A Walkthrough of ‘dplyr’

Mark Hashimoto

12/3/2017

Motivation

- For this blog post I will be discussing the package ‘dplyr’, which we have used numerous times in our stat133 class. The aim of this post will be to give an understanding of what dplyr is, how its functions can be used, and why it is so useful within the context of data science.

Introduction

- Dplyr uses a grammar for data manipulation, and is the updated version of the original ‘plyr’ package. The additional ‘d’ in the name stands for ‘data frame’, and it is with the dplyr package that we may masterfully play around with data in data frames.
- The data manipulation package is one part of a combination of packages known as ‘tidyverse’, a name you probably have heard of before when using R. Tidyverse; which includes many other data science packages we have encountered; such as ‘ggplot2’, ‘readr’, and ‘tidyr’; was created by a group of R developers including R chief scientist Hadley Wickham.

Content

- dplyr contains an impressive amount of functions, all of which can be used for different tasks:

- Reshape data
- Syntax
- Subset Observations (Rows)
- Subset Variables (Cols)
- Summarise Data
- Make New Variables
- Combine Data Sets
- Group Data

Getting Started

- For our walk-through of dplyr and its functions, we will be using a dataset of **pokemon** from generations 1 through 6. For those who are not familiar with Pokemon, it is a game based on fictitious creatures who are used for battle. Each pokemon has a specific type (grass, ground, electric, water, etc.) and specific battle statistics (attack, defense, speed, etc.). Generation refers to the different versions of Pokemon universes that have been used, generation 1 Pokemon contains all the Pokemon from the first version of Pokemon, generation 2 contains all Pokemon from the second, and so on so forth

Pokemon

- Let’s begin by loading our dplyr package and readr package using the library() function. The readr package will be used to read the csv of pokemon data

```
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(readr)
dat = read_csv('../data/Pokemon.csv')
```

```
## Parsed with column specification:
## cols(
##   `#` = col_integer(),
##   Name = col_character(),
##   `Type 1` = col_character(),
##   `Type 2` = col_character(),
##   Total = col_integer(),
##   HP = col_integer(),
##   Attack = col_integer(),
##   Defense = col_integer(),
##   `Sp. Atk` = col_integer(),
##   `Sp. Def` = col_integer(),
##   Speed = col_integer(),
##   Generation = col_integer(),
##   Legendary = col_character()
## )
```

- Let's take a glimpse at our Pokemon dataset to give a sense of what kind of data is used.

```
head(dat)
```

```
## # A tibble: 6 x 13
##      `#`                  Name `Type 1` `Type 2` Total    HP Attack Defense
##   <int>                 <chr>    <chr>    <chr> <int> <int>  <int>   <int>
## 1     1             Bulbasaur    Grass   Poison   318    45     49      49
## 2     2               Ivysaur    Grass   Poison   405    60     62      63
## 3     3              Venusaur    Grass   Poison   525    80     82      83
## 4     3 VenusaurMega Venusaur    Grass   Poison   625    80    100     123
## 5     4            Charmander     Fire     <NA>   309    39     52      43
## 6     5            Charmeleon     Fire     <NA>   405    58     64      58
## # ... with 5 more variables: `Sp. Atk` <int>, `Sp. Def` <int>,
## #   Speed <int>, Generation <int>, Legendary <chr>
```

- We observe that our data frame gives us a row of information for each pokemon
- Our columns include Pokemon's
  - '#' (id number of pokemon)
  - Name
  - type1
  - type2 (if pokemon has 2 types)
  - total stats
  - HP (health points)
  - Attack (points)
  - Defense (points)
  - Sp(ecial) Attack
  - Sp(ecial) Defense
  - Speed
  - Generation
  - Legendary (tells us if the pokemon is a legendary type, which a type of very rare, very powerful pokemon)

