lubridate

Making dates easier

lubridate::overview

- Dates notoriously are one of the toughest data types to use when starting a new programming language

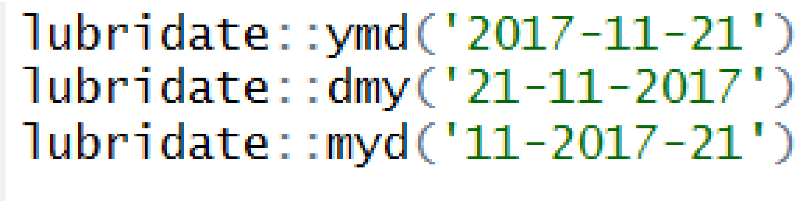
- Luckily, lubridate makes this painful process much easier!

- Content
  - Parsing functions
  - Component functions
  - Rounding
  - Date differences
  - Extracting time from timestamp

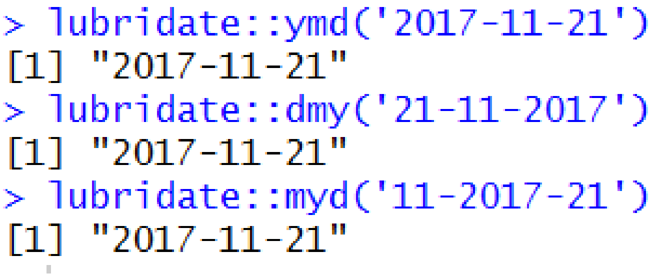
lubridate::parsing functions

- Lubridate can convert nearly any character, factor, or numeric variable to a date by using a parsing function

- The key to using these functions is correctly specifying the order in which the year, month, day appear in the variable of interest



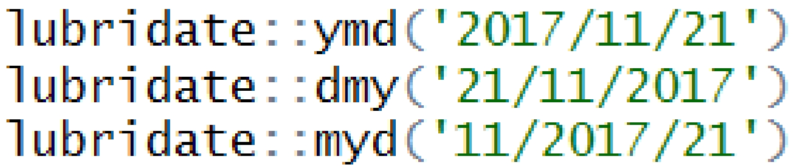
```
lubridate::ymd('2017-11-21')
lubridate::dmy('21-11-2017')
lubridate::myd('11-2017-21')
```



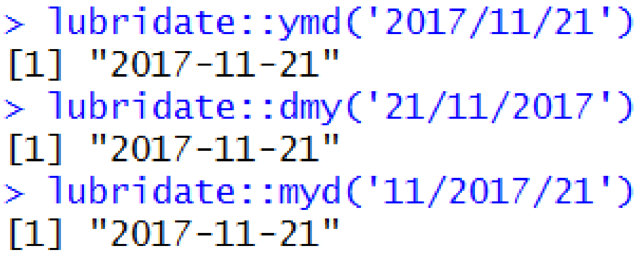
```
> lubridate::ymd('2017-11-21')
[1] "2017-11-21"
> lubridate::dmy('21-11-2017')
[1] "2017-11-21"
> lubridate::myd('11-2017-21')
[1] "2017-11-21"
```

lubridate::parsing functions

- These same functions can be used regardless of what special character is used to separate year, month, and day in the variable of interest:



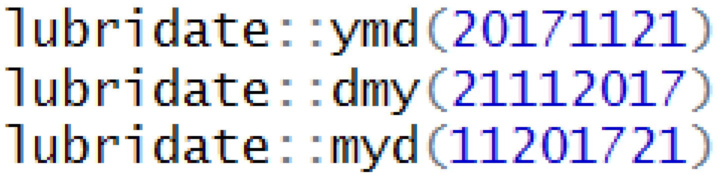
```
lubridate::ymd('2017/11/21')
lubridate::dmy('21/11/2017')
lubridate::myd('11/2017/21')
```



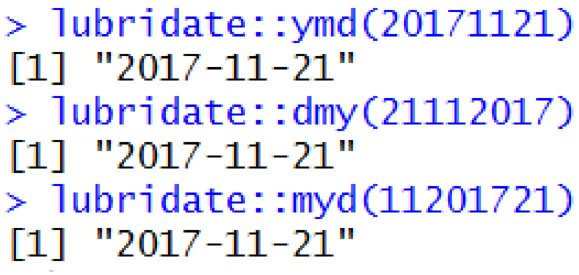
```
> lubridate::ymd('2017/11/21')
[1] "2017-11-21"
> lubridate::dmy('21/11/2017')
[1] "2017-11-21"
> lubridate::myd('11/2017/21')
[1] "2017-11-21"
```

lubridate::parsing functions

- These same functions can also be used on numeric variables that have no special character separators:



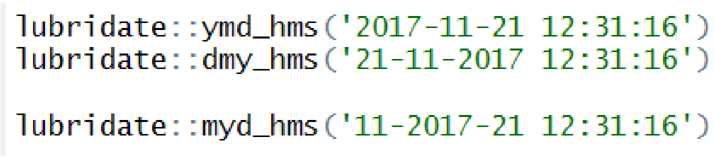
```
lubridate::ymd(20171121)
lubridate::dmy(21112017)
lubridate::myd(11201721)
```



```
> lubridate::ymd(20171121)
[1] "2017-11-21"
> lubridate::dmy(21112017)
[1] "2017-11-21"
> lubridate::myd(11201721)
[1] "2017-11-21"
```

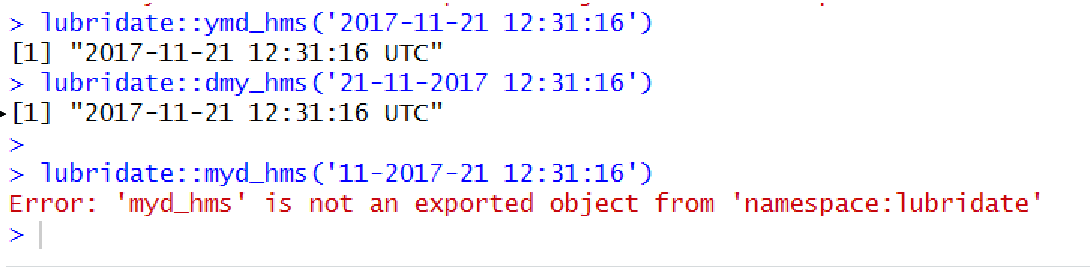
lubridate::parsing functions

- *Nearly* all of these same functions can also be used on variables that have times associated with them:



```
lubridate::ymd_hms('2017-11-21 12:31:16')
lubridate::dmy_hms('21-11-2017 12:31:16')

lubridate::myd_hms('11-2017-21 12:31:16')
```

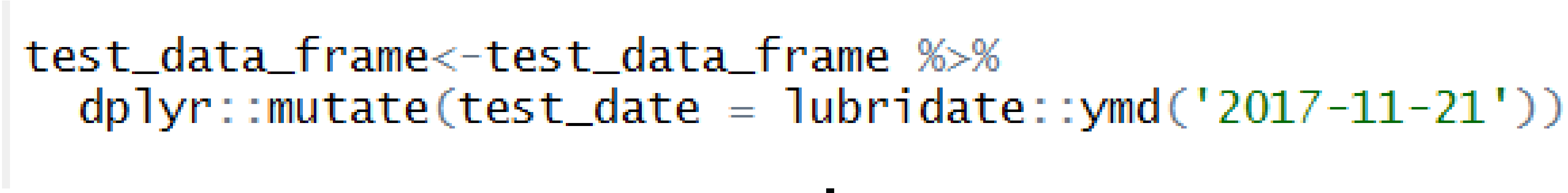


```
> lubridate::ymd_hms('2017-11-21 12:31:16')
[1] "2017-11-21 12:31:16 UTC"
> lubridate::dmy_hms('21-11-2017 12:31:16')
[1] "2017-11-21 12:31:16 UTC"
>
> lubridate::myd_hms('11-2017-21 12:31:16')
Error: 'myd_hms' is not an exported object from 'namespace:lubridate'
> |
```

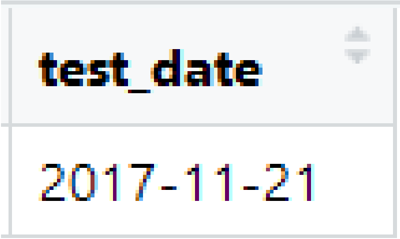
*please see complete list of lubridate functions in lubridate cheatsheet: https://www.rstudio.com/resources/cheatsheets/

lubridate::parsing functions into new variables

- You can use lubridate to create new variables within a dataframe by combining lubridate with dplyr’s “mutate” function
  - Note: The input to lubridate can also be a variable within your dataframe



```
test_data_frame<-test_data_frame %>%
  dplyr::mutate(test_date = lubridate::ymd('2017-11-21'))
```



test_date
2017-11-21

lubridate::exercise 1

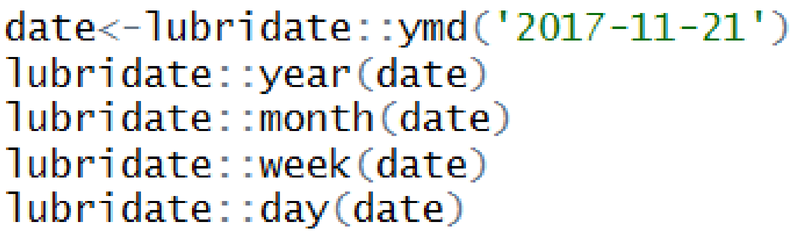
In the blood culture dataset, perform the following operations:

1. Convert BLD_CX_DT from a factor to a date variable type
2. Convert DOB from a factor to a date variable type

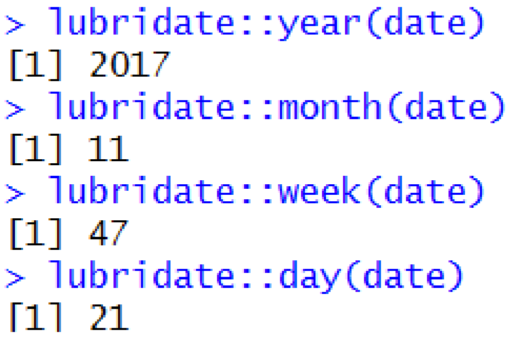
lubridate::component functions

- lubridate also allows you to extract certain components from a date/timestamp variable, such as the year, month, week, or day

- This is particularly useful if you would like to group your data at one of these date levels for analysis or visualization purposes



```
date<-lubridate::ymd('2017-11-21')
lubridate::year(date)
lubridate::month(date)
lubridate::week(date)
lubridate::day(date)
```



```
> lubridate::year(date)
[1] 2017
> lubridate::month(date)
[1] 11
> lubridate::week(date)
[1] 47
> lubridate::day(date)
[1] 21
```

lubridate::exercise.2

In the blood culture dataset, perform the following operations:

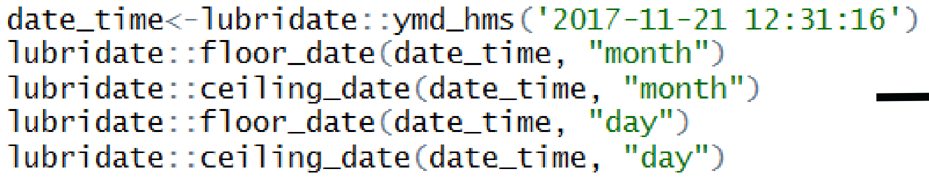
1. Extract the month from the BLD_CX_DT date field you created in exercise #1

2. Create a data frame that counts the number of unique blood cultures each calendar month (regardless of year)

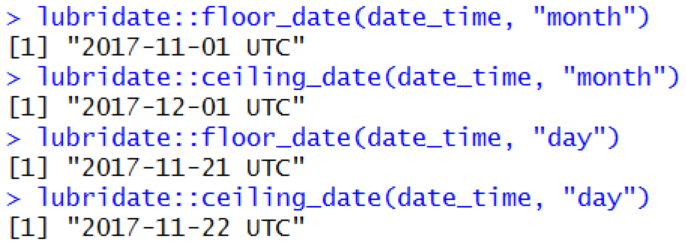
lubridate::rounding

- Similar to SQL, lubridate has a “floor_” and “ceiling_” commands that allow you to round the date either up or down, respectively, to the closest date unit

- Although there are a number of possible date units, the most relevant date units for our work will be “month” and “day”



```
date_time<-lubridate::ymd_hms('2017-11-21 12:31:16')
lubridate::floor_date(date_time, "month")
lubridate::ceiling_date(date_time, "month")
lubridate::floor_date(date_time, "day")
lubridate::ceiling_date(date_time, "day")
```



```
> lubridate::floor_date(date_time, "month")
[1] "2017-11-01 UTC"
> lubridate::ceiling_date(date_time, "month")
[1] "2017-12-01 UTC"
> lubridate::floor_date(date_time, "day")
[1] "2017-11-21 UTC"
> lubridate::ceiling_date(date_time, "day")
[1] "2017-11-22 UTC"
```

lubridate::exercise.3

To the blood culture dataset, perform the following operations:

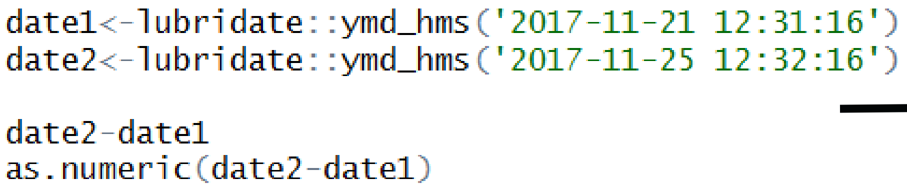
1. Create a variable that is the first day of the month shown in BLD_CX_DT

2. Create a dataframe that counts the number of blood cultures each month-year

date differences

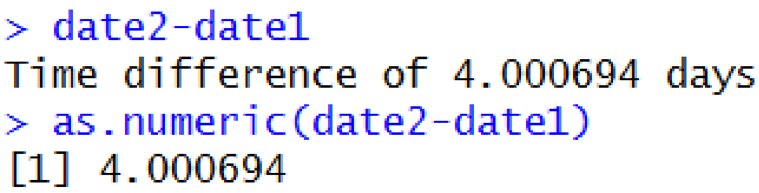
- When subtracting one date from another (i.e. calculating time between), R defaults to returning the value in text format

- To return the format as a number (in days), be sure to include a “as.numeric” statement



```
date1<-lubridate::ymd_hms('2017-11-21 12:31:16')
date2<-lubridate::ymd_hms('2017-11-25 12:32:16')

date2-date1
as.numeric(date2-date1)
```



```
> date2-date1
Time difference of 4.000694 days
> as.numeric(date2-date1)
[1] 4.000694
```

lubridate::exercise.4

To the blood culture dataset, perform the following operation:

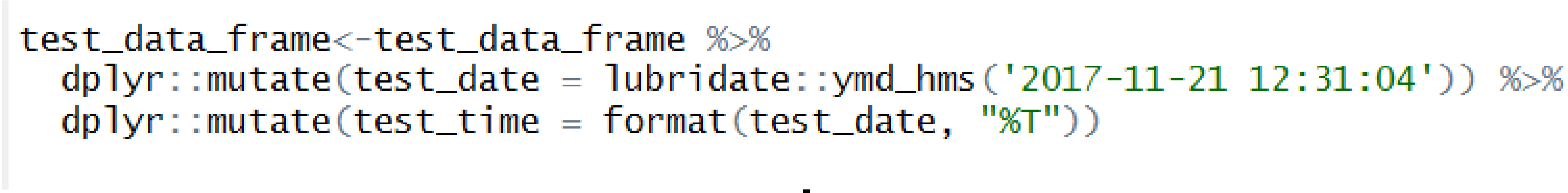
1. Create a variable that calculates the days between birth and blood culture

2. Determine the mean and median days between birth and blood culture

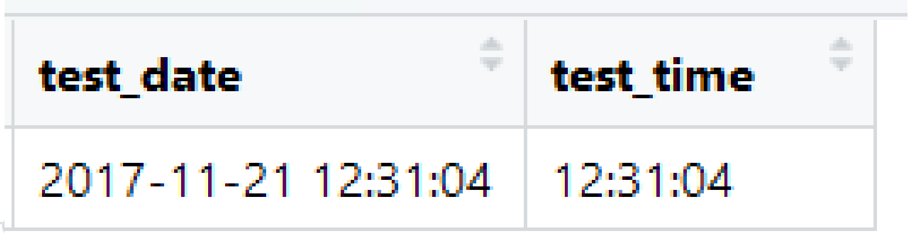
extracting time only

- Although lubridate can perform nearly all date related operations, it is not great at extracting time from a timestamp and storing the output as a time variable

- The ‘format’ function from baseR, however, accomplishes this easily



```
test_data_frame<-test_data_frame %>%
  dplyr::mutate(test_date = lubridate::ymd_hms('2017-11-21 12:31:04')) %>%
  dplyr::mutate(test_time = format(test_date, "%T"))
```



test_date	test_time
2017-11-21 12:31:04	12:31:04

lubridate::exercise.5

To the blood culture dataset, perform the following operation:

1. Use dplyr and format to create an indicator that describes whether or not the blood culture was taken after 12 PM