Intro2R

Data Wrangling and Visualisation

Song, Xiao Ping
xp.song@u.nus.edu

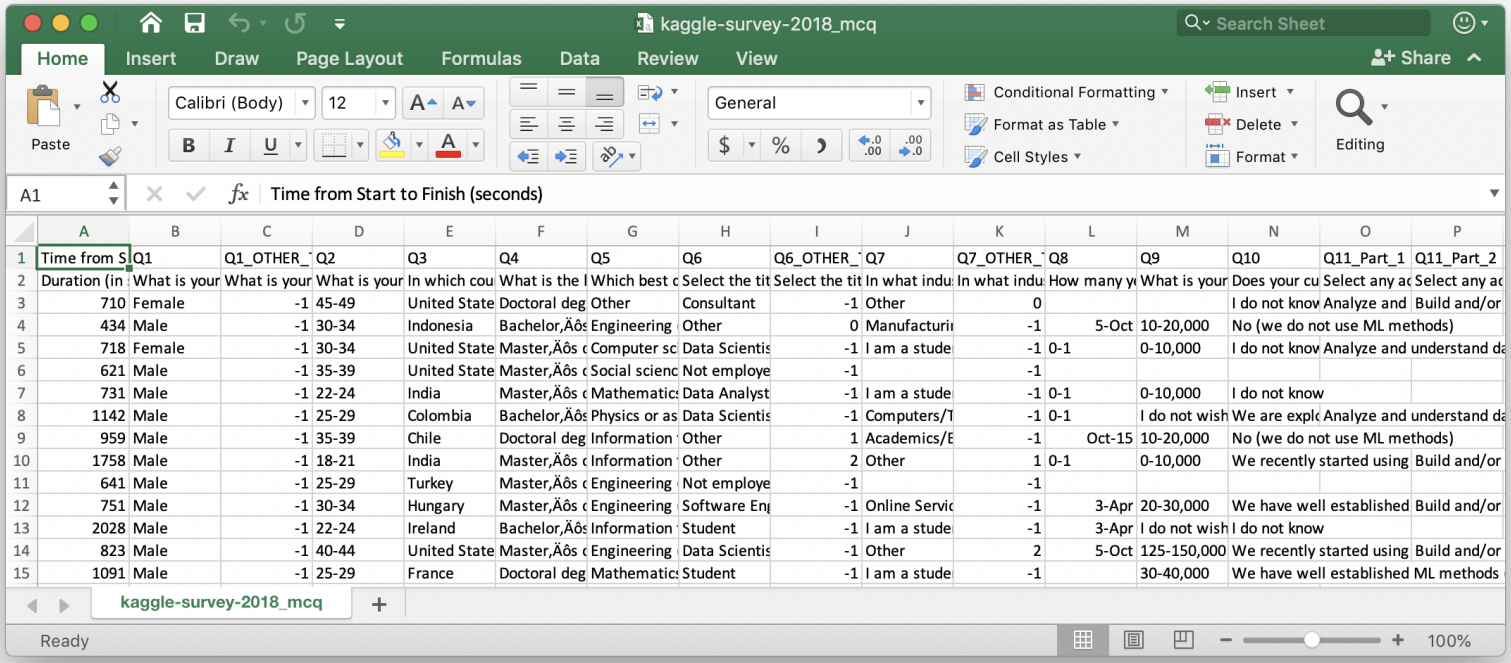
Course materials: https://github.com/xp-song/Intro2R
updated 2022-10-18

Outline

About our Dataset

Kaggle Machine Learning and Data Science Survey 2018

- The industry-wide survey presents the state of data science and machine learning
- We will be analysing multiple choice responses `/data/kaggle-survey-2018_mcq.csv`



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Time from S	Q1	Q1_OTHER_	Q2	Q3	Q4	Q5	Q6	Q6_OTHER_	Q7	Q7_OTHER_	Q8	Q9	Q10	Q11_Part_1	Q11_Part_2
2	Duration (in	What is your	What is your	What is your	In which cou	What is the l	Which best c	Select the tit	Select the tit	In what indu	In what indu	How many y	What is your	Does your cu	Select any ac	Select any ac
3	710	Female	-1	45-49	United State	Doctoral deg	Other	Consultant	-1	Other	0			I do not know	Analyze and	Build and/or
4	434	Male	-1	30-34	Indonesia	Bachelor,Âôs	Engineering	Other	0	Manufacturi	-1	5-Oct	10-20,000	No (we do not use ML methods)		
5	718	Female	-1	30-34	United State	Master,Âôs d	Computer sc	Data Scientis	-1	I am a stude	-1	0-1	0-10,000	I do not know	Analyze and understand da	
6	621	Male	-1	35-39	United State	Master,Âôs d	Social scienc	Not employe	-1		-1					
7	731	Male	-1	22-24	India	Master,Âôs d	Mathematics	Data Analyst	-1	I am a stude	-1	0-1	0-10,000	I do not know		
8	1142	Male	-1	25-29	Colombia	Bachelor,Âôs	Physics or as	Data Scientis	-1	Computers/T	-1	0-1	I do not wish	We are explo	Analyze and understand da	
9	959	Male	-1	35-39	Chile	Doctoral deg	Information	Other	1	Academics/E	-1	Oct-15	10-20,000	No (we do not use ML methods)		
10	1758	Male	-1	18-21	India	Master,Âôs d	Information	Other	2	Other	1	0-1	0-10,000	We recently started using	Build and/or	
11	641	Male	-1	25-29	Turkey	Master,Âôs d	Engineering	Not employe	-1		-1					
12	751	Male	-1	30-34	Hungary	Master,Âôs d	Engineering	Software Eng	-1	Online Servic	-1	3-Apr	20-30,000	We have well established	Build and/or	
13	2028	Male	-1	22-24	Ireland	Bachelor,Âôs	Information	Student	-1	I am a stude	-1	3-Apr	I do not wish	I do not know		
14	823	Male	-1	40-44	United State	Master,Âôs d	Engineering	Data Scientis	-1	Other	2	5-Oct	125-150,000	We recently started using	Build and/or	
15	1091	Male	-1	25-29	France	Doctoral deg	Mathematics	Student	-1	I am a stude	-1		30-40,000	We have well established	ML methods	

About our Dataset

More about Kaggle

- An online community for data science
- Owned by Google (>1 mil users in 2017)
- Users can find/publish data and analysis, and take part in data science competitions

Our analysis includes code adapted from R Notebooks created by the Kaggle users Heads or Tails and Jose Berengueres

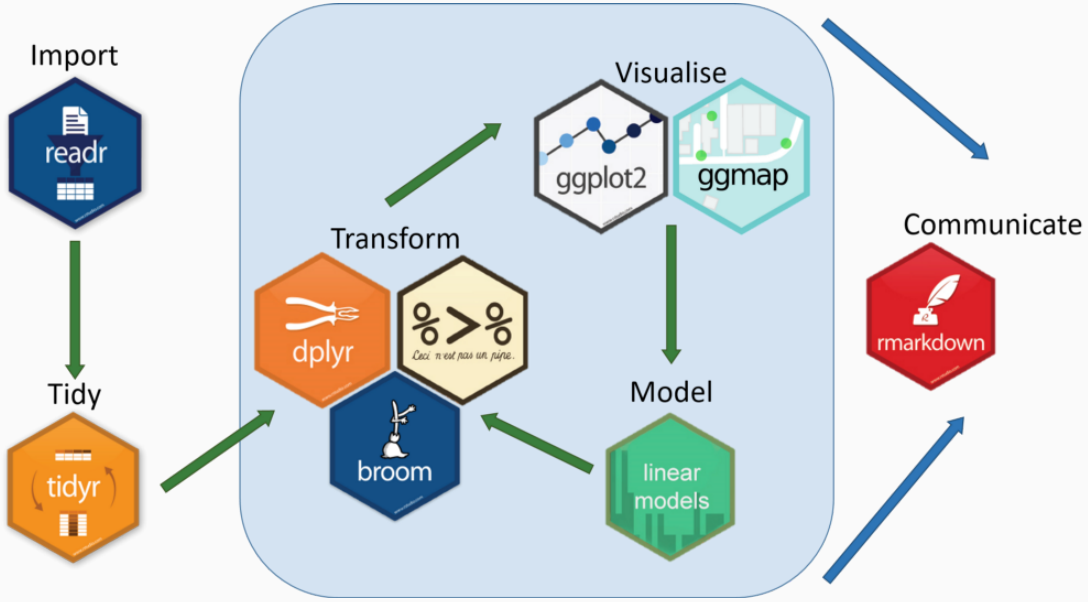
Before we begin...

Install and load packages

Before we begin...

Let's install the tidyverse collection of packages

tidyverse: A collection of packages commonly used for data analyses



Example workflow (medium.com)

Before we begin...

Let's install the tidyverse collection of packages

```
install.packages("tidyverse", dependencies = TRUE) # don't forget quotes
```

- Type `n` if you get the following prompt:
  `Do you want to install from sources the package which needs compilation?`
- Click 'Yes' if you are asked to restart R

Load these packages into R

```
library(tidyverse) # no need quotes
```

Before we begin...

"Tidy" data

- Tabular data (2D)
- Each variable is a column & each observation is a row
- Can be in long or wide format

country	year	key	value
Afghanistan	1999	infected	135
Afghanistan	1999	population	19839494
Afghanistan	2020	infected	384
Afghanistan	2020	population	21739203
Australia	1999	infected	34
Australia	1999	population	23423534
Australia	2020	infected	45
Australia	2020	population	23346436
Belgium	1999	infected	272
Belgium	1999	population	49273820
Belgium	2020	infected	274
Belgium	2020	population	48928472

country	year	infected	population
Afghanistan	1999	135	19839494
Afghanistan	2020	384	21739203
Australia	1999	34	23423534
Australia	2020	45	23346436
Belgium	1999	272	49273820
Belgium	1999	274	48928472

Outline

Load data

Import tabular data as *tibbles* using `readr::read_csv()`

Tibbles are dataframes with stricter rules that avoid hassle/errors often associated with conventional dataframes.

```
multi <- read_csv("data/kaggle-survey-2018_mcq.csv", skip = 1)
```

```
head(multi)
```

```
## # A tibble: 6 × 395
##   `Duration (in seconds)` `What is your gen… `What is your gen… `What is your a…
##                     <dbl> <chr>                           <dbl> <chr>
## 1                     710 Female                             -1 45-49
## 2                     434 Male                               -1 30-34
## 3                     718 Female                             -1 30-34
## 4                     621 Male                               -1 35-39
## 5                     731 Male                               -1 22-24
## 6                    1142 Male                               -1 25-29
## # … with 391 more variables: In which country do you currently reside? <chr>,
## #   What is the highest level of formal education that you have attained or plan to attain within the next
## #   Which best describes your undergraduate major? - Selected Choice <chr>,
## #   Select the title most similar to your current role (or most recent title if retired): - Selected Choic
```

Load data

Let's compare `read_csv()` with `read.csv()` in base R

```
multi2 <- read.csv("data/kaggle-survey-2018_mcq.csv", skip = 1)
head(multi2)
```

```
##   Duration..in.seconds. What.is.your.gender....Selected.Choice
## 1                   710                                 Female
## 2                   434                                   Male
## 3                   718                                 Female
## 4                   621                                   Male
## 5                   731                                   Male
## 6                  1142                                   Male
##   What.is.your.gender....Prefer.to.self.describe...Text
## 1                                                    -1
## 2                                                    -1
## 3                                                    -1
## 4                                                    -1
## 5                                                    -1
## 6                                                    -1
##   What.is.your.age....years.. In.which.country.do.you.currently.reside.
## 1                       45-49                 United States of America
## 2                       30-34                                Indonesia
## 3                       30-34                 United States of America
```

Examine data

Examine column names

```
head(colnames(multi))
```

```
## [1] "Duration (in seconds)"
## [2] "What is your gender? - Selected Choice"
## [3] "What is your gender? - Prefer to self-describe - Text"
## [4] "What is your age (# years)?"
## [5] "In which country do you currently reside?"
## [6] "What is the highest level of formal education that you have attained or plan to attain within the nex
```

- Analysing colnames as entire sentences is not very feasible at scale (we'll abbreviate the colnames later)

Print the first column by name

```
multi$`Duration (in seconds)`
```

- Have to wrap colname with backticks (because of white spaces)

Convert data

The pipe operator `%>%`

Frequently used to manipulate data in stages/sequence

E.g.:

```
round(exp(diff(log(x))), 1) #using nested brackets


x %>% log() %>% #using the pipe operator
    diff() %>%
    exp() %>%
    round(1)
```

Convert data

Multiple choice questions have categorical answers with discrete levels—i.e. Factors!

Convert columns with *character* data into *factors* using `mutate()`

```
multi <- multi %>%
  mutate(across(.cols = is.character, .fns = as.factor)
```

Outline

Sample Size

How many respondents are there?

```
nrow(multi)
```

```
## [1] 23859
```

Survey duration

Abbreviate the colname `Duration (in seconds)` to `duration` using `rename()`

```
multi <- multi %>%
  rename(duration = `Duration (in seconds)`)
```

Change the units from seconds to minutes using `mutate()`

```
multi <- multi %>%
  mutate(duration = duration/60) # overwrite the colname
```

Print out first few rows of `multi$duration`

```
head(multi$duration)
```

```
## [1] 11.833333  7.233333 11.966667 10.350000 12.183333 19.033333
```

Survey duration

Plot a histogram using the `ggplot2::ggplot()` function

Three basic steps:

1. Provide *data*
2. Assign your data *variables* to *aesthetics*
3. Assign the graphical *primitives*

```
multi %>%
  ggplot(aes(duration)) +
  geom_histogram()
```

Survey duration

Plot a histogram using the `ggplot2::ggplot()` function

Three basic steps:

1. Provide *data*
2. Assign your data *variables* to *aesthetics*
3. Assign the graphical *primitives*

```
multi %>%
  ggplot(aes(duration)) +
  geom_histogram() +
  geom_vline(xintercept = median(multi$duration))
```

Survey duration

Plot a histogram using the `ggplot2::ggplot()` function

Three basic steps:

1. Provide *data*
2. Assign your data *variables* to *aesthetics*
3. Assign the graphical *primitives*

```
multi %>%
  ggplot(aes(duration)) +
  geom_histogram() +
  geom_vline(xintercept = median(multi$duration)) +
  scale_x_log10()
```

Survey duration

Plot a histogram using the `ggplot2::ggplot()` function

Three basic steps:

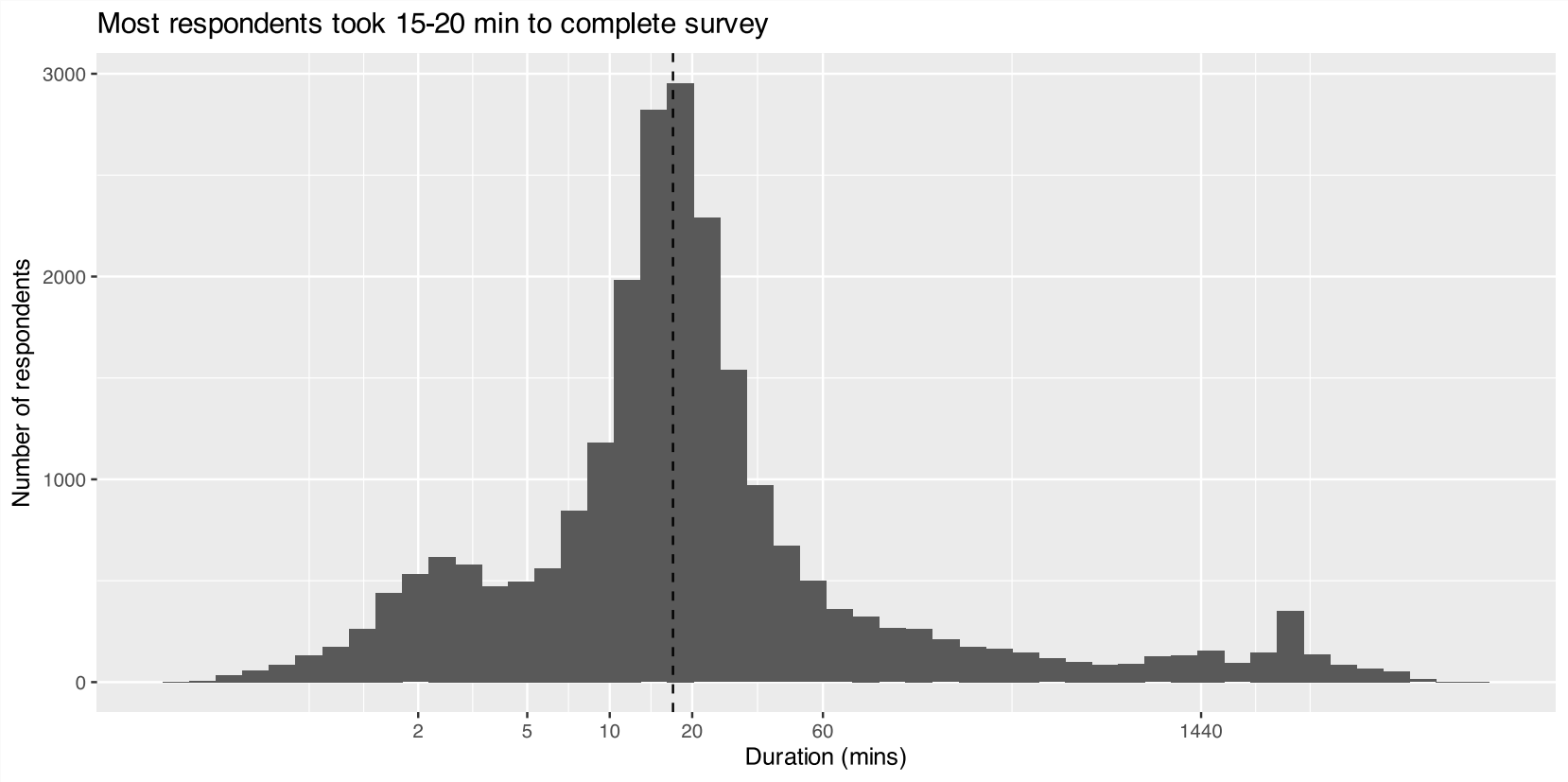
1. Provide *data*
2. Assign your data *variables* to *aesthetics*
3. Assign the graphical *primitives*

```
multi %>%
  ggplot(aes(duration)) +
  geom_histogram(bins = 50) +
  geom_vline(xintercept = median(multi$duration), linetype = 2) +

  scale_x_log10(breaks = c(2, 5, 10, 20, 60, 1440)) + #address extreme x-values

  #customisation
  labs(x = "Duration (mins)", y = "Number of respondents") + #change axis labels
  ggtitle("Most respondents took 15-20 min to complete survey") #add figure title
```

Survey duration



Note: The dashed line denotes the median survey duration. The x-axis has been transformed to a logarithmic scale.

Outline

Geographical distribution

Abbreviate the colname `In which country do you currently reside?` to `country`

```
multi <- multi %>%
  rename(country = `In which country do you currently reside?`)
```

Examine the column

```
head(multi$country)
```

```
## [1] "United States of America" "Indonesia"
## [3] "United States of America" "United States of America"
## [5] "India"                    "Colombia"
```

Geographical distribution

Summarise the number of respondents per country

```
ctry_n <- multi %>%
  count(country)
```

Geographical distribution

Summarise the number of respondents per country

Show 6 entries Search:

country	n
Argentina	119
Australia	330
Austria	62
Bangladesh	107
Belarus	91
Belgium	111

Showing 1 to 6 of 58 entries

Previous 1 2 3 4 5 ... 10 Next

Geographical distribution

Remove rows with certain answers in our summary table `ctry_n`

```
ctry_n <- ctry_n %>%
  filter(!(country %in% c("Other", "I do not wish to disclose my location")))
```

```
ctry_n
```

```
## # A tibble: 56 × 2
##    country        n
##    <chr>      <int>
##  1 Argentina    119
##  2 Australia    330
##  3 Austria       62
##  4 Bangladesh   107
##  5 Belarus       91
##  6 Belgium      111
##  7 Brazil       736
##  8 Canada       604
##  9 Chile         76
## 10 China       1644
## # … with 46 more rows
```

Geographical distribution

Map the country name to the ISO3 country code using the function `countrycode()`

- `install.packages("countrycode")`
- Add data as a new column named `iso3` using `mutate()`

```
ctry_n <- ctry_n %>%
  mutate(iso3 = countrycode(country, origin = "country.name", destination = "iso3c"))
```

```
ctry_n
```

```
## # A tibble: 56 × 3
##    country        n iso3
##    <chr>      <int> <chr>
##  1 Argentina    119 ARG
##  2 Australia    330 AUS
##  3 Austria       62 AUT
##  4 Bangladesh   107 BGD
##  5 Belarus       91 BLR
##  6 Belgium      111 BEL
##  7 Brazil       736 BRA
##  8 Canada       604 CAN
##  9 Chile         76 CHL
```

Dealing with duplicates

Let's check if the number of country names & country codes match

```
length(unique(ctry_n$country))
```

```
## [1] 56
```

```
length(unique(ctry_n$iso3))
```

```
## [1] 55
```

Check which elements are duplicated with `duplicated()`

```
duplicated(ctry_n$iso3)
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [37] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
## [49] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

Dealing with duplicates

Subset all rows with duplicates

```
ctry_n[duplicated(ctry_n$iso3) | duplicated(ctry_n$iso3, fromLast=TRUE),]
```

```
## # A tibble: 2 × 3
##   country               n iso3
##   <chr>             <int> <chr>
## 1 Republic of Korea    71 KOR
## 2 South Korea         188 KOR
```

Geographical distribution

Dealing with duplicates

Group the dataframe by `iso3`, then add up the no. of respondents `n`

```
ctry_n <- ctry_n %>%
  group_by(iso3) %>%
  summarise(country = first(country), #get the first value for country name
            n = sum(n)) #sum up duplicates
```

```
ctry_n
```

```
## # A tibble: 55 × 3
##    iso3  country         n
##    <chr> <chr>       <int>
##  1 ARG   Argentina     119
##  2 AUS   Australia     330
##  3 AUT   Austria        62
##  4 BEL   Belgium       111
##  5 BGD   Bangladesh    107
##  6 BLR   Belarus        91
##  7 BRA   Brazil        736
##  8 CAN   Canada        604
```

Geographical distribution

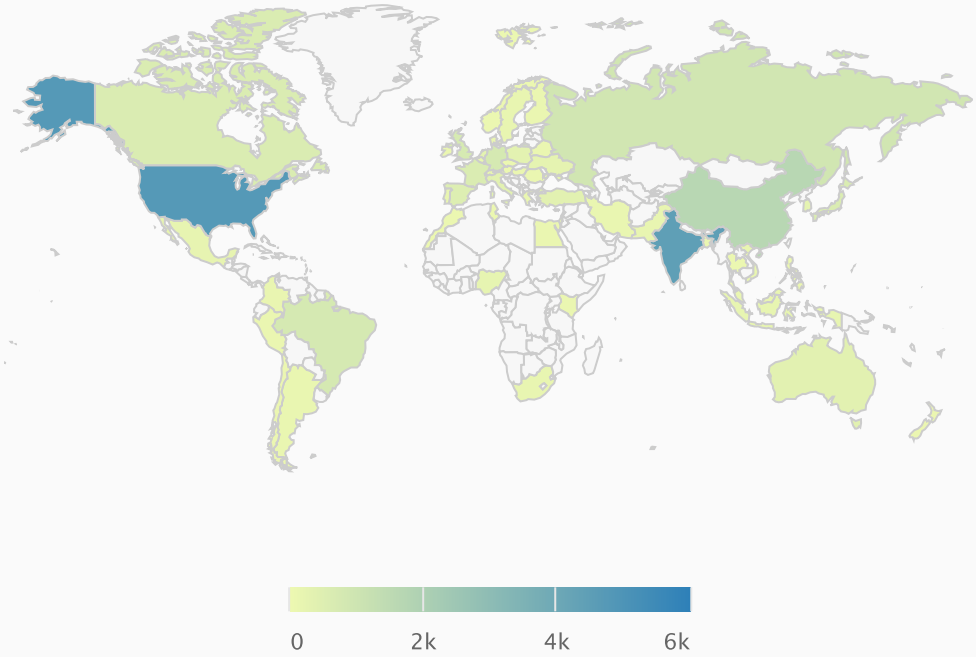
Plot `ctry_n` as an interactive map using `highcharter::highchart()`

```
highchart() %>%
  hc_add_series_map(worldgeojson, #map data to add
                    ctry_n, #our data
                    value = 'n', #colname of interest in ctry_n
                    joinBy = 'iso3') %>% #both datasets have this colname (country code)

  #customisation
  hc_title(text = 'Geographical distribution of survey respondents') %>%
  hc_colorAxis(minColor = "#edf8b1", maxColor = "#2c7fb8") %>%
  hc_tooltip(useHTML = TRUE, headerFormat = "", pointFormat = "{point.country}: {point.n} respondents")
```

Geographical distribution

Geographical distribution of survey respondents



Survey Analysis: Country-level Data

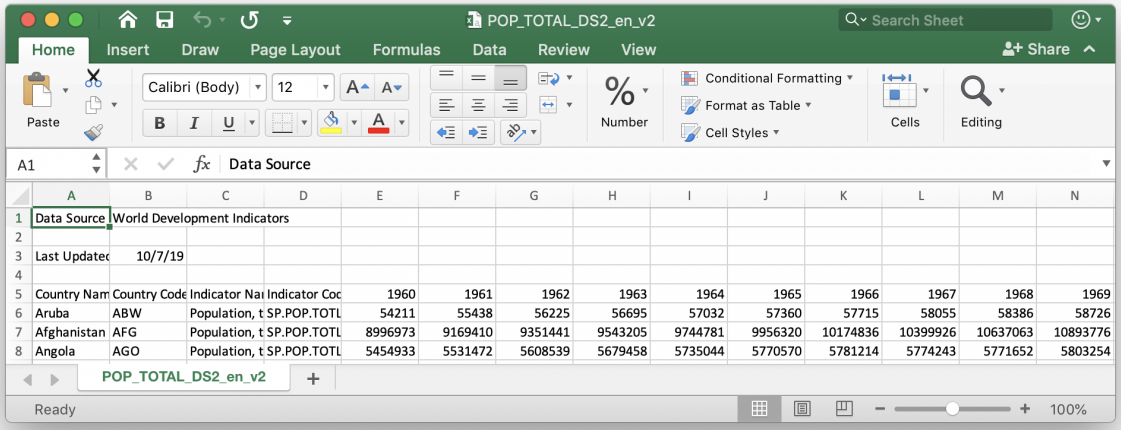
Country-level Data

Datasets

Total population count (The World Bank)

```
pops <- read_csv("data/POP_TOTAL_DS2_en_v2.csv", skip = 4)

pops <- pops %>%
  rename(country = "Country Name", population = "2018") %>%
  select(country, population) %>% #we are only interested in these 2 columns
  mutate(iso3 = countrycode(country, origin = "country.name",
                                destination = "iso3c"))
```



Country-level Data

Datasets

Total population count (The World Bank)

```
pops
```

```
## # A tibble: 264 × 3
##    country              population iso3
##    <chr>                     <dbl> <chr>
##  1 Aruba                    105845 ABW
##  2 Afghanistan            37172386 AFG
##  3 Angola                 30809762 AGO
##  4 Albania                 2866376 ALB
##  5 Andorra                   77006 AND
##  6 Arab World            419790588 <NA>
##  7 United Arab Emirates    9630959 ARE
##  8 Argentina              44494502 ARG
##  9 Armenia                 2951776 ARM
## 10 American Samoa            55465 ASM
## # … with 254 more rows
```

Country-level Data

Datasets

Total population count (The World Bank)

- **Check for `NA` values using `is.na()`**

```
pops[is.na(pops$iso3),]
```

```
## # A tibble: 49 × 3
##    country                                        population iso3
##    <chr>                                               <dbl> <chr>
##  1 Arab World                                      419790588 <NA>
##  2 Central Europe and the Baltics                  102511922 <NA>
##  3 Channel Islands                                    170499 <NA>
##  4 Caribbean small states                            7358965 <NA>
##  5 East Asia & Pacific (excluding high income)    2081651801 <NA>
##  6 Early-demographic dividend                     3249140605 <NA>
##  7 East Asia & Pacific                            2328220870 <NA>
##  8 Europe & Central Asia (excluding high income)   417797257 <NA>
##  9 Europe & Central Asia                           918793590 <NA>
## 10 Euro area                                       341783171 <NA>
## # … with 39 more rows
```

Country-level Data

Datasets

Total population count (The World Bank)

- **Remove rows with `NA` values**

```
pops <- pops[!is.na(pops$iso3),]
```

```
pops
```

```
## # A tibble: 215 × 3
##    country              population iso3
##    <chr>                     <dbl> <chr>
##  1 Aruba                    105845 ABW
##  2 Afghanistan            37172386 AFG
##  3 Angola                 30809762 AGO
##  4 Albania                 2866376 ALB
##  5 Andorra                   77006 AND
##  6 United Arab Emirates    9630959 ARE
##  7 Argentina              44494502 ARG
##  8 Armenia                 2951776 ARM
```

Country-level Data

Datasets

Global Innovation Index (INSEAD)

```
innov <- read_csv("data/INSEADGlobalinnovationIndex2018.csv")

innov <- innov %>%
  rename(country = "Economy", index = "Score") %>%
  select(country, index) %>%
  mutate(iso3 = countrycode(country, origin = "country.name",
                            destination = "iso3c"))
```

Country-level Data

Datasets

Global Innovation Index (INSEAD)

```
innov
```

```
## # A tibble: 126 × 3
##    country                   index iso3
##    <chr>                     <dbl> <chr>
##  1 Switzerland                68.4 CHE
##  2 Netherlands                63.3 NLD
##  3 Sweden                     63.1 SWE
##  4 United Kingdom             60.1 GBR
##  5 Singapore                  59.8 SGP
##  6 United States of America   59.8 USA
##  7 Finland                    59.6 FIN
##  8 Denmark                    58.4 DNK
##  9 Germany                    58   DEU
## 10 Ireland                    57.2 IRL
## # … with 116 more rows
```

Country-level Data

Datasets

Join

We have 3 summary tables with *'countries'* as data points (each row):

1. `ctry_n` Number of survey respondents
2. `pops` Total population
3. `innov` Innovation index

Country-level Data

Datasets

Join

Combine the three tables using `inner_join()`, based on the variable `iso3`

```
ctry_data <- ctry_n %>%
  inner_join(innov, by = "iso3") %>%
  inner_join(pops, by = "iso3")
```

Country-level Data

Datasets

Join

Combined table `ctry_data`

```
ctry_data
```

```
## # A tibble: 55 × 7
##    iso3  country.x       n country.y   index country     population
##    <chr> <chr>       <int> <chr>       <dbl> <chr>            <dbl>
##  1 ARG   Argentina     119 Argentina    30.7 Argentina     44494502
##  2 AUS   Australia     330 Australia    52   Australia     24992369
##  3 AUT   Austria        62 Austria      51.3 Austria        8847037
##  4 BEL   Belgium       111 Belgium      50.5 Belgium       11422068
##  5 BGD   Bangladesh    107 Bangladesh   23.1 Bangladesh   161356039
##  6 BLR   Belarus        91 Belarus      29.4 Belarus        9485386
##  7 BRA   Brazil        736 Brazil       33.4 Brazil       209469333
##  8 CAN   Canada        604 Canada       53   Canada        37058856
##  9 CHE   Switzerland   164 Switzerland  68.4 Switzerland    8516543
## 10 CHL   Chile          76 Chile        37.8 Chile         18729160
## # … with 45 more rows
```

Country-level Data

Datasets

Join

Remove `country.x` and `country.y`

```
ctry_data <- ctry_data %>%
  select(-c(country.x, country.y))
```

Country-level Data

Datasets

```
ctry_data
```

Join

```
## # A tibble: 55 × 5
##    iso3      n index country     population
##    <chr> <int> <dbl> <chr>            <dbl>
##  1 ARG     119  30.7 Argentina     44494502
##  2 AUS     330  52   Australia     24992369
##  3 AUT      62  51.3 Austria        8847037
##  4 BEL     111  50.5 Belgium       11422068
##  5 BGD     107  23.1 Bangladesh   161356039
##  6 BLR      91  29.4 Belarus        9485386
##  7 BRA     736  33.4 Brazil       209469333
##  8 CAN     604  53   Canada        37058856
##  9 CHE     164  68.4 Switzerland    8516543
## 10 CHL      76  37.8 Chile         18729160
## # … with 45 more rows
```

Country-level Data

Datasets

Join

Save

Let's save `ctry_data` on our computer!

```
write_csv(ctry_data, "output/country_data.csv")
```

Respondents per population

Calculate the no. of respondents as a *proportion of the total population*

- Save it in a new colname `respop10k`

```
ctry_data$respop10k <- ctry_data$n / ctry_data$population * 10000 #respondents per 10k ppl
```

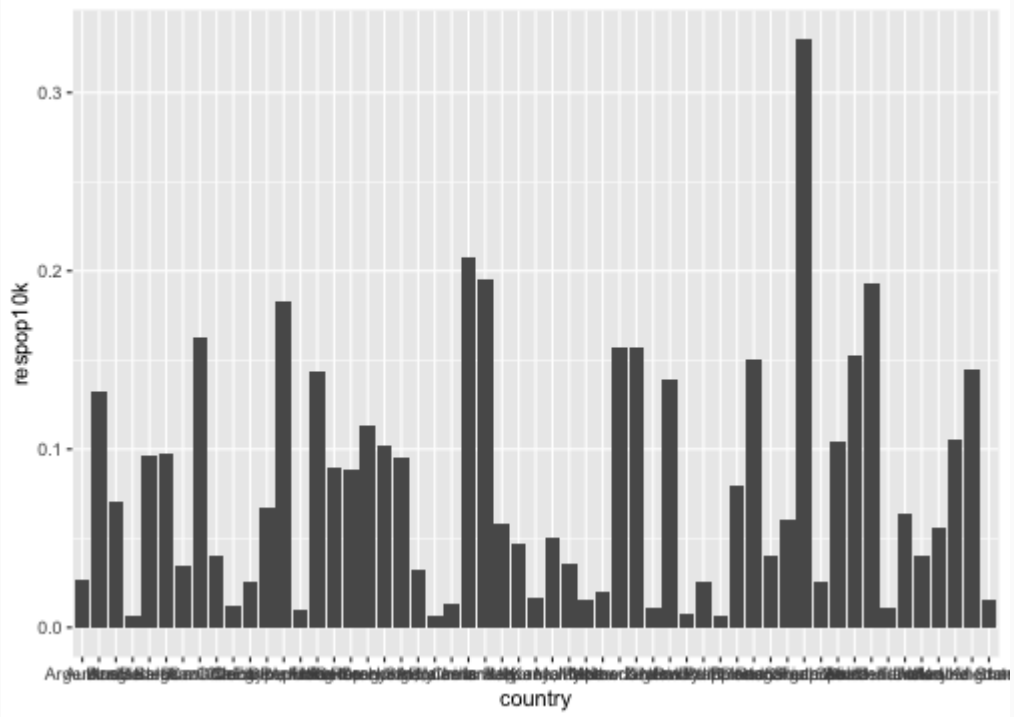
```
ctry_data
```

```
## # A tibble: 55 × 6
##    iso3      n index country     population respop10k
##    <chr> <int> <dbl> <chr>            <dbl>     <dbl>
##  1 ARG     119  30.7 Argentina     44494502   0.0267
##  2 AUS     330  52   Australia     24992369   0.132
##  3 AUT      62  51.3 Austria        8847037   0.0701
##  4 BEL     111  50.5 Belgium       11422068   0.0972
##  5 BGD     107  23.1 Bangladesh   161356039   0.00663
##  6 BLR      91  29.4 Belarus        9485386   0.0959
##  7 BRA     736  33.4 Brazil       209469333   0.0351
##  8 CAN     604  53   Canada        37058856   0.163
##  9 CHE     164  68.4 Switzerland    8516543   0.193
## 10 CHL      76  37.8 Chile         18729160   0.0406
## # … with 45 more rows
```

Respondents per population

Plot a bar chart of `respop10k` for each `country`, using `geom_col` as a graphical primitive

```
ctry_data %>%
  ggplot(aes(x = country, y = respop10k)) +
  geom_col()
```



Respondents per population

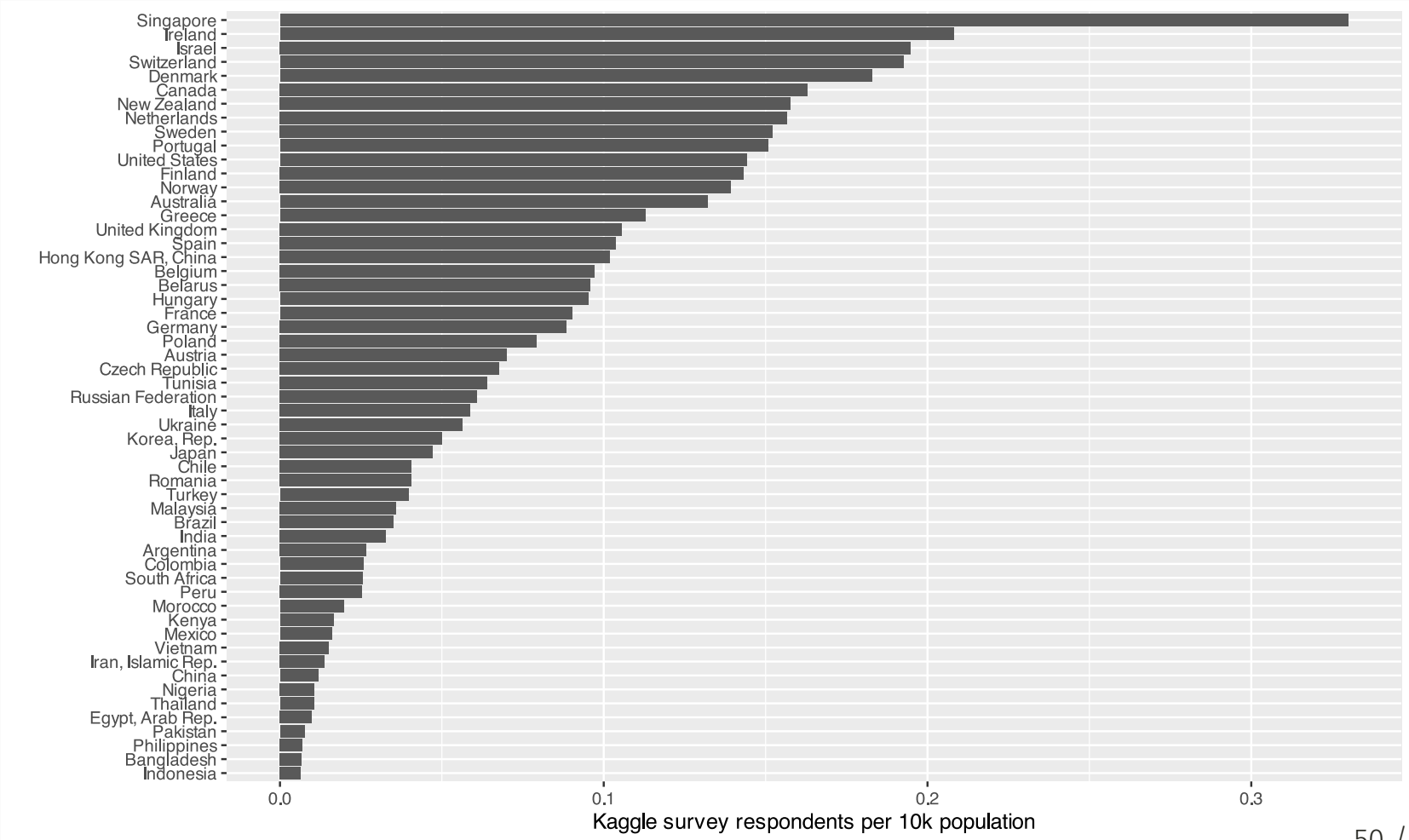
Arrange countries in descending order using `reorder()`, and swap the axes using `coord_flip()`

```
ctryplot <- ctry_data %>%
  ggplot(aes(x = reorder(country, respop10k), #plot countries in desc order
             y = respop10k)) +
  geom_col() +

  #customisation
  labs(x = "", y = "Kaggle survey respondents per 10k population") +
  coord_flip()  #switch x & y axis
```

Respondents per population

```
ctryplot
```



Respondents per population

Save the plot to an image file using `jpeg()`

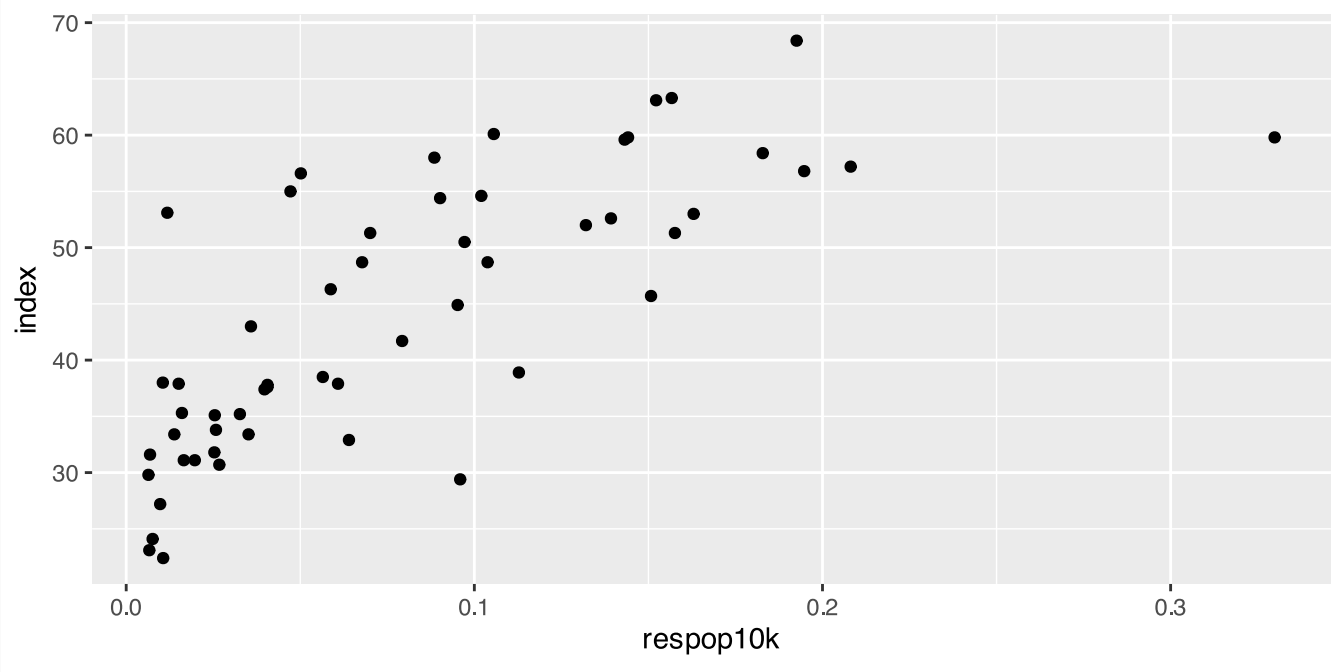
- Run the following code chunk at one go

```
jpeg(filename = "output/ctryplot.jpeg", width = 1800, height = 2000, res = 300)
ctryplot
dev.off() #finish creating the image file
```

Innovation

Make a scatter plot of `respop10k` against the innovation `index`, using `geom_point()` as graphical primitive

```
ggplot(ctry_data, aes(x = respop10k, y = index)) +
  geom_point()
```



Innovation

Add the `label` aesthetic to the `aes()` argument, and add text labels using `ggrepel::geom_text_repel()`

```
ggplot(ctry_data, aes(x = respop10k, y = index,
                      label = country)) +
  geom_point() +
  geom_text_repel()
```

```
## Warning: ggrepel: 18 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```

Innovation

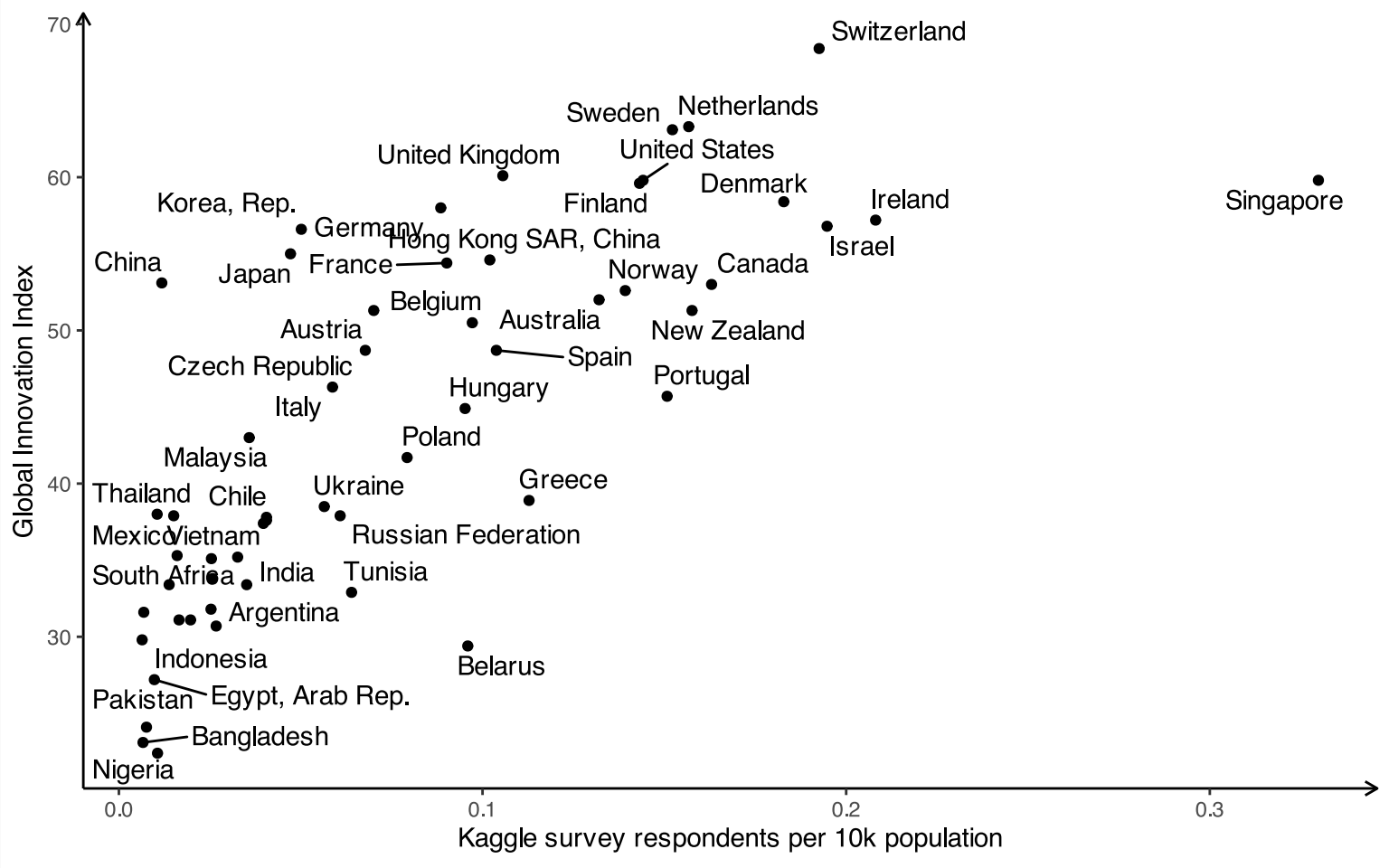
Even more customisation:

```
ggplot(ctry_data, aes(x = respop10k,
                      y = index,
                      label = country)) +
  geom_point() +
  geom_text_repel() +

  #customisation
  labs(y = "Global Innovation Index",
       x = "Kaggle survey respondents per 10k population") +
  theme(legend.position = "none",
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.line = element_line(colour = "black",
        arrow = arrow(length = unit(0.08, "inches"), type = "open")))
```

Innovation

```
## Warning: ggrepel: 9 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```



Innovation

Test the correlation between `respop10k` and the innovation `index` using `cor.test()`

```
cor.test(ctry_data$respop10k, ctry_data$index,
         method = "spearman") #ranked comparisons
```

```
##
##     Spearman's rank correlation rho
##
## data:  ctry_data$respop10k and ctry_data$index
## S = 5627.6, p-value = 3.364e-13
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##       rho
## 0.7969838
```

Outline

It's your turn!

Explore and visualise `data(diamonds, package = "ggplot2")`

Filter diamonds that are less than $3000 with a Premium cut

Hint: Use `summary()` for a summary of the dataset

It's your turn!

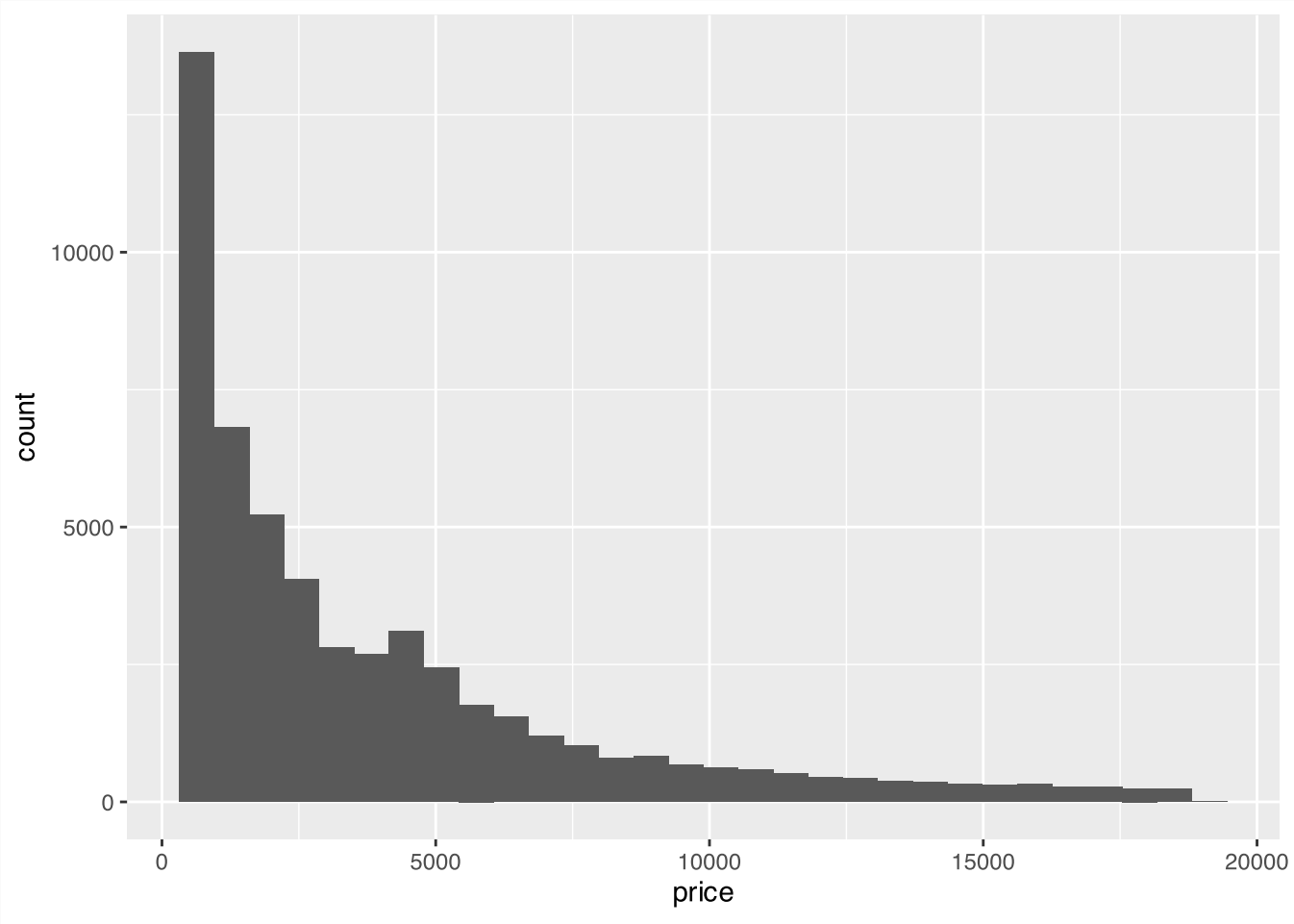
Filter diamonds that are less than $3000 with a Premium cut

```
diamonds[diamonds$price < 3000 & diamonds$cut == "Premium", ]
```

```
## # A tibble: 6,757 × 10
##    carat cut     color clarity depth table price     x     y     z
##    <dbl> <ord>   <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
##  1  0.21 Premium E     SI1      59.8    61   326  3.89  3.84  2.31
##  2  0.29 Premium I     VS2      62.4    58   334  4.2   4.23  2.63
##  3  0.22 Premium F     SI1      60.4    61   342  3.88  3.84  2.33
##  4  0.2  Premium E     SI2      60.2    62   345  3.79  3.75  2.27
##  5  0.32 Premium E     I1       60.9    58   345  4.38  4.42  2.68
##  6  0.24 Premium I     VS1      62.5    57   355  3.97  3.94  2.47
##  7  0.29 Premium F     SI1      62.4    58   403  4.24  4.26  2.65
##  8  0.22 Premium E     VS2      61.6    58   404  3.93  3.89  2.41
##  9  0.22 Premium D     VS2      59.3    62   404  3.91  3.88  2.31
## 10  0.3  Premium J     SI2      59.3    61   405  4.43  4.38  2.61
## # … with 6,747 more rows
```

It's your turn!

Example plot

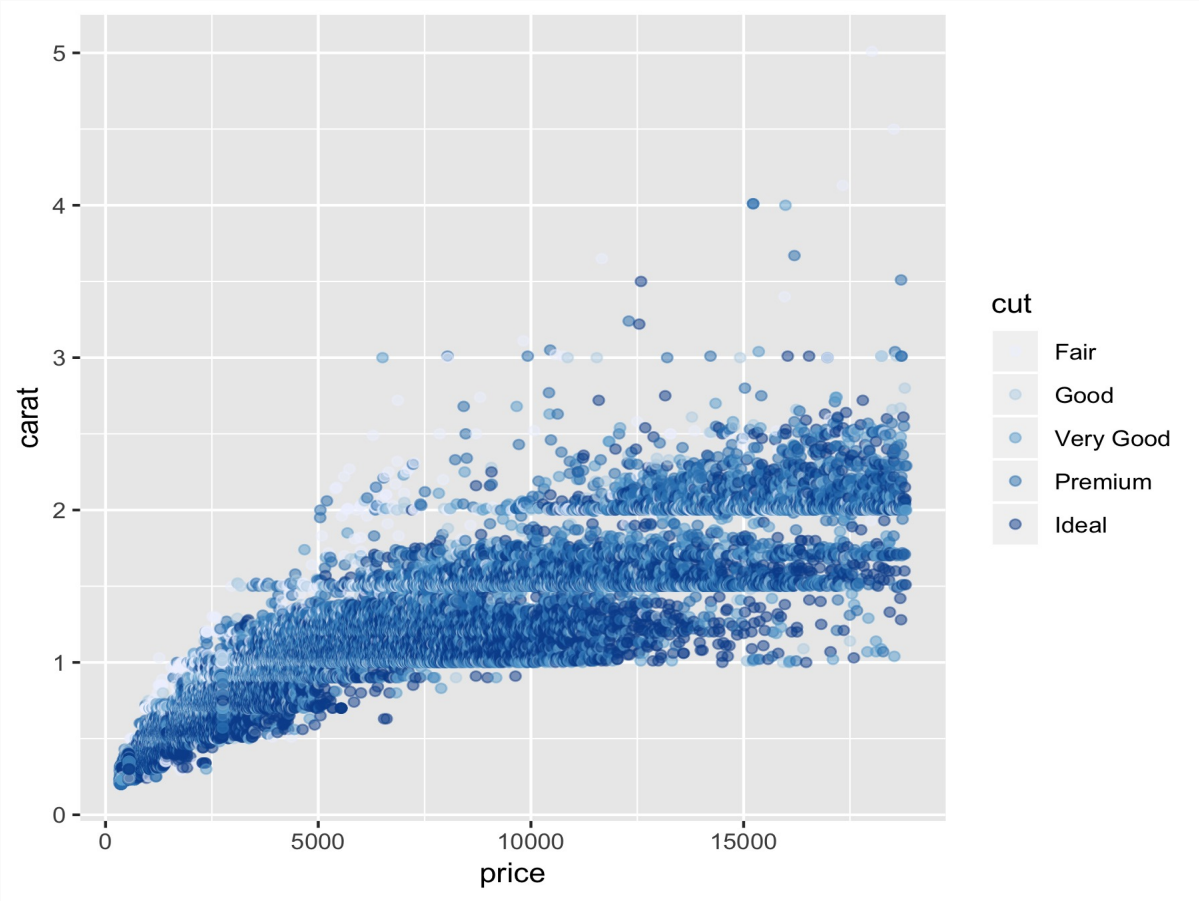


It's your turn!

```
diamonds %>%
  ggplot(aes(x = price)) +
  geom_histogram()
```

It's your turn!

Example plot



It's your turn!

```
diamonds %>%
  ggplot(aes(x = price,
             y = carat,
             color = cut)) +
  geom_point(alpha = 0.5) +

  scale_color_brewer(type = "seq")
```

Questions?

About our dataset

Data preparation

Survey Overview

Survey Analysis

It's your turn!

Further applications

Further applications

Data communication

- Interactive plots with Plotly
- Visualisation with the rCharts package
- Interactive web apps with Shiny
- Alternative outputs for R Markdown documents

Statistics in R

- r-statistics.co by Selva Prabhakaran
- R Tutorial: An Introduction to Statistics
- Learning Statistics with R by Danielle Navarro
- Statistics Fundamentals with R by Datacamp
- Statistics and R by Havard University

Other resources

- Skill tracks in R by Datacamp