-Reshape Data

- Functions used to change the layout of our dataset include:
  - arrange()
  - rename()
- arrange(data, column) takes the data frame 'data' and arranges the rows in terms of values of 'column' from lowest to highest
- arrange(data, desc(column)) will order the rows for values of the column from highest to lowest

```
head(arrange(dat, Speed))
```

```
## # A tibble: 6 x 13
##      `#`      Name `Type 1` `Type 2` Total    HP Attack Defense `Sp. Atk`
##   <int>     <chr>    <chr>    <chr> <int> <int>  <int>   <int>     <int>
## 1   213   Shuckle      Bug     Rock   505    20     10     230        10
## 2   446  Munchlax   Normal     <NA>   390   135     85      40        40
## 3   328  Trapinch   Ground     <NA>   290    45    100      45        45
## 4   438    Bonsly     Rock     <NA>   290    50     80      95        10
## 5   597 Ferroseed    Grass    Steel   305    44     50      91        24
## 6    79  Slowpoke    Water  Psychic   315    90     65      65        40
## # ... with 4 more variables: `Sp. Def` <int>, Speed <int>,
## #   Generation <int>, Legendary <chr>
```

```
head(arrange(dat, desc(Speed)))
```

```
## # A tibble: 6 x 13
##      `#`                      Name `Type 1` `Type 2` Total    HP Attack
##    <int>                     <chr>    <chr>    <chr> <int> <int>  <int>
## 1   386          DeoxysSpeed Forme  Psychic      <NA>   600    50     95
## 2   291                    Ninjask      Bug   Flying   456    61     90
## 3    65     AlakazamMega Alakazam  Psychic      <NA>   590    55     50
## 4   142 AerodactylMega Aerodactyl     Rock   Flying   615    80    135
## 5   386         DeoxysNormal Forme  Psychic      <NA>   600    50    150
## 6   386         DeoxysAttack Forme  Psychic      <NA>   600    50    180
## # ... with 6 more variables: Defense <int>, `Sp. Atk` <int>, `Sp.
## #   Def` <int>, Speed <int>, Generation <int>, Legendary <chr>
```

- The function rename(data, old_column = new_column) will take data frame 'data' and rename column 'old_column' to 'new_column'

*Lets remove the spaces between Type and # in our data frame to make referencing our columns easier.

```
data = rename(dat, Type1 = 'Type 1', Type2 = 'Type 2')
data
```

```
## # A tibble: 800 x 13
##      `#`                      Name Type1  Type2 Total    HP Attack Defense
##    <int>                     <chr> <chr>  <chr> <int> <int>  <int>   <int>
##  1     1                 Bulbasaur Grass Poison   318    45     49      49
##  2     2                   Ivysaur Grass Poison   405    60     62      63
##  3     3                  Venusaur Grass Poison   525    80     82      83
##  4     3     VenusaurMega Venusaur Grass Poison   625    80    100     123
##  5     4                Charmander  Fire   <NA>   309    39     52      43
##  6     5                Charmeleon  Fire   <NA>   405    58     64      58
##  7     6                 Charizard  Fire Flying   534    78     84      78
##  8     6 CharizardMega Charizard X  Fire Dragon   634    78    130     111
##  9     6 CharizardMega Charizard Y  Fire Flying   634    78    104      78
## 10     7                  Squirtle Water   <NA>   314    44     48      65
## # ... with 790 more variables: `Sp. Atk` <int>, `Sp.
## #   Def` <int>, Speed <int>, Generation <int>, Legendary <chr>
```

