

INSTALL & LOAD

```
# From CRAN
install.packages("ksformat")
# From Github (dev)
remotes::install_github("crow16384/ksformat")
library(ksformat)
```

FORMAT CREATION

FNEW() — DISCRETE VALUE+LABEL VALUE

```
fnew(..., name=NULL, type="auto", default=NULL, multilabel=FALSE, ignore_case=FALSE)
... => named pairs key="label" | .missing/other = special labels | name = register in library
```

fnew

| | |
|---|--|
| "M" = "Male", "F" = "Female", .missing = "Unknown", .other = "Other Gender", name = "sex" | SEX CHAR M - Male F - Female .missing - Unknown .other - Other Gender |
|---|--|

FINPUT() — REVERSE (LABEL+VALUE) INVALUE

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

finput

| | |
|--|--|
| "Male" = 1, "Female" = 2, name = "sex_inv" | sex_inv INVALUE -> NUM Male - 1 Female - 2 |
|--|--|

FNEW_BID() — BIDIRECTIONAL VALUE + INVALUE

```
fnew_bid(..., name=NULL, type="auto")
Creates both a VALUE format and matching _inv INVALUE simultaneously
```

fnew_bid

| | |
|---|--|
| "A" = "Active", "I" = "Inactive", name = "status" | Creates both: "status" VALUE format "status_inv" INVALUE |
|---|--|

FNEW_DATE() — DATE/TIME/DATETIME FORMAT

```
fnew_date(pattern, name=NULL, type="auto", .missing=NULL)
pattern = SAS format name (DATE9.) or R strftime pattern (%d.%m.%Y)
```

fnew_date

| | |
|---|--|
| "dd.mm.yy", name="r", type="date" | Pattern: %d.%m.%Y (DATE9.) |
| "MMDDYY18.", name="us", .missing="NO DATE" | SAS names auto-resolved; R patterns also supported |

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 |

FORMAT APPLICATION

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() / **FPUTC()** — TYPED APPLY

```
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" |

FPUTK() — COMPOSITE KEY APPLY

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

**fnew("A1"="2025-01-15",
"A2"="2025-02-20",
"B1"="2025-03-10",
.other="NOT FOUND",
name="vd")**

| |
|--|
| "2025-01-15" "2025-02-20" "2025-03-10" |
| "NOT FOUND" |

FPUTK() + **FMAP()** / **FPARSE()** — DATE LOOKUP

```
# Data-driven from data frame
fnew(fmap(paste(SV$SUBJ,
SV$VISIT, sep="|"),
as.Date(SV$DTC)),
name="svdtn", type="Date")
# Or via fparse() text:
# VALUE svdtn (Date)
# format: %Y-%m-%d %H:%M:%S
# "S1"|"="2025-01-15";
fputk(SV$SUBJ, SV$VISIT,
format="svdtn")
```

**fputk(c("A", "A", "B", "B"),
c(1,2,1,3), format="vd")**

| |
|------------------------------|
| "2025-01-15" "2025-02-20" NA |
|------------------------------|

FPUT_DF() — FORMAT DATA FRAME COLUMNS

```
fput_df(data, ..., suffix="fmt", replace=FALSE)
... => colformat pairs | suffix = new column suffix | replace = overwrite original
```

**fput_df(df,
sex = format_get("sex"),
age = format_get("age"),
suffix = "tbl")**

| SEX | AGE | SEX_LBL | AGE_LBL |
|-----|-----|---------|---------|
| M | 15 | Male | Child |
| F | 25 | Female | Adult |
| NA | 35 | Unknown | Adult |

MISSING VALUE HANDLING

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

IS_MISSING() — CHECK FOR MISSING VALUES

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

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

KEEP_NA OPTION

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

| |
|--------|
| "Male" |
|--------|

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

| | |
|--------|----|
| "Male" | NA |
|--------|----|

TEXT PARSING & NUMERIC RANGES

FPARSE() — PARSE VALUE/INVALUE TEXT

```
fparse(text=NULL, file=NULL)
text = format definition string | file = path to .txt file with definitions
```

fparse(text = "
VALUE age (numeric)
(0, 10) = "Child"
(10, 65) = "Adult"
(65, HIGH) = "Senior"
.missing = "Age Unknown"
")

| |
|--|
| "Child" "Adult" "Senior" "Age Unknown" |
|--|

RANGE SYNTAX REFERENCE

| SYNTAX | MEANING | MATH |
|--------|-----------------|-------------|
| [a, b] | both inclusive | a <= x <= b |
| (a, b) | right exclusive | a < x < b |
| [a, b) | left exclusive | a < x <= b |
| (a, b] | both exclusive | a < x < b |
| HIGH | => upper bound | a < x < b |
| LOW | => lower bound | |

BMI WITH DECIMALS + .MISSING

```
fparse(text = '
VALUE bmi (numeric)
(0, 10.5) = "Underweight"
(10.5, 25) = "Normal"
(25, 30) = "Overweight"
(30, HIGH) = "Obese"
.missing = "No data"
')
```

| BMI | RESULT |
|------|-------------|
| 16.2 | Underweight |
| 25.0 | Overweight |
| 35.1 | Obese |
| NA | No data |

EXCLUSIVE / INCLUSIVE bounds + .OTHER

