dable manual

Henrik Renlund

October 4, 2025

Contents

List of Tables

List of Figures

1 Introduction

A *dable*, i.e. a *descriptive table*, has up to 3 (optional) parts consisting of

- descriptions (`desc`),

and if a stratification (grouping) is given, also

- comparisons (`comp`), as well as
- tests (`test`).

The table-creating functions will respond to a `part` argument which specifies which parts are wanted, with only the descriptive part being returned by default.

Each variable is given a type (Section 2), and each type is associated with its own set of functions to create the parts (Section 3.1.1). One must always choose one of these (or accept the default) to e.g. create the description part, but it is also possible to provide user defined functions (Section 3.5.1).

It is also possible to provide a *variable table* (`vtab`, Section 2.2) that provides a label and a group for each variable. The grouping can be used to group the rows of the output tables (if generated in e.g. LaTeX).

The package initially tried to strictly separate table creation into

(1) generate the statistics/numbers that go into the table

(2) format for a desired output generating program/language

This aim has not quite been achieved. It works pretty well for default versions of *simple descriptive tables* (Section 3), but is harder to do for *baseline tables* (Section 4). The compromise reached is that one can set package parameters pointing to the desired output. As an example, this document is generated in LaTeX so I will begin by restoring all defaults[1] with that output program in mind.

```
dpset_defaults(overwrite = TRUE, style = "latex")
```

[1] Currently, this actually does very little - it's main purpose is to set the `indent` parameter to `\quad`. The most obvious problem in this context is the fact that '%' has a special meaning, but that is easy to substitute just before translating to LaTeX code.

2 Types

To determine how to deal with a variable we classify it as

- numeric (`real`),
- categorical (`catg`),
- date (`date`), or
- time-to-event (`surv`).

Of course, a time-to-event variable will actually consist of a pair of variables - more on that later. There are also subcategories of `catg`:

- *binary* variables (`bnry`) are dichotomous variables where it should be completely obvious what the two levels are if you are only given one. The idea is that for these variables, it typically suffices to describe (e.g. by a count and/or percentage) one of the levels.
- *lavish categorical* variables (`lcat`) are those where there are too many unique values to routinely include value-level information in a table (e.g. an id-variable of some sort).

There is a non-surprising general idea of a mapping from the class (in R), of a variable, to the dable *type*, given by Table 1.

Table 1: The basic mapping idea.

R class	⟶	dable type
`numeric`, `integer`		`real`
`factor`, `character`, `logical`		`catg` (or `bnry`/`lcat`)
`Date`		`date`

But the types can be changed, manually or through parameters, so that e.g. integer variables with a small number of unique values can be describes as if it was categorical. See Figure 1 for a the full set of possible automatic mappings.

The parameters one can set that will alter the general mapping are given by:

- `real.tol` is an integer n and, if set, re-types a `real` variable to a `catg` variable *if* the number of unique values is $\leq n$. Why? `real` types typically describe variables measured on a continuous scale and if the number of unique values is low, it is probably a coding for something else *or* something you want described as a `catg`.
- `catg.tol` is an integer n and, if set, re-types a `catg` variable to a `lcat` variable *if* the number of unique values is $> n$. Why? Some categorical data just have too many levels to include in a descriptive table, the `lcat` type is meant to provide alternative descriptions for these.
- `bnry.list` is a list of length 2 vectors specifying, respectively, what the set of unique values can be in order for a variable to end up with the `bnry` type. Why? The `bnry` variables will generally only show descriptiv measures for the reference level, thus it must be clear from that level alone what the non-showing level is.

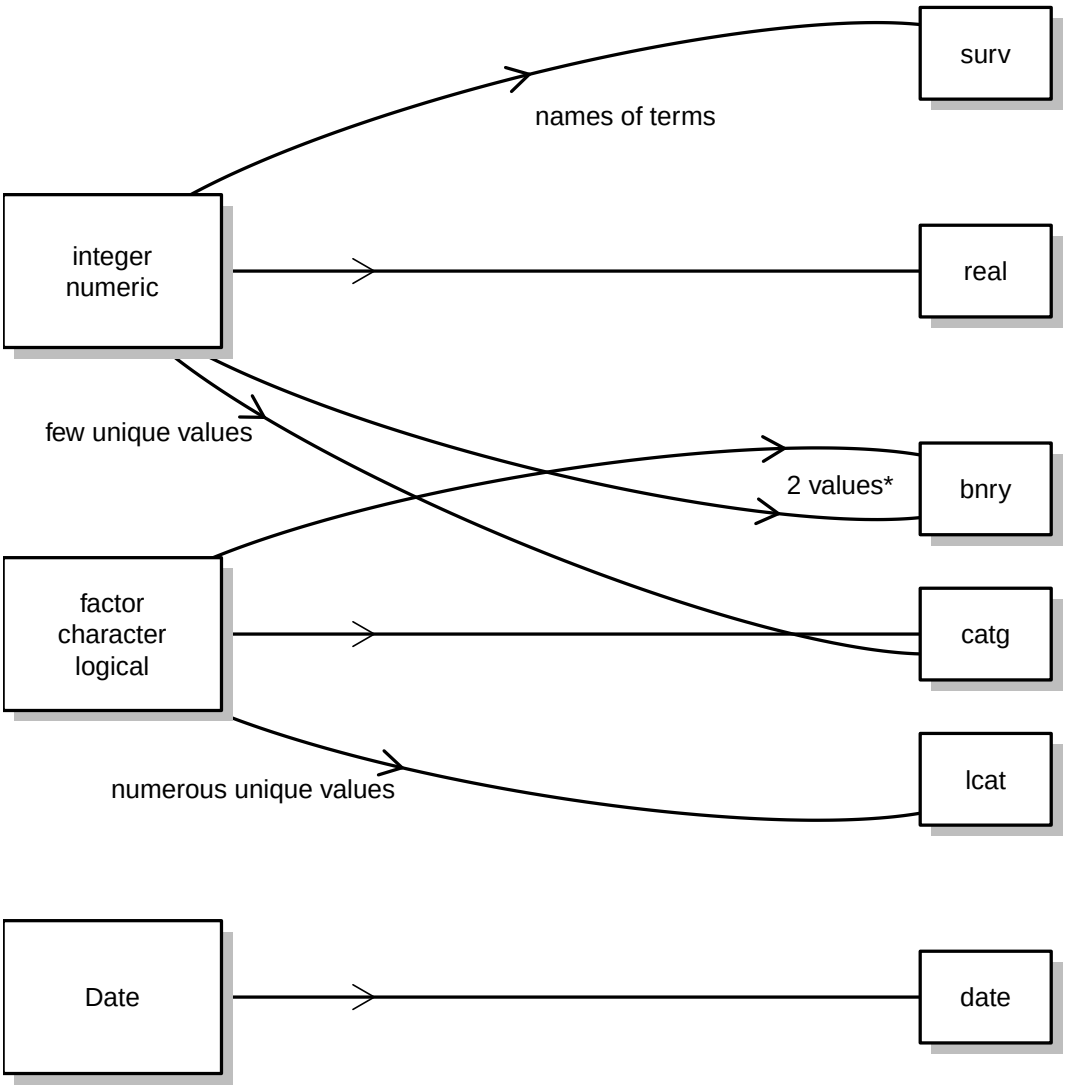

Figure 1: Possible mappings. For both numeric/integer and factor/character/logical, if there are exactly 2 unique values corresponding to one of the elements of `bnry.list`, they get type `bnry`. If number of nunique values of a numeric/integer variable is `real.tol` or less, the type is `catg`. If number of unique values of a factor/character variable is strictly greater than `catg.tol`, the type is `lcat`. The `surv` type is determined through given names or pattern matching.

We can inspect the default values thus

```
dpget("real.tol")

## [1] 0

dpget("catg.tol")

## [1] Inf

dpget("bnry.list")

## [[1]]
## [1] 0 1
##
## [[2]]
## [1] FALSE  TRUE
##
## [[3]]
## [1] "No"  "Yes"
##
## [[4]]
## [1] "no"  "yes"
```

The time-to-event variable pairs need to be specified with a *survival table* (`stab`), an example of which is given in the package

```
test_stab()

##   label  time  event
## 1   Foo t.foo ev.foo
## 2   Bar t.bar ev.bar
```

But - if one utilizes a naming scheme where the time- and status component of a time-to-event variable differes only by a consistent use of a prefix or suffix, these variables can be identified automatically!

