

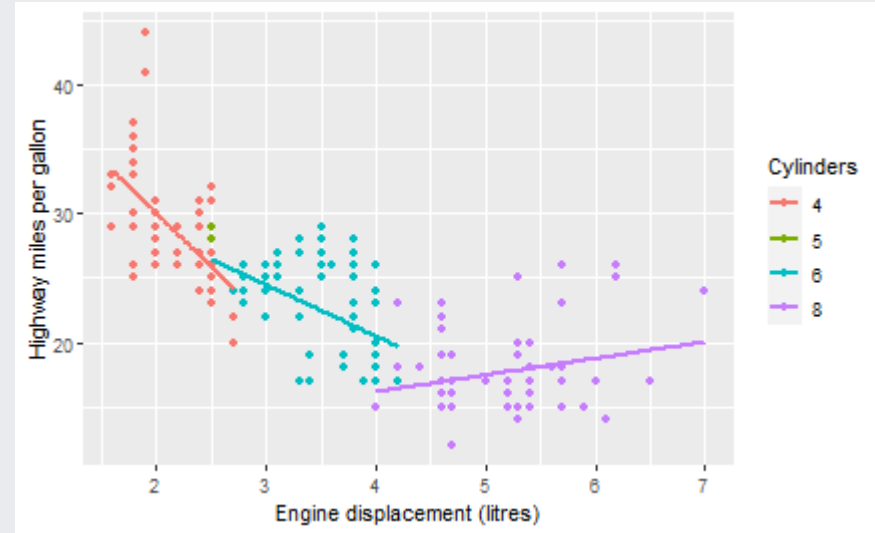
Transforming and summarising data

29/10/2024

Plotting using ggplot2

```
ggplot(data = mpg,
       mapping = aes(x = displ,
                     y = hwy,
                     colour = factor(cyl)))
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  labs(x = "Engine displacement (litres)",
       y = "Highway miles per gallon",
       colour = "Cylinders")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



Importing your data

Fear of Crime Dataset

Ellis & Renouf (2018) - the relationship between fear of crime and various personality measures.

Their data is openly available, stored as text in a *comma-separated-values* format (.csv).

Once again, we can use the import button or some code (with `read_csv()`) to load this data in and automatically format it into a *tibble*.

```
library(readr)  
FearofCrime <- read_csv("data/FearofCrime.CSV")
```

Fear of Crime Dataset

Ellis & Renouf (2018) collected data online using Qualtrics.

The file contains one column for each question that the participants answered, for a total of 169(!) columns.

Each row is a single participant's answers, and their demographic information.

```
FearofCrime
```

```
## # A tibble: 301 × 169
##   ResponseID Respo...1 Name Exter...2 Status Start...3 EndDate Finis...4 Conse...5 sex
##   <chr>         <chr>   <chr> <lgl>      <dbl> <chr>      <chr>      <dbl>    <dbl> <dbl>
## 1 R_ai4tgG1... Defaul... Anon... NA           0 19/10/... 19/10/...      1        1      2
## 2 R_d50iATV... Defaul... Anon... NA           0 20/10/... 20/10/...      1        1      2
## 3 R_aaBVZUe... Defaul... Anon... NA           0 20/10/... 20/10/...      1        1      1
## 4 R_6nxInLK... Defaul... Anon... NA           0 20/10/... 20/10/...      1        1      1
## 5 R_6SCYbh0... Defaul... Anon... NA           0 20/10/... 20/10/...      1        1      2
## 6 R_5pCxWA6... Defaul... Anon... NA           0 20/10/... 20/10/...      1        1      1
## 7 R_d1nji6V... Defaul... Anon... NA           0 20/10/... 20/10/...      1        1      2
## 8 R_9v6ZgUh... Defaul... Anon... NA           0 20/10/... 20/10/...      1        1      1
## 9 R_5Bg7VjB... Defaul... Anon... NA           0 20/10/... 20/10/...      1        1      2
## 10 R_9Sv17lQ... Defaul... Anon... NA           0 20/10/... 20/10/...      1        1      2
## # ... with 291 more rows, 159 more variables: age <dbl>, hexaco1 <dbl>,
## #   hexaco2 <dbl>, hexaco3 <dbl>, hexaco4 <dbl>, hexaco5 <dbl>, hexaco6 <dbl>,
```

Prison population

Last week, we looked at some data regarding the UK's prison population.

The data is contained in an Excel spreadsheet, downloaded from data.gov.uk.

```
library(readxl)
prison_pop <- read_excel("data/prison-population-data-tool-31-december-2017.xlsx",
                        sheet = "PT Data")
```

We use the `read_excel()` function to read Excel files.

Note how the file name and location come first, and then I specify a specific *sheet*.

Excel spreadsheets often have multiple sheets with different information.

Spaces

Your Workspace

PSY9219M; Research Method

New Space

Learn

Guide

Primers

DataCamp Courses

Cheat Sheets

Feedback and Questions

Info

Terms and Conditions

System Status

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins

R 3.5.0

FearofCrime x

Filter

	happy10	happy11	happy12	happy13	happy14	happy15	happy16	happy17
2	4	4	4	5	4	4		
2	4	4	2	4	5	5		
4	4	2	4	4	2	5		
4	4	2	4	4	2	5		
2	4	4	2	4	4	5		
5	2	1	5	5	1	4		
1	2	2	5	5	4	5		

Showing 1 to 8 of 301 entries

Console Terminal x Jobs x

/cloud/project/

```
StartDate = col_character(),
EndDate = col_character(),
hexaco_First_Click = col_double(),
hexaco_Last_Click = col_double(),
hexaco_Page_Submit = col_double(),
happy_First_Click = col_double(),
happy_Last_Click = col_double(),
happy_Page_Submit = col_double(),
crime_First_Click = col_double(),
crime_Last_Click = col_double(),
crime_Page_Submit = col_double()
)
See spec(...) for full column specifications.
> View(FearofCrime)
> |
```

Environment History Connections

Import Dataset

- From Text (base)...
- From Text (readr)...
- From Excel...
- From SPSS...
- From SAS...
- From Stata...

Files Plots Packages Help Viewer

New Folder Upload Delete Rename More

Cloud > project > data

	Name	Size	Modified
	..		
	2018-08-lincolnshire-street.csv	1.2 MB	Oct 21, 2018, 11:00 F
	Geographical_data_tool_oct05_...	18.2 MB	Oct 21, 2018, 10:54 F
	FearofCrime.csv	134 KB	Oct 22, 2018, 10:54 A
	crime.csv	23.3 KB	Oct 22, 2018, 10:56 A
	prison-population-data-tool-31-...	826.9 KB	Oct 22, 2018, 10:59 A

Spaces

Your Workspace

PSY9219M; Research Method

New Space

Learn

Guide

Primers

DataCamp Courses

Cheat Sheets

Feedback and Questions

Info

Terms and Conditions

System Status

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins R 3.5.0

Import Excel Data

File/Url: Browse...

Data Preview:

Offender Management Statistics - Prison Population Data Tool
Quarterly Prison Population at 30 June 2015 - 31 December 2017
NA
User Guide

Previewing first 50 entries.

Import Options:

Name: <input type="text" value="prison_population_data_to"/>	Max Rows: <input type="text" value=""/>	<input checked="" type="checkbox"/> First Row as Names
Sheet: <input type="text" value="Default"/>	Skip: <input type="text" value="0"/>	<input checked="" type="checkbox"/> Open Data Viewer
Range: <input type="text" value="A1:D10"/>	NA: <input type="text" value=""/>	

Code Preview:

```
library(readxl)
prison_population_data_tool_31_december_2017 <- read_excel("data/prison-population-data-tool-31-december-2017.xlsx")
```

Import Cancel

Reading Excel files using readxl

Spaces

Your Workspace

PSY9219M; Research Method

New Space

Learn

Guide

Primers

DataCamp Courses

Cheat Sheets

Feedback and Questions

Info

Terms and Conditions

System Status

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function

Addins

R 3.5.0

Import Excel Data

File/Url:

/cloud/project/data/prison-population-data-tool-31-december-2017.xlsx

Browse...

Data Preview:

View (character)	Date (character)	Establishment (character)	Sex (character)	Age / Custody / Nationality / Offence Group (character)	Population (double)
a Establishment*Sex*Age Group	2015-06	Altcourse	Male	Adults (21+)	922
a Establishment*Sex*Age Group	2015-06	Altcourse	Male	Juveniles and Young Adults (15-20)	169
a Establishment*Sex*Age Group	2015-06	Ashfield	Male	Adults (21+)	389
a Establishment*Sex*Age Group	2015-06	Askham Grange	Female	Adults (21+)	NA
a Establishment*Sex*Age Group	2015-06	Askham Grange	Female	Juveniles and Young Adults (15-20)	NA
a Establishment*Sex*Age Group	2015-06	Aylesbury	Male	Adults (21+)	113
a Establishment*Sex*Age Group	2015-06	Aylesbury	Male	Juveniles and Young Adults (15-20)	268
a Establishment*Sex*Age Group	2015-06	Bedford	Male	Adults (21+)	459
a Establishment*Sex*Age Group	2015-06	Bedford	Male	Juveniles and Young Adults (15-20)	30
a Establishment*Sex*Age Group	2015-06	Belmarsh	Male	Adults (21+)	794
a Establishment*Sex*Age Group	2015-06	Belmarsh	Male	Juveniles and Young Adults (15-20)	74

Previewing first 50 entries.

Import Options:

Name: prison_pop

Max Rows:

☒ First Row as Names

Sheet: PT Data

Skip:

0

☒ Open Data Viewer

Range: A1:D10

NA:

Code Preview:

```
library(readxl)
prison_pop <-
read_excel("data
/prison-population
-data-tool-31-december
-2017.xlsx").
```

Reading Excel files using readxl

Import

Cancel

Prison population

Once the data is imported, we have a tibble.

We can immediately see there are 6 columns with 22409 rows.

```
prison_pop
```

```
## # A tibble: 22,409 × 6
##   View      Date      Establishment Sex    Age / Cu...1 Popul...2
##   <chr>    <chr>    <chr>          <chr>  <chr>      <dbl>
## 1 a Establishment*Sex*Age Group 2015-06 Altcourse    Male  Adults (2...    922
## 2 a Establishment*Sex*Age Group 2015-06 Altcourse    Male  Juveniles...   169
## 3 a Establishment*Sex*Age Group 2015-06 Ashfield     Male  Adults (2...   389
## 4 a Establishment*Sex*Age Group 2015-06 Askham Grange Female Adults (2...    NA
## 5 a Establishment*Sex*Age Group 2015-06 Askham Grange Female Juveniles...    NA
## 6 a Establishment*Sex*Age Group 2015-06 Aylesbury    Male  Adults (2...   113
## 7 a Establishment*Sex*Age Group 2015-06 Aylesbury    Male  Juveniles...   268
## 8 a Establishment*Sex*Age Group 2015-06 Bedford      Male  Adults (2...   459
## 9 a Establishment*Sex*Age Group 2015-06 Bedford      Male  Juveniles...    30
## 10 a Establishment*Sex*Age Group 2015-06 Belmarsh     Male  Adults (2...   794
## # ... with 22,399 more rows, and abbreviated variable names
## #   1'Age / Custody / Nationality / Offence Group', 2Population
```

We need to do more work to make this file useable...

dplyr and data transformation





Data transformation

With datasets like those we've loaded, there are often organisational issues.

For example, there could be many columns or rows we don't need, or the data would make more sense if it were sorted.

This is where `dplyr` comes in!

Function	Effect
<code>select()</code>	Include or exclude variables (columns)
<code>arrange()</code>	Change the order of observations (rows)
<code>filter()</code>	Include or exclude observations (rows)
<code>mutate()</code>	Create new variables (columns)
<code>group_by()</code>	Create groups of observations
<code>summarise()</code>	Aggregate or summarise groups of observations (rows)

Selecting columns



Selecting columns

Sometimes only some columns are of interest.

The Fear of Crime dataset has 169 columns. Only some of them are useful; here are the first ten.

```
names(FearofCrime)[1:10]
```

```
## [1] "ResponseID"  
## [2] "ResponseSet"  
## [3] "Name"  
## [4] "ExternalDataReference"  
## [5] "Status"  
## [6] "StartDate"  
## [7] "EndDate"  
## [8] "Finished"  
## [9] "Consent Form / This study includes a range of questionnaires collecting / demographic and i  
## [10] "sex"
```



Selecting columns

We pass the name of the data frame that we want to select from, and the names of each column we want to keep after that.

Suppose that, first of all, we were only interested in the age and sex of our participants.

```
select(FearofCrime, age, sex)
```

```
## # A tibble: 301 × 2
##   age    sex
##   <dbl> <dbl>
## 1     26     2
## 2     66     2
## 3     41     1
## 4     46     1
## 5     53     2
## 6     33     1
## 7     41     2
## 8     39     1
## 9     38     2
## 10    19     2
## # ... with 291 more rows
```



Selecting columns

The HEXACO-PI-R is a personality questionnaire that aims to measure six factors - Honesty-Humility, Emotionality, Extraversion, Agreeableness, Conscientiousness, and Openness to Experience.

The Fear of Crime dataset has the participants answers to the 60 questions of the HEXACO-PI-R in 60 columns.

```
select(FearofCrime, hexaco1,  
       hexaco2, hexaco3)
```

```
## # A tibble: 8 × 3  
##   hexaco1 hexaco2 hexaco3  
##   <dbl>   <dbl>   <dbl>  
## 1       4       5       2  
## 2       2       4       2  
## 3       1       5       2  
## 4       1       5       2  
## 5       2       4       4  
## 6       2       4       2  
## 7       1       5       4  
## 8       2       4       3
```




Selecting columns

Typing these out one by one would be ... *laborious*.

Fortunately, there are some shorthands.

The colon (:) operator can be used to say "everything between these columns (inclusive)".

```
select(FearofCrime, hexaco1:hexaco5)
```

```
## # A tibble: 301 × 5
##   hexaco1 hexaco2 hexaco3 hexaco4 hexaco5
##   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1         4         5         2         4         1
## 2         2         4         2         4         4
## 3         1         5         2         3         2
## 4         1         5         2         4         1
## 5         2         4         4         5         5
## 6         2         4         2         2         2
## 7         1         5         4         4         4
## 8         2         4         3         2         2
## 9         1         2         4         2         5
## 10        4         4         2         3         2
## # ... with 291 more rows
```



Selecting columns

Note that you can also tell `select()` to *remove* columns using the minus (-) sign.

```
select(FearofCrime, -ResponseSet, -Name, -Status, -ExternalDataReference)
```

```
## # A tibble: 301 × 165
##   Respons...1 Start...2 EndDate Finis...3 Conse...4 sex age hexaco1 hexaco2 hexaco3
##   <chr>      <chr>      <chr>      <dbl>      <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 R_ai4tgG... 19/10/... 19/10/...      1        1      2    26      4      5      2
## 2 R_d50iAT... 20/10/... 20/10/...      1        1      2    66      2      4      2
## 3 R_aaBVZU... 20/10/... 20/10/...      1        1      1    41      1      5      2
## 4 R_6nxInL... 20/10/... 20/10/...      1        1      1    46      1      5      2
## 5 R_6SCYbh... 20/10/... 20/10/...      1        1      2    53      2      4      4
## 6 R_5pCxWA... 20/10/... 20/10/...      1        1      1    33      2      4      2
## 7 R_d1nji6... 20/10/... 20/10/...      1        1      2    41      1      5      4
## 8 R_9v6ZgU... 20/10/... 20/10/...      1        1      1    39      2      4      3
## 9 R_5Bg7Vj... 20/10/... 20/10/...      1        1      2    38      1      2      4
## 10 R_9Sv17l... 20/10/... 20/10/...      1        1      2    19      4      4      2
## # ... with 291 more rows, 155 more variables: hexaco4 <dbl>, hexaco5 <dbl>,
## #   hexaco6 <dbl>, hexaco7 <dbl>, hexaco8 <dbl>, hexaco9 <dbl>, hexaco10 <dbl>,
## #   hexaco11 <dbl>, hexaco12 <dbl>, hexaco13 <dbl>, hexaco14 <dbl>,
## #   hexaco15 <dbl>, hexaco16 <dbl>, hexaco17 <dbl>, hexaco18 <dbl>,
## #   hexaco19 <dbl>, hexaco20 <dbl>, hexaco21 <dbl>, hexaco22 <dbl>,
## #   hexaco23 <dbl>, hexaco24 <dbl>, hexaco25 <dbl>, hexaco26 <dbl>,
```

Creating new columns



Creating new columns

Here is a version of the Fear of Crime data where participants' overall scores on the various personality measures have been calculated.

```
crime
```

```
## # A tibble: 301 × 15
##   Participi...1 sex    age victi...2    H    E    X    A    C    O    SA    TA
##   <chr>      <chr> <dbl> <chr>    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 R_01TjXg... male    55 yes      3.7    3    3.4    3.9    3.2    3.6    1.15    1.55
## 2 R_0dN5Ye... fema...  20 no       2.5    3.1    2.5    2.4    2.2    3.1    2.05    2.95
## 3 R_0DPiPY... male    57 yes      2.6    3.1    3.3    3.1    4.3    2.8    2      2.6
## 4 R_0f7bSs... male    19 no       3.5    1.8    3.3    3.4    2.1    2.7    1.55    2.1
## 5 R_0rov2R... fema...  20 no       3.3    3.4    3.9    3.2    2.8    3.9    1.3     1.8
## 6 R_0wioqG... fema...  20 no       2.6    2.6    3      2.6    2.9    3.4    2.55    1.5
## 7 R_0wR08l... male    34 yes      3.2    2.5    3.2    2.8    4      3.2    1.85    1.75
## 8 R_116nEd... fema...  19 no       2.9    4      3.9    4.2    3.7    1.9    1.1     2
## 9 R_11ZmBd... fema...  19 yes      3.4    3.4    3.3    3.4    3.2    3.2    2.2     2.9
## 10 R_12i26Q... male    20 no       2.4    2.1    1.8    2.2    3.4    2.9    2.15    2.4
## # ... with 291 more rows, 3 more variables: OHQ <dbl>, FoC <dbl>, Foc2 <dbl>, and
## #   abbreviated variable names 1Participant, 2victim_crime
```



Creating new columns

```
crime_sub <- select(crime,  
                    age, SA, TA, sex)  
mutate(crime_sub, age_group = ifelse(age > 40,  
                                     "Over 40",  
                                     "40 or under"))
```

```
## # A tibble: 301 × 5  
##       age      SA      TA sex      age_group  
##   <dbl> <dbl> <dbl> <chr>   <chr>  
## 1     55  1.15  1.55 male    Over 40  
## 2     20  2.05  2.95 female  40 or under  
## 3     57  2      2.6  male    Over 40  
## 4     19  1.55  2.1  male    40 or under  
## 5     20  1.3   1.8  female  40 or under  
## 6     20  2.55  1.5  female  40 or under  
## 7     34  1.85  1.75 male    40 or under  
## 8     19  1.1   2     female  40 or under  
## 9     19  2.2   2.9  female  40 or under  
## 10    20  2.15  2.4  male    40 or under  
## # ... with 291 more rows
```



Arranging rows

Having calculated each person's *state anxiety* score, perhaps we'd now like to check who has the lowest and highest scores (note: this can be a good way to check for extreme values!).

```
arrange(crime_sub, SA)
```

```
## # A tibble: 301 × 4
##   age      SA      TA sex
##   <dbl> <dbl> <dbl> <chr>
## 1    20     1    1.05 male
## 2    53     1    1.55 female
## 3    49     1    1.65 male
## 4    19    1.05    1.5  female
## 5    19    1.1     2    female
## 6    19    1.1    1.4  male
## 7    29    1.1    1.5  female
## 8    19    1.1    1.3  female
## 9    20    1.1    1.8  female
## 10   21    1.1    2.1  male
## # ... with 291 more rows
```

```
arrange(crime_sub, desc(SA))
```

```
## # A tibble: 301 × 4
##   age      SA      TA sex
##   <dbl> <dbl> <dbl> <chr>
## 1    19    3.85    3.85 female
## 2    20    3.6     3.6  female
## 3    20    3.6     3.55 female
## 4    18    3.4     4    female
## 5    19    3.4     3.35 female
## 6    20    3.35    2.8  female
## 7    20    3.3     3.5  male
## 8    19    3.2     2.95 male
## 9    19    3.1     3.1  female
## 10   20    3.1     3.15 female
## # ... with 291 more rows
```

Grouping and summarizing



Summarising rows

A common task when analyzing data is to create summaries of statistical characteristics.

Here I calculate the *mean*, *standard deviation*, and *variance* of the State Anxiety variable.

Other possible summary functions (other than `mean()`, `sd()`, or `var()`) include `max()`, `min()`, `IQR()`, or `median()`.

```
summarise(crime,  
          mean = mean(SA),  
          standard_dev = sd(SA),  
          variance = var(SA))
```

```
## # A tibble: 1 × 3  
##   mean standard_dev variance  
##   <dbl>         <dbl>     <dbl>  
## 1  1.92         0.554     0.307
```




Grouping observations

`group_by()` is used to organise data frames into groups according to categorical variables.

```
grouped_crime <- group_by(crime, sex, victim_crime)
grouped_crime
```

```
## # A tibble: 301 × 15
## # Groups:   sex, victim_crime [4]
##   Participi...1 sex    age victi...2    H    E    X    A    C    O    SA    TA
##   <chr>      <chr> <dbl> <chr>    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 R_01TjXg... male    55 yes     3.7    3    3.4    3.9    3.2    3.6    1.15    1.55
## 2 R_0dN5Ye... fema...  20 no      2.5    3.1    2.5    2.4    2.2    3.1    2.05    2.95
## 3 R_0DPiPY... male    57 yes     2.6    3.1    3.3    3.1    4.3    2.8    2      2.6
## 4 R_0f7bSs... male    19 no      3.5    1.8    3.3    3.4    2.1    2.7    1.55    2.1
## 5 R_0rov2R... fema...  20 no      3.3    3.4    3.9    3.2    2.8    3.9    1.3    1.8
## 6 R_0wioqG... fema...  20 no      2.6    2.6    3      2.6    2.9    3.4    2.55    1.5
## 7 R_0wR08l... male    34 yes     3.2    2.5    3.2    2.8    4      3.2    1.85    1.75
## 8 R_116nEd... fema...  19 no      2.9    4      3.9    4.2    3.7    1.9    1.1    2
## 9 R_11ZmBd... fema...  19 yes     3.4    3.4    3.3    3.4    3.2    3.2    2.2    2.9
## 10 R_12i26Q... male    20 no      2.4    2.1    1.8    2.2    3.4    2.9    2.15    2.4
## # ... with 291 more rows, 3 more variables: OHQ <dbl>, FoC <dbl>, Foc2 <dbl>, and
## #   abbreviated variable names 1Participant, 2victim_crime
```



Summarising groups

Once data is *grouped*, the most common thing to do is to `summarise()` those groups.

```
summarise(grouped_crime,  
           state_anxiety = mean(SA),  
           sd_SA = sd(SA),  
           var_SA = var(SA))
```

```
## `summarise()` has grouped output by 'sex'. You can override using the `.groups`  
## argument.
```

```
## # A tibble: 4 × 5  
## # Groups:   sex [2]  
##   sex    victim_crime state_anxiety sd_SA var_SA  
##   <chr>   <chr>           <dbl> <dbl> <dbl>  
## 1 female no             1.90  0.518  0.268  
## 2 female yes            1.98  0.643  0.413  
## 3 male   no             2.02  0.553  0.306  
## 4 male   yes            1.74  0.472  0.223
```

Removing unwanted rows



Filtering rows

The `prison_pop` dataset has 22409 rows, but we don't need (or want) them all!

```
unique(prison_pop$View)
```

```
## [1] "a Establishment*Sex*Age Group"      "b Establishment*Sex*Custody type"
## [3] "c Establishment*Sex*Nationality"    "d Establishment*Sex*Offence group"
```

The data is actually *repeated* four times, but organised differently each time.

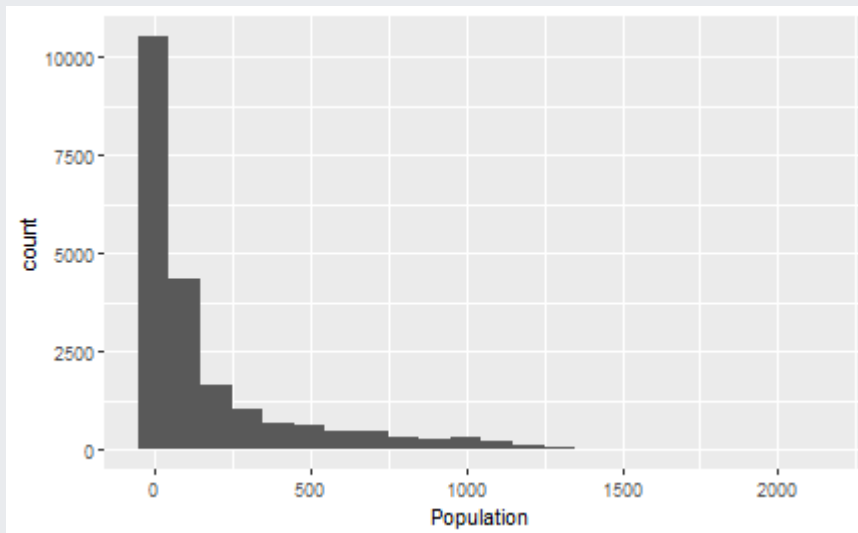
```
## # A tibble: 4 × 3
##   View                total_pop num_entries
##   <chr>                <dbl>     <int>
## 1 a Establishment*Sex*Age Group      938760      2042
## 2 b Establishment*Sex*Custody type    939314      2740
## 3 c Establishment*Sex*Nationality     938841      3215
## 4 d Establishment*Sex*Offence group   936191     14412
```



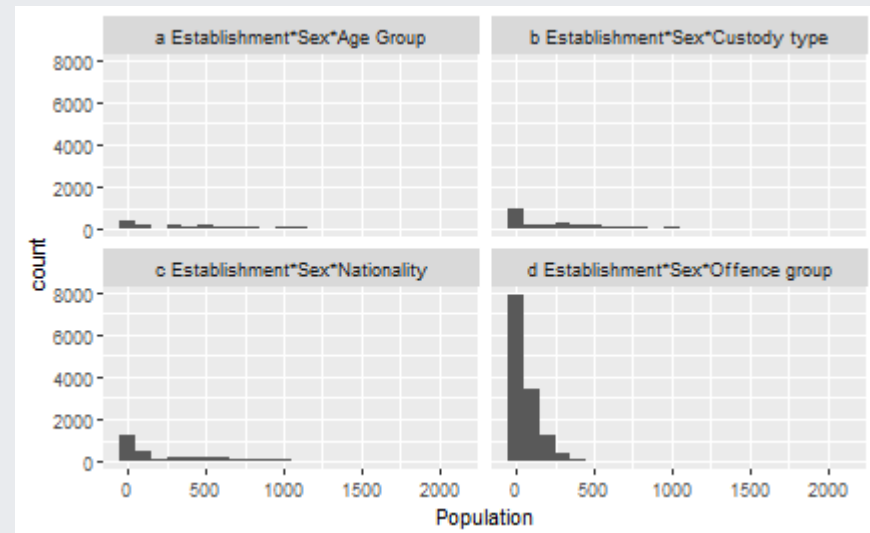
Filtering rows

If we just started investigating the data without accounting for this, it would be misleading.

```
ggplot(prison_pop, aes(x = Population)) +  
  geom_histogram(binwidth = 100)
```



```
ggplot(prison_pop, aes(x = Population)) +  
  geom_histogram(binwidth = 100) + facet_wrap
```





Filtering rows

We can use the `filter()` function to select only the rows we're interested in, using *logical conditions* and *relational operators*.

```
filter(prison_pop,  
       View == "a Establishment*Sex*Age Group")
```

```
## # A tibble: 2,042 × 6  
##   View      Date Establishment Sex Age / Cu...1 Popul...2  
##   <chr>    <chr>    <chr>    <chr> <chr>      <dbl>  
## 1 a Establishment*Sex*Age Group 2015-06 Altcourse Male Adults (2... 922  
## 2 a Establishment*Sex*Age Group 2015-06 Altcourse Male Juveniles... 169  
## 3 a Establishment*Sex*Age Group 2015-06 Ashfield Male Adults (2... 389  
## 4 a Establishment*Sex*Age Group 2015-06 Askham Grange Female Adults (2... NA  
## 5 a Establishment*Sex*Age Group 2015-06 Askham Grange Female Juveniles... NA  
## 6 a Establishment*Sex*Age Group 2015-06 Aylesbury Male Adults (2... 113  
## 7 a Establishment*Sex*Age Group 2015-06 Aylesbury Male Juveniles... 268  
## 8 a Establishment*Sex*Age Group 2015-06 Bedford Male Adults (2... 459  
## 9 a Establishment*Sex*Age Group 2015-06 Bedford Male Juveniles... 30  
## 10 a Establishment*Sex*Age Group 2015-06 Belmarsh Male Adults (2... 794  
## # ... with 2,032 more rows, and abbreviated variable names  
## #   1`Age / Custody / Nationality / Offence Group`, 2Population
```

Relational operators

Relational operators compare two (or more) things and return a **logical** value (i.e. TRUE/FALSE)

Operator	Meaning	Example
>	Greater than	5 > 4
>=	Greater than or equal to	4 >= 4
<	Less than	Population < 400
<=	Less than or equal to	Population <= 400
==	Exactly equal to	Sex == "Male"
!=	Not equal to	Establishment != "Ashfield"
%in%	Is contained in	Establishment %in% c("Bedford", "Oakwood")

Logical operators

Logical operators can be used to combine multiple relational operators or *negate* a relational operator.

Operator	Meaning	Example
&	AND	Population < 1000 & Sex == "Male"
	OR	Population > 200 Population < 500
!	NOT	!(Establishment %in% c("Bedford", "Oakwood"))



Filtering rows

We can have multiple *conditions* for selection with `filter()`.

Suppose we only wanted to include rows where Population is over 300 but under 600.

```
filter(prison_pop,  
  View == "a Establishment*Sex*Age Group",  
  Population > 300 & Population < 600)
```

```
## # A tibble: 487 × 6  
##   View      Date Establishment Sex Age / Cu...1 Popul...2  
##   <chr>    <chr>    <chr>    <chr> <chr>      <dbl>  
## 1 a Establishment*Sex*Age Group 2015-06 Ashfield      Male Adults (2... 389  
## 2 a Establishment*Sex*Age Group 2015-06 Bedford      Male Adults (2... 459  
## 3 a Establishment*Sex*Age Group 2015-06 Brinsford    Male Juveniles... 349  
## 4 a Establishment*Sex*Age Group 2015-06 Bristol      Male Adults (2... 553  
## 5 a Establishment*Sex*Age Group 2015-06 Bronzefield  Female Adults (2... 459  
## 6 a Establishment*Sex*Age Group 2015-06 Buckley Hall Male Adults (2... 440  
## 7 a Establishment*Sex*Age Group 2015-06 Coldingley  Male Adults (2... 515  
## 8 a Establishment*Sex*Age Group 2015-06 Deerbolt     Male Juveniles... 311  
## 9 a Establishment*Sex*Age Group 2015-06 Eastwood Park Female Adults (2... 331  
## 10 a Establishment*Sex*Age Group 2015-06 Erlestoke    Male Adults (2... 514  
## # ... with 477 more rows, and abbreviated variable names  
## #   1'Age / Custody / Nationality / Offence Group', 2Population
```

Putting it all together



Pipes

Often you want to conduct several steps, one after the other.

You could do this using objects to store each intermediate step.

```
temp_pris <- filter(prison_pop,  
                    View == "a Establishment*Sex*Age Group",  
                    Date == "2015-06")  
temp_pris <- group_by(temp_pris,  
                      Sex,  
                      `Age / Custody / Nationality / Offence Group`)  
temp_pris <- summarise(temp_pris,  
                      mean_pop = mean(Population, na.rm = TRUE),  
                      median_pop = median(Population, na.rm = TRUE),  
                      total_pop = sum(Population, na.rm = TRUE),  
                      max_pop = max(Population, na.rm = TRUE))
```

```
## `summarise()` has grouped output by 'Sex'. You can override using the `.groups`  
## argument.
```

Pipes



A simpler way is to use *pipes* (%>%)

pipes can be read as meaning "AND THEN"

```
prison_pop %>%  
  filter(View == "a Establishment*Sex*Age Group",  
         Date == "2015-06") %>%  
  group_by(Sex, `Age / Custody / Nationality / Offence Group`) %>%  
  summarise(mean_pop = mean(Population, na.rm = TRUE),  
            median_pop = median(Population, na.rm = TRUE),  
            total_pop = sum(Population, na.rm = TRUE),  
            max_pop = max(Population, na.rm = TRUE))
```

```
## `summarise()` has grouped output by 'Sex'. You can override using the `.groups`  
## argument.
```

```
## # A tibble: 4 × 6  
## # Groups:   Sex [2]  
##   Sex      Age / Custody / Nationality / Offence...1 mean_...2 media...3 total...4 max_pop  
##   <chr>   <chr>                                     <dbl>    <dbl>    <dbl>    <dbl>  
## 1 Female Adults (21+)                             356      333     3560     480  
## 2 Female Juveniles and Young Adults (15-20)         18.6     19      167      35  
## 3 Male   Adults (21+)                             717.     677    76730    1587
```

Reading materials

Revision

For revision of this week's concepts, see Chapter *Data transformation* in R for Data Science.

For practice, use the "Work with Data" RStudio cloud primer.

Next week

Discovering Statistics using R (Field et al.)

- Chapter 9, Comparing two means
- Chapter 5, Exploring assumptions (additional)