Workfolio

Hannah-Louise Hayman

05/12/22

Contents

Flows

Note: Flow functions are adapted from package format to standalone scripts.

CutFlow

CutFlow is designed to generate cut points for multiple variables, using a training dataset, and apply these cutpoints to both the training dataset and any validation sets supplied. Note that cutpoints are generated from the training data and applied to the training data plus other datasets (validation).

To run CutFlow, simply fill in the blanks, as in the example below;

- Save your dataset(s) as a CSV file
  - Example data can be found at flowsData/cutFlowIn/cutFlowData
- Create a subdirectory in your R directory, and place your dataset files inside
- Subdirectory is the name of the folder you placed your datasets in. It must be within your current directory
- TrainingData is the name of the dataset you wish to be used to generate the cutpoint
  - This cutpoint is then applied to all datasets within the subdirectory
- If coding multiple datasets, the respective variables must have the exact same name in all datasets
  - Additional datasets do not need to contain all variables from your training dataset, just those you wish to be coded in that dataset
- CutPointStatus is the status variable to be used
  - Coded as 0 (no event) and 1 (event)
- CutPointTime is the time variable to be used
  - Must be a continuous variable
- minprop is the minimum proportion of cases to be include either side of the cutpoint
  - Default is 0.1, exclude the argument if you don't want to change this
- Greyscale is an optional toggle to produce a greyscale variant of all plots
  - The default is colour, exclude the argument if you don't want to change this
- Variables is a list of your variables to generate cutpoints for

Table 1: cutFlowA data not coded.

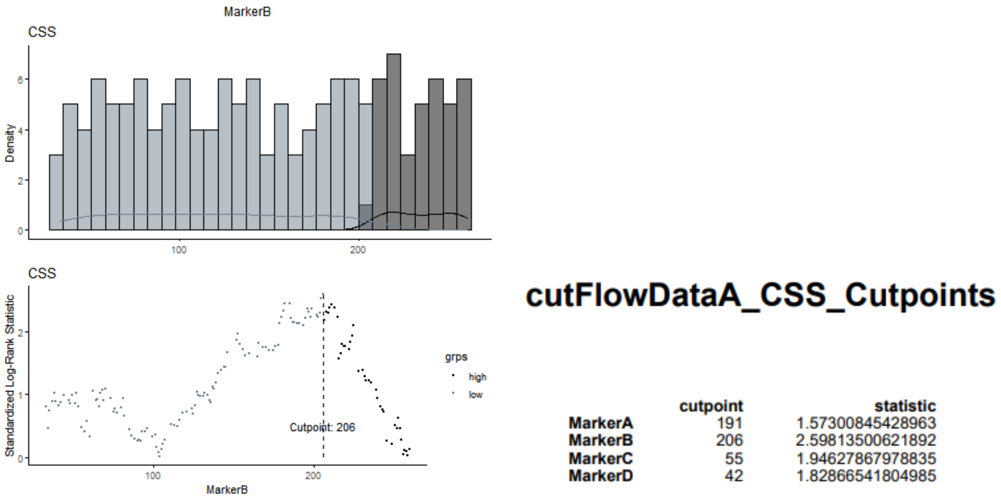
TMA_ID	CSS	CSS_Time	OS	OS_Time	MarkerA	MarkerB	MarkerC	MarkerD
TMA_1	0	50	0	92	176	278	53	273
TMA_2	0	8	1	14	246	154	225	97
TMA_3	0	89	1	6	20	95	92	234

Example syntax:

```
CutFlow(Subdirectory = "flowsData/cutFlowIn/cutFlowData", TrainingData = "cutFlowDataA",
        CutPointStatus = c("CSS"), CutPointTime = c("CSS_Time"), minprop = 0.1,
        Greyscale = TRUE, Variables = c("MarkerA",
                                        "MarkerB",
                                        "MarkerC",
                                        "MarkerD"))
```

A new folder will be created in your R directory;