Syntax

- Syntax related function of dplyr include
  - tbl_df()
  - glimpse()
- both of these functions take the dataframe as an argument
  - tblr_df converts data to table class. tables are easier to examine than data frames
  - glimpse() gives a information dense summary of table data

```
tbl_df(dat)
```

```
## # A tibble: 800 x 13
##      `#`                      Name `Type 1` `Type 2` Total    HP Attack
##    <int>                     <chr>    <chr>    <chr> <int> <int>  <int>
##  1     1                 Bulbasaur    Grass   Poison   318    45     49
##  2     2                   Ivysaur    Grass   Poison   405    60     62
##  3     3                  Venusaur    Grass   Poison   525    80     82
##  4     3     VenusaurMega Venusaur    Grass   Poison   625    80    100
##  5     4                Charmander     Fire      <NA>   309    39     52
##  6     5                Charmeleon     Fire      <NA>   405    58     64
##  7     6                 Charizard     Fire   Flying   534    78     84
##  8     6 CharizardMega Charizard X     Fire   Dragon   634    78    130
##  9     6 CharizardMega Charizard Y     Fire   Flying   634    78    104
## 10     7                  Squirtle    Water      <NA>   314    44     48
## # ... with 790 more rows, and 6 more variables: Defense <int>, `Sp.
## #   Atk` <int>, `Sp. Def` <int>, Speed <int>, Generation <int>,
## #   Legendary <chr>
```

```
glimpse(dat)
```

```
## Observations: 800
## Variables: 13
## $ `#`         <int> 1, 2, 3, 3, 4, 5, 6, 6, 6, 7, 8, 9, 9, 10, 11, 12, ...
## $ Name        <chr> "Bulbasaur", "Ivysaur", "Venusaur", "VenusaurMega V...
## $ `Type 1`    <chr> "Grass", "Grass", "Grass", "Grass", "Fire", "Fire",...
## $ `Type 2`    <chr> "Poison", "Poison", "Poison", "Poison", NA, NA, "Fl...
## $ Total       <int> 318, 405, 525, 625, 309, 405, 534, 634, 634, 314, 4...
## $ HP          <int> 45, 60, 80, 80, 39, 58, 78, 78, 78, 44, 59, 79, 79,...
## $ Attack      <int> 49, 62, 82, 100, 52, 64, 84, 130, 104, 48, 63, 83, ...
## $ Defense     <int> 49, 63, 83, 123, 43, 58, 78, 111, 78, 65, 80, 100, ...
## $ `Sp. Atk`   <int> 65, 80, 100, 122, 60, 80, 109, 130, 159, 50, 65, 85...
## $ `Sp. Def`   <int> 65, 80, 100, 120, 50, 65, 85, 85, 115, 64, 80, 105,...
## $ Speed       <int> 45, 60, 80, 80, 65, 80, 100, 100, 100, 43, 58, 78, ...
## $ Generation  <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
## $ Legendary   <chr> "False", "False", "False", "False", "False", "False...
```

Subset Observations

-We can use dplyr functions to also subset rows of our data frame + filter() + sample_frac() + sample_n + slice()

- filter() will extract rows based on logical criteria

*Lets see which pokemon have healthpoints (HP) over 100

```
head(filter(dat, HP > 100))
```

```
## # A tibble: 6 x 13
##      `#`       Name `Type 1` `Type 2` Total    HP Attack Defense `Sp. Atk`
##   <int>      <chr>    <chr>    <chr> <int> <int>  <int>   <int>     <int>
## 1    39 Jigglypuff   Normal    Fairy   270   115     45      20        45
## 2    40 Wigglytuff   Normal    Fairy   435   140     70      45        85
## 3    89        Muk   Poison     <NA>   500   105    105      75        65
## 4   112     Rhydon   Ground     Rock   485   105    130     120        45
## 5   113    Chansey   Normal     <NA>   450   250      5       5        35
## 6   115 Kangaskhan   Normal     <NA>   490   105     95      80        40
## # ... with 4 more variables: `Sp. Def` <int>, Speed <int>,
## #   Generation <int>, Legendary <chr>
```

- sample_frac(data, 'fraction', 'replace') will randomly select a fraction of rows
  - fraction argument states what fraction of original rows will be sampled
  - the logical argument 'replace' will determine whether or not rows will be replaced during sampling

```
sample_frac(dat, .01, replace = FALSE)
```

```
## # A tibble: 8 x 13
##      `#`     Name `Type 1` `Type 2` Total    HP Attack Defense `Sp. Atk`
##   <int>    <chr>    <chr>    <chr> <int> <int>  <int>   <int>     <int>
## 1   208  Steelix    Steel   Ground   510    75     85     200        55
## 2   183   Marill    Water    Fairy   250    70     20      50        20
## 3   481  Mesprit  Psychic     <NA>   580    80    105     105       105
## 4   440  Happiny   Normal     <NA>   220   100      5       5        15
## 5   427  Buneary   Normal     <NA>   350    55     66      44        44
## 6   586 Sawsbuck   Normal    Grass   475    80    100      70        60
## 7   194   Wooper    Water   Ground   210    55     45      45        25
## 8   592 Frillish    Water    Ghost   335    55     40      50        65
## # ... with 4 more variables: `Sp. Def` <int>, Speed <int>,
## #   Generation <int>, Legendary <chr>
```

```
sample_frac(dat, .01, replace = TRUE)
```

```
## # A tibble: 8 x 13
##      `#`                     Name `Type 1` `Type 2` Total    HP Attack
##   <int>                    <chr>    <chr>    <chr> <int> <int>  <int>
## 1   270                   Lotad    Water    Grass   220    40     30
## 2   376               Metagross    Steel  Psychic   600    80    135
## 3   611                  Fraxure   Dragon     <NA>   410    66    117
## 4   632                  Durant      Bug    Steel   484    58    109
## 5   720       HoopaHoopa Unbound  Psychic     Dark   680    80    160
## 6     6 CharizardMega Charizard X     Fire   Dragon   634    78    130
## 7   380                   Latias   Dragon  Psychic   600    80     80
## 8   229     HoundoomMega Houndoom     Dark     Fire   600    75     90
## # ... with 6 more variables: Defense <int>, `Sp. Atk` <int>, `Sp.
## #   Def` <int>, Speed <int>, Generation <int>, Legendary <chr>
```

