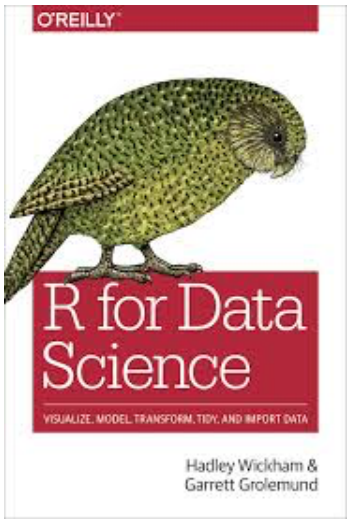
CT474: Smart Grid

Lecture 2: Data Transformation in R

Dr. Jim Duggan,
School of Engineering & Informatics
National University of Ireland Galway.
https://github.com/JimDuggan/PDAR
https://twitter.com/_jimduggan

Overview

- Visualisation is an important tool for insight generation, but it's rare that you get the data in exactly the right form you need" (Wickham and Grolemund 2017)
  - Create new variables
  - Create summaries
  - Order data
- **dplyr** package is designed for data transformation



nycflights13 Data Set

```
> nyc <- nycflights13::flights
>
> nyc
# A tibble: 336,776 × 19
    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay carrier flight
   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>     <dbl>   <chr>  <int>
1   2013     1     1      517            515         2      830            819        11      UA   1545
2   2013     1     1      533            529         4      850            830        20      UA   1714
3   2013     1     1      542            540         2      923            850        33      AA   1141
4   2013     1     1      544            545        -1     1004           1022       -18      B6    725
5   2013     1     1      554            600        -6      812            837       -25      DL    461
6   2013     1     1      554            558        -4      740            728        12      UA   1696
7   2013     1     1      555            600        -5      913            854        19      B6    507
8   2013     1     1      557            600        -3      709            723       -14      EV   5708
9   2013     1     1      557            600        -3      838            846        -8      B6     79
10  2013     1     1      558            600        -2      753            745         8      AA    301
# ... with 336,766 more rows, and 8 more variables: tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>,
#   distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

Data Elements

```
> str(nyc)
Classes ‘tbl_df’, ‘tbl’ and 'data.frame':       336776 obs. of  19 variables:
 $ year          : int  2013 2013 2013 2013 2013 2013 2013 2013 2013 2013 ...
 $ month         : int  1 1 1 1 1 1 1 1 1 1 ...
 $ day           : int  1 1 1 1 1 1 1 1 1 1 ...
 $ dep_time      : int  517 533 542 544 554 554 555 557 557 558 ...
 $ sched_dep_time: int  515 529 540 545 600 558 600 600 600 600 ...
 $ dep_delay     : num  2 4 2 -1 -6 -4 -5 -3 -3 -2 ...
 $ arr_time      : int  830 850 923 1004 812 740 913 709 838 753 ...
 $ sched_arr_time: int  819 830 850 1022 837 728 854 723 846 745 ...
 $ arr_delay     : num  11 20 33 -18 -25 12 19 -14 -8 8 ...
 $ carrier       : chr  "UA" "UA" "AA" "B6" ...
 $ flight        : int  1545 1714 1141 725 461 1696 507 5708 79 301 ...
 $ tailnum       : chr  "N14228" "N24211" "N619AA" "N804JB" ...
 $ origin        : chr  "EWR" "LGA" "JFK" "JFK" ...
 $ dest          : chr  "IAH" "IAH" "MIA" "BQN" ...
 $ air_time      : num  227 227 160 183 116 150 158 53 140 138 ...
 $ distance      : num  1400 1416 1089 1576 762 ...
 $ hour          : num  5 5 5 5 6 5 6 6 6 6 ...
 $ minute        : num  15 29 40 45 0 58 0 0 0 0 ...
 $ time_hour     : POSIXct, format: "2013-01-01 05:00:00" "2013-01-01 05:00:00"
```

tibble abbreviations

Abbreviation	Data Type
int	integers
dbl	doubles (real numbers)
chr	character vectors (strings
dttm	date-times
lgl	logical
fctr	factor (categorical variables with fixed possible values)
date	dates

dplyr Basics: 5 key functions

Function	Purpose
filter()	Pick observations by their values
arrange()	Reorder the rows
select()	Pick variables by their names
mutate()	Create new variables with functions of existing variables
summarise()	Collapse many values down to a single summary

- All verbs (functions) work similarly
  - The first argument is a data frame
  - The subsequent arguments decide what to do with the data frame
  - The result is a data frame (supports chaining of steps)

filter()

- Subset observations based on their values.

