Tidy Data Lab

Jose Bravo

March 13, 2017

Manipulating Data with dplyr

Loading the data provided by swirl.

```
##
## 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

## Warning: package 'lubridate' was built under R version 3.3.3

##
## Attaching package: 'lubridate'

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

```
#List dimensions of the dataframe and preview it.

dim(mydf)
```

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

```
head(mydf)
```

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

```
#Create a data frame tbl
cran <- tbl_df(mydf)
```

```
#The five 'verbs' that cover most fundamental data manipulation
select(cran, ip_id, package, country)
```

```
## # A tibble: 225,468 × 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
```

```
#print the variables r_arch to county
select(cran, r_arch:country)
```

```
## # A tibble: 225,468 × 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
```

```
# or in reverse
select(cran, country:r_arch)
```

```
## # A tibble: 225,468 × 5
##    country version      package      r_os r_arch
##      <chr>   <chr>        <chr>     <chr>  <chr>
## 1       US   0.2.4    htmltools   mingw32 x86_64
## 2       US 0.10-32      tseries   mingw32 x86_64
## 3       US  1.0-15        party linux-gnu x86_64
## 4       US  3.14-4        Hmisc linux-gnu x86_64
## 5       CA   0.6.4       digest linux-gnu x86_64
## 6       US   4.6-7 randomForest linux-gnu x86_64
```

```
## 7       US   1.8.1         plyr linux-gnu x86_64
## 8       US   0.3-2      whisker linux-gnu x86_64
## 9       CN  0.10.4         Rcpp      <NA>   <NA>
## 10      US     0.1     hflights linux-gnu x86_64
## # ... with 225,458 more rows
```

```
# print all the variables in the dataframe except time
select(cran, -time)
```

```
## # A tibble: 225,468 × 10
##        X       date    size r_version r_arch      r_os      package
##    <int>      <chr>   <int>     <chr>  <chr>     <chr>        <chr>
## 1      1 2014-07-08   80589     3.1.0 x86_64   mingw32    htmltools
## 2      2 2014-07-08  321767     3.1.0 x86_64   mingw32      tseries
## 3      3 2014-07-08  748063     3.1.0 x86_64 linux-gnu        party
## 4      4 2014-07-08  606104     3.1.0 x86_64 linux-gnu        Hmisc
## 5      5 2014-07-08   79825     3.0.2 x86_64 linux-gnu       digest
## 6      6 2014-07-08   77681     3.1.0 x86_64 linux-gnu randomForest
## 7      7 2014-07-08  393754     3.1.0 x86_64 linux-gnu         plyr
## 8      8 2014-07-08   28216     3.0.2 x86_64 linux-gnu      whisker
## 9      9 2014-07-08    5928      <NA>   <NA>      <NA>         Rcpp
## 10    10 2014-07-08 2206029     3.0.2 x86_64 linux-gnu     hflights
## # ... with 225,458 more rows, and 3 more variables: version <chr>,
## #   country <chr>, ip_id <int>
```

```
# omit all variables from X to size
select(cran, -(X:size))
```

```
## # A tibble: 225,468 × 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() to select subset of rows
filter(cran, package == "swirl")
```

```
## # A tibble: 820 × 11
##        X       date     time   size r_version r_arch         r_os package
##    <int>      <chr>    <chr>  <int>     <chr>  <chr>        <chr>   <chr>
## 1     27 2014-07-08 00:17:16 105350     3.0.2 x86_64      mingw32   swirl
## 2    156 2014-07-08 00:22:53  41261     3.1.0 x86_64    linux-gnu   swirl
## 3    358 2014-07-08 00:13:42 105335    2.15.2 x86_64      mingw32   swirl
```

```
## 4    593 2014-07-08 00:59:45 105465     3.1.0 x86_64 darwin13.1.0   swirl
## 5    831 2014-07-08 00:55:27 105335     3.0.3 x86_64      mingw32   swirl
## 6    997 2014-07-08 00:33:06  41261     3.1.0 x86_64      mingw32   swirl
## 7   1023 2014-07-08 00:35:36 106393     3.1.0 x86_64      mingw32   swirl
## 8   1144 2014-07-08 00:00:39 106534     3.0.2 x86_64    linux-gnu   swirl
## 9   1402 2014-07-08 00:41:41  41261     3.1.0   i386      mingw32   swirl
## 10  1424 2014-07-08 00:44:49 106393     3.1.0 x86_64    linux-gnu   swirl
## # ... with 810 more rows, and 3 more variables: version <chr>,
## #   country <chr>, ip_id <int>
```