- Folder name format is CutFlow_SystemData_Number
- Three folders are contained within;
  - 0.OriginalDatasetFiles - A copy of all datafiles fed into CutFlow, for record keeping
  - 1.CutPointOutput - A copy of all cutpoint data, including a pdf list of cutpoints
  - 2.CodedDatasets - A copy of all datasets, newly coded



cutFlowDataA_CSS_Cutpoints

	cutpoint	statistic
MarkerA	191	1.57300845428963
MarkerB	206	2.59813500621892
MarkerC	55	1.94627867978835
MarkerD	42	1.82866541804985

Figure 1: A - Cutpoint graphic per variable. B - CutPoints summary.

Table 2: cutFlowA data coded

TMA_ID	CSS	CSS_Time	OS	OS_Time	MarkerA	MarkerB	MarkerC	MarkerD	MarkerA_Coded	MarkerB_Coded	MarkerC_Coded	MarkerD_Coded
TMA_1	0	50	0	92	176	278	53	273	1	1	1	1
TMA_2	0	8	1	14	246	154	225	97	1	1	1	1
TMA_3	0	89	1	6	20	95	92	234	0	0	1	1

SurvFlow

SurvFlow takes in a dataset and runs appropriate survival analysis.

To run SurvFlow, simply fill in the blanks, as in the example below;

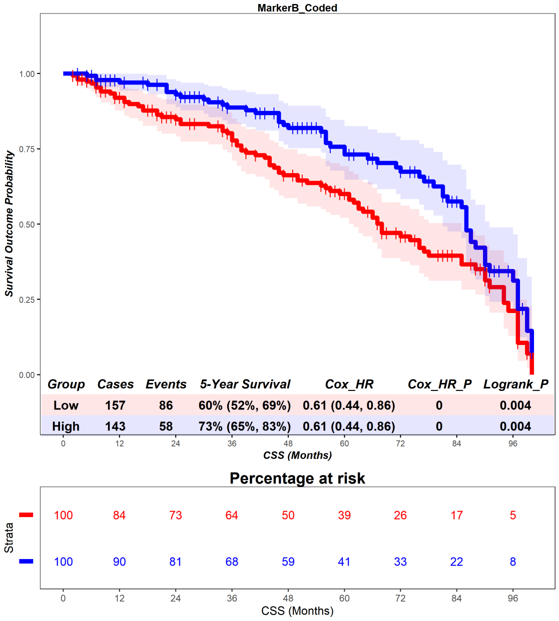
- Data is a Coded dataset in csv format. If using CutFlow, simply use the produced coded dataset
- Variables is a list of variables (coded as 0 and 1) for analysis
- LegendLabels are optional labels for your legends. Default is the value of the level (0 and 1)
- Identifier is an identifier variable for cases
- PlotTitles are optional plot titles. Default is variable name
- SurvivalStatus are status variables
  - Coded as 0 (no event) and 1 (event)
  - Must have the same number of elements as the SurvivalTime variable
- SurvivalTime are survival time variables
  - Must be a continuous variable
  - Must have the same number of elements as the SurvivalStatus variable
- SurvivalTimeUnit is the unit of time for survival time
- xYearSurvivalVar is the number of years to be used to calculate 'X' years survival. Default = 5

Example syntax:

```r
Data <- read.csv(
    paste0(
      getwd(),
      "/flowsData/cutFlowOut/CutFlow_2022-11-30_1/2.CodedDatasets/cutFlowDataA.csv"
    )
  )
SurvFlow(
  Data,
  Variables = c(
    "MarkerA_Coded",
    "MarkerB_Coded",
    "MarkerC_Coded",
    "MarkerD_Coded"
  ),
  LegendLabels = c("Low", "High"),
  Identifier = "TMA_ID",
  SurvivalStatus = c("CSS", "OS"),
  SurvivalTime = c("CSS_Time", "OS_Time"),
  xYearSurvivalVar = 5,
  SurvivalTimeUnit = "Months",
  SurvBase = TRUE
)
```

