Analysis of urban versus rural living

In [2]:

```
#importing Libery
import warnings
warnings.filterwarnings("ignore")

import pandas as pd
import copy
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns

# styling
pd.set_option('display.max_columns',150)
plt.style.use('bmh')
from IPython.display import display
%matplotlib inline
```

Data Collection

In [3]:

```
#loading dataset in pandas dataset
data = pd.read_csv('responses.csv')
```

In [4]:
```
#check first five rows of the dataset
data.head()
```

Out[4]:

	Music	Slow songs or fast songs	Dance	Folk	Country	Classical music	Musical	Pop	Rock	Metal or Hardrock	Punk	I
0	5.0	3.0	2.0	1.0	2.0	2.0	1.0	5.0	5.0	1.0	1.0	
1	4.0	4.0	2.0	1.0	1.0	1.0	2.0	3.0	5.0	4.0	4.0	
2	5.0	5.0	2.0	2.0	3.0	4.0	5.0	3.0	5.0	3.0	4.0	
3	5.0	3.0	2.0	1.0	1.0	1.0	1.0	2.0	2.0	1.0	4.0	
4	5.0	3.0	4.0	3.0	2.0	4.0	3.0	5.0	3.0	1.0	2.0	

In [5]:
```
#check last five rows of the dataset
data.tail()
```

Out[5]:

	Music	Slow songs or fast songs	Dance	Folk	Country	Classical music	Musical	Pop	Rock	Metal or Hardrock	Pun
1005	5.0	2.0	5.0	2.0	2.0	5.0	4.0	4.0	4.0	3.0	2.
1006	4.0	4.0	5.0	1.0	3.0	4.0	1.0	4.0	1.0	1.0	4.
1007	4.0	3.0	1.0	1.0	2.0	2.0	2.0	3.0	4.0	1.0	2.
1008	5.0	3.0	3.0	3.0	1.0	3.0	1.0	3.0	4.0	1.0	1.
1009	5.0	5.0	4.0	3.0	2.0	3.0	3.0	4.0	1.0	1.0	2.

In [6]:

```
#check shape
data.shape
```

Out[6]:

```
(1010, 150)
```

In [7]:

```
#check more info mation
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1010 entries, 0 to 1009
Columns: 150 entries, Music to House - block of flats
dtypes: float64(134), int64(5), object(11)
memory usage: 1.2+ MB
```

In [8]:

```
data.describe()
```

Out[8]:

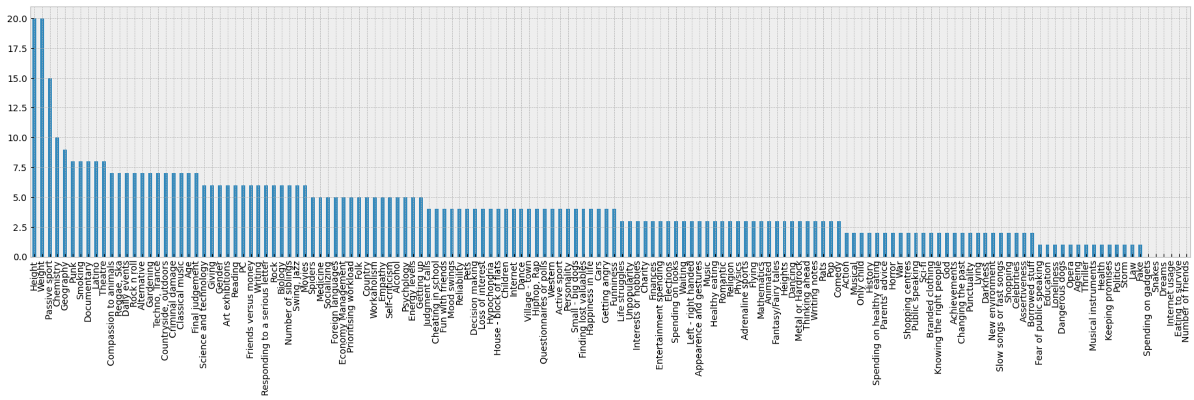
	Music	Slow songs or fast songs	Dance	Folk	Country	Classical music	
count	1007.000000	1008.000000	1006.000000	1005.000000	1005.000000	1003.000000	1008
mean	4.731877	3.328373	3.113320	2.288557	2.123383	2.956132	[illegible]
std	0.664049	0.833931	1.170568	1.138916	1.076136	1.252570	[illegible]
min	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	[illegible]
25%	5.000000	3.000000	2.000000	1.000000	1.000000	2.000000	[illegible]
50%	5.000000	3.000000	3.000000	2.000000	2.000000	3.000000	[illegible]
75%	5.000000	4.000000	4.000000	3.000000	3.000000	4.000000	[illegible]
max	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000	[illegible]

Exploratory analysis

Missing values

In [9]:
```
nulls = data.isnull().sum().sort_values(ascending=False)
nulls.plot(
    kind='bar', figsize=(23, 5))
```

Out[9]:
```
<AxesSubplot:>
```



In [11]:
```
omitted = data[(data['Weight'].isnull()) | data['Height'].isnull()]
print('Number of people with omitted weight or height: {:.0f}'.form
nas = omitted.drop(['Weight', 'Height', 'Number of siblings', 'Age'
print('Number of fields that were omitted by people who did not fil
```

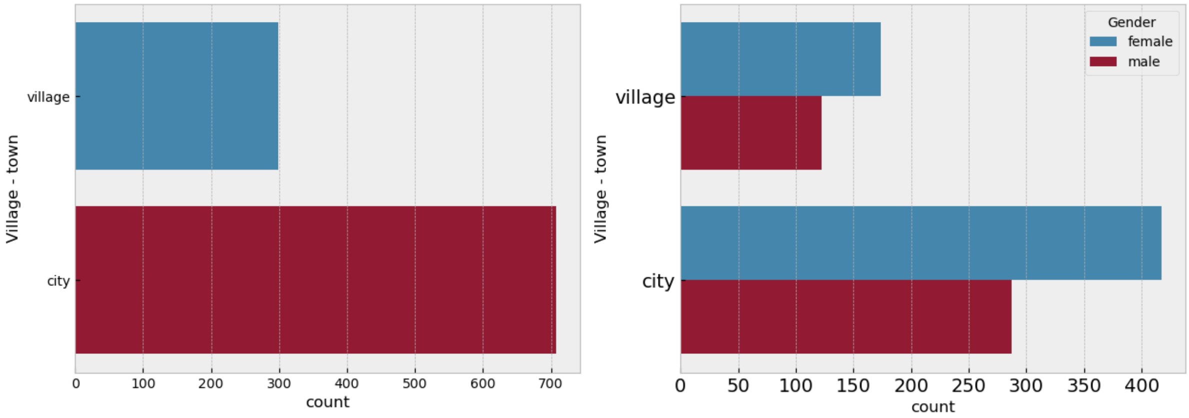
```
Number of people with omitted weight or height: 30
Number of fields that were omitted by people who did not fill Weig
ht or Height: 18
```

Understanding our goal

In [12]:
```
var_of_interest = 'Village - town'
mapping = {var_of_interest: {'city': 0, 'village': 1}}
data.dropna(subset=[var_of_interest], inplace=True)
# to be able to use hue parameter for better comparison in seaborn
data["all"] = ""
```

In [14]:
```
fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(15, 5))
sns.countplot(y=var_of_interest, data=data, ax=ax[0])
sns.countplot(y=var_of_interest, hue='Gender', data=data, ax=ax[1])
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
```

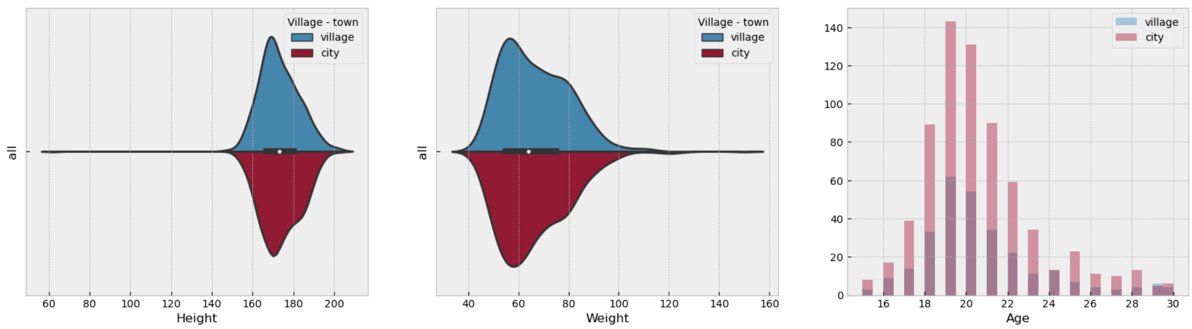
Out[14]:
```
(array([0, 1]), [Text(0, 0, 'village'), Text(0, 1, 'city')])
```



Outliers

In [15]:
```
fig, ax = plt.subplots(nrows=1, ncols=3, figsize=(20,5))
data = data.dropna(subset=['Height'])
sns.violinplot(x='Height', y = "all", hue=var_of_interest, data=dat
data = data.dropna(subset=['Weight'])
sns.violinplot(x='Weight', y = "all", hue=var_of_interest, data=dat

var_of_int_ser = data[var_of_interest]
sns.distplot(data[var_of_int_ser=='village'].Age.dropna(),
             label='village', ax=ax[2], kde=False, bins=30);

sns.distplot(data[var_of_int_ser=='city'].Age.dropna(),
             label='city', ax=ax[2], kde=False, bins=30);
ax[2].legend()
```

Out[15]:
```
<matplotlib.legend.Legend at 0x7fe0c7263ca0>
```



As we see there are some outliers that disturb the visualisation.

In [16]:
```
display(data[data['Height']<70][['Age', 'Height', 'Weight', 'Gender
display(data[data['Weight']>120][['Age', 'Height', 'Weight', 'Gende
```

	Age	Height	Weight	Gender	Village - town
676	20.0	62.0	55.0	female	city

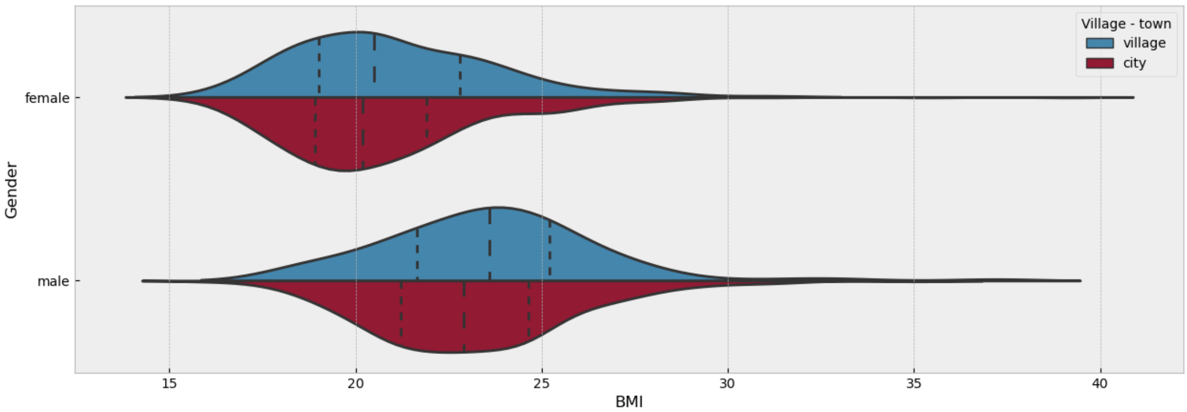
	Age	Height	Weight	Gender	Village - town
859	20.0	190.0	125.0	male	city
992	30.0	200.0	150.0	male	city

In [18]:
```
data.drop([676,992, 859], inplace = True)
```

Interestingly, there is a small secong hill in Height in city people around 185 cm. The horizontal lines are quartiles.

As TheTS mentioned in the comments, we could look at BMI of rural versus urban people. BMI is calculated as follows: weight/height^2. The hypothesis is that urban people will have a lower BMI as they might spend more times outdoors.

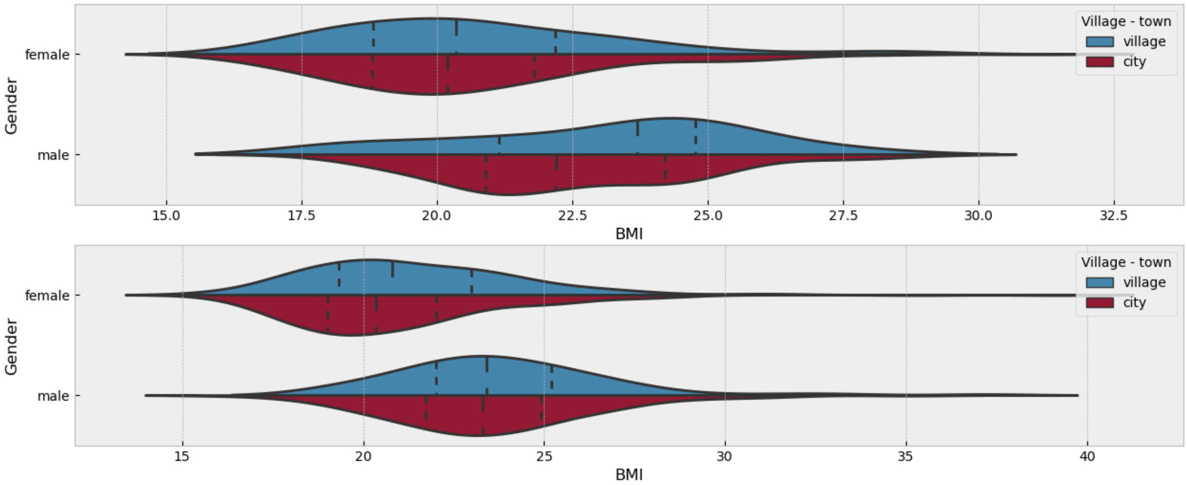
In [20]:
```
data['BMI'] = round(data['Weight']/((data['Height']/100)**2),1)
fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(15,5))
data = data.dropna(subset=['BMI'])
sns.violinplot(x='BMI', hue=var_of_interest, y='Gender', data=data,
               split=True, inner='quartile', ax=ax);
```



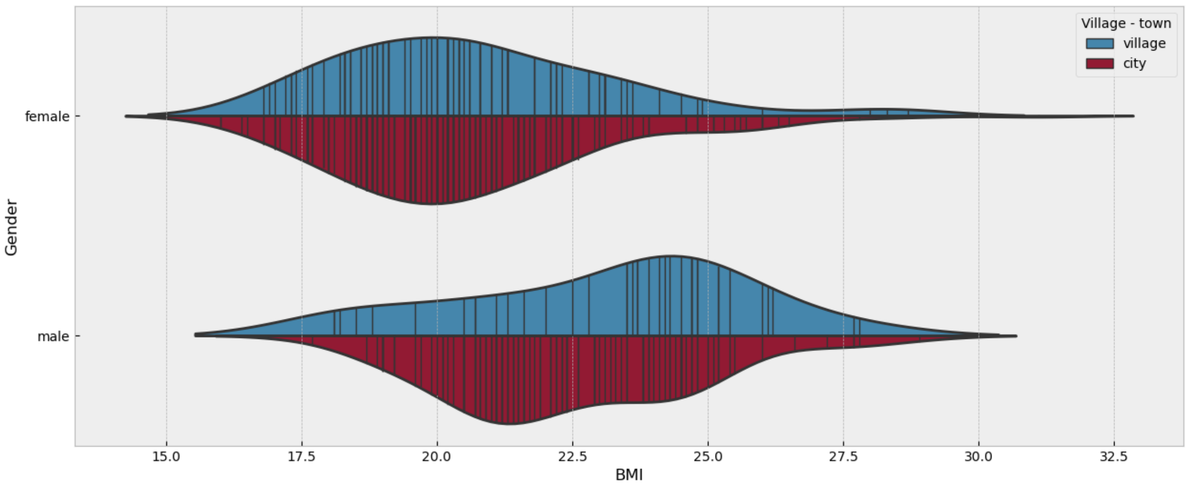
In [21]:
```
import scipy.stats as stats
city_bmi = data[data[var_of_interest]=='city'].BMI
village_bmi  = data[data[var_of_interest]=='village'].BMI
t, p = stats.ttest_ind(village_bmi, city_bmi, axis=0, equal_var=Fal
print(' t-stat = {t} \n p-value = {p}'.format(t=t,p=p/2))
```

```
 t-stat = 1.7734182239050904
 p-value = 0.03837342374443175
```

In [22]:
```
fig, ax = plt.subplots(nrows=2, ncols=1, figsize=(15,6))
data_under = data[data['Age']<20]
data_above = data[data['Age']>=20]
sns.violinplot(x='BMI', hue=var_of_interest, y='Gender', data=data_
                   inner = 'quartile', ax=ax[0], hue_order=['villag
sns.violinplot(x='BMI', hue=var_of_interest, y='Gender', data=data_
                   inner = 'quartile', ax=ax[1], hue_order=['villag
```



In [23]:
```
fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(15,6))
sns.violinplot(x='BMI', hue=var_of_interest, y='Gender', data=data_
               split=True, inner='stick', ax=ax, hue_order=['villag
```



Interesting differences

In this section we will analyze differences between the individuals based on the area of living.

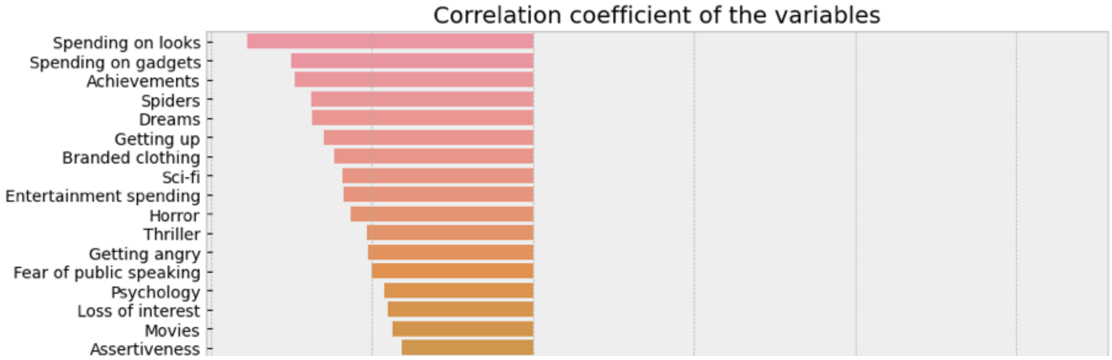
Correlation

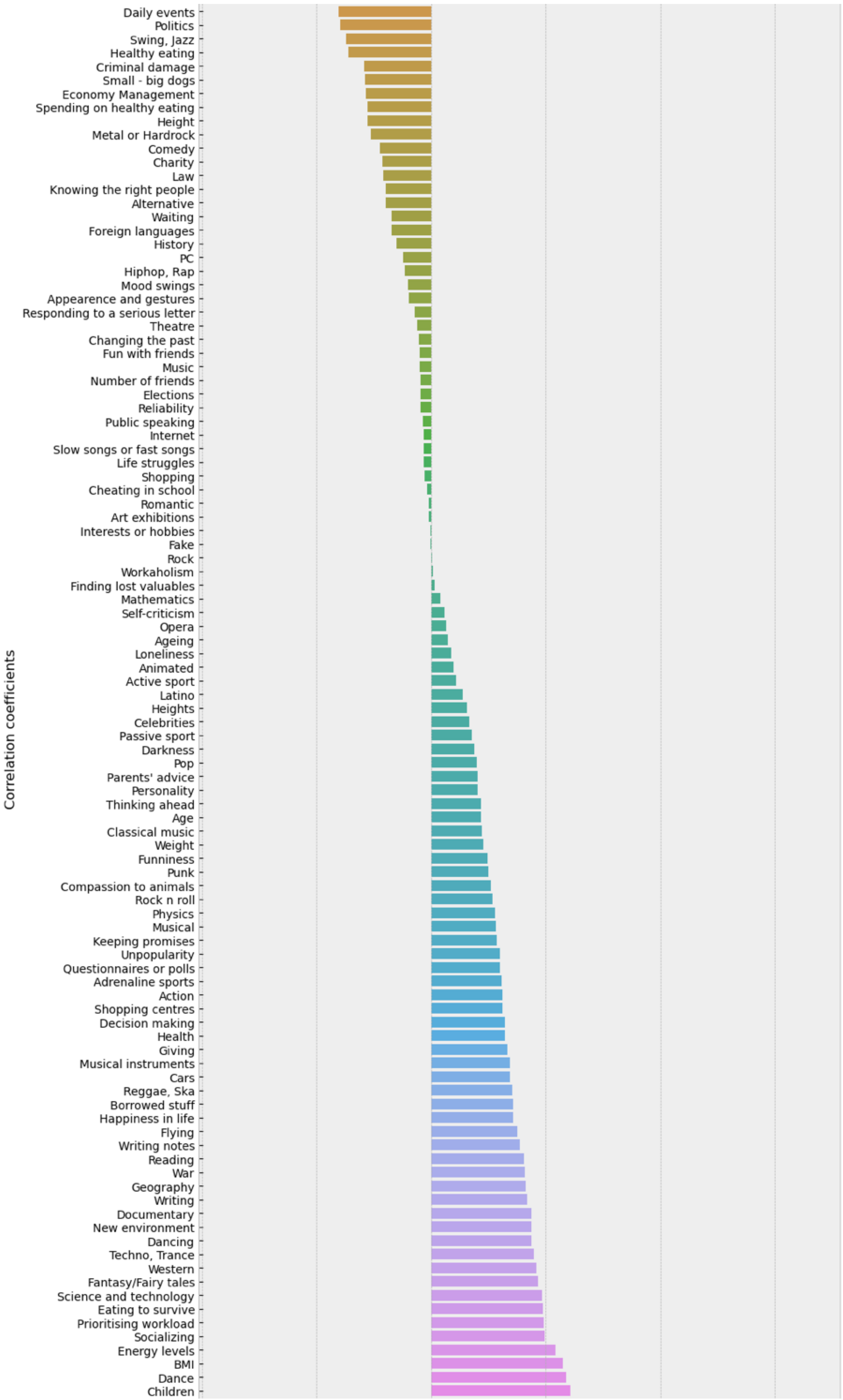
Firstly, look at correlations between the charachteristics and the urban-rural area. Correlation describes the degree of relationship between two variables. However, it tells nothing about the causuality. Just a small example, the anti-violent gaming policies say that there is a correlation between time spent on playing violent computer games and a violent behaviour. In fact, we do not know if a the computer games make a person violent ora violent person would play more violent games.

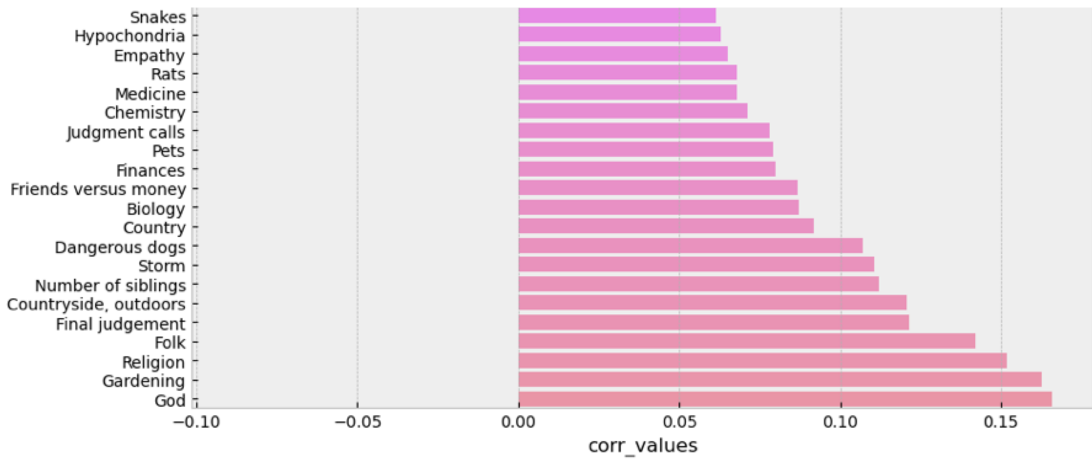
(Btw, I am a big fan of long-lasting code and functions that can be applied on different datasets :)). So, the function correlation_plot can be applied on different datasets. The function produces two plots: one for numerical features, another for categorical.

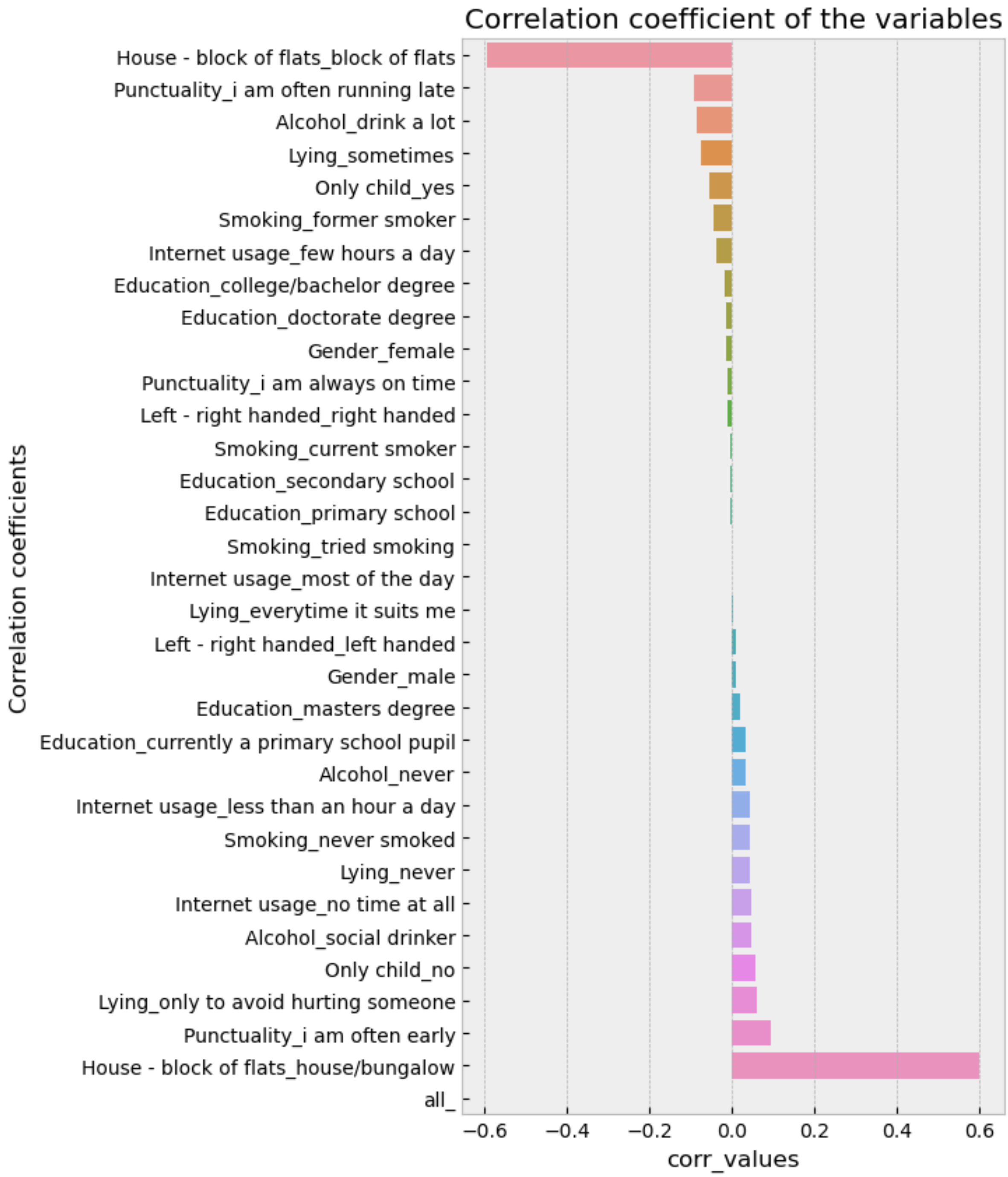
In [24]:
```
def do_ploting(x, y, figsize):
    fig, ax = plt.subplots(figsize=figsize)
    ax.set_title("Correlation coefficient of the variables")
    sns.barplot(x=x, y=y, ax=ax)
    ax.set_ylabel("Correlation coefficients")


def correlation_plot(var_of_interest, df_main, mapping, figsize=(10
    def calc_corr(var_of_interest, df, cols, figsize):
        lbls = []
        vals = []
        for col in cols:
            lbls.append(col)
            vals.append(np.corrcoef(df[col], df[var_of_interest])[0
        corrs = pd.DataFrame({'features': lbls, 'corr_values': vals
        corrs = corrs.sort_values(by='corr_values')
        do_ploting(corrs.corr_values, corrs['features'], figsize)
        return corrs

    #imputing the set
    df = copy.deepcopy(df_main)
    df.replace(mapping, inplace=True)
    mean_values = df.mean(axis=0)
    df.fillna(mean_values, inplace=True)

    #correlating non-categorical varibales
    cols_floats = [col for col in df.columns if df[col].dtype != 'o
    cols_floats.remove(var_of_interest)
    corrs_one = calc_corr(var_of_interest, df, cols_floats, figsize

    #correlating categorical variables
    cols_cats = [col for col in df.columns if df[col].dtype == 'obj
    if cols_cats:
        df_dummies = pd.get_dummies(df[cols_cats])
        cols_cats = df_dummies.columns
        df_dummies[var_of_interest] = df[var_of_interest]
        corrs_two = calc_corr(var_of_interest, df_dummies, cols_cat
    else:
        corrs_two = 0
    return [corrs_one, corrs_two]
```

In [26]:
```
corrs_area = correlation_plot(var_of_interest, data, mapping)
```




Correlation coefficients
Daily events
Politics
Swing, Jazz
Healthy eating
Criminal damage
Small - big dogs
Economy Management
Spending on healthy eating
Height
Metal or Hardrock
Comedy
Charity
Law
Knowing the right people
Alternative
Waiting
Foreign languages
History
PC
Hiphop, Rap
Mood swings
Appearence and gestures
Responding to a serious letter
Theatre
Changing the past
Fun with friends
Music
Number of friends
Elections
Reliability
Public speaking
Internet
Slow songs or fast songs
Life struggles
Shopping
Cheating in school
Romantic
Art exhibitions
Interests or hobbies
Fake
Rock
Workaholism
Finding lost valuables
Mathematics
Self-criticism
Opera
Ageing
Loneliness
Animated
Active sport
Latino
Heights
Celebrities
Passive sport
Darkness
Pop
Parents' advice
Personality
Thinking ahead
Age
Classical music
Weight
Funniness
Punk
Compassion to animals
Rock n roll
Physics
Musical
Keeping promises
Unpopularity
Questionnaires or polls
Adrenaline sports
Action
Shopping centres
Decision making
Health
Giving
Musical instruments
Cars
Reggae, Ska
Borrowed stuff
Happiness in life
Flying
Writing notes
Reading
War
Geography
Writing
Documentary
New environment
Dancing
Techno, Trance
Western
Fantasy/Fairy tales
Science and technology
Eating to survive
Prioritising workload
Socializing
Energy levels
BMI
Dance
Children


Snakes
Hypochondria
Empathy
Rats
Medicine
Chemistry
Judgment calls
Pets
Finances
Friends versus money
Biology
Country
Dangerous dogs
Storm
Number of siblings
Countryside, outdoors
Final judgement
Folk
Religion
Gardening
God
−0.10
−0.05
0.00
0.05
0.10
0.15
corr_values


Correlation coefficient of the variables
Correlation coefficients
House - block of flats_block of flats
Punctuality_i am often running late
Alcohol_drink a lot
Lying_sometimes
Only child_yes
Smoking_former smoker
Internet usage_few hours a day
Education_college/bachelor degree
Education_doctorate degree
Gender_female
Punctuality_i am always on time
Left - right handed_right handed
Smoking_current smoker
Education_secondary school
Education_primary school
Smoking_tried smoking
Internet usage_most of the day
Lying_everytime it suits me
Left - right handed_left handed
Gender_male
Education_masters degree
Education_currently a primary school pupil
Alcohol_never
Internet usage_less than an hour a day
Smoking_never smoked
Lying_never
Internet usage_no time at all
Alcohol_social drinker
Only child_no
Lying_only to avoid hurting someone
Punctuality_i am often early
House - block of flats_house/bungalow
all_
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
corr_values

The strongest correlations that we have are coming fro the house type and it is quite logical because people in the village would live most of the time in the houses. Other correlations that are not that strong are the associations with God and Spending on looks. We will dig into it.

In [27]:
```
#The strongest correlations that we have are
corr_num = corrs_area[0]
corr_cats = corrs_area[1]
display(corr_num[corr_num.corr_values == max(corr_num.corr_values)]
display(corr_num[corr_num.corr_values == min(corr_num.corr_values)]
display(corr_cats[corr_cats.corr_values == max(corr_cats.corr_value
display(corr_cats[corr_cats.corr_values == min(corr_cats.corr_value
```

	features	corr_values
101	God	0.165819

	features	corr_values
132	Spending on looks	-0.088743

	features	corr_values
31	House - block of flats_house/bungalow	0.600989

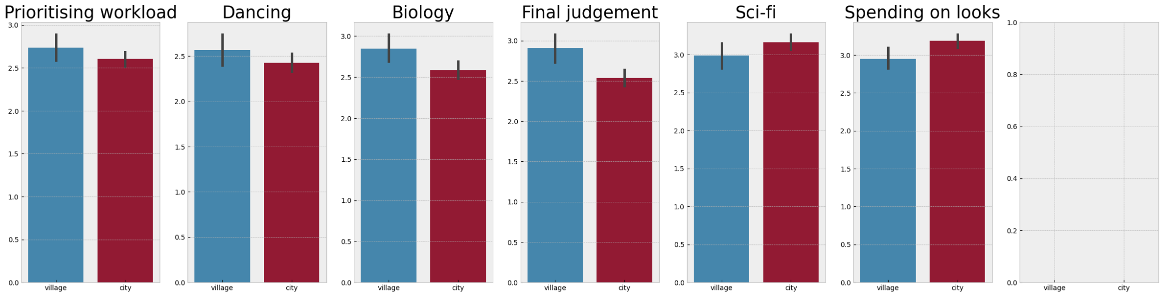
	features	corr_values
30	House - block of flats_block of flats	-0.594615

Characteristic differences

I have picked the features that were different among people from urban and rural areas. The plot of all features can be found at the end of this notebook.

In [28]:

```
good_columns = ['Dance', 'Folk', 'Techno, Trance','Religion', 'Medi
               'Spending on gadgets','Hypochondria','Western', 'Ea
               'God', 'Chemistry', 'Gardening', 'Politics','Econom
               'Branded clothing', 'Friends versus money','Number 
               'Storm', 'Rats', 'Country', 'Dangerous dogs', 'Fina
               'Entertainment spending', 'Horror', 'Pets', 'Priori
               'Biology', 'Final judgement', 'Sci-fi', 'Spending o
fig, ax = plt.subplots(nrows=5, ncols=7 ,figsize=(30,40), sharex=Tr
start = 0
for j in range(5):
    for i in range(7):
        if start == len(good_columns):
            break
        sns.barplot(y=good_columns[start], x=var_of_interest, data=
        ax[j,i].set_ylabel('')
        ax[j,i].set_xlabel('')
        ax[j,i].set_title(good_columns[start], fontsize=25)
        start += 1
```



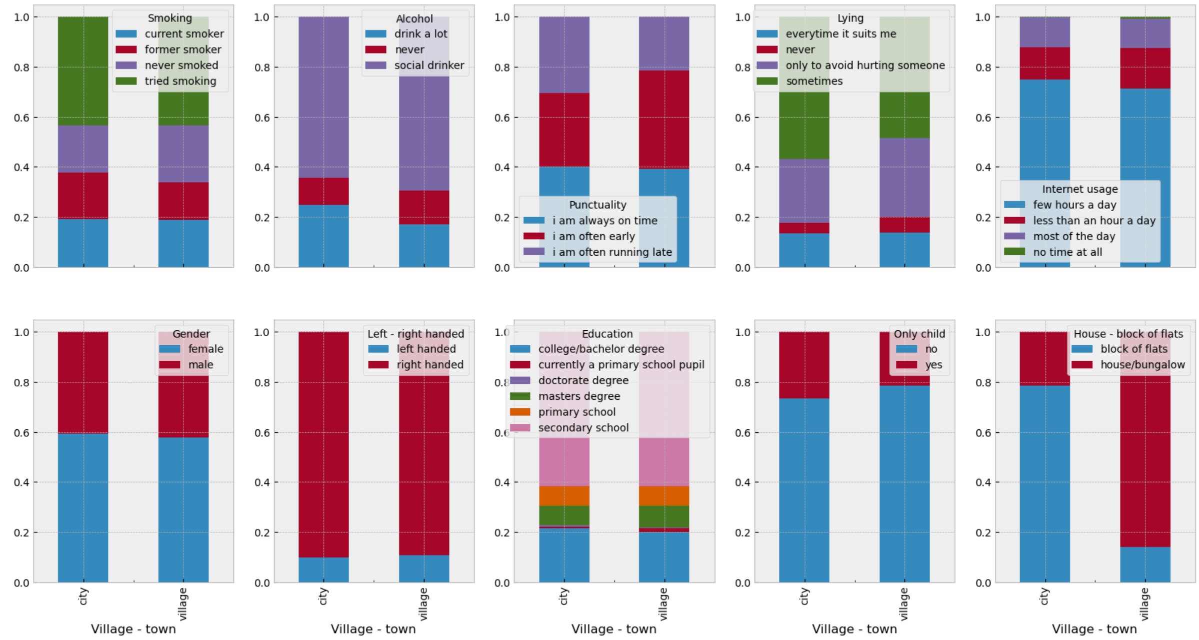
There are several interesting observations:

People from rural area are more into folk music; are more likely to spend time outdoors gardening; also there are more interested in biology, medicine and chemistry. It is quite logical as people in villages are often more close to the nature and therefore would spend time outdoors exploring. Also, the phobias of snakes, rats, dangerous dogs and storm are well-marked in people living in rural areas as they could have had a direct contact to it and know about the circumstances. On the other hand, they are less afraid of the spiders which again can be explained due to the fact that they were more exposed to the nature. Village-livers are more hypochondriac than city-livers which can be explained that there are less hospitals and doctors in villages.

What is a bit striking is that peopel from rural areas are more religios (Religion and God) than in the city. Additionaly, they are proned to "Final judgement" aka "I believe that bad people will suffer one day and good people will be rewarded". It can also have something to do with the stronger belief in God.

City-livers would spend more money on looks, entertainment and branded clothing as norally cities offer much more possibilitie to go shopping and have more entertainment facilities than the villages. Also, they would spend more time on gadgets.

In [31]:
```
features_cats = [col for col in data.columns if data[col].dtype=='o
features_cats.remove(var_of_interest)
fig, ax = plt.subplots(nrows=2, ncols=5,figsize=(20,10), sharex=Tru
start = 0
for j in range(2):
    for i in range(5):
        tab = pd.crosstab(data[var_of_interest], data[features_cats
        tab_prop = tab.div(tab.sum(1).astype(float), axis=0)
        tab_prop.plot(kind="bar", stacked=True, ax=ax[j,i] )
        start += 1
```



In the categorial features only one thing catches the eye, namelly, that village-livers mst of the time live in the houses comparing to the city livers who most live in the flat. That is also self-explanatory.

Multicollinearity

Apart from outliers and missing values, multicollinearity is another common issue. Some ML algorithms like Random Forest do not suffer from multicollinearity, whereas linear regression could have problems with it.

So, it would be quite exciting to look which characteristics are correlated in our data set.

In [32]:
```
= data.corr()
e: https://stackoverflow.com/questions/17778394/list-highest-correla
 matrix is symmetric so we need to extract upper triangle matrix wit

(corr.where(np.triu(np.ones(corr.shape), k=1).astype(np.bool))
            .stack()
            .sort_values(ascending=False))
ay(os.head(10))
ay(os.tail(10))
```

```
Weight                BMI                 0.845273
Height                Weight              0.737569
Biology               Medicine            0.724598
                      Chemistry           0.688375
Fantasy/Fairy tales   Animated            0.676642
Shopping              Shopping centres    0.649102
Chemistry             Medicine            0.632300
Classical music       Opera               0.600121
Mathematics           Physics             0.588532
Snakes                Rats                0.565351
dtype: float64
```

```
Action              Life struggles     -0.310373
Reading             Cars               -0.317057
Romantic            Height             -0.318037
Cars                Life struggles     -0.323088
Loneliness          Energy levels      -0.341980
Changing the past   Happiness in life  -0.357691
Dangerous dogs      Small - big dogs   -0.376013
Life struggles      Weight             -0.380090
                    Height             -0.401332
Loneliness          Happiness in life  -0.440306
dtype: float64
```

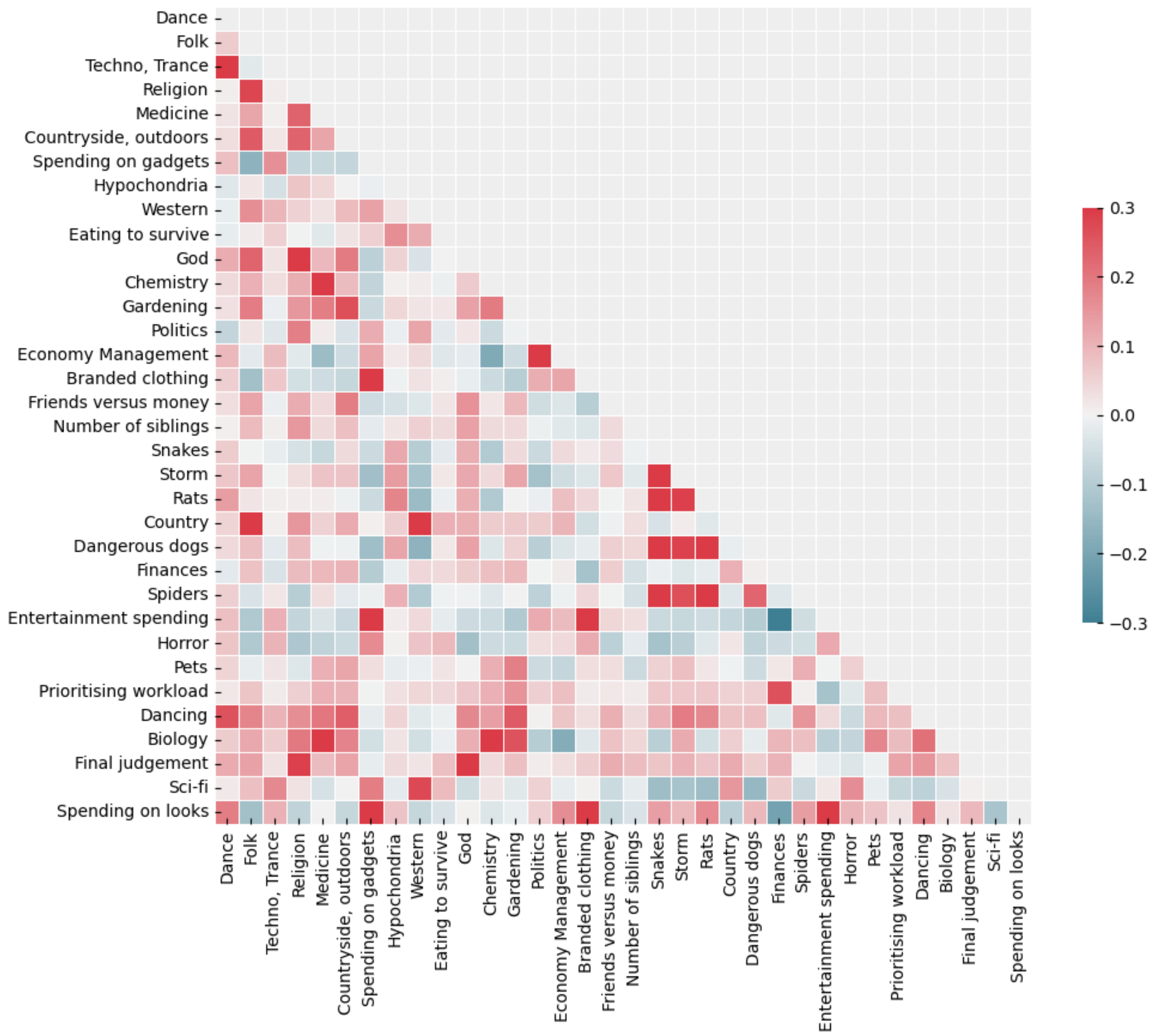
The most correlated features are Height and Weight, what again self-explanaroty. People who are interested in Biology are also interested in Medicne and Chemistry. The same for Fantasy/Fairy tales and Animated movies.

You might ask why there is a negative correlation between Life struggles, Weight and Heigt. I will disclose it a the end ;) and will confront once again with the definition of correlation.

Meanwhile lets have a look on the correlations amoung out good features.

In [33]:
```
corr = data[good_columns].corr()
mask = np.zeros_like(corr, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True
f, ax = plt.subplots(figsize=(11, 9))
cmap = sns.diverging_palette(220, 10, as_cmap=True)
sns.heatmap(
    corr,
    mask=mask,
    cmap=cmap,
    vmax=.3,
    center=0,
    square=True,
    linewidths=.5,
    cbar_kws={"shrink": .5})
```

Out[33]: <AxesSubplot:>



There is indeed a correlation between Final Judgement and God! Religios people tend to believe more that the bas ones will suffer.

In [34]:
```
print (os['Final judgement'][0:2])
```

```
God         0.491327
Charity     0.174467
dtype: float64
```

In [35]:
```
print (os['Religion'][0:2])
```

```
God                0.508850
Final judgement    0.289225
dtype: float64
```

The ones who spend money on entertainment, also spend money on looks and gadgets.

In [36]:
```
print (os['Entertainment spending'])
```

```
Spending on looks              0.402580
Spending on gadgets            0.336548
Height                         0.160816
Spending on healthy eating     0.143228
Weight                         0.131049
BMI                            0.064862
Number of siblings             0.031045
Age                           -0.042523
dtype: float64
```

Lets exclude of the element from the correlation pair once the correlation is more than 0.5

In [37]:
```
corr = data.corr()
os = (corr.where(np.triu(np.ones(corr.shape), k=1).astype(np.bool))
                 .stack()
                 .sort_values(ascending=False))
display(os[abs(os)>0.5])
drop_colinera_cols = os[abs(os)>0.5].reset_index()['level_1']
```

```
Weight                   BMI                  0.845273
Height                   Weight               0.737569
Biology                  Medicine             0.724598
                         Chemistry            0.688375
Fantasy/Fairy tales      Animated             0.676642
Shopping                 Shopping centres     0.649102
Chemistry                Medicine             0.632300
Classical music          Opera                0.600121
Mathematics              Physics              0.588532
Snakes                   Rats                 0.565351
Art exhibitions          Theatre              0.548380
Metal or Hardrock        Punk                 0.543569
Rock                     Metal or Hardrock    0.526920
Fear of public speaking  Public speaking      0.509547
Religion                 God                  0.508850
Horror                   Thriller             0.508406
Shopping                 Spending on looks    0.506264
Rock                     Punk                 0.504536
dtype: float64
```

Preparing the dataset for ML

In [38]:
```
clean_data = data.dropna(subset=[var_of_interest])
features_int = [col for col in clean_data.columns if clean_data[col
features_cats = [col for col in clean_data.columns if clean_data[co

features_int = list(set(features_int) - set(drop_colinera_cols))
print ('Number of features {:.0f}'.format(len(features_int)))
```

```
Number of features 124
```

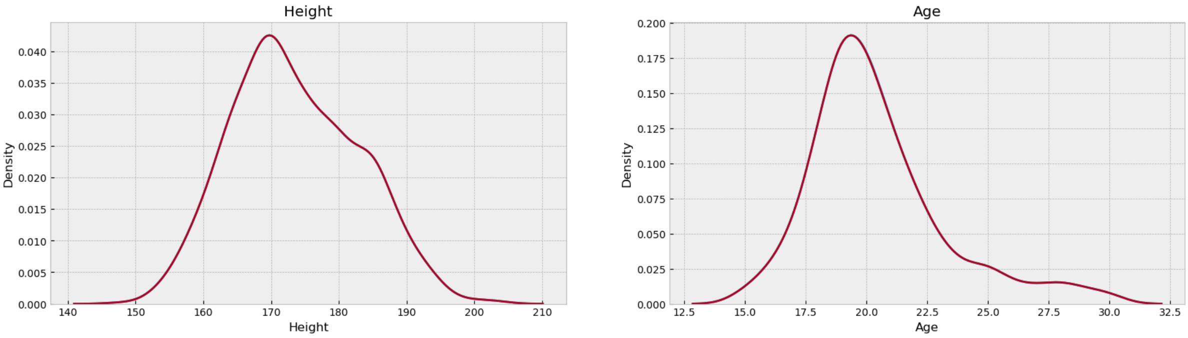
We will impute missing values with the mean, althought there are some better solutions to do it, like imputing Height and Weight according to the Gender or taking randomly a value in the range [mean - std, mean + std]

In [39]:
```
X = clean_data[features_int]
mean_values = X.mean(axis=0)
X = X.apply(lambda x: x.fillna(x.mean()),axis=0)
#X_cats = clean_data[features_cats].drop(var_of_interest, 1)
#X_cats = X_cats.drop('House - block of flats', 1)
#X_cats = pd.get_dummies(X_cats)
#print(X.shape)
#print(X_cats.shape)
```

In [41]:
```
from sklearn.metrics import make_scorer, accuracy_score, roc_auc_sc
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report


Y = clean_data[var_of_interest]
for key, val in mapping[var_of_interest].items():
    Y.replace(key,val, inplace = True)
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size
```

It is also a good idea to have a look on the distrubtion once me imputed the values to be sure that we did not disrupt it.

In [42]:
```
fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(20,5))

sns.kdeplot(X.Height,label = 'Before imputation', ax = ax[0]);
sns.kdeplot(clean_data.Height, label = 'After imputation', ax = ax[
ax[0].set_title('Height');

sns.kdeplot(X.Age,label = 'Before imputation', ax = ax[1]);
sns.kdeplot(clean_data.Age, label = 'After imputation', ax = ax[1])
ax[1].set_title('Age');

#sns.kdeplot(X.Weight,label = 'Before imputation', ax = ax[2])
#sns.kdeplot(clean_data.Weight, label = 'After imputation' , ax = a
#ax[2].set_title('Weight')
```



Machine Learning

We will standardize variables to be sure that everything is on whie scale as it plays quite an important role in regularization as we will apply logistic regression. The goal is to find variables that have more effect on the dependent variable (aka rural or urban liver).

(One also could use another metric to optimize like f1 but in fact accuracy metrix performed somehow better on this dataset regarding the f1 and accuracy score on test set.)

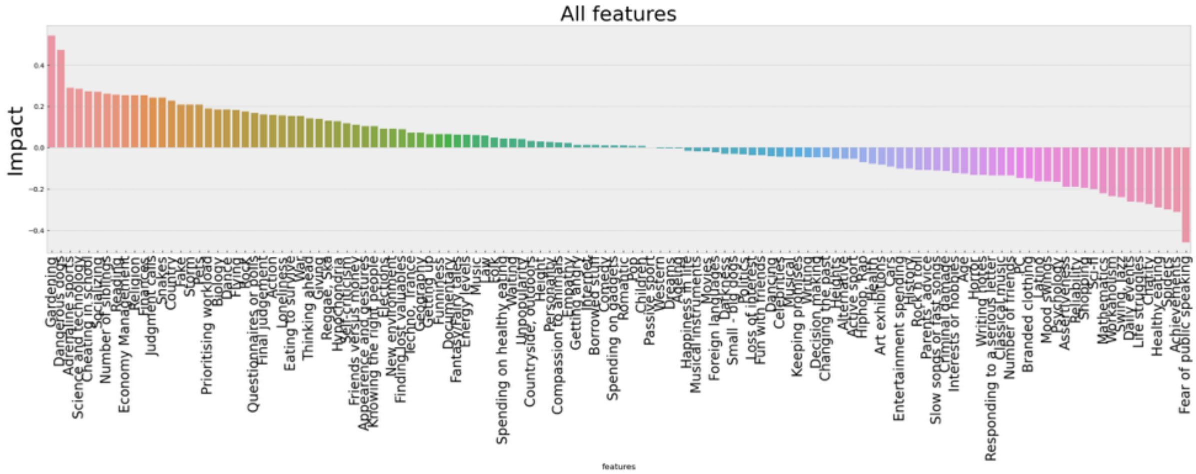
In [43]:
```
# standardization
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(x_train)
x_train = scaler.transform(x_train)
x_test = scaler.transform(x_test)
```

In [47]:
```
# gridsearch for parameter tuning
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.linear_model import LogisticRegression
clr = LogisticRegression()
KF = KFold(len(x_train))
param_grid = {'C':[.001,.01,.03,.1,.3,1,3,10]}
grsearch = GridSearchCV(clr, param_grid=param_grid, cv=KF, scoring
grsearch.fit(x_train, y_train)
print(grsearch.best_params_)

# fitting logistic regression and evaluating
clr = LogisticRegression(C=grsearch.best_params_['C'])
clr.fit(x_train, y_train)

mean_accuracy = np.mean(cross_val_score(clr, x_train, y_train, cv=K
print('Average accuracy score on CV set: {:.2f}'.format(mean_accura

mean_f1 = np.mean(cross_val_score(clr, x_train, y_train, cv=KF, sco
print('Average f1 on CV set: {:.2f}'.format(mean_f1))
print('')
print('Accuracy score on test set is: {:.2f}'.format(clr.score(x_te
recall = recall_score(y_test, clr.predict(x_test))
print ('Recall on test: {:.2f}'.format(recall))
precision = precision_score(y_test, clr.predict(x_test))
print ('Presicion on test: {:.2f}'.format(precision))
print ('F1 score on test: {:.2f}'.format((2*recall*precision /(reca
```

```
{'C': 3}
Average accuracy score on CV set: 0.67
Average f1 on CV set: 0.08

Accuracy score on test set is: 0.70
Recall on test: 0.40
Presicion on test: 0.51
F1 score on test: 0.45
```

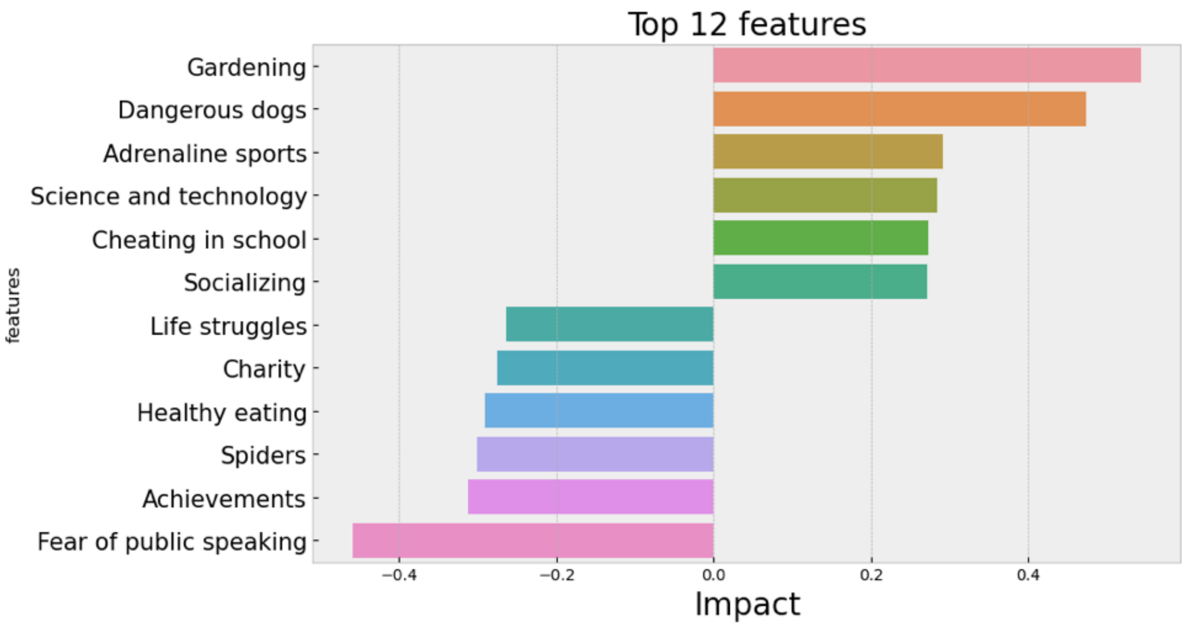
Lets look at the impact of all features on our rural-village liver classification. And no worries, we will zoom in.

In [48]:
```
feat_coeff = pd.DataFrame({'features': X.columns,'impacts': clr.coe
feat_coeff = feat_coeff.sort_values('impacts', ascending=False)

fig, ax1 = plt.subplots(1,1, figsize=(30,6));
sns.barplot(x=feat_coeff.features, y=feat_coeff.impacts, ax=ax1);
ax1.set_title('All features', size=30);
ax1.set_xticklabels(labels=feat_coeff.features, size=20, rotation=9
ax1.set_ylabel('Impact', size=30);
```



Zooming on only top 12 features based on their coefficients:

In [49]:
```
top10 = pd.concat([feat_coeff.head(6),feat_coeff.tail(6)])
fig, ax1 = plt.subplots(1,1, figsize=(10,6))
sns.barplot(y=top10.features, x=top10.impacts, ax=ax1);
ax1.set_title('Top 12 features', size=20);
ax1.set_yticklabels(labels=top10.features, size=15);
ax1.set_xlabel('Impact', size=20);
```



You can interpret it as : For every unit change in gardening (so, you thought ok, it is actually 5 not 4) the log odds of you having lived in city is decreased by 0.156.

It will get a bit simplier if we exponetiate the coefficients. The interpretation would be then: For every unit change in gardening the odds of you having lived in city is decreased by a factor exp(-0.156) = 0.8555592 (or by 85%).

Disclaimer: As we have quite lots of features and only 1K samples, it would be good to do somekind of feature selection as linear models tend to suffer from non-informative features. (see Max Kuhn "Applied Predictive Modeling", Feature Selection)

Insights:

- Gardening, fear of dogs and storm, as well as religion decrease the odds of having lived in the city. Rural area livers are more exposed to gardening and might have more experience with dangerous dogs. As well as storm might have a more devastating effect on the village than on a city. People in the village are also typically a bit more religious than in the city.
- The fear of spiders increase the odds of living in the city. It is also intuitive as city livers are less used to spiders than the village livers.

Sci-fi genre of movies might, healthy eating and fear of public speaking as well as getting angry are a bit speculative. Do urban livers get angry faster also why do they prefer Sci-fi more?

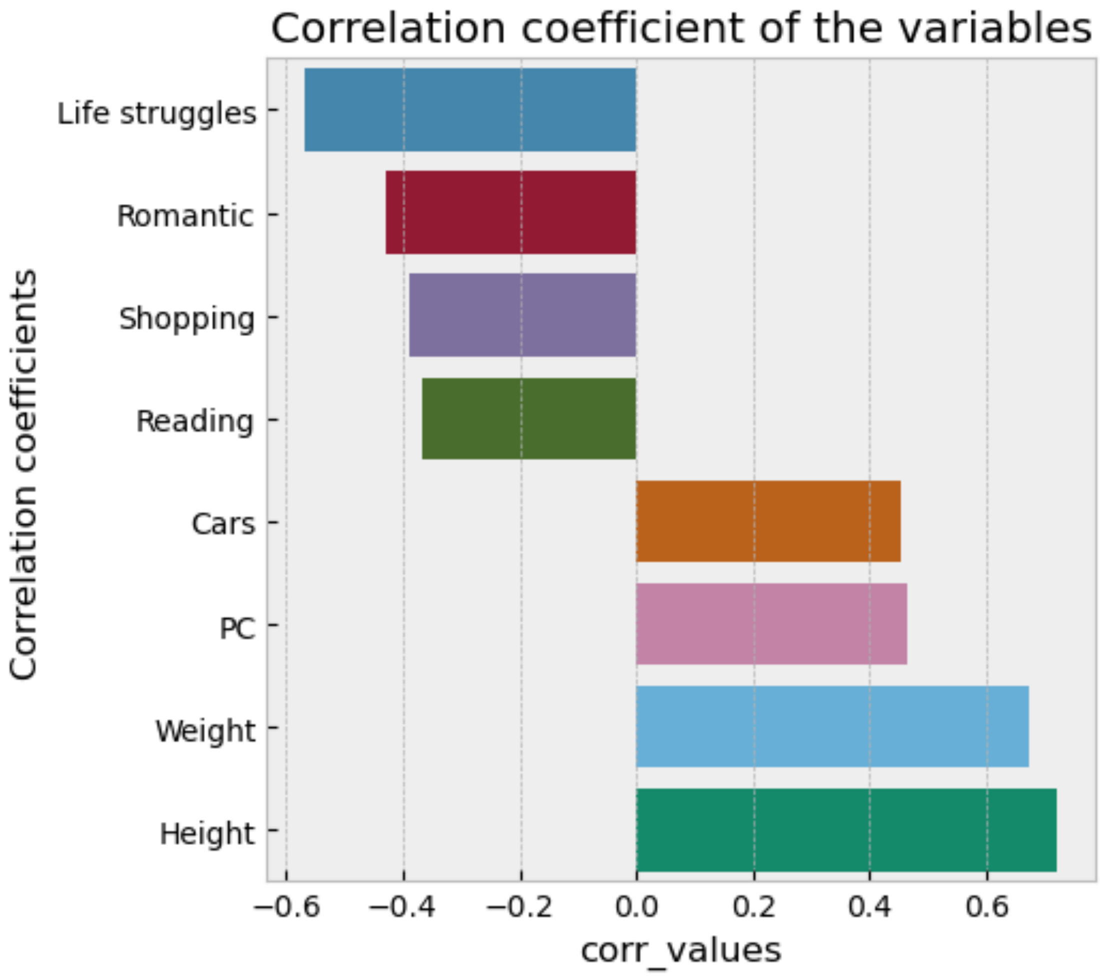
The dessert: negative correlation between Life struggles, Weight and Heigt

In [50]:
```
display(os.tail(3))
```

```
Life struggles  Weight              -0.380090
                Height              -0.401332
Loneliness      Happiness in life   -0.440306
dtype: float64
```

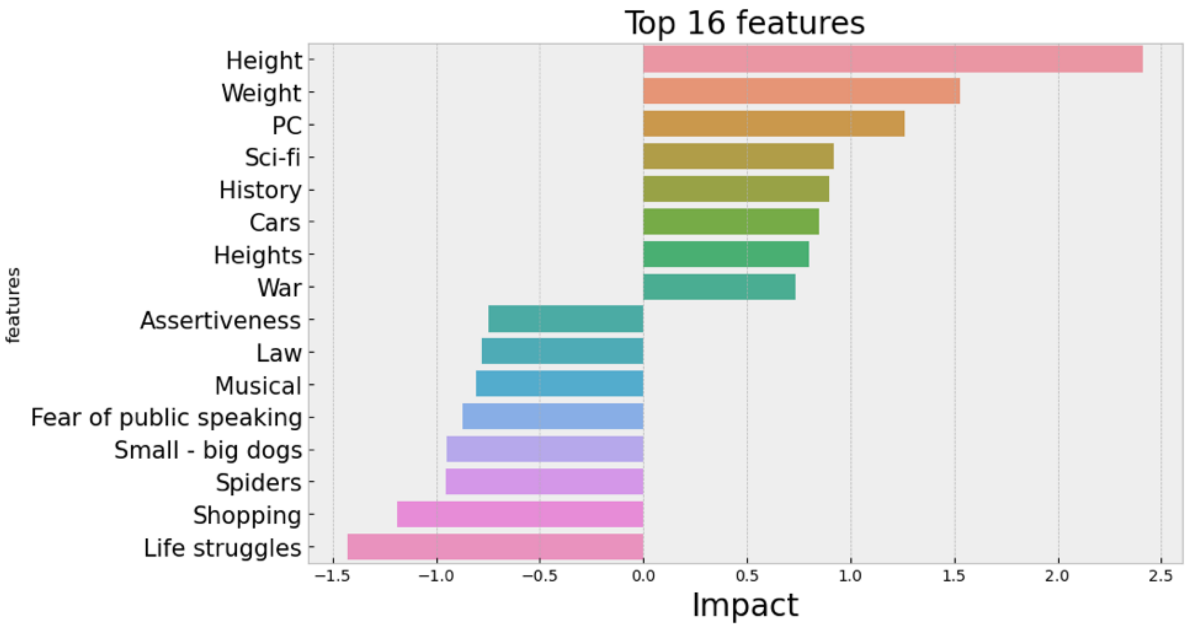
To understand why there is a negative correlation between life struggles, weight and height I will change the objectives. Now, we want to analyze the data based on the gender difference. So, lets plot correlation of each variable against the gender.

In [52]:
```
cols_to_keep = ['Life struggles', 'Romantic', 'Shopping',
                'Reading', 'Weight', 'Height', 'PC',
                'Cars', 'Gender']
gender_map = {'Gender': {'female': 0, 'male': 1}}
corrs_dfs_gender = correlation_plot('Gender', data[cols_to_keep], g
```



And a regression that will help us to get more insights.

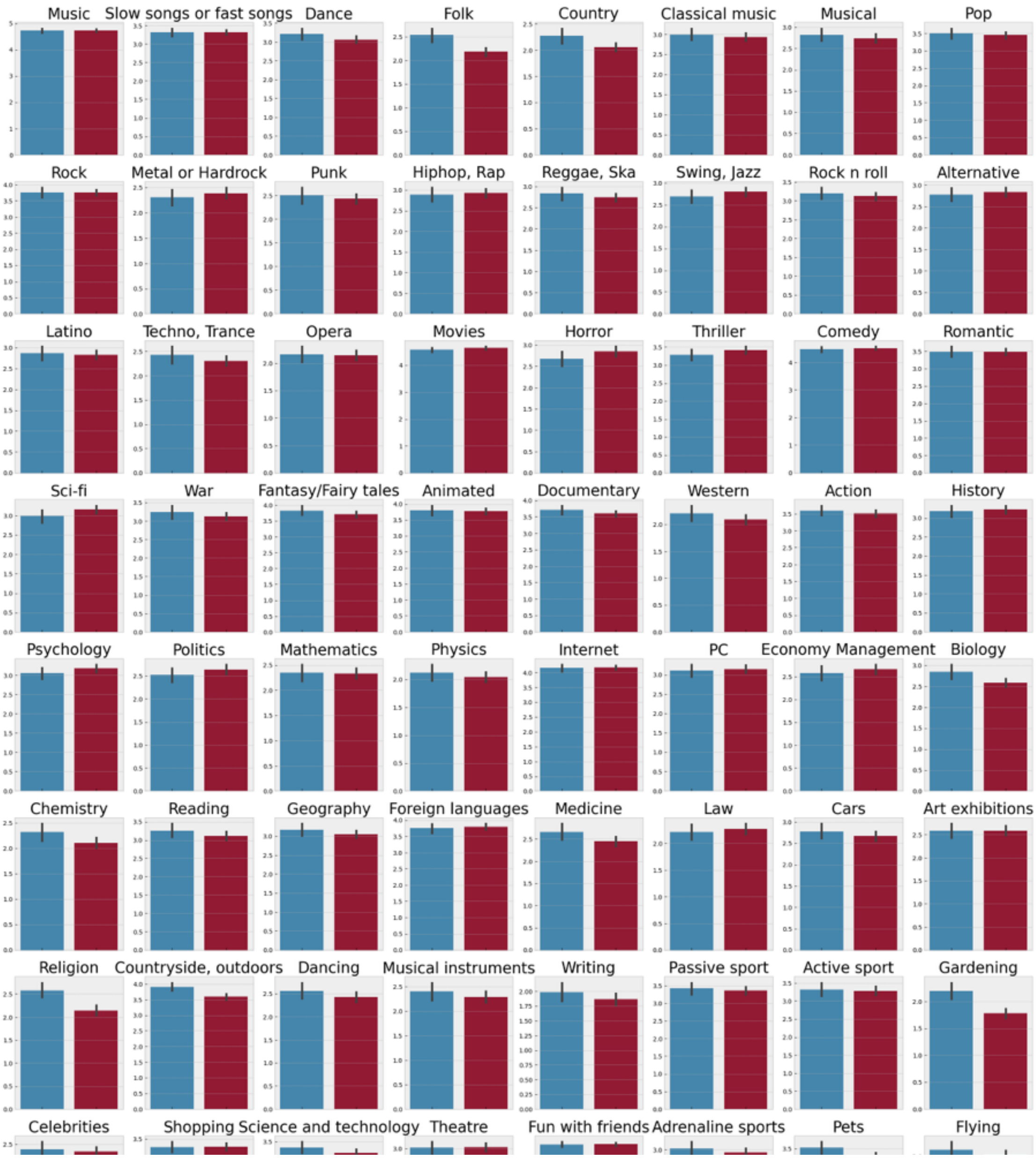
In [53]:
```
clean_data = data.dropna(subset=['Gender'])
features_int = [col for col in clean_data.columns if clean_data[col
X = clean_data[features_int]
mean_values = X.mean(axis=0)
X = X.apply(lambda x: x.fillna(x.mean()),axis=0)
Y = clean_data['Gender']
Y.replace('female',0, inplace = True)
Y.replace('male',1, inplace = True)

scaler = StandardScaler()
scaler.fit(X)
X = scaler.transform(X)

clr = LogisticRegression()
clr.fit(X, Y)
feat_coeff = pd.DataFrame({'features': features_int,'impacts': clr.
feat_coeff = feat_coeff.sort_values('impacts', ascending=False)

top10 = pd.concat([feat_coeff.head(8),feat_coeff.tail(8)])
fig, ax1 = plt.subplots(1,1, figsize=(10,6))
sns.barplot(y=top10.features, x=top10.impacts, ax=ax1);
ax1.set_title('Top 16 features', size=20);
ax1.set_yticklabels(labels=top10.features, size=15);
ax1.set_xlabel('Impact', size=20);
```

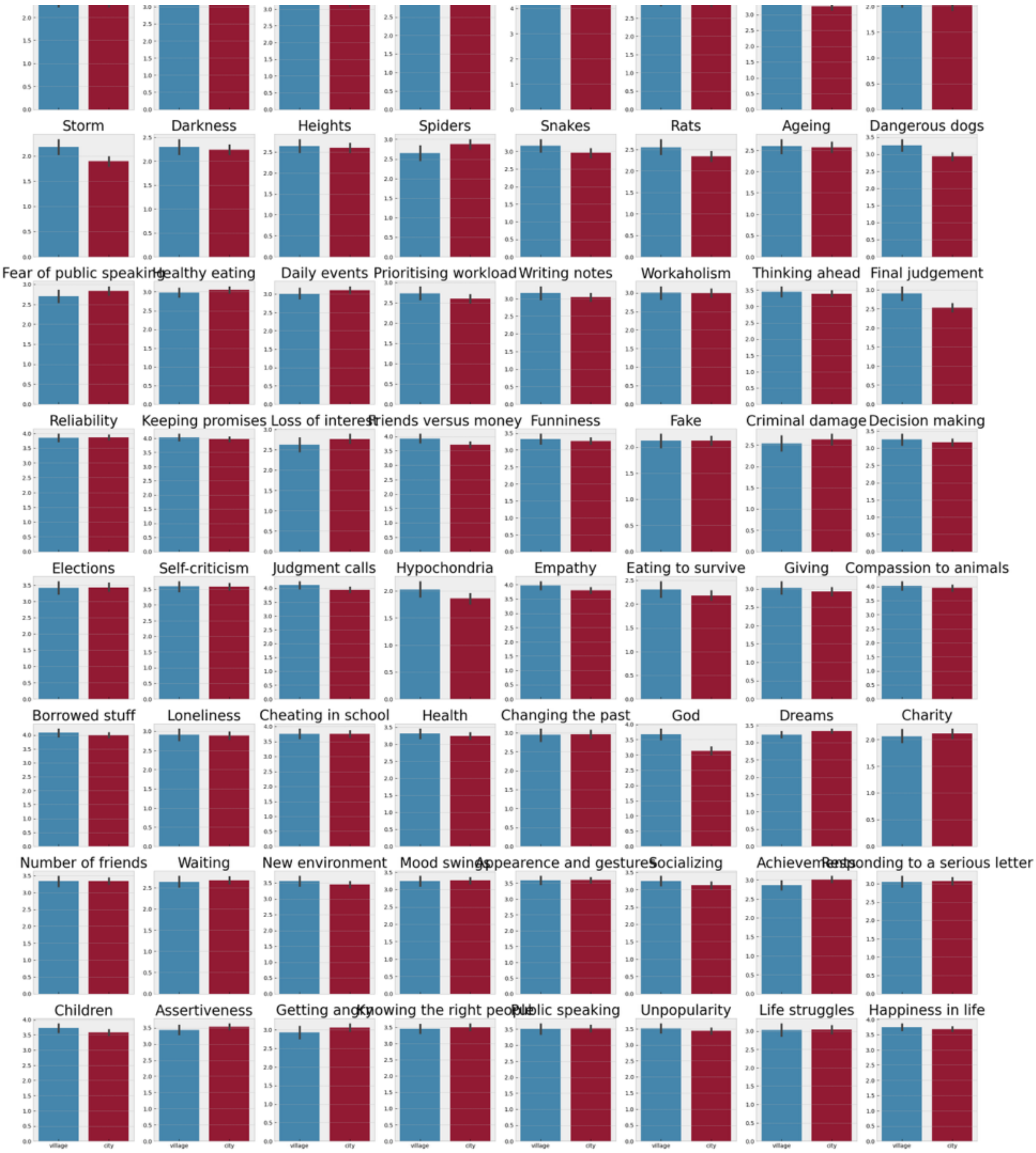


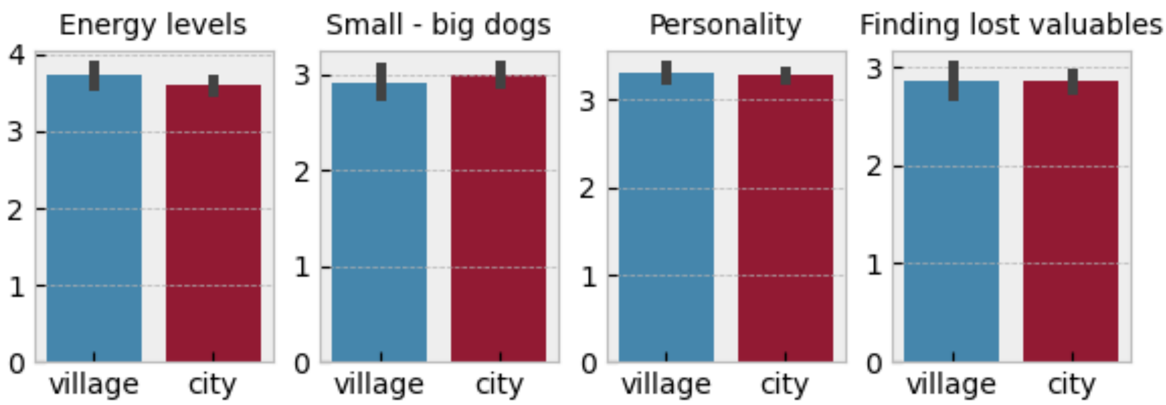
Additional plots

In [55]:

```
fig, ax = plt.subplots(nrows=15, ncols=8, figsize=(30, 70), sharex=
start = 0
for j in range(15):
    for i in range(8):
        sns.barplot(
            y=features_int[start], x=var_of_interest, data=data, ax
        ax[j, i].set_ylabel('')
        ax[j, i].set_xlabel('')
        ax[j, i].set_title(features_int[start], fontsize=25)
        start += 1

fig, ax = plt.subplots(nrows=1, ncols=4, figsize=(7, 2), sharex=Tru

for i in range(4):
    sns.barplot(y=features_int[start], x=var_of_interest, data=data
    ax[i].set_ylabel('')
    ax[i].set_xlabel('')
    ax[i].set_title(features_int[start], fontsize=10)
    start += 1
```

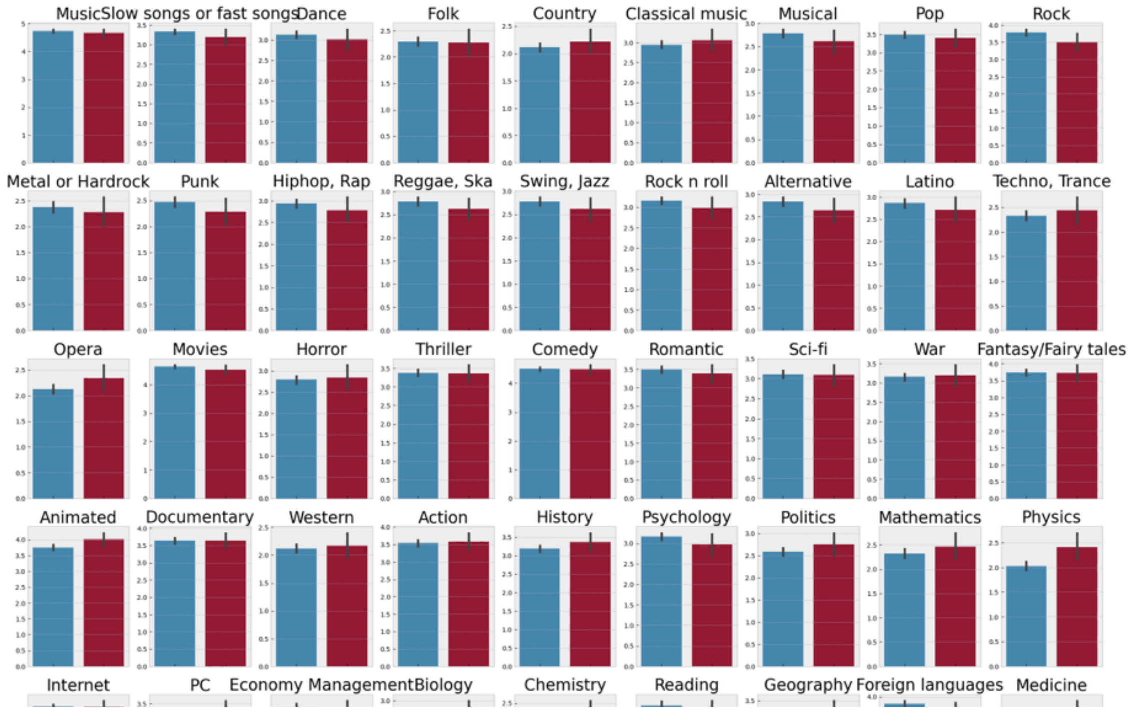






Initially I was intened to analyze the difference between left and high handed but the class distribution is even more imbalanced.

In [56]:
```
fig, ax = plt.subplots(nrows=15, ncols=9, figsize=(30, 70), sharex=
start = 0
for j in range(15):
    for i in range(9):
        sns.barplot(
            y=features_int[start],
            x='Left - right handed',
            data=data,
            ax=ax[j, i])
        ax[j, i].set_ylabel('')
        ax[j, i].set_xlabel('')
        ax[j, i].set_title(features_int[start], fontsize=25)
        start += 1

fig, ax = plt.subplots(nrows=1, ncols=4, figsize=(7, 2), sharex=Tru
for i in range(4):
    sns.barplot(
        y=features_int[start], x='Left - right handed', data=data,
    ax[i].set_ylabel('')
    ax[i].set_xlabel('')
    ax[i].set_title(features_int[start], fontsize=10)
    start += 1
```



In [ ]: