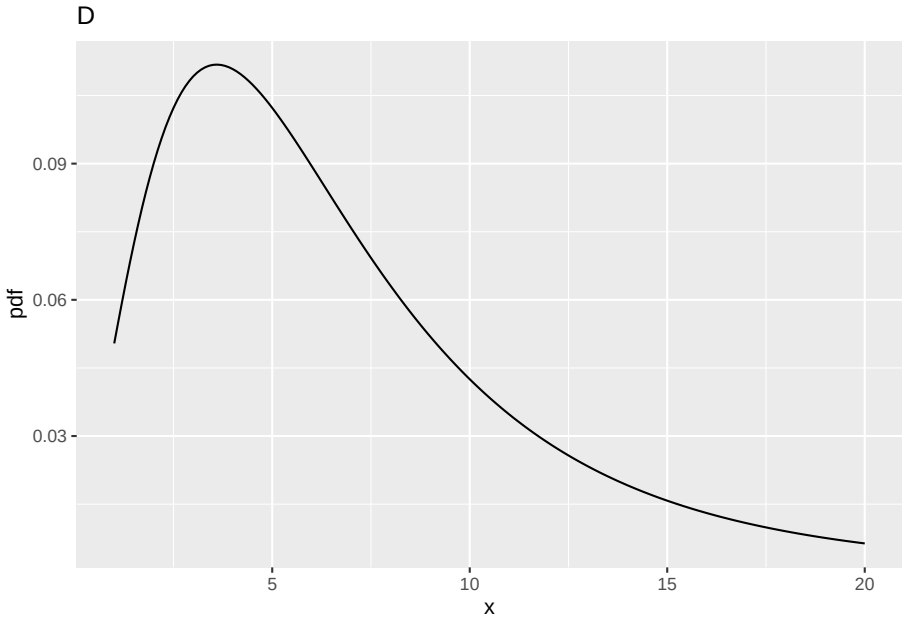
- Four values of `ncp` are considered (0, 2, 5, 10) for `ddf` = 10.
- `fCrit` is the critical value of the F distribution, i.e., that value such that fraction α of the area is to the right of the critical value.

```
ndf <- 2;ddf <- 10;ncp <- c(0,2,5,10)
alpha <- 0.05
fCrit <- qf(1-alpha, ndf,ddf)
x <- seq(1, 20, 0.1)
myLabel <- c("A", "B", "C", "D")
myLabelIndx <- 1
pFgtFCrit <- NULL
for (i in 1:length(ncp))
{
  y <- df(x,ndf,ddf,ncp=ncp[i])
  pFgtFCrit <- c(pFgtFCrit, 1-pf(fCrit, ndf, ddf, ncp = ncp[i]))
}
for (i in 1:length(ncp))
{
  y <- df(x,ndf,ddf,ncp=ncp[i])
  curveData <- data.frame(x = x, pdf = y)
  curvePlot <- ggplot(data = curveData, mapping = aes(x = x, y = pdf)) +
    geom_line() +
    ggtitle(myLabel[myLabelIndx]);myLabelIndx <- myLabelIndx + 1
  print(curvePlot)
}
fCrit_2_10 <- fCrit # convention fCrit_ndf_ddf
```




B
0.3
0.2
0.1
0.0
pdf
5
10
15
20
x


C
0.15
0.10
0.05
0.00
pdf
5
10
15
20
x



	ndf	ddf	fCrit	ncp	pFgtFCrit
A	2	10	4.102821	0	0.0500000
B	2	10	4.102821	2	0.1775840
C	2	10	4.102821	5	0.3876841
D	2	10	4.102821	10	0.6769776

8.3 Comments

8.3.1 Fig. A

- This corresponds to `ncp = 0`, i.e., the *central* F-distribution.
- The integral under this distribution is unity (this is also true for all plots in this vignette).
- The critical value, `fCrit` in the above code block, is the value of `x` such that the probability of exceeding `x` is α. The corresponding parameter `alpha` is defined above as 0.05.
- In the current example `fCrit` = 4.102821. Notice the use of the quantile function `qf()` to determine this value, and the default value of `ncp`, namely zero, is used; specifically, one does not pass a 4th argument to `qf()`.
- **The decision rule for rejecting the NH uses the NH distribution of the F-statistic**, i.e., reject the NH if F >= `fCrit`. As expected, `prob > fCrit` = 0.05 because this is how `fCrit` was defined.

8.3.2 Fig. B

- This corresponds to `ncp = 2`, `ndf` = 2 and `ddf` = 10.
- The distribution is slightly shifted to the right as compared to Fig. A, thereby making it more likely that the observed value of the F-statistic will exceed the critical value determined for the NH distribution.
- In fact, `prob > fCrit` = 0.177584, i.e., the *statistical power* (compare this to Fig. A where `prob > fCrit` was 0.05).

8.3.3 Fig. C

- This corresponds to `ncp = 5`, `ndf` = 2 and `ddf` = 10.
- Now `prob > fCrit` = 0.3876841.
- Power has increased compared to Fig. B.

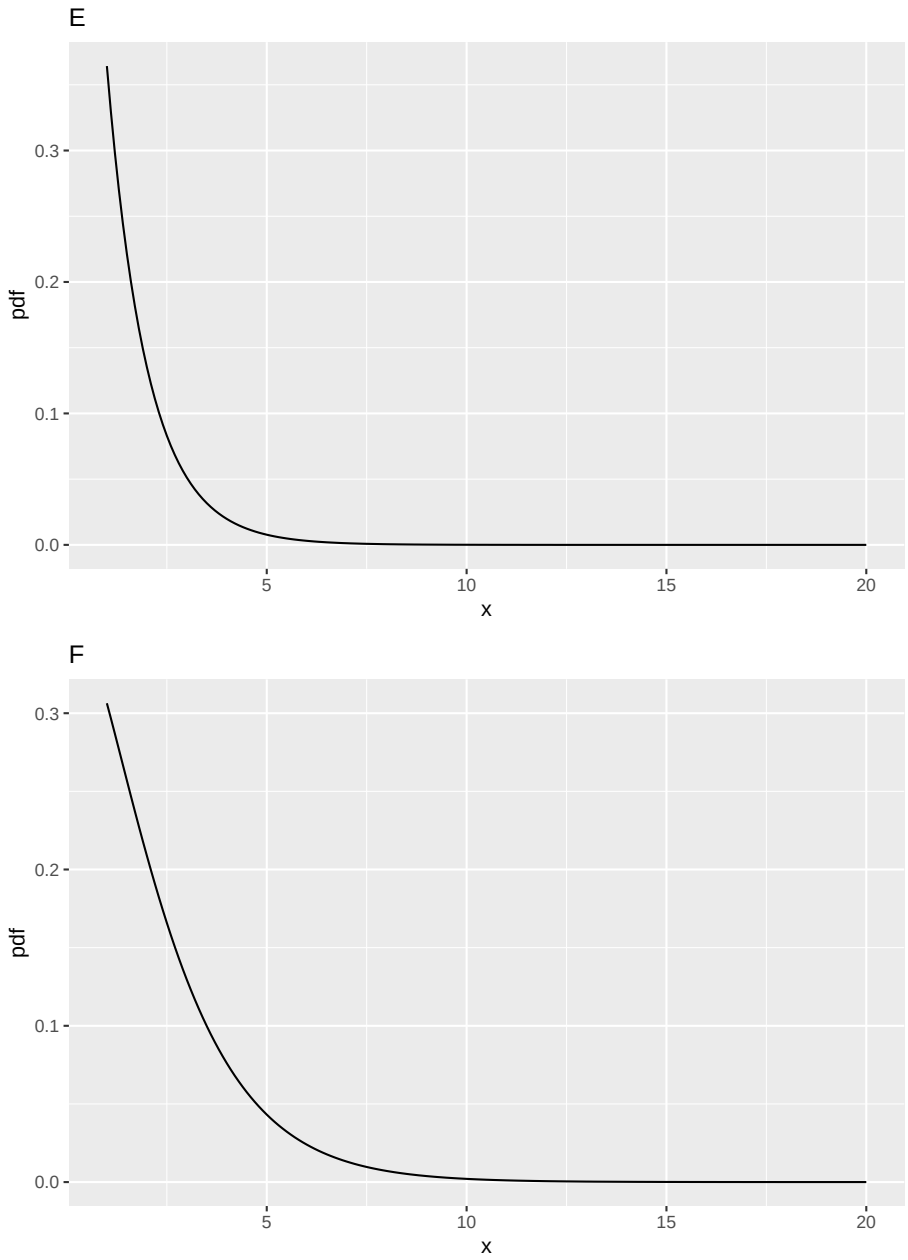
8.3.4 Fig. D

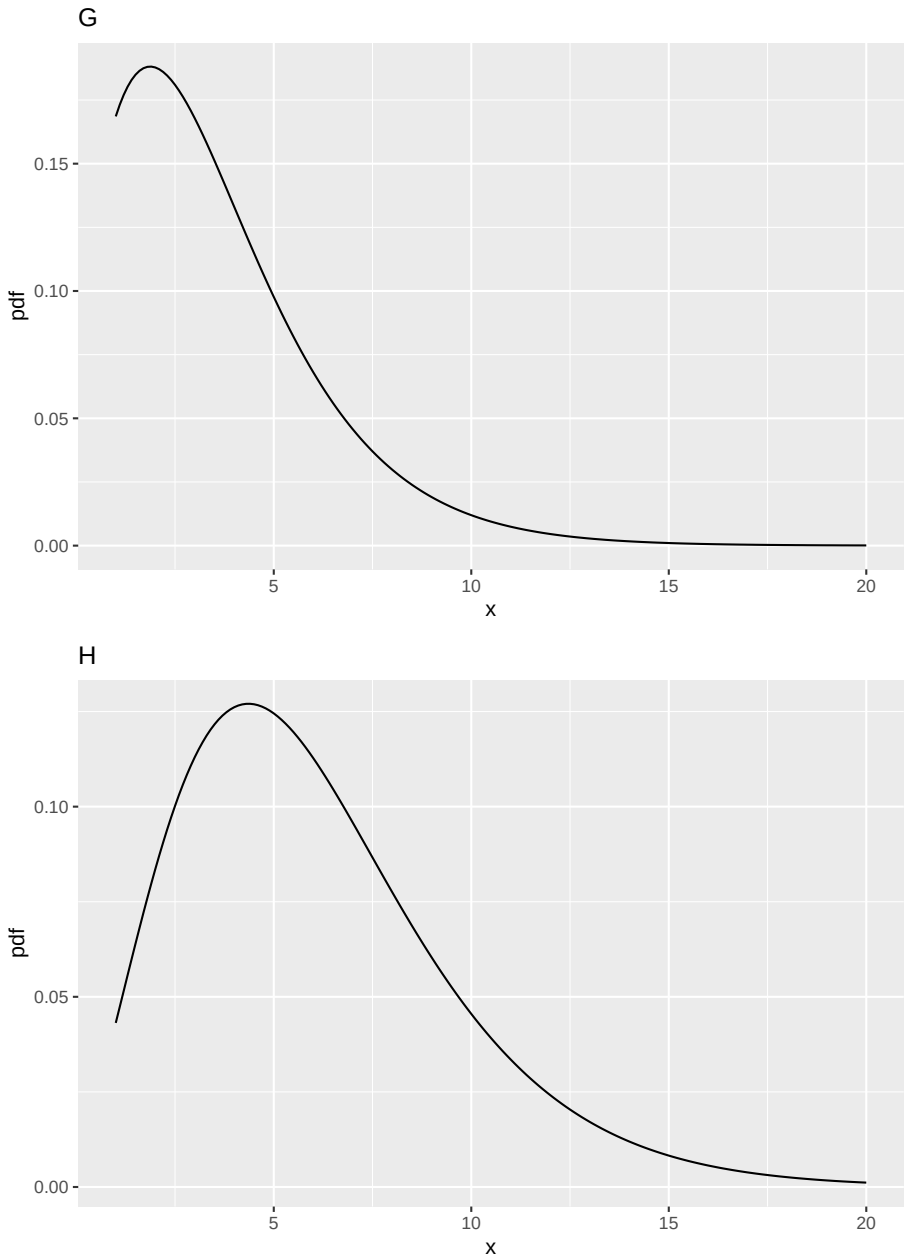
- This corresponds to `ncp = 10`, `ndf` = 2 and `ddf` = 10.
- Now `prob > fCrit` is 0.6769776.
- Power has increased compared to Fig. C.
- The effect of the shift is most obvious in Fig. C and Fig. D.
- Considering a vertical line at `x` = 4.102821, fraction 0.6769776 of the probability distribution in Fig. D lies to the right of this line
- Therefore the NH is likely to be rejected with probability 0.6769776.

8.3.5 Summary

The larger that non-centrality parameter, the greater the shift to the right of the F-distribution, and the greater the statistical power.

8.4 Effect of `ncp` for `ndf` = 2 and `ddf` = 100




G
0.15
0.10
0.05
0.00
pdf
5
10
15
20
x
H
0.10
0.05
0.00
pdf
5
10
15
20
x

	ndf	ddf	fCrit	ncp	pFgtFCrit
A	2	10	4.102821	0	0.0500000
B	2	10	4.102821	2	0.1775840
C	2	10	4.102821	5	0.3876841
D	2	10	4.102821	10	0.6769776
E	2	100	3.087296	0	0.0500000
F	2	100	3.087296	2	0.2199264
G	2	100	3.087296	5	0.4910802
H	2	100	3.087296	10	0.8029764

8.5 Comments

- All comparisons in this sections are at the same values of `ncp` defined above.
- And between `ddf` = 100 and `ddf` = 10.

8.5.1 Fig. E

- This corresponds to `ncp` = 0, `ndf` = 2 and `ddf` = 100.
- The critical value is `fCrit_2_100` = 3.0872959. Notice the decrease compared to the previous value for `ncp` = 0, i.e., 4.102821, for `ddf` = 10.
- One expects that increasing `ddf` will make it more likely that the NH will be rejected, and this is confirmed below.
- All else equal, statistical power increases with increasing `ddf`.

8.5.2 Fig. F

- This corresponds to `ncp` = 2, `ndf` = 2 and `ddf` = 100.
- The probability of exceeding the critical value is `prob > fCrit_2_100` = 0.2199264, greater than the previous value, i.e., 0.177584 for `ddf` = 10.

8.5.3 Fig. G

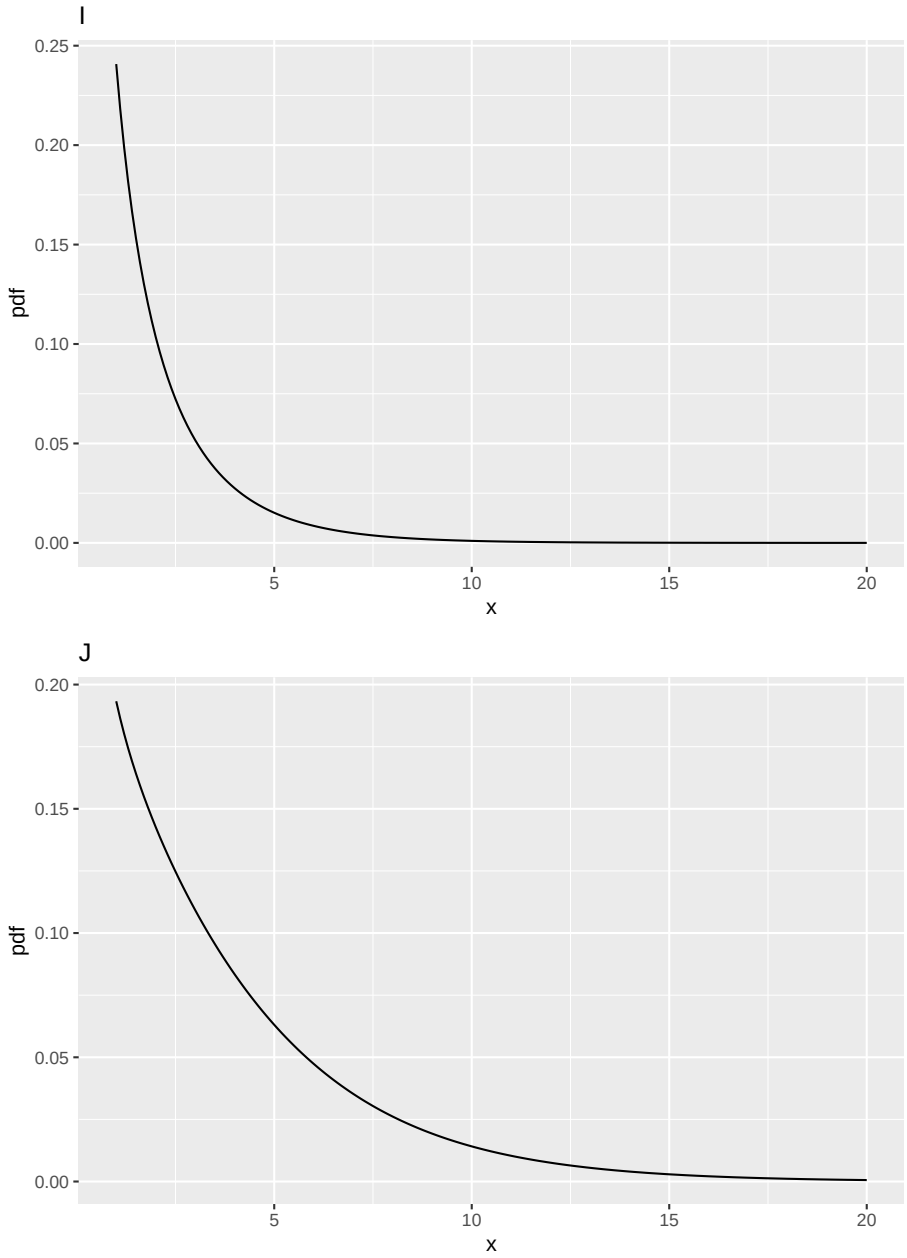
- This corresponds to `ncp = 5`, `ndf` = 2 and `ddf` = 100.
- The probability of exceeding the critical value is `prob > fCrit_2_100` = 0.4910802.
- This is greater than the previous value, i.e., 0.3876841 for `ddf` = 10.

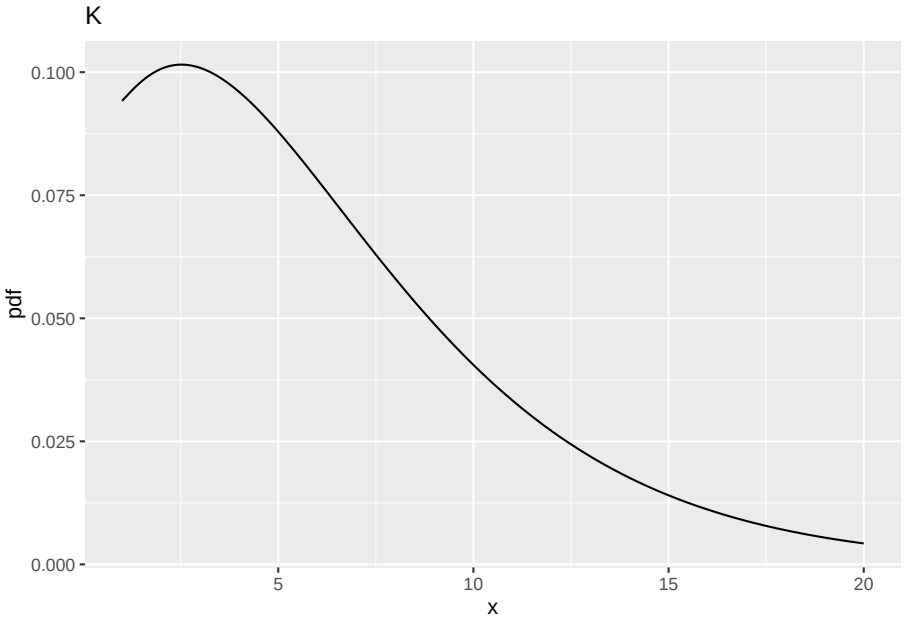
8.5.4 Fig. H

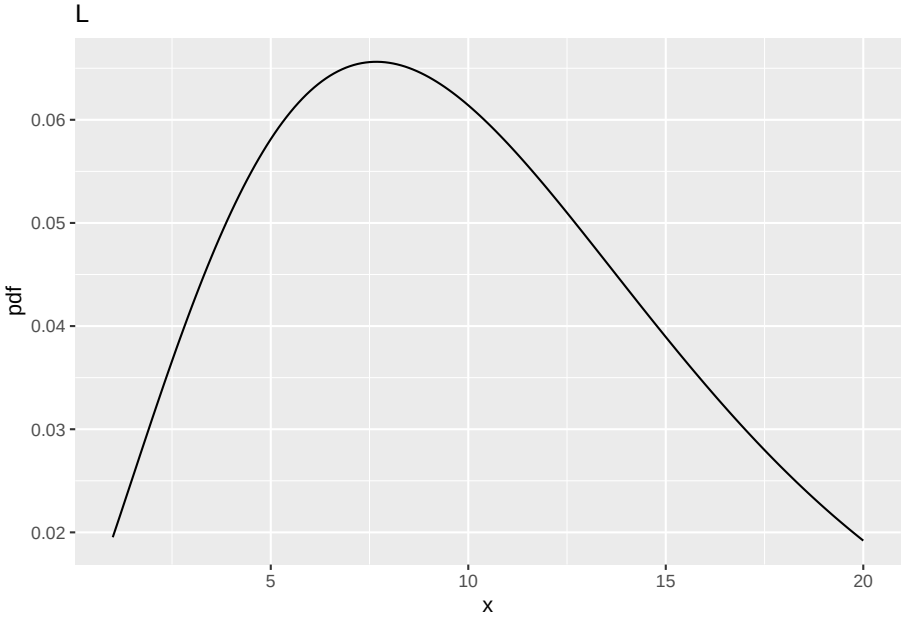
- This corresponds to `ncp = 10`, `ndf` = 2 and `ddf` = 100.

- The probability of exceeding the critical value is `prob > fCrit_2_100` is 0.8029764.
- This is greater than the previous value, i.e., 0.6769776 for `ddf` = 10.

8.6 Effect of `ncp` for `ndf` = 1, `ddf` = 100




K
0.100
0.075
0.050
0.025
0.000
pdf
5
10
15
20
x


L
0.06
0.05
0.04
0.03
0.02
pdf
5
10
15
20
x

	ndf	ddf	fCrit	ncp	pFgtFCrit
A	2	10	4.102821	0	0.0500000
B	2	10	4.102821	2	0.1775840
C	2	10	4.102821	5	0.3876841
D	2	10	4.102821	10	0.6769776
E	2	100	3.087296	0	0.0500000
F	2	100	3.087296	2	0.2199264
G	2	100	3.087296	5	0.4910802
H	2	100	3.087296	10	0.8029764
I	1	100	3.936143	0	0.0500000
J	1	100	3.936143	2	0.2883607
K	1	100	3.936143	5	0.6004962
L	1	100	3.936143	10	0.8793619

8.7 Comments

- All comparisons in this sections are at the same values of `ncp` defined above and at `ddf` = 100.
- And between `ndf` = 1 and `ndf` = 2.

8.7.1 Fig. I

- This corresponds to `ncp` = 0, `ndf` = 1 and `ddf` = 100.
- The critical value is `fCrit_1_100` = 3.936143.
- Notice the increase in the critical value as compared to the corresponding value for `ndf = 2`, i.e., 3.0872959.
- One might expect power to decrease, **but see below**.

