DATE and TIME CHEATSHEET

(R Programming Language)

as.Date()	Used to get the date format when the date is in character, numeric, POSIXlt, and POSIXct formats
	calculates date as number of days since 1970-01-01 (negative values for earlier dates)
	Doesn't store any time information
	Standard output format : %Y-%m-%d
	Syntax: as.Date(x, format, tryFormats, origin, tz, …)
POSIXct format (Portable Operating System Interface calendar time)	Handle time data along with date data
	Stores date time data as a single value in units of seconds since the time 00:00:00 UTC on Jan 1, 1970 (unix epoch)
	Standard output format : %Y-%m-%d %H:%M:%S
	Syntax: as.POSIXct(x, format, tz, origin, …)
	Example: timeDate <- as.POSIXct("2019-10-27 10:15") timeDate: "2019-10-27 10:15:00 EDT" unclass(timeDate): 1572185700
POSIXlt format (Portable Operating System Interface local time)	Similar to POSIXct class but stores date and time attributes separately in a list
	It is a vector, and hence can be used to extract specific aspects of a time (such as the day of the week)
	Syntax: as.POSIXlt(x, format, tz, origin, …)
	Example: timeDate <- as.POSIXlt("2019-10-27 10:15") timeDate: "2019-10-27 10:15:00 EDT" unclass(timeDate): $sec: 0 (seconds) $min: 15 (minutes) $hour: 10 (hours) $mday: 27 (day of month (1-31) $mon: 9 (month of the year (0-11) $year: 119 (years since 1900) $wday: 0 (day of the week (0-6 where 0 represents Sunday)) $yday: 299 (day of the year (0-365)) $isdst: 1(Daylight savings indicator, positive if it is daylight savings) $zone: "EDT" $gmtoff: NA
parse_datetime()	Similar to as.Date but support fewer datetime formats
	The dates are calculated as number of days since 1970-01-01 instead of seconds
	Returns a POSIXct vector with timezone attribute
	Elements that couldnot be parsed are returned as NA
	Syntax: parse_datetime(x, format, na = c("", "NA"), locale , trim_ws) **Note:** The format specification should match the entire string **Supported Formats:** Year - "%Y", "%y" Month - "%m", "%b", "%B" Day - "%d" Hour - "%H", "%I" Minutes - "%M" Seconds - "%S" Time zone - "%Z", "%z" AM/PM indicator - "%p"

Strptime	Convert the character string to date time format Character input is first converted to class "POSIXlt" and numeric input is first converted to "POSIXct" by strptime
	Syntax: strptime(x, format, tz = "")
	Example: strptime("2017-02-09", format="%Y-%d-%m", tz = "UTC") Output: "2017-09-02 UTC"
strftime	Convert time data type to a string
	Syntax: strftime(x, format, usetz = FALSE, ...)
	The default format for both is "%Y-%m-%d %H:%M:%S"

Standard Date/Time

Code	Meaning	Code	Meaning
%a	Abbreviated weekday	%A	Full weekday
%b	Abbreviated month	%B	Full month
%c	Locale-specific date and time	%d	Day of the month (decimal number)
%H	Decimal hours (00-24 hour)	%I	Decimal hours (01-12 hour)
%j	Decimal day of the year	%m	Decimal month
%M	Decimal minute (00-59)	%p	Locale-specific AM/PM. Used with %I and not with %H
%S	Decimal second (00–61)	%U	Decimal week of the year (starting on Sunday)
%w	Decimal Weekday (0=Sunday)	%W	Decimal week of the year (starting on Monday)
%x	Locale-specific Date	%X	Locale-specific Time
%y	2-digit year	%Y	4-digit year
%z	Offset from GMT e.g. +0800	%Z	Time zone (character e.g. "America/Chicago")

Example date formats:

%m/%d/%y – 10/27/19
%B %d %Y – October 27 2019
%Y-%m-%d - 2016-01-29
%b %d, %Y – Oct 27, 2019
%A, %B %d, %Y - Sunday, October 27, 2019

Standard Notations:

x: Object to be converted

Format: Format x is currently stored in. It's character string

tryFormats: Character vector of format strings. Provided if format is not specified

origin: Origin date from where to calculate the number of days if x is a numeric value

tz: Time zone name. By default, data is the stored in the local time zone

Usetz: Logical. Mention if the timezone should be appended to the output

For parse_datetime()

Na: Vector of strings that should be interpreted as missing values

Locale: To specify default time zones, day/month names etc. The default_locale is UTC

Trim_ws : True if the trailing and leading whitespaces needs to be trimmed

Date formatting:

Use format function to change the format of the date from the standard %Y-%m-%d. Returns the output in character format

Example: z = as.Date("2019-10-29")

	Input	Output
Change the date format	format(z,"%a %b %d")	"Tue Oct 29"
extract month and day	format(z,"%b %d")	"Oct 29"
extract year value	as.numeric(format(z, "%Y")	2019
First day of the month	as.Date(format(z, "%Y-%m-01"))	"2019-10-01"
change year value to 2020 (z)	as.POSIXct(format(date, "2020-%m-%d"))	"2020-10-29 EDT"
Day of week	as.numeric(format(z,"%w"))	2 (# Sun = 0)
Day of year	as.numeric(format(z,"%j"))	302

Date Parsing

	Example	as.Date	as.POSIXct
When date is character class	z = "2019-10-27"	as.Date(z)	as.POSIXct(z)
	z = "10/27/2019"	as.Date(z, "%m/%d/%Y")	as.POSIXct(z, format= "%m/%d/%Y") or as.POSIXct(strptime(z, "%m/%d/%Y"))
	z = 27102019	as.Date(as.character(z),format = "%d%m%Y")	as.POSIXct(as.character(z), format = "%d%m%Y")
When date is in number of days	z= 18196	as.Date(z, origin = "1970-01-01")	(as.POSIXct(z * 86400, origin = "1970-01-01"))
When date is in number of seconds	Z = 1572188800	as.Date(z / 86400, origin = "1970-01-01")	(as.POSIXct(z, origin = "1970-01-01"))

Note 1: Specify the origin date from where to start counting days or seconds.
For example: For numeric data imported from excel we might need the origin date that Excel starts counting from (In Windows it is December 30, 1899 and in Mac the origin date is January 1, 1904)

Note2: For POSIXct, time zone and origin both while converting ‘z’ into date time format. By default, it considers the local time zone
Example: z = 1572188800
(as.POSIXct(z, origin = "1970-01-01")) Output: "2019-10-27 11:06:40 EDT"
(as.POSIXct(z, "UTC", origin = "1970-01-01")) Output: "2019-10-27 15:06:40 UTC"

Date/Duration comparison or Difference:

	Input	Output
Compare two dates	as.Date("2019-10-02") > as.Date("2019-10-29")	False
Number of days between dates	as.Date("2019-10-02") - as.Date("2019-10-29")	Time difference of -27 days

Time difference between 2 dates	z = as.Date("2019-10-20")
	difftime(Sys.Date(), z)
		Time difference of 7 days

	difftime(Sys.time(), start)	Time difference of 7.661039 days
	difftime(Sys.time(), start, units = “hours”)	Time difference of 183.9032 hours

Note: By default, difftime gives the output in days. Specify the “units” argument to get the time difference in other formats

Other functions:

	as.Date	as.POSIXct
Next Day	date + 1	seq(date, length = 2, by = "day")[2]
Previous Day	date – 1	seq(date, length = 2, by = "-1 day")[2]
x days since date (1 day is 24 hrs)	date + x	seq(date, length = 2, by = paste(x, "day"))[2]
display date in new time zone (TZ)		as.POSIXct(format(as.POSIXct(date), tz = "TZ"), tz = "TZ")
Sequence of 5 dates	seq(date, length = 5, by = "day")	seq(date, length = 5, by = "day")
Sequence of 5 dates every 1st week	seq(date, length = 5, by = "1 week")	seq(date, length = 5, by = "1 week")
change year value to z	as.Date(format(date, "z-%m-%d"))	as.POSIXct(format(date, "z-%m-%d"))

BASE R

sys.date()	returns the current day in the current time zone as an object of class POSIXct.
sys.time()	returns absolute data-time value (can be converted to various time zones) as an object of class Date.
sys.timezone()	returns current time zone as a character string.

difftime(b2,b1,units)	b1, b2 – dates, units: require output in what units? (‘week’, ‘days’, ‘hours’, ‘mins’, ‘secs’)
seq()	seq(as.Date('1976-7-4'),by='days',length=10)
mean(DATES)	Output the mean of the DATES vector
max(MAX)	Output the maximum of the DATES vector
min(DATES)	Output the minimum of the DATES vector

plot(df$DATE, <df-column>, type = "l")	y-axis is on the order of years.
plot(df$DATE[1:30], <df-column>, type = "l")	switch x-axis to months (from years)

df = dataframe, <df-column> = some data column/list of entries

weekdays(x, abbreviate)	return a character vector of name
months(x, abbreviate)	return a character vector of name
quarters(x, abbreviate)	returns a character vector of “Q1” to “Q4” that represent quarter of an year.
julian(x, origin = as.POSIXct(“1970-01-01”, tz=”GMT”), …)	returns the number of days since the origin in POSIXct format.
julian(x, origin = as.Date(“1970-01-01”), …)	returns the number of days since the origin in DATE format.

x = an object inheriting from class “POSIXt” or “Date”

Abbreviate = logical vector. Should the names be abbreviated?

All time calculations are done ignoring leap-seconds

LUBRIDATE

NOTE: Few cheat sheets are already available for lubridate. Here we included the most commonly used functions only.
library(lubridate)

as_datetime(S)	S = seconds, Date-time is stored as seconds since 1970-01-01 00:00:00 UTC
as_date(D)	D = days, A date is a day stored as the number of days since 1970-01-01
t <- hms::as.hms(S)	S = seconds, An hms is a time stored as the number of seconds since 00:00:00

now(tzone = "")	Current time in tz (defaults to system tz). now()
today(tzone = "")	Current date in a tz (defaults to system tz). today()
fast_strptime()	Faster strptime. fast_strptime('9/1/01', '%y/%m/%d')
parse_date_time()	Easier strptime. parse_date_time("9/1/01", "ymd")

PARSE DATE-TIMES (Convert strings or numbers to date-times)
1. Identify the order of the year (y), month (m), day (d), hour (h), minute (m) and second (s) elements in your data.
2. Use the function below whose name replicates the order. Each accepts a wide variety of input formats.

ymd_hms(), ymd_hm(), ymd_h()	ymd_hms("2017-11-28T14:02:00")
mdy_hms(), mdy_hm(), mdy_h()	mdy_hms("11/28/2017 1:02:03")
dmy_hms(), dmy_hm(), dmy_h()	dmy_hms("1 Jan 2017 23:59:59")
mdy(), myd()	mdy("July 4th, 2000")

ROUND DATE-TIMES

floor_date(x, unit)	Round down to nearest unit.
round_date(x, unit)	Round to nearest unit.
ceiling_date(x, unit, change_on_boundary = NULL)	Round up to nearest unit.
rollback(dates, roll_to_first = FALSE, preserve_hms = TRUE)	Roll back to last day of previous month. rollback(dt)

x = date, unit = hour/month/..

GET AND SET COMPONENTS

Use an accessor function to get a component. Assign into an accessor function to change a component in place.

d ## "2019-10-30"
day(d) ## 30
day(d) <- 1
d ## "2019-10-01"

2019-10-30 11:59:59	date(x)	Date component
2019-10-30 11:59:59	year(x)	Year
2019-**10**-30 11:59:59	month(x, label, abbr)	Month
2019-10-**30** 11:59:59	day(x)	Day
2019-10-30 **11**:59:59	hour(x)	Hour
2019-10-30 11:**59**:59	minute(x)	Minutes
2019-10-30 11:59:**59**	second(x)	Seconds
week(x)	Week of the year. (nth week of the year)	
am(x)	Is it in the am?	
pm(x)	Is it in the pm?	
leap_year(x)	Is it a leap year?	
update(x, ..., simple = FALSE)	Updates the date x. update(dt, day = 2, hour = 1)	

STAMP DATE-TIMES

stamp() Derive a template from an example string and return a new function that will apply the template to date-times. Also stamp_date() and stamp_time().
1. Derive a template, create a function sf <- stamp("Created Sunday, Jan 17, 1999 3:34")
2. Apply the template to dates sf(ymd("2010-04-05"))
## [1] "Created Monday, Apr 05, 2010 00:00"
Tip: use a date with day > 12

GGPLOT

g <- ggplot(df, **aes(DATE, <column>)**) + geom_line()	plot with date on x axis

LABELS AND ANNOTATIONS

g + **geom_vline**(xintercept = ymd("k"), color) + **annotate**("text", x = ymd("k"), y = 3.75, label, color, hjust = 0)	To mark specific events in a time series, use geom_vline, annotate. #k=yyyy-mm-dd

LIMITS and BREAKS

ggplot(df %>% **filter(DATE < as.Date("K")**), aes(DATE, <column>))
+ geom_line()
	limit the DATE to be less than some K (yyyy-mm-dd format). x-axis labels might switch from years to months.

Using Lubridate

library(lubridate) g + **scale_x_date(limits = c(ymd("k1"), ymd("k2")))**	k1, k2 : yyyy-mm-dd
g + **scale_x_date(date_breaks = "k years")**	breaks on the x-axis, each of difference of k years
g + **scale_x_date(date_labels = "%Y-%m")** #yyyy-mm	to alter the x-axis label representation. The format can be any like “%b / %Y”.

scale_x_time() is similar to scale_x_date()