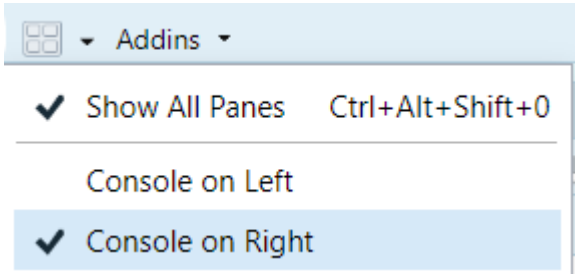
Chapter 2 Workshop

Table of contents

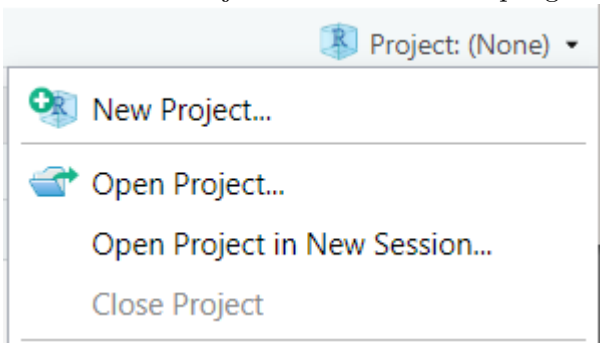
Setting up a Quarto project

It is a good idea to get into the habit of using Quarto projects, rather than just R scripts. Here is a step-by-step guide to creating a project for your workshops. You don't have to use projects, but they are very useful.

1. Open RStudio. (Optional: click on the little window symbol at the top and select "Console on Right")

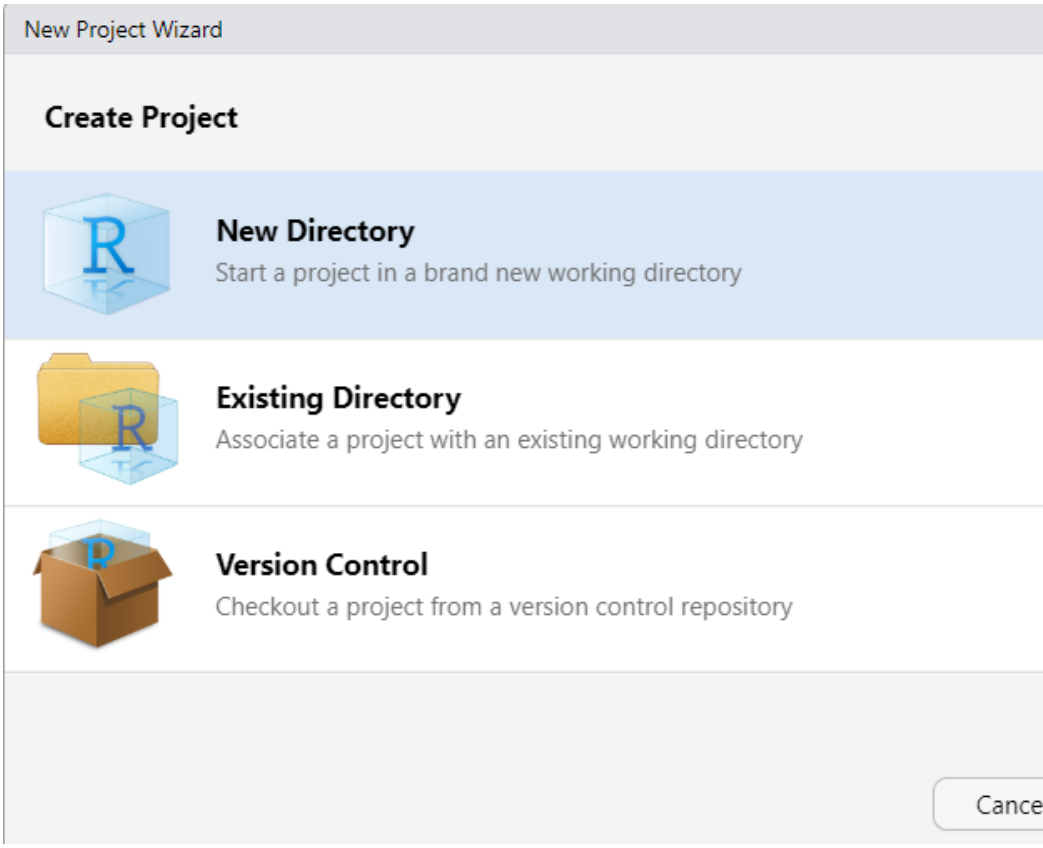


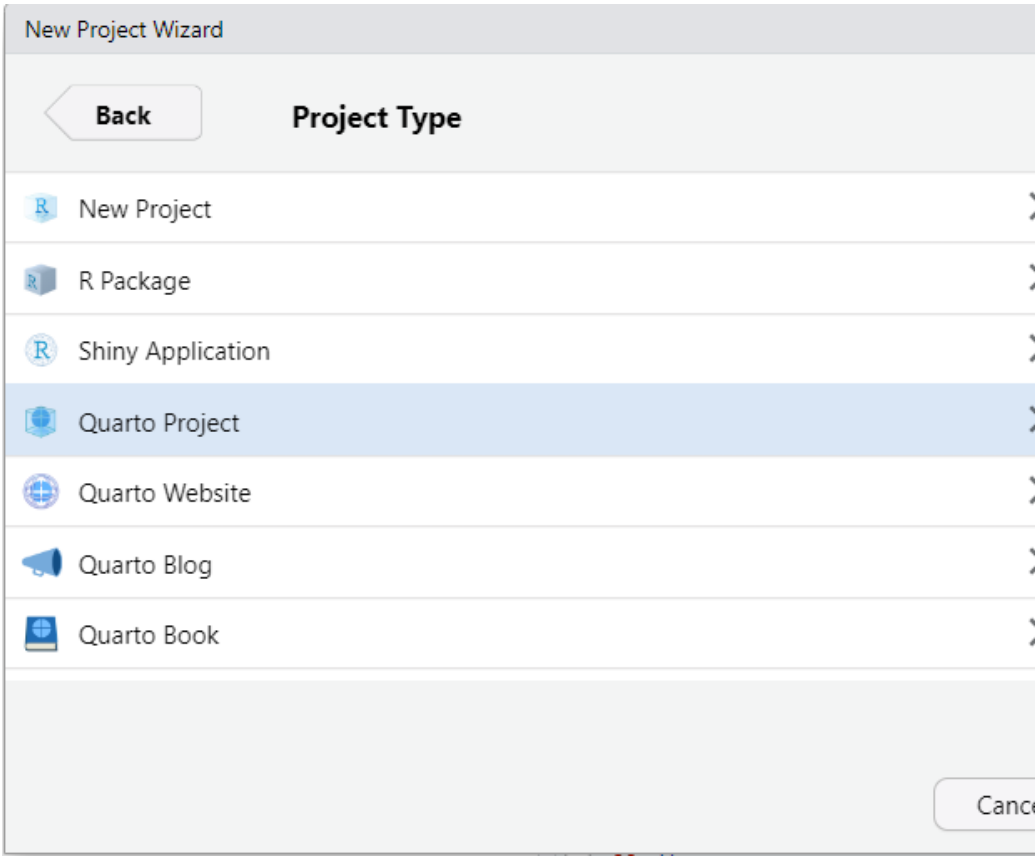
2. If you haven't already, make a directory on your computer where you want to keep your code for this course.
3. Make a new project. Select the "Project" button at the top-right of Rstudio, and select "New Project...". \



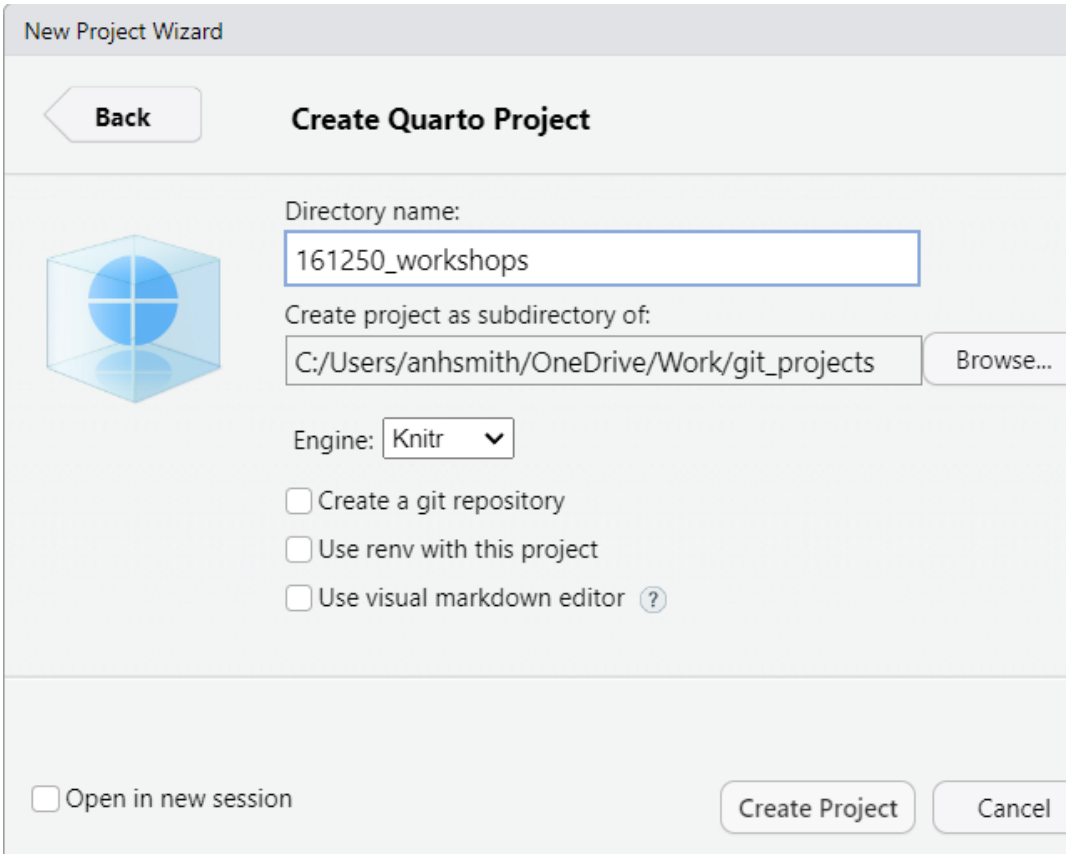
4. In the pop-up window:

- Select “New Directory”





- Select "Quarto Project"
- Choose your directory via the "Browse", and then give the project a name like

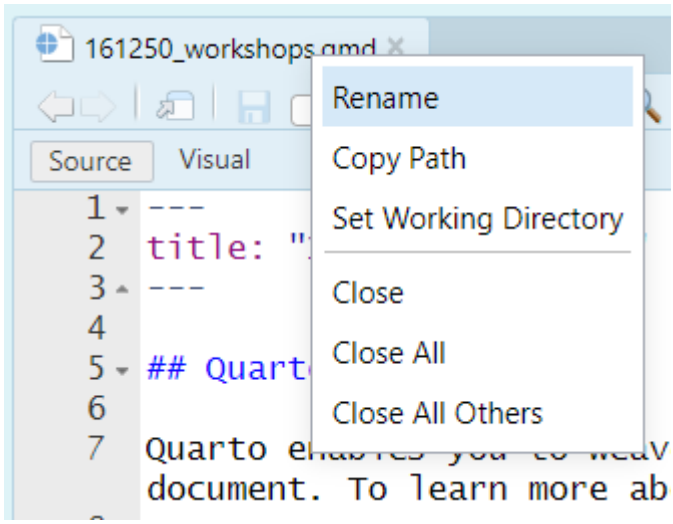


"161250_workshops"

- Finish by clicking on "Create Project".

The project should now be created, and you'll likely have an open *.qmd file (something like "161250_workshops.qmd") in the top-right window of Rstudio. We want to make a *.qmd file for this workshop.

5. Right-click on the qmd tab and select "Rename", and rename it "workshop2.qmd" or something similar. (Alternatively, just make a new file via the menus: *File > New File > Quarto Document.*)



Now you have a document for your Workshop 2 work. You can:

- Write headings with lines beginning with "#".
- Write text in the main part of the document.
- Make a code chunk for your R code using *Ctrl-Alt-i*. Write R code in the code chunks.

Like so:

````
````default

# Heading

normal text


```{r}

# this is a code chunk -- R code goes here

```




````
````

There are lots of tutorials online covering the basics of Quarto, and we'll discuss them during our own workshops. Here are a couple for starters:

https://quarto.org/docs/get-started/hello/rstudio.html

https://www.youtube.com/watch?v=c654j7aQjcg

There are many advantages of Quarto projects. One is that you can put datasets into the project folder, and they'll be easily accessible within your project, without having to worry about file paths.

You can easily open a recent past projects via the "Projects" button on the top-right of Rstudio.

Dataset `Prestige`

As you work through this workshop, you can copy the code and paste it into a code chunk. Write notes and observations to your self as you go.

We will be using a well-known dataset called `Prestige` from the `car` R package. This dataset deals with prestige ratings of Canadian occupations. The `Prestige` dataset has 102 rows and 6 columns. Each row (or 'observation') is an occupation.

This data frame contains the following columns:

- `education` - Average education of occupational incumbents, years, in 1971.
- `income` - Average income of incumbents, dollars, in 1971.
- `women` - Percentage of incumbents who are women.
- `prestige` - Pineo-Porter prestige score for occupation, from a social survey conducted in the mid-1960s.
- `census` - Canadian Census occupational code.
- `type` - Type of occupation. A factor with levels: bc, Blue Collar; prof, Professional, Managerial, and Technical; wc, White Collar. (includes four missing values).

First we'll load the data. The dataset sits in the `car` package, so you need to load the `car` package first.

```
library(car)
data(Prestige)
```

Exercise 2.1

Draw a bar chart for `type`. These plots show the count or relative frequency of blue collar (`bc`), professional (`prof`), and white collar (`wc`) professions in the dataset.

```
library(tidyverse)

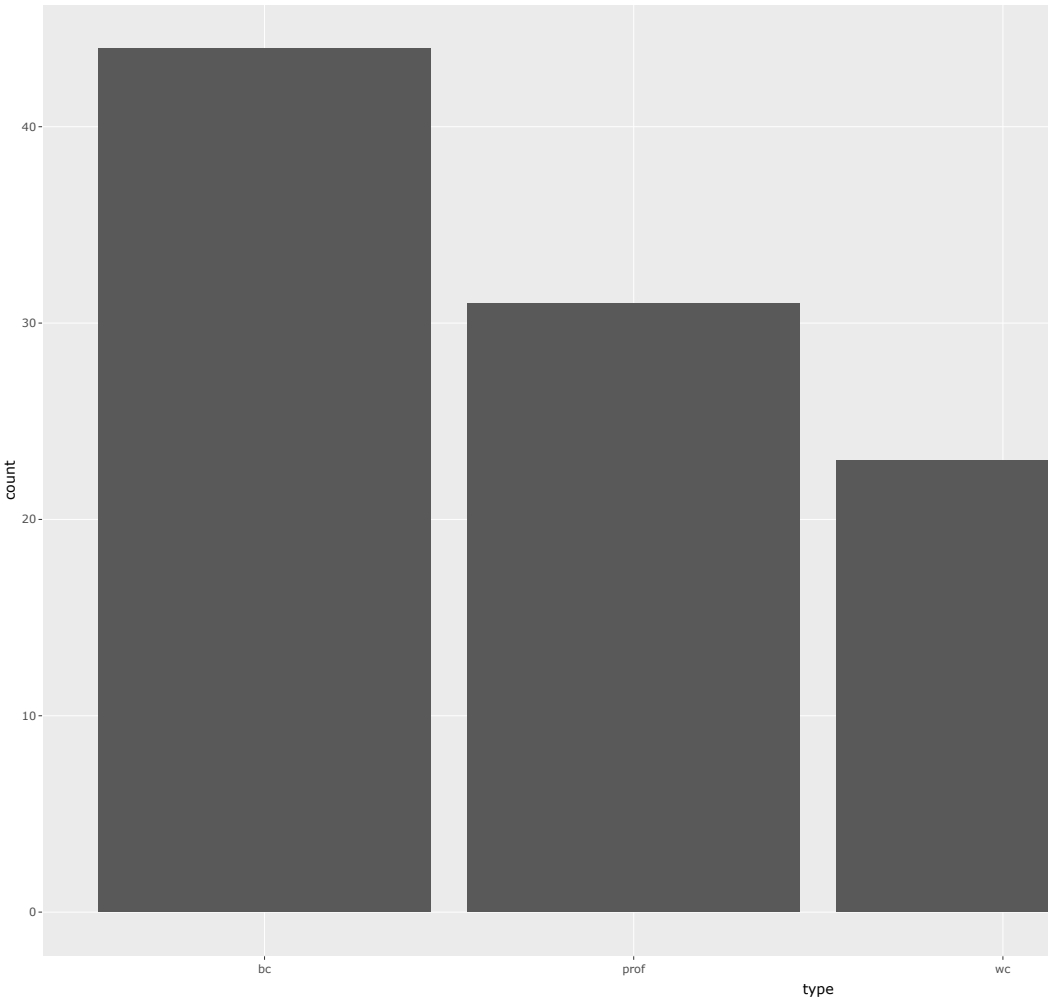
p <- Prestige |>
  ggplot() +
  aes(type) +
  geom_bar()

p
```

Or with `plotly` (which works for HTML, not for PDF)

```
library(plotly)

ggplotly(p)
```


40
30
20
10
0
count
bc
prof
wc
type

Or with old-style R plot

```
# or
library(car)
barplot(table(Prestige$type))
```

Exercise 2.2

Draw a histogram of `prestige`.

Below demonstrates the flexibility of `ggplot` code. You can specify the `data` argument by piping it into `ggplot`, or by putting it as an argument to `ggplot` or a `geom_`. Likewise, the `mapping` or `aes` information, which determines which variables are used where, can be added as an extra line or specified inside the `ggplot` or `geom_` function.

All of these methods give the same result.

```
Prestige |>
  ggplot() +
  aes(x = prestige) +
  geom_histogram(bins=10)
```

```
ggplot(
  data = Prestige,
  mapping = aes(x = prestige)
  ) +
  geom_histogram(bins=10)
```

```
ggplot(Prestige) +
  aes(x = prestige) +
  geom_histogram(bins=10)
```

```
ggplot() +
  geom_histogram(
    data = Prestige,
    mapping = aes(x = prestige),
    bins = 10
    )
```

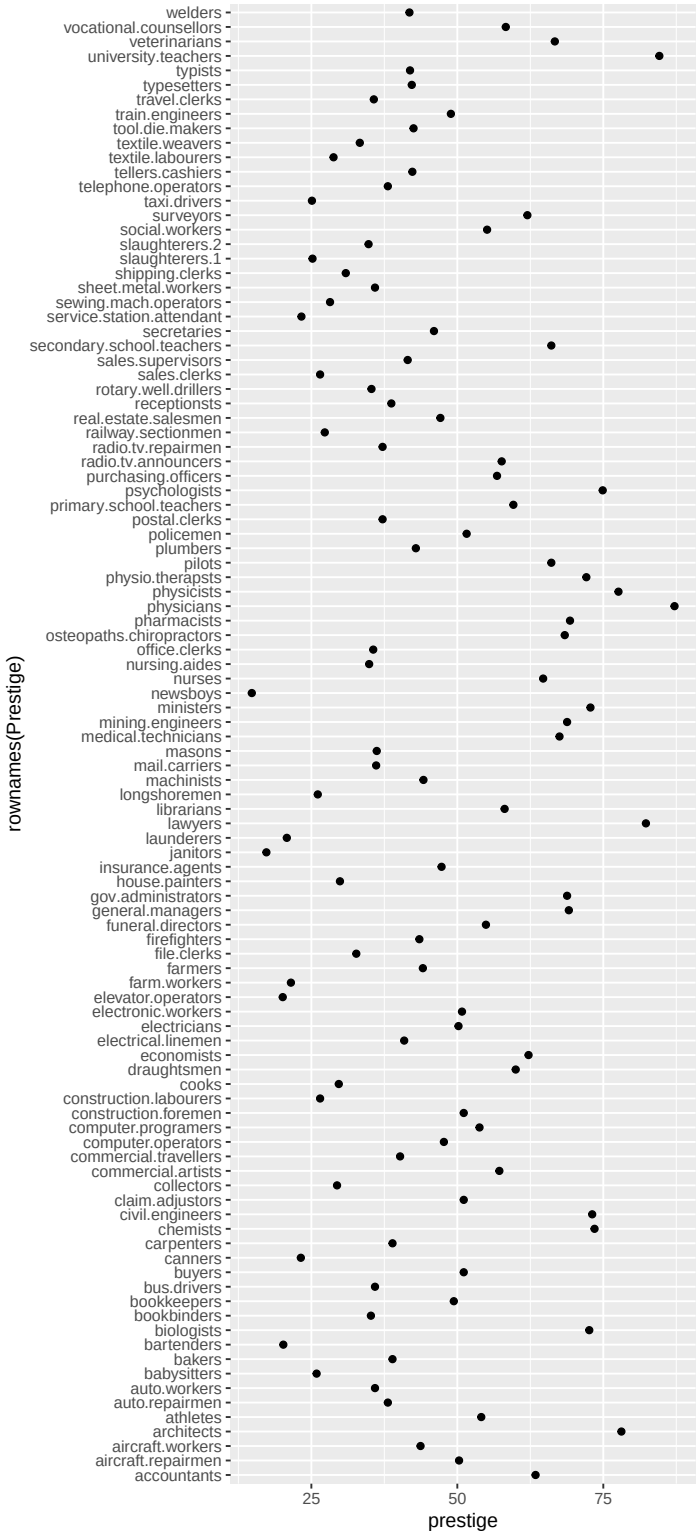
```
# or
# library(plotly)
```

```
# p <- Prestige |>
#   ggplot() +
#   aes(prestige) +
#   geom_histogram(bins=10)
#
# ggplotly(p)

# or
# hist(Prestige$prestige)
```

We can display this information as a dot plot.

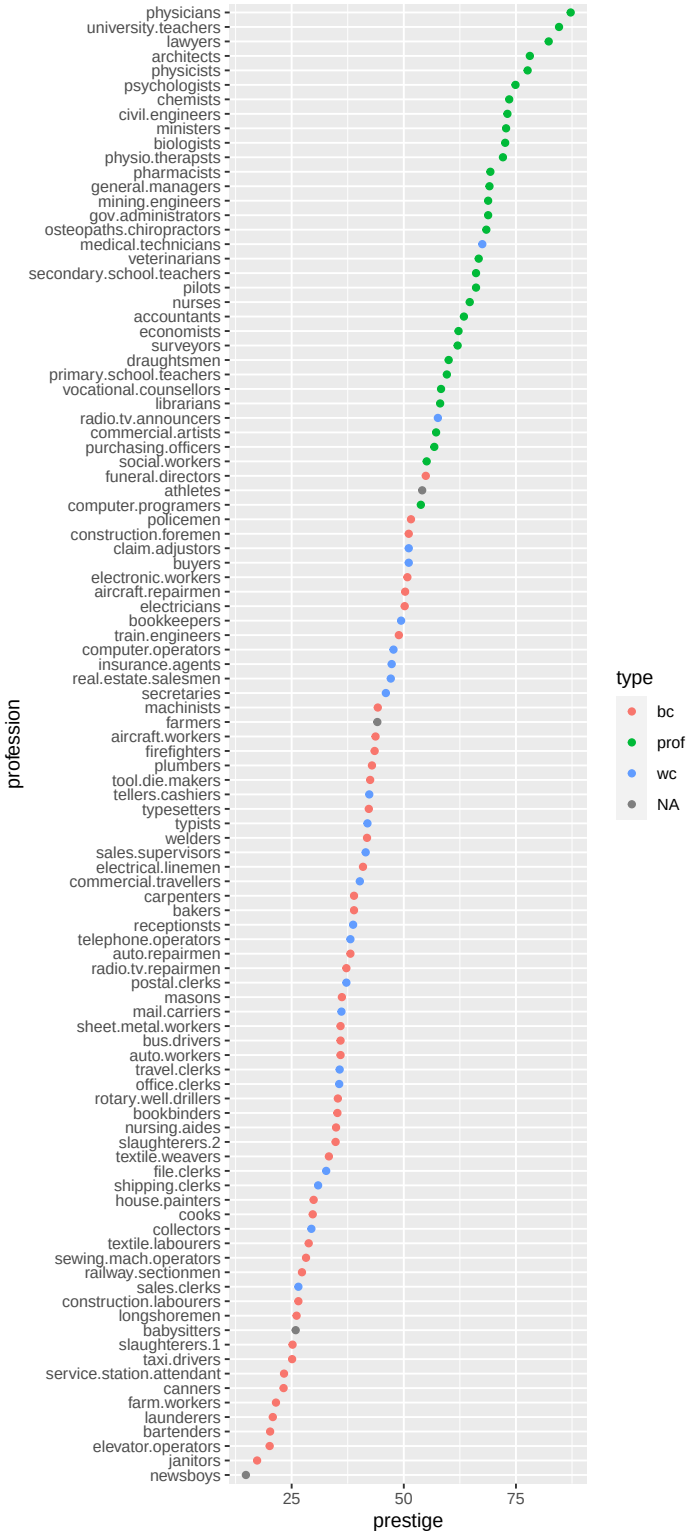
```
Prestige |>
  ggplot() +
  aes(x = rownames(Prestige), y = prestige) +
  geom_point() +
  coord_flip()
```


welders
vocational.counsellors
veterinarians
university.teachers
typists
typesetters
travel.clerks
train.engineers
tool.die.makers
textile.weavers
textile.labourers
tellers.cashiers
telephone.operators
taxi.drivers
surveyors
social.workers
slaughterers.2
slaughterers.1
shipping.clerks
sheet.metal.workers
sewing.mach.operators
service.station.attendant
secretaries
secondary.school.teachers
sales.supervisors
sales.clerks
rotary.well.drillers
receptionsts
real.estate.salesmen
railway.sectionmen
radio.tv.repairmen
radio.tv.announcers
purchasing.officers
psychologists
primary.school.teachers
postal.clerks
policemen
plumbers
pilots
physio.therapsts
physicists
physicians
pharmacists
osteopaths.chiropractors
office.clerks
nursing.aides
nurses
newsboys
ministers
mining.engineers
medical.technicians
masons
mail.carriers
machinists
longshoremen
librarians
lawyers
launderers
janitors
insurance.agents
house.painters
gov.administrators
general.managers
funeral.directors
firefighters
file.clerks
farmers
farm.workers
elevator.operators
electronic.workers
electricians
electrical.linemen
economists
draughtsmen
cooks
construction.labourers
construction.foremen
computer.programers
computer.operators
commercial.travellers
commercial.artists
collectors
claim.adjustors
civil.engineers
chemists
carpenters
canners
buyers
bus.drivers
bookkeepers
bookbinders
biologists
bartenders
bakers
babysitters
auto.workers
auto.repairmen
athletes
architects
aircraft.workers
aircraft.repairmen
accountants
rownames(Prestige)
25
50
75
prestige

What a mess!

We can tidy it up by ordering the professions on the plot according to `prestige`. First, we move the professions from rownames to a variable. Then, we `fct_reorder` the professions using `prestige`. Then, the resulting data gets piped into `ggplot`.

```
Prestige |>
  rownames_to_column(var = "profession") |>
  mutate(
    profession = fct_reorder(profession, prestige)
    ) |>
  ggplot() +
  aes(x = profession, y = prestige, colour = type) +
  geom_point() +
  coord_flip()
```


physicians
university.teachers
lawyers
architects
physicists
psychologists
chemists
civil.engineers
ministers
biologists
physio.therapsts
pharmacists
general.managers
mining.engineers
gov.administrators
osteopaths.chiropractors
medical.technicians
veterinarians
secondary.school.teachers
pilots
nurses
accountants
economists
surveyors
draughtsmen
primary.school.teachers
vocational.counsellors
librarians
radio.tv.announcers
commercial.artists
purchasing.officers
social.workers
funeral.directors
athletes
computer.programers
policemen
construction.foremen
claim.adjustors
buyers
electronic.workers
aircraft.repairmen
electricians
bookkeepers
train.engineers
computer.operators
insurance.agents
real.estate.salesmen
secretaries
machinists
farmers
aircraft.workers
firefighters
plumbers
tool.die.makers
tellers.cashiers
typesetters
typists
welders
sales.supervisors
electrical.linemen
commercial.travellers
carpenters
bakers
receptionsts
telephone.operators
auto.repairmen
radio.tv.repairmen
postal.clerks
masons
mail.carriers
sheet.metal.workers
bus.drivers
auto.workers
travel.clerks
office.clerks
rotary.well.drillers
bookbinders
nursing.aides
slaughterers.2
textile.weavers
file.clerks
shipping.clerks
house.painters
cooks
collectors
textile.labourers
sewing.mach.operators
railway.sectionmen
sales.clerks
construction.labourers
longshoremen
babysitters
slaughterers.1
taxi.drivers
service.station.attendant
canners
farm.workers
launderers
bartenders
elevator.operators
janitors
newsboys
profession
25
50
75
prestige
type
bc
prof
wc
NA

Exercise 2.3

Obtain some summary statistics for `prestige`. There are a few options for this.

```
summary(Prestige)

library(psych)

describe(Prestige)

describeBy(education + income + women + prestige ~ type,
           data = Prestige)
```

Exercise 2.4

Make a boxplot of `prestige ~ type`:

```
Prestige |>
  ggplot() +
  aes(y=prestige, x=type) +
  geom_boxplot()
```

```
# or
# library(plotly)
# p <- Prestige |> ggplot() +
#   aes(y=prestige, x=type) + geom_boxplot()
# ggplotly(p)

# or
# library(lattice)
# bwplot(prestige ~ type, data=Prestige)

# as violin plots
Prestige |>
  ggplot() +
  aes(y=prestige, x=type) +
  geom_violin()
```

```
# Or put it all together
Prestige |>
  ggplot() +
  aes(y=prestige, x=type) +
  geom_violin() +
  geom_boxplot(col = 2, alpha = .2) +
  geom_jitter(alpha = .2, width = .2, height = 0, colour = 4)
```

Exercise 2.5

Obtain the Empirical Cumulative Distribution Function (ECDF) graphs of `prestige ~ type`:

```
Prestige |>
  ggplot() +
  aes(prestige, colour=type) +
  stat_ecdf()
```

```
Prestige |>
  ggplot() +
  aes(prestige) +
  stat_ecdf() +
  facet_wrap(~type)
```

```
# or
library(latticeExtra)
ecdfplot(~ prestige | type, data = Prestige)
```

```
Prestige |>
  ggplot() +
  aes(
    x = prestige, # these aes settings are used
    col = type    # by both geoms
    ) +
  geom_density(
    aes(fill = type), # the 'fill' aes goes here because
    alpha = .2        # geom_rug doesn't use 'fill'
    ) +
  geom_rug()
```

Exercise 2.6

Obtain the {0.05, 0.1, 0.25, 0.5, 0.75, 0.9, 0.95} quantiles of `prestige`:

```
pr <- c(0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 0.9, 0.95, 0.99)

Prestige |>
  summarise(
    probs = pr,
    quants = quantile(prestige, pr)
    )

# or simply
quantile(Prestige$prestige, pr)
```

Exercise 2.7

Obtain the scatter plot (with and without marginal boxplots) **prestige vs. education** :

```
library(ggExtra)

p1 <- Prestige |>
  ggplot() +
  aes(x = education, y = prestige) +
  geom_point() +
  geom_smooth(col = 2) +
  geom_smooth(method = "lm", se = FALSE)

ggMarginal(p1, type="boxplot")
```

```
library(car)

scatterplot(education ~ prestige, data = Prestige)
```

The later plot will show prediction interval ribbon while the first plot will show the confidence interval ribbon.

Exercise 2.8

Obtain the bubble or balloon plot **prestige vs. education vs. income** (income forming the bubble size):

```
library(ggplot2)

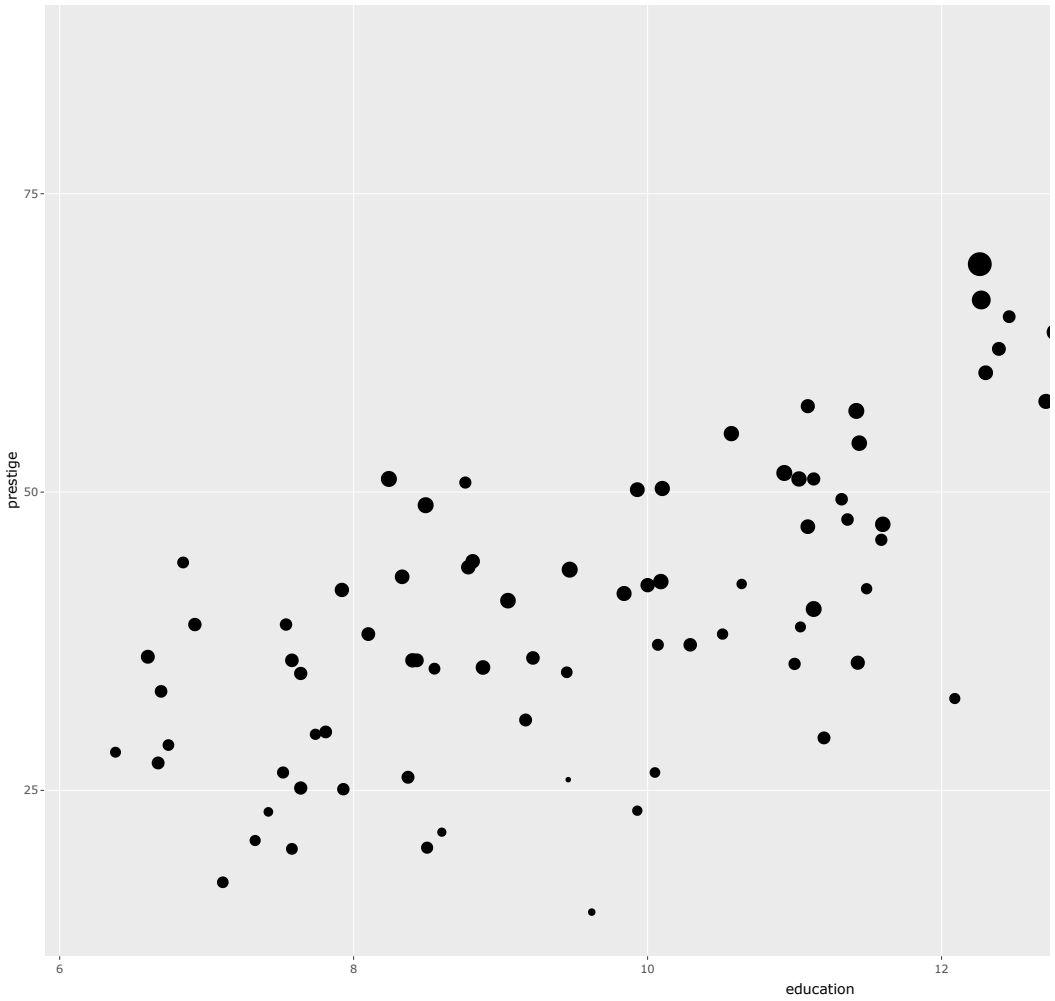
Prestige |>
  ggplot() +
  aes(x = education, y = prestige, size = income) +
  geom_point()
```

```
# or

library(plotly)

p <- Prestige |>
  ggplot() +
  aes(x = education, y = prestige, size = income) +
  geom_point()

ggplotly(p)
```


75
50
25
prestige
6
8
10
12
education

Exercise 2.9

Obtain the contour plot **prestige vs. education vs. income** :

```
library(plotly)

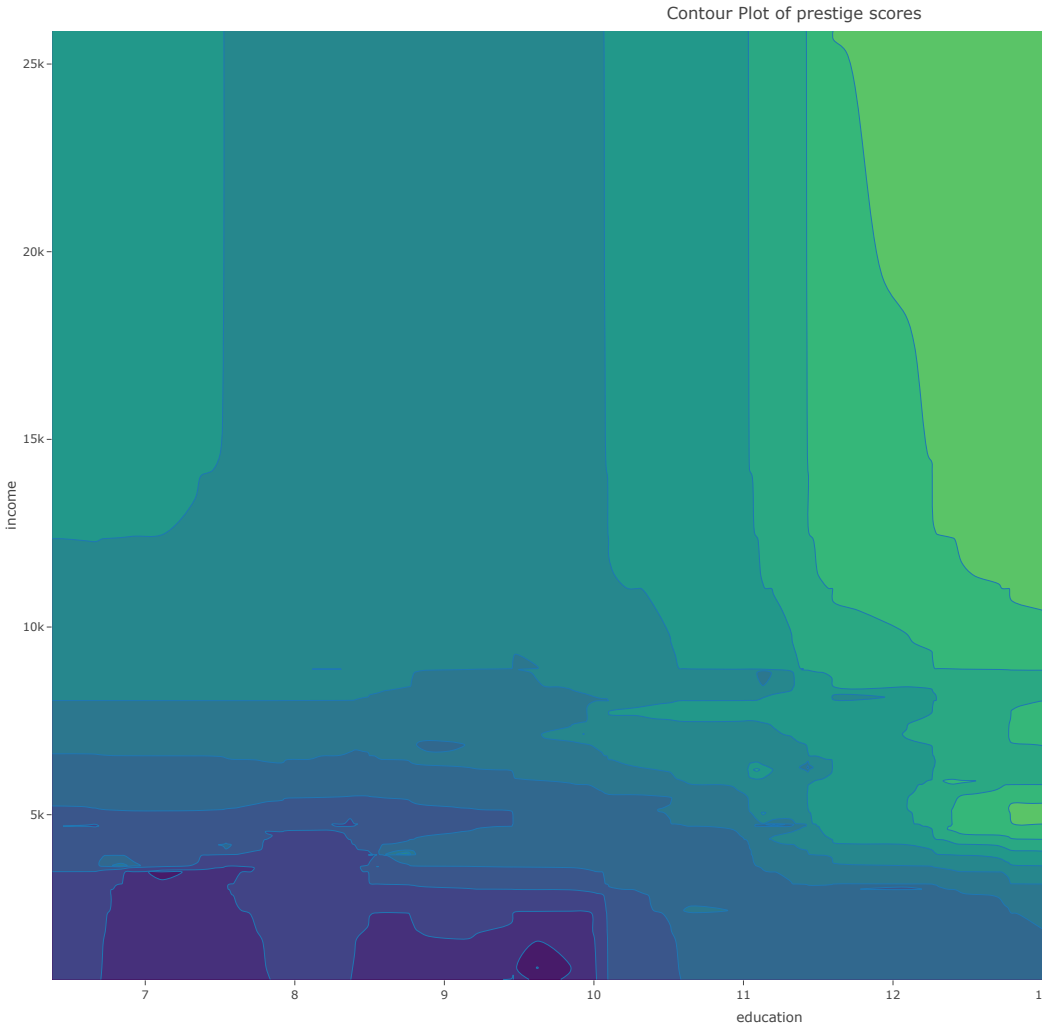
plot_ly(type = 'contour',
        x = Prestige$education,
        y = Prestige$income,
        z = Prestige$prestige)
```


25k
20k
15k
10k
5k
7
8
9
10
11
12
1

To add axes labels and titles, try-

```
library(plotly)

plot_ly(
  Prestige,
  type = 'contour',
  x = Prestige$education,
  y = Prestige$income,
  z = Prestige$prestige
) |> layout(
  title = 'Contour Plot of prestige scores',
  xaxis = list(title = 'education'),
  yaxis = list(title = 'income')
)
```


Contour Plot of prestige scores
25k
20k
15k
10k
5k
income
7
8
9
10
11
12
1
education

We can also define our own function for the contour approximation.

```
library(modelr)

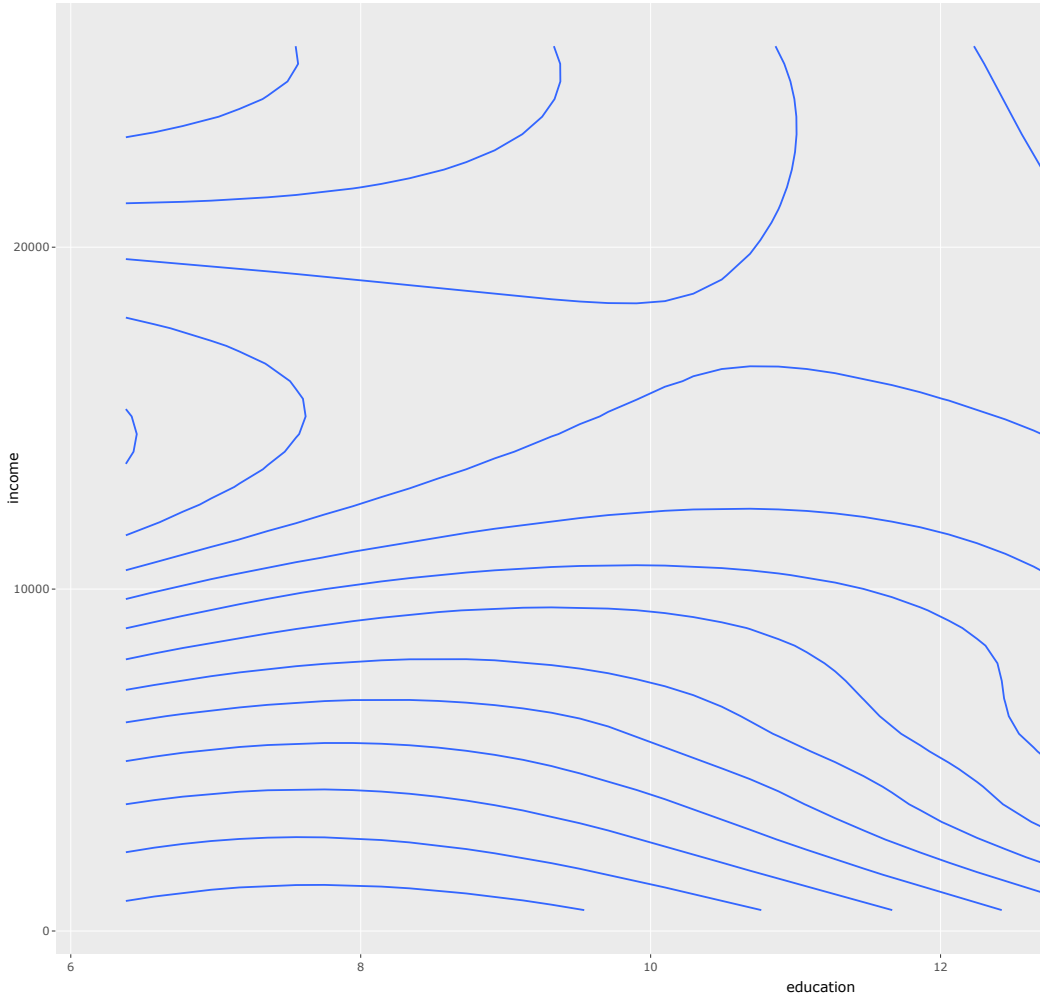
# make a smooth model
y.m = loess(prestige ~ education * income, data = Prestige)

# make a regular grid of all combinations of education and income
mygrid <- Prestige |>
  data_grid(
    education = seq_range(education, 50),
    income = seq_range(income, 50)
  ) |>
  # add predicted prestige using the smooth model
  add_predictions(y.m, var = "predicted prestige")

# make ggplot contour plot
p <- mygrid |>
  ggplot() +
  aes(x = education, y = income, z = `predicted prestige`) +
  geom_contour()

p
```

```
# make a plotly version
library(plotly)
ggplotly(p)
```


20000
10000
0
income
6
8
10
12
education

```r
# filled contour ggplot
mygrid |>
  ggplot() +
  aes(x=education, y=income, z=`predicted prestige`) +
  stat_contour_filled()
```

```r
# or the older-style lattice graphs

library(lattice)

contourplot(`predicted prestige` ~ education * income,
            data = mygrid,
            cuts = 10, region = TRUE,
            xlab = "education ", ylab = "income ")
```

```r
wireframe(`predicted prestige` ~ education * income,
          data = mygrid,
          cuts = 10, region = TRUE,
          xlab = "education ", ylab = "income ")
```

```r
levelplot(`predicted prestige` ~ education * income,
          data = mygrid,
          cuts = 10, region = TRUE,
          xlab = "education ", ylab = "income ")
```

```r
cloud(`predicted prestige` ~ income * education,
      data = mygrid)
```

Exercise 2.10

Obtain the 3-D plot **prestige vs. education vs. income** :

```
library(car)

scatter3d(prestige ~ education + income,
          data = Prestige)
```

Exercise 2.11

Create `prestige ~ education | type` graphs. That is, `prestige ~ education` grouped by `type` as colours and/or panels.

```
Prestige |>
  ggplot() +
  aes(x = education, y = prestige, color = type) +
  geom_point() +
  facet_wrap(~ type)
```

```
# or
# library(plotly)
#
# p <- Prestige |>
#   ggplot() +
#   aes(x = education, y = prestige, color = type) +
#   geom_point() +
#   facet_wrap(~ type)
#
# ggplotly(p)
```

```
p <- Prestige |>
  ggplot() +
  aes(x = education, y = prestige, color = type) +
  geom_point()

p
```

```
# OR
#
# library(plotly)
# ggplotly(p)
```

```
scatterplot(prestige ~ education | type,
            data=Prestige)
```

```
xyplot(prestige ~ education | type,
       auto.key = TRUE,
       data = Prestige)
```

```
xyplot(prestige ~ education,
       group = type,
       auto.key = TRUE,
       data = Prestige)
```

Exercise 2.12

Time-series data EDA based on RBNZ house sales data.

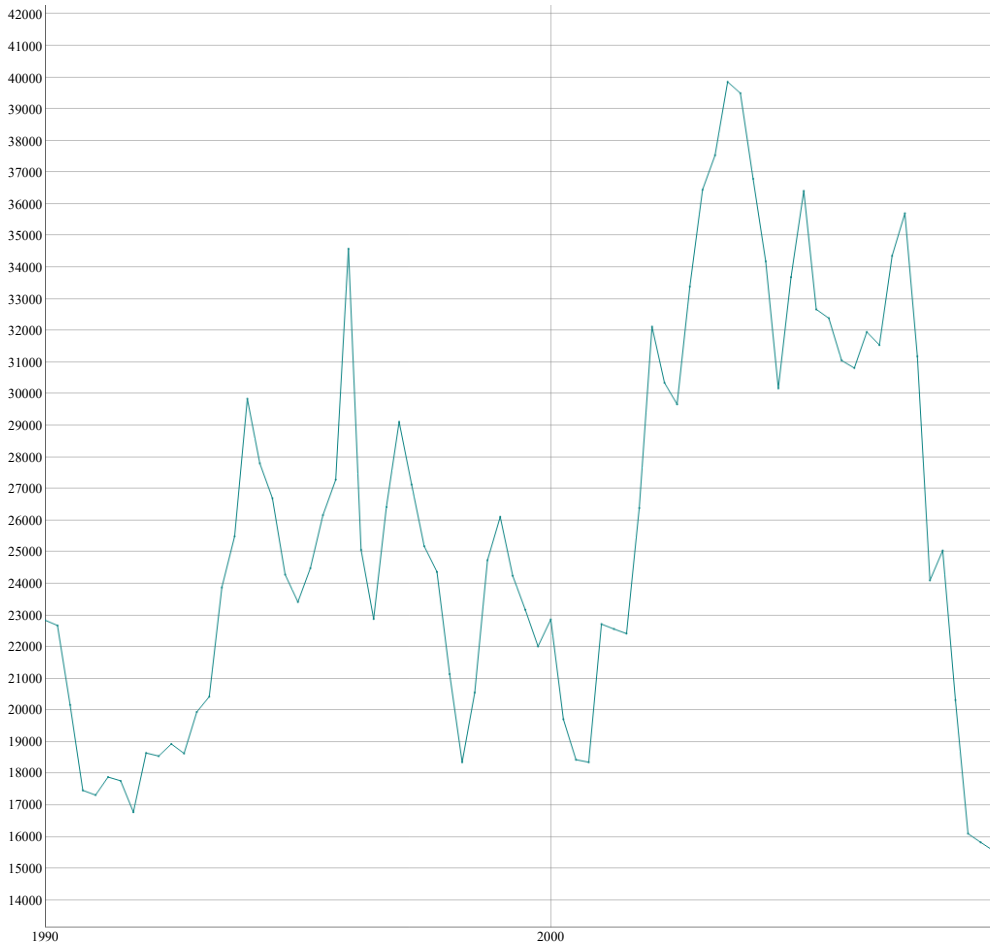
```
url1 <- "https://www.massey.ac.nz/~anhsmith/data/housesales.RData"
download.file(url = url1, destfile = "housesales.RData")
load("housesales.RData")
```

```
library(forecast)
autoplot(housesales)
```

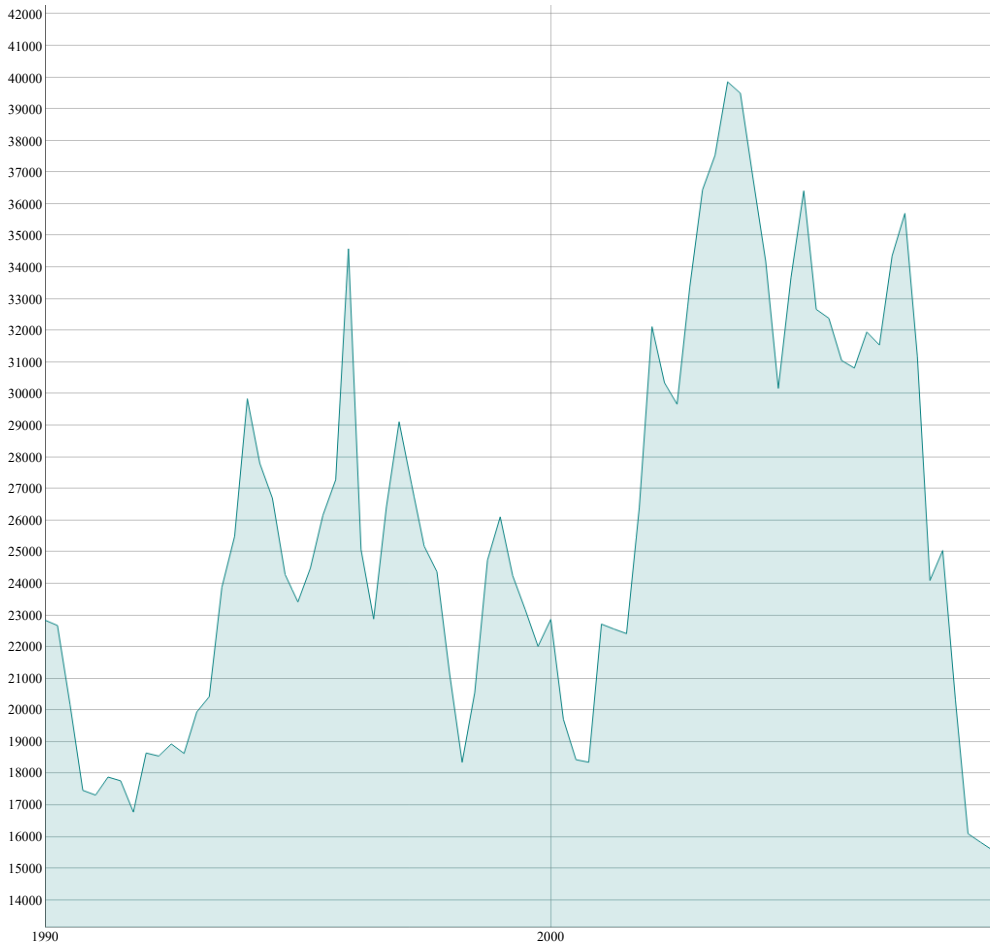
```
# Dynamic graphing
# https://rstudio.github.io/dygraphs/index.html

library(dygraphs)

dygraph(housesales) |>
  dyOptions(drawPoints = TRUE)
```


42000
41000
40000
39000
38000
37000
36000
35000
34000
33000
32000
31000
30000
29000
28000
27000
26000
25000
24000
23000
22000
21000
20000
19000
18000
17000
16000
15000
14000
1990
2000

```
dygraph(housesales) |>
  dyOptions(fillGraph=TRUE)
```


42000
41000
40000
39000
38000
37000
36000
35000
34000
33000
32000
31000
30000
29000
28000
27000
26000
25000
24000
23000
22000
21000
20000
19000
18000
17000
16000
15000
14000
1990
2000

```
ggseasonplot(housesales)
```

```
ggsubseriesplot(housesales)
```

Series Decomposition

```
library(tidyverse)

housesales |>
  decompose(type="additive")  |>
  forecast::autoplot() +
  ggtitle("")
```

```
housesales |>
  decompose(type="multiplicative") |>
  forecast::autoplot() +
  ggtitle("")
```

lag & ACF plots

```
gglagplot(housesales)
```

```
gglagplot(housesales, seasonal=FALSE, lag=1)
```

```
ggAcf(housesales)
```

```
ggPacf(housesales)
```

```
ggtsdisplay(housesales)
```

More graphing examples are here (R code file).