8.7.2 Fig. J

- This corresponds to `ncp` = 2, `ndf` = 1 and `ddf` = 100.
- Now `prob > fCrit_1_100` = 0.2883607, larger than the previous value 0.2199264.
- The power has actually increased.

8.7.3 Fig. K

- This corresponds to `ncp` = 5, `ndf` = 1 and `ddf` = 100‘’,
- Now `prob > fCrit_1_100` = 0.6004962, larger than the previous value 0.4910802.
- Again, the power has actually increased.

8.7.4 Fig. L

- This corresponds to `ncp` = 10, `ndf` = 1 and `ddf` = 100
- Now `prob > fCrit_1_100` is 0.8793619, larger than the previous value 0.8029764.
- The power has actually increased.

8.8 Summary

- Power increases with increasing `ddf` and `ncp`.
- The effect of increasing `ncp` is quite dramatic. This is because power depends on the square of `ncp`.
- Decreasing `ndf` also **increases** power. At first glance this may seem counterintuitive, as `fCrit` has gone up, but is explained by the differing shapes of the two distributions: the pdf is broader for `ndf` = 1 as compared to `ndf` = 2 (compare Fig. L to H).

8.9 References

Chapter 9

ROC-DBMH sample size from first principles

9.1 Introduction

The starting point is a **pilot** study. The variability in this dataset (specifically the variance components, subsequently converted to mean squares), obtained by running the significance testing function `StSignificanceTesting()`, is used to extrapolate to the necessary numbers of readers and cases, in the **pivotal** study, to achieve the desired power. In this example, the observed effect size in the pilot study is used as the anticipated effect size for the pivotal study – this is generally not a good idea as discussed in **Chapter 11** under "Cautionary notes". Shown below, and the reader should confirm, is a first principles implementation of the relevant formulae in **Chapter 11**.

9.2 Sample size estimation using the DBMH method

The Van Dyke dataset in file `VanDyke.lrc`, in `"MRMC"` format, is regarded as a pilot study. The command `rocData <- DfReadDataFile(fileName, format = "MRMC")` reads the data and saves it to a `dataset` object `rocData`. For more on data formats click here. The next line uses the function `StSignificanceTesting()` to apply `method = "DBMH"` analysis, the default, using the `FOM = "Wilcoxon"` figure of merit. The next line extracts the variance components `varYTR`, `varYTC` and `varYEps` (the Y's denote pseudovalue based values). The next line extracts the effect size.

```
alpha <- 0.05
rocData <- dataset02 ##"VanDyke.lrc"
#fileName <- dataset03 ## "Franken1.lrc"
retDbm <- StSignificanceTesting(dataset = rocData, FOM = "Wilcoxon", method = "DBMH")
varYTR <- retDbm$varComp$varTR;varYTC <- retDbm$varComp$varTC;varYEps <- retDbm$varCo
effectSize <- retDbm$ciDiffTrtRRRC$Estimate
```

The *observed* effect size is `effectSize` = -0.0438003, which, in this example, is used as the *anticipated* effect size, generally not a good idea. **See Chapter 11 for nuances regarding the choice of this all important value.** The following code snippet reveals the names and array indexing of the pseudovalue variance components.

```
retDbm$varComp
#>          varR       varC        varTR     varTC      varRC     varErr
#> 1 0.001534999 0.02724923 0.0002004025 0.0119753 0.01226473 0.0399716
```

For example, the treatment-reader pseudovalue variance component is the third element of `retDbm$varComp`.

9.2.1 Random reader random case (RRRC)

This illustrates random reader random case sample size estimation. Assumed are 10 readers and 163 cases in the pivotal study. The non-centrality parameter is defined by:

TBA

The sampling distribution of the F-statistic under the AH is:

TBA

Also,

TBA

where `d` is the observed effect size, i.e., `effectSize`. The formulae for calculating the mean-squares are in (Hillis and Berbaum, 2004), implemented in `UtilMeanSquares()`.

```
#RRRC
J <- 10;K <- 163
ncp <- (0.5*J*K*(effectSize)^2)/(K*varYTR+max(J*varYTC,0)+varYEps)
MS <- UtilMeanSquares(rocData, FOM = "Wilcoxon", method = "DBMH")
ddf <- (MS$msTR+max(MS$msTC-MS$msTRC,0))^2/(MS$msTR^2)*(J-1)
FCrit <- qf(1 - alpha, 1, ddf)
Power1 <- 1-pf(FCrit, 1, ddf, ncp = ncp)
```

The next line calculates the non centrality parameter, `ncp` = 8.1269825. Note that `effectSize` enters as the **square**. The `UtilMeanSquares()` function returns the mean-squares as a `list` (ignore the last two rows of output for now).

```
str(MS)
#> List of 9
#>  $ msT        : num 0.547
#>  $ msR        : num 0.437
#>  $ msC        : num 0.397
#>  $ msTR       : num 0.0628
#>  $ msTC       : num 0.0521
#>  $ msRC       : num 0.0645
#>  $ msTRC      : num 0.04
#>  $ msCSingleT: num [1:2] 0.336 0.16
#>  $ msCSingleR: num [1:5] 0.1222 0.2127 0.1365 0.0173 0.1661
```

The next line calculates `ddf` = 12.822129. The remaining lines calculate the critical value of the F-distribution, `FCrit` = 4.680382 and statistical power = 0.7494133, which by design is close to 80%, i.e., the numbers of readers and cases were chosen to achieve this value.

9.2.2 Fixed reader random case (FRRC)

This code illustrates fixed reader random case sample size estimation. Assumed are 10 readers and 133 cases in the pivotal study. The formulae are:

TBA

The sampling distribution of the F-statistic under the AH is:

TBA

```
#FRRC
ncp <- (0.5*J*K*(effectSize)^2)/(max(J*varYTC,0)+varYEps)
ddf <- (K-1)
FCrit <- qf(1 - alpha, 1, ddf)
Power2 <- 1-pf(FCrit, 1, ddf, ncp = ncp)
```

This time non centrality parameter, `ncp` = 7.9873835, `ddf` = 132, `FCrit` = 3.912875 and statistical power = 0.8011167. Again, be design, this is close to 80%. Note that when readers are regarded as a fixed effect, fewer cases are needed to achieve the desired power. Freezing out a source of variability results in a more stable measurement and hence fewer cases are needed to achieve the desired power.

9.2.3 Random reader fixed case (RRFC)

This code illustrates random reader random case sample size estimation. Assumed are 10 readers and 53 cases in the pivotal study. The formulae are:

TBA

The sampling distribution of the F-statistic under the AH is:

TBA

```
#RRFC
ncp <- (0.5*J*K*(effectSize)^2)/(K*varYTR+varYEps)
ddf <- (J-1)
FCrit <- qf(1 - alpha, 1, ddf)
Power3 <- 1-pf(FCrit, 1, ddf, ncp = ncp)
```

This time non centrality parameter, `ncp` = 10.0487164, `ddf` = 9, `FCrit` = 5.117355 and statistical power = 0.8049666. Again, be design, this is close to 80%.

9.3 Summary

For 10 readers, the numbers of cases needed for 80% power is largest (163) for RRRC, intermediate (133) for FRRC and least for RRFC (53). For all three analyses, the expectation of 80% power is met.

9.4 References

Chapter 10

ROC-DBMH sample size using RJafroc

10.1 Introduction

This illustrates the `RJafroc` implementation of sample-size estimation. Default α is 0.05 and default power (1-β) is 0.8. Three functions are provided. Each of these functions can be used with `method "DBMH"` (illustrated here, the default) or `method = "ORH"` (next vignette). Illustrated below, for the most part, is the random-reader random-case (RRRC) option, i.e., `option = "RRRC"`. The last two examples illustrate fixed-reader random-case (FRRC) `option = "FRRC"` and random-reader fixed-case (RRFC) `option = "RRFC"` options.

- `SsPowerGivenJK()` Statistical power for specified numbers of readers and cases in an ROC study.
- `SsPowerTable()` Generate a power table, i.e., combinations of numbers of readers and cases yielding the desired power.
- `SsSampleSizeKGivenJ` Number of cases, for specified number of readers, to achieve desired power.

10.2 Illustration of `SsPowerGivenJK()` using `method = "DBMH"`

The selected dataset corresponds to the Van Dyke data.

```
power <- SsPowerGivenJK(dataset02, FOM = "Wilcoxon", J = 6, K = 112, option = "RRRC")
```

The returned value is a list containing the expected power `power`, the non-centrality parameter `ncp`, the denominator degrees of freedom `ddf` and the F-statistic `f`. The numerator degrees of freedom `ndf` is always `I - 1`, i.e., unity for this dataset.

```
str(power)
#> 'data.frame':    1 obs. of  4 variables:
#>  $ powerRRRC: num 0.556
#>  $ ncpRRRC  : num 4.8
#>  $ ddfHRRRC : num 23.1
#>  $ fRRRC    : num 4.28
```

Expected power is 0.5555789.

10.3 Illustration of `SsPowerTable()` using `method = "DBMH"`

```
powTab <- SsPowerTable(dataset02, FOM = "Wilcoxon", method = "DBMH", option = "RRRC")
```

Now show the power table `powTab`. Note that the last column is always close to 0.8, the desired power. The 2nd and 3rd columns show the number of readers and number of cases to achieve the desired power.

```
powTab
#>    numReaders numCases power
#> 1           3    >2000  <NA>
#> 2           3    >2000  <NA>
#> 3           4     1089   0.8
#> 4           4     1089   0.8
#> 5           5      344 0.801
#> 6           5      344 0.801
#> 7           6      251 0.801
#> 8           6      251 0.801
#> 9           7      211 0.801
#> 10          7      211 0.801
#> 11          8      188 0.801
#> 12          8      188 0.801
#> 13          9      173 0.801
#> 14          9      173 0.801
```

```
#> 15          10      163 0.802
#> 16          10      163 0.802
#> 17          11      155 0.801
#> 18          11      155 0.801
#> 19          12      149 0.802
#> 20          12      149 0.802
#> 21          13      144 0.801
#> 22          13      144 0.801
#> 23          14      140 0.802
#> 24          14      140 0.802
#> 25          15      137 0.802
#> 26          15      137 0.802
#> 27          16      134 0.802
#> 28          16      134 0.802
#> 29          17      131 0.801
#> 30          17      131 0.801
#> 31          18      129 0.801
#> 32          18      129 0.801
#> 33          19      127 0.801
#> 34          19      127 0.801
#> 35          20      126 0.802
#> 36          20      126 0.802
#> 37          21      124 0.801
#> 38          21      124 0.801
#> 39          22      123 0.802
#> 40          22      123 0.802
#> 41          23      122 0.802
#> 42          23      122 0.802
#> 43          24      121 0.803
#> 44          24      121 0.803
#> 45          25      120 0.802
#> 46          25      120 0.802
#> 47          26      119 0.802
#> 48          26      119 0.802
#> 49          27      118 0.802
#> 50          27      118 0.802
#> 51          28      117 0.801
#> 52          28      117 0.801
#> 53          29      117 0.803
#> 54          29      117 0.803
#> 55          30      116 0.802
#> 56          30      116 0.802
#> 57          31      115 0.801
#> 58          31      115 0.801
#> 59          32      115 0.803
```

```
#> 60          32      115 0.803
#> 61          33      114 0.801
#> 62          33      114 0.801
#> 63          34      114 0.803
#> 64          34      114 0.803
#> 65          35      113 0.801
#> 66          35      113 0.801
#> 67          36      113 0.802
#> 68          36      113 0.802
#> 69          37      112   0.8
#> 70          37      112   0.8
#> 71          38      112 0.802
#> 72          38      112 0.802
#> 73          39      112 0.803
#> 74          39      112 0.803
#> 75          40      111 0.801
#> 76          40      111 0.801
#> 77          41      111 0.802
#> 78          41      111 0.802
#> 79          42      111 0.803
#> 80          42      111 0.803
#> 81          43      110 0.801
#> 82          43      110 0.801
#> 83          44      110 0.802
#> 84          44      110 0.802
#> 85          45      110 0.802
#> 86          45      110 0.802
#> 87          46      110 0.803
#> 88          46      110 0.803
#> 89          47      109 0.801
#> 90          47      109 0.801
#> 91          48      109 0.802
#> 92          48      109 0.802
#> 93          49      109 0.802
#> 94          49      109 0.802
#> 95          50      109 0.803
#> 96          50      109 0.803
#> 97          51      108   0.8
#> 98          51      108   0.8
#> 99          52      108 0.801
#> 100         52      108 0.801
#> 101         53      108 0.802
#> 102         53      108 0.802
#> 103         54      108 0.802
#> 104         54      108 0.802
```

```
#> 105          55      108 0.803
#> 106          55      108 0.803
#> 107          56      107   0.8
#> 108          56      107   0.8
#> 109          57      107 0.801
#> 110          57      107 0.801
#> 111          58      107 0.801
#> 112          58      107 0.801
#> 113          59      107 0.802
#> 114          59      107 0.802
#> 115          60      107 0.802
#> 116          60      107 0.802
#> 117          61      107 0.803
#> 118          61      107 0.803
#> 119          62      107 0.803
#> 120          62      107 0.803
#> 121          63      106   0.8
#> 122          63      106   0.8
#> 123          64      106 0.801
#> 124          64      106 0.801
#> 125          65      106 0.801
#> 126          65      106 0.801
#> 127          66      106 0.802
#> 128          66      106 0.802
#> 129          67      106 0.802
#> 130          67      106 0.802
#> 131          68      106 0.802
#> 132          68      106 0.802
#> 133          69      106 0.803
#> 134          69      106 0.803
#> 135          70      106 0.803
#> 136          70      106 0.803
#> 137          71      106 0.804
#> 138          71      106 0.804
#> 139          72      105   0.8
#> 140          72      105   0.8
#> 141          73      105 0.801
#> 142          73      105 0.801
#> 143          74      105 0.801
#> 144          74      105 0.801
#> 145          75      105 0.801
#> 146          75      105 0.801
#> 147          76      105 0.802
#> 148          76      105 0.802
#> 149          77      105 0.802
```

```
#> 150         77      105 0.802
#> 151         78      105 0.802
#> 152         78      105 0.802
#> 153         79      105 0.803
#> 154         79      105 0.803
#> 155         80      105 0.803
#> 156         80      105 0.803
#> 157         81      105 0.803
#> 158         81      105 0.803
#> 159         82      105 0.803
#> 160         82      105 0.803
#> 161         83      104   0.8
#> 162         83      104   0.8
#> 163         84      104   0.8
#> 164         84      104   0.8
#> 165         85      104 0.801
#> 166         85      104 0.801
#> 167         86      104 0.801
#> 168         86      104 0.801
#> 169         87      104 0.801
#> 170         87      104 0.801
#> 171         88      104 0.801
#> 172         88      104 0.801
#> 173         89      104 0.802
#> 174         89      104 0.802
#> 175         90      104 0.802
#> 176         90      104 0.802
#> 177         91      104 0.802
#> 178         91      104 0.802
#> 179         92      104 0.802
#> 180         92      104 0.802
#> 181         93      104 0.802
#> 182         93      104 0.802
#> 183         94      104 0.803
#> 184         94      104 0.803
#> 185         95      104 0.803
#> 186         95      104 0.803
#> 187         96      104 0.803
#> 188         96      104 0.803
#> 189         97      104 0.803
#> 190         97      104 0.803
#> 191         98      104 0.804
#> 192         98      104 0.804
#> 193         99      104 0.804
#> 194         99      104 0.804
```

```
#> 195         100      103   0.8
#> 196         100      103   0.8
```

10.4 Illustration of SsSampleSizeKGivenJ() using method = "DBMH"

This function illustrates how the number of cases for 10 readers, used in Vignette 2, were chosen. In all but one example the default value of the desiredPower argument is used, namely 0.8 (if the argument is absent, its default value is used).

10.4.1 RRRC

```
ncases <- SsSampleSizeKGivenJ(dataset02, FOM = "Wilcoxon", J = 10, method = "DBMH", option = "RRR
str(ncases)
#> 'data.frame':    1 obs. of  2 variables:
#>  $ KRRRC    : num 163
#>  $ powerRRRC: num 0.802
```

ncases is a list containing the number of cases 163 and expected power 0.8015625. Compare the number of cases to the RRRC value used in vignette 2.

10.4.1.1 Non default value of desiredPower

This is illustrated below for 90% desired power.

```
ncases <- SsSampleSizeKGivenJ(dataset02, FOM = "Wilcoxon", J = 10, method = "DBMH", option = "RRR
str(ncases)
#> 'data.frame':    1 obs. of  2 variables:
#>  $ KRRRC    : num 236
#>  $ powerRRRC: num 0.9
```

The required number of cases is 236 and expected power is 0.9003501.

10.4.2 FRRC

```
ncases <- SsSampleSizeKGivenJ(dataset02, FOM = "Wilcoxon", J = 10, method = "DBMH", opt
str(ncases)
#> 'data.frame':    1 obs. of  2 variables:
#>  $ KFRRC    : num 133
#>  $ powerFRRC: num 0.801
```

The required number of cases is 133 and expected power is 0.8011167. Compare the number of cases to the FRRC value used in vignette 2.

10.4.3 RRFC

```
ncases <- SsSampleSizeKGivenJ(dataset02, FOM = "Wilcoxon", J = 10, method = "DBMH", opt
str(ncases)
#> 'data.frame':    1 obs. of  2 variables:
#>  $ KRRFC    : num 53
#>  $ powerRRFC: num 0.805
```

The required number of cases is 53 and expected power is 0.8049666. Compare the number of cases to the RRFC value used in vignette 2.

Chapter 11

ROC-ORH sample size using RJafroc

11.1 Introduction

The use of the functions introduced in vignette 3, but this time using the ORH method to estimate the variance components, is illustrated here. The reader should confirm that these give the same results as the corresponding ones obtained using the DBMH method. When the figure of merit is the empirical AUC, the two methods can be shown to be identical.

11.2 Illustration of `SsPowerGivenJK()` using `method = "ORH"`

```
power <- SsPowerGivenJK(dataset02, FOM = "Wilcoxon", J = 6, K = 251, method = "ORH", option = "RR
```

The returned value is a `list` containing the expected power, the non-centrality parameter, the denominator degrees of freedom and the F-statistic (the numerator degrees of freedom is always one less than the number of treatments, i.e., unity in this example).

```
str(power)
#> 'data.frame':    1 obs. of  4 variables:
#>  $ powerRRRC: num 0.801
#>  $ ncpRRRC  : num 8.91
```

```
#>  $ ddfHRRRC : num 16.1
#>  $ fRRRC    : num 4.49
```

Expected power is 0.8005403.

11.3 Illustration of `SsPowerTable()` using `method = "ORH"`

```
powTab <- SsPowerTable(dataset02, FOM = "Wilcoxon", method = "ORH", option = "RRRC")
```

Now show the power table `powTab`.

```
powTab
#>    numReaders numCases power
#> 1           3    >2000  <NA>
#> 2           3    >2000  <NA>
#> 3           4     1089   0.8
#> 4           4     1089   0.8
#> 5           5      344 0.801
#> 6           5      344 0.801
#> 7           6      251 0.801
#> 8           6      251 0.801
#> 9           7      211 0.801
#> 10          7      211 0.801
#> 11          8      188 0.801
#> 12          8      188 0.801
#> 13          9      173 0.801
#> 14          9      173 0.801
#> 15         10      163 0.802
#> 16         10      163 0.802
#> 17         11      155 0.801
#> 18         11      155 0.801
#> 19         12      149 0.802
#> 20         12      149 0.802
#> 21         13      144 0.801
#> 22         13      144 0.801
#> 23         14      140 0.802
#> 24         14      140 0.802
#> 25         15      137 0.802
#> 26         15      137 0.802
#> 27         16      134 0.802
```

```
#> 28          16      134 0.802
#> 29          17      131 0.801
#> 30          17      131 0.801
#> 31          18      129 0.801
#> 32          18      129 0.801
#> 33          19      127 0.801
#> 34          19      127 0.801
#> 35          20      126 0.802
#> 36          20      126 0.802
#> 37          21      124 0.801
#> 38          21      124 0.801
#> 39          22      123 0.802
#> 40          22      123 0.802
#> 41          23      122 0.802
#> 42          23      122 0.802
#> 43          24      121 0.803
#> 44          24      121 0.803
#> 45          25      120 0.802
#> 46          25      120 0.802
#> 47          26      119 0.802
#> 48          26      119 0.802
#> 49          27      118 0.802
#> 50          27      118 0.802
#> 51          28      117 0.801
#> 52          28      117 0.801
#> 53          29      117 0.803
#> 54          29      117 0.803
#> 55          30      116 0.802
#> 56          30      116 0.802
#> 57          31      115 0.801
#> 58          31      115 0.801
#> 59          32      115 0.803
#> 60          32      115 0.803
#> 61          33      114 0.801
#> 62          33      114 0.801
#> 63          34      114 0.803
#> 64          34      114 0.803
#> 65          35      113 0.801
#> 66          35      113 0.801
#> 67          36      113 0.802
#> 68          36      113 0.802
#> 69          37      112   0.8
#> 70          37      112   0.8
#> 71          38      112 0.802
#> 72          38      112 0.802
```

```
#> 73          39      112 0.803
#> 74          39      112 0.803
#> 75          40      111 0.801
#> 76          40      111 0.801
#> 77          41      111 0.802
#> 78          41      111 0.802
#> 79          42      111 0.803
#> 80          42      111 0.803
#> 81          43      110 0.801
#> 82          43      110 0.801
#> 83          44      110 0.802
#> 84          44      110 0.802
#> 85          45      110 0.802
#> 86          45      110 0.802
#> 87          46      110 0.803
#> 88          46      110 0.803
#> 89          47      109 0.801
#> 90          47      109 0.801
#> 91          48      109 0.802
#> 92          48      109 0.802
#> 93          49      109 0.802
#> 94          49      109 0.802
#> 95          50      109 0.803
#> 96          50      109 0.803
#> 97          51      108   0.8
#> 98          51      108   0.8
#> 99          52      108 0.801
#> 100         52      108 0.801
#> 101         53      108 0.802
#> 102         53      108 0.802
#> 103         54      108 0.802
#> 104         54      108 0.802
#> 105         55      108 0.803
#> 106         55      108 0.803
#> 107         56      107   0.8
#> 108         56      107   0.8
#> 109         57      107 0.801
#> 110         57      107 0.801
#> 111         58      107 0.801
#> 112         58      107 0.801
#> 113         59      107 0.802
#> 114         59      107 0.802
#> 115         60      107 0.802
#> 116         60      107 0.802
#> 117         61      107 0.803
```

```
#> 118          61      107 0.803
#> 119          62      107 0.803
#> 120          62      107 0.803
#> 121          63      106   0.8
#> 122          63      106   0.8
#> 123          64      106 0.801
#> 124          64      106 0.801
#> 125          65      106 0.801
#> 126          65      106 0.801
#> 127          66      106 0.802
#> 128          66      106 0.802
#> 129          67      106 0.802
#> 130          67      106 0.802
#> 131          68      106 0.802
#> 132          68      106 0.802
#> 133          69      106 0.803
#> 134          69      106 0.803
#> 135          70      106 0.803
#> 136          70      106 0.803
#> 137          71      106 0.804
#> 138          71      106 0.804
#> 139          72      105   0.8
#> 140          72      105   0.8
#> 141          73      105 0.801
#> 142          73      105 0.801
#> 143          74      105 0.801
#> 144          74      105 0.801
#> 145          75      105 0.801
#> 146          75      105 0.801
#> 147          76      105 0.802
#> 148          76      105 0.802
#> 149          77      105 0.802
#> 150          77      105 0.802
#> 151          78      105 0.802
#> 152          78      105 0.802
#> 153          79      105 0.803
#> 154          79      105 0.803
#> 155          80      105 0.803
#> 156          80      105 0.803
#> 157          81      105 0.803
#> 158          81      105 0.803
#> 159          82      105 0.803
#> 160          82      105 0.803
#> 161          83      104   0.8
#> 162          83      104   0.8
```

```
#> 163         84      104   0.8
#> 164         84      104   0.8
#> 165         85      104 0.801
#> 166         85      104 0.801
#> 167         86      104 0.801
#> 168         86      104 0.801
#> 169         87      104 0.801
#> 170         87      104 0.801
#> 171         88      104 0.801
#> 172         88      104 0.801
#> 173         89      104 0.802
#> 174         89      104 0.802
#> 175         90      104 0.802
#> 176         90      104 0.802
#> 177         91      104 0.802
#> 178         91      104 0.802
#> 179         92      104 0.802
#> 180         92      104 0.802
#> 181         93      104 0.802
#> 182         93      104 0.802
#> 183         94      104 0.803
#> 184         94      104 0.803
#> 185         95      104 0.803
#> 186         95      104 0.803
#> 187         96      104 0.803
#> 188         96      104 0.803
#> 189         97      104 0.803
#> 190         97      104 0.803
#> 191         98      104 0.804
#> 192         98      104 0.804
#> 193         99      104 0.804
#> 194         99      104 0.804
#> 195        100      103   0.8
#> 196        100      103   0.8
```

Since the default `FOM = "Wilcoxon"`, the table is identical to that generated in vignette 3, which used `method = "DBMH"`.

