

### INSTALL & LOAD

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

### FORMAT CREATION

```
fneww[...] - 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
```

```
finput[...] - REVERSE (LABEL+VALUE) [INVALUE]
finput(..., name=NULL, target_type="numeric", missing_value=NA)
... = named pairs label=value | target_type = "numeric" | character output type
```

```
fnew_bid[...] - BIDIRECTIONAL [VALUE] + [INVALUE]
fnew_bid(..., name=NULL, type="auto")
... = named pairs value=fmt and existing_inw INVALUE simultaneously
```

```
fnew_date[...] - DATE/TIME/DATETIME FORMAT
fnew_date(pattern, name=NULL, type="auto", .missing=NULL)
... = SAS format name (DATE9.) or R strftime pattern (dd.MM.yyyy)
```

Table with 3 columns: PARAM, DEFAULT, WHAT IT DOES. Lists options for fneww like name, type, multilabel, ignore\_case, .missing, .other.

### FORMAT APPLICATION

```
fput[...] - APPLY FORMAT (GENERAL)
fput(x, format, ..., keep_na=FALSE)
... = values | format = name or object | ... = extra args for expression labels
```

```
fputn[...] - TYPED APPLY
fputn(x, format_name, ..., fputc(x, format_name, ...))
... = Numeric/character shorthand - format_name is always a string name
```

```
fputk[...] - COMPOSITE KEY APPLY
fputk(..., format, sep="|", keep_na=FALSE)
... = vectors passed into composite key (e.g. SUBJ|VISIT) before lookup
```

```
fputk[...] + fmap[...] / fparse[...] - DATE LOOKUP
# Data-driven from data frame
fnew(fmap(paste(SV5SUB),
SV5VISIT, sep="|"),
as.Date(SV5CTC1),
name="sv5ctn", type="Date")
... = Returns native Date = arithmetic works
```

```
fput_df[...] - FORMAT DATA FRAME COLUMNS
fput_df(data, ..., suffix="fmt", replace=FALSE)
... = col=format pairs | suffix = new column suffix | replace = overwrite original
```

Table showing columns: SEX, AGE, SEX\_LBL, AGE\_LBL. Rows show data for Male, Female, Unknown.

### MISSING VALUE HANDLING

Table with 3 columns: #, CONDITION, RESULT. Shows how NA, NaN, exact and range matches are handled.

```
is_missing[...] - CHECK FOR MISSING VALUES
is_missing(x)
Returns TRUE for NA, NaN, and empty string ""
```

```
KEEP_NA OPTION
fputc("M", NA, "sex")
fputc("M", NA, "sex", keep_na = TRUE)
```

### TEXT PARSING & NUMERIC RANGES

```
fparse[...] - PARSE VALUE/INVALUE TEXT
fparse(text=NULL, file=NULL)
text = format definition string | file = path to text file with definitions
```

Table with 3 columns: SYNTAX, MEANING, MATH. Explains range notations like (a,b), [a,b], (a,b], [a,b].

```
BMI WITH DECIMALS + .MISSING
fparse(text = '
VALUE bmi (numeric)
(0, 18) = "Child"
(18, 65) = "Adult"
(65, HIGH) = "Senior"
.missing = "Age Unknown"
')
fputn(c(5, 18, 65, NA), "age")
```

Table with 3 columns: EXCLUSIVE / INCLUSIVE BOUNDS + .OTHER. Shows how ranges like (0, 50) and [50, 180] are defined.

```
PARSE MULTIPLE FORMATS AT ONCE
fparse(text = '
VALUE race (character)
"0"="White" "1"="Black"
"A"="Asian" "B"="Black"
.missing = "Unknown"
')
INVALUE race_inw
"White"="0" "Black"="1"
"Asian"="3"
')
```

```
RANGE_SPEC[...] - BUILD RANGE STRINGS
range_spec(low, high, label, inc_low=TRUE, inc_high=FALSE)
... = Programmatic range key builder for fneww() - returns "low,high,inc_low,inc_high"
range_spec(0, 18)
range_spec(18, Inf, inc_high = FALSE)
```

### REVERSE FORMATTING (INVALUE)

```
finputn[...] - LABELS + NUMERIC
finputn(x, invalue_name)
... = Numeric invalue, returns numeric vector
```

```
finputn[...] - LABELS + CHARACTER
finputn(x, invalue_name)
... = Numeric invalue, returns character vector
```

```
ROUND-TRIP: VALUE ↔ INVALUE
"A" → fputc("status") → "Active" → finputn("status_inw") → "A"
```

### DATE, TIME & DATETIME

```
SAS DATE FORMATS (AUTO-RESOLVED)
fputn(Sys.Date(), "DATE9.")
fputn(Sys.Date(), "MMDDYY18.")
fputn(Sys.Date(), "YYMMDD18.")
fputn(Sys.Date(), "WORDDATE.")
fputn(Sys.Date(), "QTR.")
```

Table showing SAS DATE FORMATS (AUTO-RESOLVED) with columns: FORMAT, RESULT. Includes DATE9, MMDDYY18, YYMMDD18, WORDDATE, QTR.

```
TIME FORMATS (SECONDS SINCE MIDNIGHT)
SECS TIMEER TIMES HHMM
3600 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(), "DATEIME20.")
fputn(Sys.time(), "DATEIME20.")
```

Table showing DATETIME FORMATS with columns: FORMAT, RESULT. Includes DATEIME20, DTDATE, DTYYMMDD.

```
CUSTOM DATE FORMAT + .MISSING
v <- fnew_date("DATE9.",
name = "vfmt",
.missing = "NOT RECORDED")
fput_df(patients,
visit_date = v)
fnew(
"1" = "date9.",
"2" = "mddy18.",
name = "vfmt",
type = "numeric")
```

Table showing CUSTOM DATE FORMAT + .MISSING with columns: ID, VISIT\_DATE, VISIT\_FMT. Includes rows for 2025-01-10, 2025-02-15, and <NA>.

```
PUTN WORKFLOW (SAS-STYLE DYNAMIC DISPATCH)
fnew(
"1" = "date9.",
"2" = "mddy18.",
name = "vfmt",
type = "numeric")
datefmt <- fputn(key, "vfmt")
date <- fputn(number, datefmt)
```

### FORMAT LIBRARY

```
fprint[...] - INSPECT REGISTERED FORMATS
fprint(name=NULL)
No args = list all | name = show detail for one format
```

```
FORMAT_DET() FCLEAR()
format_get(name) - retrieve format object fclear(name=NULL) - remove one or all
fmat <- format_get("sex")
fclear("sex") # remove one
fclear() # clear all
```

```
FEEXPORT() - EXPORT TO PARSEABLE TEXT
fexport(..., formats=NULL, file=NULL)
... = list of formats = format objects | file = write to file (else returns string)
cat(fexport(bmi = bmi_fmt))
```

```
fimport[...] - SAS CNTLOUT CSV
fimport(file, register=TRUE, overwrite=TRUE)
file = CNTLOUT CSV path | register = auto-load to library | overwrite = replace existing
x <- fimport("cntout.csv")
```

```
m <- fimport(csv, register = FALSE)
fprint(m[,"GENER"])
```

Table showing columns: AGEGRP, BRCAT, GENDER. Rows show values like 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.