Getting and cleaning Data Swirl exercises

Hariharan

This PDF contains the code and output generated for swirl exercises in the course Find the Getting-and-cleaning-data-swirl.Rmd in the same folder as this file to interact with the code and make changes for a better learning experience

1. Manipulating Data with dplyr

Setting up the environment

```
path2csv<-"C:/Users/MAHE/Documents/R/win-library/3.6/swirl/Courses/Getting_and_Cleaning_Data/Manipulati
mydf <- read.csv(path2csv, stringsAsFactors = FALSE)
dim(mydf)
```

```
## [1] 225468     11
```

```
head(mydf)
```

```
##   X       date     time   size r_version r_arch      r_os      package version
## 1 1 2014-07-08 00:54:41  80589     3.1.0 x86_64   mingw32    htmltools   0.2.4
## 2 2 2014-07-08 00:59:53 321767     3.1.0 x86_64   mingw32      tseries 0.10-32
## 3 3 2014-07-08 00:47:13 748063     3.1.0 x86_64 linux-gnu        party  1.0-15
## 4 4 2014-07-08 00:48:05 606104     3.1.0 x86_64 linux-gnu        Hmisc  3.14-4
## 5 5 2014-07-08 00:46:50  79825     3.0.2 x86_64 linux-gnu       digest   0.6.4
## 6 6 2014-07-08 00:48:04  77681     3.1.0 x86_64 linux-gnu randomForest   4.6-7
##   country ip_id
## 1      US     1
## 2      US     2
## 3      US     3
## 4      US     3
## 5      CA     4
## 6      US     3
```

```
# loading dplyr
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.6.3
```

```
##
## Attaching package: 'dplyr'
## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
packageVersion("dplyr")
```

```
## [1] '0.8.5'
```

The first step of working with data in dplyr is to load the data into what the package authors call a ‘data frame tbl’ or ‘tbl_df’. Use the following code to create a new tbl_df called cran:

```
cran <- tbl_df(mydf)
rm("mydf")
cran
```

```
## # A tibble: 225,468 x 11
##        X date  time    size r_version r_arch r_os  package version country ip_id
##    <int> <chr> <chr>  <int> <chr>     <chr>  <chr> <chr>   <chr>   <chr>   <int>
##  1     1 2014~ 00:5~ 8.06e4 3.1.0     x86_64 ming~ htmlto~ 0.2.4   US          1
##  2     2 2014~ 00:5~ 3.22e5 3.1.0     x86_64 ming~ tseries 0.10-32 US          2
##  3     3 2014~ 00:4~ 7.48e5 3.1.0     x86_64 linu~ party   1.0-15  US          3
##  4     4 2014~ 00:4~ 6.06e5 3.1.0     x86_64 linu~ Hmisc   3.14-4  US          3
##  5     5 2014~ 00:4~ 7.98e4 3.0.2     x86_64 linu~ digest  0.6.4   CA          4
##  6     6 2014~ 00:4~ 7.77e4 3.1.0     x86_64 linu~ random~ 4.6-7   US          3
##  7     7 2014~ 00:4~ 3.94e5 3.1.0     x86_64 linu~ plyr    1.8.1   US          3
##  8     8 2014~ 00:4~ 2.82e4 3.0.2     x86_64 linu~ whisker 0.3-2   US          5
##  9     9 2014~ 00:5~ 5.93e3 <NA>      <NA>   <NA>  Rcpp    0.10.4  CN          6
## 10    10 2014~ 00:1~ 2.21e6 3.0.2     x86_64 linu~ hfligh~ 0.1     US          7
## # ... with 225,458 more rows
```

using select

```
select(cran, ip_id, package, country)
```

```
## # A tibble: 225,468 x 3
##    ip_id package      country
##    <int> <chr>        <chr>
##  1     1 htmltools    US
##  2     2 tseries      US
##  3     3 party        US
##  4     3 Hmisc        US
##  5     4 digest       CA
##  6     3 randomForest US
##  7     3 plyr         US
##  8     5 whisker      US
##  9     6 Rcpp         CN
## 10     7 hflights     US
## # ... with 225,458 more rows
```

```r
select(cran, r_arch:country)
```

```
## # A tibble: 225,468 x 5
##    r_arch r_os      package      version country
##    <chr>  <chr>     <chr>        <chr>   <chr>  
##  1 x86_64 mingw32   htmltools    0.2.4   US     
##  2 x86_64 mingw32   tseries      0.10-32 US     
##  3 x86_64 linux-gnu party        1.0-15  US     
##  4 x86_64 linux-gnu Hmisc        3.14-4  US     
##  5 x86_64 linux-gnu digest       0.6.4   CA     
##  6 x86_64 linux-gnu randomForest 4.6-7   US     
##  7 x86_64 linux-gnu plyr         1.8.1   US     
##  8 x86_64 linux-gnu whisker      0.3-2   US     
##  9 <NA>   <NA>      Rcpp         0.10.4  CN     
## 10 x86_64 linux-gnu hflights     0.1     US     
## # ... with 225,458 more rows
```

```r
select(cran, -time)
```

```
## # A tibble: 225,468 x 10
##        X date       size r_version r_arch r_os   package   version country ip_id
##    <int> <chr>     <int> <chr>     <chr>  <chr>  <chr>     <chr>   <chr>   <int>
##  1     1 2014-0~   80589 3.1.0     x86_64 mingw~ htmltools 0.2.4   US          1
##  2     2 2014-0~  321767 3.1.0     x86_64 mingw~ tseries   0.10-32 US          2
##  3     3 2014-0~  748063 3.1.0     x86_64 linux~ party     1.0-15  US          3
##  4     4 2014-0~  606104 3.1.0     x86_64 linux~ Hmisc     3.14-4  US          3
##  5     5 2014-0~   79825 3.0.2     x86_64 linux~ digest    0.6.4   CA          4
##  6     6 2014-0~   77681 3.1.0     x86_64 linux~ randomFo~ 4.6-7   US          3
##  7     7 2014-0~  393754 3.1.0     x86_64 linux~ plyr      1.8.1   US          3
##  8     8 2014-0~   28216 3.0.2     x86_64 linux~ whisker   0.3-2   US          5
##  9     9 2014-0~    5928 <NA>      <NA>   <NA>   Rcpp      0.10.4  CN          6
## 10    10 2014-0~ 2206029 3.0.2     x86_64 linux~ hflights  0.1     US          7
## # ... with 225,458 more rows
```

```r
select(cran, -(X:size))
```

```
## # A tibble: 225,468 x 7
##    r_version r_arch r_os      package      version country ip_id
##    <chr>     <chr>  <chr>     <chr>        <chr>   <chr>   <int>
##  1 3.1.0     x86_64 mingw32   htmltools    0.2.4   US          1
##  2 3.1.0     x86_64 mingw32   tseries      0.10-32 US          2
##  3 3.1.0     x86_64 linux-gnu party        1.0-15  US          3
##  4 3.1.0     x86_64 linux-gnu Hmisc        3.14-4  US          3
##  5 3.0.2     x86_64 linux-gnu digest       0.6.4   CA          4
##  6 3.1.0     x86_64 linux-gnu randomForest 4.6-7   US          3
##  7 3.1.0     x86_64 linux-gnu plyr         1.8.1   US          3
##  8 3.0.2     x86_64 linux-gnu whisker      0.3-2   US          5
##  9 <NA>      <NA>   <NA>      Rcpp         0.10.4  CN          6
## 10 3.0.2     x86_64 linux-gnu hflights     0.1     US          7
## # ... with 225,458 more rows
```