A new folder will be created in your R directory;

- Folder name format is SurvFlow_Filename_SystemData_Number
- Inside is a folder per SurvFlow module, for example BaseSurv
  - At the next level is a folder per survival status/time pair, containing the survival plots
- The plot can be seen as below;



Group	Cases	Events	5-Year Survival	Cox_HR	Cox_HR_P	Logrank_P
Low	157	86	60% (52%, 69%)	0.61 (0.44, 0.86)	0	0.004
High	143	58	73% (65%, 83%)	0.61 (0.44, 0.86)	0	0.004

	0	12	24	36	48	60	72	84	96
🟥	100	84	73	64	50	39	26	17	5
🟦	100	90	81	68	59	41	33	22	8

Figure 2: survFlow example output.

Omics

intSurvLymph

clinEnrich carries out clinical enrichment using MafTools' 'clinicalEnrichment' function and outputs an S4 object containing five items:

- sigGenes
    - A list of genes with p value < 0.05 and an odds ratio > 2
- sigGenesFDR
    - sigGenes with p value replaced by FDR
- sigGenesData
    - Enrichment results for significant genes
- sigGenesFDRData
    - sigGenesData with p value replaced by FDR
- enrichData
    - Complete enrichment data
- Also outputs a plot of clinical enrichment to the provided filepath

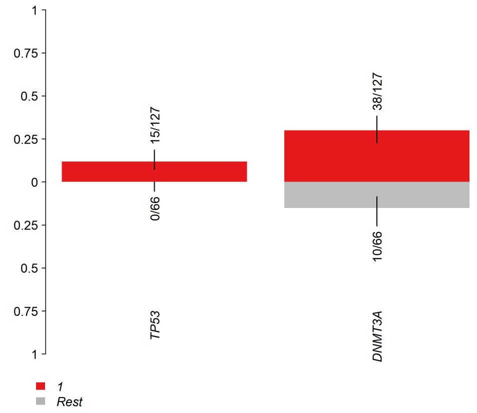


Figure 3: A - Histogram. B - qqPlot.

geneCombHR builds cox regression models to generate hazard ratios and associated p values for each gene resulting from clinical enrichment:

- Results
  - Full results
- sigResults
  - Significant results with a hazard ratio >2 or <0.5
- sigAdjResults
  - Significant (adjusted) results with a hazard ratio >2 or <0.5
- cox.zph.res
  - Results of proportional hazard tests

intSurvLymph takes in lymphData (from clinEnrich) and survData (from geneCombHR) and outputs three plots:

- Lymph plot
  - Bubble plot of log(OR) on x and log(p) on y
- Surv plot
  - Bubble plot of log(HR) on x and log(p) on y
- Dual plot
  - Bubble plot of log(OR) on x and log(HR) on y

