Science files fraud detection

О Проекте

Тестовое задание для специалиста по Data Science.

Описание данных

Часть базы данных наблюдательного исследования (основные данные по визиту Скрининг) в формате Excel. В файле будет 2 вкладки: вкладка с описанием полей для сбора данных и вкладка с данными.

Задача

Проверить имеющиеся данные на их возможную фальсификацию.

Если фальсификация будет обнаружена, то вам необходимо указать:

- какие именно данные, возможно, были сфальсифицированы;
- отметить номера пациентов, для которых данные, возможно, были сфальсифицированы.

Выполнил

Алексей Сейкин

profile

Первоначальная стратегия

- Задача 1: Определение пациентов с фальсифицированными данными
  - Статистические методы:
    - Z-оценка: Подсчитать Z-оценку для каждого значения и идентифицировать значения, которые превышают порог (например, 3 или -3 для двухстороннего теста).
    - IQR (межквартильный размах): Вычислить IQR и определить значения за пределами 1.5*IQR от квартилей.
  - Визуализация:
    - Ящик с усами (Boxplot): Визуализировать данные для каждого признака и искать выбросы.
    - Scatter plot: Построить график зависимостей между различными признаками и поискать необычные точки.
  - Машинное обучение:
  - Изолирующий лес (Isolation Forest): Алгоритм, основанный на деревьях решений, который изолирует выбросы.
  - Метод опорных векторов для одного класса (One-Class SVM): Определяет границу нормальных данных и выявляет выбросы.
  - DBSCAN или OPTICS: Алгоритмы кластеризации, которые могут обнаруживать выбросы как шум в данных.
- Задача 2: Определение сфальсифицированных признаков
  - Feature Importance:
    - Использование алгоритмов машинного обучения, таких как Random Forest или Gradient Boosting, для определения важности признаков, которые могут указывать на фальсификацию.

- Анализ аномалий:
  - Использование алгоритмов детекции аномалий на уровне признаков для выявления необычных паттернов.
  - Ручная проверка:
  - Ручной аудит признаков с экспертами в соответствующей области для интерпретации подозрительных значений.
- Correlation Analysis:
  - Провести корреляционный анализ для определения необычных связей между признаками, которые могут указывать на манипулирование данными.
- Temporal Analysis:
  - Анализ изменений в данных со временем для выявления необычных паттернов, которые могут быть связаны с фальсификацией.

Шаг 1. Импорты

In [ ]:
```
import pandas as pd
import seaborn as sns
import numpy as np
import plotly.express as px
import scipy
import matplotlib.pyplot as plt
from scipy.stats import zscore
import warnings
from sklearn.cluster import DBSCAN
from sklearn.linear_model import LinearRegression
warnings.filterwarnings('ignore')

# кастомные функции
from utils import df_info # Функция для получения информации о датафрейме
from utils import low_information_features # Функция для определения неинформативных признаков
```

In [ ]:
```
# Загрузка данных
df = pd.read_excel('data/ProjectToOneFile_for_test.xlsx', sheet_name=1)
legend = pd.read_excel('data/ProjectToOneFile_for_test.xlsx', sheet_name=0)
```

In [ ]:
```
# Описание полей
legend
```

Out[ ]:

	Код поля	Описание поля	Всплывающая подсказка	Тип поля
0	SCR_ICF_SVDAT_ICF	Дата подписания ИС	Дата подписания ИС	DATE
1	SCR_ICF_SVDAT	Дата визита 1	Дата визита 1	DATE
2	SCR_ICF_BRTHDAT	Дата рождения	Дата рождения	DATE
3	SCR_ICF_AGE	Возраст, полных лет	Возраст, полных лет	TEXTBOX
4	SCR_ICF_VSORRES_HEIGHT	Рост, см	Рост, см	TEXTBOX
5	SCR_ICF_VSORRES_WEIGHT	Масса тела, кг	Масса тела, кг	TEXTBOX
6	SCR_ICF_VSORRES_BMI	Индекс массы тела, кг/м2	Индекс массы тела, кг/м2	TEXTBOX
7	SCR_LB_SPERMOGR_LBDAT_SPERM	Дата проведения исследования	Дата проведения исследования	DATE
8	SCR_LB_SPERMOGR_LBORRES_VOL_EJACUL	Объем эякулята, мл	Объем эякулята, мл	TEXTBOX
9	SCR_LB_SPERMOGR_LBORRES_TOTCONC_SP	Общая концентрация сперматозоидов\nв эякуляте,...	Общая концентрация сперматозоидов\nв эякуляте,...	TEXTBOX
10	SCR_LB_SPERMOGR_LBORRES_COCENTR_SP	Концентрация сперматозоидов, млн/м\nл\	Концентрация сперматозоидов, млн/м\nл	TEXTBOX
11	SCR_LB_SPERMOGR_LBORRES_TOTMOTILIT_SP	Общая подвижность сперматозоидов,\n%	Общая подвижность сперматозоидов,\n%	TEXTBOX
12	SCR_LB_SPERMOGR_LBORRES_PROGR_MOBIL	Группа1. Прогрессивно-подвижные, %	Группа1. Прогрессивно-подвижные (Быст\nрые про...	TEXTBOX
13	SCR_LB_SPERMOGR_LBORRES_PNONPROGR_MOBIL	Группа 2. Непрогрессивно-подвижные,\n%	Группа 2. Непрогрессивно-подвижные,\n%	TEXTBOX
14	SCR_LB_SPERMOGR_LBORRES_FIXED	Группа 3. Неподвижные, %	Группа 3. Неподвижные, %	TEXTBOX

Шаг 2. Исследовательский анализ данных

In [ ]:
```
df_info(df)
```

```
Количество записей:      602
Количество столбцов:     18
Явных дубликатов:        0
Пропуски присутствуют в 13 столбцах из 18:
```

