Investigating risk of suicide in patients with cancer using routine data

Notes on Stata/R/Python

Justin C Yang

justin.yang@ucl.ac.uk

University College London

Datasets

- Cancer registry data (linked to civil mortality registers or otherwise including data on suicide mortality) with sex and age
  - e.g. The Simulacrum, a synthetic cancer dataset
- Civil mortality registers for the general population with sex and age
  - e.g. Registered suicide deaths in England

Variables

- Required
  - Age (at cancer diagnosis, during follow-up, at death)
  - Sex
  - Cancer type
  - Date/year of cancer diagnosis
  - Date/year of suicide death
  - Date/year of loss to follow-up
- Confounders
  - Deprivation
  - Ethnicity
  - Tumour grade
  - Treatment type
  - Comorbidities
  - Others

Computations

Standardised Mortality Ratio (SMR)

The SMR compares rates of suicide deaths among cancer patients and rates of suicide deaths among the general population, standardised for age, sex, and time period.

$$\text{SMR} = \frac{\text{observed number of suicides}}{\text{expected number of suicides}}$$

Absolute Excess Risk (AER)

The AER or attributable risk is the difference between two absolute risks over a specific time period. In this case, it is the difference between observed suicides and expected suicides among patients with cancer.

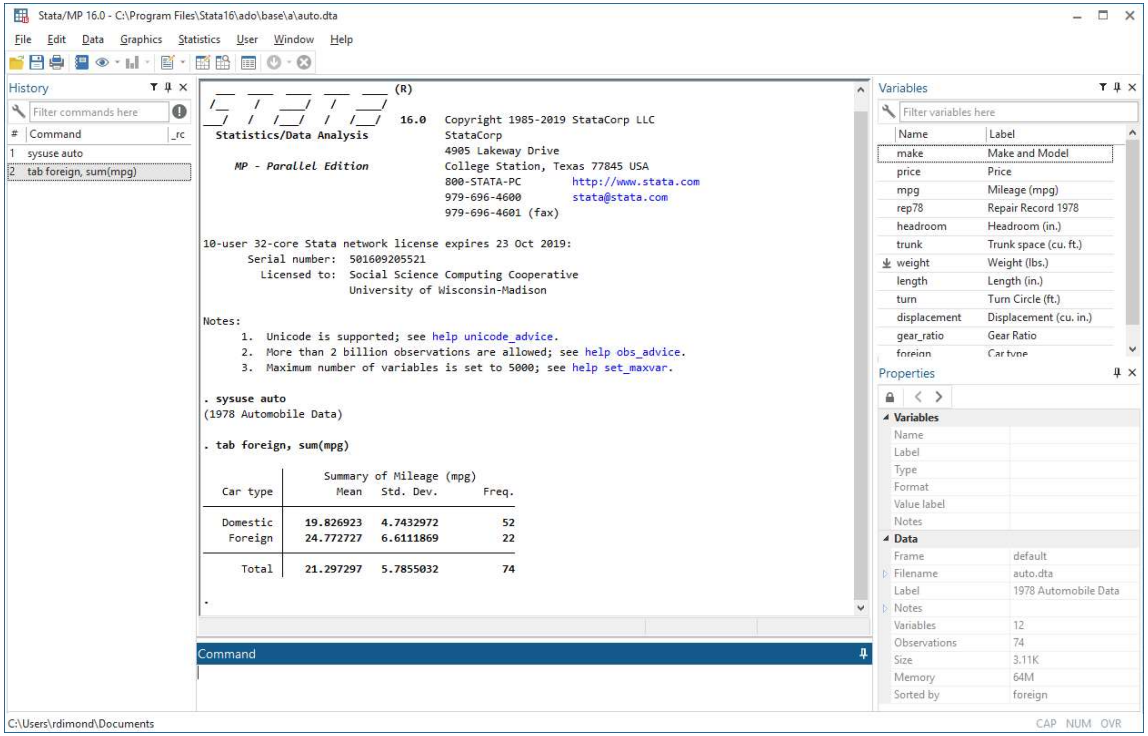
$$\text{AER} = \frac{\text{observed number of suicides} - \text{expected number of suicides}}{\text{person-years at risk}}$$

