Application of Simple Regression Models to Body Composition Measurements

J.R. Powers-Luhn

***Abstract*—A dataset of body composition measurements is used to explore linear regression techniques. Several regression models are developed, predicting body fat percentage within 4%, performing unfavorably compared to the work of Penrose, et al [1]. The sensitivity of the model to the selection of training, testing, and validation data is briefly discussed.**

I. Introduction

Measurements of body fat percentage are desirable as they may be good predictors of heart attack risk later in life [2][3]. Currently the most accurate methods to measure body fat percentage require accurate measurements of overall density [4]. This can involve techniques as extreme as whole-body immersion in a pool of deuterated or tritiated water [2].

A. Linear Regression

In general, a linear regression seeks the coefficients $\vec{\alpha}$ which minimize the error on equation (1):

$$\mathbf{x}\vec{\alpha} = \vec{y} \tag{1}$$

where $\mathbf{x}$ represents the measured inputs from which we seek to predict the output, y. We can determine the best values for α by choosing to minimize the difference between our predicted and true values as in equation (2), leading to equation (3).

$$\begin{aligned} E(\alpha) &= (\mathbf{x}\vec{\alpha} - \vec{y})^2 \qquad (2)\\ \frac{\mathrm{d}E}{\mathrm{d}\alpha} &= 0 = 2(\mathbf{x}\vec{\alpha} - \vec{y})\mathbf{x} \\ \mathbf{x}\vec{\alpha}\mathbf{x} &= \vec{y}\mathbf{x} \\ \mathbf{x}^{-1}\mathbf{x}\vec{\alpha}\mathbf{x}\mathbf{x}^{-1} &= \mathbf{x}^{-1}\vec{y}\mathbf{x}\mathbf{x}^{-1} \\ \vec{\alpha} &= \mathbf{x}^{-1}\vec{y} \qquad (3) \end{aligned}$$

In order to have a unique solution to this equation, $\mathbf{x}$ must have at least as many rows as it has columns. In practice, the number of samples (rows) and the dimensionality of each sample (columns) are not linked to each other. In the case of the dataset examined in this paper, for example, there are many times more samples (247) than there are dimensions (14). In order to simplify this process, we therefore multiply $\mathbf{x}$ in equation (1) by its transpose $\mathbf{x}^T$ in order to obtain equation (4).

$$\vec{\alpha} = (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\vec{y} \tag{4}$$

Since the relationship between two quantities will not in general have a zero intercept, new "input" is added to the measured values prior to the regression. This input consists of a column of non-varying values (ones) which allows for the last element of α to correspond to the axis intercept when all inputs are zero.

B. Solution Stability and Covariance

Another complication arises in practice. In order to solve (1), it is necessary to have at least as many *linearly independent* rows as there are columns. In the event that some of the measured inputs covary, the solution becomes unstable and can vary wildly based on the noise in the sampled data. It may be desirable, therefore, to leave some sampled values out of the matrix in favor of others which capture the same information. A useful tool to choose variables to eliminate from consideration is the *covariance matrix* or its normalized cousin, the *correlation matrix.*

The covariance of two variables is calculated using equation (5).

$$cov(\vec{x}, \vec{y}) = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y}) \tag{5}$$

It is difficult to compare the covariance across the entire dimensionality of a dataset since the units are expected to differ. Therefore the more commonly employed value is the correlation (equation (6)).

$$corr(\vec{x}, \vec{y}) = \frac{cov(\vec{x}, \vec{y})}{\sigma_x \sigma_y} \tag{6}$$

C. Linear in parameters

Of course, often the variable which is to be predicted does not depend linearly on the input variables–for example, square or higher-order polynomial terms (an example is shown in equation (7)).

$$y = \alpha_1 x_1 + \alpha_2 x_1^2 + \alpha_3 x_2 + \alpha_4 x_1 x_2 \tag{7}$$

In such a case it is still possible to fit the model using equation (4). New "inputs" are created to take the non-linear terms (8). The coefficients can then be calculated.

$$\begin{aligned} &y = \alpha_1 x_1 + \alpha_2 x_3 + \alpha_3 x_2 + \alpha_4 x_4 \\ &x_3 = x_1^2 \\ &x_4 = x_1 x_2 \end{aligned} \tag{8}$$

While this technique does not work if the equation cannot be parameterized (e.g. $y = \sin(\alpha x)$), it can allow for the fitting of more complex datasets.

D. Dataset examined

The dataset analyzed was a subset of body measurements taken by Penrose[1]. This data set consists of 247 samples each of fourteen quantities:

- Age (yr)
- Weight (lbs)
- Height (inches)
- Adiposity index ($\mathrm{kg/m^2}$)
- Neck circumference (cm)
- Chest circumference (cm)
- Ab circumference (cm)
- Hip circumference (cm)
- Thigh circumference (cm)
- Knee circumference (cm)
- Ankle circumference (cm)
- Extended bicep circumference (cm)
- Forearm circumference (cm)
- Wrist circumference (cm)

143 of these measured values were used to derive the lean body weight equation (9), with abdominal (Ab) and wrist (Wr) circumferences measured in cm, weight (Wt) measured in kg, and height (Hgt) measured in m. This was then validated with 109 additional measurements.

$$LBW = 17.298 + 0.89946(Wt) - 0.2783(Age) \\ + 0.002617(Age^2) + 17.819(Hgt) - 0.6798(Ab - Wr) \tag{9}$$

Outliers from the data were removed in order to provide a more representative analysis.

A variety of regressions were performed on linear and non-linear combinations of these measured input variables.

II. Methodology

The data were split into three groups for the purposes of this analysis. All regression was performed on the **training** group. Peformance scores for each regression were calculated using the **testing** group, and final results were reported using the **validation** group. Since each sample corresponded to a single patient there was deemed to be no relationship between rows in the dataset. This meant that the set could be divided by shuffling the rows and selecting the first 60% as training data, the second 20% as testing data, and the remaining 20% as validation data. Since these fractions did not match whole numbers of training rows, the floor of each fraction ($\frac{trainrows}{totalrows}$, $\frac{testrows}{totalrows}$, $\frac{valrows}{totalrows}$) was taken and the remaining rows added to the training set. Performance for each regression was calculated using the root mean squared error (RMSE) for that prediction (10), where x_i and y_i are from the test set and α is calculated from the training set.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=0}^{N}(x_i\alpha - y_i)^2} \tag{10}$$

In order to test the one-variable regression performance, each column was used to perform a regression and the performance of each regression on the testing set was calculated.

