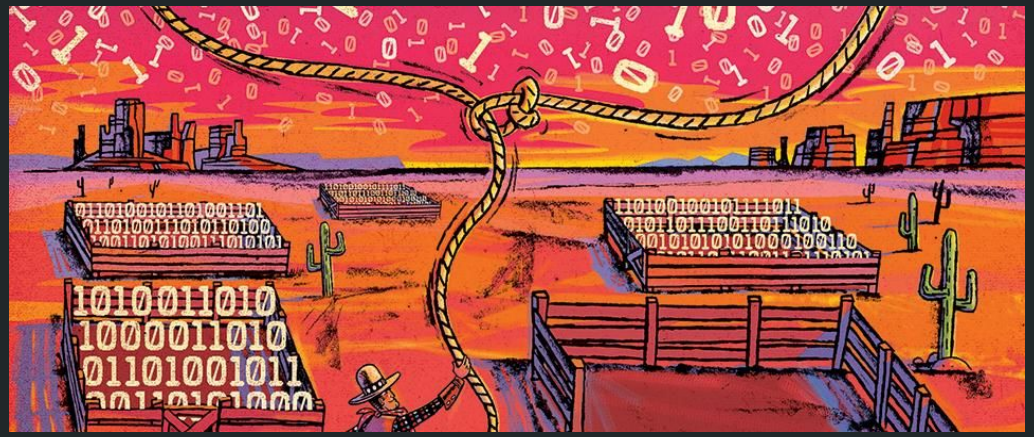
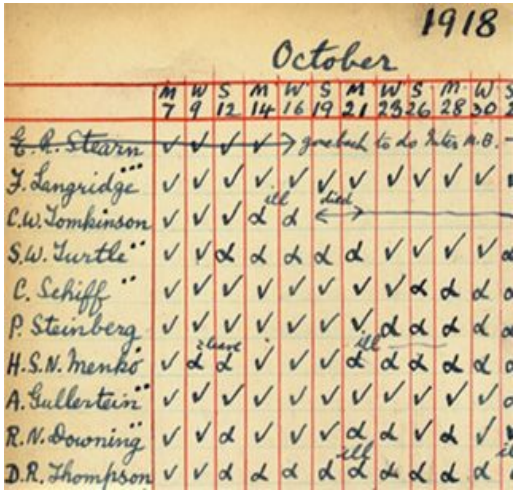
Data Wrangling



Some definitions

- A data table is a collection of measurements
- A variable is a column in a data table
- An observation is a row in a data table
- A measurement, or scalar, is a value for each (observation, variable) pair



	A	B	C	D	E	F	G
7			Monday	Tuesday	Wednesday	Thursday	Friday
8		Student Name	15 Nov-15	16 Nov-15	17 Nov-15	18 Nov-15	19 Nov-15
9	01	Lee, John	PR	PR	PR	PR	PR
10	02	Williams, Adam	PR	PR	PR	PR	PR
11	03	Williams, Sarah	PR	PR	PR	PR	PR
12	04	Doe, Jane	PR	AB	AB	PR	AB
13	05	Liu, Kat	PR	AB	PR	PR	PR
14	06	Smith, Elizabeth	PR	PR	PR	PR	PR
15	07	Sanders, Shane	AB	PR	PR	PR	PR
16	08	White, Andrea	PR	PR	PR	PR	PR
17	09	Aaronson, Alison	PR	PR	PR	PR	PR
18	10	Maggie, Noah	PR	PR	PR	PR	PR
19	11	Grande, John	PR	PR	PR	PR	PR
20	12	McDonell, Josh	PR	PR	PR	PR	PR
21	13	Spear, Terry	PR	PR	PR	PR	PR
22	14	Greene, Tom	PR	PR	PR	PR	PR
23	15	Snow, Daniel	PR	PR	PR	PR	PR
24	16	Daniels, Jack	PR	PR	AB	PR	PR
25	17	Daniels, Bob	PR	PR	PR	PR	PR
26	18	West, Williams	PR	PR	PR	PR	PR
27	19						
28	20						
29	21						
30	22						
31	23						
32	24						
33	25						
34	26						
35	27						
36	28						
37	29						
38	30						
39		# PResent per Day:	17	16	16	18	17
40	18	# ABsent per Day:	1	2	2	0	1
41		Total:	18	18	18	18	18
42		Daily Attendance %:	94	89	89	100	94

ATTENDANCE:
Select,
PR if Present
AB if Absent

Data wrangling

- Reduction/Aggregation: reduces a variable to a scalar
  - Summarizing (e.g., How many students are in the class? And how many had perfect attendance?)
- Transformation/Mutation: creates a new variable based on one or more existing variables
  - From a student's total absences and total number of school days, compute the proportion of days the student was absent

Data wrangling (cont'd)

- Data verb: transforms a data table into a new one, usually by applying a reduction or transformation
  - Adding or deleting variables or observations
  - Sorting a variable (e.g., names) in, say, ascending order
  - Filtering a variable (e.g., who was absent on the day of the test?)
- Grouping data is another data verb whereby data are grouped before a reduction or transformation is applied
  - "Group by" students whose last name begins with A to M and N to Z

dplyr

What is dplyr?

