gClinBiomarker Package Overivew

Ning Leng, Alexey Pronin, Christina Rabe, Doug Kelkhoff, Kwame Okrah

2018-03-19

Contents

1 Overview

gClinBiomarker is an R package that allows users to easily perform biomarker analyses and generate high-quality figures and tables. It contains a set of functions covering essential biomarker analyses conducted in both oncology and non-oncology trials. It also provides an R markdown template that allows users to plug in their data set and generate a biomarker analysis report with "one-click". The report contains demographics checking, biomarker property characterization, cutoff exploration (for continuous biomarkers), subgroup analysis, and longitudinal analysis. The package takes either VADs (AdAM) or a customized csv/data frame.

gClinBiomarker is built on top of the existing Baseline R package (biomarkers). The bulk of the work done on this package is on the back-end:

- streamline the original code,
- create robust documentation, and
- modularize the package for future extensions.

This vignette provides general guidelines of using gClinBiomarker package. Example workflows are also provided in the vignette, which cover essential biomarker analyses.

2 Install and load the package

To install this package from R, use install_github() function from the devtools package

In R, type:

```r
library(devtools)
install_github("lengning/gClinBiomarker")
```

Before analysis can proceed, the gClinBiomarker package must be loaded into the working space:

```r
library(gClinBiomarker)
```

3 Input

3.1 Input requirement

Functions in gClinBiomarkers package take data frame as the input format. In this data frame, clinical data and biomarker data should all be included as columns. Rows are samples.

If only baseline biomarker analysis will be performed, the data frame should have one patient per row, without duplicated entries for any patient.

If longitudinal analysis will be performed, one patient may have multiple enteries for multiple visits.

Functions `ReadData()` and `ReadVAD()` may also be used to read in data in csv format and sas7bdat format.

3.2 Example data set: baseline only

An example data set is included in the package:

```r
head(input)
```

```
##   Patient.ID Sample.ID Arm Age Weight ECOG Sex Baseline.SLD         Country
## 1     PID001    SID001 TRT  40   61.0    0   F    550.4970 Western Europe
## 2     PID002    SID002 TRT  60   87.0    1   F    549.7751            USA
## 3     PID003    SID003 TRT  53   57.6    1   M    548.8829 Western Europe
## 4     PID004    SID004 TRT  44   57.0    0   F    549.6050            USA
## 5     PID005    SID005 TRT  49   72.5    2   F    551.5498          Other
## 6     PID006    SID006 TRT  62   76.0    0   M    549.2565          Other
##   CD8.ihc KRAS.mutant KRAS.exprs       OS      PFS  Response BEP OS.event
## 1       3      Mutant    171.25 9.691992 5.782341        PR   1        0
## 2       2   Wild Type      2.77 4.435318 1.412731        PD   1        1
## 3       3   Wild Type     61.82 9.691992 1.215606        PD   1        0
## 4       1      Mutant     71.51 1.905544 1.905544        PD   1        0
## 5      NA        <NA>        NA 2.628337 2.628337        SD   0        1
## 6       2      Mutant     44.63 4.829569 4.139630 NON CR/PD   1        0
##   PFS.event Lab_ontrt KRAS.exprs.2group
## 1         0 28.993922              >=100
## 2         1  9.890427               <100
## 3         1 26.896020               <100
## 4         1  5.421392               <100
```

```
## 5         1 14.741382            <NA>
## 6         0 16.957035            <100
```

str(input)

```
## 'data.frame':    550 obs. of  20 variables:
##  $ Patient.ID      : chr  "PID001" "PID002" "PID003" "PID004" ...
##  $ Sample.ID       : chr  "SID001" "SID002" "SID003" "SID004" ...
##  $ Arm             : Factor w/ 2 levels "CTRL","TRT": 2 2 2 2 2 2 2 2 2 1 ...
##  $ Age             : int  40 60 53 44 49 62 70 49 62 45 ...
##  $ Weight          : num  61 87 57.6 57 72.5 76 46 69.3 65 54 ...
##  $ ECOG            : int  0 1 1 0 2 0 0 1 0 1 ...
##  $ Sex             : chr  "F" "F" "M" "F" ...
##  $ Baseline.SLD    : num  550 550 549 550 552 ...
##  $ Country         : Factor w/ 3 levels "Other","USA",..: 3 2 3 2 1 1 3 3 3 1 ...
##  $ CD8.ihc         : num  3 2 3 1 NA 2 NA 1 NA NA ...
##  $ KRAS.mutant     : Factor w/ 2 levels "Mutant","Wild Type": 1 2 2 1 NA 1 NA 2 NA NA ...
##  $ KRAS.exprs      : num  171.25 2.77 61.82 71.51 NA ...
##  $ OS              : num  9.69 4.44 9.69 1.91 2.63 ...
##  $ PFS             : num  5.78 1.41 1.22 1.91 2.63 ...
##  $ Response        : Factor w/ 6 levels "CR","NE","NON CR/PD",..: 5 4 4 4 6 3 4 3 4 6 ...
##  $ BEP             : num  1 1 1 1 0 1 0 1 0 0 ...
##  $ OS.event        : int  0 1 0 0 1 0 0 0 1 0 ...
##  $ PFS.event       : int  0 1 1 1 1 0 1 0 1 0 ...
##  $ Lab_ontrt       : num  28.99 9.89 26.9 5.42 14.74 ...
##  $ KRAS.exprs.2group: Factor w/ 2 levels ">=100","<100": 1 2 2 2 NA 2 2 2 NA 1 ...
```

The columns indicates:

- Patient.ID: patient ID
- Sample.ID: sample ID
- Arm: treatment arm
- Age: age (baseline clinical variable, continuous)
- Age: baseline body Age (baseline clinical variable, continuous)
- ECOG: ECOG (baseline clinical variable, ordinal)
- Sex: gender (baseline clinical variable, binary)
- Baseline.SLD: baseline SLD (baseline clinical variable, continuous)
- Country: region (baseline clinical variable, categorical)
- CD8.ihc: CD8 IHC measurement (biomarker, protein level, ordinal)
- KRAS.mutant: KRAS mutation status (biomarker, DNA level, binary)
- KRAS.exprs: KRAS gene expression (biomarker, mRNA level, continuous)
- OS: overall survival (clinical outcome)
- OS.event: censoring indicator for OS, 1 indicates event and 0 indicates censoring (clinical outcome)
- PFS: progression free survival (clinical outcome)
- PFS.event: censoring indicator for PFS, 1 indicates event and 0 indicates censoring (clinical outcome)
- Lab_ontrt: example continuous surrogate endpoint - on treatment lab data; one record per patient
- Response: best overall overall response (clinical outcome, ordinal)
- BEP: Biomarker evaluable population indicator. 1: in biomarker evaluable population, 2: not in. (When evaluable multiple biomarkers, we may want to take the same set of patients for all evaluations. Then the can be defined as the patients who have valid measurement for all biomarkers. In this example data set, the BEP is defined as patients who have CD8 IHC data, KRAS mutation data, and KRAS expression data. If only one biomarker is of interest, BEP could be defined as patients who have valid measure for this biomarker. A BEP column is required to perform certain analyses (e.g. selection bias examination of BEP).)

4 Biomarker workflow

4.1 Continuous biomarker in a two-arm study with survival outcome

We use PFS as our primary endpoint, KRAS.exprs as the biomarker of interest.

4.1.1 Representativeness: Selection Bias of Biomarker Population

In this section, we are trying to answer the question: *Are biomarker evaluable population representative of the full population population?*

Key baseline demographics and prognostic characteristics (including stratification variables and any variables with known prognostic effect) and efficacy outcomes should be summarized by treatment groups and compared between biomarker evaluable population (BEP) and the full population. These analyses are conducted to investigate any potential selection bias associated with the availability of the biomarker (e.g. we may not get enough tissue for patients whose tumor size is small. Therefore they may be exlcuded from BEP).

4.1.1.1 Check selection bias in terms of key clinical variables, between full population and BEP

Function `SumamryVars()` can be used to perform selection bias checking of clinical variables. For example, if Age and Sex are two key clinical variables, we can perform the selection bias checking by:

```
SummaryVars(data=input, trt='Arm', subgroup='BEP', var=c('Sex','Age'),
    var.class=c('categorical','numeric'))
```

```
##                 All(CTRL)    BEP(CTRL)    All(TRT)     BEP(TRT)
## Sex             ""           ""           ""           ""
## Total (non-NA)  "182"        "81"         "368"        "176"
## NA's            "0"          "0"          "0"          "0"
## F               "89 (48.9%)" "39 (48.15%)" "184 (50%)" "90 (51.14%)"
## M               "93 (51.1%)" "42 (51.85%)" "184 (50%)" "86 (48.86%)"
## Age             ""           ""           ""           ""
## N               "182"        "81"         "368"        "176"
## Mean            "52.54"      "53.1"       "54.03"      "54.42"
## Median          "51.5"       "52"         "54"         "54"
## Min-Max         "27...85"    "32...85"    "30...89"    "33...89"
## NA's            "0"          "0"          "0"          "0"
```

`kable()` function from knitr package may be used in Rmarkdown file for better table display:

```
kable(
  SummaryVars(data=input, trt='Arm', subgroup='BEP', var=c('Sex','Age'),
      var.class=c('categorical','numeric'))
)
```

	All(CTRL)	BEP(CTRL)	All(TRT)	BEP(TRT)
Sex				
Total (non-NA)	182	81	368	176
NA's	0	0	0	0
F	89 (48.9%)	39 (48.15%)	184 (50%)	90 (51.14%)

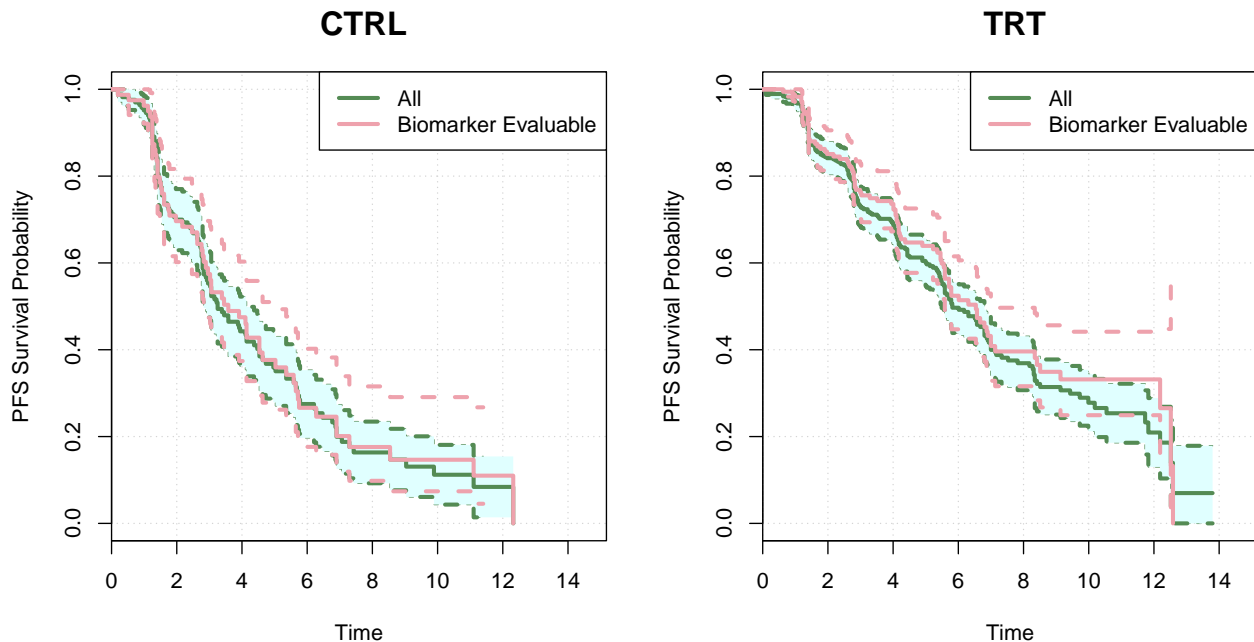|	All(CTRL)	BEP(CTRL)	All(TRT)	BEP(TRT)
M	93 (51.1%)	42 (51.85%)	184 (50%)	86 (48.86%)
Age				
N	182	81	368	176
Mean	52.54	53.1	54.03	54.42
Median	51.5	52	54	54
Min-Max	27…85	32…85	30…89	33…89
NA's	0	0	0	0

Here we specify the treatment column via parameter `trt`, and specify the BEP column via parameter `subgroup`. The comparision will be between BEP and the full population (BEP + nonBEP). If BEP vs. non BEP comparison is of interet, the user may specify parameter `compare.subgroup=TRUE`.

The clinical variables of interest can be specified via `var`. Parameter `var.class` can be used to specify class of the variable. Possible classes are "categorical","numeric","ordered.factor". If var.class is not specified, the program will define the variable class based on class of the column (class()). Note that for ordinal variable whose levels are numbers, the column need to be converted to ordered factor. Otherwise it will be treated as continuous variable.

4.1.1.2 Check whether the clinical outcome in BEP is comparable to the full population

Function `CompareKM()` can be used to compare survival outcome in BEP vs. the full population:

```
CompareKM(data=input,tte='PFS', cen='PFS.event',trt='Arm', bep='BEP')
```
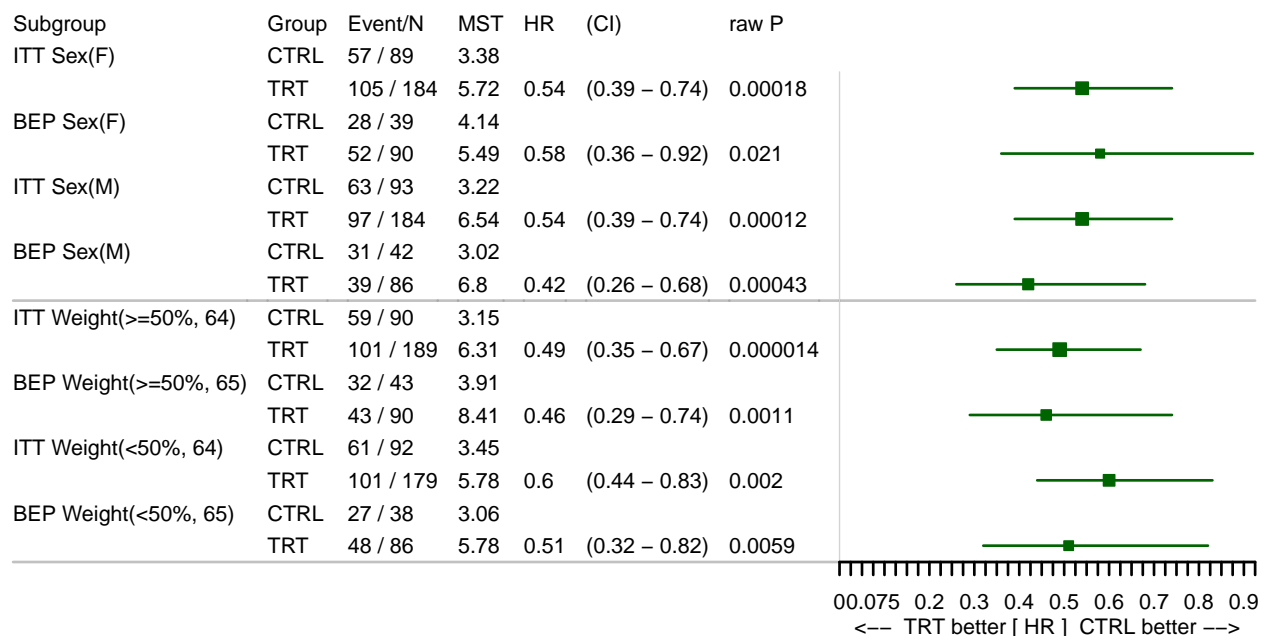

By default, the KM curve and 95% CI will be plotted for each arm. The BEP KM curve is expected to be within the full population confidence bands.