11.4 Illustrations of SsSampleSizeKGivenJ() using method = "ORH"

11.4.1 For RRRC generalization

```
ncases <- SsSampleSizeKGivenJ(dataset02, FOM = "Wilcoxon", J = 10, method = "ORH", option = "RRRC
```

ncases is a list containing the number of cases ncases$KRRRC and expected power ncases$powerRRRC.

```
str(ncases)
#> 'data.frame':    1 obs. of  2 variables:
#>  $ KRRRC    : num 163
#>  $ powerRRRC: num 0.802
```

The required number of cases is 163 and expected power is 0.8015625.

11.4.2 For FRRC generalization

```
ncases <- SsSampleSizeKGivenJ(dataset02, FOM = "Wilcoxon", J = 10, method = "ORH", option = "FRRC
```

The required number of cases is 133 and expected power is 0.8011167.

11.4.3 For RRFC generalization

```
ncases <- SsSampleSizeKGivenJ(dataset02, FOM = "Wilcoxon", J = 10, method = "ORH", option = "RRFC
```

The required number of cases is 53 and expected power is 0.8049666.

Chapter 12

Choosing a realistic effect size

12.1 Introduction

- The value of the true `FOM` difference between the treatments, i.e., the true effect-size (ES) is, of course, unknown. If it were known, there would be no need to conduct an ROC study. One would simply adopt the treatment with the higher `FOM`. Sample-size estimation involves making an educated guess regarding the ES, called the *anticipated* ES, and denoted by `d`. To quote (ICRU, 2008): "any calculation of power amounts to specification of the anticipated effect-size". Increasing the anticipated ES will increase statistical power but may represent an unrealistic expectation of the true difference between the treatments, in the sense that it overestimates the ability of technology to achieve this much improvement. An unduly small might be clinically insignificant, besides requiring a very large sample-size to achieve sufficient power.

- There is a key difference between *statistical* significance and *clinical* significance. An effect-size in AUC units could be so small, e.g., 0.001, as to be clinically insignificant, but by employing a sufficiently large sample size one could design a study to detect this small and clinically meaningless difference with high probability, i.e., high statistical power.

- What determines clinical significance? A small effect-size, e.g., 0.01 AUC units, could be clinically significant if it applies to a large population, where the small benefit in detection rate is amplified by the number of patients benefiting from the new treatment. In contrast, for an "orphan" disease, i.e., one with very low prevalence, an effect-size of 0.05 might not

be enough to justify the additional cost of the new treatment. The improvement might have to be 0.1 before it is worth it for a new treatment to be brought to market. One hates to monetize life and death issues, but there is no getting away from it, as cost/benefit issues determine clinical significance. The arbiters of clinical significance are engineers, imaging scientists, clinicians, epidemiologists, insurance companies and those who set government health care policies. The engineers and imaging scientists determine whether the effect-size the clinicians would like is feasible from technical and scientific viewpoints. The clinician determines, based on incidence of disease and other considerations, e.g., altruistic, malpractice, cost of the new device and insurance reimbursement, what effect-size is justifiable. Cohen has suggested that d values of 0.2, 0.5, and 0.8 be considered small, medium, and large, respectively, but he has also argued against their indiscriminate usage. However, after a study is completed, clinicians often find that an effect-size that biostatisticians label as small may, in certain circumstances, be clinically significant and an effect-size that they label as large may in other circumstances be clinically insignificant. Clearly, this is a complex issue. Some suggestions on choosing a clinically significant effect size are made in **Chapter 11**.

- Does one even need to perform a pivotal study? If the pilot study returns a significant difference, one has rejected the NH and that is all there is to it. There is no need to perform the pivotal study, unless one "tweaks" the new treatment and/or casts a wider sampling net to make a stronger argument, perhaps to the FDA, that the treatments are indeed generalizable, and that the difference is in the right direction (new treatment FOM $>$ conventional treatment FOM). If a significant difference is observed in the opposite direction (e.g., new treatment FOM $<$ conventional treatment FOM) one cannot justify a pivotal study with an expected effect-size in the "other or favored" direction; see example below. Since the Van Dyke pilot study came close to rejecting the NH and the observed effect size, see below, is not too small, a pivotal study is justified.

- This vignette discusses choosing a realistic effect size based on the pilot study. Illustrated first is using Van Dyke dataset, regarded as the pilot study.

12.2 Illustration of `SsPowerGivenJK()` using `method = "ORH"`

```
rocData <- dataset02 ##"VanDyke.lrc"
#fileName <- dataset03 ## "Franken1.lrc"
retDbm <- StSignificanceTesting(dataset = rocData, FOM = "Wilcoxon", method = "DBMH")
```

```
str(retDbm$ciDiffTrtRRRC)
#> 'data.frame':    1 obs. of  8 variables:
#>  $ TrtDiff : chr "Trt0-Trt1"
#>  $ Estimate: num -0.0438
#>  $ StdErr  : num 0.0207
#>  $ DF      : num 15.3
#>  $ t       : num -2.11
#>  $ PrGTt   : num 0.0517
#>  $ CILower : num -0.088
#>  $ CIUpper : num 0.000359
```

- Lacking any other information, the observed effect-size is the best estimate of the effect-size to be anticipated. The output shows that the FOM difference, for treatment 0 minus treatment 1, is -0.0438003. In the actual study treatment 1 is the new modality which hopes to improve upon 0, the conventional modality. Since the sign is negative, the difference is going the right way and is justified in moving forward with planning a pivotal study. [If the difference went the other way, there is little justification for a pivotal study].
- The standard error of the difference is 0.0207486.
- An optimistic, but not unduly so, effect size is given by:

```
effectSizeOpt <- abs(retDbm$ciDiffTrtRRRC$Estimate) + 2*retDbm$ciDiffTrtRRRC$StdErr
```

- The observed effect-size is a realization of a random variable. The lower limit of the 95% confidence interval is given by -0.0879595 and the upper limit by 3.5885444×10^{-4}. CI's generated like this, with independent sets of data, are expected to encompass the true value with 95% probability. The lower end (greatest magnitude of the difference) of the confidence interval is -0.0852976, and this is the optimistic estimate. Since the sign is immaterial, one uses as the optimistic estimate the value 0.0852976.
- **While the sign is immaterial for sample size estimates, the decision to conduct the pivotal most certainly is material. If the sign went the other way, with the new modality lower than the conventional modality, one would be unjustified in conducting a pivotal study.**

12.3 References

Chapter 13

FROC sample size estimation and comparison to ROC

13.1 Introduction

- FROC sample size estimation is not fundamentally different from the previously outlined procedure (see vignettes corresponding to Chapter 11) for the ROC paradigm. To recapitulate, based on analysis of a pilot ROC dataset and using a specified FOM, e.g., `FOM = Wilcoxon`, and either `method = "DBMH"` or `method = "ORH"` for significance testing, one estimates the intrinsic variability of the data expressed in terms of variance components or the covariance matrix. The second step is to postulate a clinically realistic effect-size, e.g., the anticipated AUC difference between the two treatments. Given these values, the sample size functions implemented in `RJafroc` (beginning with `Ss`) allow one to estimate the number of readers and cases necessary to detect (i.e., reject the null hypothesis) the specified effect size at specified Type II error rate, typically chosen to be 20% (corresponding to 80% statistical power) and specified Type I error rate, typically chosen to be 5%.

- In FROC analysis the only difference, **indeed the critical difference**, is the choice of FOM; e.g., `FOM = "wAFROC"` instead of the inferred ROC-AUC, `FOM = "HrAuc"`. The FROC dataset is analyzed using either the DBMH or the ORH method. This yields the necessary variance components or the covariance matrix corresponding to the wAFROC-AUC. The next step is to specify the effect-size **in wAFROC-AUC units**, and therein lies the rub. What value does one use? The ROC-AUC has a his-

torically well-known interpretation: the classification ability at separating diseased patients from non-diseased patients, while the wAFROC-AUC does not. Needed is a way of relating the effect-size in ROC-AUC units to one in wAFROC-AUC units: as should be obvious this requires a physical model, e.g., the RSM, that predicts both ROC and wAFROC curves and the respective AUCs.

1. One chooses an ROC-AUC effect-size that is realistic, one that clinicians understand and can therefore participate in, in the effect-size postulation process.
2. One converts the ROC effect-size to a wAFROC-AUC effect-size. The method for this is described in the next section.
3. One uses the sample size tools in in `RJafroc` to determine sample size or power.

- **It is important to recognize is that all quantities have to be in the same units**. When performing ROC analysis, everything (variance components and effect-size) has to be in units of the selected FOM, e.g., `FOM = "Wilcoxon"` which is identical to the empirical ROC-AUC. When doing wAFROC analysis, everything has to be in units of the wAFROC-AUC. The variance components and effect-size in wAFROC-AUC units will be different from their corresponding ROC counterparts. In particular, as shown next, an ROC-AUC effect-size of 0.05 generally corresponds to a larger effect-size in wAFROC-AUC units. The reason for this is that the range over which wAFROC-AUC can vary, namely 0 to 1, is twice the corresponding ROC-AUC range.
- The next section explains the steps used to implement step #2 above.

13.2 Relating an ROC effect-size to a wAFROC effect-size

- If the original data is FROC, one needs to first convert it to ROC, using `DfFroc2Roc()`: **the RSM fits ROC data**.
- For each treatment and reader the inferred ROC data is fitted by `FitRsmRoc()`, yielding estimates of the RSM *physical* (or primed) parameters (not the *intrinsic* values).
- The following example uses the *first two* treatments of the "FED" dataset, `dataset04`, which is a 5 treatment 4 radiologist FROC dataset acquired by Dr. Federica Zanca et. al. (Zanca et al., 2009). The dataset has 5 treatments and 4 readers and 200 cases and was acquired on a 5-point integer scale, i.e., it is already binned. If not one needs to bin the dataset

using `DfBinDataset()`. I need to emphasize this point: **if the dataset represents continuous ratings, as with a CAD algorithm, one must bin the dataset to (ideally) about 5 bins**. The number of parameters that must be estimated increases with the number of bins (for each bin one needs to estimate a cutoff parameter).

- The reason for using RSM parameter values only for the first two treatments is that these were found (Zanca et al., 2009) to be almost equivalent (more precisely, the NH could not be rejected for the first two treatments, so it makes sense to regard them as "almost" NH treatments.
- The following code block defines the pilot FROC data `frocData` (corresponding to `dataset04`, which is the "FED" dataset, but with only treatments 1 and 2 extracted, using `DfExtractDataset()`) and `rocData`, i.e., the highest-rating ROC dataset inferred from the FROC dataset using `DfFroc2Roc()`.

```
frocData <- DfExtractDataset(dataset04, trts = c(1,2))
rocData <- DfFroc2Roc(frocData)
```

The next code block determines `lesDist`, the lesion distribution array, which has `Lmax` (maximum number of lesions per diseased case over the dataset) rows and two columns. The first column contains the integers 1, 2, ..., `Lmax` and the second column contains the fraction of diseased cases with the number of lesions per case specified in the first column. The second column will sum to unity. The RSM fitting algorithm needs to know how lesion-rich the dataset is, as the RSM predicted ROC-AUC depends on the lesion-richness of the dataset. For reasons that will become clear below, one also needs `lesWghts`, the distribution of the lesion weights.

```
lesDistr <- UtilLesionDistr(frocData)
lesWghts <- UtilLesionWeightsDistr(frocData) # this is needed later
```

The meanings of `lesDistr` and `lesWghts` is clear from examining their values:

```
print(lesDistr)
#>      [,1] [,2]
#> [1,]    1 0.69
#> [2,]    2 0.20
#> [3,]    3 0.11
print(lesWghts)
#>      [,1]      [,2]      [,3]      [,4]
#> [1,]    1 1.0000000      -Inf      -Inf
#> [2,]    2 0.5000000 0.5000000      -Inf
#> [3,]    3 0.3333333 0.3333333 0.3333333
```

For this dataset `Lmax` is 3, and 69 percent of the diseased cases have one lesion, 20 percent have two lesions and 11 percent have three lesions. Since the lesions are equally weighted, on cases with one lesion the weight of the lesion is unity, on cases with two lesions the weights of each lesion is 0.5 and on cases with three lesions the weight of each lesion is 1/3.

The next code block determines the number of treatments and readers (`I` and `J`) from the dimensions of the `frocData$NL` array. It creates an array `RsmParms` to hold the RSM fitted parameter values. For each treatment and reader it applies the fitting algorithm `FitRsmRoc()`. The first three returned values are `mu`, `lambdaP` and `nuP`, corresponding to RSM parameters μ, λ' and ν' .

```
I <- dim(frocData$NL)[1]
J <- dim(frocData$NL)[2]
RsmParms <- array(dim = c(I,J,3))
for (i in 1:I) {
  for (j in 1:J)  {
    x1 <- FitRsmRoc(rocData, trt = i, rdr = j, lesDistr)
    RsmParms[i,j,1] <- x1[[1]] # mu
    RsmParms[i,j,2] <- x1[[2]] # lambdaP
    RsmParms[i,j,3] <- x1[[3]] # nuP
  }
}
```

I recommend taking the median of each of the parameters, over all treatment-reader indices, as representing the average NH dataset. The median is less sensitive to outliers than the mean.

```
muMed <- median(RsmParms[,,1])
lambdaPMed <- median(RsmParms[,,2])
nuPMed <- median(RsmParms[,,3])
```

The defining values of the fitting model are `muMed` = 3.3105557, `lambdaPMed` = 1.714368 and `nuPMed` = 0.7036567. Note that these obey the constraints `lambdaPMed > 0` and `0 < nuP < 1`. One then converts the physical parameters to the intrinsic values:

```
temp <- UtilPhysical2IntrinsicRSM(muMed, lambdaPMed, nuPMed)
lambdaMed <- temp$lambda
nuMed <- temp$nu
```

In terms of intrinsic parameters, the defining values of the fitting model are `muMed` = 3.3105557, `lambdaMed` = 5.6755108 and `nuMed` = 0.3673814. We are now ready to calcuate the expected NH FOMs using the ROC -AUC and the wAFROC FOM.

```
aucRocNH <- PlotRsmOperatingCharacteristics(muMed, lambdaMed, nuMed,
                                            lesDistr = lesDistr,
                                            lesWghtDistr = lesWghts, OpChType = "ROC")$aucROC
aucwAfrocNH <- PlotRsmOperatingCharacteristics(muMed, lambdaMed, nuMed,
                                              lesDistr = lesDistr,
                                              lesWghtDistr = lesWghts, OpChType = "wAFROC")$aucwA
```

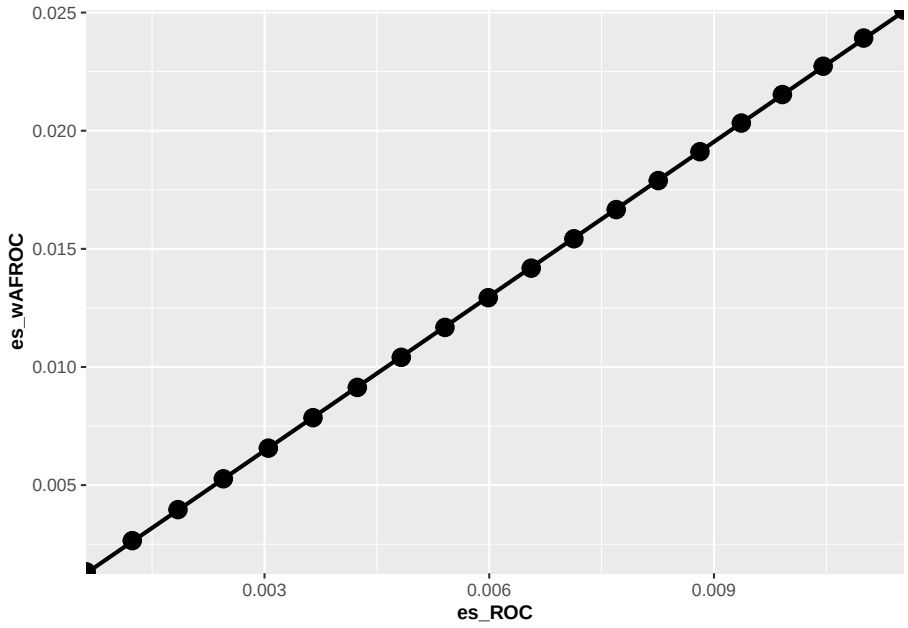
- The plotting function `PlotRsmOperatingCharacteristics()` returns a number of other objects, most importantly the plot, but here we use only the AUC, which is obtained by numerical integration of the predicted operating characteristics. However, it calls for the **intrinsic** RSM parameters, which is why we had to convert the **physical** to the **intrinsic** values.
- One has `aucRocNH = 0.8791301` and `aucwAfrocNH = 0.7198311`. Note that the wAFROC-FOM is smaller than the ROC-FOM as it includes the localization constraint.
- To induce the alternative hypothesis condition, one increments μ_{NH} by Δ_μ. The resulting ROC-AUC and wAFROC-AUC are calculated, again by numerical integration of the RSM predicted ROC and wAFROC curves, leading to the corresponding effect-sizes (note that in each equation below one takes the difference between the AH value minus the NH value):
- The next step is to calculate the effect size (new value minus the NH value) using ROC and wAFROC FOMs for a series of specifed `deltaMu` values. This generates values that can be used to interpolate a wAFROC effect size for a specified ROC effect size.

```
deltaMu <- seq(0.01, 0.2, 0.01) # values of deltaMu to scan below
esRoc <- array(dim = length(deltaMu));eswAfroc <- array(dim = length(deltaMu))
for (i in 1:length(deltaMu)) {
  esRoc[i] <- PlotRsmOperatingCharacteristics(
    muMed + deltaMu[i], lambdaMed, nuMed, lesDistr = lesDistr,
    lesWghtDistr = lesWghts, OpChType = "ROC")$aucROC - aucRocNH
  eswAfroc[i] <- PlotRsmOperatingCharacteristics(
    muMed+ deltaMu[i], lambdaMed, nuMed, lesDistr = lesDistr,
    lesWghtDistr = lesWghts, OpChType = "wAFROC")$aucwAFROC - aucwAfrocNH
  cat("ES_ROC = ", esRoc[i], ", ES_wAFROC = ", eswAfroc[i],"\n")
}
#> ES_ROC =  0.0006197813 , ES_wAFROC =  0.001329066
#> ES_ROC =  0.001234752 , ES_wAFROC =  0.002650005
#> ES_ROC =  0.00184496 , ES_wAFROC =  0.003962878
#> ES_ROC =  0.002450451 , ES_wAFROC =  0.005267748
#> ES_ROC =  0.003051273 , ES_wAFROC =  0.006564677
#> ES_ROC =  0.003647472 , ES_wAFROC =  0.007853725
```

```
#> ES_ROC =  0.004239094 , ES_wAFROC =  0.009134954
#> ES_ROC =  0.004826184 , ES_wAFROC =  0.01040842
#> ES_ROC =  0.005408788 , ES_wAFROC =  0.0116742
#> ES_ROC =  0.00598695 , ES_wAFROC =  0.01293233
#> ES_ROC =  0.006560717 , ES_wAFROC =  0.01418289
#> ES_ROC =  0.007130131 , ES_wAFROC =  0.01542592
#> ES_ROC =  0.007695238 , ES_wAFROC =  0.0166615
#> ES_ROC =  0.00825608 , ES_wAFROC =  0.01788967
#> ES_ROC =  0.008812702 , ES_wAFROC =  0.0191105
#> ES_ROC =  0.009365145 , ES_wAFROC =  0.02032404
#> ES_ROC =  0.009913453 , ES_wAFROC =  0.02153036
#> ES_ROC =  0.01045767 , ES_wAFROC =  0.0227295
#> ES_ROC =  0.01099783 , ES_wAFROC =  0.02392152
#> ES_ROC =  0.01153399 , ES_wAFROC =  0.02510649
```

Here is a plot of wAFROC effect size (y-axis) vs. ROC effect size.

```
df <- data.frame(es_ROC = esRoc, es_wAFROC = eswAfroc)
p <- ggplot(data = df, aes(x = es_ROC, y = es_wAFROC)) +
  geom_smooth(method = "lm", se = FALSE, color = "black", formula = y ~ x) +
  geom_point(size = 4) +
  scale_color_manual(values = "black") +
  theme(axis.title.y = element_text(size = 10,face="bold"),
        axis.title.x = element_text(size = 10,face="bold")) +
  scale_x_continuous(expand = c(0, 0)) +
  scale_y_continuous(expand = c(0, 0))
print(p)
```



The plot is very close to linear. This makes it easy to design an interpolation function. In the following code block the first line fits `eswAfroc` vs. `esRoc` using the linear model `lm()` function constrained to pass through the origin (the minus one): `scaleFactor <- lm(eswAfroc ~ -1 + esRoc)`. One expects this constraint since for `deltaMu = 0` the effect size must be zero no matter how it is measured.

```
scaleFactor<-lm(eswAfroc~-1+esRoc) # fit values to straight line thru origin
effectSizeROC <- seq(0.01, 0.1, 0.01)
effectSizewAFROC <- effectSizeROC*scaleFactor$coefficients[1] # r2 = summary(scaleFactor)$r.squa
```

The scaleFactor of the straight line fit is `scaleFactor`, where scaleFactor = 2.1688609 and `R2` = 0.9999904. Therefore, the conversion from ROC to wAFROC effect size is: `effectSizewAFROC = scaleFactor * effectSizeROC`. **The wAFROC effect size is twice the ROC effect size.** All that remains is to calculate the variance componenents using the two FOMs:

13.3 Computing the respective variance components

The code block applies `StSignificanceTesting()` to `rocData` and `frocData`, using the appropriate FOM, and extracts the variance components.

```
temp1 <- StSignificanceTesting(rocData, FOM = "Wilcoxon", method = "DBMH", option = "RI
temp2 <- StSignificanceTesting(frocData, FOM = "wAFROC", method = "DBMH", option = "RRI
varCompROC <- temp1$varComp
varCompwAFROC <- temp2$varComp
```

The observed wAFROC effect-size is -0.0068562. This is a very small effect size; the corresponding ROC effect-size is -0.0051; the sign does not affect the calculations, which is too small to reach 80% power. It is not surprising that the study (Zanca et al., 2009) did not find a significant difference between these two treatments

The respective variance components are:

```
print(varCompROC)
#>           varR       varC        varTR       varTC       varRC     varErr
#> 1 0.000827738 0.03812335 0.0001526507 0.009644327 0.003544196 0.09484637
print(varCompwAFROC)
#>           varR       varC         varTR      varTC      varRC    varErr
#> 1 0.001854229 0.06117805 -0.0004439279 0.01016519 0.01355883 0.0967256
```

Only terms involving treatment are relevant to sample size. The wAFROC `varTC` and `varError` values are slightly larger than the ROC ones - as expected because, again, the range of the wAFROC FOM is twice that of the ROC FOM.