	Пропущено %
SCR_ICF_VSORRES_HEIGHT	75.913621
SCR_ICF_VSORRES_WEIGHT	75.913621
SCR_ICF_VSORRES_BMI	75.913621
SCR_LB_SPERMOGR_LBORRES_VOL_EJACUL	15.614618
SCR_LB_SPERMOGR_LBORRES_TOTMOTILIT_SP	8.471761
SCR_LB_SPERMOGR_LBORRES_PROGR_MOBIL	8.139535
SCR_LB_SPERMOGR_LBORRES_PNONPROGR_MOBIL	8.139535
SCR_LB_SPERMOGR_LBORRES_FIXED	7.807309
SCR_LB_SPERMOGR_LBORRES_TOTCONC_SP	7.142857
SCR_LB_SPERMOGR_LBORRES_COCENTR_SP	6.976744
SCR_LB_SPERMOGR_LBDAT_SPERM	3.654485
SCR_ICF_BRTHDAT	0.498339
SCR_ICF_AGE	0.498339

```
Обобщенная информация:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 602 entries, 0 to 601
Columns: 18 entries, Screening # to SCR_LB_SPERMOGR_LBORRES_FIXED
dtypes: float64(11), int64(1), object(6)
memory usage: 84.8+ KB
None
Первые 3 строки:
```

	Screening #	Initials	Site #	SCR_ICF_SVDAT_ICF	SCR_ICF_SVDAT	SCR_ICF_BRTHDAT	SCR_ICF_AGE	SCR_ICF_VSORRES_HEIGHT	SCR_ICF_VSORRES_WEIGHT	SCR_ICF_VSORRES_BMI	SCR_LB_SPERMOGR_LBDAT_SPERM	SCR_LB_SPERMOGR
0	001-001	РКВ	1	2023-06-26	2023-06-26	1986-04-12	37.0	183.0	100.0	29.86	2023-06-21	
1	001-002	КАД	1	2023-06-24	2023-06-24	1993-09-09	29.0	184.0	91.0	26.88	2023-05-27	
2	001-003	АДР	1	2023-10-23	2023-10-23	1983-01-09	40.0	173.0	78.0	26.06	2023-10-23	

Анализ пропусков

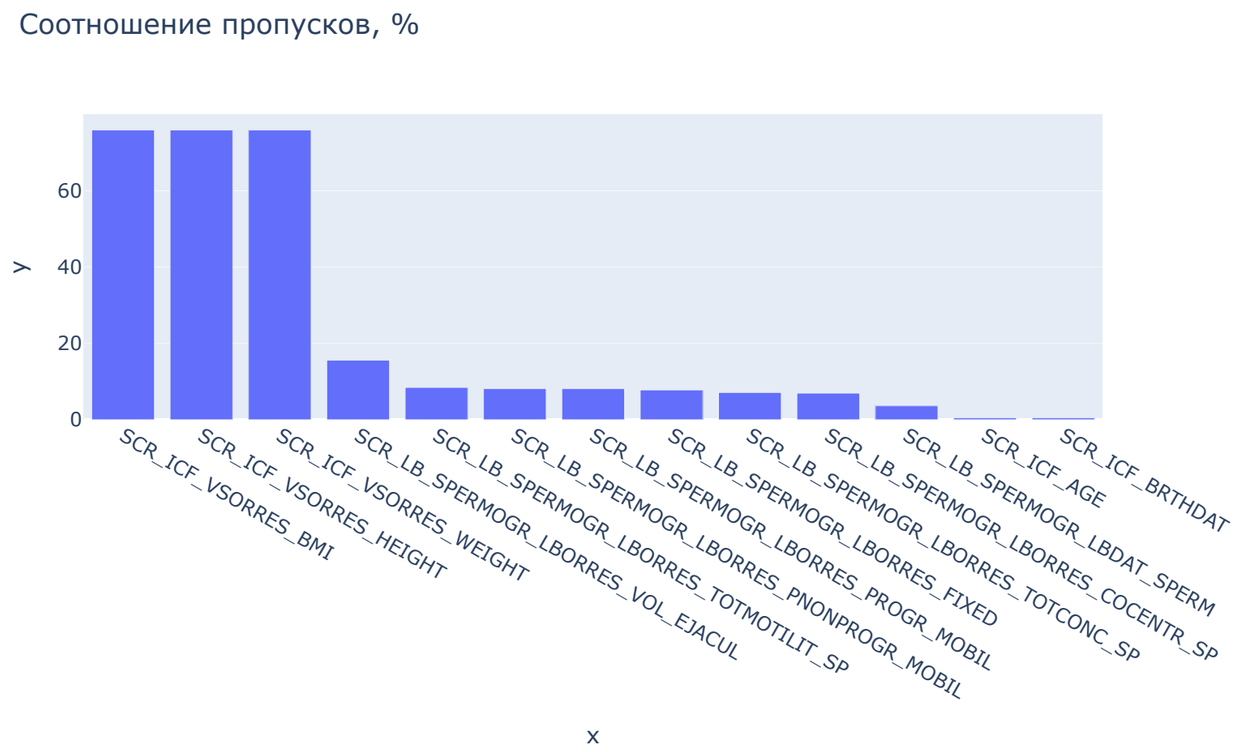
In [ ]:
```
# Общее количество пропусков в колонках
df.isna().sum()
```

Out[ ]:
```
Screening #                          0
Initials                             0
Site #                               0
SCR_ICF_SVDAT_ICF                    0
SCR_ICF_SVDAT                        0
SCR_ICF_BRTHDAT                      3
SCR_ICF_AGE                          3
SCR_ICF_VSORRES_HEIGHT             457
SCR_ICF_VSORRES_WEIGHT             457
SCR_ICF_VSORRES_BMI                457
SCR_LB_SPERMOGR_LBDAT_SPERM         22
SCR_LB_SPERMOGR_LBORRES_VOL_EJACUL  94
SCR_LB_SPERMOGR_LBORRES_TOTCONC_SP  43
SCR_LB_SPERMOGR_LBORRES_COCENTR_SP  42
SCR_LB_SPERMOGR_LBORRES_TOTMOTILIT_SP  51
SCR_LB_SPERMOGR_LBORRES_PROGR_MOBIL  49
SCR_LB_SPERMOGR_LBORRES_PNONPROGR_MOBIL  49
SCR_LB_SPERMOGR_LBORRES_FIXED       47
dtype: int64
```