4.1.1.3 Examine whether the prognostic/predictive/null trend of key clinical variables holds in BEP

Function PlotTanForestMulti may be used to examine whether any of the key clinical variables show predictive trend in BEP:

```
PlotTabForestMulti(data=input,
                             outcome.class="survival",
                             outcome.var=c("PFS","PFS.event"),
                             trt="Arm",
                             var=c("Sex","Weight"),
                   bep="BEP",bep.indicator=1, compare.bep.itt=TRUE
                   )
```

Across arm, Compare BEP vs. All
PFS
Unadjusted, unstratified analysis

Subgroup	Group	Event/N	MST	HR	(CI)	raw P
ITT Sex(F)	CTRL	57 / 89	3.38			
	TRT	105 / 184	5.72	0.54	(0.39 – 0.74)	0.00018
BEP Sex(F)	CTRL	28 / 39	4.14			
	TRT	52 / 90	5.49	0.58	(0.36 – 0.92)	0.021
ITT Sex(M)	CTRL	63 / 93	3.22			
	TRT	97 / 184	6.54	0.54	(0.39 – 0.74)	0.00012
BEP Sex(M)	CTRL	31 / 42	3.02			
	TRT	39 / 86	6.8	0.42	(0.26 – 0.68)	0.00043
ITT Weight(>=50%, 64)	CTRL	59 / 90	3.15			
	TRT	101 / 189	6.31	0.49	(0.35 – 0.67)	0.000014
BEP Weight(>=50%, 65)	CTRL	32 / 43	3.91			
	TRT	43 / 90	8.41	0.46	(0.29 – 0.74)	0.0011
ITT Weight(<50%, 64)	CTRL	61 / 92	3.45			
	TRT	101 / 179	5.78	0.6	(0.44 – 0.83)	0.002
BEP Weight(<50%, 65)	CTRL	27 / 38	3.06			
	TRT	48 / 86	5.78	0.51	(0.32 – 0.82)	0.0059

00.075 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9
<−− TRT better [HR] CTRL better −−>

From the plot above, we can see that variable Sex doesn't show a predictive trend in full population (similar trt/ctrl HR in male and female group). However, in BEP the variable Sex has numerically smaller trt/ctrl HR in Male group. Although from the SummaryVars() result, the percentages of Male are comparable between full population and BEP, it is still possible that the male patients have better clinical outcome than the female group. This is likely due to some selection bias that we weren't be able to capture in previous steps (e.g. unmeasured variable). In downstream analyses, if the biomarker subgroup also show a predictive effect, we will need to check that whether it is confounded by Sex.

4.1.1.4 Compare treatment effect estimation in full population and in BEP, adjusted for key clinical variables

Another important summary statistic to look at is the trt/ctrl (target/reference) HR in full population and trt/ctrl HR in BEP. Although highly overlapped curves in figures generated by CompareKM() function suggests that the unadjusted HR in full population and BEP will be comparable, it will be necessary to check adjusted HR as well. The following code chunk shows how to use the function CoxTab() to fit a multivariate Cox model by including treatment variable, Sex and Age.

```
input.bep <- subset(input, BEP==1) # dataset with only BEP samples

kable(
  CoxTab(data=input,tte="PFS", cens="PFS.event",  var='Arm',
       var.class="categorical"),
caption="full population, unadjusted"
)
```

Table 2: full population, unadjusted

	HR	CI.low	CI.high	p-value	n.trt	n.ref
Arm (TRT/CTRL)	0.54	0.43	0.67	0.000000078	368	182

```
kable(
  CoxTab(data=input,tte="PFS", cens="PFS.event",  var=c('Arm','Sex',"Weight"),
       var.class=c("categorical","categorical","numeric")),
caption="full population, adjusted for Sex, Weight"
)
```

Table 3: full population, adjusted for Sex, Weight

	HR	CI.low	CI.high	p-value	n.trt	n.ref
Arm (TRT/CTRL)	0.53	0.42	0.67	0.000000056	368	182
Sex (M/F)	0.9	0.72	1.12	0.35	277	273
Weight	1	0.99	1	0.39		

```
kable(
  CoxTab(data=input.bep,tte="PFS", cens="PFS.event",  var='Arm',
       var.class="categorical"),
caption="BEP, unadjusted"
)
```

Table 4: BEP, unadjusted

	HR	CI.low	CI.high	p-value	n.trt	n.ref
Arm (TRT/CTRL)	0.49	0.36	0.69	0.000031	176	81

```
kable(
  CoxTab(data=input.bep,tte="PFS", cens="PFS.event",  var=c('Arm','Sex',"Weight"),
       var.class=c("categorical","categorical","numeric")),
caption="BEP, adjusted for Sex, Weight"
)
```

Table 5: BEP, adjusted for Sex, Weight

	HR	CI.low	CI.high	p-value	n.trt	n.ref
Arm (TRT/CTRL)	0.48	0.34	0.67	0.000014	176	81

	HR	CI.low	CI.high	p-value	n.trt	n.ref
Sex (M/F)	0.78	0.56	1.08	0.13	128	129
Weight	0.99	0.98	1	0.12		

If any selection bias is suspected, the user may consider to stratify for the imbalanced factor in downstream analysis (e.g. unstratified analysis as primary analysis and stratified analysis as sensitivity analysis).

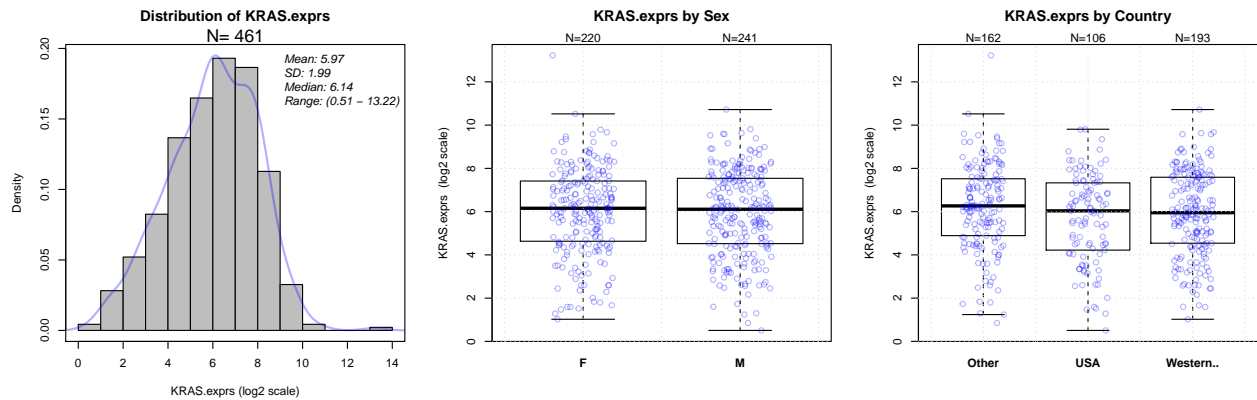### 4.1.2 Biomarker property and its association to clinical variables

Before performing cutoff exploraotry analysis, it is important to check a biomarker's property. For example, whether this biomarker has a bi-modal or multi modal distribution - if so, this biomarker may has natural cutoff.
Relationship between the biomarker and key demographic and prognostic variables should also be investigated using bivariate plots. Prognostic property of the biomarker should also be assessed, by estimates of the clinical efficacy in the control arm.

The `PlotProperty()` function could be used to generate single variate plot for biomaker and clinical covariates. It can also be used to generate bi-variate plots to investigate biomarker-clinical variable relationship.

4.1.2.1 Biomarker property and relationship to clinical variable

```
PlotProperty(data=input, biomarker.var="KRAS.exprs", biomarker.class="numeric",
             var=c("Sex", "Country"), var.class=c("categorical", "categorical"),
             log2=TRUE, par.param = list(mfrow=c(2,3)))
```


4.1.2.2 Whether the biomarker show within-arm trend

Function `PlotTabForestBiomarker()` can be used to investigate whether the biomarker shows within-arm trend:

```
input.ctrl <- subset(input, Arm=="CTRL") ## Data with only ctrl samples
res.multicut.ctrl <- PlotTabForestBiomarker(data=input.ctrl,
                                outcome.class="survival",
                                outcome.var=c("PFS","PFS.event"),
                                var="KRAS.exprs",
                                var.class="numeric",
                                percentile.cutoff=c(.25,.5,.75), main.prefix="CTRL",
                                greater=TRUE, less=FALSE)
```

```
## Some NAs in var column, will define the non NA entries as BEP
```

```
## only 1 arm; show.itt is set to FALSE
```

```
## only 1 arm; show.bep is set to FALSE
```

CTRL Within−arm Effect of Biomarker
PFS, KRAS.exprs
Unadjusted, unstratified analysis

Subgroup	Group	Event/N	MST	HR	(CI)	raw P
KRAS.exprs(>=25%, 22.63)	Less	26 / 37	4.14			
	Greater	76 / 115	3.38	1.06	(0.68 – 1.67)	0.79
KRAS.exprs(>=50%, 64)	Less	55 / 76	3.91			
	Greater	47 / 76	3.45	1.08	(0.73 – 1.59)	0.72
KRAS.exprs(>=75%, 171.86)	Less	80 / 114	3.45			
	Greater	22 / 38	4.73	0.83	(0.52 – 1.33)	0.44

```
          0.37   0.61   1.0   1.65   2.72
                        HR
```

```r
input.trt <- subset(input, Arm=="TRT") ## Data with only trt samples
res.multicut.trt <- PlotTabForestBiomarker(data=input.trt,
                            outcome.class="survival",
                            outcome.var=c("PFS","PFS.event"),
                            var="KRAS.exprs",
                            var.class="numeric",
                            percentile.cutoff=c(.25,.5,.75), main.prefix="TRT",
                            greater=TRUE, less=FALSE)
```
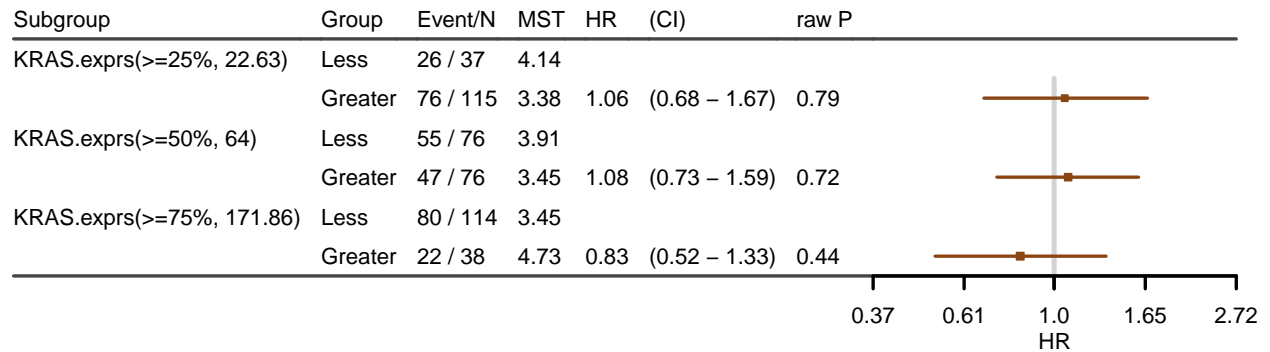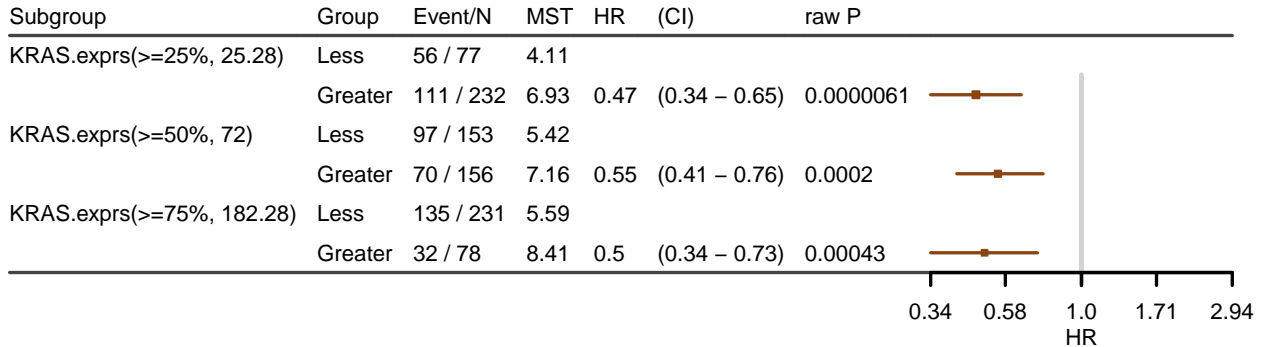
```
## Some NAs in var column, will define the non NA entries as BEP
```

```
## only 1 arm; show.itt is set to FALSE
```

```
## only 1 arm; show.bep is set to FALSE
```

```
## Warning in par(old.par): calling par(new=TRUE) with no plot
```

TRT Within–arm Effect of Biomarker
PFS, KRAS.exprs
Unadjusted, unstratified analysis

Subgroup	Group	Event/N	MST	HR	(CI)	raw P
KRAS.exprs(>=25%, 25.28)	Less	56 / 77	4.11			
	Greater	111 / 232	6.93	0.47	(0.34 – 0.65)	0.0000061
KRAS.exprs(>=50%, 72)	Less	97 / 153	5.42			
	Greater	70 / 156	7.16	0.55	(0.41 – 0.76)	0.0002
KRAS.exprs(>=75%, 182.28)	Less	135 / 231	5.59			
	Greater	32 / 78	8.41	0.5	(0.34 – 0.73)	0.00043

```
        0.34   0.58   1.0   1.71   2.94
                      HR
```

We first get dataset input.ctrl and input.trt, which only contains patients in ctrl arm and trt arm, respectively. In function `PlotTabForestBiomarker()`, time to event variable and censoring variable need to be specified via `outcome.var`. The biomarker variable can be specified via `var`. By specifying `percentile.cutoff` as `c(.25,.5,.75)` and specifying `greater` TRUE+, the program calculates within-arm HR of biomarker >= 25% vs <25% group, biomarker >= 50% vs <50% group, and biomarker >= 75% vs <75% group.

In ctrl arm, patients who have expression >= 75% have slightly better PFS - >= 75% vs <75% HR is 0.83, and the median PFS is slightly higher in >= 75% group (4.73) vs <75% (3.45).

In trt arm, within-arm trend is seen in all three comparisons - for all 3 cutoffs, patients with higher expression have longer PFS than patients with low expression. The biomarker high vs. low HR are all around 0.5.

If strong within-arm effect is seen in both arms, the biomarker may be prognostic but not predictive.

If stratified or adjusted analyses are of interest, they can be specified via parameters `covariate` and `strata`.

4.1.3   Seeking for predictive trend: Biomarker cutoff exploration/selection

Example codes in this section can be used to examine multiple candidate cutoffs for a continuous biomarker. The need for cut-off determination should be rooted in the development strategy. In general, an exhaustive search looking at all possible cut-off values is not recommended for decision making. Over-optimized cutoff using one set of clinical data may lead to hard-to-reproduce results. When determining a cutoff, biomarker property should be considered - e.g. cut at a low-dense point may be more robust to population shift. The cutoff selection should also fit the program's stratigitic considerations. There is always a prevalence-effect size trade-off, inputs from multiple functions are needed - for example whether the team is willing to take more risk in PTS (high prevalence, weaker signal) or the team is willing to target at smaller population (lower prevalence, stronger signal)

4.1.3.1   Try different cutoffs - look for consistent trend

Function `PlotTabForestBiomarker()` can also be used to investigate whether the biomarker is predictive across arm. To perform cross-arm analysis, parameter `trt` needs to be specified. By specifying `trt` ane `greater=TRUE`, trt/ctrl HR of biomarker >= 25% group, biomarker >= 50% group, and biomarker >= 75% group are calculated:

```
res.multicut <- PlotTabForestBiomarker(data=input,
                                outcome.class="survival",
```

```
                    outcome.var=c("PFS","PFS.event"),
                    trt="Arm",
                    var="KRAS.exprs",
                    var.class="numeric",
                    percentile.cutoff=c(.25,.5,.75),
                    greater=TRUE, less=FALSE,
                    show.itt=TRUE, show.bep=TRUE)
```

```
## Some NAs in var column, will define the non NA entries as BEP
```

Across–arm Effect of Biomarker
PFS, KRAS.exprs
Unadjusted, unstratified analysis

Subgroup	Group	Event/N	MST	HR	(CI)	raw P
All	CTRL	120 / 182	3.25			
	TRT	202 / 368	5.82	0.54	(0.43 – 0.67)	0.000000078
BEP	CTRL	102 / 152	3.58			
	TRT	167 / 309	6.01	0.56	(0.43 – 0.71)	0.0000035
KRAS.exprs(>=25%, 23.26)	CTRL	73 / 111	3.38			
	TRT	112 / 236	6.93	0.45	(0.33 – 0.61)	0.00000015
KRAS.exprs(>=50%, 70.52)	CTRL	43 / 70	3.58			
	TRT	73 / 162	7.16	0.42	(0.29 – 0.62)	0.0000096
KRAS.exprs(>=75%, 177.29)	CTRL	22 / 37	4.73			
	TRT	33 / 79	8.41	0.38	(0.22 – 0.66)	0.00054

```
0.22        0.47        1.0        2.13        4.55
        <–– TRT better [HR]  CTRL better ––>
```

