Week 3: Tutorial on Reading and Wrangling Data

Ben Best

2016-01-22 07:26

Contents

Github Workflows Recap

- "conversation" of code amongst us with pull requests & issues, with every change logged in the version history
- flows:

1. **fork** and **pull request**: *fork*, clone, pull, (branch,) commit, push and *pull request*
   - ***read only*** (no write) permissions on original repository
   - eg `bren-ucsb/env-info` *fork* to `bbest/env-info`
   - ie `<org>/<repo>` *fork* to `<user>/<repo>`
   - to update:
     * *pull request* `<user>/<repo>` -> `<org>/<repo>`, or
     * *pull request* `<org>/<repo>` -> `<user>/<repo>`
2. **pull** and **push**: clone, pull, (branch,) commit and push
   - read and ***write*** permissions on original repository
   - eg `bbest` *push* directly to `whaleroute/whaleroute.github.io`
   - ie `<user>` *push* directly to `<org>/<repo>`
   - see Github Flow for branching model

Assignment

For the individual assignment today, you'll need to first pull

Where am I? Getting around in the Command Line

Knowing your present working directory is critical to using "relative" paths, ie relative to your present working directory. Relative paths (eg `somedir/somefile.csv` are often preferred over "absolute" paths (eg `C:/somedir/somefile.csv`) since the project's root folder can move around on the machine or even to a different machine and still work, whereas an absolute path is locked down to a very exact machine-specific path. Here are a couple of aspects to keep in mind however when knitting Rmarkdown (*.Rmd) files:

- When you open an RStudio project, the default present working directory is the top level folder for that project (and contains the `*.Rproj` file).
- When you "Knit" an Rmarkdown file (`*.Rmd`), the working directory is set to the folder containing the `*.Rmd` and a new workspace is used.

The above differences mean that when writing chunks of R code, a path can work in the Console and fail when you go to "Knit" the Rmarkdown file (`*.Rmd`), or vice versa.

So let's review some basic commands for navigating directories in both shell commands and R commands.

Bash Shell

The bash shell is the most common Unix-based command shell, found in Linux and Mac machines. It gets emulated for Windows in the Git Bash Shell application when installing git. Natively, Windows uses the less powerful Windows DOS command prompt, which uses `cd` (for `pwd` and `cd`) and `dir` (instead of `ls`).

```
# present working directory
pwd

# change working directory
cd

# list files
ls

# list files that end in '.jpg'
ls *.jpg
```

Note the use of the wildcard `*` to indicate any set of characters.

R

Now play with the same commands commented above, but in R.

```
# present working directory
getwd()

# change working directory
```

```
setwd('.')

# list files
list.files()

# list files that end in '.jpg'
list.files(pattern=glob2rx('*.jpg'))

# file exists
file.exists('test.png')
```

Look at the help for `list.files()` (`?list.files` or F1 with cursor over `list.files()` in editing window) to see that the `pattern` argument expects a regular expression and `glob2rx()` changes the wildcard or globbing pattern into a regular expression.

Install Packages

```
# Run this chunk only once in your Console
# Do not evaluate when knitting Rmarkdown

# list of packages
pkgs = c(
  'readr',        # read csv
  'readxl',       # read xls
  'dplyr',        # data frame manipulation
  'tidyr',        # data tidying
  'nycflights13', # test dataset of NYC flights for 2013
  'gapminder')    # test dataset of life expectancy and popultion

# install packages if not found
for (p in pkgs){
  if (!require(p, character.only=T)){
    install.packages(p)
  }
}
```

The **gapminder** dataset is "an excerpt of the data available at Gapminder.org. For each of 142 countries, the package provides values for life expectancy, GDP per capita, and population, every five years, from 1952 to 2007" (CRAN). Gapminder was the brain child of Hans Rosling who famously gave the TED Talk: The best stats you've ever seen - Hans Rosling.

Readings

These are the main R packages we'll be learning about this week:

- `readr`: column types
- `dplyr`: introduction
- `tidyr`: tidy data
- `dplyr` & `tidyr`: data wrangling cheatsheet

Reading CSV

`utils::read.csv`

Traditionally, you would read a CSV like so:

`readr::read_csv`

- hell inferno

Get CO_2 Emissions

```
library(readxl)
url = 'http://edgar.jrc.ec.europa.eu/news_docs/CO2_1970-2014_dataset_of_CO2_report_2015.xls'
xls = 'data/CO2_1970-2014_dataset_of_CO2_report_2015.xls'

if (!file.exists(xls)){
  download.file(url, xls)
}
d = read_excel(xls, skip=12)
```

```
## DEFINEDNAME: 21 00 00 01 0b 00 00 00 01 00 00 00 00 00 00 0d 3b 00 00 0c 00 e0 00 00 00 2c 00
## DEFINEDNAME: 21 00 00 01 0b 00 00 00 01 00 00 00 00 00 00 0d 3b 00 00 0c 00 e0 00 00 00 2c 00
## DEFINEDNAME: 21 00 00 01 0b 00 00 00 01 00 00 00 00 00 00 0d 3b 00 00 0c 00 e0 00 00 00 2c 00
## DEFINEDNAME: 21 00 00 01 0b 00 00 00 01 00 00 00 00 00 00 0d 3b 00 00 0c 00 e0 00 00 00 2c 00
```

CO2 time series per region/country | EUROPA

- join with population over time. get at emissions per capita
- rank worst 10
- get at cumulative emissions
- join with GDP. fit model, who excessively bad/goodget?
- references relevant to UCOP21:
- United Nations Framework Convention on Climate Change - Wikipedia, the free encyclopedia
- 6 Graphs Explain the World's Top 10 Emitters | World Resources Institute
- Global Emissions | Climate Change | US EPA
- just USA

