Data Summarization

Data Wrangling in R

Quick Data read in

We can use the Charm City Circulator Dataset from "http://sisbid.github.io/Data-Wrangling/data/Charm_City_Circulator_Ridership.csv".

```
circ = read_csv(paste0("http://sisbid.github.io/Data-Wrangling/",
                       "data/Charm_City_Circulator_Ridership.csv"))
# or circ = read_csv("../data/Charm_City_Circulator_Ridership.csv")
```

Head and Tail Commands

The `head`/`tail` commands displays the first/last `6` (default) rows:

```
head(circ)
```

```
# A tibble: 6 x 15
  day     date    orangeBoardings orangeAlightings orangeAverage purpleBoardir
  <chr>   <chr>             <dbl>            <dbl>         <dbl>          <db
1 Monday  01/11/…             877             1027          952
2 Tuesday 01/12/…             777              815          796
3 Wednes… 01/13/…            1203             1220         1212.
4 Thursd… 01/14/…            1194             1233         1214.
5 Friday  01/15/…            1645             1643         1644
6 Saturd… 01/16/…            1457             1524         1490.
# … with 9 more variables: purpleAlightings <dbl>, purpleAverage <dbl>,
#   greenBoardings <dbl>, greenAlightings <dbl>, greenAverage <dbl>,
#   bannerBoardings <dbl>, bannerAlightings <dbl>, bannerAverage <dbl>,
#   daily <dbl>
```

Head and Tail Commands

The `head`/`tail` commands displays the first/last `6` (default) rows:

```
tail(circ, 10)
```

```
# A tibble: 10 x 15
   day      date    orangeBoardings orangeAlightings orangeAverage purpleBoardir
   <chr>    <chr>             <dbl>            <dbl>         <dbl>         <db
 1 Wednes… 02/20…            3374             3491          3432.          47
 2 Thursd… 02/21…            3569             3705          3637           47
 3 Friday  02/22…            3910             4006          3958           42
 4 Saturd… 02/23…            3456             3669          3562.          27
 5 Sunday  02/24…            2128             2079          2104.          23
 6 Monday  02/25…            3962             3987          3974.          51
 7 Tuesday 02/26…            3423             3487          3455           49
 8 Wednes… 02/27…            3974             4063          4018.          49
 9 Thursd… 02/28…            3820             3966          3893           48
10 Friday  03/01…            4506             4449          4478.          54
# … with 9 more variables: purpleAlightings <dbl>, purpleAverage <dbl>,
#   greenBoardings <dbl>, greenAlightings <dbl>, greenAverage <dbl>,
#   bannerBoardings <dbl>, bannerAlightings <dbl>, bannerAverage <dbl>,
#   daily <dbl>
```

Data Summarization

- Basic statistical summarization is key after cleaning data!
  - `mean(x)`: takes the mean of x
  - `sd(x)`: takes the standard deviation of x
  - `median(x)`: takes the median of x
  - `quantile(x)`: displays sample quantiles of x. Default is min, IQR, max
  - `range(x)`: displays the range. Same as `c(min(x), max(x))`
  - `sum(x)`: sum of x
  - **all have a** `na.rm` for missing data
- Transformations
  - `log`, `log2`, `log10` - log transformation
  - `sqrt` - square root

Statistical summarization

Remember `NA` is “missing” so it’s unknown what the mean or sum of something is (by default). `na.rm` argument (“remove NAs”).

```r
mean(circ$daily)
```

```
[1] NA
```

```r
sum(circ$daily)
```

```
[1] NA
```

```r
mean(circ$daily, na.rm = TRUE)
```

```
[1] 7233.48
```

```r
sum(circ$daily, na.rm = TRUE)
```

```
[1] 7392617
```

Statistical summarization

```
quantile(circ$daily, na.rm = TRUE)
```

```
      0%      25%      50%      75%     100% 
    0.00  4293.25  6701.75 10500.75 22074.50 
```

```
quantile(circ$daily, na.rm = TRUE, probs = c(0.6, 0.84))
```

```
     60%      84% 
 8208.00 12045.92 
```

```
median(circ$daily, na.rm = TRUE)
```

```
[1] 6701.75
```

Length and unique

`unique(x)` will return the unique elements of `x`

```
unique(circ$day)
```

```
[1] "Monday"    "Tuesday"   "Wednesday" "Thursday"  "Friday"    "Saturday"
[7] "Sunday"
```

`length` will tell you the length of a vector. Combined with `unique`, tells you the number of unique elements:

```
length(unique(circ$date))
```

```
[1] 1146
```

