

```
new() ~ DISCRETE VALUE=LABEL VALUE
few(...., name=NULL, type="auto", default=NULL, multilabel=FALSE, ignore_case=FALSE)
... = named pairs key=label | .missing=other = special labels | name = register in library

few(
  "M" = "Male",
  "F" = "Female",
  .missing = "Unknown",
  other = "Other Gender",
  name = "sex"
)

sex CHAR
M - Male
F - Female
.missing - Unknown
other - Other Gender
```

```
finfun(..., name=NULL, target_type="numeric", missing_value=NA)
... = named pairs label=value | target_type = "numeric"|"character" output type

finfun(
  "Male" = 1,
  "Female" = 2,
  name = "sex_inv"
)
```

<pre>new_bid, name=NAME, type=ACTIVE) Creates both a VALUE format and matching _inv INVALUE simultaneously</pre>	<pre>Creates both "status" VALUE format and "status_inv" INVALUE</pre>
<pre>new_bid, name = "Active", "AI" = "Inactive", name = "status"</pre>	
<pre>NEW_DATE[] → DATE/TIME/DATE/TIME FORMAT</pre>	
<pre>new_date:pattern, name=NAME, type=DATE, _missing(NAME) pattern = SAS format name (DATE) or R strftime pattern (%d,%M,%Y)</pre>	
<pre>new_date("DATE\$*", name="bday") new_date("%d-%M-%Y", name="ru", type="date") new_date("PDT\$T18", name="ru", _missing("% DATE")</pre>	<pre>bday DATE Pattern: %bday (DATE). SAS names auto-resolved; R patterns also</pre>

OPTIONS FOR FNEW()		
PARAM	DEFAULT	WHAT IT DOES
name	NULL	Register in format library
type	"auto"	"numeric"/"character" key type
multilabel	FALSE	Allow multiple labels per value
ignore_case	FALSE	Case-insensitive matching
.missing	-	Label for NA/NaN/" "
.other	-	Fallback for unmatched values

```
fput() — APPLY FORMAT (GENERIC)
```

```
fput(x, format, ..., keep_na=FALSE)
```

x = values | format = name or object | ... = extra args for expression labels

```
fput(
  c("M", "F", "M", NA, "X"),
  "sex"
```

"Male"

"Female"

"Male"

"Unknown"

"Other Gender"

```
fputn(x, format_name, ...) ~ fputc(x, format_name, ...)
```

Numeric/character shorthand - format_name is always a string name

```
fputn(c(5, 30, 70), "age")
```


Child	Adult	Senior
Male	Female	Eyes1a

```
fputx(..., format, sep="", keep_nas=FALSE)
... = vectors pasted into composite key (e.g. SUBJ|VISIT) before lookup

fnew("A|1"=="2025-01-15",
     "A|2"=="2025-02-20",
     "B|1"=="2025-03-10",
     .other="NOT FOUND",
     name="vd")
fputx(c("A", "A", "B", "B"),
      c(1, 2, 1, 3), format="w",
```

```
data-driven from data frame
fmap(paste(SV$SUBJ,
           SV$VISIT, sep=" ",
           as.Date(SV$DTC)),
     name="svdtn", type="date")
#> OLS formula: Date ~
#> VALUE SVdtn (Date)
#> format: %Y-%m-%d, nocase)
#> "S11"="2025-01-15";
#> fpar1("SSSUBJ", Q$SVISIT,
#> format="svdtn")
```

```
fput_df(data, ..., suffix="_fmt", replace=FALSE)
... = col-format pairs | suffix = new column suffix | replace = overwrite original
```

```
fput_df(df,
sex = format_get("sex"),
age = format_get("age"),
suffix = "_lbl")
```

SEX	AGE	SEX_LBL
M	15	Male
F	25	Female
NA	35	Unknown

#	CONDITION	RESULT
1	NA / NaN / ""	.missing label
2	Exact match	Mapped label
3	Range match	Range label
4	No match	.other / original

```
is_missing(x)
Returns TRUE for NA, NaN, and empty string ""
```

is_missing(NA)	# TRUE	TRUE
is_missing(NaN)	# TRUE	TRUE
is_missing("")	# TRUE	TRUE
is_missing("x")	# FALSE	FALSE

```
fput(c("M", NA), "sex")
```

Male Unknown

```
fput(c("M", NA), "sex",  
      keep_na = TRUE)
```

Male NA

FPARSE() — PARSE VALUE/INVALUE TEXT

```
fparse(text=NULL, file=NULL)
```

```

parsetext(text = '
VALUE age (numeric)
[0, 18)      = "Child"
[18, 65)     = "Adult"
[65, HIGH]   = "Senior"
.missing     = "Age Unknown"
;')
fputn(c(5,18,65,NA), "age")

```

RANGE SYNTAX REFERENCE		
SYNTAX	MEANING	MATH
[a, b]	both inclusive	$a \leq x \leq b$
[a, b)	right exclusive	$a \leq x < b$
(a, b]	left exclusive	$a < x \leq b$
(a, b)	both exclusive	$a < x < b$
HIGH	== upper bound	
LOW	== lower bound	

	BMI	RESULT
[0, 18.5) = "Underweight"	16.2	Underweight
[18.5, 25) = "Normal"	25.0	Overweight
[25, 30) = "Overweight"	35.1	Obese
[30, HIGH] = "Obese"	NA	No data
.missing = "No data"		

EXCLUSIVE / INCLUSIVE BOUNDS + .OTHER	
<code>fparse(text = "</code>	
<code>VALUE score (numeric)</code>	
<code>(0, 50) = "Low"</code>	SCORE LABEL
<code>(50, 100) = "High"</code>	0 Out of range
<code> other = "Out of range"</code>	50 Low
<code>;</code>	51 High
	101 Out of range

```

parse(text = "
  VALUE race(character)
    "W"="White" "B"="Black"
    "A"="Asian"
    .missing = "Unknown"
;
INVALUE race_inv

```

Registers **both** in library:

race	VALUE	(char)
race_inv	INVALUE	(num)

```

RANGE_SPEC() := BUILD RANGE STRINGS
range_spec(low, high, label, inc_lowTRUE, inc_highFALSE)
Programatic range key builder for fnew() - returns "low,high,inc_low,inc_high"
range_spec(0, 10)
range_spec(10, Inf,
inc_high = FALSE)

```

```
FINPUTN() — LABELS → NUMERIC
```

```
finputn(x, invalue_name)
```

Applies named invalue, returns numeric vector

```
inputs(
  c("Male", "Female", "Unknown"),
  "sex_inv"
)
```

```
FINPUTC() — LABELS → CHARACTER
```

```
finputc(x, invalname)
```

Applies named invalname, returns character vector

```
finputc(
  c("Active", "Pending"),
  "status_inv"
```

"A" **"P"**

ROUND-TRIP: VALUE ↔ INVALUE

"A" → fputc "status" → "Active" → fputc "status_inv" → "A"

DATE FORMATS (AUTO-RESOLVED)	
<code>sn(Sys.Date(), "DATE.")</code>	DATE.
<code>sn(Sys.Date(), "MMDDYY10.")</code>	MMDDYY10.
<code>sn(Sys.Date(), "YYMMDD10.")</code>	YYMMDD10.
<code>sn(Sys.Date(), "WORDDATE.")</code>	WORDDATE.
<code>sn(Sys.Date(), "QTR.")</code>	QTR.

CS	TIME0	TIMES	HMM
	0:00:00	0:00	00:00
00	1:00:00	1:00	01:00
000	12:30:00	12:30	12:30
999	23:59:59	23:59	23:59

FORMAT	→ RESULT
DATETIME20.	18MAR2026:13:48:58
DATETIME13.	18MAR26:13:48
DTDATE.	18MAR2026
DTYYMMDD.	2026-03-18

```
new_date("DATE9",
name = "vfmt",
missing = "NOT RECORDED")

df(patients,
sit_date = v)

ID VISIT_DATE VISIT_FMT
1 2025-01-10 10JAN2025
2 2025-02-15 15FEB2025
3 <NA> NOT RECORDED
```

```

"date9.",
"mmdyy10.",
name = "wfmt",
type = "numeric")

wfmt <- fputn(key, "wfmt")
datef <- fputn(number, datef)

```

```
PRINT() — INSPECT REGISTERED FORMATS
print(name=NULL)
args = list all | name = show detail for one format
```

```

> list() # list all
> list("sex") # detail

```

age	VALUE	(num)
sex	VALUE	(char)
sex_inv	INVALUE	(num)

```
format_get() # Clear()
format_get(name) - retrieve format object      fclear(name=NULL) - remove one or all
format_get("sex")
format_get("sex") # remove one
```

```
export(..., formats=NULL, file=NULL)
# or formats = format objects | file = write to file (else returns string)
fexport(bmi = bmi_fmt)
```

```
[0, 18.5) = "Underweight"  
[18.5, 25) = "Normal"  
[25, 30) = "Overweight"  
[30, HIGH) = "Obese"  
;
```

```
fimport(file, register=TRUE, overwrite=TRUE)
```

```

# Import the data
df = fimport(csv,
             register = FALSE)

# Print the first 5 rows of the data
print(v_m[["GENDER"]])

```