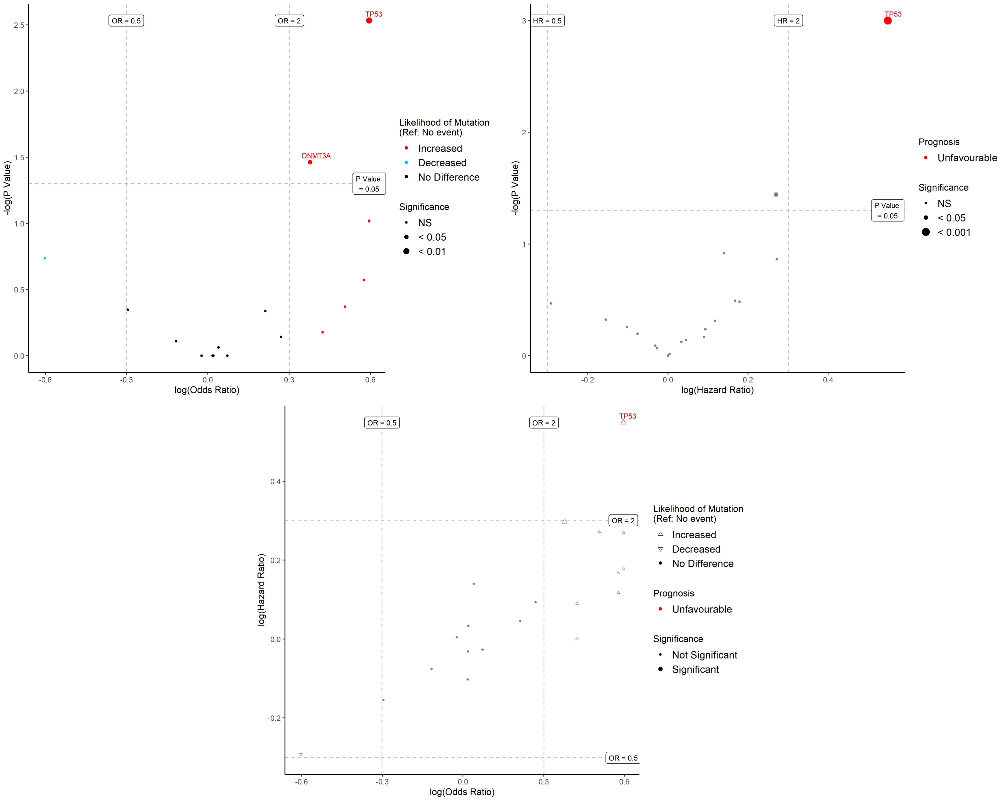


Figure 4: A - Lymph plot. B - Surv plot. C - Dual plot.

Example syntax:

```r
laml.maf = system.file('extdata', 'tcga_laml.maf.gz', package = 'maftools')
laml.clin = system.file('extdata', 'tcga_laml_annot.tsv', package = 'maftools')
laml = read.maf(maf = laml.maf, clinicalData = laml.clin)

enrichMain <- clinEnrich(
  maf = laml,
  variable = "Overall_Survival_Status",
  minMut = 5,
  filePath = "omicsData/"
)

lymphData <-
  enrichMain@enrichData[enrichMain@enrichData$Group1 == "1",]

hrOutput <-
  geneCombHR(
    maf = laml,
    genes = lymphData$Hugo_Symbol,
    maxN = 1,
    Time = "days_to_last_followup",
    Event = "Overall_Survival_Status",
    CImin = 0.75,
    CImax = 2,
    minMut = 5,
    adjust = "Yes",
    fileName = "Bubble-",
    filePath = "omicsData/"
  )

intSurvLymphOutput <- intSurvLymph(
  lymphData = lymphData,
  survData = hrOutput@Results,
  cellType = "Status",
  reference = "No event",
  labelSize = 3,
  filePath = "omicsData/"
)
```

Additional example:

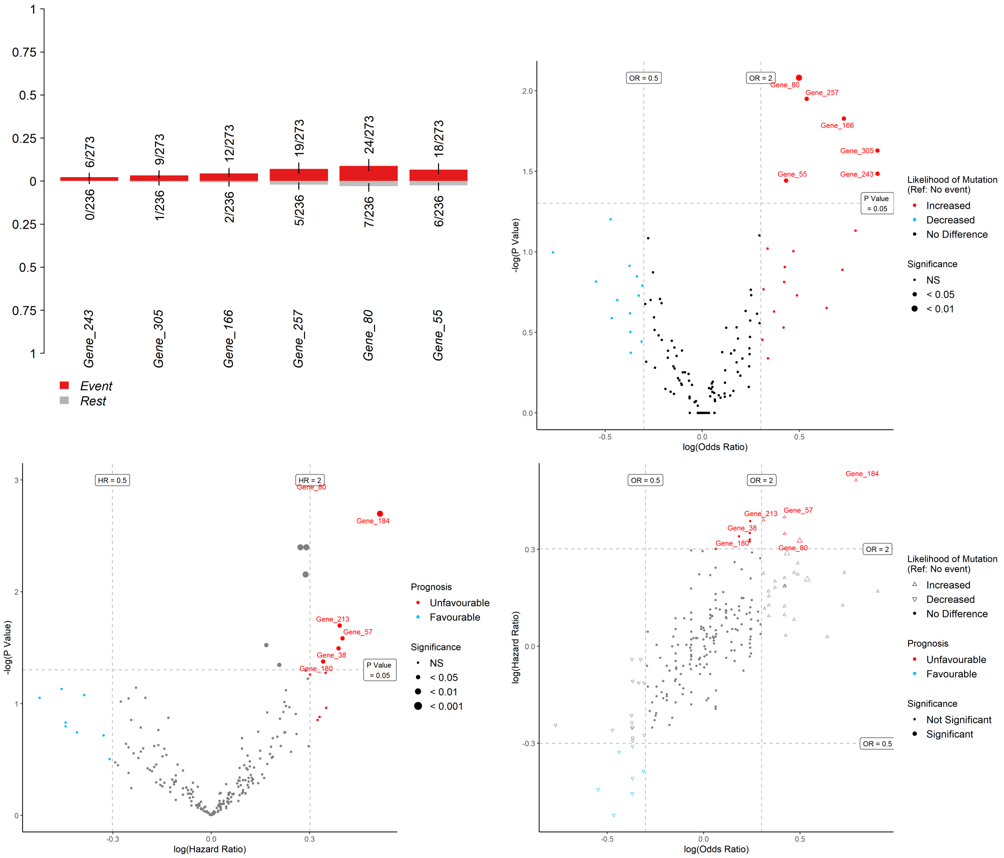


Figure 5: A - Clinical enrichment. B - Lymph plot. C - Surv plot. D - Dual plot.

mutMatrix

Example syntax:

```r
mutMatrix(
  mafA = mafPrimary,
  mafB = mafMetastatic,
  mafNameA = "Primary",
  mafNameB = "Metastatic",
  path = "omicsData/",
  varIDs = varIDs,
  data = data,
  pathways = genesTotal
)
```

geneStatus

geneStatus takes a maf file (see MafTools package) and summarises the number and variant classification of mutations in the selected genes, for each patient. If 'genes' = NULL (default), all genes in the maf are used.

Example syntax:

```r
mutDataIndex <- geneStatus(
  maf = laml,
  ID = "Tumor_Sample_Barcode",
  genes = c("FLT3", "NPM1", "DNTM3A", "TP53")
)
```

Table 3: Output of geneStatus ordered by ID

Gene	ID	variantCode	Frame_Shift_Del	Frame_Shift_Ins	In_Frame_Del	In_Frame_Ins	Missense_Mutation	Nonsense_Mutation	Splice_Site	TotalMut
NPM1	TCGA-AB-2802	8	0	2	0	0	0	0	0	2
FLT3	TCGA-AB-2812	8	0	0	0	1	0	0	0	1
NPM1	TCGA-AB-2812	8	0	1	0	0	0	0	0	1
TP53	TCGA-AB-2813	8	0	0	0	0	1	0	0	1
FLT3	TCGA-AB-2814	8	0	0	0	0	1	0	0	1
FLT3	TCGA-AB-2816	8	0	0	0	0	1	0	0	1
FLT3	TCGA-AB-2818	8	0	0	0	0	1	0	0	1
TP53	TCGA-AB-2820	8	0	1	0	0	0	0	0	1
NPM1	TCGA-AB-2824	8	0	1	0	0	0	0	0	1
FLT3	TCGA-AB-2825	8	0	0	0	0	0	0	1	1
NPM1	TCGA-AB-2825	8	0	1	0	0	0	0	0	1
TP53	TCGA-AB-2829	8	0	0	0	0	1	0	1	2
FLT3	TCGA-AB-2830	8	0	0	0	1	0	0	0	1
FLT3	TCGA-AB-2834	8	0	0	0	0	1	0	0	1
NPM1	TCGA-AB-2835	8	0	1	0	0	0	0	0	1
FLT3	TCGA-AB-2836	8	0	0	0	1	0	0	0	1
TP53	TCGA-AB-2838	8	0	0	0	0	0	0	1	1
NPM1	TCGA-AB-2839	8	0	1	0	0	0	0	0	1
FLT3	TCGA-AB-2840	8	0	0	0	1	0	0	0	1
NPM1	TCGA-AB-2848	8	0	1	0	0	0	0	0	1

