Data manipulation: Filtering and arranging

Research Methods for Human Inquiry

Andrew Perfors

Today's story…

Hi, I'm Flopsy.

Yes I know it's a ridiculous name and I hate it but I didn't choose it and blame my parents.

Anyway. I'm worried about Bunny and Gladly.

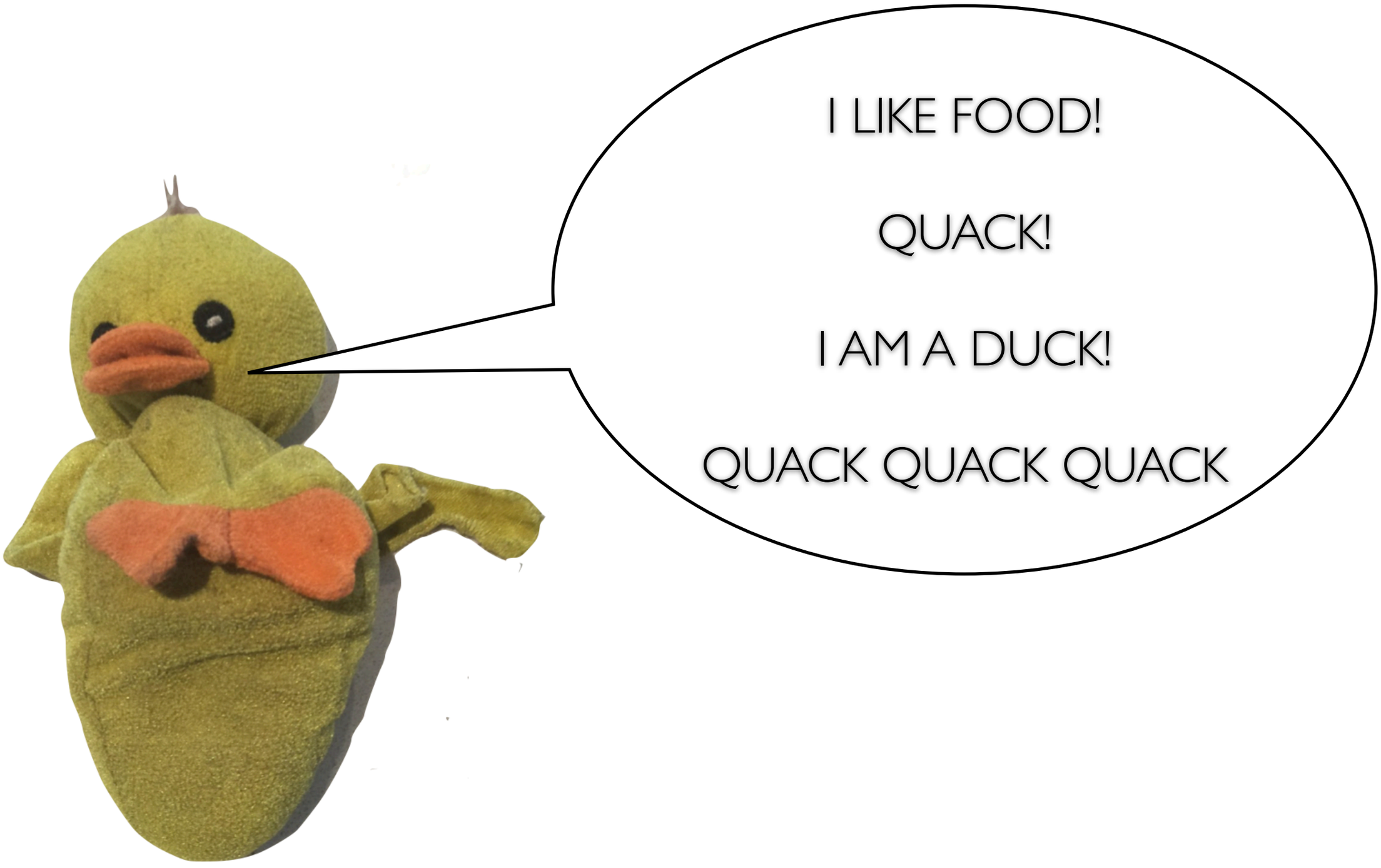
Today's story…

They're fighting a lot and partly it's because Gladly stole Bunny's idea but I think mainly it's because Gladly is hungry all of the time and he turns into a meanie then.

But we don't have much food to give him. That's a problem, isn't it? Is he being all selfish or are we really running out?

Cause I'm kind of hungry too.

Today's story…



Today's story…

I'm hungry too, but I find it kind of hard to deal with these datasets. There is so much stuff in them!

Is there any way to simplify things?

What if we don’t want all of our data?

What if we don't want all of our data?

```{r speciesagesummary, echo=FALSE}
gdata %>%
  group_by(species) %>%
  summarise(mnAge = mean(age),
            sdAge = sd(age),
            nAge = n()) %>%
  ungroup()
```

species <chr>	mnAge <dbl>	sdAge <dbl>	nAge <int>
bear	6.428571	2.070197	7
bunny	7.000000	2.309401	16
cat	10.000000	2.828427	2
dog	7.500000	2.081666	4
duck	6.000000	*NA*	1
fox	13.000000	*NA*	1
frog	7.000000	*NA*	1
hedgehog	8.000000	*NA*	1
monkey	6.000000	*NA*	1
turtle	6.000000	*NA*	1

1–10 of 10 rows

This is really irritating - perhaps we don't want to worry about the species that are really rare

How can we **filter** our data to keep just the rows we want, and arrange them nicely?

Could do this with some of our data manipulation skills from last week, but as we've seen that's fiddly and often difficult

`filter()` and `arrange()`

General idea: `filter()` lets you select a subset of the rows of the tibble, and `arrange()` tells you how to sort them

The name of the data tibble we want to use →

We only want the bears

```
gdata %>%
  filter(species=="bear")
```

```
# A tibble: 7 × 9
  name      species ageCat food  carrot  cake   mud   age gender
  <chr>     <chr>   <chr>  <chr>  <dbl> <dbl> <dbl> <dbl> <chr>
1 pink bear bear    medium chic…      8     6     1    10 male
2 gladly    bear    medium honey      7    10     2     8 male
3 snowy     bear    medium fish       8     7     2     7 male
4 cuddly    bear    medium meat       7     9     1     6 male
5 shaggy    bear    young  honey      5    10     1     5 female
6 bamboo    bear    young  bamb…      7     9     2     5 female
7 panda     bear    young  meat       6     9     1     4 male
```

`filter()` and `arrange()`

General idea: `filter()` lets you select a subset of the rows of the tibble, and `arrange()` tells you how to sort them

The name of the data tibble we want to use →

We only want the bears

```
gdata %>%
  filter(species=="bear") %>%
  arrange(gender)
```

Within the bears, arranges the rows by gender

```
# A tibble: 7 × 9
  name      species ageCat food  carrot  cake   mud   age gender
  <chr>     <chr>   <chr>  <chr>  <dbl> <dbl> <dbl> <dbl> <chr>
1 shaggy    bear    young  honey      5    10     1     5 female
2 bamboo    bear    young  bamb…      7     9     2     5 female
3 pink bear bear    medium chic…      8     6     1    10 male
4 gladly    bear    medium honey      7    10     2     8 male
5 snowy     bear    medium fish       8     7     2     7 male
6 cuddly    bear    medium meat       7     9     1     6 male
7 panda     bear    young  meat       6     9     1     4 male
```

`filter()` and `arrange()`

Suppose we want bears *and* bunnies! Use the logical operators…

```{r nextfilter}
gdata %>%
  filter(species=="bear" | species=="bunny") %>%
  arrange(gender)
```

The logical operators in `filter()` can get confusing. This means "keep all rows where species is "bear" or species is "bunny"

name <chr>	species <chr>	ageCat <chr>	food <chr>	carrot <dbl>	cake <dbl>	m… <dbl>	age <dbl>	gender <chr>
shaggy	bear	young	honey	5	10	1	5	female
bamboo	bear	young	bamboo	7	9	2	5	female
holly	bunny	old	carrot	10	8	2	11	female
bunny	bunny	old	carrot	10	10	1	10	female
lfb	bunny	medium	lettuce	10	9	2	9	female
cuddly paws	bunny	medium	*NA*	*NA*	*NA*	*NA*	9	female
pink bunny	bunny	medium	lettuce	9	7	1	7	female
purple bunny	bunny	medium	carrot	10	8	1	7	female
blue bunny	bunny	medium	pizza	8	7	2	7	female
grey	bunny	medium	cake	9	10	1	6	female

1–10 of 23 rows Previous 1 2 3 Next

`filter()` and `arrange()`

Can arrange by multiple things (first comes first…)

```
```{r nextfilter}
gdata %>%
  filter(species=="bear" | species=="bunny") %>%
  arrange(species,gender)
```
```

name <chr>	species <chr>	ageCat <chr>	food <chr>	carrot <dbl>	cake <dbl>	m… <dbl>	age <dbl>	gender <chr>
shaggy	bear	young	honey	5	10	1	5	female
bamboo	bear	young	bamboo	7	9	2	5	female
pink bear	bear	medium	chicken	8	6	1	10	male
gladly	bear	medium	honey	7	10	2	8	male
snowy	bear	medium	fish	8	7	2	7	male
cuddly	bear	medium	meat	7	9	1	6	male
panda	bear	young	meat	6	9	1	4	male
holly	bunny	old	carrot	10	8	2	11	female
bunny	bunny	old	carrot	10	10	1	10	female
lfb	bunny	medium	lettuce	10	9	2	9	female

1–10 of 23 rows Previous 1 2 3 Next

`filter()` and `arrange()`

Can filter by multiple columns at once, and use other logical operators. For instance, this selects only the female bears:

```{r filterpractice}
gdata %>%
  filter(species=="bear" & gender=="female")
```

The logical operators in `filter()` can get confusing. This means "keep all rows where species is "bear" AND the gender is female"

A tibble: 2 × 9

name <chr>	species <chr>	ageCat <chr>	food <chr>	carrot <dbl>	cake <dbl>	mud <dbl>	age <dbl>	gender <chr>
shaggy	bear	young	honey	5	10	1	5	female
bamboo	bear	young	bamboo	7	9	2	5	female

2 rows

`filter()` and `arrange()`

As always, you can assign the tibble that is created to a new variable

```
```{r filterpractice}
ladybears <- gdata %>%
  filter(species=="bear" & gender=="female")
print(ladybears)
```
```

A tibble: 2 × 9

name <chr>	species <chr>	ageCat <chr>	food <chr>	carrot <dbl>	cake <dbl>	mud <dbl>	age <dbl>	gender <chr>
shaggy	bear	young	honey	5	10	1	5	female
bamboo	bear	young	bamboo	7	9	2	5	female

2 rows

`filter()` and `arrange()`

As always, you can assign the tibble that is created to a new variable… and then do other things to that tibble!

```
```{r simplefilter, echo=FALSE}
# create a tibble called fOnly that contains only females
fOnly <- gdata %>%
  filter(gender=="female")

# calculate the mean age of all species in that tibble (of just females)
fOnly %>%
  group_by(species) %>%
  summarise(mnAge = mean(age),
            nAge = n()) %>%
  ungroup()
```
```

species <chr>	mnAge <dbl>	nAge <int>
bear	5.0	2
bunny	7.4	10
cat	10.0	2
fox	13.0	1
frog	7.0	1
turtle	6.0	1

`filter()` and `arrange()`

As always, you can assign the tibble that is created to a new variable… and then do other things to that tibble!

species <chr>	mnAge <dbl>	sdAge <dbl>	nAge <int>
bear	6.428571	2.070197	7
bunny	7.000000	2.309401	16
cat	10.000000	2.828427	2
dog	7.500000	2.081666	4
duck	6.000000	*NA*	1
fox	13.000000	*NA*	1
frog	7.000000	*NA*	1
hedgehog	8.000000	*NA*	1
monkey	6.000000	*NA*	1
turtle	6.000000	*NA*	1

1–10 of 10 rows

This is different from the summary statistics we calculated at the beginning because those included the male bears as well!

species <chr>	mnAge <dbl>	nAge <int>
bear	5.0	2
bunny	7.4	10
cat	10.0	2
fox	13.0	1
frog	7.0	1
turtle	6.0	1

Walk through everything with a simple example now!

`filter()` and `arrange()` example

Example dataset `ds`

Ratings are on a scale of 1-10 (higher = more of that emotion)

person	gender	time	happy	anger	joy
bunny	female	morning	7	3	6
bunny	female	evening	4	3	2
gladly	male	morning	5	NA	1
gladly	male	evening	8	3	2
shadow	female	morning	4	3	3
shadow	female	evening	9	1	5
quackers	male	morning	NA	5	4
quackers	male	evening	6	5	4

`filter()` removes rows

```
ds %>%
   filter(gender==“female”)
```

person	gender	time	happy	anger	joy
bunny	female	morning	7	3	6
bunny	female	evening	4	3	2
shadow	female	morning	4	3	3
shadow	female	evening	9	1	5

Keeps only rows where the gender is female

`filter()` and `arrange()` example

Example dataset `ds`

Ratings are on a scale of 1-10 (higher = more of that emotion)

person	gender	time	happy	anger	joy
bunny	female	morning	7	3	6
bunny	female	evening	4	3	2
gladly	male	morning	5	NA	1
gladly	male	evening	8	3	2
shadow	female	morning	4	3	3
shadow	female	evening	9	1	5
quackers	male	morning	NA	5	4
quackers	male	evening	6	5	4

`filter()` removes rows

```
ds %>%
   filter(gender==“female”) %>%
   arrange(happy)
```

person	gender	time	happy	anger	joy
bunny	female	evening	4	3	2
shadow	female	morning	4	3	3
bunny	female	morning	7	3	6
shadow	female	evening	9	1	5

Sorts the rows in increasing order of happiness

`filter()` and `arrange()` example

Example dataset `ds`

Ratings are on a scale of 1-10 (higher = more of that emotion)

person	gender	time	happy	anger	joy
bunny	female	morning	7	3	6
bunny	female	evening	4	3	2
gladly	male	morning	5	NA	1
gladly	male	evening	8	3	2
shadow	female	morning	4	3	3
shadow	female	evening	9	1	5
quackers	male	morning	NA	5	4
quackers	male	evening	6	5	4

`filter()` removes rows

```
ds %>%
   filter(gender==“female”) %>%
   arrange(desc(happy))
```

person	gender	time	happy	anger	joy
shadow	female	evening	9	1	5
bunny	female	morning	7	3	6
bunny	female	evening	4	3	2
shadow	female	morning	4	3	3

Sorts the rows in *decreasing* order of happiness

`filter()` and `arrange()` example

Example dataset `ds`

Ratings are on a scale of 1-10 (higher = more of that emotion)

person	gender	time	happy	anger	joy
bunny	female	morning	7	3	6
bunny	female	evening	4	3	2
gladly	male	morning	5	NA	1
gladly	male	evening	8	3	2
shadow	female	morning	4	3	3
shadow	female	evening	9	1	5
quackers	male	morning	NA	5	4
quackers	male	evening	6	5	4

`filter()` removes rows

```
filterEx <- ds %>%
   filter(gender=="female") %>%
   arrange(desc(happy))
```

`filterEx`

person	gender	time	happy	anger	joy
shadow	female	evening	9	1	5
bunny	female	morning	7	3	6
bunny	female	evening	4	3	2
shadow	female	morning	4	3	3

Sorts the rows in *decreasing* order of happiness

Assign the output to a tibble

`filter()` and `arrange()` example

Example dataset `ds`

Ratings are on a scale of 1-10 (higher = more of that emotion)

person	gender	time	happy	anger	joy
bunny	female	morning	7	3	6
bunny	female	evening	4	3	2
gladly	male	morning	5	NA	1
gladly	male	evening	8	3	2
shadow	female	morning	4	3	3
shadow	female	evening	9	1	5
quackers	male	morning	NA	5	4
quackers	male	evening	6	5	4

`filter()` removes rows

```
ds %>%
   filter(gender==“male” & joy>3)
```

person	gender	time	happy	anger	joy
quackers	male	morning	NA	5	4
quackers	male	evening	6	5	4

Keeps only rows where the gender is male AND the joy is greater than 3

`filter()` and `arrange()` example

Example dataset `ds`

Ratings are on a scale of 1-10 (higher = more of that emotion)

person	gender	time	happy	anger	joy
bunny	female	morning	7	3	6
bunny	female	evening	4	3	2
gladly	male	morning	5	NA	1
gladly	male	evening	8	3	2
shadow	female	morning	4	3	3
shadow	female	evening	9	1	5
quackers	male	morning	NA	5	4
quackers	male	evening	6	5	4

`filter()` removes rows

```
ds %>%
   filter(person==“bunny” | person==“gladly”)
```

`filter()` and `arrange()` example

Example dataset `ds`

Ratings are on a scale of 1-10 (higher = more of that emotion)

person	gender	time	happy	anger	joy
bunny	female	morning	7	3	6
bunny	female	evening	4	3	2
gladly	male	morning	5	NA	1
gladly	male	evening	8	3	2
shadow	female	morning	4	3	3
shadow	female	evening	9	1	5
quackers	male	morning	NA	5	4
quackers	male	evening	6	5	4

`filter()` removes rows

```
ds %>%
   filter(person=="bunny" | person=="gladly")
```

person	gender	time	happy	anger	joy
bunny	female	morning	7	3	6
bunny	female	evening	4	3	2
gladly	male	morning	5	NA	1
gladly	male	evening	8	3	2

Keeps only rows where the person is "bunny" OR the person is "gladly"

`filter()` and `arrange()` example

Example dataset `ds`

Ratings are on a scale of 1-10 (higher = more of that emotion)

person	gender	time	happy	anger	joy
bunny	female	morning	7	3	6
bunny	female	evening	4	3	2
gladly	male	morning	5	NA	1
gladly	male	evening	8	3	2
shadow	female	morning	4	3	3
shadow	female	evening	9	1	5
quackers	male	morning	NA	5	4
quackers	male	evening	6	5	4

`filter()` removes rows

```
ds %>%
   filter(is.na(anger))
```

person	gender	time	happy	anger	joy
gladly	male	morning	5	NA	1

Keeps only rows where there is an NA for anger

`filter()` and `arrange()` example

Example dataset `ds`

Ratings are on a scale of 1-10 (higher = more of that emotion)

person	gender	time	happy	anger	joy
bunny	female	morning	7	3	6
bunny	female	evening	4	3	2
gladly	male	morning	5	NA	1
gladly	male	evening	8	3	2
shadow	female	morning	4	3	3
shadow	female	evening	9	1	5
quackers	male	morning	NA	5	4
quackers	male	evening	6	5	4

`filter()` removes rows

```
ds %>%
   filter(!is.na(anger))
```

person	gender	time	happy	anger	joy
bunny	female	morning	7	3	6
bunny	female	evening	4	3	2
gladly	male	evening	8	3	2
shadow	female	morning	4	3	3
shadow	female	evening	9	1	5
quackers	male	morning	NA	5	4
quackers	male	evening	6	5	4

Keeps only rows where there is *not* an NA for anger

`filter()` and `arrange()` example

Example dataset `ds`

Ratings are on a scale of 1-10 (higher = more of that emotion)

person	gender	time	happy	anger	joy
bunny	female	morning	7	3	6
bunny	female	evening	4	3	2
gladly	male	morning	5	NA	1
gladly	male	evening	8	3	2
shadow	female	morning	4	3	3
shadow	female	evening	9	1	5
quackers	male	morning	NA	5	4
quackers	male	evening	6	5	4

`filter()` removes rows

```
ds %>%
   na.omit()
```

person	gender	time	happy	anger	joy
bunny	female	morning	7	3	6
bunny	female	evening	4	3	2
gladly	male	evening	8	3	2
shadow	female	morning	4	3	3
shadow	female	evening	9	1	5
quackers	male	evening	6	5	4

Keeps only rows where there is no NA anywhere

Exercises

Do the questions found in
the `w3day2-exercises.Rmd` file!