```
> filter(nyc,month==1,day==1)
# A tibble: 842 × 19
    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay carrier
   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>     <dbl>   <chr>
1   2013     1     1      517            515         2      830            819        11      UA
2   2013     1     1      533            529         4      850            830        20      UA
3   2013     1     1      542            540         2      923            850        33      AA
4   2013     1     1      544            545        -1     1004           1022       -18      B6
5   2013     1     1      554            600        -6      812            837       -25      DL
6   2013     1     1      554            558        -4      740            728        12      UA
7   2013     1     1      555            600        -5      913            854        19      B6
8   2013     1     1      557            600        -3      709            723       -14      EV
9   2013     1     1      557            600        -3      838            846        -8      B6
10  2013     1     1      558            600        -2      753            745         8      AA
# ... with 832 more rows, and 9 more variables: flight <int>, tailnum <chr>, origin <chr>,
#   dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

Exercises

- Find all flights that:
  - Had an arrival delay of two or more hours
  - Flew to Houston (IAH or HOU)
  - Were operated by United, American or Delta
  - Departed in the summer (July, August and September)
  - Arrived more than 2 hours late, but didn't leave late
  - Departed between midnight and 6AM (inclusive)

Had an arrival delay of two or more hours

```
> filter(nyc,arr_delay > 120)
# A tibble: 10,034 × 19
    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay carrier
   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>     <dbl>   <chr>
1   2013     1     1      811            630       101     1047            830       137      MQ
2   2013     1     1      848           1835       853     1001           1950       851      MQ
3   2013     1     1      957            733       144     1056            853       123      UA
4   2013     1     1     1114            900       134     1447           1222       145      UA
5   2013     1     1     1505           1310       115     1638           1431       127      EV
6   2013     1     1     1525           1340       105     1831           1626       125      B6
7   2013     1     1     1549           1445        64     1912           1656       136      EV
8   2013     1     1     1558           1359       119     1718           1515       123      EV
9   2013     1     1     1732           1630        62     2028           1825       123      EV
10  2013     1     1     1803           1620       103     2008           1750       138      MQ
# ... with 10,024 more rows, and 9 more variables: flight <int>, tailnum <chr>, origin <chr>,
#   dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

Flew to Houston (IAH or HOU)

```
> filter(nyc,dest=="IAH" | dest == "HOU")
# A tibble: 9,313 × 19
    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay carrier
   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>     <dbl>   <chr>
1   2013     1     1      517            515         2      830            819        11      UA
2   2013     1     1      533            529         4      850            830        20      UA
3   2013     1     1      623            627        -4      933            932         1      UA
4   2013     1     1      728            732        -4     1041           1038         3      UA
5   2013     1     1      739            739         0     1104           1038        26      UA
6   2013     1     1      908            908         0     1228           1219         9      UA
7   2013     1     1     1028           1026         2     1350           1339        11      UA
8   2013     1     1     1044           1045        -1     1352           1351         1      UA
9   2013     1     1     1114            900       134     1447           1222       145      UA
10  2013     1     1     1205           1200         5     1503           1505        -2      UA
# ... with 9,303 more rows, and 9 more variables: flight <int>, tailnum <chr>, origin <chr>,
#   dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