- An R package full of data verbs
- Some examples of things it can do
  - Select (variables)
  - Filter (observations)
  - Sort (rearrange data)
  - Summarize (e.g., mean)
  - Transform (e.g., add columns)
  - The all-powerful `group_by` operation

Aside: packages

- A package is a collection of functions which are not built-in to a language, but which can be imported easily, and then used as if they were built-in.
- To install a package into R, type the following in the console:
  - `install.packages("<packagename>")`
- Then, to load the package `library(<packagename>)`
- For example, the `datasets` package has loads of built-in data sets
  - `install.packages("datasets")`
  - `library(datasets)`
  - `install.packages("dplyr")`
  - `library(dplyr)`

Our data: A survey sent to Slovakian youth in 2013

- The data set consists of responses to 150 questions, covering topics such as music/movie preferences, hobbies and interests, spending habits, etc.
- Contains 1010 rows and 150 columns

HEALTH HABITS

1. **Smoking habits**: Never smoked - Tried smoking - Former smoker - Current smoker (categorical)
2. **Drinking**: Never - Social drinker - Drink a lot (categorical)
3. **I live a very healthy lifestyle.**: Strongly disagree 1-2-3-4-5 Strongly agree (integer)

PERSONALITY TRAITS, VIEWS ON LIFE & OPINIONS

1. **I take notice of what goes on around me.**: Strongly disagree 1-2-3-4-5 Strongly agree (integer)
2. **I try to do tasks as soon as possible and not leave them until last minute.**: Strongly disagree 1-2-3-4-5 Strongly agree (integer)
3. **I always make a list so I don't forget anything.**: Strongly disagree 1-2-3-4-5 Strongly agree (integer)
4. **I often study or work even in my spare time.**: Strongly disagree 1-2-3-4-5 Strongly agree (integer)
5. **I look at things from all different angles before I go ahead.**: Strongly disagree 1-2-3-4-5 Strongly agree (integer)
6. **I believe that bad people will suffer one day and good people will be rewarded.**: Strongly disagree 1-2-3-4-5 Strongly agree (integer)
7. **I am reliable at work and always complete all tasks given to me.**: Strongly disagree 1-2-3-4-5 Strongly agree (integer)
8. **I always keep my promises.**: Strongly disagree 1-2-3-4-5 Strongly agree (integer)
9. **I can fall for someone very quickly and then completely lose interest.**: Strongly disagree 1-2-3-4-5 Strongly agree (integer)
10. **I would rather have lots of friends than lots of money.**: Strongly disagree 1-2-3-4-5 Strongly agree (integer)
11. **I always try to be the funniest one.**: Strongly disagree 1-2-3-4-5 Strongly agree (integer)
12. **I can be two faced sometimes.**: Strongly disagree 1-2-3-4-5 Strongly agree (integer)

A sample of questions from the survey. Complete information is available here.

Sample survey questions

Column Name	Question
Reliability	I am reliable at work and always complete all tasks given to me.
Keeping promises	I always keep my promises.
Loss of interest	I can fall for someone very quickly and then completely lose interest.
Friends versus money	I would rather have lots of friends than lots of money.
Funniness	I always try to be the funniest one.
Fake	I can be two faced sometimes.
Criminal damage	I damaged things in the past when angry.
Decision making	I take my time making decisions.
Empathy	I am an empathetic person.

An excerpt of the data

Reliability	Keeping promises	Loss of interest	Friends versus money	Funniness	Fake	Criminal damage	Decision making
4	4	1	3	5	1	1	3
4	4	3	4	3	2	1	2
4	5	1	5	2	4	1	3
3	4	5	2	1	1	5	5
5	4	2	3	3	2	1	3
3	4	3	2	3	1	4	2
4	5	3	4	4	1	2	2
3	3	1	4	4	2	1	3
5	4	1	4	2	2	1	4
4	5	3	4	3	1	2	5
4	4	1	3	2	1	1	5
3	3	3	3	5	3	5	3
5	5	4	4	3	1	2	5
5	4	3	3	3	2	2	5
5	4	1	5	4	1	4	5
5	4	5	3	5	1	5	3
2	3	5	4	5	3	3	5
5	3	2	2	3	5	5	2
4	5	3	5	3	2	1	3
4	5	3	4	3	1	3	4
2	4	2	1	3	1	4	2
4	4	2	3	4	2	3	1
4	3	1	3	3	3	1	5
4	4	2	3	4	3	1	4
3	5	1	5	2	2	5	3
5	4	5	4	3	2	1	5
4	4	3	5	5	2	1	4
4	2	2	4	3	1	1	1
4	4	4	2	4	2	5	2

