Lecture 20: Dates and Times

STAT 450, Fall 2021

Reading: Chapter 16 from *R for Data Science*

Useful reference: https://lubridate.tidyverse.org/

`lubridate` is an R package that provides easy-to-use functions for dealing with date-time data in R. It's not part of the core tidyverse so you'll first need to install the package, and then use `library()` to load it into your R session.

```
library(tidyverse)
library(lubridate)
```

Parsing Dates

`lubridate` contains helper functions that convert character vectors (strings) into date objects in R. To use them, identify the order of the year (**y**), month (**m**), and day (**d**). For example:

```
ymd("2020-11-15")
```

```
## [1] "2020-11-15"
```

```
mdy("November 15, 2020")
```

```
## [1] "2020-11-15"
```

```
mdy("11/15/2020")
```

```
## [1] "2020-11-15"
```

Dates in R are represented with the `Date` class. For example:

```
date1 <- "2020-11-15"
class(date1)
```

```
## [1] "character"
```

```
date1 <- ymd("2020-11-15")
class(date1)
```

```
## [1] "Date"
```

Parsing Date-Times

```
ymd_hms("2020-11-15 19:21:22")
```

```
## [1] "2020-11-15 19:21:22 UTC"
```

```
mdy_hms("11/15/2020 19:21:22")
```

```
## [1] "2020-11-15 19:21:22 UTC"
```

Date-time objects in R are represented with the POSIXct class:

```
t1 <- "2020-11-15 19:21:22"
class(t1)
```

```
## [1] "character"
```

```
t1 <- ymd_hms("2020-11-15 19:21:22")
class(t1)
```

```
## [1] "POSIXct" "POSIXt"
```

The class name, which is somewhat cryptic, comes from Unix. POSIX (pronounced poz-icks) is an acronym that stands for "Portable Operating System Interface [for Unix]," which refers to a set of standards for the Unix operating system. The ct in POSIXct stands for calendar time.

Fun fact: internally, date-times are stored as the number of seconds since the so-called Unix epoch on January 1, 1970:

```
now()
```

```
## [1] "2021-11-14 19:05:48 PST"
```

```
as.numeric(now()) # number of seconds since January 1, 1970
```

```
## [1] 1636945549
```

Extracting Components

```
t1 <- ymd_hms("2020-11-15 19:21:22")
year(t1)
```

```
## [1] 2020
```

```
month(t1)
```

```
## [1] 11
```

```
month(t1, label = TRUE)
```

```
## [1] Nov
## 12 Levels: Jan < Feb < Mar < Apr < May < Jun < Jul < Aug < Sep < ... < Dec
```

```
mday(t1)
```

```
## [1] 15
```

```
wday(t1, label = TRUE)
```

```
## [1] Sun
## Levels: Sun < Mon < Tue < Wed < Thu < Fri < Sat
```

```
hour(t1)
```

```
## [1] 19
```

Exercise: Use the appropriate `lubridate` function to parse each of the following dates (i.e., convert from a character vector to a date or date-time object in R):

```
t1 <- "January 1, 2010"
t2 <- "2015-Mar-07"
t3 <- "06-Jun-2017"
t4 <- c("11/14/2020", "11/15/2020")
t5 <- c("11/14/2020 11:30:00", "11/14/2020 12:30:00")
t6 <- c("11/14/2020 1:30:00 AM", "11/14/2020 1:30:00 PM")
```

San Francisco Crime Data

To demonstrate working with date-times in R we'll use a data set on crimes that occurred in San Francisco in 2018. The data was obtained from
https://data.sfgov.org/Public-Safety/Police-Department-Incident-Reports-2018-to-Present/wg3w-h783

The original data set contains 26 columns, but we'll only work with 2 of those columns: the date-time when the crime incident occurred, and the type of crime.

```
sfcrimes <- readRDS(url("https://ericwfox.github.io/data/sfcrimes.rds"))
```

```
sfcrimes
```

```
## # A tibble: 153,520 x 2
##    date_time              type                                 
##    <chr>                  <chr>                                
##  1 2018/10/05 04:15:00 PM Other Offenses                       
##  2 2018/10/05 07:13:00 PM Offences Against The Family And Children
##  3 2018/10/05 07:13:00 PM Disorderly Conduct                   
##  4 2018/10/05 06:33:00 PM Other Miscellaneous                  
##  5 2018/10/05 06:33:00 PM Warrant                              
##  6 2018/10/05 06:33:00 PM Traffic Violation Arrest             
##  7 2018/10/05 04:36:00 PM Traffic Violation Arrest             
##  8 2018/10/05 04:36:00 PM Other Miscellaneous                  
##  9 2018/10/05 05:14:00 PM Traffic Violation Arrest             
## 10 2018/10/05 05:14:00 PM Other Miscellaneous                  
## # ... with 153,510 more rows
```

```
crime_tb <- sort(table(sfcrimes$type), decreasing = TRUE)
crime_tb[1:10]
```

```
## 
##       Larceny Theft Other Miscellaneous        Non-Criminal             Assault 
##               48789               11785                9622                9043 
##  Malicious Mischief            Burglary       Lost Property             Warrant 
##                8864                7103                5778                5579 
## Motor Vehicle Theft               Fraud 
##                5289                4659
```

```
# subset burglaries
sfcrimes2 <- filter(sfcrimes, type == "Burglary")
sfcrimes2
```

```
## # A tibble: 7,103 x 2
##    date_time              type    
##    <chr>                  <chr>   
##  1 2018/10/06 05:42:00 AM Burglary
##  2 2018/10/06 08:45:00 PM Burglary
##  3 2018/10/06 08:45:00 PM Burglary
##  4 2018/10/08 04:25:00 AM Burglary
##  5 2018/10/06 05:00:00 PM Burglary
##  6 2018/10/07 04:08:00 PM Burglary
##  7 2018/10/09 07:27:00 AM Burglary
##  8 2018/10/09 10:25:00 AM Burglary
##  9 2018/09/05 07:40:00 AM Burglary
## 10 2018/10/08 07:30:00 PM Burglary
## # ... with 7,093 more rows
```

```r
# parse date-times of burglaries
t <- ymd_hms(sfcrimes2$date_time, tz = "America/Los_Angeles")
t[1:10]
```

```
##  [1] "2018-10-06 05:42:00 PDT" "2018-10-06 20:45:00 PDT"
##  [3] "2018-10-06 20:45:00 PDT" "2018-10-08 04:25:00 PDT"
##  [5] "2018-10-06 17:00:00 PDT" "2018-10-07 16:08:00 PDT"
##  [7] "2018-10-09 07:27:00 PDT" "2018-10-09 10:25:00 PDT"
##  [9] "2018-09-05 07:40:00 PDT" "2018-10-08 19:30:00 PDT"
```

```r
class(t)
```

```
## [1] "POSIXct" "POSIXt"
```

```r
# get local time zone
Sys.timezone()
```

```
## [1] "America/Los_Angeles"
```

Time zone reference: https://en.wikipedia.org/wiki/List_of_tz_database_time_zones

Extracting Components

```r
# hour of the day
table(hour(t))
```

```
## 
##   0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19 
## 361 225 277 321 325 298 174 207 246 271 288 243 332 262 269 310 348 415 404 379 
##  20  21  22  23 
## 364 259 294 231
```

```r
# day of the week
table(wday(t, label = T))
```

```
## 
##  Sun  Mon  Tue  Wed  Thu  Fri  Sat 
##  865 1055  991 1077 1020 1167  928
```

```r
# month
table(month(t, label = T))
```

```
## 
## Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec 
## 618 550 592 596 649 578 638 707 594 555 499 527
```

```r
# extract date
head(date(t))
```

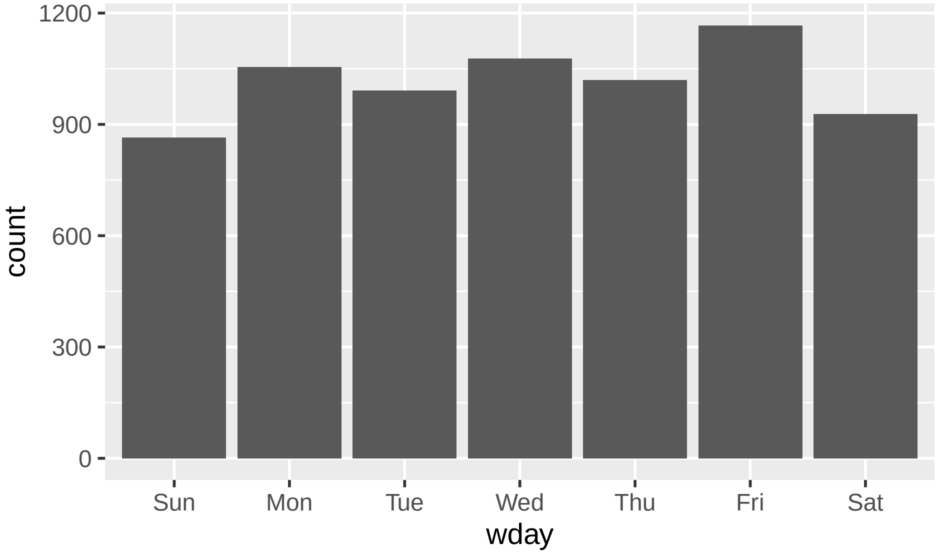
```
## [1] "2018-10-06" "2018-10-06" "2018-10-06" "2018-10-08" "2018-10-06"
## [6] "2018-10-07"
```

```
# make new tibble with different date-time components
sfcrimes3 <- tibble(
  date = date(t),
  month = month(t, label = T),
  wday = wday(t, label = T),
  hour = hour(t)
)
sfcrimes3
```

```
## # A tibble: 7,103 x 4
##    date       month wday   hour
##    <date>     <ord> <ord> <int>
##  1 2018-10-06 Oct   Sat       5
##  2 2018-10-06 Oct   Sat      20
##  3 2018-10-06 Oct   Sat      20
##  4 2018-10-08 Oct   Mon       4
##  5 2018-10-06 Oct   Sat      17
##  6 2018-10-07 Oct   Sun      16
##  7 2018-10-09 Oct   Tue       7
##  8 2018-10-09 Oct   Tue      10
##  9 2018-09-05 Sep   Wed       7
## 10 2018-10-08 Oct   Mon      19
## # ... with 7,093 more rows
```

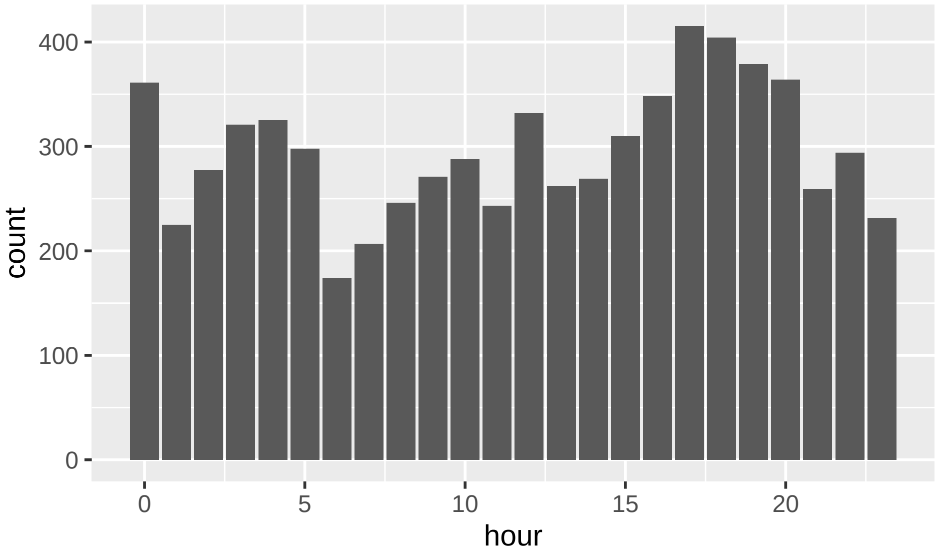
Bar plot of number of burglaries that occurred each day of the week:

```
ggplot(sfcrimes3, aes(wday)) + geom_bar()
```



Bar plot of number of burglaries that occurred each hour of the day:

```
ggplot(sfcrimes3, aes(hour)) + geom_bar()
```



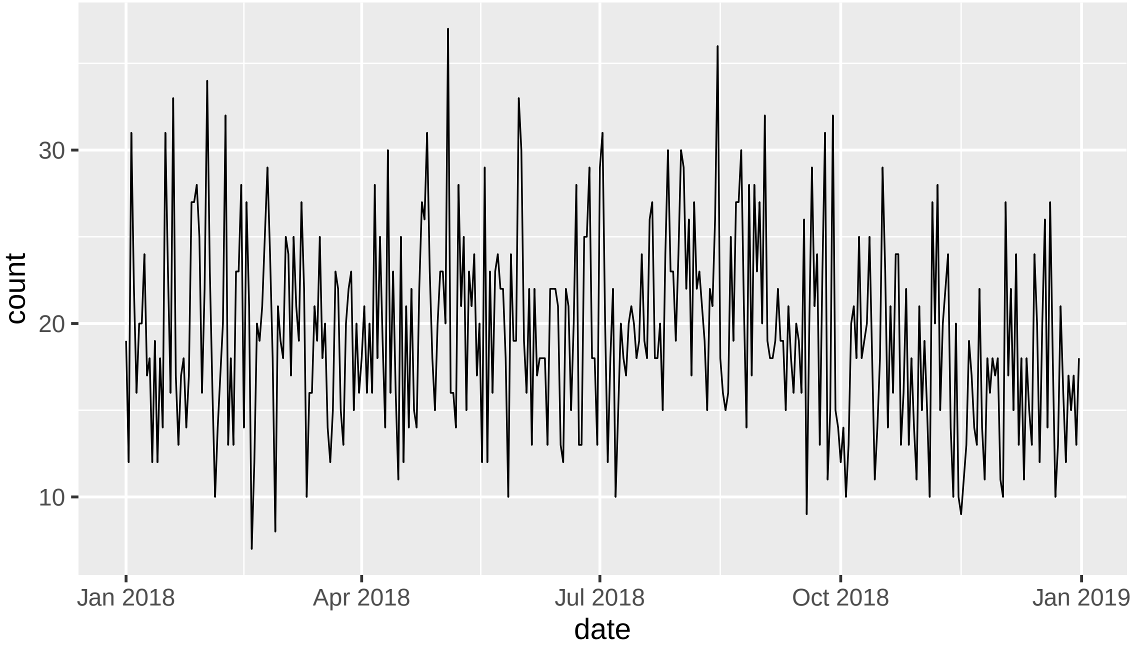
Time Series Plot

We can use `group_by()` and `summarise()` to count the number of burglaries that occurred on each date.

```
daily_crimes <- sfcrimes3 %>%
  group_by(date) %>%
  summarize(count = n())
daily_crimes
```

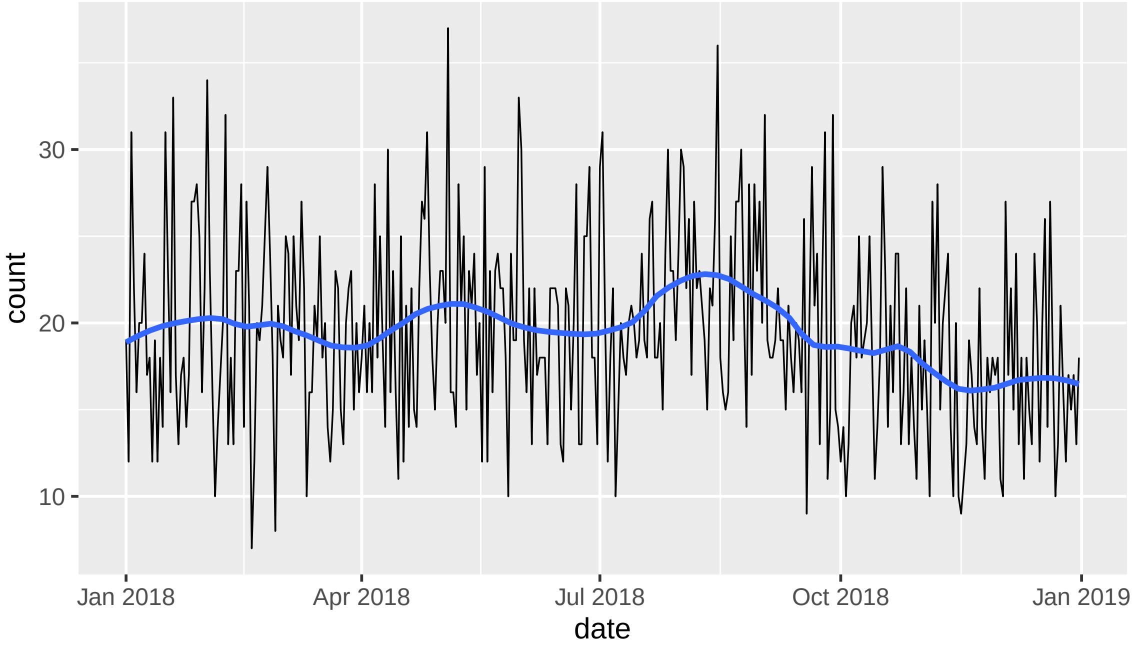
```
## # A tibble: 365 x 2
##    date       count
##    <date>     <int>
##  1 2018-01-01    19
##  2 2018-01-02    12
##  3 2018-01-03    31
##  4 2018-01-04    22
##  5 2018-01-05    16
##  6 2018-01-06    20
##  7 2018-01-07    20
##  8 2018-01-08    24
##  9 2018-01-09    17
## 10 2018-01-10    18
## # ... with 355 more rows
```

```
ggplot(daily_crimes, aes(x=date, y=count)) +
  geom_line(size=0.3)
```



Use `geom_smooth()` to better visualize the trend, and `span` to adjust smoothness.

```
ggplot(daily_crimes, aes(x=date, y=count)) +
  geom_line(size=0.3) +
  geom_smooth(span = 0.2, se = FALSE)
```


count
Jan 2018
Apr 2018
Jul 2018
Oct 2018
Jan 2019
date