# *Exercises*

## *Jumping Rivers*

### *Exercise 1: Slopes and intercepts*

For graphs A, B & C, draw what you think is the line of best fit through the points, then calculate the slope and the gradient

| Graph | $\beta_0$ | $\beta_1$ |
|---|---|---|
| A | | |
| B | | |
| C | | |

### *Exercise 2: Residuals*

With your new knowledge of residuals, redraw the lines of best fits for graphs A, B & C. Calculate the residuals and RSS for each line you've drawn. Do you think there's a better line with a smaller RSS?

| Graph | $\beta_0$ | $\beta_1$ | RSS |
|---|---|---|---|
| A | | | |
| B | | | |
| C | | | |

### *Exercise 3: Loading the data*

Create a new ipython notebook, and call it "python_exercises". You'll benefit from having the workings from these exercise stored in a different notebook to the rest of your working. The following code will load the data in.

```
import pandas as pd
import jrpyml
hd = jrpyml.datasets.load_head_size()
hd.head()
```

```
##   gender age_range  head_size  brain_weight  height
## 0   Male     20-46       4512          1530      69
## 1   Male     20-46       3738          1297      72
## 2   Male     20-46       4261          1335      77
## 3   Male     20-46       3777          1282      72
## 4   Male     20-46       4177          1590      65
```

This is a data set consisting of 237 people, where we have collected their gender, age range, head size (cm^3), brain weight (grams) and height (inches). Let's say we're a scientist who is interested in predicting a persons brain weight based upon their head size.

a) Write down the simple linear regression model we could use to do this.

b) Set up the correct training data and response variable for this.

*Exercise 4: Visualising the data*

Using **seaborn**, produce a scatter plot of the head size against brain weight. What does the graph tell you?

*Exercise 5: fitting the model*

Fit the simple linear regression model you described in exercise 3

*Exercise 6: predictions*

a) A patient comes into a doctors with a head size of 5000 cm^3. After assessing an MRI, Dr Frankenstein predicts that his brain weight is between 1500-1550 grams. Is he right?

b) After seeing how great your model is, Dr Frankenstein would like some help in assessing the brain weight of his 3 new patients. Their head sizes are 2500, 3000 & 4500. Do this inside one call to `model.predict()`.

*Exercise 7: fitted values*

Using the fitted values, overlay the model line over the scatter plot you produced in exercise 4

*Exercise 8: residuals*

Using the fitted values, calculate the residual sum of squares

*Exercise 9: model coefficients*

a) Can you write down $\beta_0$ and $\beta_1$ for your model? We've seen how to extract $\beta_1$ in the notes. For $\beta_0$, take a look at the methods available with $dir(model)$. Remember, $\beta_0$ is the y-intercept.

b) What do these coefficients tell you about the relationship between head size and brain weight?

*Exercise 10: multiple linear regression*

Dr Frankenstein thinks that not only does head size affect brain weight, but that their also might be a relationship between a persons height and their brain weight.

a) Write down the multiple linear regression model you would now fit to examine this claim.

b) Train the model.

c) What are the coefficients for your model?

d) What is the residual sum of squares for your model? How does this compare to your previous model?

e) Dr Frankenstein has 3 new patients:

| Patient | Head size | Height |
|---|---|---|
| 1 | 4000 | 80 |
| 2 | 3000 | 70 |
| 3 | 2000 | 60 |

Give him an estimate of each patients brain weight using your model. Hint: create a **pandas** DataFrame containing the new values to pass to `model.predict()`

*Exercise 11: standardised residuals*

Calculate the standardised residuals for the model in exercise 10.

*Exercise 12: polynomials*

Come up with graphs for the third and fourth order polynomials, $y = X^3$ and $y = X^4$

*Exercise 13: More polynomials*

The following code generates some data and places it in a pandas data frame.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
x = np.arange(-5, 5.1, 0.1)
y = pow(x, 3) + np.sin(x) + np.random.normal(loc=0, scale=20, size=101)
```

```
example = pd.DataFrame({
    "x": x,
    "y": y
})
```

a) Create a scatter plot of x against y. Do you think a linear line would accurately represent the data? If not, what order polynomial do you think would suit this data?

b) Using the `PolynomialFeatures()` function, appropriately transform your predictor, `x`, and fit your model.

c) Using the fitted values, overlay your model line onto the scatter plot produced earlier in the exercise.

*Exercise 14: Standardisation*

Given that this is now day 2, we've probably lost the data. So here is the code to load it in again

```
import pandas as pd
import jrpyml
hd = jrpyml.datasets.load_head_size()
```

a) Create a new variable called "`head_size_stand`", that is the standardised version of `head_size`. For now, don't use the *preprocessing* module. Do it using only **NumPy** functionality.

b) Produce a boxplot of the new standardised head size, comparing males and females. Which gender has the biggest head (Metaphorically it's men, obviously)? ***Hint***: use `sns.boxplot()`

c) Let's say we want to fit the model $brainweight = \beta_0 + \beta_1 \times headsize + \beta_2 \times height$. Set up your `X_train` and `y_train` objects, and standardise both `head_size` and `height` but this time using `StandardScaler()`

d) Fit the model and predict the brain weight for a person with head size $4000 cm^3$ and 70 inches tall.

*Exercise 15: Min-Max*

Repeat exercise 14, but using the min max transform instead of the standardisation transform. Do you get the same prediction as exercise 14?

*Exercise 16: Pipelines*

Repeat the previous prediction, but this time do it by setting up a pipeline that first does a min-max transformation, and then second performs linear regression. You should get the same result.

*Exercise 17: Categorical data*

Up until this point, we've pretty much ignored the fact that we have two categorical variables in the data, `gender` and `age_range`. Let's say we want to build a model, that is able to esimate someones brain weight by their gender and nothing else.

a) What do you think the predicted value of brain weight for each gender will be?

b) Set up your `X_train` variable such that is only has `gender` in it

c) Set up a model pipeline that preprocesses the data via a one hot encoding scheme, then using linear regression.

d) Train the model and then predict what brain sizes a `Male` and `Female` would have.

e) What model have you trained? ***Hint***: Look at `model.named_steps["regression"].intercept_` and `model.named_steps["regression"].coef_`.

f) What is the average of brain weight for men and women in this data set? How does this compare with your previous predictions? Why do you think this is?

g) What is the RSS for this model? How does this compare to the RSS you calculated in question 8?

*Exercise 18: Combining steps*

Let's say we want to try both gender and head size as predictors. a) Set up the correct `X_train` variable

b) Using `ColumnTransformer`, create a preprocessor that one hot encodes gender and does a standardisation transform on head_size.

c) Create a pipeline where the preprocessor is the first step, and a linear regression model is the second step

d) Fit the model, and predict the brain weight of a Male with a $4000cm^3$ head and a Female with a $2000cm^3$ head.

e) What is the residual sum of squares of this model? How does it compare to the RSS in questions 8 & 17?

f) Draw the fitted values against head_size, what do you notice?

*Exercise 19: Validation set approach*

Let's go back to our very first model, $brainweight = \beta_0 + \beta_1 \times headsize$. Set up your predictor and response like so

```
X = hd["head_size"]
y = hd["brain_weight"]
```

a) Split the data into a training and validation set, using `train_test_split()`. Remember to reshape `X_train` and `X_test` after using `.reshape(-1,1)`.

b) Set up a pipeline to Standardise `head_size`, and then run a linear regression. Fit the model, then calculate the training and test error. Which is bigger? Can you think why this might be?

*Exercise 20: cross validation*

a) Set up the same model as you used in exercise 19.

b) We're going to use 10-fold cross validation to estimate the test error of our model. Set up a scoring function to do this.

c) Run 10-fold cross validation on the model.

d) What is the average test error?

*Exercise 21: bootstrap*

Using the bootstrap method, obtain a density plot, like the ones in section 4.6, of the coefficient to the predictor $head_size$ in your model.

*Exercise 22: Logistic regression*

We're going to switch this model around now, and instead of modelling the brain weight of a patient, we're going to the model the gender of a patient using their head size. The following code will set up the X_train and y_train objects for you

```
import pandas as pd
import jrpyml
hd = jrpyml.datasets.load_head_size()
X = hd.drop(columns="gender")
y = hd["gender"]
X_train = hd[["head_size"]]
y_train = y
```

a) The following code will show you a boxplot of head sizes per gender. What does this tell you about the relationship between gender and head size?

```
sns.boxplot(y="head_size", x="gender", data=hd)
```

b) Write a pipeline to standardise the predictor then perform logistic regression. Use that to fit the model

c) If a patient was to have a head size of 3500, what gender would you predict they were? What is the associated probability?

*Exercise 23: More logistic regression*

a) For the model in exercise 22. What percentage of predictions did you get right in the training data?

b) Of those the model classified as Male, what percentage were actually Male?

c) The following code will set up and perform 10-fold cross validation on the data. How does the average estimate of the accuracy on the test set compare to the accuracy in part a) ?

```
from sklearn.model_selection import cross_validate
from sklearn.metrics import make_scorer, accuracy_score, precision_score, recall_score
import pandas as pd

acc = make_scorer(accuracy_score)

def precision(y_true, y_pred):
    return precision_score(y_true, y_pred, pos_label="Male")

def recall(y_true, y_pred):
    return recall_score(y_true, y_pred, pos_label="Male")

prec = make_scorer(precision)
rec = make_scorer(recall)
output = cross_validate(model, X_train, y_train, scoring={
   'acc': acc,
   'prec': prec,
   'rec': rec
}, cv=10, return_train_score=False)
```