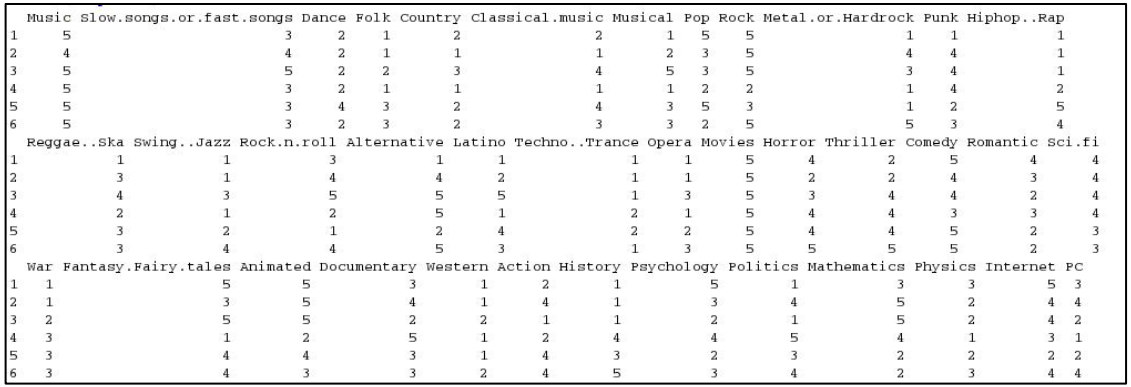
Importing data into RStudio

To store the data set in a data frame called `responses`:

```
responses <- read.table("~/R/dplyr/responses.csv", sep = ";", header = TRUE)
```

To see the head, or the first six rows, of `responses`:

```
head(responses)
```



	Music	Slow.songs.or.fast.songs	Dance	Folk	Country	Classical.music	Musical	Pop	Rock	Metal.or.Hardrock	Punk	Hiphop..Rap
1	5		3	2	1	2	2	1	5	5	1	1	1
2	4		4	2	1	1	1	2	3	5	4	4	1
3	5		5	2	2	3	4	5	3	5	3	4	1
4	5		3	2	1	1	1	1	2	2	1	4	2
5	5		3	4	3	2	4	3	5	3	1	2	5
6	5		3	2	3	2	3	3	2	5	5	3	4

	Reggae..Ska	Swing..Jazz	Rock.n.roll	Alternative	Latino	Techno..Trance	Opera	Movies	Horror	Thriller	Comedy	Romantic	Sci.fi
1	1	1	3	1	1	1	1	5	4	2	5	4	4
2	3	1	4	4	2	1	1	5	2	2	4	3	4
3	4	3	5	5	5	1	3	5	3	4	4	2	4
4	2	1	2	5	1	2	1	5	4	4	3	3	4
5	3	2	1	2	4	2	2	5	4	4	5	2	3
6	3	4	4	5	3	1	3	5	5	5	5	2	3

	War	Fantasy.Fairy.tales	Animated	Documentary	Western	Action	History	Psychology	Politics	Mathematics	Physics	Internet	PC
1	1	5	5	3	1	2	1	5	1	3	3	5	3
2	1	3	5	4	1	4	1	3	4	5	2	4	4
3	2	5	5	2	2	1	1	2	1	5	2	4	2
4	3	1	2	5	1	2	4	4	5	4	1	3	1
5	3	4	4	3	1	4	3	2	3	2	2	2	2
6	3	4	3	3	2	4	5	3	4	2	3	4	4

Five basic verbs in dplyr

- `select()` selects one or more columns in a data frame
- `filter()` filters rows from a data frame according to some criterion
- `arrange()` arranges rows in a data frame in ascending or descending order, based on the value(s) in one or more columns
- `group by()` and `summarize()` are usually used together, and allow you to summarize a column of values grouped by some other variable, e.g., compute the mean of all the values in a column, per group
- `mutate()` adds new columns to a data frame by transforming existing ones

The two most basic functions are `select()` and `filter()`, which select columns and filter rows, respectively.