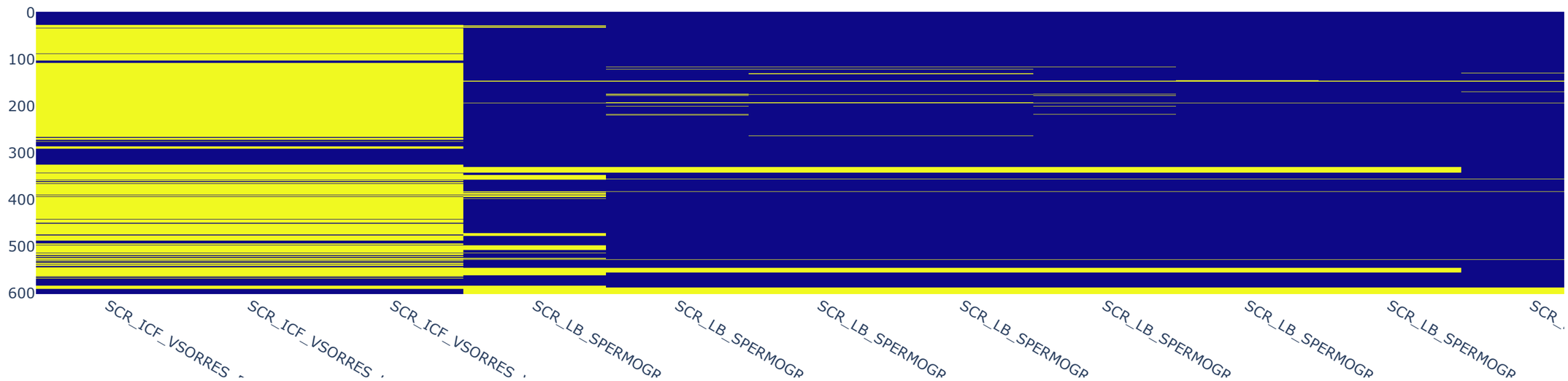
In [ ]:
```
# Количество наблюжений хотя бы с одним пропуском
df.isnull().any(axis=1).sum()
```

Out[ ]:
```
472
```

In [ ]:
```
# Соотношение пропусков, %
nans = 100 * df.isnull().mean().sort_values(ascending=False)[df.isnull().mean().sort_values(ascending=False) > 0]

fig = px.bar(x=nans.index, y=nans.values, title='Соотношение пропусков, %', height=500, width=800)
fig.show()
```



In [ ]:
```
fig = px.imshow(df[nans.index].isnull(), title='Тепловая карта пропусков')
fig.show()
```



Анализ типов данных

In [ ]:
```
df.dtypes
```

Out[ ]:
```
Screening #                             object
Initials                                object
Site #                                   int64
SCR_ICF_SVDAT_ICF                       object
SCR_ICF_SVDAT                           object
SCR_ICF_BRTHDAT                         object
SCR_ICF_AGE                            float64
SCR_ICF_VSORRES_HEIGHT                 float64
SCR_ICF_VSORRES_WEIGHT                 float64
SCR_ICF_VSORRES_BMI                    float64
SCR_LB_SPERMOGR_LBDAT_SPERM             object
SCR_LB_SPERMOGR_LBORRES_VOL_EJACUL     float64
SCR_LB_SPERMOGR_LBORRES_TOTCONC_SP     float64
SCR_LB_SPERMOGR_LBORRES_COCENTR_SP     float64
SCR_LB_SPERMOGR_LBORRES_TOTMOTILIT_SP  float64
SCR_LB_SPERMOGR_LBORRES_PROGR_MOBIL    float64
SCR_LB_SPERMOGR_LBORRES_PNONPROGR_MOBIL float64
SCR_LB_SPERMOGR_LBORRES_FIXED          float64
dtype: object
```

Анализ малоинформативных признаков

In [ ]:
```
# Получение неинформативных признаков
low_information_features(df)
```

```
Screening #: 100.0% уникальных значений
['Screening #']
```

Out[ ]:

Выводы по EDA

При первичном анализе датасета на предмет пропусков обнаружены признаки с количеством пропусков более 70%. Так-же присутствует признак screenening - похоже это индекс пациента. Так-же часть данных, содержащих даты - не приведенf а нужному типу.

Поскольку у некоторых пациентов пропущены все значения в результатах - нет смысла оставлять их в датасете.

Исправим эти моменты, сохранив общее число наблюдений

In [ ]:
```
df_cleaned = df.copy()
```

In [ ]:
```
# Удаление строк, где пропуски присутствуют во всех отобранных столбцах
spermo_columns = [col for col in df_cleaned.columns if 'SCR_LB_SPERMOGR_LBORRES' in col]
df_cleaned = df_cleaned.dropna(subset=spermo_columns, how='all')
```

In [ ]:
```
# Устанавливаем столбец 'Screening #' в качестве индекса DataFrame
df_cleaned.set_index('Screening #', inplace=True)

# Определяем столбцы, содержащие строки с датами в формате 'YYYY-MM-DD'
date_cols = []
for col in df_cleaned.columns:
    # Если хотя бы одно значение в столбце соответствует шаблону даты, добавляем его в список date_cols
    if df_cleaned[col].astype(str).str.match(r'^\d{4}-\d{2}-\d{2}$').any():
        date_cols.append(col)

# Конвертируем все обнаруженные столбцы с датами из строк в datetime
for col in date_cols:
    df_cleaned[col] = pd.to_datetime(df_cleaned[col])

# Удаляем столбцы, где более 70% значений пропущены
# Сначала рассчитываем порог: столбцы должны содержать минимум 30% непустых значений
threshold = len(df_cleaned) * 0.3
# Удаляем столбцы, не соответствующие порогу
df_cleaned.dropna(axis=1, thresh=threshold, inplace=True)

# Приводим возраст к int - целому числу
df_cleaned['SCR_ICF_AGE'] = df_cleaned['SCR_ICF_AGE'].astype(int, errors='ignore')
```

In [ ]:
```
df_info(df_cleaned)
```

```
Количество записей:      560
Количество столбцов:     14
Явных дубликатов:        0
Пропуски присутствуют в 9 столбцах из 14:
```