using filter

```r
filter(cran, package == "swirl")
```

```
## # A tibble: 820 x 11
##         X date  time    size r_version r_arch r_os  package version country ip_id
##     <int> <chr> <chr>  <int> <chr>     <chr>  <chr> <chr>   <chr>   <chr>   <int>
##  1     27 2014~ 00:1~ 105350 3.0.2     x86_64 ming~ swirl   2.2.9   US         20
##  2    156 2014~ 00:2~  41261 3.1.0     x86_64 linu~ swirl   2.2.9   US         66
##  3    358 2014~ 00:1~ 105335 2.15.2    x86_64 ming~ swirl   2.2.9   CA        115
##  4    593 2014~ 00:5~ 105465 3.1.0     x86_64 darw~ swirl   2.2.9   MX        162
##  5    831 2014~ 00:5~ 105335 3.0.3     x86_64 ming~ swirl   2.2.9   US         57
##  6    997 2014~ 00:3~  41261 3.1.0     x86_64 ming~ swirl   2.2.9   US         70
##  7   1023 2014~ 00:3~ 106393 3.1.0     x86_64 ming~ swirl   2.2.9   BR        248
##  8   1144 2014~ 00:0~ 106534 3.0.2     x86_64 linu~ swirl   2.2.9   US        261
##  9   1402 2014~ 00:4~  41261 3.1.0     i386   ming~ swirl   2.2.9   US        234
## 10   1424 2014~ 00:4~ 106393 3.1.0     x86_64 linu~ swirl   2.2.9   US        301
## # ... with 810 more rows
```

```r
filter(cran, r_version == "3.1.1", country == "US")
```

```
## # A tibble: 1,588 x 11
##         X date  time    size r_version r_arch r_os  package version country ip_id
##     <int> <chr> <chr>  <int> <chr>     <chr>  <chr> <chr>   <chr>   <chr>   <int>
##  1   2216 2014~ 00:4~ 3.85e5 3.1.1     x86_64 darw~ colors~ 1.2-4   US        191
##  2  17332 2014~ 03:3~ 1.97e5 3.1.1     x86_64 darw~ httr    0.3     US       1704
##  3  17465 2014~ 03:2~ 2.33e4 3.1.1     x86_64 darw~ snow    0.3-13  US         62
##  4  18844 2014~ 03:5~ 1.91e5 3.1.1     x86_64 darw~ maxLik  1.2-0   US       1533
##  5  30182 2014~ 04:1~ 7.77e4 3.1.1     i386   ming~ random~ 4.6-7   US        646
##  6  30193 2014~ 04:0~ 2.35e6 3.1.1     i386   ming~ ggplot2 1.0.0   US          8
##  7  30195 2014~ 04:0~ 2.99e5 3.1.1     i386   ming~ fExtre~ 3010.81 US       2010
##  8  30217 2014~ 04:3~ 5.68e5 3.1.1     i386   ming~ rJava   0.9-6   US         98
##  9  30245 2014~ 04:1~ 5.27e5 3.1.1     i386   ming~ LPCM    0.44-8  US          8
## 10  30354 2014~ 04:3~ 1.76e6 3.1.1     i386   ming~ mgcv    1.8-1   US       2122
## # ... with 1,578 more rows
```

```r
filter(cran, r_version <= "3.0.2", country == "IN")
```

```
## # A tibble: 4,139 x 11
##         X date  time    size r_version r_arch r_os  package version country ip_id
##     <int> <chr> <chr>  <int> <chr>     <chr>  <chr> <chr>   <chr>   <chr>   <int>
##  1    348 2014~ 00:4~ 1.02e7 3.0.0     x86_64 ming~ BH      1.54.0~ IN        112
##  2   9990 2014~ 02:1~ 3.97e5 3.0.2     x86_64 linu~ equate~ 1.1     IN       1054
##  3   9991 2014~ 02:1~ 1.19e5 3.0.2     x86_64 linu~ ggdend~ 0.1-14  IN       1054
##  4   9992 2014~ 02:1~ 8.18e4 3.0.2     x86_64 linu~ dfcrm   0.2-2   IN       1054
##  5  10022 2014~ 02:1~ 1.56e6 2.15.0    x86_64 ming~ RcppAr~ 0.4.32~ IN       1060
##  6  10023 2014~ 02:1~ 1.18e6 2.15.1    i686   linu~ foreca~ 5.4     IN       1060
##  7  10189 2014~ 02:3~ 9.09e5 3.0.2     x86_64 linu~ editru~ 2.7.2   IN       1054
##  8  10199 2014~ 02:3~ 1.78e5 3.0.2     x86_64 linu~ energy  1.6.1   IN       1054
##  9  10200 2014~ 02:3~ 5.18e4 3.0.2     x86_64 linu~ ENmisc  1.2-7   IN       1054
## 10  10201 2014~ 02:3~ 6.52e4 3.0.2     x86_64 linu~ entropy 1.2.0   IN       1054
## # ... with 4,129 more rows
```

```r
filter(cran, country == "US" | country == "IN")
```

```
## # A tibble: 95,283 x 11
##        X date  time    size r_version r_arch r_os  package version country ip_id
##    <int> <chr> <chr>  <int> <chr>     <chr>  <chr> <chr>   <chr>   <chr>   <int>
##  1     1 2014~ 00:5~ 8.06e4 3.1.0     x86_64 ming~ htmlto~ 0.2.4   US          1
##  2     2 2014~ 00:5~ 3.22e5 3.1.0     x86_64 ming~ tseries 0.10-32 US          2
##  3     3 2014~ 00:4~ 7.48e5 3.1.0     x86_64 linu~ party   1.0-15  US          3
##  4     4 2014~ 00:4~ 6.06e5 3.1.0     x86_64 linu~ Hmisc   3.14-4  US          3
##  5     6 2014~ 00:4~ 7.77e4 3.1.0     x86_64 linu~ random~ 4.6-7   US          3
##  6     7 2014~ 00:4~ 3.94e5 3.1.0     x86_64 linu~ plyr    1.8.1   US          3
##  7     8 2014~ 00:4~ 2.82e4 3.0.2     x86_64 linu~ whisker 0.3-2   US          5
##  8    10 2014~ 00:1~ 2.21e6 3.0.2     x86_64 linu~ hfligh~ 0.1     US          7
##  9    11 2014~ 00:1~ 5.27e5 3.0.2     x86_64 linu~ LPCM    0.44-8  US          8
## 10    12 2014~ 00:1~ 2.35e6 2.14.1    x86_64 linu~ ggplot2 1.0.0   US          8
## # ... with 95,273 more rows
```

```r
filter(cran, size > 100500, r_os == "linux-gnu")
```

```
## # A tibble: 33,683 x 11
##        X date  time    size r_version r_arch r_os  package version country ip_id
##    <int> <chr> <chr>  <int> <chr>     <chr>  <chr> <chr>   <chr>   <chr>   <int>
##  1     3 2014~ 00:4~ 7.48e5 3.1.0     x86_64 linu~ party   1.0-15  US          3
##  2     4 2014~ 00:4~ 6.06e5 3.1.0     x86_64 linu~ Hmisc   3.14-4  US          3
##  3     7 2014~ 00:4~ 3.94e5 3.1.0     x86_64 linu~ plyr    1.8.1   US          3
##  4    10 2014~ 00:1~ 2.21e6 3.0.2     x86_64 linu~ hfligh~ 0.1     US          7
##  5    11 2014~ 00:1~ 5.27e5 3.0.2     x86_64 linu~ LPCM    0.44-8  US          8
##  6    12 2014~ 00:1~ 2.35e6 2.14.1    x86_64 linu~ ggplot2 1.0.0   US          8
##  7    14 2014~ 00:1~ 3.10e6 3.0.2     x86_64 linu~ Rcpp    0.9.7   VE         10
##  8    15 2014~ 00:1~ 5.68e5 3.1.0     x86_64 linu~ rJava   0.9-6   US         11
##  9    16 2014~ 00:1~ 1.60e6 3.1.0     x86_64 linu~ RSQLite 0.11.4  US          7
## 10    18 2014~ 00:2~ 1.87e5 3.1.0     x86_64 linu~ ipred   0.9-3   DE         13
## # ... with 33,673 more rows
```

```r
filter(cran, !is.na(r_version))
```

```
## # A tibble: 207,205 x 11
##        X date  time    size r_version r_arch r_os  package version country ip_id
##    <int> <chr> <chr>  <int> <chr>     <chr>  <chr> <chr>   <chr>   <chr>   <int>
##  1     1 2014~ 00:5~ 8.06e4 3.1.0     x86_64 ming~ htmlto~ 0.2.4   US          1
##  2     2 2014~ 00:5~ 3.22e5 3.1.0     x86_64 ming~ tseries 0.10-32 US          2
##  3     3 2014~ 00:4~ 7.48e5 3.1.0     x86_64 linu~ party   1.0-15  US          3
##  4     4 2014~ 00:4~ 6.06e5 3.1.0     x86_64 linu~ Hmisc   3.14-4  US          3
##  5     5 2014~ 00:4~ 7.98e4 3.0.2     x86_64 linu~ digest  0.6.4   CA          4
##  6     6 2014~ 00:4~ 7.77e4 3.1.0     x86_64 linu~ random~ 4.6-7   US          3
##  7     7 2014~ 00:4~ 3.94e5 3.1.0     x86_64 linu~ plyr    1.8.1   US          3
##  8     8 2014~ 00:4~ 2.82e4 3.0.2     x86_64 linu~ whisker 0.3-2   US          5
##  9    10 2014~ 00:1~ 2.21e6 3.0.2     x86_64 linu~ hfligh~ 0.1     US          7
## 10    11 2014~ 00:1~ 5.27e5 3.0.2     x86_64 linu~ LPCM    0.44-8  US          8
## # ... with 207,195 more rows
```