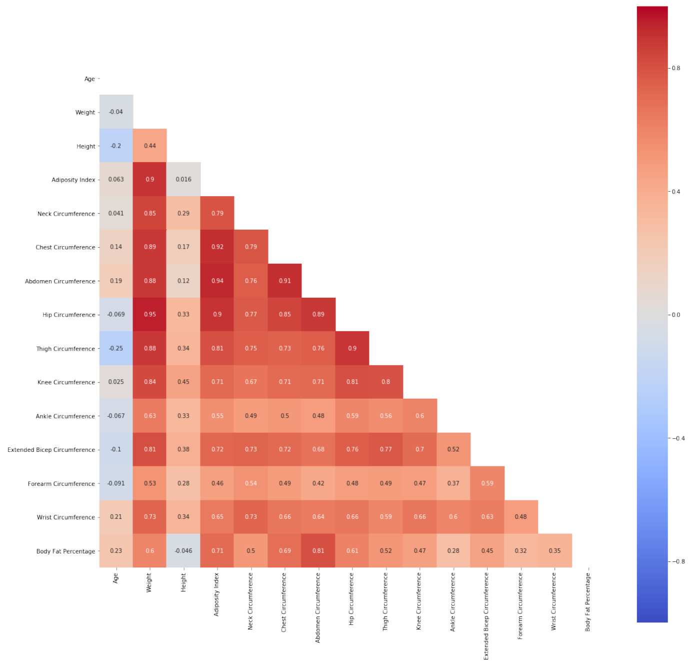


Fig. 1. Correlations between each pair of data. Darker colors represent more correlation. All values except height and age are positively correlated with each other.

Next, two-variable regressions were performed on every combination of two inputs. An additional regression was performed using all fourteen of the input variables.

A correlation matrix was computed for the input and output values in order to inform a "best guess" of factors to use in a linear model. The inputs were selected based on both their correlation to body fat percentage and their lack of correlation to each other. This model was constructed and its performance measured.

In order to test the performance of non-linear terms, inverse and quadratic combinations of the input variables were calculated and used to perform additional regressions.

III. Results

The correlation matrix calculated on the training set revealed that most values were somewhat correlated with body fat percentage (¿0.3), but only abdominal circumference and adiposity index were strongly correlated (¿0.7). Age, height, and ankle circumference were uncorrelated (¡0.3) with body fat. Forearm and wrist circumference were borderline (0.32 and 0.35, respectively). A representation of the complete correlation matrix is shown in figure 1.

Fourteen regressions were performed using a single column as input. The best single input was determined to be abdominal circumference based on its test error of 4.47. The single worst input was determined to be height with an error of 6.74. The validation performance of the abdominal circumference model was 4.14. The performance of all models on the test set can be seen in figure 2.

The regressions against paired columns showed slightly better results. The performance of these models ranged from a

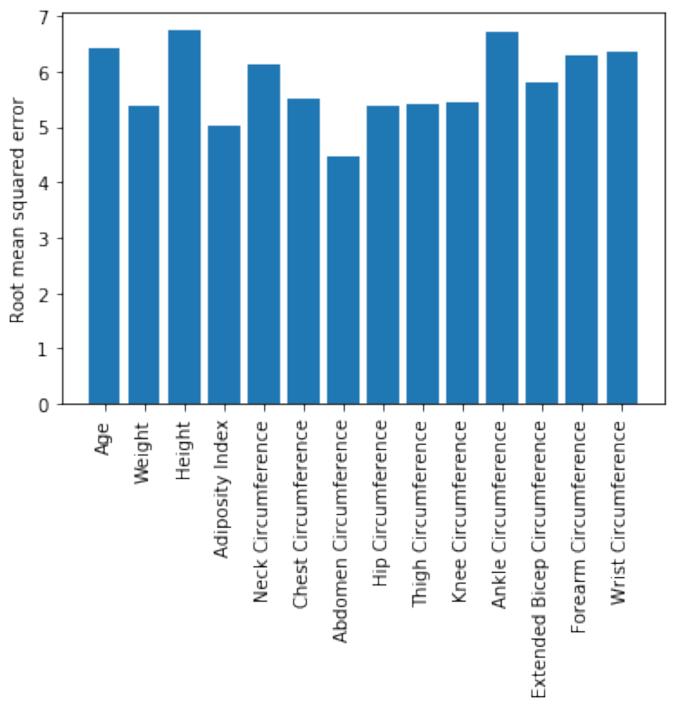


Fig. 2. Performance of one-variable regression models on the test set as a function of input variable. The range of errors was 2.3 percentage points. Inputs with the highest correlation to the output were unsuprisingly higher performing.

minimum of 4.20 (abdomen circumference and wrist circumference) to a maximum of 6.84 (height and ankle circumference). Validation performance of the best model produced an RMSE of 4.07. While wrist circumference was particularly uncorrelated with body fat percentage, it is theorized that including it in this model improved performance because it was also uncorrelated with abdomen circumference and thus contained information not captured in the abdomen circumference value alone. Height was also uncorrelated with abdomen circumference and body fat percentage, but the model that included abdomen circumference and height had a validation error of only 3.67. This suggests that choosing variables based on their orthogonality and not simply correlation can have a positive effect on a model.

Regression against the entire range of fourteen inputs produced performance on the test set with an RMSE of 4.55, but it fared quite a bit better on the validation set (3.75). This trend was followed by the "best guess" regression against abdomen circumference, wrist circumference, and height, whose performance on the validation set (3.83) was also better than the performance on the test set (4.14).

Based on the theory that body fat percentage should be inversely correlated with weight, a regression was performed against abdomen circumference, height, and wrist circumference with the addition of $\frac{1}{weight}$. This theory was not born out, with a test error of 4.11 but a validation error of 3.87, performing approximately as well as the "best guess" regression.

While it would not be appropriate to choose a model based on the results of the validation set, the swings in accuracy between test and validation performance indicate that the statistics of their respective data sets may be different. Based on the small variation in accuracy when comparing these models, it is not possible to select a "best" model from the data presented in this paper.

A comparison was made of the errors presented to the standard deviation of body fat percentage in the training set. The standard deviation of body fat percentage in the training set was 7.88. This means that the model which performed the best on the test set (abdomen and wrist circumference, height, $weight^{-1}$) had a validation error of approximately 1/2 standard deviations. By contrast, the worst-performing single-variable regression (height) had a test error of 0.87 standard deviations.

IV. Conclusions

Several models were generated, predicting body fat percentage from height, weight, and circumference measurments to an error of approximately 4%. Performance measurements were limited to root mean squared error, but will be extended in the future to include variance in the error in an attempt to capture this. More sophisticated cross-validation techniques may be appropriate when examining datasets with high variance in their values.

References

[1] K W Penrose, A G Nelson, and A G Fisher. "Generalized Body Composition Prediction Equation For Men Using Simple Measurement Techniques". In: *Medicine & Science in Sports & Exercise* 17.2 (1985). ISSN: 0195-9131.