4.1.3.2 Estimations within non-overlapping bins

Function `PlotTabForestBiomarker()` can also be used to calculate trt/ctrl HR within percentile bins. For example, by specifying `within.bin=TRUE`, trt/ctrl HR of biomarker 0-25% group, 25-50% group, 50-75% group, and 75-100% group are calculated:

```
res.withinbin <- PlotTabForestBiomarker(data=input,
                    outcome.class="survival",
                    outcome.var=c("PFS","PFS.event"),
                    trt="Arm",
                    var="KRAS.exprs",
                    var.class="numeric",
                    percentile.cutoff=c(.25,.5,.75),
                    within.bin=TRUE,
                    show.itt=TRUE, show.bep=TRUE)
```
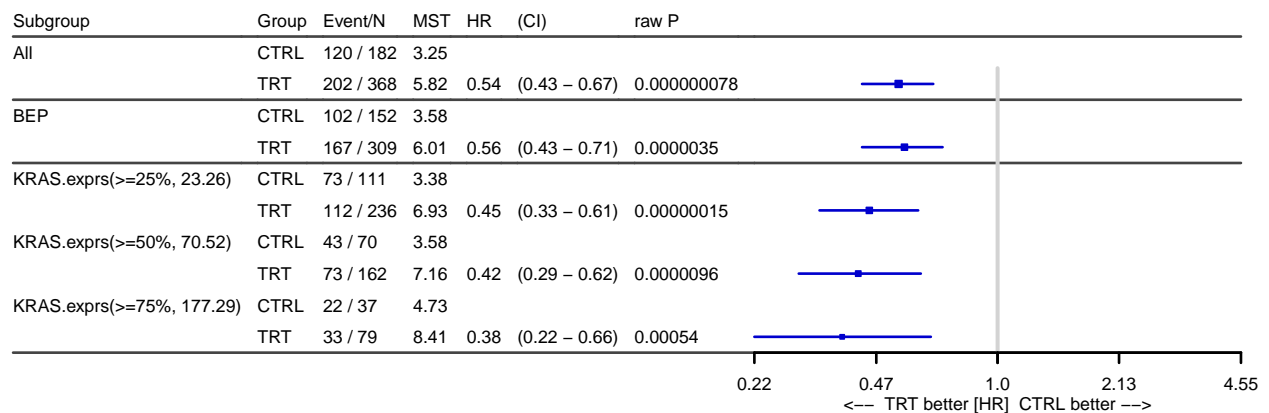
```
## Some NAs in var column, will define the non NA entries as BEP
```

```
## within.bin is TRUE, greater and less will be ignored
```

11

Across–arm Effect of Biomarker
PFS, KRAS.exprs
Unadjusted, unstratified analysis

Subgroup	Group	Event/N	MST	HR	(CI)	raw P
All	CTRL	120 / 182	3.25			
	TRT	202 / 368	5.82	0.54	(0.43 – 0.67)	0.000000078
BEP	CTRL	102 / 152	3.58			
	TRT	167 / 309	6.01	0.56	(0.43 – 0.71)	0.0000035
KRAS.exprs[0–25%, 1.41–23.26)	CTRL	29 / 41	3.94			
	TRT	55 / 73	4.07	1.04	(0.66 – 1.64)	0.87
KRAS.exprs[25–50%, 23.26–70.52)	CTRL	30 / 41	3.15			
	TRT	39 / 74	6.67	0.51	(0.31 – 0.82)	0.0057
KRAS.exprs[50–75%, 70.52–177.29)	CTRL	21 / 33	3.22			
	TRT	40 / 83	5.82	0.46	(0.27 – 0.79)	0.0049
KRAS.exprs[75–100%, 177.29–9541.51]	CTRL	22 / 37	4.73			
	TRT	33 / 79	8.41	0.38	(0.22 – 0.66)	0.00054

```
0.22        0.47        1.0        2.13        4.55
      <–– TRT better [HR]  CTRL better ––>
```

4.1.3.3   Estimations within overlapped sliding windows - STEPP plot
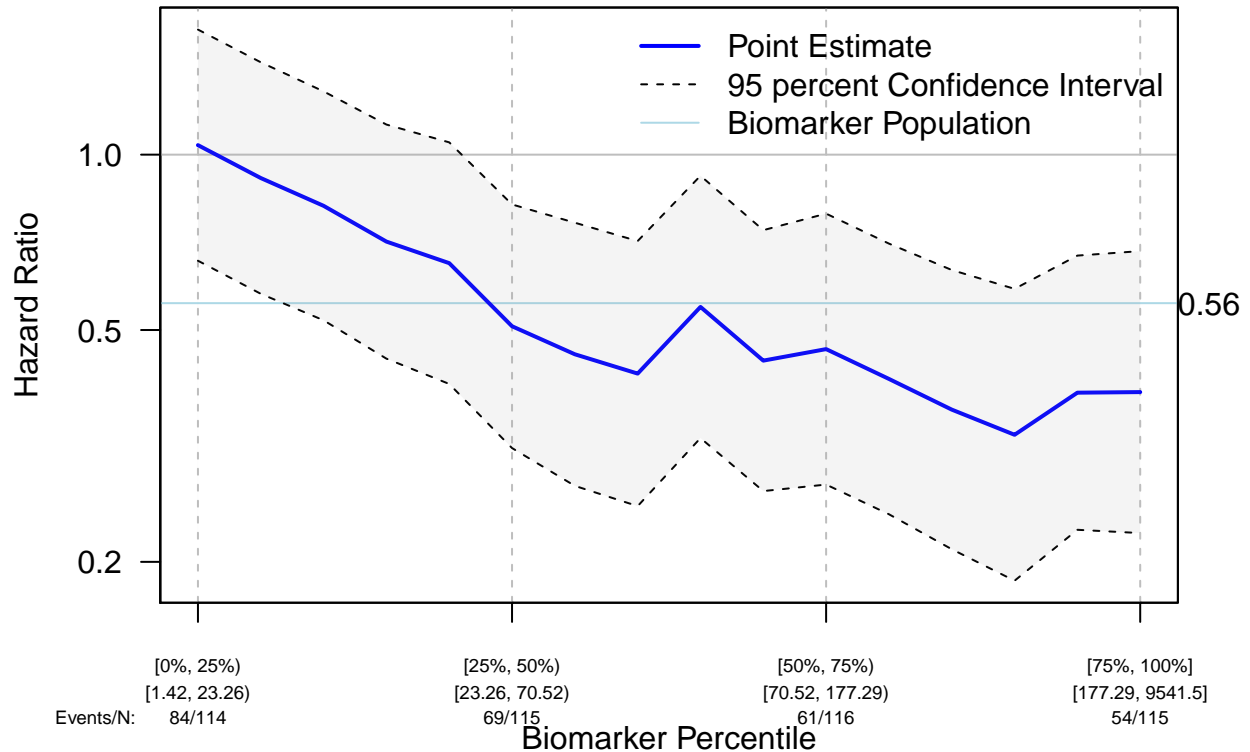
STEPP refers to Subgroup Treatment Effect Pattern Plot and it investigates relationship between biomarker and treatment effect. Only continuous biomarkers are suitable for STEPP analysis. STEPP performs treatment effect estimation on overlapping subsets of patients defined according to the biomarker level. The default setting of run.STEPP slides subgroup windows by 5% for each step and the subgroup size is 25% of the whole population.

A monotone trend is expected to be seen for an ideal biomarker.

```
stepp.out <- PlotSTEPP(data = input,
        outcome.var = c("PFS", "PFS.event"),
        outcome.class = "survival",
        trt = "Arm",
        var = "KRAS.exprs",
        placebo.code = "CTRL",
        active.code = "TRT"
)
```

```
## some NA in var column, will ignore NA entries
```

STEPP: Subgroup Treatment Effect Pattern Plot

Here we can see the trt/ctrl HR decreases as the biomarker expression increases.

4.1.4 Biomarker subgroup analysis (using selected cutoff)

When a cutoff is selected, the following functions can be used to perfrom subgroups analysis. Suppose that we decided to use 100 as cutoff.

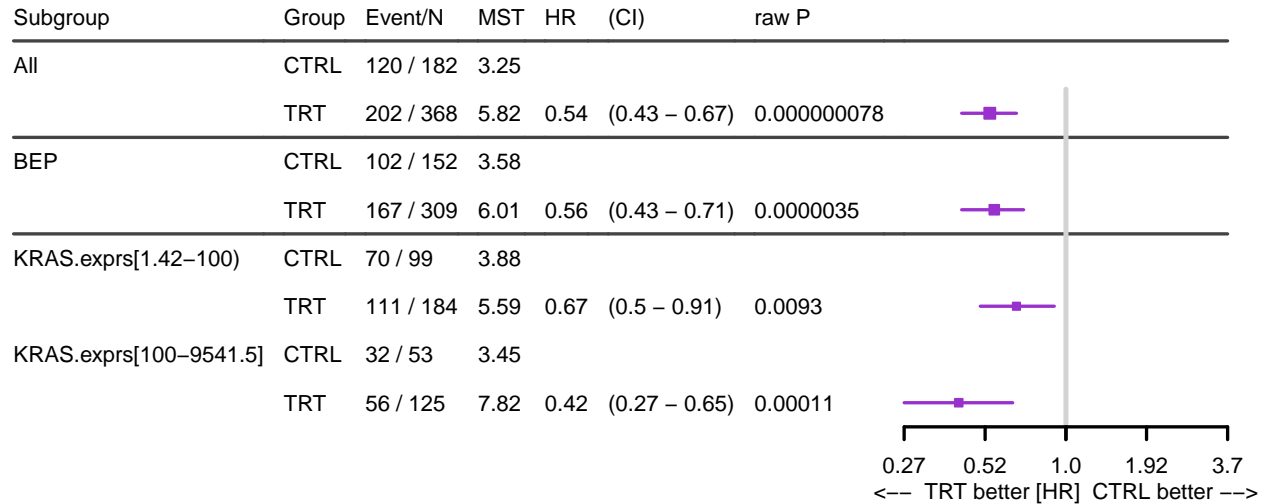#### 4.1.4.1 Estimations within each subgroup

`PlotTabForestBiomarker()` function can again be used to estimate the treatment effect in biomarker subgroups. Here given that we have a numerical cutoff, it can be specified using parameter `numerical.cutoff`. And `percentile.cutoff` can be left as NULL.

```
res.subgroups.cut <- PlotTabForestBiomarker(data=input,
                                    outcome.class="survival",
                                    outcome.var=c("PFS","PFS.event"),
                                    trt="Arm",
                                    var="KRAS.exprs",
                                    var.class="numeric",
                                    percentile.cutoff=NULL,
                                    numerical.cutoff=100, within.bin=TRUE,
                                    show.itt=TRUE, show.bep=TRUE)
```

```
## Some NAs in var column, will define the non NA entries as BEP
```

```
## within.bin is TRUE, greater and less will be ignored
```

Across−arm Effect of Biomarker
PFS, KRAS.exprs
Unadjusted, unstratified analysis

Subgroup	Group	Event/N	MST	HR	(CI)	raw P
All	CTRL	120 / 182	3.25			
	TRT	202 / 368	5.82	0.54	(0.43 − 0.67)	0.000000078
BEP	CTRL	102 / 152	3.58			
	TRT	167 / 309	6.01	0.56	(0.43 − 0.71)	0.0000035
KRAS.exprs[1.42−100)	CTRL	70 / 99	3.88			
	TRT	111 / 184	5.59	0.67	(0.5 − 0.91)	0.0093
KRAS.exprs[100−9541.5]	CTRL	32 / 53	3.45			
	TRT	56 / 125	7.82	0.42	(0.27 − 0.65)	0.00011

```
                      0.27   0.52   1.0   1.92   3.7
                      <−−  TRT better [HR]  CTRL better −−>
```

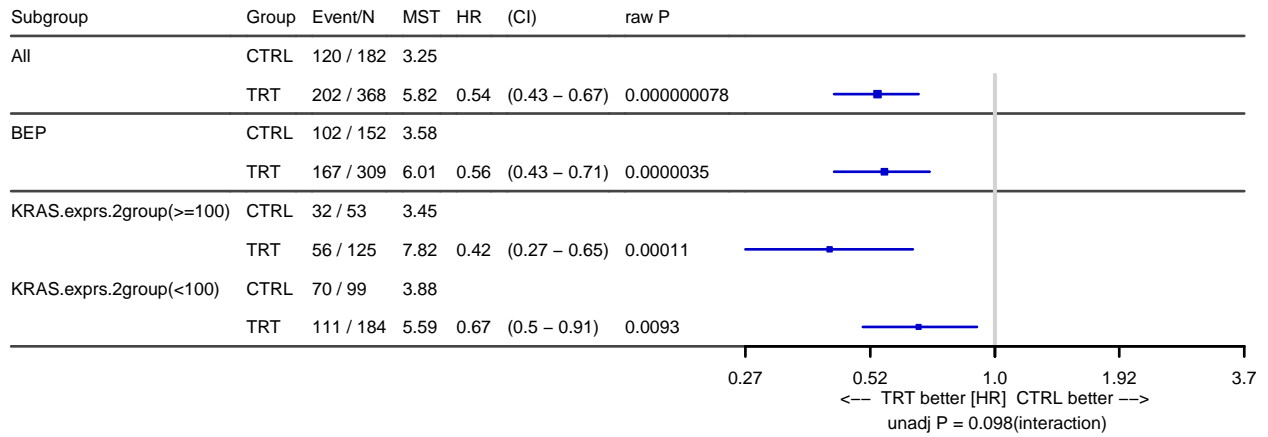This can also be done by converting the continuous marker to a binary variable:

```r
input$KRAS.exprs.2group <- ifelse(input$KRAS.exprs >= 100, ">=100","<100")
input$KRAS.exprs.2group <- factor(input$KRAS.exprs.2group, levels=c(">=100","<100")) # ">=100" as Dx+
res.subgroups.bi <- PlotTabForestBiomarker(data=input,
                                outcome.class="survival",
                                outcome.var=c("PFS","PFS.event"),
                                trt="Arm",
                                var="KRAS.exprs.2group",
                                var.class="categorical",
                                show.itt=TRUE, show.bep=TRUE)
```

```
## Some NAs in var column, will define the non NA entries as BEP
```

Across–arm Effect of Biomarker
PFS, KRAS.exprs.2group
Unadjusted, unstratified analysis

Subgroup	Group	Event/N	MST	HR	(CI)	raw P
All	CTRL	120 / 182	3.25			
	TRT	202 / 368	5.82	0.54	(0.43 – 0.67)	0.000000078
BEP	CTRL	102 / 152	3.58			
	TRT	167 / 309	6.01	0.56	(0.43 – 0.71)	0.0000035
KRAS.exprs.2group(>=100)	CTRL	32 / 53	3.45			
	TRT	56 / 125	7.82	0.42	(0.27 – 0.65)	0.00011
KRAS.exprs.2group(<100)	CTRL	70 / 99	3.88			
	TRT	111 / 184	5.59	0.67	(0.5 – 0.91)	0.0093

```
0.27        0.52      1.0       1.92      3.7
      <-- TRT better [HR] CTRL better -->
            unadj P = 0.098(interaction)
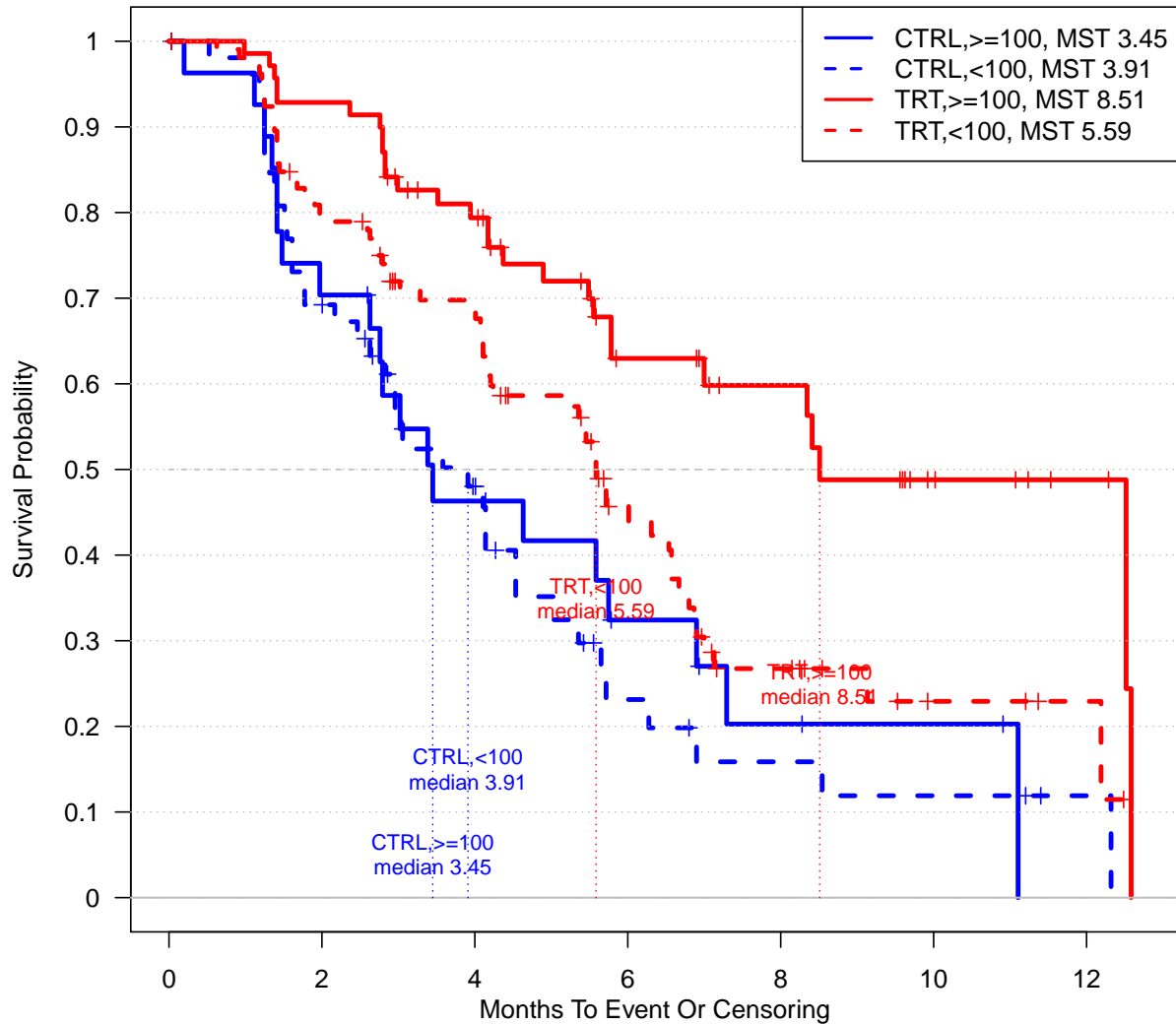```

### 4.1.4.2 KM curves

Function `PlotKM()` can be used to generate the KM curves of biomarker subgroups:

```r
PlotKM(data=input, tte="PFS",cen="PFS.event", bep="BEP",
          main="PFS BEP by treatment, by KRAS expression subgroups",
          var=c("Arm","KRAS.exprs.2group"), legend.loc="topright",
      plot.median=TRUE)
```

```
## more than one elements in 'var', trt parameter will be ignored


## 'varlist' is specified, trt and var parameters will be ignored
```

PFS BEP by treatment, by KRAS expression subgroups

CTRL,>=100	28	19	11	6	3	2	0
CTRL,<100	53	36	21	7	4	3	1
TRT,>=100	71	64	49	24	17	7	3
TRT,<100	105	82	64	27	12	4	2

```
## [1] ""
```

4.1.4.3 Check whether biomarker subgroup is confounded with key clinical variables

Functions `SummaryVars()` and `PlotTabForestMulti()` can again be used to check whether the biomarker subgroup is confounded with clinical variable. Using sex variable as an exmaple, instead of checking whether the F/M percentage is comparable in the full population vs. BEP, here we check the F/M percentage is comparable in biomarker high and low group. To perform such analysis we can specify `subgroup` as `"KRAS.exprs.2group"`, and specify `compare.subgroup=TRUE`. By doing so the program will compare the ">=100" group vs. others.
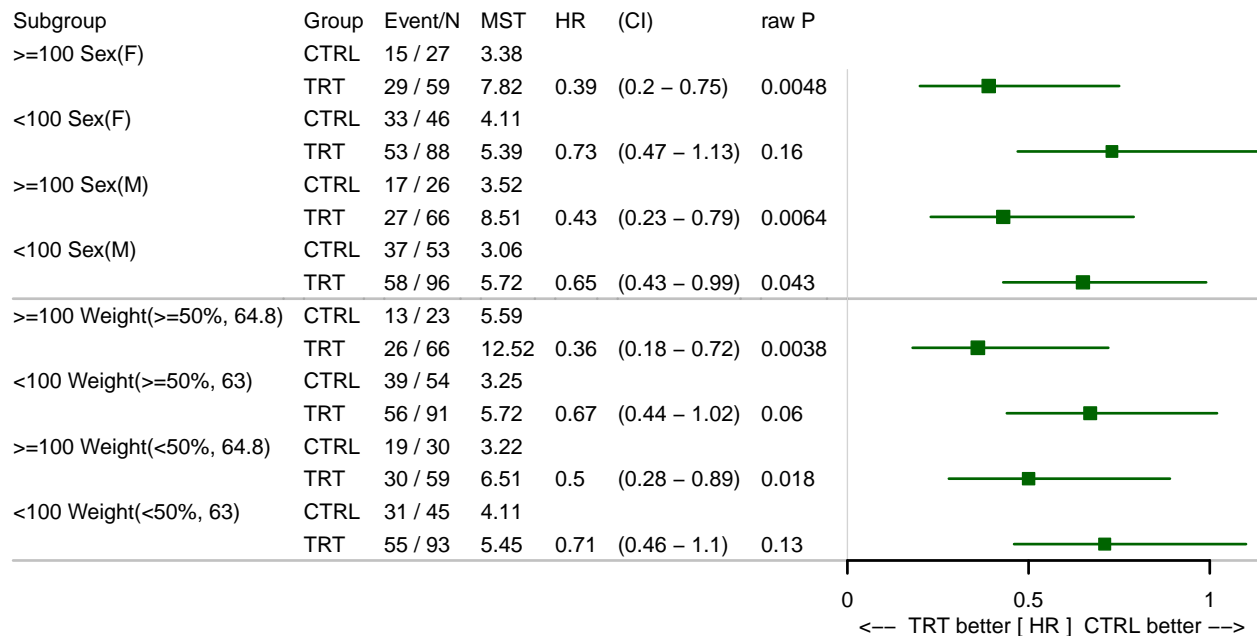
```
input.bep <- subset(input, BEP==1)
kable(
SummaryVars(data=input.bep,trt='Arm', subgroup='KRAS.exprs.2group', var=c('Sex'),
        var.class="categorical", subgroup.indicator=">=100",compare.subgroup=TRUE, subgroup.name="exprs")
)
```

	exprs_>=100(CTRL)	exprs_<100(CTRL)	exprs_>=100(TRT)	exprs_<100(TRT)
Sex				
Total (non-NA)	28	53	71	105
NA's	0	0	0	0
F	15 (53.57%)	24 (45.28%)	36 (50.7%)	54 (51.43%)
M	13 (46.43%)	29 (54.72%)	35 (49.3%)	51 (48.57%)

`PlotTabForestMulti()` function could be used to look at treatment effect in smaller subgroups defined by both biomarker and clinical variables. For example, if we are interested in the trt/ctrl HR in following groups: biomarker high Male; biomarker high female; biomarker low male; biomarker low female, etc. We may specify `subgroup` as the biomarker variable, and `var` as the clinical variables of interest. By setting `compare.bep.itt=FALSE` and `compare.subgroup=TRUE`, the program will calculate the summary statistics within each subgroup. For numerical clinical variable, it will be dichotomized by its median.

```
res.subgroup.cov <- PlotTabForestMulti(data=input,
                            outcome.class="survival",
                            outcome.var=c("PFS","PFS.event"),
                            trt="Arm",
                            var=c("Sex","Weight"),
                  compare.bep.itt=FALSE,
                  compare.subgroup=TRUE,
                  subgroup="KRAS.exprs.2group"
                  )
```

Across arm, Compare KRAS.exprs.2group subgroup
PFS
Unadjusted, unstratified analysis

Subgroup	Group	Event/N	MST	HR	(CI)	raw P	
>=100 Sex(F)	CTRL	15 / 27	3.38				
	TRT	29 / 59	7.82	0.39	(0.2 – 0.75)	0.0048	
<100 Sex(F)	CTRL	33 / 46	4.11				
	TRT	53 / 88	5.39	0.73	(0.47 – 1.13)	0.16	
>=100 Sex(M)	CTRL	17 / 26	3.52				
	TRT	27 / 66	8.51	0.43	(0.23 – 0.79)	0.0064	
<100 Sex(M)	CTRL	37 / 53	3.06				
	TRT	58 / 96	5.72	0.65	(0.43 – 0.99)	0.043	
>=100 Weight(>=50%, 64.8)	CTRL	13 / 23	5.59				
	TRT	26 / 66	12.52	0.36	(0.18 – 0.72)	0.0038	
<100 Weight(>=50%, 63)	CTRL	39 / 54	3.25				
	TRT	56 / 91	5.72	0.67	(0.44 – 1.02)	0.06	
>=100 Weight(<50%, 64.8)	CTRL	19 / 30	3.22				
	TRT	30 / 59	6.51	0.5	(0.28 – 0.89)	0.018	
<100 Weight(<50%, 63)	CTRL	31 / 45	4.11				
	TRT	55 / 93	5.45	0.71	(0.46 – 1.1)	0.13	

0   0.5   1

<–– TRT better [HR] CTRL better ––>

In this case, no treatment effect difference is observed between the Female vs. Male subgroups (as well as weight high & low).

4.2 Continuous biomarker in a single-arm study with survival outcome

We generate a data set with only patients from the treatment arm to mimic the case of single arm study:

```
input.trt <- subset(input, Arm=="TRT")
```

We use PFS as primary endpoint, KRAS.exprs as biomarker of interest. We only take the treat

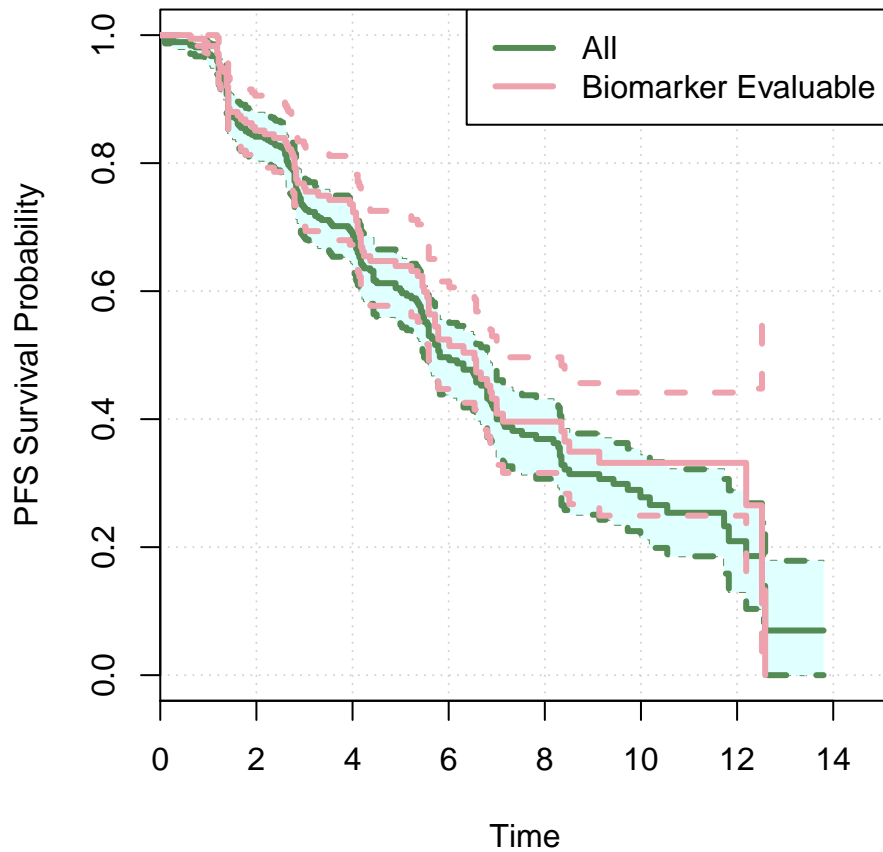### 4.2.1 Representativeness: Selection Bias of Biomarker Population

4.2.1.1 Check selection bias in terms of key clinical variables, between the full population and BEP

```
kable(
  SummaryVars(data=input.trt, trt=NULL, subgroup='BEP', var=c('Age','Sex'),
        var.class=c('numeric','categorical'))
)
```

	All	BEP
Age		
N	368	176
Mean	54.03	54.42
Median	54	54
Min-Max	30...89	33...89
NA's	0	0
Sex		
Total (non-NA)	368	176
NA's	0	0
F	184 (50%)	90 (51.14%)
M	184 (50%)	86 (48.86%)

4.2.1.2 Check whether the clinical outcome in BEP is comparable to the full population
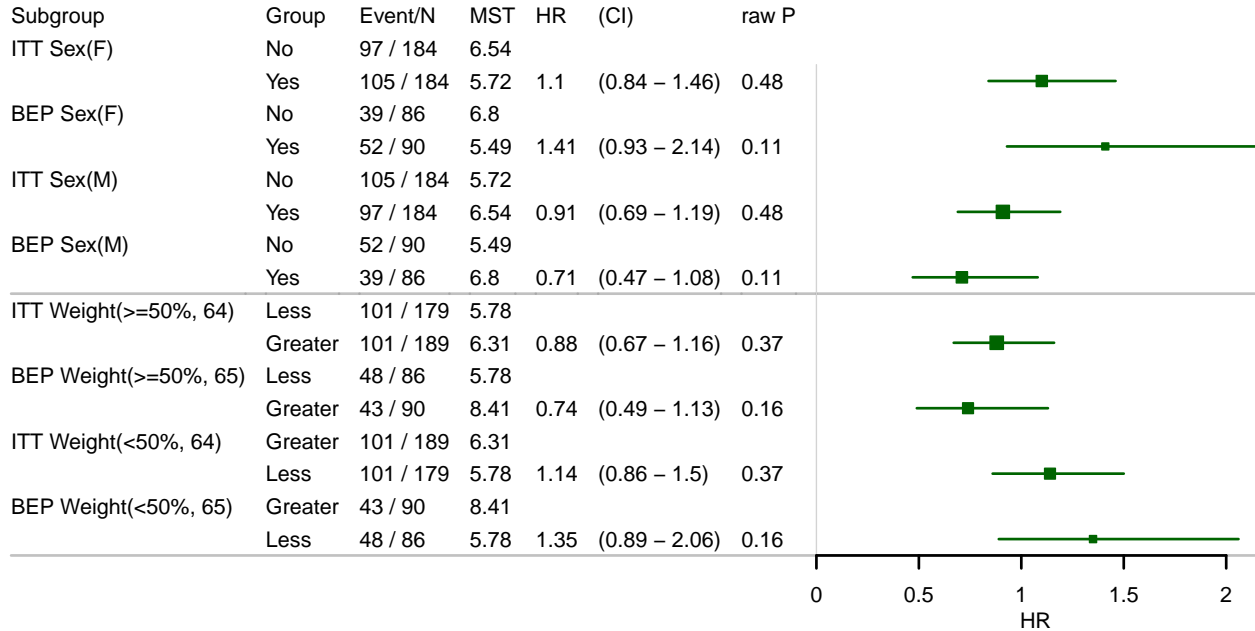
```
CompareKM(data=input.trt,tte='PFS', cen='PFS.event',trt=NULL, bep='BEP')
```

4.2.1.3 Examine whether the prognostic/null trend of key clinical variables holds in BEP

```r
PlotTabForestMulti(data=input.trt,
                            outcome.class="survival",
                            outcome.var=c("PFS","PFS.event"),
                            trt=NULL,
                            var=c("Sex","Weight"),
                  bep="BEP",bep.indicator=1, compare.bep.itt=TRUE
                            )
```

Within arm, Compare BEP vs. All
PFS
Unadjusted, unstratified analysis

Subgroup	Group	Event/N	MST	HR	(CI)	raw P
ITT Sex(F)	No	97 / 184	6.54			
	Yes	105 / 184	5.72	1.1	(0.84 – 1.46)	0.48
BEP Sex(F)	No	39 / 86	6.8			
	Yes	52 / 90	5.49	1.41	(0.93 – 2.14)	0.11
ITT Sex(M)	No	105 / 184	5.72			
	Yes	97 / 184	6.54	0.91	(0.69 – 1.19)	0.48
BEP Sex(M)	No	52 / 90	5.49			
	Yes	39 / 86	6.8	0.71	(0.47 – 1.08)	0.11
ITT Weight(>=50%, 64)	Less	101 / 179	5.78			
	Greater	101 / 189	6.31	0.88	(0.67 – 1.16)	0.37
BEP Weight(>=50%, 65)	Less	48 / 86	5.78			
	Greater	43 / 90	8.41	0.74	(0.49 – 1.13)	0.16
ITT Weight(<50%, 64)	Greater	101 / 189	6.31			
	Less	101 / 179	5.78	1.14	(0.86 – 1.5)	0.37
BEP Weight(<50%, 65)	Greater	43 / 90	8.41			
	Less	48 / 86	5.78	1.35	(0.89 – 2.06)	0.16

Age is a prognostic factor in both the full population and BEP. Sex show stronger prognostic trend in BEP, but not in the full population.