```
fparse(text = '
VALUE score (numeric)
(0, 50) = "Low"
(50, 100) = "High"
.other = "Out of range"
')
```

| SCORE | LABEL |
|-------|--------------|
| 0 | Out of range |
| 50 | Low |
| 51 | High |
| 101 | Out of range |

PARSE MULTIPLE FORMATS AT ONCE

```
fparse(text = '
VALUE race (character)
"R"="White" "B"="Black"
"A"="Asian"
.missing = "Unknown"
')
INVALUE race_inv
"White"=1 "Black"=2
"Asian"=3
')
```

Registers both in library:

| | |
|----------|------------------|
| race | EXCLUSIVE (char) |
| race_inv | INVALUE (num) |

RANGE_SPEC() — BUILD RANGE STRINGS

```
range_spec(low, high, label, inc_low=TRUE, inc_high=FALSE)
Programmatic range key builder for fnew() - returns "low,high,inc_low,inc_high"
```

range_spec(0, 10)

| |
|-------------------|
| "0,10,TRUE,FALSE" |
|-------------------|

**range_spec(10, Inf,
inc_high = FALSE)**

| |
|---------------------|
| "10,Inf,TRUE,FALSE" |
|---------------------|

REVERSE FORMATTING (INVALUE)

FINPUTN() — LABELS + NUMERIC

```
finputn(x, invalue_name)
Applies named invalue, returns numeric vector
```

**finputn(c("Male", "Female", "Unknown"),
c("Male", "Female", "Unknown"),
"sex_inv")**

| |
|--------|
| 1 2 NA |
|--------|

FINPUTC() — LABELS + CHARACTER

```
finputc(x, invalue_name)
Applies named invalue, returns character vector
```

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

| |
|---------|
| "A" "P" |
|---------|

ROUND-TRIP: VALUE <- INVALUE

"A" -> fputc("status" -> "Active" -> finputc("status_inv" -> "A"

DATE, TIME & DATETIME

SAS DATE FORMATS (AUTO-RESOLVED)

| fputn(Sys.Date(), "DATE9.") | FORMAT | -> RESULT |
|--------------------------------|-----------|----------------|
| DATE9. | DATE9. | 10MAR2026 |
| fputn(Sys.Date(), "MMDDYY10.") | MMDDYY10. | 03/18/2026 |
| fputn(Sys.Date(), "YMDDD10.") | YMDDD10. | 2026-03-18 |
| fputn(Sys.Date(), "WORDDATE.") | WORDDATE. | March 18, 2026 |
| fputn(Sys.Date(), "QTR.") | QTR. | 1 |

TIME FORMATS (SECONDS SINCE MIDNIGHT)

| SECS | TIMEB. | TIME5. | HHMM. |
|-------|----------|--------|-------|
| 0 | 0:00:00 | 0:00 | 00:00 |
| 3600 | 1:00:00 | 1:00 | 01:00 |
| 45000 | 12:30:00 | 12:30 | 12:30 |
| 86399 | 23:59:59 | 23:59 | 23:59 |

DATETIME FORMATS

| fputn(Sys.time(), "DATETIME20.") | FORMAT | -> RESULT |
|----------------------------------|-------------|--------------------|
| DATETIME20. | DATETIME20. | 10MAR2026:13:48:58 |
| DATETIME13. | DATETIME13. | 10MAR26:13:48 |
| DTDATE. | DTDATE. | 10MAR2026 |
| DTYMDD. | DTYMDD. | 2026-03-18 |

CUSTOM DATE FORMAT + .MISSING

```
v <- fnew_date("DATE9.",  
name = "vfmt",  
.missing = "NOT RECORDED")
```

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

PUTN WORKFLOW (SAS-STYLE DYNAMIC DISPATCH)

```
fnew("1" = "date9.",  
"2" = "mmdyy10.",  
name = "vfmt",  
type = "numeric")
```

datefmt <- fputn(key, "vfmt")
date <- fputn(number, datefmt)

| NUM | KEY | FMT | DATE |
|-------|-----|----------|------------|
| 12103 | 1 | date9. | 03FEB2003 |
| 10899 | 2 | mmdyy10. | 11/07/1999 |

FORMAT LIBRARY

FPRINT() — INSPECT REGISTERED FORMATS

```
fprint(name=NULL)
No args = list all | name = show detail for one format
```

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

| | | |
|---------|---------------|---|
| age | VALUE (num) | 3 |
| sex | VALUE (char) | 2 |
| sex_inv | INVALUE (num) | 2 |

FORMAT_GET() / **FCLEAR()**

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

fclear("sex") # remove one
fclear() # clear all

Retrieve format object
Remove single format
Clear entire library

EXPORT() — EXPORT TO PARSEABLE TEXT

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

cat(fexport(bmi = bmi_fmt))

| |
|---------------------------|
| VALUE bmi (numeric) |
| (0, 10.5) = "Underweight" |
| (10.5, 25) = "Normal" |
| (25, 30) = "Overweight" |
| (30, HIGH) = "Obese" |
| . |

FIMPORT() — SAS CNTLOUT CSV