[2] Dympna Gallagher, Steven B Heymsfield, Moonseong Heo, et al. "Healthy percentage body fat ranges: an approach for developing guidelines based on body mass index". In: *The American Journal of Clinical Nutrition* 72.3 (2000), pp. 694–701. DOI: 10.1093/ajcn/72.3.694. eprint: /oup/backfile/content_public/journal/ajcn/72/3/10.1093_ajcn_72.3.694/3/694.pdf. URL: http://dx.doi.org/10.1093/ajcn/72.3.694.

[3] Eric B. Rimm, Meir J. Stampfer, Edward Giovannucci, et al. "Body Size and Fat Distribution as Predictors of Coronary Heart Disease among Middle-aged and Older US Men". In: *American Journal of Epidemiology* 141.12 (1995), pp. 1117–1127. DOI: 10.1093/oxfordjournals.aje.a117385. eprint: /oup/backfile/content_public/journal/aje/141/12/10.1093/oxfordjournals.aje.a117385/2/141-12-1117.pdf. URL: http://dx.doi.org/10.1093/oxfordjournals.aje.a117385.

[4] Roger W. Johnson. "Fitting Percentage of Body Fat to Simple Body Measurements". In: *Journal of Statistics Education* 4.1 (1996), null. DOI: 10.1080/10691898.1996.11910505. eprint: https://doi.org/10.1080/10691898.1996.11910505. URL: https://doi.org/10.1080/10691898.1996.11910505.

Homework 3 Pandas

September 25, 2018

Due to the random selection of rows, some variation is introduced in the selection of the training, test, and validation sets each time this notebook is run. This is the source of any deviation between the results recorded here and those reported in the attached paper.

1 Imports

```
In [1]: %matplotlib inline
        import matplotlib.pyplot as plt
        import seaborn as sns
        import pandas as pd
        import numpy as np
        import numpy.linalg as la
        import scipy.io as sio
        from math import floor
```

2 Load the data

```
In [2]: file = 'hwkdataNEW.mat'
        data_dict = sio.loadmat(file)
        print(data_dict.keys())
```

```
dict_keys(['__header__', '__version__', '__globals__', 'x', 'y'])
```

Extract the parts we want and examine them

```
In [3]: x = data_dict['x']
        y = data_dict['y']

In [4]: x.shape

Out[4]: (247, 14)

In [5]: y.shape

Out[5]: (247, 1)
```

We want to put this all into the pandas data format

```
In [6]: column_names = ['Age', 'Weight', 'Height', 'Adiposity Index',
                        'Neck Circumference', 'Chest Circumference',
                        'Abdomen Circumference', 'Hip Circumference',
                        'Thigh Circumference', 'Knee Circumference',
                        'Ankle Circumference', 'Extended Bicep Circumference',
                        'Forearm Circumference', 'Wrist Circumference',
                        'Body Fat Percentage']
```

```
In [7]: data = pd.DataFrame(x, columns=column_names[:-1]) # Last column contains y
```

```
In [8]: data[column_names[-1]] = y
```

```
In [9]: data.head()
```

```
Out[9]:     Age  Weight  Height  Adiposity Index  Neck Circumference  \
        0  23.0  154.25   67.75             23.7                36.2
        1  22.0  173.25   72.25             23.4                38.5
        2  22.0  154.00   66.25             24.7                34.0
        3  26.0  184.75   72.25             24.9                37.4
        4  24.0  184.25   71.25             25.6                34.4

           Chest Circumference  Abdomen Circumference  Hip Circumference  \
        0                 93.1                   85.2               94.5
        1                 93.6                   83.0               98.7
        2                 95.8                   87.9               99.2
        3                101.8                   86.4              101.2
        4                 97.3                  100.0              101.9

           Thigh Circumference  Knee Circumference  Ankle Circumference  \
        0                 59.0                37.3                 21.9
        1                 58.7                37.3                 23.4
        2                 59.6                38.9                 24.0
        3                 60.1                37.3                 22.8
        4                 63.2                42.2                 24.0

           Extended Bicep Circumference  Forearm Circumference  Wrist Circumference  \
        0                          32.0                   27.4                 17.1
        1                          30.5                   28.9                 18.2
        2                          28.8                   25.2                 16.6
        3                          32.4                   29.4                 18.2
        4                          32.2                   27.7                 17.7

           Body Fat Percentage
        0                 12.6
        1                  6.9
        2                 24.6
        3                 10.9
        4                 27.8
```

Data is now a pandas DataFrame that contains the information we need

3 Sort into Train, Test, and Validation

Since the rows of data should be indpendent and uncorrelated, I have chosen to simply shuffle them then tag them as belonging to one of the named data sets (`train`, `test`, or `validation`).

```
In [10]: train_frac = 0.6
         test_frac = 0.2
         val_frac = 0.2

         train_len = floor(len(data) * train_frac)
         test_len = floor(len(data) * test_frac)
         val_len = floor(len(data) * val_frac)
         rows_used = train_len + test_len + val_len
         # Make sure all rows being used by adding unused
         # rows into the training set
         while rows_used < len(data):
             train_len += 1
             rows_used = train_len + test_len + val_len
         print(f'Rows in original dataset: {len(data)}')
         print(f'Rows in training dataset: {train_len}')
         print(f'Rows in testing dataset: {test_len}')
         print(f'Rows in validation dataset: {val_len}')
         print(f'Rows used: {train_len + test_len + val_len}')
```

```
Rows in original dataset: 247
Rows in training dataset: 149
Rows in testing dataset: 49
Rows in validation dataset: 49
Rows used: 247
```