```r
input.trt.bep <- subset(input.trt, BEP==1) # dataset with only BEP samples

kable(
  CoxTab(data=input.trt,tte="PFS", cens="PFS.event",  var=c('Sex',"Weight"),
       var.class=c("categorical","numeric")),
caption="the full population, model of Sex, Weight"
)
```

Table 8: the full population, model of Sex, Weight

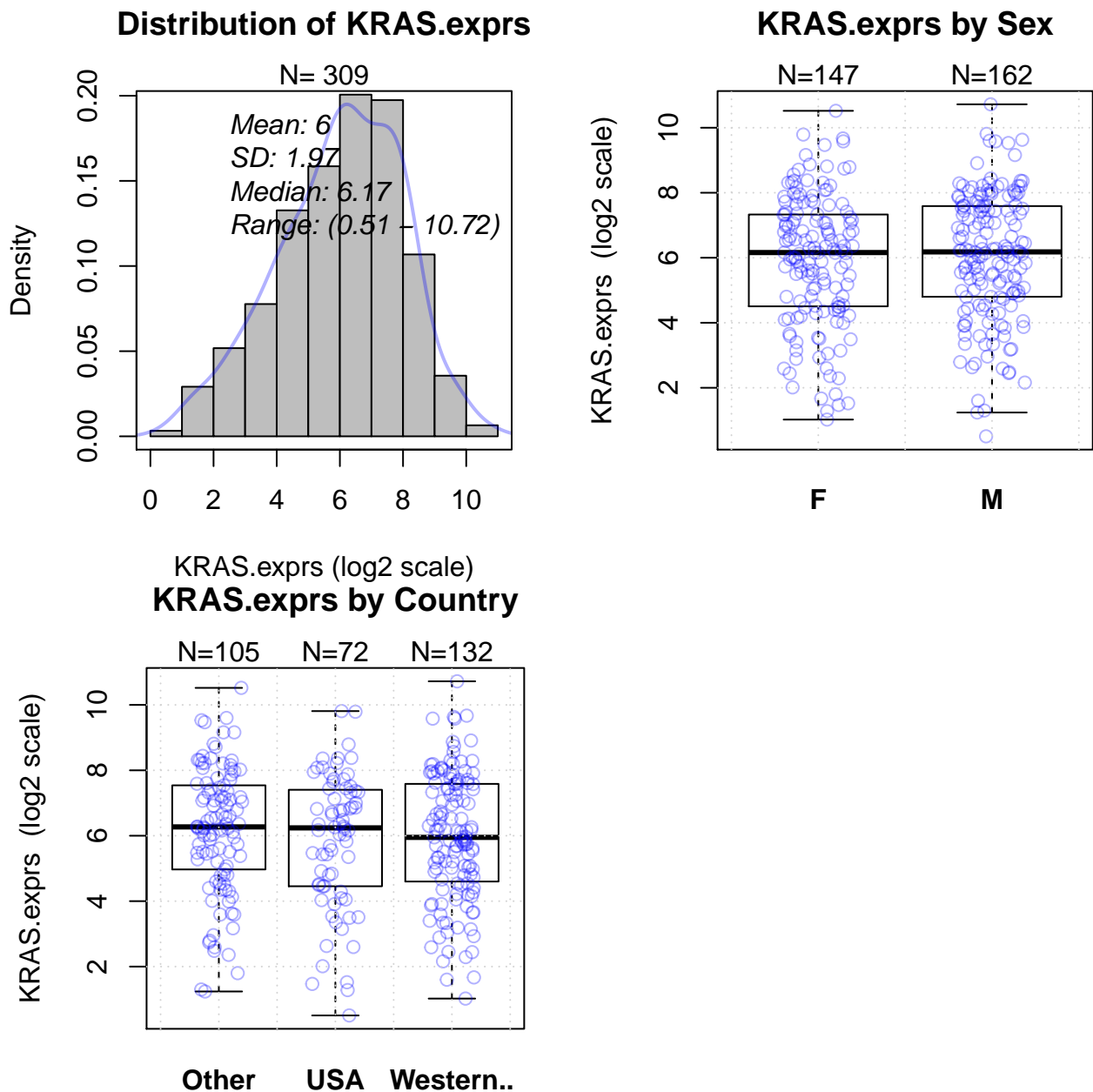|	HR	CI.low	CI.high	p-value	n.trt	n.ref
Sex (M/F)	0.91	0.69	1.2	0.49	184	184
Weight	1	0.99	1	0.33		

```r
kable(
  CoxTab(data=input.trt.bep,tte="PFS", cens="PFS.event",  var=c('Sex',"Weight"),
       var.class=c("categorical","numeric")),
caption="BEP, model of Sex, Weight"
)
```

Table 9: BEP, model of Sex, Weight

	HR	CI.low	CI.high	p-value	n.trt	n.ref
Sex (M/F)	0.7	0.46	1.06	0.09	86	90
Weight	0.99	0.97	1	0.094		

4.2.2 Biomarker property and association with clinical variables

```
PlotProperty(data=input.trt, biomarker.var="KRAS.exprs", biomarker.class="numeric",
             var=c("Sex", "Country"), var.class=c("categorical", "categorical"),
             log2=TRUE, par.param = list(mfrow=c(1,2)))
```

4.2.3 Biomarker cutoff exploration/selection

4.2.3.1 Try different cutoffs - look for consistent trend

```
res.multicut <- PlotTabForestBiomarker(data=input.trt,
                          outcome.class="survival",
                          outcome.var=c("PFS","PFS.event"),
                          trt=NULL,
                          var="KRAS.exprs",
                          var.class="numeric",
                          percentile.cutoff=c(.25,.5,.75),
                          greater=TRUE, less=FALSE,
                          show.itt=FALSE, show.bep=FALSE)
```

```
## Some NAs in var column, will define the non NA entries as BEP
```

Within–arm Effect of Biomarker
PFS, KRAS.exprs
Unadjusted, unstratified analysis

Subgroup	Group	Event/N	MST	HR	(CI)	raw P
KRAS.exprs(>=25%, 25.28)	Less	56 / 77	4.11			
	Greater	111 / 232	6.93	0.47	(0.34 – 0.65)	0.0000061
KRAS.exprs(>=50%, 72)	Less	97 / 153	5.42			
	Greater	70 / 156	7.16	0.55	(0.41 – 0.76)	0.0002
KRAS.exprs(>=75%, 182.28)	Less	135 / 231	5.59			
	Greater	32 / 78	8.41	0.5	(0.34 – 0.73)	0.00043


```
0.34   0.58   1.0   1.71   2.94
              HR
```

Within-bin analysis and STEPP are not available in single-arm analysis.

4.2.4 Biomarker subgroup analysis (using selected cutoff)

4.2.4.1 Subgroup analysis
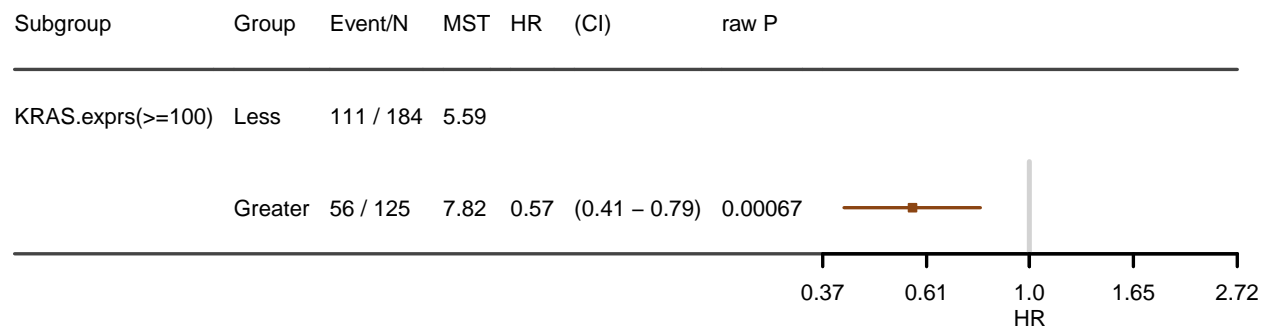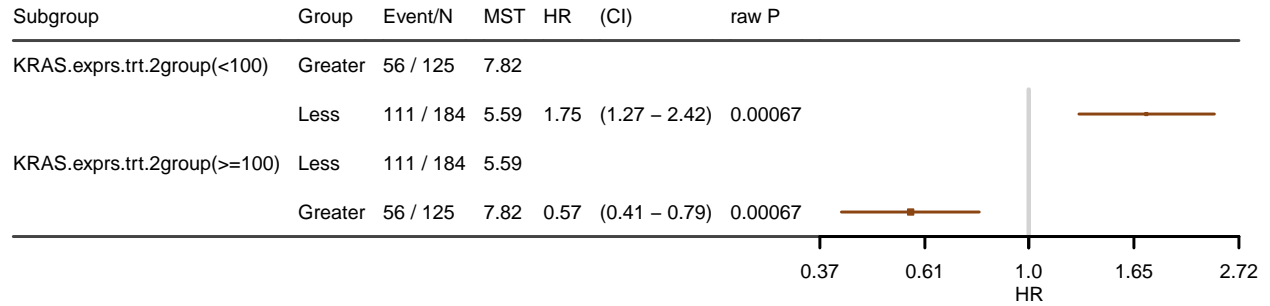
Suppose the proposed cutoff is 100

```
res.subgroups.cut <- PlotTabForestBiomarker(data=input.trt,
                          outcome.class="survival",
                          outcome.var=c("PFS","PFS.event"),
                          trt=NULL,
                          var="KRAS.exprs",
                          var.class="numeric",
```

```
                              percentile.cutoff=NULL,
                              numerical.cutoff=100,
                              greater=TRUE,less=FALSE,
                              show.itt=FALSE, show.bep=FALSE)
```

Some NAs in var column, will define the non NA entries as BEP

Within–arm Effect of Biomarker
PFS, KRAS.exprs
Unadjusted, unstratified analysis

Subgroup	Group	Event/N	MST	HR	(CI)	raw P
KRAS.exprs(>=100)	Less	111 / 184	5.59			
	Greater	56 / 125	7.82	0.57	(0.41 – 0.79)	0.00067

```
              0.37      0.61      1.0      1.65     2.72
                                  HR
```

This can also be done by converting the continous marker to a binary variable

```
input.trt$KRAS.exprs.trt.2group <- ifelse(input.trt$KRAS.exprs >= 100, ">=100","<100")
input.trt$KRAS.exprs.2group <- factor(input.trt$KRAS.exprs.2group, levels=c(">=100","<100")) # ">=100"

res.subgroups.bi <- PlotTabForestBiomarker(data=input.trt,
                                outcome.class="survival",
                                outcome.var=c("PFS","PFS.event"),
                                trt=NULL,
                                var="KRAS.exprs.trt.2group",
                                var.class="categorical",
                                show.itt=FALSE, show.bep=FALSE)
```

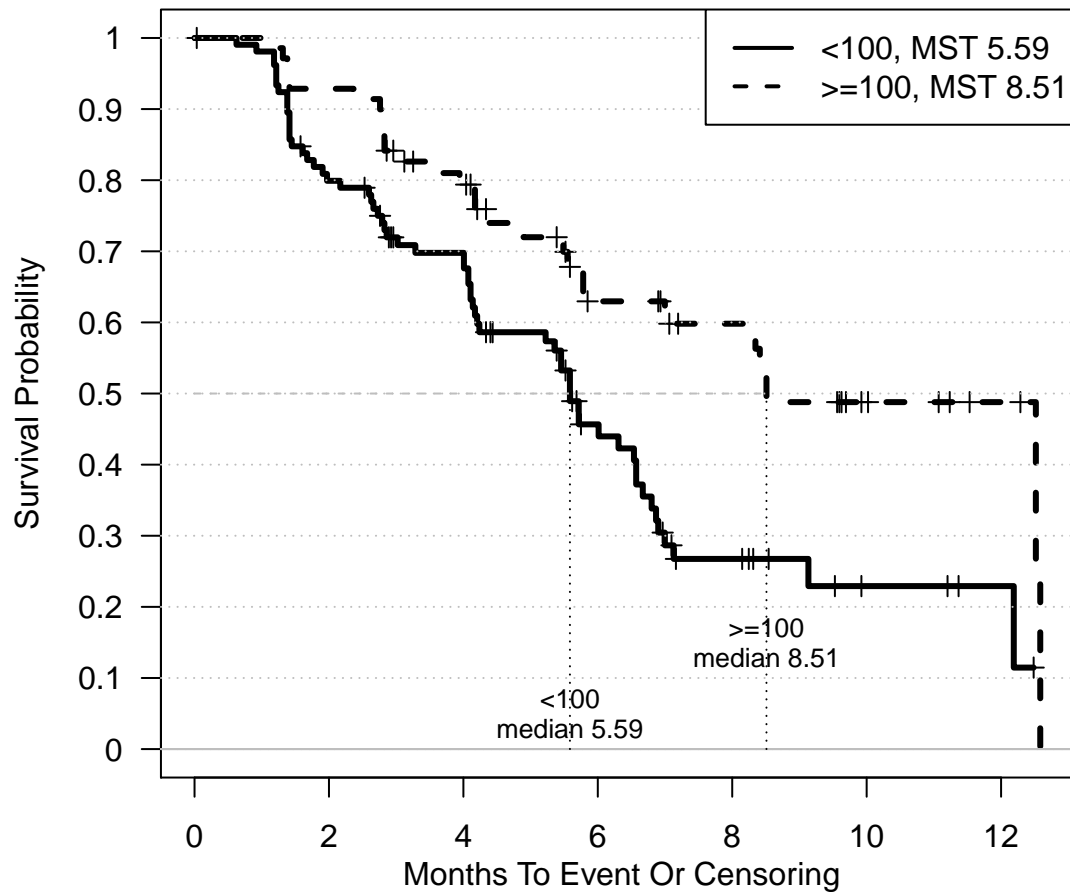## Some NAs in var column, will define the non NA entries as BEP
```

Within–arm Effect of Biomarker
PFS, KRAS.exprs.trt.2group
Unadjusted, unstratified analysis

Subgroup	Group	Event/N	MST	HR	(CI)	raw P
KRAS.exprs.trt.2group(<100)	Greater	56 / 125	7.82			
	Less	111 / 184	5.59	1.75	(1.27 – 2.42)	0.00067
KRAS.exprs.trt.2group(>=100)	Less	111 / 184	5.59			
	Greater	56 / 125	7.82	0.57	(0.41 – 0.79)	0.00067

```
          0.37      0.61     1.0      1.65     2.72
                             HR
```

4.2.4.2 KM curves

```
PlotKM(data=input.trt, tte="PFS",cen="PFS.event", bep="BEP",
        main="PFS in BEP, by KRAS expression subgroups, single arm",
        var=c("KRAS.exprs.trt.2group"), legend.loc="topright",
    plot.median=TRUE)
```

PFS in BEP, by KRAS expression subgroups, single arm

<100	105	82	64	27	12	4	2
>=100	71	64	49	24	17	7	3

```
## [1] ""
```

4.2.4.3 Check whether biomarker subgroup is confounded with clinical variables

```
input.trt.bep <- subset(input.trt, BEP==1)
kable(
SummaryVars(data=input.trt.bep,trt=NULL, subgroup='KRAS.exprs.2group', var=c('Sex'),
      var.class="categorical", subgroup.indicator=">=100",compare.subgroup=TRUE, subgroup.name="exprs")
)
```

	exprs_>=100	exprs_<100
Sex		
Total (non-NA)	71	105
NA's	0	0
F	36 (50.7%)	54 (51.43%)
M	35 (49.3%)	51 (48.57%)

4.3 Categorical biomarker with survival outcome

4.3.1 Two-arm study

Analyses in section 4.1 can also be applied to categorial biomarker, except the cutoff exploration part (section 4.1.3).

For example, for categorical biomarker KRAS.mutant, the subgroup analysis can be performed using the `PlotTabForestBiomarker()` function:

```
res.subgroups.cat <- PlotTabForestBiomarker(data=input,
                                outcome.class="survival",
                                outcome.var=c("PFS","PFS.event"),
                                trt="Arm",
                                var="KRAS.mutant",
                                var.class="categorical",
                                show.itt=TRUE, show.bep=TRUE)
```

```
## Some NAs in var column, will define the non NA entries as BEP
```

Across–arm Effect of Biomarker
PFS, KRAS.mutant
Unadjusted, unstratified analysis

Subgroup	Group	Event/N	MST	HR	(CI)	raw P
All	CTRL	120 / 182	3.25			
	TRT	202 / 368	5.82	0.54	(0.43 – 0.67)	0.000000078
BEP	CTRL	73 / 103	3.58			
	TRT	118 / 223	6.54	0.5	(0.37 – 0.66)	0.0000028
KRAS.mutant(Mutant)	CTRL	26 / 34	3.22			
	TRT	31 / 61	6.01	0.48	(0.28 – 0.8)	0.0055
KRAS.mutant(Wild Type)	CTRL	47 / 69	3.94			
	TRT	87 / 162	6.57	0.5	(0.35 – 0.72)	0.00019

```
0.28        0.53      1.0       1.89      3.57
      <-- TRT better [HR]  CTRL better -->
           unadj P = 0.87(interaction)
```

The forest plot above provides HR estimates across two arms

4.3.2 Single-arm study

Similarly, most functions in section 4.2 can be applied to categorical biomarker analysis (except the cutoff exploration part in section 4.2.3). For example, taking the `input.trt` data set which only contains patients from treatment arm:

```
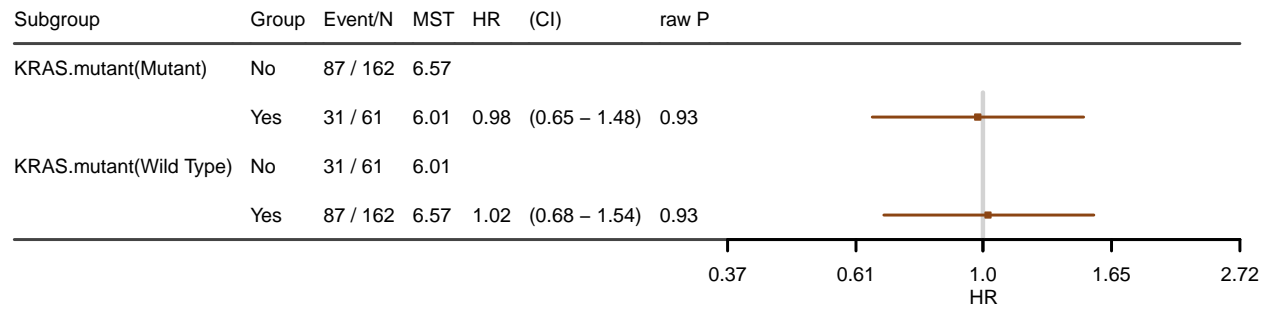res.subgroups.cat <- PlotTabForestBiomarker(data=input.trt,
                                outcome.class="survival",
                                outcome.var=c("PFS","PFS.event"),
                                trt=NULL,
                                var="KRAS.mutant",
                                var.class="categorical")
```

```
## Some NAs in var column, will define the non NA entries as BEP
```

```
## only 1 arm; show.itt is set to FALSE
```

```
## only 1 arm; show.bep is set to FALSE
```

Within−arm Effect of Biomarker
PFS, KRAS.mutant
Unadjusted, unstratified analysis

Subgroup	Group	Event/N	MST	HR	(CI)	raw P
KRAS.mutant(Mutant)	No	87 / 162	6.57			
	Yes	31 / 61	6.01	0.98	(0.65 − 1.48)	0.93
KRAS.mutant(Wild Type)	No	31 / 61	6.01			
	Yes	87 / 162	6.57	1.02	(0.68 − 1.54)	0.93

```
        0.37      0.61      1.0      1.65      2.72
                            HR
```

4.4 Categorical outcome (e.g. response)

4.4.1 Check selection bias in terms of key clinical variables, between full population and BEP

Majority of functions from section 4.1.1 can be directly applied on analyses of categorical endpoints.

Function `SumamryVars()` in section 4.1.1 can still be used to perform selection bias checking of clinical variables. To check whether clinical outcome in BEP is comparable to the full population, user can use function `PlotRspBar()` when the clinical outcome is response assessment. The parameter `rep.levels` may be used to define order of the levels.

```
Rsp.out <- PlotRspBar(data=input, outcome.var="Response", binary=FALSE,
rsp.levels=c("CR", "PR","SD","NON CR/PD", "PD","NE"),trt="Arm",
compare.bep.itt=TRUE, bep = "BEP")
```

```
## entries with missing outcome.var are removed! 524 entries left
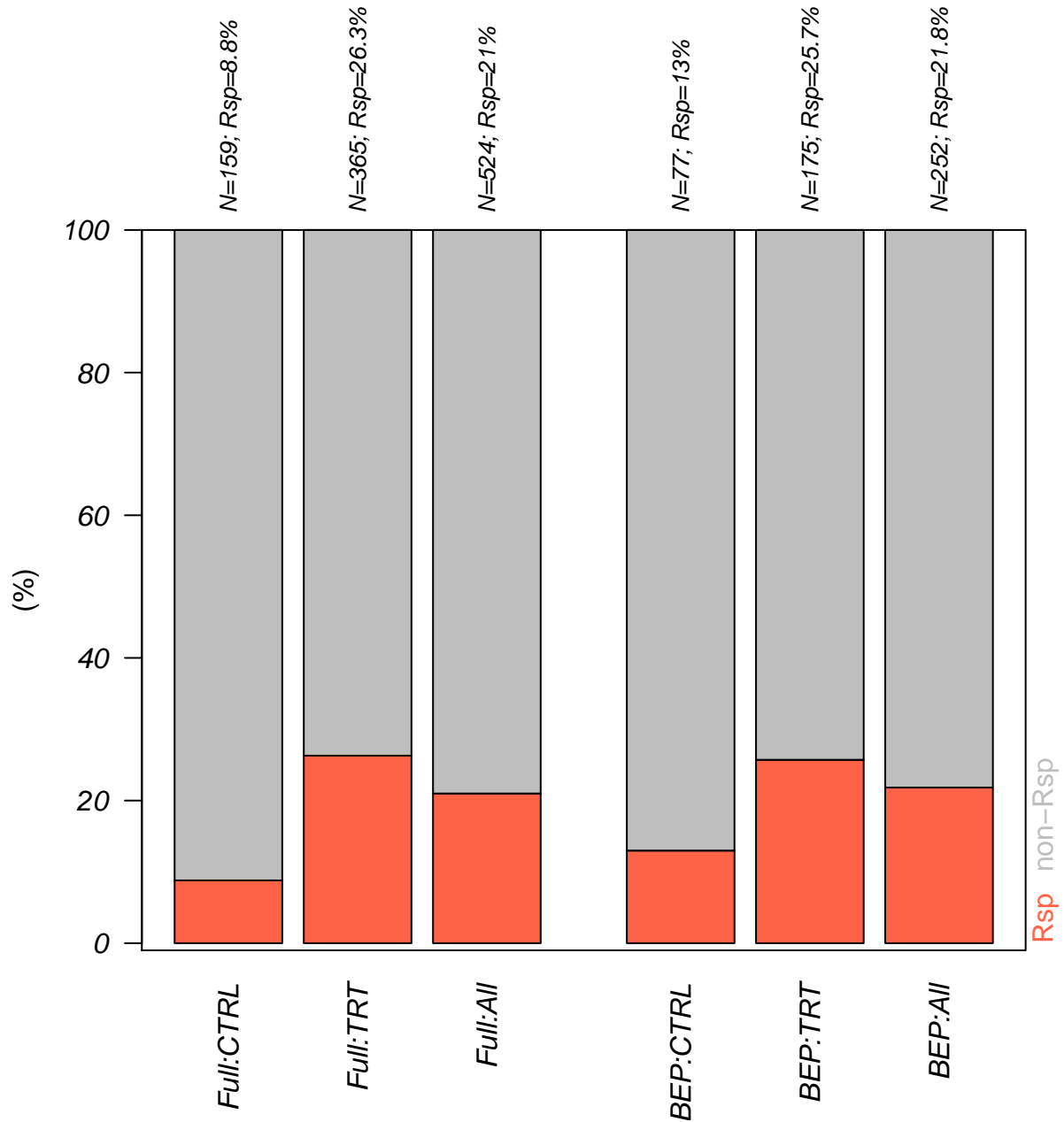```

## Association of response rate



The summary statistics can be obtained by:

```
kable(Rsp.out$count,caption="count")
```

Table 11: count

	CR	PR	SD	NON CR/PD	PD	NE
Full:CTRL	0	14	64	22	58	1
Full:TRT	6	90	145	49	74	1
Full:All	6	104	209	71	132	2
BEP:CTRL	0	10	30	7	30	0

	CR	PR	SD	NON CR/PD	PD	NE
BEP:TRT	4	41	72	27	31	0
BEP:All	4	51	102	34	61	0

```r
kable(round(Rsp.out$perc,2), caption="percentage")
```

Table 12: percentage

	CR	PR	SD	NON CR/PD	PD	NE
Full:CTRL	0.00	0.09	0.40	0.14	0.36	0.01
Full:TRT	0.02	0.25	0.40	0.13	0.20	0.00
Full:All	0.01	0.20	0.40	0.14	0.25	0.00
BEP:CTRL	0.00	0.13	0.39	0.09	0.39	0.00
BEP:TRT	0.02	0.23	0.41	0.15	0.18	0.00
BEP:All	0.02	0.20	0.40	0.13	0.24	0.00

To plot counts instead of percentage, one can specify the `plot.count` parameter to TRUE:

```r
PlotRspBar(data=input, outcome.var="Response", binary=FALSE,
rsp.levels=c("CR", "PR","SD","NON CR/PD", "PD","NE"),trt="Arm",
compare.bep.itt=TRUE, bep = "BEP", plot.count=TRUE)
```

```
## entries with missing outcome.var are removed! 524 entries left
```

Association of response rate

To summarize the response into two classes - responder vs. non-responder, user may set the parameter `binary` to TRUE. Parameter `rsp.response` (`rsp.nonresponse`) can be used to define levels to be considered as responder (non-responder). Patients whose response outcome don't fall into these defined levels will be elminated from the analysis.

```
Rsp.out.2 <- PlotRspBar(data=input, outcome.var="Response", binary=TRUE,
rsp.response = c("CR","PR"),
rsp.nonresponse = c("SD", "PD","NON CR/PD","NE"),trt="Arm",
compare.bep.itt=TRUE, bep = "BEP")
```

```
## entries with missing outcome.var are removed! 524 entries left
```

Association of response rate

The summary statistics can be obtained by:

```
kable(Rsp.out.2$count,caption="count")
```

Table 13: count

	Rsp	non-Rsp
Full:CTRL	14	145
Full:TRT	96	269
Full:All	110	414
BEP:CTRL	10	67

	Rsp	non-Rsp
BEP:TRT	45	130
BEP:All	55	197

```
kable(round(Rsp.out.2$perc,2), caption="percentage")
```

Table 14: percentage

	Rsp	non-Rsp
Full:CTRL	0.09	0.91
Full:TRT	0.26	0.74
Full:All	0.21	0.79
BEP:CTRL	0.13	0.87
BEP:TRT	0.26	0.74
BEP:All	0.22	0.78

To plot counts instead of percentage, one can specify the `plot.count` parameter to TRUE:

```
PlotRspBar(data=input, outcome.var="Response", binary=TRUE,
rsp.response = c("CR","PR"),
rsp.nonresponse = c("SD", "PD","NON CR/PD","NE"),trt="Arm",
compare.bep.itt=TRUE, bep = "BEP", plot.count = TRUE)
```

```
## entries with missing outcome.var are removed! 524 entries left
```

Association of response rate

4.4.2 Continuous biomarker in a two-arm study

Most of functions in workflow in section 4.1 may be used in analyses of continuous biomarker in a 2-arm study with categorical response. The following sections show examples of categorical-response specific use cases

4.4.2.1 Check within-arm trend

To check the within-arm trend associated with categorical outcome, user may use the same function `PlotTabForestBiomarker()` as in section 4.1.2.2. Instead of specifying outcome.class as "survival", user can specify outcome.class as "binary".

```
input.ctrl <- subset(input, Arm=="CTRL") ## Data with only ctrl samples
res.multicut.ctrl <- PlotTabForestBiomarker(data=input.ctrl,
                                    outcome.class=c("binary"),
                                    outcome.var=c("Response"),
                                    rsp.cat = TRUE,
                                    rsp.response = c("CR","PR"),
                                    rsp.nonresponse = c("SD", "PD","NON CR/PD","NE",NA),
                                    var="KRAS.exprs",
                                    var.class="numeric",
                                    percentile.cutoff=c(.25,.5,.75), main.prefix="CTRL",
                                    greater=TRUE, less=FALSE)
```

```
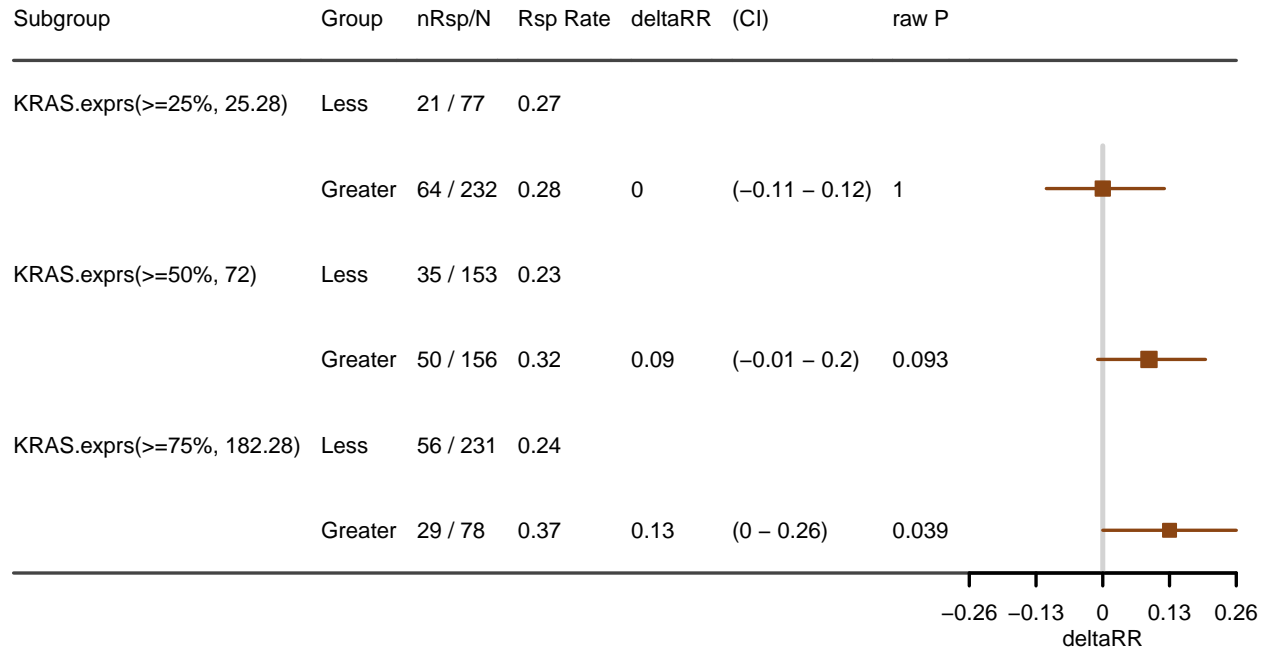## Covariate adjustment and stratification are not supported for binary outcome
```

```
## Some NAs in var column, will define the non NA entries as BEP
```

```
## only 1 arm; show.itt is set to FALSE
```

```
## only 1 arm; show.bep is set to FALSE
```

```
## Warning in prop.test(c(r1, r2), c(n1, n2), conf.level = 1 - alpha,
## correct = fit.para[["prop.test.use.continuity.correction"]]): Chi-squared
## approximation may be incorrect
```

```
## Warning in prop.test(c(r1, r2), c(n1, n2), conf.level = 1 - alpha,
## correct = fit.para[["prop.test.use.continuity.correction"]]): Chi-squared
## approximation may be incorrect
```

CTRL Within−arm Effect of Biomarker
Response, KRAS.exprs
Unadjusted, unstratified analysis

Subgroup	Group	nRsp/N	Rsp Rate	deltaRR	(CI)	raw P
KRAS.exprs(>=25%, 22.63)	Less	1 / 37	0.03			
	Greater	12 / 115	0.1	0.08	(−0.02 − 0.17)	0.26
KRAS.exprs(>=50%, 64)	Less	3 / 76	0.04			
	Greater	10 / 76	0.13	0.09	(−0.01 − 0.19)	0.082
KRAS.exprs(>=75%, 171.86)	Less	9 / 114	0.08			
	Greater	4 / 38	0.11	0.03	(−0.1 − 0.15)	0.87

−0.19 −0.1   0   0.1 0.19
deltaRR

```
input.trt <- subset(input, Arm=="TRT") ## Data with only trt samples
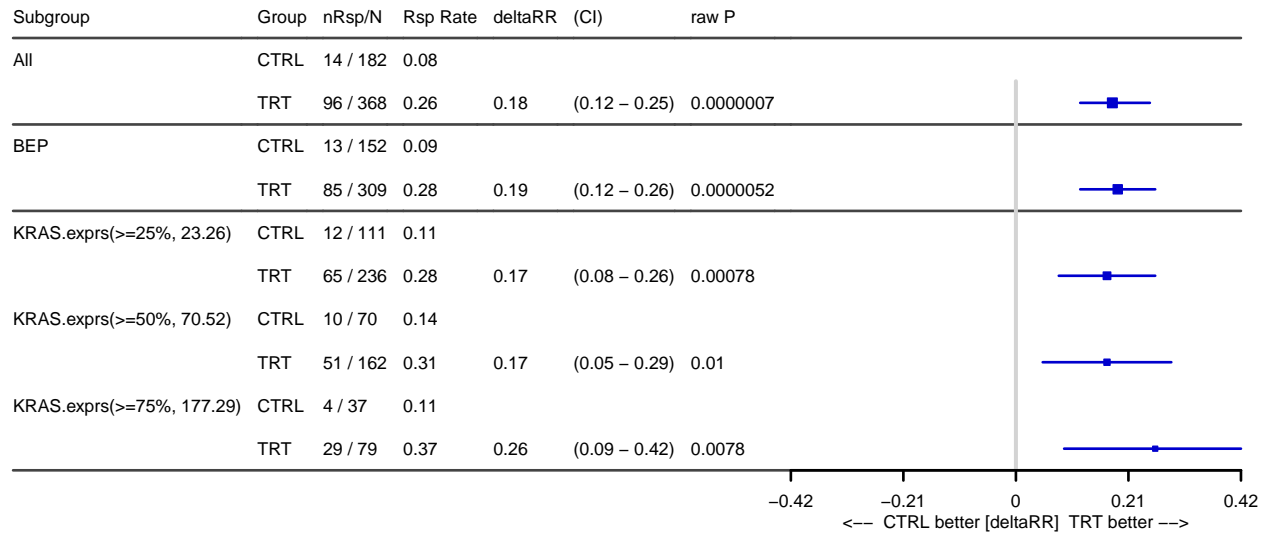res.multicut.trt <- PlotTabForestBiomarker(data=input.trt,
                          outcome.class=c("binary"),
                          outcome.var=c("Response"),
                          rsp.cat = TRUE,
                          rsp.response = c("CR","PR"),
                          rsp.nonresponse = c("SD", "PD","NON CR/PD","NE",NA),
                          var="KRAS.exprs",
                          var.class="numeric",
                          percentile.cutoff=c(.25,.5,.75), main.prefix="TRT",
                          greater=TRUE, less=FALSE)
```

Covariate adjustment and stratification are not supported for binary outcome

Some NAs in var column, will define the non NA entries as BEP

only 1 arm; show.itt is set to FALSE

only 1 arm; show.bep is set to FALSE

Warning in par(old.par): calling par(new=TRUE) with no plot

TRT Within−arm Effect of Biomarker
Response, KRAS.exprs
Unadjusted, unstratified analysis

Subgroup	Group	nRsp/N	Rsp Rate	deltaRR	(CI)	raw P	
KRAS.exprs(>=25%, 25.28)	Less	21 / 77	0.27				
	Greater	64 / 232	0.28	0	(−0.11 − 0.12)	1	
KRAS.exprs(>=50%, 72)	Less	35 / 153	0.23				
	Greater	50 / 156	0.32	0.09	(−0.01 − 0.2)	0.093	
KRAS.exprs(>=75%, 182.28)	Less	56 / 231	0.24				
	Greater	29 / 78	0.37	0.13	(0 − 0.26)	0.039	

```
            −0.26  −0.13    0    0.13   0.26
                         deltaRR
```

The summary statistics shown in the forest plot is

deltaRR = [Response Rate in biomarker high] - [Response Rate in biomarker low]

To calculate the response rate, user needs to binarize the response classes into two classes.

Parameter `rsp.response` (`rsp.nonresponse`) can be used to define levels to be considered as responder (non-responder). Patients whose response outcome don't fall into these defined levels will be elminated from the analysis.

In both arms, we observe that patients with higher expression tend to have higher response rate. This indicates that in terms of response, KRAS.exprs shows prognostic trend.

4.4.2.2 Seeking for predictive trend: cutoff exploration/selection

Similar to the example in section 4.1.3,

function `PlotTabForestBiomarker()` also can be used for cutoff exploration.

```
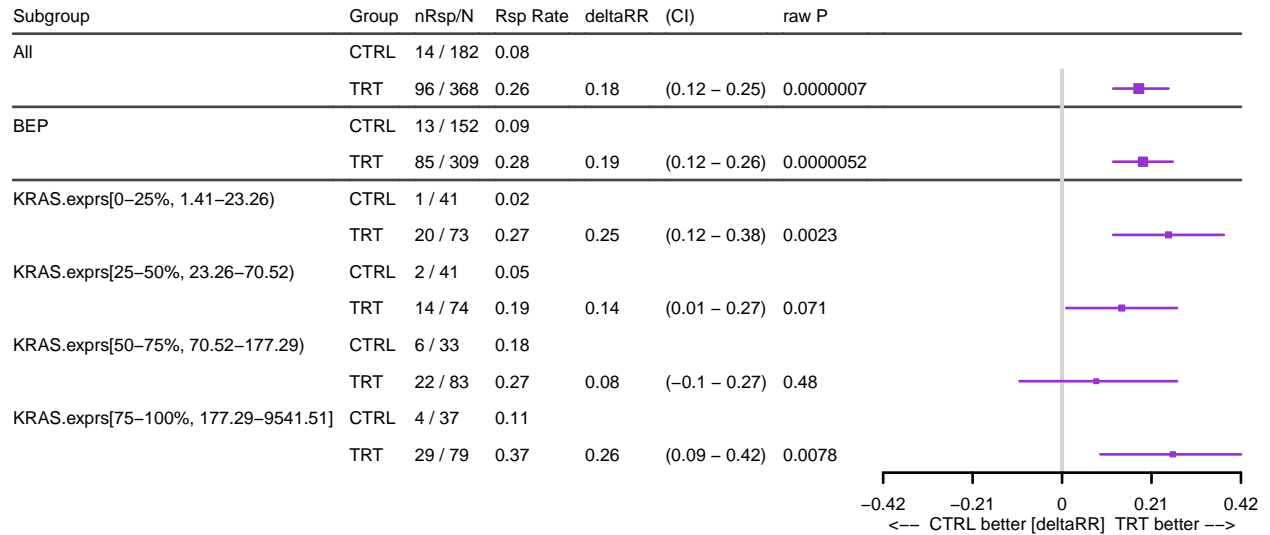res.multicut.2arm <- PlotTabForestBiomarker(data=input,
                            outcome.class=c("binary"),
                            outcome.var=c("Response"),
                            trt = "Arm",
                            rsp.cat = TRUE,
                            rsp.response = c("CR","PR"),
                            rsp.nonresponse = c("SD", "PD","NON CR/PD","NE",NA),
                            var="KRAS.exprs",
                            var.class="numeric",
```

```
                            percentile.cutoff=c(.25,.5,.75),
                            greater=TRUE, less=FALSE)
```

Covariate adjustment and stratification are not supported for binary outcome

Some NAs in var column, will define the non NA entries as BEP

Across–arm Effect of Biomarker
Response, KRAS.exprs
Unadjusted, unstratified analysis

Subgroup	Group	nRsp/N	Rsp Rate	deltaRR	(CI)	raw P
All	CTRL	14 / 182	0.08			
	TRT	96 / 368	0.26	0.18	(0.12 – 0.25)	0.0000007
BEP	CTRL	13 / 152	0.09			
	TRT	85 / 309	0.28	0.19	(0.12 – 0.26)	0.0000052
KRAS.exprs(>=25%, 23.26)	CTRL	12 / 111	0.11			
	TRT	65 / 236	0.28	0.17	(0.08 – 0.26)	0.00078
KRAS.exprs(>=50%, 70.52)	CTRL	10 / 70	0.14			
	TRT	51 / 162	0.31	0.17	(0.05 – 0.29)	0.01
KRAS.exprs(>=75%, 177.29)	CTRL	4 / 37	0.11			
	TRT	29 / 79	0.37	0.26	(0.09 – 0.42)	0.0078

–0.42   –0.21   0   0.21   0.42
<–– CTRL better [deltaRR] TRT better ––>

In across-arm analysis, for a given subgroup, the delta RR is calculated as [Response Rate in treatment] - [Response Rate in control]

From the figure above, we can observe that no matter where we set the biomarker cutoff, the biomarker high group always shows similar delta RR as the BEP delta RR (as well as full population delta RR)

We can further examine the predictive trend by looking at the within-bin analysis:

```
res.multicut.2arm <- PlotTabForestBiomarker(data=input,
                            outcome.class=c("binary"),
                            outcome.var=c("Response"),
                            trt = "Arm",
                            rsp.cat = TRUE,
                            rsp.response = c("CR","PR"),
                            rsp.nonresponse = c("SD", "PD","NON CR/PD","NE",NA),
                            var="KRAS.exprs",
                            var.class="numeric",
                            percentile.cutoff=c(.25,.5,.75),
                            within.bin=TRUE)
```

Covariate adjustment and stratification are not supported for binary outcome

Some NAs in var column, will define the non NA entries as BEP

```
## within.bin is TRUE, greater and less will be ignored
```

Across–arm Effect of Biomarker
Response, KRAS.exprs
Unadjusted, unstratified analysis

Subgroup	Group	nRsp/N	Rsp Rate	deltaRR	(CI)	raw P
All	CTRL	14 / 182	0.08			
	TRT	96 / 368	0.26	0.18	(0.12 – 0.25)	0.0000007
BEP	CTRL	13 / 152	0.09			
	TRT	85 / 309	0.28	0.19	(0.12 – 0.26)	0.0000052
KRAS.exprs[0–25%, 1.41–23.26)	CTRL	1 / 41	0.02			
	TRT	20 / 73	0.27	0.25	(0.12 – 0.38)	0.0023
KRAS.exprs[25–50%, 23.26–70.52)	CTRL	2 / 41	0.05			
	TRT	14 / 74	0.19	0.14	(0.01 – 0.27)	0.071
KRAS.exprs[50–75%, 70.52–177.29)	CTRL	6 / 33	0.18			
	TRT	22 / 83	0.27	0.08	(–0.1 – 0.27)	0.48
KRAS.exprs[75–100%, 177.29–9541.51]	CTRL	4 / 37	0.11			
	TRT	29 / 79	0.37	0.26	(0.09 – 0.42)	0.0078

```
      -0.42      -0.21        0        0.21      0.42
         <--  CTRL better [deltaRR]  TRT better -->
```

This indicates that there is no clear predictive trend in terms of the response outcome - the result doesn't show that the higher (lower) the expression, the more benefit in treatment arm compare to control arm.

4.4.2.3 Subgroup analysis

Suppose that we are interested in looking at subgroup analysis using cutoff = 100, function `PlotRspBar()` can again be used to perform the subgroup analysis. To do so, user can input the dichotomized biomarker variable into parameter `var`, and specify `compare.var=TRUE`

```
input$KRAS.exprs.2group <- ifelse(input$KRAS.exprs >= 100, ">=100","<100")
input$KRAS.exprs.2group <- factor(input$KRAS.exprs.2group, levels=c(">=100","<100")) # ">=100" as Dx+
Rsp.out.2 <- PlotRspBar(data=input, outcome.var="Response", binary=TRUE,
rsp.response = c("CR","PR"),
rsp.nonresponse = c("SD", "PD","NON CR/PD","NE"),trt="Arm",
 bep = "BEP",compare.var=TRUE, var="KRAS.exprs.2group")
```

```
## entries with missing outcome.var are removed! 524 entries left
```

Association of response rate

The subgroup analysis again confirmed the prognostic effect - when biomarker is high (>=100), patients in both TRT and CTRL arms have improved response outcome.

Compare to results shown in section 4.1, we see that teh KRAS.exprs show predictive trend in terms of PFS, but prognostic trend in terms of response outcome. Although those results are based on simulated data, in empirical study it is also possible that the biomarker shows different trend when associating to different endpoints (e.g. response vs. PFS, PFS vs. OS, investigator PFS vs. IRF PFS). Therefore when analyzing biomarker data, it is important to make sure the analysis is using appropriate endpoint.

4.4.3   Other scenarios

4.4.3.1   Continuous biomarker in a single-arm study

See section 4.2 and section 4.2.2

4.4.3.2 Categorical biomarker in a two-arm study

See section 4.3.1 and section 4.4.2

4.4.3.3 Categorical biomarker in a single-arm study

See section 4.3.2 and section 4.4.2

4.5 Continuous outcome

The similar workflow can be applied to cases with continuous endpoint.

For example, function `PlotTabForestBiomarker()` can also be used for continuous endpoint. Using Lab_ontrt as continuous surrogate endpoint:

```
res.multicut <- PlotTabForestBiomarker(data=input,
                            outcome.class="continuous",
                            outcome.var="Lab_ontrt",
                            trt="Arm",
                            var="KRAS.exprs",
                            var.class="numeric",
                            percentile.cutoff=c(.25,.5,.75),
                            greater=TRUE, less=FALSE,
                            show.itt=TRUE, show.bep=TRUE)
```

```
## Stratification is not supported for continuous outcome
```

```
## Some NAs in var column, will define the non NA entries as BEP
```

Across–arm Effect of Biomarker
Lab_ontrt, KRAS.exprs
Unadjusted, unstratified analysis

Subgroup	Group	Mean	delta	(CI)	raw P
All	CTRL	19.22			
	TRT	22.17	2.95	(1.64 – 4.26)	0.000012
BEP	CTRL	19.7			
	TRT	22.48	2.77	(1.37 – 4.18)	0.00012
KRAS.exprs(>=25%, 23.26)	CTRL	19.99			
	TRT	22.58	2.59	(0.95 – 4.23)	0.002
KRAS.exprs(>=50%, 70.52)	CTRL	19.87			
	TRT	22.88	3.01	(0.97 – 5.05)	0.004
KRAS.exprs(>=75%, 177.29)	CTRL	19.61			
	TRT	24.66	5.04	(2.32 – 7.77)	0.00038

```
                    −7.77 −3.88   0   3.88  7.77
                    <−−  CTRL better [delta]  TRT better
```

The forest plot above shows the difference in mean between treatment arm and control arm.

Function `PlotSTEPP()` can also be used for continuous endpoint:

```
stepp.out <- PlotSTEPP(data = input,
        outcome.var = "Lab_ontrt",
        outcome.class = "continuous",
        trt = "Arm",
        var = "KRAS.exprs",
        placebo.code = "CTRL",
        active.code = "TRT"
)
```

```
## some NA in var column, will ignore NA entries
```

STEPP: Subgroup Treatment Effect Pattern Plot

Mean Difference

2.8

[0%, 25%)	[25%, 50%)	[50%, 75%)	[75%, 100%]
[1.42, 23.26)	[23.26, 70.52)	[70.52, 177.29)	[177.29, 9541.5]
N: 114	N: 115	N: 116	N: 115

Biomarker Percentile

Point Estimate
95 percent Confidence Interval
Biomarker Population

4.6 Longitudinal analysis (e.g. PD biomarker)

4.6.1 Longitudinal Sample Data (`longbmkr`)

```
head(longbmkr)
```

```
## # A tibble: 6 x 8
##     pid   trt    sex     age   edu     bmkr    vm       ep
##   <int> <fctr> <fctr>  <dbl> <dbl>    <dbl> <dbl>    <dbl>
## 1     1     0      f 46.24018    11 1.405647     0 161.8545
## 2     1     0      f 46.24018    11 1.405647     6 245.1100
## 3     1     0      f 46.24018    11 1.405647    12 253.2516
## 4     1     0      f 46.24018    11 1.405647    18 341.5073
## 5     1     0      f 46.24018    11 1.405647    24 355.0323
## 6     1     0      f 46.24018    11 1.405647    30 453.0348
```

Table 15: Longitudinal Biomarker Sample Data

Column Name	English Name	Description
pid	PatientID	Unique Patient Identifier
trt	Treatment	Binary Classification of Treatment (1) or Control (0)
sex	Sex	Male ("m") or Female ("f") binary classification of sex
age	Age	Age of Patient
edu	Education	Years of Education for the Patient
bmkr	Biomarker Measurement	Biomarker Measurement|Mock Biomarker Measurement Reading
vm	Visitation Month	Timepoint that data was collected
ep	Endpoint Measurement	Mock Endpoint Measurement Reading

4.6.2 Baseline Plots of Subsetted Longitudinal Data

The baseline data for longitudinal data can be accessed by subsetting the data for only the visitation that took place at month 0. This can be done in a couple ways. In base-R (without any extra packages), this can be done by slicing the data on the condition.