13.4 Comparing ROC power to wAFROC power for equivalent effect-sizes

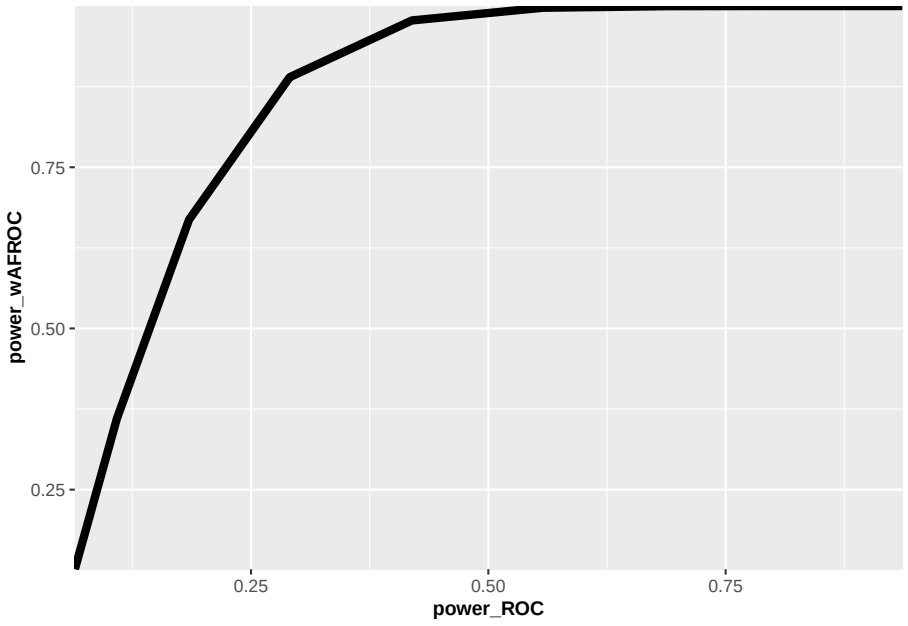
We are now ready to compare ROC and wAFROC powers for equivalent effect sizes. The following example is for 5 readers (`JTest`) and 100 cases (`KTest`) in the **pivotal study**.

```
powerROC <- array(dim = length(effectSizeROC));powerwAFROC <- array(dim = length(effect

JTest <- 5;KTest <- 100
for (i in 1:length(effectSizeROC)) {
  varYTR <- varCompROC$varTR # these are pseudovalue based variance components assumin
  varYTC <- varCompROC$varTC
  varYEps <- varCompROC$varErr
  ret <- SsPowerGivenJKDbmVarComp (J = JTest, K = KTest, effectSize = effectSizeROC[i]
  powerROC[i] <- ret$powerRRRC

  varYTR <- varCompwAFROC$varTR # these are pseudovalue based variance components assu
  varYTC <- varCompwAFROC$varTC
```

```
  varYEps <- varCompwAFROC$varErr
  ret <- SsPowerGivenJKDbmVarComp (J = JTest, K = KTest, effectSize = effectSizewAFROC[i], varYTR
  powerwAFROC[i] <- ret$powerRRRC

  cat("ROC-ES = ", effectSizeROC[i], ", wAFROC-ES = ", effectSizewAFROC[i],
      ", Power-ROC = ", powerROC[i], ", Power-wAFROC = ", powerwAFROC[i], "\n")
}
#> ROC-ES =  0.01 , wAFROC-ES =  0.02168861 , Power-ROC =  0.06443046 , Power-wAFROC =  0.126631
#> ROC-ES =  0.02 , wAFROC-ES =  0.04337722 , Power-ROC =  0.108789 , Power-wAFROC =  0.3605744
#> ROC-ES =  0.03 , wAFROC-ES =  0.06506583 , Power-ROC =  0.1847115 , Power-wAFROC =  0.6686874
#> ROC-ES =  0.04 , wAFROC-ES =  0.08675444 , Power-ROC =  0.2907927 , Power-wAFROC =  0.8897125
#> ROC-ES =  0.05 , wAFROC-ES =  0.108443 , Power-ROC =  0.4195443 , Power-wAFROC =  0.9777308
#> ROC-ES =  0.06 , wAFROC-ES =  0.1301317 , Power-ROC =  0.5573812 , Power-wAFROC =  0.9973522
#> ROC-ES =  0.07 , wAFROC-ES =  0.1518203 , Power-ROC =  0.6881601 , Power-wAFROC =  0.9998172
#> ROC-ES =  0.08 , wAFROC-ES =  0.1735089 , Power-ROC =  0.7983611 , Power-wAFROC =  0.9999927
#> ROC-ES =  0.09 , wAFROC-ES =  0.1951975 , Power-ROC =  0.8809508 , Power-wAFROC =  0.9999998
#> ROC-ES =  0.1 , wAFROC-ES =  0.2168861 , Power-ROC =  0.936068 , Power-wAFROC =  1
```

Since the wAFROC effect size is about a factor of two larger than the ROC effect size, wAFROC power is larger than that for ROC. The effect is magnified as the effect size enters as the square in the formula for the power (this overwhelms the slight increase in variability of wAFROC-FOM relative to ROC-FOM noted previously). The following is a plot of the respective powers.

```
df <- data.frame(power_ROC = powerROC, power_wAFROC = powerwAFROC)
p <- ggplot(mapping = aes(x = power_ROC, y = power_wAFROC)) +
  geom_line(data = df, size = 2)+
  scale_color_manual(values = "black") +
  theme(axis.title.y = element_text(size = 10,face="bold"),
        axis.title.x = element_text(size = 10,face="bold"))  +
  scale_x_continuous(expand = c(0, 0)) +
  scale_y_continuous(expand = c(0, 0))
print(p)
```



13.5 References

Chapter 14

FROC sample size estimation using specified ROC effect

14.1 Introduction

This example uses the FED dataset as a pilot FROC study and function `SsFrocNhRsmModel()` to construct the NH model (encapsulating some of the code in the previous vignette).

14.2 Constructing the NH model for the dataset

One starts by extracting the first two treatments from `dataset04`, which represent the NH dataset, see previous vignette. Next one constructs the NH model - note that the lesion distribution `lesionPmf` can be specified here independently of that in the pilot dataset. This allows some control over selection of the diseased cases in the pivotal study.

```
frocNhData <- DfExtractDataset(dataset04, trts = c(1,2))
ret <- SsFrocNhRsmModel(frocNhData, lesionPmf = c(0.7, 0.2, 0.1))
muMed <- ret$muMed
lambdaMed <- ret$lambdaMed
nuMed <- ret$nuMed
lesDistr <- ret$lesDistr
lesWghtDistr <- ret$lesWghtDistr
scaleFactor <- ret$scaleFactor
```

The fitting model is defined by `muMed = r muMed`, `lambdaMed = 5.6140942` and `nuMed = 0.3696988` and `lesionPmf`. The effect size scale factor is 2.1542102.

```
aucRocNH <- PlotRsmOperatingCharacteristics(muMed, lambdaMed, nuMed,
                                            lesDistr = lesDistr,
                                            lesWghtDistr = lesWghtDistr, OpChType = "RO
aucwAfrocNH <- PlotRsmOperatingCharacteristics(muMed, lambdaMed, nuMed,
                                               lesDistr = lesDistr,
                                               lesWghtDistr = lesWghtDistr, OpChType =
```

The null hypothesis ROC AUC is 0.8790725 and the corresponding NH wAFROC AUC is 0.7231709.

14.3 Extracting the wAFROC variance components

The next code block applies `StSignificanceTesting()` to `frocNhData`, using `FOM = "wAFROC"` and extracts the variance components.

```
varCompwAFROC  <- StSignificanceTesting(frocNhData, FOM = "wAFROC", method = "DBMH", op
```

14.4 wAFROC power for specified ROC effect size, number of readers J and number of cases K

The following example is for ROC effect size = 0.05, 5 readers (`J`) and 100 cases (`K`) in the **pivotal study**.

```
ROC_ES <- 0.05
effectSizewAFROC <- scaleFactor * ROC_ES
J <- 5;K <- 100

varYTR <- varCompwAFROC$varTR
varYTC <- varCompwAFROC$varTC
varYEps <- varCompwAFROC$varErr
ret <- SsPowerGivenJKDbmVarComp (J = J, K = K, effectSize = effectSizewAFROC,
                                 varYTR, varYTC, varYEps, option = "RRRC")
powerwAFROC <- ret$powerRRRC

cat("ROC-ES = ", ROC_ES, ", wAFROC-ES = ", ROC_ES * scaleFactor, ", Power-wAFROC = ", p
#> ROC-ES =  0.05 , wAFROC-ES =  0.1077105 , Power-wAFROC =  0.976293
```

14.5 wAFROC number of cases for 80% power for a given number of readers J

```
varYTR <- varCompwAFROC$varTR
varYTC <- varCompwAFROC$varTC
varYEps <- varCompwAFROC$varErr
ret2 <- SsSampleSizeKGivenJ(dataset = NULL, J = 6, effectSize = effectSizewAFROC, method = "DBMH"
                      list(varYTR = varYTR, varYTC = varYTC, varYEps = varYEps))

cat("ROC-ES = ", ROC_ES, ", wAFROC-ES = ", ROC_ES * scaleFactor,
    ", K80RRRC = ", ret2$KRRRC, ", Power-wAFROC = ", ret2$powerRRRC, "\n")
#> ROC-ES =  0.05 , wAFROC-ES =  0.1077105 , K80RRRC =  42 , Power-wAFROC =  0.804794
```

14.6 wAFROC Power for a given number of readers J and cases K

```
ret3 <- SsPowerGivenJK(dataset = NULL, J = 6, K = ret2$KRRRC, effectSize = effectSizewAFROC, meth
                    list(varYTR = varYTR, varYTC = varYTC, varYEps = varYEps))

cat("ROC-ES = ", ROC_ES, ", wAFROC-ES = ", ROC_ES * scaleFactor,
    ", powerRRRC = ", ret3$powerRRRC, "\n")
#> ROC-ES =  0.05 , wAFROC-ES =  0.1077105 , powerRRRC =  0.804794
```

The estimated power is close to 80% as the number of cases (`ret2$KRRRC = 42`) was chosen deliberately from the previous code block.

14.7 References

Chapter 15

RSM predicted operating characteristics

15.1 Introduction

- The purpose of this vignette is to explain the operating characteristics predicted by the RSM. It relates to Chapter 17 in my book (Chakraborty, 2017).
- This vignette is under development ...
- Also to explain the difference between `dataset` members (`lesionID`, `lesionWeight`) and (`lesDist`, `lesWghtDistr`), which are RSM model parameters.

15.2 The distinction between predicted curves and empirical curves

- Operating characteristics predicted by a model have zero sampling variability.

- Empirical operating characteristics, which apply to datasets, have non-zero sampling variability.
- If the model is corect, as the numbers of cases in the dataset increases, the empirical operating characteristic asymptotically approaches the predicted curve.

15.3 The RSM model

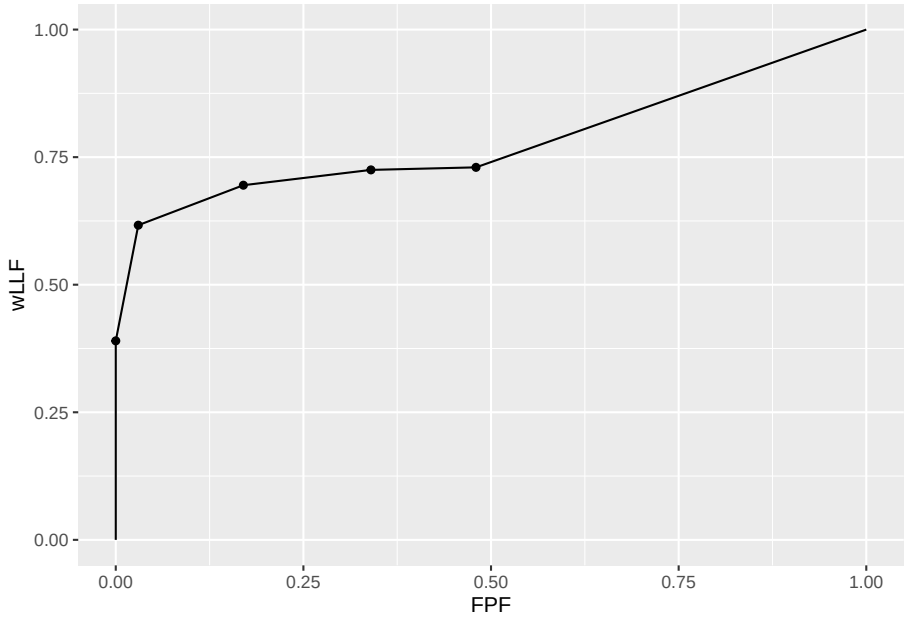
- The 3 RSM parameters and two additional parameters characterizing the dataset determine the wAFROC curve.
- The 3 RSM parameters are μ, λ and ν.
- The two dataset parameters are:
  - The distribution of number of lesions per diseased case, `lesDist`.
  - The distribution of lesion weights, `lesWghtDistr`.
- These parameters do not apply to individual cases; rather they refer to a large population (asymptotically infinite in size) of cases.

```
str(dataset04$lesionID)
#>  num [1:100, 1:3] 1 1 1 1 1 1 1 1 1 1 ...
str(dataset04$lesionWeight)
#>  num [1:100, 1:3] 1 1 1 1 1 1 1 1 1 1 ...
```

- Note that the first index of both arrays is the case index for the 100 abnormal cases in this dataset.
- With finite number of cases the empirical operating characteristic (or for that matter any fitted operating characteristic) will have sampling variability as in the following example.

15.4 The empirical wAFROC

```
p <- PlotEmpiricalOperatingCharacteristics(dataset04, opChType = "wAFROC")
p$Plot
```

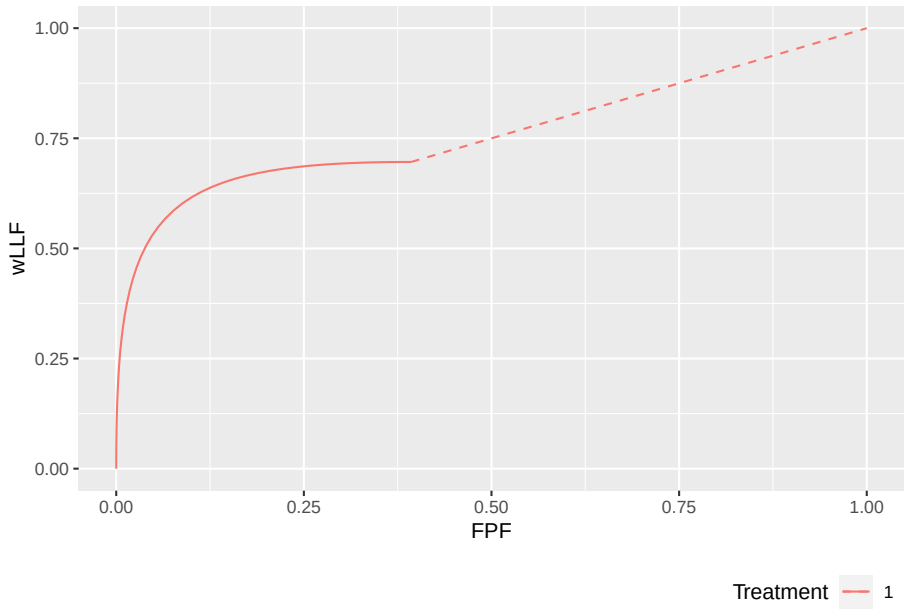


- The piecewise linear nature of the plot, with sharp breaks, indicates that this is due to a finite dataset.
- In contrast the following code shows a smooth plot, because it is a model *predicted* plot.

15.5 The predicted wAFROC

```
## Following example is for mu = 2, lambda = 1, nu = 0.6. 20% of the diseased
## cases have a single lesion, 40% have two lesions, 10% have 3 lesions,
## and 30% have 4 lesions.
lesDistr <- rbind(c(1, 0.2), c(2, 0.4), c(3, 0.1), c(4, 0.3))

## On cases with one lesion the weights are 1, on cases with 2 lesions the weights
## are 0.4 and 0.6, on cases with three lesions the weights are 0.2, 0.3 and 0.5, and
## on cases with 4 lesions the weights are 0.3, 0.4, 0.2 and 0.1:
lesWghtDistr <- rbind(c(1, 1.0, -Inf, -Inf, -Inf),
                      c(2, 0.4,  0.6, -Inf, -Inf),
                      c(3, 0.2,  0.3,  0.5, -Inf),
                      c(4, 0.3,  0.4, 0.2,  0.1))
p <- PlotRsmOperatingCharacteristics(mu = 2, lambda = 1, nu = 0.6, OpChType = "wAFROC",
                                     lesDistr = lesDistr, lesWghtDistr = lesWghtDistr,
```

```
                                        legendPosition = "bottom", nlfRange = c(0, 1), l
p$wAFROCPlot
```



15.6 The distribution of number of lesions and weights

```
lesDistr
#>      [,1] [,2]
#> [1,]    1  0.2
#> [2,]    2  0.4
#> [3,]    3  0.1
#> [4,]    4  0.3
lesWghtDistr
#>      [,1] [,2] [,3] [,4] [,5]
#> [1,]    1  1.0 -Inf -Inf -Inf
#> [2,]    2  0.4  0.6 -Inf -Inf
#> [3,]    3  0.2  0.3  0.5 -Inf
#> [4,]    4  0.3  0.4  0.2  0.1
```

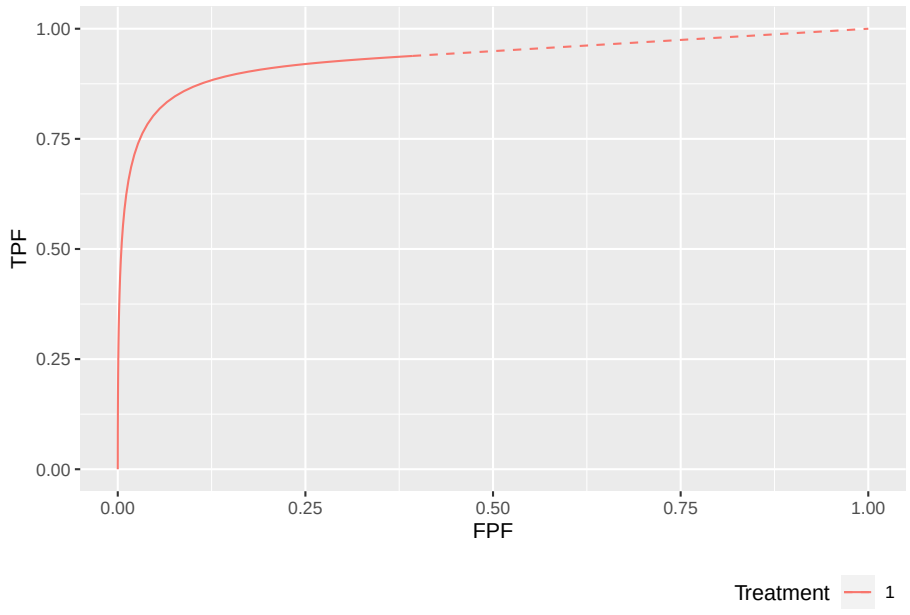
- The second column of `lesDistr` specifies the fraction of diseased cases with the number of lesions specified in the first column.

- The first column of `lesWghtDistr` is a copy of the first column of `lesDistr`. The remaining non-`-Inf` entries are the weights.
- For cases with 1 lesion, the weight is 1.
- For cases with 2 lesions, the first lesion has weight 0.4 and the second lesion has weight 0.6, which sum to unity.
- For cases with 3 lesions, the respective weights are 0.2, 0.3 and 0.5, which sum to unity.
- For cases with 4 lesions, the respective weights are 0.3, 0.4, 0.2 and 0.1, which sum to unity.

15.7 Other operating characteristics

- By changing `OpChType` one can generate other operating characteristics.
- Note that lesiion weights argument is not needed for ROC curves. It is only needed for `wAFROC` and `wAFROC1` curves.

```
lesDistr <- rbind(c(1, 0.2), c(2, 0.4), c(3, 0.1), c(4, 0.3))
p <- PlotRsmOperatingCharacteristics(mu = 2, lambda = 1, nu = 0.6, OpChType = "ROC",
                                     lesDistr = lesDistr,
                                     legendPosition = "bottom")
p$ROCPlot
```



15.8 Summary

15.9 References

Chapter 16

Improper ROCs

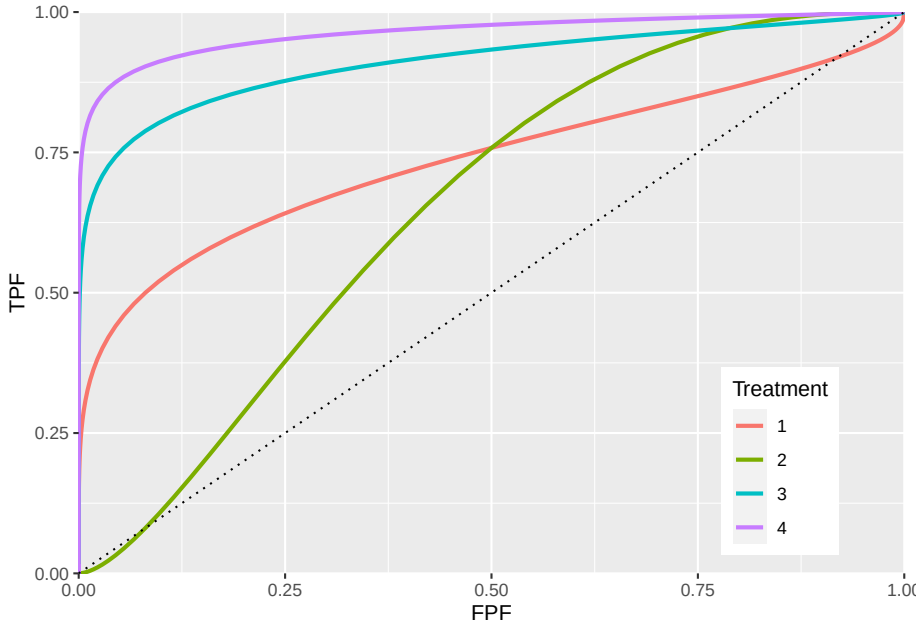
16.1 The binormal model

The binormal model has two parameters, `a` and `b`. The signal (or diseased cases) distribution has unit standard deviation. The noise (or non-diseased cases) distribution has standard deviation `b`. The `a` parameter is the separation of the two distributions.

16.2 Improper ROCs

Binormal model fits invariably lead to ROC curves that inappropriately cross the chance diagonal, leading to a prediction of a region of the ROC curve where performance is worse than chance, even for expert observers. By convention, such curves are termed *improper*. This vignette illustrates improper ROCs predicted by the binormal model.

