Cheatography

Data management in R Cheat Sheet

by xeonkai via cheatography.com/31513/cs/9676/

Manipulating dataframes

Adding new columns

```
mydata <- transform(mydata, sumx = x1 + x2, meanx = (x1 + x2)/2)
```

`total <- cbind(A,B)` - each object to have same no. of rows and sorted in same order

```
merge(dfA, dfB, by = c("ID","Country")
```

Adding new rows

`total <- rbind(A, B)` - each object to have same variables

Recoding variables

```
leadership <- within(leadership, { agecat <- NA agecat[age > 75] <- "Elder" agecat[age >= 55 & age <=75] <- "Middle Aged" agecat[age < 55] <- "Young"})
```

Other recoding functions: `car` package - `recode()`, `doBy` package - `recodevar()`, `cut()` in R

Renaming variables

`reshape` package - `rename()`

```
rename(dataframe, c(oldname="newname", oldname="newname",...)
```

```
names() - names(leadership)[6:8] <- c("item1","item2","item3")
```

Missing values

`is.na()`, `na.rm=TRUE`, `na.omit()` - deletes any row with missing data

Date values

```
as.Date(x, "input_format")
```

Default format: yyyy-mm-dd

Manipulating dataframes (cont)

```
Sys.Date(), date(), difftime(date1, date2, units="weeks")
```

Converting character to dates: `help(as.Date)`, `help(strftime)`

Formatting dates and time: `help(ISOdatetime)`

`lubridate` and `fcalendar` package

Sorting data

`order()`: default ascending, prepend sorting variable with `-` for descending

e.g. `df2 <- df[order(df$gender, -df$age),]`

Date formats

%d	Day as a number (0-31)	01-31
%a	Abbreviated weekday	Mon
%A	Unabbreviated weekday	Monday
%m	Month (00-12)	00-12
%b	Abbreviated month	Jan
%B	Unabbreviated month	January
%y	2-digit year	07
%Y	4-digit year	2007

Type conversions

`is.numeric()`	`as.numeric()`
`is.character()`	`as.character()`
`is.vector()`	`as.vector()`
`is.matrix()`	`as.matrix()`
`is.data.frame()`	`as.data.frame()`
`is.factor()`	`as.factor()`
`is.logical()`	`as.logical()`

Subsetting datasets

Selecting variables

```
new <- df[, c(6:10)]
```

```
new <- df[c("q1","q2","q3")]
```

```
myvars <- paste("q", 1:3, sep="")
```

```
new <- df[myvars]
```

Excluding variables

```
myvars <- names(leadership) %in% c("q1","q2")
```

```
new <- df[!myvars]
```

```
new <- df[c(-1,-2)]
```

```
df$q1 <- NULL
```

Selecting observations

```
new <- df[1:3,]
```

```
new <- df[which(df$q1=="M" & df$q2 >30),]
```

By **xeonkai**
cheatography.com/xeonkai/

Not published yet.
Last updated 8th November, 2016.

Cheatography

Data management in R Cheat Sheet

by xeonkai via cheatography.com/31513/cs/9676/

Subsetting datasets (cont)

Random Samples

```
mysample <-
df[sample(1:nrow(df), 3,
replace=FALSE),]
```

`sampling` and `survey` package

`subset()` function

e.g.

```
new <- subset(df, age >=35 | age <
24, select = c(q1, q2, q3))
new <- subset(df, gender == "M" &
age >25, select = gender:q3)
```

SQL in R

`sqldf` package

```
library(sqldf)
new <- sqldf("select * from mtcars
where carb=1 order by mpg",
row.names=TRUE)
sqldf("select avg(mpg) as avg_mpg,
avg(disp) as avg_disp, gear from
mtcars where cyl in (4,6) group by
gear")
```

Mathematical functions

`abs(x)`	Absolute value
`sqrt(x)`	Square root. Same as `25^(0.5)`.
`ceiling(x)`	Smallest integer not less than x
`floor(x)`	Largest integer not greater than x

Mathematical functions (cont)

`trunc(x)`	Integer formed by truncating values in `x` towards 0
`round(x, digits=n)`	Round `x` to the specified number of decimal places
`signif(x, digits=n)`	Round `x` to the specified number of significant digits
`cos(x)`, `sin(x)`, `tan(x)`	Cosine, sine, and tangent
`acos(x)`, `asin(x)`, `atan(x)`	Arc-cosine, arc-sine and arc-tangent
`cosh(x)`, `sinh(x)`, `tanh(x)`	Hyperbolic cosine, sine, and tangent
`acosh(x)`, `asinh(x)`, `atanh(x)`	Hyperbolic arc-cosine, arc-sine, and arc-tangent
`log(x, base=n)`	Logarithm of `x` to the base `n`
`log(x)`	Natural logarithm
`log10(x)`	Common logarithm

Mathematical functions (cont)

`exp(x)`	Exponential function

Statistical functions

`mean(x)`	Mean
`median(x)`	Median
`sd(x)`	Standard deviation
`var(x)`	Variance
`mad(x)`	Mean absolute deviation
`quantile(x, probs)`	Quantiles where `x` is the numeric vector of quantiles and `probs` is a numeric vector with probabilities in [0,1]
	`y <- quantile(x, c(.3,.84))`
`range(x)`	Range
	`diff(range(x))` returns difference between extreme values
`sum(x)`	Sum

By **xeonkai**
cheatography.com/xeonkai/

Not published yet.
Last updated 8th November, 2016.