	Пропущено %
SCR_LB_SPERMOGR_LBORRES_VOL_EJACUL	9.285714
SCR_LB_SPERMOGR_LBORRES_TOTMOTILIT_SP	1.607143
SCR_LB_SPERMOGR_LBORRES_PROGR_MOBIL	1.250000
SCR_LB_SPERMOGR_LBORRES_PNONPROGR_MOBIL	1.250000
SCR_LB_SPERMOGR_LBORRES_FIXED	0.892857
SCR_ICF_BRTHDAT	0.535714
SCR_ICF_AGE	0.535714
SCR_LB_SPERMOGR_LBDAT_SPERM	0.357143
SCR_LB_SPERMOGR_LBORRES_TOTCONC_SP	0.178571

```
Обобщенная информация:
<class 'pandas.core.frame.DataFrame'>
Index: 560 entries, 001-001 to 071-004
Columns: 14 entries, Initials to SCR_LB_SPERMOGR_LBORRES_FIXED
dtypes: datetime64[ns](4), float64(8), int64(1), object(1)
memory usage: 65.6+ KB
None
Первые 3 строки:
```

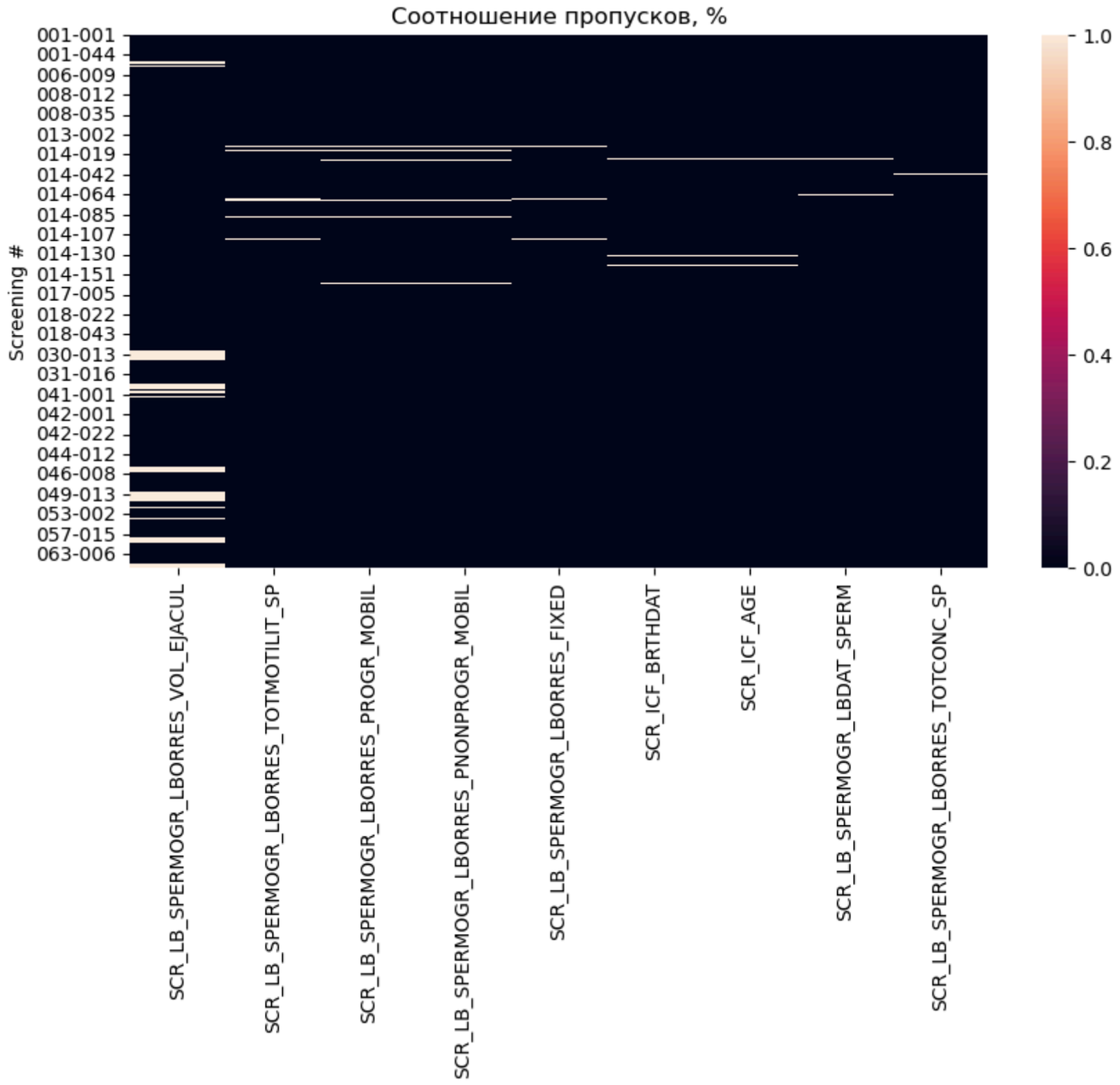
Screening #	Initials	Site #	SCR_ICF_SVDAT_ICF	SCR_ICF_SVDAT	SCR_ICF_BRTHDAT	SCR_ICF_AGE	SCR_LB_SPERMOGR_LBDAT_SPERM	SCR_LB_SPERMOGR_LBORRES_VOL_EJACUL	SCR_LB_SPERMOGR_LBORRES_TOTCONC_SP	SCR_LB_SPERM
001-001	РКВ	1	2023-06-26	2023-06-26	1986-04-12	37.0	2023-06-21	3.2	139.17	
001-002	КАД	1	2023-06-24	2023-06-24	1993-09-09	29.0	2023-05-27	0.4	15.20	
001-003	АДР	1	2023-10-23	2023-10-23	1983-01-09	40.0	2023-10-23	3.0	103.00	