```
# You can specify many conditions
filter(cran, r_version == "3.1.1", country == "US")
```

```
## # A tibble: 1,588 × 11
##        X       date     time    size r_version r_arch         r_os
##    <int>      <chr>    <chr>   <int>     <chr>  <chr>        <chr>
## 1   2216 2014-07-08 00:48:58  385112     3.1.1 x86_64 darwin13.1.0
## 2  17332 2014-07-08 03:39:57  197459     3.1.1 x86_64 darwin13.1.0
## 3  17465 2014-07-08 03:25:38   23259     3.1.1 x86_64 darwin13.1.0
## 4  18844 2014-07-08 03:59:17  190594     3.1.1 x86_64 darwin13.1.0
## 5  30182 2014-07-08 04:13:15   77683     3.1.1   i386      mingw32
## 6  30193 2014-07-08 04:06:26 2351969     3.1.1   i386      mingw32
## 7  30195 2014-07-08 04:07:09  299080     3.1.1   i386      mingw32
## 8  30217 2014-07-08 04:32:04  568036     3.1.1   i386      mingw32
## 9  30245 2014-07-08 04:10:41  526858     3.1.1   i386      mingw32
## 10 30354 2014-07-08 04:32:51 1763717     3.1.1   i386      mingw32
## # ... with 1,578 more rows, and 4 more variables: package <chr>,
## #   version <chr>, country <chr>, ip_id <int>
```

```
# Alter the previous command but specify India, and r versions less than 3.0.8
filter(cran, r_version <= "3.0.2", country == "IN")
```

```
## # A tibble: 4,139 × 11
##        X       date     time     size r_version r_arch      r_os
##    <int>      <chr>    <chr>    <int>     <chr>  <chr>     <chr>
## 1    348 2014-07-08 00:44:04 10218907     3.0.0 x86_64   mingw32
## 2   9990 2014-07-08 02:11:32   397497     3.0.2 x86_64 linux-gnu
## 3   9991 2014-07-08 02:11:32   119199     3.0.2 x86_64 linux-gnu
## 4   9992 2014-07-08 02:11:33    81779     3.0.2 x86_64 linux-gnu
## 5  10022 2014-07-08 02:19:45  1557078    2.15.0 x86_64   mingw32
## 6  10023 2014-07-08 02:19:46  1184285    2.15.1   i686 linux-gnu
## 7  10189 2014-07-08 02:38:06   908854     3.0.2 x86_64 linux-gnu
## 8  10199 2014-07-08 02:38:28   178436     3.0.2 x86_64 linux-gnu
## 9  10200 2014-07-08 02:38:29    51811     3.0.2 x86_64 linux-gnu
## 10 10201 2014-07-08 02:38:29    65245     3.0.2 x86_64 linux-gnu
## # ... with 4,129 more rows, and 4 more variables: package <chr>,
## #   version <chr>, country <chr>, ip_id <int>
```

```
# Filter can also take boolean operators
filter(cran, country == "US" | country == "IN")
```

```
## # A tibble: 95,283 × 11
```

```
##        X       date     time    size r_version r_arch      r_os
##    <int>      <chr>    <chr>   <int>     <chr>  <chr>     <chr>
## 1      1 2014-07-08 00:54:41   80589     3.1.0 x86_64   mingw32
## 2      2 2014-07-08 00:59:53  321767     3.1.0 x86_64   mingw32
## 3      3 2014-07-08 00:47:13  748063     3.1.0 x86_64 linux-gnu
## 4      4 2014-07-08 00:48:05  606104     3.1.0 x86_64 linux-gnu
## 5      6 2014-07-08 00:48:04   77681     3.1.0 x86_64 linux-gnu
## 6      7 2014-07-08 00:48:35  393754     3.1.0 x86_64 linux-gnu
## 7      8 2014-07-08 00:47:30   28216     3.0.2 x86_64 linux-gnu
## 8     10 2014-07-08 00:15:35 2206029     3.0.2 x86_64 linux-gnu
## 9     11 2014-07-08 00:15:25  526858     3.0.2 x86_64 linux-gnu
## 10    12 2014-07-08 00:14:45 2351969    2.14.1 x86_64 linux-gnu
## # ... with 95,273 more rows, and 4 more variables: package <chr>,
## #   version <chr>, country <chr>, ip_id <int>
```