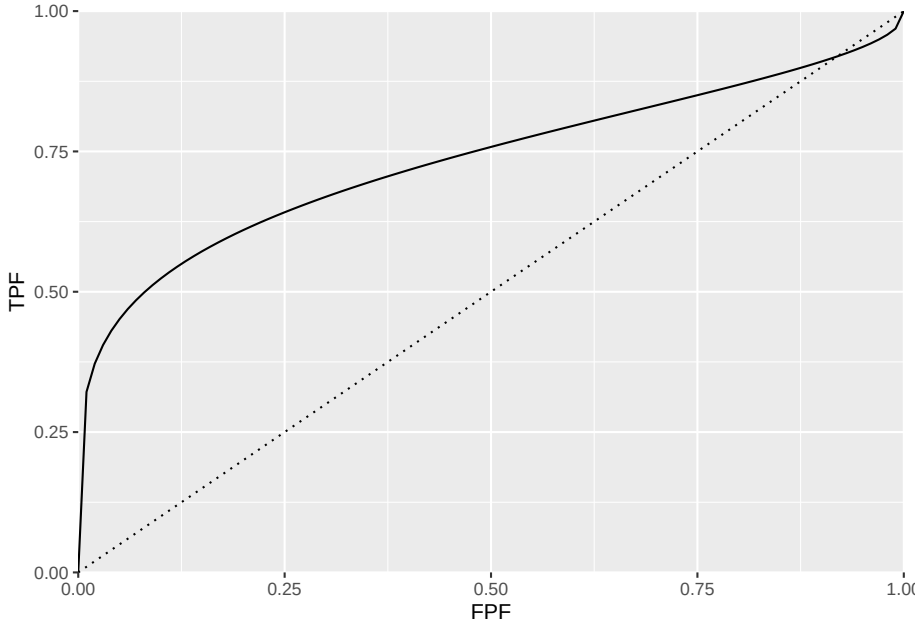
```
  aArray <- c(0.7, 0.7, 1.5, 2)
  bArray <- c(0.5, 1.5, 0.5, 0.5)
  chance_diag <- data.frame(x = c(0,1), y = c(0,1))
  p <- PlotBinormalFit(aArray, bArray) +
    scale_x_continuous(expand = c(0, 0)) +
    scale_y_continuous(expand = c(0, 0)) +
    theme(legend.position = c(0.85, 0.2))
p <- p + geom_line(data = chance_diag, aes(x = x, y = y), linetype="dotted")
print(p)
```

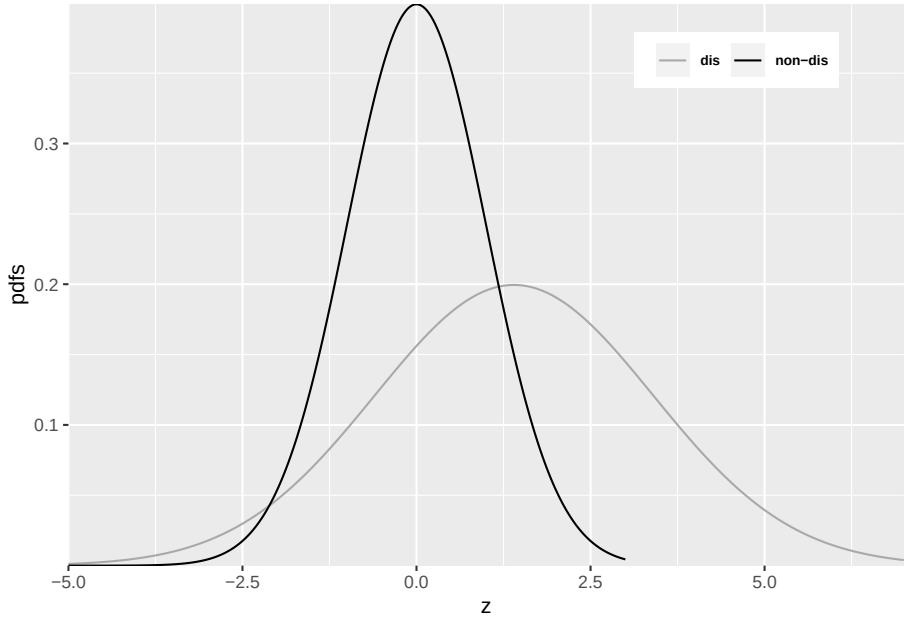


The red plot is the clearest example of an improper ROC. This type of curve occurs whenever `b < 1`. The chance line crossing near the upper right corner, around (0.919,0.919), and the fact that the ROC curve must eventually reach (1, 1) implies the curve must turn upwards as one approaches (1, 1), thereby displaying a "hook". Whenever `b != 1` the hook is there, regardless of whether it is easily visible or not. If `b < 1` the hook is near the upper right corner. If `b > 1` the hook is near the origin (see green line, corresponding to `b = 1.5`). With increasing `a` the hook is less prominent (blue line corresponding to `a = 1.5, b = 0.5` and purple line corresponding to `a = 2, b = 0.5`). But it is there.

16.3 Reason for improper ROCs

The reason for the "hook"" becomes apparent upon examination of the pdfs.






```
#> a =  0.7 , b =  0.5
```

Since `b < 1`, the diseased *pdf* is broader and has a lower peak (since the integral under each distribution is unity) than the non-diseased pdf. Sliding an imaginary threshold to the left, starting from the extreme right, one sees that initially, just below `z = 7`, the diseased distribution starts being *"picked up"* while the non-diseased distribution is not *"picked up"*, causing the ROC to start with infinite slope near the origin (because TPF is increasing while FPF is not). Around `z = 2.5` the non-diseased distribution starts being *"picked up"*, causing the ROC slope to decrease. Around `z = -3`, almost all of the non-diseased distribution has been *"picked up"* which means FPF is near unity, but since not all of the broader diseased distribution has been *"picked up"*, TPF is less than unity. Here is a region where `TPF < FPF`, meaning *the operating point is below the chance diagonal*. As the threshold is lowered further, TPF continues to increase, as the rest of the diseased distribution is *"picked up"* while FPF stays almost constant at unity. In this region, the ROC curve is approaching the upper right corner with almost infinite slope (because TPF is increasing but FPF is not).

Chapter 17

Degenerate datasets in the binormal model

17.1 Two helper functions

17.2 Degenerate datasets

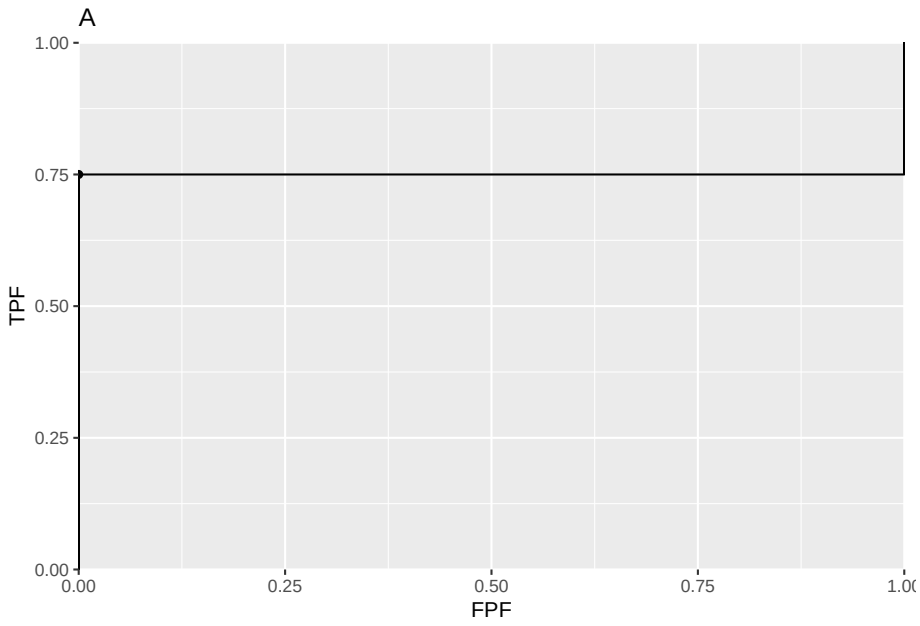
Metz defined binormal degenerate data sets as those that result in exact-fit binormal ROC curves of inappropriate shape consisting of a series of horizontal and/or vertical line segments in which the ROC "curve" crosses the chance line. The crossing of the chance line occurs because the degenerate data sets can be fitted exactly by infinite or zero values for the model slope parameter `b`, and infinite values for the decision thresholds, or both.

17.3 Understanding degenerate datasets

To understand this, consider that the non-diseased distribution is a Dirac delta function centered at zero (by definition such a function integrates to unity) and the unit variance diseased distribution is centered at 0.6744898. In other words this binormal model is characterized by `a = 0.6744898` and `b = 0`. What is the expected ROC curve? As the threshold ζ is moved from the far right, gradually to the left, TPF will increase but FPF is stuck at zero until the threshold reaches zero. Just before reaching this point, the coordinates of the ROC operating point are (0, 0.75). The 0.75 is due to the fact that `z = 0` is -0.6744898 units relative to the center of the diseased distribution, so the area under the diseased distribution below `z = 0` is 0.25. Since `pnorm` is the probability *below* the threshold, TPF must be its complement, namely 0.75.

This explains the operating point (0,0.75), which lies on the y-axis. As the threshold crosses the zero-width delta function, FPF shoots up from 0 to 1, but TPF stays constant. Therefore, the operating point has jumped from (0, 0.75) to (1, 0.75). When the threshold is reduced further, the operating point moves up vertically, along the right side of the ROC plot, until the threshold is so small that virtually all of diseased distribution exceeds it and the operating point reaches (1, 1). The ROC curve is illustrated in plot A.

```
plotOP <- data.frame(FPF = 0, TPF = 0.75)
a <- 0.6744898; b <- 0
plotCurve <- BMPoints(a, b)
figA <- ggplot(mapping = aes(x = FPF, y = TPF)) +
  geom_line(data = plotCurve) +
  geom_point(data = plotOP)  +
  scale_x_continuous(expand = c(0, 0)) +
  scale_y_continuous(expand = c(0, 0)) +
  ggtitle("A")
print(figA)
```

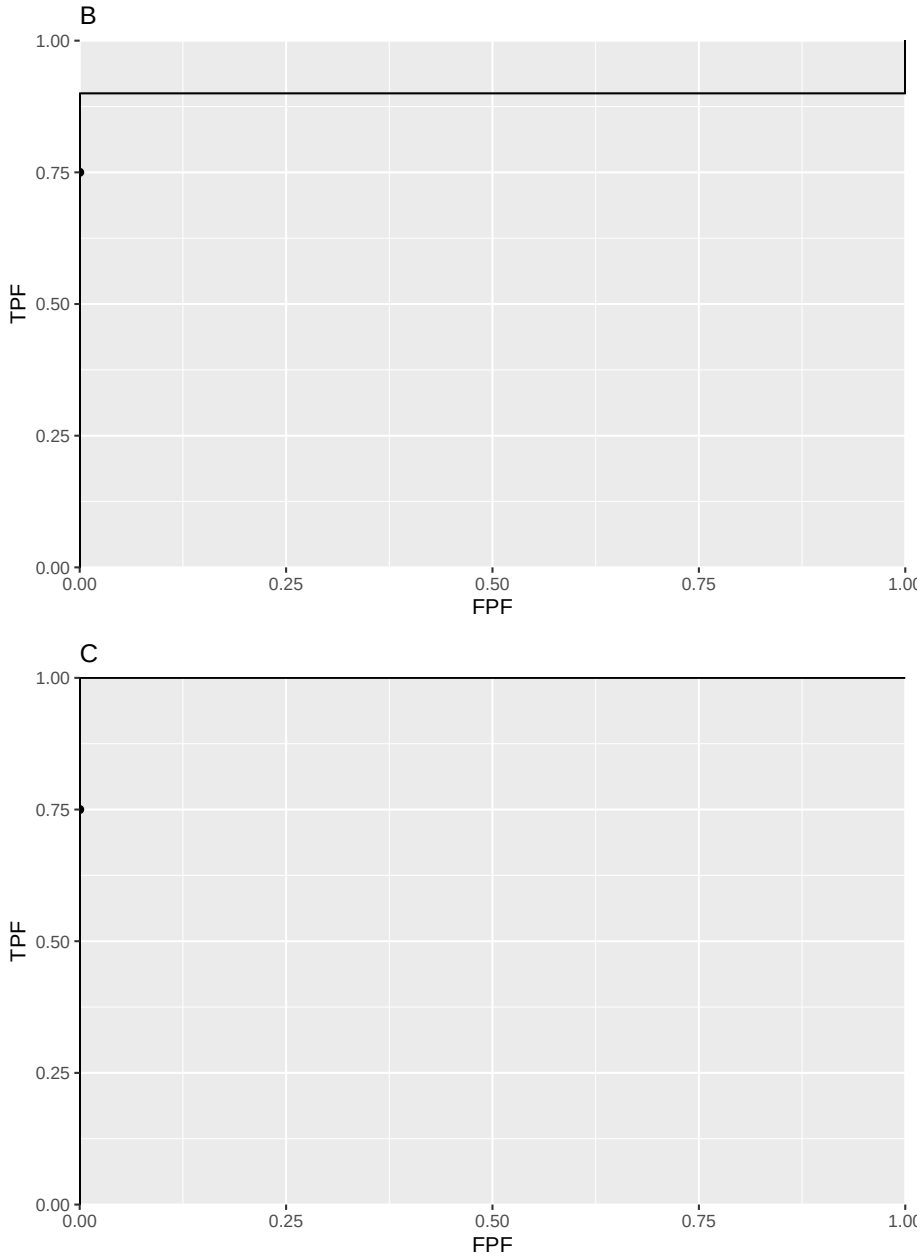


This is an extreme example of an ROC curve with a "hook". If the data is such that the only operating point provided by the observer is (0,0.75) then this curve will be an exact fit to the operating point.

17.4 The exact fit is not unique

Actually, given one operating point (0, 0.75) the preceding fit is not even unique. If the diseased distribution is shifted appropriately to the right of its previous position, and one can determine the necessary value of a, then the ROC curve will shoot upwards through the operating point (0, 0.75) to (0, 0.9), as in plot B, before proceeding horizontally to (1, 0.9) and then completing the curve to (1, 1). If the diseased distribution is shifted well to the right, i.e., a is very large, then the ROC curve will shoot upwards past the operating point, as in plot C, all the way to (0,1) before proceeding horizontally to (1, 1).

```
a <- 1.281552; b <- 0
plotCurve <- BMPoints(a, b)
figB <- ggplot(mapping = aes(x = FPF, y = TPF)) +
  geom_line(data = plotCurve) +
  geom_point(data = plotOP)  +
  scale_x_continuous(expand = c(0, 0)) +
  scale_y_continuous(expand = c(0, 0)) +
  ggtitle("B")

a <- Inf; b <- 0
plotCurve <- BMPoints(a, b)
figC <- ggplot(mapping = aes(x = FPF, y = TPF)) +
  geom_line(data = plotCurve) +
  geom_point(data = plotOP)  +
  scale_x_continuous(expand = c(0, 0)) +
  scale_y_continuous(expand = c(0, 0)) +
  ggtitle("C")
print(figB);print(figC)
```



All of these represent exact fits to the observed operating point, with `b = 0` and different values of `a`. Not one of them is reasonable.

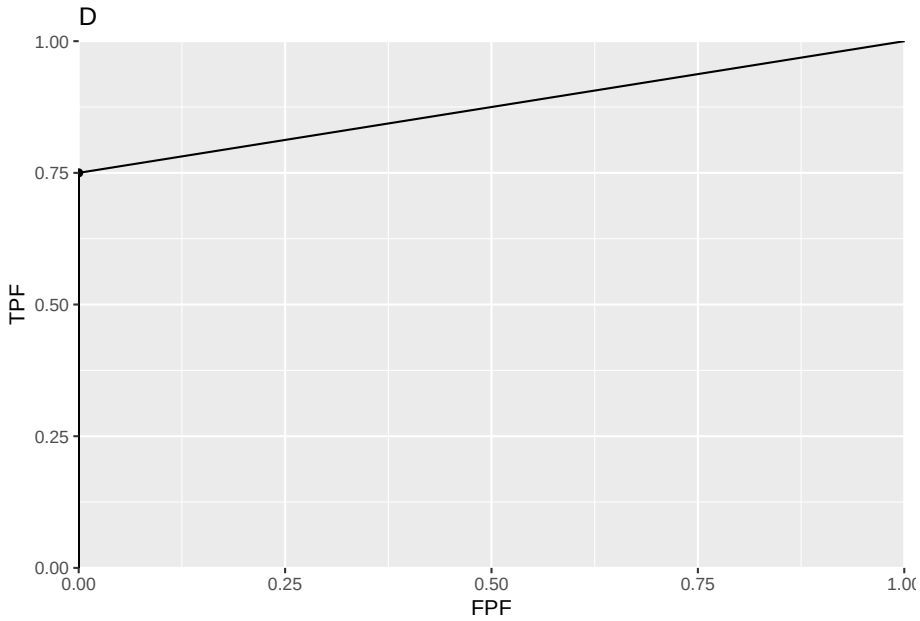
17.5 Comments on degeneracy

Degeneracy occurs if the observer does not provide any interior operating points. So why worry about it? Perhaps one has a non-cooperating observer, who is not heeding the instructions to *spread the ratings, use all the bins*. A simple example shows that the observer could if fact be cooperating fully and is still unable to provide any interior data points. Consider 100 diseased cases consisting of 75 easy cases and 25 difficult ones and 100 easy non-diseased cases. The observer is expected to rate the 75 easy diseased cases as *fives*, the difficult ones as *ones* and the 100 non-diseased cases are rated *ones*. No amount of coaxing *please, please spread your ratings* is going to convince this observer to rate with twos, threes and fours any of the 75 easy diseased cases. If the cases are obviously diseased, and that is what is meant by *easy cases*, they are supposed to be rated fives: *definitely diseased*. Forcing them to rate some of them as *probably diseased* or *possibly diseased* would be irrational and guilty of bending the reading paradigm to fit the convenience of the researcher (early in his research career, the author used to believe in the existence of non-cooperating observers, so Metz's advice to *spread the ratings* did not seem unreasonable at that time).

17.6 A reasonable fit to the degenerate dataset

If the dataset yields a single operating point (0, 0.75), what is a reasonable ROC plot? There is a theorem that given an observed operating point, the line connecting that point to (1, 1) represents a lower bound on achievable performance by the observer. The observer using a guessing mechanism to classify the remaining cases achieves the lower bound. Here is an explanation of this theorem. Having rated the 75 easy diseased cases as fives, the observer is left with 25 diseased cases and 100 non-diseased cases, all of which appear definitely non-diseased to the observer. Suppose the observer randomly rates 20% of the remaining cases as fours. This would pick up five of the actually diseased cases and 20 non-diseased ones. Therefore, the total number of diseased cases rated four or higher is 80, and the corresponding number of non-diseased cases is 20. The new operating point of the observer is (0.20, 0.80). Now, one has two operating points, the original one on the y-axis at (0, 0.75) and an interior point (0.20, 0.80). Next, instead of randomly rating 20% of the remaining cases as fours, the observer rates 40% of them as fours, then the interior point would have been (0.40, 0.85). The reader can appreciate that simply by increasing the fraction of remaining cases that are randomly rated fours, the observer can move the operating point along the straight line connecting (0, 0.75) and (1, 1), as in plot D. Since a guessing mechanism is being used, this must represent a lower bound on performance. The resulting ROC curve is proper and the net AUC = 0.875.

```
mu <- Inf; alpha <- 0.75
plotCurve <- CBMPoints(mu, alpha)
figD <- ggplot(mapping = aes(x = FPF, y = TPF)) +
  geom_line(data = plotCurve) +
  geom_point(data = plotOP)  +
  scale_x_continuous(expand = c(0, 0)) +
  scale_y_continuous(expand = c(0, 0)) +
  ggtitle("D")
print(figD)
```



For this dataset this is in fact the fit yielded by the contaminated binormal model (CBM) and the radiological search model (RSM). Why should one select the lowest possible performance consistent with the data? Because it yields a *unique* value for performance: any higher performance would not be unique.