In [ ]:
```
df_cleaned.dtypes
```

Out[ ]:
```
Initials                                object
Site #                                   int64
SCR_ICF_SVDAT_ICF                datetime64[ns]
SCR_ICF_SVDAT                    datetime64[ns]
SCR_ICF_BRTHDAT                  datetime64[ns]
SCR_ICF_AGE                            float64
SCR_LB_SPERMOGR_LBDAT_SPERM      datetime64[ns]
SCR_LB_SPERMOGR_LBORRES_VOL_EJACUL     float64
SCR_LB_SPERMOGR_LBORRES_TOTCONC_SP     float64
SCR_LB_SPERMOGR_LBORRES_COCENTR_SP     float64
SCR_LB_SPERMOGR_LBORRES_TOTMOTILIT_SP  float64
SCR_LB_SPERMOGR_LBORRES_PROGR_MOBIL    float64
SCR_LB_SPERMOGR_LBORRES_PNONPROGR_MOBIL float64
SCR_LB_SPERMOGR_LBORRES_FIXED          float64
dtype: object
```

In [ ]:
```
# Визуализация соотношений пропусков
nans = 100 * df_cleaned.isnull().mean().sort_values(ascending=False)[df_cleaned.isnull().mean().sort_values(ascending=False) > 0]
# Столбчатая диаграмма
nans.plot(
    kind='bar',
    logy=False,
    figsize=(10,5),
    title='Соотношение пропусков, %');
```

```
# Построение тепловой карты пропусков
sns.heatmap(df_cleaned[nans.index].isnull());
```



Шаг 3 Поиск аномалий

Задача 1: Определение пациентов с фальсифицированными данными

Проблемы с датами

In [ ]:
```python
# Пересчитаем возраст
df_cleaned['recalculated_age'] = (df_cleaned['SCR_ICF_SVDAT_ICF'] - df_cleaned['SCR_ICF_BRTHDAT']).dt.days / 365.25
```

In [ ]:
```python
# Выведем количество наблюдений с отклонением более 2 года

# Вычисляем абсолютное отклонение возраста относительно колонки 'SCR_ICF_AGE'
df_cleaned['age_deviation'] = (df_cleaned['recalculated_age'] - df_cleaned['SCR_ICF_AGE']).abs()

# Подсчитываем количество наблюдений с отклонением в 2 года и более
deviation_count = df_cleaned[df_cleaned['age_deviation'] >= 2].shape[0]

print(f"Количество наблюдений с отклонением в 2 года и более: {deviation_count}")
```

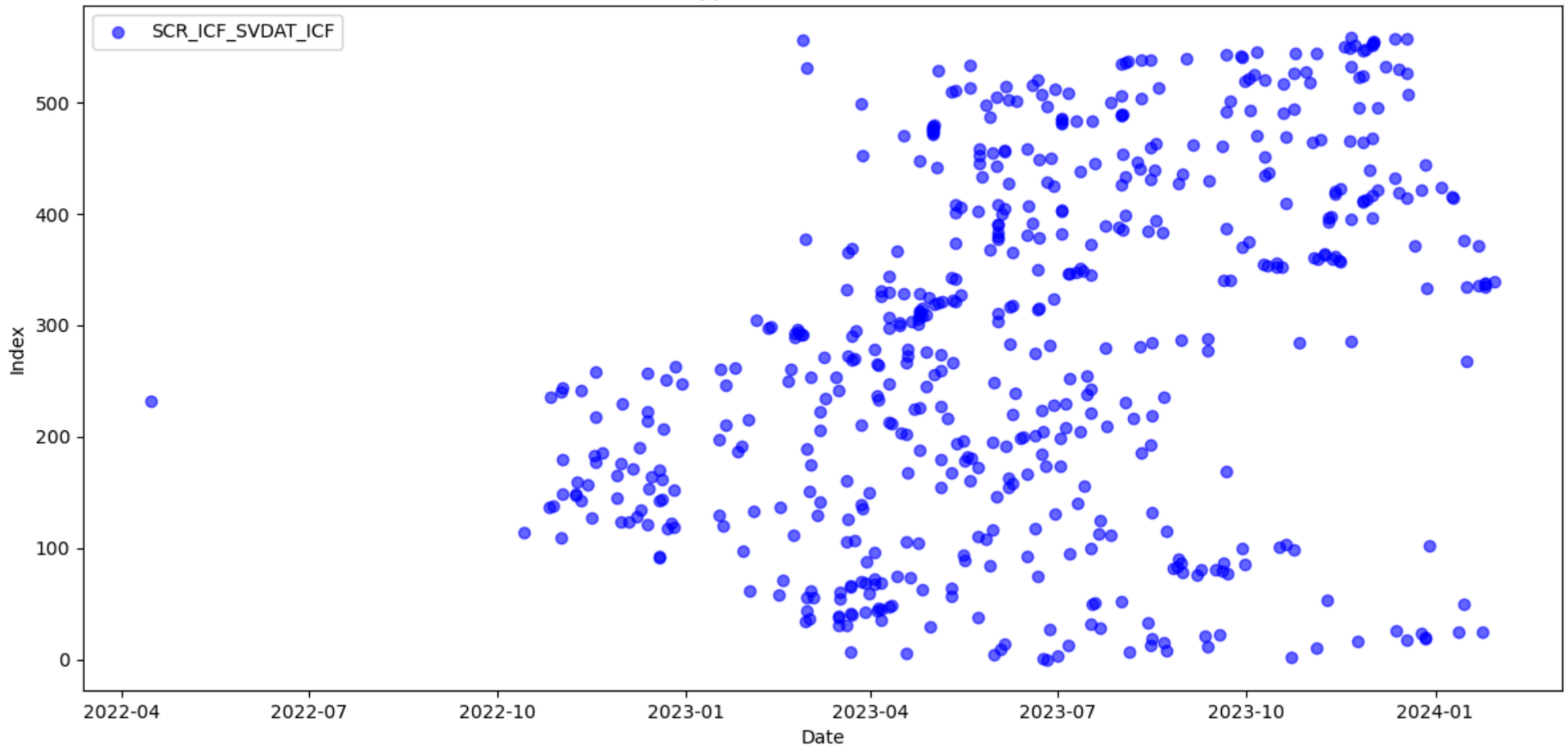
```
Количество наблюдений с отклонением в 2 года и более: 0
```