Selecting columns with `select()`

We can select columns by listing their names explicitly: age, gender, and education

```
head(select(responses, Age, Gender, Education))
```

	Age	Gender	Education
1	20	female	college/bachelor degree
2	19	female	college/bachelor degree
3	20	female	secondary school
4	22	female	college/bachelor degree
5	20	female	secondary school
6	20	male	secondary school

And we can select a range of columns using the ‘:’ operator:

```
head(select(responses, Reliability:Decision.making))
```

	Reliability	Keeping.promises	Loss.of.interest	Friends.versus.money	Funniness	Fake	Criminal.damage	Decision.making
1	4	4	1	3	5	1	1	3
2	4	4	3	4	3	2	1	2
3	4	5	1	5	2	4	1	3
4	3	4	5	2	1	1	5	5
5	5	4	2	3	3	2	1	3
6	3	4	3	2	3	1	4	2

`select()`, cont'd

To select all columns that start with some character string, use the function `starts_with()`.

Maybe for some reason we want to select all the columns that start with the letter 'a' :

```
head(select(responses, starts_with("a")))
```

	Alternative	Animated	Action	Art.exhibitions	Active.sport	Adrenaline.sports	Ageing	Alcohol	Appearence.and.gestures
1	1	5	2	1	5	4	1	drink a lot	4
2	4	5	4	2	1	2	3	drink a lot	4
3	5	5	1	5	2	5	1	drink a lot	3
4	5	2	2	5	1	1	4	drink a lot	3
5	2	4	4	1	1	2	2	social drinker	3
6	5	3	4	2	4	3	1	never	3

	Achievements	Assertiveness	Age
1	4	1	20
2	2	2	19
3	3	3	20
4	3	5	22
5	3	4	20
6	2	4	20

`select()`, cont.

Some additional options to select columns based on a specific criteria include:

1. `ends_with()` Select columns that end with a character string
2. `contains()` Select columns that contain a character string
3. `matches()` Select columns that match a regular expression
4. `one_of()` Select columns names that are from a group of names

Selecting rows with `filter()`

Filter the rows for young people who consider themselves empathetic:

```
head(filter(responses, Empathy > 3))
```

	Life.struggles	Happiness.in.life	Energy.levels	Small...big.dogs	Personality	Finding.lost.valuables	Getting.up
1	4	4	4	3	3	3	4
2	3	3	4	4	3	3	3
3	5	5	4	3	3	2	2
4	4	3	1	5	3	5	5
5	5	4	4	1	3	3	4
6	5	4	3	2	3	2	4

	Interests.or.hobbies	Parents..advice	Questionnaires.or.polls	Internet.usage	Finances	Shopping.centres
1	5	3	1	few hours a day	2	4
2	5	3	4	few hours a day	2	3
3	4	4	5	less than an hour a day	4	3
4	1	4	3	few hours a day	2	1
5	3	3	3	few hours a day	4	4
6	3	4	4	less than an hour a day	2	4

	Branded.clothing	Entertainment.spending	Spending.on.looks	Spending.on.gadgets	Spending.on.healthy.eating	Age	Heig
1	1	4	3	4	2	20	1
2	3	3	1	4	4	20	1
3	1	3	4	1	5	20	1
4	3	2	1	3	4	18	1
5	4	2	3	2	4	19	1
6	2	3	4	2	2	19	1

	Number.of.siblings	Gender	Left...right.handed	Education	Only.child	Village...town	House...block.of.flats
1	2	female	right handed	secondary school	no	city	block of flats
2	1	male	right handed	secondary school	no	city	block of flats
3	1	female	right handed	secondary school	no	village	house/bungalow
4	1	female	right handed	secondary school	no	city	house/bungalow
5	3	female	right handed	secondary school	no	city	block of flats
6	2	female	left handed	secondary school	no	city	block of flats

Some potentially interesting findings here: These individuals skew towards agreement with the statement “I find it very difficult to get up in the morning.” All of them responded neutrally to the statement “I believe all my personality traits are positive.” And, ⅚ of them are female. (Of course, these are only the first 6 individuals who consider themselves empathetic.)