using arrange

```
cran2 <- select(cran, size:ip_id)
arrange(cran2, ip_id)#ip_id is in ascending order
```

```
## # A tibble: 225,468 x 8
##      size r_version r_arch r_os         package     version country ip_id
##     <int> <chr>     <chr>  <chr>        <chr>       <chr>   <chr>   <int>
##  1  80589 3.1.0     x86_64 mingw32      htmltools   0.2.4   US          1
##  2 180562 3.0.2     x86_64 mingw32      yaml        2.1.13  US          1
##  3 190120 3.1.0     i386   mingw32      babel       0.2-6   US          1
##  4 321767 3.1.0     x86_64 mingw32      tseries     0.10-32 US          2
##  5  52281 3.0.3     x86_64 darwin10.8.0 quadprog    1.5-5   US          2
##  6 876702 3.1.0     x86_64 linux-gnu    zoo         1.7-11  US          2
##  7 321764 3.0.2     x86_64 linux-gnu    tseries     0.10-32 US          2
##  8 876702 3.1.0     x86_64 linux-gnu    zoo         1.7-11  US          2
##  9 321768 3.1.0     x86_64 mingw32      tseries     0.10-32 US          2
## 10 784093 3.1.0     x86_64 linux-gnu    strucchange 1.5-0   US          2
## # ... with 225,458 more rows
```

```
arrange(cran2, desc(ip_id))#To do the same, but in descending order
```

```
## # A tibble: 225,468 x 8
##       size r_version r_arch r_os         package      version country ip_id
##      <int> <chr>     <chr>  <chr>        <chr>        <chr>   <chr>   <int>
##  1    5933 <NA>      <NA>   <NA>         CPE          1.4.2   CN      13859
##  2  569241 3.1.0     x86_64 mingw32      multcompView 0.1-5   US      13858
##  3  228444 3.1.0     x86_64 mingw32      tourr        0.5.3   NZ      13857
##  4  308962 3.1.0     x86_64 darwin13.1.0 ctv          0.7-9   CN      13856
##  5  950964 3.0.3     i386   mingw32      knitr        1.6     CA      13855
##  6   80185 3.0.3     i386   mingw32      htmltools    0.2.4   CA      13855
##  7 1431750 3.0.3     i386   mingw32      shiny        0.10.0  CA      13855
##  8 2189695 3.1.0     x86_64 mingw32      RMySQL       0.9-3   US      13854
##  9 4818024 3.1.0     i386   mingw32      igraph       0.7.1   US      13853
## 10  197495 3.1.0     x86_64 mingw32      coda         0.16-1  US      13852
## # ... with 225,458 more rows
```

```
arrange(cran2, package, ip_id)# first arrange by package names (ascending alphabetically), then by ip_i
```

```
## # A tibble: 225,468 x 8
##     size r_version r_arch r_os         package version country ip_id
##    <int> <chr>     <chr>  <chr>        <chr>   <chr>   <chr>   <int>
##  1 71677 3.0.3     x86_64 darwin10.8.0 A3      0.9.2   CN       1003
##  2 71672 3.1.0     x86_64 linux-gnu    A3      0.9.2   US       1015
##  3 71677 3.1.0     x86_64 mingw32      A3      0.9.2   IN       1054
##  4 70438 3.0.1     x86_64 darwin10.8.0 A3      0.9.2   CN       1513
##  5 71677 <NA>      <NA>   <NA>         A3      0.9.2   BR       1526
##  6 71892 3.0.2     x86_64 linux-gnu    A3      0.9.2   IN       1542
##  7 71677 3.1.0     x86_64 linux-gnu    A3      0.9.2   ZA       2925
##  8 71672 3.1.0     x86_64 mingw32      A3      0.9.2   IL       3889
##  9 71677 3.0.3     x86_64 mingw32      A3      0.9.2   DE       3917
## 10 71672 3.1.0     x86_64 mingw32      A3      0.9.2   US       4219
## # ... with 225,458 more rows
```

```
arrange(cran2, country, desc(r_version), ip_id)
```

```
## # A tibble: 225,468 x 8
##       size r_version r_arch r_os      package       version   country ip_id
##      <int> <chr>     <chr>  <chr>     <chr>         <chr>     <chr>   <int>
##  1 1556858 3.1.1     i386   mingw32   RcppArmadillo 0.4.320.0 A1       2843
##  2 1823512 3.1.0     x86_64 linux-gnu mgcv          1.8-1     A1       2843
##  3   15732 3.1.0     i686   linux-gnu grnn          0.1.0     A1       3146
##  4 3014840 3.1.0     x86_64 mingw32   Rcpp          0.11.2    A1       3146
##  5  660087 3.1.0     i386   mingw32   xts           0.9-7     A1       3146
##  6  522261 3.1.0     i386   mingw32   FNN           1.1       A1       3146
##  7  522263 3.1.0     i386   mingw32   FNN           1.1       A1       3146
##  8 1676627 3.1.0     x86_64 linux-gnu rgeos         0.3-5     A1       3146
##  9 2118530 3.1.0     x86_64 linux-gnu spacetime     1.1-0     A1       3146
## 10 2217180 3.1.0     x86_64 mingw32   gstat         1.0-19    A1       3146
## # ... with 225,458 more rows
```