In [ ]:
```python
# Переводим колонки с датами в формат datetime
# df_cleaned['SCR_LB_SPERMOGR_LBDAT_SPERM'] = pd.to_datetime(df_cleaned['SCR_LB_SPERMOGR_LBDAT_SPERM'])
df_cleaned['SCR_ICF_SVDAT'] = pd.to_datetime(df_cleaned['SCR_ICF_SVDAT'])

# Фильтруем DataFrame, чтобы оставить только те наблюдения, где дата анализа спермограммы идет до даты визита
observations_before_visit = df_cleaned[df_cleaned['SCR_LB_SPERMOGR_LBDAT_SPERM'] < df_cleaned['SCR_ICF_SVDAT']]

# Подсчитываем количество таких наблюдений
count_observations_before_visit = observations_before_visit.shape[0]

print(f"Количество наблюдений, где анализ спермограммы был сделан до визита: {count_observations_before_visit}")
```

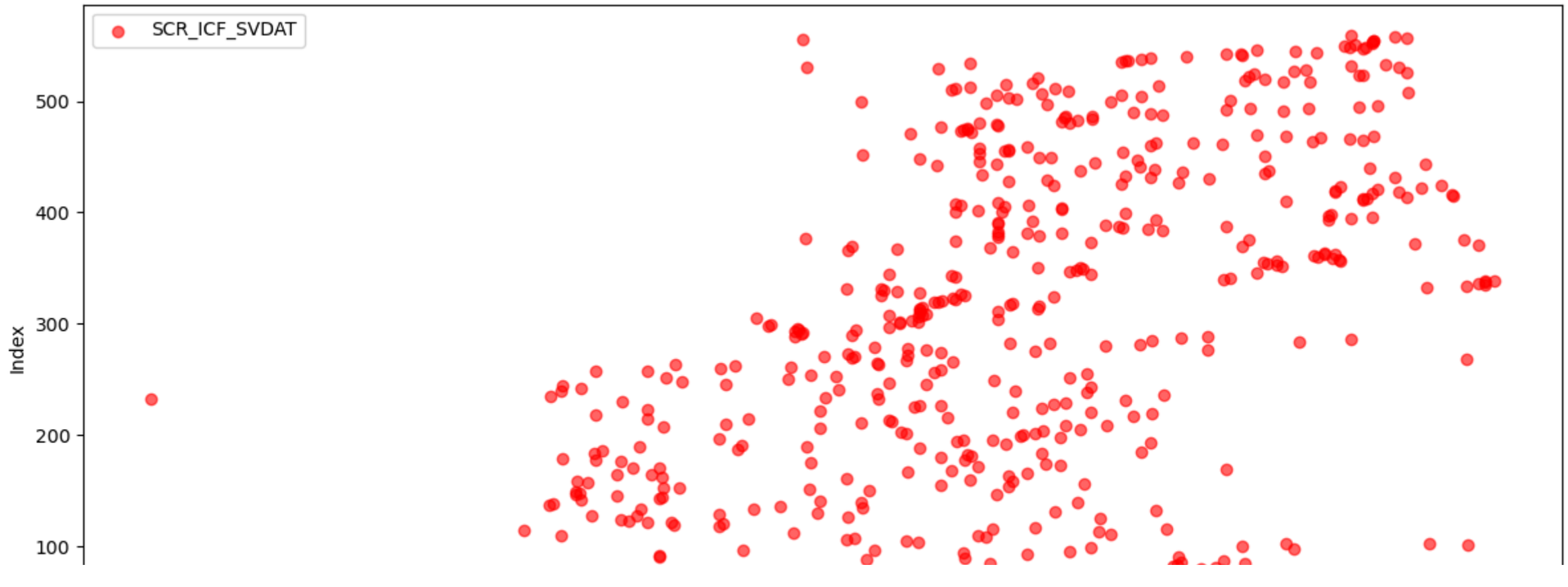
```
Количество наблюдений, где анализ спермограммы был сделан до визита: 101
```

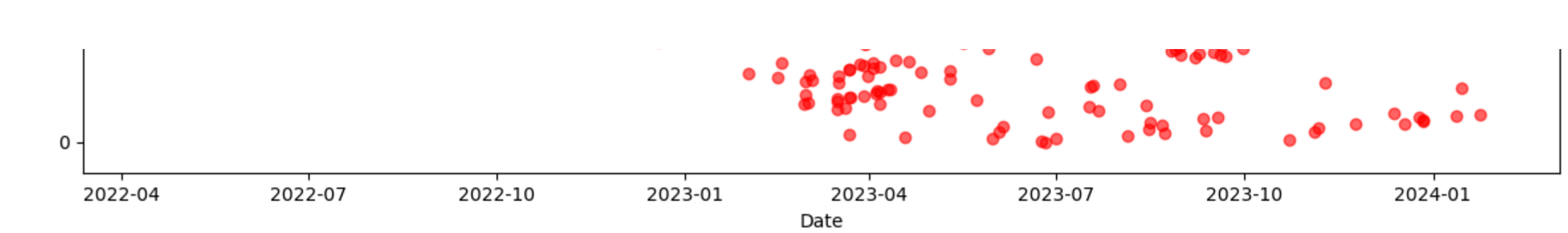
In [ ]:
```python
# Prepare a series of numbers to represent the y-axis in the scatter plot
indices = range(len(df_cleaned))

# Creating scatter plots
plt.figure(figsize=(12, 18))

# Scatter plot for SCR_ICF_SVDAT_ICF
plt.subplot(3, 1, 1)  # 3 rows, 1 column, 1st subplot
plt.scatter(df_cleaned['SCR_ICF_SVDAT_ICF'], indices, color='blue', alpha=0.6, label='SCR_ICF_SVDAT_ICF')
plt.title('Дата подписания ИС')
plt.xlabel('Date')
plt.ylabel('Index')
plt.legend()

# Scatter plot for SCR_ICF_SVDAT
plt.subplot(3, 1, 2)  # 3 rows, 1 column, 2nd subplot
plt.scatter(df_cleaned['SCR_ICF_SVDAT'], indices, color='red', alpha=0.6, label='SCR_ICF_SVDAT')
plt.title('Дата первого визита')
plt.xlabel('Date')
plt.ylabel('Index')
plt.legend()

# Scatter plot for SCR_ICF_BRTHDAT
plt.subplot(3, 1, 3)  # 3 rows, 1 column, 3rd subplot
plt.scatter(df_cleaned['SCR_LB_SPERMOGR_LBDAT_SPERM'], indices, color='green', alpha=0.6, label='SCR_LB_SPERMOGR_LBDAT_SPERM')
plt.title('Дата лабораторного исследования')
plt.xlabel('Date')
plt.ylabel('Index')
plt.legend()



# Show the plots
plt.tight_layout()
plt.show()
```

