TEXAS McCombs

MASTER OF SCIENCE IN BUSINESS ANALYTICS

Data wrangling basics

Outline

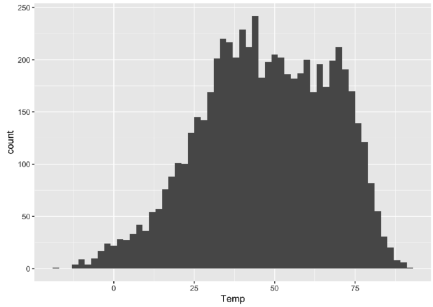
Complex summaries: two concrete examples

Six key data verbs

Summaries

- → typical value: what's the temperature on a typical day in Rapid City? (mean, median)
- → shape: is the distribution symmetric or skewed? Does it have multiple peaks?
- → variation: how much do the individual days vary from a "typical" day? (sd, IQR)
- → extremes: what temperatures should we expect on days that are unusually hot or unusually cold? (min, max, quantile, z-score)

```
ggplot(rapidcity) +
  geom_histogram(aes(x=Temp), binwidth=2)
```



An important note about summaries

Variation is reality.
Averages are abstractions.
Always plot your data.

Data wrangling: An example

- Consider a question like this:

  What were the five coldest individual months in Rapid City between 1995 and 2011, as measured by the average daily temperature? What were the lowest and highest daily temperatures within each of those months?

  At right are 10 random rows from the data set.

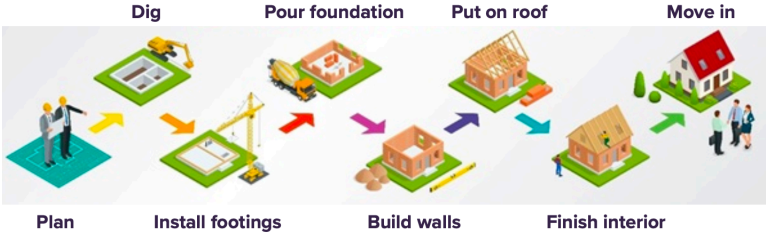
- How would you go about this complex task?

10 random rows

Year	Month	Day	Temp
1996	8	2	78.3
1996	9	15	59.6
1998	2	1	33.2
2000	6	2	50.8
2001	8	26	76.8
2003	9	8	71.9
2003	12	6	35.5
2005	12	16	18.6
2010	6	20	68.2
2010	10	16	53.2

+ more rows

An analogy: Building a house



There's only one way to manage something so complex:

→ Break down the complex task into simpler tasks.
→ Sequence the tasks so that each one builds on prior tasks and feeds into subsequent tasks.

Data wrangling: An example

- Consider a question like this:

  What were the five coldest individual months in Rapid City between 1995 and 2011, as measured by the average daily temperature? What were the lowest and highest daily temperatures within each of those months?

  At right are 10 random rows from the data set.

- **Remember our advice - break down the complex tasks into simpler ones!**

10 random rows

Year	Month	Day	Temp
1996	8	2	78.3
1996	9	15	59.6
1998	2	1	33.2
2000	6	2	50.8
2001	8	26	76.8
2003	9	8	71.9
2003	12	6	35.5
2005	12	16	18.6
2010	6	20	68.2
2010	10	16	53.2

+ more rows

Data wrangling: An example

Here are the simple tasks:

→ **Group** the data set into individual months in individual years: January 1995, February 1995, ..., all the way through December 2011.

10 random rows

Year	Month	Day	Temp
1996	8	2	78.3
1996	9	15	59.6
1998	2	1	33.2
2000	6	2	50.8
2001	8	26	76.8
2003	9	8	71.9
2003	12	6	35.5
2005	12	16	18.6
2010	6	20	68.2
2010	10	16	53.2

+ more rows

Data wrangling: An example

Here are the simple tasks:

→ **Group** the data set into individual months in individual years: January 1995, February 1995, ..., all the way through December 2011.

