In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.collections as mc
import gzip
```

Multiple linear regression on IQ data followed by PCA, FA and LDA

The IQ data contains data of countries about the IQ level, socio economic, health related, and climate related variables. If we want to see if these variables have a certain relatioship with IQ we must consider IQ as the dependent variable and the other variables as independent. We can perform multiple linear regression on this dataset to fit a model between IQ and the independent variables and see if their is a relationship is.

I would expect that socio economic variables would have a positive effect on the IQ level. As a better income and a better rights, like the right on education would positively effect the IQ level. A higher income gives an individual a higher chance of getting to college.

In [2]:

```
# Loading the data in
IQ_data = pd.read_csv("IQ(1).dat", sep="\s+")
IQ_data.head()
```

Out[2]:

	Country	IQ	Income	Expenditure	Temperature	Length	Body	Percentage	Volume	BMI	...	Stability	Rights
1	Singapore	108	25407	905.0	31.5	11.53	172	6.70	126	23.2	...	92	89
2	South_Korea	106	13710	520.0	18.2	9.66	174	5.55	146	23.1	...	75	77
3	Japan	105	36785	1242.0	19.8	10.92	171	6.39	179	21.9	...	90	98
4	Switzerland	102	49994	2465.0	12.7	14.35	178	8.06	359	23.8	...	97	100
5	Netherlands	102	33383	1681.0	14.3	15.87	183	8.67	801	25.3	...	86	100

5 rows × 26 columns

In [3]:

```
# Checking for NaNs
for column in IQ_data.columns:
    percentage = 100 / len(IQ_data[column]) * len(IQ_data[column].dropna())
    print(f"{column:>12}:  {percentage:.2f}%")

#expenditure does contain some NAs
```

```
     Country:  100.00%
          IQ:  100.00%
      Income:  100.00%
 Expenditure:  98.11%
 Temperature:  100.00%
      Length:  100.00%
        Body:  100.00%
  Percentage:  100.00%
      Volume:  100.00%
         BMI:  100.00%
    M_Height:  100.00%
    M_Weight:  100.00%
       M_BMI:  100.00%
    F_Height:  100.00%
    F_Weight:  100.00%
       F_BMI:  100.00%
   Stability:  100.00%
      Rights:  100.00%
      Health:  100.00%
    Security:  100.00%
     Climate:  100.00%
       Costs:  100.00%
  Popularity:  100.00%
       Total:  100.00%
       code2:  100.00%
       code3:  100.00%
```

In [4]:

```
# Dropping non-numerical data, later this can be used for labeling data points
IQ_values = IQ_data.drop(columns=["Country","code2","code3"])
# Checking if the rest of the data is numerical
print(IQ_values.dtypes)
y = IQ_values.IQ
# Removing regressand from regressors
IQ_values = IQ_values.drop(columns=["IQ"])
```

```
IQ              int64
Income          int64
Expenditure   float64
Temperature   float64
Length        float64
Body            int64
Percentage    float64
Volume          int64
BMI           float64
M_Height      float64
M_Weight      float64
M_BMI         float64
F_Height      float64
F_Weight      float64
F_BMI         float64
Stability       int64
Rights          int64
Health          int64
Security        int64
Climate         int64
Costs           int64
Popularity      int64
Total           int64
dtype: object
```

In [5]:

```
if IQ_values["Expenditure"].isnull().values.any():
           y = y[~np.isnan(IQ_values["Expenditure"].values)]
           IQ_values["Expenditure"] = IQ_values["Expenditure"][IQ_values["Expenditure"].notna()]
           print(len(y))
```

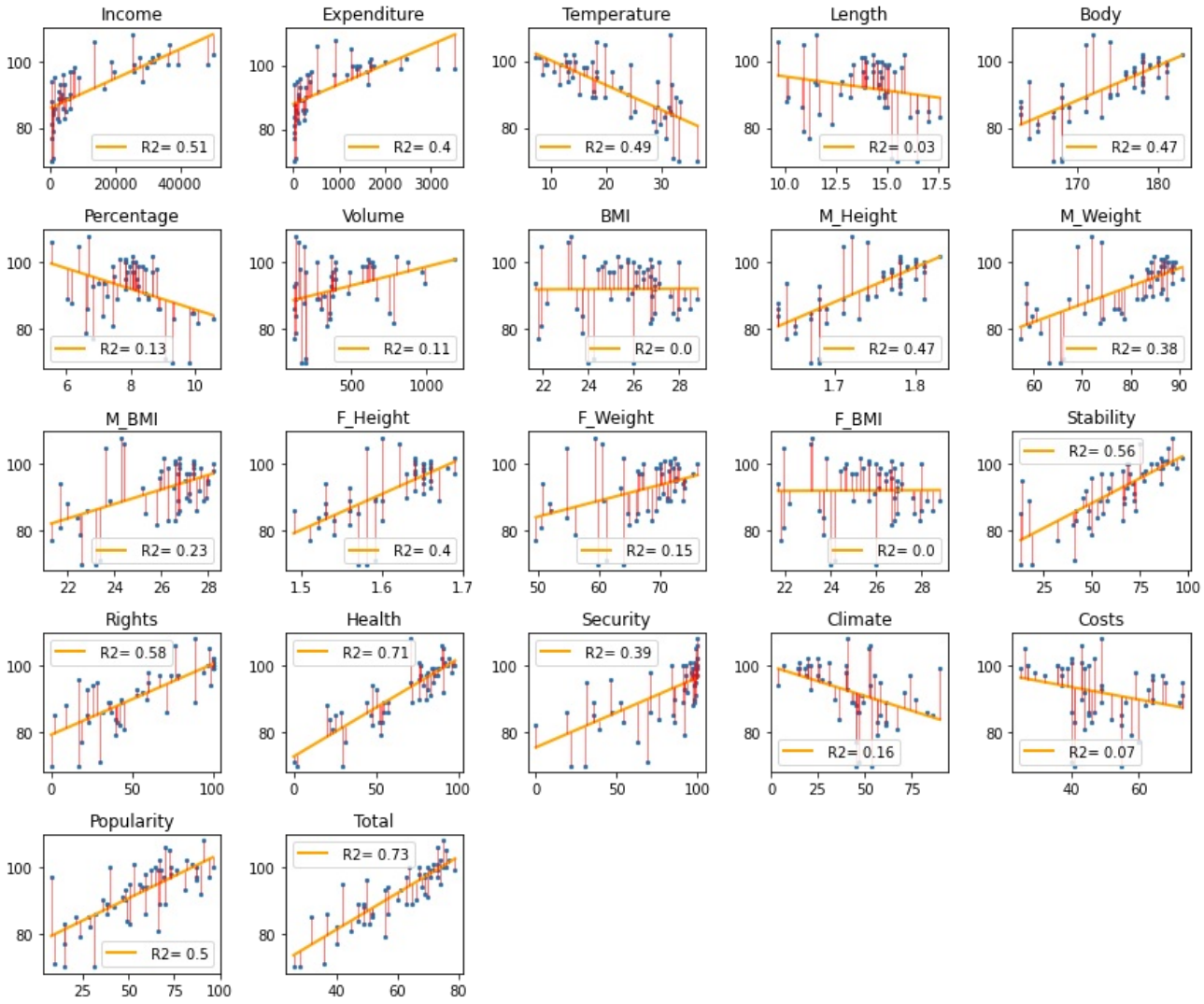
```
52
```

Checking if there is a linear relationship between the variables and IQ using simple linear regression.

In [6]:

```
fig, axes = plt.subplots(nrows=5, ncols=5,figsize=(12,10))
variable_names = [i for i in IQ_values.columns]
residuals_all = []
for i, ax in enumerate(axes.flatten()):
    try:
        if IQ_values[variable_names[i]].isnull().values.any() == True:
            print(variable_names[i]," contains NaNs. NaNs are removed in the x variable and the corresponding y-v
alue")
            x = IQ_values[variable_names[i]].dropna()
            y = y[~np.isnan(IQ_values[variable_names[i]].values)]
        else:
            y = IQ_data["IQ"]
            x = IQ_values[variable_names[i]]
        X = np.stack((np.ones(len(x)),x),axis=1)
        a, b = np.linalg.inv(X.T @ X)  @ X.T @ y
        r_xs = np.linspace(x.min(), x.max())
        regression_ys = a + b * x
        RSS = np.sum(np.square(y -regression_ys))
        TSS = np.sum(np.square(y-np.mean(y)))
        residuals = y - regression_ys
        residuals_all.append(residuals)
        # Regression line
        R2 = 1 - (RSS/TSS)
        ax.plot(r_xs,a+b * r_xs, c="orange", linewidth=2 ,label = f" R2= {round(R2,2)}")
        ax.scatter(x, y,s=5)
        ax.set_title(variable_names[i])
        ax.vlines(x,y,y-residuals,color="red",linewidth=0.5)
        ax.legend()
    except IndexError:
        fig.delaxes(ax)
fig.tight_layout()
plt.show()
```

```
Expenditure  contains NaNs. NaNs are removed in the x variable and the corresponding y-value
```



In [7]:

```
# It seems that BMI and F_BMI is actually the same thing, one should be removed, however,
# they do not show a relations ship because the R2 is 0. So we will discard them both.
uniq_comparison = str(*np.unique(IQ_values["BMI"] == IQ_values["F_BMI"]))
if uniq_comparison == "True":
    print("Column is a duplicate")
```

```
Column is a duplicate
```

Income and expenditure are showing a clear trend that is not linearly related to IQ but rather follows a square root relationship. Actually this makes sense, IQ level reaches a plateau phase, I think this has to do with the IQ-test scoring, at some point the score can't get any higher. First I thought that I needed to discard these variables, but after assessing this stack exchange link (https://stats.stackexchange.com/questions/412414/should-one-drop-independent-variables-if-they-dont-have-linear-relationship-wit) I do think differently. There seems to be a discussion about the question if an independent variable does not follow a linear relationship with the dependent varibale it should be removed or not. Let's include it in our multiple linear regression model and try without it to see the performance.

In [8]:

```
print(IQ_values.columns)
IQ_values_dropped = IQ_values.drop(columns=["BMI", "F_BMI"])
```