- sample_n(dat, number, replace) gives a random sample of a stated number of rows
  - number argument states how many rows will be sampled
  - the logical argument 'replace' will determine whether or not rows will be replaced during sampling

```
sample_n(dat, 10, replace = TRUE)
```

```
## # A tibble: 10 x 13
##      `#`                Name `Type 1` `Type 2` Total    HP Attack Defense
##    <int>               <chr>    <chr>    <chr> <int> <int>  <int>   <int>
##  1   160          Feraligatr    Water     <NA>   530    85    105     100
##  2   334 AltariaMega Altaria   Dragon    Fairy   590    75    110     110
##  3   147             Dratini   Dragon     <NA>   300    41     64      45
##  4   373           Salamence   Dragon   Flying   600    95    135      80
##  5   179              Mareep Electric     <NA>   280    55     40      40
##  6   240               Magby     Fire     <NA>   365    45     75      37
##  7   359               Absol     Dark     <NA>   465    65    130      60
##  8   532             Timburr Fighting     <NA>   305    75     80      55
##  9    69           Bellsprout    Grass   Poison   300    50     75      35
## 10    36            Clefable    Fairy     <NA>   483    95     70      73
## # ... with 5 more variables: `Sp. Atk` <int>, `Sp. Def` <int>,
## #   Speed <int>, Generation <int>, Legendary <chr>
```

- slice(data, index) function will select rows by position

- data argument states which dataframe to be used
- index vector states which row positions to be selected

- Let's select rows from position 1 to 5

```
slice(dat, 5:10)
```

```
## # A tibble: 6 x 13
##      `#`                      Name `Type 1` `Type 2` Total    HP Attack
##   <int>                     <chr>    <chr>    <chr> <int> <int>  <int>
## 1     4                Charmander     Fire     <NA>   309    39     52
## 2     5                Charmeleon     Fire     <NA>   405    58     64
## 3     6                 Charizard     Fire   Flying   534    78     84
## 4     6 CharizardMega Charizard X     Fire   Dragon   634    78    130
## 5     6 CharizardMega Charizard Y     Fire   Flying   634    78    104
## 6     7                  Squirtle    Water     <NA>   314    44     48
## # ... with 6 more variables: Defense <int>, `Sp. Atk` <int>, `Sp.
## #   Def` <int>, Speed <int>, Generation <int>, Legendary <chr>
```

Subset Variables

- dplyr functions that can be used to subset certain variables (columns) of a data frame:
  - select()
- select(dat, column1, column1, contain, column1:column3)
  - will return the dataframe with columns 'columns1', and 'column'2
  - one column to n columns can be selected in a data frame with n columns
  - index of columns can be used or title of column
  - the '-' operator can be used in front of a column name to choose all columns except that one
  - the contains('char') argument can be used to select all columns that contain 'char'
  - we can use a vector of columns (column1: column4) to select all columns between column 1 and 4
- selecting variable of index 1,2,3, and 4

```
head(select(dat, 1,2,3,4))
```

```
## # A tibble: 6 x 4
##      `#`                  Name `Type 1` `Type 2`
##   <int>                 <chr>    <chr>    <chr>
## 1     1             Bulbasaur    Grass   Poison
## 2     2               Ivysaur    Grass   Poison
## 3     3              Venusaur    Grass   Poison
## 4     3 VenusaurMega Venusaur    Grass   Poison
## 5     4            Charmander     Fire     <NA>
## 6     5            Charmeleon     Fire     <NA>
```