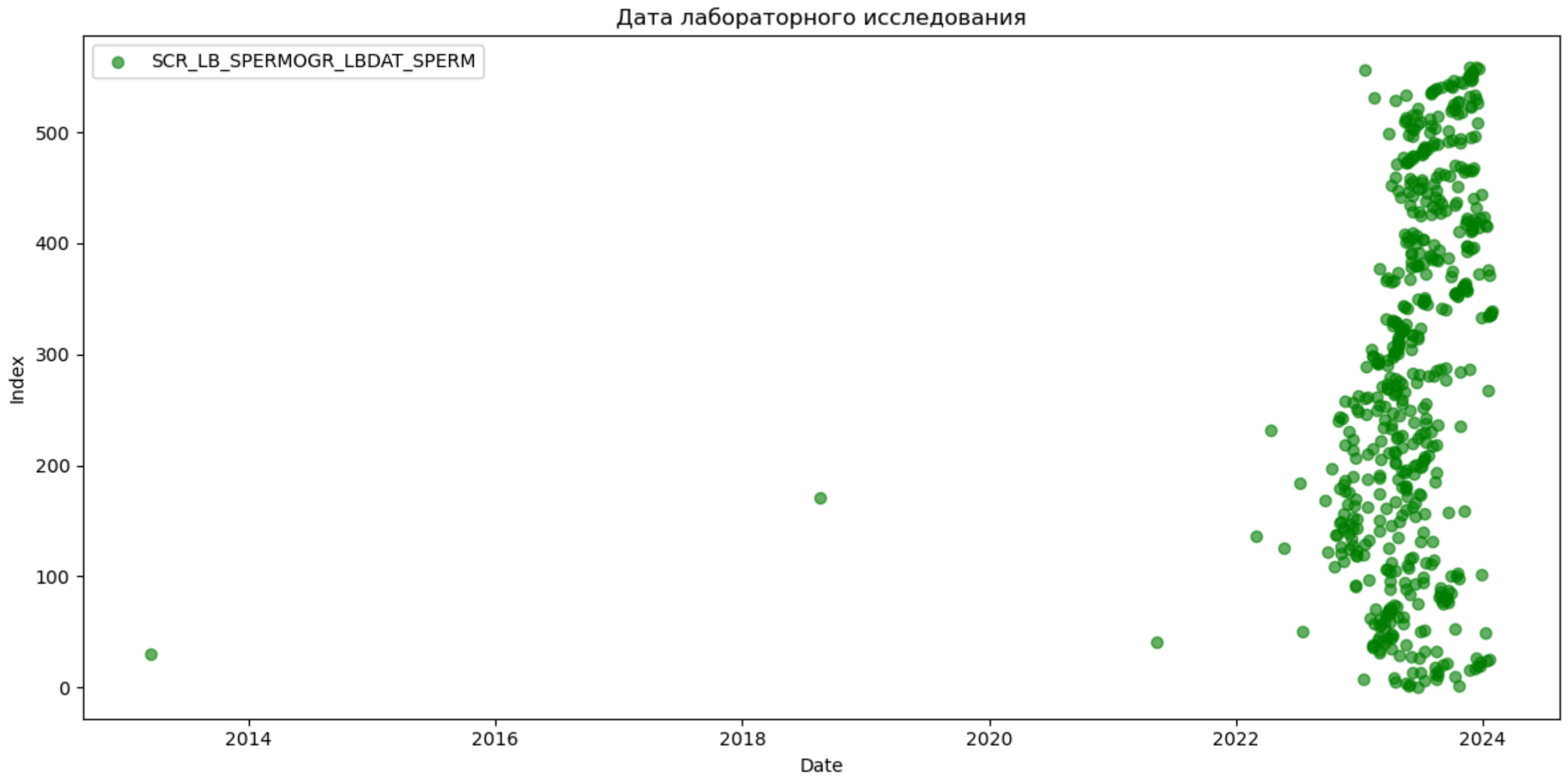
Дата подписания ИС



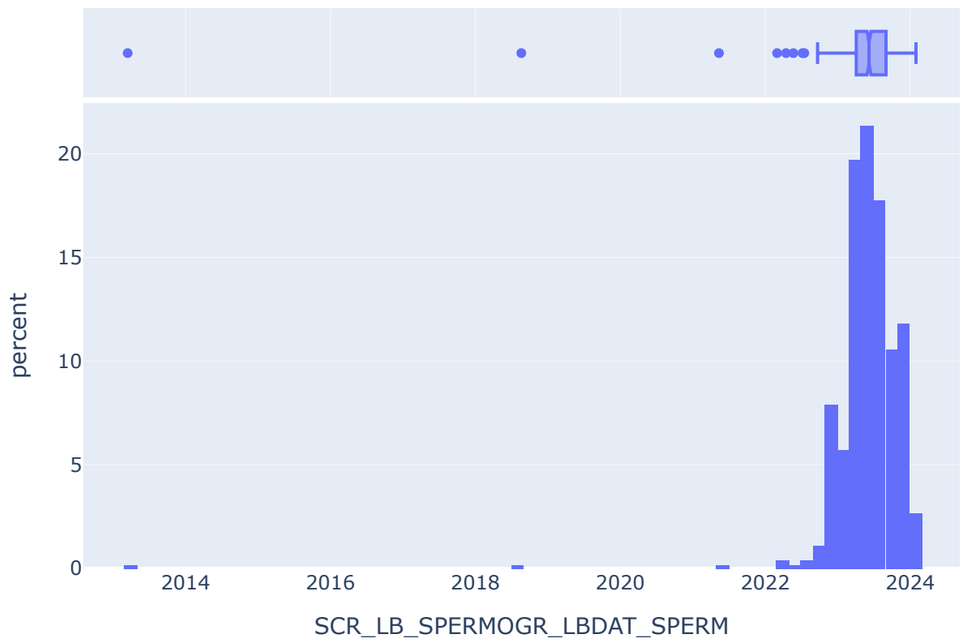
Дата первого визита





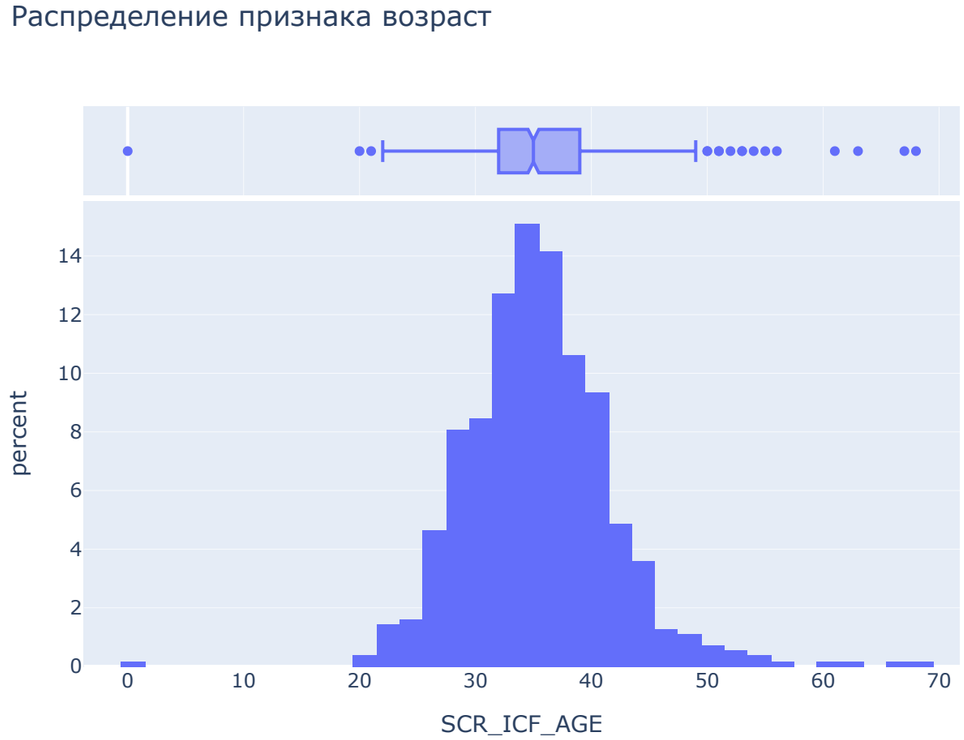


In [ ]:
```
fig = px.histogram(
    data_frame=df_cleaned,
    x = 'SCR_LB_SPERMOGR_LBDAT_SPERM',
    histnorm = 'percent',
    marginal = 'box',
    width = 700,
    title = 'Распределение дат лабораторных исследований'

)
fig.show()
```



Обзор данных по возрасту

In [ ]:
```
fig = px.histogram(
    data_frame=df_cleaned,
    x = 'SCR_ICF_AGE',
    histnorm = 'percent',
    marginal = 'box',
    width = 700,
    title = 'Распределение признака возраст'

)
fig.show()
```



In [ ]:
```
df_cleaned.SCR_ICF_AGE[df_cleaned.SCR_ICF_AGE < 15]
```

Out[ ]:
```
Screening #