Table

`table(x)` will return a frequency table of unique elements of `x`

```
table(circ$day)
```

```
   Friday    Monday  Saturday    Sunday  Thursday   Tuesday Wednesday 
      164       164       163       163       164       164       164 
```

The `tidy` way: `dplyr`: `count`

```
circ %>% count(day)
```

```
# A tibble: 7 x 2
  day           n
  <chr>     <int>
1 Friday      164
2 Monday      164
3 Saturday    163
4 Sunday      163
5 Thursday    164
6 Tuesday     164
7 Wednesday   164
```

The tidy way: dplyr: count

```
circ %>% mutate(many_riders = daily > 1000) %>% count(many_riders, day)
```

```
# A tibble: 21 x 3
   many_riders day           n
   <lgl>       <chr>     <int>
 1 FALSE       Friday        1
 2 FALSE       Monday        5
 3 FALSE       Saturday      6
 4 FALSE       Sunday       13
 5 FALSE       Thursday      2
 6 FALSE       Tuesday       4
 7 FALSE       Wednesday     2
 8 TRUE        Friday      145
 9 TRUE        Monday      141
10 TRUE        Saturday    140
# … with 11 more rows
```

Summarize the data: **`dplyr summarize/summarise`** function

`dplyr::summarise` will allow you to summarize data. Format is `new_column = SUMMARY`. If you don't set a `new` name, it will be a messy output:

```
circ %>%
  summarize(mean_purple = mean(purpleAverage, na.rm = TRUE),
            mean(bannerAverage, na.rm = TRUE))
```

```
# A tibble: 1 x 2
  mean_purple `mean(bannerAverage, na.rm = TRUE)`
        <dbl>                               <dbl>
1       4017.                                827.
```

`across` - summarize multiple columns!

If you would like to a bunch of columns, you can use `across` and pass in a function (with other arguments) with select helpers:

```
circ %>% summarise(across(ends_with("Boardings"), mean, na.rm = TRUE))
```

```
# A tibble: 1 x 4
  orangeBoardings purpleBoardings greenBoardings bannerBoardings
            <dbl>           <dbl>          <dbl>           <dbl>
1           3031.           4127.          1929.            830.
```

Perform Operations By Groups: dplyr

`group_by` allows you group the data set by grouping variables:

```
sub_circ = circ %>% group_by(day)
head(sub_circ)
```

```
# A tibble: 6 x 15
# Groups:   day [6]
  day      date     orangeBoardings orangeAlightings orangeAverage purpleBoardi
  <chr>    <chr>              <dbl>            <dbl>         <dbl>            <d
1 Monday   01/11/…              877             1027          952
2 Tuesday  01/12/…              777              815          796
3 Wednes… 01/13/…             1203             1220         1212.
4 Thursd… 01/14/…             1194             1233         1214.
5 Friday   01/15/…             1645             1643         1644
6 Saturd… 01/16/…             1457             1524         1490.
# … with 9 more variables: purpleAlightings <dbl>, purpleAverage <dbl>,
#   greenBoardings <dbl>, greenAlightings <dbl>, greenAverage <dbl>,
#   bannerBoardings <dbl>, bannerAlightings <dbl>, bannerAverage <dbl>,
#   daily <dbl>
```

- doesn't change the data in any way, but how **functions operate on it**

Summarize the data

It’s grouped!

```
sub_circ %>% summarize(avg_daily = mean(daily, na.rm = TRUE))
```

```
# A tibble: 7 x 2
  day       avg_daily
  <chr>         <dbl>
1 Friday        8961.
2 Monday        7340.
3 Saturday      6743.
4 Sunday        4531.
5 Thursday      7639.
6 Tuesday       7642.
7 Wednesday     7779.
```

Using the **`pipe`**

Pipe `sub_circ` into `group_by`, then pipe that into `summarise`:

```
day_avgs = circ %>%
  group_by(day) %>%
  summarize(avg_daily = mean(daily, na.rm = TRUE))
head(day_avgs)
```

```
# A tibble: 6 x 2
  day       avg_daily
  <chr>         <dbl>
1 Friday        8961.
2 Monday        7340.
3 Saturday      6743.
4 Sunday        4531.
5 Thursday      7639.
6 Tuesday       7642.
```

Ungroup the data

You usually want to perform operations on groups and may want to redefine the groups. The `ungroup` function will allow you to clear the groups from the data:

```
sub_circ = ungroup(sub_circ)
sub_circ
```

```
# A tibble: 1,146 x 15
   day     date   orangeBoardings orangeAlightings orangeAverage purpleBoardir
   <chr>   <chr>            <dbl>            <dbl>         <dbl>           <db
 1 Monday  01/11…             877             1027          952
 2 Tuesday 01/12…             777              815          796
 3 Wednes… 01/13…            1203             1220         1212.
 4 Thursd… 01/14…            1194             1233         1214.
 5 Friday  01/15…            1645             1643         1644
 6 Saturd… 01/16…            1457             1524         1490.
 7 Sunday  01/17…             839              938          888.
 8 Monday  01/18…             999             1000         1000.
 9 Tuesday 01/19…            1023             1047         1035
10 Wednes… 01/20…            1375             1416         1396.
# … with 1,136 more rows, and 9 more variables: purpleAlightings <dbl>,
#   purpleAverage <dbl>, greenBoardings <dbl>, greenAlightings <dbl>,
#   greenAverage <dbl>, bannerBoardings <dbl>, bannerAlightings <dbl>,
#   bannerAverage <dbl>, daily <dbl>
```

`group_by` with `mutate`

We can use `mutate` instead of `summarise` to add the summary back in to the original data!

```
circ %>%
  group_by(day) %>%
  mutate(mean = mean(daily, na.rm = TRUE)) %>%
  select(day, date, mean, daily)
```

```
# A tibble: 1,146 x 4
# Groups:   day [7]
   day       date        mean daily
   <chr>     <chr>      <dbl> <dbl>
 1 Monday    01/11/2010 7340.  952 
 2 Tuesday   01/12/2010 7642.  796 
 3 Wednesday 01/13/2010 7779. 1212.
 4 Thursday  01/14/2010 7639. 1214.
 5 Friday    01/15/2010 8961. 1644 
 6 Saturday  01/16/2010 6743. 1490.
 7 Sunday    01/17/2010 4531.  888.
 8 Monday    01/18/2010 7340. 1000.
 9 Tuesday   01/19/2010 7642. 1035 
10 Wednesday 01/20/2010 7779. 1396.
# … with 1,136 more rows
```

Counting with `n()`

Standard statistics can be calculated: `n()` counts the number of observations.

```
circ %>%
  group_by(day) %>%
  summarize(n = n(),
            mean = mean(daily, na.rm = TRUE)) %>%
  head
```

```
# A tibble: 6 x 3
  day           n  mean
  <chr>     <int> <dbl>
1 Friday      164 8961.
2 Monday      164 7340.
3 Saturday    163 6743.
4 Sunday      163 4531.
5 Thursday    164 7639.
6 Tuesday     164 7642.
```

Bonus material

Statistical summarization

`t.test` is good for t-tests, but also gives a mean and 95% CI:

```
t.test(circ$daily)
```

```

    One Sample t-test

data:  circ$daily
t = 56.642, df = 1021, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 6982.884 7484.076
sample estimates:
mean of x
  7233.48
```

```
broom::tidy(t.test(circ$daily))
```

```
# A tibble: 1 x 8
  estimate statistic   p.value parameter conf.low conf.high method   alternati
     <dbl>     <dbl>     <dbl>     <dbl>    <dbl>     <dbl> <chr>    <chr>
1    7233.      56.6 2.27e-317      1021    6983.     7484. One Sam… two.sidec
```

Data Summarization on matrices/data frames

- Basic statistical summarization
  - `rowMeans(x)`: takes the means of each row of x
  - `colMeans(x)`: takes the means of each column of x
  - `rowSums(x)`: takes the sum of each row of x
  - `colSums(x)`: takes the sum of each column of x
  - `summary(x)`: for data frames, displays the quantile information
- The `matrixStats` package has additional `row*` and `col*` functions
  - Like `rowSds`, `colQuantiles`

Column and Row means

`colMeans` and `rowMeans` must work on **all numeric data**. We will subset the boardings

```
avgs = circ %>% select(ends_with("Boardings"))
colMeans(avgs, na.rm = TRUE)
```

```
orangeBoardings purpleBoardings  greenBoardings bannerBoardings 
      3031.1196       4127.3964       1928.9979        829.5963 
```

```
circ = circ %>% mutate(mean_boarding = rowMeans(avgs, na.rm = TRUE))
head(circ %>% select(day, mean_boarding))
```

```
# A tibble: 6 x 2
  day       mean_boarding
  <chr>             <dbl>
1 Monday              877
2 Tuesday             777
3 Wednesday          1203
4 Thursday           1194
5 Friday             1645
6 Saturday           1457
```

Basic Plots

Plotting is an important component of exploratory data analysis.

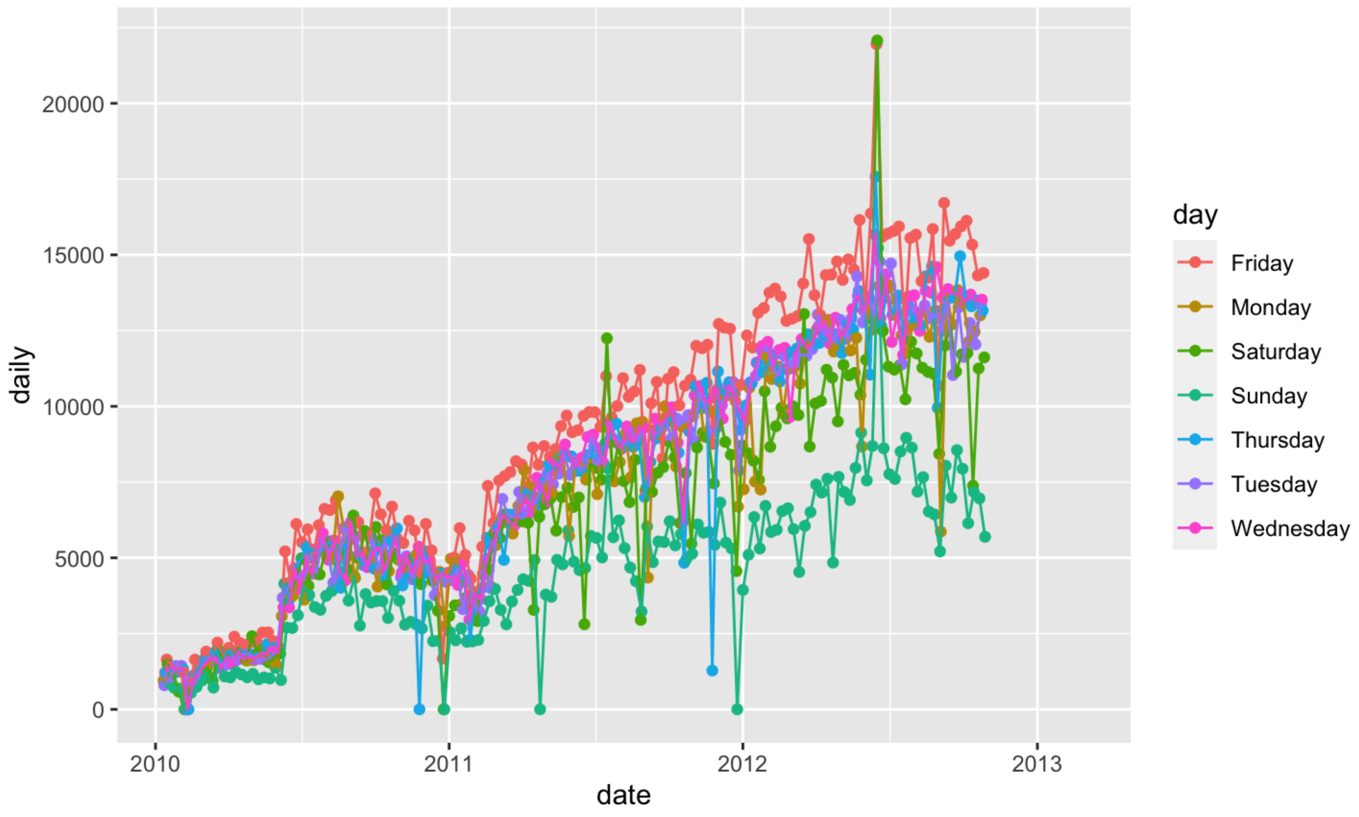
`ggplot2` is a package of plotting that is very popular and powerful (using the **g**rammar of **g**raphics). We will use `qplot` (“quick plot”) for most of the basic examples:

```
qplot
```

```
function (x, y, ..., data, facets = NULL, margins = FALSE, geom = "auto",
    xlim = c(NA, NA), ylim = c(NA, NA), log = "", main = NULL,
    xlab = NULL, ylab = NULL, asp = NA, stat = NULL, position = NULL)
NULL
```

Scatterplot

```r
library(ggplot2)
circ %>%
  mutate(date = lubridate::mdy(date)) %>%
  qplot(x = date, y = daily, colour = day, data = .) + geom_line()
```


daily
date
day
Friday
Monday
Saturday
Sunday
Thursday
Tuesday
Wednesday