Python to R practice

about the data

the data we generated is called `data.csv`. it has the following columns:

- `id`: unique identifier
- `zip_code`: seattle area ZIP code
- `age`: age in years
- `smoking`: binary variable (0 = does not smoke, 1 = smokes)
- `years_smoked`: number of years smoked (NA if non-smoker)

python code to translate

```
import pandas as pd
import numpy as np
import scipy.stats as stats

df = pd.read_csv("data.csv")

# loop over columns and print out means for age and smoking
for column in list(df.columns.values):
    if column in ['age', 'smoking']:
        colmean = np.mean(df[column])
        print("{col} mean is {mean}\n".format(col=column, mean=colmean))
```

age mean is 62.797

smoking mean is 0.484

```
# rename a variable
df.rename(columns={'age': 'age_in_years'}, inplace=True)

# subset based on a value
df = df[df['age_in_years'] > 50]

# create a new variable by adding or multiplying two variables
df['age_first_smoked'] = df['age_in_years'] - df['years_smoked']

# compute a summary statistic (chi-square test in this example)
crosstab = pd.crosstab(df['age_in_years'], df['age_first_smoked'])
chi2_stat, p_val, dof, expected = stats.chi2_contingency(crosstab)

print(f"chi-square test statistic: {chi2_stat}")
```

chi-square test statistic: 3552.287468281278

```
print(f"P-value: {p_val}")
```

P-value: 4.787214107528798e-33

```
print(f"degrees of freedom: {dof}")
```

degrees of freedom: 2597

python output preview

the first 5 rows of the output from the python code:

```
head = df.head()
print(head.to_markdown())
```

	id	zip_code	age_in_years	smoking	years_smoked	age_first_smoked
0	1	98103	55	0	nan	nan
1	2	98104	75	1	9	66
3	4	98122	91	0	nan	nan
4	5	98106	66	1	9	57
5	6	98178	74	0	nan	nan

your turn: write the r translation

write a code chunk below that does the same thing as the python code above. your code should:

1. read in the data
2. loop over columns and print means for `age` and `smoking`
3. rename `age` to `age_in_years`
4. subset to only include rows where `age_in_years > 50`
5. create a new variable `age_first_smoked` = `age_in_years - years_smoked`
6. run a chi-square test and print the results

appendix: all code

```r
# Setup
knitr::opts_chunk$set(results = 'hide', echo = FALSE, fig.keep = 'none',
                      warning = FALSE, message = FALSE)

library(pacman)
pacman::p_load(tidyverse, reticulate, knitr, kableExtra)
use_condaenv("tutorial")
set.seed(123)
n <- 1000

ids <- 1:n
zip_codes <- c(
  "98101", "98102", "98103", "98104", "98105", "98106", "98107", "98108", "98109", "98112",
  "98115", "98116", "98117", "98118", "98119", "98121", "98122", "98125", "98126", "98133",
  "98134", "98136", "98144", "98146", "98154", "98164", "98174", "98177", "98178", "98195",
  "98199"
)
ages <- sample(25:100, n, replace = TRUE)
smoking <- sample(0:1, n, replace = TRUE)
years_smoked <- sample(0:10, n, replace = TRUE)

data <- data.frame(
  id = ids,
  zip_code = sample(zip_codes, n, replace = TRUE),
  age = ages,
  smoking = smoking,
  years_smoked = ifelse(smoking == 1, sample(0:10, n, replace = TRUE), NA)
)

write.csv(data, "data.csv", row.names = FALSE)
```

```python
import pandas as pd
import numpy as np
import scipy.stats as stats

df = pd.read_csv("data.csv")

# loop over columns and print out means for age and smoking
for column in list(df.columns.values):
    if column in ['age', 'smoking']:
        colmean = np.mean(df[column])
        print("{col} mean is {mean}\n".format(col=column, mean=colmean))

# rename a variable
df.rename(columns={'age': 'age_in_years'}, inplace=True)

# subset based on a value
df = df[df['age_in_years'] > 50]

# create a new variable by adding or multiplying two variables
df['age_first_smoked'] = df['age_in_years'] - df['years_smoked']

# compute a summary statistic (chi-square test in this example)
```

```
crosstab = pd.crosstab(df['age_in_years'], df['age_first_smoked'])
chi2_stat, p_val, dof, expected = stats.chi2_contingency(crosstab)

print(f"chi-square test statistic: {chi2_stat}")
print(f"P-value: {p_val}")
print(f"degrees of freedom: {dof}")
head = df.head()
print(head.to_markdown())
# YOUR CODE HERE
```