Chapter 18

Proper ROCs

18.1 Helper functions

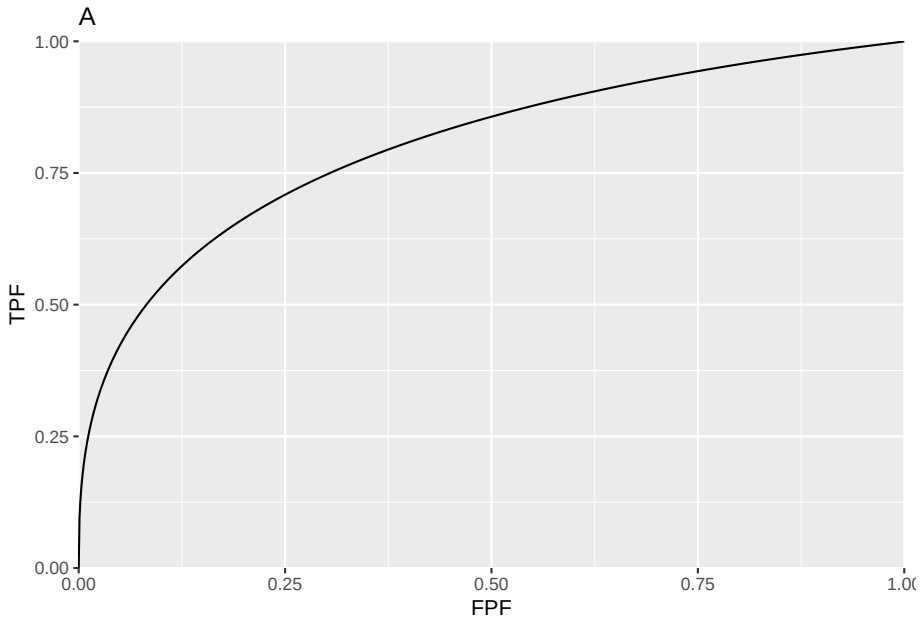
18.2 Definitions of PROPROC parameters in terms of binormal model parameters

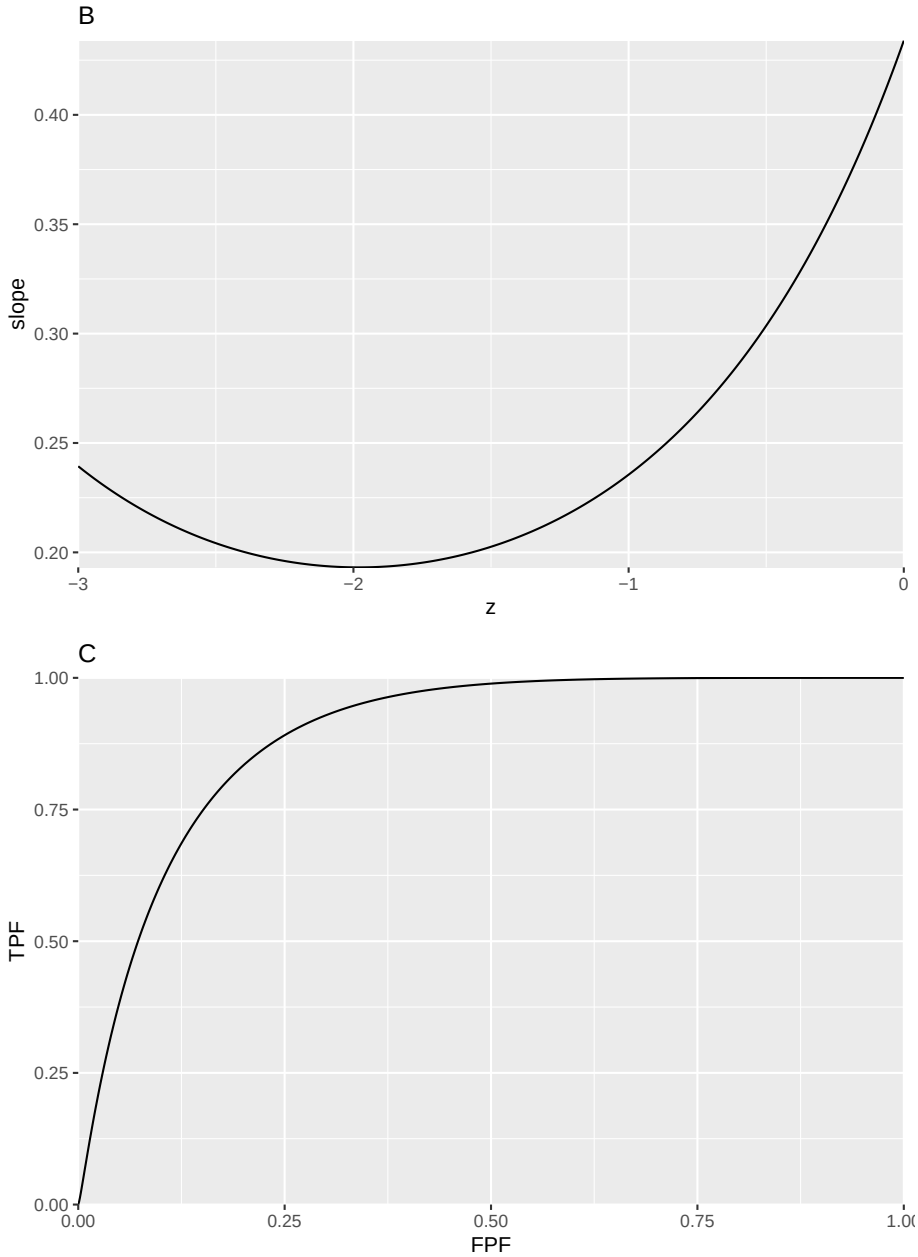
TBA

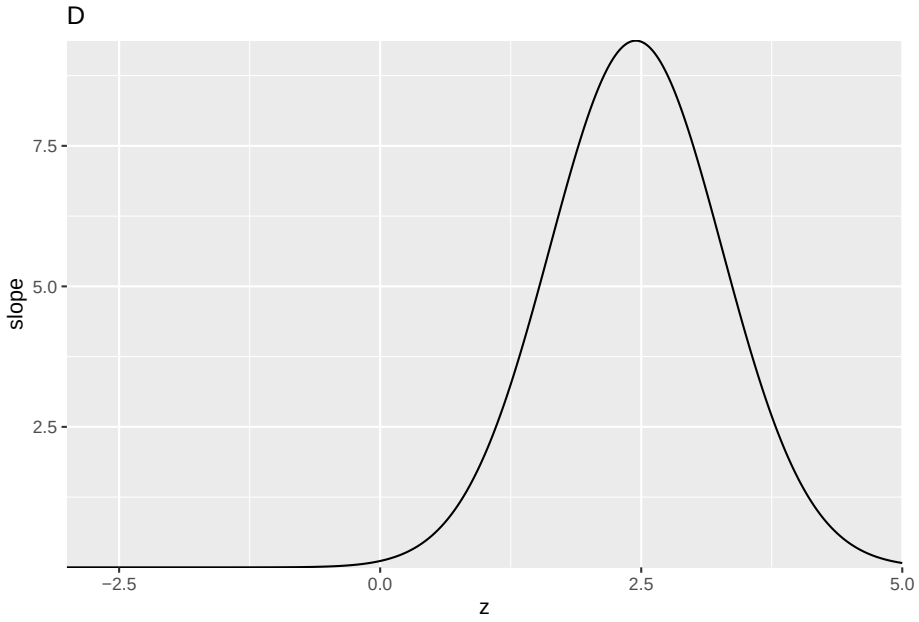
18.3 Main code and output

```
c1Arr <-   c(-0.1322804, 0.2225588); daArr  <-  c(1.197239, 1.740157)
myLabel <- c("A", "B", "C", "D")
myLabelIndx <- 1
for (i in 1:2)
{
  c1 <- c1Arr[i]
  da <- daArr[i]
  ret <- Transform2ab(da, c1)
  a <- ret$a;b <- ret$b
  if (i == 1) z <- seq(-3, 0, by = 0.01) # may need to adjust limits to view detail of slope plot
  if (i == 2) z <- seq(-3, 5, by = 0.01) # may need to adjust limits to view detail of slope plot

  FPF <- seq(0.0, 1, 0.001)
  TPF <- rocY(FPF, a, b)

  rocPlot <- data.frame(FPF = FPF, TPF = TPF)
```

```
  plotRoc <- ggplot(rocPlot, aes(x = FPF, y = TPF)) +
    geom_line()  +
   scale_x_continuous(expand = c(0, 0)) +
    scale_y_continuous(expand = c(0, 0))   +
    ggtitle(myLabel[myLabelIndx]);myLabelIndx <- myLabelIndx + 1

  slope <-b*dnorm(a-b*z)/dnorm(-z) # same as likelihood ratio

  slopePlot <- data.frame(z = z, slope = slope)
  p <- ggplot(slopePlot, aes(x = z, y = slope)) +
    geom_line()  +
    scale_x_continuous(expand = c(0, 0)) +
    scale_y_continuous(expand = c(0, 0))  +
    ggtitle(myLabel[myLabelIndx]);myLabelIndx <- myLabelIndx + 1
  print(plotRoc);print(p)
}
```




B
slope
0.40
0.35
0.30
0.25
0.20
−3
−2
−1
0
z
C
TPF
1.00
0.75
0.50
0.25
0.00
0.00
0.25
0.50
0.75
1.0
FPF



18.4 Discussion

Plot A is for `c1` = -0.1322804, `da` = 1.197239 while plot C is for `c1` = 0.2225588, `da` = 1.740157. Plots B and D are the corresponding slope plots as functions of the binormal model z-sample. In plot A, the slope is infinite near the origin and the curve approaches the upper-right corner with finite slope. The situation is reversed in plot C where the slope is finite near the origin and the curve approaches the upper-right corner with zero slope.

These two readers are from a clinical dataset, `dataset01`. Highest rating inferred ROC data from original FROC data, were analyzed by PROPROC and the resulting parameter values are coded here. They were chosen as they demonstrate key differences in the shapes of proper ROC plots. Plot A corresponds to a negative value of `c1`, which implies `b < 1`. The slope of the proper ROC is infinite near the origin and approaches a positive constant near the upper right corner of the ROC. Plot C is for a positive value of `c1`, i.e., for `b > 1`. Now the slope of the proper ROC is finite near the origin and approaches zero near the upper right corner.

Considering plot D, as one "cuts" the slope axis horizontally with a sliding threshold, starting with very high values and moving downwards, the slope of the ROC curve starts at the origin with a large but finite value. This corresponds to the peak in plot D. Above the peak, there are no solutions for `z`. The slope decreases monotonically to zero, corresponding to the flattening out of the slope

at zero for `z ~ -2`.

The two values of `z` corresponding to each cut implies, of course, that the binormal model based proper algorithm has to do a lot of bookkeeping, since each horizontal cut splits the decision axis into 3 regions. One can think of shrinking each of plots B & D horizontally to zero width, and all that remains is the slope axis with a thick vertical line superimposed on it, corresponding to the horizontally collapsed curves. In plot B the vertical line extends from positive infinity down to about 0.1, and represents the range of decision variable samples encountered by the observer on the likelihood ratio scale. In plot D the vertical line extends from a finite value (~9.4) to zero. For the stated binormal model parameters values outside of these ranges are not possible.

Chapter 19

Metz Eqn36 numerical check

19.1 Helper functions

19.2 Main code and output

```
npts <-  10000
for (i in 1:2) {
  for (j in 1:5) {
    C  <-  c1[i,j]
    da  <-  d_a1[i,j]
    ret <- GetLimits(da,C)
    LL <- ret$LL;UL <- ret$UL
    vc  <-  seq (LL, UL, length.out = npts)
    TPF  <-  TruePositiveFraction (vc, da, C)
    FPF <- FalsePositiveFraction (vc, da, C)
    FPF <- rev(FPF);TPF <- rev(TPF)
    df2 <- data.frame(FPF = FPF, TPF = TPF)
    # do integral numerically
    numAuc <- trapz(FPF, TPF)
    # Implement Eqn. 36 from Metz-Pan paper
    rho <- -(1-C^2)/(1+C^2);sigma <- rbind(c(1, rho), c(rho, 1))
    lower <- rep(-Inf,2);upper <- c(-da/sqrt(2),0)
    aucProproc <- pnorm(da/sqrt(2)) + 2 * pmvnorm(lower, upper, sigma = sigma)
    aucProproc <-  as.numeric(aucProproc)
    cat("i = ", i,"j = ", j,"C = ", C, ", da = ", da, "aucProproc =", aucProproc, "Norm. Diff. =
```

```
  }
}
#> i =  1 j =  1 C =  -0.1322804 , da =  1.197239 aucProproc = 0.8014164 Norm. Diff. =
#> i =  1 j =  2 C =  -0.08696513 , da =  1.771176 aucProproc = 0.8947898 Norm. Diff. =
#> i =  1 j =  3 C =  -0.1444419 , da =  1.481935 aucProproc = 0.8526605 Norm. Diff. =
#> i =  1 j =  4 C =  0.08046016 , da =  1.513757 aucProproc = 0.8577776 Norm. Diff. =
#> i =  1 j =  5 C =  0.2225588 , da =  1.740157 aucProproc = 0.8909392 Norm. Diff. =
#> i =  2 j =  1 C =  -0.08174248 , da =  0.6281251 aucProproc = 0.6716574 Norm. Diff.
#> i =  2 j =  2 C =  0.04976448 , da =  0.9738786 aucProproc = 0.7544739 Norm. Diff. =
#> i =  2 j =  3 C =  -0.1326126 , da =  1.155871 aucProproc = 0.7931787 Norm. Diff. =
#> i =  2 j =  4 C =  0.1182226 , da =  1.620176 aucProproc = 0.8740274 Norm. Diff. =
#> i =  2 j =  5 C =  0.0781033 , da =  0.8928816 aucProproc = 0.7360989 Norm. Diff. =
```

19.3 Discussion

Note the close correspondence between the formula, Eqn. 36 in the Metz-Pan paper and the numerical estimate. As a historical note, Eqn. 31 and Eqn. 36 (they differ only in parameterizations) in the referenced publication are provided without proof – it was probably obvious to Prof Metz or he wanted to leave it to us "mere mortals" to figure it out, as a final parting gesture of his legacy. The author once put a significant effort into proving it and even had a bright graduate student from the biostatistics department work on it to no avail. The author has observed that these equations always yield very close to the numerical estimates, to within numerical precisions, so the theorem is correct empirically, but he has been unable to prove it analytically. It is left as an exercise for a gifted reader to prove/disprove these equations.

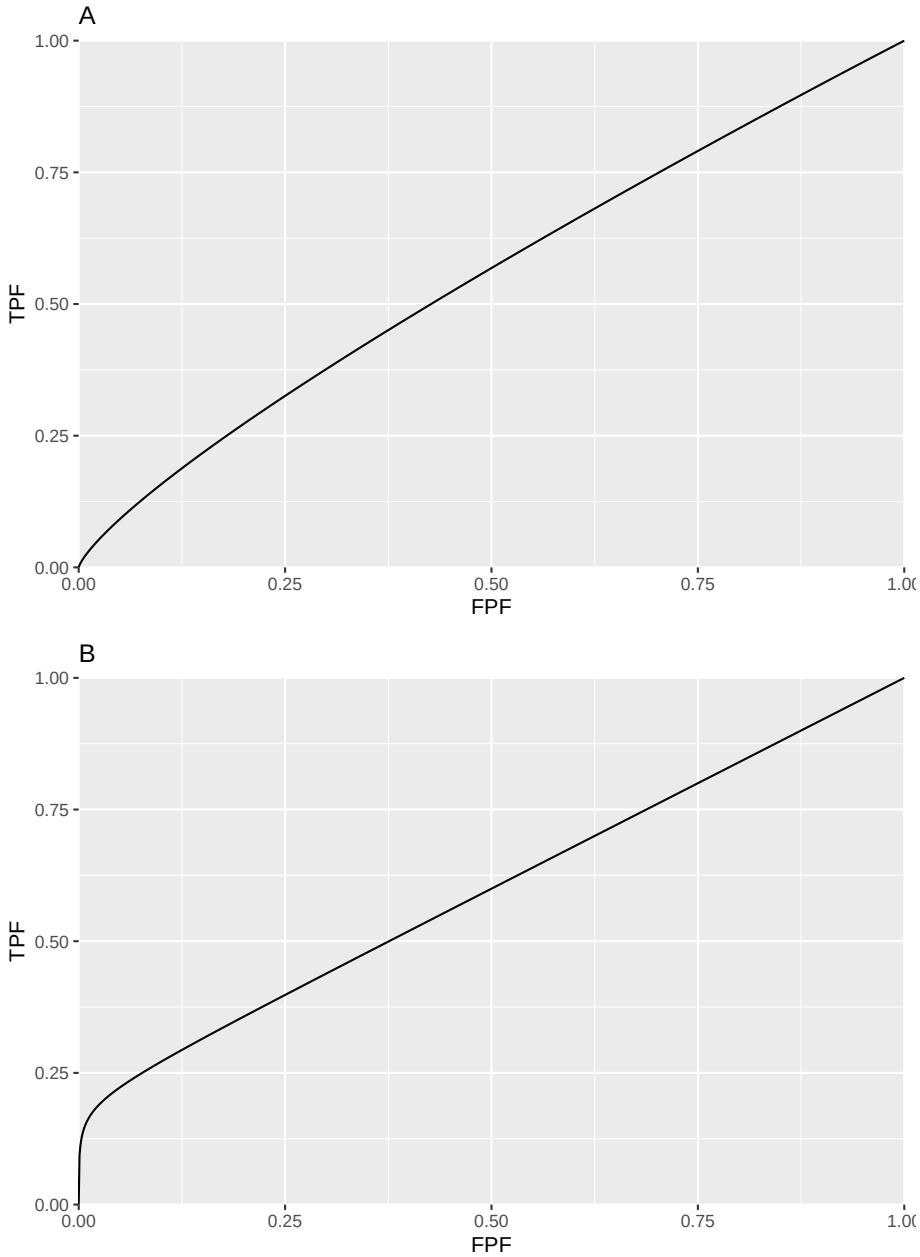
Chapter 20

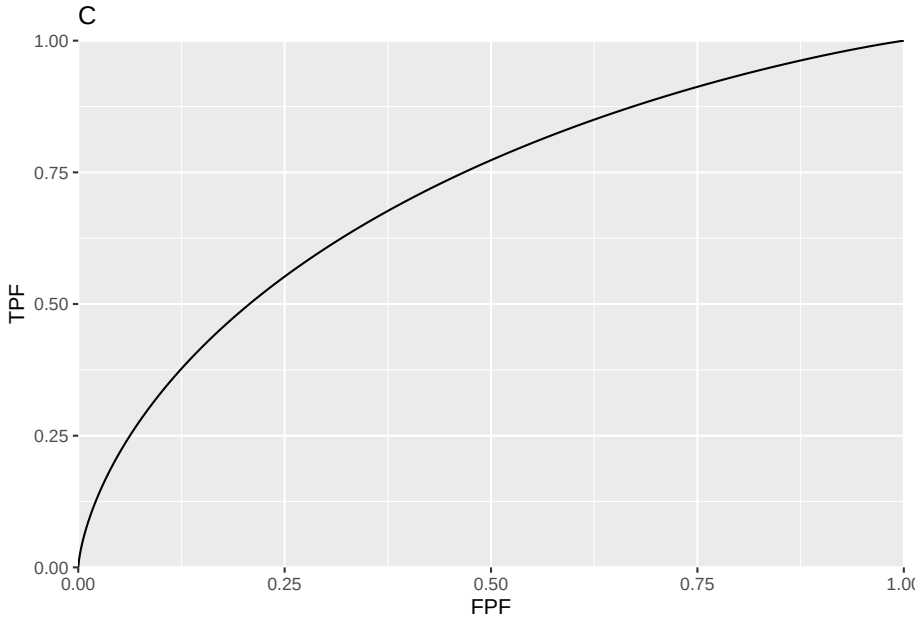
CBM Plots

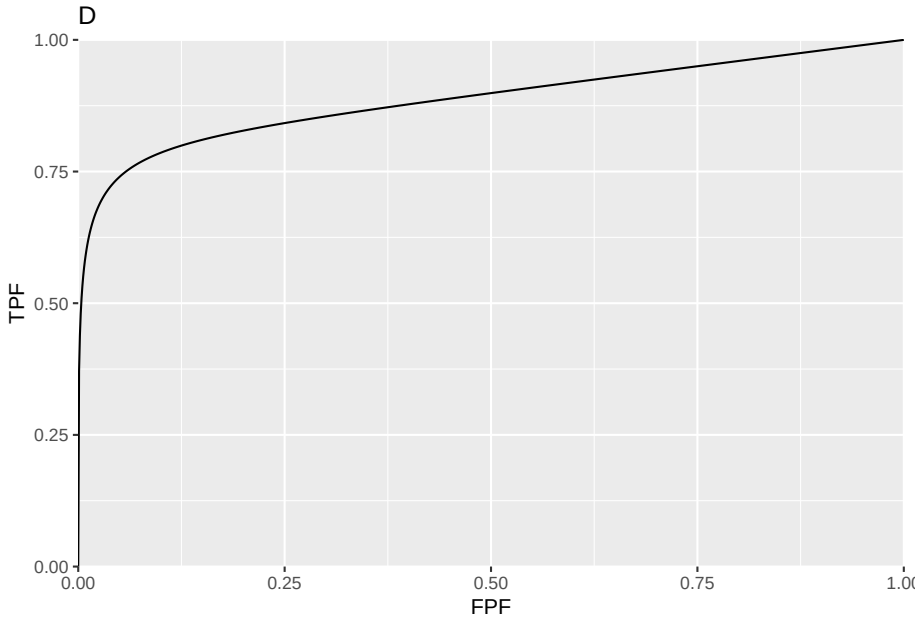
20.1 Helper functions

20.2 Main code and output

```
#> Fig. A : mu =  1 , alpha =  0.2
#> Fig. B : mu =  3 , alpha =  0.2
#> Fig. C : mu =  1 , alpha =  0.8
#> Fig. D : mu =  3 , alpha =  0.8
```


A
1.00
0.75
0.50
0.25
0.00
TPF
0.00
0.25
0.50
0.75
FPF
B
1.00
0.75
0.50
0.25
0.00
TPF
0.00
0.25
0.50
0.75
FPF


C
1.00
0.75
0.50
0.25
0.00
TPF
0.00
0.25
0.50
0.75
FPF

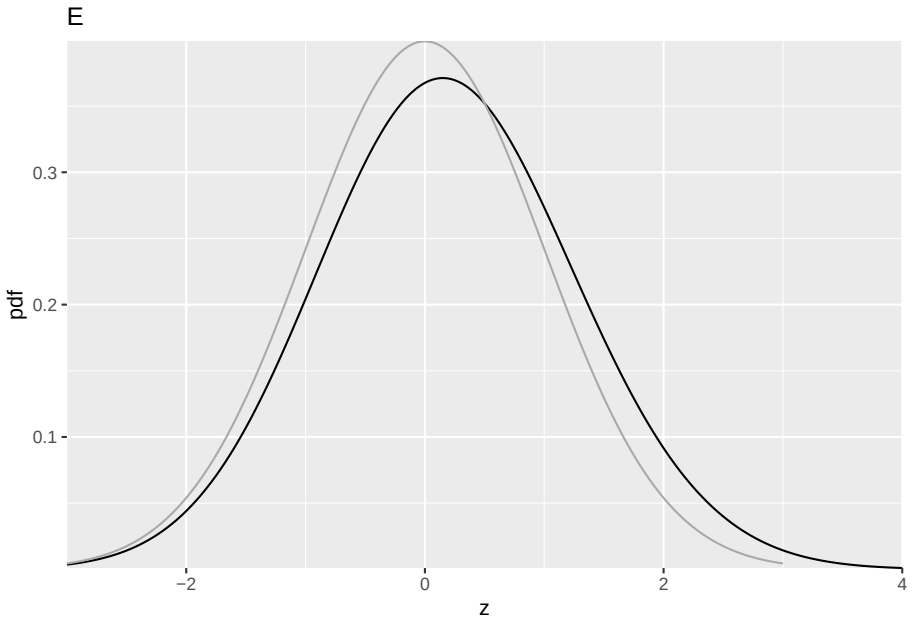

D
1.00
0.75
0.50
0.25
0.00
TPF
0.00
0.25
0.50
0.75
FPF

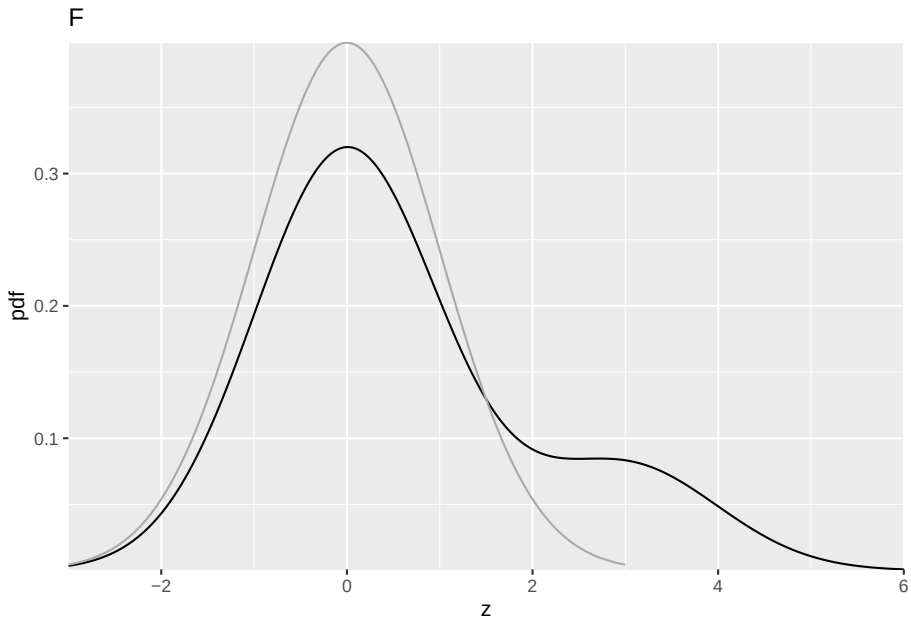
20.3 Comments

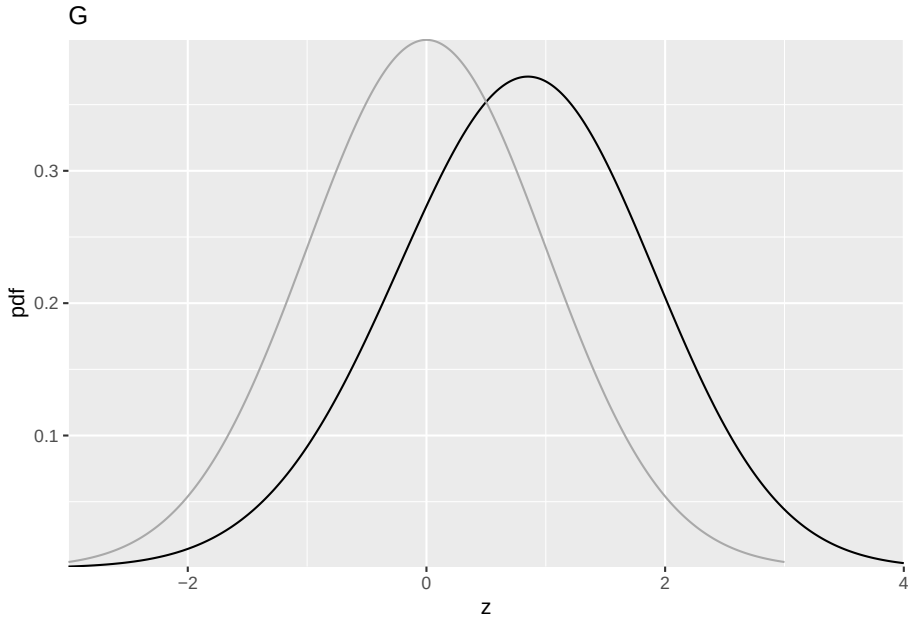
Plots A - D show ROC curves predicted by the CBM model; the corresponding values of the *mu* and *alpha* parameters are indicated above the plots. For small *mu* and/or *alpha* the curve approaches the chance diagonal, consistent with the notion that if the lesion is not visible, performance can be no better than chance level.