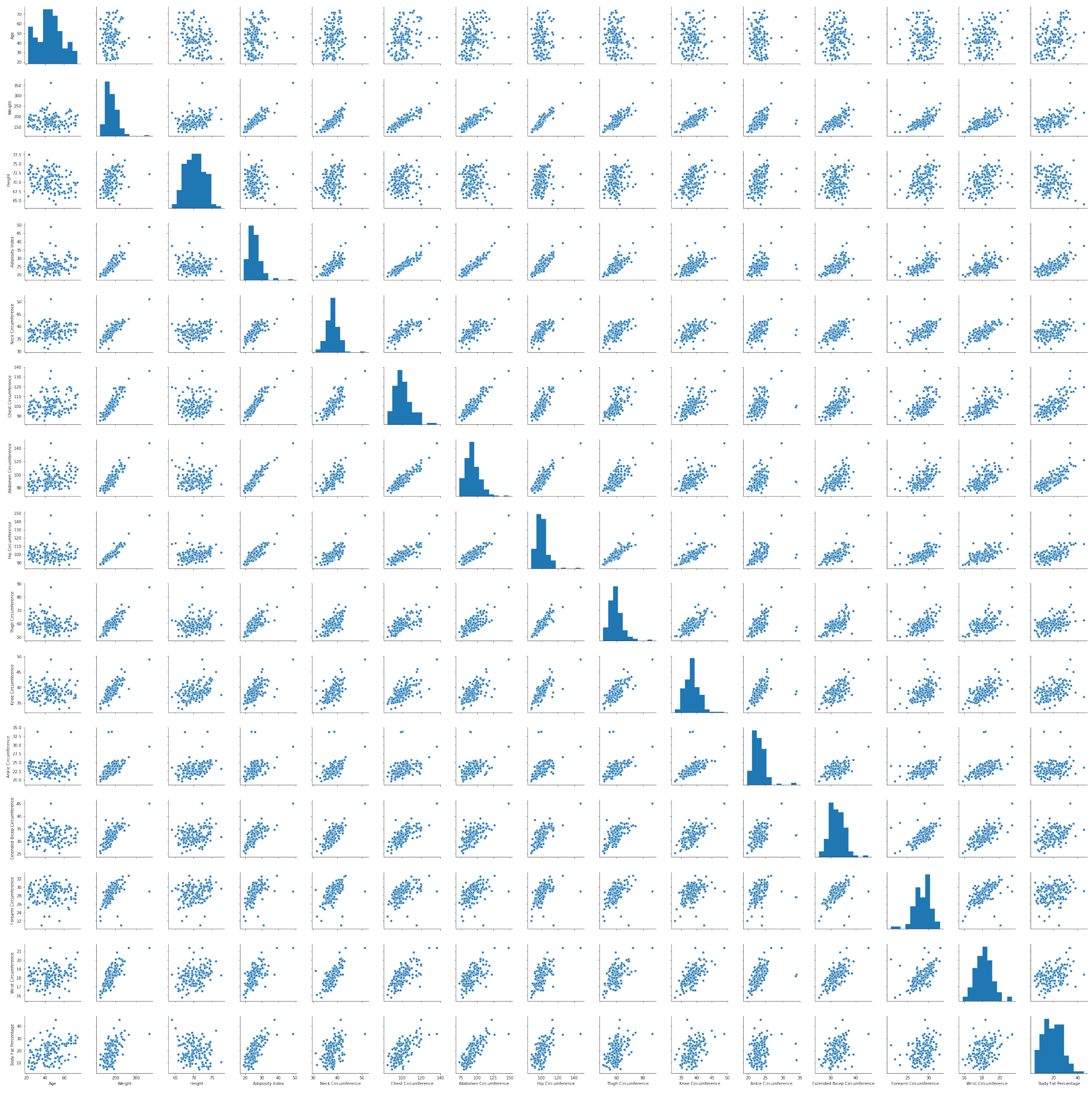
```
In [11]: # Shuffle and reindex the dataset
         data = data.sample(frac=1).reset_index(drop=True)
         # Split the data into three datasets
         dataset_labels = np.zeros(len(data)).astype('str')
         dataset_labels[:train_len] = 'train'
         dataset_labels[train_len:train_len + test_len] = 'test'
         dataset_labels[-val_len:] = 'val'
         data['sample'] = dataset_labels
```

3.1 TO DO: Make sure max and min for each column is in training set

4 Regression

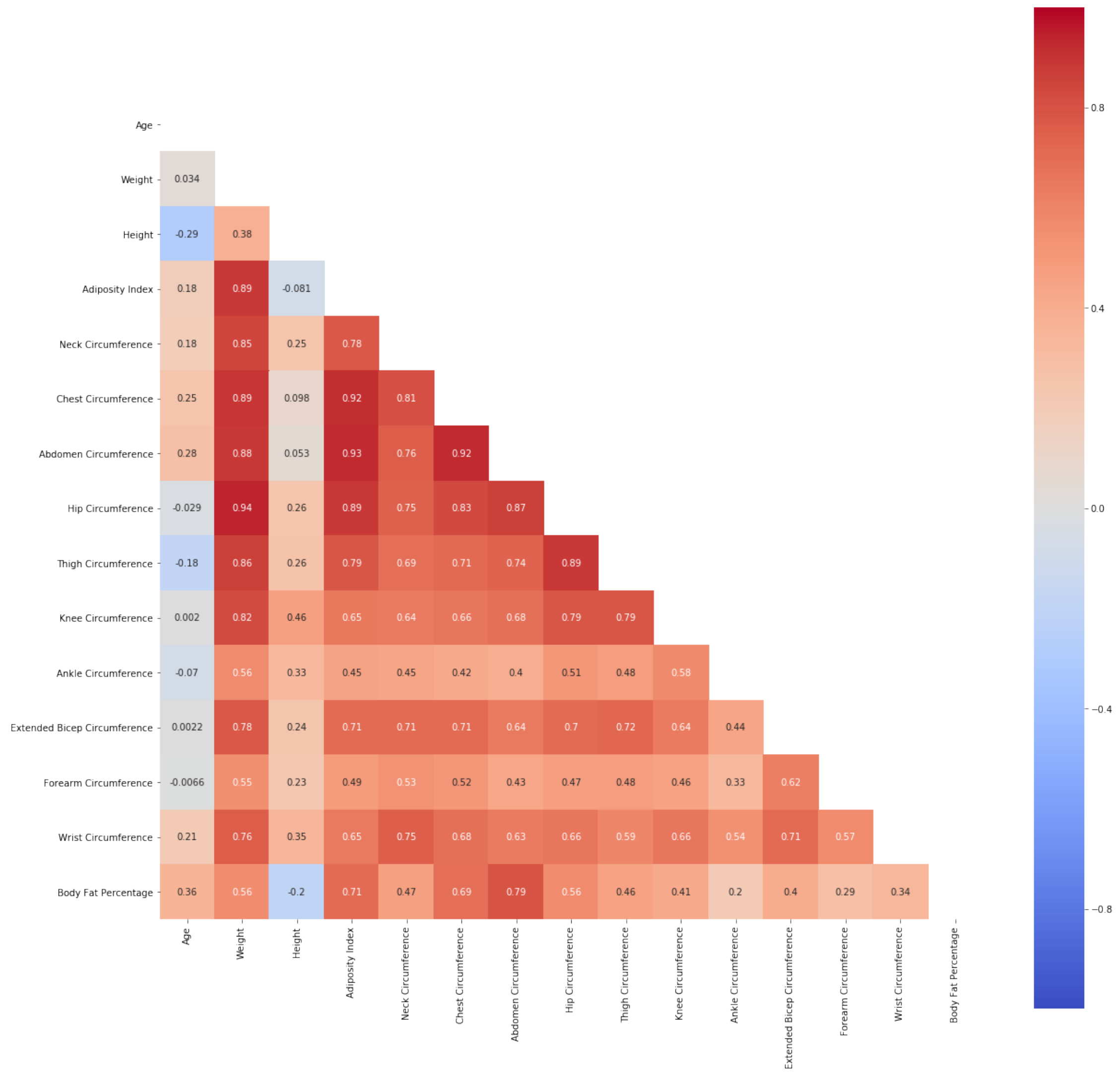
Explanation of regression

```
In [12]: train = data[data['sample']=='train']
In [13]: def regress(x, y):
             """Returns coefficient of regression"""
             # One-pad x
             o = np.ones_like(x[:,0]).reshape((-1, 1))

             x = np.hstack([x, o])

             # Perform regression
             coefficients = la.inv(x.T @ x) @ x.T @ y

             return coefficients

In [14]: def performance(x_test, y_test, coefficients):
             """Root mean squared error"""
             # One-pad x
             o = np.ones_like(x_test[:,0]).reshape((-1, 1))
             x_test = np.hstack([x_test, o])

             # Make predictions
             y_pred = x_test @ coefficients

             # Calculate RMSE
             return np.sqrt(np.sum((y_test - y_pred) ** 2) / y_test.shape[0])

In [15]: sns.pairplot(train)

Out[15]: <seaborn.axisgrid.PairGrid at 0x1a22347ac8>
```