```
# can also filter numeric values
filter(cran, size > 100500, r_os == "linux-gnu")
```

```
## # A tibble: 33,683 × 11
##        X       date     time    size r_version r_arch      r_os  package
##    <int>      <chr>    <chr>   <int>     <chr>  <chr>     <chr>    <chr>
## 1      3 2014-07-08 00:47:13  748063     3.1.0 x86_64 linux-gnu    party
## 2      4 2014-07-08 00:48:05  606104     3.1.0 x86_64 linux-gnu    Hmisc
## 3      7 2014-07-08 00:48:35  393754     3.1.0 x86_64 linux-gnu     plyr
## 4     10 2014-07-08 00:15:35 2206029     3.0.2 x86_64 linux-gnu hflights
## 5     11 2014-07-08 00:15:25  526858     3.0.2 x86_64 linux-gnu     LPCM
## 6     12 2014-07-08 00:14:45 2351969    2.14.1 x86_64 linux-gnu  ggplot2
## 7     14 2014-07-08 00:15:35 3097729     3.0.2 x86_64 linux-gnu     Rcpp
## 8     15 2014-07-08 00:14:37  568036     3.1.0 x86_64 linux-gnu    rJava
## 9     16 2014-07-08 00:15:50 1600441     3.1.0 x86_64 linux-gnu  RSQLite
## 10    18 2014-07-08 00:26:59  186685     3.1.0 x86_64 linux-gnu    ipred
## # ... with 33,673 more rows, and 3 more variables: version <chr>,
## #   country <chr>, ip_id <int>
```

```
# filter out NAs from the r_version variable
filter(cran, !is.na(r_version))
```

```
## # A tibble: 207,205 × 11
##        X       date     time    size r_version r_arch      r_os
##    <int>      <chr>    <chr>   <int>     <chr>  <chr>     <chr>
## 1      1 2014-07-08 00:54:41   80589     3.1.0 x86_64   mingw32
## 2      2 2014-07-08 00:59:53  321767     3.1.0 x86_64   mingw32
## 3      3 2014-07-08 00:47:13  748063     3.1.0 x86_64 linux-gnu
## 4      4 2014-07-08 00:48:05  606104     3.1.0 x86_64 linux-gnu
## 5      5 2014-07-08 00:46:50   79825     3.0.2 x86_64 linux-gnu
## 6      6 2014-07-08 00:48:04   77681     3.1.0 x86_64 linux-gnu
## 7      7 2014-07-08 00:48:35  393754     3.1.0 x86_64 linux-gnu
## 8      8 2014-07-08 00:47:30   28216     3.0.2 x86_64 linux-gnu
## 9     10 2014-07-08 00:15:35 2206029     3.0.2 x86_64 linux-gnu
## 10    11 2014-07-08 00:15:25  526858     3.0.2 x86_64 linux-gnu
## # ... with 207,195 more rows, and 4 more variables: package <chr>,
## #   version <chr>, country <chr>, ip_id <int>
```

```
#using arrange()
arrange(cran2, ip_id)
```

```
## # A tibble: 225,468 × 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
```

```
# now in descending order
arrange(cran2, desc(ip_id))
```

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

```
# arrange multiple variables
arrange(cran2, country, desc(r_version), ip_id)
```

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

```
# Using the mutate() function to add additional column
mutate(cran3, size_mb = size / 2^20)
```

```
## # A tibble: 225,468 × 4
##    ip_id      package    size     size_mb
##    <int>        <chr>   <int>       <dbl>
## 1      1    htmltools   80589 0.076855659
## 2      2      tseries  321767 0.306860924
## 3      3        party  748063 0.713408470
## 4      3        Hmisc  606104 0.578025818
## 5      4       digest   79825 0.076127052
## 6      3 randomForest   77681 0.074082375
## 7      3         plyr  393754 0.375513077
## 8      5      whisker   28216 0.026908875
## 9      6         Rcpp    5928 0.005653381
## 10     7     hflights 2206029 2.103833199
## # ... with 225,458 more rows
```

```
# add one more column of size_gb
mutate(cran3, size_mb = size / 2^20, size_gb = size_mb / 2^10)
```