Deriving SMR

1. Calculate the number of suicide deaths among cancer patients by age group and by sex for each cancer group. These are the observed number of suicides.
2. Using age- and sex-standardised suicide rates in the general population, derive the expected number of suicide deaths among cancer patients for each cancer group, multiplying by the number of person-years at risk.
3. Express the cancer-specific SMRs as a fraction of the observed number of suicides to the expected number of suicides.

Software Options: Stata

- Commercial statistical software package widely used in econometrics and epidemiology with many statistical functions
- Lower barrier to use with a graphical user interface and programmable automation
- Extensible using user-written programs
- Historically, slower to update features based on version updates but continuous release option is available

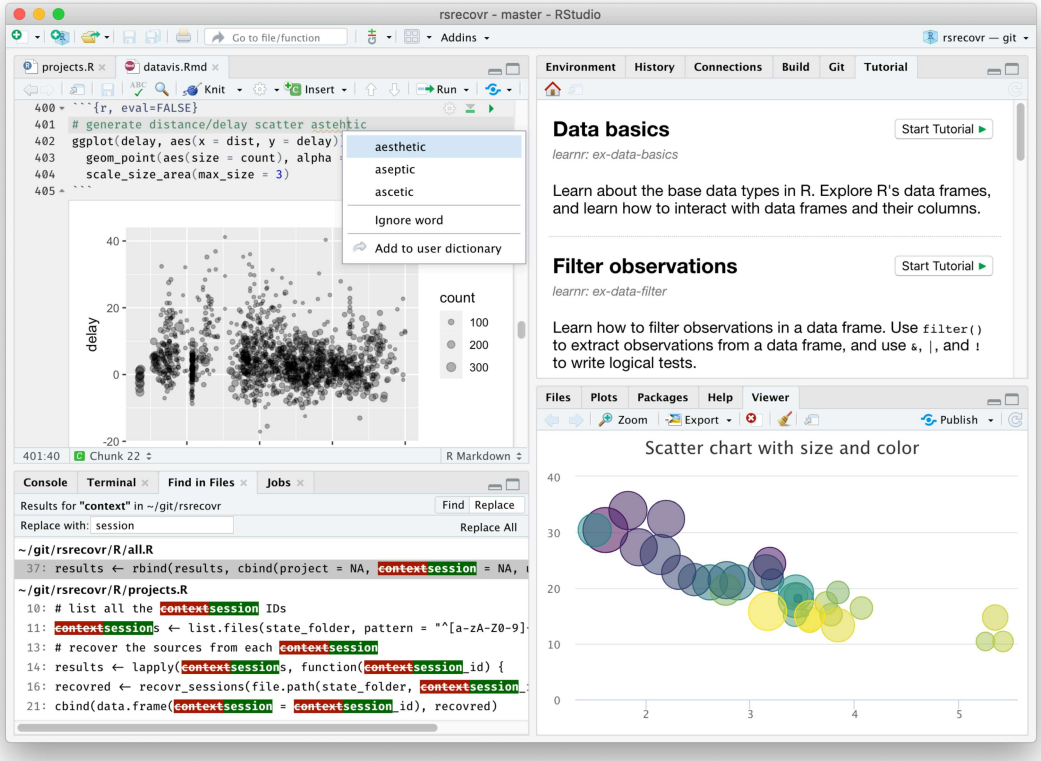


Example: Stata

```
* Calculate expected numbers of deaths using population rates

    gen E_suicide = (_t-_t0) * (poprate / 100000)

*--OBSERVED NO
            gen obstr = string(_dsuicide) + " / " + string(E_suicic

            gen smr     = (_dsuicide/E_suicide)
            gen smrll       = ((invgammap( _dsuicide,     (0.05)/2)
            gen smrul       = ((invgammap((_dsuicide+ 1), (1.95)/2)
            gen str smrstr = string(smr , "%9.1f") + " (" + string(

            gen aer     = cond(((_dsuicide- E_suicide)/pyrs)>0 , ((
            gen aerll       = aer - (1.96*(sqrt(_dsuicide)/pyrs))
            gen aerul       = aer + (1.96*(sqrt(_dsuicide)/pyrs))
            gen str aerstr = string(aer , "%9.1f") + " (" + string(

            sort `v'
            decode `v', gen(strdiag)
```

Software Options: R

- Open-source interpreted programming language for statistical computing and data visualisation characterised by a very large number of extension packages (20,752!)
- Learning syntax can be harder but there are many resources to learn different syntactic paradigms (e.g. base R, Tidyverse, data.table, Bioconductor)
- Very good community support and integrates new statistical methods quickly, even for domain-specific functions

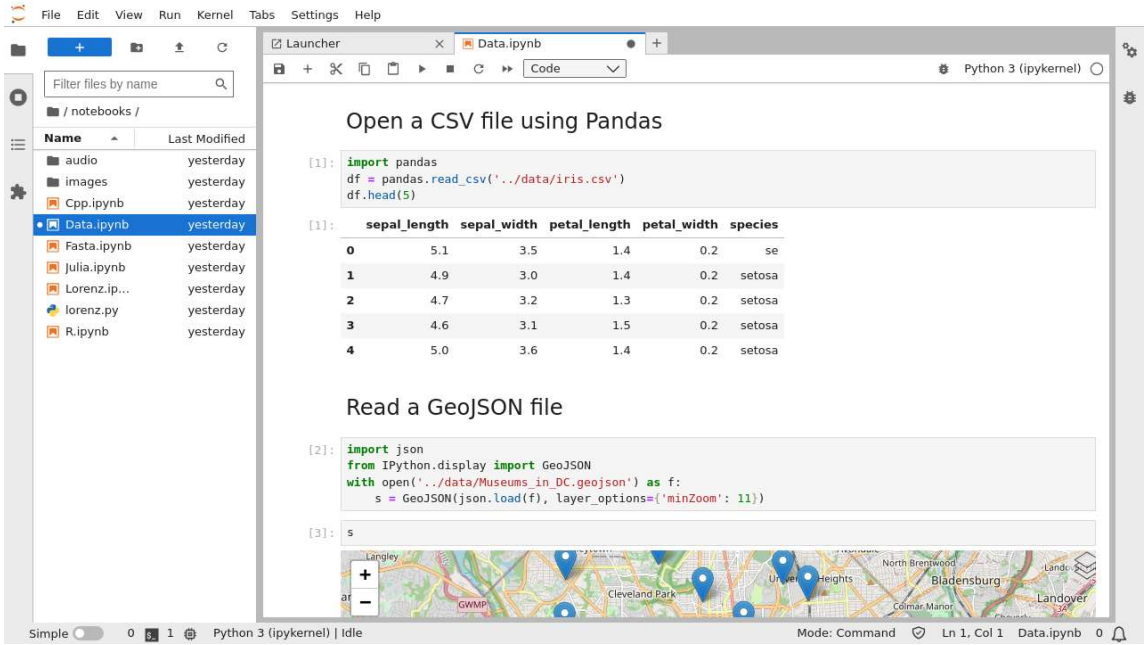


Example: R

```r
# Calculate SMRs
cancer_smr <- cancer_suicides |>
  left_join(reference_suicides) |>
  mutate(expected_suicides = (person_years / 10000) * suicides_per_
  group_by(cancer_group) |>
  summarise (
    observed_suicides = sum(suicide_deaths),
    expected_suicides = round(sum(expected_suicides)),
    person_years = sum(person_years)
  ) |>
  mutate(
    smr = round(observed_suicides / expected_suicides, digits = 2),
    aer = round((observed_suicides - expected_suicides) / person_ye
                digits = 2
    )
  ) |>
  select(cancer_group,
         person_years,
         observed_suicides
```

Software Options: Python

- High-level, general purpose object-oriented programming language commonly used in data science, particularly deep learning and machine learning
- Harder to learn as a full programming language but has mature features such as unit testing and debugging
- Good packages for major data science functions though not all statistical methods are available
- As a general purpose language, strong at handling large amounts of data and performing non-statistical tasks, such as natural language processing



Example: Python

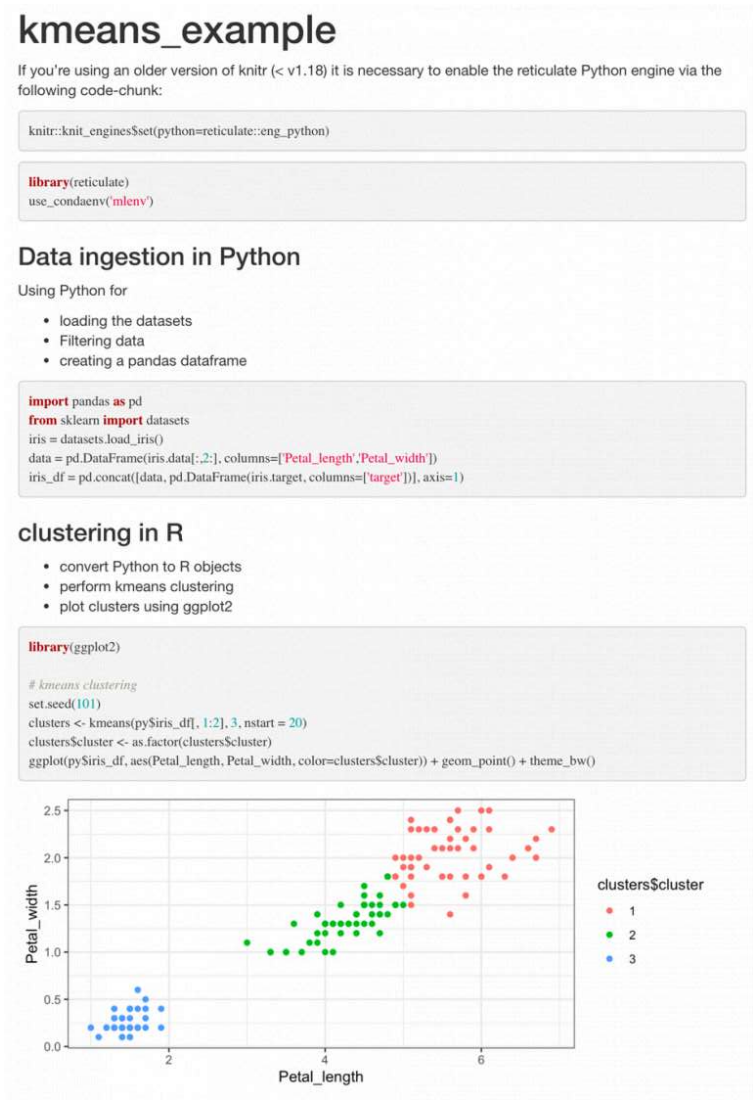
```python
# Calculate suicides among cancer patients
df_survival['attained_age'] = df_survival['age_category_at_death']
df_survival['sex'] = df_survival['gender']
df_survival['year'] = pd.to_datetime(df_survival['vital_status_date
df_survival['suicide'] = df_survival['suicide'] - 1

# Group by and summarize
grouped = df_survival.groupby(['year', 'sex', 'attained_age', 'canc
cancer_suicides = grouped.agg(
    person_years=pd.NamedAgg(column='time', aggfunc=lambda x: round
    suicide_deaths=pd.NamedAgg(column='suicide', aggfunc='sum')
).reset_index()
cancer_suicides.dropna(inplace=True)
cancer_suicides = cancer_suicides.drop_duplicates()
cancer_suicides['sex'] = cancer_suicides['sex'].astype(str)
cancer_suicides['attained_age'] = cancer_suicides['attained_age'].a

# Prepare for merging by ensuring alignment of the key columns used
```

Software Options: Summary

Combining Software

While most analysts become fluent in one or more statistical software packages, analyses are often conducted using a single software option. However, several packages exist to allow calling upon different statistical software in the same document:

- Use Stata with R using the RStata package
- Use Python with R using the reticulate package
- Use R with Stata (and vice versa) using the RCall package
- Use Python with Stata (and vice versa) using the PyStata package



kmeans_example

If you're using an older version of knitr (< v1.18) it is necessary to enable the reticulate Python engine via the following code-chunk:

```
knitr::knit_engines$set(python=reticulate::eng_python)
```

```
library(reticulate)
use_condaenv('mlenv')
```

Data ingestion in Python

Using Python for

- loading the datasets
- Filtering data
- creating a pandas dataframe

```
import pandas as pd
from sklearn import datasets
iris = datasets.load_iris()
data = pd.DataFrame(iris.data[:,2:], columns=['Petal_length','Petal_width'])
iris_df = pd.concat([data, pd.DataFrame(iris.target, columns=['target'])], axis=1)
```

clustering in R

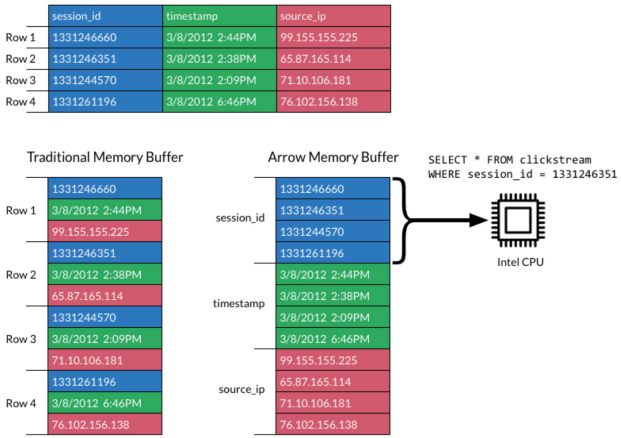
- convert Python to R objects
- perform kmeans clustering
- plot clusters using ggplot2

```
library(ggplot2)

# kmeans clustering
set.seed(101)
clusters <- kmeans(py$iris_df[, 1:2], 3, nstart = 20)
clusters$cluster <- as.factor(clusters$cluster)
ggplot(py$iris_df, aes(Petal_length, Petal_width, color=clusters$cluster)) + geom_point() + theme_bw()
```

Notes on Low-Resource Computing

Dealing with large datasets can be challenging, especially datasets which may be larger-than-memory. General advice for dealing with includes:

- Loading only the minimum amount of variables required to run a model
- Parallelising code wherever possible to take advantage of multi-core computing
- Using columnar memory formats such as Apache Arrow in R, Python, or Julia
- Using packages, Python and R can work directly with databases, including use of SQL; depending on version, Stata can handle large numbers of variables and observations, depending on edition



Final Remarks

- Each software options has its own advantages and disadvantages and each analyst brings their own experiences and familiarity with different software
- R and Python are less user-friendly but are free and have powerful extension packages; Stata is more user-friendly but has an associated cost/subscription
- Stata will be most familiar for econometricians; Python will be most familiar for machine learning and data science; and R is used across a wide variety of subject domains
- You can mix and match software as needed if you know multiple languages (e.g. Reticulate allows you to run Python from R while RCall allows you to call R from within Stata)

Links

- Henson et al. Code
  - Stata
  - R and Python and Github repository
- Tutorials
  - Introduction to computational causal inference using reproducible Stata, R, and Python code
  - Principal Component Analysis and Regression
- Reference
  - Software Carpentry
  - Data Carpentry
  - UCLA OARC Stats