→ **Summarize** each individual month by calculating the average, min, and max of the Temp variable.

January 2007

Year	Month	Day	Temp
2007	1	1	24.2
2007	1	2	27.6
2007	1	3	40.3
2007	1	4	39.2
2007	1	5	31.7
2007	1	6	23.5
2007	1	7	31.1
2007	1	8	37.0
2007	1	9	32.7
2007	1	10	41.2

+ more rows

mean(Temp)	min(Temp)	max(Temp)
26.28065	0.6	42.2

Data wrangling: An example

Here are the simple tasks:

→ **Group** the data set into individual months in individual years: January 1995, February 1995, ..., all the way through December 2011.

→ **Summarize** each individual month by calculating the average, min, and max of the Temp variable. Put all of the monthly summaries in a table

Monthly summaries

Year	Month	mean_Temp	min_Temp	max_Temp
1995	1	28.0	6.2	50.9
1995	2	31.8	2.7	54.0
1995	3	33.1	-2.1	55.9
1995	4	40.0	24.4	50.8
1995	5	51.5	38.6	59.4
1995	6	63.1	43.3	77.1
1995	7	70.2	58.9	79.6
1995	8	73.1	58.6	85.0
1995	9	60.2	35.2	79.3
1995	10	46.1	29.4	65.8
1995	11	33.6	8.5	52.0
1995	12	24.7	-2.1	47.6
1996	1	14.9	-11.0	46.1
1996	2	26.8	-19.0	47.9
1996	3	25.9	-4.0	45.8
1996	4	43.5	27.2	62.6
1996	5	50.1	32.5	65.6
1996	6	65.4	54.0	77.8
1996	7	71.5	62.3	80.3
1996	8	72.8	66.1	80.8
1996	9	59.2	39.5	81.7
1996	10	47.1	25.5	64.1
1996	11	24.2	6.5	43.4
1996	12	17.5	-10.8	40.4
1997	1	18.0	-10.2	45.1

Data wrangling: An example

Here are the simple tasks:

→ **Group** the data set into individual months in individual years: January 1995, February 1995, ..., all the way through December 2011.

→ **Summarize** each individual month by calculating the average, min, and max of the Temp variable. Put all of the monthly summaries in a table

→ **Arrange** the months by mean temperature and examine the top 5.

Monthly summaries

Year	Month	mean_Temp	min_Temp	max_Temp
1996	1	14.9	-11.0	46.1
2009	12	16.4	-2.6	35.6
2000	12	17.3	-9.0	38.8
1996	12	17.5	-10.8	40.4
2001	2	17.6	-3.9	40.8
1997	1	18.0	-10.2	45.1
2008	12	18.1	-12.2	37.1
2011	2	19.9	-8.7	49.9
2011	1	20.6	-3.0	42.3
2010	2	20.7	2.7	34.7
2007	2	21.1	-9.9	41.5
2010	1	21.3	-9.7	39.4
2005	1	21.4	-3.9	50.0
2008	1	22.0	-0.7	44.7
2003	2	22.4	-5.0	44.9
2004	1	22.5	-5.6	40.7
2002	3	22.7	2.5	48.2
2007	12	23.4	11.0	48.4
1998	1	24.1	-4.9	53.1
1996	11	24.2	6.5	43.4
2010	12	24.2	0.9	39.8
1995	12	24.7	-2.1	47.6
1999	1	24.8	3.6	40.5
2009	1	24.9	-3.9	46.2
2005	12	25.1	-0.9	47.8

The result!

The five coldest months in Rapid City, 1996-2011

Year	Month	mean_Temp	min_Temp	max_Temp
1996	1	14.9	-11.0	46.1
2009	12	16.4	-2.6	35.6
2000	12	17.3	-9.0	38.8
1996	12	17.5	-10.8	40.4
2001	2	17.6	-3.9	40.8

This is data wrangling

- Our original data set wasn't in the form we needed to answer the question directly. This is the rule, rather than the exception, in data science.
- To get our data into a form where we could answer to our question, we had to break our complex task into simpler tasks:

  Identify the simple tasks (group, summarize, arrange, etc.)
  Sequence those tasks in the right order.
- This process is part of what's called data wrangling.

  → Data wrangling: the process of getting your data into a useful form for visualization, summary, and modeling.
  → Wrangling is an huge part of data science, because data rarely comes in precisely the form that suits some particular analysis.

How does this look in R

```
rapidcity %>%
  group_by(Year, Month) %>%
  summarize(avg_temp = mean(Temp),
            coldest_day = min(Temp),
            warmest_day = max(Temp)) %>%
  arrange(avg_temp) %>%
  head(5) %>%
  round(1)
```

Group the data set according to all combos of Year and Month

How does this look in R

```
rapidcity %>%
  group_by(Year, Month) %>%
  summarize(avg_temp = mean(Temp),
            coldest_day = min(Temp),
            warmest_day = max(Temp)) %>%
  arrange(avg_temp) %>%
  head(5) %>%
  round(1)
```

Calculate summary statistics for each group

Right-hand side: the summary we want to calculate

How does this look in R

```
rapidcity %>%
  group_by(Year, Month) %>%
  summarize(avg_temp = mean(Temp),
            coldest_day = min(Temp),
            warmest_day = max(Temp)) %>%
  arrange(avg_temp) %>%
  head(5) %>%
  round(1)
```

Calculate summary statistics for each group

Left-hand side: the name we want to give to each summary

How does this look in R

```
rapidcity %>%
  group_by(Year, Month) %>%
  summarize(avg_temp = mean(Temp),
            coldest_day = min(Temp),
            warmest_day = max(Temp)) %>%
  arrange(avg_temp) %>%
  head(5) %>%
  round(1)
```

Calculate summary statistics for each group

Multiple summaries separated by commas

How does this look in R

```
rapidcity %>%
  group_by(Year, Month) %>%
  summarize(avg_temp = mean(Temp),
            coldest_day = min(Temp),
            warmest_day = max(Temp)) %>%
  arrange(avg_temp) %>%
  head(5) %>%
  round(1)
```

Arrange the table in ascending order of `avg_temp` summary

How does this look in R

```
rapidcity %>%
  group_by(Year, Month) %>%
  summarize(avg_temp = mean(Temp),
            coldest_day = min(Temp),
            warmest_day = max(Temp)) %>%
  arrange(avg_temp) %>%
  head(5) %>%
  round(1)
```

Take the first 5 entries in the table and round to 1 decimal place.

How does this look in R

```
rapidcity %>%
  group_by(Year, Month) %>%
  summarize(avg_temp = mean(Temp),
            coldest_day = min(Temp),
            warmest_day = max(Temp)) %>%
  arrange(avg_temp) %>%
  head(5) %>%
  round(1)
```

```
##    Year Month avg_temp coldest_day warmest_day
##   <dbl> <dbl>    <dbl>       <dbl>       <dbl>
## 1  1996     1     14.9         -11        46.1
## 2  2009    12     16.4          -2.6      35.6
## 3  2000    12     17.3          -9        38.8
## 4  1996    12     17.5         -10.8      40.4
## 5  2001     2     17.6          -3.9      40.8
```

The result

Remember our mantra ...

Manage complexity by breaking down complex tasks into simpler tasks.