Wow, that's a little unweildy. Let's try just the correlation matrix

```
In [16]: plt.figure(figsize=(20,20))
         mask = np.zeros_like(train.corr(), dtype=np.bool)
         mask[np.triu_indices_from(mask)] = True
         sns.heatmap(train.corr(), vmin=-1, vmax=1, center=0,
                     square=True, cmap='coolwarm', annot=True, mask=mask)
         plt.show()
```

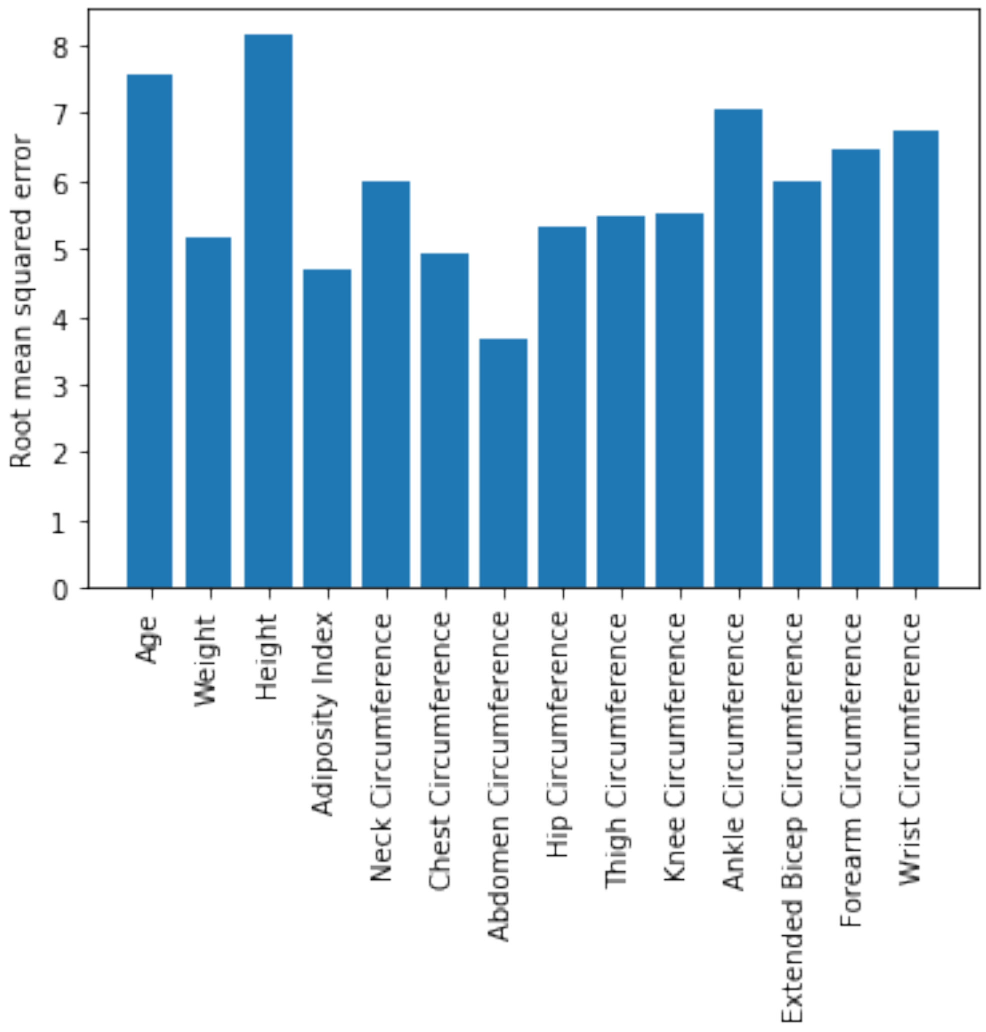


4.1 Attempt a 1-value regression for each column

```
In [17]: test = data[data['sample'] == 'test']
         test_performance = {'one_factor': {}}
         for name in column_names[:-1]:
             test_performance['one_factor'][name] = performance(
                 test[name].values.reshape(-1, 1),
                 test['Body Fat Percentage'].values,
                 regress(train[name].values.reshape(-1, 1), train['Body Fat Percentage'].value
             )
         print('Root mean squared error vs regressor')
         print('-------------------------------------')
         for f, p in test_performance['one_factor'].items():
             print(f'{f}\t\t{p:0.2f}')
```

```
Root mean squared error vs regressor
------------------------------------
Age                 7.57
Weight                 5.16
Height                 8.15
Adiposity Index                 4.71
Neck Circumference                 6.00
Chest Circumference                 4.93
Abdomen Circumference                 3.67
Hip Circumference                 5.34
Thigh Circumference                 5.50
Knee Circumference                 5.51
Ankle Circumference                 7.06
Extended Bicep Circumference                 5.99
Forearm Circumference                 6.48
Wrist Circumference                 6.75
```

```
In [18]: f = plt.figure()
         plt.bar(x=column_names[:-1], height=[v for v in test_performance['one_factor'].values
         plt.xticks(rotation=90)
         plt.ylabel('Root mean squared error')
         plt.show()
```



5 Pairwise Regression

```
In [19]: from itertools import combinations

In [20]: combs = combinations(column_names[:-1], 2)

In [21]: two_factor = {}
         for _ in combs:
             c = [i for i in _]
             p = performance(test[c].values, test['Body Fat Percentage'].values,
                             regress(train[c].values, train['Body Fat Percentage'].values))
             two_factor[tuple(c)] = p
             print(f'{c} \t\t\t\t {p:0.2f}')
```

```
['Age', 'Weight']                                  5.02
['Age', 'Height']                                  7.90
['Age', 'Adiposity Index']                                  4.71
['Age', 'Neck Circumference']                                  6.00
['Age', 'Chest Circumference']                                  4.89
['Age', 'Abdomen Circumference']                                  3.77
['Age', 'Hip Circumference']                                  5.14
['Age', 'Thigh Circumference']                                  5.18
['Age', 'Knee Circumference']                                  5.80
['Age', 'Ankle Circumference']                                  7.03
['Age', 'Extended Bicep Circumference']                                  5.90
['Age', 'Forearm Circumference']                                  6.40
['Age', 'Wrist Circumference']                                  7.02
['Weight', 'Height']                                  5.19
['Weight', 'Adiposity Index']                                  5.03
['Weight', 'Neck Circumference']                                  5.15
['Weight', 'Chest Circumference']                                  5.22
['Weight', 'Abdomen Circumference']                                  3.72
['Weight', 'Hip Circumference']                                  5.18
['Weight', 'Thigh Circumference']                                  5.21
['Weight', 'Knee Circumference']                                  5.30
['Weight', 'Ankle Circumference']                                  5.04
['Weight', 'Extended Bicep Circumference']                                  5.18
['Weight', 'Forearm Circumference']                                  5.19
['Weight', 'Wrist Circumference']                                  5.15
['Height', 'Adiposity Index']                                  5.03
['Height', 'Neck Circumference']                                  6.55
['Height', 'Chest Circumference']                                  5.20
['Height', 'Abdomen Circumference']                                  3.89
['Height', 'Hip Circumference']                                  5.58
['Height', 'Thigh Circumference']                                  5.96
['Height', 'Knee Circumference']                                  5.96
['Height', 'Ankle Circumference']                                  7.69
['Height', 'Extended Bicep Circumference']                                  6.54
['Height', 'Forearm Circumference']                                  7.08
['Height', 'Wrist Circumference']                                  7.61
['Adiposity Index', 'Neck Circumference']                                  4.79
['Adiposity Index', 'Chest Circumference']                                  4.62
['Adiposity Index', 'Abdomen Circumference']                                  3.61
['Adiposity Index', 'Hip Circumference']                                  4.98
['Adiposity Index', 'Thigh Circumference']                                  5.00
['Adiposity Index', 'Knee Circumference']                                  4.85
['Adiposity Index', 'Ankle Circumference']                                  4.72
['Adiposity Index', 'Extended Bicep Circumference']                                  4.86
['Adiposity Index', 'Forearm Circumference']                                  4.83
['Adiposity Index', 'Wrist Circumference']                                  4.78
['Neck Circumference', 'Chest Circumference']                                  5.16
['Neck Circumference', 'Abdomen Circumference']                                  3.67
```

```
['Neck Circumference', 'Hip Circumference']                                   5.29
['Neck Circumference', 'Thigh Circumference']                                   5.47
['Neck Circumference', 'Knee Circumference']                                   5.51
['Neck Circumference', 'Ankle Circumference']                                   6.01
['Neck Circumference', 'Extended Bicep Circumference']                                    5.83
['Neck Circumference', 'Forearm Circumference']                                   5.94
['Neck Circumference', 'Wrist Circumference']                                   6.00
['Chest Circumference', 'Abdomen Circumference']                                  3.64
['Chest Circumference', 'Hip Circumference']                                   4.97
['Chest Circumference', 'Thigh Circumference']                                   5.05
['Chest Circumference', 'Knee Circumference']                                   5.03
['Chest Circumference', 'Ankle Circumference']                                   4.93
['Chest Circumference', 'Extended Bicep Circumference']                                    5.10
['Chest Circumference', 'Forearm Circumference']                                   5.06
['Chest Circumference', 'Wrist Circumference']                                   5.11
['Abdomen Circumference', 'Hip Circumference']                                   3.70
['Abdomen Circumference', 'Thigh Circumference']                                   3.85
['Abdomen Circumference', 'Knee Circumference']                                   3.76
['Abdomen Circumference', 'Ankle Circumference']                                   3.60
['Abdomen Circumference', 'Extended Bicep Circumference']                                    3.8
['Abdomen Circumference', 'Forearm Circumference']                                   3.75
['Abdomen Circumference', 'Wrist Circumference']                                   3.56
['Hip Circumference', 'Thigh Circumference']                                    5.51
['Hip Circumference', 'Knee Circumference']                                    5.47
['Hip Circumference', 'Ankle Circumference']                                    5.31
['Hip Circumference', 'Extended Bicep Circumference']                                     5.33
['Hip Circumference', 'Forearm Circumference']                                    5.29
['Hip Circumference', 'Wrist Circumference']                                    5.36
['Thigh Circumference', 'Knee Circumference']                                   5.29
['Thigh Circumference', 'Ankle Circumference']                                   5.49
['Thigh Circumference', 'Extended Bicep Circumference']                                    5.44
['Thigh Circumference', 'Forearm Circumference']                                   5.36
['Thigh Circumference', 'Wrist Circumference']                                   5.51
['Knee Circumference', 'Ankle Circumference']                                   5.47
['Knee Circumference', 'Extended Bicep Circumference']                                    5.37
['Knee Circumference', 'Forearm Circumference']                                   5.38
['Knee Circumference', 'Wrist Circumference']                                   5.63
['Ankle Circumference', 'Extended Bicep Circumference']                                   5.98
['Ankle Circumference', 'Forearm Circumference']                                   6.36
['Ankle Circumference', 'Wrist Circumference']                                   6.73
['Extended Bicep Circumference', 'Forearm Circumference']                                   5.9
['Extended Bicep Circumference', 'Wrist Circumference']                                   5.98
['Forearm Circumference', 'Wrist Circumference']                                   6.39
```

```
In [22]: print(max(two_factor, key=two_factor.get))
         print(max([v for v in two_factor.values()]))
```

```
('Age', 'Height')
```

```
7.901445978984846
```

```
In [23]: print(min(two_factor, key=two_factor.get))
         print(min([v for v in two_factor.values()]))
```

```
('Abdomen Circumference', 'Wrist Circumference')
3.5561491191764545
```

6 All 14 Variables

```
In [24]: p = performance(x_test=test[column_names[:-1]].values,
                         y_test=test['Body Fat Percentage'].values,
                         coefficients=regress(
                             train[column_names[:-1]].values,
                             train['Body Fat Percentage'].values,
                         ))
         test_performance['All inputs performance'] = p
         print(f'All inputs performance\t\t{p:0.2f}')
```

```
All inputs performance                3.41
```

7 My best guess

```
In [25]: predictive_columns = ['Abdomen Circumference', 'Wrist Circumference', 'Height']
         p = performance(x_test=test[predictive_columns].values,
                         y_test=test['Body Fat Percentage'].values,
                         coefficients=regress(
                             train[predictive_columns].values,
                             train['Body Fat Percentage'].values,
                         ))
         test_performance['Predictive columns performance'] = p
         print(f'Predictive columns performance\t\t{p:0.2f}')
```

```
Predictive columns performance                3.70
```

8 Non-linear terms

In order to improve the model it's worth trying higher-order (non-linear) terms.

Add a term for the square of each input variable

```
In [26]: for c in column_names[:-1]:
             data[f'{c}^2'] = data[c] * data[c]
```

Add a term for the product of each combination of two variables

```
In [27]: combs = combinations(column_names[:-1], 2)
         for c1, c2 in combs:
             data[f'{c1} x {c2}'] = data[c1] * data[c2]

In [28]: for c in column_names[:-1]:
             data[f'{c}^-1'] = data[c] ** -1

In [29]: train = data[data['sample']=='train']
         test = data[data['sample'] == 'test']

In [30]: train.head()

Out[30]:     Age  Weight  Height  Adiposity Index  Neck Circumference  \
         0  43.0  164.25   73.25             21.3                35.7
         1  36.0  226.75   71.75             31.0                41.5
         2  45.0  135.75   68.50             20.4                32.8
         3  48.0  173.75   72.00             23.6                37.0
         4  24.0  208.50   72.75             27.7                39.2

            Chest Circumference  Abdomen Circumference  Hip Circumference  \
         0                 96.6                   81.5               97.2
         1                115.3                  108.8              114.4
         2                 92.3                   83.4               90.4
         3                 99.1                   92.0               98.3
         4                102.0                   99.1              110.1

            Thigh Circumference  Knee Circumference           ...              \
         0                 58.4                38.2           ...
         1                 69.2                42.4           ...
         2                 52.0                35.8           ...
         3                 59.3                38.4           ...
         4                 71.2                43.5           ...

            Neck Circumference^-1  Chest Circumference^-1  Abdomen Circumference^-1  \
         0               0.028011                0.010352                  0.012270
         1               0.024096                0.008673                  0.009191
         2               0.030488                0.010834                  0.011990
         3               0.027027                0.010091                  0.010870
         4               0.025510                0.009804                  0.010091

            Hip Circumference^-1  Thigh Circumference^-1 Knee Circumference^-1  \
         0              0.010288                0.017123              0.026178
         1              0.008741                0.014451              0.023585
         2              0.011062                0.019231              0.027933
         3              0.010173                0.016863              0.026042
         4              0.009083                0.014045              0.022989

            Ankle Circumference^-1  Extended Bicep Circumference^-1  \
         0                0.042735                         0.033670
```

```
1                 0.041667                             0.028249
2                 0.048544                             0.034722
3                 0.044643                             0.035842
4                 0.039683                             0.027701

   Forearm Circumference^-1  Wrist Circumference^-1
0                  0.036496                0.054645
1                  0.047619                0.049751
2                  0.039216                0.061350
3                  0.038168                0.058824
4                  0.033003                0.053476

[5 rows x 135 columns]
```

```
In [31]: train.columns
```

```
Out[31]: Index(['Age', 'Weight', 'Height', 'Adiposity Index', 'Neck Circumference',
                'Chest Circumference', 'Abdomen Circumference', 'Hip Circumference',
                'Thigh Circumference', 'Knee Circumference',
                ...
                'Neck Circumference^-1', 'Chest Circumference^-1',
                'Abdomen Circumference^-1', 'Hip Circumference^-1',
                'Thigh Circumference^-1', 'Knee Circumference^-1',
                'Ankle Circumference^-1', 'Extended Bicep Circumference^-1',
                'Forearm Circumference^-1', 'Wrist Circumference^-1'],
               dtype='object', length=135)
```

```
In [32]: train_x = train.drop(['Body Fat Percentage', 'sample'], axis=1)
         print(*train_x.columns, sep='\n')
```

```
Age
Weight
Height
Adiposity Index
Neck Circumference
Chest Circumference
Abdomen Circumference
Hip Circumference
Thigh Circumference
Knee Circumference
Ankle Circumference
Extended Bicep Circumference
Forearm Circumference
Wrist Circumference
Age^2
Weight^2
Height^2
Adiposity Index^2
Neck Circumference^2
```

```
Chest Circumference^2
Abdomen Circumference^2
Hip Circumference^2
Thigh Circumference^2
Knee Circumference^2
Ankle Circumference^2
Extended Bicep Circumference^2
Forearm Circumference^2
Wrist Circumference^2
Age x Weight
Age x Height
Age x Adiposity Index
Age x Neck Circumference
Age x Chest Circumference
Age x Abdomen Circumference
Age x Hip Circumference
Age x Thigh Circumference
Age x Knee Circumference
Age x Ankle Circumference
Age x Extended Bicep Circumference
Age x Forearm Circumference
Age x Wrist Circumference
Weight x Height
Weight x Adiposity Index
Weight x Neck Circumference
Weight x Chest Circumference
Weight x Abdomen Circumference
Weight x Hip Circumference
Weight x Thigh Circumference
Weight x Knee Circumference
Weight x Ankle Circumference
Weight x Extended Bicep Circumference
Weight x Forearm Circumference
Weight x Wrist Circumference
Height x Adiposity Index
Height x Neck Circumference
Height x Chest Circumference
Height x Abdomen Circumference
Height x Hip Circumference
Height x Thigh Circumference
Height x Knee Circumference
Height x Ankle Circumference
Height x Extended Bicep Circumference
Height x Forearm Circumference
Height x Wrist Circumference
Adiposity Index x Neck Circumference
Adiposity Index x Chest Circumference
Adiposity Index x Abdomen Circumference
Adiposity Index x Hip Circumference
Adiposity Index x Thigh Circumference
Adiposity Index x Knee Circumference
Adiposity Index x Ankle Circumference
Adiposity Index x Extended Bicep Circumference
Adiposity Index x Forearm Circumference
Adiposity Index x Wrist Circumference
Neck Circumference x Chest Circumference
Neck Circumference x Abdomen Circumference
Neck Circumference x Hip Circumference
Neck Circumference x Thigh Circumference
Neck Circumference x Knee Circumference
Neck Circumference x Ankle Circumference
Neck Circumference x Extended Bicep Circumference
Neck Circumference x Forearm Circumference
Neck Circumference x Wrist Circumference
Chest Circumference x Abdomen Circumference
Chest Circumference x Hip Circumference
Chest Circumference x Thigh Circumference
Chest Circumference x Knee Circumference
Chest Circumference x Ankle Circumference
Chest Circumference x Extended Bicep Circumference
Chest Circumference x Forearm Circumference
Chest Circumference x Wrist Circumference
Abdomen Circumference x Hip Circumference
Abdomen Circumference x Thigh Circumference
Abdomen Circumference x Knee Circumference
Abdomen Circumference x Ankle Circumference
Abdomen Circumference x Extended Bicep Circumference
Abdomen Circumference x Forearm Circumference
Abdomen Circumference x Wrist Circumference
Hip Circumference x Thigh Circumference
Hip Circumference x Knee Circumference
Hip Circumference x Ankle Circumference
Hip Circumference x Extended Bicep Circumference
Hip Circumference x Forearm Circumference
Hip Circumference x Wrist Circumference
Thigh Circumference x Knee Circumference
Thigh Circumference x Ankle Circumference
Thigh Circumference x Extended Bicep Circumference
Thigh Circumference x Forearm Circumference
Thigh Circumference x Wrist Circumference
Knee Circumference x Ankle Circumference
Knee Circumference x Extended Bicep Circumference
Knee Circumference x Forearm Circumference
Knee Circumference x Wrist Circumference
Ankle Circumference x Extended Bicep Circumference
Ankle Circumference x Forearm Circumference
Ankle Circumference x Wrist Circumference
Extended Bicep Circumference x Forearm Circumference
Extended Bicep Circumference x Wrist Circumference
Forearm Circumference x Wrist Circumference
Age^-1
Weight^-1
Height^-1
Adiposity Index^-1
Neck Circumference^-1
Chest Circumference^-1
Abdomen Circumference^-1
Hip Circumference^-1
Thigh Circumference^-1
Knee Circumference^-1
Ankle Circumference^-1
Extended Bicep Circumference^-1
Forearm Circumference^-1
Wrist Circumference^-1
```

```
In [33]: theory_1_columns = ['Abdomen Circumference', 'Weight^-1', 'Height', 'Wrist Circumfere
         c = theory_1_columns
         performance(test[c].values, test['Body Fat Percentage'].values,
                     regress(train[c].values, train['Body Fat Percentage'].values))

Out[33]: 3.683762778355379

In [34]: theory_2_columns = ['Abdomen Circumference', 'Weight^-1', 'Height', 'Wrist Circumfere
         c = theory_2_columns
         performance(test[c].values, test['Body Fat Percentage'].values,
                     regress(train[c].values, train['Body Fat Percentage'].values))

Out[34]: 3.683762778355379

In [35]: theory_3_columns = train_x.columns
         c = theory_3_columns
         performance(test[c].values, test['Body Fat Percentage'].values,
                     regress(train[c].values, train['Body Fat Percentage'].values))

Out[35]: 20.737652502714663
```

9 Final Validation Performance Calculations

The final performance for the "best" models from the categories above will now be calculated on the validation set

```
In [36]: val = data[data['sample'] == 'val']
```

9.1 Single variable performance

```
In [37]: one_variable_columns = ['Abdomen Circumference']
         c = one_variable_columns
         performance(val[c].values, val['Body Fat Percentage'].values,
                                    regress(train[c].values, train['Body Fat Percentage'].values))
```

```
Out[37]: 4.265121424789481
```

9.2 Two variable performance

```
In [38]: two_variable_columns = ['Abdomen Circumference', 'Wrist Circumference']
         c = two_variable_columns
         performance(val[c].values, val['Body Fat Percentage'].values,
                                    regress(train[c].values, train['Body Fat Percentage'].values))
```

```
Out[38]: 3.9050699059210032
```

```
In [39]: other_two_variable_columns = ['Abdomen Circumference', 'Height']
         c = other_two_variable_columns
         performance(val[c].values, val['Body Fat Percentage'].values,
                                    regress(train[c].values, train['Body Fat Percentage'].values))
```

```
Out[39]: 4.52611272089604
```

```
In [40]: another_two_variable_columns = ['Abdomen Circumference', 'Age']
         c = another_two_variable_columns
         performance(val[c].values, val['Body Fat Percentage'].values,
                                    regress(train[c].values, train['Body Fat Percentage'].values))
```

```
Out[40]: 4.300874979416831
```

9.3 14 variable performance

```
In [41]: fourteen_variable_columns = column_names[:-1]
         c = fourteen_variable_columns
         performance(val[c].values, val['Body Fat Percentage'].values,
                                    regress(train[c].values, train['Body Fat Percentage'].values))
```

```
Out[41]: 4.265218559119979
```

Just goes to show the value of a validation set…

9.4 "Best guess" variables

```
In [42]: best_guess_columns = ['Abdomen Circumference', 'Wrist Circumference', 'Height']
         c = best_guess_columns
         performance(val[c].values, val['Body Fat Percentage'].values,
                                    regress(train[c].values, train['Body Fat Percentage'].values))
```

```
Out[42]: 4.223629635987778
```

9.5 Linear in parameters

```
In [43]: linear_in_parameters_columns = ['Abdomen Circumference', 'Weight^-1', 'Height', 'Wris
         c = linear_in_parameters_columns
         performance(val[c].values, val['Body Fat Percentage'].values,
                                   regress(train[c].values, train['Body Fat Percentage'].values))

Out[43]: 4.189345933006309
```

10 Statistical Perspective

```
In [44]: s = train['Body Fat Percentage'].std()

In [45]: s

Out[45]: 7.902927289914957

In [46]: s/2

Out[46]: 3.9514636449574785

In [47]: 6.84 / s

Out[47]: 0.8655020790496991

In [48]: 3.87 / s

Out[48]: 0.4896919657781193

In [49]: from scipy.stats import norm

In [50]: norm.cdf(0.5) - norm.cdf(-0.5)

Out[50]: 0.38292492254802624

In [51]: norm.cdf(1.0) - norm.cdf(-1.0)

Out[51]: 0.6826894921370859

In [52]: norm.cdf(6.84 / s) - norm.cdf(-6.84 / s)

Out[52]: 0.6132367246886898
```

11 Reproducing Penrose

```
In [54]: penrose_columns = [
             'Weight', 'Age', 'Age^2',
             'Height', 'Abdomen Circumference',
             'Wrist Circumference'
         ]
         c = penrose_columns
         performance(val[c].values, val['Body Fat Percentage'].values,
                                   regress(train[c].values, train['Body Fat Percentage'].values))

Out[54]: 4.3673819767976205
```