```
## # A tibble: 225,468 × 5
##    ip_id      package    size     size_mb      size_gb
##    <int>        <chr>   <int>       <dbl>        <dbl>
## 1      1    htmltools   80589 0.076855659 7.505435e-05
## 2      2      tseries  321767 0.306860924 2.996689e-04
## 3      3        party  748063 0.713408470 6.966880e-04
## 4      3        Hmisc  606104 0.578025818 5.644783e-04
## 5      4       digest   79825 0.076127052 7.434282e-05
## 6      3 randomForest   77681 0.074082375 7.234607e-05
## 7      3         plyr  393754 0.375513077 3.667120e-04
## 8      5      whisker   28216 0.026908875 2.627820e-05
## 9      6         Rcpp    5928 0.005653381 5.520880e-06
## 10     7     hflights 2206029 2.103833199 2.054525e-03
## # ... with 225,458 more rows
```

```
# add another column with the correct size of the package
mutate(cran3, correct_size = size + 1000)
```

```
## # A tibble: 225,468 × 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
```

Grouping and Chaining with dplyr

```
#Using group_by function
by_package <- group_by(cran, package)
# summarize by mean(size)
summarize(by_package, mean(size))
```

```
## # A tibble: 6,023 × 2
##        package `mean(size)`
##          <chr>        <dbl>
## 1           A3     62194.96
## 2          abc   4826665.00
## 3     abcdeFBA    455979.87
## 4  ABCExtremes     22904.33
## 5     ABCoptim     17807.25
## 6        ABCp2     30473.33
## 7     abctools   2589394.00
## 8          abd    453631.24
## 9         abf2     35692.62
## 10       abind     32938.88
## # ... with 6,013 more rows
```

```
# modifying swirl script
pack_sum <- summarize(by_package, count = n(), unique = n_distinct(ip_id), countries = n_distinct(counti


# print pack_sum
pack_sum
```

```
## # A tibble: 6,023 × 5
##        package count unique countries  avg_bytes
##          <chr> <int>  <int>     <int>      <dbl>
## 1           A3    25     24        10   62194.96
## 2          abc    29     25        16 4826665.00
## 3     abcdeFBA    15     15         9  455979.87
## 4  ABCExtremes    18     17         9   22904.33
## 5     ABCoptim    16     15         9   17807.25
## 6        ABCp2    18     17        10   30473.33
## 7     abctools    19     19        11 2589394.00
## 8          abd    17     16        10  453631.24
## 9         abf2    13     13         9   35692.62
## 10       abind   396    365        50   32938.88
## # ... with 6,013 more rows
```

```
# print the one percentile of the pack_sum
quantile(pack_sum$count, probs = 0.99)
```

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

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

```
## # A tibble: 61 × 5
##       package count unique countries   avg_bytes
##         <chr> <int>  <int>     <int>       <dbl>
## 1      bitops  1549   1408        76   28715.046
## 2         car  1008    837        64 1229122.307
## 3     caTools   812    699        64  176589.018
## 4  colorspace  1683   1433        80  357411.197
## 5  data.table   680    564        59 1252721.215
## 6         DBI  2599    492        48  206933.250
## 7    devtools   769    560        55  212932.640
## 8   dichromat  1486   1257        74  134731.938
## 9      digest  2210   1894        83  120549.294
## 10     doSNOW   740     75        24    8363.755
## # ... with 51 more rows
```

```
# Using arrange to sort top_counts into new variable top_counts_sorted
top_counts_sorted <- arrange(top_counts, desc(count))

#view the data
View(top_counts_sorted)

# Apply filter to pack_sum to select all rows corresponding to values of 'unique' that are strictly gre
top_unique <- filter(pack_sum, unique > 465)

#Now arrange() top_unique by the 'unique' column, in descending order
top_unique_sorted <- arrange(top_unique, desc(unique))
```

```
# The following code is completing the scripts that Swirl() has generated for the user to complete.
# chain1.R completed
cran %>%
  select(ip_id, country, package, size) %>%
    print
```

```
## # A tibble: 225,468 × 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
```

```
# chain2.R completed
cran %>%
  select(ip_id, country, package, size) %>%
  mutate(size_mb = size / 2^20)
```