*selecting variables of name 'Name' and 'Legendary'

```
head(select(dat, 'Name', 'Legendary'))
```

```
## # A tibble: 6 x 2
##                   Name Legendary
##                  <chr>     <chr>
## 1             Bulbasaur     False
## 2               Ivysaur     False
## 3              Venusaur     False
## 4 VenusaurMega Venusaur     False
## 5            Charmander     False
## 6            Charmeleon     False
```

- selecting all variable except 'Name' and 'Legendary'

```
head(select(dat, -Name, -Legendary))
```

```
## # A tibble: 6 x 11
##      `#` `Type 1` `Type 2` Total    HP Attack Defense `Sp. Atk` `Sp. Def`
##   <int>    <chr>    <chr> <int> <int>  <int>   <int>     <int>     <int>
## 1     1    Grass   Poison   318    45     49      49        65        65
## 2     2    Grass   Poison   405    60     62      63        80        80
## 3     3    Grass   Poison   525    80     82      83       100       100
## 4     3    Grass   Poison   625    80    100     123       122       120
## 5     4     Fire     <NA>   309    39     52      43        60        50
## 6     5     Fire     <NA>   405    58     64      58        80        65
## # ... with 2 more variables: Speed <int>, Generation <int>
```

- selecting variables that contain the character 'Def'

```
head(select(dat, contains('Def')))
```

```
## # A tibble: 6 x 2
##   Defense `Sp. Def`
##     <int>     <int>
## 1      49        65
## 2      63        80
## 3      83       100
## 4     123       120
## 5      43        50
## 6      58        65
```

*selecting variable including and inbetween Name and HP

```
head(select(dat, Name:HP))
```

```
## # A tibble: 6 x 5
##                    Name `Type 1` `Type 2` Total    HP
##                   <chr>    <chr>    <chr> <int> <int>
## 1             Bulbasaur    Grass   Poison   318    45
## 2               Ivysaur    Grass   Poison   405    60
## 3              Venusaur    Grass   Poison   525    80
## 4 VenusaurMega Venusaur    Grass   Poison   625    80
## 5            Charmander     Fire     <NA>   309    39
## 6            Charmeleon     Fire     <NA>   405    58
```

Summarise Data

- functions in dplyr can also be used to summarise data in our dataframe:
  - summarise(data, avg = mean(column))
  - summarise_each(data, funs(mean))
  - count(data, column)
- The summarise(data, avg = function(column)) function will summarise 'data' into a single row of values using a function 'function' applied to all values of column 'column'
- Summarise_all(data, funs(function)) will apply the function 'function' to each column in dataframe 'data'. Keep in mind, this will only for numeric functions work if all columns are a numeric type.
- the count function will count the number of rows with each unique value of variable
- Let's summarise our data by finding the average HP of all our Pokemon in the dataframe
- then find the average value of all our numeric values in our data frame
- Finally, we will count the number of pokemon with 'type 1'

```
summarise(dat, avg = mean(HP))
```

```
## # A tibble: 1 x 1
##        avg
##      <dbl>
## 1 69.25875
```

```
numericdat = select(dat, -c(1,2,3,4,13))
summarise_all(numericdat, funs(mean))
```

```
## # A tibble: 1 x 8
##      Total       HP   Attack Defense `Sp. Atk` `Sp. Def`   Speed
##      <dbl>    <dbl>    <dbl>   <dbl>     <dbl>     <dbl>   <dbl>
## 1 435.1025 69.25875 79.00125 73.8425     72.82   71.9025 68.2775
## # ... with 1 more variables: Generation <dbl>
```

```
count(dat, 'Type 1')
```

```
## # A tibble: 1 x 2
##   `"Type 1"`     n
##        <chr> <int>
## 1     Type 1   800
```

Make New Variables

- We can also use function in dplyr to make new variables (columns) and add them to our dataframe:
  - mutate(data, new_col = col1 + col2)
  - transmute(data, new_col = col1 + col2)
- mutate(data, new_col = col1 + col2) will take the dataframe 'data', and add a new column 'new_col', whose values are 'col1' added by 'col2'
- Here, we will create a new variable 'Strength', that will be the combined number of Attack and Defense points each Pokemon has

```
head(mutate(dat, Strength = Attack + Defense))
```

```
## # A tibble: 6 x 14
##      `#`                   Name `Type 1` `Type 2` Total    HP Attack Defense
##   <int>                  <chr>    <chr>    <chr> <int> <int>  <int>   <int>
## 1     1              Bulbasaur    Grass   Poison   318    45     49      49
## 2     2                Ivysaur    Grass   Poison   405    60     62      63
## 3     3               Venusaur    Grass   Poison   525    80     82      83
## 4     3 VenusaurMega Venusaur    Grass   Poison   625    80    100     123
## 5     4             Charmander     Fire     <NA>   309    39     52      43
## 6     5             Charmeleon     Fire     <NA>   405    58     64      58
## # ... with 6 more variables: `Sp. Atk` <int>, `Sp. Def` <int>,
## #   Speed <int>, Generation <int>, Legendary <chr>, Strength <int>
```

- transmute(data, new_col = col1 + col2) will replace orignal columns in dataframe ‘data’ with the newly computed column ‘new_col’

```
head(transmute(dat, Strength = Attack + Defense))
```

```
## # A tibble: 6 x 1
##   Strength
##      <int>
## 1       98
## 2      125
## 3      165
## 4      223
## 5       95
## 6      122
```

Group Data

- Dplyr functions can be also be used to group or ungroup data in dataframes:
  - group_by()
  - ungroup()
- group_by(data, variable, function) will take a dataframe ‘data’ and group the data into rows with the same values of ‘variable’, and will apply the ‘function’ to all the other variables

*The result of grouping will create a dataframe identical to our original data, but will create groups categorized by the selected variable.

- Grouping is very useful for showing a summary of statistical information. For our example we will see the average attack points for pokemon of each type.lets group our pokemon dataset by ‘Type 1’, and then show the average Attack points for eac type of pokemon

```
grouped = group_by(data, Type1)
summarise(grouped, avg_attack = mean(Attack))
```

```
## # A tibble: 18 x 2
##       Type1 avg_attack
##       <chr>      <dbl>
##  1      Bug   70.97101
##  2     Dark   88.38710
##  3   Dragon  112.12500
##  4 Electric   69.09091
##  5    Fairy   61.52941
##  6 Fighting   96.77778
##  7     Fire   84.76923
##  8   Flying   78.75000
##  9    Ghost   73.78125
## 10    Grass   73.21429
## 11   Ground   95.75000
## 12      Ice   72.75000
## 13   Normal   73.46939
## 14   Poison   74.67857
## 15  Psychic   71.45614
## 16     Rock   92.86364
## 17    Steel   92.70370
## 18    Water   74.15179
```

- The ungroup(data) function will ungroup any grouped data frame ‘data’
- Notice how when we summarise our data by average attack points, it now gives us the average Attack of all pokemon, and not the average attack of each group.

```
ungrouped = ungroup(grouped)
summarise(ungrouped, avg_attack = mean(Attack))
```

```
## # A tibble: 1 x 1
##   avg_attack
##        <dbl>
## 1   79.00125
```

Conclusion

- Overall, the versatility and expansiveness of dplyr makes manipulating data frames simple and quick. With its many functions we may be able to perform many different types of tasks on data frames; including: Reshape data, Syntax, Subset Observations (Rows), Subset

Variables (Cols), Summarise Data, Make New Variables, Combine Data Sets, and Group Data.

- From this blog I hope you have noticed the many ways in which dplyr and its functions can be utilized to manipulate our data. In Data Science it is crucial for us to be able to manipulate data in order to extract key information, and by using dplyr, this process has become extremeley easy. Thanks dplyr!

References

Syntax and Documentation:

1. https://github.com/ucb-stat133/stat133-fall-2017/blob/master/slides/12-dplyr-introduction.pdf
2. https://www.rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf
3. http://dplyr.tidyverse.org/
4. https://www.rdocumentation.org/packages/dplyr/versions/0.5.0
5. http://r4ds.had.co.nz/transform.html

Dataset:

6. https://www.kaggle.com/abcsds/pokemon

Picture:

7. https://www.google.com/search?q=pokemon&source=lnms&tbm=isch&sa=X&ved=0ahUKEwjb9bCkk-_XAhXIKCYKHRpMDI8Q_AUICygC&biw=1280&bih=640#imgrc=JJPo-pmtd4sjTM: