Open in Colab

Author: Phu Dang

Date: 01.20.2023

Mentor: Niema Moshiri, Niema Lab

In [1]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [2]:
```
# Importing libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import math
import seaborn as sns
from pandas.plotting import scatter_matrix
import scipy.stats as stats

import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
cd /content/drive/MyDrive/NIEMA_Lab/WHEL_Data/Baseline-Data
```

```
/content/drive/MyDrive/NIEMA_Lab/WHEL_Data/Baseline-Data
```

In [4]:
```
!ls
```

```
carot0.xls        ffq0.xls           Medical.xls  phbase.xls      TFbase.xls
demographics.xls  Health_Status.xls  NDSDIET.xls  REPROHX.xls
FAMHX.xls         LIFESTYLE.xls      Nsupps.xls   suppintake.XLS
```

In [5]:
```
# Loading baseline datasets

# Health Status at baseline
hs = pd.read_excel("Health_Status.xls")

# Demographics
dm = pd.read_excel("demographics.xls")

# Medical/clinic measurements at baseline
me = pd.read_excel("Medical.xls")

# Food from NDS at baseline
fd = pd.read_excel("NDSDIET.xls")

# Supplements intake at baseline
suppf = pd.read_excel("Nsupps.xls")
supp = pd.read_excel("suppintake.XLS")

# Thoughts and feelings at baseline
tf = pd.read_excel("TFbase.xls")
```

```
# Personal habits at baseline
ph = pd.read_excel("phbase.xls")

# Lifestyle at baseline
ls = pd.read_excel("LIFESTYLE.xls")

# Family history at baseline
fh = pd.read_excel("FAMHX.xls")

# Reproductive history at baseline
rh = pd.read_excel("REPROHX.xls")

# Food frequency at baseline
ff = pd.read_excel("ffq0.xls")

# Blood carotenoids at baseline
bc = pd.read_excel("carot0.xls")
```

In [6]:
```
cd /content/drive/MyDrive/NIEMA_Lab/WHEL_Data/Year-1-Data
```

```
/content/drive/MyDrive/NIEMA_Lab/WHEL_Data/Year-1-Data
```

In [7]:
```
# Loading Year 1 datasets

# Health Status at Year 1
hs_y1 = pd.read_excel("healthstaty1.XLS")

# Medical/clinic measurements at Year 1
me_y1 = pd.read_excel("clinicaly1.xls")

# Food from NDS at Year 1
fd_y1 = pd.read_excel("ndsfoody1.xls")

# Supplements intake at Year 1
suppf_y1 = pd.read_excel("nsuppsy1.xls")
supp_y1 = pd.read_excel("suppintakey1.xls")

# Thoughts and feelings at Year 1
tf_y1 = pd.read_excel("tfy1.xls")

# Personal habits at Year 1
ph_y1 = pd.read_excel("phy1.xls")

# Food frequency at Year 1
ff_y1 = pd.read_excel("ffq1.xls")

# Blood carotenoids at Year 1
bc_y1 = pd.read_excel("caroty1.xls")
```

In [8]:
```
cd /content/drive/MyDrive/NIEMA_Lab/WHEL_Data/Year-4-Data
```

```
/content/drive/MyDrive/NIEMA_Lab/WHEL_Data/Year-4-Data
```

In [9]:
```
# Loading Year 4 datasets

# Health Status at Year 4
hs_y4 = pd.read_excel("healthstaty4.XLS")

# Medical/clinic measurements at Year 4
me_y4 = pd.read_excel("clinicaly4.xls")
```

```
# Food from NDS at Year 4
fd_y4 = pd.read_excel("ndsfoody4.xls")

# Supplements intake at Year 4
suppf_y4 = pd.read_excel("nsuppsy4.xls")
supp_y4 = pd.read_excel("suppintakey4.xls")

# Thoughts and feelings at Year 4
tf_y4 = pd.read_excel("tfy4.xls")

# Personal habits at Year 4
ph_y4 = pd.read_excel("phy4.xls")

# Food frequency at Year 4
ff_y4 = pd.read_excel("ffq4.xls")

# Blood carotenoids at Year 4
bc_y4 = pd.read_excel("caroty4.xls")
```

In [10]:
```
cd /content/drive/MyDrive/NIEMA_Lab/WHEL_Data/
```

```
/content/drive/MyDrive/NIEMA_Lab/WHEL_Data
```

In [11]:
```
# Loading endpoints (results) dataset

ep = pd.read_excel("endpoints.xls")
```

In [12]:
```
dm.columns
```

Out[12]:
```
Index(['ID', 'age at dx', 'age at rand', 'Ethnicity', 'Education',
       'Marital Status', 'Empl. Status', 'Yr Rand'],
      dtype='object')
```

In [13]:
```
def examine_loss(df_before, df_after, operation_description):

  """
      Function to examine data (row) loss after some operation

  """

  num_rows_prior = df_before.shape[0]
  num_rows_after = df_after.shape[0]
  loss = num_rows_prior - num_rows_after

  ids_prior, ids_after = list(df_before['ID']), list(df_after['ID'])
  loss_ids = []
  for id in ids_prior:
    if id not in ids_after:
      loss_ids.append(id)

  return f"\nOperation: {operation_description}\nBefore: {num_rows_prior} rows\nAfter:
```

Data cleaning, preprocessing & EDA

In [14]:
```
hs.head(10)
```

Out[14]:

	ID	hospital_	Exam_Freq	Proph_Mast	Blood_Sugar	Cardiovascular	Digestive	Arthritis	Osteo
0	13245	0.0	1.0	0.0	0.0	0.0	0.0	0.0	
1	13244	1.0	1.0	0.0	0.0	0.0	0.0	0.0	
2	13243	0.0	1.0	0.0	0.0	0.0	0.0	0.0	
3	13242	0.0	1.0	0.0	0.0	0.0	0.0	0.0	
4	13241	0.0	3.0	0.0	0.0	0.0	0.0	0.0	
5	13240	0.0	1.0	0.0	0.0	0.0	0.0	0.0	
6	13239	0.0	3.0	0.0	1.0	0.0	0.0	1.0	
7	13238	0.0	2.0	0.0	0.0	0.0	0.0	0.0	
8	13237	0.0	1.0	0.0	0.0	0.0	0.0	1.0	
9	13236	0.0	2.0	0.0	0.0	0.0	0.0	0.0	

In [15]:

```
x = hs['Cardiovascular']
print(len(x))
x = [i for i in x if not math.isnan(i)]
```

```
3088
```

In [16]:

```
len(x)
```

Out[16]:

```
2543
```

In [17]:

```
np.unique(x)
```

Out[17]:

```
array([0., 1.])
```

In [18]:

```
cols = [i for i in hs.columns if i not in ['ID', 'hospital_', 'Exam_Freq']]
cols
```

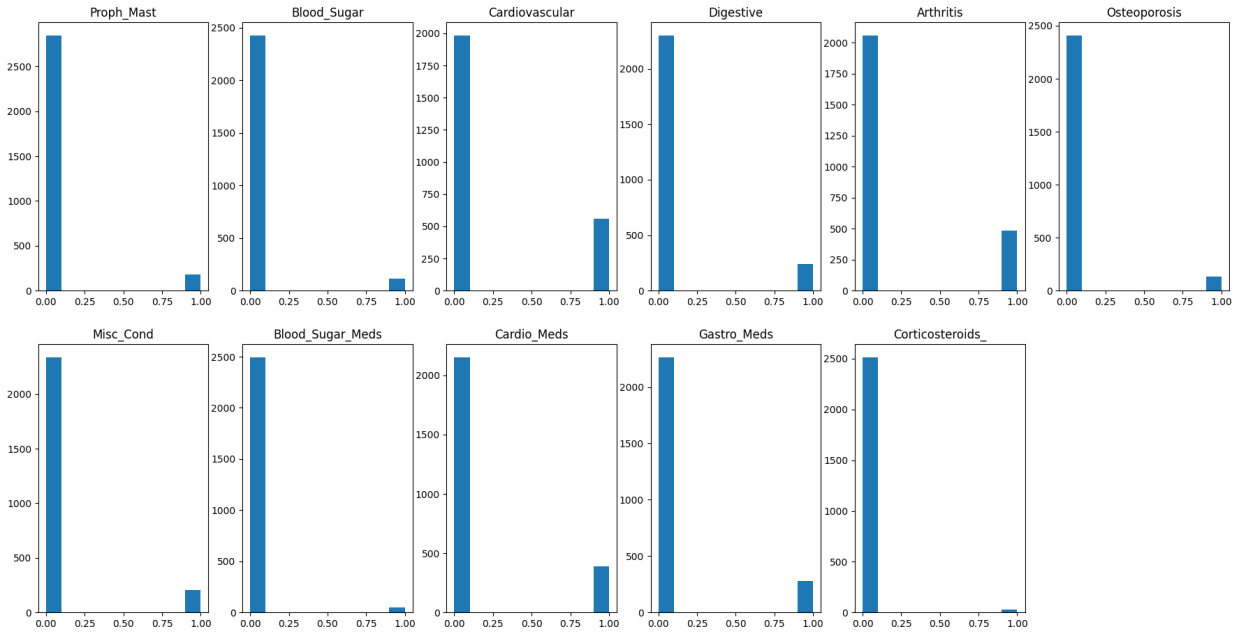
Out[18]:

```
['Proph_Mast',
 'Blood_Sugar',
 'Cardiovascular',
 'Digestive',
 'Arthritis',
 'Osteoporosis',
 'Misc_Cond',
 'Blood_Sugar_Meds',
 'Cardio_Meds',
 'Gastro_Meds',
 'Corticosteroids_']
```

In [19]:

```
index = 1
plt.figure(figsize=(22, 11))
for col in cols:
  plt.subplot(2, 6, index)
  values = [i for i in hs[col] if not math.isnan(i)]
  plt.hist(values)
```

```
    plt.title(col)
    index += 1
```



In [20]:
```
hs.sort_values(by='ID', ascending=True)
```

Out[20]:

	ID	hospital_	Exam_Freq	Proph_Mast	Blood_Sugar	Cardiovascular	Digestive	Arthritis	Os
3087	1002	NaN	NaN	0.0	NaN	NaN	NaN	NaN	
3086	1003	NaN	NaN	0.0	NaN	NaN	NaN	NaN	
3085	1005	NaN	NaN	0.0	NaN	NaN	NaN	NaN	
3084	1007	NaN	NaN	0.0	NaN	NaN	NaN	NaN	
3083	1008	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
...	...	...	...	...	...	...	...	...	
4	13241	0.0	3.0	0.0	0.0	0.0	0.0	0.0	
3	13242	0.0	1.0	0.0	0.0	0.0	0.0	0.0	
2	13243	0.0	1.0	0.0	0.0	0.0	0.0	0.0	
1	13244	1.0	1.0	0.0	0.0	0.0	0.0	0.0	
0	13245	0.0	1.0	0.0	0.0	0.0	0.0	0.0	

3088 rows × 14 columns

In [21]:
```
tf
```

Out[21]:

	ID	ss_tot	ss_emot	ss_aff	ss_tang	ss_pos	lv_alone	petownr	church	religion	...	sympto
0	1002	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	N
1	1003	45	20.0	5.0	10.0	10.0	0.0	1.0	5.0	3.0	...	0.0
2	1005		NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	N
3	1007		NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	N
4	1008		NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	N
...	...	...	...	...	...	...	...	...	...	...	...	
3083	13241	36	13.0	5.0	10.0	8.0	1.0	1.0	1.0	3.0	...	0.7
3084	13242	40	16.0	5.0	9.0	10.0	0.0	1.0	1.0	1.0	...	0.0
3085	13243	37	14.0	5.0	8.0	10.0	0.0	1.0	1.0	2.0	...	0.4
3086	13244	38	16.0	4.0	10.0	8.0	0.0	1.0	1.0	2.0	...	0.4
3087	13245	26	12.0	4.0	4.0	6.0	0.0	1.0	3.0	3.0	...	0.5

3088 rows × 38 columns

In [22]:
```
tf.columns
```

Out[22]:
```
Index(['ID', 'ss_tot', 'ss_emot', 'ss_aff', 'ss_tang', 'ss_pos', 'lv_alone',
       'petownr', 'church', 'religion', 'club', 'caregvB', 'strain',
       'optimism', 'nee', 'aee', 'hostil', 'QOL', 'physhealth', 'physfctn',
       'lim_phys', 'pain', 'genhlth', 'mental health', 'lim_emot', 'energy',
       'emotwell', 'socfctn', 'symptom', 'bloat', 'sweats', 'aches',
       'hotflash', 'tired', 'totlfevt', 'dep_log', 'whi_irs_old',
       'WHI_irs_new'],
      dtype='object')
```

In [23]:
```
tf['ss_tot'].unique()
```

Out[23]:
```
array([nan, 45, '  ', 43, 42, 31, 33, 32, 38, 24, 35, 40, 23, 41, 30, 36,
       26, 29, 44, 27, 39, 37, 21, 22, 34, 19, 17, 28, 25, ' ', 12, 18,
       20, 16, 11, 14, 13, 15], dtype=object)
```

In [24]:
```
tf = tf[tf['ss_tot'] != '  ']
tf = tf[tf['QOL'] != ' ']
tf = tf[tf['ss_tot'] != ' ']
```

In [25]:
```
tf['QOL'] = tf['QOL'].astype(float)
tf['physhealth'] = tf['physhealth'].astype(float)
tf['mental health'] = tf['mental health'].astype(float)
tf['ss_tot'] = tf['ss_tot'].astype(float)
```

In [26]:
```
tf.dtypes
```

Out[26]:
```
ID                 int64
ss_tot           float64
ss_emot          float64
ss_aff           float64
ss_tang          float64
ss_pos           float64
lv_alone         float64
petownr          float64
church           float64
religion         float64
club             float64
caregvB          float64
strain           float64
optimism         float64
nee              float64
aee              float64
hostil           float64
QOL              float64
physhealth       float64
physfctn         float64
lim_phys         float64
pain             float64
genhlth          float64
mental health    float64
lim_emot         float64
energy           float64
emotwell         float64
socfctn          float64
symptom          float64
bloat            float64
sweats           float64
aches            float64
hotflash         float64
tired            float64
totlfevt         float64
dep_log          float64
whi_irs_old      float64
WHI_irs_new      float64
dtype: object
```

In [27]:
```
tf.astype(float)
tf.dropna(inplace=True)
```

In [28]:
```
datasets = {"Health Status": hs, "Demographics": dm, "Medical Measurements": me,
            "Food from NDS": fd, "Supplements": supp, "Thoughts and Feelings": tf,
            "Personal Habits": ph, "Lifestyle": ls, "Family History": fh, "Reproductiv
            "Food Frequency": ff, "Blood Carotenoids": bc}
```

In [29]:
```
datasets_y1 = {"Health Status": hs_y1, "Medical Measurements": me_y1,
               "Food from NDS": fd_y1, "Supplements": supp_y1, "Thoughts and Feelings"
               "Personal Habits": ph_y1, "Food Frequency": ff_y1, "Blood Carotenoids":
```

In [30]:
```
datasets_y4 = {"Health Status": hs_y4, "Medical Measurements": me_y4,
               "Food from NDS": fd_y4, "Supplements": supp_y4, "Thoughts and Feelings"
               "Personal Habits": ph_y4, "Food Frequency": ff_y4, "Blood Carotenoids":
```

In [31]:
```
for label, dataset in datasets.items():
  print(label)
```

```
print(dataset.columns)
print()
```

```
Health Status
Index(['ID', 'hospital_', 'Exam_Freq', 'Proph_Mast', 'Blood_Sugar',
       'Cardiovascular', 'Digestive', 'Arthritis', 'Osteoporosis', 'Misc_Cond',
       'Blood_Sugar_Meds', 'Cardio_Meds', 'Gastro_Meds', 'Corticosteroids_'],
      dtype='object')

Demographics
Index(['ID', 'age at dx', 'age at rand', 'Ethnicity', 'Education',
       'Marital Status', 'Empl. Status', 'Yr Rand'],
      dtype='object')

Medical Measurements
Index(['ID', 'Lump/Mast', 'Radiation', 'Chemo', 'Tumor Type', 'Estr Recep',
       'Prog Recep', 'Node Exam', 'Node Pos', 'Stage', 'Tumor Size', 'Height',
       'Weight', 'BMI', 'Waist', 'Hip', 'Pulse', 'Bld Pres -Sys',
       'Bld Pres - Dias', 'Menopause', 'Anti-Estr Use', 'Anti-Estro Type'],
      dtype='object')

Food from NDS
Index(['ID', 'Recalls', 'last date', 'Kcal', 'Protein(g', 'Total Fat',
       'Total Carb', 'Alcohol', '% Protein', '% Fat', '% Carb', '% Alcohol',
       'Caffeine', 'Calcium', 'Iron', 'Magnesium', 'Phosphorus', 'Potassium',
       'Sodium', 'Zinc', 'Copper', 'Selenium', 'Total Vit A', 'Retinol',
       'Alpha-Toc Eq', 'alpha-Toc', 'Gamma-Toc', 'Vit D', 'Vitamin C',
       'Thiamin', 'Riboflavin', 'Niacin', 'Folacin', 'Panto. Acid', 'Vit B6',
       'Vit B12', 'Cholesterol', 'Total SFA', 'Total MFA', 'Total PFA',
       '% SFA', '%MFA', '% PFA', 'PFA 18:2', 'PFA 18:3', 'PFA 20:5',
       'PFA 22:6', 'Diet Fiber', 'Soluble Fiber', 'Insol. Fiber', 'Starch',
       'Sucrose', 'Galactose', 'Fructose', 'Lactose', 'Maltose', 'Water',
       'Anim. Protein', 'Veg Prot', 'Vit K', 'Trans Fat Acids', 'Beta-Carot',
       'Alpha-Carot', 'Beta-Cryptox', 'Lut + Zeax', 'Lycopene', 'Green Veg',
       'Iceber', 'Tomato', 'Orange Veg', 'Other Veg', 'Wh. Pota',
       'Veg_woJ,PI,Iceberg', 'VJ', 'Legume', 'Soy Legume', 'Citrus', 'Berry',
       'Melon', 'Other Fruit', 'FJ', 'Fruit_noJ', 'Whole Grain',
       'Refined Grain'],
      dtype='object')

Supplements
Index(['Case Id No.', 'NSUPPREC', 'Retinol (mcg)', 'Calcium (mg)', 'Iron (mg)',
       'Magnesium (mg)', 'Phosphorus (mg)', 'Beta-carotene(mcg)',
       'Vitamin D (mcg)', 'Total alpha-Toc eg', 'Vitamin C (mg)',
       'Thiamin (mg)', 'Niacin (mg)', 'Riboflavin (mg)', 'Vitamin B6 (mg)',
       'Folacin (mcg)', 'Vitamin B12 (mcg)', 'Zinc (mg)', 'Selenium (mg)',
       'Potassium (mg)', 'Sodium (mg)', 'Phosphorus (mg).1', 'Copper (mg)',
       'Pantothenic Acid (mg)'],
      dtype='object')

Thoughts and Feelings
Index(['ID', 'ss_tot', 'ss_emot', 'ss_aff', 'ss_tang', 'ss_pos', 'lv_alone',
       'petownr', 'church', 'religion', 'club', 'caregvB', 'strain',
       'optimism', 'nee', 'aee', 'hostil', 'QOL', 'physhealth', 'physfctn',
       'lim_phys', 'pain', 'genhlth', 'mental health', 'lim_emot', 'energy',
       'emotwell', 'socfctn', 'symptom', 'bloat', 'sweats', 'aches',
       'hotflash', 'tired', 'totlfevt', 'dep_log', 'whi_irs_old',
       'WHI_irs_new'],
      dtype='object')

Personal Habits
Index(['ID', 'smoking status', 'pack years', 'NEW METS', 'wgt history'], dtype='objec
t')
```

```
Lifestyle
Index(['ID', 'main meals', 'snacks', 'meals out', 'cooking time', 'adults',
       'children', 'I cook?', 'cooking help', 'I clean up?', 'cleanup help',
       'shopping freq', 'grocery bill', 'I buy food?', 'shopping help',
       'weight', 'time at weight', 'pre-BC weight', 'age18 weight',
       'max weight', 'time at max wgt', 'health insurance',
       'insurance change?', 'insurance satisfaction', 'years in state',
       'where born', 'age15 residence', 'age35 residence', 'age50 residence',
       'longest residence'],
      dtype='object')

Family History
Index(['id', 'adopted?', '# first deg relatives',
       '# first deg relatives with breast cancer',
       '# first deg relatives with any cancer',
       '# second deg relatives (granparents, aunts, uncles)',
       '# second deg relatives with breast cancer',
       '# second deg relatives with any cancer', 'mom had BC?',
       'mom had ovarian cancer?', '# sisters', '# sisters with breast cancer',
       '# sisters with ovarian cancer',
       '# first deg relatives with colorectal cancer',
       '# second deg relatives with colorectal cancer'],
      dtype='object')

Reproductive History
Index(['ID', 'Menarch Age', '# Times Pregnant', '# Live Births',
       'Age at First Live Birth', 'Months Full Breast Feeding',
       'Months Partial Breastfeeding', 'Age at Menopause', 'Hysterectomy',
       '# Oophorectomies', 'Hormonal Birth Control', 'Birth Control Type',
       'Years of Birth Control', 'HRT', 'HRT Type', 'Years of HRT',
       '# Previous Breast Biopsies'],
      dtype='object')

Food Frequency
Index(['id', 'ener', 'prot', 'tfat', 'carb', 'alco', 'prot_pcal', 'tfat_pcal',
       'carb_pcal', 'alco_pcal', 'caff', 'calc', 'iron', 'magn', 'phos',
       'pota', 'sodi', 'zinc', 'copp', 'sele', 'vita', 'reti', 'vite', 'gtoc',
       'vitd', 'vitc', 'thia', 'ribo', 'niac', 'fola', 'pant', 'vib6', 'vb12',
       'chol', 'sfat', 'mufa', 'pufa', 'sfat_pcal', 'mufa_pcal', 'pufa_pcal',
       'f182n', 'alin', 'epa', 'dha', 'tfib', 'star', 'sucr', 'gala', 'fruc',
       'lact', 'malt', 'watr', 'vitk', 'trans', 'bcar', 'acar', 'cryp', 'lute',
       'lyco', 'veg', 'vj', 'fruit', 'frj'],
      dtype='object')

Blood Carotenoids
Index(['ID', 'lutein', 'cryptox', 'lycopene', 'alphacar', 'betacar'], dtype='object')
```

Compare visualizations of biomarkers between hospitalized and not hospitalized patients

Datasets: thoughts and feelings, lifestyle, family history, reproductive history, blood carotenoids

(Baseline exclusively)

In [32]:
```
# Merging heath status and blood carotenoids
```

```
health_bc = hs.merge(right=bc, how="inner", on="ID")
health_bc.columns
health_bc.shape
```

Out[32]:
```
(3088, 19)
```

In [33]:
```
# Cleaning and preprocessing health_tf (health status and thoughts/feelings merged)

health_bc.dropna(inplace=True)
drop_idx = health_bc[health_bc['hospital_'] == 6.].index
health_bc.drop(index=drop_idx, inplace=True)
health_bc.shape

# for col in hs.columns:
#   if col != 'ID':
    # print(col)
    # print(health_tf[col].unique())
    # print()

health_bc.head(5)
```

Out[33]:

	ID	hospital_	Exam_Freq	Proph_Mast	Blood_Sugar	Cardiovascular	Digestive	Arthritis	Osteo
0	13245	0.0	1.0	0.0	0.0	0.0	0.0	0.0	
1	13244	1.0	1.0	0.0	0.0	0.0	0.0	0.0	
2	13243	0.0	1.0	0.0	0.0	0.0	0.0	0.0	
3	13242	0.0	1.0	0.0	0.0	0.0	0.0	0.0	
4	13241	0.0	3.0	0.0	0.0	0.0	0.0	0.0	

In [34]:
```
health_bc.shape
```

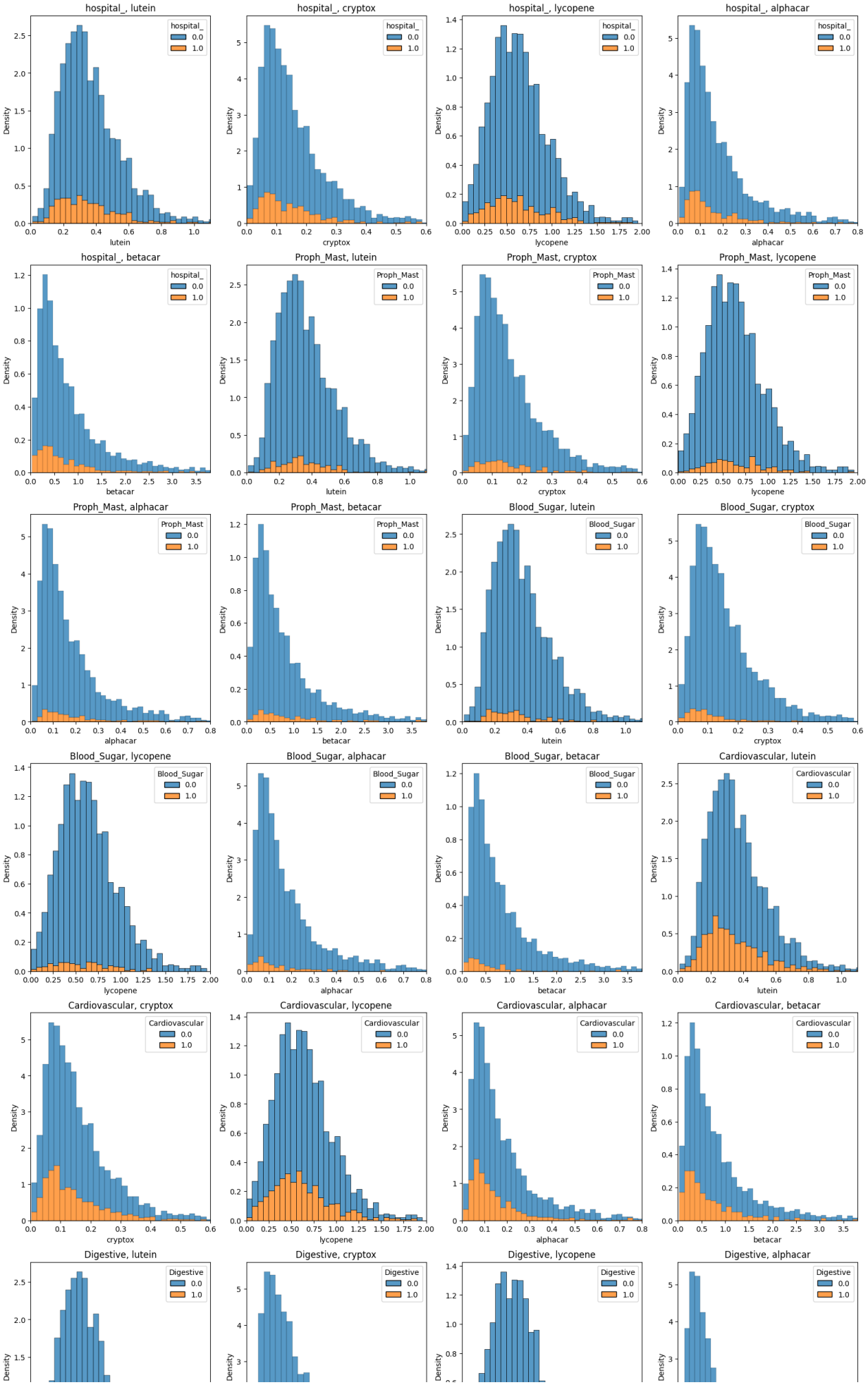
Out[34]:
```
(2498, 19)
```

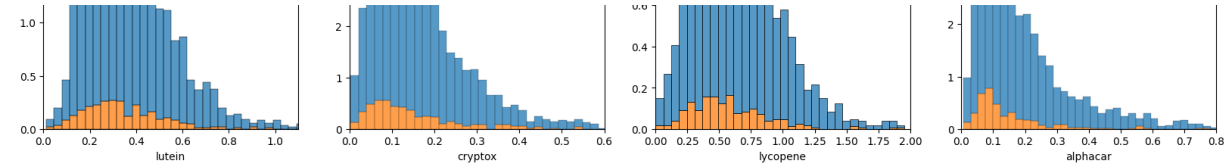
In [35]:
```
# Visualize health_tf

# plt.figure(figsize = (31, 19))
# ax = sns.heatmap(health_tf.corr(), annot = True, fmt = '.2f', linewidth = 0.5)
# props = plt.setp(ax.xaxis.get_majorticklabels(), rotation=45, ha='right', rotation_m
# plt.title('Health Status and Thoughts/Feelings Correlation', size = 15)
```

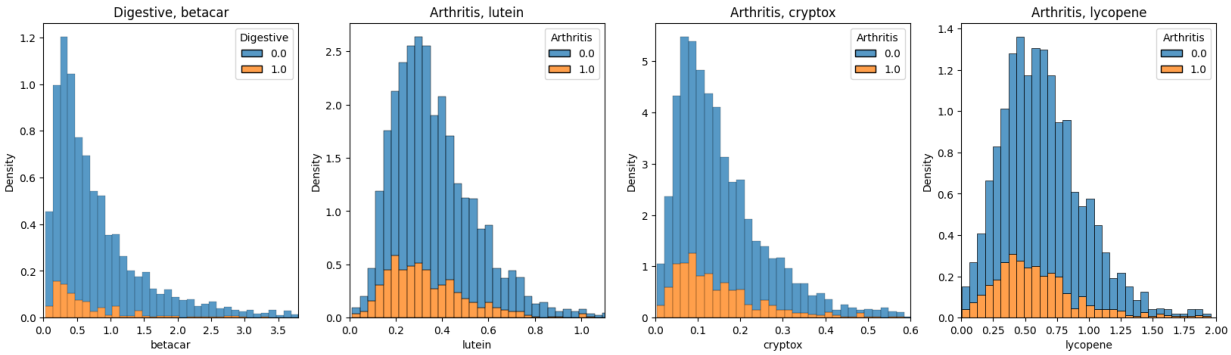
In [36]:
```
# cols = hospitalized_bc.columns
# sns.pairplot(hospitalized_bc[[col for col in cols if col != 'ID']], hue = 'hospital_
```

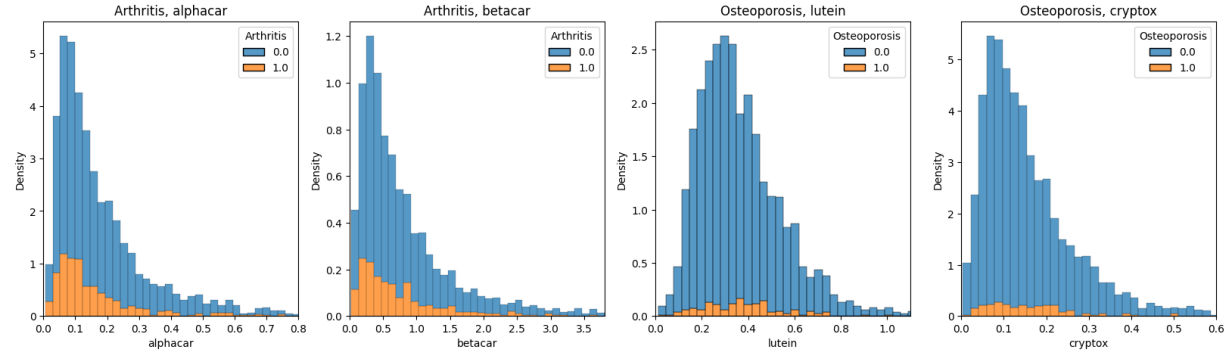
In [37]:
```
index = 1
plt.figure(figsize=(20, 90)) # sharey=False
for condition in hs.columns:
  if condition not in ['ID', 'Exam_Freq']:
    for col in bc.columns:
      if col != 'ID':
        if col == 'lutein':
          x = 1.1
        elif col == 'cryptox':
```

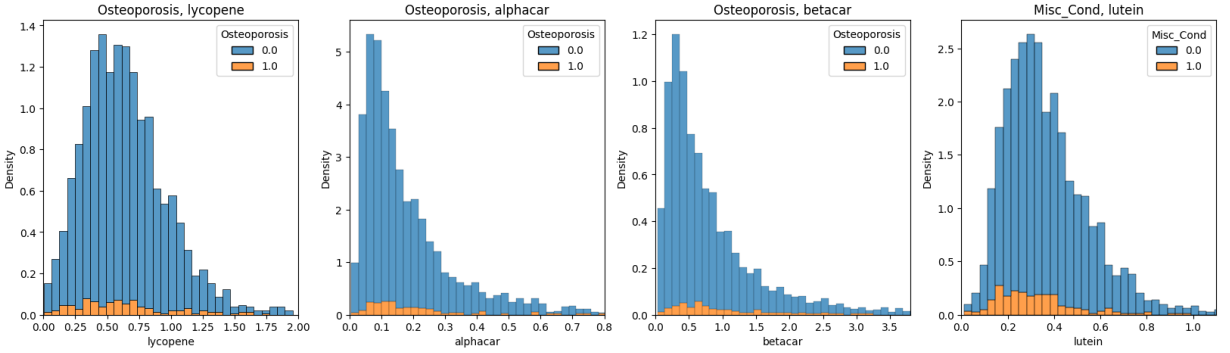
```
            x = 0.6
        elif col == 'alphacar':
            x = 0.8
        elif col == 'betacar':
            x = 3.8
        else:
            x = 2
        plt.subplot(15, 4, index)
        sns.histplot(health_bc, x=col, hue=condition, multiple='stack', stat='density'
        plt.xlim(0, x)
        plt.title(f"{condition}, {col}")
        index += 1
```

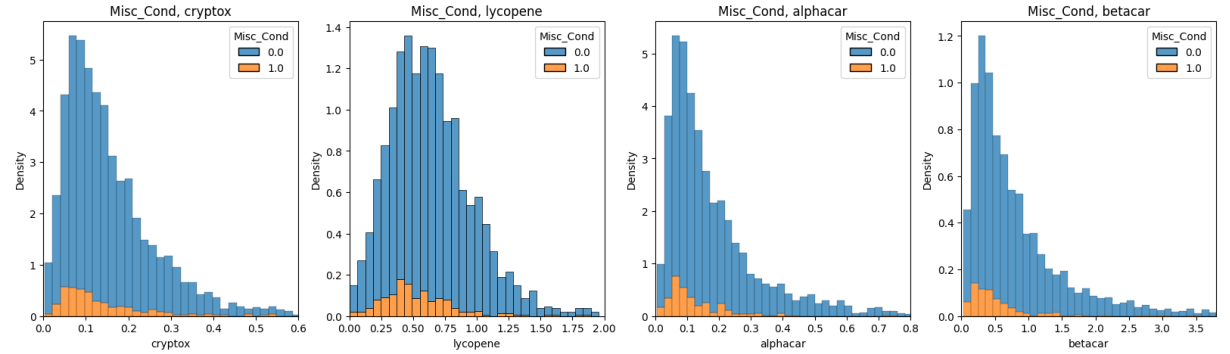

hospital_, lutein
hospital_, cryptox
hospital_, lycopene
hospital_, alphacar
hospital_, betacar
Proph_Mast, lutein
Proph_Mast, cryptox
Proph_Mast, lycopene
Proph_Mast, alphacar
Proph_Mast, betacar
Blood_Sugar, lutein
Blood_Sugar, cryptox
Blood_Sugar, lycopene
Blood_Sugar, alphacar
Blood_Sugar, betacar
Cardiovascular, lutein
Cardiovascular, cryptox
Cardiovascular, lycopene
Cardiovascular, alphacar
Cardiovascular, betacar
Digestive, lutein
Digestive, cryptox
Digestive, lycopene
Digestive, alphacar
Density
0.0
1.0

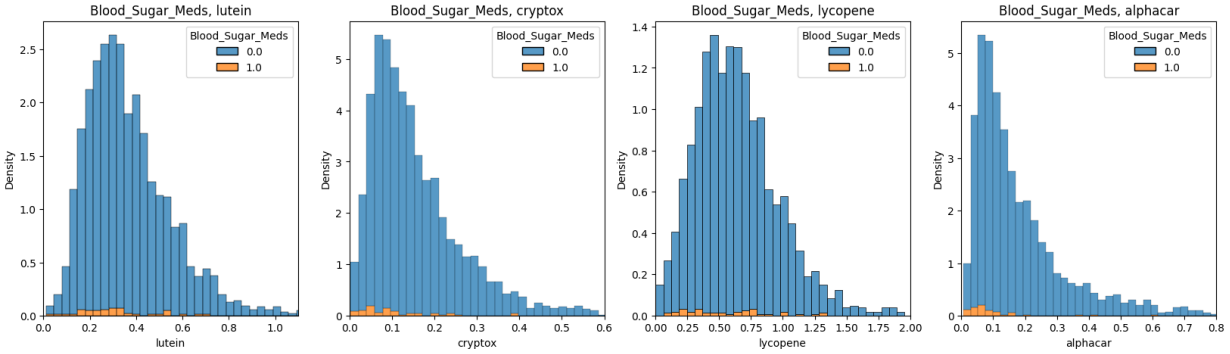

lutein
cryptox
lycopene
alphacar


Digestive, betacar
Arthritis, lutein
Arthritis, cryptox
Arthritis, lycopene
Digestive
Arthritis
0.0
1.0
Density
betacar
lutein
cryptox
lycopene


Arthritis, alphacar
Arthritis, betacar
Osteoporosis, lutein
Osteoporosis, cryptox
Arthritis
Osteoporosis
0.0
1.0
Density
alphacar
betacar
lutein
cryptox


Osteoporosis, lycopene
Osteoporosis, alphacar
Osteoporosis, betacar
Misc_Cond, lutein
Osteoporosis
Misc_Cond
0.0
1.0
Density
lycopene
alphacar
betacar
lutein


Misc_Cond, cryptox
Misc_Cond, lycopene
Misc_Cond, alphacar
Misc_Cond, betacar
Misc_Cond
0.0
1.0
Density
cryptox
lycopene
alphacar
betacar


Blood_Sugar_Meds, lutein
Blood_Sugar_Meds, cryptox
Blood_Sugar_Meds, lycopene
Blood_Sugar_Meds, alphacar
Blood_Sugar_Meds
0.0
1.0
Density
lutein
cryptox
lycopene
alphacar

Blood_Sugar_Meds, betacar
Cardio_Meds, lutein
Cardio_Meds, cryptox
Cardio_Meds, lycopene

In [38]:
```
# Merging thoughts/feelings and lifestyle

tfls = tf.merge(right=ls, how="inner", on="ID")
tfls.columns
tfls.shape
```

Out[38]:
```
(2861, 67)
```

In [39]:
```
tfls.dropna(inplace=True)
tfls.shape
```

Out[39]:
```
(2086, 67)
```

In [40]:
```
tfls.columns
```

Out[40]:
```
Index(['ID', 'ss_tot', 'ss_emot', 'ss_aff', 'ss_tang', 'ss_pos', 'lv_alone',
       'petownr', 'church', 'religion', 'club', 'caregvB', 'strain',
       'optimism', 'nee', 'age', 'hostil', 'QOL', 'physhealth', 'physfctn',
       'lim_phys', 'pain', 'genhlth', 'mental health', 'lim_emot', 'energy',
       'emotwell', 'socfctn', 'symptom', 'bloat', 'sweats', 'aches',
       'hotflash', 'tired', 'totlfevt', 'dep_log', 'whi_irs_old',
       'WHI_irs_new', 'main meals', 'snacks', 'meals out', 'cooking time',
       'adults', 'children', 'I cook?', 'cooking help', 'I clean up?',
       'cleanup help', 'shopping freq', 'grocery bill', 'I buy food?',
       'shopping help', 'weight', 'time at weight', 'pre-BC weight',
       'age18 weight', 'max weight', 'time at max wgt', 'health insurance',
       'insurance change?', 'insurance satisfaction', 'years in state',
       'where born', 'age15 residence', 'age35 residence', 'age50 residence',
       'longest residence'],
      dtype='object')
```

In [41]:
```
# tfls.rename(columns={'ss_tot':'total'})
```

In [42]:
```
tfls = tfls[tfls['longest residence'] != ' ']
```

In [43]:
```
# Convert string-type columns in tfls to floats

# for col in tfls.columns:
#   if not isinstance(list(tfls[col])[0], float):
#     for x in list(tfls[col]):
#       if isinstance(x, str):
#         print(col)
#         print(f"{x} +{x}+++ {len(x)}")
#       new_x = np.float64(x)

tfls = tfls.astype(float)
```

In [44]:
```
tfls.QOL
```

Out[44]:

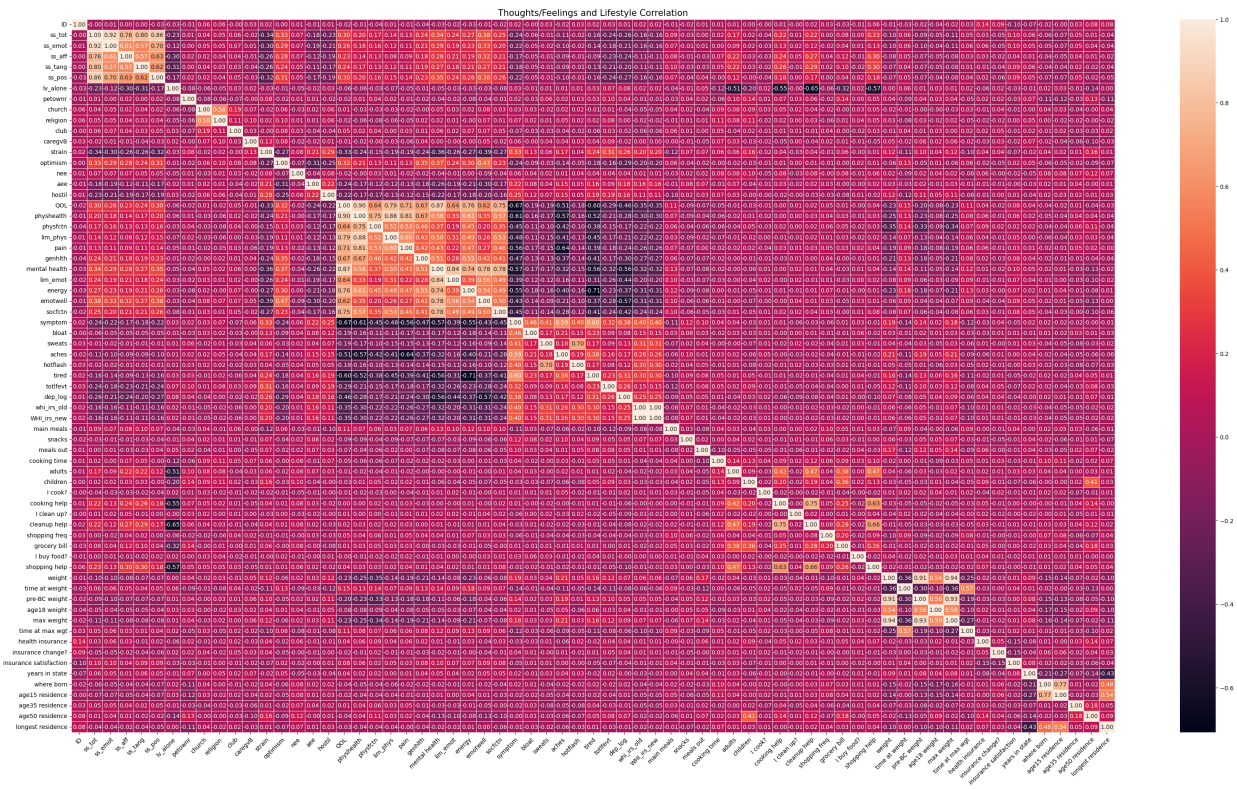
```
1       85.062500
2       74.957031
8       79.019531
9       68.062500
12      89.062500
           ...
2853    93.125000
2855    93.187500
2856    77.269531
2857    92.875000
2859    73.062500
Name: QOL, Length: 2083, dtype: float64
```

In [45]:

```
# Visualize tfls

plt.figure(figsize = (40, 22))
ax = sns.heatmap(tfls.corr(), annot = True, fmt = '.2f', linewidth = 0.5)
props = plt.setp(ax.xaxis.get_majorticklabels(), rotation=45, ha='right', rotation_moc
plt.title('Thoughts/Feelings and Lifestyle Correlation', size = 15)
```

Out[45]:

```
Text(0.5, 1.0, 'Thoughts/Feelings and Lifestyle Correlation')
```

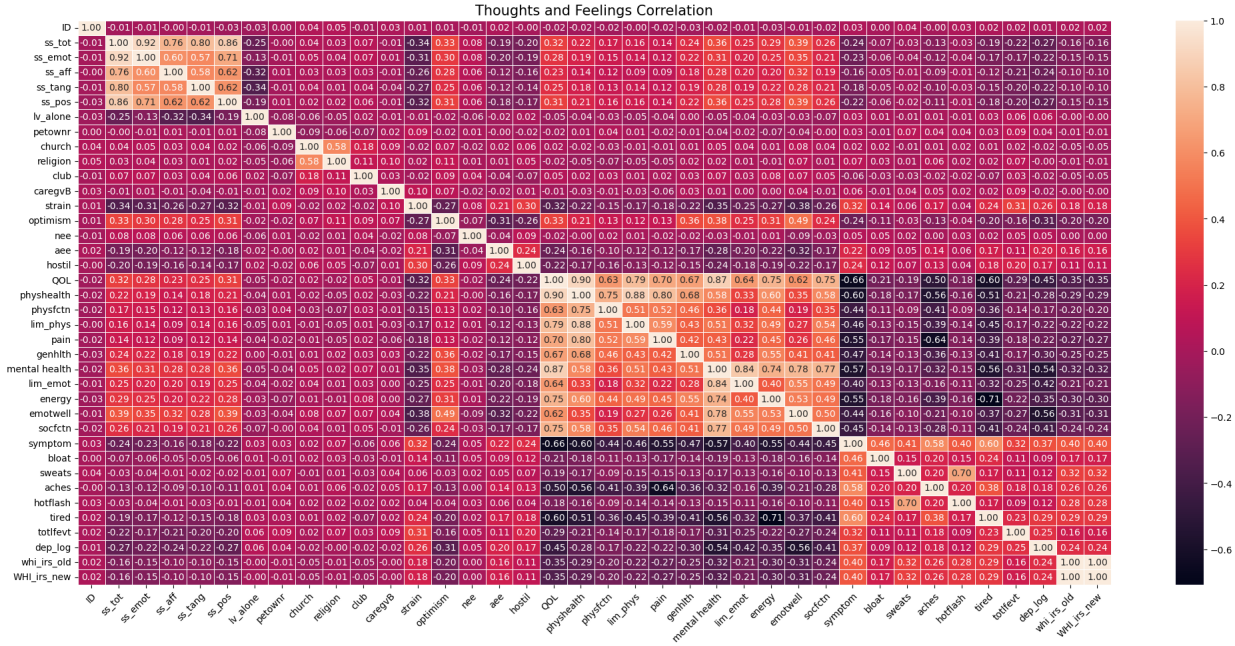


In [46]:

```
# Visualize tf

plt.figure(figsize = (25, 11))
ax = sns.heatmap(tf.corr(), annot = True, fmt = '.2f', linewidth = 0.5)
props = plt.setp(ax.xaxis.get_majorticklabels(), rotation=45, ha='right', rotation_moc
plt.title('Thoughts and Feelings Correlation', size = 15)
```

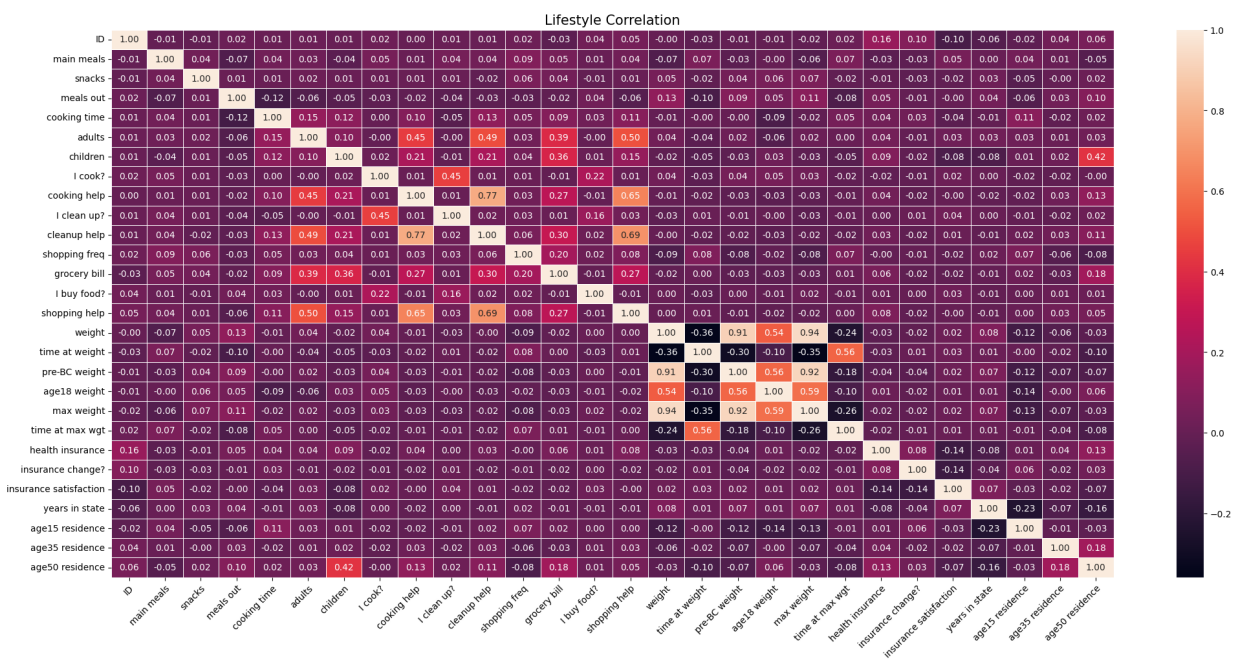
Out[46]:

```
Text(0.5, 1.0, 'Thoughts and Feelings Correlation')
```



In [47]:
```
# Visualize ls

plt.figure(figsize = (25, 11))
ax = sns.heatmap(ls.corr(), annot = True, fmt = '.2f', linewidth = 0.5)
props = plt.setp(ax.xaxis.get_majorticklabels(), rotation=45, ha='right', rotation_moc
plt.title('Lifestyle Correlation', size = 15)
```

Out[47]:
```
Text(0.5, 1.0, 'Lifestyle Correlation')
```

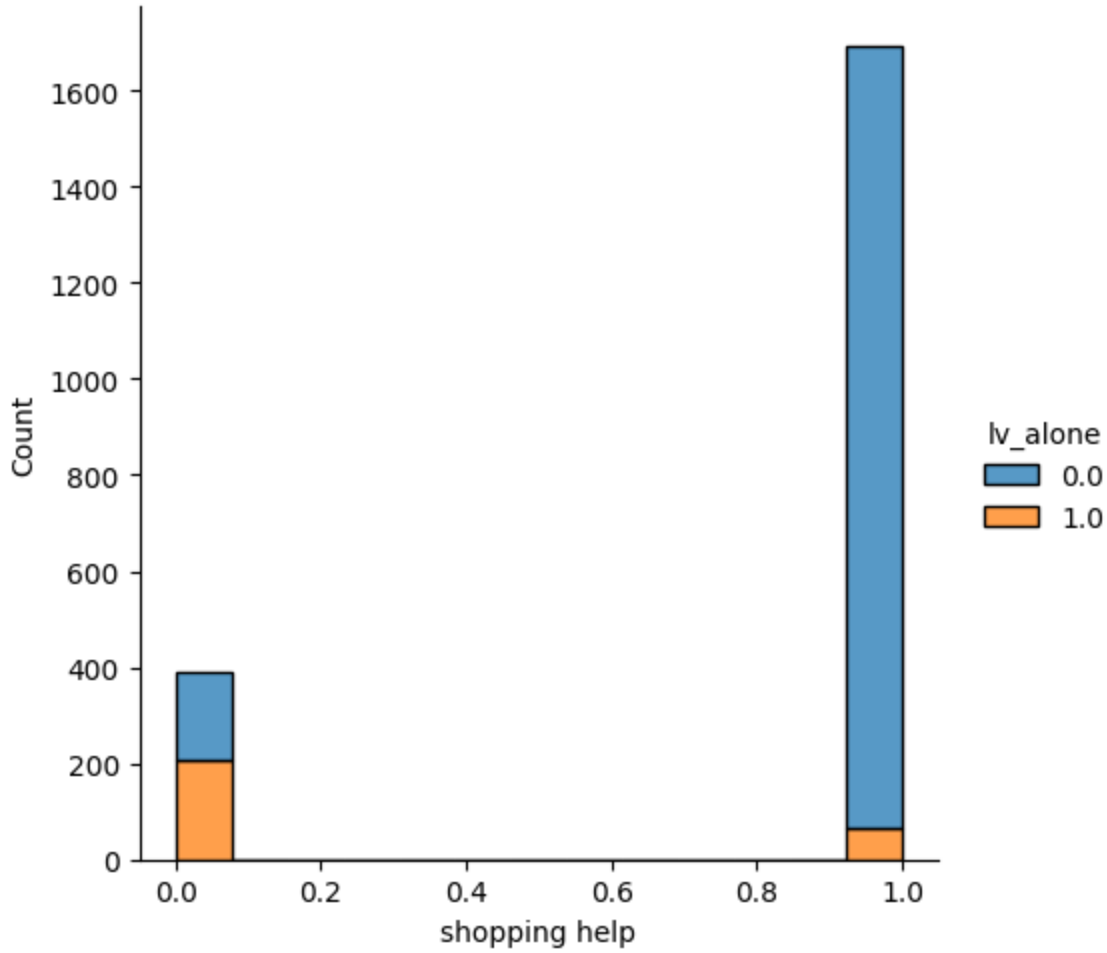


In [48]:
```
tfls['lv_alone'].unique()
```

Out[48]:
```
array([0., 1.])
```

In [49]:
```
sns.displot(data=tfls, x='shopping help', hue='lv_alone', multiple='stack')
```

Out[49]:
```
<seaborn.axisgrid.FacetGrid at 0x7fb005b4f7f0>
```



In [50]:
```
# Check for columns with <= 10 unique values to plot histograms with comparable condit

columns = []
for col in tfls.columns:
  if col != 'ID':
    length = len(tfls[col].unique())
    if length <= 10:
      columns.append(col)
      print(col)
      print(tfls[col].unique())
      print()
```

```
ss_aff
[5. 3. 4. 2. 1.]

ss_tang
[10.  9.  2.  4.  7.  8.  5.  6.  3.]

ss_pos
[ 9.  6. 10.  7.  8.  4.  5.  2.  3.]

lv_alone
[0. 1.]

petownr
[1. 0.]

church
[1. 2. 4. 3. 5. 6.]

religion
[2. 3. 1.]

club
[1. 3. 2. 5. 4. 6.]

caregvB
[0. 1. 4. 3. 2.]

lim_phys
[ 25.       100.         0.        50.        75.        33.328125
  66.65625 ]

pain
[ 87.5  75.   62.5  50.    0.  100.   37.5  25.   12.5]

lim_emot
[100.        66.65625   33.328125   0.      ]

socfctn
[100.   75.   87.5  62.5  37.5  50.   12.5  25.    0. ]

bloat
[0. 1. 2. 3.]

sweats
[0. 1. 3. 2.]

aches
[1. 3. 2. 0.]

hotflash
[1. 3. 0. 2.]

tired
[1. 2. 0. 3.]

totlfevt
[0. 4. 1. 2. 5. 3. 7. 6. 8.]

adults
[3. 2. 1. 4. 6. 5. 7.]
```

```
children
[ 2.  1.  0.  3. 12.  4.  5.  6.]

I cook?
[1. 0.]

cooking help
[1. 0.]

I clean up?
[1. 0.]

cleanup help
[1. 0.]

shopping freq
[3. 2. 1. 4.]

I buy food?
[1. 0.]

shopping help
[1. 0.]

health insurance
[5. 4. 1. 3. 7. 6. 8. 2.]

insurance change?
[1. 4. 5. 2. 3. 7. 6.]

insurance satisfaction
[6. 9. 2. 8. 7. 1. 5. 0. 4. 3.]

years in state
[3. 4. 2. 1.]
```

In [51]:

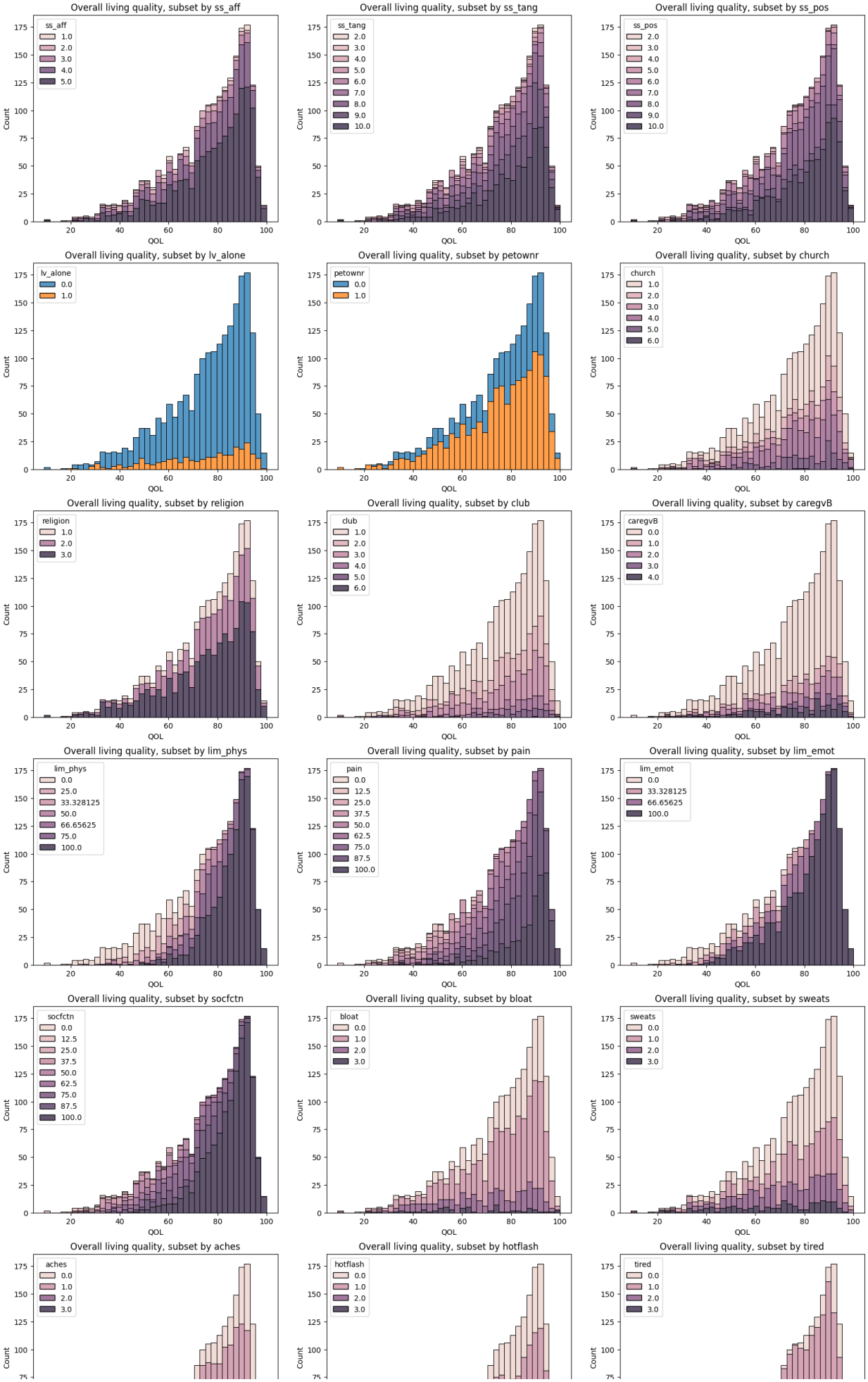
```
len(columns)
```

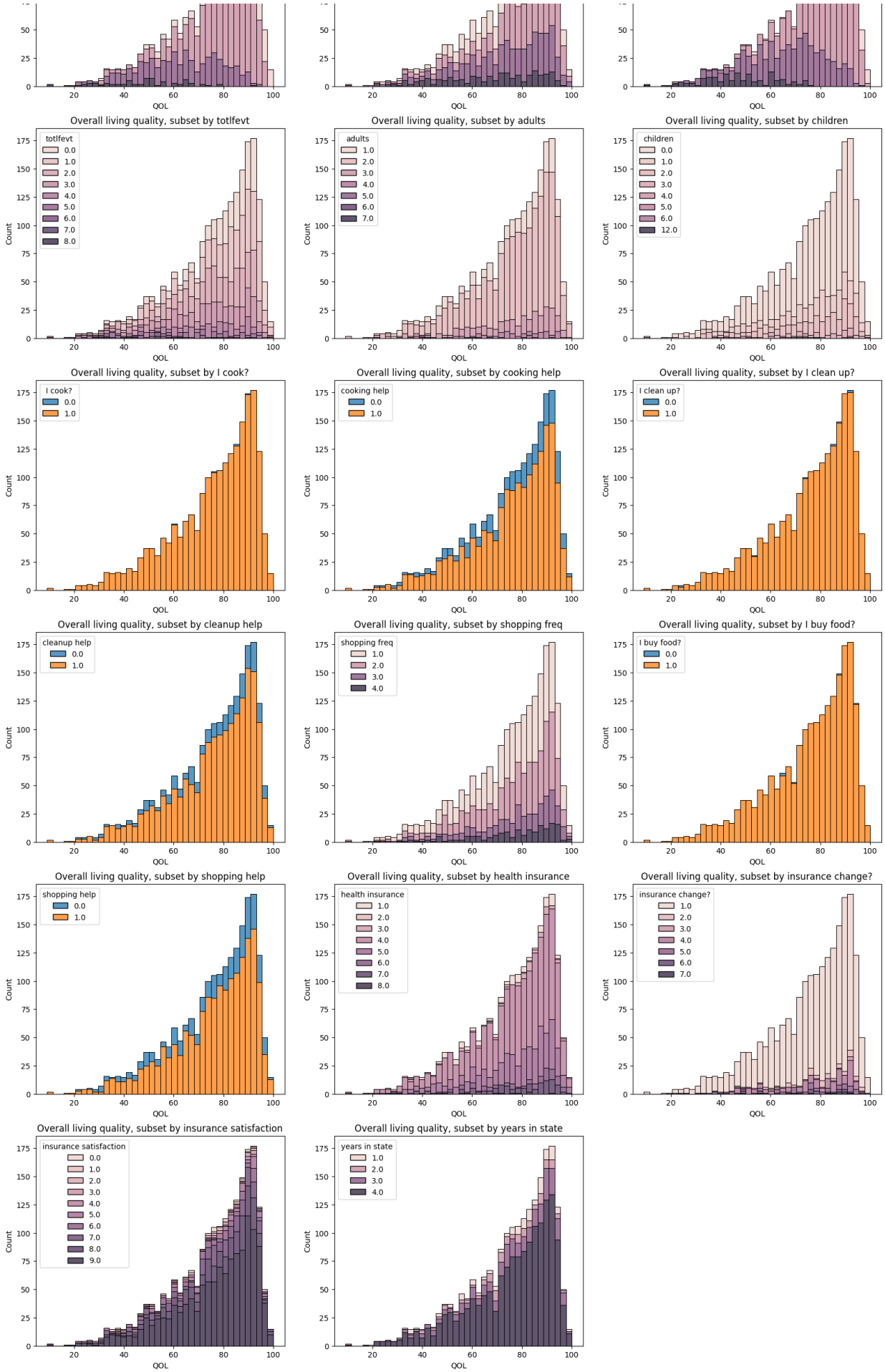
Out[51]:

```
32
```

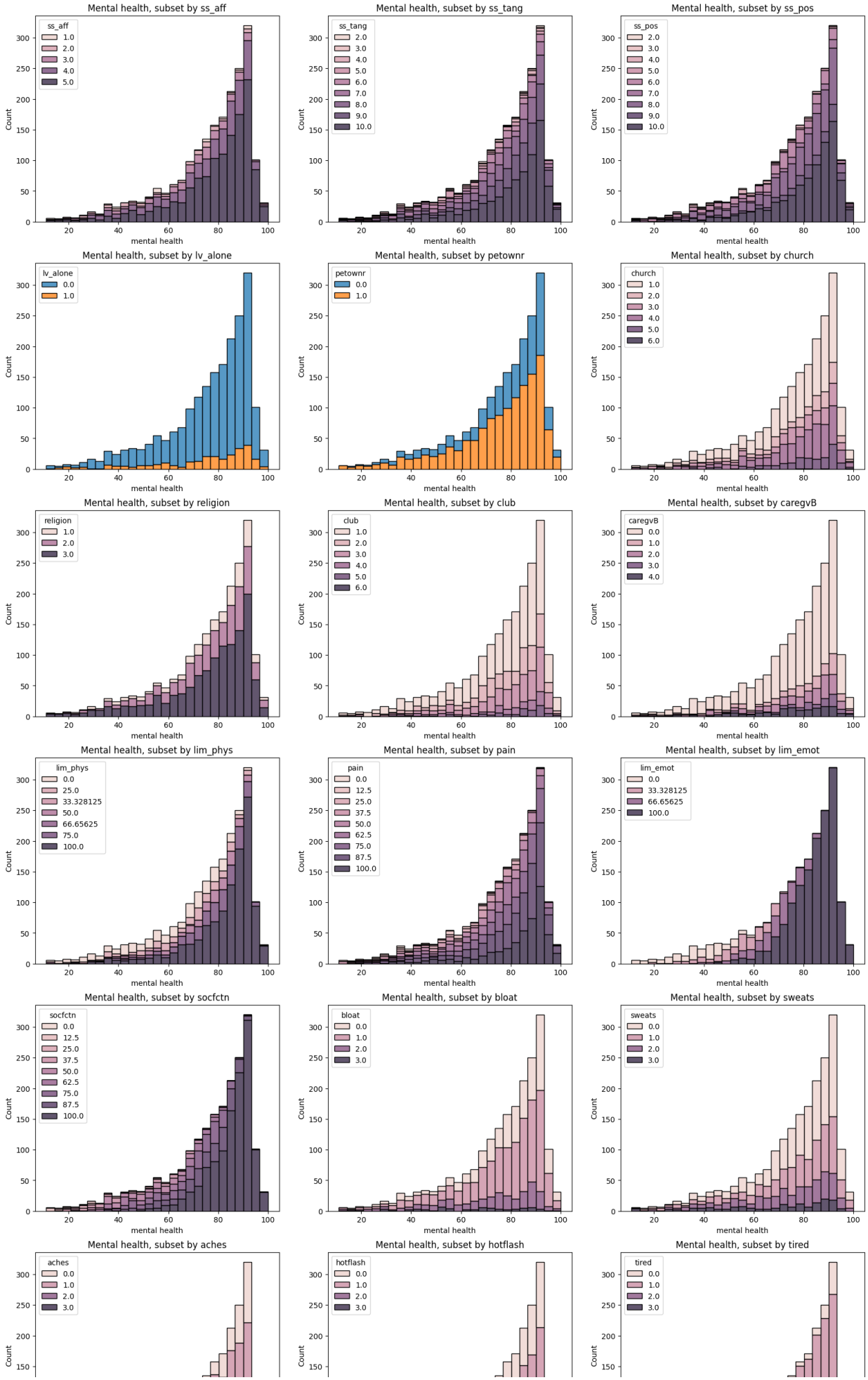
In [52]:

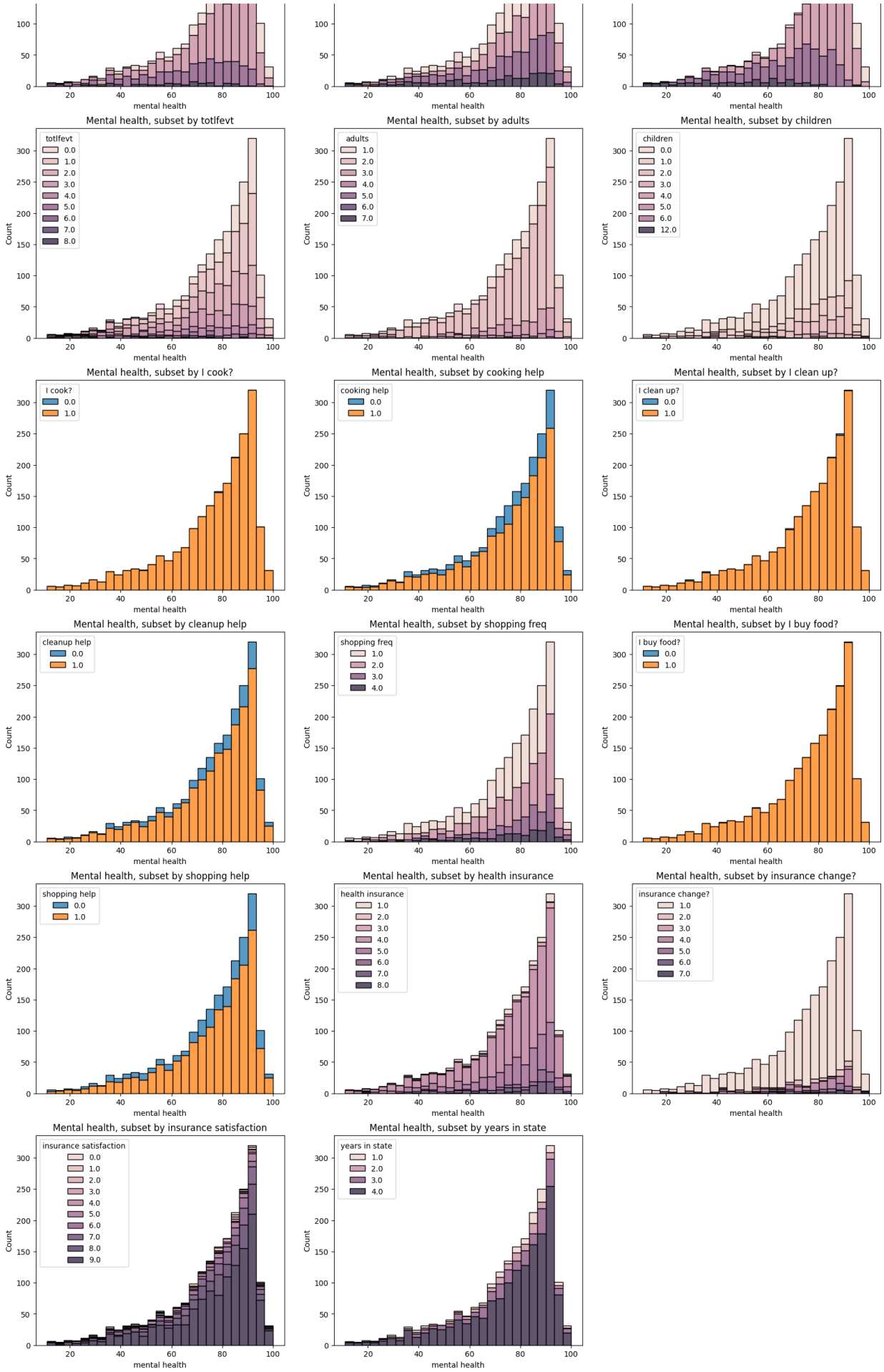
```
# Plotting the above

index = 1
plt.figure(figsize=(20, 65))
for col in columns:
  plt.subplot(11, 3, index)
  sns.histplot(x=tfls['QOL'], hue=tfls[col], multiple='stack', bins=40)
  plt.title(f"Overall living quality, subset by {col}")
  index += 1
```


Overall living quality, subset by ss_aff
ss_aff
1.0
2.0
3.0
4.0
5.0
Overall living quality, subset by ss_tang
ss_tang
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
10.0
Overall living quality, subset by ss_pos
ss_pos
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
10.0
Overall living quality, subset by lv_alone
lv_alone
0.0
1.0
Overall living quality, subset by petownr
petownr
0.0
1.0
Overall living quality, subset by church
church
1.0
2.0
3.0
4.0
5.0
6.0
Overall living quality, subset by religion
religion
1.0
2.0
3.0
Overall living quality, subset by club
club
1.0
2.0
3.0
4.0
5.0
6.0
Overall living quality, subset by caregvB
caregvB
0.0
1.0
2.0
3.0
4.0
Overall living quality, subset by lim_phys
lim_phys
0.0
25.0
33.328125
50.0
66.65625
75.0
100.0
Overall living quality, subset by pain
pain
0.0
12.5
25.0
37.5
50.0
62.5
75.0
87.5
100.0
Overall living quality, subset by lim_emot
lim_emot
0.0
33.328125
66.65625
100.0
Overall living quality, subset by socfctn
socfctn
0.0
12.5
25.0
37.5
50.0
62.5
75.0
87.5
100.0
Overall living quality, subset by bloat
bloat
0.0
1.0
2.0
3.0
Overall living quality, subset by sweats
sweats
0.0
1.0
2.0
3.0
Overall living quality, subset by aches
aches
0.0
1.0
2.0
3.0
Overall living quality, subset by hotflash
hotflash
0.0
1.0
2.0
3.0
Overall living quality, subset by tired
tired
0.0
1.0
2.0
3.0
Count
QOL


Overall living quality, subset by totlfevt
totlfevt
0.0
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
Overall living quality, subset by adults
adults
1.0
2.0
3.0
4.0
5.0
6.0
7.0
Overall living quality, subset by children
children
0.0
1.0
2.0
3.0
4.0
5.0
6.0
12.0
Overall living quality, subset by I cook?
I cook?
0.0
1.0
Overall living quality, subset by cooking help
cooking help
0.0
1.0
Overall living quality, subset by I clean up?
I clean up?
0.0
1.0
Overall living quality, subset by cleanup help
cleanup help
0.0
1.0
Overall living quality, subset by shopping freq
shopping freq
1.0
2.0
3.0
4.0
Overall living quality, subset by I buy food?
I buy food?
0.0
1.0
Overall living quality, subset by shopping help
shopping help
0.0
1.0
Overall living quality, subset by health insurance
health insurance
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
Overall living quality, subset by insurance change?
insurance change?
1.0
2.0
3.0
4.0
5.0
6.0
7.0
Overall living quality, subset by insurance satisfaction
insurance satisfaction
0.0
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
Overall living quality, subset by years in state
years in state
1.0
2.0
3.0
4.0
Count
QOL

In [53]:
```
# Plotting the above for general health

index = 1
plt.figure(figsize=(20, 65))
for col in columns:
  plt.subplot(11, 3, index)
  sns.histplot(x=tfls['mental health'], hue=tfls[col], multiple='stack')
  plt.title(f"Mental health, subset by {col}")
  index += 1
```


Mental health, subset by ss_aff
ss_aff
1.0
2.0
3.0
4.0
5.0
Mental health, subset by ss_tang
ss_tang
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
10.0
Mental health, subset by ss_pos
ss_pos
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
10.0
Mental health, subset by lv_alone
lv_alone
0.0
1.0
Mental health, subset by petownr
petownr
0.0
1.0
Mental health, subset by church
church
1.0
2.0
3.0
4.0
5.0
6.0
Mental health, subset by religion
religion
1.0
2.0
3.0
Mental health, subset by club
club
1.0
2.0
3.0
4.0
5.0
6.0
Mental health, subset by caregvB
caregvB
0.0
1.0
2.0
3.0
4.0
Mental health, subset by lim_phys
lim_phys
0.0
25.0
33.328125
50.0
66.65625
75.0
100.0
Mental health, subset by pain
pain
0.0
12.5
25.0
37.5
50.0
62.5
75.0
87.5
100.0
Mental health, subset by lim_emot
lim_emot
0.0
33.328125
66.65625
100.0
Mental health, subset by socfctn
socfctn
0.0
12.5
25.0
37.5
50.0
62.5
75.0
87.5
100.0
Mental health, subset by bloat
bloat
0.0
1.0
2.0
3.0
Mental health, subset by sweats
sweats
0.0
1.0
2.0
3.0
Mental health, subset by aches
aches
0.0
1.0
2.0
3.0
Mental health, subset by hotflash
hotflash
0.0
1.0
2.0
3.0
Mental health, subset by tired
tired
0.0
1.0
2.0
3.0
Count
mental health


Mental health, subset by totlfevt
totlfevt
0.0
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
Mental health, subset by adults
adults
1.0
2.0
3.0
4.0
5.0
6.0
7.0
Mental health, subset by children
children
0.0
1.0
2.0
3.0
4.0
5.0
6.0
12.0
Mental health, subset by I cook?
I cook?
0.0
1.0
Mental health, subset by cooking help
cooking help
0.0
1.0
Mental health, subset by I clean up?
I clean up?
0.0
1.0
Mental health, subset by cleanup help
cleanup help
0.0
1.0
Mental health, subset by shopping freq
shopping freq
1.0
2.0
3.0
4.0
Mental health, subset by I buy food?
I buy food?
0.0
1.0
Mental health, subset by shopping help
shopping help
0.0
1.0
Mental health, subset by health insurance
health insurance
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
Mental health, subset by insurance change?
insurance change?
1.0
2.0
3.0
4.0
5.0
6.0
7.0
Mental health, subset by insurance satisfaction
insurance satisfaction
0.0
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
Mental health, subset by years in state
years in state
1.0
2.0
3.0
4.0
Count
mental health

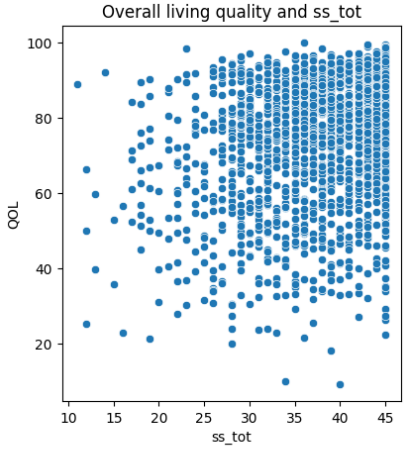
In [54]:
```
# Get columns not plotted above

scatter_columns = []
for col in tfls.columns:
  if col not in ['ID'] and col not in columns:
    scatter_columns.append(col)
len(scatter_columns)
```

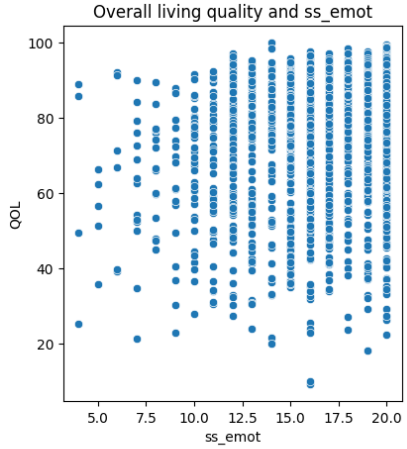
Out[54]:
```
34
```

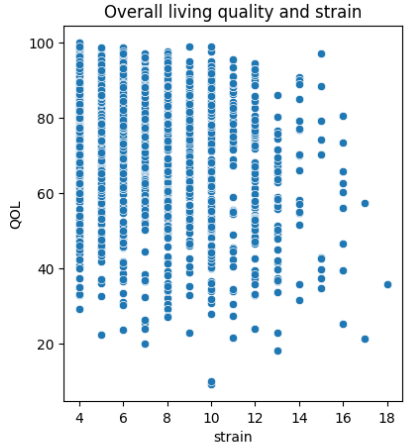
In [55]:
```
columns
```

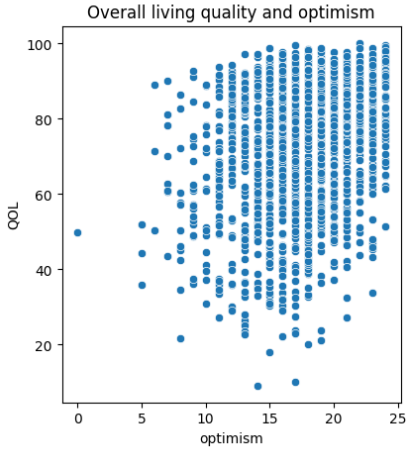
Out[55]:
```
['ss_aff',
 'ss_tang',
 'ss_pos',
 'lv_alone',
 'petownr',
 'church',
 'religion',
 'club',
 'caregvB',
 'lim_phys',
 'pain',
 'lim_emot',
 'socfctn',
 'bloat',
 'sweats',
 'aches',
 'hotflash',
 'tired',
 'totlfevt',
 'adults',
 'children',
 'I cook?',
 'cooking help',
 'I clean up?',
 'cleanup help',
 'shopping freq',
 'I buy food?',
 'shopping help',
 'health insurance',
 'insurance change?',
 'insurance satisfaction',
 'years in state']
```

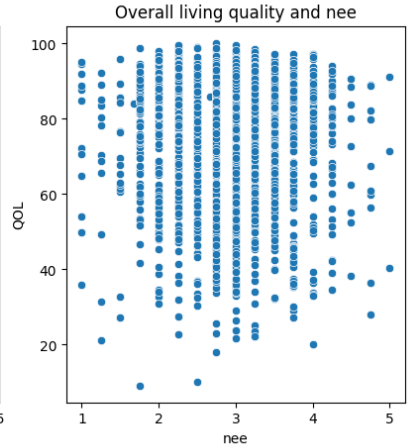
In [56]:
```
# Plotting the above

index = 1
plt.figure(figsize=(15, 70))
for col in scatter_columns[:-5]:
  plt.subplot(12, 3, index)
  sns.scatterplot(data=tfls, x=col, y='QOL')
  plt.title(f"Overall living quality and {col}")
  index += 1
```

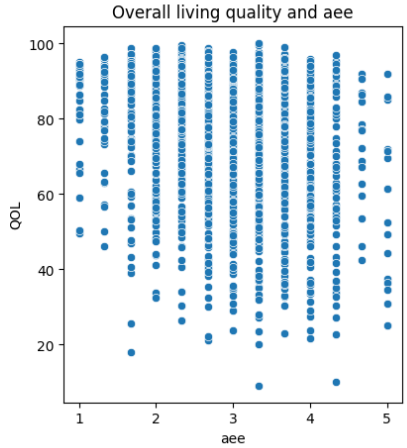

Overall living quality and ss_tot
QOL
ss_tot

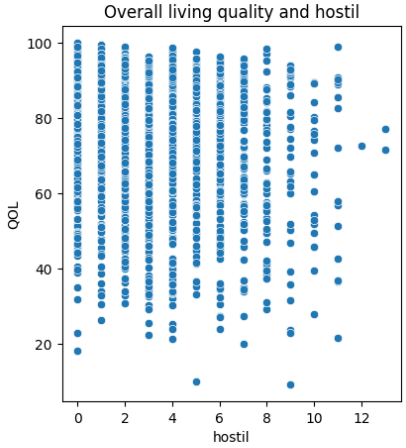

Overall living quality and ss_emot
QOL
ss_emot

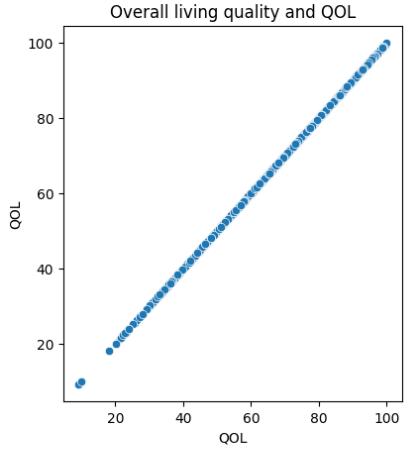

Overall living quality and strain
QOL
strain

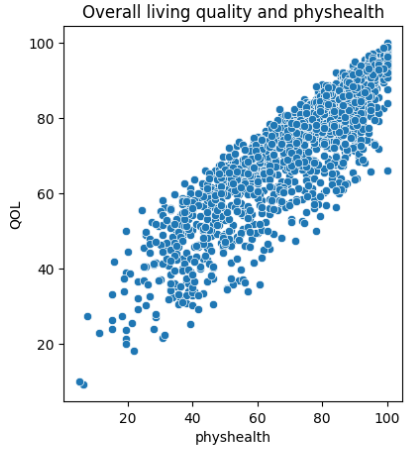

Overall living quality and optimism
QOL
optimism

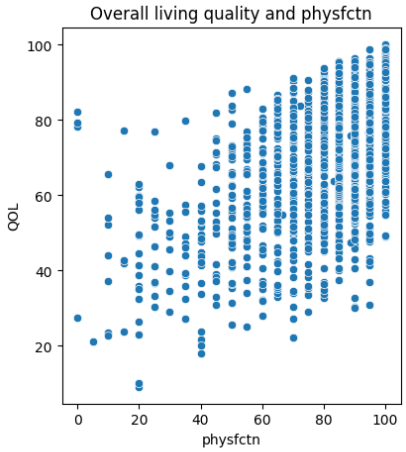

Overall living quality and nee
QOL
nee

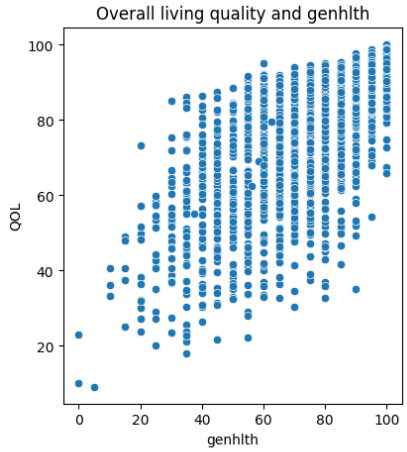

Overall living quality and aee
QOL
aee

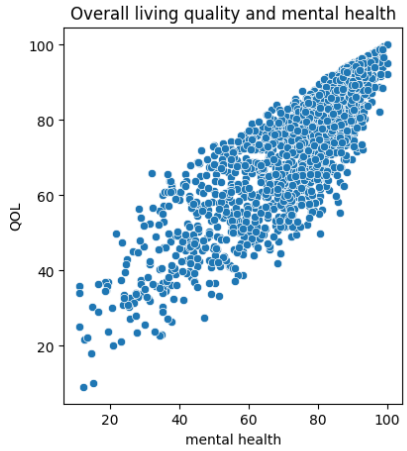

Overall living quality and hostil
QOL
hostil

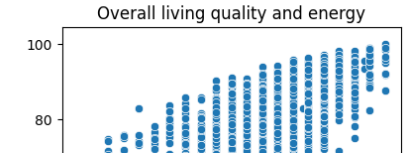

Overall living quality and QOL
QOL
QOL

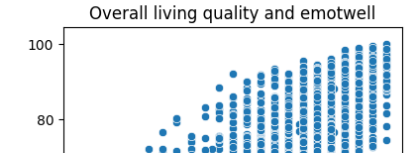

Overall living quality and physhealth
QOL
physhealth

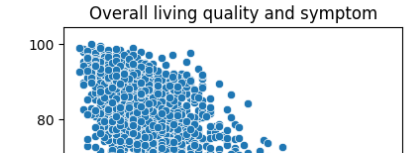

Overall living quality and physfctn
QOL
physfctn

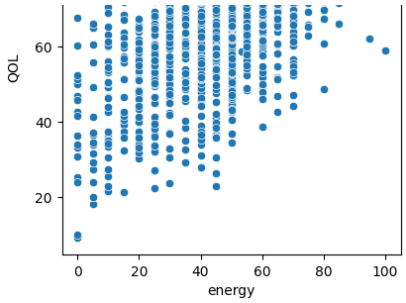

Overall living quality and genhlth
QOL
genhlth

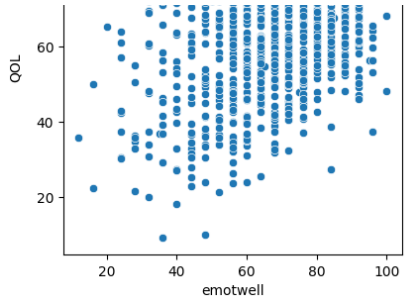

Overall living quality and mental health
QOL
mental health

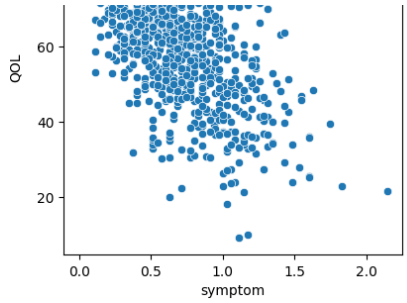

Overall living quality and energy

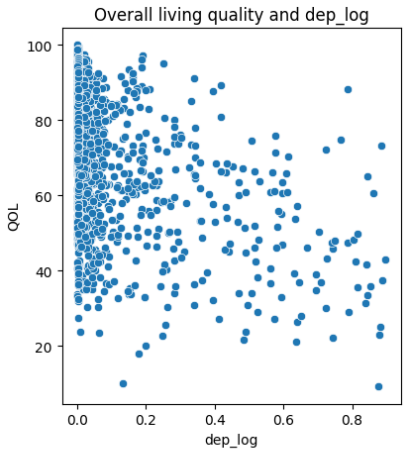

Overall living quality and emotwell

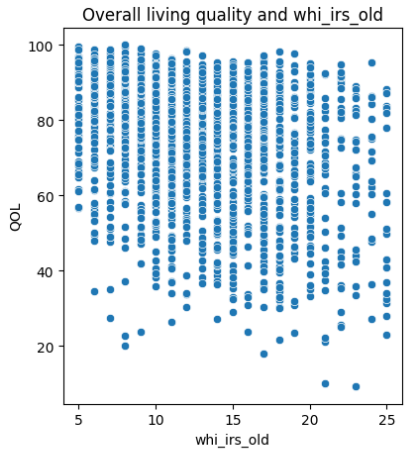

Overall living quality and symptom

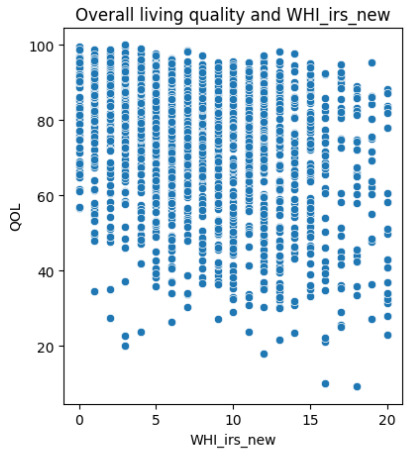

QOL
energy

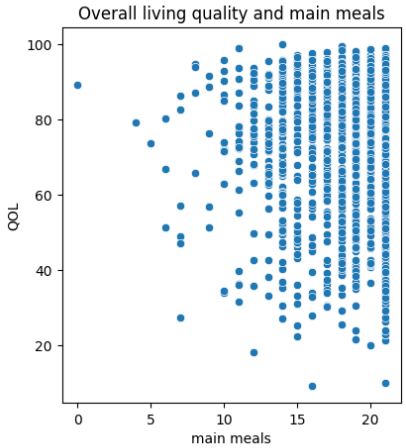

QOL
emotwell

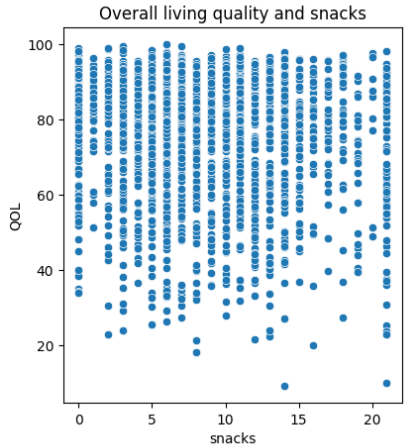

QOL
symptom

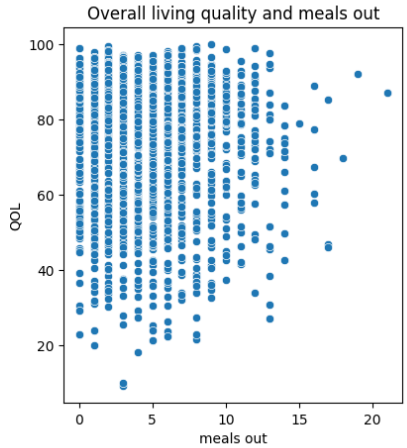

Overall living quality and dep_log
QOL
dep_log

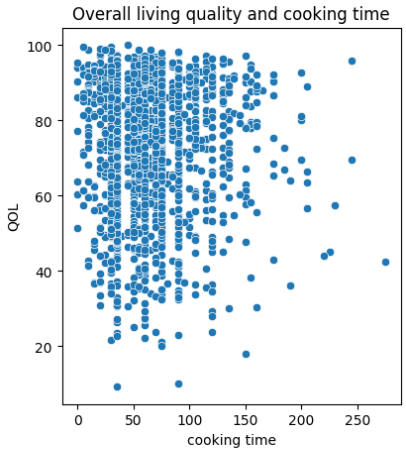

Overall living quality and whi_irs_old
QOL
whi_irs_old

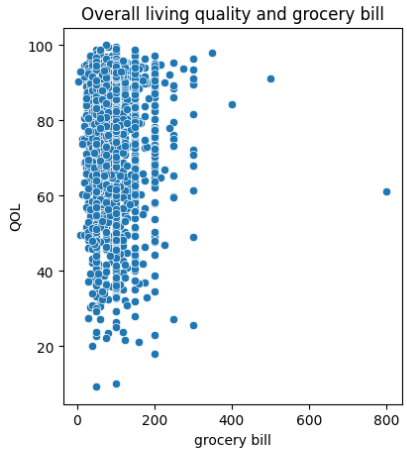

Overall living quality and WHI_irs_new
QOL
WHI_irs_new

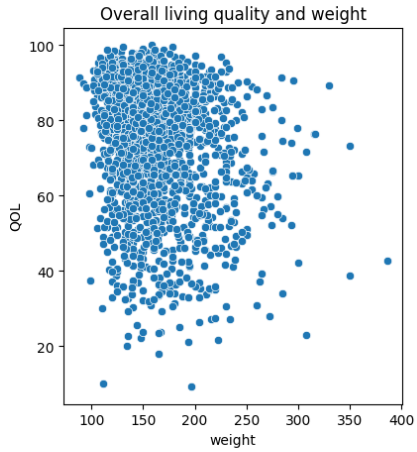

Overall living quality and main meals
QOL
main meals

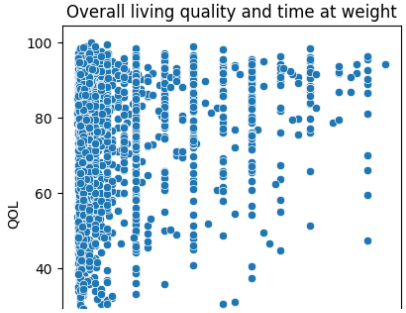

Overall living quality and snacks
QOL
snacks

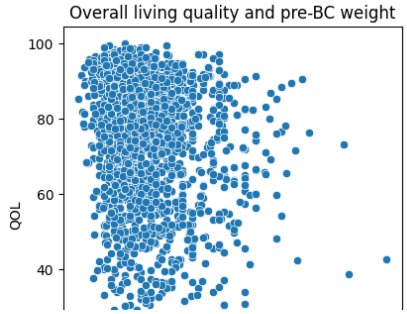

Overall living quality and meals out
QOL
meals out

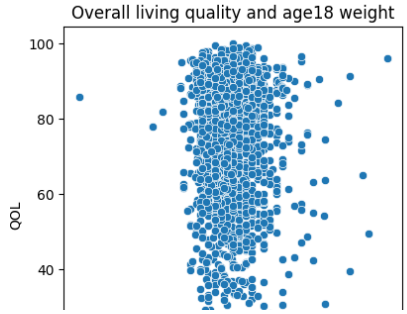

Overall living quality and cooking time
QOL
cooking time


Overall living quality and grocery bill
QOL
grocery bill

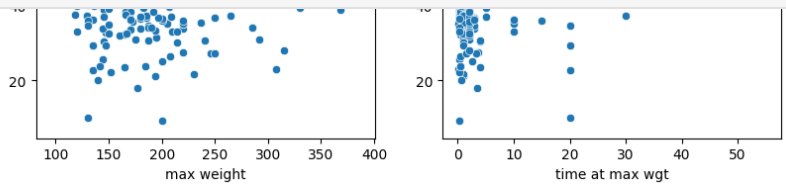

Overall living quality and weight
QOL
weight

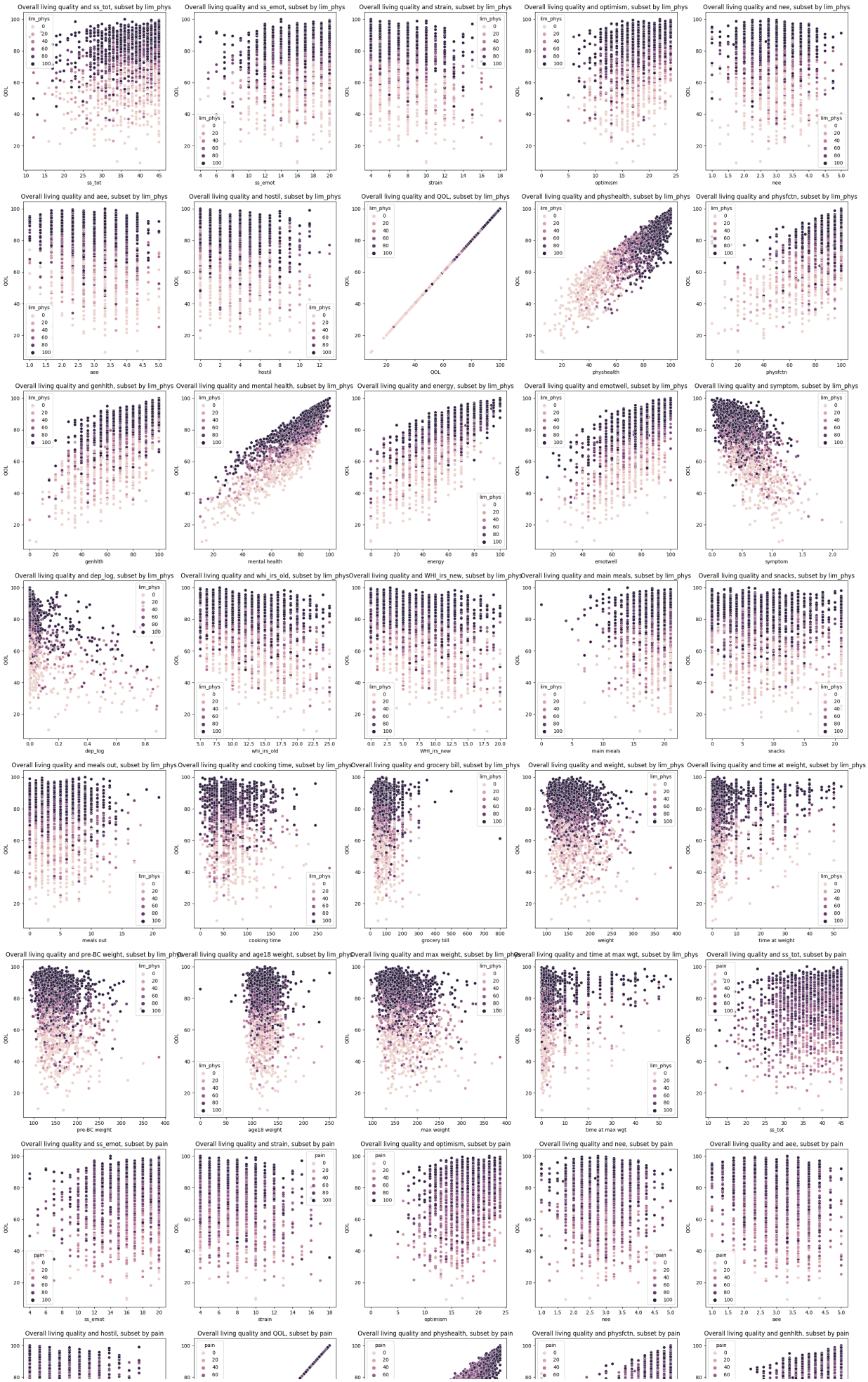

Overall living quality and time at weight
QOL

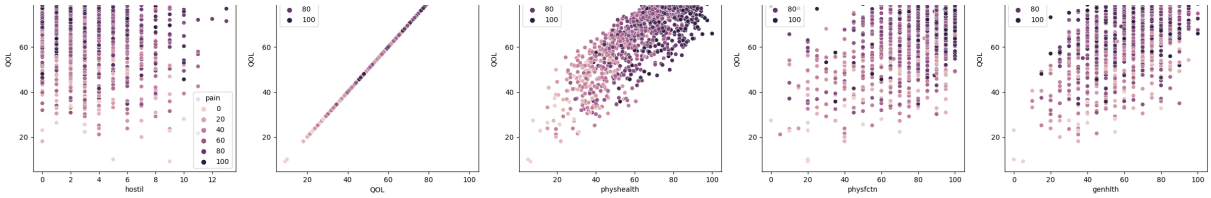

Overall living quality and pre-BC weight
QOL

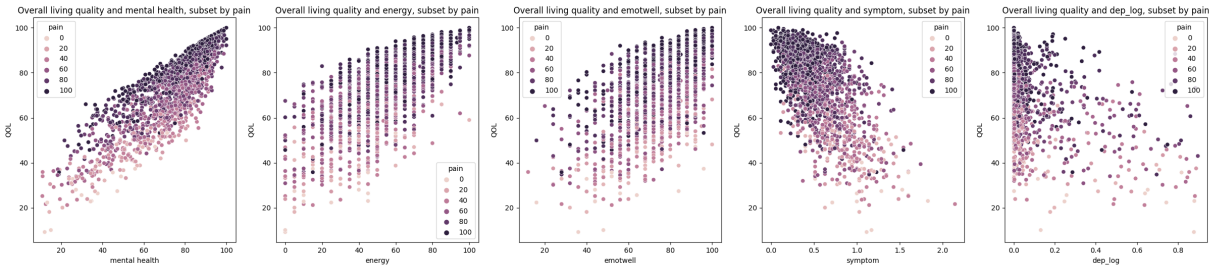

Overall living quality and age18 weight
QOL

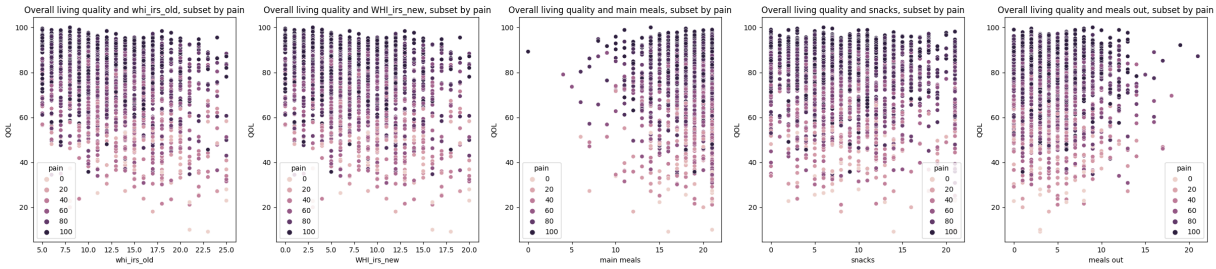
In [57]:
```
# Plotting the above, subset by variables in [columns]

subsets = ['lim_phys', 'pain', 'lim_emot', 'socfctn']
index = 1
plt.figure(figsize=(30, 200))
for subset in subsets:
  for col in scatter_columns[:-5]:
    plt.subplot(29, 5, index)
    sns.scatterplot(data=tfls, x=col, y='QOL', hue=subset)
    plt.title(f"Overall living quality and {col}, subset by {subset}")
    index += 1
```

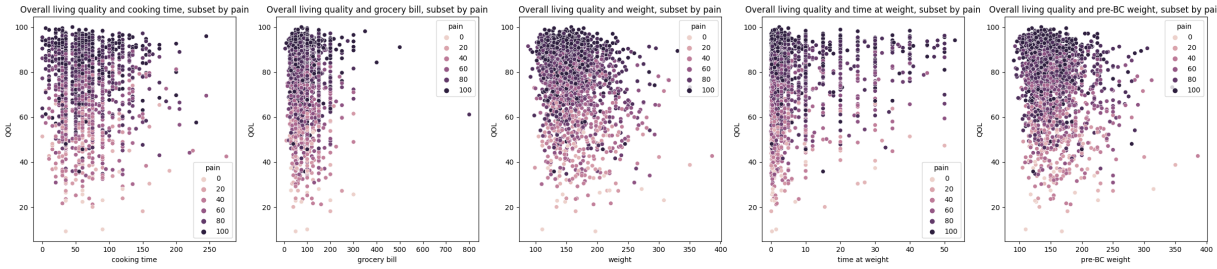


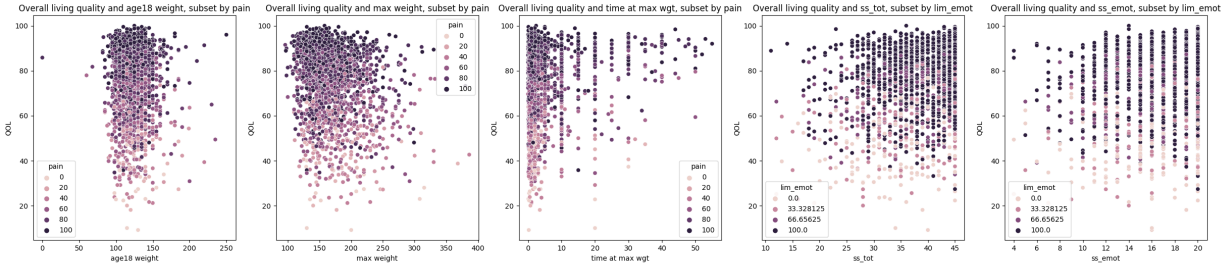

Overall living quality and ss_tot, subset by lim_phys
Overall living quality and ss_emot, subset by lim_phys
Overall living quality and strain, subset by lim_phys
Overall living quality and optimism, subset by lim_phys
Overall living quality and nee, subset by lim_phys
Overall living quality and aee, subset by lim_phys
Overall living quality and hostil, subset by lim_phys
Overall living quality and QOL, subset by lim_phys
Overall living quality and physhealth, subset by lim_phys
Overall living quality and physfctn, subset by lim_phys
Overall living quality and genhlth, subset by lim_phys
Overall living quality and mental health, subset by lim_phys
Overall living quality and energy, subset by lim_phys
Overall living quality and emotwell, subset by lim_phys
Overall living quality and symptom, subset by lim_phys
Overall living quality and dep_log, subset by lim_phys
Overall living quality and whi_irs_old, subset by lim_phys
Overall living quality and WHI_irs_new, subset by lim_phys
Overall living quality and main meals, subset by lim_phys
Overall living quality and snacks, subset by lim_phys
Overall living quality and meals out, subset by lim_phys
Overall living quality and cooking time, subset by lim_phys
Overall living quality and grocery bill, subset by lim_phys
Overall living quality and weight, subset by lim_phys
Overall living quality and time at weight, subset by lim_phys
Overall living quality and pre-BC weight, subset by lim_phys
Overall living quality and age18 weight, subset by lim_phys
Overall living quality and max weight, subset by lim_phys
Overall living quality and time at max wgt, subset by lim_phys
Overall living quality and ss_tot, subset by pain
Overall living quality and ss_emot, subset by pain
Overall living quality and strain, subset by pain
Overall living quality and optimism, subset by pain
Overall living quality and nee, subset by pain
Overall living quality and aee, subset by pain
Overall living quality and hostil, subset by pain
Overall living quality and QOL, subset by pain
Overall living quality and physhealth, subset by pain
Overall living quality and physfctn, subset by pain
Overall living quality and genhlth, subset by pain
QOL
lim_phys
pain
0
20
40
60
80
100
ss_tot
ss_emot
strain
optimism
nee
aee
hostil
QOL
physhealth
physfctn
genhlth
mental health
energy
emotwell
symptom
dep_log
whi_irs_old
WHI_irs_new
main meals
snacks
meals out
cooking time
grocery bill
weight
time at weight
pre-BC weight
age18 weight
max weight
time at max wgt

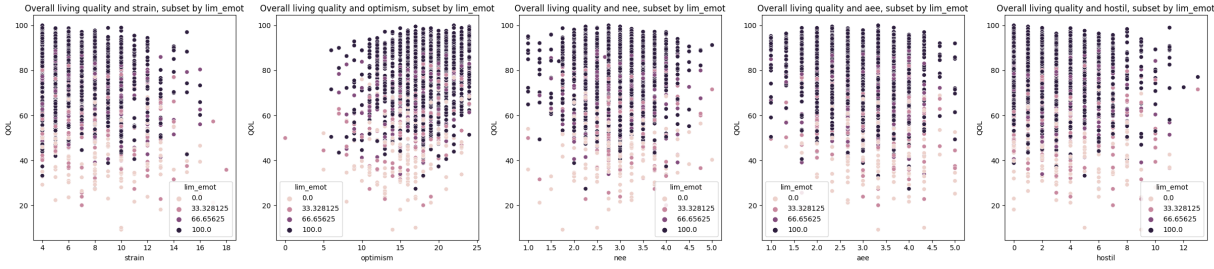

QOL
hostil
pain
0
20
40
60
80
100
QOL
physhealth
physfctn
genhlth

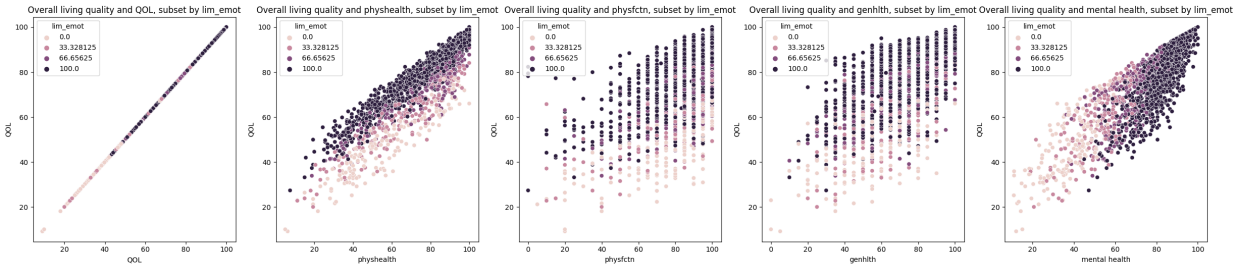

Overall living quality and mental health, subset by pain
Overall living quality and energy, subset by pain
Overall living quality and emotwell, subset by pain
Overall living quality and symptom, subset by pain
Overall living quality and dep_log, subset by pain
QOL
mental health
energy
emotwell
symptom
dep_log
pain

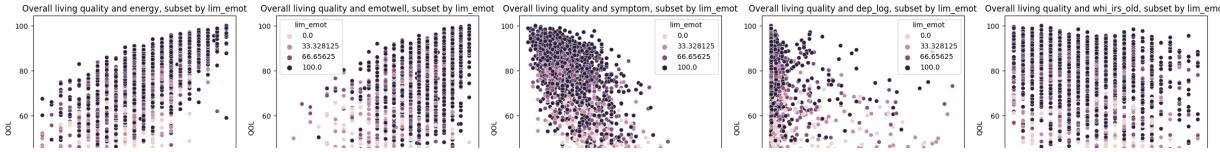

Overall living quality and whi_irs_old, subset by pain
Overall living quality and WHI_irs_new, subset by pain
Overall living quality and main meals, subset by pain
Overall living quality and snacks, subset by pain
Overall living quality and meals out, subset by pain
QOL
whi_irs_old
WHI_irs_new
main meals
snacks
meals out
pain


Overall living quality and cooking time, subset by pain
Overall living quality and grocery bill, subset by pain
Overall living quality and weight, subset by pain
Overall living quality and time at weight, subset by pain
Overall living quality and pre-BC weight, subset by pain
QOL
cooking time
grocery bill
weight
time at weight
pre-BC weight
pain


Overall living quality and age18 weight, subset by pain
Overall living quality and max weight, subset by pain
Overall living quality and time at max wgt, subset by pain
Overall living quality and ss_tot, subset by lim_emot
Overall living quality and ss_emot, subset by lim_emot
QOL
age18 weight
max weight
time at max wgt
ss_tot
ss_emot
pain
lim_emot
0.0
33.328125
66.65625
100.0

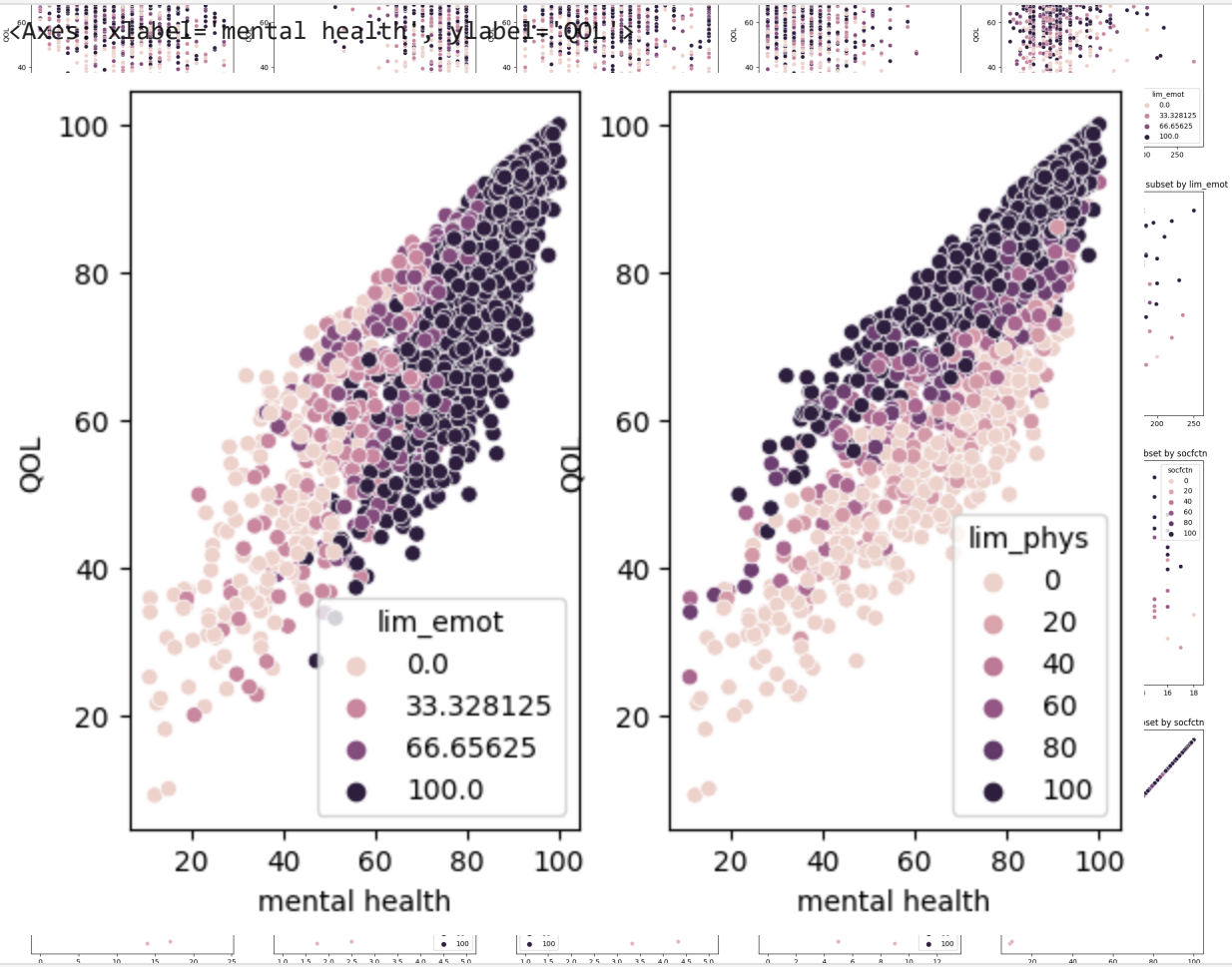

Overall living quality and strain, subset by lim_emot
Overall living quality and optimism, subset by lim_emot
Overall living quality and nee, subset by lim_emot
Overall living quality and aee, subset by lim_emot
Overall living quality and hostil, subset by lim_emot
QOL
strain
optimism
nee
aee
hostil
lim_emot
0.0
33.328125
66.65625
100.0


Overall living quality and QOL, subset by lim_emot
Overall living quality and physhealth, subset by lim_emot
Overall living quality and physfctn, subset by lim_emot
Overall living quality and genhlth, subset by lim_emot
Overall living quality and mental health, subset by lim_emot
QOL
physhealth
physfctn
genhlth
mental health
lim_emot
0.0
33.328125
66.65625
100.0

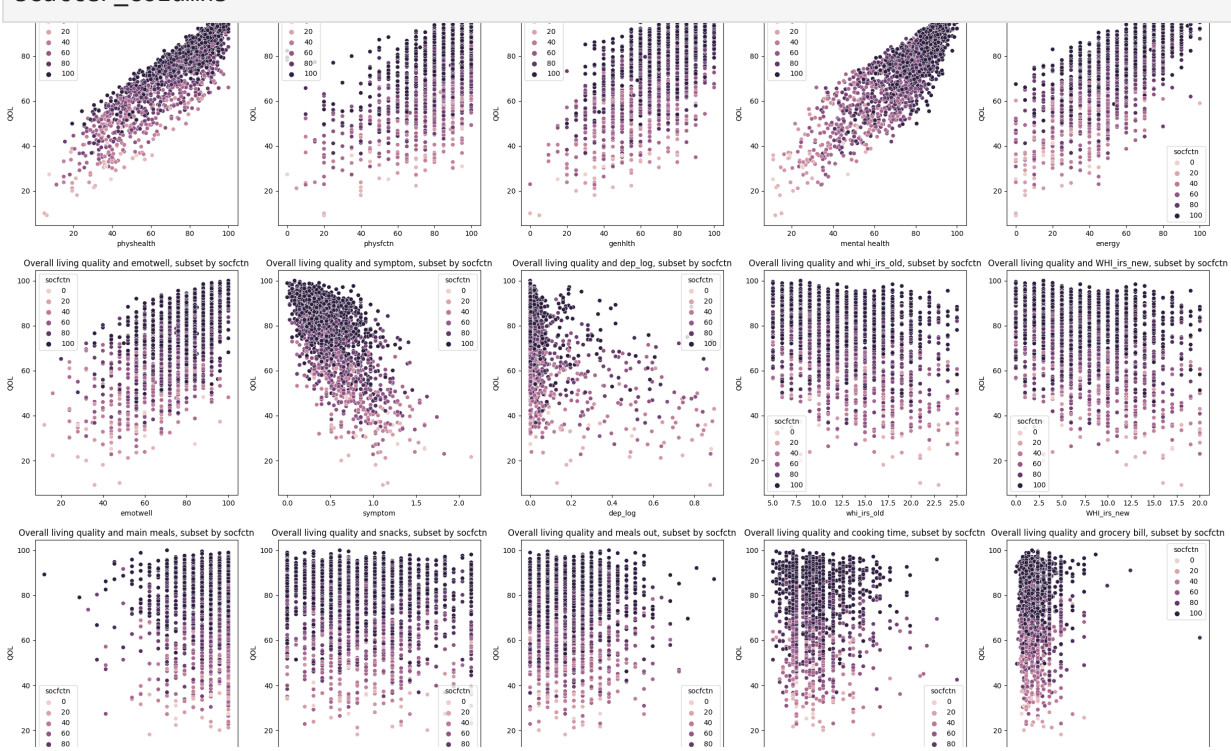

Overall living quality and energy, subset by lim_emot
Overall living quality and emotwell, subset by lim_emot
Overall living quality and symptom, subset by lim_emot
Overall living quality and dep_log, subset by lim_emot
Overall living quality and whi_irs_old, subset by lim_emot
QOL
lim_emot
0.0
33.328125
66.65625
100.0

In [58]:
```
plt.subplot(1, 2, 1)
sns.scatterplot(data=tfls, x='mental health', y='QOL', hue='lim_emot')

plt.subplot(1, 2, 2)
sns.scatterplot(data=tfls, x='mental health', y='QOL', hue='lim_phys')
```

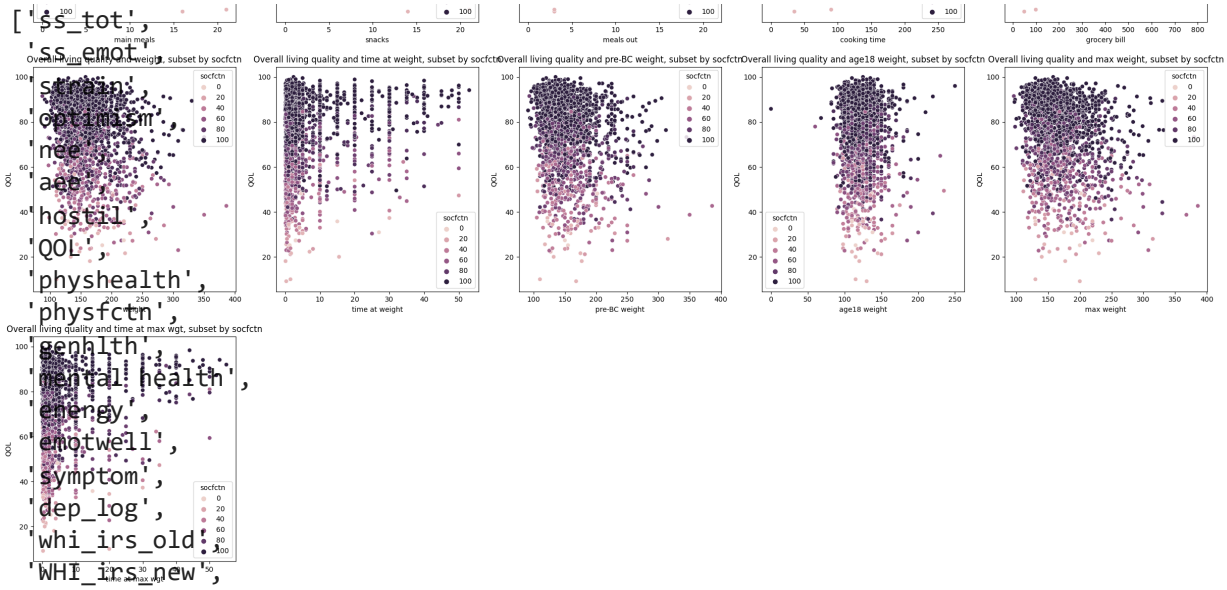
Out[58]:
```
<Axes: xlabel='mental health', ylabel='QOL'>
```



In [59]:
```
scatter_columns
```



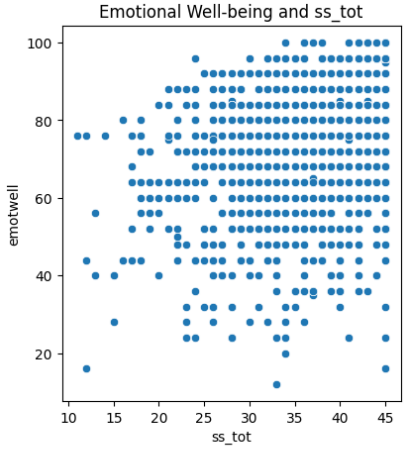
Out[59]:

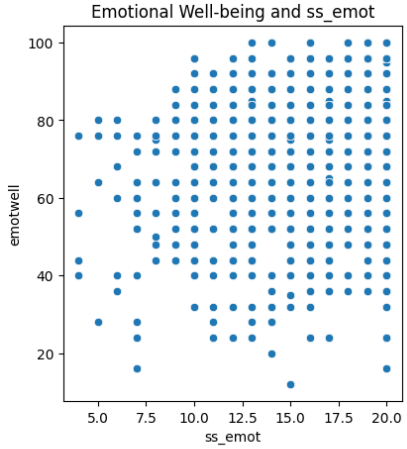


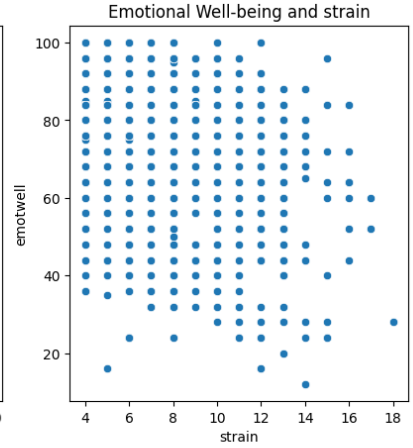

```
['ss_tot',
 'ss_emot',
 'strain',
 'optimism',
 'nee',
 'aee',
 'hostil',
 'QOL',
 'physhealth',
 'physfctn',
 'genhlth',
 'mental health',
 'energy',
 'emotwell',
 'symptom',
 'dep_log',
 'whi_irs_old',
 'WHI_irs_new',
 'main meals',
 'snacks',
 'meals out',
 'cooking time',
 'grocery bill',
 'weight',
 'time at weight',
 'pre-BC weight',
 'age18 weight',
 'max weight',
 'time at max wgt',
 'where born',
 'age15 residence',
 'age35 residence',
 'age50 residence',
 'longest residence']
```

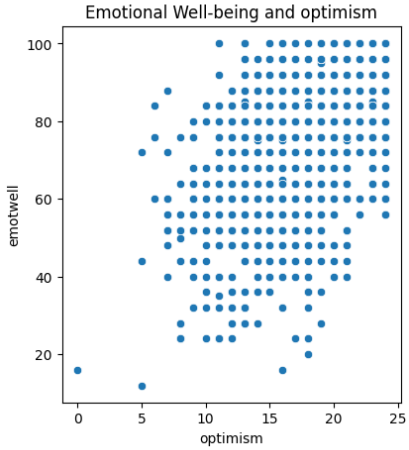
In [60]:

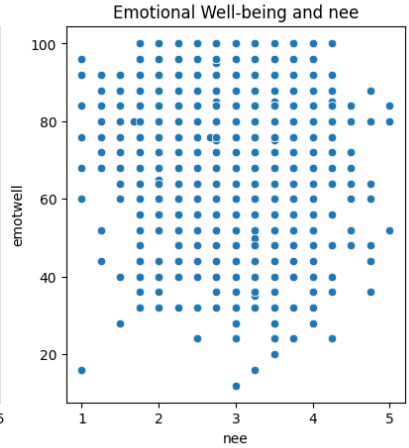
```
# Plotting general health and other vars in scatter_columns

index = 1
plt.figure(figsize=(15, 70))
for col in scatter_columns[:-5]:
  plt.subplot(12, 3, index)
  sns.scatterplot(data=tfls, x=col, y='emotwell')
  plt.title(f"Emotional Well-being and {col}")
  index += 1
```

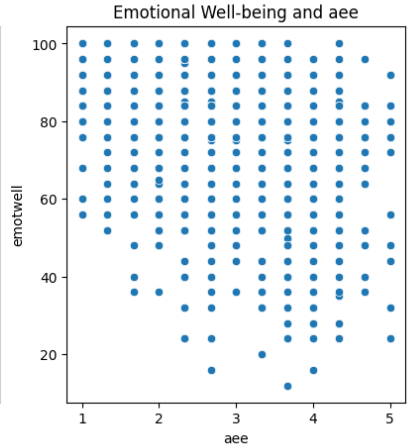

Emotional Well-being and ss_tot
emotwell
ss_tot

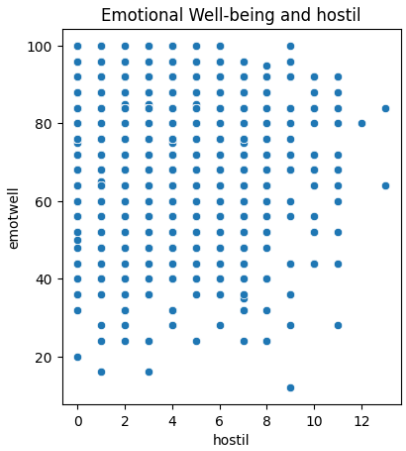

Emotional Well-being and ss_emot
emotwell
ss_emot

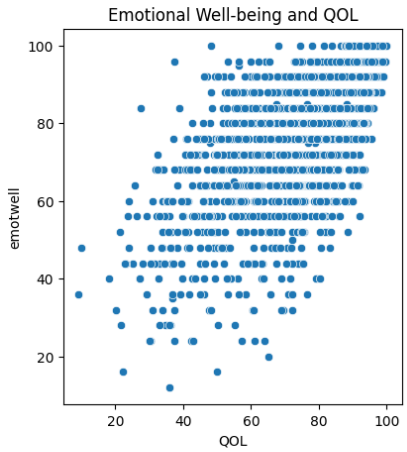

Emotional Well-being and strain
emotwell
strain

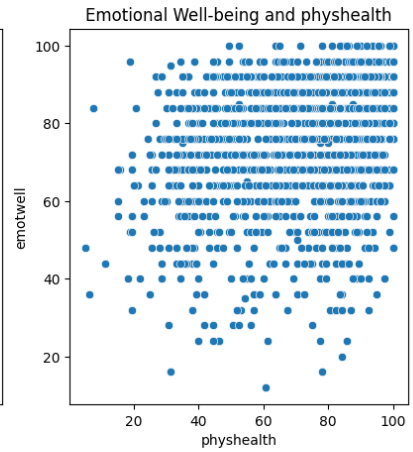

Emotional Well-being and optimism
emotwell
optimism

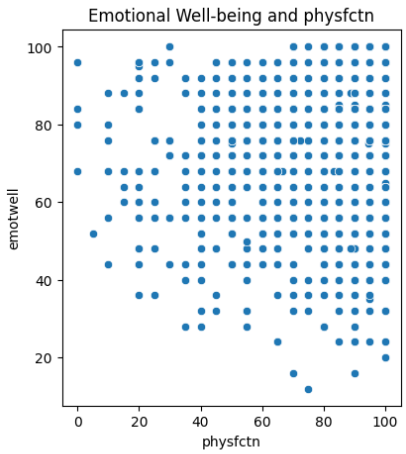

Emotional Well-being and nee
emotwell
nee

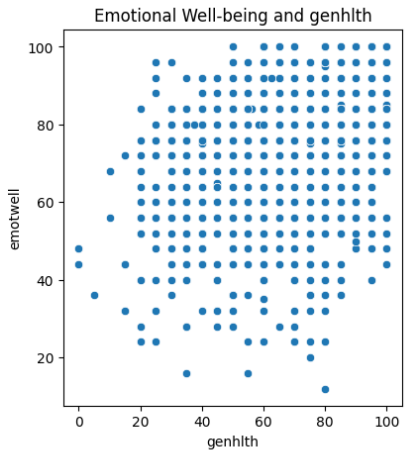

Emotional Well-being and aee
emotwell
aee

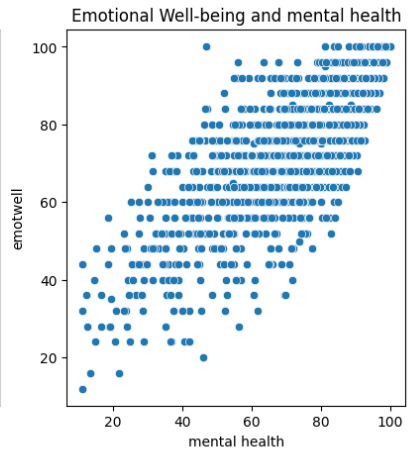

Emotional Well-being and hostil
emotwell
hostil

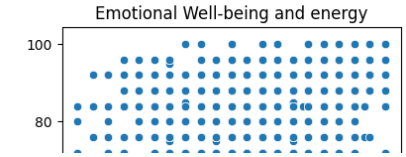

Emotional Well-being and QOL
emotwell
QOL

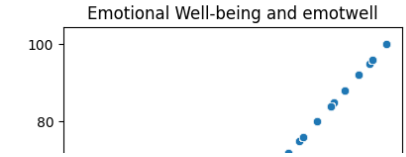

Emotional Well-being and physhealth
emotwell
physhealth


Emotional Well-being and physfctn
emotwell
physfctn

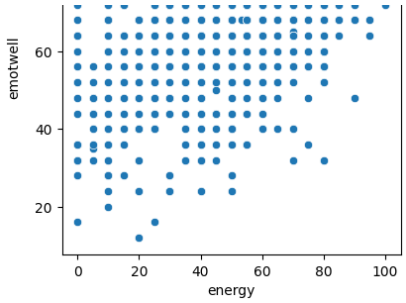

Emotional Well-being and genhlth
emotwell
genhlth

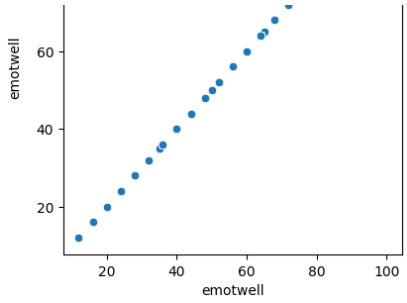

Emotional Well-being and mental health
emotwell
mental health

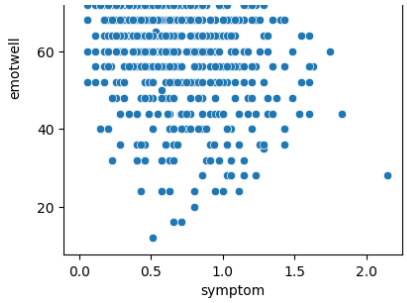

Emotional Well-being and energy


Emotional Well-being and emotwell

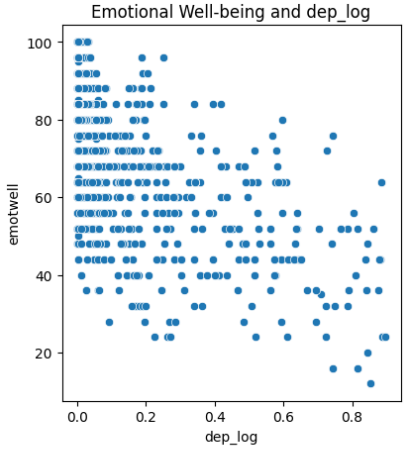
Emotional Well-being and symptom



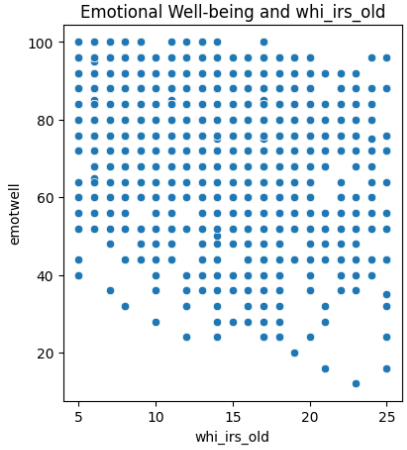




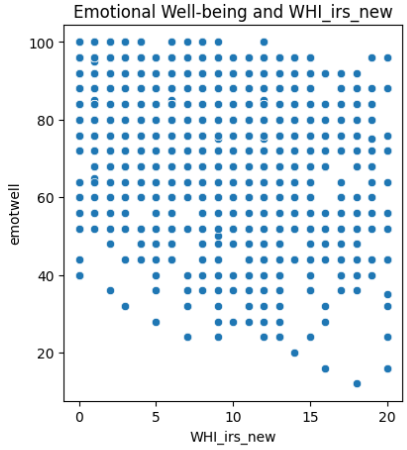
Emotional Well-being and dep_log



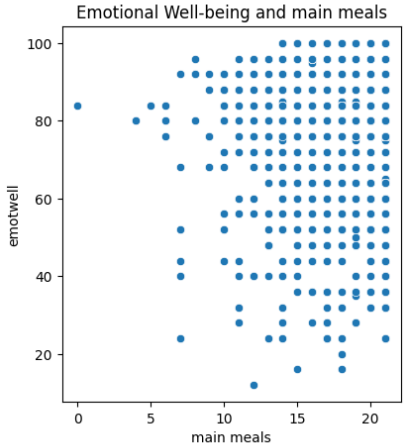
Emotional Well-being and whi_irs_old



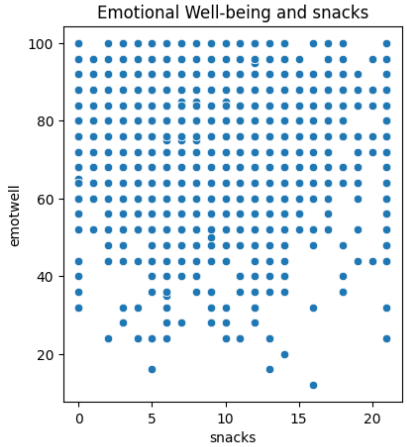
Emotional Well-being and WHI_irs_new



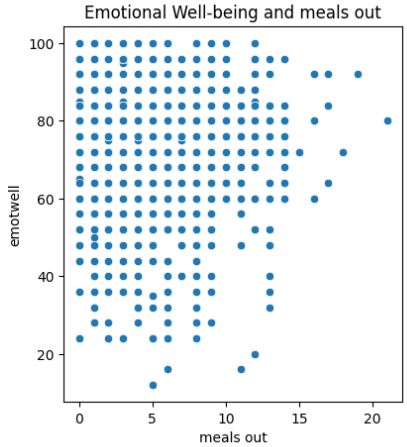
Emotional Well-being and main meals



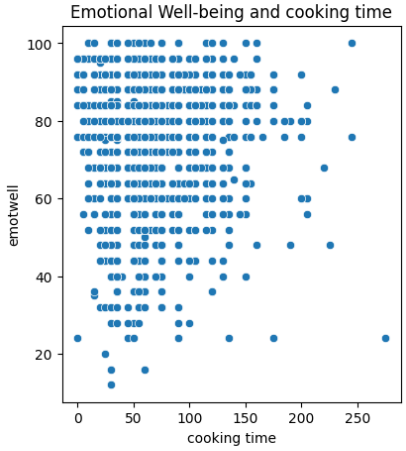
Emotional Well-being and snacks



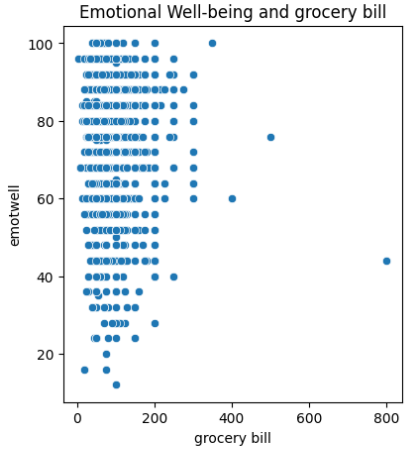
Emotional Well-being and meals out



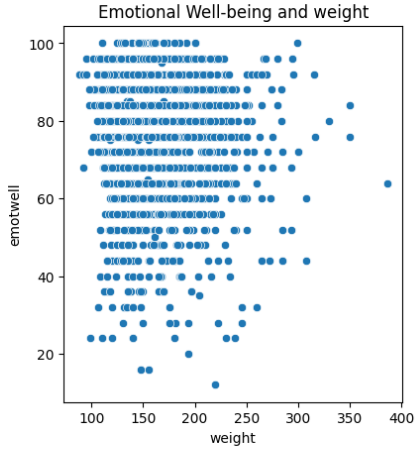
Emotional Well-being and cooking time



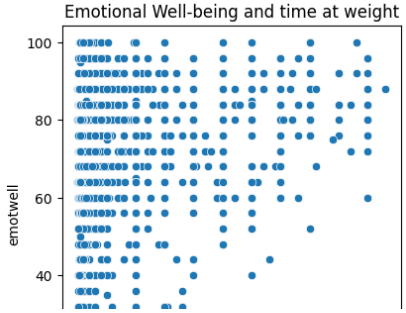
Emotional Well-being and grocery bill



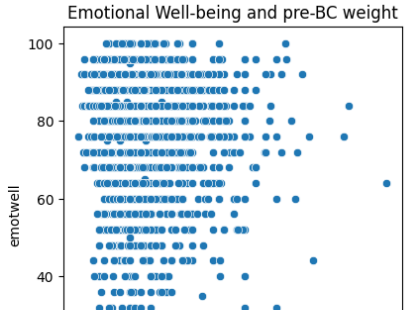
Emotional Well-being and weight



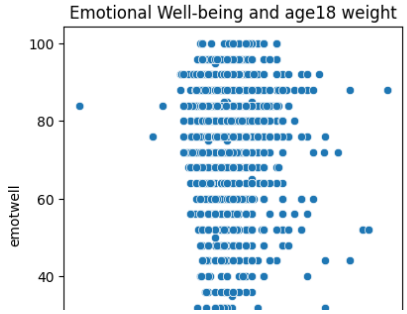
Emotional Well-being and time at weight



Emotional Well-being and pre-BC weight



Emotional Well-being and age18 weight

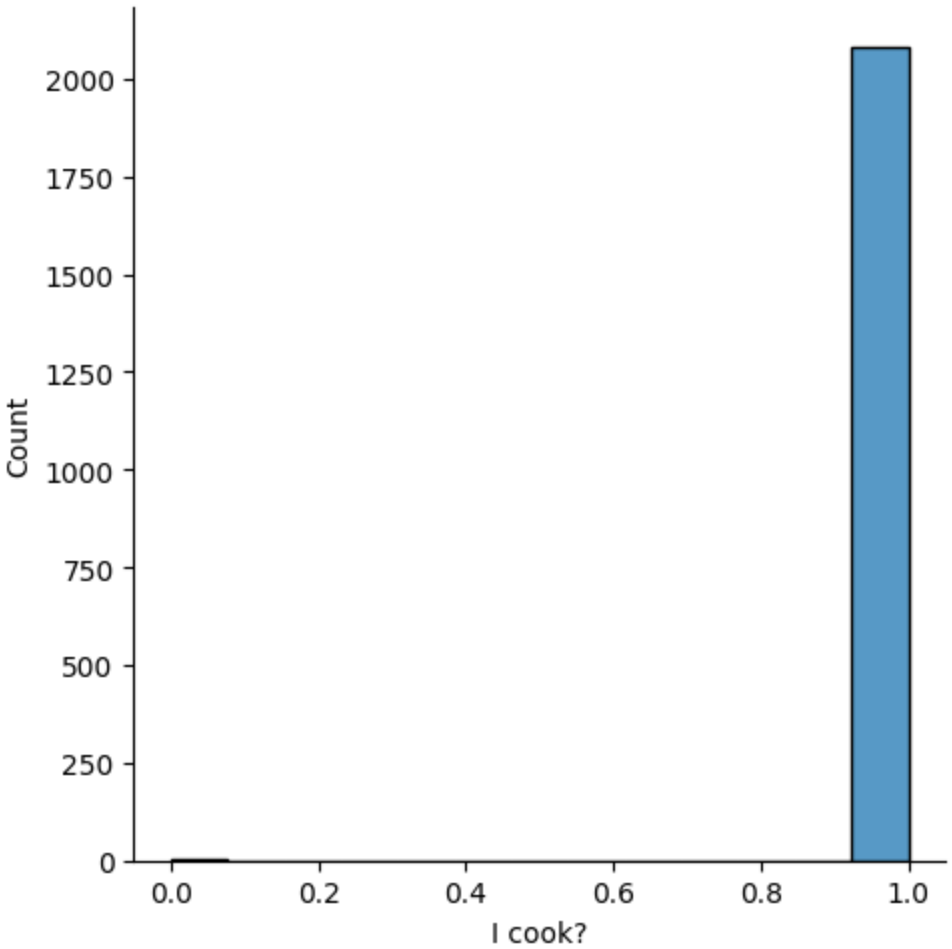


In [61]:
```
tfls.columns
```

Out[61]:
```
Index(['ID', 'ss_tot', 'ss_emot', 'ss_aff', 'ss_lang', 'ss_pos', 'lv_alone',
       'petownr', 'church', 'religion', 'club', 'canegvB', 'strain',
       'optimism', 'nee', 'aee', 'hostil', 'QOL', 'physhealth', 'physfctn',
       'lim_phys', 'pain', 'genhlth', 'mental_health', 'lim_emot', 'energy',
       'emotwell', 'socfctn', 'symptom', 'bloat', 'sweats', 'aches',
       'hotflash', 'tired', 'totlfevt', 'dep_log', 'whi_irs_old',
       'WHI_irs_new', 'main meals', 'snacks', 'meals out', 'cooking time',
       'adults', 'children', 'I cook?', 'cooking help', 'I clean up?',
       'cleanup help', 'shopping freq', 'grocery bill', 'I buy food?',
       'shopping help', 'weight', 'time at weight', 'pre-BC weight',
       'age18 weight', 'max weight', 'time at max wgt', 'health insurance',
       'insurance change?', 'insurance satisfaction', 'years in state',
       'where born', 'age15 residence', 'age35 residence', 'age50 residence',
       'longest residence'],
      dtype='object')
```

In [62]:
```
sns.displot(x=tfls['I cook?'])
```

Out[62]:
```
<seaborn.axisgrid.FacetGrid at 0x7fb01436b9a0>
```



In [63]:
```
corr = tfls.corr().columns
```

In [64]:
```
og = tfls.columns
```

In [65]:
```
for x in og:
  if x not in corr:
    print(x)
```

In [66]:
```
stats.pearsonr(tfls['tired'], tfls['energy'])
```

Out[66]:
```
PearsonRResult(statistic=-0.7069727541123129, pvalue=2.20132507e-315)
```

Cleaning data and visualizing relationships between biomarkers and demographics

(marital status and employment status)

In [67]:
```
dm.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3088 entries, 0 to 3087
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   ID              3088 non-null   int64
 1   age at dx       3088 non-null   int64
 2   age at rand     3088 non-null   int64
 3   Ethnicity       3088 non-null   int64
 4   Education       3088 non-null   int64
 5   Marital Status  3088 non-null   int64
 6   Empl. Status    3088 non-null   int64
 7   Yr Rand         3088 non-null   int64
dtypes: int64(8)
memory usage: 193.1 KB
```

In [68]:
```
print(dm['Marital Status'].unique())
print(dm['Empl. Status'].unique())
```

```
[2 4 5 1 3 6]
[1 0 2]
```

In [69]:
```
# Merging thoughts/feelings and demographics [dm]

tfdm = tf.merge(right=dm, how='inner', on='ID')
tfdm.head()
tfdm.shape
```

Out[69]:
```
(2861, 45)
```

In [70]:
```
tfdm.dropna(inplace=True)
```

In [71]:
```
tfdm.shape
```

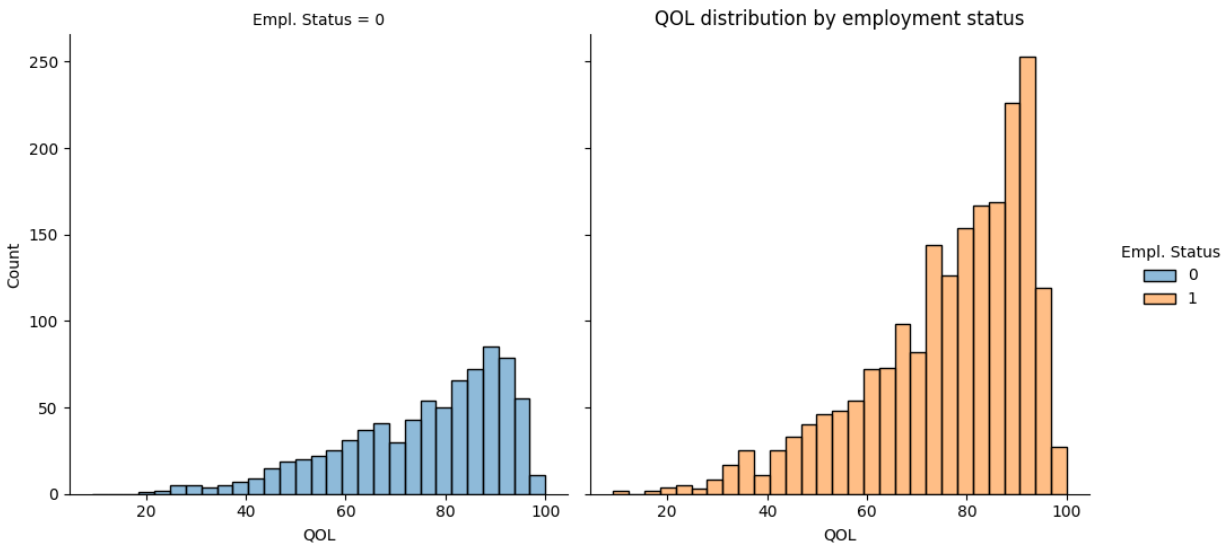
Out[71]:
```
(2861, 45)
```

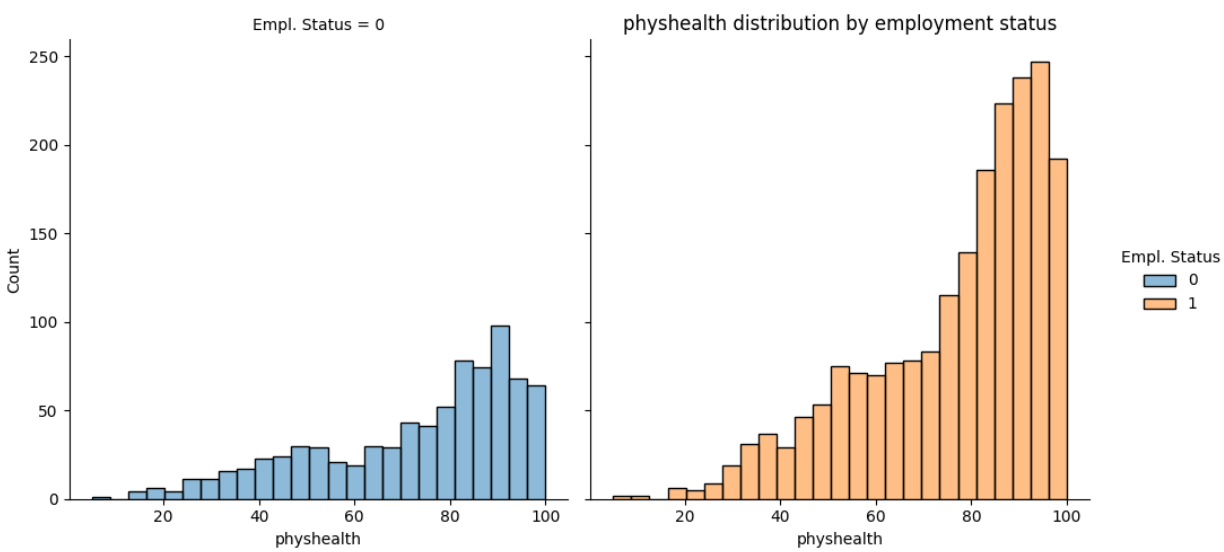
In [72]:
```
# Remove rows with unknown marital status and/or unknown employment status

tfdm = tfdm[(tfdm['Marital Status'] != 6) & (tfdm['Empl. Status'] != 2)]
```

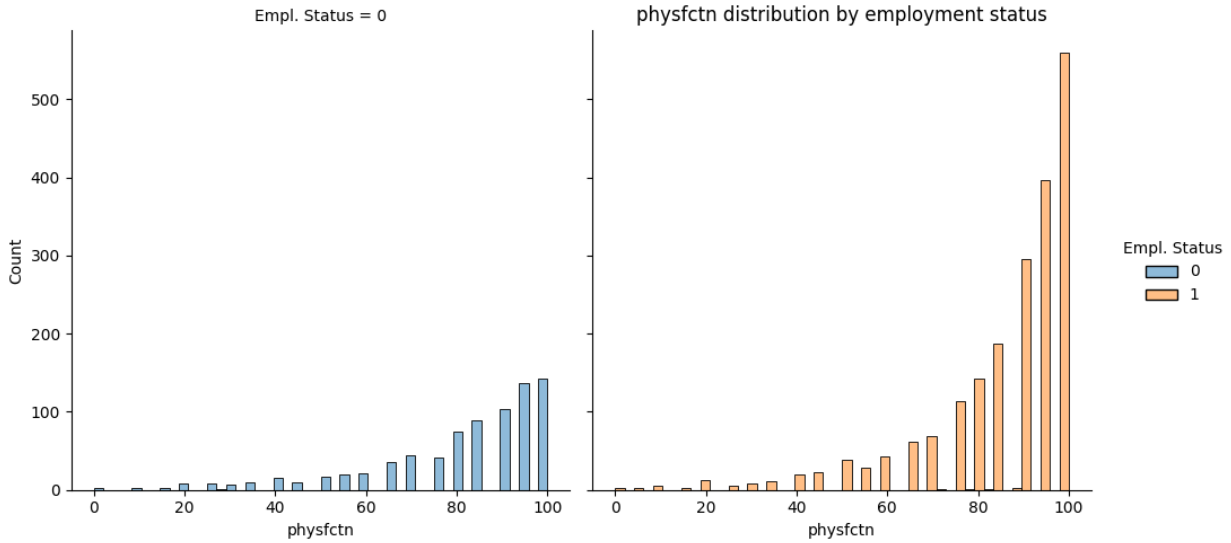
In [73]:
```
tfdm['Empl. Status'].unique()
```

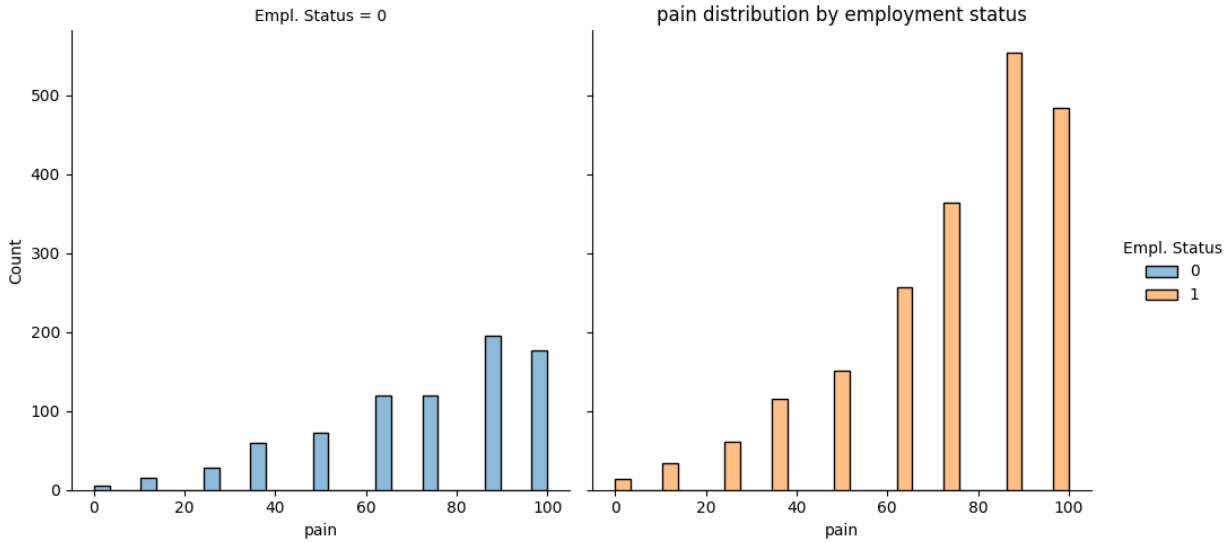
Out[73]:
```
array([1, 0])
```

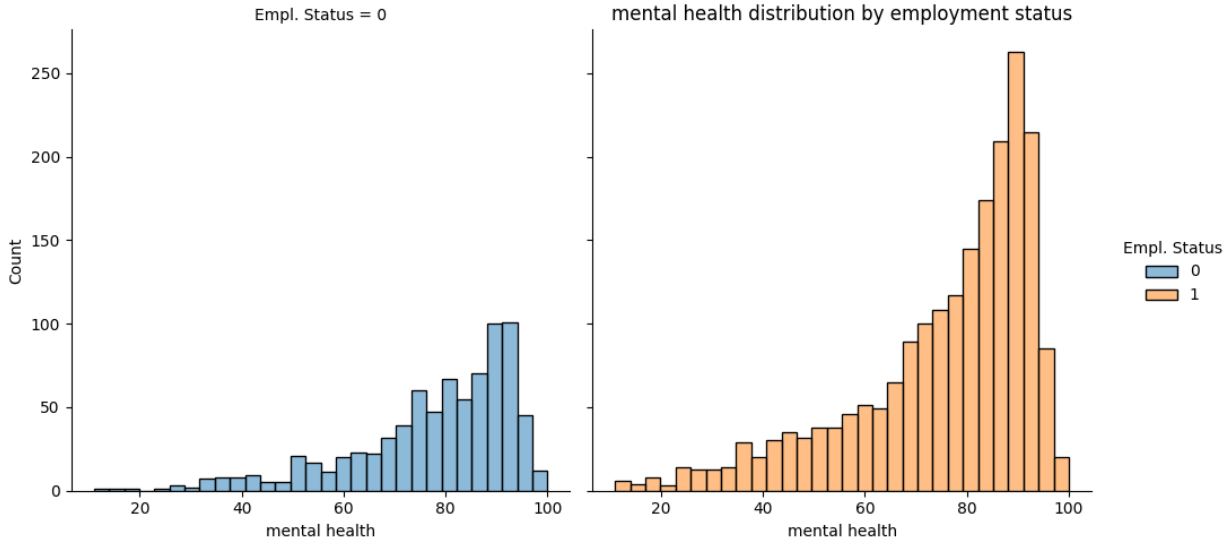
In [74]:
```
# Visualize thoughts/feelings subsetted by employment status

columns = ['QOL', 'physhealth', 'physfctn', 'pain', 'mental health', 'emotwell']

for col in columns:
  sns.displot(data=tfdm, x=col, hue='Empl. Status', col='Empl. Status',)
  plt.title(f"{col} distribution by employment status")
```

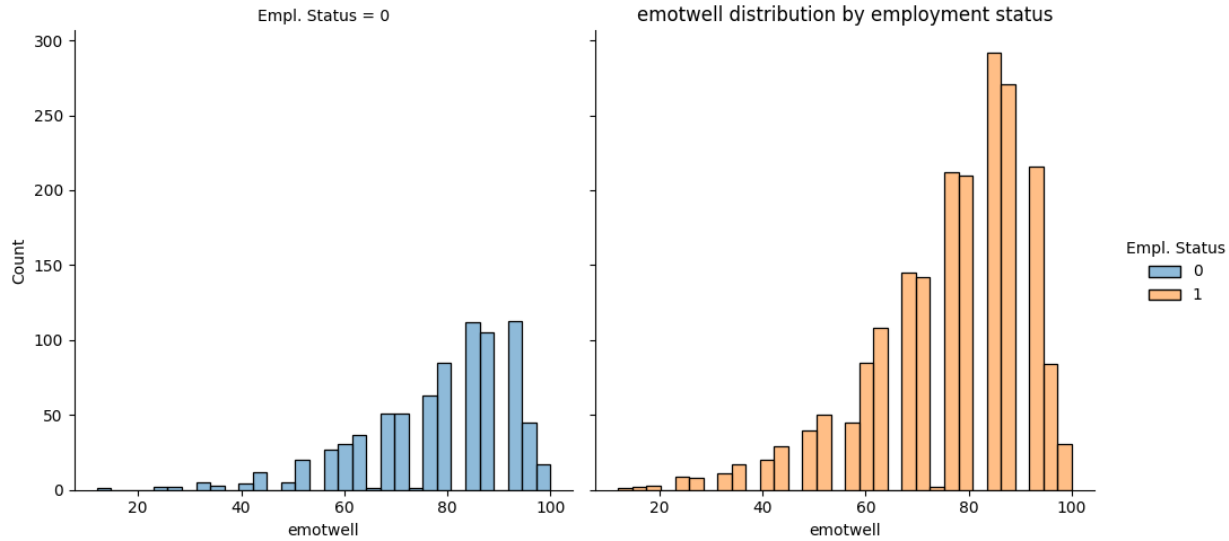




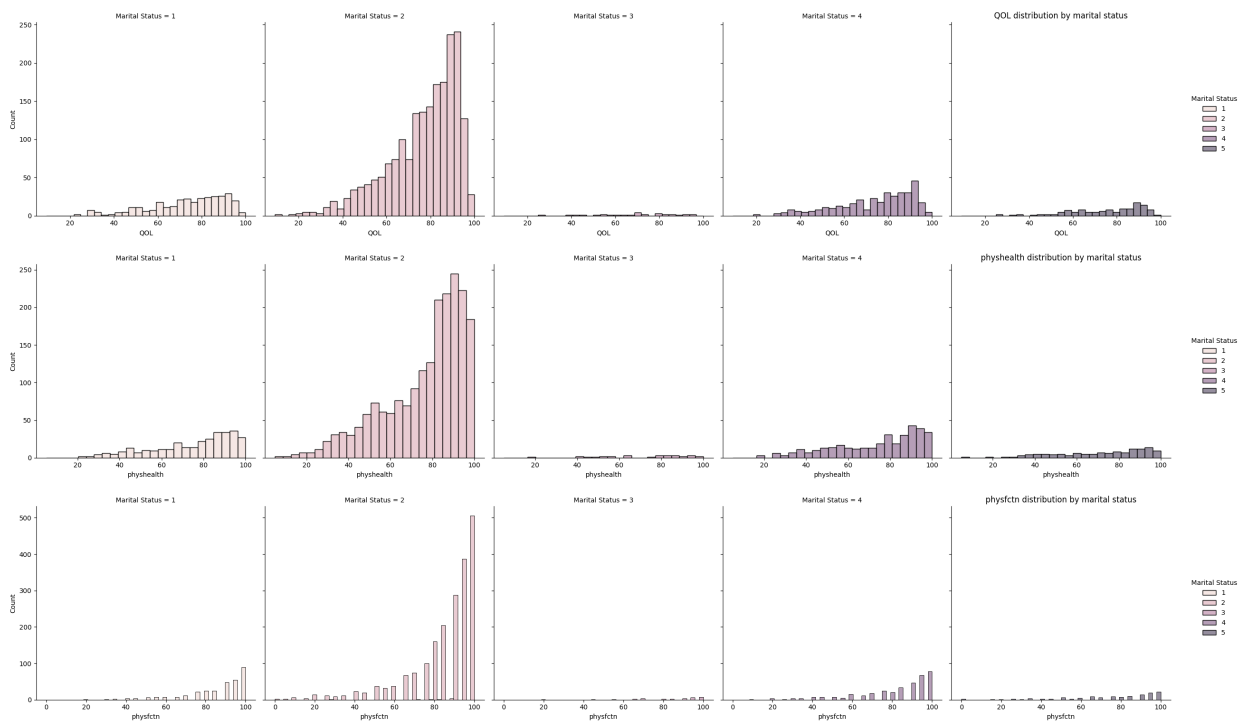

Empl. Status = 0
physfctn distribution by employment status
Count
physfctn
Empl. Status
0
1

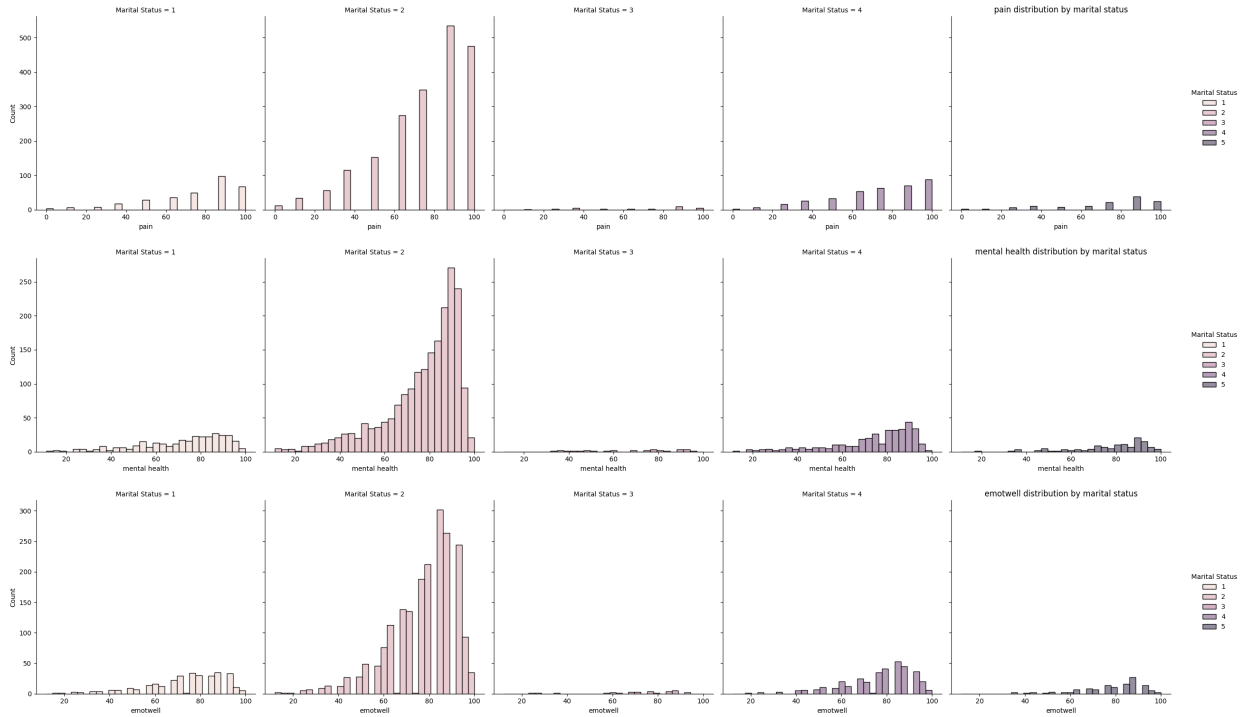

Empl. Status = 0
pain distribution by employment status
Count
pain
Empl. Status
0
1


Empl. Status = 0
mental health distribution by employment status
Count
mental health
Empl. Status
0
1



In [75]:
```
# Above, subsetted by marital status

# Marital Status
#                1 Single
#                2 Married
#                3 Separated
#                4 Divorced
#                5 Widowed

for col in columns:
  sns.displot(data=tfdm, x=col, hue='Marital Status', col='Marital Status',);
  plt.title(f"{col} distribution by marital status")
```





In [76]:
```
tfdm['ss_pos'].unique()
```

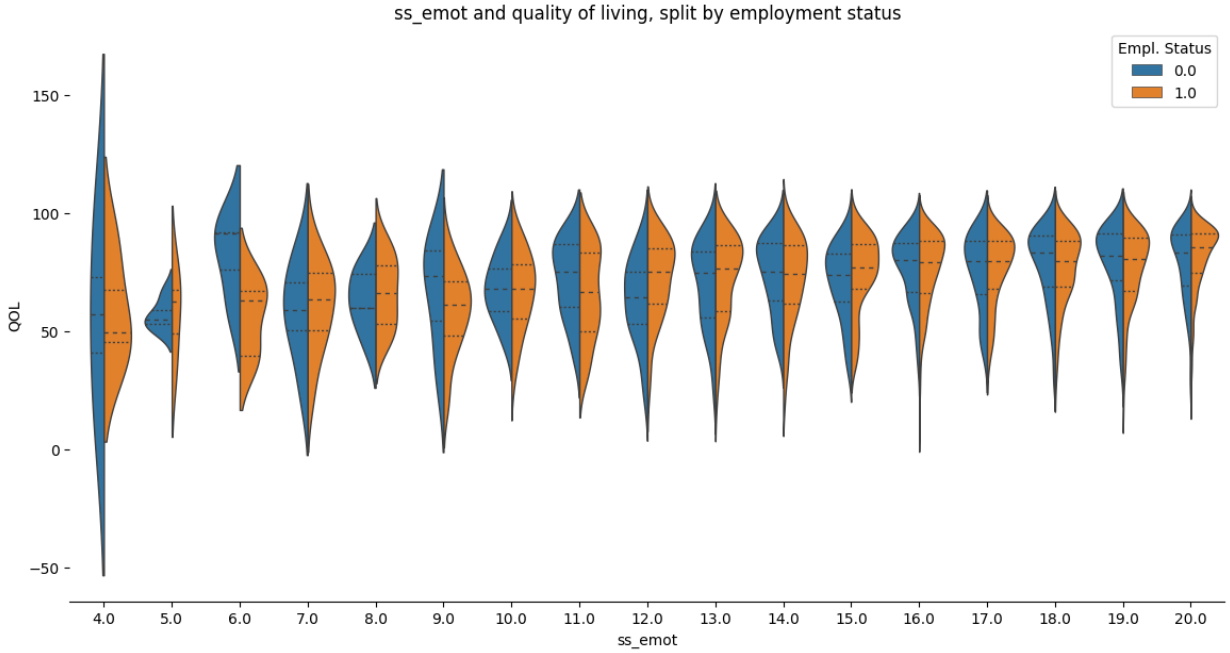
Out[76]:
```
array([10.,  9.,  8.,  7.,  6.,  4.,  5.,  2.,  3.])
```

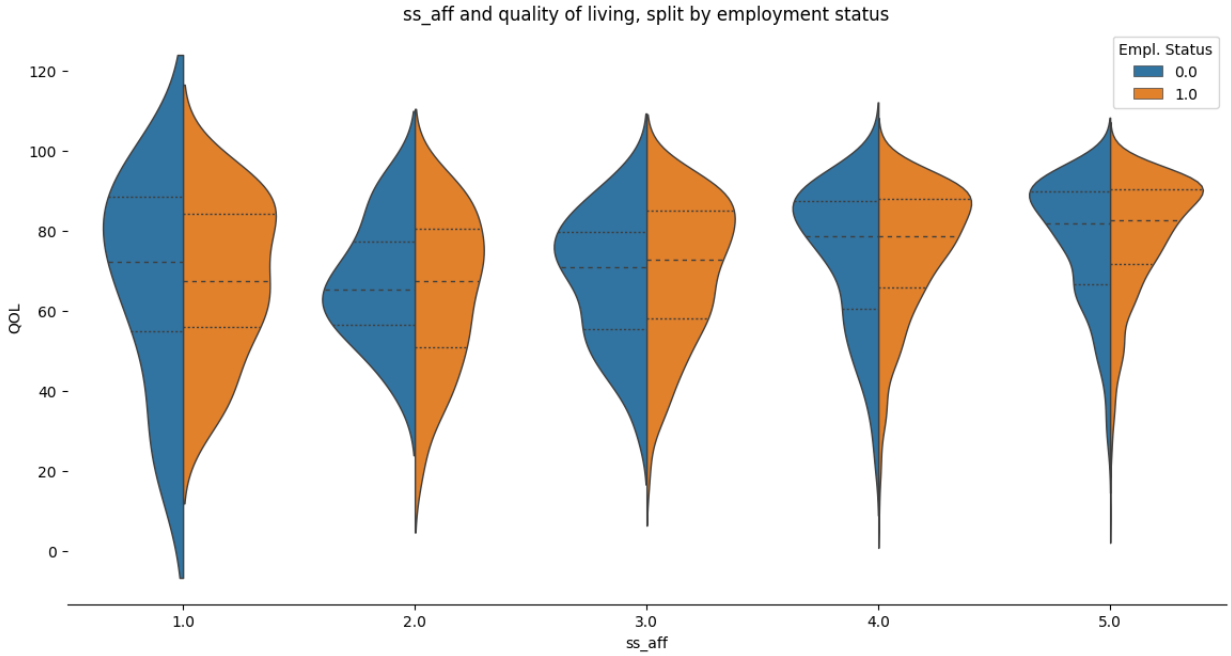
In [77]:
```
tfdm = tfdm.astype(float)
```

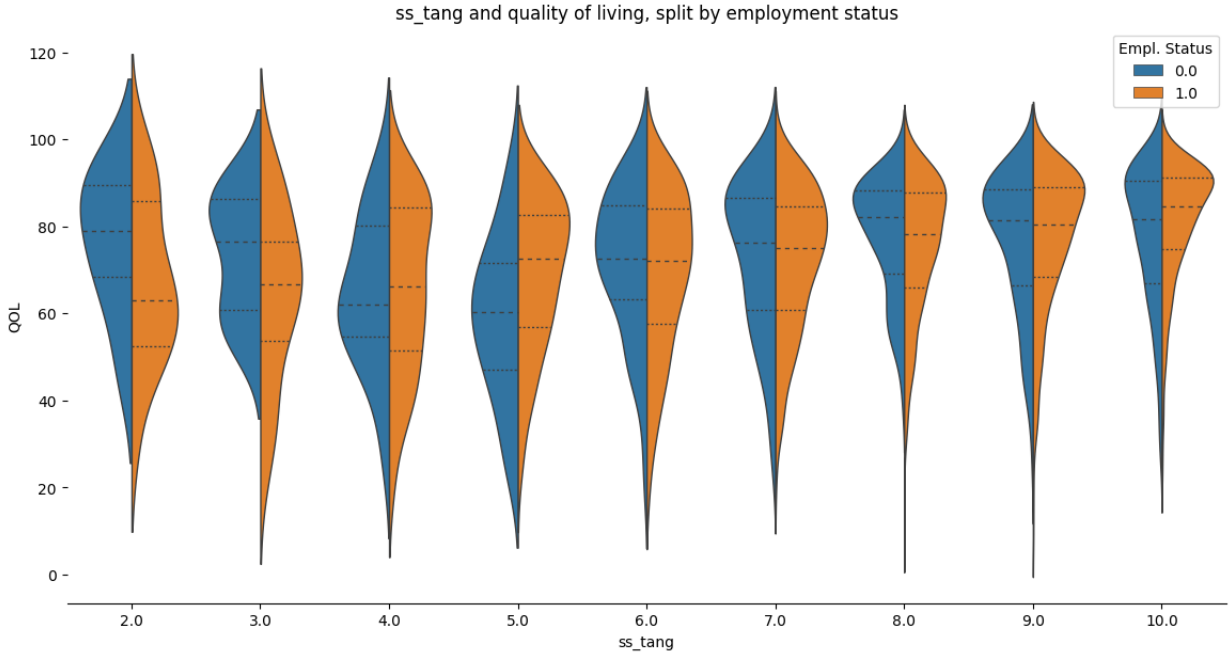
In [78]:
```
type(list(tfdm['ID'])[0])
```

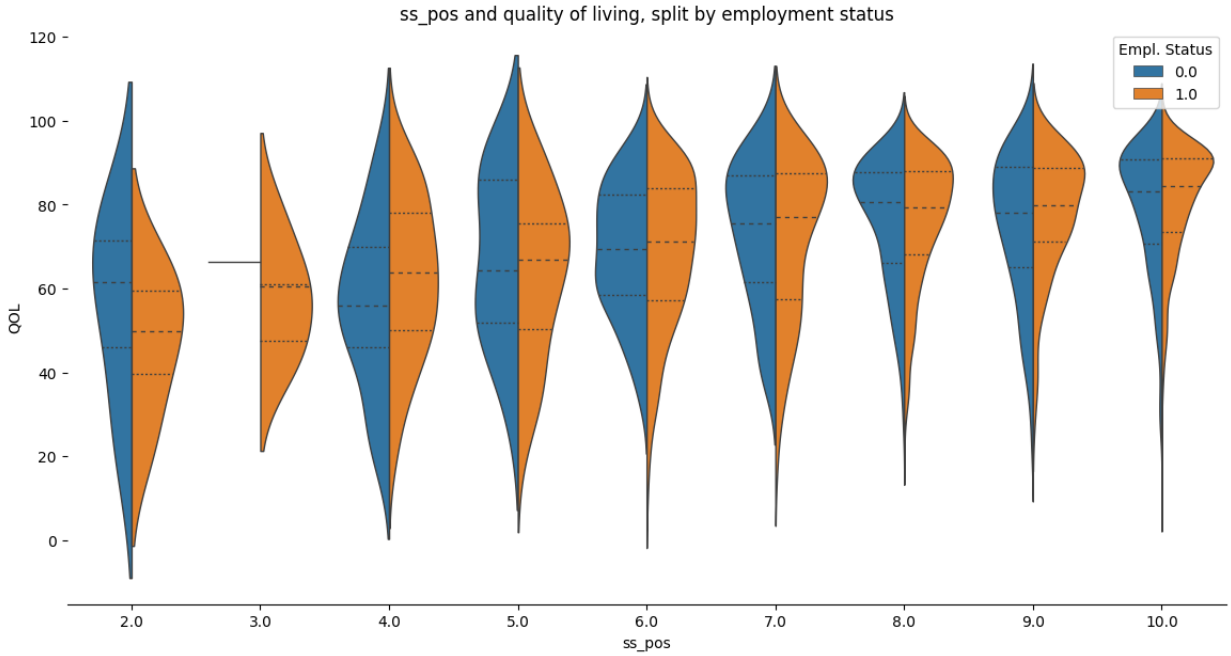
Out[78]:
```
float
```

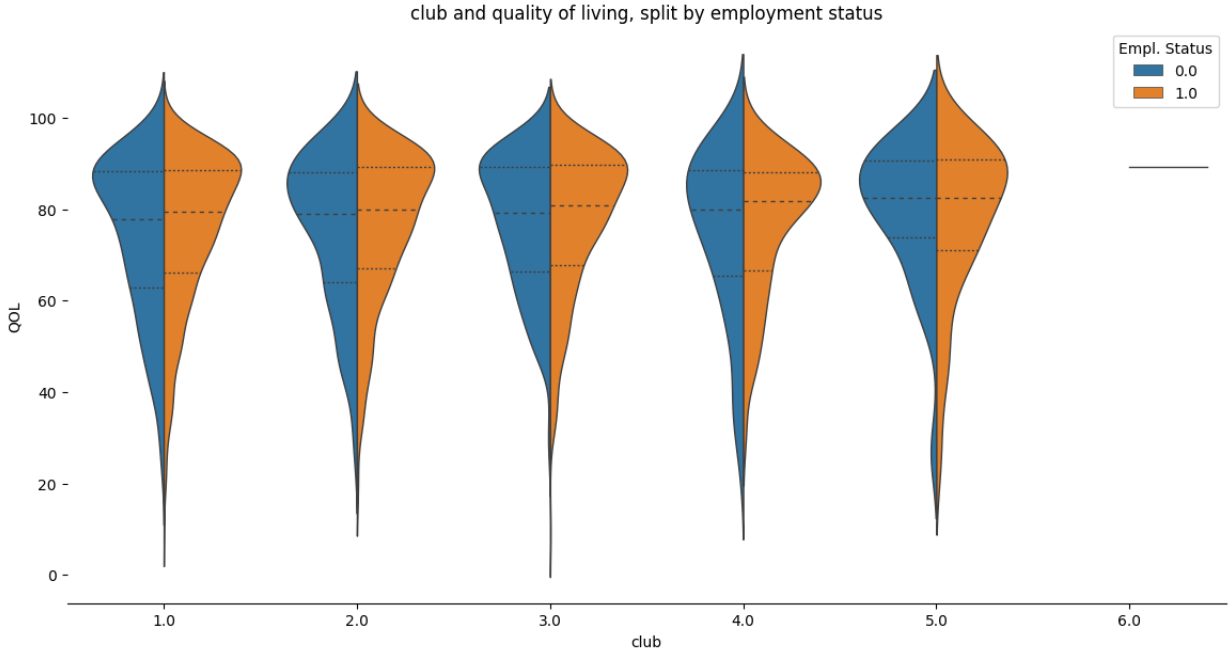
In [79]:
```
# Plot violinplot between living alone and overall living quality, split by employment

columns = ['ss_emot', 'ss_aff', 'ss_tang', 'ss_pos', 'club', 'caregvB',
           'strain', 'optimism', 'aee', 'hostil', 'pain', 'bloat', 'sweats',
           'aches', 'tired', 'WHI_irs_new']

for col in columns:
  plt.figure(figsize=(14, 7))
  sns.violinplot(data=tfdm, x=col, y='QOL', hue='Empl. Status', split=True,
                 inner='quart', linewidth=1);
  sns.despine(left=True)
  plt.title(f"{col} and quality of living, split by employment status")
```

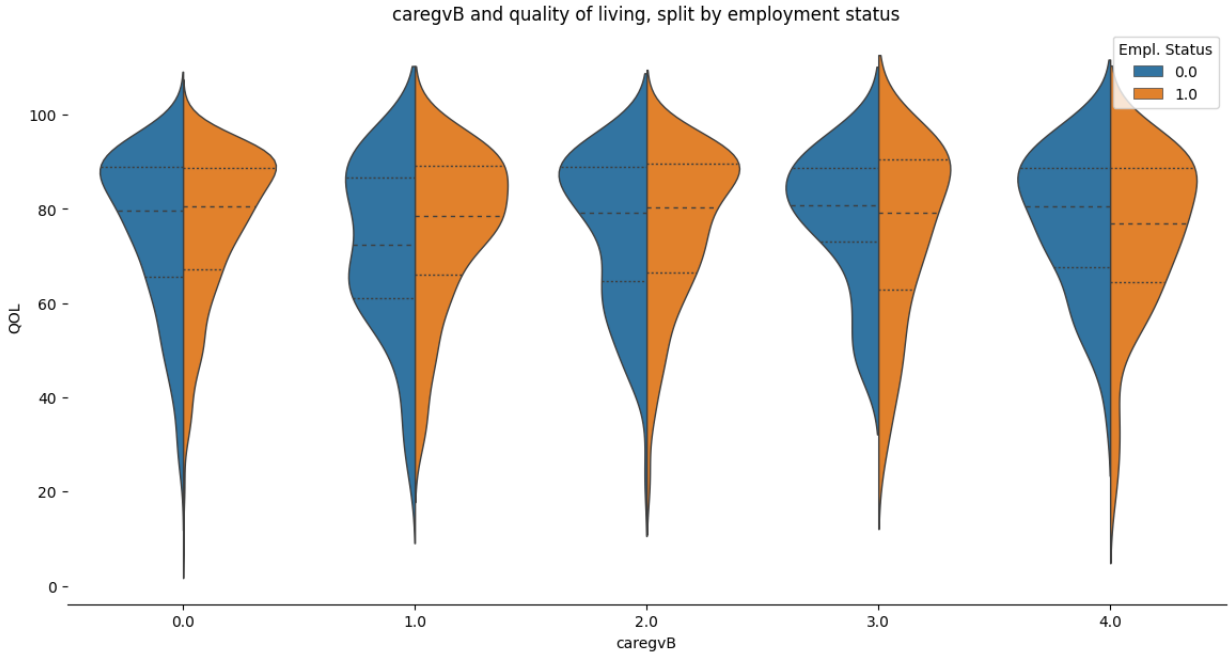

ss_emot and quality of living, split by employment status
Empl. Status
0.0
1.0
QOL
150
100
50
0
−50
4.0
5.0
6.0
7.0
8.0
9.0
10.0
11.0
12.0
13.0
14.0
15.0
16.0
17.0
18.0
19.0
20.0
ss_emot

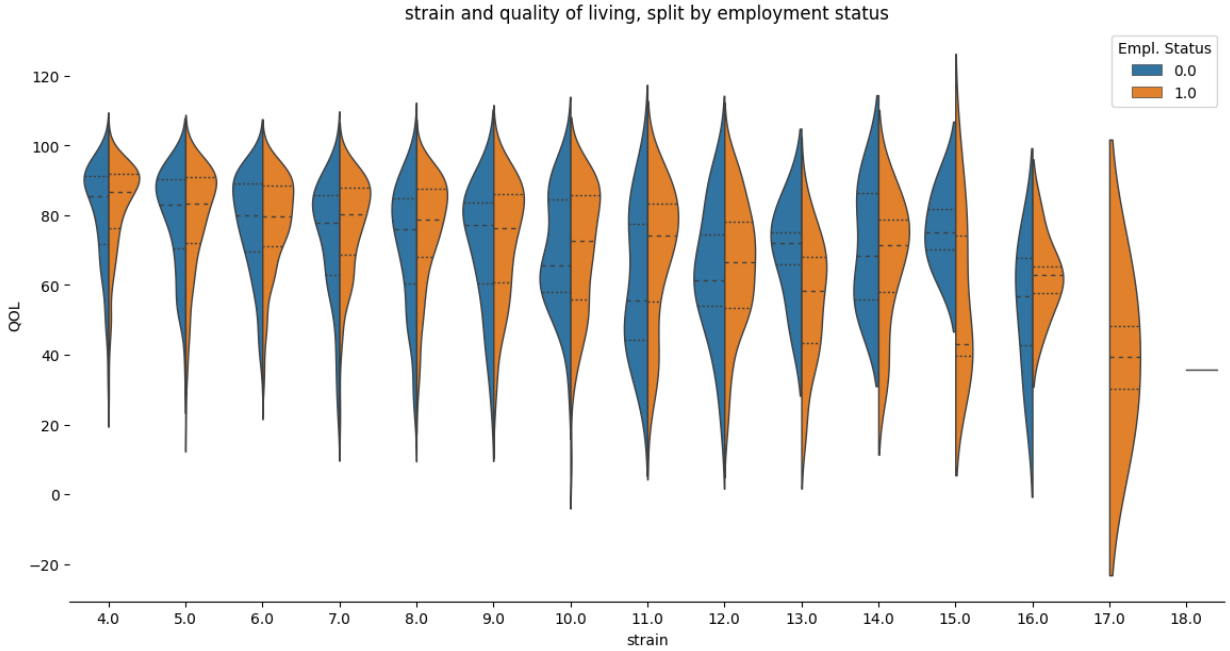

ss_aff and quality of living, split by employment status
Empl. Status
0.0
1.0
QOL
120
100
80
60
40
20
0
1.0
2.0
3.0
4.0
5.0
ss_aff

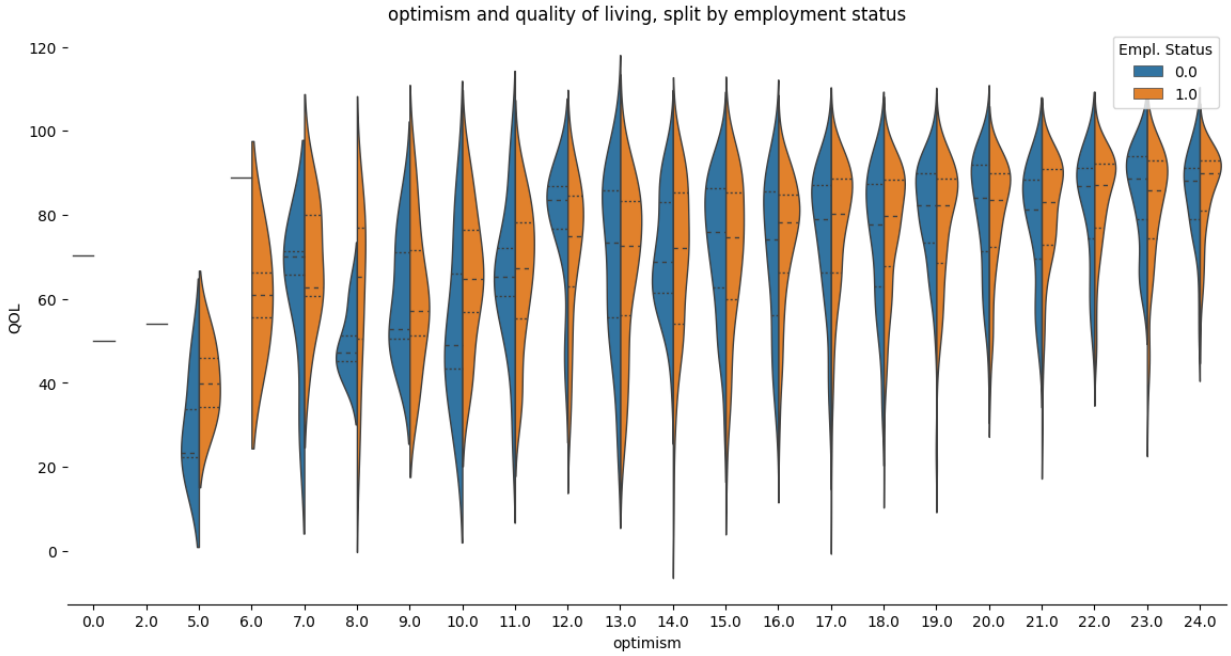

ss_tang and quality of living, split by employment status
Empl. Status
0.0
1.0
QOL
ss_tang
0
20
40
60
80
100
120
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
10.0

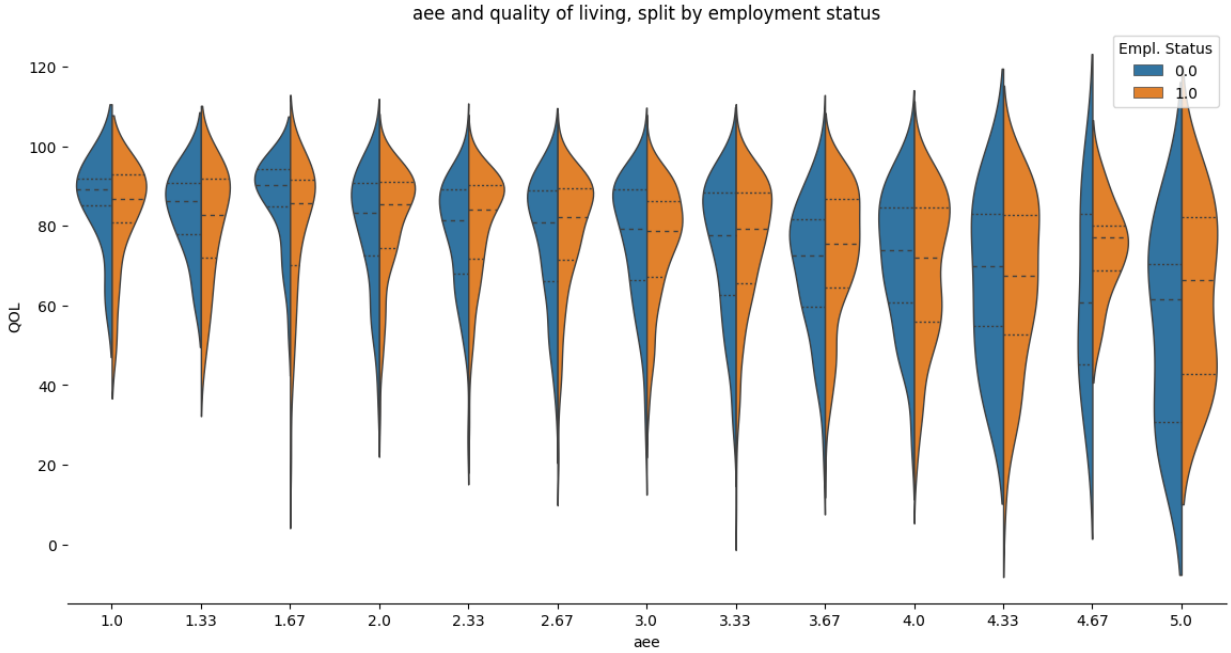

ss_pos and quality of living, split by employment status
Empl. Status
0.0
1.0
QOL
ss_pos
0
20
40
60
80
100
120
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
10.0

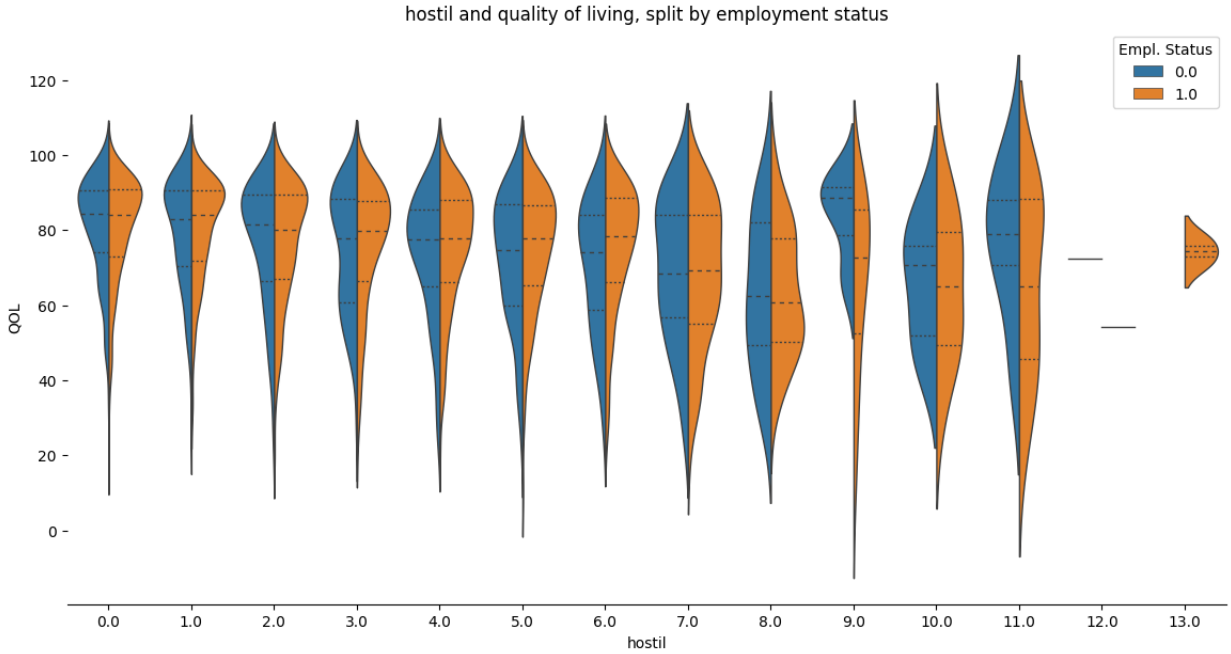

club and quality of living, split by employment status
Empl. Status
0.0
1.0
QOL
0
20
40
60
80
100
1.0
2.0
3.0
4.0
5.0
6.0
club

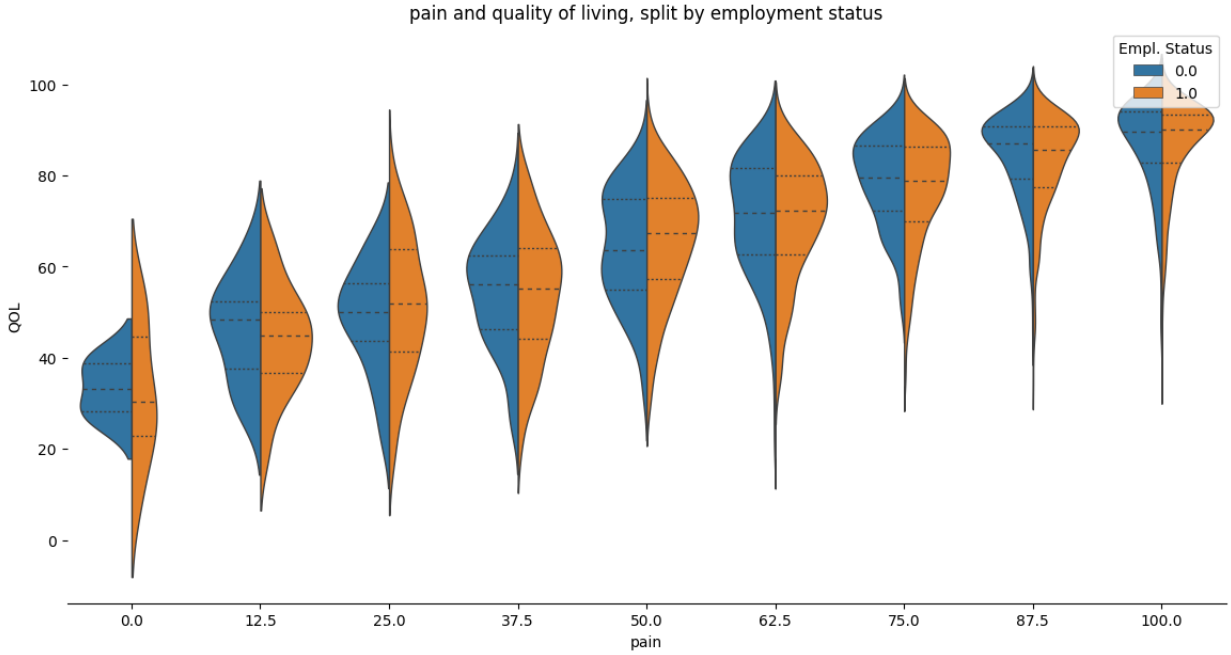

caregvB and quality of living, split by employment status
Empl. Status
0.0
1.0
QOL
0
20
40
60
80
100
0.0
1.0
2.0
3.0
4.0
caregvB

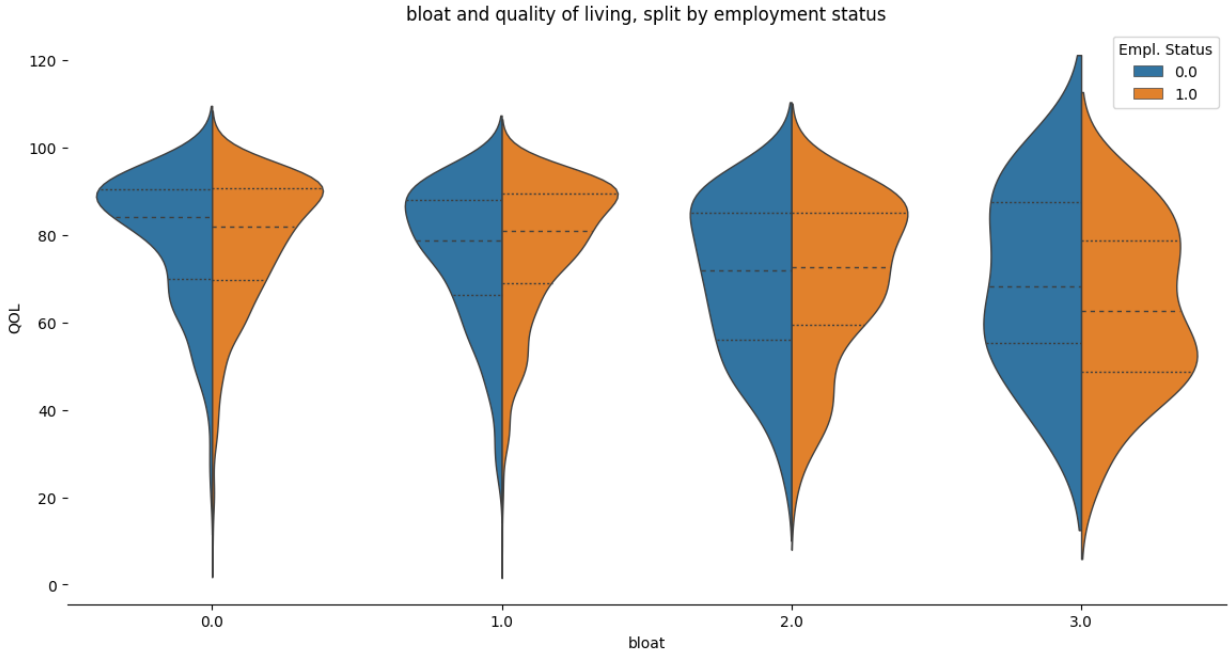

strain and quality of living, split by employment status
Empl. Status
0.0
1.0
QOL
strain
120
100
80
60
40
20
0
−20
4.0
5.0
6.0
7.0
8.0
9.0
10.0
11.0
12.0
13.0
14.0
15.0
16.0
17.0
18.0

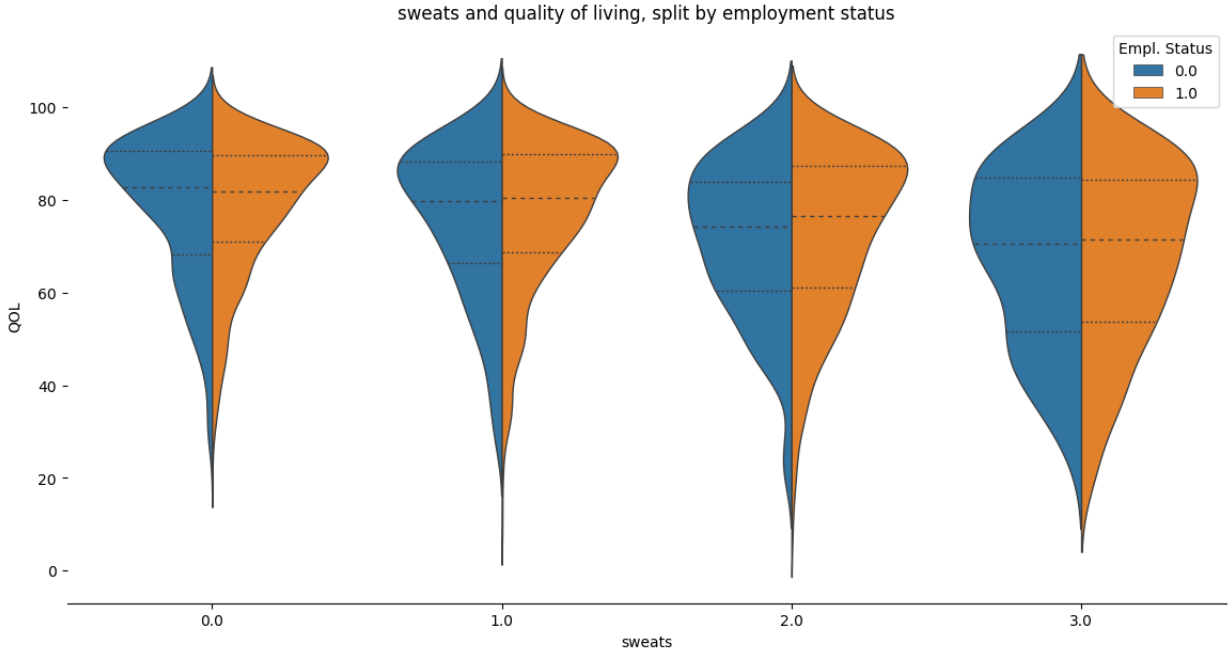

optimism and quality of living, split by employment status
Empl. Status
0.0
1.0
QOL
optimism
120
100
80
60
40
20
0
0.0
2.0
5.0
6.0
7.0
8.0
9.0
10.0
11.0
12.0
13.0
14.0
15.0
16.0
17.0
18.0
19.0
20.0
21.0
22.0
23.0
24.0

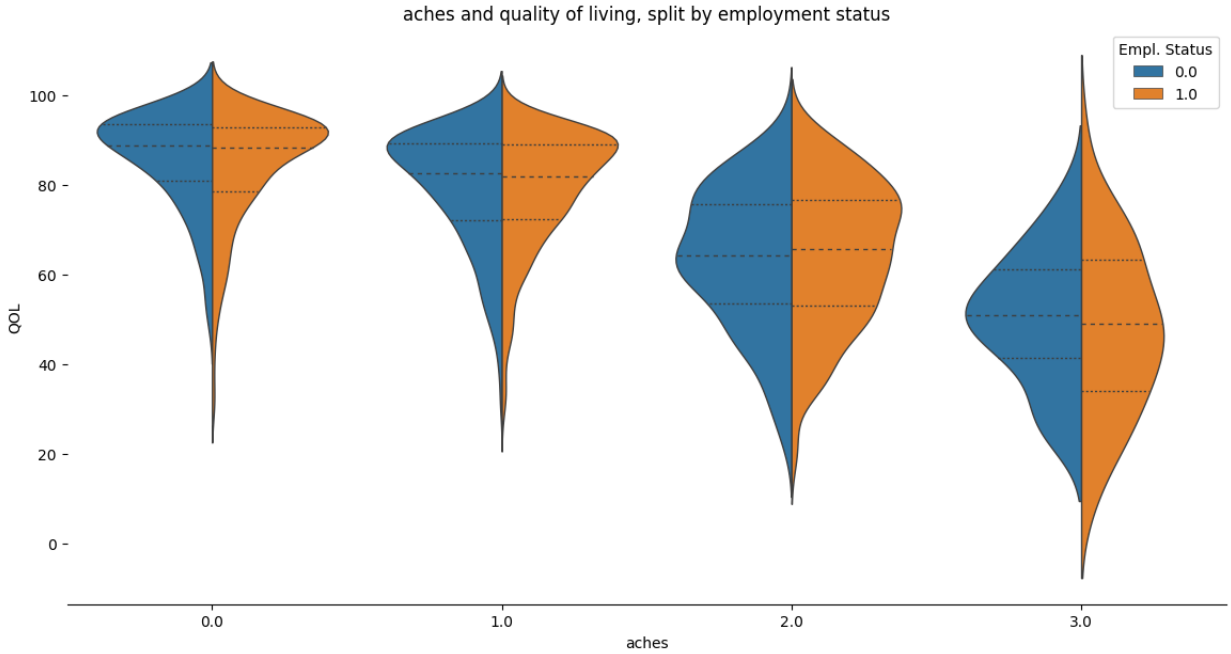

aee and quality of living, split by employment status
Empl. Status
0.0
1.0
QOL
120
100
80
60
40
20
0
1.0
1.33
1.67
2.0
2.33
2.67
3.0
3.33
3.67
4.0
4.33
4.67
5.0
aee

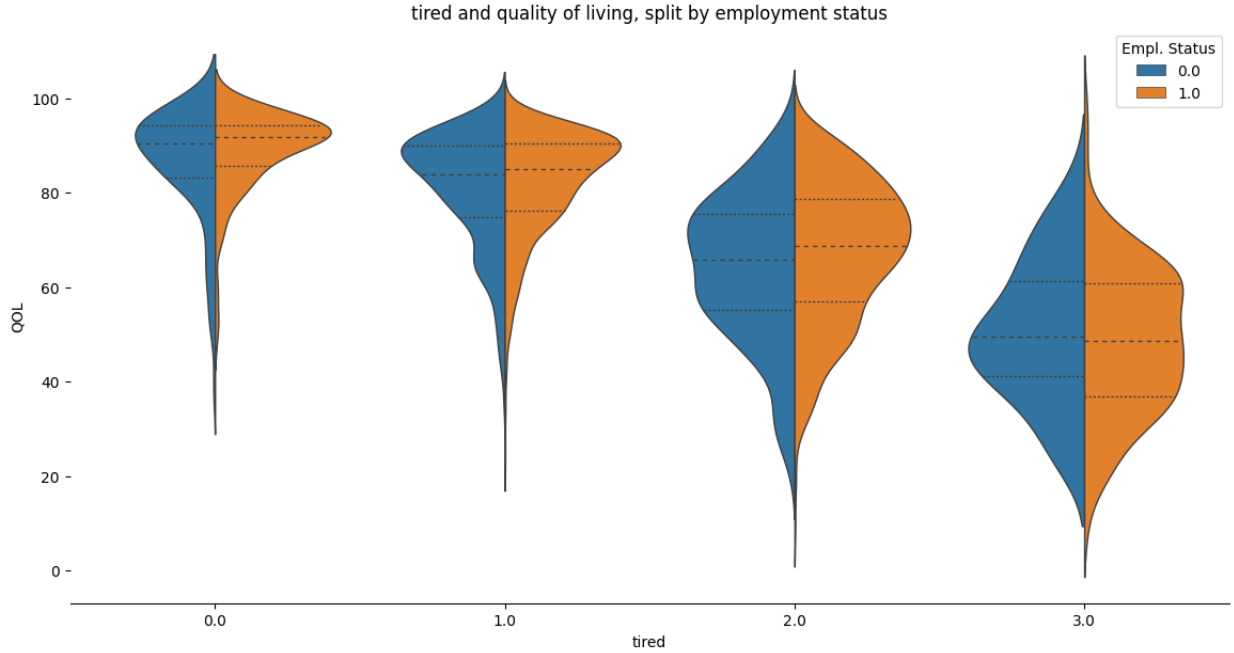

hostil and quality of living, split by employment status
Empl. Status
0.0
1.0
QOL
120
100
80
60
40
20
0
0.0
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
10.0
11.0
12.0
13.0
hostil

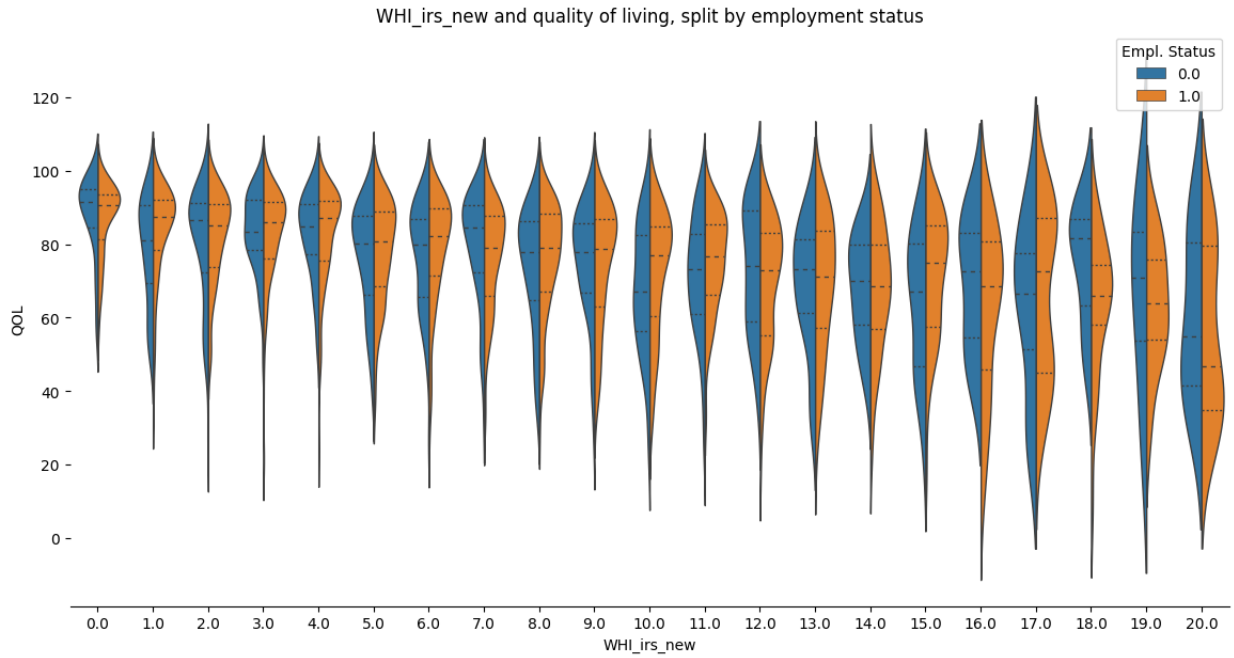

pain and quality of living, split by employment status
Empl. Status
0.0
1.0
QOL
100
80
60
40
20
0
0.0
12.5
25.0
37.5
50.0
62.5
75.0
87.5
100.0
pain


bloat and quality of living, split by employment status
Empl. Status
0.0
1.0
QOL
120
100
80
60
40
20
0
0.0
1.0
2.0
3.0
bloat


sweats and quality of living, split by employment status
Empl. Status
0.0
1.0
QOL
100
80
60
40
20
0
0.0
1.0
2.0
3.0
sweats


aches and quality of living, split by employment status
Empl. Status
0.0
1.0
QOL
100
80
60
40
20
0
0.0
1.0
2.0
3.0
aches



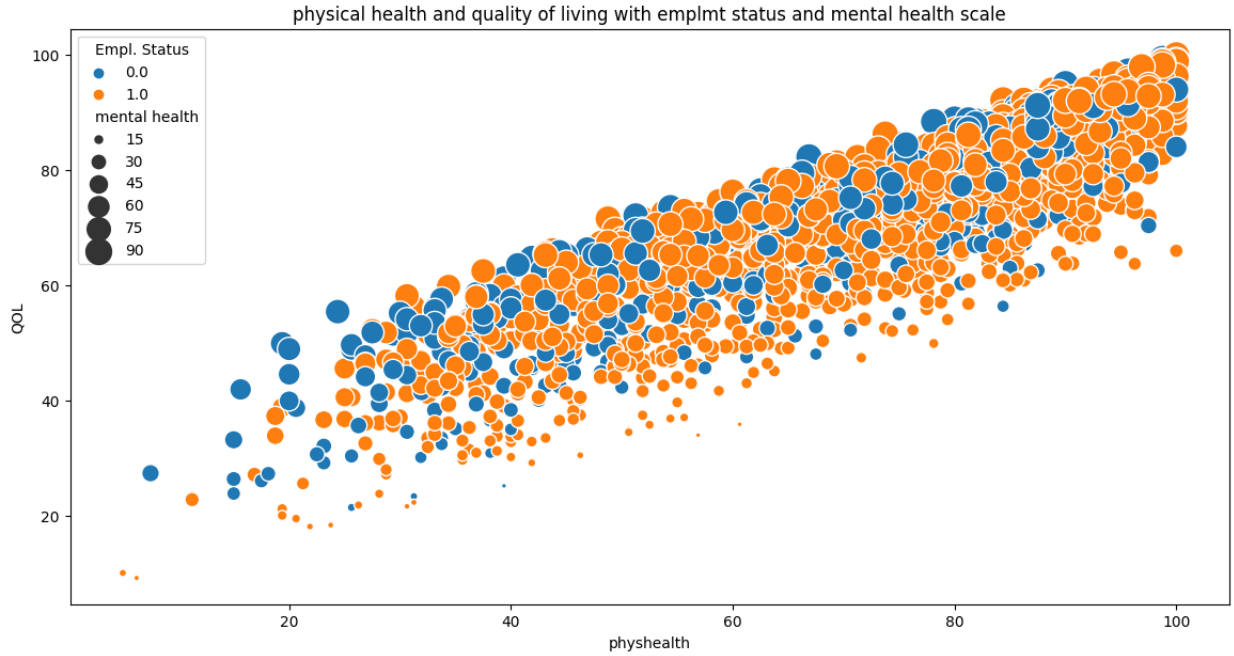


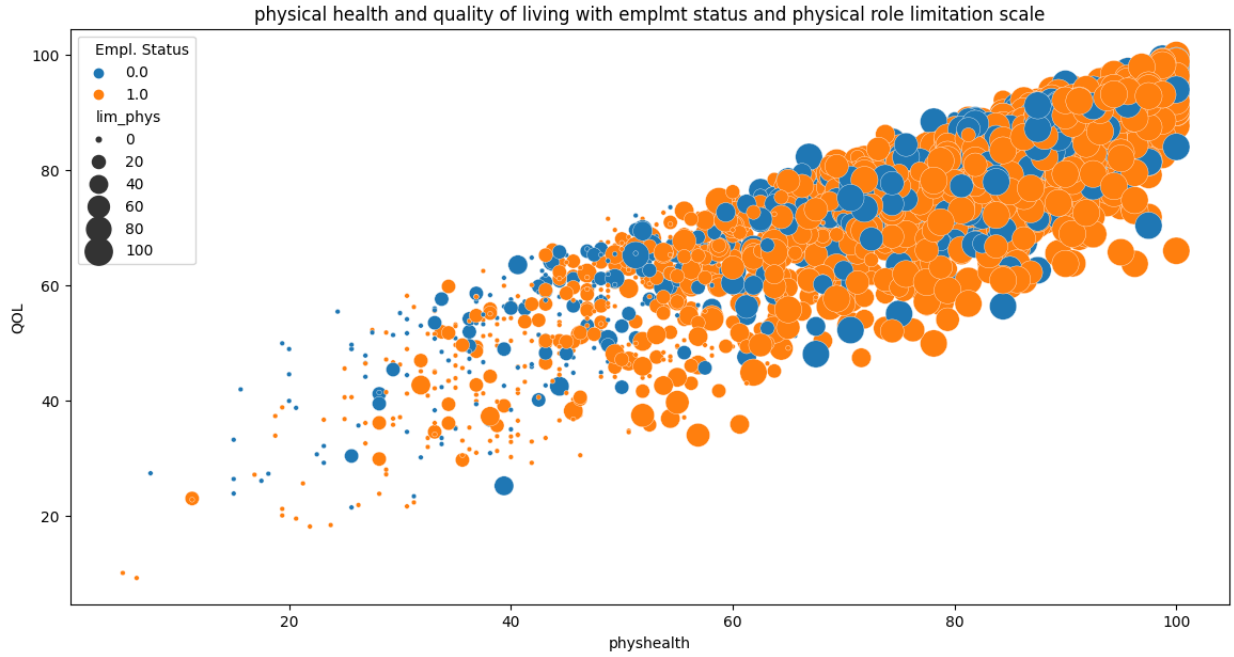
In [80]:
```
# Plot physical health and quality of living with emplmt status and mental health scal

plt.figure(figsize=(14, 7))
sns.scatterplot(data=tfdm, x='physhealth', y='QOL', hue='Empl. Status', size='mental h
            sizes=(11,333))
plt.title("physical health and quality of living with emplmt status and mental health


# Plot physical health and quality of living with emplmt status and physical role limi

plt.figure(figsize=(14, 7))
sns.scatterplot(data=tfdm, x='physhealth', y='QOL', hue='Empl. Status', size='lim_phys
            sizes=(11,333))
plt.title("physical health and quality of living with emplmt status and physical role


# Plot physical health and quality of living with emplmt status and emotional role lim
```

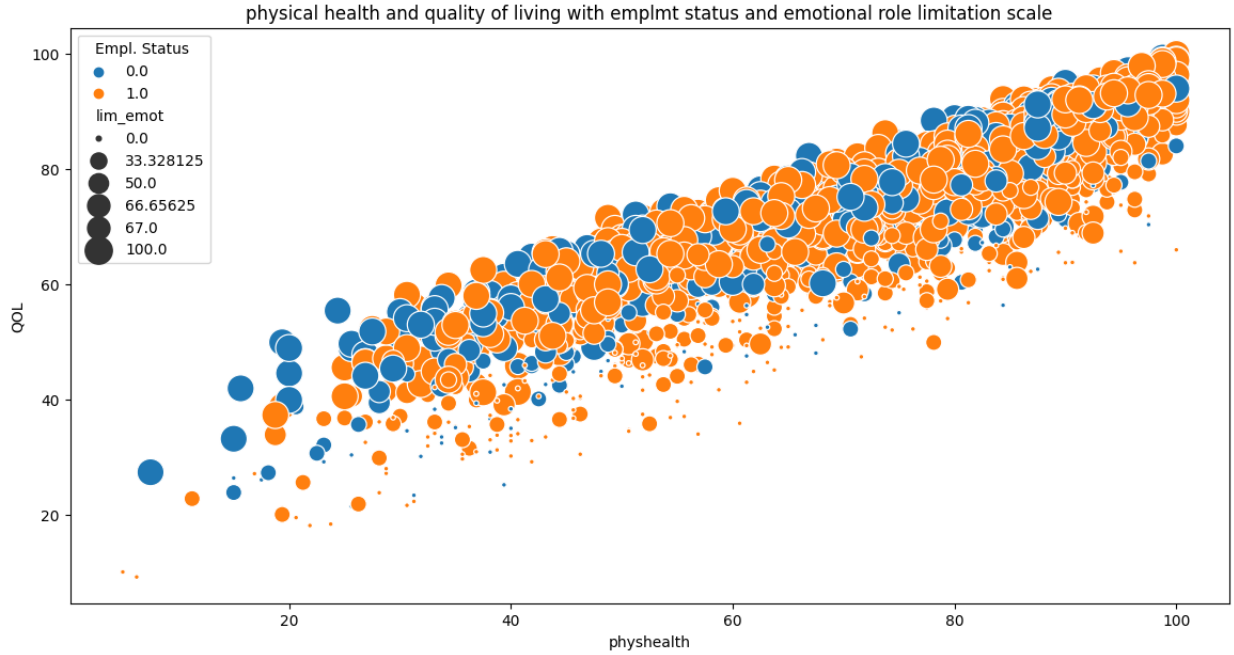
```
plt.figure(figsize=(14, 7))
sns.scatterplot(data=tfdm, x='physhealth', y='QOL', hue='Empl. Status', size='lim_emot
            sizes=(11,333))
plt.title("physical health and quality of living with emplmt status and emotional role
```

Out[80]:

```
Text(0.5, 1.0, 'physical health and quality of living with emplmt status and emotiona
l role limitation scale')
```







In [81]:
```
tfdm['pain'].unique()
```

Out[81]:
```
array([100. ,  87.5,  75. ,  62.5,  12.5,  50. ,   0. ,  37.5,  25. ])
```

In [82]:
```
tfdm.columns
```

Out[82]:
```
Index(['ID', 'ss_tot', 'ss_emot', 'ss_aff', 'ss_tang', 'ss_pos', 'lv_alone',
       'petownr', 'church', 'religion', 'club', 'caregvB', 'strain',
       'optimism', 'nee', 'aee', 'hostil', 'QOL', 'physhealth', 'physfctn',
       'lim_phys', 'pain', 'genhlth', 'mental health', 'lim_emot', 'energy',
       'emotwell', 'socfctn', 'symptom', 'bloat', 'sweats', 'aches',
       'hotflash', 'tired', 'totlfevt', 'dep_log', 'whi_irs_old',
       'WHI_irs_new', 'age at dx', 'age at rand', 'Ethnicity', 'Education',
       'Marital Status', 'Empl. Status', 'Yr Rand'],
      dtype='object')
```

Cleaning data and visualizing relationships between biomarkers and family history

In [83]:
```
fh.rename(columns={'id':'ID'}, inplace=True)
fh.columns
```

Out[83]:
```
Index(['ID', 'adopted?', '# first deg relatives',
       '# first deg relatives with breast cancer',
       '# first deg relatives with any cancer',
       '# second deg relatives (granparents, aunts, uncles)',
       '# second deg relatives with breast cancer',
       '# second deg relatives with any cancer', 'mom had BC?',
       'mom had ovarian cancer?', '# sisters', '# sisters with breast cancer',
       '# sisters with ovarian cancer',
       '# first deg relatives with colorectal cancer',
       '# second deg relatives with colorectal cancer'],
      dtype='object')
```

In [84]:
```
fhhs = hs.merge(right=fh, how='inner', on='ID')
fhhs.shape
```

Out[84]: (3088, 28)

In [85]:
```
fhhs.dropna(inplace=True)
fhhs.shape
```

Out[85]: (2241, 28)

In [86]:
```
fhhs.head()
```

Out[86]:

	ID	hospital_	Exam_Freq	Proph_Mast	Blood_Sugar	Cardiovascular	Digestive	Arthritis	Osteo
0	13245	0.0	1.0	0.0	0.0	0.0	0.0	0.0	
1	13244	1.0	1.0	0.0	0.0	0.0	0.0	0.0	
2	13243	0.0	1.0	0.0	0.0	0.0	0.0	0.0	
3	13242	0.0	1.0	0.0	0.0	0.0	0.0	0.0	
4	13241	0.0	3.0	0.0	0.0	0.0	0.0	0.0	

5 rows × 28 columns

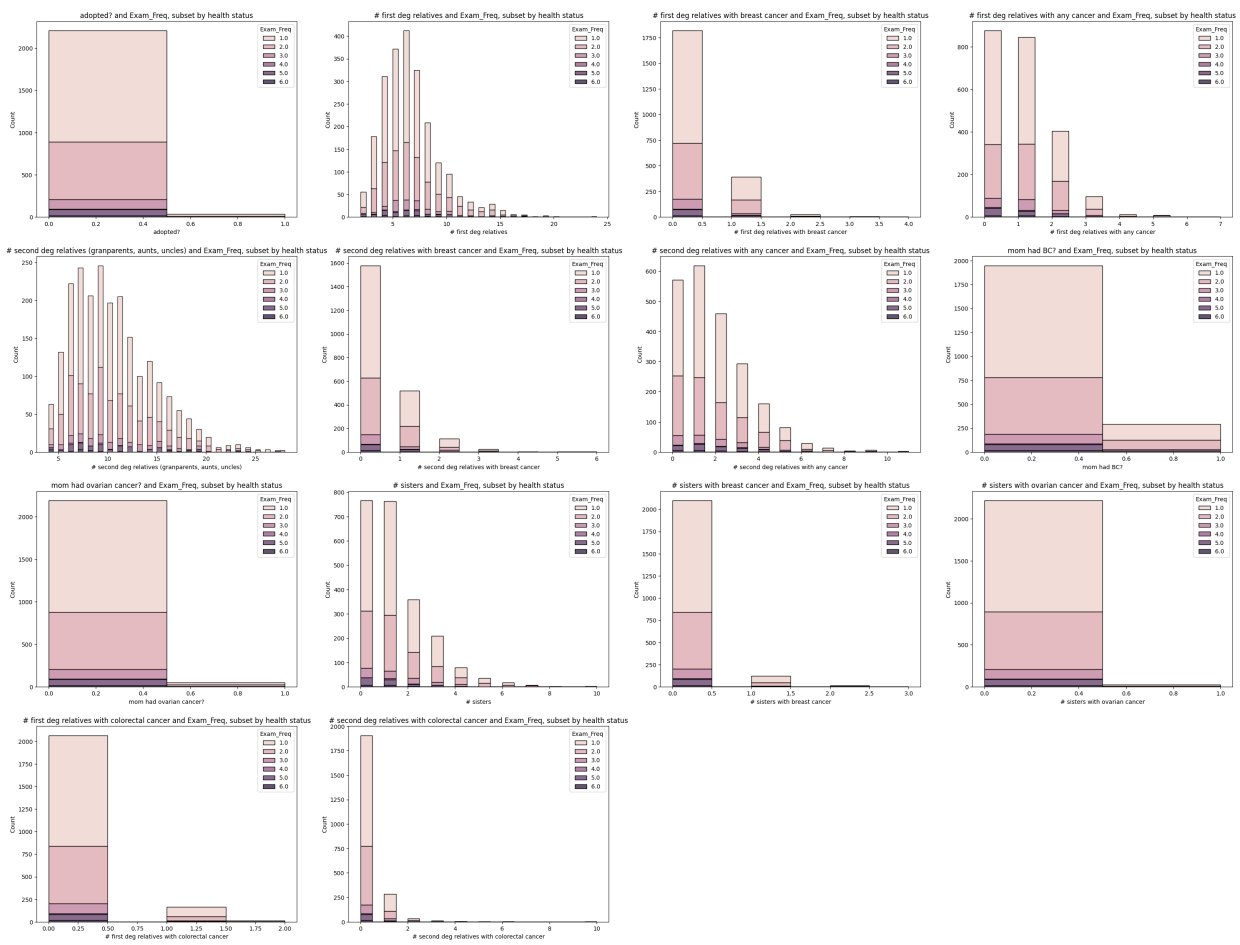
In [87]:
```
for col in fhhs.columns:
  if col not in hs.columns and col != 'ID':
    print(col)
    print(fhhs[col].unique())
    print()
```

```
adopted?
[0. 1.]

# first deg relatives
[ 8.  5.  4.  7. 11.  6.  9. 13.  3. 10. 15. 14. 12.  2. 17. 16. 20. 18.
 19. 24.]

# first deg relatives with breast cancer
[0. 1. 2. 4. 3.]

# first deg relatives with any cancer
[1. 2. 0. 3. 6. 4. 5. 7.]

# second deg relatives (granparents, aunts, uncles)
[ 6.  8. 12. 13. 10. 18. 11.  9.  5. 16.  7.  4. 14. 17. 15. 20. 28. 22.
 19. 23. 25. 26. 24. 27. 21.]

# second deg relatives with breast cancer
[1. 0. 3. 2. 6. 5. 4.]

# second deg relatives with any cancer
[ 3.  1.  2.  4.  0.  5. 11.  6.  9.  7.  8.]

mom had BC?
[0. 1.]

mom had ovarian cancer?
[0. 1.]

# sisters
[ 1.  0.  3.  4.  2.  5.  7.  6. 10.  9.  8.]

# sisters with breast cancer
[0. 1. 3. 2.]

# sisters with ovarian cancer
[0. 1.]

# first deg relatives with colorectal cancer
[0. 1. 2.]

# second deg relatives with colorectal cancer
[ 1.  0.  3.  2.  6.  4.  5. 10.]
```

In [88]:
```
fhhs.columns
```

Out[88]:
```
Index(['ID', 'hospital_', 'Exam_Freq', 'Proph_Mast', 'Blood_Sugar',
       'Cardiovascular', 'Digestive', 'Arthritis', 'Osteoporosis', 'Misc_Cond',
       'Blood_Sugar_Meds', 'Cardio_Meds', 'Gastro_Meds', 'Corticosteroids_',
       'adopted?', '# first deg relatives',
       '# first deg relatives with breast cancer',
       '# first deg relatives with any cancer',
       '# second deg relatives (granparents, aunts, uncles)',
       '# second deg relatives with breast cancer',
       '# second deg relatives with any cancer', 'mom had BC?',
       'mom had ovarian cancer?', '# sisters', '# sisters with breast cancer',
       '# sisters with ovarian cancer',
       '# first deg relatives with colorectal cancer',
       '# second deg relatives with colorectal cancer'],
      dtype='object')
```

In [89]:
```
# Plot histograms of family history attributes and health status

plt.figure(figsize=(37, 50))
index = 1
hs_col = 'Exam_Freq'
for col in fhhs.columns:
  if col not in hs.columns and col != 'ID':
    plt.subplot(7, 4, index)
    sns.histplot(data=fhhs, x=col,
                 hue=hs_col,
                 binwidth=0.5,
                 multiple='stack')
    plt.title(f"{col} and {hs_col}, subset by health status")
    index += 1
```



In [90]:
```
fhhs.columns
```

Out[90]:
```
Index(['ID', 'hospital_', 'Exam_Freq', 'Proph_Mast', 'Blood_Sugar',
       'Cardiovascular', 'Digestive', 'Arthritis', 'Osteoporosis', 'Misc_Cond',
       'Blood_Sugar_Meds', 'Cardio_Meds', 'Gastro_Meds', 'Corticosteroids_',
       'adopted?', '# first deg relatives',
       '# first deg relatives with breast cancer',
       '# first deg relatives with any cancer',
       '# second deg relatives (granparents, aunts, uncles)',
       '# second deg relatives with breast cancer',
       '# second deg relatives with any cancer', 'mom had BC?',
       'mom had ovarian cancer?', '# sisters', '# sisters with breast cancer',
       '# sisters with ovarian cancer',
       '# first deg relatives with colorectal cancer',
       '# second deg relatives with colorectal cancer'],
      dtype='object')
```

In [91]:
```
rh.head()
```

Out[91]:

	ID	Menarch Age	# Times Pregnant	# Live Births	Age at First Live Birth	Months Full Breast Feeding	Months Partial Breastfeeding	Age at Menopause	Hysterectomy	Ooph
0	1002	17.0	2.0	2.0	26.0	0.0	0.0	NaN	0.0	
1	1003	12.0	2.0	2.0	27.0	12.0	0.0	50.0	0.0	
2	1005	11.0	0.0	0.0	NaN	0.0	0.0	42.0	1.0	
3	1007	12.0	2.0	2.0	28.0	13.0	3.0	53.0	0.0	
4	1008	11.0	2.0	2.0	NaN	0.0	0.0	48.0	1.0	

Look at changes in biomarkers over time (baseline --> year 1 --> year 4), as well as study results (endpoints)

Working datasets: baseline, Year 1, Year 4

- Arithmetic method: (Year 4 - Baseline)
- Naming: example - “alphacar” (original) → “alphacar_ovt” (new column)

'ovt' ~ overtime

Data cleaning methods: Drop rows with NaN or unclear inputs (e.g., unknown)

In [92]:
```
hs.tail()
```

Out[92]:

	ID	hospital_	Exam_Freq	Proph_Mast	Blood_Sugar	Cardiovascular	Digestive	Arthritis	Ost
3083	1008	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
3084	1007	NaN	NaN	0.0	NaN	NaN	NaN	NaN	
3085	1005	NaN	NaN	0.0	NaN	NaN	NaN	NaN	
3086	1003	NaN	NaN	0.0	NaN	NaN	NaN	NaN	
3087	1002	NaN	NaN	0.0	NaN	NaN	NaN	NaN	

In [93]:
```
hs_y1.sort_values(by='id', ascending=False).tail()
```

Out[93]:

	id	hospital_	Exam_Freq	Proph_Mast	Blood_Sugar	Cardiovascular	Digestive	Arthritis	Osteop
4	1045	0	1	0	0	0	0	0	
3	1044	0	1	0	0	0	0	0	
2	1043	0	1	0	0	0	0	0	
1	1042	0	3	0	0	0	0	0	
0	1039	0	1	0	0	0	0	0	

In [94]:
```
hs_y4.sort_values(by='id', ascending=False).tail()
```

Out[94]:

	id	hospital_	Exam_Freq	Proph_Mast	Blood_Sugar	Cardiovascular	Digestive	Arthritis	Osteop
4	1012	0.0	3	0.0	0	0	0	1	
3	1010	1.0	3	0.0	0	0	0	1	
2	1007	0.0	2	0.0	0	0	0	0	
1	1005	0.0	3	0.0	0	0	0	0	
0	1002	0.0	2	0.0	0	0	0	0	

In [95]:
```
bc.head()
```

Out[95]:

	ID	lutein	cryptox	lycopene	alphacar	betacar
0	1002	0.2092	0.0597	0.5196	0.2083	0.5758
1	1003	0.2916	0.1067	0.9841	0.1786	0.4987
2	1005	0.3766	0.3836	0.9130	0.3229	1.0148
3	1007	0.4710	0.3985	0.5773	0.4239	0.9255
4	1008	0.8979	0.4957	0.7539	0.3842	1.3843

In [96]:
```
bc_y1.head()
```

Out[96]:

	ID	lutein	cryptox	lycopene	alphacar	betacar
0	1007	0.6372	0.2622	0.5536	1.0655	2.0221
1	1009	0.3964	0.1499	0.7195	0.1049	0.2694
2	1012	0.5388	0.4482	0.7514	0.1696	2.7975
3	1018	0.3734	0.2387	0.3903	0.3460	0.9311
4	1019	0.6775	0.0738	0.6714	0.1213	0.7233

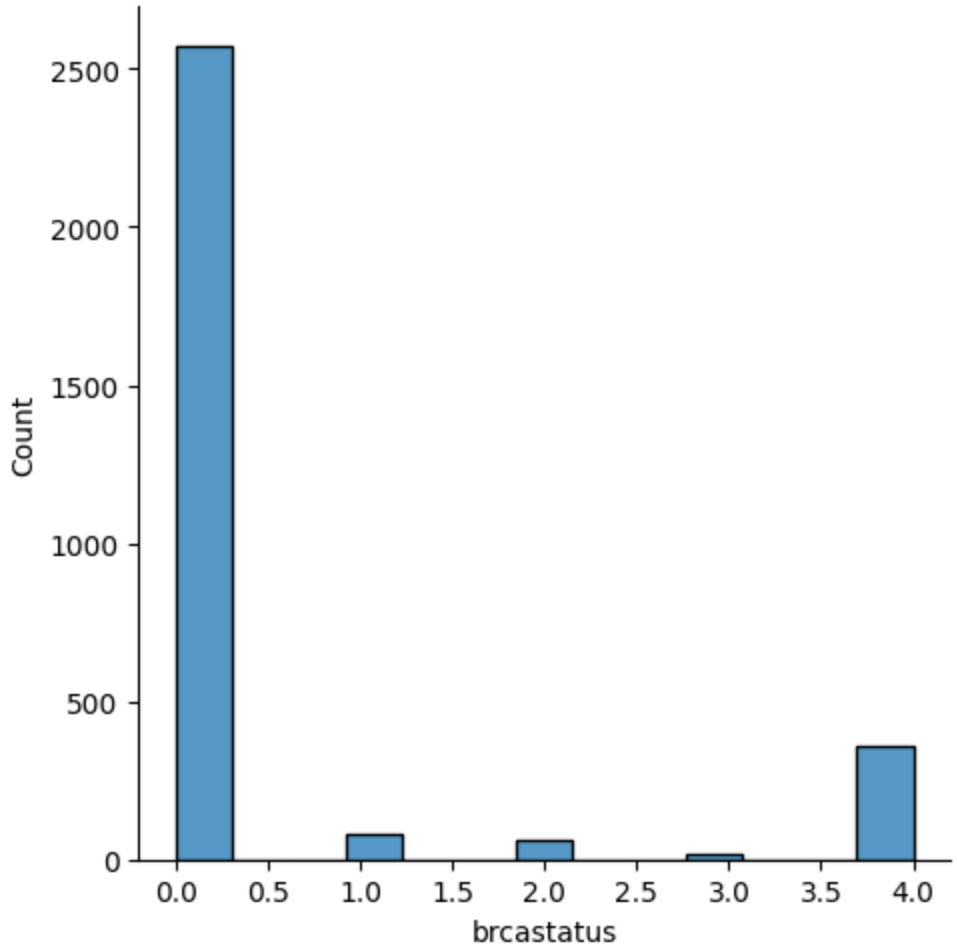
In [97]:
```
bc_y4.head()
```

Out[97]:

	ID	lutein	cryptox	lycopene	alphacar	betacar
0	1002	0.1894	0.0711	0.6854	0.1140	0.3465
1	1005	0.2592	0.0820	1.1692	0.1744	1.4004
2	1007	0.4459	0.2227	0.4298	0.5708	1.2759
3	1012	0.5416	0.4340	1.0422	0.0836	2.7283
4	1018	0.2659	0.1262	0.4447	0.1456	1.3234

In [98]:
```
sns.displot(x=ep['brcastatus'])
```

Out[98]:
```
<seaborn.axisgrid.FacetGrid at 0x7fb014ae27a0>
```



In [99]:
```
ep.columns
```

Out[99]:
```
Index(['id', 'intgrp', 'vitality', 'brcastatus', 'othcstatus',
       'brcacontribute', 'Year_Dx', 'grade', 'grade2', 'grade3', 'grademiss',
       'newstage', 'stage2', 'stage3', 'site0', 'recur_flag', 'yrsdx_rand',
       'yrsrand_endr', 'yrsdx_endr', 'yrsdx_endd'],
      dtype='object')
```

In [100…
```
ep.groupby(by="brcastatus").count().get('id')

# Breast Cancer Status as of 6/1/2006 or last prior contact
#               0 Confirmed - No Evidence of Recurrence
#               1 Confirmed – New Primary Breast Cancer
#               2 Confirmed – Local Recurrence
#               3 Confirmed – Regional Recurrence
#               4 Confirmed – Distant Recurrence

#               0 Confirmed - No Evidence of Recurrence
#               1 Confirmed - Recurrent (1+2+3+4)
```

Out[100]:
```
brcastatus
0    2570
1      78
2      63
3      20
4     357
Name: id, dtype: int64
```

In [101…
```
ep.groupby(by="recur_flag").count().get('id')

# Recurrence Flag
#       0 No Invasive Breast Cancer Events
#       1 Invasive Breast Cancer Event
```

Out[101]:
```
recur_flag
0    2570
1     518
Name: id, dtype: int64
```

In [102…
```
ep.groupby(by="grade").count().get('id')

# Cancer Grade
#               0 - Grade Not Applicable or Not Available
#               1 - Grade I, Well Differentiated
#               2 - Grade II, Moderately Differentiated
#               3 - Grade III, Poorly Differentiated
```

Out[102]:
```
grade
0     256
1     484
2    1240
3    1108
Name: id, dtype: int64
```

In [103…
```
ep.groupby(by="newstage").count().get('id')

# Cancer Stage, AJCC 6th
#               1 Stage I
#               2 Stage IIA
#               3 Stage IIB
#               4 Stage IIIA
#               5 Stage IIIC
```

Out[103]:
```
newstage
1    1191
2    1026
3     384
4     373
5     114
Name: id, dtype: int64
```

In [104…
```
x = bc[bc['ID'] == 1002]['lutein'].values[0]
print(x)
```

```
0.2092
```

In [105…
```
datasets['Thoughts and Feelings']
```

Out[105]:

	ID	ss_tot	ss_emot	ss_aff	ss_tang	ss_pos	lv_alone	petownr	church	religion	...	sympto
1	1003	45.0	20.0	5.0	10.0	10.0	0.0	1.0	5.0	3.0	...	0.0
7	1011	43.0	19.0	5.0	10.0	9.0	0.0	1.0	1.0	2.0	...	0.4
10	1017	42.0	19.0	5.0	9.0	9.0	0.0	1.0	1.0	2.0	...	0.6
11	1018	31.0	10.0	5.0	8.0	8.0	0.0	0.0	1.0	3.0	...	0.3
12	1019	33.0	13.0	4.0	8.0	8.0	1.0	0.0	1.0	2.0	...	0.2
...	...	...	...	...	...	...	...	...	...	...	...	
3083	13241	36.0	13.0	5.0	10.0	8.0	1.0	1.0	1.0	3.0	...	0.7
3084	13242	40.0	16.0	5.0	9.0	10.0	0.0	1.0	1.0	1.0	...	0.0
3085	13243	37.0	14.0	5.0	8.0	10.0	0.0	1.0	1.0	2.0	...	0.4
3086	13244	38.0	16.0	4.0	10.0	8.0	0.0	1.0	1.0	2.0	...	0.4
3087	13245	26.0	12.0	4.0	4.0	6.0	0.0	1.0	3.0	3.0	...	0.5

2861 rows × 38 columns

In [106…

```
def calc_changes_ovt(dataset_type, col_name, new_dataset):

  """
      Calculate change over time in column [col_name] for every patient

      dataset_type ~ type of dataset, such as column name "alphacar" would be
      under dataset type blood carotenoids

      col_name ~ the column on which changes over time are to be calculated

      new_dataset ~ new dataset to store the changes over time
  """

  df = datasets[dataset_type]
  # df_y1 = datasets_y1[dataset_type]
  df_y4 = datasets_y4[dataset_type]

  id_col_name = 'id' if 'id' in df.columns else 'ID'
  # id_col_name_y1 = 'id' if 'id' in df_y1.columns else 'ID'
  id_col_name_y4 = 'id' if 'id' in df_y4.columns else 'ID'

  changes = []
  ids = df[id_col_name]
  # print(ids)

  if 'ID' not in new_dataset.columns:
    new_dataset['ID'] = ids

  for id in ids:

    start = df[df[id_col_name] == id][col_name].values[0]

    if id in df_y4[id_col_name_y4].values:
```

```
      end = df_y4[df_y4[id_col_name_y4] == id][col_name].values[0]
    # elif id in df_y1[id_col_name_y1].values:
    #   end = df_y1[df_y1[id_col_name_y4] == id][col_name].values[0]
    else:
      change = np.float('nan')
      changes.append(change)
      continue

    # print(id)
    # print(end)
    # print('*****end*****')
    # print(type(end))
    # print(start)
    # print('*****start*****')
    # print(type(start))

    change = end - start
    changes.append(change)

  new_col_name = f"{col_name}_ovt"
  new_dataset[new_col_name] = changes

  return None
```

In [107…

```
tf.head()
```

Out[107]:

	ID	ss_tot	ss_emot	ss_aff	ss_tang	ss_pos	lv_alone	petownr	church	religion	...	symptom
1	1003	45.0	20.0	5.0	10.0	10.0	0.0	1.0	5.0	3.0	...	0.057
7	1011	43.0	19.0	5.0	10.0	9.0	0.0	1.0	1.0	2.0	...	0.457
10	1017	42.0	19.0	5.0	9.0	9.0	0.0	1.0	1.0	2.0	...	0.629
11	1018	31.0	10.0	5.0	8.0	8.0	0.0	0.0	1.0	3.0	...	0.343
12	1019	33.0	13.0	4.0	8.0	8.0	1.0	0.0	1.0	2.0	...	0.200

5 rows × 38 columns

In [108…

```
# Calculate changes in blood carotenoids over time

bc_ovt = pd.DataFrame()
for col in bc.columns:
  if col not in ['id', 'ID']:
    calc_changes_ovt('Blood Carotenoids', col, bc_ovt)

bc_ovt
```

Out[108]:

	ID	lutein_ovt	cryptox_ovt	lycopene_ovt	alphacar_ovt	betacar_ovt
0	1002	-0.0198	0.0114	0.1658	-0.0943	-0.2293
1	1003	NaN	NaN	NaN	NaN	NaN
2	1005	-0.1174	-0.3016	0.2562	-0.1485	0.3856
3	1007	-0.0251	-0.1758	-0.1475	0.1469	0.3504
4	1008	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...
3083	13241	0.0065	0.0827	0.4094	0.0154	0.2861
3084	13242	0.0932	-0.1789	0.2133	0.1499	0.8042
3085	13243	-0.0045	0.1245	-0.1687	0.0116	-0.1035
3086	13244	-0.2045	-0.0240	-0.0402	-0.1501	-0.7837
3087	13245	-0.1546	-0.0867	-0.4676	-0.4169	-0.4602

3088 rows × 6 columns

In [109…

```
# Drop all rows with nan

bc_ovt.dropna(inplace = True)
```

In [110…

```
# Merge bc_ovt with endpoint
bc_ovt.rename(columns={'ID':'id'}, inplace=True)
bcep = bc_ovt.merge(right=ep, how='inner', on='id')
```

In [111…

```
bcep
```

Out[111]:

	id	lutein_ovt	cryptox_ovt	lycopene_ovt	alphacar_ovt	betacar_ovt	intgrp	vitality	brcasta
0	1002	-0.0198	0.0114	0.1658	-0.0943	-0.2293	3	1	
1	1005	-0.1174	-0.3016	0.2562	-0.1485	0.3856	4	1	
2	1007	-0.0251	-0.1758	-0.1475	0.1469	0.3504	3	1	
3	1012	0.0285	-0.2532	0.0965	0.0203	1.6420	4	1	
4	1018	-0.0347	0.0303	-0.0820	0.0918	0.8330	3	1	
...	...	...	...	...	...	...	...	...	
2084	13241	0.0065	0.0827	0.4094	0.0154	0.2861	4	1	
2085	13242	0.0932	-0.1789	0.2133	0.1499	0.8042	3	1	
2086	13243	-0.0045	0.1245	-0.1687	0.0116	-0.1035	3	1	
2087	13244	-0.2045	-0.0240	-0.0402	-0.1501	-0.7837	4	1	
2088	13245	-0.1546	-0.0867	-0.4676	-0.4169	-0.4602	3	1	

2089 rows × 25 columns

In [112…

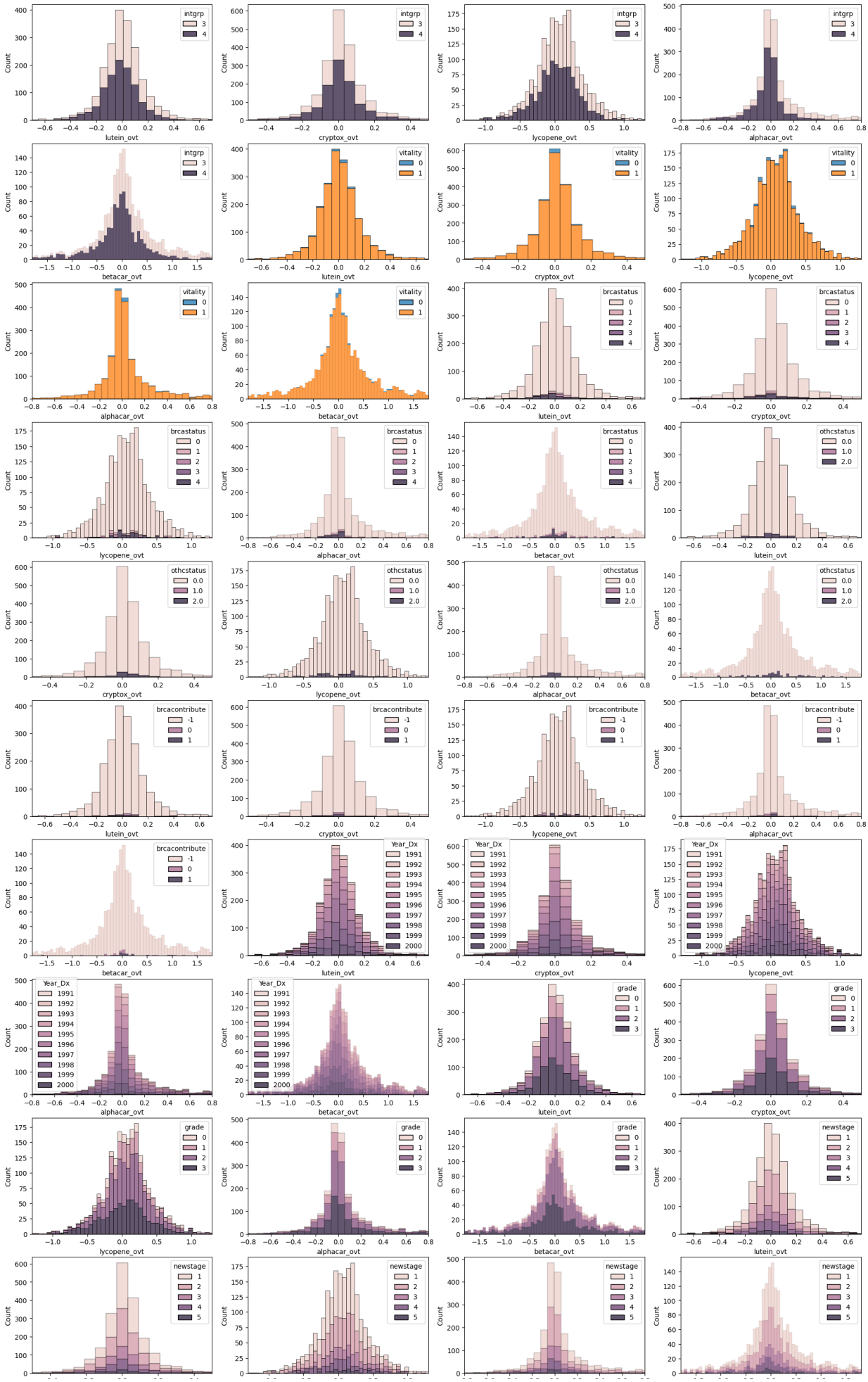
```
ep.columns
```

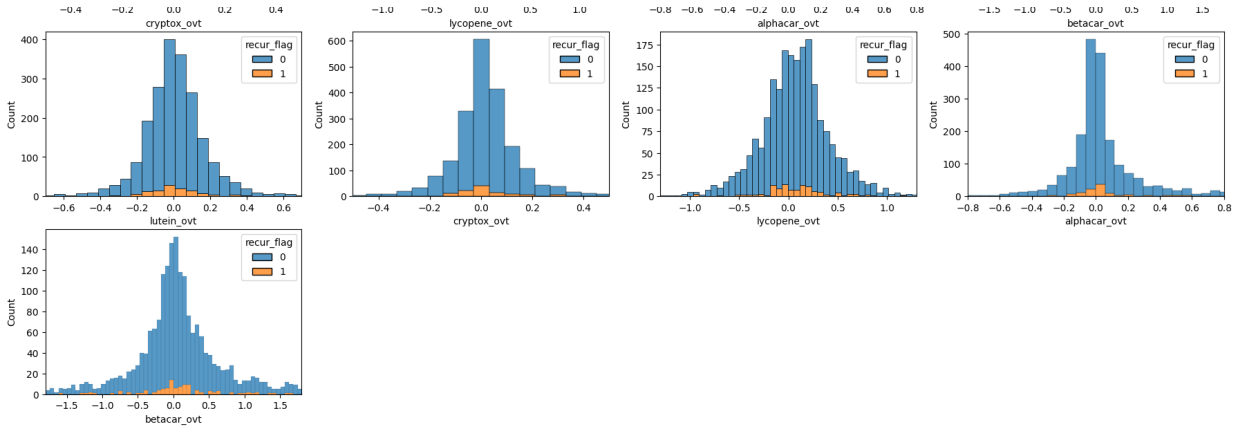
Out[112]:

```
Index(['id', 'intgrp', 'vitality', 'brcastatus', 'othcstatus',
       'brcacontribute', 'Year_Dx', 'grade', 'grade2', 'grade3', 'grademiss',
       'newstage', 'stage2', 'stage3', 'site0', 'recur_flag', 'yrsdx_rand',
       'yrsrand_endr', 'yrsdx_endr', 'yrsdx_endd'],
      dtype='object')
```

In [113…

```
# Plot bcep to measure relationships

columns = ['intgrp', 'vitality', 'brcastatus', 'othcstatus',
       'brcacontribute', 'Year_Dx', 'grade', 'newstage', 'recur_flag']
idx = 1
plt.figure(figsize=(22, 44))

for ep_col in columns:
  for bc_col in bc_ovt.columns[1:]:
    plt.subplot(12, 4, idx)
    sns.histplot(data=bcep, x=bc_col, hue=ep_col, binwidth=0.06, multiple='stack')
    if bc_col == 'lutein_ovt':
      plt.xlim(-0.7, 0.7)
    elif bc_col == 'cryptox_ovt':
      plt.xlim(-0.5, 0.5)
    elif bc_col == 'lycopene_ovt':
      plt.xlim(-1.3, 1.3)
    elif bc_col == 'alphacar_ovt':
      plt.xlim(-0.8, 0.8)
    else:
      plt.xlim(-1.8, 1.8)
    idx += 1
```


intgrp
3
4
vitality
0
1
brcastatus
0
1
2
3
4
othcstatus
0.0
1.0
2.0
brcacontribute
-1
0
1
Year_Dx
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
grade
0
1
2
3
newstage
1
2
3
4
5
Count
lutein_ovt
cryptox_ovt
lycopene_ovt
alphacar_ovt
betacar_ovt



In [114…

```
# Merge quality of living column with bcep

QOL = tf[['ID', 'QOL', 'optimism', 'physhealth', 'mental health', 'genhlth']]
QOL.rename(columns={'ID':'id'}, inplace=True)
bcep_tf = bcep.merge(right=QOL, how='inner', on='id')
```

In [115…

```
bcep_tf.dropna(inplace=True)
bcep_tf = bcep_tf[bcep_tf['QOL'] != ' ']
bcep_tf
```

Out[115]:

	id	lutein_ovt	cryptox_ovt	lycopene_ovt	alphacar_ovt	betacar_ovt	intgrp	vitality	brcasta
0	1018	-0.0347	0.0303	-0.0820	0.0918	0.8330	3	1	
1	1019	0.4003	0.0880	-0.0739	0.1937	0.3823	4	1	
2	1022	0.0056	-0.0503	-0.2200	0.0528	0.3148	3	1	
3	1026	-0.1215	-0.3689	-0.3149	0.0722	0.3063	4	1	
4	1027	0.1232	0.1394	0.4131	0.0773	0.9563	4	1	
...	...	...	...	...	...	...	...	...	
1939	13241	0.0065	0.0827	0.4094	0.0154	0.2861	4	1	
1940	13242	0.0932	-0.1789	0.2133	0.1499	0.8042	3	1	
1941	13243	-0.0045	0.1245	-0.1687	0.0116	-0.1035	3	1	
1942	13244	-0.2045	-0.0240	-0.0402	-0.1501	-0.7837	4	1	
1943	13245	-0.1546	-0.0867	-0.4676	-0.4169	-0.4602	3	1	

1937 rows × 30 columns

In [116…

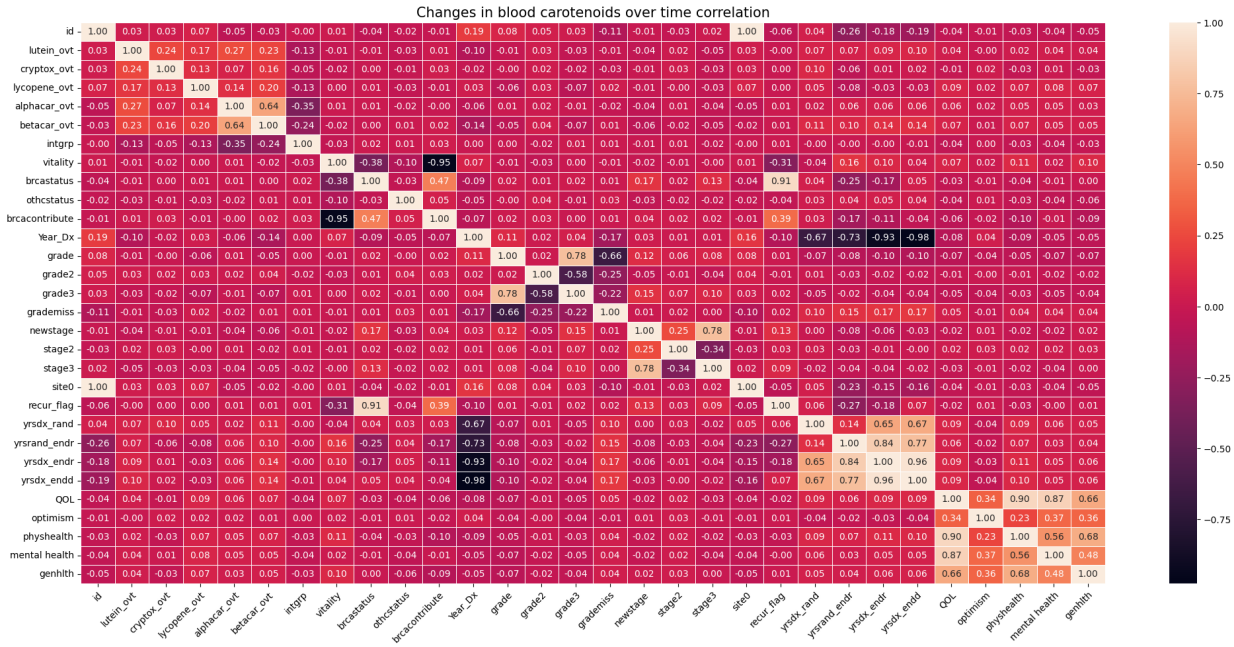
```
# Convert 'QOL', 'physical health', 'mental health' to float

bcep_tf['QOL'] = bcep_tf['QOL'].astype(float)
bcep_tf['physhealth'] = bcep_tf['physhealth'].astype(float)
bcep_tf['mental health'] = bcep_tf['mental health'].astype(float)
```

In [117…
```
bcep_tf.dtypes
```

Out[117]:
```
id                  int64
lutein_ovt        float64
cryptox_ovt       float64
lycopene_ovt      float64
alphacar_ovt      float64
betacar_ovt       float64
intgrp              int64
vitality            int64
brcastatus          int64
othcstatus        float64
brcacontribute      int64
Year_Dx             int64
grade               int64
grade2              int64
grade3              int64
grademiss           int64
newstage            int64
stage2              int64
stage3              int64
site0               int64
recur_flag          int64
yrsdx_rand        float64
yrsrand_endr      float64
yrsdx_endr        float64
yrsdx_endd        float64
QOL               float64
optimism          float64
physhealth        float64
mental health     float64
genhlth           float64
dtype: object
```

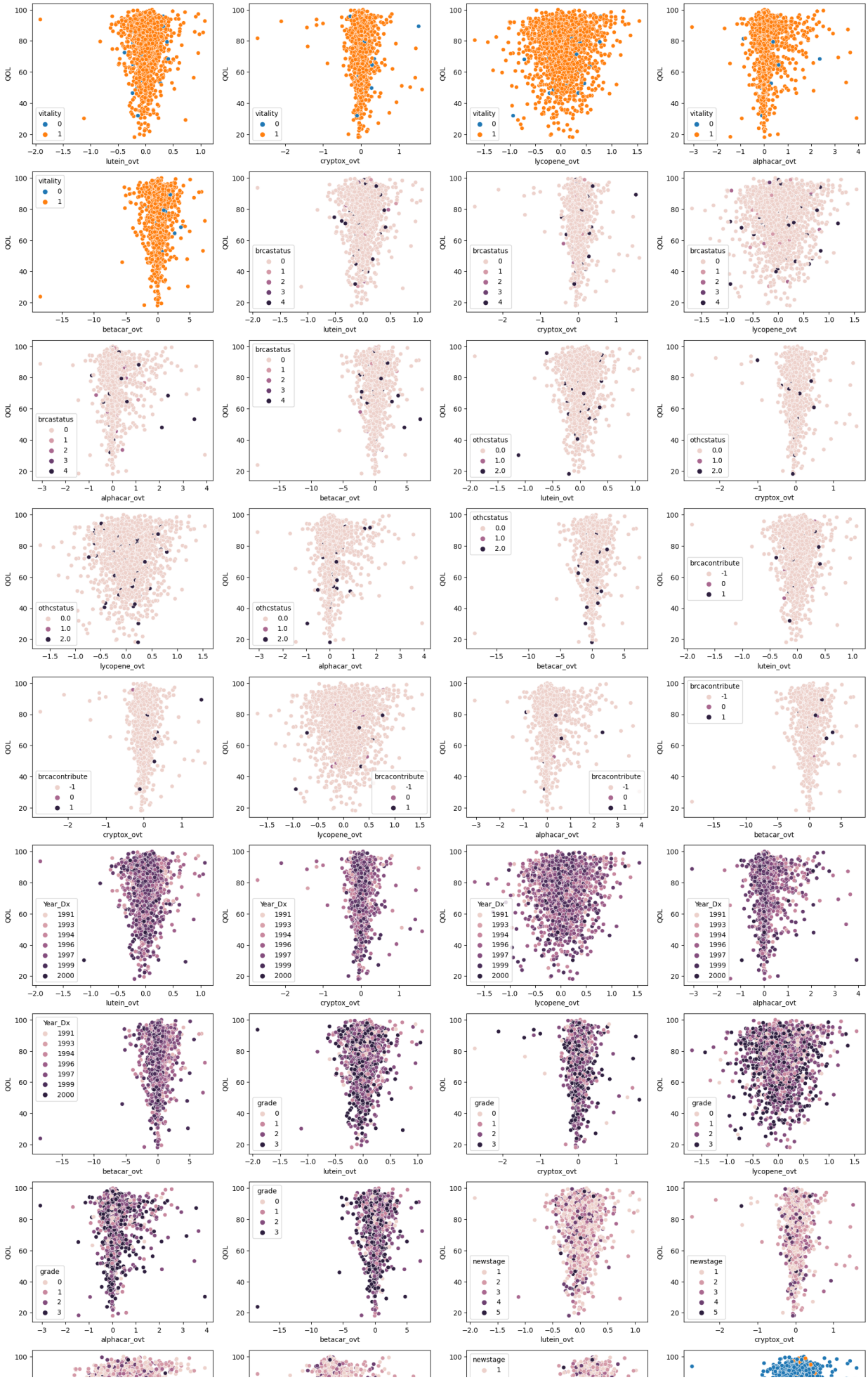
In [118…
```
# Plot a heat map of bcep

plt.figure(figsize = (25, 11))
ax = sns.heatmap(bcep_tf.corr(), annot = True, fmt = '.2f', linewidth = 0.5)
props = plt.setp(ax.xaxis.get_majorticklabels(), rotation=45, ha='right', rotation_moc
plt.title('Changes in blood carotenoids over time correlation', size = 15)
```

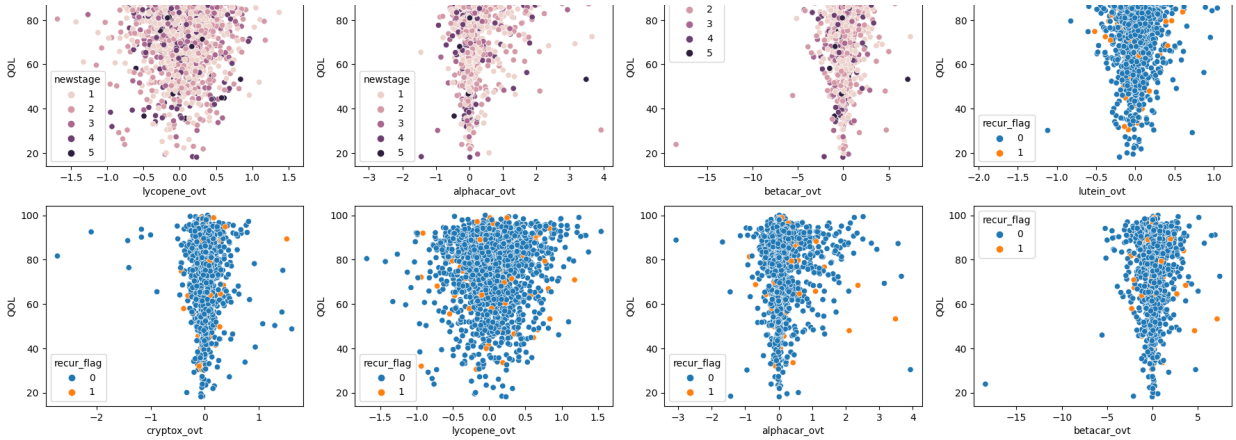
Out[118]:
```
Text(0.5, 1.0, 'Changes in blood carotenoids over time correlation')
```



In [119…

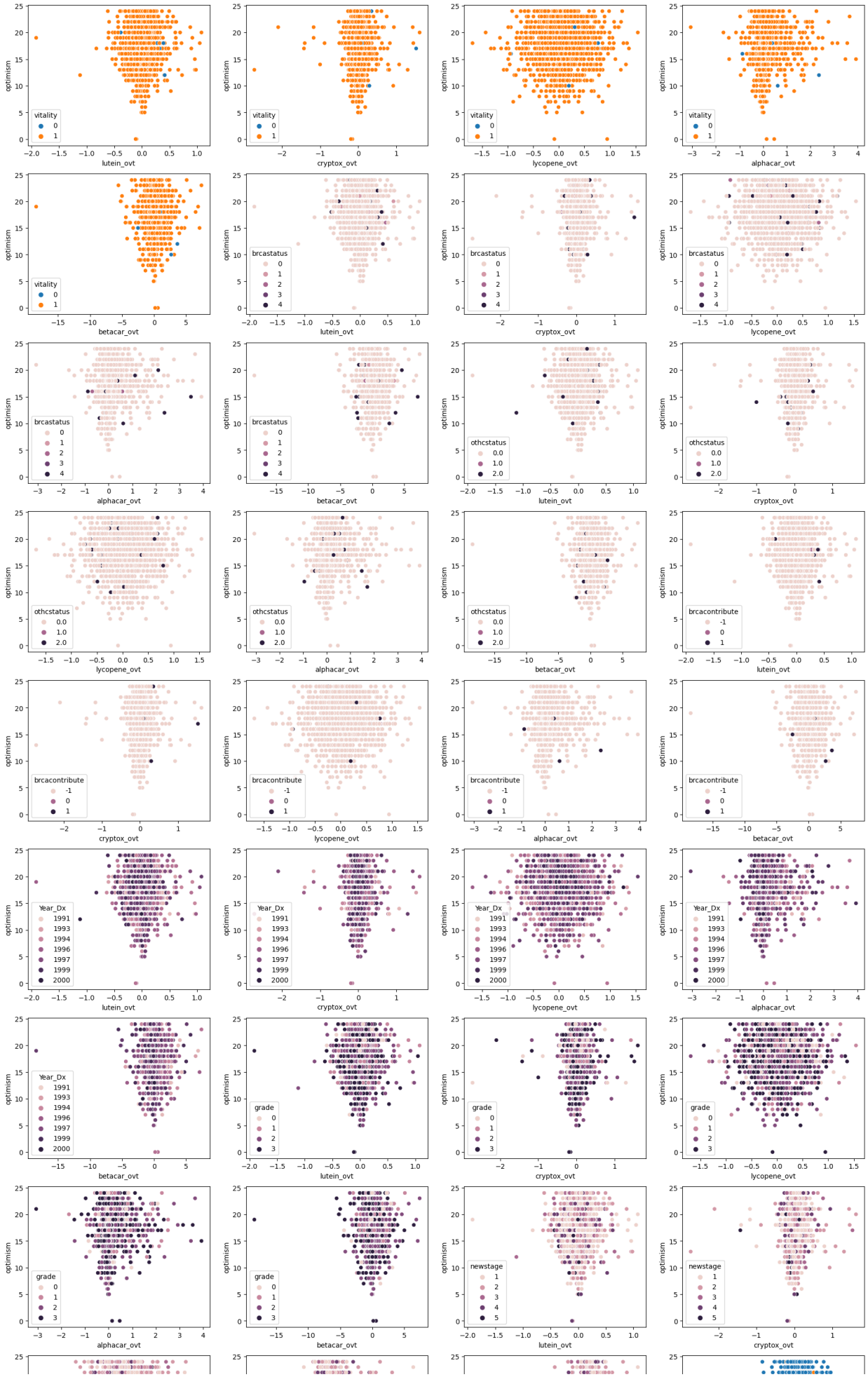
```
# Plot bcep_tf to measure relationships wrt quality of living

columns = ['vitality', 'brcastatus', 'othcstatus',
       'brcacontribute', 'Year_Dx', 'grade', 'newstage', 'recur_flag']
idx = 1
plt.figure(figsize=(22, 44))

for ep_col in columns:
  for bc_col in bc_ovt.columns[1:]:
    plt.subplot(10, 4, idx)
    sns.scatterplot(data=bcep_tf, x=bc_col, y="QOL", hue=ep_col)
    idx += 1
```

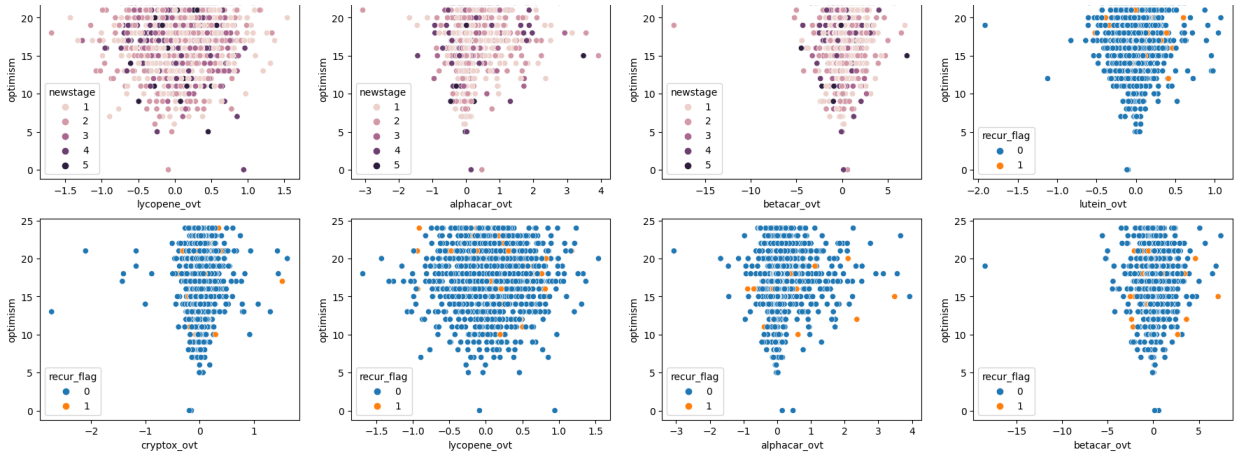

QOL
lutein_ovt
cryptox_ovt
lycopene_ovt
alphacar_ovt
betacar_ovt
vitality
0
1
brcastatus
0
1
2
3
4
othcstatus
0.0
1.0
2.0
brcacontribute
-1
0
1
Year_Dx
1991
1993
1994
1996
1997
1999
2000
grade
0
1
2
3
newstage
1
2
3
4
5



In [120…

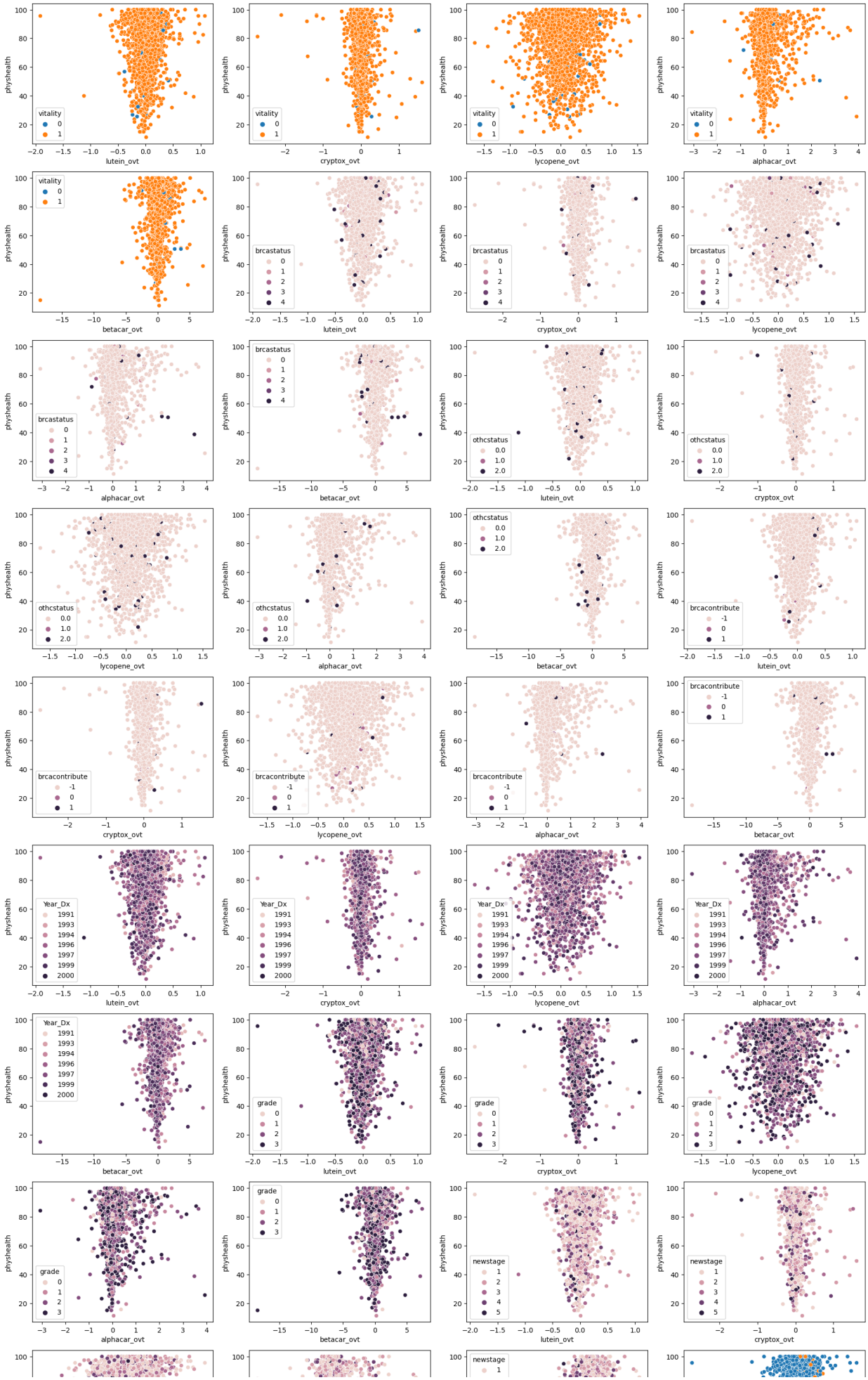
```
# Plot bcep_tf to measure relationships wrt optimism

columns = ['vitality', 'brcastatus', 'othcstatus',
       'brcacontribute', 'Year_Dx', 'grade', 'newstage', 'recur_flag']
idx = 1
plt.figure(figsize=(22, 44))

for ep_col in columns:
  for bc_col in bc_ovt.columns[1:]:
    plt.subplot(10, 4, idx)
    sns.scatterplot(data=bcep_tf, x=bc_col, y="optimism", hue=ep_col)
    idx += 1
```

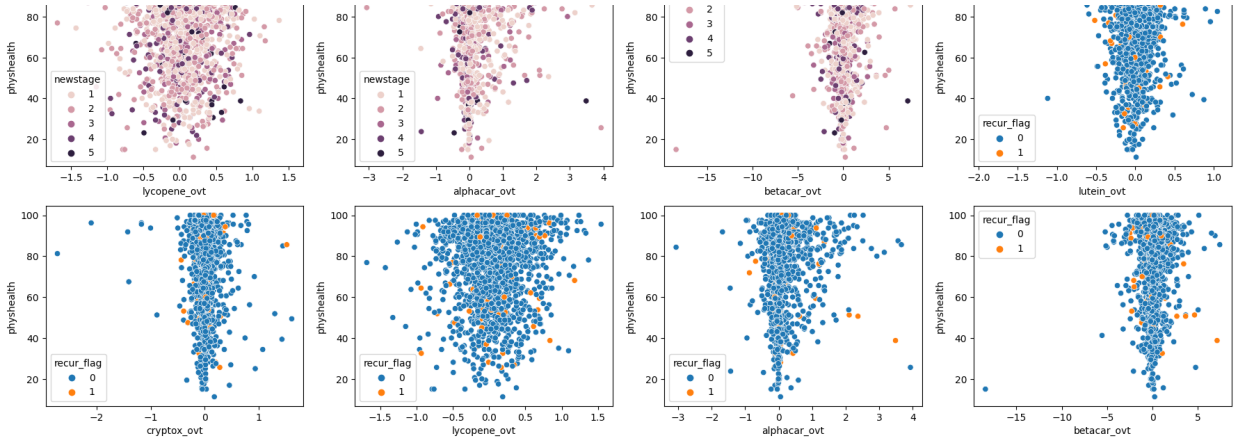

optimism
lutein_ovt
cryptox_ovt
lycopene_ovt
alphacar_ovt
betacar_ovt
vitality
0
1
brcastatus
0
1
2
3
4
othcstatus
0.0
1.0
2.0
brcacontribute
-1
0
1
Year_Dx
1991
1993
1994
1996
1997
1999
2000
grade
0
1
2
3
newstage
1
2
3
4
5



In [121…

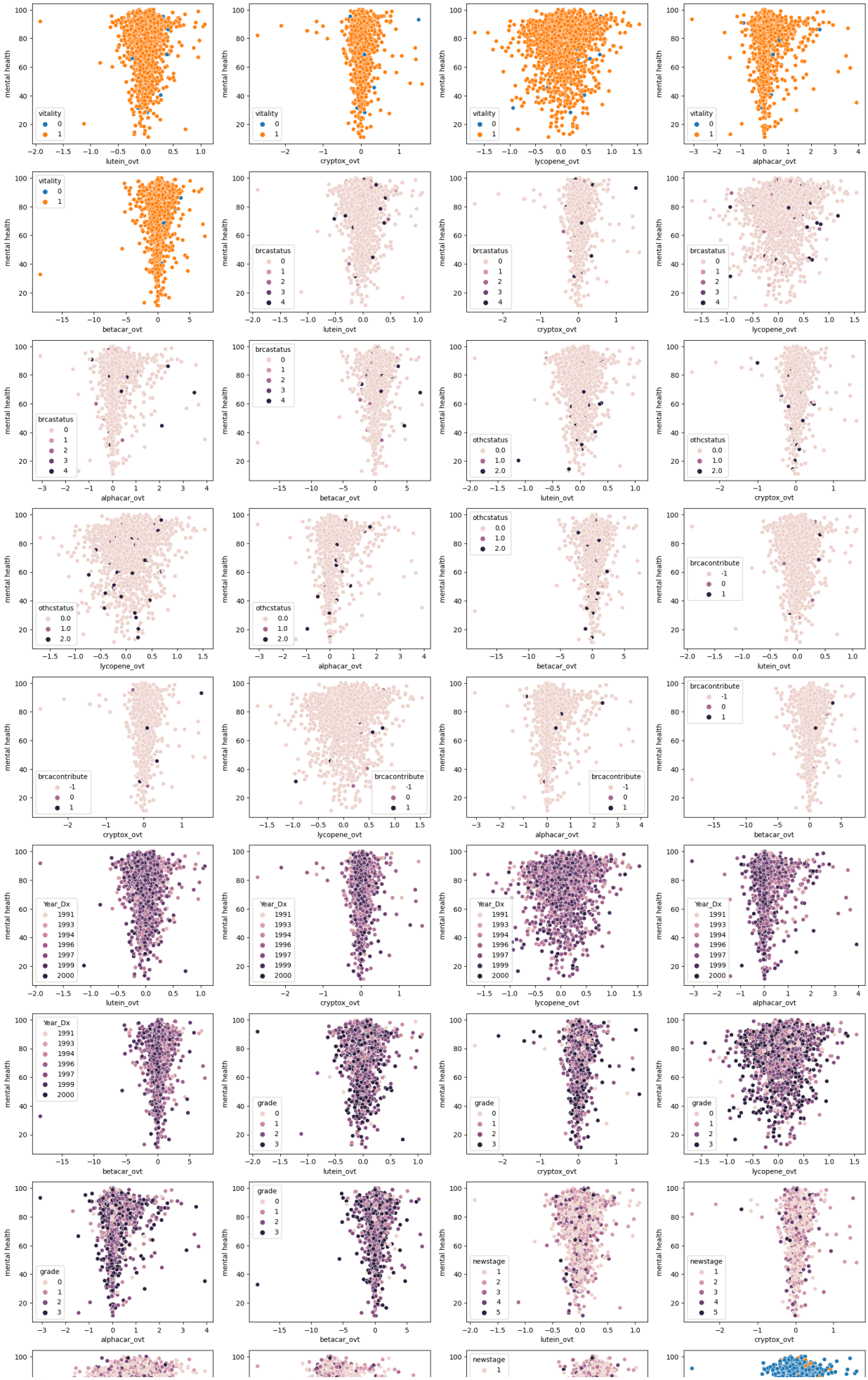
```
# Plot bcep_ls to measure relationships wrt to physical health

columns = ['vitality', 'brcastatus', 'othcstatus',
       'brcacontribute', 'Year_Dx', 'grade', 'newstage', 'recur_flag']
idx = 1
plt.figure(figsize=(22, 44))

for ep_col in columns:
  for bc_col in bc_ovt.columns[1:]:
    plt.subplot(10, 4, idx)
    sns.scatterplot(data=bcep_tf, x=bc_col, y="physhealth", hue=ep_col)
    idx += 1
```

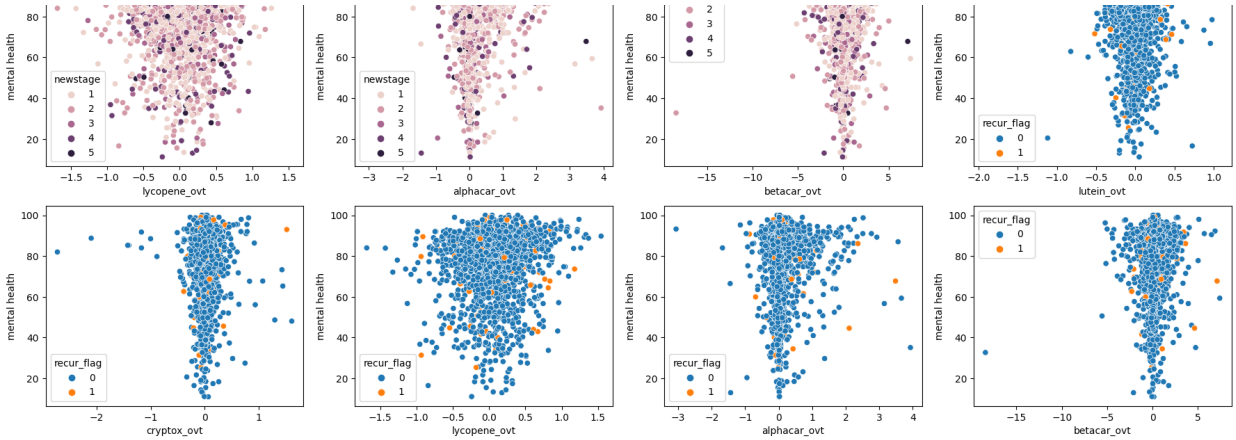

physhealth
lutein_ovt
cryptox_ovt
lycopene_ovt
alphacar_ovt
betacar_ovt
vitality
0
1
brcastatus
0
1
2
3
4
othcstatus
0.0
1.0
2.0
brcacontribute
-1
0
1
Year_Dx
1991
1993
1994
1996
1997
1999
2000
grade
0
1
2
3
newstage
1
2
3
4
5



In [122…

```
# Plot bcep_ls to measure relationships wrt mental health

columns = ['vitality', 'brcastatus', 'othcstatus',
       'brcacontribute', 'Year_Dx', 'grade', 'newstage', 'recur_flag']
idx = 1
plt.figure(figsize=(22, 44))

for ep_col in columns:
  for bc_col in bc_ovt.columns[1:]:
    plt.subplot(10, 4, idx)
    sns.scatterplot(data=bcep_tf, x=bc_col, y="mental health", hue=ep_col)
    idx += 1
```


mental health
lutein_ovt
cryptox_ovt
lycopene_ovt
alphacar_ovt
betacar_ovt
vitality
0
1
brcastatus
0
1
2
3
4
othcstatus
0.0
1.0
2.0
brcacontribute
-1
0
1
Year_Dx
1991
1993
1994
1996
1997
1999
2000
grade
0
1
2
3
newstage
1
2
3
4
5



In [123…

```
tf.columns
```

Out[123]:

```
Index(['ID', 'ss_tot', 'ss_emot', 'ss_aff', 'ss_tang', 'ss_pos', 'lv_alone',
       'petownr', 'church', 'religion', 'club', 'caregvB', 'strain',
       'optimism', 'nee', 'aee', 'hostil', 'QOL', 'physhealth', 'physfctn',
       'lim_phys', 'pain', 'genhlth', 'mental health', 'lim_emot', 'energy',
       'emotwell', 'socfctn', 'symptom', 'bloat', 'sweats', 'aches',
       'hotflash', 'tired', 'totlfevt', 'dep_log', 'whi_irs_old',
       'WHI_irs_new'],
      dtype='object')
```

In [124…

```
# Exporting bcep_tf to work in "exploratory-analysis-2.ipynb"

path = '/content/drive/MyDrive/NIEMA_Lab/WHEL_Data/Preprocessed-Data/bcep_tf.csv'

with open(path, 'w') as f:
  bcep_tf.to_csv(f)
```

In [125…

```
# Calculate changes in QOL, mental health, physical health, general health,
# optimism, pain, and social functioning over time

ovt = pd.DataFrame()
columns = ['QOL', 'energy', 'genhlth', 'optimism', 'pain', 'socfctn']
for col in columns:
  calc_changes_ovt('Thoughts and Feelings', col, ovt);

ovt.head()
```

Out[125]:

	ID	QOL_ovt	energy_ovt	genhlth_ovt	optimism_ovt	pain_ovt	socfctn_ovt
1	1003	NaN	NaN	NaN	NaN	NaN	NaN
7	1011	NaN	NaN	NaN	NaN	NaN	NaN
10	1017	NaN	NaN	NaN	NaN	NaN	NaN
11	1018	-15.750000	5.0	0.0	0.0	-37.5	-12.5
12	1019	1.855469	15.0	-15.0	0.0	-12.5	0.0

Any cells below related to analyzing lost rows through preprocessing operations are conducted to examine potential biases in the study, specifically to see which segments of the

study population are *unintentionally* excluded from final (cleaned) dataset.

In [126…

```
df_before = ovt.copy()
ovt.dropna(inplace=True)
df_after = ovt.copy()
row_loss, loss_ids = examine_loss(df_before, df_after, "calculate changes over time \r
print(row_loss)
print(loss_ids)
```

```
Operation: calculate changes over time
for thoughts & feelings
Before: 2861 rows
After: 1989 rows
Loss: 872 rows

[1003, 1011, 1017, 1034, 1039, 1041, 1045, 1046, 1052, 1054, 1056, 1057, 1058, 1061,
1065, 1067, 1070, 1071, 1074, 1076, 1077, 1080, 1083, 1088, 1092, 1098, 1099, 1102, 1
103, 1108, 1110, 1113, 1119, 1125, 1126, 1129, 1130, 1131, 1133, 1135, 1136, 1137, 11
38, 1139, 1144, 1145, 1149, 1151, 1155, 1158, 1159, 1160, 1162, 1164, 1165, 1173, 117
4, 1177, 1181, 1184, 1190, 1193, 1194, 1198, 1201, 1202, 1206, 1209, 1213, 1216, 122
0, 1225, 1231, 1232, 1233, 1235, 1236, 1237, 1238, 1239, 1241, 1244, 1246, 1247, 125
5, 1260, 1263, 1264, 1267, 1268, 1269, 1272, 1275, 1280, 1283, 1284, 1291, 1293, 129
4, 1299, 1306, 1312, 1313, 1314, 1316, 1317, 1319, 1324, 1330, 1341, 1343, 1345, 134
6, 1348, 1349, 1351, 1353, 1354, 1355, 1360, 1361, 1363, 1364, 1368, 1373, 1377, 137
8, 1380, 1381, 1383, 1384, 1385, 1389, 1391, 1397, 1402, 1404, 1406, 1410, 1412, 141
3, 1415, 1421, 1422, 1423, 1425, 1430, 1438, 1441, 1443, 1444, 1450, 1453, 1454, 145
7, 1458, 1461, 1464, 1467, 1468, 1469, 1472, 1474, 1476, 1477, 1482, 1484, 1485, 149
1, 1496, 1499, 1504, 1506, 1507, 1508, 1510, 1513, 1514, 1516, 1523, 1524, 1526, 152
7, 1529, 1531, 1532, 1534, 1535, 1540, 1542, 1544, 1549, 1550, 1552, 1554, 1555, 155
7, 1561, 1562, 3001, 3008, 3009, 3010, 3014, 3017, 3019, 3020, 3027, 3038, 3040, 304
5, 3051, 3052, 3070, 3081, 3087, 3088, 3089, 3090, 3099, 3105, 3112, 3114, 3120, 312
1, 3126, 3128, 3129, 3132, 3133, 3135, 3136, 3148, 3149, 3151, 3154, 3155, 3160, 317
3, 3177, 3181, 3182, 3188, 3189, 3199, 3202, 3207, 3208, 3209, 3211, 3215, 3219, 322
0, 3221, 3222, 3228, 3230, 3231, 3234, 3242, 3244, 3248, 3260, 3273, 3274, 3279, 328
3, 3288, 3290, 3291, 3294, 3296, 3297, 3300, 3307, 3313, 3320, 3321, 3323, 3334, 334
5, 3350, 3356, 3361, 3364, 3367, 3374, 3378, 3384, 3389, 3392, 3395, 3397, 3398, 339
9, 3400, 3403, 3406, 3411, 3415, 3418, 3425, 3434, 3436, 3445, 3452, 5001, 5007, 501
2, 5044, 5047, 5048, 5051, 5054, 5068, 5074, 5076, 5085, 5088, 5093, 5095, 5103, 510
8, 5112, 5115, 5116, 5118, 5120, 5122, 5126, 5128, 5130, 5131, 5134, 5135, 5137, 513
8, 5141, 5150, 5155, 5156, 5158, 5165, 5168, 5171, 5174, 5175, 5176, 5177, 5193, 520
5, 5209, 5210, 5211, 5225, 5233, 5238, 5243, 5244, 5246, 5249, 5251, 5254, 5256, 526
0, 5266, 5273, 5276, 5277, 5282, 5283, 5286, 5287, 5289, 5290, 5292, 5296, 5298, 530
0, 5301, 5302, 5304, 5305, 5310, 5311, 5312, 5314, 5316, 5319, 5320, 5324, 5330, 533
1, 5334, 5336, 5341, 5357, 5358, 5361, 5362, 5363, 5364, 5365, 5366, 5369, 5378, 537
9, 5387, 5390, 5391, 5392, 5394, 5401, 5402, 5403, 5409, 5415, 5418, 5422, 5425, 542
7, 5428, 5433, 5436, 5439, 5440, 5441, 5444, 5446, 5448, 5457, 5459, 5465, 5471, 548
7, 5490, 5492, 5499, 5506, 5515, 5524, 5532, 5537, 5538, 5541, 5548, 5549, 5554, 556
3, 5575, 7001, 7002, 7005, 7017, 7019, 7023, 7031, 7033, 7034, 7035, 7038, 7043, 704
4, 7048, 7051, 7053, 7059, 7066, 7068, 7069, 7079, 7092, 7094, 7098, 7099, 7106, 710
7, 7108, 7114, 7121, 7123, 7124, 7125, 7127, 7130, 7134, 7135, 7138, 7144, 7148, 715
0, 7152, 7158, 7160, 7161, 7162, 7164, 7165, 7170, 7178, 7179, 7181, 7183, 7184, 720
3, 7208, 7210, 7213, 7215, 7220, 7237, 7239, 7242, 7243, 7245, 7246, 7250, 7274, 727
8, 7280, 7283, 7287, 7291, 7296, 7299, 7300, 7304, 7309, 7310, 7315, 7321, 7324, 732
5, 7326, 7330, 7331, 7332, 7336, 7344, 7345, 7346, 7348, 7349, 7352, 7356, 7358, 736
4, 7365, 7366, 7369, 7378, 7380, 7381, 7385, 7386, 7388, 7391, 7392, 7394, 7396, 740
2, 7403, 7406, 7410, 7414, 7421, 7423, 7432, 7433, 7435, 7439, 7441, 7448, 7449, 745
4, 7456, 7458, 7460, 7463, 7464, 7465, 7470, 7479, 7483, 7488, 7491, 7492, 7493, 750
0, 7501, 7507, 7512, 7513, 7519, 9001, 9002, 9004, 9014, 9015, 9019, 9021, 9023, 903
1, 9043, 9052, 9054, 9055, 9057, 9059, 9064, 9070, 9074, 9077, 9081, 9085, 9090, 909
2, 9097, 9103, 9107, 9108, 9109, 9112, 9116, 9121, 9138, 9141, 9145, 9146, 9150, 915
4, 9157, 9159, 9160, 9162, 9170, 9176, 9182, 9190, 9204, 9209, 9231, 9232, 9238, 924
1, 9242, 9245, 9249, 9253, 9255, 9258, 9269, 9273, 9276, 9281, 9283, 9289, 9291, 929
5, 9298, 9299, 9327, 9334, 9335, 9342, 9352, 9353, 9354, 9356, 9362, 9369, 9373, 937
5, 9376, 9381, 9391, 9393, 9400, 9401, 9413, 9415, 9425, 9431, 9439, 9446, 9450, 945
2, 9461, 9464, 9465, 9471, 9479, 9482, 9494, 9503, 9510, 9511, 9517, 11004, 11009, 11
011, 11012, 11014, 11015, 11016, 11017, 11022, 11023, 11024, 11026, 11027, 11029, 110
31, 11036, 11041, 11045, 11046, 11048, 11049, 11051, 11052, 11057, 11061, 11064, 1106
9, 11071, 11072, 11076, 11077, 11087, 11094, 11107, 11111, 11115, 11118, 11123, 1113
5, 11141, 11142, 11146, 11151, 11157, 11158, 11164, 11167, 11168, 11171, 11172, 1117
```

```
5, 11179, 11183, 11185, 11192, 11193, 11194, 11195, 11199, 11200, 11202, 11205, 1120
8, 11210, 11211, 11212, 11214, 11215, 11218, 11220, 11221, 11226, 11227, 11231, 1123
2, 11233, 11235, 11236, 11241, 11248, 11250, 11251, 11255, 11258, 11261, 11262, 1126
4, 11272, 11273, 11274, 11275, 11276, 11277, 11284, 11285, 11289, 11290, 11291, 1129
4, 11298, 11300, 11310, 11311, 11312, 11313, 11319, 11321, 11322, 11323, 11330, 1133
2, 11334, 11337, 11340, 11343, 11344, 11346, 11348, 11349, 11351, 11354, 11362, 1136
7, 11371, 11375, 11378, 11379, 11380, 11384, 11387, 11388, 11389, 11392, 11395, 1139
7, 11399, 13010, 13012, 13017, 13027, 13033, 13045, 13056, 13071, 13076, 13077, 1308
6, 13091, 13093, 13096, 13097, 13100, 13102, 13108, 13116, 13123, 13124, 13134, 1313
9, 13141, 13151, 13154, 13161, 13173, 13175, 13176, 13178, 13181, 13183, 13185, 1319
4, 13213, 13221, 13238]
```

In [127…

```
dm.columns
```

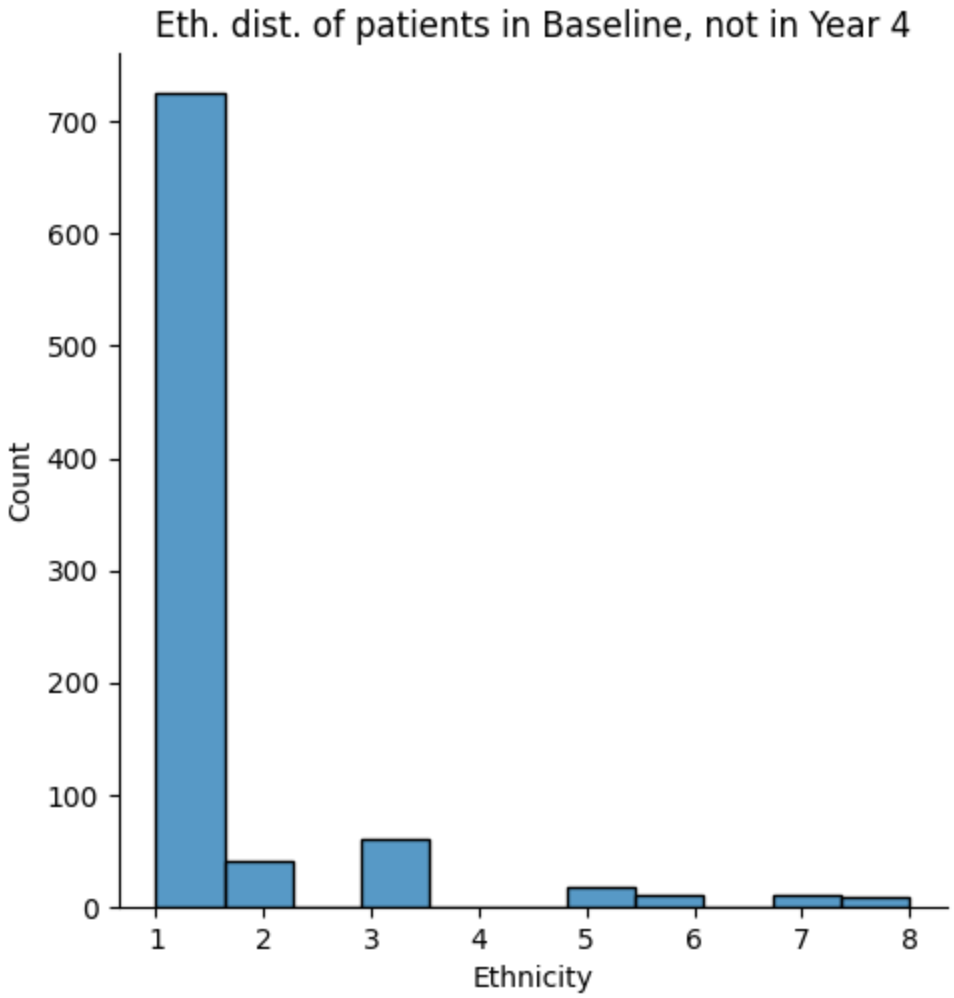
Out[127]:

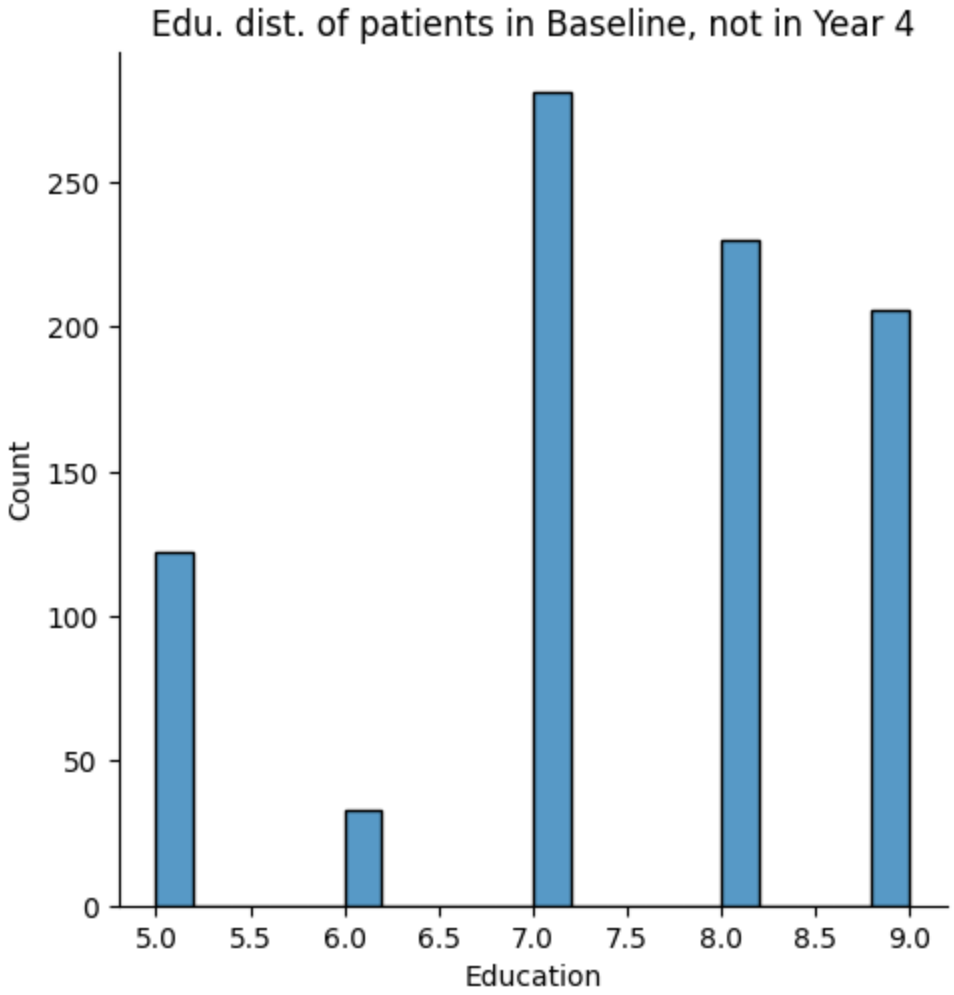
```
Index(['ID', 'age at dx', 'age at rand', 'Ethnicity', 'Education',
       'Marital Status', 'Empl. Status', 'Yr Rand'],
      dtype='object')
```

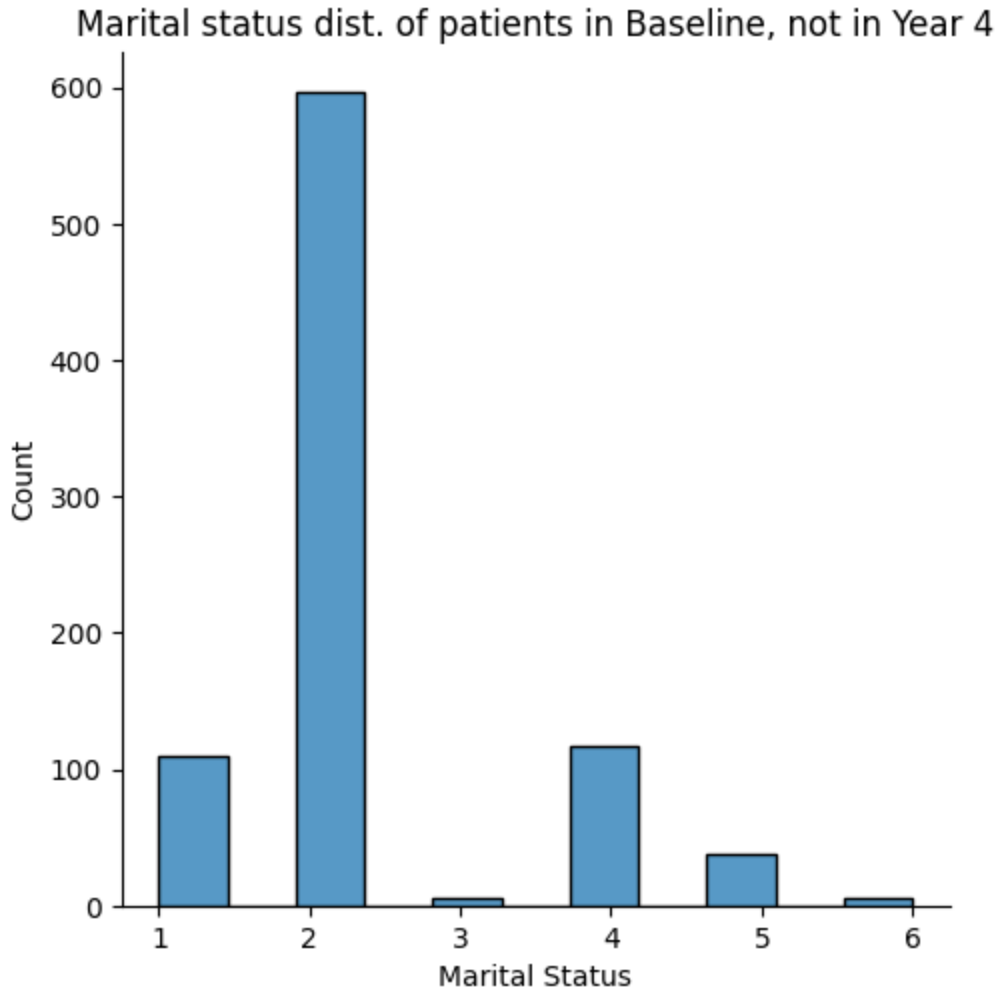
In [128…

```
# Plotting patterns of patients in Baseline but not in Year 4 (tf)

# Descriptions
      # Ethnicity
      #                 1 White, not Hispanic
      #                 2 Black not Hispanic
      #                 3 Hispanic
      #                 4 American Indian/Alaskan
      #                 5 Asian
      #                 6 Pacific Islander
      #                 7 Mixed Race
      #                 8 Other

      # Education
      #                 5 High School Graduate or less
      #                 6 Post High School Training
      #                 7 Some College Education
      #                 8 College/University Graduate
      #                 9 Post College/University Education

      # Marital Status
      #     1 Single
      #     2 Married
      #     3 Separated
      #     4 Divorced
      #     5 Widowed
      #     6 Unknown


ovt_id = list(ovt['ID'])

query = dm[dm['ID'].isin(loss_ids)]

sns.displot(data=query, x='Ethnicity')
plt.title('Eth. dist. of patients in Baseline, not in Year 4')

sns.displot(data=query, x='Education')
plt.title('Edu. dist. of patients in Baseline, not in Year 4')

sns.displot(data=query, x='Marital Status')
plt.title('Marital status dist. of patients in Baseline, not in Year 4')
```

Out[128]: Text(0.5, 1.0, 'Marital status dist. of patients in Baseline, not in Year 4')




Edu. dist. of patients in Baseline, not in Year 4
Count
0
50
100
150
200
250
5.0
5.5
6.0
6.5
7.0
7.5
8.0
8.5
9.0
Education

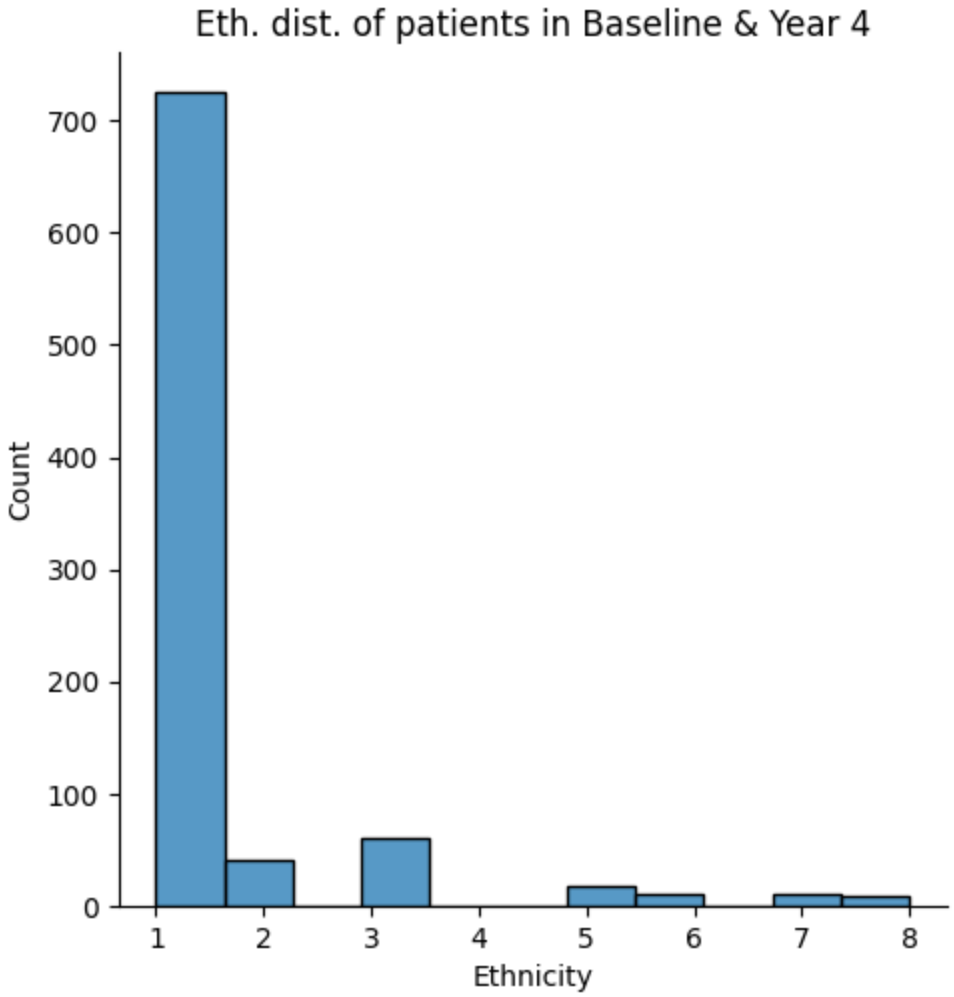


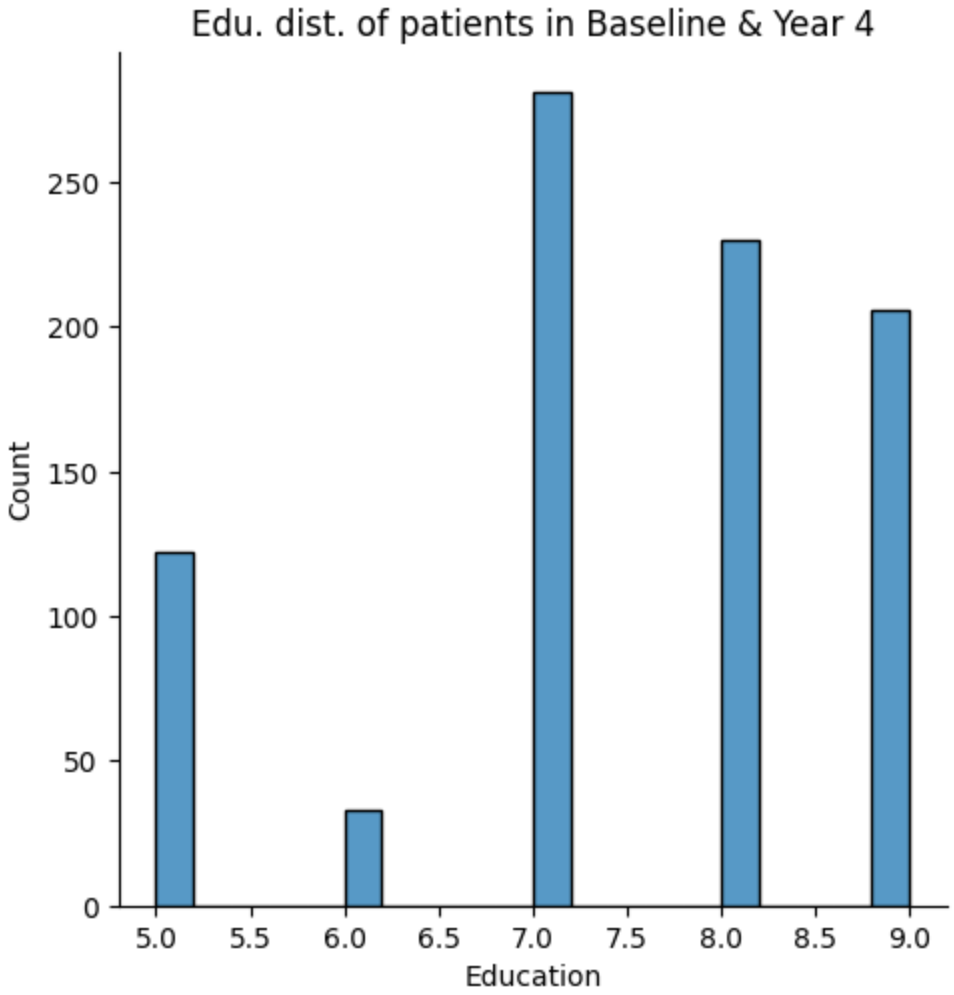
In [129…

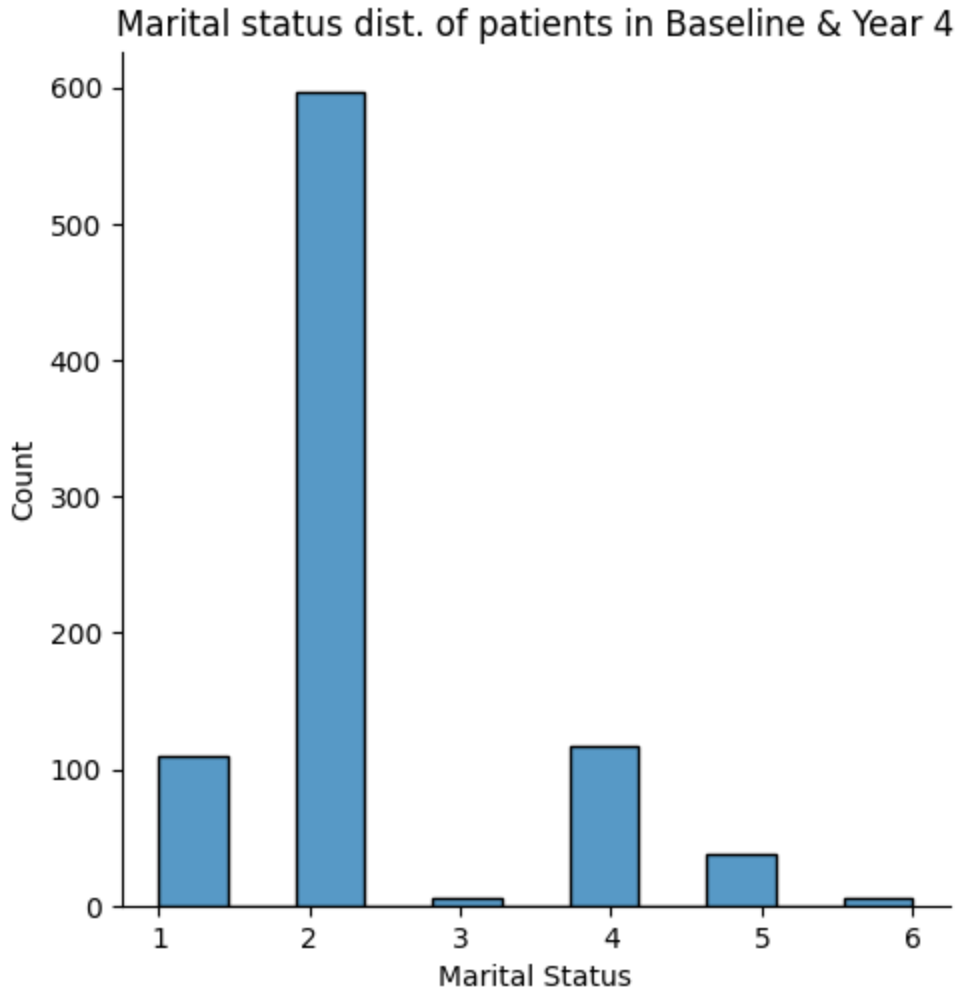
```
# Plotting patterns of patients in both Baseline and Year 4 (tf)

query_returned = dm[(dm['ID'].isin(ovt_id)) & (~dm['ID'].isin(loss_ids))]

sns.displot(data=query, x='Ethnicity')
plt.title('Eth. dist. of patients in Baseline & Year 4')

sns.displot(data=query, x='Education')
plt.title('Edu. dist. of patients in Baseline & Year 4')

sns.displot(data=query, x='Marital Status')
plt.title('Marital status dist. of patients in Baseline & Year 4')
```

Out[129]:

```
Text(0.5, 1.0, 'Marital status dist. of patients in Baseline & Year 4')
```


Eth. dist. of patients in Baseline & Year 4
Count
0
100
200
300
400
500
600
700
1
2
3
4
5
6
7
8
Ethnicity


Edu. dist. of patients in Baseline & Year 4
Count
0
50
100
150
200
250
5.0
5.5
6.0
6.5
7.0
7.5
8.0
8.5
9.0
Education

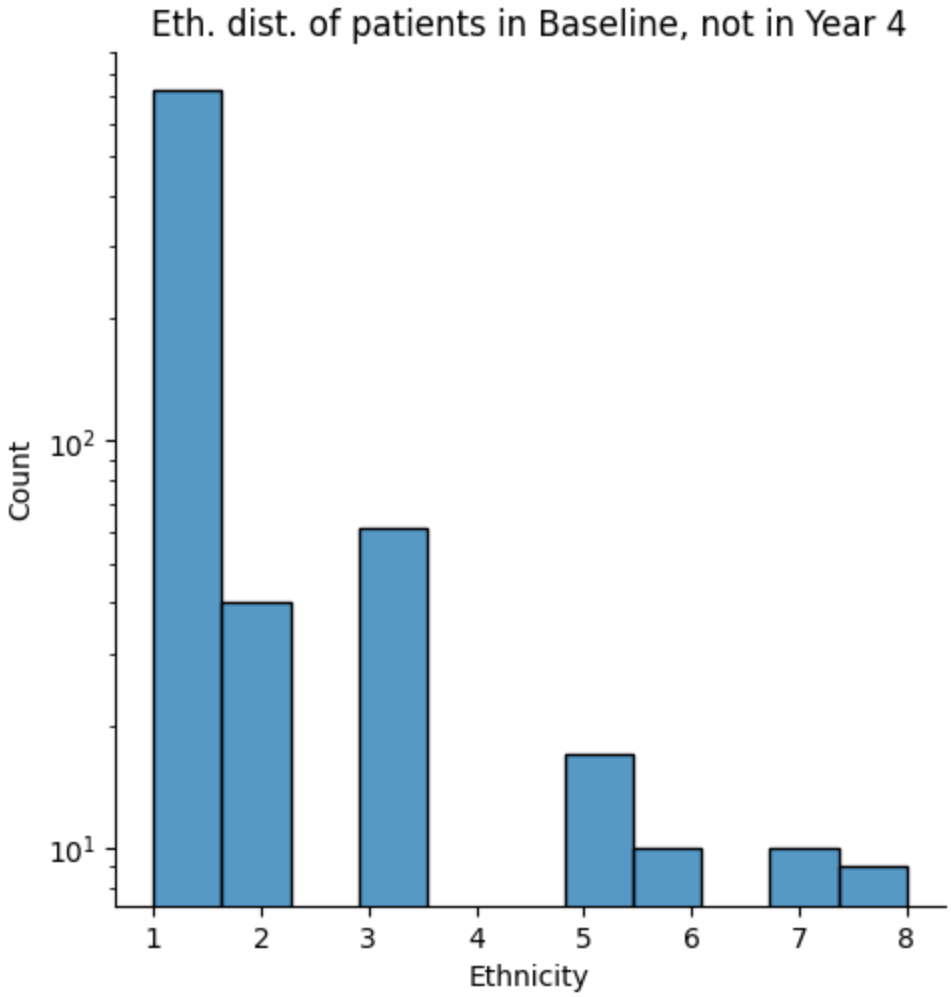


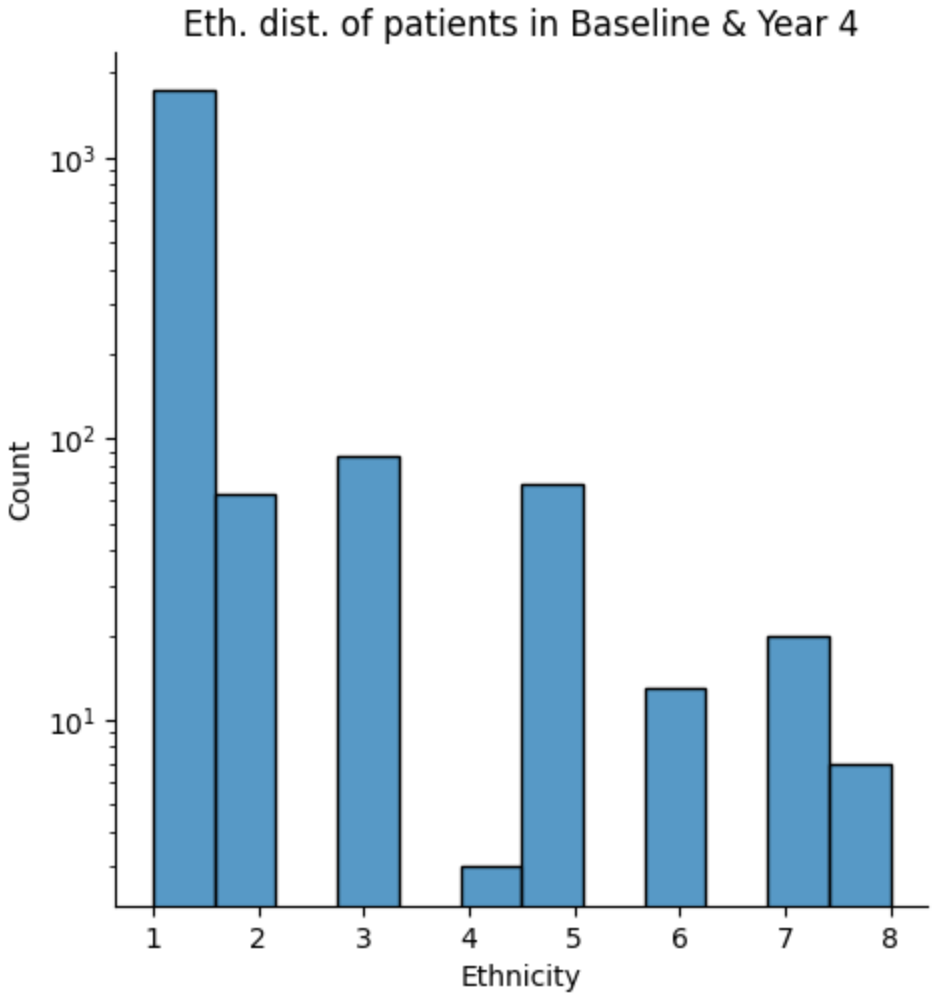
In [130…

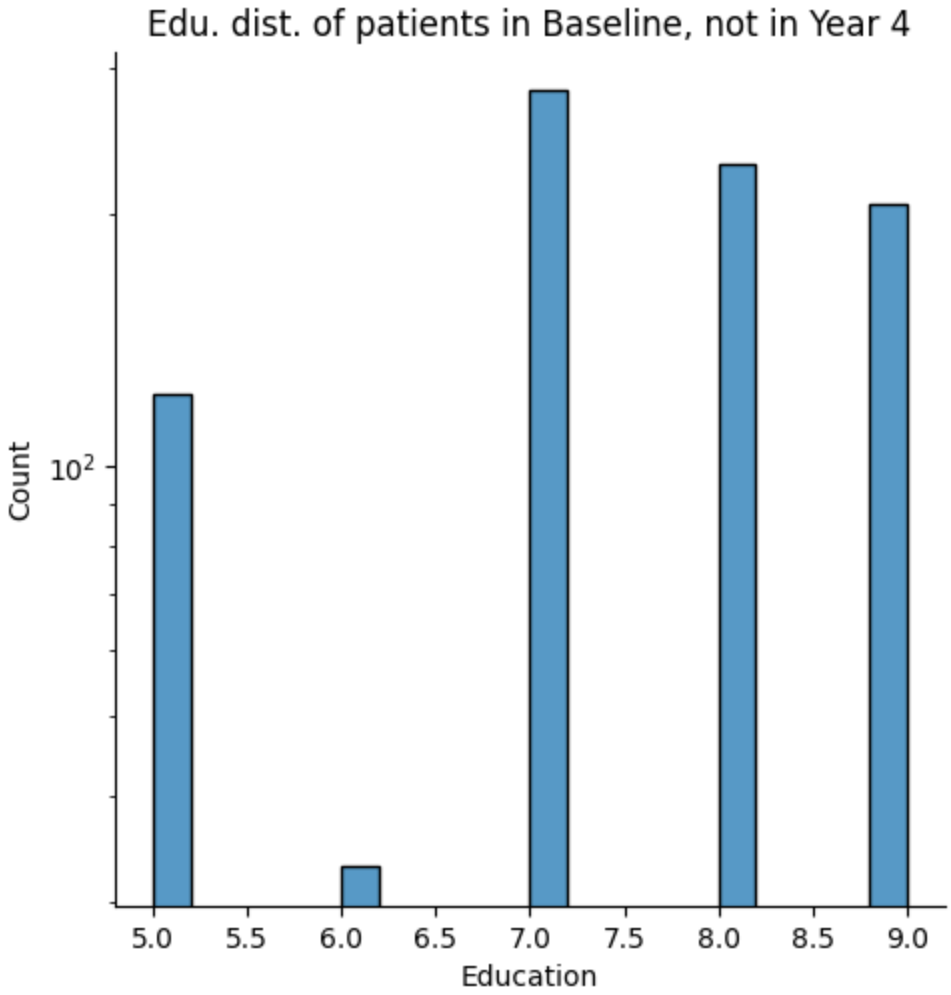
```
# Plot the above in log scale

sns.displot(data=query, x='Ethnicity')
plt.yscale("log")
plt.title('Eth. dist. of patients in Baseline, not in Year 4')

sns.displot(data=query_returned, x='Ethnicity')
plt.yscale('log')
plt.title('Eth. dist. of patients in Baseline & Year 4')

sns.displot(data=query, x='Education')
plt.yscale("log")
plt.title('Edu. dist. of patients in Baseline, not in Year 4')

sns.displot(data=query_returned, x='Education')
plt.yscale('log')
plt.title('Edu. dist. of patients in Baseline & Year 4')

sns.displot(data=query, x='Marital Status')
plt.yscale("log")
plt.title('Marital status dist. of patients in Baseline, not in Year 4')

sns.displot(data=query_returned, x='Marital Status')
plt.yscale('log')
plt.title('Marital status dist. of patients in Baseline & Year 4')
```

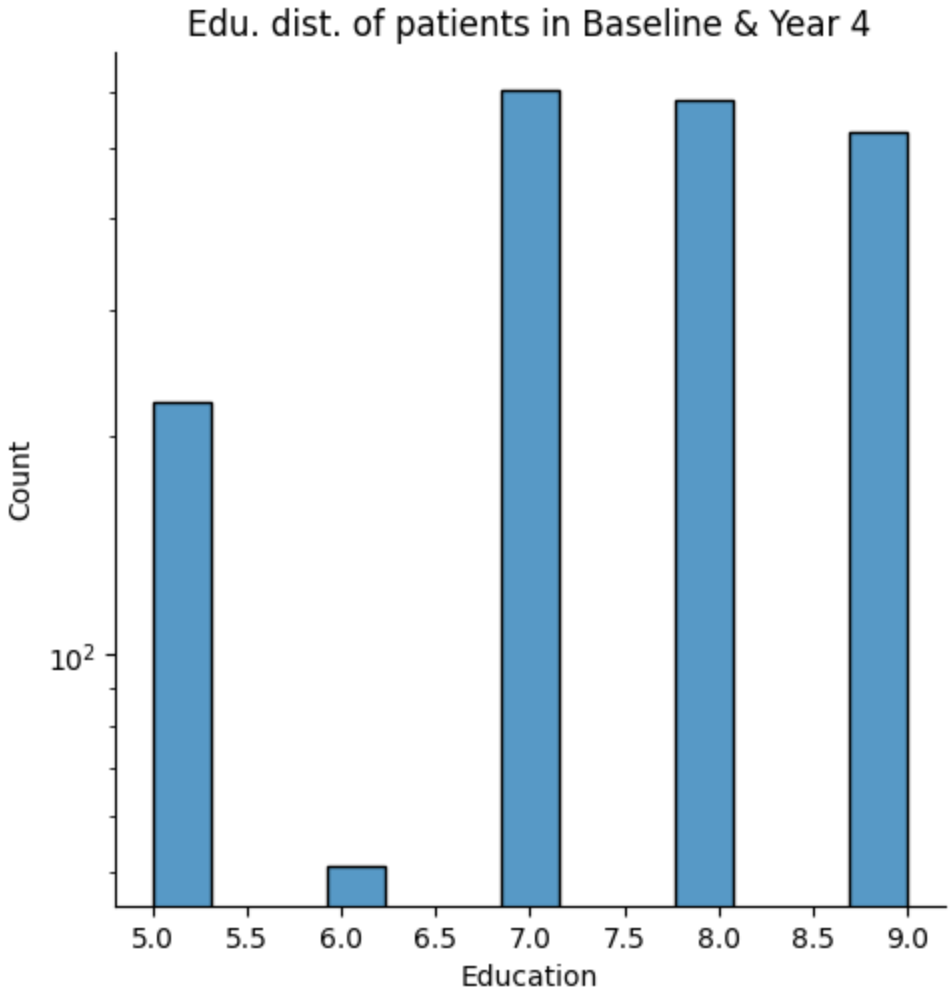
Out[130]:

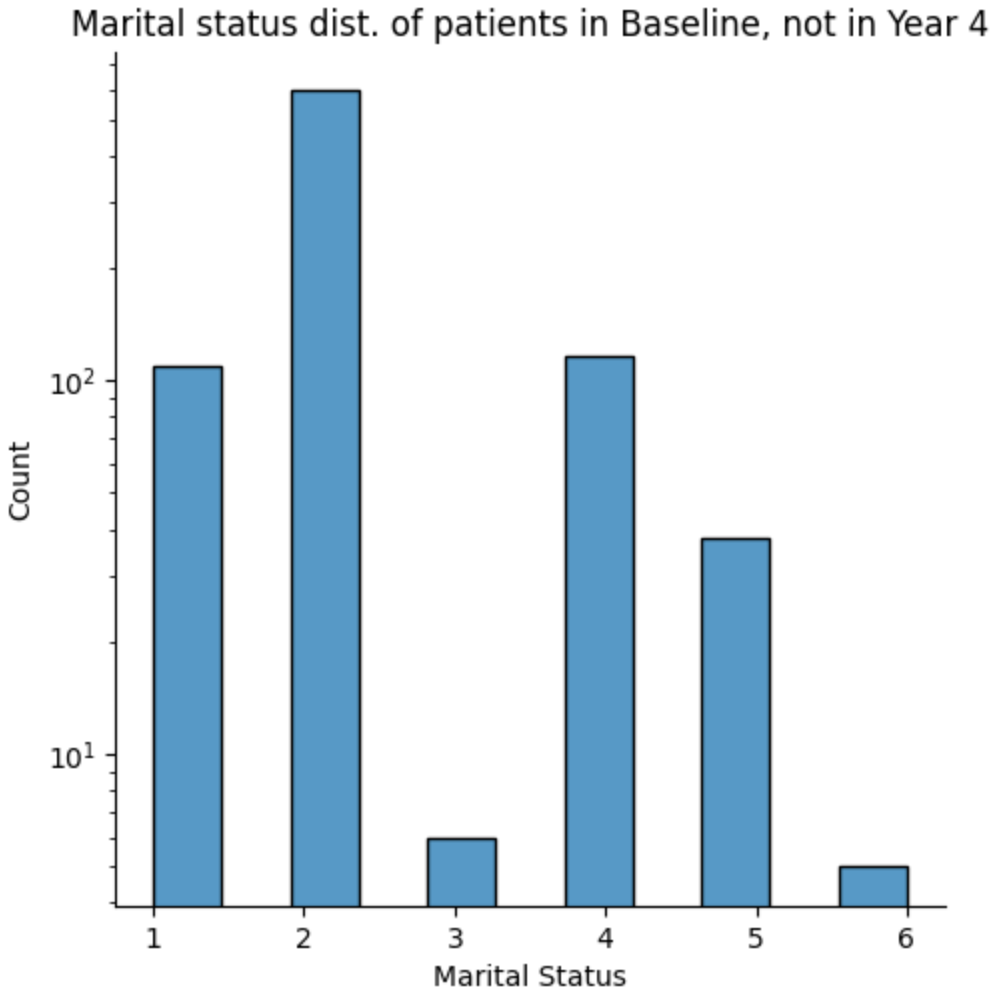
```
Text(0.5, 1.0, 'Marital status dist. of patients in Baseline & Year 4')
```

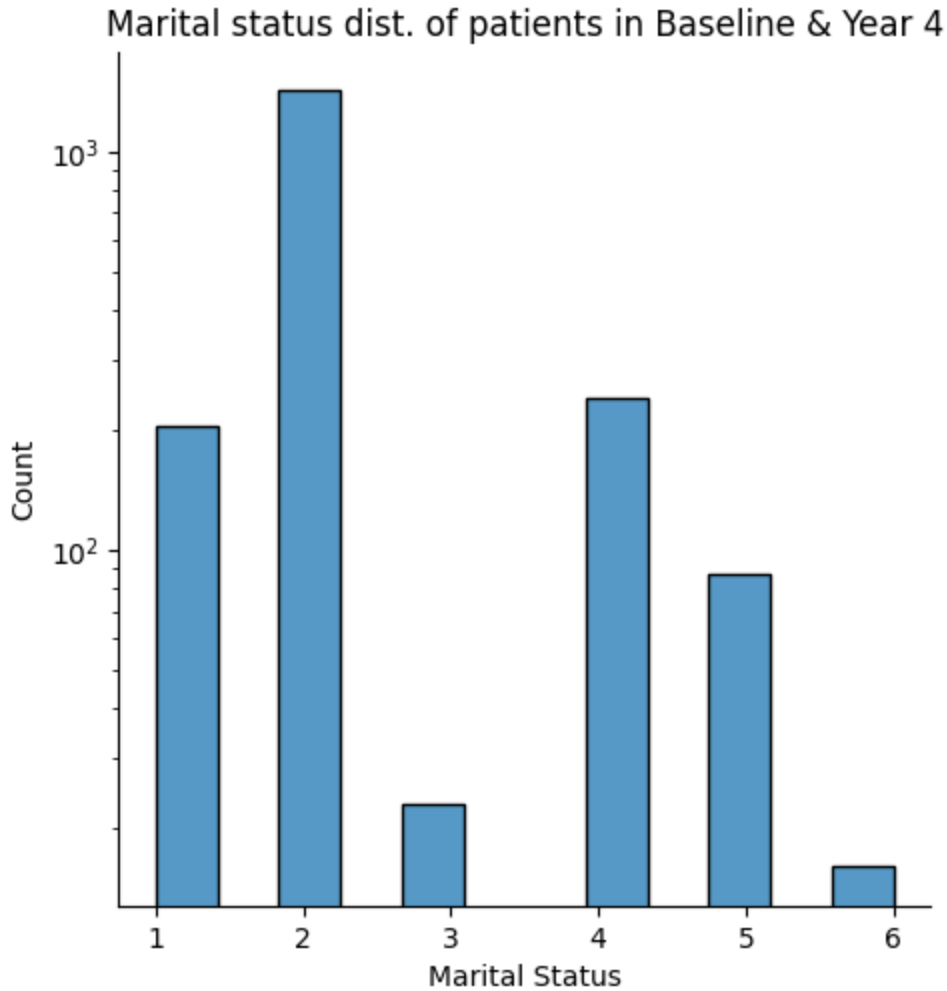

Eth. dist. of patients in Baseline, not in Year 4
Count
10^2
10^1
1
2
3
4
5
6
7
8
Ethnicity


Eth. dist. of patients in Baseline & Year 4
Count
10^3
10^2
10^1
1
2
3
4
5
6
7
8
Ethnicity


Edu. dist. of patients in Baseline, not in Year 4
Count
10^2
5.0
5.5
6.0
6.5
7.0
7.5
8.0
8.5
9.0
Education


Edu. dist. of patients in Baseline & Year 4
Count
10^2
5.0
5.5
6.0
6.5
7.0
7.5
8.0
8.5
9.0
Education

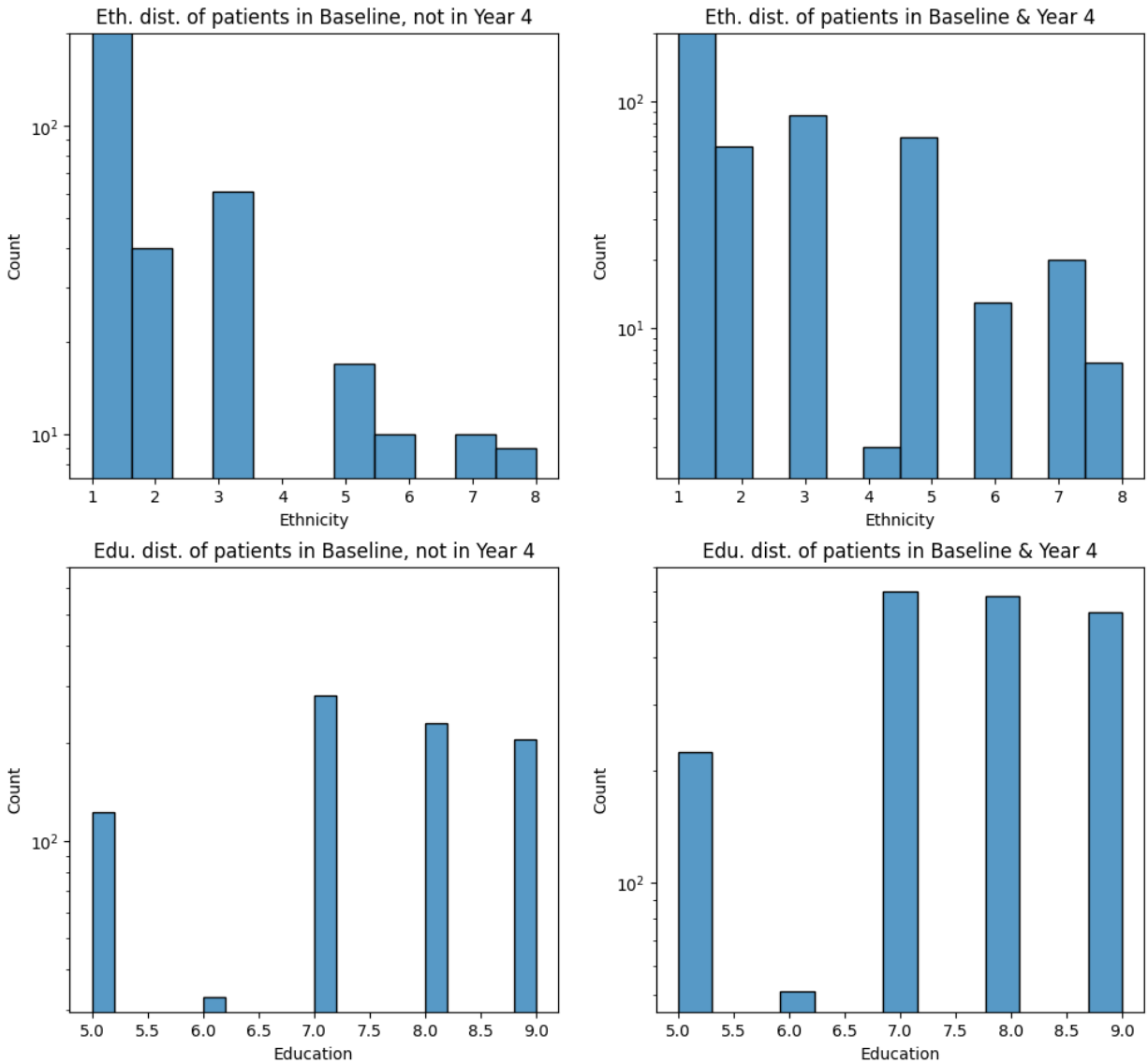

Marital status dist. of patients in Baseline, not in Year 4
Count
10^2
10^1
1
2
3
4
5
6
Marital Status

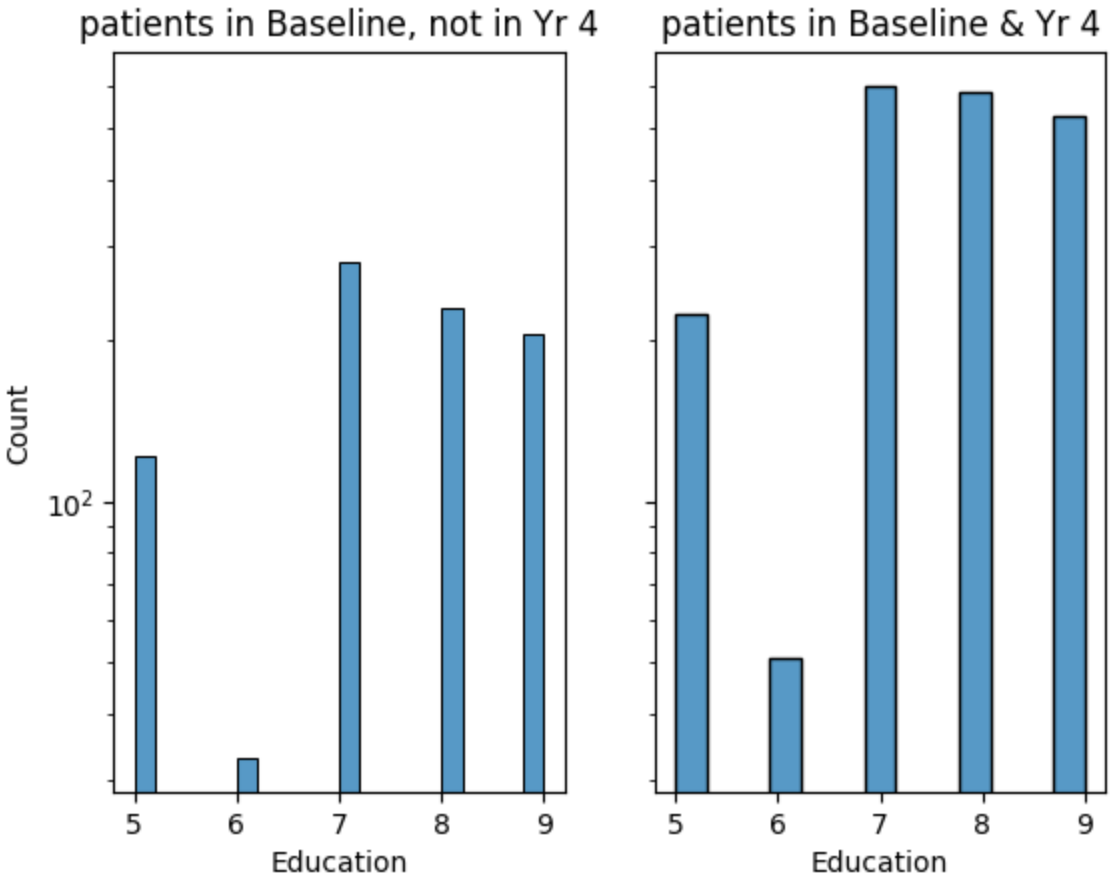


In [131…

```
plt.figure(figsize=(12, 11))

plt.subplot(2, 2, 1)
sns.histplot(x=query['Ethnicity'])
plt.yscale("log")
plt.ylim(0, 2*(10**2))
plt.title('Eth. dist. of patients in Baseline, not in Year 4')

plt.subplot(2, 2, 2)
sns.histplot(x=query_returned['Ethnicity'])
plt.yscale('log')
plt.ylim(0, 2*(10**2))
plt.title('Eth. dist. of patients in Baseline & Year 4')

plt.subplot(2, 2, 3)
sns.histplot(data=query, x='Education')
plt.yscale("log")
plt.ylim(0, 7*(10**2))
plt.title('Edu. dist. of patients in Baseline, not in Year 4')

plt.subplot(2, 2, 4)
sns.histplot(data=query_returned, x='Education')
plt.yscale('log')
plt.ylim(0, 7*(10**2))
plt.title('Edu. dist. of patients in Baseline & Year 4')

fig, ax = plt.subplots(1,2, sharey=True)
```

```
sns.histplot(x="Education", data=query, ax=ax[0]).set(title='patients in Baseline, not
sns.histplot(x="Education", data=query_returned, ax=ax[1]).set(title='patients in Base
plt.yscale('log')
fig.show()
```





In [156…
```
query.head()
```

Out[156]:

	ID	age at dx	age at rand	Ethnicity	Education	Marital Status	Empl. Status	Yr Rand
1	1003	51	53	1	8	2	1	1995
7	1011	42	42	5	7	2	1	1995
10	1017	45	45	1	9	2	1	1995
23	1034	61	62	1	5	4	0	1995
28	1039	42	43	1	7	1	1	1995

In [158…
```
query_returned.head()
```

Out[158]:

	ID	age at dx	age at rand	Ethnicity	Education	Marital Status	Empl. Status	Yr Rand
11	1018	56	58	1	5	2	1	1995
12	1019	67	71	1	7	4	0	1995
14	1022	50	53	1	7	4	1	1995
16	1026	51	52	1	7	2	1	1995
17	1027	64	67	1	7	2	0	1995

In [132…
```
query.shape
```

Out[132]: (872, 8)

In [159…
```
query_returned.shape
```

Out[159]: (1989, 8)

In [164…
```
# Exporting query & query_returned to perform hypothesis testing

path_non_returners = '/content/drive/MyDrive/NIEMA_Lab/WHEL_Data/Preprocessed-Data/nor
path_returners = '/content/drive/MyDrive/NIEMA_Lab/WHEL_Data/Preprocessed-Data/returne

with open(path_non_returners, 'w') as f:
  query.to_csv(f)

with open(path_returners, 'w') as f:
  query_returned.to_csv(f)
```

In [133…
```
# sanity check

for id in loss_ids:
  if id in list(tf_y4['ID']):
    print(id)
```

```
1236
3361
3398
3400
5276
9401
```

In [134…
```
tf[tf['ID'] == 1236].loc[212]
```

Out[134]:

```
ID               1236.000000
ss_tot             25.000000
ss_emot            11.000000
ss_aff              4.000000
ss_tang             4.000000
ss_pos              6.000000
lv_alone            1.000000
petownr             1.000000
church              2.000000
religion            3.000000
club                3.000000
caregvB             0.000000
strain              5.000000
optimism           17.000000
nee                 2.250000
aee                 1.670000
hostil              2.000000
QOL                90.187500
physhealth         89.375000
physfctn           95.000000
lim_phys          100.000000
pain               87.500000
genhlth            75.000000
mental health      91.000000
lim_emot          100.000000
energy             80.000000
emotwell           84.000000
socfctn           100.000000
symptom             0.114000
bloat               0.000000
sweats              0.000000
aches               1.000000
hotflash            0.000000
tired               1.000000
totlfevt            3.000000
dep_log             0.045957
whi_irs_old         7.000000
WHI_irs_new         2.000000
Name: 212, dtype: float64
```

In [135…

```
tf_y4[tf_y4['ID'] == 1236].loc[122]
```

Out[135]:
```
ID              1236.000000
ss_tot            32.000000
ss_emot           13.000000
ss_aff             5.000000
ss_tang            7.000000
ss_pos             7.000000
lv_alone           1.000000
petownr            1.000000
church             2.000000
religion           3.000000
club               3.000000
caregvB            1.000000
strain            11.000000
optimism                NaN
NEE                2.000000
AEE                2.000000
hostil             2.000000
QOL               95.250000
QOL_physical      96.250000
physfctn          95.000000
lim_phys         100.000000
pain             100.000000
genhlth           90.000000
QOL_mental        94.250000
lim_emot         100.000000
energy            85.000000
emotwell          92.000000
socfctn          100.000000
symptom            0.181793
bloating           0.000000
nightsweats        1.000000
aches              1.000000
hotflash           0.000000
tired              1.000000
totlfevt           7.000000
dep_log            0.001087
whi_irs_old        7.000000
whi_irs_new        2.000000
Name: 122, dtype: float64
```

In [136…
```
tf_y4[tf_y4['ID'] == 1236]['optimism'].values[0] - tf[tf['ID'] == 1236]['optimism'].va
```

Out[136]:
```
nan
```

In [137…
```
tf_y4['ID']
```

Out[137]:
```
0        1002
1        1005
2        1007
3        1010
4        1012
        ...
2129    13241
2130    13242
2131    13243
2132    13244
2133    13245
Name: ID, Length: 2134, dtype: int64
```

In [138…
```
tf.shape[0]
```

Out[138]:
```
2861
```

In [139…
```
tf_y4.shape[0]
```

Out[139]:
```
2134
```

In [140…
```
baseline_ids = tf['ID']
y4_ids = tf_y4['ID']

not_in_baseline = []
for id in list(y4_ids):
  if id not in list(baseline_ids):
    not_in_baseline.append(id)

not_in_y4 = []
for id in list(baseline_ids):
  if id not in list(y4_ids):
    not_in_y4.append(id)

print(not_in_baseline)
print(len(not_in_baseline))
print()
print(not_in_y4)
print(len(not_in_y4))
```

```
[1002, 1005, 1007, 1010, 1012, 1044, 1101, 1123, 1127, 1179, 1180, 1217, 1249, 1253,
1295, 1323, 1327, 1328, 1332, 1344, 1352, 1399, 1414, 1437, 1501, 1522, 1530, 3158, 3
206, 3241, 3275, 3276, 3336, 3431, 5016, 5060, 5066, 5069, 5075, 5082, 5092, 5096, 51
23, 5132, 5139, 5142, 5151, 5152, 5153, 5182, 5189, 5199, 5245, 5278, 5322, 5464, 549
4, 5500, 5568, 7009, 7013, 7022, 7029, 7056, 7058, 7062, 7116, 7128, 7175, 7195, 719
8, 7227, 7262, 7270, 7281, 7298, 7312, 7320, 7333, 7359, 7447, 7480, 9009, 9034, 903
9, 9040, 9042, 9046, 9060, 9098, 9104, 9105, 9110, 9122, 9123, 9132, 9137, 9155, 916
1, 9178, 9186, 9188, 9212, 9222, 9234, 9236, 9250, 9261, 9264, 9306, 9312, 9319, 933
2, 9337, 9341, 9377, 9378, 9379, 9419, 9449, 9478, 9516, 9525, 9527, 11054, 11059, 11
063, 11089, 11095, 11112, 11136, 11140, 11166, 11206, 11303, 11304, 11305, 11307, 113
08]
139

[1003, 1011, 1017, 1034, 1039, 1041, 1045, 1046, 1052, 1054, 1056, 1057, 1058, 1061,
1065, 1067, 1070, 1071, 1074, 1076, 1077, 1080, 1083, 1088, 1092, 1098, 1099, 1102, 1
103, 1108, 1110, 1113, 1119, 1125, 1126, 1129, 1130, 1131, 1133, 1135, 1136, 1137, 11
38, 1139, 1144, 1145, 1149, 1151, 1155, 1158, 1159, 1160, 1162, 1164, 1165, 1173, 117
4, 1177, 1181, 1184, 1190, 1193, 1194, 1198, 1201, 1202, 1206, 1209, 1213, 1216, 122
0, 1225, 1231, 1232, 1233, 1235, 1237, 1238, 1239, 1241, 1244, 1246, 1247, 1255, 126
0, 1263, 1264, 1267, 1268, 1269, 1272, 1275, 1280, 1283, 1284, 1291, 1293, 1294, 129
9, 1306, 1312, 1313, 1314, 1316, 1317, 1319, 1324, 1330, 1341, 1343, 1345, 1346, 134
8, 1349, 1351, 1353, 1354, 1355, 1360, 1361, 1363, 1364, 1368, 1373, 1377, 1378, 138
0, 1381, 1383, 1384, 1385, 1389, 1391, 1397, 1402, 1404, 1406, 1410, 1412, 1413, 141
5, 1421, 1422, 1423, 1425, 1430, 1438, 1441, 1443, 1444, 1450, 1453, 1454, 1457, 145
8, 1461, 1464, 1467, 1468, 1469, 1472, 1474, 1476, 1477, 1482, 1484, 1485, 1491, 149
6, 1499, 1504, 1506, 1507, 1508, 1510, 1513, 1514, 1516, 1523, 1524, 1526, 1527, 152
9, 1531, 1532, 1534, 1535, 1540, 1542, 1544, 1549, 1550, 1552, 1554, 1555, 1557, 156
1, 1562, 3001, 3008, 3009, 3010, 3014, 3017, 3019, 3020, 3027, 3038, 3040, 3045, 305
1, 3052, 3070, 3081, 3087, 3088, 3089, 3090, 3099, 3105, 3112, 3114, 3120, 3121, 312
6, 3128, 3129, 3132, 3133, 3135, 3136, 3148, 3149, 3151, 3154, 3155, 3160, 3173, 317
7, 3181, 3182, 3188, 3189, 3199, 3202, 3207, 3208, 3209, 3211, 3215, 3219, 3220, 322
1, 3222, 3228, 3230, 3231, 3234, 3242, 3244, 3248, 3260, 3273, 3274, 3279, 3283, 328
8, 3290, 3291, 3294, 3296, 3297, 3300, 3307, 3313, 3320, 3321, 3323, 3334, 3345, 335
0, 3356, 3364, 3367, 3374, 3378, 3384, 3389, 3392, 3395, 3397, 3399, 3403, 3406, 341
1, 3415, 3418, 3425, 3434, 3436, 3445, 3452, 5001, 5007, 5012, 5044, 5047, 5048, 505
1, 5054, 5068, 5074, 5076, 5085, 5088, 5093, 5095, 5103, 5108, 5112, 5115, 5116, 511
8, 5120, 5122, 5126, 5128, 5130, 5131, 5134, 5135, 5137, 5138, 5141, 5150, 5155, 515
6, 5158, 5165, 5168, 5171, 5174, 5175, 5176, 5177, 5193, 5205, 5209, 5210, 5211, 522
5, 5233, 5238, 5243, 5244, 5246, 5249, 5251, 5254, 5256, 5260, 5266, 5273, 5277, 528
2, 5283, 5286, 5287, 5289, 5290, 5292, 5296, 5298, 5300, 5301, 5302, 5304, 5305, 531
0, 5311, 5312, 5314, 5316, 5319, 5320, 5324, 5330, 5331, 5334, 5336, 5341, 5357, 535
8, 5361, 5362, 5363, 5364, 5365, 5366, 5369, 5378, 5379, 5387, 5390, 5391, 5392, 539
4, 5401, 5402, 5403, 5409, 5415, 5418, 5422, 5425, 5427, 5428, 5433, 5436, 5439, 544
0, 5441, 5444, 5446, 5448, 5457, 5459, 5465, 5471, 5487, 5490, 5492, 5499, 5506, 551
5, 5524, 5532, 5537, 5538, 5541, 5548, 5549, 5554, 5563, 5575, 7001, 7002, 7005, 701
7, 7019, 7023, 7031, 7033, 7034, 7035, 7038, 7043, 7044, 7048, 7051, 7053, 7059, 706
6, 7068, 7069, 7079, 7092, 7094, 7098, 7099, 7106, 7107, 7108, 7114, 7121, 7123, 712
4, 7125, 7127, 7130, 7134, 7135, 7138, 7144, 7148, 7150, 7152, 7158, 7160, 7161, 716
2, 7164, 7165, 7170, 7178, 7179, 7181, 7183, 7184, 7203, 7208, 7210, 7213, 7215, 722
0, 7237, 7239, 7242, 7243, 7245, 7246, 7250, 7274, 7278, 7280, 7283, 7287, 7291, 729
6, 7299, 7300, 7304, 7309, 7310, 7315, 7321, 7324, 7325, 7326, 7330, 7331, 7332, 733
6, 7344, 7345, 7346, 7348, 7349, 7352, 7356, 7358, 7364, 7365, 7366, 7369, 7378, 738
0, 7381, 7385, 7386, 7388, 7391, 7392, 7394, 7396, 7402, 7403, 7406, 7410, 7414, 742
1, 7423, 7432, 7433, 7435, 7439, 7441, 7448, 7449, 7454, 7456, 7458, 7460, 7463, 746
4, 7465, 7470, 7479, 7483, 7488, 7491, 7492, 7493, 7500, 7501, 7507, 7512, 7513, 751
9, 9001, 9002, 9004, 9014, 9015, 9019, 9021, 9023, 9031, 9043, 9052, 9054, 9055, 905
7, 9059, 9064, 9070, 9074, 9077, 9081, 9085, 9090, 9092, 9097, 9103, 9107, 9108, 910
9, 9112, 9116, 9121, 9138, 9141, 9145, 9146, 9150, 9154, 9157, 9159, 9160, 9162, 917
0, 9176, 9182, 9190, 9204, 9209, 9231, 9232, 9238, 9241, 9242, 9245, 9249, 9253, 925
5, 9258, 9269, 9273, 9276, 9281, 9283, 9289, 9291, 9295, 9298, 9299, 9327, 9334, 933
```

```
5, 9342, 9352, 9353, 9354, 9356, 9362, 9369, 9373, 9375, 9376, 9381, 9391, 9393, 940
0, 9413, 9415, 9425, 9431, 9439, 9446, 9450, 9452, 9461, 9464, 9465, 9471, 9479, 948
2, 9494, 9503, 9510, 9511, 9517, 11004, 11009, 11011, 11012, 11014, 11015, 11016, 110
17, 11022, 11023, 11024, 11026, 11027, 11029, 11031, 11036, 11041, 11045, 11046, 1104
8, 11049, 11051, 11052, 11057, 11061, 11064, 11069, 11071, 11072, 11076, 11077, 1108
7, 11094, 11107, 11111, 11115, 11118, 11123, 11135, 11141, 11142, 11146, 11151, 1115
7, 11158, 11164, 11167, 11168, 11171, 11172, 11175, 11179, 11183, 11185, 11192, 1119
3, 11194, 11195, 11199, 11200, 11202, 11205, 11208, 11210, 11211, 11212, 11214, 1121
5, 11218, 11220, 11221, 11226, 11227, 11231, 11232, 11233, 11235, 11236, 11241, 1124
8, 11250, 11251, 11255, 11258, 11261, 11262, 11264, 11272, 11273, 11274, 11275, 1127
6, 11277, 11284, 11285, 11289, 11290, 11291, 11294, 11298, 11300, 11310, 11311, 1131
2, 11313, 11319, 11321, 11322, 11323, 11330, 11332, 11334, 11337, 11340, 11343, 1134
4, 11346, 11348, 11349, 11351, 11354, 11362, 11367, 11371, 11375, 11378, 11379, 1138
0, 11384, 11387, 11388, 11389, 11392, 11395, 11397, 11399, 13010, 13012, 13017, 1302
7, 13033, 13045, 13056, 13071, 13076, 13077, 13086, 13091, 13093, 13096, 13097, 1310
0, 13102, 13108, 13116, 13123, 13124, 13134, 13139, 13141, 13151, 13154, 13161, 1317
3, 13175, 13176, 13178, 13181, 13183, 13185, 13194, 13213, 13221, 13238]
866
```

In [141…

```
1 in list(y4_ids)
```

Out[141]:

```
False
```

In [142…

```
datasets.keys()
```

Out[142]:

```
dict_keys(['Health Status', 'Demographics', 'Medical Measurements', 'Food from NDS',
'Supplements', 'Thoughts and Feelings', 'Personal Habits', 'Lifestyle', 'Family Histo
ry', 'Reproductive History', 'Food Frequency', 'Blood Carotenoids'])
```

In [143…

```
ep.rename(columns={"id": "ID"}, inplace=True)
bc_ovt.rename(columns={"id": "ID"}, inplace=True)
```

In [160…

```
ovt.shape
```

Out[160]:

```
(1989, 7)
```

In [144…

```
# Merging ovt with endpoints data

ovt_ep = ovt.merge(right=ep, how='inner', on='ID')

for col in ep.columns:
  if col not in ['ID', 'brcastatus']:
    ovt_ep.drop(columns=[col], inplace=True)
```

In [163…

```
ovt_ep
```

Out[163]:

	ID	QOL_ovt	energy_ovt	genhlth_ovt	optimism_ovt	pain_ovt	socfctn_ovt	brcastatus	br
0	1018	-15.750000	5.0	0.0	0.0	-37.5	-12.5	0	
1	1019	1.855469	15.0	-15.0	0.0	-12.5	0.0	0	
2	1022	-2.312500	0.0	-10.0	-2.0	-12.5	0.0	0	
3	1026	-7.769531	5.0	-5.0	2.0	0.0	-37.5	0	
4	1027	9.625000	20.0	0.0	-1.0	12.5	37.5	0	
...	...	...	...	...	...	...	...	...	
1984	13241	1.437500	5.0	5.0	-1.0	12.5	0.0	0	
1985	13242	2.187500	5.0	5.0	2.0	-12.5	0.0	0	
1986	13243	9.937500	15.0	10.0	1.0	12.5	0.0	0	
1987	13244	8.500000	0.0	-20.0	1.0	25.0	0.0	0	
1988	13245	16.937500	0.0	5.0	1.0	12.5	0.0	0	

1989 rows × 9 columns

In [145…

```
# Within ovt_ep, map brcastatus values to binary (0 stays 0, >0 becomes 1)

new = []
curr = ovt_ep['brcastatus']

for status in curr:
  if status == 0:
    new.append(status)
  else:
    new.append(1)

ovt_ep['brcastatus_binary'] = new
```

In [146…

```
# Merging ovt_ep with changes in blood carotenoids over time

ovrt = ovt_ep.merge(right=bc_ovt, how='inner', on='ID')
```

In [147…

```
ovrt
```

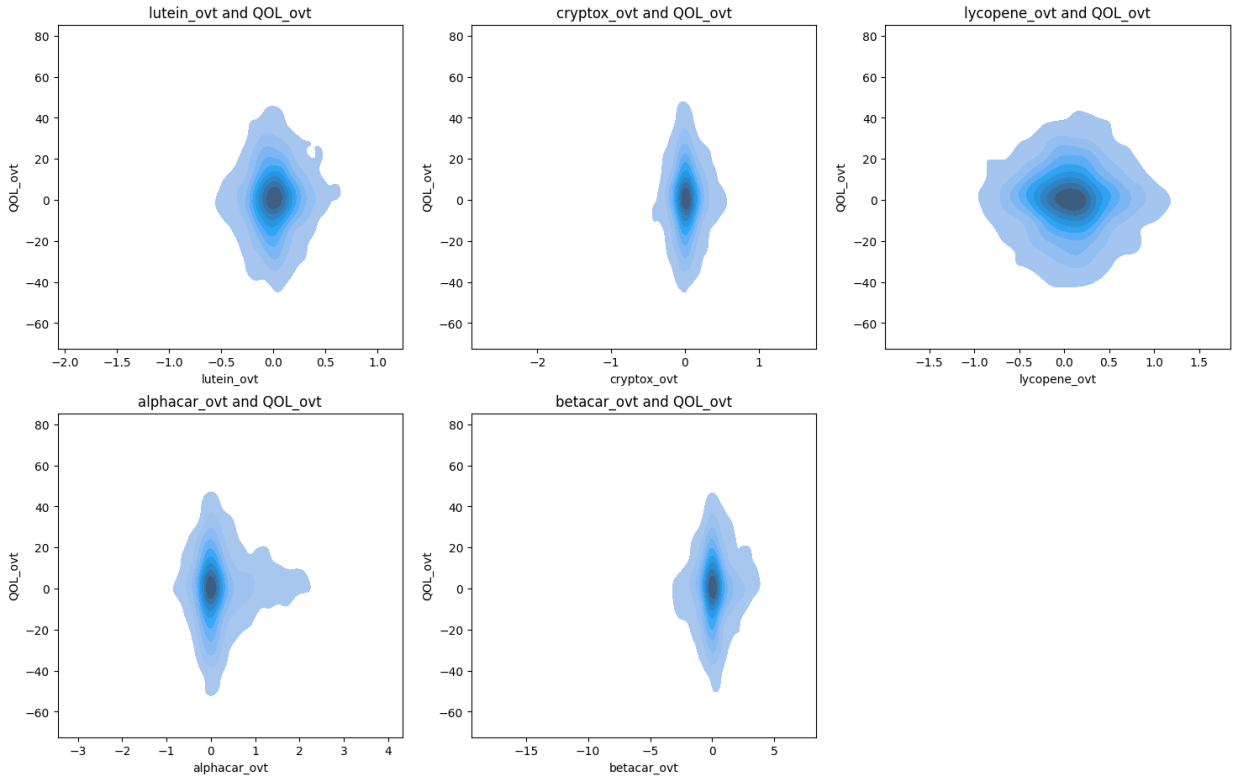
Out[147]:

	ID	QOL_ovt	energy_ovt	genhlth_ovt	optimism_ovt	pain_ovt	socfctn_ovt	brcastatus	b
0	1018	-15.750000	5.0	0.0	0.0	-37.5	-12.5	0	
1	1019	1.855469	15.0	-15.0	0.0	-12.5	0.0	0	
2	1022	-2.312500	0.0	-10.0	-2.0	-12.5	0.0	0	
3	1026	-7.769531	5.0	-5.0	2.0	0.0	-37.5	0	
4	1027	9.625000	20.0	0.0	-1.0	12.5	37.5	0	
...	...	...	...	...	...	...	...	...	
1839	13241	1.437500	5.0	5.0	-1.0	12.5	0.0	0	
1840	13242	2.187500	5.0	5.0	2.0	-12.5	0.0	0	
1841	13243	9.937500	15.0	10.0	1.0	12.5	0.0	0	
1842	13244	8.500000	0.0	-20.0	1.0	25.0	0.0	0	
1843	13245	16.937500	0.0	5.0	1.0	12.5	0.0	0	

1844 rows × 14 columns

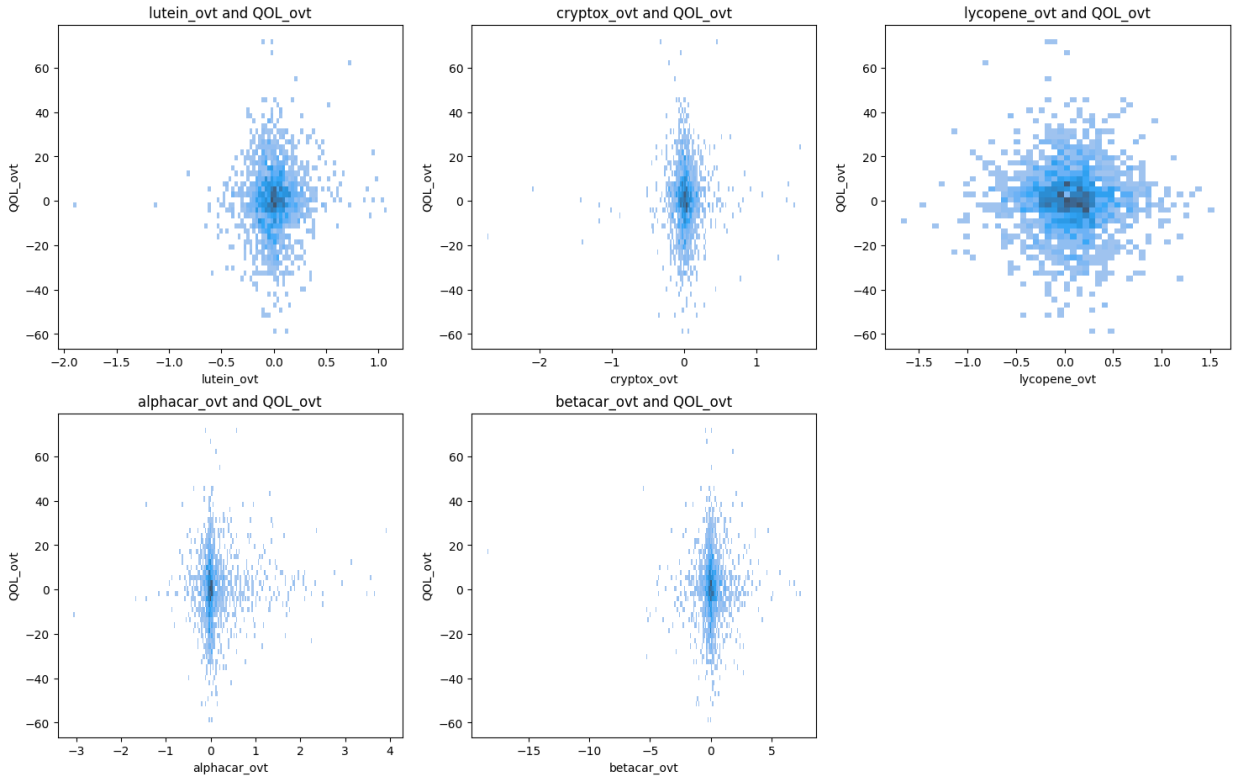
In [148…

```
# Plotting changes in blood carotenoids and changes in thoughts/feelings metrics

idx = 1
yaxis = 'QOL_ovt'
plt.figure(figsize=(18, 11))
for col in bc_ovt.columns:
  if col not in ['ID']:
    plt.subplot(2, 3, idx)
    sns.kdeplot(data=ovrt, x=col, y=yaxis, fill=True)
    plt.title(f'{col} and {yaxis}')
    idx += 1
```



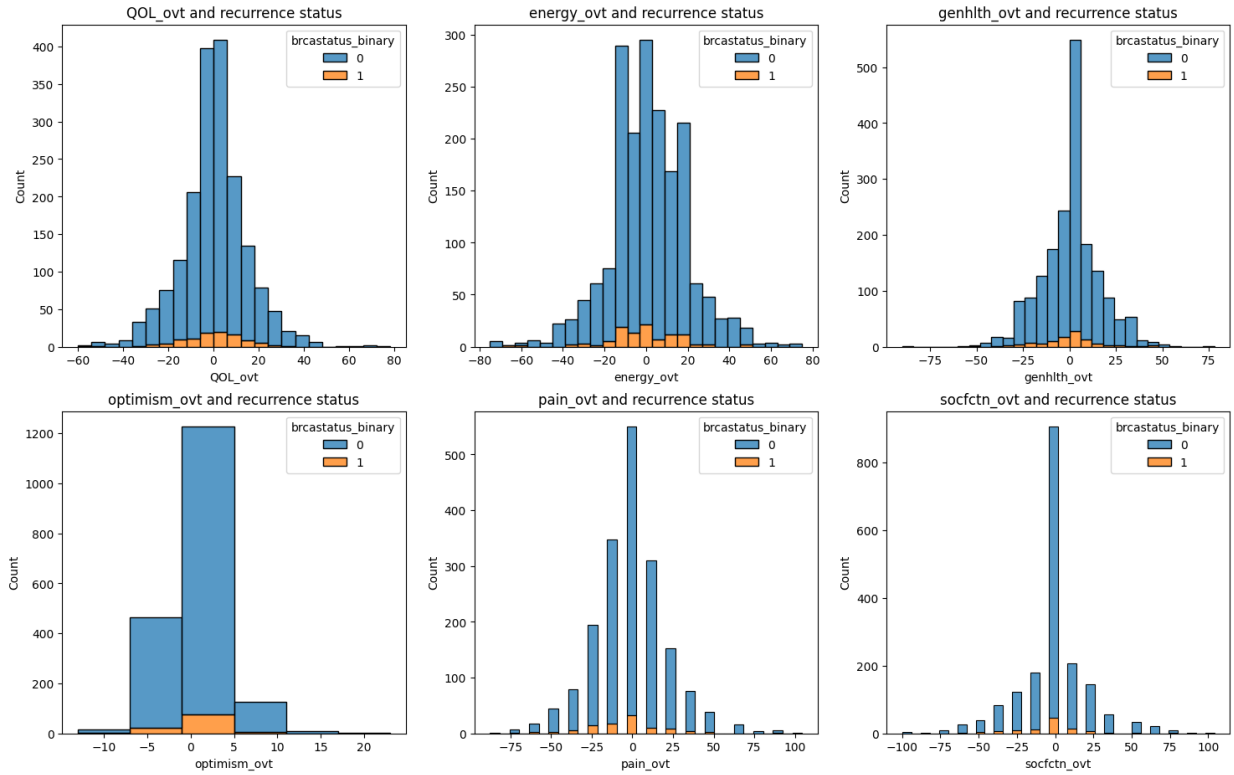
In [149…

```
# Plotting changes in blood carotenoids and changes in thoughts/feelings metrics

idx = 1
yaxis = 'QOL_ovt'
plt.figure(figsize=(18, 11))
for col in bc_ovt.columns:
  if col not in ['ID']:
    plt.subplot(2, 3, idx)
    sns.histplot(data=ovrt, x=col, y=yaxis, fill=True)
    plt.title(f'{col} and {yaxis}')
    idx += 1
```



In [150…

```
# Plotting changes in thoughts/feelings metrics over time and breast cancer status

columns = ['QOL_ovt', 'energy_ovt', 'genhlth_ovt', 'optimism_ovt', 'pain_ovt', 'socfct

idx = 1
plt.figure(figsize=(18, 11))
for col in columns:
  plt.subplot(2, 3, idx)
  sns.histplot(data=ovrt, x=col, hue='brcastatus_binary', stat='count', multiple='stac
  plt.title(f'{col} and recurrence status')
  idx += 1
```



In [150…

In [151…

```
# Exporting ovrt to work in "exploratory-analysis-2.ipynb"

path = '/content/drive/MyDrive/NIEMA_Lab/WHEL_Data/Preprocessed-Data/ovrt_phu.csv'

with open(path, 'w') as f:
  ovrt.to_csv(f)
```

In [152…

```
# sanity check

fd_y4[fd_y4['ID'] == 13243]['Lycopene'].values[0] - fd[fd['ID'] == 13243]['Lycopene'].
```

Out[152]:

```
20269.3076171875
```

In [152…

```
examine_loss()
```

In [153…

```
fd.columns
```

Out[153]:
```
Index(['ID', 'Recalls', 'last date', 'Kcal', 'Protein(g', 'Total Fat',
       'Total Carb', 'Alcohol', '% Protein', '% Fat', '% Carb', '% Alcohol',
       'Caffeine', 'Calcium', 'Iron', 'Magnesium', 'Phosphorus', 'Potassium',
       'Sodium', 'Zinc', 'Copper', 'Selenium', 'Total Vit A', 'Retinol',
       'Alpha-Toc Eq', 'alpha-Toc', 'Gamma-Toc', 'Vit D', 'Vitamin C',
       'Thiamin', 'Riboflavin', 'Niacin', 'Folacin', 'Panto. Acid', 'Vit B6',
       'Vit B12', 'Cholesterol', 'Total SFA', 'Total MFA', 'Total PFA',
       '% SFA', '%MFA', '% PFA', 'PFA 18:2', 'PFA 18:3', 'PFA 20:5',
       'PFA 22:6', 'Diet Fiber', 'Soluble Fiber', 'Insol. Fiber', 'Starch',
       'Sucrose', 'Galactose', 'Fructose', 'Lactose', 'Maltose', 'Water',
       'Anim. Protein', 'Veg Prot', 'Vit K', 'Trans Fat Acids', 'Beta-Carot',
       'Alpha-Carot', 'Beta-Cryptox', 'Lut + Zeax', 'Lycopene', 'Green Veg',
       'Iceber', 'Tomato', 'Orange Veg', 'Other Veg', 'Wh. Pota',
       'Veg_woJ,PI,Iceberg', 'VJ', 'Legume', 'Soy Legume', 'Citrus', 'Berry',
       'Melon', 'Other Fruit', 'FJ', 'Fruit_noJ', 'Whole Grain',
       'Refined Grain'],
      dtype='object')
```

Unused Code Archive

In [154…
```
# test = np.ones_like(ovrt['QOL_ovt']) / len(ovrt['QOL_ovt'])
# test
```

In [154…