nycflights13

```
library(dplyr)
```

```
##
## 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
```

```
library(nycflights13)

# Convert data to a tbl_df so that it uses dplyr's nice print method
flights <- tbl_df(flights)
flights
```

```
## Source: local data frame [336,776 x 16]
##
##     year month   day dep_time dep_delay arr_time arr_delay carrier tailnum
##    (int) (int) (int)    (int)     (dbl)    (int)     (dbl)   (chr)   (chr)
## 1   2013     1     1      517         2      830        11      UA  N14228
## 2   2013     1     1      533         4      850        20      UA  N24211
## 3   2013     1     1      542         2      923        33      AA  N619AA
## 4   2013     1     1      544        -1     1004       -18      B6  N804JB
## 5   2013     1     1      554        -6      812       -25      DL  N668DN
## 6   2013     1     1      554        -4      740        12      UA  N39463
## 7   2013     1     1      555        -5      913        19      B6  N516JB
## 8   2013     1     1      557        -3      709       -14      EV  N829AS
## 9   2013     1     1      557        -3      838        -8      B6  N593JB
## 10  2013     1     1      558        -2      753         8      AA  N3ALAA
## ..   ...   ...   ...      ...       ...      ...       ...     ...     ...
## Variables not shown: flight (int), origin (chr), dest (chr), air_time
##   (dbl), distance (dbl), hour (dbl), minute (dbl)
```

```
# Use glimpse to few some values in each column
glimpse(flights)
```

```
## Observations: 336,776
## Variables: 16
## $ year      (int) 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013...
## $ month     (int) 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
## $ day       (int) 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
## $ dep_time  (int) 517, 533, 542, 544, 554, 554, 555, 557, 557, 558, 55...
## $ dep_delay (dbl) 2, 4, 2, -1, -6, -4, -5, -3, -3, -2, -2, -2, -2, -2,...
## $ arr_time  (int) 830, 850, 923, 1004, 812, 740, 913, 709, 838, 753, 8...
## $ arr_delay (dbl) 11, 20, 33, -18, -25, 12, 19, -14, -8, 8, -2, -3, 7,...
## $ carrier   (chr) "UA", "UA", "AA", "B6", "DL", "UA", "B6", "EV", "B6"...
## $ tailnum   (chr) "N14228", "N24211", "N619AA", "N804JB", "N668DN", "N...
## $ flight    (int) 1545, 1714, 1141, 725, 461, 1696, 507, 5708, 79, 301...
## $ origin    (chr) "EWR", "LGA", "JFK", "JFK", "LGA", "EWR", "EWR", "LG...
## $ dest      (chr) "IAH", "IAH", "MIA", "BQN", "ATL", "ORD", "FLL", "IA...
## $ air_time  (dbl) 227, 227, 160, 183, 116, 150, 158, 53, 140, 138, 149...
## $ distance  (dbl) 1400, 1416, 1089, 1576, 762, 719, 1065, 229, 944, 73...
## $ hour      (dbl) 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 6...
## $ minute    (dbl) 17, 33, 42, 44, 54, 54, 55, 57, 57, 58, 58, 58, 58, ...
```

```
# filter: inspect subsets of data
# How many flights flew to Madison in 2013?
```

```
flights %>%
  filter(dest == "MSN")
```

```
## Source: local data frame [572 x 16]
##
##     year month   day dep_time dep_delay arr_time arr_delay carrier tailnum
##    (int) (int) (int)    (int)     (dbl)    (int)     (dbl)   (chr)   (chr)
## 1   2013     1     1     1353        -4     1549        24      EV  N14105
## 2   2013     1     2     1422        25     1604        39      EV  N16911
## 3   2013     1     3     1415        24     1540        21      EV  N13968
## 4   2013     1     4     1345        -5     1525         7      EV  N16911
## 5   2013     1     6     1340        -5     1506        -7      EV  N13913
## 6   2013     1     7     1348        -2     1625        67      EV  N13958
## 7   2013     1     8     1347        -3     1526         8      EV  N29906
## 8   2013     1     9     1428        38     1630        72      EV  N13955
## 9   2013     1    10     1344        -6     1510        -8      EV  N22971
## 10  2013     1    11     1344        -6       NA        NA      EV  N15985
## ..   ...   ...   ...      ...       ...      ...       ...     ...     ...
## Variables not shown: flight (int), origin (chr), dest (chr), air_time
##   (dbl), distance (dbl), hour (dbl), minute (dbl)
```

```
# filter: multiple conditions
# How many flights flew to Madison in first week of January?
# Comma separated conditions are combined with '&'
flights %>%
  filter(dest == "MSN", month == 1, day <= 7)
```

```
## Source: local data frame [6 x 16]
##
##    year month   day dep_time dep_delay arr_time arr_delay carrier tailnum
##   (int) (int) (int)    (int)     (dbl)    (int)     (dbl)   (chr)   (chr)
## 1  2013     1     1     1353        -4     1549        24      EV  N14105
## 2  2013     1     2     1422        25     1604        39      EV  N16911
## 3  2013     1     3     1415        24     1540        21      EV  N13968
## 4  2013     1     4     1345        -5     1525         7      EV  N16911
## 5  2013     1     6     1340        -5     1506        -7      EV  N13913
## 6  2013     1     7     1348        -2     1625        67      EV  N13958
## Variables not shown: flight (int), origin (chr), dest (chr), air_time
##   (dbl), distance (dbl), hour (dbl), minute (dbl)
```

```
# QuestionCommas in the filter statement are implicit & (and) operators. Is there anything similar for
# Logical or statements are supported, but there's no shorthand.
flights %>%
  filter(dest == "MSN" | dest == "ORD" | dest == "MDW")
```

```
## Source: local data frame [21,968 x 16]
##
##     year month   day dep_time dep_delay arr_time arr_delay carrier tailnum
##    (int) (int) (int)    (int)     (dbl)    (int)     (dbl)   (chr)   (chr)
## 1   2013     1     1      554        -4      740        12      UA  N39463
## 2   2013     1     1      558        -2      753         8      AA  N3ALAA
## 3   2013     1     1      608         8      807        32      MQ  N9EAMQ
```

```
## 4   2013     1     1      629        -1      824        14      AA  N3CYAA
## 5   2013     1     1      656        -4      854         4      AA  N4WNAA
## 6   2013     1     1      709         9      852        20      UA  N26226
## 7   2013     1     1      715         2      911        21      UA  N841UA
## 8   2013     1     1      739        -6      918       -12      AA  N4WPAA
## 9   2013     1     1      749        39      939        49      MQ  N508MQ
## 10  2013     1     1      828        -2     1027        15      B6  N274JB
## ..   ...   ...   ...      ...       ...      ...       ...     ...     ...
## Variables not shown: flight (int), origin (chr), dest (chr), air_time
##   (dbl), distance (dbl), hour (dbl), minute (dbl)
```

```
# For more complicated checks, I would try a set operation.
flights %>%
  filter(dest %in% c("MSN", "ORD", "MDW"))
```

```
## Source: local data frame [21,968 x 16]
##
##     year month   day dep_time dep_delay arr_time arr_delay carrier tailnum
##    (int) (int) (int)    (int)     (dbl)    (int)     (dbl)   (chr)   (chr)
## 1   2013     1     1      554        -4      740        12      UA  N39463
## 2   2013     1     1      558        -2      753         8      AA  N3ALAA
## 3   2013     1     1      608         8      807        32      MQ  N9EAMQ
## 4   2013     1     1      629        -1      824        14      AA  N3CYAA
## 5   2013     1     1      656        -4      854         4      AA  N4WNAA
## 6   2013     1     1      709         9      852        20      UA  N26226
## 7   2013     1     1      715         2      911        21      UA  N841UA
## 8   2013     1     1      739        -6      918       -12      AA  N4WPAA
## 9   2013     1     1      749        39      939        49      MQ  N508MQ
## 10  2013     1     1      828        -2     1027        15      B6  N274JB
## ..   ...   ...   ...      ...       ...      ...       ...     ...     ...
## Variables not shown: flight (int), origin (chr), dest (chr), air_time
##   (dbl), distance (dbl), hour (dbl), minute (dbl)
```

```
# arrange: sort columns
# Sort by which airport they departed from in NYC, then year, month, day.
flights %>%
  arrange(origin, year, month, day)
```

```
## Source: local data frame [336,776 x 16]
##
##     year month   day dep_time dep_delay arr_time arr_delay carrier tailnum
##    (int) (int) (int)    (int)     (dbl)    (int)     (dbl)   (chr)   (chr)
## 1   2013     1     1      517         2      830        11      UA  N14228
## 2   2013     1     1      554        -4      740        12      UA  N39463
## 3   2013     1     1      555        -5      913        19      B6  N516JB
## 4   2013     1     1      558        -2      923       -14      UA  N53441
## 5   2013     1     1      559        -1      854        -8      UA  N76515
## 6   2013     1     1      601         1      844        -6      B6  N644JB
## 7   2013     1     1      606        -4      858       -12      AA  N633AA
## 8   2013     1     1      607         0      858       -17      UA  N53442
## 9   2013     1     1      608         8      807        32      MQ  N9EAMQ
## 10  2013     1     1      615         0      833        -9      DL  N326NB
## ..   ...   ...   ...      ...       ...      ...       ...     ...     ...
```

```
## Variables not shown: flight (int), origin (chr), dest (chr), air_time
##   (dbl), distance (dbl), hour (dbl), minute (dbl)
```

```
# desc: reverses sorting of a column
# Find longest delayed flights to Madison.
flights %>%
  filter(dest == "MSN") %>%
  arrange(desc(dep_delay))
```

```
## Source: local data frame [572 x 16]
##
##     year month   day dep_time dep_delay arr_time arr_delay carrier tailnum
##    (int) (int) (int)    (int)     (dbl)    (int)     (dbl)   (chr)   (chr)
## 1   2013    12     5     2000       340     2132       337      EV  N14953
## 2   2013    11    17       45       310      206       274      EV  N722EV
## 3   2013     3     8     1907       302     2031       298      EV  N13989
## 4   2013     9    12     1841       291     2135       364      EV  N719EV
## 5   2013    10     7     1912       287     2048       301      EV  N14923
## 6   2013     3    14     1845       280     2026       293      EV  N11192
## 7   2013     6    30     1855       280     2013       274      EV  N13994
## 8   2013    10    11     1857       272     2011       264      EV  N16919
## 9   2013     2     6     2242       257       15       247      EV  N612QX
## 10  2013    12     2     1823       228     1951       206      EV  N709EV
## ..   ...   ...   ...      ...       ...      ...       ...     ...     ...
## Variables not shown: flight (int), origin (chr), dest (chr), air_time
##   (dbl), distance (dbl), hour (dbl), minute (dbl)
```

```
# select
# Select the columns you want.
flights %>%
  select(origin, year, month, day)
```

```
## Source: local data frame [336,776 x 4]
##
##    origin  year month   day
##     (chr) (int) (int) (int)
## 1     EWR  2013     1     1
## 2     LGA  2013     1     1
## 3     JFK  2013     1     1
## 4     JFK  2013     1     1
## 5     LGA  2013     1     1
## 6     EWR  2013     1     1
## 7     EWR  2013     1     1
## 8     LGA  2013     1     1
## 9     JFK  2013     1     1
## 10    LGA  2013     1     1
## ..    ...   ...   ...   ...
```

```
# select's helpers
# select has many helper functions. See ?select.
flights %>%
  select(origin, year:day, starts_with("dep"))
```

```
## Source: local data frame [336,776 x 6]
##
##    origin  year month   day dep_time dep_delay
##     (chr) (int) (int) (int)    (int)     (dbl)
## 1     EWR  2013     1     1      517         2
## 2     LGA  2013     1     1      533         4
## 3     JFK  2013     1     1      542         2
## 4     JFK  2013     1     1      544        -1
## 5     LGA  2013     1     1      554        -6
## 6     EWR  2013     1     1      554        -4
## 7     EWR  2013     1     1      555        -5
## 8     LGA  2013     1     1      557        -3
## 9     JFK  2013     1     1      557        -3
## 10    LGA  2013     1     1      558        -2
## ..    ...   ...   ...   ...      ...       ...
```

```
# negative selecting
# We can drop columns by "negating" the name. Since helpers give us column names, we can negate them to
flights %>%
  select(-dest, -starts_with("arr"),
         -ends_with("time"))
```

```
## Source: local data frame [336,776 x 11]
##
##     year month   day dep_delay carrier tailnum flight origin distance
##    (int) (int) (int)     (dbl)   (chr)   (chr)  (int)  (chr)    (dbl)
## 1   2013     1     1         2      UA  N14228   1545    EWR     1400
## 2   2013     1     1         4      UA  N24211   1714    LGA     1416
## 3   2013     1     1         2      AA  N619AA   1141    JFK     1089
## 4   2013     1     1        -1      B6  N804JB    725    JFK     1576
## 5   2013     1     1        -6      DL  N668DN    461    LGA      762
## 6   2013     1     1        -4      UA  N39463   1696    EWR      719
## 7   2013     1     1        -5      B6  N516JB    507    EWR     1065
## 8   2013     1     1        -3      EV  N829AS   5708    LGA      229
## 9   2013     1     1        -3      B6  N593JB     79    JFK      944
## 10  2013     1     1        -2      AA  N3ALAA    301    LGA      733
## ..   ...   ...   ...       ...     ...     ...    ...    ...      ...
## Variables not shown: hour (dbl), minute (dbl)
```

Recap: Verbs for inspecting data - convert to a `tbl_df` - nice print method - `glimpse` - some of each column - `filter` - subsetting - `arrange` - sorting (desc to reverse the sort) - `select` - picking (and omiting) columns

```
# rename
# Rename columns with rename(NewName = OldName). To keep the order correct, read/remember the renaming
flights %>%
  rename(y = year, m = month, d = day)
```

```
## Source: local data frame [336,776 x 16]
##
##        y     m     d dep_time dep_delay arr_time arr_delay carrier tailnum
##    (int) (int) (int)    (int)     (dbl)    (int)     (dbl)   (chr)   (chr)
## 1   2013     1     1      517         2      830        11      UA  N14228
## 2   2013     1     1      533         4      850        20      UA  N24211
```

```
## 3   2013     1     1      542         2      923        33      AA  N619AA
## 4   2013     1     1      544        -1     1004       -18      B6  N804JB
## 5   2013     1     1      554        -6      812       -25      DL  N668DN
## 6   2013     1     1      554        -4      740        12      UA  N39463
## 7   2013     1     1      555        -5      913        19      B6  N516JB
## 8   2013     1     1      557        -3      709       -14      EV  N829AS
## 9   2013     1     1      557        -3      838        -8      B6  N593JB
## 10  2013     1     1      558        -2      753         8      AA  N3ALAA
## ..   ...   ...   ...      ...       ...      ...       ...     ...     ...
## Variables not shown: flight (int), origin (chr), dest (chr), air_time
##   (dbl), distance (dbl), hour (dbl), minute (dbl)
```

```
# mutate
# How much departure delay did the flight make up for in the air?
flights %>%
  mutate(
    gain = arr_delay - dep_delay,
    speed = (distance / air_time) * 60,
    gain_per_hour = gain / (air_time / 60)) %>%
  select(gain:gain_per_hour)
```

```
## Source: local data frame [336,776 x 3]
##
##     gain    speed gain_per_hour
##    (dbl)    (dbl)         (dbl)
## 1      9 370.0441      2.378855
## 2     16 374.2731      4.229075
## 3     31 408.3750     11.625000
## 4    -17 516.7213     -5.573770
## 5    -19 394.1379     -9.827586
## 6     16 287.6000      6.400000
## 7     24 404.4304      9.113924
## 8    -11 259.2453    -12.452830
## 9     -5 404.5714     -2.142857
## 10    10 318.6957      4.347826
## ..   ...      ...           ...
```

```
# group_by
# Let's compute the average delay per month of flights to Madison.
# Normally-in aggregate, by or plyr's d*ply functions-you specify the grouping as an argument to the ag
aggregate(dep_delay ~ month, flights, mean,
          subset = flights$dest == "MSN")
```

```
##    month dep_delay
## 1      1  18.07692
## 2      2  20.11111
## 3      3  41.87097
## 4      4  29.40741
## 5      5  21.54839
## 6      6  29.83333
## 7      7  11.13333
## 8      8  19.06667
## 9      9  15.97183
```

```
## 10    10  19.26190
## 11    11  15.96250
## 12    12  42.68831
```

```
# group_by
# In dplyr, grouping is its own action. It is done as its own step in the pipeline. Here, we filter to
msn_by_month <- flights %>%
  filter(dest == "MSN") %>%
  group_by(month)
msn_by_month
```

```
## Source: local data frame [572 x 16]
## Groups: month [12]
##
##     year month   day dep_time dep_delay arr_time arr_delay carrier tailnum
##    (int) (int) (int)    (int)     (dbl)    (int)     (dbl)   (chr)   (chr)
## 1   2013     1     1     1353        -4     1549        24      EV  N14105
## 2   2013     1     2     1422        25     1604        39      EV  N16911
## 3   2013     1     3     1415        24     1540        21      EV  N13968
## 4   2013     1     4     1345        -5     1525         7      EV  N16911
## 5   2013     1     6     1340        -5     1506        -7      EV  N13913
## 6   2013     1     7     1348        -2     1625        67      EV  N13958
## 7   2013     1     8     1347        -3     1526         8      EV  N29906
## 8   2013     1     9     1428        38     1630        72      EV  N13955
## 9   2013     1    10     1344        -6     1510        -8      EV  N22971
## 10  2013     1    11     1344        -6       NA        NA      EV  N15985
## ..   ...   ...   ...      ...       ...      ...       ...     ...     ...
## Variables not shown: flight (int), origin (chr), dest (chr), air_time
##   (dbl), distance (dbl), hour (dbl), minute (dbl)
```

```
# summarise
# Now we use summarise to compute (several) aggregate values within each group (month). summarise retur
msn_by_month %>%
  summarise(
    flights = n(),
    avg_delay = mean(dep_delay, na.rm = TRUE),
    n_planes = n_distinct(tailnum))
```

```
## Source: local data frame [12 x 4]
##
##    month flights avg_delay n_planes
##    (int)   (int)     (dbl)    (int)
## 1      1      27  18.07692       23
## 2      2      47  20.11111       36
## 3      3      31  41.87097       29
## 4      4      27  29.40741       25
## 5      5      31  21.54839       29
## 6      6      30  29.83333       28
## 7      7      30  11.13333       25
## 8      8      31  19.06667       26
## 9      9      72  15.97183       57
## 10    10      85  19.26190       61
## 11    11      81  15.96250       54
## 12    12      80  42.68831       58
```

```
# tally
# tally is a shortcut for counting number of items per group.
# Number of flights from NYC by destination by month:
flights %>%
  group_by(dest, month) %>%
  tally
```

```
## Source: local data frame [1,113 x 3]
## Groups: dest [?]
##
##     dest month     n
##    (chr) (int) (int)
## 1    ABQ     4     9
## 2    ABQ     5    31
## 3    ABQ     6    30
## 4    ABQ     7    31
## 5    ABQ     8    31
## 6    ABQ     9    30
## 7    ABQ    10    31
## 8    ABQ    11    30
## 9    ABQ    12    31
## 10   ACK     5    21
## ..   ...   ...   ...
```

```
# ungroup
# Remove the grouping structure with ungroup.
msn_by_month %>% ungroup
```

```
## Source: local data frame [572 x 16]
##
##     year month   day dep_time dep_delay arr_time arr_delay carrier tailnum
##    (int) (int) (int)    (int)     (dbl)    (int)     (dbl)   (chr)   (chr)
## 1   2013     1     1     1353        -4     1549        24      EV  N14105
## 2   2013     1     2     1422        25     1604        39      EV  N16911
## 3   2013     1     3     1415        24     1540        21      EV  N13968
## 4   2013     1     4     1345        -5     1525         7      EV  N16911
## 5   2013     1     6     1340        -5     1506        -7      EV  N13913
## 6   2013     1     7     1348        -2     1625        67      EV  N13958
## 7   2013     1     8     1347        -3     1526         8      EV  N29906
## 8   2013     1     9     1428        38     1630        72      EV  N13955
## 9   2013     1    10     1344        -6     1510        -8      EV  N22971
## 10  2013     1    11     1344        -6       NA        NA      EV  N15985
## ..   ...   ...   ...      ...       ...      ...       ...     ...     ...
## Variables not shown: flight (int), origin (chr), dest (chr), air_time
##   (dbl), distance (dbl), hour (dbl), minute (dbl)
```

```
# Summarizing undoes grouping.
# Each summarise statement peels off one layer of grouping (from the right of the list of groups).
# day gets peeled off
per_day <- flights %>%
  group_by(dest, year, month, day) %>%
  summarise(flights = n())
per_day
```

```
## Source: local data frame [31,229 x 5]
## Groups: dest, year, month [?]
##
##     dest  year month   day flights
##    (chr) (int) (int) (int)   (int)
## 1    ABQ  2013     4    22       1
## 2    ABQ  2013     4    23       1
## 3    ABQ  2013     4    24       1
## 4    ABQ  2013     4    25       1
## 5    ABQ  2013     4    26       1
## 6    ABQ  2013     4    27       1
## 7    ABQ  2013     4    28       1
## 8    ABQ  2013     4    29       1
## 9    ABQ  2013     4    30       1
## 10   ABQ  2013     5     1       1
## ..   ...   ...   ...   ...     ...
```

```
# Peel off month grouping
per_month <- per_day %>%
  summarise(flights = sum(flights))
per_month
```

```
## Source: local data frame [1,113 x 4]
## Groups: dest, year [?]
##
##     dest  year month flights
##    (chr) (int) (int)   (int)
## 1    ABQ  2013     4       9
## 2    ABQ  2013     5      31
## 3    ABQ  2013     6      30
## 4    ABQ  2013     7      31
## 5    ABQ  2013     8      31
## 6    ABQ  2013     9      30
## 7    ABQ  2013    10      31
## 8    ABQ  2013    11      30
## 9    ABQ  2013    12      31
## 10   ACK  2013     5      21
## ..   ...   ...   ...     ...
```

That covers 80% of dplyr - select - filter - arrange - glimpse - rename - mutate - group_by, ungroup - summarise

Other 20%

- assembly: bind_rows, bind_cols
- column-wise operations: mutate_each, summarise_each
- join tables together: left_join, right_join, inner_join, full_join
- filtering joins: semi_join, anti_join
- do: arbitrary code on each chunk
- different types of tabular data (databases, data.tables)

References

Command Line

- The Unix Shell | Software Carpentry

Data Management

- Best Practices Primer | DataONE
- Data Management Guide for Public Participation | DataONE
- Education Modules | DataONE

Data Wrangling in R

Footnotes

1. bash (bourne again shell)