Six key data verbs

1. `summarize`, for calculating summary statistics
2. `group_by`, for splitting a data set into groups
3. `filter`, for looking at specific rows (cases)
4. `select`, for looking at specific columns (variables)
5. `mutate`, for defining new variables from old ones
6. `arrange`, for sorting according to some specific variable

summarize

You've met this before: it's use to calculate summary statistics

```
rapidcity %>%
  summarize(avg_temp = mean(Temp),
            median_temp = median(Temp),
            sd_temp = sd(Temp),
            iqr_temp = IQR(Temp),
            min_temp = min(Temp),
            max_temp = max(Temp))
```

Right-hand side: the summary we want to calculate

Left-hand side: the name we want to give that summary

```
##   avg_temp median_temp  sd_temp iqr_temp min_temp max_temp
## 1 47.28159        47.6 20.05404    30.65      -19     91.9
```

group_by

Used to split the rows of a data set into groups

Specify groups with `group_by()`, then use `summarize()` to calculate something for each group, and return it in a nice table

Biggest powerhouse combo in R

```
rapidcity %>%
  group_by(Month) %>%
  summarize(avg_temp = mean(Temp),
            sd_temp = sd(Temp)) %>%
  round(1)
```

	Month	avg_temp	sd_temp
	<dbl>	<dbl>	<dbl>
1	1	24.4	13.5
2	2	27.4	13
3	3	34.2	12.7
4	4	44.5	9.7
5	5	54.3	8.3
6	6	64.3	7.7
7	7	73.7	6.6
8	8	71.9	6.1
9	9	61.4	9.1
10	10	47.9	9.7
11	11	35.1	11.5
12	12	25.7	12.4

filter

Keep rows that satisfy your conditions; ignore everything else

```
rapidcity2009 = rapidcity %>%
  filter(Year == 2009)
```

```
head(rapidcity2009)
```

```
##   Year Month Day Temp
## 1 2009     1   1 30.7
## 2 2009     1   2 20.3
## 3 2009     1   3 16.9
## 4 2009     1   4  8.0
## 5 2009     1   5 13.9
## 6 2009     1   6 28.2
```

The double-equals sign (==) inside filter is used to test for equality. That is, we are filtering the data frame to include only those cases where the Year variable is equal to 2009.

select

Used to select specific columns (variables) in your data frame

A frequent use case is to de-clutter output

```
rapidcity2009 %>%
  select(Month, Day, Temp) %>%
  head(5)
```

```
##   Month Day Temp
## 1     1   1 30.7
## 2     1   2 20.3
## 3     1   3 16.9
## 4     1   4  8.0
## 5     1   5 13.9
```

mutate

Add a column defined in terms of existing columns


```
rapidcity = rapidcity %>%
  mutate(Summer = ifelse(Month == 6 | Month == 7 | Month == 8,
                         yes="summer", no=“not_summer"))

sample_n(rapidcity, 5)
```

```
Year Month Day Temp     Summer
1 2003    12   6 35.5 not_summer
2 2001     8  26 76.8     summer
3 2005    12  16 18.6 not_summer
4 1996     9  15 59.6 not_summer
5 2010     6  20 68.2     summer
```

arrange

Used to sort according to a variable or set of variables

```
rapidcity %>%
  arrange(Temp) %>%
  head(10)
```

```
##    Year Month Day  Temp
## 1  1996     2   2 -19.0
## 2  2008    12  15 -12.2
## 3  1996     2   3 -11.8
## 4  2006     2  18 -11.5
## 5  1996     1  30 -11.0
## 6  1996    12  26 -10.8
## 7  1996     1  19 -10.6
## 8  1996    12  24 -10.6
## 9  1996     1  29 -10.4
## 10 1997     1  11 -10.2
```

ascending order

```
rapidcity %>%
  arrange(desc(Temp)) %>%
  head(10)
```

```
##    Year Month Day Temp
## 1  2007     7   7 91.9
## 2  2006     7  16 90.7
## 3  2006     7  30 89.8
## 4  2007     7  23 89.5
## 5  2007     7  24 89.5
## 6  2002     6  29 89.4
## 7  2002     7  15 89.3
## 8  2006     7  15 89.0
## 9  2003     8  23 88.9
## 10 2002     7  16 88.4
```

descending order