The default is to use a prefix `t.` and `ev.` for the time- and event/status component, respectively.

```
dpget("surv.prefix") ## is prefix used? (else suffix)

## [1] TRUE

dpget("surv.affix") ## what is the affix used?

##  time event
##  "t." "ev."
```

2.1 The guide

The packages provides a test data set

```
d <- test_data()
str(d)

## 'data.frame': 1000 obs. of  24 variables:
##  $ id        : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ pid       : chr  "id1" "id2" "id3" "id4" ...
##  $ status    : chr  "included" "included" "included" "included" ...
##  $ country   : Factor w/ 3 levels "SW","NO","DE": 1 3 1 2 1 3 3 2 2 3 ...
##  $ area      : chr  "urban" "rural" "urban" "urban" ...
##  $ offspring : logi  TRUE TRUE FALSE TRUE FALSE FALSE ...
##  $ age       : num  50 54 44 75 37 64 36 42 67 56 ...
##  $ male      : int  1 0 0 0 1 0 0 0 0 1 ...
##  $ measA     : num  10.9 42.4 91.1 16 77.4 ...
##  $ measNA    : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ forgotten : num  -0.899 -0.433 0.753 0.905 -0.803 ...
##  $ importance: num  2.7 2 3.3 4.4 4.3 3.6 5.2 6 5.1 4.6 ...
##  $ index     : Date, format: "2017-08-14" "2017-08-25" ...
##  $ posix     : POSIXct, format: "2017-04-04 23:51:11" "2017-05-15 15:28:34" ...
##  $ ev.foo    : int  1 1 1 0 0 1 1 0 0 0 ...
##  $ ev.bar    : int  0 0 0 1 0 0 0 0 0 0 ...
##  $ t.bar     : num  365 365 365 157 365 ...
##  $ t.foo     : num  130.5 21.5 16.3 365 365 ...
##  $ end       : Date, format: "2018-08-14" "2018-08-25" ...
##  $ measB     : num  58.5 13.3 26.8 NA 29.4 ...
##  $ measM     : num  2.99 NA NA NA 84.5 ...
##  $ gender    : Factor w/ 2 levels "male","female": 1 2 2 2 1 2 2 2 2 1 ...
##  $ kids      : int  1 2 0 3 0 0 2 0 0 0 ...
##  $ region    : chr  "A" "E" "B" "C" ...
```

We can use this data set to illustrate the *data guide* (`dguide`).

```
(g <- dguide(d))
```

```
## data guide:
##              term    type             class      label         group
## 1              id    real           integer         id    Covariates
## 2             pid    catg         character        pid    Covariates
## 3          status    catg         character     status    Covariates
## 4         country    catg            factor    country    Covariates
## 5            area    catg         character       area    Covariates
## 6       offspring    bnry           logical  offspring    Covariates
## 7             age    real           numeric        age    Covariates
## 8            male    bnry           integer       male    Covariates
## 9           measA    real           numeric      measA    Covariates
## 10         measNA missing           numeric     measNA    Covariates
## 11      forgotten    real           numeric  forgotten    Covariates
## 12     importance    real           numeric importance    Covariates
## 13          index    date              Date      index    Covariates
## 14          posix unknown           POSIXct      posix    Covariates
## 17            end    date              Date        end    Covariates
## 18          measB    real           numeric      measB    Covariates
## 19          measM    real           numeric      measM    Covariates
## 20         gender    catg            factor     gender    Covariates
## 21           kids    real           integer       kids    Covariates
## 22         region    catg         character     region    Covariates
## 16 ev.bar / t.bar    surv integer / numeric        bar Time-to-event
## 15 ev.foo / t.foo    surv integer / numeric        foo Time-to-event
## with stab:
##   label  time  event         group
## 1   bar t.bar ev.bar Time-to-event
## 2   foo t.foo ev.foo Time-to-event
```

Note that the guide provides an automatically generated survival table based on the naming of variables. Also, a default grouping of variables is always created with the value

```
dpget("vtab.group.name")
```

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

for all but the time-to-event variables which are grouped as

```
dpget("stab.group.name")
```

```
## [1] "Time-to-event"
```

One can manually update the guide

```
g$type[g$term == "pid"] <- "lcat"
```

or provide some parameters to influence the types

```
subset(dguide(d, catg.tol = 10), subset = term == "pid") ## type now lcat

## data guide:
##   term type     class label      group
## 2  pid lcat character   pid Covariates
```

Arguments to `dguide`:

- `data` the data (duh!)
- `id` a character vector of the terms that are id variables; these will be typed as `lcat`. (Also, the first entry will be singled out as the "unit id" and hidden from tables unless you set the dable option `unit.id.rm` to `FALSE`.)
- `elim.set` a character vector of variables to ignore
- `vtab` a "variable table", i.e. a `data.frame` with columns `term`, `label`, an optionally `group`. This table informs the `label` and `group` column of the guide.
- `stab` a "survival table", i.e. a `data.frame` with columns `label`, `time`, `event`, an optionally `group`.
- `...` arguments passed to `term_type`.

2.2 Variable and survival tables

There are examples of `vtab` and `stab` in the package

```
(vt <- test_vtab())

##         term           label         group
## 1         id    ID (integer)     Meta data
## 2        pid              ID     Meta data
## 3     status      A constant     Meta data
## 4      index           Index     Meta data
## 5      posix   Unkown format     Meta data
## 6        end             End     Meta data
## 7        age             Age  Demographics
## 8       male            Male  Demographics
## 9     gender          Gender  Demographics
## 10   country         Country  Demographics
## 11      area            Area  Demographics
## 12    region          Region  Demographics
## 13 offspring       Offspring Measures etc.
## 14      kids       Kid count Measures etc.
## 15     measA       A measure Measures etc.
## 16     measB Another measure Measures etc.
```

```
## 17     measM   Manly measure Measures etc.
## 18    measNA Missing measure Measures etc.

(st <- test_stab())

##   label  time  event
## 1   Foo t.foo ev.foo
## 2   Bar t.bar ev.bar
```

The survival table has no grouping, so we can provide one

```
st$group <- "Outcomes"
```

Now we can see what the guide looks like, if we add some additional arguments:

```
(g <- dguide(d, id = c("id", "pid"), vtab = vt, stab = st))

## data guide:
##              term    type            class           label         group
## 1              id unit.id          integer    ID (integer)     Meta data
## 2             pid    lcat        character              ID     Meta data
## 3          status    catg        character      A constant     Meta data
## 13          index    date             Date           Index     Meta data
## 14          posix unknown          POSIXct   Unkown format     Meta data
## 17            end    date             Date             End     Meta data
## 7             age    real          numeric             Age  Demographics
## 8            male    bnry          integer            Male  Demographics
## 20         gender    catg           factor          Gender  Demographics
## 4         country    catg           factor         Country  Demographics
## 5            area    catg        character            Area  Demographics
## 22         region    catg        character          Region  Demographics
## 6       offspring    bnry          logical       Offspring Measures etc.
## 21           kids    real          integer       Kid count Measures etc.
## 9           measA    real          numeric       A measure Measures etc.
## 18          measB    real          numeric Another measure Measures etc.
## 19          measM    real          numeric   Manly measure Measures etc.
## 10         measNA missing          numeric Missing measure Measures etc.
## 15 ev.foo / t.foo    surv integer / numeric             Foo      Outcomes
## 16 ev.bar / t.bar    surv integer / numeric             Bar      Outcomes
## 11      forgotten    real          numeric       forgotten    Covariates
## 12     importance    real          numeric      importance    Covariates
## with stab:
##   label  time  event    group
## 1   Foo t.foo ev.foo Outcomes
## 2   Bar t.bar ev.bar Outcomes
```

Note that the variables not specified by the variable table gets the default grouping and are placed at the end.

To see the usefulness of setting up a "variable table", notice that with these preparations the following code generates Table 2:

```
b <- baseline(d, guide = g, gtab = "gender", part = c(T,T,T))
blatex(b, where = "!ht", caption = "Default baseline table",
       label = "tab:default-bsl", size = "small")
```

Table 2: Default baseline table

	male	**female**	**Comparison**	**Test**
	n = 489	n = 511	male/female	male/female
Meta data				
ID	=489=	=511=		
A constant: included	489 (100.0%)	511 (100.0%)		
Index	170101/181229 [1]	170102/181225	-0.035	0.58$^{\forall}$
End	180101/191229 [1]	180102/191225	-0.035	0.58$^{\forall}$
Demographics				
Age	55.0 (48.0 - 55.0)	56.0 (49.0 - 56.0)	-0.069	0.28$^{\flat}$
Male: 1	489 (100.0%)	0 (0%)	Inf	<0.0001$^{\perp}$
Country: SW	173 (35.4%)	178 (34.8%)	0.031	0.88$^{\perp}$
NO	168 (34.4%)	171 (33.5%)		
DE	148 (30.3%)	162 (31.7%)		
Area: rural	101 (20.7%) [2]	103 (20.3%) [3]	0.011	0.92$^{\perp}$
urban	386 (79.3%)	405 (79.7%)		
Region: A	147 (30.1%)	157 (30.7%)	0.044	0.97$^{\perp}$
B	105 (21.5%)	108 (21.1%)		
C	65 (13.3%)	61 (11.9%)		
D	86 (17.6%)	93 (18.2%)		
E	86 (17.6%)	92 (18.0%)		
Measures etc.				
Offspring: TRUE	358 (73.2%)	382 (74.8%)	0.035	0.63$^{\perp}$
Kid count	1.00 (0 - 1.00)	1.00 (0 - 1.00)	-0.019	0.76$^{\flat}$
A measure	26.9 (11.5 - 26.9) [5]	30.3 (12.6 - 30.3) [2]	-0.14	0.029$^{\flat}$
Another measure	38.7 (18.4 - 38.7) [168]	46.7 (20.3 - 46.7) [171]	-0.13	0.088$^{\flat}$
Manly measure	33.6 (14.3 - 33.6)	- [511]		
Outcomes				
Foo	92; 0.00057	109; 0.00066	-0.0036	0.31$^{\sharp}$
Bar	140; 0.00091	160; 0.0010	-0.0031	0.38$^{\sharp}$
Covariates				
forgotten	0.025 (-0.47 - 0.025)	-0.023 (-0.50 - -0.023)	0.022	0.72$^{\flat}$
importance	3.10 (1.60 - 3.10)	3.30 (1.70 - 3.30)	-0.091	0.15$^{\flat}$

Rows: 1000 (male:489, female:511). No duplicate subjects. $^{\perp}$*Chi-square.* $^{\forall}$*Wilcoxon.* $^{\flat}$*t-test.* $^{\sharp}$*Log rank. Categorical variable: =unique values=. [n] is missing count (if applicable). Categorical variable: Count (Percent). Date variables: min/max. Numeric variable: Median (Q1-Q3). Time-to-event variable: events; rate*

3 Simple descriptive tables

A *simple descriptive table* describes only 1 type. Tables describing *all* variables will be refered to as *baseline tables* - so we will cut the "simple" and just refer to the tables in this section as *descriptive*. The workhorse for all tables is the `dable` function. To create the default descriptive table for the real type we use

```
dable(d, type = "real", guide = g)

##         term         Mean        SD
## 1        age 55.332000000 10.072399
## 2       kids  1.472000000  1.206097
## 3      measA 41.058453071 42.242859
## 4      measB 61.475041370 58.837329
## 5      measM 47.721624171 44.726907
## 6  forgotten  0.001249743  0.577495
## 7 importance  3.168900000  1.739416
```

where the output is ordered as the guide since we supply the guide as an argument.

There is a shorthand for each type, and some .

```
dreal(d, guide = g)

##         term         Mean        SD
## 1        age 55.332000000 10.072399
## 2       kids  1.472000000  1.206097
## 3      measA 41.058453071 42.242859
## 4      measB 61.475041370 58.837329
## 5      measM 47.721624171 44.726907
## 6  forgotten  0.001249743  0.577495
## 7 importance  3.168900000  1.739416

dcatg(d, guide = g)

##       term    Level Count Proportion
## 1   status included  1000  1.0000000
## 2   gender     male   489  0.4890000
## 3   gender   female   511  0.5110000
## 4  country       SW   351  0.3510000
## 5  country       NO   339  0.3390000
## 6  country       DE   310  0.3100000
## 7     area    rural   204  0.2050251
## 8     area    urban   791  0.7949749
## 9   region        A   304  0.3040000
## 10  region        B   213  0.2130000
## 11  region        C   126  0.1260000
## 12  region        D   179  0.1790000
## 13  region        E   178  0.1780000

dbnry(d, guide = g)
```

```
##        term Level Count Proportion
## 1      male     1   489      0.489
## 2 offspring  TRUE   740      0.740

dlcat(d, guide = g)

##   term Unique
## 1  pid   1000

ddate(d, guide = g)

##    term    min    max
## 1 index 170101 181229
## 2   end 180101 191229

dsurv(d, guide = g)

##   term     Time Events         Rate
## 1  Foo 325923.1    201 0.0006167099
## 2  Bar 311600.9    300 0.0009627700
```

Some of these accept type specific extra arguments, which can be of interest

```
ddate(d, guide = g, date.format = "%Y-%m-%d")

##    term        min        max
## 1 index 2017-01-01 2018-12-29
## 2   end 2018-01-01 2019-12-29

dsurv(d, guide = g, time.unit = 365.25)

##   term     Time Events      Rate
## 1  Foo 892.3288    201 0.2252533
## 2  Bar 853.1167    300 0.3516518
```

3.1 Table stratification

Tables can be stratified using a *grouping table* (`gtab`). We are slightly overloading the "group" terminology here; the groups use to stratify a table have nothing to do with `group` column of the guide - those can be used to group rows of the output table created in e.g. LaTeX, whereas the `gtab` group columns of the descriptive table.

The simplest way to create a stratification is to point to a grouping variable

```
(dt <- dreal(d, guide = g, gtab = "gender"))

##         term         Mean         SD         Mean         SD
## 1        age 54.977505112 10.3251198 55.671232877  9.8226197
## 2       kids  1.460122699  1.1886350  1.483365949  1.2236306
## 3      measA 38.059724024 38.8539428 43.909896801 45.0839527
## 4      measB 57.459770884 55.8057254 65.265929095 61.4038813
## 5      measM 47.721624171 44.7269066          NaN         NA
```

```
## 6  forgotten  0.007871068  0.5604131 -0.005086515  0.5938614
## 7 importance  3.088343558  1.7199192  3.245988258  1.7560926
```

Note that column names become duplicated, but there is "meta data" in the attributes that keeps track of which columns belong to which grouping.

```
attr(dt, "part")

## [1] "meta"        "desc:male"   "desc:male"   "desc:female" "desc:female"
```

In, fact there is even more meta data behind the scene; if we pass this object to `datex`, it generates the LaTeX code for a table, using some of this additional information as can be seen in Table 3.

```
datex(dt, label = "tab:datex",
      caption = "Default \\LaTeX\\, output for descriptive table.")
```

Table 3: Default LaTeX output for descriptive table.

	male		female	
	Mean	SD	Mean	SD
Demographics				
Age	55.0	10.3	55.7	9.82
Measures etc.				
Kid count	1.46	1.19	1.48	1.22
A measure	38.1	38.9	43.9	45.1
Another measure	57.5	55.8	65.3	61.4
Manly measure	47.7	44.7		
Covariates				
forgotten	0.0079	0.56	-0.0051	0.59
importance	3.09	1.72	3.25	1.76

Rows: 1000 (male:489, female:511). No duplicate subjects

The package assumes utilization of meta data is wanted. If not, one typically have to turn these features off, see Table 4.

```
datex(dt, row.group = FALSE, lab = FALSE, label = "tab:datex-less",
      caption = "Descriptive table using less meta data.")
```

Table 4: Descriptive table using less meta data.

	male		female	
	Mean	SD	Mean	SD
Age	55.0	10.3	55.7	9.82
Kid count	1.46	1.19	1.48	1.22
A measure	38.1	38.9	43.9	45.1
Another measure	57.5	55.8	65.3	61.4
Manly measure	47.7	44.7		
forgotten	0.0079	0.56	-0.0051	0.59
importance	3.09	1.72	3.25	1.76

Rows: 1000 (male:489, female:511). No duplicate subjects

But we are digressing. Another way to create a grouping is to use `gtab_maker`, or create one manually - it is simply a data frame with logical columns of length equal to the data set.

```
gt <- gtab_maker("area", data = d, all = TRUE)
head(gt)

##   rural urban  All
## 1 FALSE  TRUE TRUE
## 2  TRUE FALSE TRUE
## 3 FALSE  TRUE TRUE
## 4 FALSE  TRUE TRUE
## 5 FALSE  TRUE TRUE
## 6 FALSE  TRUE TRUE
```

Lets use a less frequent type (`lcat`) to illustrate the next point; although the presentation defaults to the order of the `gtab`,

```
(dt <- dlcat(d, guide = g, gtab = gt))

##   term Unique Unique Unique
## 1  pid    204    791   1000

attr(dt, "part") ## ordered as gtab

## [1] "meta"       "desc:rural" "desc:urban" "desc:All"
```

it is possible to change it within the `part` argument.

```
dt <- dlcat(d, guide = g, gtab = gt, part = list("desc" = c(3,1,2)))
attr(dt, "part") ## order now permutated

## [1] "meta"       "desc:All"   "desc:rural" "desc:urban"
```

It is also possible to specify `desc` as one of "all" (default), "first" or "last" ("none" is also accepted).

```
dt <- dlcat(d, guide = g, gtab = gt, part = "last")
attr(dt, "part") ## now only last group included

## [1] "meta"     "desc:All"
```

3.1.1 Other descriptive functions

If you are not happy with the default description, there are other describer functions to choose from, use `?desc-real` to see all build-in describers for the `real` type, and similar help pages exists for other types and the other parts (`comp`/`test`). In section 3.5 you can see how to define your own functions. The name of the new function must be supplied to the `fnc` argument. As an example, `min_max` is a built-in function that one can choose

```
dreal(d, fnc = list(desc = "min_max"))

##         term          Min          Max
## 1         id  1.000000000 1000.0000000
## 2        age 25.000000000   85.0000000
## 3      measA  0.018912588  367.3780576
## 4  forgotten -0.999758688    0.9965274
## 5 importance  0.100000000    6.0000000
## 6      measB  0.004979325  362.1742406
## 7      measM  0.195807952  306.1464719
## 8       kids  0.000000000    5.0000000
```

3.1.2 Change default functions

The default values can be viewed with `dpget`, e.g. the default describer, compararer and tester for `real`.

```
dpget("real.desc")

## [1] "mean_sd"

dpget("real.comp")

## [1] "real.std"

dpget("real.test")

## [1] "param"
```

New values can be set with `dpset`.

```
dpset("real.desc", "name_of_new_function") ## no eval
```

You can view *all* package parameters with `dpget_all()`.

3.2 An emphasis on weight

Most built-in describers and comparers has a `weight` argument (when it makes sense) - at the the of writing this is not implemented for the testers. This parameter is explicit in the higher order `dable` function so always exists (even if null), as can be seen in Secion 3.6.

Currently, weights can only be specified as a variable in data.

```
dreal(d, guide = g, weight = "importance")
```

3.3 Table comparisons

The comparison part of a decriptive table is accessed by the `part` argument. This can be given either a logical vector of length up to 3 - if shorter, it fills in the missing values with `FALSE`. The position of the elements is associated with the parts: first *descriptions*, second *comparisons* and third *tests*.

```
dt <- dreal(d, guide = g, gtab = "gender", part = c(TRUE, TRUE))
datex(dt, label = "tab:desc-n-comp", row.group = FALSE,
      caption = "Descriptive table with comparison.")
```

Table 5: Descriptive table with comparison.

	male		**female**		**Std. diff.**
	Mean	SD	Mean	SD	male/female
Age	55.0	10.3	55.7	9.82	-0.069
Kid count	1.46	1.19	1.48	1.22	-0.019
A measure	38.1	38.9	43.9	45.1	-0.14
Another measure	57.5	55.8	65.3	61.4	-0.13
Manly measure	47.7	44.7			
forgotten	0.0079	0.56	-0.0051	0.59	0.022
importance	3.09	1.72	3.25	1.76	-0.091

Rows: 1000 (male:489, female:511). No duplicate subjects

But, as we've seen in Section 3.1, one can manipulate the order of the descriptive presentation, and one can change the "order" of the comparison as well. To do so, one provides a list to `part` instead. The first list entry corresponds to descriptions and is `TRUE/FALSE` or a vector of the wanted ordering. The second list entry corresponds to comparison and can be

- `TRUE/FALSE`
- A list of length 2 vectors for the comparison pairs
- A single character value "across" (Table 7) or "adjacent" (Table 8), "none" is also accepted.

So, to change to order of the comparison in Table 5 you use

```
dt <- dreal(d, guide = g, gtab = "gender",
            part = list(TRUE, list(c(2,1))))
datex(dt, label = "tab:desc-n-comp-2", row.group = FALSE,
      caption = "Descriptive table with comparison order changed.")
```

Table 6: Descriptive table with comparison order changed.

	male		**female**		**Std. diff.**
	Mean	SD	Mean	SD	female/male
Age	55.0	10.3	55.7	9.82	0.069
Kid count	1.46	1.19	1.48	1.22	0.019
A measure	38.1	38.9	43.9	45.1	0.14
Another measure	57.5	55.8	65.3	61.4	0.13
Manly measure	47.7	44.7			
forgotten	0.0079	0.56	-0.0051	0.59	-0.022
importance	3.09	1.72	3.25	1.76	0.091

Rows: 1000 (male:489, female:511). No duplicate subjects

to get Table 6.

To make comparisons "across":

```
dt <- dbnry(d, guide = g, gtab = "region",
            part = list(FALSE, "across"))
datex(dt, label = "tab:across", row.group = FALSE,
      caption = "Comparison \\emph{across} the groups.")
```

Table 7: Comparison *across* the groups.

	Std. diff.			
	A/B	A/C	A/D	A/E
Male	0.019	0.065	0.0062	0.00081
Offspring	0.15	0.13	0.099	0.14

Rows: 1000. No duplicate subjects

To make comparisons "adjacent":

```
dt <- dbnry(d, guide = g, gtab = "region",
            part = list(FALSE, "adjacent"))
datex(dt, label = "tab:adjacent", row.group = FALSE,
      caption = "Comparison between \\emph{adjacent} groups.")
```

Table 8: Comparison between *adjacent* groups.

	Std. diff.			
	A/B	B/C	C/D	D/E
Male	0.019	0.046	0.071	0.0054
Offspring	0.15	0.019	0.034	0.041

Rows: 1000. No duplicate subjects

To make your own comparison:

```
dt <- dbnry(d, guide = g, gtab = "region",
            part = list(FALSE, list(c(1,3), c(5,2))))
datex(dt, label = "tab:comp-list", row.group = FALSE,
      caption = "Comparison between chosen groups.")
```

Table 9: Comparison between chosen groups.

	Std. diff.	
	A/C	E/B
Male	0.065	0.020
Offspring	0.13	0.012

Rows: 1000. No duplicate subjects

3.4 Table tests

The testing part of the `part` argument cam be given as `TRUE/FALSE`, with `TRUE` meaning to test all groups

```
dt <- dreal(d, guide = g, gtab = "country", part = c(TRUE,FALSE,TRUE))
datex(dt, label = "tab:desc-test", row.group = FALSE,
      caption = "Descriptive table with test.")
```

Table 10: Descriptive table with test.

	SW		NO		DE		F-test
	Mean	SD	Mean	SD	Mean	SD	SW/NO/DE
Age	55.5	9.78	55.4	10.0	55.1	10.5	0.88
Kid count	1.46	1.16	1.56	1.28	1.38	1.18	0.17
A measure	38.2	39.8	41.9	45.6	43.4	40.9	0.26
Another measure	63.3	62.8			59.4	54.1	0.4
Manly measure	50.8	49.4	46.1	43.2	45.9	40.6	0.53
forgotten	-0.038	0.59	0.022	0.59	0.023	0.55	0.29
importance	3.30	1.67	3.10	1.76	3.09	1.79	0.2

Rows: 1000 (SW:351, NO:339, DE:310). No duplicate subjects

One can also specify wich groups to include in the test

```
dt <- dreal(d, guide = g, gtab = "country", part = list(TRUE,FALSE,c(2,3)))
datex(dt, label = "tab:desc-test-2", row.group = FALSE,
      caption = "Descriptive table with selected testing.")
```

Table 11: Descriptive table with selected testing.

	SW		NO		DE		t-test
	Mean	SD	Mean	SD	Mean	SD	NO/DE
Age	55.5	9.78	55.4	10.0	55.1	10.5	0.71
Kid count	1.46	1.16	1.56	1.28	1.38	1.18	0.068
A measure	38.2	39.8	41.9	45.6	43.4	40.9	0.66
Another measure	63.3	62.8			59.4	54.1	
Manly measure	50.8	49.4	46.1	43.2	45.9	40.6	0.97
forgotten	-0.038	0.59	0.022	0.59	0.023	0.55	0.97
importance	3.30	1.67	3.10	1.76	3.09	1.79	0.95

Rows: 1000 (SW:351, NO:339, DE:310). No duplicate subjects

3.5 Custom made functions

In general, all custom made functions must take a `...` argument - since specific function arguments will be passed on through higher order functions, these additional arguments will be supplied to all describer, comparers, and testers. Therefore, they must tolerate being overfed arguments.

3.5.1 Custom made describers

The recommended way is to define a function that returns a `data.frame`, even if a single value is wanted. This function can be pointed to in the `fnc` argument.

Describers need only take an `x` argument (as well as `...`)

```
IQR <- function(x, ...){
    x <- quantile(x, probs = c(.25, .75), na.rm = TRUE)
    data.frame(IQR = x[2] - x[1])
}
dreal(d, fnc = list(desc = "IQR")) |> head(n = 3)

##    term       IQR
## 1    id 499.50000
## 2   age  14.25000
## 3 measA  43.98454
```

You can point to non-`data.frame`-returning functions as well (as long as they accept "`...`"), but they will not get great column names

```
dreal(d, fnc = list(desc = "mean"), na.rm = TRUE) |> head(n = 3)

##    term         r
## 1    id 500.50000
## 2   age  55.33200
## 3 measA  41.05845
```

unless you give them a label

```
myMean <- function(x, ...) mean(x, na.rm = TRUE, ...)
attr(myMean, "label") <- "The Mean Value"
dreal(d, fnc = list(desc = "myMean")) |> head(n = 3)

##    term The Mean Value
## 1    id      500.50000
## 2   age       55.33200
## 3 measA       41.05845
```

3.5.2 The 'meta' attribute

Some things a describer returns might be of a meta data character, e.g. the levels of a categorial variable - these are wasteful to repeat in each strata. To that end, there is a 'meta' attribute that can be set (on the function) to tell the program which columns not to repeat across a stratification.

```
myBnry <- function(x, ...){
    if(!is.factor(x)) x <- factor(x)
    data.frame(Level = levels(x),
               Count = as.integer(table(x)))
}
attr(myBnry, "meta") <- "Level"
dbnry(d, gtab = "gender", fnc = list(desc = "myBnry"))

##        term Level Count Count
## 1 offspring FALSE   131   129
## 2 offspring  TRUE   358   382
## 3      male     0     0   511
## 4      male     1   489     0
```

3.5.3 Custom made describers for `surv`

Decribers for the `surv` type must take a `time` and `event` argument. Note that all functions *can* take extra arguments, and that these can be passed from the higher order functions.

```
Foo <- function(time, event, my.unit.of.time = 'time units', ... ){
    data.frame(Info = sprintf(paste0("%s events in %.2f ", my.unit.of.time),
                              sum(event), sum(time)))
}
dsurv(d, fnc = list(desc = "Foo"))

##   term                              Info
## 1  bar 300 events in 311600.89 time units
## 2  foo 201 events in 325923.10 time units

dsurv(d, fnc = list(desc = "Foo"), my.unit.of.time = "days")

##   term                          Info
## 1  bar 300 events in 311600.89 days
## 2  foo 201 events in 325923.10 days
```

3.5.4 Custom made comparers/testers

Comparers, and testers, must take an `x` and `g` argument. Conceptually, comparers always deal with 2 groups, whereas a tester can deal with any number of groups.

```
Delta <- function(x, g, ...){
    if(!is.factor(g)) g <- factor(g)
    x_i <- g == levels(g)[1]
    y_i <- g == levels(g)[2]
    data.frame(MinDelta = min(x[x_i], na.rm = TRUE) - min(x[y_i], na.rm = TRUE),
               MaxDelta = max(x[x_i], na.rm = TRUE) - max(x[y_i], na.rm = TRUE))
}
dreal(d, gtab = "gender", part = c(F,T), fnc = list(desc=NULL, comp = "Delta"))
```

```
##         term     MinDelta      MaxDelta
## 1         id -1.000000000  -1.000000000
## 2        age  3.000000000   0.000000000
## 3      measA -0.043488640 -97.310299769
## 4  forgotten  0.003472922  -0.006842324
## 5 importance  0.000000000   0.000000000
## 6      measB  0.054081002  -7.272849083
## 7      measM         -Inf           Inf
## 8       kids  0.000000000   0.000000000
```

This function could also be supplied as a tester (although it is not testing anything); here is an example where we keep the default comparer:

```
(dt <- dreal(d, gtab = "gender", part = c(F,T,T),
             fnc = list(desc=NULL, comp = NULL, test = "Delta")))
```

```
##         term  Std. diff.     MinDelta      MaxDelta
## 1         id -0.03630706 -1.000000000  -1.000000000
## 2        age -0.06884267  3.000000000   0.000000000
## 3      measA -0.13901052 -0.043488640 -97.310299769
## 4  forgotten  0.02244206  0.003472922  -0.006842324
## 5 importance -0.09069944  0.000000000   0.000000000
## 6      measB -0.13304830  0.054081002  -7.272849083
## 7      measM         NaN         -Inf           Inf
## 8       kids -0.01926886  0.000000000   0.000000000
```

Recall that what is set as tester or comparer is remembered by the `part` attribute.

```
attr(dt, "part")
```

```
## [1] "meta"             "comp:male/female" "test:male/female" "test:male/female"
```

3.5.5 Custom made comparers/testers for `surv`

Comparers, and testers, for `surv` must take a `time`, `event`, and `g` argument.

```
HR <- function(time, event, g, weight = NULL, ...){
    if(!is.factor(g)) g <- factor(g)
    mod <- survival::coxph(survival::Surv(time, event) ~ g, weight = weight)
    data.frame(HR = as.numeric(exp(mod$coefficients)[1]))
}
dsurv(d, gtab = "gender", part = c(F,T), fnc = list(desc=NULL, comp = "HR"))

##   term       HR
## 1  bar 1.107036
## 2  foo 1.155531
```

3.6 The information functions are privy to

Internally, what is passed to each function is not only what the user supplies in the "`...`" of the higher order functions, but also some additional stuff; e.g. the name of the current variable (`.term`), the label (`.label`), if the variable has any missing values (`.missing`), etc.

```
theDots <- function(x, ...){
    dots <- list(...)
    foo <- function(z) if(is.null(z)) "NULL" else z
    data.frame(name = names(dots),
               value = unlist(lapply(dots, foo)))
}
dlcat(d, guide = g, fnc = list(desc = "theDots"),
      pass_this_along = "this was passed by the user")

##   term            name                        value
## 1  pid          weight                         NULL
## 2  pid pass_this_along this was passed by the user
## 3  pid     .table.type                         lcat
## 4  pid           .term                          pid
## 5  pid          .label                           ID
## 6  pid           .type                         lcat
## 7  pid          .group                    Meta data
## 8  pid        .missing                        FALSE
```

This allows for silly functions

```
silly <- function(x, ...){
    dots <- list(...)
    r <- paste0("'I'm %s (a %s type) but I prefer my to use my label: %s'")
    data.frame(Presentation = sprintf(r, dots$.term, dots$.type, dots$.label))
}
dlcat(d, guide = g, fnc = list(desc = "silly"))

##   term                                                Presentation
## 1  pid 'I'm pid (a lcat type) but I prefer my to use my label: ID'
```

3.7 Some notes on formatting

To appear.

4 Baseline tables

Baseline tables describe several types in one table - the name is chosen from the clinical world where a study population's various traits are typically described by the characteristics collected at the start of the study, at baseline.

Combining various types adds a layer of complexity for the functions that generate the parts of the table. For one, they have to to return the same number of columns (per part). This typically means mixing numbers and text, and so formatting of numbers has to be done already at the table-generating phase - in contrast, the default simple descriptive tables could be populated with all numbers without the need for formatting.

Another problem is how to know what the presented numbers are. In a simple descriptive table, we name the columns to indicate what is in them; e.g. "Mean" and "SD" when we present the mean values and standard deviations for the variables. But in a baseline table, we mix the types - so the columns typically are not enough to explain this.

A third problem in the context of this package, is that it becomes much more involved to pass user defined functions to baseline tables (so much so that it is currently not possible) - we need a at least a function per type.

The current approach is that there is a set of default functions for baseline tables that is distinct from the set of default funcions for simple descriptive tables. And thus, to choose other functions for baseline tables means to set them as package parameters. To "manually" implement a completely new set of baseline functions is somewhat cumbersome. However, the aim is to put together a number of baseline *themes* that the user easily can choose between without having to reset any default values - and if a new theme is needed, it would be best to implement it *within* the package.

Before going further into the details, lets look at the two (actually three, but really just two) themes that exist currently.

4.1 Baseline theme 0 (default)

The distinct feature of the default theme is that it tries to be compact.

The default baseline table was shown in Table 2, but we'll do a version of it again. We reuse the "variable table" stored in `vt` and we will also trim down the data so that the table becomes a bit smaller, see Table 12.

```
sel <- subset(g, !group %in% c("Measures etc.", "Outcomes", "Covariates"))$term
d2 <- d[, sel]
g2 <- dguide(d2, id = c("id", "pid"), vtab = vt,
             elim.set = c("end", "region", "area"))
b <- baseline(d2, theme = 0, guide = g2, gtab = "gender", part = c(T,T,T))
blatex(b, caption = "Baseline table theme 0 (default)",
       label = "tab:bsl-0", size = "small")
```

Table 12: Baseline table theme 0 (default)

	male	**female**	**Comparison**	**Test**
	n = 489	n = 511	male/female	male/female
Meta data				
ID	=489=	=511=		
A constant: included	489 (100.0%)	511 (100.0%)		
Index	170101/181229 [1]	170102/181225	-0.035	0.58$^{\forall}$
Demographics				
Age	55.0 (48.0 - 55.0)	56.0 (49.0 - 56.0)	-0.069	0.28$^{\flat}$
Male: 1	489 (100.0%)	0 (0%)	Inf	<0.0001$^{\perp}$
Country: SW	173 (35.4%)	178 (34.8%)	0.031	0.88$^{\perp}$
NO	168 (34.4%)	171 (33.5%)		
DE	148 (30.3%)	162 (31.7%)		

Rows: 1000 (male:489, female:511). No duplicate subjects. $^{\perp}$*Chi-square.* $^{\forall}$*Wilcoxon.* $^{\flat}$*t-test. Categorical variable: =unique values=. [n] is missing count (if applicable). Categorical variable: Count (Percent). Date variables: min/max. Numeric variable: Median (Q1-Q3)*

Notice that each type has a unique form of the numbers presented in the descriptive part of the table. This allows for the explanation of those numbers to be in the "foot" of the table. The comparison part only contain 1 kind of numbers (the standardized differences - but this is actually not clear). The test part contains different kinds of test, all of which are explained through footnotes.

ALso, notice that baseline tables have their own LaTeX-code-generating function `blatex`.

4.1.1 Baseline theme 1, a slight variation on 0

The only difference between theme 0 and 1 is that the `real` type is described as "Median (Q1-Q3)" in 0 and "Mean (SD)" in 1, see Table 13.

```
b <- baseline(d2, theme = 1, guide = g2, gtab = "gender", part = c(T,T,T))
blatex(b, caption = "Baseline table theme 1.",
       label = "tab:bsl-1", size = "small")
```

Table 13: Baseline table theme 1.

	male	**female**	**Comparison**	**Test**
	n = 489	n = 511	male/female	male/female
Meta data				
ID	=489=	=511=		
A constant: included	489 (100.0%)	511 (100.0%)		
Index	170101/181229 [1]	170102/181225	-0.035	0.58$^{\forall}$
Demographics				
Age	55.0 (10.3)	55.7 (9.82)	-0.069	0.28$^{\flat}$
Male: 1	489 (100.0%)	0 (0%)	Inf	<0.0001$^{\perp}$
Country: SW	173 (35.4%)	178 (34.8%)	0.031	0.88$^{\perp}$
NO	168 (34.4%)	171 (33.5%)		
DE	148 (30.3%)	162 (31.7%)		

Rows: 1000 (male:489, female:511). No duplicate subjects. $^{\perp}$*Chi-square.* $^{\forall}$*Wilcoxon.* $^{\flat}$*t-test. Categorical variable: =unique values=. [n] is missing count (if applicable). Categorical variable: Count (Percent). Date variables: min/max. Numeric variable: Mean (SD)*

4.2 Baseline theme 2

The baseline theme 2 takes more space, but is perhaps simpler to parse.

```
b <- baseline(d2, theme = 2, guide = g2, gtab = "gender", part = c(T,T,T))
blatex(b, caption = "Baseline table theme 2.",
       label = "tab:bsl-2", size = "small")
```

Table 14: Baseline table theme 2.

	male	**female**	**Comparison**	**Test**
	n = 489	n = 511	male/female	male/female
Meta data				
ID; Unique	489	511		
A constant, n (%): included	489 (100.0%)	511 (100.0%)		
Index; Min	2017-01-01	2017-01-02	-0.035	0.58$^{\forall}$
Max	2018-12-29	2018-12-25		
Missing	1	0		
Demographics				
Age; Mean (SD)	55.0 (10.3)	55.7 (9.82)	-0.069	0.28$^{\flat}$
Median (Q1,Q3)	55.0 (48.0, 63.0)	56.0 (49.0, 62.0)		
Min, Max	28.0, 85.0	25.0, 85.0		
Male, n (%): 0	0 (0%)	511 (100.0%)	Inf	<0.0001$^{\perp}$
1	489 (100.0%)	0 (0%)		
Country, n (%): SW	173 (35.4%)	178 (34.8%)	0.031	0.88$^{\perp}$
NO	168 (34.4%)	171 (33.5%)		
DE	148 (30.3%)	162 (31.7%)		

Rows: 1000 (male:489, female:511). No duplicate subjects. $^{\perp}$*Chi-square.* $^{\forall}$*Wilcoxon.* $^{\flat}$*t-test*

4.3 Adjusting the baseline table

This section is just a reminder of some of the features that were presented for simple descriptive tables. E.g. suppose you want a default baseline table by `gender` but also a column showing the data for everyone, as well as a test - but, of course, the test should only contrast the gender levels. Then you can use `gtab_maker` and supply the correct input for the `part` argument.

```
gt <- gtab_maker("gender", data = d2, all = TRUE, all.first = TRUE)
b <- baseline(d2, theme = 0, guide = g2, gtab = gt,
                  part = list(T, F, c(2,3)))
blatex(b, label = "tab:adj-bsl", caption = "Adjusted baseline table")
```

Table 15: Adjusted baseline table

	All	**male**	**female**	**Test**
	n = 1000	n = 489	n = 511	male/female
Meta data				
ID	=1000=	=489=	=511=	
A constant: included	1000 (100.0%)	489 (100.0%)	511 (100.0%)	
Index	170101/181229 [1]	170101/181229 [1]	170102/181225	0.58$^{\forall}$
Demographics				
Age	55.0 (48.8 - 55.0)	55.0 (48.0 - 55.0)	56.0 (49.0 - 56.0)	0.28$^{\flat}$
Male: 1	489 (48.9%)	489 (100.0%)	0 (0%)	<0.0001$^{\perp}$
Gender: male	489 (48.9%)	489 (100.0%)	0 (0%)	<0.0001$^{\perp}$
female	511 (51.1%)	0 (0%)	511 (100.0%)	
Country: SW	351 (35.1%)	173 (35.4%)	178 (34.8%)	0.88$^{\perp}$
NO	339 (33.9%)	168 (34.4%)	171 (33.5%)	
DE	310 (31.0%)	148 (30.3%)	162 (31.7%)	

Rows: 1000 (All:1000, male:489, female:511). No duplicate subjects. $^{\perp}$*Chi-square.* $^{\forall}$*Wilcoxon.* $^{\flat}$*t-test. Categorical variable: =unique values=. [n] is missing count (if applicable). Categorical variable: Count (Percent). Date variables: min/max. Numeric variable: Median (Q1-Q3)*

4.4 Behind the scenes of baseline tables

To appear: Describe the tidy nature of current baselines and the functions that turn them into presentation ready tables (`dable_prune`, `dable_fnote`, etc.).

A Loose ends

- Most testing functions do not use weighting - should a warning be issued from the built-in testers when being passed a non-null weight?
- Baseline theme 0: the 'Comparison' is never identified as having standardized differences. Should the name of 'Comparison' and 'Test' perhaps be carried in the object until presentation?
- Subsetting the guide destroys the attributes (stab, levels, missing).

B Testing

This section is for testing all the defaults only, mainly to see that the code goes through.

B.1 Default describers

```
dable(d, type = "real", guide = g)
```

```
##         term         Mean        SD
## 1        age 55.332000000 10.072399
## 2       kids  1.472000000  1.206097
## 3      measA 41.058453071 42.242859
## 4      measB 61.475041370 58.837329
## 5      measM 47.721624171 44.726907
## 6  forgotten  0.001249743  0.577495
## 7 importance  3.168900000  1.739416
```

```
dable(d, type = "catg", guide = g)
```

```
##       term    Level Count Proportion
## 1   status included  1000  1.0000000
## 2   gender     male   489  0.4890000
## 3   gender   female   511  0.5110000
## 4  country       SW   351  0.3510000
## 5  country       NO   339  0.3390000
## 6  country       DE   310  0.3100000
## 7     area    rural   204  0.2050251
## 8     area    urban   791  0.7949749
## 9   region        A   304  0.3040000
## 10  region        B   213  0.2130000
## 11  region        C   126  0.1260000
## 12  region        D   179  0.1790000
## 13  region        E   178  0.1780000
```

```
dable(d, type = "bnry", guide = g)
```

```
##        term Level Count Proportion
## 1      male     1   489      0.489
## 2 offspring  TRUE   740      0.740
```

```
dable(d, type = "date", guide = g)
```

```
##    term    min    max
## 1 index 170101 181229
## 2   end 180101 191229
```

```
dable(d, type = "surv", guide = g)
```

```
##   term     Time Events         Rate
## 1  Foo 325923.1    201 0.0006167099
## 2  Bar 311600.9    300 0.0009627700
```

```
dable(d, type = "lcat", guide = g)
```

```
##   term Unique
## 1  pid   1000
```

B.2 Default comparers

```
dable(d, type = "real", guide = g, gtab = "gender", part = c(F,T))

##         term  Std. diff.
## 1        age -0.06884267
## 2       kids -0.01926886
## 3      measA -0.13901052
## 4      measB -0.13304830
## 5      measM         NaN
## 6  forgotten  0.02244206
## 7 importance -0.09069944

dable(d, type = "catg", guide = g, gtab = "gender", part = c(F,T))

##      term Std. diff.
## 1  status         NA
## 2 country 0.03139396
## 3    area 0.01148310
## 4  region 0.04429413

dable(d, type = "bnry", guide = g, gtab = "gender", part = c(F,T))

##        term Std. diff.
## 1      male        Inf
## 2 offspring 0.03521521

dable(d, type = "date", guide = g, gtab = "gender", part = c(F,T))

##    term  Std. diff.
## 1 index -0.03490307
## 2   end -0.03490307

dable(d, type = "surv", guide = g, gtab = "gender", part = c(F,T))

##   term   Std. diff.
## 1  Foo -0.003611166
## 2  Bar -0.003116198

dable(d, type = "lcat", guide = g, gtab = "gender", part = c(F,T))

##   term Std. diff.
## 1  pid         NA
```

B.3 Default testers

```
dable(d, type = "real", guide = g, gtab = "gender", part = c(F,F,T))

## [META_param] testing 'measM' against 'gender' does not compute; with error:
## Error in `contrasts<-`(`*tmp*`, value = contr.funs[1 + isOF[nn]]):  contrasts
can be applied only to factors with 2 or more levels

##         term     t-test
## 1        age 0.27649031
## 2       kids 0.76081326
## 3      measA 0.02908696
## 4      measB 0.08824107
## 5      measM         NA
## 6  forgotten 0.72301814
## 7 importance 0.15205441

dable(d, type = "catg", guide = g, gtab = "gender", part = c(F,F,T))

## [META_chisq] testing 'status' against 'gender' does not compute; with error:
## Error in stats::chisq.test(x, g):  'x' and 'y' must have at least 2 levels

##      term         p
## 1  status        NA
## 2 country 0.8841969
## 3    area 0.9183285
## 4  region 0.9744276

dable(d, type = "bnry", guide = g, gtab = "gender", part = c(F,F,T))

##        term             p
## 1      male 1.328233e-218
## 2 offspring  6.279693e-01

dable(d, type = "date", guide = g, gtab = "gender", part = c(F,F,T))

##    term  Wilcoxon
## 1 index 0.5834804
## 2   end 0.5834804

dable(d, type = "surv", guide = g, gtab = "gender", part = c(F,F,T))

##   term   Logrank
## 1  Foo 0.3068142
## 2  Bar 0.3793844

dable(d, type = "lcat", guide = g, gtab = "gender", part = c(F,F,T))

##   term no.test
## 1  pid      NA
```

B.4 Default baseline

```
baseline(d, guide = g, gtab = "gender", part = c(F,F,F))

## no table produced

## data frame with 0 columns and 0 rows

baseline(d, guide = g, gtab = "gender", part = c(T,F,F))

##          term             Variable
## 21        pid                   ID
## 10     status A constant: included
## 24      index                Index
## 25        end                  End
## 1         age                  Age
## 8        male              Male: 1
## 11    country          Country: SW
## 12    country            \\quad NO
## 13    country            \\quad DE
## 14       area          Area: rural
## 15       area         \\quad urban
## 16     region            Region: A
## 17     region             \\quad B
## 18     region             \\quad C
## 19     region             \\quad D
## 20     region             \\quad E
## 9   offspring      Offspring: TRUE
## 2        kids            Kid count
## 3       measA            A measure
## 4       measB      Another measure
## 5       measM        Manly measure
## 6   forgotten            forgotten
## 7  importance           importance
## 22       Foo                  Foo
## 23       Bar                  Bar
##                                                                    Summary.info
## 21 Categorical variable: =unique values=. [n] is missing count (if applicable)
## 10 Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 24              Date variables: min/max. [n] is missing count (if applicable)
## 25              Date variables: min/max. [n] is missing count (if applicable)
## 1       Numeric variable: Median (Q1-Q3). [n] is missing count (if applicable)
## 8  Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 11 Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 12 Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 13 Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 14 Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 15 Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 16 Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 17 Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 18 Categorical variable: Count (Percent). [n] is missing count (if applicable)
```

```
## 19 Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 20 Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 9  Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 2       Numeric variable: Median (Q1-Q3). [n] is missing count (if applicable)
## 3       Numeric variable: Median (Q1-Q3). [n] is missing count (if applicable)
## 4       Numeric variable: Median (Q1-Q3). [n] is missing count (if applicable)
## 5       Numeric variable: Median (Q1-Q3). [n] is missing count (if applicable)
## 6       Numeric variable: Median (Q1-Q3). [n] is missing count (if applicable)
## 7       Numeric variable: Median (Q1-Q3). [n] is missing count (if applicable)
## 22  Time-to-event variable: events; rate. [n] is missing count (if applicable)
## 23  Time-to-event variable: events; rate. [n] is missing count (if applicable)
##                     Summary                   Summary
## 21                    =489=                     =511=
## 10              489 (100.0%)              511 (100.0%)
## 24         170101/181229 [1]             170102/181225
## 25         180101/191229 [1]             180102/191225
## 1         55.0 (48.0 - 55.0)        56.0 (49.0 - 56.0)
## 8               489 (100.0%)                   0 (0%)
## 11               173 (35.4%)               178 (34.8%)
## 12               168 (34.4%)               171 (33.5%)
## 13               148 (30.3%)               162 (31.7%)
## 14          101 (20.7%) [2]           103 (20.3%) [3]
## 15              386 (79.3%)               405 (79.7%)
## 16               147 (30.1%)               157 (30.7%)
## 17               105 (21.5%)               108 (21.1%)
## 18                65 (13.3%)                61 (11.9%)
## 19                86 (17.6%)                93 (18.2%)
## 20                86 (17.6%)                92 (18.0%)
## 9                358 (73.2%)               382 (74.8%)
## 2           1.00 (0 - 1.00)           1.00 (0 - 1.00)
## 3    26.9 (11.5 - 26.9) [5]    30.3 (12.6 - 30.3) [2]
## 4  38.7 (18.4 - 38.7) [168] 46.7 (20.3 - 46.7) [171]
## 5         33.6 (14.3 - 33.6)                    - [511]
## 6     0.025 (-0.47 - 0.025)  -0.023 (-0.50 - -0.023)
## 7         3.10 (1.60 - 3.10)        3.30 (1.70 - 3.30)
## 22               92; 0.00057               109; 0.00066
## 23              140; 0.00091               160; 0.0010
```

```
baseline(d, guide = g, gtab = "gender", part = c(F,T,F))
```

```
##          term   Std. diff.
## 14        pid           NA
## 10     status           NA
## 17      index -0.034903068
## 18        end -0.034903068
## 1         age -0.068842671
## 8        male          Inf
## 11    country  0.031393958
## 12       area  0.011483102
## 13     region  0.044294127
## 9   offspring  0.035215213
```

```
## 2        kids -0.019268860
## 3       measA -0.139010518
## 4       measB -0.133048304
## 5       measM          NaN
## 6   forgotten  0.022442056
## 7  importance -0.090699441
## 15       Foo -0.003611166
## 16       Bar -0.003116198
```

```
baseline(d, guide = g, gtab = "gender", part = c(F,F,T))
```

```
## [META_param] testing 'measM' against 'gender' does not compute; with error:
## Error in 'contrasts<-'('*tmp*', value = contr.funs[1 + isOF[nn]]):  contrasts
can be applied only to factors with 2 or more levels
## [META_chisq] testing 'status' against 'gender' does not compute; with error:
## Error in stats::chisq.test(x, g):  'x' and 'y' must have at least 2 levels
```

```
##          term     p.info             p
## 14        pid       <NA>            NA
## 10     status Chi-square            NA
## 17      index   Wilcoxon  5.834804e-01
## 18        end   Wilcoxon  5.834804e-01
## 1         age     t-test  2.764903e-01
## 8        male Chi-square 1.328233e-218
## 11    country Chi-square  8.841969e-01
## 12       area Chi-square  9.183285e-01
## 13     region Chi-square  9.744276e-01
## 9   offspring Chi-square  6.279693e-01
## 2        kids     t-test  7.608133e-01
## 3       measA     t-test  2.908696e-02
## 4       measB     t-test  8.824107e-02
## 5       measM     t-test            NA
## 6   forgotten     t-test  7.230181e-01
## 7  importance     t-test  1.520544e-01
## 15       Foo   Log rank  3.068142e-01
## 16       Bar   Log rank  3.793844e-01
```

```
baseline(d, guide = g, gtab = "gender", part = c(T,T,F))
```

```
##          term             Variable
## 21        pid                   ID
## 10     status A constant: included
## 24      index                Index
## 25        end                  End
## 1         age                  Age
## 8        male              Male: 1
## 11    country          Country: SW
## 12    country            \\quad NO
## 13    country            \\quad DE
## 14       area          Area: rural
## 15       area         \\quad urban
```

```
## 16     region            Region: A
## 17     region             \\quad B
## 18     region             \\quad C
## 19     region             \\quad D
## 20     region             \\quad E
## 9   offspring      Offspring: TRUE
## 2        kids            Kid count
## 3       measA            A measure
## 4       measB      Another measure
## 5       measM        Manly measure
## 6   forgotten            forgotten
## 7  importance           importance
## 22       Foo                  Foo
## 23       Bar                  Bar
##                                                                  Summary.info
## 21 Categorical variable: =unique values=. [n] is missing count (if applicable)
## 10 Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 24               Date variables: min/max. [n] is missing count (if applicable)
## 25               Date variables: min/max. [n] is missing count (if applicable)
## 1       Numeric variable: Median (Q1-Q3). [n] is missing count (if applicable)
## 8  Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 11 Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 12 Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 13 Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 14 Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 15 Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 16 Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 17 Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 18 Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 19 Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 20 Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 9  Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 2       Numeric variable: Median (Q1-Q3). [n] is missing count (if applicable)
## 3       Numeric variable: Median (Q1-Q3). [n] is missing count (if applicable)
## 4       Numeric variable: Median (Q1-Q3). [n] is missing count (if applicable)
## 5       Numeric variable: Median (Q1-Q3). [n] is missing count (if applicable)
## 6       Numeric variable: Median (Q1-Q3). [n] is missing count (if applicable)
## 7       Numeric variable: Median (Q1-Q3). [n] is missing count (if applicable)
## 22  Time-to-event variable: events; rate. [n] is missing count (if applicable)
## 23  Time-to-event variable: events; rate. [n] is missing count (if applicable)
##                     Summary                   Summary   Std. diff.
## 21                    =489=                     =511=           NA
## 10             489 (100.0%)              511 (100.0%)           NA
## 24        170101/181229 [1]             170102/181225 -0.034903068
## 25        180101/191229 [1]             180102/191225 -0.034903068
## 1        55.0 (48.0 - 55.0)       56.0 (49.0 - 56.0) -0.068842671
## 8              489 (100.0%)                   0 (0%)          Inf
## 11              173 (35.4%)               178 (34.8%)  0.031393958
## 12              168 (34.4%)               171 (33.5%)           NA
## 13              148 (30.3%)               162 (31.7%)           NA
```

```
## 14          101 (20.7%) [2]          103 (20.3%) [3]  0.011483102
## 15             386 (79.3%)              405 (79.7%)           NA
## 16              147 (30.1%)              157 (30.7%)  0.044294127
## 17              105 (21.5%)              108 (21.1%)           NA
## 18               65 (13.3%)               61 (11.9%)           NA
## 19               86 (17.6%)               93 (18.2%)           NA
## 20               86 (17.6%)               92 (18.0%)           NA
## 9               358 (73.2%)              382 (74.8%)  0.035215213
## 2           1.00 (0 - 1.00)          1.00 (0 - 1.00) -0.019268860
## 3    26.9 (11.5 - 26.9) [5]   30.3 (12.6 - 30.3) [2] -0.139010518
## 4  38.7 (18.4 - 38.7) [168] 46.7 (20.3 - 46.7) [171] -0.133048304
## 5        33.6 (14.3 - 33.6)                  - [511]          NaN
## 6     0.025 (-0.47 - 0.025)  -0.023 (-0.50 - -0.023)  0.022442056
## 7        3.10 (1.60 - 3.10)       3.30 (1.70 - 3.30) -0.090699441
## 22              92; 0.00057             109; 0.00066 -0.003611166
## 23             140; 0.00091              160; 0.0010 -0.003116198
```

```
baseline(d, guide = g, gtab = "gender", part = c(T,F,T))
```

```
## [META_param] testing 'measM' against 'gender' does not compute; with error:
## Error in `contrasts<-`(`*tmp*`, value = contr.funs[1 + isOF[nn]]):  contrasts
can be applied only to factors with 2 or more levels
##
## [META_chisq] testing 'status' against 'gender' does not compute; with error:
## Error in stats::chisq.test(x, g):  'x' and 'y' must have at least 2 levels
```

```
##          term             Variable
## 21        pid                   ID
## 10     status A constant: included
## 24      index                Index
## 25        end                  End
## 1         age                  Age
## 8        male              Male: 1
## 11    country          Country: SW
## 12    country            \\quad NO
## 13    country            \\quad DE
## 14       area          Area: rural
## 15       area         \\quad urban
## 16     region            Region: A
## 17     region             \\quad B
## 18     region             \\quad C
## 19     region             \\quad D
## 20     region             \\quad E
## 9   offspring      Offspring: TRUE
## 2        kids            Kid count
## 3       measA            A measure
## 4       measB      Another measure
## 5       measM        Manly measure
## 6   forgotten             forgotten
## 7  importance           importance
## 22       Foo                  Foo
```

```
## 23        Bar                  Bar
##                                                                  Summary.info
## 21 Categorical variable: =unique values=. [n] is missing count (if applicable)
## 10 Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 24              Date variables: min/max. [n] is missing count (if applicable)
## 25              Date variables: min/max. [n] is missing count (if applicable)
## 1       Numeric variable: Median (Q1-Q3). [n] is missing count (if applicable)
## 8  Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 11 Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 12 Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 13 Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 14 Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 15 Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 16 Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 17 Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 18 Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 19 Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 20 Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 9  Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 2       Numeric variable: Median (Q1-Q3). [n] is missing count (if applicable)
## 3       Numeric variable: Median (Q1-Q3). [n] is missing count (if applicable)
## 4       Numeric variable: Median (Q1-Q3). [n] is missing count (if applicable)
## 5       Numeric variable: Median (Q1-Q3). [n] is missing count (if applicable)
## 6       Numeric variable: Median (Q1-Q3). [n] is missing count (if applicable)
## 7       Numeric variable: Median (Q1-Q3). [n] is missing count (if applicable)
## 22  Time-to-event variable: events; rate. [n] is missing count (if applicable)
## 23  Time-to-event variable: events; rate. [n] is missing count (if applicable)
##        p.info                   Summary                   Summary             p
## 21       <NA>                     =489=                     =511=            NA
## 10 Chi-square              489 (100.0%)              511 (100.0%)            NA
## 24   Wilcoxon         170101/181229 [1]             170102/181225  5.834804e-01
## 25   Wilcoxon         180101/191229 [1]             180102/191225  5.834804e-01
## 1      t-test        55.0 (48.0 - 55.0)        56.0 (49.0 - 56.0)  2.764903e-01
## 8  Chi-square              489 (100.0%)                    0 (0%) 1.328233e-218
## 11 Chi-square               173 (35.4%)               178 (34.8%)  8.841969e-01
## 12       <NA>               168 (34.4%)               171 (33.5%)            NA
## 13       <NA>               148 (30.3%)               162 (31.7%)            NA
## 14 Chi-square           101 (20.7%) [2]           103 (20.3%) [3]  9.183285e-01
## 15       <NA>               386 (79.3%)               405 (79.7%)            NA
## 16 Chi-square               147 (30.1%)               157 (30.7%)  9.744276e-01
## 17       <NA>               105 (21.5%)               108 (21.1%)            NA
## 18       <NA>                65 (13.3%)                61 (11.9%)            NA
## 19       <NA>                86 (17.6%)                93 (18.2%)            NA
## 20       <NA>                86 (17.6%)                92 (18.0%)            NA
## 9  Chi-square               358 (73.2%)               382 (74.8%)  6.279693e-01
## 2      t-test           1.00 (0 - 1.00)           1.00 (0 - 1.00)  7.608133e-01
## 3      t-test    26.9 (11.5 - 26.9) [5]    30.3 (12.6 - 30.3) [2]  2.908696e-02
## 4      t-test 38.7 (18.4 - 38.7) [168] 46.7 (20.3 - 46.7) [171]  8.824107e-02
## 5      t-test        33.6 (14.3 - 33.6)                  - [511]            NA
## 6      t-test     0.025 (-0.47 - 0.025)  -0.023 (-0.50 - -0.023)  7.230181e-01
```

```
## 7      t-test       3.10 (1.60 - 3.10)       3.30 (1.70 - 3.30)  1.520544e-01
## 22   Log rank             92; 0.00057             109; 0.00066  3.068142e-01
## 23   Log rank            140; 0.00091              160; 0.0010  3.793844e-01
```

```
baseline(d, guide = g, gtab = "gender", part = c(F,T,T))
```

```
## [META_param] testing 'measM' against 'gender' does not compute; with error:
## Error in `contrasts<-`(`*tmp*`, value = contr.funs[1 + isOF[nn]]):  contrasts
can be applied only to factors with 2 or more levels
##
## [META_chisq] testing 'status' against 'gender' does not compute; with error:
## Error in stats::chisq.test(x, g):  'x' and 'y' must have at least 2 levels
```

```
##          term     p.info   Std. diff.             p
## 14        pid       <NA>           NA            NA
## 10     status Chi-square           NA            NA
## 17      index   Wilcoxon -0.034903068  5.834804e-01
## 18        end   Wilcoxon -0.034903068  5.834804e-01
## 1         age     t-test -0.068842671  2.764903e-01
## 8        male Chi-square          Inf 1.328233e-218
## 11    country Chi-square  0.031393958  8.841969e-01
## 12       area Chi-square  0.011483102  9.183285e-01
## 13     region Chi-square  0.044294127  9.744276e-01
## 9   offspring Chi-square  0.035215213  6.279693e-01
## 2         kids     t-test -0.019268860  7.608133e-01
## 3        measA     t-test -0.139010518  2.908696e-02
## 4        measB     t-test -0.133048304  8.824107e-02
## 5        measM     t-test          NaN            NA
## 6    forgotten     t-test  0.022442056  7.230181e-01
## 7   importance     t-test -0.090699441  1.520544e-01
## 15        Foo   Log rank -0.003611166  3.068142e-01
## 16        Bar   Log rank -0.003116198  3.793844e-01
```

```
baseline(d, guide = g, gtab = "gender", part = c(T,T,T))
```

```
## [META_param] testing 'measM' against 'gender' does not compute; with error:
## Error in `contrasts<-`(`*tmp*`, value = contr.funs[1 + isOF[nn]]):  contrasts
can be applied only to factors with 2 or more levels
##
## [META_chisq] testing 'status' against 'gender' does not compute; with error:
## Error in stats::chisq.test(x, g):  'x' and 'y' must have at least 2 levels
```

```
##          term             Variable
## 21        pid                   ID
## 10     status A constant: included
## 24      index                Index
## 25        end                  End
## 1         age                  Age
## 8        male             Male: 1
## 11    country          Country: SW
## 12    country            \\quad NO
## 13    country            \\quad DE
```

```
## 14       area          Area: rural
## 15       area         \\quad urban
## 16     region            Region: A
## 17     region             \\quad B
## 18     region             \\quad C
## 19     region             \\quad D
## 20     region             \\quad E
## 9   offspring      Offspring: TRUE
## 2        kids            Kid count
## 3       measA            A measure
## 4       measB      Another measure
## 5       measM        Manly measure
## 6   forgotten            forgotten
## 7  importance           importance
## 22       Foo                  Foo
## 23       Bar                  Bar
##                                                                  Summary.info
## 21 Categorical variable: =unique values=. [n] is missing count (if applicable)
## 10 Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 24              Date variables: min/max. [n] is missing count (if applicable)
## 25              Date variables: min/max. [n] is missing count (if applicable)
## 1       Numeric variable: Median (Q1-Q3). [n] is missing count (if applicable)
## 8  Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 11 Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 12 Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 13 Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 14 Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 15 Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 16 Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 17 Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 18 Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 19 Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 20 Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 9  Categorical variable: Count (Percent). [n] is missing count (if applicable)
## 2       Numeric variable: Median (Q1-Q3). [n] is missing count (if applicable)
## 3       Numeric variable: Median (Q1-Q3). [n] is missing count (if applicable)
## 4       Numeric variable: Median (Q1-Q3). [n] is missing count (if applicable)
## 5       Numeric variable: Median (Q1-Q3). [n] is missing count (if applicable)
## 6       Numeric variable: Median (Q1-Q3). [n] is missing count (if applicable)
## 7       Numeric variable: Median (Q1-Q3). [n] is missing count (if applicable)
## 22  Time-to-event variable: events; rate. [n] is missing count (if applicable)
## 23  Time-to-event variable: events; rate. [n] is missing count (if applicable)
##        p.info                   Summary                   Summary   Std. diff.
## 21       <NA>                     =489=                     =511=           NA
## 10 Chi-square              489 (100.0%)              511 (100.0%)           NA
## 24   Wilcoxon         170101/181229 [1]             170102/181225 -0.034903068
## 25   Wilcoxon         180101/191229 [1]             180102/191225 -0.034903068
## 1      t-test       55.0 (48.0 - 55.0)       56.0 (49.0 - 56.0) -0.068842671
## 8  Chi-square              489 (100.0%)                   0 (0%)          Inf
## 11 Chi-square               173 (35.4%)              178 (34.8%)  0.031393958
```

```
## 12       <NA>              168 (34.4%)              171 (33.5%)           NA
## 13       <NA>              148 (30.3%)              162 (31.7%)           NA
## 14 Chi-square          101 (20.7%) [2]          103 (20.3%) [3]  0.011483102
## 15       <NA>             386 (79.3%)              405 (79.7%)           NA
## 16 Chi-square              147 (30.1%)              157 (30.7%)  0.044294127
## 17       <NA>              105 (21.5%)              108 (21.1%)           NA
## 18       <NA>               65 (13.3%)               61 (11.9%)           NA
## 19       <NA>               86 (17.6%)               93 (18.2%)           NA
## 20       <NA>               86 (17.6%)               92 (18.0%)           NA
## 9  Chi-square              358 (73.2%)              382 (74.8%)  0.035215213
## 2      t-test          1.00 (0 - 1.00)          1.00 (0 - 1.00) -0.019268860
## 3      t-test   26.9 (11.5 - 26.9) [5]   30.3 (12.6 - 30.3) [2] -0.139010518
## 4      t-test 38.7 (18.4 - 38.7) [168] 46.7 (20.3 - 46.7) [171] -0.133048304
## 5      t-test       33.6 (14.3 - 33.6)                  - [511]          NaN
## 6      t-test    0.025 (-0.47 - 0.025)  -0.023 (-0.50 - -0.023)  0.022442056
## 7      t-test       3.10 (1.60 - 3.10)       3.30 (1.70 - 3.30) -0.090699441
## 22   Log rank              92; 0.00057             109; 0.00066 -0.003611166
## 23   Log rank             140; 0.00091              160; 0.0010 -0.003116198
##                p
## 21            NA
## 10            NA
## 24  5.834804e-01
## 25  5.834804e-01
## 1   2.764903e-01
## 8  1.328233e-218
## 11  8.841969e-01
## 12            NA
## 13            NA
## 14  9.183285e-01
## 15            NA
## 16  9.744276e-01
## 17            NA
## 18            NA
## 19            NA
## 20            NA
## 9   6.279693e-01
## 2   7.608133e-01
## 3   2.908696e-02
## 4   8.824107e-02
## 5             NA
## 6   7.230181e-01
## 7   1.520544e-01
## 22  3.068142e-01
## 23  3.793844e-01
```