```
## # A tibble: 225,468 × 5
##    ip_id country      package    size     size_mb
##    <int>   <chr>        <chr>   <int>       <dbl>
## 1      1      US    htmltools   80589 0.076855659
## 2      2      US      tseries  321767 0.306860924
## 3      3      US        party  748063 0.713408470
## 4      3      US        Hmisc  606104 0.578025818
## 5      4      CA       digest   79825 0.076127052
## 6      3      US randomForest   77681 0.074082375
## 7      3      US         plyr  393754 0.375513077
## 8      5      US      whisker   28216 0.026908875
## 9      6      CN         Rcpp    5928 0.005653381
## 10     7      US     hflights 2206029 2.103833199
## # ... with 225,458 more rows
```

```
# chain3.R completed
cran %>%
  select(ip_id, country, package, size) %>%
  mutate(size_mb = size / 2^20) %>%
  # Your call to filter() goes here
  filter(size_mb <= 0.5)
```

```
## # A tibble: 142,021 × 5
##    ip_id country      package   size     size_mb
##    <int>   <chr>        <chr>  <int>       <dbl>
## 1      1      US    htmltools  80589 0.076855659
## 2      2      US      tseries 321767 0.306860924
## 3      4      CA       digest  79825 0.076127052
## 4      3      US randomForest  77681 0.074082375
## 5      3      US         plyr 393754 0.375513077
## 6      5      US      whisker  28216 0.026908875
## 7      6      CN         Rcpp   5928 0.005653381
## 8     13      DE        ipred 186685 0.178036690
## 9     14      US       mnormt  36204 0.034526825
## 10    16      US    iterators 289972 0.276538849
## # ... with 142,011 more rows
```

```
# chain4.R
cran %>%
  select(ip_id, country, package, size) %>%
  mutate(size_mb = size / 2^20) %>%
  filter(size_mb <= 0.5) %>%
  # Your call to arrange() goes here
  arrange(desc(size_mb))
```

```
## # A tibble: 142,021 × 5
##    ip_id country              package   size   size_mb
```

```
##    <int>   <chr>                 <chr>  <int>     <dbl>
## 1  11034      DE                  phia 524232 0.4999466
## 2   9643      US                   tis 524152 0.4998703
## 3   1542      IN               RcppSMC 524060 0.4997826
## 4  12354      US                 lessR 523916 0.4996452
## 5  12072      US            colorspace 523880 0.4996109
## 6   2514      KR              depmixS4 523863 0.4995947
## 7   1111      US              depmixS4 523858 0.4995899
## 8   8865      CR              depmixS4 523858 0.4995899
## 9   5908      CN RcmdrPlugin.KMggplot2 523852 0.4995842
## 10 12354      US RcmdrPlugin.KMggplot2 523852 0.4995842
## # ... with 142,011 more rows
```

Tidying Data with tidyr

```
# unfortunately, these courses are bugged within markdown, so I have to create the dataframes myself
grade <- c("A","B","C","D","E")
male <- c(1,5,5,5,7)
female <- c(5,0,2,5,4)
students <- data.frame(grade, male, female)
# Print the data frame students
students
```

```
##   grade male female
## 1     A    1      5
## 2     B    5      0
## 3     C    5      2
## 4     D    5      5
## 5     E    7      4
```

```
# call gather() with the following arguments (in order):students, sex, count, -grade
gather(students, sex, count, -grade)
```

```
##    grade    sex count
## 1      A   male     1
## 2      B   male     5
## 3      C   male     5
## 4      D   male     5
## 5      E   male     7
## 6      A female     5
## 7      B female     0
## 8      C female     2
## 9      D female     5
## 10     E female     4
```

```
male_1 <- c(3,6,7,4,1)
female_1 <- c(4,4,4,0,1)
male_2 <- c(3,3,3,8,2)
female_2 <- c(4,5,8,1,7)
students2 <- data.frame(grade, male_1, female_1, male_2, female_2)
```

```
# Print students2
students2
```

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

```
#Call gather() with the following arguments (in order): students2, sex_class, count, -grade). Store the
res <- gather(students2, sex_class, count, -grade)
res
```

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

```
# Call separate() on res to split the sex_class column into sex and class.
separate(res, col = sex_class, into = c("sex", "class"))
```

```
##    grade    sex class count
## 1      A   male     1     3
## 2      B   male     1     6
## 3      C   male     1     7
## 4      D   male     1     4
## 5      E   male     1     1
## 6      A female     1     4
## 7      B female     1     4
## 8      C female     1     4
## 9      D female     1     0
## 10     E female     1     1
## 11     A   male     2     3
## 12     B   male     2     3
```