using mutate

```
cran3 <- select(cran, ip_id, package, size)
mutate(cran3, size_mb = size / 2^20)
```

```
## # A tibble: 225,468 x 4
##    ip_id package         size size_mb
##    <int> <chr>          <int>   <dbl>
##  1     1 htmltools      80589 0.0769
##  2     2 tseries       321767 0.307
##  3     3 party         748063 0.713
##  4     3 Hmisc         606104 0.578
##  5     4 digest         79825 0.0761
##  6     3 randomForest   77681 0.0741
##  7     3 plyr          393754 0.376
##  8     5 whisker        28216 0.0269
##  9     6 Rcpp            5928 0.00565
## 10     7 hflights     2206029 2.10
## # ... with 225,458 more rows
```

```
mutate(cran3, size_mb = size / 2^20, size_gb = size_mb / 2^10)
```

```
## # A tibble: 225,468 x 5
##    ip_id package         size size_mb    size_gb
##    <int> <chr>          <int>   <dbl>      <dbl>
##  1     1 htmltools      80589 0.0769  0.0000751
##  2     2 tseries       321767 0.307   0.000300
##  3     3 party         748063 0.713   0.000697
##  4     3 Hmisc         606104 0.578   0.000564
##  5     4 digest         79825 0.0761  0.0000743
##  6     3 randomForest   77681 0.0741   0.0000723
##  7     3 plyr          393754 0.376    0.000367
##  8     5 whisker        28216 0.0269   0.0000263
##  9     6 Rcpp            5928 0.00565  0.00000552
## 10     7 hflights     2206029 2.10     0.00205
## # ... with 225,458 more rows
```

```
mutate(cran3, correct_size = size + 1000)
```

```
## # A tibble: 225,468 x 4
##    ip_id package         size correct_size
##    <int> <chr>          <int>        <dbl>
##  1     1 htmltools      80589        81589
##  2     2 tseries       321767       322767
##  3     3 party         748063       749063
##  4     3 Hmisc         606104       607104
##  5     4 digest         79825        80825
##  6     3 randomForest   77681        78681
##  7     3 plyr          393754       394754
##  8     5 whisker        28216        29216
##  9     6 Rcpp            5928         6928
## 10     7 hflights     2206029      2207029
## # ... with 225,458 more rows
```

using summarize

```
summarize(cran, avg_bytes = mean(size))
```

```
## # A tibble: 1 x 1
##   avg_bytes
##       <dbl>
## 1   844086.
```

2. Grouping and Chaining with dplyr

Setting up the environment

```
path2csv<-"C:/Users/MAHE/Documents/R/win-library/3.6/swirl/Courses/Getting_and_Cleaning_Data/Manipulati
mydf <- read.csv(path2csv, stringsAsFactors = FALSE)
dim(mydf)
```

```
## [1] 225468     11
```

```
head(mydf)
```

```
##   X       date     time   size r_version r_arch      r_os      package version
## 1 1 2014-07-08 00:54:41  80589     3.1.0 x86_64   mingw32    htmltools   0.2.4
## 2 2 2014-07-08 00:59:53 321767     3.1.0 x86_64   mingw32      tseries 0.10-32
## 3 3 2014-07-08 00:47:13 748063     3.1.0 x86_64 linux-gnu        party  1.0-15
## 4 4 2014-07-08 00:48:05 606104     3.1.0 x86_64 linux-gnu        Hmisc  3.14-4
## 5 5 2014-07-08 00:46:50  79825     3.0.2 x86_64 linux-gnu       digest   0.6.4
## 6 6 2014-07-08 00:48:04  77681     3.1.0 x86_64 linux-gnu randomForest   4.6-7
##   country ip_id
## 1      US     1
## 2      US     2
## 3      US     3
## 4      US     3
## 5      CA     4
## 6      US     3
```

```
# loading dplyr
library(dplyr)
packageVersion("dplyr")
```

```
## [1] '0.8.5'
```

```
cran <- tbl_df(mydf)
```