`filter()`, cont'd

Filter for young people who consider themselves empathetic and feel lonely in life:

```
head(filter(responses, Empathy > 3, Loneliness > 3))
```

	Prioritising.workload	Writing.notes	Workaholism	Thinking.ahead	Final.judgement	Reliability	Keeping.promises
1	2	5	3	5	3	4	5
2	2	1	4	2	5	5	4
3	1	5	3	1	1	3	3
4	3	5	4	5	1	5	5
5	NA	NA	5	3	1	5	4
6	4	3	4	5	3	4	5

	Loss.of.interest	Friends.versus.money	Funniness	Fake	Criminal.damage	Decision.making	Elections	Self.criticism
1	1	5	2	4	1	3	5	4
2	1	4	2	2	1	4	1	3
3	3	3	5	3	5	3	1	5
4	4	4	3	1	2	5	5	5
5	5	3	5	1	5	3	5	5
6	3	5	3	2	1	3	1	2

	Judgment.calls	Hypochondria	Empathy	Eating.to.survive	Giving	Compassion.to.animals	Borrowed.stuff	Loneliness
1	4	1	5	5	5	4	2	5
2	2	2	5	1	1	4	5	4
3	3	5	5	1	1	2	2	4
4	2	5	5	3	2	5	5	5
5	4	1	4	5	3	5	5	4
6	5	1	4	3	1	2	5	4

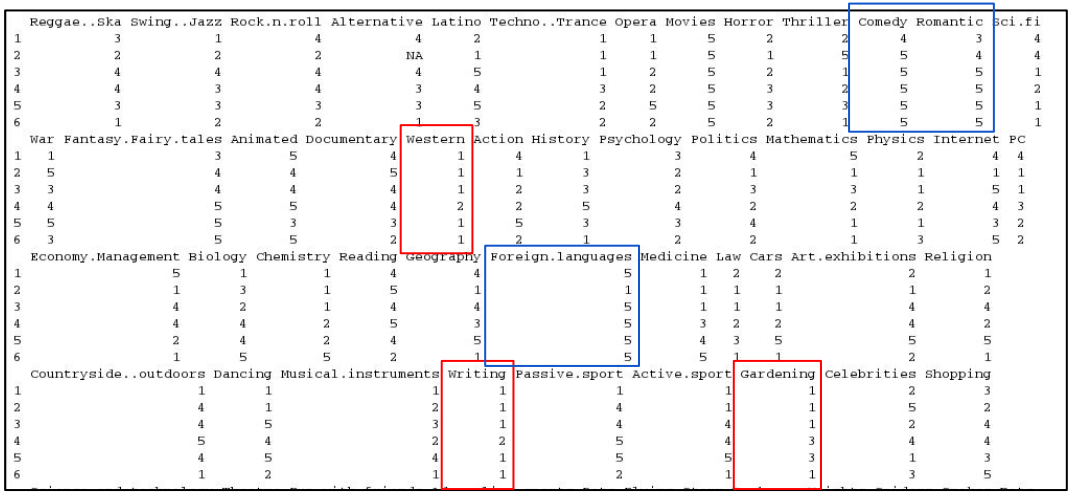
	Cheating.in.school	Health	Changing.the.past	God	Dreams	Charity	Number.of.friends	Punctuality
1	3	2	5	5	1	3	3	i am often running late
2	5	4	5	5	4	1	2	i am often early
3	5	2	3	1	3	3	4	i am often running late
4	2	5	3	1	3	2	3	i am often running late
5	5	2	5	1	3	1	3	i am always on time
6	4	3	4	5	3	1	2	i am often early

They mostly consider themselves procrastinators, would rather have friends than money, and would change the past if they could.

`filter()`, cont'd

Filter for people who are under the age of 20, enjoy meeting new people, and drink alcohol:

```
head(filter(responses, Age < 20, Socializing > 3, Alcohol %in% c("social drinker", "drink a lot")))
```



	Reggae..Ska	Swing..Jazz	Rock.n.roll	Alternative	Latino	Techno..Trance	Opera	Movies	Horror	Thriller	Comedy	Romantic	Sci.fi
1	3	1	4	4	2	1	1	5	2	2	4	3	4
2	2	2	2	NA	1	1	1	5	1	5	5	4	4
3	4	4	4	4	5	1	2	5	2	1	5	5	1
4	4	3	4	3	4	3	2	5	3	2	5	5	2
5	3	3	3	3	5	2	5	5	3	3	5	5	1
6	1	2	2	1	3	2	2	5	2	1	5	5	1

	War	Fantasy.Fairy.tales	Animated	Documentary	Western	Action	History	Psychology	Politics	Mathematics	Physics	Internet	PC
1	1	3	5	4	1	4	1	3	4	5	2	4	4
2	5	4	4	5	1	1	3	2	1	1	1	1	1
3	3	4	4	4	1	2	3	2	3	3	1	5	1
4	4	5	5	4	2	2	5	4	2	2	2	4	3
5	5	5	3	3	1	5	3	3	4	1	1	3	2
6	3	5	5	2	1	2	1	2	2	1	3	5	2