```
## 13     C   male     2     3
## 14     D   male     2     8
## 15     E   male     2     2
## 16     A female     2     4
## 17     B female     2     5
## 18     C female     2     8
## 19     D female     2     1
## 20     E female     2     7
```

```
# Completing the generated Swirl() scripts.
# script1.R complete
students2 %>%
  gather(sex_class ,count ,-grade ) %>%
  separate( col = sex_class, into = c("sex", "class")) %>%
  print
```

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

```
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)
class4 <- c(NA,NA,"A","C",NA,NA,"A","A",NA,NA)
class5 <- c(NA,NA,NA,NA,"B","A",NA,NA,"A","C")
students3 <- data.frame(name, test,class1,class2,class3,class4,class5)

# print students3
students3
```

```
##     name    test class1 class2 class3 class4 class5
## 1  Sally midterm      A   <NA>      B   <NA>   <NA>
## 2  Sally   final      C   <NA>      C   <NA>   <NA>
```

```
## 3   Jeff midterm   <NA>      D   <NA>      A   <NA>
## 4   Jeff   final   <NA>      E   <NA>      C   <NA>
## 5  Roger midterm   <NA>      C   <NA>   <NA>      B
## 6  Roger   final   <NA>      A   <NA>   <NA>      A
## 7  Karen midterm   <NA>   <NA>      C      A   <NA>
## 8  Karen   final   <NA>   <NA>      C      A   <NA>
## 9  Brian midterm      B   <NA>   <NA>   <NA>      A
## 10 Brian   final      B   <NA>   <NA>   <NA>      C
```

```
# script2.R complete
students3 %>%
  gather( class,grade , class1:class5 , na.rm= TRUE) %>%
  print
```

```
## Warning: attributes are not identical across measure variables; they will
## be dropped
```

```
##     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
## 33  Jeff midterm class4     A
## 34  Jeff   final class4     C
## 37 Karen midterm class4     A
## 38 Karen   final class4     A
## 45 Roger midterm class5     B
## 46 Roger   final class5     A
## 49 Brian midterm class5     A
## 50 Brian   final class5     C
```

```
# script3.R complete
# 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).
students3 %>%
  gather(class, grade, class1:class5, na.rm = TRUE) %>%
  spread( test, grade ) %>%
  print
```

```
## Warning: attributes are not identical across measure variables; they will
## be dropped
```

```
##     name  class final midterm
```

```
## 1  Brian class1     B       B
## 2  Brian class5     C       A
## 3   Jeff class2     E       D
## 4   Jeff class4     C       A
## 5  Karen class3     C       C
## 6  Karen class4     A       A
## 7  Roger class2     A       C
## 8  Roger class5     A       B
## 9  Sally class1     C       A
## 10 Sally class3     C       B
```

```
# 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.
# script4.R complete
students3 %>%
  gather(class, grade, class1:class5, na.rm = TRUE) %>%
  spread(test, grade) %>%
  mutate(class = parse_number(class)) %>%
  print
```

```
## Warning: attributes are not identical across measure variables; they will
## be dropped
```

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

```
#
id <- 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 <- 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")
students4 <- data.frame(id,name,sex,class,midterm, final)

#
# script5.R completed
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
```

```
# script6.R completed
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
```

```
# script7.R completed
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
```

```
# Again, I have to create the dataframes myself because markdown and swirl variables don't play nice
name <- c("Brian","Roger","Roger","Karen")
class <- c(1,2,5,4)
final <- c("B","A","A","A")
passed <- data.frame(name,class,final)

name <- c("Brian","Sally","Sally","Jeff","Jeff","Karen")
class <- c(5,1,3,2,4,3)
final <- c("C","C","C","E","C","C")
failed <- data.frame(name,class,final)
```

```
#Use dplyr's mutate() to add a new column to the passed table.
passed <- passed %>% mutate(status = "passed")
# Now, do the same for the failed table
failed <- failed %>% mutate(status = "failed")

#Call bind_rows() with two arguments, passed and failed (in that order), to join the two tables.
bind_rows(passed, failed)
```

```
## Warning in bind_rows_(x, .id): Unequal factor levels: coercing to character

## Warning in bind_rows_(x, .id): Unequal factor levels: coercing to character

##     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
```

```
##
score_range <- c("700-800","600-690","500-590","400-490","300-390","200-290")
read_male <- c(40151,121950,227141,242554,113568,30728)
read_fem <- c(38898, 126084, 259553,296793,133473,29154)
read_total <- c(79049, 248034, 486694, 539347, 247041, 59882)
math_male <- c(74461, 162564, 233141,204670, 82468,18788)
math_fem <- c(46040, 133954, 257678,288696,131025,26562)
math_total <- c(120501,296518,490819,493366,213493,25350)
write_male <- c(31574, 100963, 202326, 262623, 146106, 32500)
write_fem <- c(39101, 125368, 247239, 302933, 144381, 24933)
write_total <- c(70675, 226331, 449565, 565556, 290481, 57433)
sat <- data.frame(score_range,read_male,read_fem, read_total, math_male,math_fem, math_total, write_mal

##

# script8.R completed
sat %>%
  select(-contains("total")) %>%
  gather(part_sex, count, -score_range) %>%
  separate(part_sex, c("part","sex")) %>%
  print
```

```
##    score_range  part  sex  count
## 1      700-800  read male  40151
## 2      600-690  read male 121950
## 3      500-590  read male 227141
## 4      400-490  read male 242554
```

```
## 5      300-390  read male 113568
## 6      200-290  read male  30728
## 7      700-800  read  fem  38898
## 8      600-690  read  fem 126084
## 9      500-590  read  fem 259553
## 10     400-490  read  fem 296793
## 11     300-390  read  fem 133473
## 12     200-290  read  fem  29154
## 13     700-800  math male  74461
## 14     600-690  math male 162564
## 15     500-590  math male 233141
## 16     400-490  math male 204670
## 17     300-390  math male  82468
## 18     200-290  math male  18788
## 19     700-800  math  fem  46040
## 20     600-690  math  fem 133954
## 21     500-590  math  fem 257678
## 22     400-490  math  fem 288696
## 23     300-390  math  fem 131025
## 24     200-290  math  fem  26562
## 25     700-800 write male  31574
## 26     600-690 write male 100963
## 27     500-590 write male 202326
## 28     400-490 write male 262623
## 29     300-390 write male 146106
## 30     200-290 write male  32500
## 31     700-800 write  fem  39101
## 32     600-690 write  fem 125368
## 33     500-590 write  fem 247239
## 34     400-490 write  fem 302933
## 35     300-390 write  fem 144381
## 36     200-290 write  fem  24933
```

```
# script9.R completed
sat %>%
  select(-contains("total")) %>%
  gather(part_sex, count, -score_range) %>%
  separate(part_sex, c("part", "sex")) %>%
  group_by(part,sex) %>%
  mutate(total = sum(count),
         prop = count / total
  ) %>% print
```

```
## Source: local data frame [36 x 6]
## Groups: part, sex [6]
##
##    score_range  part   sex  count  total       prop
##          <fctr> <chr> <chr>  <dbl>  <dbl>      <dbl>
## 1      700-800  read  male  40151 776092 0.05173485
## 2      600-690  read  male 121950 776092 0.15713343
## 3      500-590  read  male 227141 776092 0.29267278
## 4      400-490  read  male 242554 776092 0.31253253
## 5      300-390  read  male 113568 776092 0.14633317
## 6      200-290  read  male  30728 776092 0.03959324
```

```
## 7      700-800  read   fem  38898 883955 0.04400450
## 8      600-690  read   fem 126084 883955 0.14263622
## 9      500-590  read   fem 259553 883955 0.29362694
## 10     400-490  read   fem 296793 883955 0.33575578
## # ... with 26 more rows
```

Dates and Times with lubridate

```
# print today() and put it in variable this_day
this_day <- today()

# use other time functions
year(this_day)
```

```
## [1] 2017
```

```
wday(this_day)
```

```
## [1] 4
```

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

```
## [1] Wed
## Levels: Sun < Mon < Tues < Wed < Thurs < Fri < Sat
```

```
# lubridate also records date and time combinations
this_moment <- now()
hour(this_moment)
```

```
## [1] 17
```

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

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

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

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

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

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

```
dmy(25081985)
```

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

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

```
## [1] "2017-03-15 08:34:55 EDT"
```

```
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] "8y 9m 1d 21H 24M 22.6667380332947S"
```