Statistical functions (cont)

`diff(x, lag=n)`	Lagged differences, with `lag` indicating which lag to use. Default lag is 1.
`min(x)`	Minimum
`max(x)`	Maximum
`scale(x, center=TRUE, scale=TRUE)`	Column center (`center=TRUE`) or standardize (`center=TRUE, scale=TRUE`) data object `x`, i.e. to a mean of 0 and std of 1

Trimmed mean - dropping top and lowest 5% and missing values

```
y <- mean(x, trim=0.05, na.rm=TRUE)
```

Probability functions

`beta`	Beta
`binom`	Binomial
`cauchy`	Cauchy
`chisq`	Chi-squared (noncentral)
`exp`	Exponential
`f`	F
`gamma`	Gamma

Probability functions (cont)

`geom`	Geometric
`hyper`	Hypergeometric
`lnorm`	Lognormal
`logis`	Logistic
`multinom`	Multinomial
`nbinom`	Negative binomial
`norm`	Normal
`pois`	Poisson
`signrank`	Wilcoxon Signed Rank
`t`	T
`unif`	Uniform
`weibull`	Weibull
`wilcox`	Wilcoxon Rank Sum

General form of probability function:

```
[dpqr]distribution_abbreviation()
```

`d` = density

`p` = distribution function

`q` = quantile function

`r` = random generation (random deviates)

Character functions

`nchar(x)`	Counts the no. of characters of `x`
`substr(x, start, stop)`	Extract or replace substrings in a character vector
	`x <- "abcdef"`
	`substr(x, 2, 4)` returns `"bcd"`
	`substr(x, 2, 4) <- "22222"` produces `"a222ef`
`grep(pattern, x, ignore.case=FALSE, fixed=FALSE)`	Search for pattern in `x`. `fixed=FALSE` - `pattern` is regex. `fixed=TRUE` - `pattern` is text string. Returns matching indices.
`sub(pattern, replacement, x, ignore.case=FALSE, fixed=FALSE)`	Find `pattern` in `x` and substitute with `replacement` text
`strsplit(x, split, fixed=FALSE)`	Split the elements of `x` at `split`
	`y <- strsplit("abc", "")` returns 1-component, 3-element list containing `"a" "b" "c"`.

Cheatography

Data management in R Cheat Sheet

by xeonkai via cheatography.com/31513/cs/9676/

Character functions (cont)

	`unlist(y)[2]` and `sapply(y, "[", 2)` both return "b".
`paste(..., sep="")`	Concatenate strings after using `sep` string to separate them
	`paste("x", 1:3, sep="M")` returns `c("xM1","xM2","xM3")`
`toupper(x)`	Uppercase
`tolower(x)`	Lowercase

Other useful functions

`length(x)`	Length of object x`
`seq(from, to, by)`	Generate a sequence
`rep(x, n)`	Repeat `x` `n` times
`cut(x, n)`	Divide continuous variable `x` into factor with `n` levels. `ordered_result = TRUE` creates an ordered factor.

Other useful functions (cont)

`pretty(x, n)`	Create pretty breakpoints. Divide a continuous variable `x` into `n` intervals, by selecting `n+1` equally spaced rounded values. Often used in plotting.
`cat(..., file="myfile", append=FALSE)`	Concatenates the objects in ... and outputs them to the screen or to a file
`apply(x, MARGIN, FUN, ...)`	Apply function to data objects

Escape characters:
`\n` - new lines
`\t` - tabs
`\'` - single quote
`\b` - backspace

Control flow

FOR	`for (var in seq) statement`
WHILE	`while (cond) statement`
IF-ELSE	`if (cond) statement`

Control flow (cont)

	`if (cond) statement1 else statement2`
IFELSE	`ifelse(cond, statement1, statement2)`
SWITCH	`switch(expr, ...)`

`statement` - single R statement or compound statement enclosed in {} and separated by ;
`cond` - expression that resolves to `TRUE` or `FALSE`
`expr` - statement that evaluates to number or character string
`seq` - sequence of numbers or character strings

Example for switch

```
feelings <- c("sad","afraid")
for (i in feelings)
  print(
    switch(i,
         happy = "I am glad you are happy",
         afraid = "There is nothing to fear",
         sad = "Cheer up",
         angry = "Calm down now"
            )
        )
```

By **xeonkai**
cheatography.com/xeonkai/

Not published yet.
Last updated 8th November, 2016.

Aggregation and restructuring

`t()`	Transpose
`aggregate(x, by, FUN)`	Aggregate (`by` variables must be a list)

```
new <- aggregate(mtcars,
by=list(Group.cyl=cyl,
Group.gears=gear), FUN=mean,
na.rm=TRUE)
```

Reshape package