	Economy.Management	Biology	Chemistry	Reading	Geography	Foreign.languages	Medicine	Law	Cars	Art.exhibitions	Religion
1	5	1	1	4	4	5	1	2	2	2	1
2	1	3	1	5	1	1	1	1	1	1	2
3	4	2	1	4	4	5	1	1	1	4	4
4	4	4	2	5	3	5	3	2	2	4	2
5	2	4	2	4	5	5	4	3	5	5	5
6	1	5	5	2	1	5	5	1	1	2	1

	Countryside..outdoors	Dancing	Musical.instruments	Writing	Passive.sport	Active.sport	Gardening	Celebrities	Shopping
1	1	1	1	1	1	1	1	2	3
2	4	1	2	1	4	1	1	5	2
3	4	5	3	1	4	4	1	2	4
4	5	4	2	2	5	4	3	4	4
5	4	5	4	1	5	5	3	1	3
6	1	2	1	1	2	1	1	3	5

They aren't so interested in Western movies, writing, or gardening; but they like comedies, romantic movies, and foreign languages.

%>%, cont'd

Recall how to select:

```
head(select(responses, Age, Gender, Education))
```

	Age	Gender	Education
1	20	female	college/bachelor degree
2	19	female	college/bachelor degree
3	20	female	secondary school
4	22	female	college/bachelor degree
5	20	female	secondary school
6	20	male	secondary school

Here it is again, using piping:

```
responses %>%
    select(Age, Gender, Education) %>%
    head
```

We are piping the `responses` data frame to the `select()` function to extract the three columns (age, gender, and education), and then we are piping the ensuing data frame to the `head()` function.

Pipe operator: %>%

- The pipe operator allows you to pipe the output from one function to the input of another function
- Instead of nesting functions as we've been doing (reading from the inside to the outside), the idea of of piping is to read the functions from left to right

Pipe operator: %>%

To see the head, or the first six rows, of `responses`:

```
head(responses)
```

```
responses %>% head
```

	Music	Slow.songs.or.fast.songs	Dance	Folk	Country	Classical.music	Musical	Pop	Rock	Metal.or.Hardrock	Punk	Hiphop..Rap
1	5	3	2	1	2	2	1	5	5	1	1	1
2	4	4	2	1	1	1	2	3	5	4	4	1
3	5	5	2	2	3	4	5	3	5	3	4	1
4	5	3	2	1	1	1	1	2	2	1	4	2
5	5	3	4	3	2	4	3	5	3	1	2	5
6	5	3	2	3	2	3	3	2	5	5	3	4

	Reggae..Ska	Swing..Jazz	Rock.n.roll	Alternative	Latino	Techno..Trance	Opera	Movies	Horror	Thriller	Comedy	Romantic	Sci.fi
1	1	1	3	1	1	1	1	5	4	2	5	4	4
2	3	1	4	4	2	1	1	5	2	2	4	3	4
3	4	3	5	5	5	1	3	5	3	4	4	2	4
4	2	1	2	5	1	2	1	5	4	4	3	3	4
5	3	2	1	2	4	2	2	5	4	4	5	2	3
6	3	4	4	5	3	1	3	5	5	5	5	2	3

	War	Fantasy.Fairy.tales	Animated	Documentary	Western	Action	History	Psychology	Politics	Mathematics	Physics	Internet	PC
1	1	5	5	3	1	2	1	5	1	3	3	5	3
2	1	3	5	4	1	4	1	3	4	5	2	4	4
3	2	5	5	2	2	1	1	2	1	5	2	4	2
4	3	1	2	5	1	2	4	4	5	4	1	3	1
5	3	4	4	3	1	4	3	2	3	2	2	2	2
6	3	4	3	3	2	4	5	3	4	2	3	4	4

Arrange rows with `arrange()`

To arrange rows by the values in a particular column or columns, use the `arrange()` function, with the name(s) of the column(s) you want to arrange the rows by as argument(s):

```
responses %>% arrange(Age) %>% head
```

	Interests.or.hobbies	Parents..advice	Questionnaires.or.polls	Internet.usage	Finances	Shopping.centres	Branded.clothing
1	5	3	5	few hours a day	5	5	3
2	2	2	5	most of the day	5	4	3
3	5	3	2	few hours a day	2	5	4
4	3	2	3	few hours a day	3	3	5
5	1	3	5	few hours a day	1	5	3
6	3	3	3	most of the day	5	5	3

	Entertainment.spending	Spending.on.looks	Spending.on.gadgets	Spending.on.healthy.eating	Age	Height	Weight
1	5	3	2	5	15	173	49
2	1	2	2	2	15	170	51
3	3	5	5	5	15	160	48
4	2	4	2	2	15	181	63
5	4	5	4	2	15	176	53
6	3	3	2	3	15	170	49

Taking the head of the data frame arranged by age yields the responses of six 15-year-olds, who aren't super likely to spend their money on healthy eating, and prefer branded clothing to non-branded.

`arrange()`, cont'd

By default, `arrange()` sorts in ascending order, but you can also arrange in descending order:

```
responses %>% arrange(desc(Age)) %>% head
```

	Interests.or.hobbies	Parents..advice	Questionnaires.or.polls	Internet.usage	Finances	Shopping.centres
1	4	2	3	few hours a day	1	3
2	2	3	3	less than an hour a day	4	4
3	4	2	3	most of the day	3	2
4	3	4	3	less than an hour a day	3	2
5	2	4	4	most of the day	5	5
6	3	1	1	less than an hour a day	1	1

	Branded.clothing	Entertainment.spending	Spending.on.looks	Spending.on.gadgets	Spending.on.healthy.eating	Age	He
1	1	3	1	1	5	30	
2	2	2	3	1	3	30	
3	4	4	3	4	4	30	
4	1	3	2	2	4	30	
5	5	5	5	5	5	30	
6	1	5	3	2	2	30	

This group of 30-year-old respondents are much more likely to spend money on healthy eating, and less likely to care about branded clothing.

Pipe operator: %>%

- The real power of piping is that it enables us to seamlessly combine dplyr data verbs

`arrange()`, cont'd

Let's select a few columns and arrange the rows by number of friends:

```
responses %>%
      select(Age, Gender, Number.of.friends, Happiness.in.life) %>%
      arrange(Number.of.friends) %>%
      head
```

	Age	Gender	Number.of.friends	Happiness.in.life
1	22	female	1	2
2	21	male	1	4
3	21	female	1	3
4	18	female	1	3
5	17	female	1	2
6	21	female	1	2

Again, in descending order:

```
responses %>%
      select(Age, Gender, Number.of.friends, Happiness.in.life) %>%
      arrange(desc(Number.of.friends)) %>%
      head
```

	Age	Gender	Number.of.friends	Happiness.in.life
1	22	female	5	5
2	20	male	5	5
3	20	female	5	4
4	25	female	5	4
5	21	female	5	4
6	21	female	5	2

(Note the correlation between number of friends and happiness in life.)

`arrange()`, cont'd

Same as before, except now let's also filter for respondents over the age of 25:

```
responses %>%
    select(Age, Gender, Number.of.friends, Happiness.in.life) %>%
    arrange(Number.of.friends) %>%
    filter(Age > 25) %>%
    head
```

	Age	Gender	Number.of.friends	Happiness.in.life
1	28	female	1	3
2	26	male	1	4
3	27	male	2	4
4	29	male	2	NA
5	28	female	2	4
6	28	female	2	3

Hypothesis: people's happiness levels depend less on their number of friends as they age?

Create summaries using `summarise()`

The `summarise()` function creates a new data frame that summaries a column in the given data frame. For example, to compute the mean `happiness`, apply the `mean()` function to the column `Happiness.in.life` and call the value `avg_happiness`. (Because some survey respondents left this field blank, we remove the `N/A` responses using `filter()` and `!is.na`.)

```
responses %>% summarise(avg_happiness = mean(Happiness.in.life))
```

```
  avg_happiness
1            NA
```

```
responses %>% summarise(avg_happiness = mean(Happiness.in.life, na.rm = TRUE))
```

```
  avg_happiness
1      3.705765
```

Create summaries using `summarise()`

If we want to compare the average happiness of female vs. male respondents, we can also filter by gender:

```
responses %>%
    filter(Gender == "female") %>%
    summarise(avg_happiness = mean(Happiness.in.life, na.rm = TRUE))
```

```
  avg_happiness
1      3.681895
```

```
responses %>%
    filter(Gender == "male") %>%
    summarise(avg_happiness = mean(Happiness.in.life, na.rm = TRUE))
```

```
  avg_happiness
1      3.748166
```

Gender doesn't appear to be a major determinant of happiness.

`summarise()`, cont'd

There are many other summary statistics, such as:

- `sd()`
- `min()`
- `max()`
- `median()`
- `sum()`
- `length()` (the length of a column)
- `first()` (the first value in a column)
- `last()` (the last value in a column)
- `unique()` (the number of distinct values in a column)

Very common paradigm

- Split: break down your data set by groups, or levels
- Apply: apply some function (e.g., summarize) to each group
- Combine: put your data back together again

Cereals:

- Split up the data by shelf
- Calculate the average sugar content per shelf
- Put your data back together again

Group operations using `group_by()`

The `group_by()` operation makes it easy to implement the split-apply-combine paradigm.

We can split a data frame by some variable, apply a function to the ensuing split data frames, and then combine the output back into a single data frame.

To group by gender, and summarize the average happiness of each gender group:

```
responses %>%
    group_by(Gender) %>%
    summarise(avg_happiness = mean(Happiness.in.life, na.rm = TRUE))
```

	Gender	avg_happiness
	<fct>	<dbl>
1	""	3.17
2	female	3.68
3	male	3.75

`group_by()`, cont.

Now, combining functions, we can split the data frame by age, gender, number of siblings, or any other factor, and then ask for the average scores to the following questions, from left to right:

1. I am 100% happy with my life.
2. I live a very healthy lifestyle.
3. I look at things from all different angles before I g
4. I always try to vote in elections.
5. I am not afraid to give my opinion if I feel strongly
6. I feel lonely in life.
7. I spend a lot of money on my appearance.

	Age	avg_happiness	avg_health	avg_thinking	avg_voting	avg_assertiveness	avg_loneliness	avg_spending_looks	total
	<int>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<int>
1	15	3.833333	2.166667	3.000000	1.500000	3.166667	3.666667	3.166667	6
2	16	3.333333	2.933333	3.466667	2.133333	3.133333	3.266667	3.866667	15
3	17	3.547619	3.023810	3.095238	1.690476	3.523810	3.047619	3.309524	42
4	18	3.701149	3.068966	3.459770	2.908046	3.494253	3.137931	3.183908	87
5	19	3.771812	2.932886	3.248322	3.791946	3.328859	2.744966	3.033557	149
6	20	3.708955	2.880597	3.425373	3.731343	3.470149	2.783582	3.111940	134
7	21	3.611765	3.105882	3.411765	3.729412	3.376471	2.964706	3.117647	85
8	22	3.622642	3.037736	3.641509	3.792453	3.641509	2.867925	2.981132	53
9	23	3.735294	3.058824	3.294118	3.676471	3.470588	2.823529	2.852941	34
10	24	3.842105	2.947368	4.000000	3.789474	3.789474	2.947368	2.789474	19
11	25	3.650000	3.200000	3.250000	4.050000	3.650000	3.000000	3.150000	20
12	26	3.700000	3.100000	3.800000	3.500000	3.600000	2.600000	3.500000	10
13	27	3.875000	2.875000	3.875000	4.125000	3.500000	2.625000	3.125000	8
14	28	3.916667	3.250000	3.916667	3.416667	3.833333	2.750000	2.666667	12
15	29	3.833333	3.000000	3.500000	4.500000	4.000000	2.333333	2.833333	6
16	30	3.833333	3.166667	3.666667	4.333333	4.500000	3.166667	2.833333	6

This gives us a set of summary statistics grouped by age.

The rightmost column reports the total number of respondents by age.

Create new columns using `mutate()`

The `mutate()` function adds new columns to a data frame. Let's create a new column called `decision_regrets`, which will be the ratio of responses to the statement
"I take my time to make decisions" to "I often think about and regret the decisions I make":

```
responses %>%
    mutate(decision_regrets = Decision.making / Self.criticism) %>%
    select(Age, decision_regrets) %>%
    head
```

	Age	decision_regrets
1	20	3.00
2	19	0.50
3	20	0.75
4	22	1.00
5	20	0.60
6	20	0.50

Note the selection for `Age` and the new `decision_regrets` columns.
Otherwise, the new data frame would contain all 150 columns.

Create new columns using `mutate()`

The `mutate()` function adds new columns to a data frame.

In this example, we use it to mutate `Age` from an integer to a factor:

```
responses %>%
    mutate(decision_regrets = Decision.making / Self.criticism) %>%
    mutate(age_factor = cut(Age, breaks = c(15, 19, 23, 27))) %>%
    select(age_factor, decision_regrets) %>%
    head
```

	age_factor	decision_regrets
1	(19,23]	3.00
2	(15,19]	0.50
3	(19,23]	0.75
4	(19,23]	1.00
5	(19,23]	0.60
6	(19,23]	0.50

Note the selection for `age_factor` and the new `decision_regrets` columns.

Otherwise, the new data frame would contain all 150 columns.

Benefits of dplyr

- R already contains many built-in functions, like `split()`, `subset()`, `summary()`, and so on, but the dplyr equivalents are easier to work with and targeted specifically at data frames
- Additionally, dplyr built-in functions have more intuitive syntax
  - From a data frame listing all flights that departed from New York City in 2013, you can select all flights that departed on January 1st in dplyr with:

    ```
    filter(flights, month == 1, day == 1)
    flights %>% filter(month == 1, day == 1)
    ```

  - In base R, this query would be more verbose and more difficult to read:

    ```
    flights[flights$month == 1 & flights$day == 1, ]
    ```