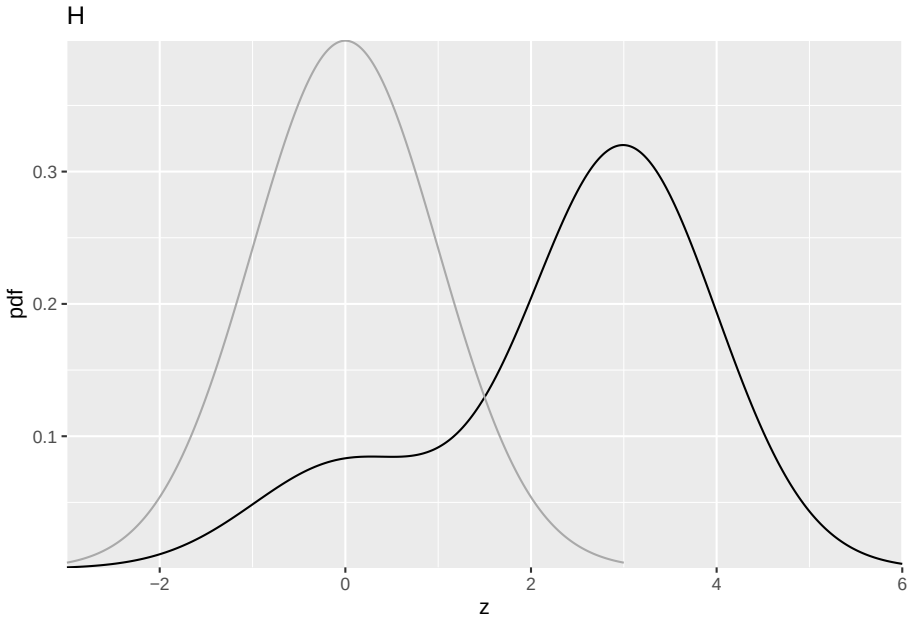
20.4 pdf plots

```
#> Fig. E : mu =  1 , alpha =  0.2
#> Fig. F : mu =  3 , alpha =  0.2
#> Fig. G : mu =  1 , alpha =  0.8
#> Fig. H : mu =  3 , alpha =  0.8
```




F
pdf
0.3
0.2
0.1
−2
0
2
4
6
z


G
pdf
0.3
0.2
0.1
−2
0
2
4
z

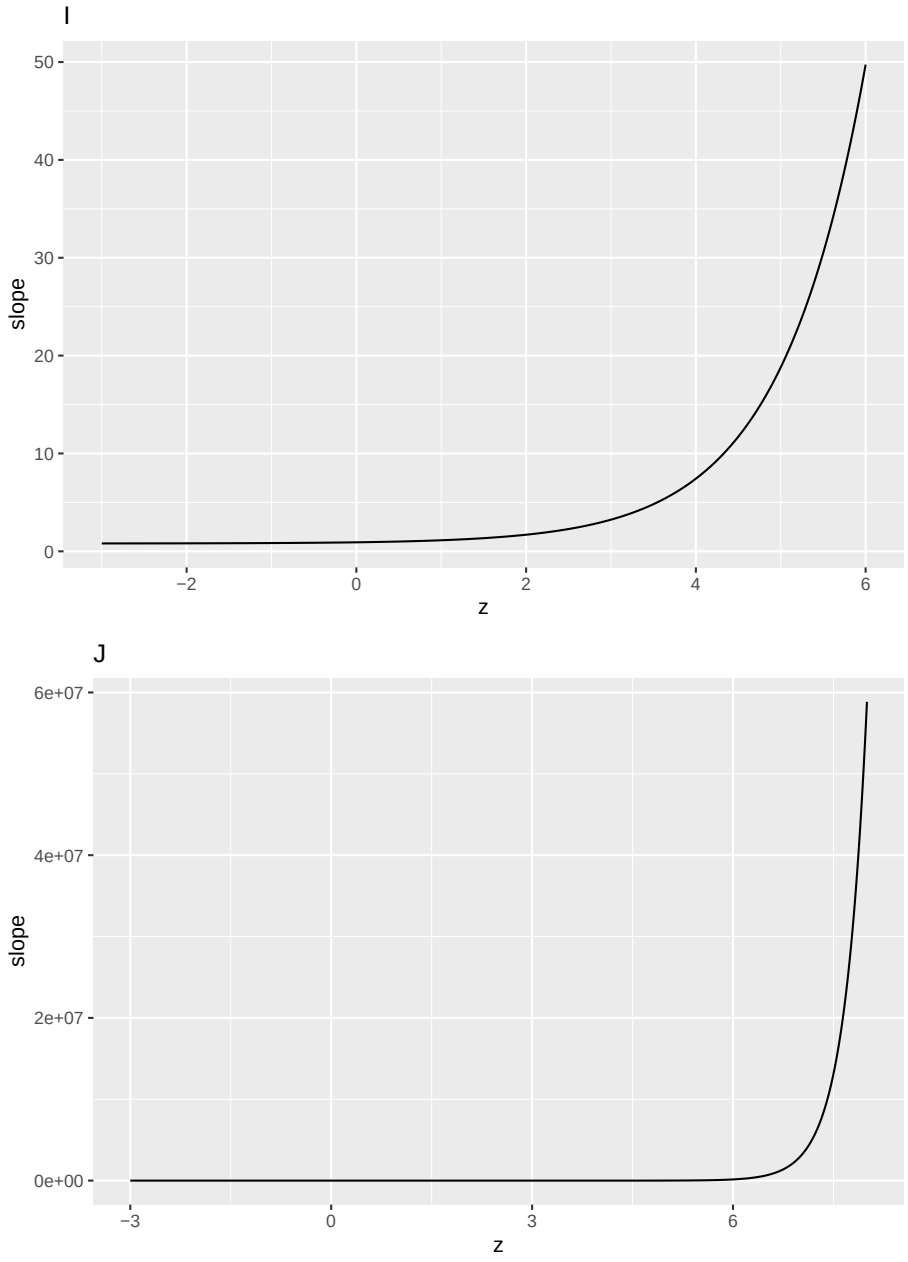


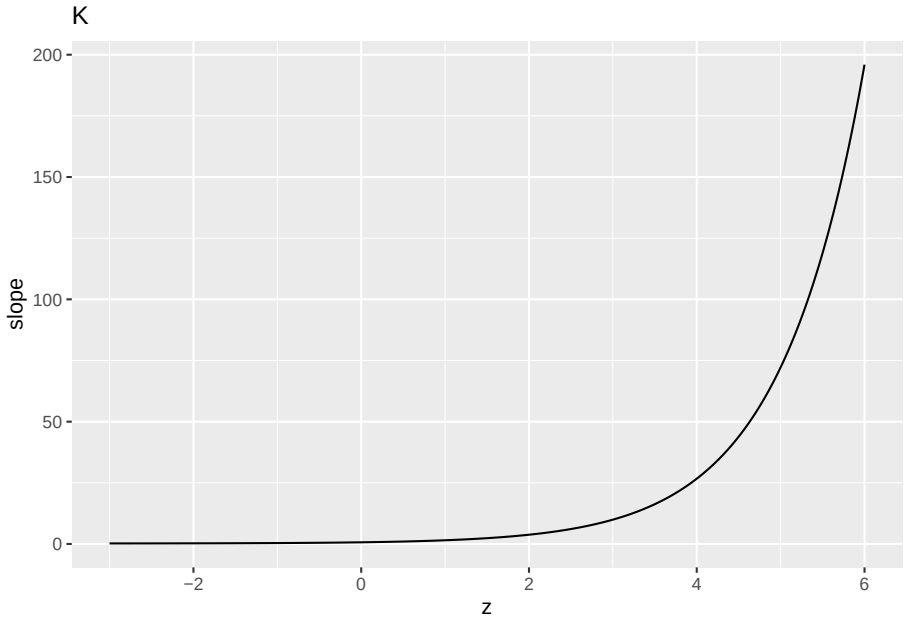
20.5 Comments

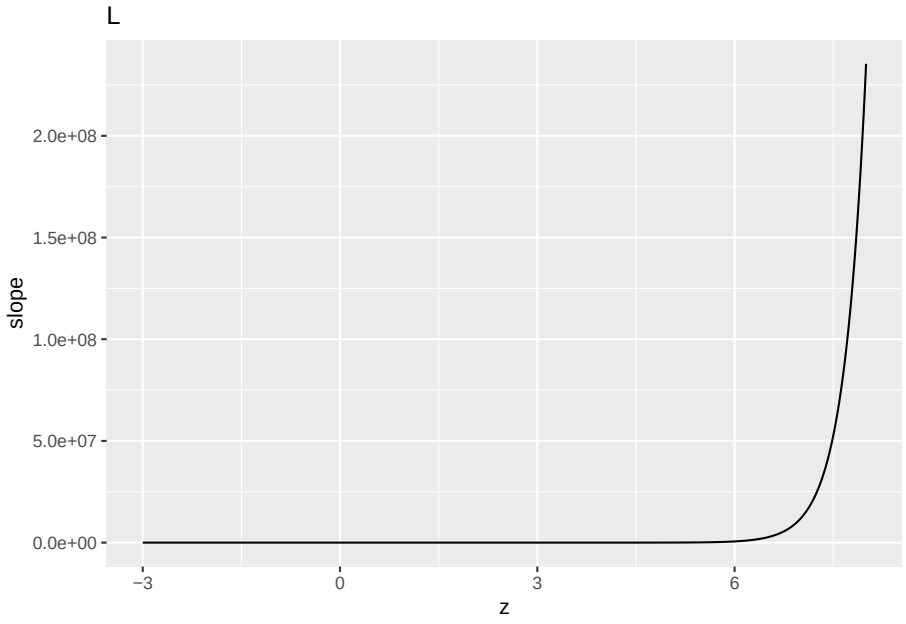
The dark line is the diseased distribution. The grey line is the non-diseased distribution. The bimodal diseased distribution is clearly evident in plots F and H.

20.6 likelihood ratio plots

```
#> Fig. I : mu =  1 , alpha =  0.2
#> Fig. J : mu =  3 , alpha =  0.2
#> Fig. K : mu =  1 , alpha =  0.8
#> Fig. L : mu =  3 , alpha =  0.8
```


I
50
40
30
20
10
0
slope
−2
0
2
4
6
z
J
6e+07
4e+07
2e+07
0e+00
slope
−3
0
3
6
z


K
200
150
100
50
0
slope
−2
0
2
4
6
z


L
2.0e+08
1.5e+08
1.0e+08
5.0e+07
0.0e+00
slope
−3
0
3
6
z

20.7 Comments

Close examination of the region near the flat part shows it does not plateau at zero; rather the minimum is at 1 - *alpha*, explaining the non-zero limiting slope of the predicted curve near (1, 1).

Chapter 21

ROI paradigm data

21.1 Introduction; this vignette is under construction!

- In the region-of-interest (ROI) paradigm (Obuchowski, 1997, @RN55) each case is regarded as consisting of Q_k ($Q_k \geq 1$) "quadrants" or "regions-of-interest" or ROIs, where k is the case index ($k = 1, 2, ..., K$) and K is the total number of cases (i.e., case-level non-diseased plus case-level diseased cases). Each ROI needs to be classified, by the investigator, as either ROI-level-non-diseased (i.e., it has no lesions) or ROI-level-diseased (i.e., it has at least one lesion). **Note the distinction between case-level and ROI-level truth states.** One can have ROI-level non-diseased regions in a case-level diseased case. A case-level diseased case must contain at least one ROI-level diseased region and a case-level non-diseased case cannot have any ROI-level diseased regions.
- The observer gives a single rating (in fact an ordered label) to each ROI, denoted R_{kr} ($r = 1, 2, ..., Q_k$). Here r is the ROI index and k is the case index. The rating 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 one or more lesions in the ROI.
- The ROI paradigm is not restricted to 4 or even a constant number of ROIs per case. That is the reason for the k subscript in Q_k.
- The ROI data structure is a special case of the FROC data structure, the essential difference being that the number of ratings per case is an a-priori known value, equal to Q_k.
- ROI-level non-diseased region ratings are stored in the `NL` field and ROI-level diseased region ratings are stored in the `LL` field.

- One can think of the ROI paradigm as similar to the FROC paradigm, but

with localization accuracy restricted to belonging to a region (one cannot distinguish multiple lesions within a region). Unlike the FROC paradigm, a rating *is required* for every ROI.

21.2 An example ROI dataset

An example simulated ROI dataset is included as `datasetROI`.

```
str(datasetROI)
#> List of 8
#>  $ NL          : num [1:2, 1:5, 1:90, 1:4] 0.95 0.927 0.556 0.805 1.421 ...
#>  $ LL          : num [1:2, 1:5, 1:40, 1:4] 1.57 2.31 2.3 2.34 2.34 ...
#>  $ lesionVector: int [1:40] 2 3 2 2 3 3 1 2 3 3 ...
#>  $ lesionID    : num [1:40, 1:4] 2 1 1 1 1 2 4 1 1 1 ...
#>  $ lesionWeight: num [1:40, 1:4] 0.5 0.333 0.5 0.5 0.333 ...
#>  $ dataType    : chr "ROI"
#>  $ modalityID  : Named chr [1:2] "1" "2"
#>   ..- attr(*, "names")= chr [1:2] "1" "2"
#>  $ readerID    : Named chr [1:5] "1" "2" "3" "4" ...
#>   ..- attr(*, "names")= chr [1:5] "1" "2" "3" "4" ...
datasetROI$NL[1,1,1,]
#> [1]  0.9498680 -0.0582497 -0.7763780  0.0120730
mean(datasetROI$NL[,,1:50,])
#> [1] 0.1014348
datasetROI$NL[1,1,51,]
#> [1] 1.01867 0.34710    -Inf    -Inf
datasetROI$lesionVector[1]
#> [1] 2
datasetROI$LL[1,1,1,]
#> [1] 1.56928 2.05945    -Inf    -Inf
x <- datasetROI$LL;mean(x[is.finite(x)])
#> [1] 1.815513
```

Examination of the output reveals that:

- This is a 2-treatment 5-reader dataset, with 50 non-diseased cases and 40 diseased cases, and $Q_k = 4$ for all k.
- For treatment 1, reader 1, case 1 (the 1st non-diseased case) the 4 ratings are 0.949868, -0.0582497, -0.776378, 0.012073. The mean of all ratings on non-diseased cases is 0.1014348.
- For treatment 1, reader 1, case 51 (the 1st diseased case) the NL ratings are 1.01867, 0.3471. There are only two finite values because this case

has two ROI-level-diseased regions, and 2 plus 2 makes for the assumed 4-regions per case. The corresponding `$lesionVector` field is 2.

- The ratings of the 2 ROI-level-diseased ROIs on this case are 1.56928, 2.05945. The mean rating over all ROI-level-diseased ROIs is 1.8155127.

21.3 The ROI Excel data file

- An Excel file in JAFROC format containing simulated ROI data corresponding to `datasetROI`, is included with the distribution. The first command (below) finds the location of the file and the second command reads it and saves it to a dataset object `ds`. !!!DPC!!!
- The `DfReadDataFile` function automatically recognizes that this is an *ROI* dataset. Its structure is similar to the JAFROC format Excel file, with some important differences, noted below. It contains three worksheets:

```
## fileName <- system.file(
##     "extdata", "RoiData.xlsx", package = "RJafroc", mustWork = TRUE)
## ds <- DfReadDataFile(fileName)
## ds$dataType
```

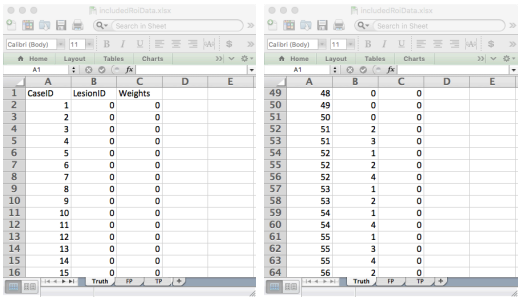
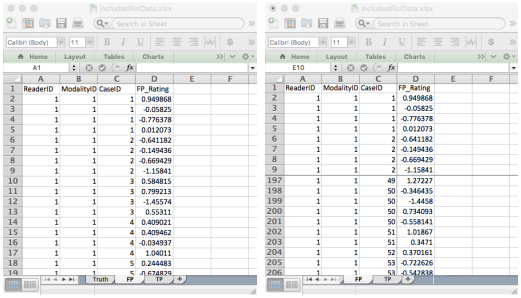


Figure 21.1: Fig. 1 two views of Truth worksheet

- The `Truth` worksheet, Fig. 1, indicates which cases are diseased and which are non-diseased and the number of ROI-level-diseased region on each case.
  - There are 50 non-diseased cases (labeled 1-50) under column `CaseID` and 40 diseased cases (labeled 51-90).
  - The `LesionID` field for each non-diseased case (e.g., `CaseID` = 1) is zero and there is one row per case. For diseased cases, this field has a variable number of entries, ranging from 1 to 4. As an example,

there are two rows for `CaseID` = 51 in the Excel file: one with `LesionID` = 2 and one with `LesionID` = 3.

– The `Weights` field is always zero (this field is not used in ROI analysis).



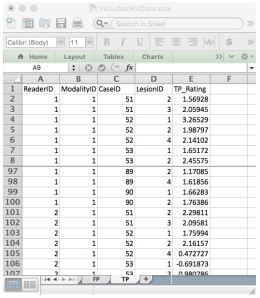
	A	B	C	D	E	F
1	ReaderID	ModalityID	CaseID	FP_Rating		
2	1	1	1	0.949868		
3	1	1	1	-0.05825		
4	1	1	1	-0.776378		
5	1	1	1	0.012073		
6	1	1	2	-0.641182		
7	1	1	2	-0.149436		
8	1	1	2	-0.669429		
9	1	1	2	-1.15841		
10	1	1	3	0.584815		
11	1	1	3	0.799213		
12	1	1	3	-1.45574		
13	1	1	3	0.55311		
14	1	1	4	0.409021		
15	1	1	4	0.409462		
16	1	1	4	-0.034937		
17	1	1	4	1.04011		
18	1	1	5	0.244483		
19	1	1	5	-0.674879		

	A	B	C	D	E	F
1	ReaderID	ModalityID	CaseID	FP_Rating		
2	1	1	1	0.949868		
3	1	1	1	-0.05825		
4	1	1	1	-0.776378		
5	1	1	1	0.012073		
6	1	1	2	-0.641182		
7	1	1	2	-0.149436		
8	1	1	2	-0.669429		
9	1	1	2	-1.15841		
197	1	1	49	1.27227		
198	1	1	50	-0.346435		
199	1	1	50	-1.4458		
200	1	1	50	0.734093		
201	1	1	50	-0.558141		
202	1	1	51	1.01867		
203	1	1	51	0.3471		
204	1	1	52	0.370161		
205	1	1	53	-0.722626		
206	1	1	53	-0.142838		

Figure 21.2: Fig. 2 two views of FP worksheet

- The `FP` (or `NL`) worksheet - this lists the ratings of ROI-level-non-diseased regions.
  – For `ReaderID` = 1, `ModalityID` = 1 and `CaseID` = 1 there are 4 rows, corresponding to the 4 ROI-level-non-diseased regions in this case. The corresponding ratings are 0.949868, -0.0582497, -0.776378, 0.012073. The pattern repeats for other treatments and readers, but the rating are, of course, different.

  – Each `CaseID` is represented in the `FP` worksheet (a rare exception could occur if a case-level diseased case has 4 diseased regions).



	A	B	C	D	E	F
1	ReaderID	ModalityID	CaseID	LesionID	TP_Rating	
2	1	1	51	2	1.56928	
3	1	1	51	3	2.05945	
4	1	1	52	1	3.26529	
5	1	1	52	2	1.98797	
6	1	1	52	4	2.14102	
7	1	1	53	1	1.65172	
8	1	1	53	2	2.45575	
97	1	1	89	2	1.17085	
98	1	1	89	4	1.61856	
99	1	1	90	1	1.66283	
100	1	1	90	2	1.76386	
101	2	1	51	2	2.29811	
102	2	1	51	3	2.09581	
103	2	1	52	1	1.75994	
104	2	1	52	2	2.16157	
105	2	1	52	4	0.472727	
106	2	1	53	1	-0.691873	
107	2	1	53	2	0.980786	

Figure 21.3: Fig. 2 TP worksheet

- The `TP` (or `LL`) worksheet - this lists the ratings of ROI-level-diseased regions.

- Because non-diseased cases generate TPs, one does not find any entry with `CaseID` = 1-50 in the `TP` worksheet.

- The lowest `CaseID` in the `TP` worksheet is 51, which corresponds to the first diseased case.

- There are two entries for this case, corresponding to the two ROI-level-diseased regions present in this case. Recall that corresponding to this `CaseID` in the `Truth` worksheet there were two entries with `LesionID` = 2 and 3. These must match the `LesionID`'s listed for this case in the `TP` worksheet. Complementing these two entries, in the `FP` worksheet for `CaseID` = 51, there are 2 entries corresponding to the two ROI-level-non-diseased regions in this case.

- One should confirm that for each diseased case the sum of the number of entries in the `TP` and `FP` worksheets is always 4.

21.4 Next, TBA

The next vignette illustrates significance testing for this paradigm.

21.5 References

Chapter 22

Analyzing data acquired according to the ROI paradigm

22.1 Introduction; this vignette is under construction!

22.2 Note to self (10/29/19) !!!DPC!!!

The FOM and DeLong method implementations need checking with a toy dataset.

22.3 Introduction

- For an ROI dataset `StSignificanceTesting()` automatically defaults to `method = "ORH"`, `covEstMethod = "DeLong"` and `FOM = "ROI"`.
- The covariance estimation method is based on the original DeLong method (DeLong et al., 1988), which is valid only for the trapezoidal AUC, i.e. ROC data, as extended by (Obuchowski, 1997) to ROI data, see formula below.
- The essential differences from conventional ROC analyses are in the definition of the ROI figure of merit, see below, and the procedure developed by (Obuchowski, 1997) for estimating the covariance matrix. Once the

covariances are known, `method = "ORH"` can be applied to perform significance testing, as described in (Obuchowski and Rockette, 1995) and (Chakraborty, 2017, Chapter 10).

22.4 The ROI figure of merit

Let X_{kr} denote the rating for the r$^{\text{th}}$ **lesion-containing** ROI in the k$^{\text{th}}$ case and let n_k^L be the total number of lesion-containing ROIs in the k$^{\text{th}}$ case. Similarly, let Y_{kr} denote the rating for the r$^{\text{th}}$ **lesion-free** ROI in the k$^{\text{th}}$ case and n_k^N denote the total number of lesion-free ROIs in the k$^{\text{th}}$ case. Let N_L denote the total number of lesion-containing ROIs in the image set and N_N denote the total number of lesion-free ROIs. These are given by

$$N_L = \sum\nolimits_k n_k^L$$

and

$$N_N = \sum\nolimits_k n_k^N$$

. The ROI figure of merit θ is defined by: ab

The kernel function $\Psi(X, Y)$ is defined by:

The ROIs are *effectively regarded as mini-cases* and one calculates the FOM as the Wilcoxon statistic considering the mini-cases as actual cases. The correlations between the ratings of ROIs on the same case are accounted for in the analysis.

22.5 Calculation of the ROI figure of merit.

```
UtilFigureOfMerit(datasetROI, FOM = "ROI")
#>           Rdr1      Rdr2      Rdr3      Rdr4      Rdr5
#> Trt1 0.9057239 0.8842834 0.8579279 0.9350207 0.8352103
#> Trt2 0.9297186 0.9546035 0.8937652 0.9531716 0.8770076
fom <- UtilFigureOfMerit(datasetROI, FOM = "ROI")
```

- If the correct `FOM` is not supplied, it defaults to `FOM = ROI`.
- This is a 2-treatment 5-reader dataset.
- For treatment 1, reader 1 the figure of merit is 0.9057239.
- For treatment 2, reader 5 the figure of merit is 0.8770076.
- Etc.

22.6 Significance testing

When `dataset$dataType == "ROI"` the FOM defaults to "ROI" (meaning the above formula) and the covariance estimation method defaults to `covEstMethod = "DeLong"`.

```
ret <- StSignificanceTesting(datasetROI, FOM = "Wilcoxon")
#> ROI dataset: forcing method = `ORH`, covEstMethod = `DeLong` and FOM = `ROI`.
str(ret)
#> List of 14
#>  $ fomArray            : num [1:2, 1:5] 0.906 0.93 0.884 0.955 0.858 ...
#>   ..- attr(*, "dimnames")=List of 2
#>   .. ..$ : chr [1:2] "Trt1" "Trt2"
#>   .. ..$ : chr [1:5] "Rdr1" "Rdr2" "Rdr3" "Rdr4" ...
#>  $ meanSquares         :'data.frame':    1 obs. of  3 variables:
#>   ..$ msT : num 0.00361
#>   ..$ msR : num 0.00256
#>   ..$ msTR: num 0.000207
#>  $ varComp             :'data.frame':    1 obs. of  6 variables:
#>   ..$ varR : num 0.00108
#>   ..$ varTR: num 0.000153
#>   ..$ cov1 : num 0.000247
#>   ..$ cov2 : num 0.000187
#>   ..$ cov3 : num 0.000154
#>   ..$ var  : num 0.000333
#>  $ FTestStatsRRRC      :'data.frame':    1 obs. of  4 variables:
#>   ..$ fRRRC  : num 9.76
#>   ..$ ndfRRRC: num 1
#>   ..$ ddfRRRC: num 12.8
#>   ..$ pRRRC  : num 0.00817
#>  $ ciDiffTrtRRRC       :'data.frame':    1 obs. of  8 variables:
#>   ..$ Treatment: chr "Trt1-Trt2"
#>   ..$ Estimate : num -0.038
#>   ..$ StdErr   : num 0.0122
#>   ..$ DF       : num 12.8
#>   ..$ t        : num -3.12
#>   ..$ PrGTt    : num 0.00817
#>   ..$ CILower  : num -0.0643
#>   ..$ CIUpper  : num -0.0117
#>  $ ciAvgRdrEachTrtRRRC :'data.frame':    2 obs. of  6 variables:
#>   ..$ Treatment: Factor w/ 2 levels "Trt1","Trt2": 1 2
#>   ..$ Area     : num [1:2] 0.884 0.922
#>   ..$ StdErr   : num [1:2] 0.0232 0.0197
#>   ..$ DF       : num [1:2] 12.2 10.1
#>   ..$ CILower  : num [1:2] 0.833 0.878
```

```
#>   ..$ CIUpper  : num [1:2] 0.934 0.966
#>  $ FTestStatsFRRC      :'data.frame':    1 obs. of  4 variables:
#>   ..$ fFRRC  : num 16.6
#>   ..$ ndfFRRC: num 1
#>   ..$ ddfFRRC: num Inf
#>   ..$ pFRRC  : num 4.58e-05
#>  $ ciDiffTrtFRRC       :'data.frame':    1 obs. of  8 variables:
#>   ..$ Treatment: chr "Trt1-Trt2"
#>   ..$ Estimate : num -0.038
#>   ..$ StdErr   : num 0.00933
#>   ..$ DF       : num Inf
#>   ..$ t        : num -4.08
#>   ..$ PrGTt    : num 4.58e-05
#>   ..$ CILower  : num -0.0563
#>   ..$ CIUpper  : num -0.0197
#>  $ ciAvgRdrEachTrtFRRC :'data.frame':    2 obs. of  6 variables:
#>   ..$ Treatment: Factor w/ 2 levels "Trt1","Trt2": 1 2
#>   ..$ Area     : num [1:2] 0.884 0.922
#>   ..$ StdErr   : num [1:2] 0.0163 0.0129
#>   ..$ DF       : num [1:2] Inf Inf
#>   ..$ CILower  : num [1:2] 0.852 0.896
#>   ..$ CIUpper  : num [1:2] 0.916 0.947
#>  $ ciDiffTrtEachRdrFRRC:'data.frame':    5 obs. of  9 variables:
#>   ..$ Reader   : Factor w/ 5 levels "Rdr1","Rdr2",..: 1 2 3 4 5
#>   ..$ Treatment: Factor w/ 1 level "Trt1-Trt2": 1 1 1 1 1
#>   ..$ Estimate : num [1:5] -0.024 -0.0703 -0.0358 -0.0182 -0.0418
#>   ..$ StdErr   : num [1:5] 0.01025 0.01448 0.01648 0.00928 0.01398
#>   ..$ DF       : num [1:5] Inf Inf Inf Inf Inf
#>   ..$ t        : num [1:5] -2.34 -4.86 -2.17 -1.96 -2.99
#>   ..$ PrGTt    : num [1:5] 1.93e-02 1.20e-06 2.97e-02 5.05e-02 2.79e-03
#>   ..$ CILower  : num [1:5] -0.0441 -0.0987 -0.0681 -0.0363 -0.0692
#>   ..$ CIUpper  : num [1:5] -3.90e-03 -4.19e-02 -3.53e-03 3.88e-05 -1.44e-02
#>  $ varCovEachRdr       :'data.frame':    5 obs. of  3 variables:
#>   ..$ Reader: Factor w/ 5 levels "Rdr1","Rdr2",..: 1 2 3 4 5
#>   ..$ Var   : num [1:5] 0.000269 0.000227 0.000481 0.000168 0.000522
#>   ..$ Cov1  : num [1:5] 0.000216 0.000122 0.000345 0.000125 0.000424
#>  $ FTestStatsRRFC      :'data.frame':    1 obs. of  4 variables:
#>   ..$ fRRFC  : num 17.5
#>   ..$ ndfRRFC: num 1
#>   ..$ ddfRRFC: num 4
#>   ..$ pRRFC  : num 0.0139
#>  $ ciDiffTrtRRFC       :'data.frame':    1 obs. of  8 variables:
#>   ..$ Treatment: chr "Trt1-Trt2"
#>   ..$ Estimate : num -0.038
#>   ..$ StdErr   : num 0.00909
```

```
#>   ..$ DF        : num 4
#>   ..$ t         : num -4.18
#>   ..$ PrGTt     : num 0.0139
#>   ..$ CILower   : num -0.0633
#>   ..$ CIUpper   : num -0.0128
#>  $ ciAvgRdrEachTrtRRFC :'data.frame':    2 obs. of  6 variables:
#>   ..$ Treatment: Factor w/ 2 levels "Trt1","Trt2": 1 2
#>   ..$ Area      : num [1:2] 0.884 0.922
#>   ..$ StdErr    : num [1:2] 0.0175 0.0157
#>   ..$ DF        : num [1:2] 4 4
#>   ..$ CILower   : num [1:2] 0.835 0.878
#>   ..$ CIUpper   : num [1:2] 0.932 0.965
```

- While `ret` is a list with many (22) members, their meanings should be clear from the notation. As an example:
- The variance components are given by:

```
ret$varComp
#>           varR        varTR         cov1         cov2         cov3          var
#> 1 0.001082359 0.0001526084 0.0002465125 0.0001870571 0.0001543764 0.0003333119
```

22.6.1 RRRC analysis

```
ret$FTestStatsRRRC$fRRRC
#> [1] 9.763602
ret$FTestStatsRRRC$ndfRRRC
#> [1] 1
ret$FTestStatsRRRC$ddfRRRC
#> [1] 12.82259
ret$FTestStatsRRRC$pRRRC
#> [1] 0.008173042
```

- The F-statistic is , with `ndf = 1` and `ddf` = , which yields a p-value of .
- The confidence interval for the reader averaged difference between the two treatments is given by:

```
ret$ciDiffTrtRRRC
#>   Treatment    Estimate     StdErr       DF         t       PrGTt     CILower
#> 1 Trt1-Trt2 -0.03802005 0.01216768 12.82259 -3.124676 0.008173042 -0.06434373
#>       CIUpper
#> 1 -0.01169636
```

- The FOM difference (treatment 1 minus 2) is -0.03802, which is significant, p-value = 0.008173, F-statistic = 9.7636016, ddf = 12.8225898. The confidence interval is (-0.0643437, -0.0116964).

22.6.2 FRRC analysis

```
ret$FTestStatsFRRC$fFRRC
#> [1] 16.6135
ret$FTestStatsFRRC$ndfFRRC
#> [1] 1
ret$FTestStatsFRRC$ddfFRRC
#> [1] Inf
ret$FTestStatsFRRC$pFRRC
#> [1] 4.582365e-05
```

- The F-statistic is 16.6135014, with `ndf = 1` and `ddf = Inf`, which yields a p-value of 4.5823651×10^{-5}.
- The confidence interval for the reader averaged difference between the two treatments is given by:

```
ret$ciDiffTrtFRRC
#>   Treatment    Estimate      StdErr  DF         t        PrGTt     CILower
#> 1 Trt1-Trt2 -0.03802005 0.009327861 Inf -4.075966 4.582365e-05 -0.05630232
#>       CIUpper
#> 1 -0.01973778
```

22.6.3 RRFC analysis

```
ret$FTestStatsRRFC$fRRFC
#> [1] 17.48107
ret$FTestStatsRRFC$ndfRRFC
#> [1] 1
ret$FTestStatsRRFC$ddfRRFC
#> [1] 4
ret$FTestStatsRRFC$pRRFC
#> [1] 0.01390667
```

- The F-statistic is 17.4810666, with `ndf = 1` and `ddf` = 4, which yields a p-value of 0.0139067.

- The confidence interval for the reader averaged difference between the two treatments is given by:

```
ret$ciDiffTrtRRFC
#>   Treatment    Estimate     StdErr DF         t      PrGTt     CILower
#> 1 Trt1-Trt2 -0.03802005 0.00909345  4 -4.181037 0.01390667 -0.06326751
#>       CIUpper
#> 1 -0.01277258
```

22.7 Summary

TBA

22.8 References

Chapter 23

Simulate an FROC split plot dataset

23.1 This vignette is under construction!!

- This is a follow-up on the recently added (v1.3.1) capability to read a split-plot dataset.
- Lacking an actual split-plot dataset to test the routines, I decided to simulate one.
- The simulated dataset is of dataType FROC and the number of cases interpreted by each reader is reader-dependent.
- This makes it *really* exercise the validity of the `DfReadDataFile` function.
- In my experience, the `dataset$truthTableStr` member is invaluable in catching data entry errors so much of this vignette focuses on it.

23.2 The starting point is an actual crossed FROC dataset

The example shown below begins with the Excel file `inst/extdata/FrocData.xlsx` in the project directory (this corresponds to the 5-modality FED dataset `dataseet04` (Zanca et al., 2009) with modalities 1, 2 and 3 removed). 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 `DfReadDataFile()` and saves it to object `x1`. The next statement extracts the `truthTableStr` list member, saves it to `t1` and shows its structure.

```
fed <- system.file("extdata", "FrocData.xlsx",
                        package = "RJafroc", mustWork = TRUE)
x1 <- DfReadDataFile(fed, newExcelFileFormat = TRUE)
t1 <- x1$truthTableStr
str(t1)
#>  num [1:2, 1:4, 1:200, 1:4] 1 1 1 1 1 1 1 1 1 1 ...
```

- There are 100 normal and 100 abnormal cases in this two-modality four-reader crossed dataset.
- Note that `t1` is the original crossed dataset `truthTableStr`.
- Recall from earlier vignette that for the fourth subscript of `t1` the value 1 applies to cases with no lesions (normals), value 2 applies to cases with one lesion, value 3 applies to cases with two lesions and 4 applies to cases with three lesions.
- The value for any allowed interpretation is 1 and otherwise it is `NA`.

23.3 Understanding `truthTableStr` object `t1`

- The following line yields 200 (=2*100) as reader 1 (second subscript) provides interpretations in both modalities (first subscript is blank meaning both modalities) for all 100 normal cases (third subscript is 1:100 and fourth subscript is 1) and therefore each of these interpretations yields a `TRUE` (i.e., 1).

```
sum(!is.na(t1[,1,1:100,1]))
#> [1] 200
```

- The following line yields 0 as the third subscript is 1:100, implying normal cases, but the fourth subscript is 2:4, implyng abnormal cases and therefore each of these interpretations yields an NA and `!is.na(NA)` equals `FALSE` (i.e., zero).

```
sum(!is.na(t1[,1,1:100,2:4]))
#> [1] 0
```

- The following line also yields 800 (=2x4x100) as readers 1:4 provide interpretations in both modalities for all normal cases and each interpretation yields a 1.

```
sum(!is.na(t1[,,1:100,1]))
#> [1] 800
```

- The following line yields 200 (=2*100) because the fourth subscript (2) applies to abnormal cases with at least one lesion, and each abnormal case is guaranteed to have at least one lesion (i.e., a 1 entry in the `LesionID` column of the Excel Truth worksheet) and each of these interpretations yields a 1.

```
sum(!is.na(t1[,1,101:200,2]))
#> [1] 200
```

- The following line yields 62 (=2x31) because the fourth subscript (3) applies to abnormal cases with at least two lesions, and inspection of the `LesionID` column in the original Excel file reveals that 31 abnormal cases have two lesions.

```
sum(!is.na(t1[,1,101:200,3]))
#> [1] 62
```

- The following line yields 22 (=2x11) because the fourth subscript (4) applies to abnormal cases with three lesions. Inspection of the `LesionID` column reveals that 11 abnormal cases have three lesions.

```
sum(!is.na(t1[,1,101:200,4]))
#> [1] 22
```

- The following line yields 242 (=200+31+11), the number of rows in the `Truth` worksheet.

```
sum(!is.na(t1[1,1,,]))
#> [1] 242
```

23.4 Modify a crossed FROC workbook to simulate a split-plot FROC design

- We create a simulated split-plot FROC dataset starting from a crossed FROC dataset.
- The basic idea is to modify interpretations that do not belong to a specified split-plot design.
- This was done (one could say the "hard way") by manually making appppropriate changes to `inst/extdata/FrocData.xlsx` and saving the results to `inst/extdata/toyFiles/FROC/FrocDataSpVaryK1K2.xlsx`. The filename is intended to emphasize that the numbers of normal and abnormal cases can be reader-dependent, as long as they individually add up to 100.

- We divided the 100 normal and 100 abnormal cases into 4 groups of normal and abnormal cases, where each group is interpreted by one reader only.
- The first groups of cases, interpreted by reader 1 (label "1"), consists of 23 normal (case labels "100:122") and 24 abnormal (case labels "0:23") cases.
- The second groups of cases, interpreted by reader 2 (label "3"), consists of 27 normal (case labels "123:149") and 26 abnormal (case labels "24:49") cases.
- The third groups of cases, interpreted by reader 3 (label "4"), consists of 22 normal (case labels "150:171") and 23 abnormal (case labels "51:73") cases.
- The fourth groups of cases, interpreted by reader 4 (label "5"), consists of 28 normal (case labels "172:199") and 27 abnormal (case labels c("50,74:99")) cases.

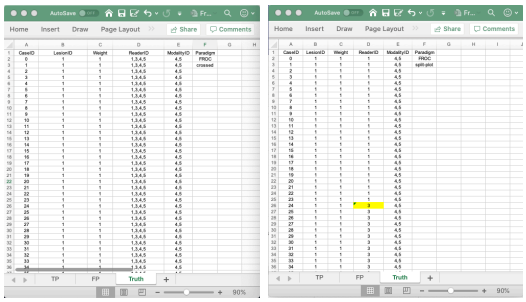


Figure 23.1: Truth worksheets; (a) LEFT=FrocData.xlsx, original crossed dataset; (b) RIGHT=FrocDataSpVaryK1K2.xlsx, modified split-plot dataset

- The above figure shows that the `ReaderID` column for cases `0:23` has been replaced with a 1, meaning that only reader 1 interprets these cases in both modalities. This yields 24 abnormal cases for reader 1 in each modality. Normal cases for this reader are `100:122`.
- Not shown above: **all interpretations by reader 1 occurring for cases outside of `0:23` and `100:122` in the other two worksheets (`TP` and `FP`) are deleted**.
- The `ReaderID` column for cases `24:49` are replaced by 3, corresponding to the second reader. All interpretations by this reader for cases outside of `24:49` in the other two worksheets are deleted. Normal cases for this reader are `123:149` and observations outside of this range in the `TP` and `FP` worsheets are deleted.
- The `ReaderID` column for cases `51:73` are replaced by 4, corresponding to the third reader. All interpretations by this reader for cases outside of `51:73` in the other two worksheets are deleted. Normal cases for this reader are `150:171` and observations outside of this range in the `TP` and `FP` worsheets are deleted.

- The `ReaderID` column for cases `50` and `74:79` are replaced by 5, corresponding to the fourth reader. All interpretations by this reader for cases outside of the specified range in the other two worksheets are deleted. Normal cases for this reader are `172:199` and observations outside of this range in the `TP` and `FP` worsheets are deleted.
- The modified file is read by the code chunk below. The read function explicitly tests that observations outside of the specified ranges in the `Truth` sheet are not present in the other two worksheets.

23.5 Example of deletion of interpretations

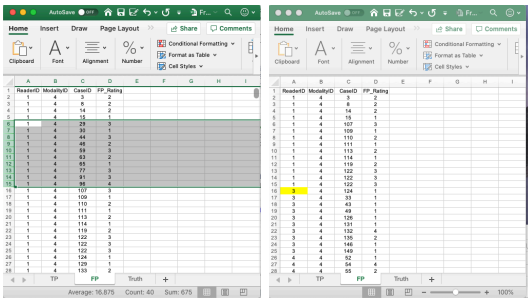


Figure 23.2: FP worksheets; (a) LEFT=FrocDataFP.xlsx, original crossed dataset; (b) RIGHT=FrocDataSpVaryK1K2FP.xlsx, modified split-plot dataset

- The two figures above illustrate deletion of interpretations.
- The left panel shows the `FP` worksheet for the original crossed data.
- For reader 1 and modality 4 it contains cases 29, 30, 44, ..., 91, 96 that do not belong to the split-plot dataset for this reader.
- Specifically, they are outside of `0:23` and `100:122`, the allowed ranges for this reader.
- These are deleted, see right panel of above figure. The next allowed cases for this reader are `107, 109,...., 122`.
- The procedure is repeated for all readers and both `TP` and `FP` sheets.

```
fedsp <- system.file("extdata", "toyFiles/FROC/FrocDataSpVaryK1K2.xlsx",
                     package = "RJafroc", mustWork = TRUE)
x2 <- DfReadDataFile(fedsp, newExcelFileFormat = TRUE)
t2 <- x2$truthTableStr
```

23.6 Understanding `truthTableStr` object `t2`

- The following line below yields 46 (= 2x23) as reader 1 (second subscript) provides interpretations in both modalities (first subscript is blank) for all normal cases (third subscript is 1:100 and fourth subscript is 1) and there are 23 normal cases interpreted by reader 1.

```
sum(!is.na(t2[,1,1:100,1]))
#> [1] 46
```

- The following line confirms the first line, with a 1 contribution coming from each case in range 1:23.

```
sum(!is.na(t2[,1,1:23,1]))
#> [1] 46
```

- The following line yields 48 (= 2x24) because the fourth subscript (2) applies to abnormal cases with at least one lesion, and we know that this reader has interpreted 24 abnormal cases.

```
sum(!is.na(t2[,1,101:124,2]))
#> [1] 48
```

- The following line yields 54 (= 2x27) because the fourth subscript (1) applies to normal cases and we know that reader 2 has interpreted 27 normal cases.

```
sum(!is.na(t2[,2,,1]))
#> [1] 54
```

- The following line yields 52 (= 2x26) because the fourth subscript (2:4) applies to abnormal cases with at least one lesion, and we know that reader 2 has interpreted 26 abnormal cases.

```
sum(!is.na(t2[,2,,2:4]))
#> [1] 52
```

23.7 References

Bibliography

Chakraborty, D. P. (1989). Maximum likelihood analysis of free-response receiver operating characteristic (froc) data. *Medical Physics*, 16(4):561–568.

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

DeLong, E. R., DeLong, D. M., and Clarke-Pearson, D. L. (1988). Comparing the areas under two or more correlated receiver operating characteristic curves: A nonparametric approach. *Biometrics*, 44:837–845.

Franken, Edmund A., J., Berbaum, K. S., Marley, S. M., Smith, W. L., Sato, Y., Kao, S. C. S., and Milam, S. G. (1992). Evaluation of a digital workstation for interpreting neonatal examinations: A receiver operating characteristic study. *Investigative Radiology*, 27(9):732–737.

Hillis, S. L. and Berbaum, K. S. (2004). Power estimation for the dorfman-berbaum-metz method. *Acad. Radiol.*, 11(11):1260–1273.

ICRU (2008). *Statistical Analysis and Power Estimation*, volume 8, pages 37–40.

Metz, C. (1978). Basic principles of roc analysis. *Seminars in Nuclear Medicine*, 8(4):283–298.

Obuchowski, N. A. (1997). Nonparametric analysis of clustered roc curve data. *Biometrics*, 53:567–578.

Obuchowski, N. A. and Rockette, H. (1995). Hypothesis testing of the diagnostic accuracy for multiple diagnostic tests: An anova approach with dependent observations. *Communications in Statistics: Simulation and Computation*, 24:285–308.

Zanca, F., Jacobs, J., Van Ongeval, C., Claus, F., Celis, V., Geniets, C., Provost, V., Pauwels, H., Marchal, G., and Bosmans, H. (2009). Evaluation of clinical image processing algorithms used in digital mammography. *Medical Physics*, 36(3):765–775.