Table 4: Output of geneStatus ordered by gene

Gene	ID	variantCode	Frame_Shift_Del	Frame_Shift_Ins	In_Frame_Del	In_Frame_Ins	Missense_Mutation	Nonsense_Mutation	Splice_Site	TotalMut
FLT3	TCGA-AB-2869	8	0	0	0	1	0	0	0	1
FLT3	TCGA-AB-2934	8	0	0	0	1	0	0	0	1
FLT3	TCGA-AB-2931	8	0	0	0	0	0	0	1	1
FLT3	TCGA-AB-2906	8	0	0	0	1	0	0	0	1
FLT3	TCGA-AB-2945	8	0	0	0	0	1	0	0	1
FLT3	TCGA-AB-2913	8	0	0	0	1	0	0	0	1
FLT3	TCGA-AB-2910	8	0	0	0	0	1	0	0	1
FLT3	TCGA-AB-2926	8	0	0	0	0	1	0	0	1
FLT3	TCGA-AB-2853	8	0	0	0	1	0	0	0	1
FLT3	TCGA-AB-2900	8	0	0	0	0	1	0	0	1
FLT3	TCGA-AB-2976	8	0	0	0	1	0	0	0	1
FLT3	TCGA-AB-2921	8	0	0	0	1	0	0	0	1
FLT3	TCGA-AB-2930	8	0	0	0	1	0	0	0	1
FLT3	TCGA-AB-2994	8	0	0	0	1	0	0	0	1
FLT3	TCGA-AB-2915	8	0	0	0	1	0	0	0	1
FLT3	TCGA-AB-2922	8	0	0	0	1	0	0	0	1
FLT3	TCGA-AB-2870	8	0	0	1	0	0	0	0	1
FLT3	TCGA-AB-2895	8	0	0	0	0	0	0	1	1
FLT3	TCGA-AB-2814	8	0	0	0	0	1	0	0	1
FLT3	TCGA-AB-2924	8	0	0	0	0	1	0	0	1

Utilities

assumpFun

assumpFun provides some exploratory analysis to consider some basic assumptions of data prior to running statistical tests.

assumpFun takes the following arguments:

- localData = in the format below
  - Example data can be found at utilitiesData/assumpFun/assumpFunData.csv
- columns = vector of column names for IHC markers
- pathName = path to a directory containing two subdirectories; ‘qqPlots’ and ‘Histograms’

Table 5: assumpFun example data.

TMA_ID	MarkerA	MarkerB	MarkerC	MarkerD
TMA_1	124	252	8	292
TMA_2	86	280	131	32
TMA_3	86	173	170	250

assumpFun will return a histogram and qqplot for each marker named in ‘columns’ and outputs Shapiro-Wilks test results.

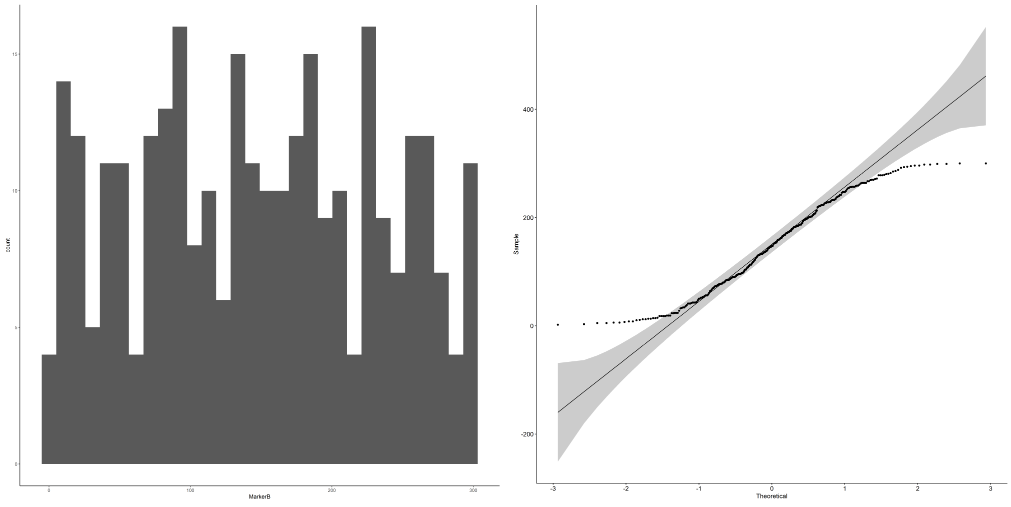


Figure 6: A - Histogram. B - qqPlot.

Table 6: Shapiro-Wilks output

Marker	statistic	p.value	sig	normal
MarkerA	0.9591511	2e-07	Sig	Not normal
MarkerB	0.9482429	0e+00	Sig	Not normal
MarkerC	0.9483907	0e+00	Sig	Not normal

groupScore

groupScore takes in data consisting of an ID, the names of two groups, variables associated with each group and a ground truth to validate the score. The data is returned with a score denoting the direction of a case towards group A (-1) or group B (1).

- data = in the format below
  - Example data can be found at utilitiesData/groupScore/groupScoreDataIn.csv
- groupNames = the names to reference groups of interest
- groupVars = a list of vectors (one per group) naming columns which relate to one or the other of the groups
  - Coded as 1 = associates with group and 0 = does not associate with group
- groupValidator = a column which provides a ground truth to validate the score
  - Optional
- groupValidatorValues = values for ‘groupValidator’ to match each group of interest
- scoreName = a name for the score
- groupNames, groupsVars and groupValidatorValues should be in the same order

Example syntax:

```
groupScoreDataOut <- groupScoreDataIn %>% groupScore(
  groupNames = c("Primary", "Metastatic"),
  groupVars = list(
    c("mutationD", "mutationE", "markerA", "markerC", "markerE"),
    c("mutationA", "mutationB", "mutationC", "markerB", "markerD")
  ),
  groupValidator = "Truth",
  groupValidatorValues = c("Primary", "Metastatic"),
  scoreName = "metastasisScore"
)
```

Table 7: groupScoreDataIn

TMA_ID	Truth	mutationA	mutationB	mutationC	mutationD	mutationE	markerA	markerB	markerC	markerD	markerE
TMA_1	Primary	0	1	1	0	1	1	0	0	1	0
TMA_2	Metastatic	1	0	0	1	0	0	0	0	0	0
TMA_3	Primary	1	1	0	1	0	0	0	0	1	0
TMA_4	Primary	1	1	0	1	0	1	0	0	1	0
TMA_5	Primary	0	0	1	0	1	0	0	0	1	0
TMA_6	Primary	0	1	0	0	0	1	1	0	0	0
TMA_7	Metastatic	1	1	1	1	0	0	1	0	0	1
TMA_8	Primary	1	1	1	1	1	1	1	0	1	0
TMA_9	Primary	1	1	0	1	0	1	1	0	0	0
TMA_10	Metastatic	1	1	1	1	0	0	0	0	0	0

Table 8: groupScoreDataOut

TMA_ID	Truth	mutationA	mutationB	mutationC	mutationD	mutationE	markerA	markerB	markerC	markerD	markerE	metastasisScore	metastasisScorePred
TMA_1	Primary	0.0	-0.1	-0.1	0.0	0.1	0.1	0.0	0	-0.1	0.0	-0.1	Metastatic
TMA_2	Metastatic	-0.1	0.0	0.0	0.1	0.0	0.0	0.0	0	0.0	0.0	0.0	NA
TMA_3	Primary	-0.1	-0.1	0.0	0.1	0.0	0.0	0.0	0	-0.1	0.0	-0.2	Metastatic
TMA_4	Primary	-0.1	-0.1	0.0	0.1	0.0	0.1	0.0	0	-0.1	0.0	-0.1	Metastatic
TMA_5	Primary	0.0	0.0	-0.1	0.0	0.1	0.0	0.0	0	-0.1	0.0	-0.1	Metastatic
TMA_6	Primary	0.0	-0.1	0.0	0.0	0.0	0.1	-0.1	0	0.0	0.0	-0.1	Metastatic
TMA_7	Metastatic	-0.1	-0.1	-0.1	0.1	0.0	0.0	-0.1	0	0.0	0.1	-0.2	Metastatic
TMA_8	Primary	-0.1	-0.1	-0.1	0.1	0.1	0.1	-0.1	0	-0.1	0.0	-0.2	Metastatic
TMA_9	Primary	-0.1	-0.1	0.0	0.1	0.0	0.1	-0.1	0	0.0	0.0	-0.1	Metastatic
TMA_10	Metastatic	-0.1	-0.1	-0.1	0.1	0.0	0.0	0.0	0	0.0	0.0	-0.2	Metastatic

groupLadder

groupScore takes in two maf files (see maftools) and a vector of genes. Output is a ladder plot showing the difference in mutational status between the two groups, such as colorectal tumours that have arisen in different parts of the colon.

- mafA = a maf file for group A
- mafB = a maf file for group B
- mafNameA = a name for mafA
- mafNameB = a name for mafB
- ID = an identifier variable
- ySize = the font size for the y axis - adjust for your number of cases
- genes = a vector of genes, which must be present in the maf files
- pathname = a path to a directory for outputs to be deposited
- outputname = a name for the output file

Example syntax:

```
groupLadder <- groupLadder(
  mafA = subsetMaf(maf = laml, clinQuery = "Overall_Survival_Status %in% '0'"),
  mafB = subsetMaf(maf = laml, clinQuery = "Overall_Survival_Status %in% '1'"),
  mafNameA = "Alive",
  mafNameB = "Dead",
  ID = "Tumor_Sample_Barcode",
  ySize = 8,
  genes = c("FLT3", "NPM1", "DNTM3A", "TP53"),
  pathname = "utilitiesData/groupLadder/",
  outputname = "ladderPlot.tiff"
)
```

Table 9: groupLadder data

Status	ID	Group	Paired	fieldChange	fieldChangeLine
Widltype	TCGA-AB-2802	DNTM3A Alive	1	No Change	solid
Widltype	TCGA-AB-2802	DNTM3A Dead	1	No Change	solid
Widltype	TCGA-AB-2802	FLT3 Alive	2	No Change	solid
Widltype	TCGA-AB-2802	FLT3 Dead	2	No Change	solid
Widltype	TCGA-AB-2802	NPM1 Alive	3	Change	dotted

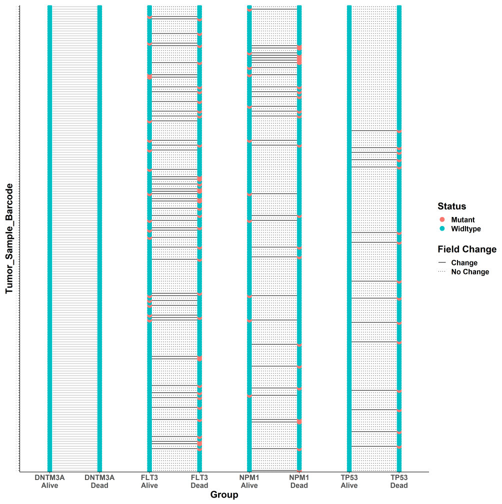


Figure 7: groupLadder plot