```
Index(['Income', 'Expenditure', 'Temperature', 'Length', 'Body', 'Percentage',
       'Volume', 'BMI', 'M_Height', 'M_Weight', 'M_BMI', 'F_Height',
       'F_Weight', 'F_BMI', 'Stability', 'Rights', 'Health', 'Security',
       'Climate', 'Costs', 'Popularity', 'Total'],
      dtype='object')
```

It is an assumption for linear regression that the residuals of variables of interest are normally distributed around zero. If this is not the case. We should not include the respective variable(s).

In [9]:

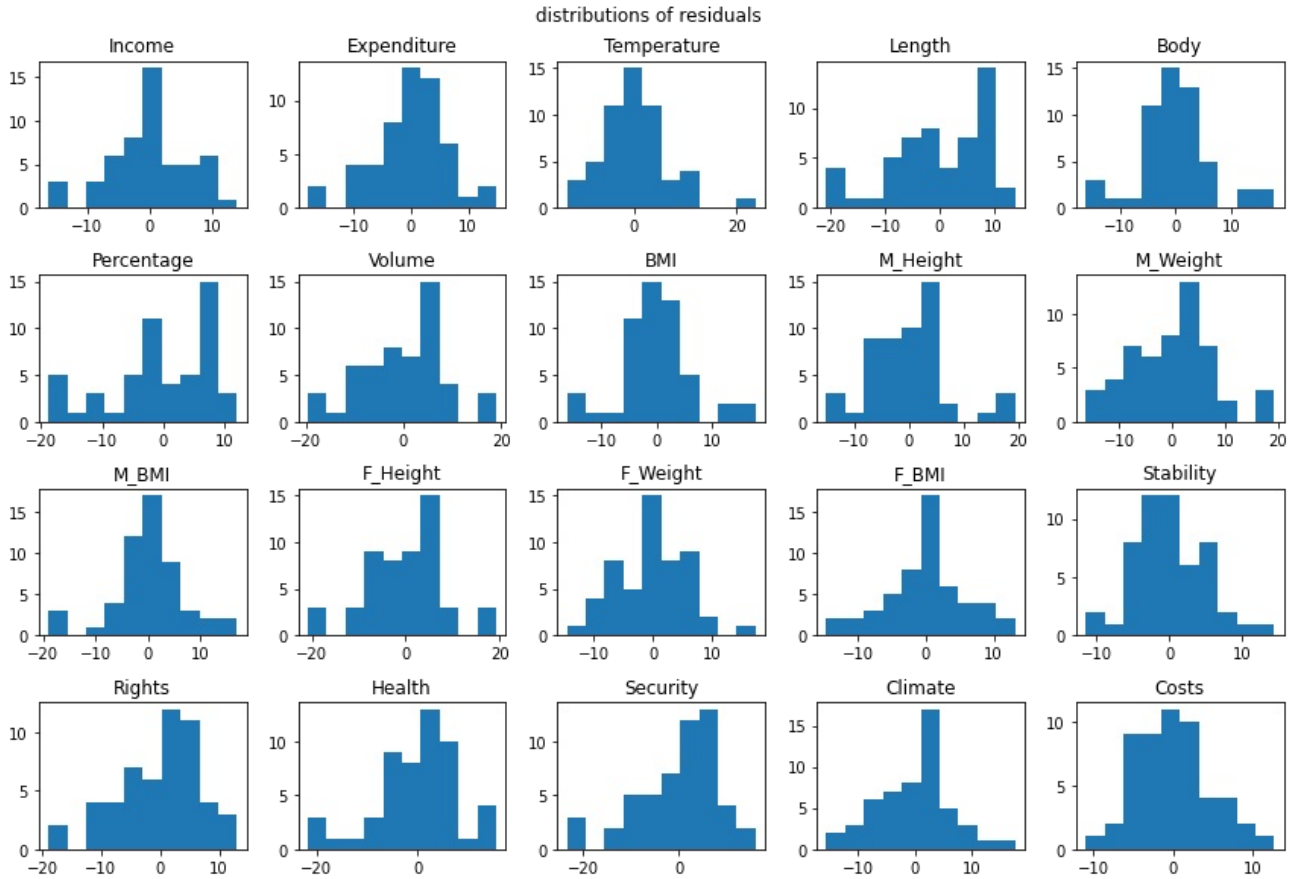
```
# Deleting BMI and F_BMI from the collection of residuals
indexes = [7,13]
for index in sorted(indexes, reverse=True):
    del residuals_all[index]
len(residuals_all)
```

Out[9]:

```
20
```

In [10]:

```
fig, axes = plt.subplots(nrows=5, ncols=5,figsize=(12,10))
for i, ax in enumerate(axes.flatten()):
    try:
        ax.hist(residuals_all[i],bins=10)
        ax.set_title(IQ_values.columns[i])
    except ValueError:
        fig.delaxes(ax)
    except IndexError:
        fig.delaxes(ax)
fig.suptitle("distributions of residuals")
plt.tight_layout()
plt.show()
```



In the plots above we can see that the residuals of the predicted y are to some extend normal distributed. Let's include these variables in the multi linear regression model to check if the variables are related to IQ.

In [11]:

```
# For dropping in np.stack
copuu = []
for name in IQ_values_dropped.columns:
    copuu.append(f"IQ_values.{name},")
" ".join(copuu)
```

Out[11]:

```
'IQ_values.Income, IQ_values.Expenditure, IQ_values.Temperature, IQ_values.Length, IQ_values.Body, I
Q_values.Percentage, IQ_values.Volume, IQ_values.M_Height, IQ_values.M_Weight, IQ_values.M_BMI, IQ_v
alues.F_Height, IQ_values.F_Weight, IQ_values.Stability, IQ_values.Rights, IQ_values.Health, IQ_valu
es.Security, IQ_values.Climate, IQ_values.Costs, IQ_values.Popularity, IQ_values.Total,'
```

In [12]:

```
# All variables, so with Income and Expenditure
# We have to delete the NaN last row unfortunately, due to an NaN.
IQ_values = IQ_values_dropped.dropna()
X = np.stack((np.ones(IQ_values.shape[0]),IQ_values.Income,IQ_values.Expenditure,IQ_values.Temperature, IQ_values
.Length, IQ_values.Body, IQ_values.Percentage, IQ_values.Volume, IQ_values.M_Height, IQ_values.M_Weight, IQ_value
s.M_BMI, IQ_values.F_Height, IQ_values.F_Weight, IQ_values.Stability, IQ_values.Rights, IQ_values.Health, IQ_valu
es.Security, IQ_values.Climate, IQ_values.Costs, IQ_values.Popularity, IQ_values.Total),axis=1)
y = IQ_data.IQ[:52]
b = np.linalg.inv(X.T @ X) @ X.T @ y
yhat = X @ b
residuals = y - yhat
R2 = 1 - yhat.var() / y.var()
print(f"R2= {R2:.2f}")
```

```
R2= -14.54
```

Using these features in the model does not seem to have any relationship with IQ because of the low R2

In [13]:

```
# Using all samples but without expenditure and income, we can use all samples because expenditure does contain
# a NaN and is not included here.
IQ_values = IQ_values_dropped
X = np.stack((np.ones(IQ_values.shape[0]),IQ_values.Temperature, IQ_values.Length, IQ_values.Body, IQ_values.Perc
entage, IQ_values.Volume, IQ_values.M_Height, IQ_values.M_Weight, IQ_values.M_BMI, IQ_values.F_Height, IQ_values.
F_Weight, IQ_values.Stability, IQ_values.Rights, IQ_values.Health, IQ_values.Security, IQ_values.Climate, IQ_valu
es.Costs, IQ_values.Popularity, IQ_values.Total),axis=1)
y = IQ_data.IQ[:len(IQ_values)]
b = np.linalg.inv(X.T @ X) @ X.T @ y
yhat = X @ b
residuals = y - yhat
R2 = 1 - yhat.var() / y.var()
print(f"R2= {R2:.2f}")
```

```
R2= -0.29
```

When using all features the model shows to have no relationship with IQ. The R2 score suggests that the model fitted is worse slightly worse than a horizontal line.

So removing the two non-linear related variables did some improvement although the fit is still nog performing well. Meaning that when we use all of these variables there seems no relationship.

In [14]:

```
# Socio economic variables, we can use all samples because expenditure does contain
# a NaN and is not included here.
IQ_values = IQ_values_dropped
X = np.stack((np.ones(IQ_values.shape[0]),IQ_values.Income,IQ_values.Stability, IQ_values.Rights,IQ_values.Health
, IQ_values.Security, IQ_values.Costs, IQ_values.Popularity),axis=1)
y = IQ_data.IQ[:len(IQ_values)]
b = np.linalg.inv(X.T @ X) @ X.T @ y
yhat = X @ b
residuals = y - yhat
R2 = 1 - yhat.var() / y.var()
print(f"R2= {R2:.2f}")
```

```
R2= 0.17
```

When using socio economic features the model does seem to detect very a weak relationship. 17% percent gets explained by the socio economic variables, I think that health is also part of a socio economic feature, as health can be related to rights. For example, the right to have free healthcare may increase health of the individual.

In [15]:

```
# Health related variables,we can use all samples because expenditure does contain
# a NaN and is not included here.
IQ_values = IQ_values_dropped
X = np.stack((np.ones(IQ_values.shape[0]), IQ_values.Length, IQ_values.Health,IQ_values.Body, IQ_values.Volume, I
Q_values.M_Height, IQ_values.M_Weight, IQ_values.M_BMI, IQ_values.F_Height, IQ_values.F_Weight),axis=1)
y = IQ_data.IQ[:len(IQ_values)]
b = np.linalg.inv(X.T @ X) @ X.T @ y
yhat = X @ b
residuals = y - yhat
R2 = 1 - yhat.var() / y.var()
print(f"R2= {R2:.2f}")
```

```
R2= -0.37
```

Health related variables does not seem to be related to IQ as the R2 score is low.

In [16]:

```
# Climate related variables
X = np.stack((np.ones(IQ_values.shape[0]),IQ_values.Temperature, IQ_values.Climate),axis=1)
y = IQ_data.IQ
b = np.linalg.inv(X.T @ X) @ X.T @ y
yhat = X @ b
residuals = y - yhat
R2 = 1 - yhat.var() / y.var()
print(R2)
```

```
0.5227121677546432
```

This surprises me that climate variables seems to be related to IQ. in the first figure simple regression is used to fit a line between the features and IQ. This shows that the higher the temperature the lower the IQ. However this seems to be very biased. As in hotter continents, such as Africa, the quality of education is often not as high as in western countries. In additon, education costs a substantial amount of money often individuals do not have this kind of money in hotter continents such as Africa. A study performed in 2018 by Ritchie & Tucker-Drob revelead that an additional year of education shows to improve IQ with 1 - 5 points. Meaning that education influences the IQ in a positive manner.

Overall I think it is overall very hard to assess the relationship between IQ and the features provided here using multi linear regression. We can try to perform PCA in order to identify clusters, expected is that we can identify clusters of with high income and low income. Or at least a seperation of countries having a good economic status.

In [17]:

```
labels = IQ_data[["Country","code2","code3"]]
IQ = IQ_data.IQ
IQ_values = IQ_data.drop(columns=["IQ","Country","code2","code3"]).dropna()
```

Before performing PCA, it must be considered if the data needs to be normalised or not. Here, it is necessary due to the fact that the variables are heterogenous. I.e., length is different from income.

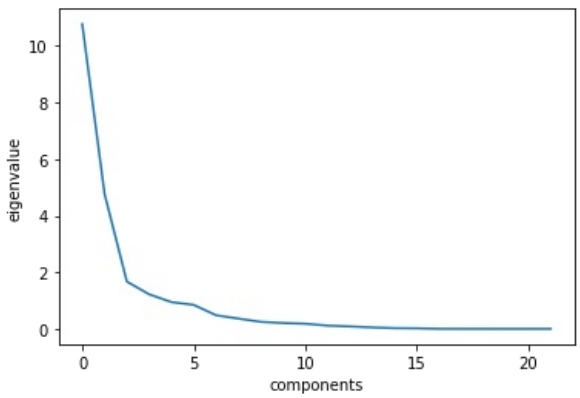
In [18]:

```
# Extract values of IQ without IQ included
values = IQ_values
# the data consistst of 52 samples and 22 variables
print(values.shape)
# Normalisation
X = (values - values.mean(axis=0)) / values.std(axis=0)
# Extract eigenvalues and eigenvectors
vals, vecs = np.linalg.eigh(np.cov(X.T))
order = vals.argsort()[::-1]
vals = vals[order]
vecs = vecs[:,order]
```

```
(52, 22)
```

In [19]:

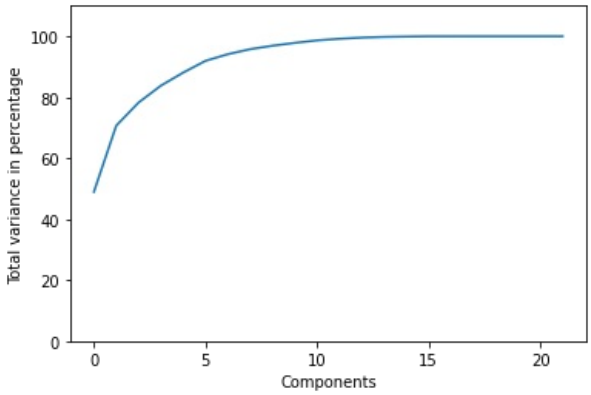
```
# The first two components explain most of the variance
plt.plot(vals)
plt.xlabel("components")
plt.ylabel("eigenvalue")
plt.show()
```



The first two components explain most of the variance.

In [20]:

```
# First two components explain 70% of the variance
plt.plot(100* vals.cumsum()/sum(vals))
plt.ylim(0,110)
plt.xlabel("Components")
plt.ylabel("Total variance in percentage")
plt.show()
```



The first two components explain 70% of the variance

In [21]:

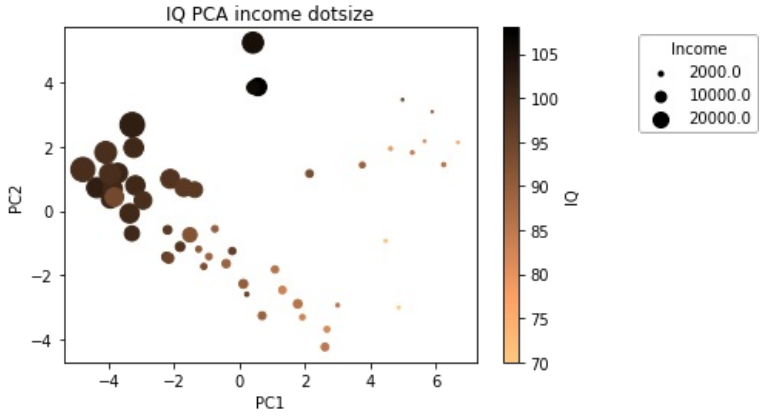
```
projections = np.dot(X, vecs)
```

In [22]:

```
income = IQ_values.Income
sizes_income = income / 200
```

In [23]:

```
# plt.figure(figsize=(10,8))
sc = plt.scatter(projections[:,0],projections[:,1], s= sizes_income, c=IQ[:52],cmap="copper_r")
plt.title("IQ PCA income dotsize")
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.colorbar(sc,label="IQ")
for x,s in zip([10,50,100],[2e3, 1e4, 2e4]):
    plt.scatter([], [], c="black", alpha=1, s=x, label=str(s))
legend = plt.legend(title="Income",bbox_to_anchor=(1.7, 1))
legend.get_frame().set_alpha(None)
plt.show()
```



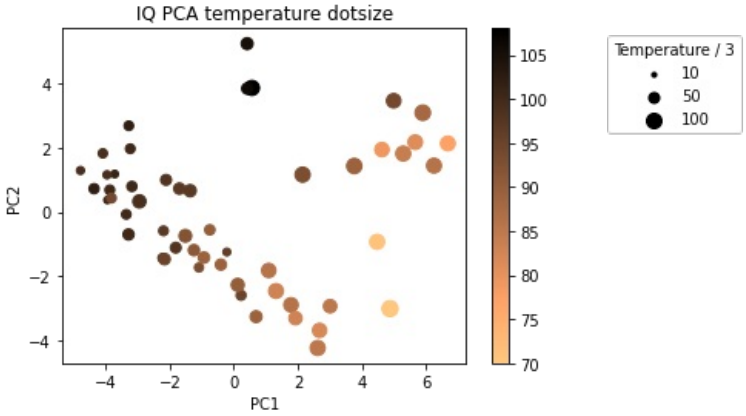
In the figure above shows the projections on PC1 and PC2. Here, the size of the dot represents the income. The bigger the income, the bigger the dot. The colors represent the IQ level. The darker the color the higher the IQ of the corresponding population is. We can clearly see that low incomes have a lower IQ. The high incomes have a greater IQ.However, this must also be correlated to education. And also to rights this is again to some extend related to health. E.g., the health of an individual again. E.g., the right of healthcare.

In [24]:

```
# Extract temperature for sizes of the dots
sizes_temp = IQ_values.Temperature *3
```

In [25]:

```
plt.axes().set_aspect("equal")
sc = plt.scatter(projections[:,0],projections[:,1], s= sizes_temp, c=IQ[:52],cmap="copper_r")
plt.title("IQ PCA temperature dotsize")
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.colorbar(sc)

for x in [10, 50, 100]:
    plt.scatter([], [], c="black", alpha=1, s=x, label=str(x))
plt.legend(title="Temperature / 3",bbox_to_anchor=(1.7, 1))
legend.get_frame().set_alpha(None)
plt.show()
```



In the figure above the projections of PC1 and PC2 are shown. The dot sizes represent temperature. What can be seen here is that smaller dots seem to be forming on the left side of PC1 and larger dots on the right side. Meaning that temperature is highly accociated with PC1. It seems to be that the higher the temperature the lower the IQ.

Again, Africa has to deal with great poverty and as a consequence the development of an individual is negatively affected by this, affecting the IQ.

Templer and Arikawa (https://www.sciencedirect.com/science/article/pii/S0160289605000917?via%3Dihub) supports this.

"*The great poverty of Africa produces conditions that can unquestionably lower IQ such as malnutrition, disease, inferior prenatal care, and inferior perinatal care. The incidence of epilepsy is very high in Africa, and this has been attributed to malnutrition, parasitic and other brain infections, febrile convulsions, and birth injuries*"

In addition Templer and Arikawa used MLR to see if there is a relationship between IQ and GDP, pigment color and temperature. They are stating that pigment color is highly correlated to climate. Maybe we can use income, climate temperature to perform multiple regression to validate this is the case with our data.

When performing factor analysis I expect that health, rights and income will somewhat point in the same direction as I expect that these are highly correlated variables. I think climate and temperature will be pointing to the same direction as these are highly correlated but do not have much to do with the other variables.

In [26]:

```
# MLR using Templer and Arikawa-like variables
# here the last sample can also be used as Nigeria does not contain Nans in cncome, climate and temperature
X = np.stack((np.ones(IQ_data.shape[0]),  IQ_data.Income,IQ_data.Climate,IQ_data.Temperature),axis=1)
y = IQ_data.IQ
b = np.linalg.inv(X.T @ X) @ X.T @ y
yhat = X @ b
residuals = y - yhat
R2 = 1 - yhat.var() / y.var()
print(R2.round(3))
```

```
0.384
```

38.4% of variance is explained by this. So it tells us something but not as much as in the paper of Templer and Arikawa. They had an R2 of 0.841 to 0.850. They had over double the amount of countries included in their analysis and this could explain why their model does perform better than ours. And there variables were slightly different compared to ours, they used minimum winter temperature and maximum summer temperature.

In [27]:

```
def varimax(components, gamma=1, maxiter=20, tol=1e-8):
    """Perform VariMax (gamma=1) or OrthoMax (gamma=0) rotation on components"""
    p,k = components.shape
    R   = np.eye(k)
    f   = float(gamma)/p
    d   = 0
    for i in range(maxiter):
        d_old = d
        L = np.dot(components, R)
        A = L**3 - f * (L*(L**2).sum(axis=0))**2
        U,s,V = np.linalg.svd(np.dot(components.T,A))
        R = np.dot(U,V)
        d = sum(s)
        if (d - d_old)**2 < tol:
            break
    return np.dot(components, R)
```

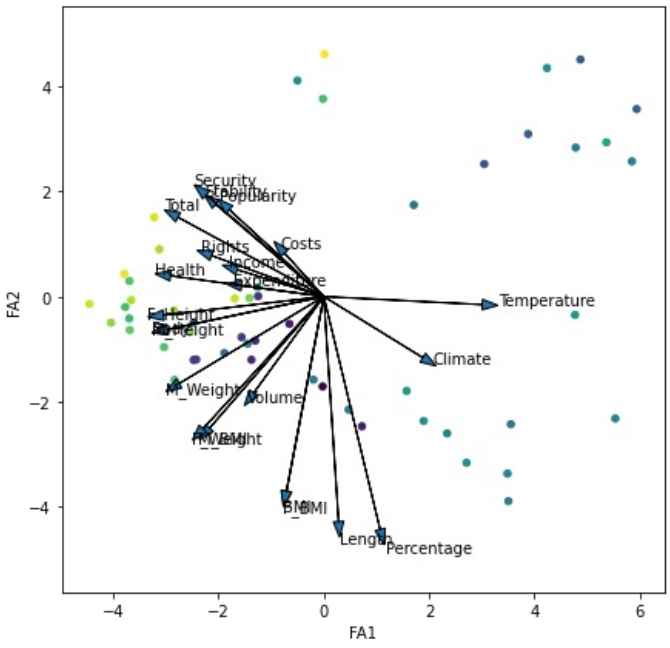
In [28]:

```
# Extract values of IQ without IQ included
values = IQ_values
# the data consistst of 52 samples and 22 variables
print(values.shape)
# Normalisation
X = (values - values.mean(axis=0)) / values.std(axis=0)
```

```
(52, 22)
```

In [29]:

```
fig, ax = plt.subplots(figsize=(7,7))
ax.axis("equal")
V = varimax(vecs[:,:3])

fa_projections = np.dot(X, V)
# plt.axis("equal")
ax.scatter(fa_projections[:,0], fa_projections[:,1], c=fa_projections[:, 2], s=20)

ax.set_xlabel("FA1")
ax.set_ylabel("FA2")
for var, (x,y) in zip(IQ_values.columns, 10*V[:, :2]):
    ax.arrow(0, 0, x, y, head_width=0.2)
    ax.text(1.1*x, 1.1*y, var)
plt.show()
```



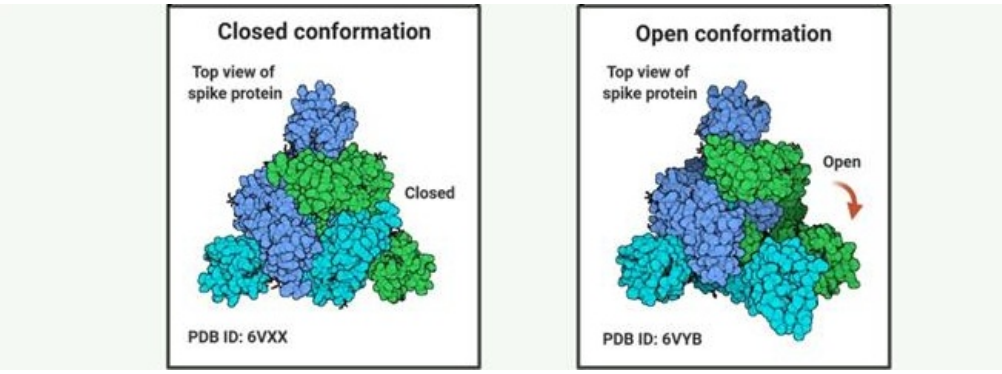
It is much as I had expected Rights income and health point to the same direction as is temperature and climate. So PC1 is clearly associated with temperature. Length seems to be related the second component, which is strange. But maybe also logical, maybe longer people have bigger brains. However, wouldn't it be more logical to look at brain sizes instead of the length of individuals. This paper (https://journals.sagepub.com/doi/10.1177/0956797618808470) states that brain size volume is positively related to IQ. But maybe that is what is present in the column volume, unfortunately I do not know what this means, total volume of the heart, body, gut? I could not find it on the website the data is coming from

Muli dimensional scaling on SARS-CoV2-Spike

The data of SARS-Cov2 Spike proteins are given in the format of a distance dissimilarity matrix of open versus closed. For this reason we can try to visualize the open versus the close conformations using multi dimensional scaling. The data given is already a dissimilarity matrix I think based on xyz values of open conformations and closed conformations of this spike protein.

The spike can bind to ACE2 using S1 this can bind tightly to ACE2 and place plays a key role in the invasion of this virus in human cells. In the figure below it seems that a slight change can be observed. The right blue part seems change a bit lower.

So the expectations using that with MDS we could see visualize this kind of representation.



(https://www.ijbs.com/v17p0097.htm (https://www.ijbs.com/v17p0097.htm))

In [30]:

```
cov2 = np.loadtxt("SARS-CoV2-Spike-closed-vs-open-distance-difference.dat")
cov2.shape
```

Out[30]:

```
(2991, 2991)
```

In [31]:

```
# No NaNs present in data
print(f"{len(cov2[np.isnan(cov2)])} NaN present")
```

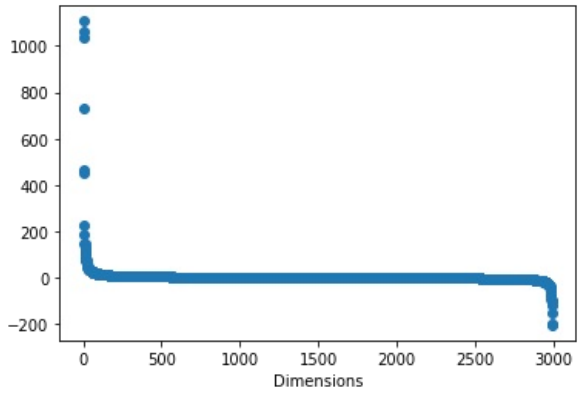
```
0 NaN present
```

In [32]:

```
M = cov2.mean(axis=0)
A = -0.5 * (cov2-M[:,None] - M[None,:] +M.mean())
# A = -0.5 * (distmat - m[None, :] - m[:, None] + m.mean())
vals, vecs = np.linalg.eigh(A)
```

In [33]:

```
# there are 6 important dimensions
order = vals.argsort()[::-1]
plt.scatter(range(1, len(vals)+1), vals[order])
plt.xlabel("Dimensions")
plt.show()
```



In [34]:

```
X = vecs[:, order] * np.power(vals[order],0.5)
```

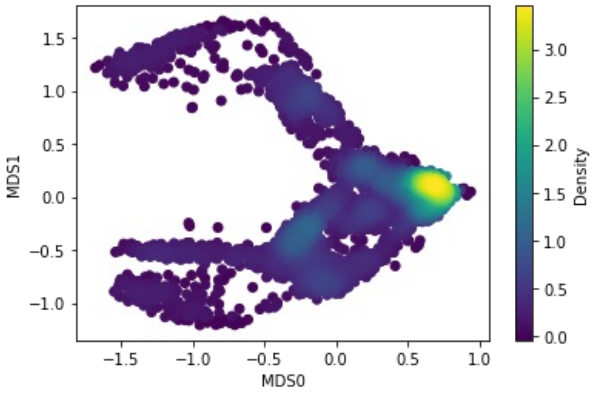
```
<ipython-input-34-b907b5e693d6>:1: RuntimeWarning: invalid value encountered in power
  X = vecs[:, order] * np.power(vals[order],0.5)
```

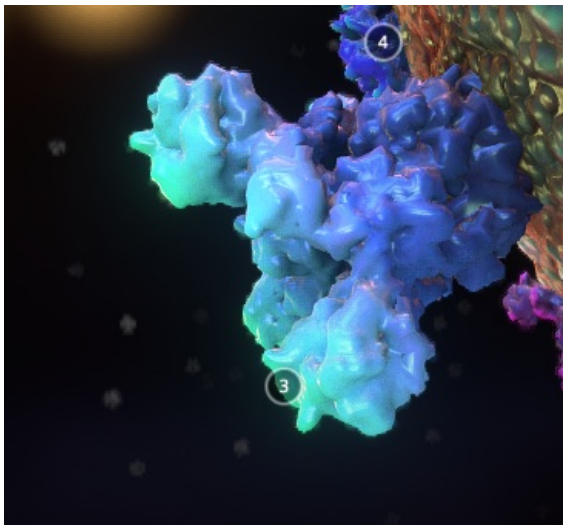
In [35]:

```
# Code from
# https://stackoverflow.com/questions/20105364/how-can-i-make-a-scatter-plot-colored-by-density-in-matplotlib
# with this function we can easily identify dens regions in scatter plots

from matplotlib import cm
from matplotlib.colors import Normalize
from scipy.interpolate import interpn
def density_scatter( x , y, ax = None, sort = True, bins = 20, title=None ,**kwargs )   :
    """
    Scatter plot colored by 2d histogram
    """
    if ax is None :
        fig , ax = plt.subplots()
        ax.set_title(title)
    data , x_e, y_e = np.histogram2d( x, y, bins = bins, density = True )
    z = interpn( ( 0.5*(x_e[1:] + x_e[:-1]) , 0.5*(y_e[1:]+y_e[:-1]) ) , data , np.vstack([x,y]).T , method = "sp
linef2d", bounds_error = False)

    #To be sure to plot all data
    z[np.where(np.isnan(z))] = 0.0

    # Sort the points by density, so that the densest points are plotted last
    if sort :
        idx = z.argsort()
        x, y, z = x[idx], y[idx], z[idx]

    ax.scatter( x, y, c=z, **kwargs )

    norm = Normalize(vmin = np.min(z), vmax = np.max(z))
    cbar = fig.colorbar(cm.ScalarMappable(norm = norm), ax=ax)
    cbar.ax.set_ylabel('Density')

    return ax
```

In [36]:

```
mds1 = density_scatter(X[:,0],X[:,1],title=None)
mds1.set_ylabel("MDS1")
mds1.set_xlabel("MDS0")

plt.show()
```

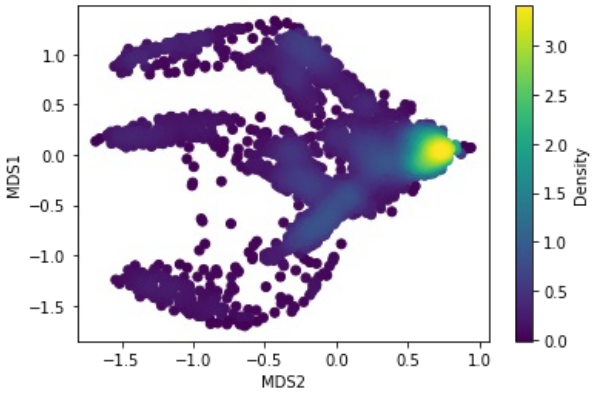


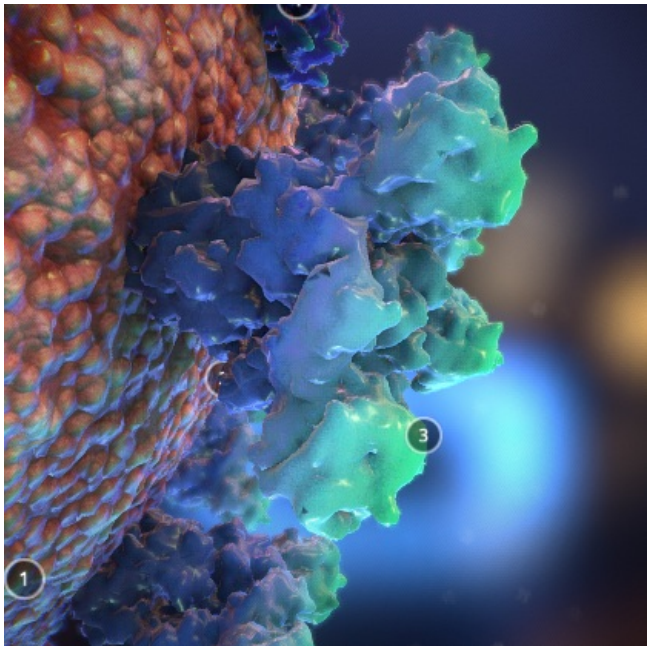


It seems that this is a side representation of the spike protein. We can clearly see three "arms" or peptides and a dens region in the middle towards the right side. This region is very dense and this could be the anchor of a receptor, anchored in the membrane. This plot suggests that the anchor is not changing in closed and open conformations.

In [37]:

```
mds2=density_scatter(X[:,0], X[:,2],title=None)
mds2.set_ylabel("MDS1")
mds2.set_xlabel("MDS2")
plt.show()
```

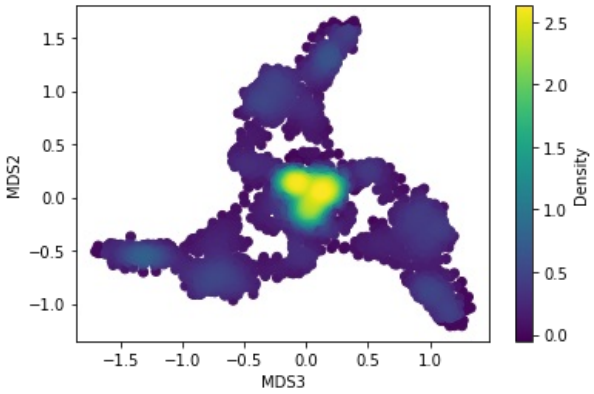


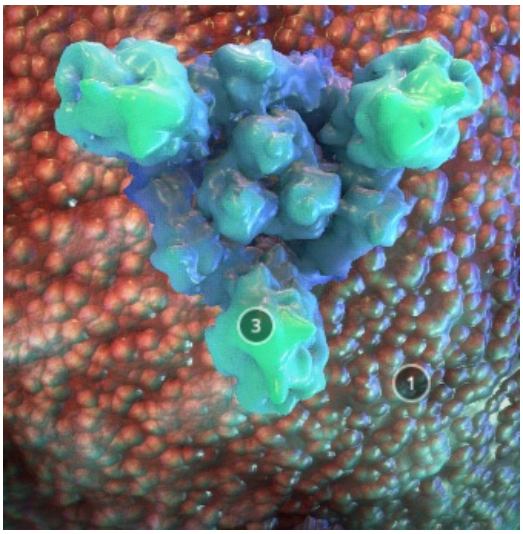


Again it looks like a similar representation of side representation of the receptor.

In [38]:

```
mds3 = density_scatter(X[:,2],X[:,1],title=None)
mds3.set_ylabel("MDS2")
mds3.set_xlabel("MDS3")
plt.show()
```





The plot above shows that in the middle there is not much difference between the distance of atoms because they are at 0 meaning that these atoms remain on the same location in open versus closed. It looks like it is the surface of the protein. In addition we can see a very dens region in the middle. This can be due to that the fact that the "root" of a receptor is often anchored within the membrane of the virus, like an anchor. What we can also see is are three "arms". And three dense regions in the anchor of the receptor

I think it is a bit hard to see where the actual RBP is located. I cannot lay the finger on the location where it is happening.

In [39]:

```
fig, ax = plt.subplots(figsize=(8,8))
ax.axis("equal")
ax.scatter(X[:,2],X[:,1], c=np.arange(0,len(cov2)) )
plt.show()
```