```
fimport(file, register=TRUE, overwrite=TRUE)
file = CNTLOUT CSV path | register = auto-add to library | overwrite = replace existing
```

x <- fimport("cntlout.csv")

**m <- fimport(csv,
register = FALSE)**

fput(v, m[["GENDER"]])

| AGEGRP | VALUE (num) |
|--------|--------------|
| BRECAT | VALUE (num) |
| GENDER | VALUE (char) |

MULTILABEL FORMATS

FPUT_ALL() — ONE VALUE + MULTIPLE LABELS MULTILABEL=TRUE

```
fput_all(x, format, ..., keep_na=FALSE)
Returns list of character vectors - all matching labels per value
```

fnew

| | |
|--|---|
| "0,5,7,1" = "Infant", "6,11,17,1" = "Child", "12,17,17,1" = "Adolescent", "0,17,17,1" = "Pediatric", "18,64,7,1" = "Adult", "65,Inf,1,1" = "Elderly", "18,Inf,1,1" = "Non-Pediatric", name = "age_cat", type = "numeric", multilabel = TRUE | 3 -> Infant Pediatric 14 -> Adolescent Pediatric 25 -> Adult Non-Pediatric 70 -> Elderly Non-Pediatric |
|--|---|

fput() -> first match only -> fput_all() -> all matching labels (list)

AT SEVERITY GRADING (CLINICAL)

```
fnew("1,1,1,1" = "Mild",  
"2,2,2,1" = "Moderate",  
"3,3,1,1" = "Severe",  
"4,4,1,1" = "Life-threat.",  
"5,5,1,1" = "Fatal",  
"3,5,1,1" = "Serious",  
"1,2,1,1" = "Non-serious",  
name = "age",  
type = "numeric",  
multilabel = TRUE)
```

| |
|---------------------------|
| 1 -> Mild Non-serious |
| 2 -> Moderate Non-serious |
| 3 -> Severe Serious |
| 4 -> Life-threat Serious |
| 5 -> Fatal Serious |

EXPRESSION LABELS

SPRINT WITH .X1 (EVALUATED AT APPLY-TIME)

```
stat <- fnew("p" = "sprintf('%s', .x1)",  
"pct" = "sprintf('%s.1f%%',  
name = "stat"
```

| |
|-----------------------------|
| "42%" "5.3%" "100%" "25.5%" |
|-----------------------------|

fput

| | |
|---|--|
| c("n", "pct", "n", "pct"), stat, c(42, .053, 100, .255) | |
|---|--|

TWO ARGS (.X1, .X2) + SCALAR RECYCLING

```
r <- fnew("ratio" = "sprintf('%s/%s',  
.x1, .x2)")
```

fput("ratio", r, 3, 10)

| |
|-------------------------|
| "3/10" |
| "42(N=100)" "55(N=100)" |

.x2=100 recycled for all elements

**lbl <- fnew("val" = "sprintf('%s(N=%s)',
.x1, .x2)")**

**fput(c("val", "val"), lbl,
c(42, 55), 100)**

VECTORIZED FORMAT NAMES

EACH ELEMENT USES A DIFFERENT FORMAT (SAS PUTC/PUTN)

```
fnew("1" = "groupa",  
"2" = "groupb",  
"3" = "groupc",  
name = "typefmt",  
type = "numeric")
```

define group; y, z ...
resprfmt <- fput(type,
"typefmt")

**word <- fputc(response,
resprfmt)**

| TYPE | RESP | FMT | WORD |
|------|----------|--------|----------|
| 1 | positive | groupa | agree |
| 1 | negative | groupa | disagree |
| 2 | positive | groupb | accept |
| 2 | negative | groupb | reject |

FUNCTION REFERENCE

| CATEGORY | FUNCTION | PURPOSE |
|----------|--------------------------|-------------------------|
| Create | fnew(..., name) | Discrete value->label |
| | fmap(keys, values) | Data-driven key->value |
| | finput(..., name) | Reverse label->value |
| | fnew_bid(..., name) | Both format + invalue |
| | fnew_date(pattern, name) | Date/time/datetime |
| Apply | fput(x, format, ...) | Generic apply |
| | fputn(x, name, ...) | Numeric apply |
| | fputc(x, name, ...) | Character apply |
| | fputk(..., format, sep) | Composite key apply |
| | fput_all(x, format) | All labels (multilabel) |
| | fput_df(data, ...) | Data frame columns |
| Reverse | finput(x, name) | Labels -> numeric |
| | finputc(x, name) | Labels -> character |
| Library | format_get(name) | Get from library |
| | fprint(name) | List/inspect |
| | fclear(name) | Remove format(s) |
| | fimport(file) | SAS CNTLOUT CSV |
| Utils | fparse(text, file) | Parse definitions |
| | fexport(..., file) | Export to text |
| | range_spec(lo, hi) | Build range key |
| Check | is_missing(x) | NA/NaN/"" -> TRUE |