using group-by

```
by_package <- group_by(cran, package)
by_package
```

```
## # A tibble: 225,468 x 11
## # Groups:   package [6,023]
##        X date  time    size r_version r_arch r_os  package version country ip_id
##    <int> <chr> <chr>  <int> <chr>     <chr>  <chr> <chr>   <chr>   <chr>   <int>
##  1     1 2014~ 00:5~ 8.06e4 3.1.0     x86_64 ming~ htmlto~ 0.2.4   US          1
##  2     2 2014~ 00:5~ 3.22e5 3.1.0     x86_64 ming~ tseries 0.10-32 US          2
##  3     3 2014~ 00:4~ 7.48e5 3.1.0     x86_64 linu~ party   1.0-15  US          3
##  4     4 2014~ 00:4~ 6.06e5 3.1.0     x86_64 linu~ Hmisc   3.14-4  US          3
##  5     5 2014~ 00:4~ 7.98e4 3.0.2     x86_64 linu~ digest  0.6.4   CA          4
##  6     6 2014~ 00:4~ 7.77e4 3.1.0     x86_64 linu~ random~ 4.6-7   US          3
##  7     7 2014~ 00:4~ 3.94e5 3.1.0     x86_64 linu~ plyr    1.8.1   US          3
##  8     8 2014~ 00:4~ 2.82e4 3.0.2     x86_64 linu~ whisker 0.3-2   US          5
##  9     9 2014~ 00:5~ 5.93e3 <NA>      <NA>   <NA>  Rcpp    0.10.4  CN          6
## 10    10 2014~ 00:1~ 2.21e6 3.0.2     x86_64 linu~ hfligh~ 0.1     US          7
## # ... with 225,458 more rows
```

```
summarize(by_package, mean(size))
```

```
## # A tibble: 6,023 x 2
##    package     `mean(size)`
##    <chr>              <dbl>
##  1 A3                 62195.
##  2 abc              4826665
##  3 abcdeFBA          455980.
##  4 ABCExtremes        22904.
##  5 ABCoptim           17807.
##  6 ABCp2              30473.
##  7 abctools         2589394
##  8 abd               453631.
##  9 abf2               35693.
## 10 abind              32939.
## # ... with 6,013 more rows
```

```
pack_sum <- summarize(by_package,
                      count = n(),
                      unique = n_distinct(ip_id),
                      countries = n_distinct(country),
                      avg_bytes = mean(size))
pack_sum
```

```
## # A tibble: 6,023 x 5
##    package     count unique countries avg_bytes
##    <chr>       <int>  <int>     <int>     <dbl>
##  1 A3             25     24        10    62195.
##  2 abc            29     25        16  4826665
##  3 abcdeFBA       15     15         9   455980.
##  4 ABCExtremes    18     17         9    22904.
##  5 ABCoptim       16     15         9    17807.
##  6 ABCp2          18     17        10    30473.
##  7 abctools       19     19        11  2589394
##  8 abd            17     16        10   453631.
##  9 abf2           13     13         9    35693.
## 10 abind         396    365        50    32939.
## # ... with 6,013 more rows
```

The ‘count’ column, created with n(), contains the total number of rows (i.e.downloads) for each package. The ‘unique’ column, created with n_distinct(ip_id), gives the total number of unique downloads for each package, as measured by the number of distinct ip_id’s. The ‘countries’ column, created with n_distinct(country), provides the number of countries in which each package was downloaded. And finally, the ‘avg_bytes’ column, created with mean(size), contains the mean download size (in bytes) for each package.

Naturally, we’d like to know which packages were most popular on the day these data were collected (July 8, 2014). Let’s start by isolating the top 1% of packages, based on the total number of downloads as measured by the ‘count’ column.

```
quantile(pack_sum$count, probs = 0.99)
```

```
##    99%
## 679.56
```

```
top_counts <- filter(pack_sum, count > 679)
top_counts
```

```
## # A tibble: 61 x 5
##    package    count unique countries avg_bytes
##    <chr>      <int>  <int>     <int>     <dbl>
##  1 bitops      1549   1408        76    28715.
##  2 car         1008    837        64  1229122.
##  3 caTools      812    699        64   176589.
##  4 colorspace  1683   1433        80   357411.
##  5 data.table   680    564        59  1252721.
##  6 DBI         2599    492        48   206933.
##  7 devtools     769    560        55   212933.
##  8 dichromat   1486   1257        74   134732.
##  9 digest      2210   1894        83   120549.
## 10 doSNOW       740     75        24     8364.
## # ... with 51 more rows
```

```
View(top_counts)
top_counts_sorted <- arrange(top_counts, desc(count))
top_counts_sorted
```

```
## # A tibble: 61 x 5
##    package  count unique countries avg_bytes
##    <chr>    <int>  <int>     <int>     <dbl>
##  1 ggplot2   4602   1680        81  2427716.
##  2 Rcpp      3195   2044        84  2512100.
##  3 plyr      2908   1754        81   799123.
##  4 rJava     2773    963        70   633522.
##  5 DBI       2599    492        48   206933.
##  6 LPCM      2335     17        10   526814.
##  7 stringr   2267   1948        82    65277.
##  8 digest    2210   1894        83   120549.
##  9 reshape2  2032   1652        76   330128.
## 10 foreach   1984    485        53   358070.
## # ... with 51 more rows
```

```
quantile(pack_sum$unique, probs = 0.99)
```

```
## 99%
## 465
```

```
top_unique <- filter(pack_sum, unique > 465)
top_unique_sorted <- arrange(top_unique, desc(unique))
top_unique_sorted
```

```
## # A tibble: 60 x 5
##    package      count unique countries avg_bytes
##    <chr>        <int>  <int>     <int>     <dbl>
##  1 Rcpp          3195   2044        84  2512100.
##  2 stringr       2267   1948        82    65277.
##  3 digest        2210   1894        83   120549.
##  4 plyr          2908   1754        81   799123.
##  5 ggplot2       4602   1680        81  2427716.
##  6 reshape2      2032   1652        76   330128.
##  7 RColorBrewer  1890   1584        79    22764.
##  8 colorspace    1683   1433        80   357411.
##  9 bitops        1549   1408        76    28715.
## 10 scales        1726   1408        77   126819.
## # ... with 50 more rows
```

Chaining

```
by_package <- group_by(cran, package)
pack_sum <- summarize(by_package,
                      count = n(),
                      unique = n_distinct(ip_id),
                      countries = n_distinct(country),
                      avg_bytes = mean(size))
top_countries <- filter(pack_sum, countries > 60)
result1 <- arrange(top_countries, desc(countries), avg_bytes)
print(result1)
```

```
## # A tibble: 46 x 5
##    package      count unique countries avg_bytes
##    <chr>        <int>  <int>     <int>     <dbl>
##  1 Rcpp          3195   2044        84  2512100.
##  2 digest        2210   1894        83   120549.
##  3 stringr       2267   1948        82    65277.
##  4 plyr          2908   1754        81   799123.
##  5 ggplot2       4602   1680        81  2427716.
##  6 colorspace    1683   1433        80   357411.
##  7 RColorBrewer  1890   1584        79    22764.
##  8 scales        1726   1408        77   126819.
##  9 bitops        1549   1408        76    28715.
## 10 reshape2      2032   1652        76   330128.
## # ... with 36 more rows
```

Same operations as above but using function call embedding

```
result2 <-
  arrange(
    filter(
      summarize(
        group_by(cran,
                 package
        ),
        count = n(),
        unique = n_distinct(ip_id),
        countries = n_distinct(country),
        avg_bytes = mean(size)
      ),
      countries > 60
    ),
    desc(countries),
    avg_bytes
  )

print(result2)
```

```
## # A tibble: 46 x 5
##    package      count unique countries avg_bytes
##    <chr>        <int>  <int>     <int>     <dbl>
##  1 Rcpp          3195   2044        84  2512100.
##  2 digest        2210   1894        83   120549.
##  3 stringr       2267   1948        82    65277.
##  4 plyr          2908   1754        81   799123.
##  5 ggplot2       4602   1680        81  2427716.
##  6 colorspace    1683   1433        80   357411.
##  7 RColorBrewer  1890   1584        79    22764.
##  8 scales        1726   1408        77   126819.
##  9 bitops        1549   1408        76    28715.
## 10 reshape2      2032   1652        76   330128.
## # ... with 36 more rows
```

In this script, we've used a special chaining operator, %>%

```
# you read it, you can pronounce the %>% operator as
# the word 'then'.

result3 <-
  cran %>%
  group_by(package) %>%
  summarize(count = n(),
            unique = n_distinct(ip_id),
            countries = n_distinct(country),
            avg_bytes = mean(size)
  ) %>%
  filter(countries > 60) %>%
  arrange(desc(countries), avg_bytes)

# Print result to console
print(result3)
```

```
## # A tibble: 46 x 5
##    package      count unique countries avg_bytes
##    <chr>        <int>  <int>     <int>     <dbl>
##  1 Rcpp          3195   2044        84  2512100.
##  2 digest        2210   1894        83   120549.
##  3 stringr       2267   1948        82    65277.
##  4 plyr          2908   1754        81   799123.
##  5 ggplot2       4602   1680        81  2427716.
##  6 colorspace    1683   1433        80   357411.
##  7 RColorBrewer  1890   1584        79    22764.
##  8 scales        1726   1408        77   126819.
##  9 bitops        1549   1408        76    28715.
## 10 reshape2      2032   1652        76   330128.
## # ... with 36 more rows
```

```
# select() the following columns from cran. Keep in mind
# that when you're using the chaining operator, you don't
# need to specify the name of the data tbl in your call to
# select().
#
# 1. ip_id
# 2. country
# 3. package
# 4. size
#
# The call to print() at the end of the chain is optional,
# but necessary if you want your results printed to the
# console. Note that since there are no additional arguments
# to print(), you can leave off the parentheses after
# the function name. This is a convenient feature of the %>%
# operator.

cran %>%
  select(ip_id, country, package, size) %>%
    print
```

```
## # A tibble: 225,468 x 4
##    ip_id country package         size
##    <int> <chr>   <chr>          <int>
##  1     1 US      htmltools      80589
##  2     2 US      tseries       321767
##  3     3 US      party         748063
##  4     3 US      Hmisc         606104
##  5     4 CA      digest         79825
##  6     3 US      randomForest   77681
##  7     3 US      plyr          393754
##  8     5 US      whisker        28216
##  9     6 CN      Rcpp            5928
## 10     7 US      hflights     2206029
## # ... with 225,458 more rows
```

```
# Use mutate() to add a column called size_mb that contains
# the size of each download in megabytes (i.e. size / 2^20).
#
# If you want your results printed to the console, add
# print to the end of your chain.

cran %>%
  select(ip_id, country, package, size) %>%
  mutate(size_mb = size / 2^20)
```

```
## # A tibble: 225,468 x 5
##    ip_id country package         size size_mb
##    <int> <chr>   <chr>          <int>   <dbl>
##  1     1 US      htmltools      80589 0.0769
##  2     2 US      tseries       321767 0.307
##  3     3 US      party         748063 0.713
##  4      3 US      Hmisc          606104 0.578
##  5      4 CA      digest          79825 0.0761
##  6      3 US      randomForest    77681 0.0741
##  7      3 US      plyr           393754 0.376
##  8      5 US      whisker         28216 0.0269
##  9      6 CN      Rcpp             5928 0.00565
## 10      7 US      hflights      2206029 2.10
## # ... with 225,458 more rows
```

```
# Use filter() to select all rows for which size_mb is
# less than or equal to (<=) 0.5.
#
# If you want your results printed to the console, add
# print to the end of your chain.

cran %>%
  select(ip_id, country, package, size) %>%
  mutate(size_mb = size / 2^20) %>%
  filter(size_mb <= 0.5)
```

```
## # A tibble: 142,021 x 5
##    ip_id country package        size size_mb
##    <int> <chr>   <chr>         <int>   <dbl>
##  1     1 US      htmltools     80589 0.0769
##  2     2 US      tseries      321767 0.307
##  3     4 CA      digest        79825 0.0761
##  4     3 US      randomForest  77681 0.0741
##  5     3 US      plyr         393754 0.376
##  6     5 US      whisker       28216 0.0269
##  7     6 CN      Rcpp           5928 0.00565
##  8    13 DE      ipred        186685 0.178
##  9    14 US      mnormt        36204 0.0345
## 10    16 US      iterators    289972 0.277
## # ... with 142,011 more rows
```

```
# arrange() the result by size_mb, in descending order.
#
# If you want your results printed to the console, add
# print to the end of your chain.

cran %>%
  select(ip_id, country, package, size) %>%
  mutate(size_mb = size / 2^20) %>%
  filter(size_mb <= 0.5) %>%
  arrange(desc(size_mb))
```

```
## # A tibble: 142,021 x 5
##    ip_id country package                  size size_mb
##    <int> <chr>   <chr>                   <int>   <dbl>
##  1 11034 DE      phia                   524232   0.500
##  2  9643 US      tis                    524152   0.500
##  3  1542 IN      RcppSMC                524060   0.500
##  4 12354 US      lessR                  523916   0.500
##  5 12072 US      colorspace            523880   0.500
##  6  2514 KR      depmixS4              523863   0.500
##  7  1111 US      depmixS4              523858   0.500
##  8  8865 CR      depmixS4              523858   0.500
##  9  5908 CN      RcmdrPlugin.KMggplot2 523852   0.500
## 10 12354 US      RcmdrPlugin.KMggplot2 523852   0.500
## # ... with 142,011 more rows
```

3.Tidying Data with tidyr

Setting up

```
library(tidyr)
```

```
## Warning: package 'tidyr' was built under R version 3.6.3
```

```
# recreating dataset usend in lesson
students<- data.frame("grade" =c('A','B','C','D','E') , "male" = as.integer(c(5,4,8,4,5)), "female" = as

students
```

```
##   grade male female
## 1     A    5      3
## 2     B    4      1
## 3     C    8      6
## 4     D    4      5
## 5     E    5      5
```

```
gather(students, sex, count, -grade)
```

```
##    grade    sex count
## 1      A   male     5
## 2      B   male     4
## 3      C   male     8
## 4      D   male     4
## 5      E   male     5
## 6      A female     3
## 7      B female     1
## 8      C female     6
## 9      D female     5
## 10     E female     5
```

```
students2<- data.frame("grade" =c('A','B','C','D','E') , "male_1" = as.integer(c(7,4,7,8,8)), "female_1

students2
```

```
##   grade male_1 female_1 male_2 female_2
## 1     A      7        0      5        8
## 2     B      4        0      5        8
## 3     C      7        4      5        6
## 4     D      8        2      8        1
## 5     E      8        4      1        0
```

```
res <- gather(students2, sex_class, count, -grade)
res
```

```
##    grade sex_class count
## 1      A    male_1     7
## 2      B    male_1     4
## 3      C    male_1     7
## 4      D    male_1     8
## 5      E    male_1     8
## 6      A  female_1     0
## 7      B  female_1     0
## 8      C  female_1     4
## 9      D  female_1     2
## 10     E  female_1     4
## 11     A    male_2     5
## 12     B    male_2     5
## 13     C    male_2     5
## 14     D    male_2     8
## 15     E    male_2     1
## 16     A  female_2     8
## 17     B  female_2     8
## 18     C  female_2     6
## 19     D  female_2     1
## 20     E  female_2     0
```

```
separate(res, sex_class, c("sex", "class"))
```

```
##    grade    sex class count
## 1      A   male     1     7
## 2      B   male     1     4
## 3      C   male     1     7
## 4      D   male     1     8
## 5      E   male     1     8
## 6      A female     1     0
## 7      B female     1     0
## 8      C female     1     4
## 9      D female     1     2
## 10     E female     1     4
## 11     A   male     2     5
## 12     B   male     2     5
## 13     C   male     2     5
## 14     D   male     2     8
## 15     E   male     2     1
## 16     A female     2     8
## 17     B female     2     8
## 18     C female     2     6
## 19     D female     2     1
## 20     E female     2     0
```

using chaining

```
students2 %>%
  gather(sex_class, count, -grade) %>%
  separate(sex_class, c("sex", "class")) %>%
  print
```

```
##    grade    sex class count
## 1      A   male     1     7
## 2      B   male     1     4
## 3      C   male     1     7
## 4      D   male     1     8
## 5      E   male     1     8
## 6      A female     1     0
## 7      B female     1     0
## 8      C female     1     4
## 9      D female     1     2
## 10     E female     1     4
## 11     A   male     2     5
## 12     B   male     2     5
## 13     C   male     2     5
## 14     D   male     2     8
## 15     E   male     2     1
## 16     A female     2     8
## 17     B female     2     8
## 18     C female     2     6
## 19     D female     2     1
## 20     E female     2     0
```

```
students3<- data.frame(
  "name" = c("Sally","Sally","Jeff","Jeff","Roger","Roger","Karen","Karen","Brian","Brian"),
  "test" = c("midterm","final","midterm","final","midterm","final","midterm","final","midterm","final")
  "class1" = c("A","C",NA,NA,NA,NA,NA,NA,"B","B"),
  "class2" = c(NA,NA,"D","E","C","A",NA,NA,NA,NA),
  "class3" = c("B","C",NA,NA,NA,NA,"C","C",NA,NA),
  "class5" = c(NA,NA,NA,NA,"B","A",NA,NA,"A","C"),
  stringsAsFactors = FALSE
)
students3
```

```
##     name    test class1 class2 class3 class5
## 1  Sally midterm      A   <NA>      B   <NA>
## 2  Sally   final      C   <NA>      C   <NA>
## 3   Jeff midterm   <NA>      D   <NA>   <NA>
## 4   Jeff   final   <NA>      E   <NA>   <NA>
## 5  Roger midterm   <NA>      C   <NA>      B
## 6  Roger   final   <NA>      A   <NA>      A
## 7  Karen midterm   <NA>   <NA>      C   <NA>
## 8  Karen   final   <NA>   <NA>      C   <NA>
## 9  Brian midterm      B   <NA>   <NA>      A
## 10 Brian   final      B   <NA>   <NA>      C
```

```
# Call gather() to gather the columns class1
# through class5 into a new variable called class.
# The 'key' should be class, and the 'value'
# should be grade.
#
# tidyr makes it easy to reference multiple adjacent
# columns with class1:class5, just like with sequences
# of numbers.
#
# Since each student is only enrolled in two of
# the five possible classes, there are lots of missing
# values (i.e. NAs). Use the argument na.rm = TRUE
# to omit these values from the final result.
#
# Remember that when you're using the %>% operator,
# the value to the left of it gets inserted as the
# first argument to the function on the right.
#
# Consult ?gather and/or ?chain if you get stuck.
#
students3 %>%
  gather(class, grade, class1:class5, na.rm = TRUE) %>%
  print
```

```
##     name    test  class grade
## 1  Sally midterm class1     A
## 2  Sally   final class1     C
## 9  Brian midterm class1     B
## 10 Brian   final class1     B
## 13  Jeff midterm class2     D
## 14  Jeff   final class2     E
## 15 Roger midterm class2     C
## 16 Roger   final class2     A
## 21 Sally midterm class3     B
## 22 Sally   final class3     C
## 27 Karen midterm class3     C
## 28 Karen   final class3     C
## 35 Roger midterm class5     B
## 36 Roger   final class5     A
## 39 Brian midterm class5     A
## 40 Brian   final class5     C
```

```
# This script builds on the previous one by appending
# a call to spread(), which will allow us to turn the
# values of the test column, midterm and final, into
# column headers (i.e. variables).
#
# You only need to specify two arguments to spread().
# Can you figure out what they are? (Hint: You don't
# have to specify the data argument since we're using
# the %>% operator.
#
students3 %>%
  gather(class, grade, class1:class5, na.rm = TRUE) %>%
  spread(test, grade) %>%
  print
```

```
##    name  class final midterm
## 1 Brian class1     B       B
## 2 Brian class5     C       A
## 3  Jeff class2     E       D
## 4 Karen class3     C       C
## 5 Roger class2     A       C
## 6 Roger class5     A       B
## 7 Sally class1     C       A
## 8 Sally class3     C       B
```

```
library(readr)
```

```
## Warning: package 'readr' was built under R version 3.6.3
```

```
parse_number("class5")
```

```
## [1] 5
```

```
# We want the values in the class columns to be
# 1, 2, ..., 5 and not class1, class2, ..., class5.
#
# Use the mutate() function from dplyr along with
# parse_number(). Hint: You can "overwrite" a column
# with mutate() by assigning a new value to the existing
# column instead of creating a new column.
#
# Check out ?mutate and/or ?parse_number if you need
# a refresher.
#
students3 %>%
  gather(class, grade, class1:class5, na.rm = TRUE) %>%
  spread(test, grade) %>%
  mutate(class = parse_number(class)) %>%
  print
```

```
##    name class final midterm
## 1 Brian     1     B       B
## 2 Brian     5     C       A
## 3  Jeff     2     E       D
## 4 Karen     3     C       C
## 5 Roger     2     A       C
## 6 Roger     5     A       B
## 7 Sally     1     C       A
## 8 Sally     3     C       B
```

```
students4 <- data.frame(
  "id" = as.integer(c(168,168,588,588,710,710,731,731,908,908)),
  "name" = c("Brian","Brian","Sally","Sally","Jeff","Jeff","Roger","Roger","Karen","Karen"),
  "sex" = c("F","F","M","M","M","M","F","F","M","M"),
  "class" = as.integer(c(1,5,1,3,2,4,2,5,3,4)),
  "midterm" = c("B","A","A","B","D","A","C","B","C","A"),
  "final" = c("B","C","C","C","E","C","A","A","C","A"),
  stringsAsFactors = FALSE

)
students4
```

```
##     id  name sex class midterm final
## 1  168 Brian   F     1       B     B
## 2  168 Brian   F     5       A     C
## 3  588 Sally   M     1       A     C
## 4  588 Sally   M     3       B     C
## 5  710  Jeff   M     2       D     E
## 6  710  Jeff   M     4       A     C
## 7  731 Roger   F     2       C     A
## 8  731 Roger   F     5       B     A
## 9  908 Karen   M     3       C     C
## 10 908 Karen   M     4       A     A
```

```
# selecting the id, name, and sex column from students4
# and storing the result in student_info.
#
student_info <- students4 %>%
  select(id, name, sex) %>%
  print
```

```
##     id  name sex
## 1  168 Brian   F
## 2  168 Brian   F
## 3  588 Sally   M
## 4  588 Sally   M
## 5  710  Jeff   M
## 6  710  Jeff   M
## 7  731 Roger   F
## 8  731 Roger   F
## 9  908 Karen   M
## 10 908 Karen   M
```

```
# Add a call to unique() below, which will remove
# duplicate rows from student_info.
#
# Like with the call to the print() function below,
# you can omit the parentheses after the function name.
# This is a nice feature of %>% that applies when
# there are no additional arguments to specify.
#
student_info <- students4 %>%
  select(id, name, sex) %>%
  unique %>%
  print
```

```
##    id  name sex
## 1 168 Brian   F
## 3 588 Sally   M
## 5 710  Jeff   M
## 7 731 Roger   F
## 9 908 Karen   M
```

```
# select() the id, class, midterm, and final columns
# (in that order) and store the result in gradebook.
#
gradebook <- students4 %>%
  select(id, class, midterm, final) %>%
  print
```

```
##     id class midterm final
## 1  168     1       B     B
## 2  168     5       A     C
## 3  588     1       A     C
## 4  588     3       B     C
## 5  710     2       D     E
## 6  710     4       A     C
## 7  731     2       C     A
## 8  731     5       B     A
## 9  908     3       C     C
## 10 908     4       A     A
```

```
passed<- data.frame(
  "name" = c("Brian","Roger","Roger","Karen"),
  "class" = as.integer(c(1,2,5,4)),
  "final" = c("B","A","A","A"),
  stringsAsFactors = FALSE
  )
failed <- data.frame(
  "name" = c("Brian","Sally","Sally","Jeff","Jeff","Karen"),
  "class" = as.integer(c(5,1,3,2,4,3)),
  "final" = c("C","C","C","E","C","C"),
   stringsAsFactors = FALSE
)
passed
```

```
##    name class final
## 1 Brian     1     B
## 2 Roger     2     A
## 3 Roger     5     A
## 4 Karen     4     A
```

```
failed
```

```
##    name class final
## 1 Brian     5     C
## 2 Sally     1     C
## 3 Sally     3     C
## 4  Jeff     2     E
## 5  Jeff     4     C
## 6 Karen     3     C
```

```
passed <- passed %>% mutate(status = "passed")
failed <- failed %>% mutate(status = "failed")
bind_rows(passed, failed)
```

```
##     name class final status
## 1  Brian     1     B passed
## 2  Roger     2     A passed
## 3  Roger     5     A passed
## 4  Karen     4     A passed
## 5  Brian     5     C failed
## 6  Sally     1     C failed
## 7  Sally     3     C failed
## 8   Jeff     2     E failed
## 9   Jeff     4     C failed
## 10 Karen     3     C failed
```

4. Dates and Times with lubridate

```
library(lubridate)
```

```
## Warning: package 'lubridate' was built under R version 3.6.3
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

```
this_day <- today()
this_day
```

```
## [1] "2020-05-15"
```

```
month(this_day)
```

```
## [1] 5
```

```r
wday(this_day)
```

```
## [1] 6
```

```r
wday(this_day, label = TRUE)
```

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

```r
this_moment <- now()
this_moment
```

```
## [1] "2020-05-15 17:59:16 IST"
```

```r
second(this_moment)
```

```
## [1] 16.8503
```

```r
my_date <- ymd("1989-05-17")
my_date
```

```
## [1] "1989-05-17"
```

```r
class(my_date)
```

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

```r
ymd("1989 May 17")
```

```
## [1] "1989-05-17"
```

```r
mdy("March 12, 1975")
```

```
## [1] "1975-03-12"
```

```r
dmy(25081985)
```

```
## [1] "1985-08-25"
```

```r
ymd("1920/1/2")
```

```
## [1] "1920-01-02"
```

```r
dt1 <- "2014-08-23 17:23:02"
ymd_hms(dt1)
```

```
## [1] "2014-08-23 17:23:02 UTC"
```

```r
hms("03:22:14")
```

```
## [1] "3H 22M 14S"
```

```r
dt2<-c("2014-05-14","2014-09-22","2014-07-11")
ymd(dt2)
```

```
## [1] "2014-05-14" "2014-09-22" "2014-07-11"
```

```r
update(this_moment, hours = 8, minutes = 34, seconds = 55)
```

```
## [1] "2020-05-15 08:34:55 IST"
```

```r
this_moment
```

```
## [1] "2020-05-15 17:59:16 IST"
```

```r
this_moment <- update(this_moment, hours = 10, minutes = 16, seconds = 0)
nyc <- now("America/New_York")
depart <- nyc + days(2)
depart <- update(depart, hours = 17, minutes = 34)
arrive <- depart + hours(15) + minutes(50)
arrive <- with_tz(arrive, "Asia/Hong_Kong")
last_time <- mdy("June 17, 2008", tz = "Singapore")
how_long <- interval(last_time, arrive)
as.period(how_long)
```

```
## [1] "11y 11m 1d 21H 24M 16S"
```