Arrived more than 2 hours late, but didn't leave late

```
> filter(nyc, dep_delay <=0 & arr_delay > 120)
# A tibble: 29 x 19
    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay carrier
   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>     <dbl>   <chr>
1   2013     1    27     1419           1420        -1     1754           1550       124      MQ
2   2013    10     7     1350           1350         0     1736           1526       130      EV
3   2013    10     7     1357           1359        -2     1858           1654       124      AA
4   2013    10    16      657            700        -3     1258           1056       122      B6
5   2013    11     1      658            700        -2     1329           1015       194      VX
6   2013     3    18     1844           1847        -3       39           2219       140      UA
7   2013     4    17     1635           1640        -5     2049           1845       124      MQ
8   2013     4    18      558            600        -2     1149            850       179      AA
9   2013     4    18      655            700        -5     1213            950       143      AA
10  2013     5    22     1827           1830        -3     2217           2010       127      MQ
# ... with 19 more rows, and 9 more variables: flight <int>, tailnum <chr>, origin <chr>,
#   dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

arrange()

- Changes the order of rows. Takes a data frame and a set of column names to order by

```
> arrange(nyc,year,month,day)
# A tibble: 336,776 × 19
    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay carrier
   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>     <dbl>   <chr>
1   2013     1     1      517            515         2      830            819        11      UA
2   2013     1     1      533            529         4      850            830        20      UA
3   2013     1     1      542            540         2      923            850        33      AA
4   2013     1     1      544            545        -1     1004           1022       -18      B6
5   2013     1     1      554            600        -6      812            837       -25      DL
6   2013     1     1      554            558        -4      740            728        12      UA
7   2013     1     1      555            600        -5      913            854        19      B6
8   2013     1     1      557            600        -3      709            723       -14      EV
9   2013     1     1      557            600        -3      838            846        -8      B6
10  2013     1     1      558            600        -2      753            745         8      AA
# ... with 336,766 more rows, and 9 more variables: flight <int>, tailnum <chr>, origin <chr>,
#   dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

Using desc()

```
> arrange(nyc,desc(arr_delay))
# A tibble: 336,776 × 19
    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay carrier
   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>     <dbl>   <chr>
1   2013     1     9      641            900      1301     1242           1530      1272      HA
2   2013     6    15     1432           1935      1137     1607           2120      1127      MQ
3   2013     1    10     1121           1635      1126     1239           1810      1109      MQ
4   2013     9    20     1139           1845      1014     1457           2210      1007      AA
5   2013     7    22      845           1600      1005     1044           1815       989      MQ
6   2013     4    10     1100           1900       960     1342           2211       931      DL
7   2013     3    17     2321            810       911      135           1020       915      DL
8   2013     7    22     2257            759       898      121           1026       895      DL
9   2013    12     5      756           1700       896     1058           2020       878      AA
10  2013     5     3     1133           2055       878     1250           2215       875      MQ
# ... with 336,766 more rows, and 9 more variables: flight <int>, tailnum <chr>, origin <chr>,
#   dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

select()

- Allows you to rapidly zoom in on a useful subset using operations based on the variable names

```
> select(nyc,year, month,day,origin,dest)
# A tibble: 336,776 × 5
    year month   day origin  dest
   <int> <int> <int>  <chr> <chr>
1   2013     1     1    EWR   IAH
2   2013     1     1    LGA   IAH
3   2013     1     1    JFK   MIA
4   2013     1     1    JFK   BQN
5   2013     1     1    LGA   ATL
6   2013     1     1    EWR   ORD
7   2013     1     1    EWR   FLL
8   2013     1     1    LGA   IAD
9   2013     1     1    JFK   MCO
10  2013     1     1    LGA   ORD
# ... with 336,766 more rows
```

mutate()

- It is often useful to add new columns that are functions of existing columns

```
> flights_sml <- select(nyc,year:day,ends_with("delay"),distance,air_time)
>
> flights_sml
# A tibble: 336,776 × 7
    year month   day dep_delay arr_delay distance air_time
   <int> <int> <int>     <dbl>     <dbl>    <dbl>    <dbl>
1   2013     1     1         2        11     1400      227
2   2013     1     1         4        20     1416      227
3   2013     1     1         2        33     1089      160
4   2013     1     1        -1       -18     1576      183
5   2013     1     1        -6       -25      762      116
6   2013     1     1        -4        12      719      150
7   2013     1     1        -5        19     1065      158
8   2013     1     1        -3       -14      229       53
9   2013     1     1        -3        -8      944      140
10  2013     1     1        -2         8      733      138
# ... with 336,766 more rows
```

Example of mutate()