```
baseline_data <- longbmkr[longbmkr['vm'] == 0,]

kable(
    head(baseline_data),
    caption = 'longitudinal data subset by visit month to produce baseline data'
)
```

Table 16: longitudinal data subset by visit month to produce baseline data

pid	trt	sex	age	edu	bmkr	vm	ep
1	0	f	46.24018	11	1.4056474	0	161.8545
3	0	f	66.74478	12	2.3057527	0	157.4129
4	1	m	46.76299	12	2.1865537	0	184.9616
5	1	f	67.24289	14	0.8460209	0	179.3956

44

pid	trt	sex	age	edu	bmkr	vm	ep
6	1	m	55.74087	11	1.1060709	0	204.9887
7	1	f	62.23837	13	1.4616352	0	173.9895

Baseline data from longitudinal data can be used for any of the plots described herein similar to the examples above.

4.6.3 Longitudinal Plots with `PlotLong`

The `PlotLong` function is called with only a small set of arguments. The first is the data to plot and the second is an aesthetic mapping (using the function `aes()`) to map columns to visual outputs. In it's most simple form, you can plot the distribution of endpoint readings over the timecourse, for the entire patient population. By default, the `PlotLong` function plots a line with points at the timepoints for which data exists with a ribbon showing the mean +/- standard error at that timestamp.

```
PlotLong(longbmkr, x=vm, y=ep)
```


4.6.3.1 Tukey Hinges and Whiskers

However, we may be more interested in the population distribution progression where we'd want to plot the Tukey boxplot hinges and whiskers over time. From this plot, we can see that although there is a slow rise in population mean, the patient response also becomes more distributed. You can specify, using the `fun.data` argument, exactly what the ribbons represent by using any keyword that can be passed to `stat_summary_funs` including "mean_sd" (mean +/- standard deviation), "mean_sd" (mean +/- standard

error of the mean), "median_iqr" (median +/- interquartile range), "tukey" (tukey boxplot hinges and whiskers), "quartiles" and "deciles".

```
PlotLong(longbmkr, x=vm, y=ep, fun.data = 'tukey',
        xlab = 'Visitation Month',
        ylab = 'Biomarker Endpoint',
        labs.title = 'Tukey Whiskers and Hinges')
```


4.6.3.2 Plotting Subpopulations (Two-Armed Trials)

By giving aesthetic mappings to flags for subpopulations, you can easily vary the colors by adding aesthetics. In this case, we map the line color and ribbon fill to the treatment arm data, allowing a two arm trial representation.

```
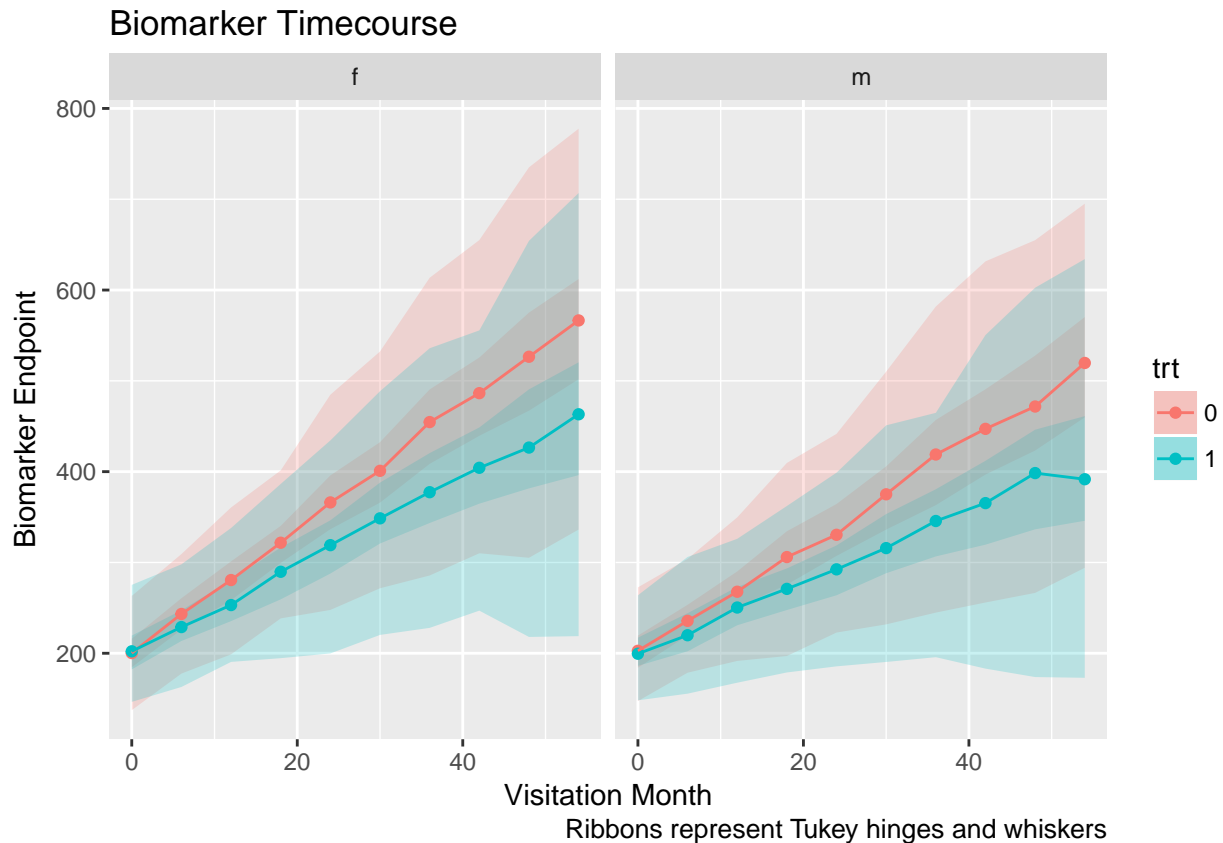PlotLong(longbmkr, x=vm, y=ep, color=trt, fill=trt,
        fun.data = 'tukey',
        xlab = 'Visitation Month',
        ylab = 'Biomarker Endpoint',
        labs.title = 'Biomarker Timecourse',
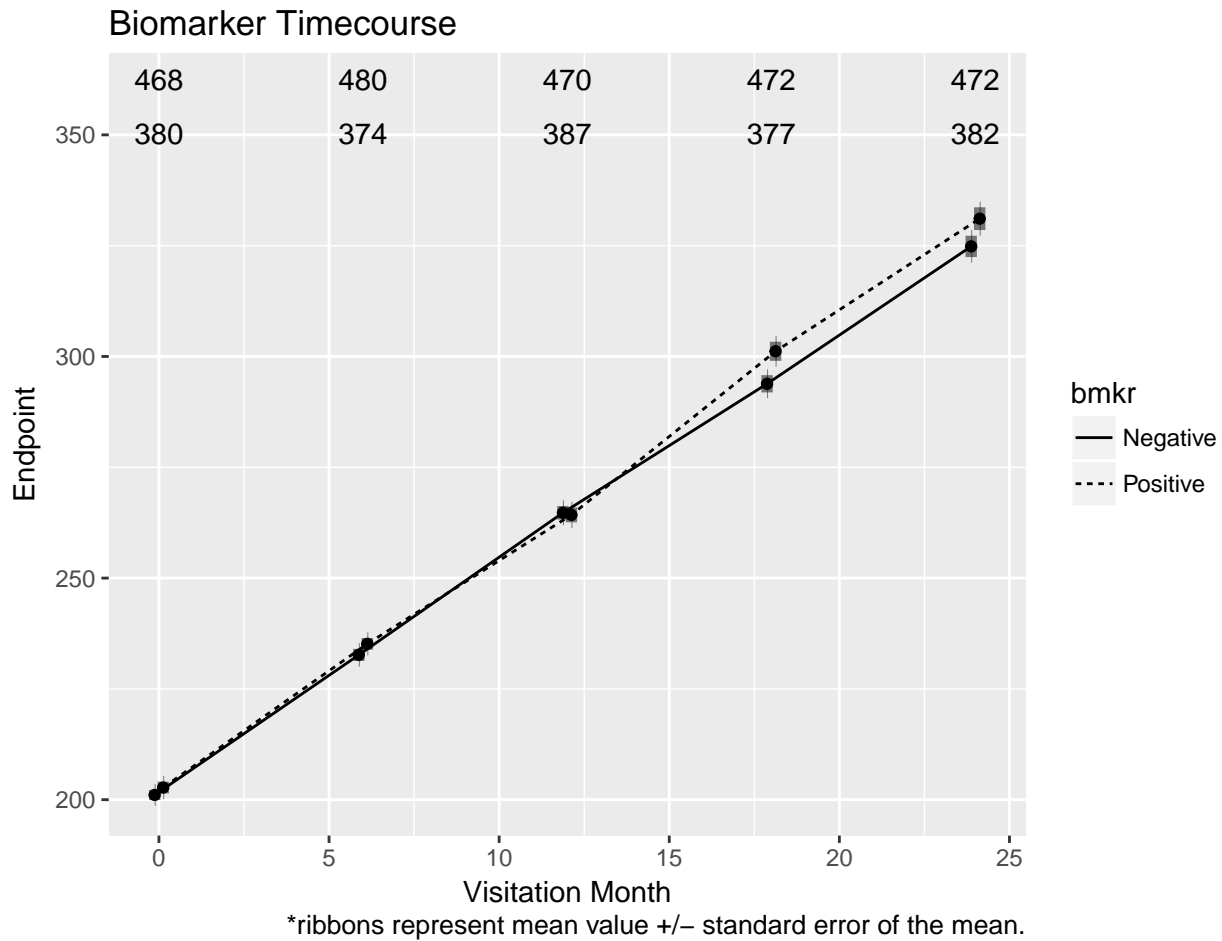        labs.caption = 'Ribbons represent Tukey hinges and whiskers')
```

Biomarker Timecourse

Ribbons represent Tukey hinges and whiskers

Similarly, we could also choose to subset our treatment arms by sex and represent this as two separate plots, one for each sex classification. In this case, we facet the plot by a formula (which is passed to ggplot::facet_grid()).

```
PlotLong(longbmkr, x=vm, y=ep, group=trt, color=trt, fill=trt,
        fun.data = 'tukey', facet.fun = . ~ sex,
        xlab = 'Visitation Month',
        ylab = 'Biomarker Endpoint',
        labs.title = 'Biomarker Timecourse',
        labs.caption = 'Ribbons represent Tukey hinges and whiskers')
```

Biomarker Timecourse

Ribbons represent Tukey hinges and whiskers

4.6.3.3 Including sample counts

The `PlotLong` function also accepts inputs to the argument `show.counts`, accepting either "label" or "table". This is used to represent sample number in each timepoint, for each grouping. For this and the followign plots, we'll do some preprocessing for which we'll use the `dplyr` package.

```r
library(dplyr)
```

```r
cutoff = 1.5

longbmkr %>%
    filter(vm <= 24) %>%
    mutate(bmkr = ifelse(bmkr > cutoff, 'Positive', 'Negative')) %>%
    mutate(trt  = ifelse(trt == 0, 'CTRL', 'TRT')) %>%

    PlotLong(x=vm, y=ep, color=trt, fill=trt, linetype=bmkr,
             fun.data = 'mean_se',
             show.counts = 'table',
             plot.style = 'errorbars',
             errorbar.linetype = 1,
             xlab = 'Visitation Month',
             ylab = 'Endpoint',
             labs.title = 'Biomarker Timecourse',
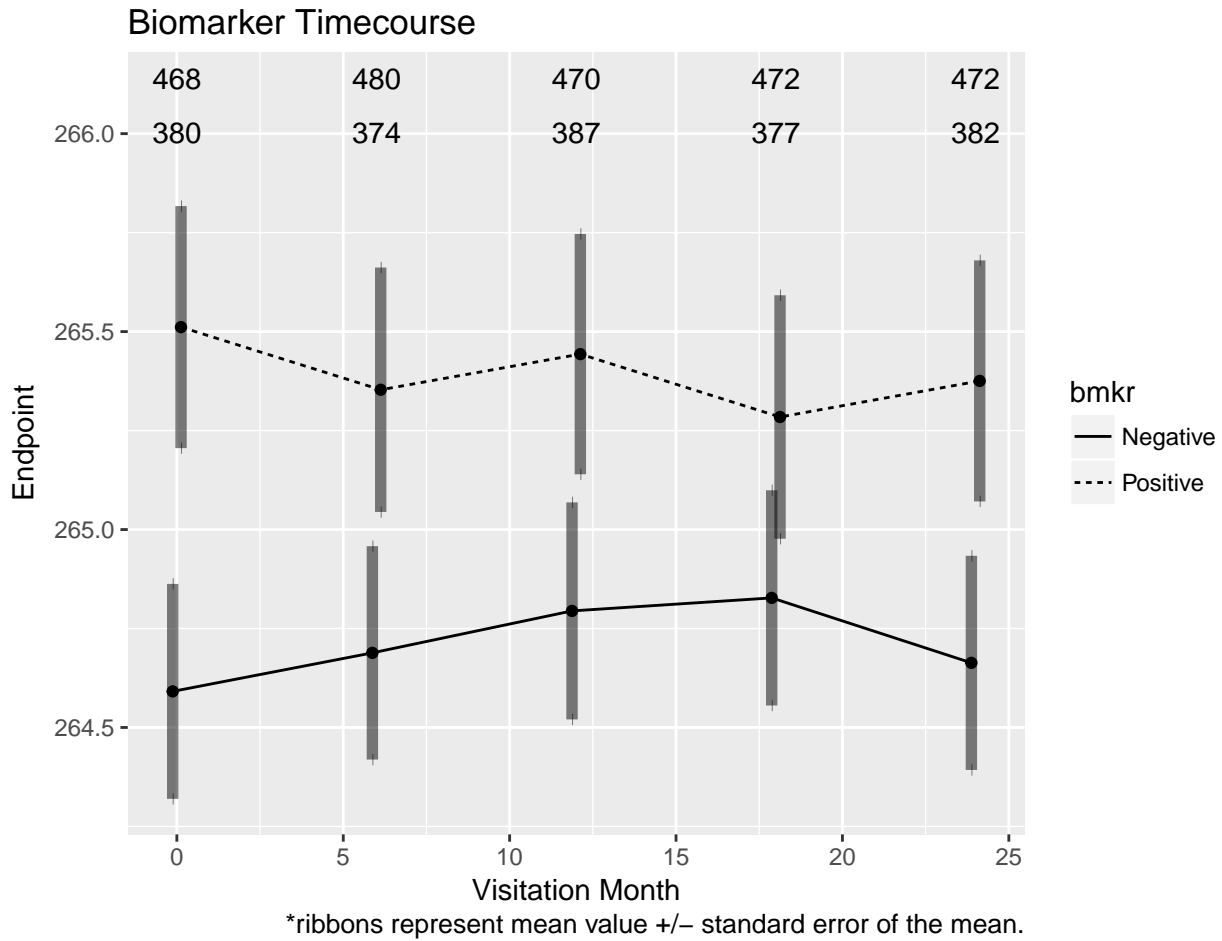             labs.caption = '*ribbons represent mean value +/- standard error of the mean.')
```

Biomarker Timecourse

*ribbons represent mean value +/– standard error of the mean.

4.6.3.4 Specifying a function for adjustment

(Currently a work-in-progress!)

```
cutoff = 1.5

longbmkr %>%
    filter(vm <= 24) %>%
    mutate(bmkr = ifelse(bmkr > cutoff, 'Positive', 'Negative')) %>%
    mutate(trt  = ifelse(trt == 0, 'CTRL', 'TRT')) %>%

    PlotLong(x=vm, y=ep, color=trt, fill=trt, linetype=bmkr,
             model = lm,
             model.formula = ep ~ sex + edu + age,
             fun.data = 'mean_se',
             show.counts = 'table',
             plot.style = 'errorbars',
             errorbar.linetype = 1,
             xlab = 'Visitation Month',
             ylab = 'Endpoint',
             labs.title = 'Biomarker Timecourse',
             labs.caption = '*ribbons represent mean value +/- standard error of the mean.')
```

```
## Warning: Deprecated: please use `purrr::possibly()` instead
```

```
## Warning: Deprecated: please use `purrr::possibly()` instead

## Warning: Deprecated: please use `purrr::possibly()` instead

## Warning: Deprecated: please use `purrr::possibly()` instead

## Warning: Deprecated: please use `purrr::possibly()` instead
```

Biomarker Timecourse

*ribbons represent mean value +/− standard error of the mean.

5 Acknowledgement