```
> mutate(flights_sml,
+        gain = arr_delay - dep_delay,
+        speed = distance / air_time * 60)
# A tibble: 336,776 × 9
    year month   day dep_delay arr_delay distance air_time  gain    speed
   <int> <int> <int>     <dbl>     <dbl>    <dbl>    <dbl> <dbl>    <dbl>
1   2013     1     1         2        11     1400      227     9 370.0441
2   2013     1     1         4        20     1416      227    16 374.2731
3   2013     1     1         2        33     1089      160    31 408.3750
4   2013     1     1        -1       -18     1576      183   -17 516.7213
5   2013     1     1        -6       -25      762      116   -19 394.1379
6   2013     1     1        -4        12      719      150    16 287.6000
7   2013     1     1        -5        19     1065      158    24 404.4304
8   2013     1     1        -3       -14      229       53   -11 259.2453
9   2013     1     1        -3        -8      944      140    -5 404.5714
10  2013     1     1        -2         8      733      138    10 318.6957
# ... with 336,766 more rows
```

summarise()

```
> by_day <- group_by(nyc,year,month,day)
>
> summarize(by_day, AverageDelay = mean(dep_delay,na.rm=T))
Source: local data frame [365 x 4]
Groups: year, month [?]

    year month   day AverageDelay
   <int> <int> <int>        <dbl>
1   2013     1     1    11.548926
2   2013     1     2    13.858824
3   2013     1     3    10.987832
4   2013     1     4     8.951595
5   2013     1     5     5.732218
6   2013     1     6     7.148014
7   2013     1     7     5.417204
8   2013     1     8     2.553073
9   2013     1     9     2.276477
10  2013     1    10     2.844995
# ... with 355 more rows
```

Pipe operator %>%

- x %>% y *turns into* f(x,y)

```
> group_by(nyc,year,month,day) %>%
+   summarize(AverageDelay = mean(dep_delay,na.rm=T))
Source: local data frame [365 x 4]
Groups: year, month [?]

    year month   day AverageDelay
   <int> <int> <int>        <dbl>
1   2013     1     1    11.548926
2   2013     1     2    13.858824
3   2013     1     3    10.987832
4   2013     1     4     8.951595
5   2013     1     5     5.732218
6   2013     1     6     7.148014
7   2013     1     7     5.417204
8   2013     1     8     2.553073
9   2013     1     9     2.276477
10  2013     1    10     2.844995
# ... with 355 more rows
```

Example: Get subject average & student average

Student ID	CX1000	CX1001	CX1002	CX1003	CX1004	CX1005	CX1006	CX1007	CX1008	CX1009
1111111	56	51	78	85	63	45	55	59	52	76
1111112	56	64	68	80	70	39	46	60	55	74
1111113	52	61	63	81	71	49	54	61	54	76
1111114	50	42	72	81	63	44	62	59	56	68
1111115	67	53	77	84	65	52	63	62	52	71
1111116	45	57	62	32	61	56	62	51	55	79
1111117	67	58	54	77	75	44	58	62	57	77
1111118	69	50	66	78	72	39	60	58	57	84
1111119	70	56	62	80	71	52	60	63	54	70
1111120	51	52	46	82	74	42	66	63	55	73
1111121	71	89	90	72	99	86	67	81	79	79
1111122	66	62	80	85	67	49	60	59	54	77
1111123	62	56	75	88	70	46	54	57	57	72
1111124	61	77	62	79	70	43	71	59	61	79
1111125	72	56	48	78	57	45	56	63	53	75
1111126	67	56	68	79	63	41	42	64	56	70
1111127	64	67	74	84	69	44	48	61	55	70
1111128	77	56	66	82	59	44	61	61	54	64
1111129	64	52	66	72	64	45	84	60	51	61
1111130	67	65	70	79	67	44	54	56	55	73
1111131	55	38	79	84	66	44	58	63	51	74
1111132	73	41	52	82	55	42	65	59	55	79

Bring into R

```
> data <- as_data_frame(read.xls("data/energy/ExamData.xlsx"),stringsAsFactors=F)
>
> data
# A tibble: 50 × 11
   Student.ID CX1000 CX1001 CX1002 CX1003 CX1004 CX1005 CX1006 CX1007 CX1008 CX1009
        <int>  <int>  <int>  <int>  <int>  <int>  <int>  <int>  <int>  <int>  <int>
1     1111111     56     51     78     85     63     45     55     59     52     76
2     1111112     56     64     68     80     70     39     46     60     55     74
3     1111113     52     61     63     81     71     49     54     61     54     76
4     1111114     50     42     72     81     63     44     62     59     56     68
5     1111115     67     53     77     84     65     52     63     62     52     71
6     1111116     45     57     62     32     61     56     62     51     55     79
7     1111117     67     58     54     77     75     44     58     62     57     77
8     1111118     69     50     66     78     72     39     60     58     57     84
9     1111119     70     56     62     80     71     52     60     63     54     70
10    1111120     51     52     46     82     74     42     66     63     55     73
# ... with 40 more rows
```

Create a tidy data set
One observation per row

```
> tdata <- gather(data,key = Subject, value = Result,CX1000:CX1009)
>
> tdata
# A tibble: 500 × 3
   Student.ID Subject Result
        <int>   <chr>  <int>
1     1111111  CX1000     56
2     1111112  CX1000     56
3     1111113  CX1000     52
4     1111114  CX1000     50
5     1111115  CX1000     67
6     1111116  CX1000     45
7     1111117  CX1000     67
8     1111118  CX1000     69
9     1111119  CX1000     70
10    1111120  CX1000     51
# ... with 490 more rows
```

Student Summaries

```
> tdata %>% group_by(Student.ID) %>%
+             summarise(Average=mean(Result),Minimum=min(Result),Maximum=max(Result))
# A tibble: 50 × 4
   Student.ID Average Minimum Maximum
        <int>   <dbl>   <int>   <int>
1     1111111    62.0      45      85
2     1111112    61.2      39      80
3     1111113    62.2      49      81
4     1111114    59.7      42      81
5     1111115    64.6      52      84
6     1111116    56.0      32      79
7     1111117    62.9      44      77
8     1111118    63.3      39      84
9     1111119    63.8      52      80
10    1111120    60.4      42      82
# ... with 40 more rows
```

Subject Summaries

```
>
> tdata %>% group_by(Subject) %>%
+   summarise(Average=mean(Result),Minimum=min(Result),Maximum=max(Result))
# A tibble: 10 × 4
   Subject Average Minimum Maximum
     <chr>   <dbl>   <int>   <int>
1   CX1000   60.44      34      81
2   CX1001   57.42      32      89
3   CX1002   68.10      45      90
4   CX1003   78.78      22      90
5   CX1004   68.04      35      99
6   CX1005   46.52      39      86
7   CX1006   57.30      14      84
8   CX1007   60.22      29      81
9   CX1008   55.42      31      79
10  CX1009   73.40      43      84
```

Adding Extra Info.

```
>
> tdata %>% group_by(Student.ID) %>%
+   summarise(Average=mean(Result), Minimum=min(Result),Maximum=max(Result),
+             ExamsTaken= n(),Fails=sum(Result<40),Passed=sum(Result>=40 & Result<50),
+             H2.2=sum(Result>=50 & Result<60), H2.1=sum(Result>=60 & Result<70),
+             H1.1=sum(Result>=70 & Result<100))
# A tibble: 50 × 10
   Student.ID Average Minimum Maximum ExamsTaken Fails Passed  H2.2  H2.1  H1.1
        <int>   <dbl>   <int>   <int>      <int> <int>  <int> <int> <int> <int>
1     1111111    62.0      45      85         10     0      1     5     1     3
2     1111112    61.2      39      80         10     1      1     2     3     3
3     1111113    62.2      49      81         10     0      1     3     3     3
4     1111114    59.7      42      81         10     0      2     3     3     2
5     1111115    64.6      52      84         10     0      0     3     4     3
6     1111116    56.0      32      79         10     1      1     4     3     1
7     1111117    62.9      44      77         10     0      1     4     2     3
8     1111118    63.3      39      84         10     1      0     3     3     3
9     1111119    63.8      52      80         10     0      0     3     3     4
10    1111120    60.4      42      82         10     0      2     3     2     3
# ... with 40 more rows
```