014-135    0.0
Name: SCR_ICF_AGE, dtype: float64
```

In [ ]:
```
df_cleaned.recalculated_age[df_cleaned.recalculated_age < 15].index[0]
```

Out[ ]:
```
'014-135'
```

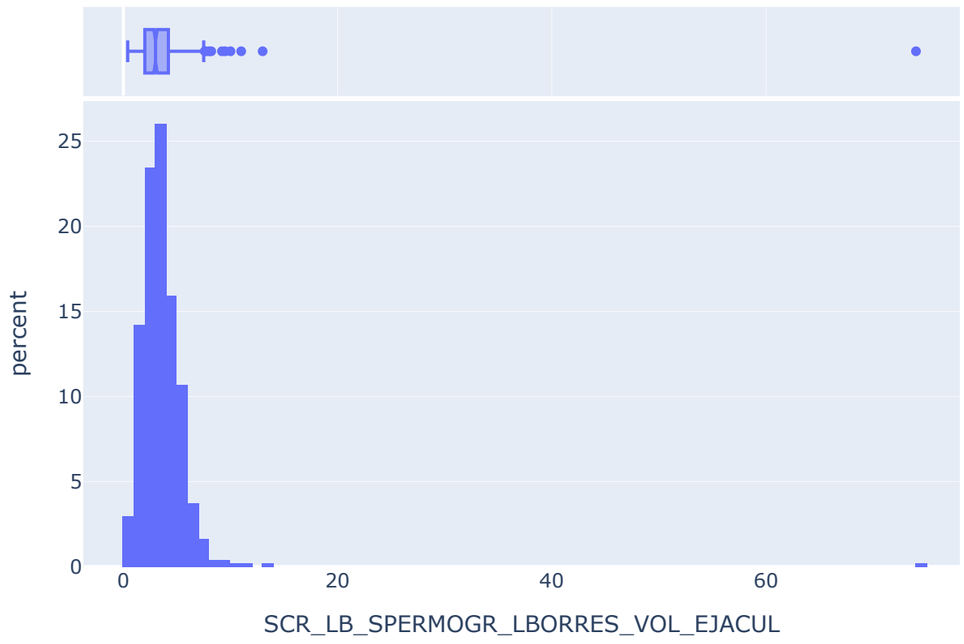
In [ ]:
```
df_cleaned.loc[df_cleaned.recalculated_age[df_cleaned.recalculated_age < 15].index[0]]
```

Out[ ]:
```
Initials                                              СГА
Site #                                                 14
SCR_ICF_SVDAT_ICF                  2023-08-22 00:00:00
SCR_ICF_SVDAT                      2023-08-22 00:00:00
SCR_ICF_BRTHDAT                    2023-03-08 00:00:00
SCR_ICF_AGE                                        0.0
SCR_LB_SPERMOGR_LBDAT_SPERM        2023-08-22 00:00:00
SCR_LB_SPERMOGR_LBORRES_VOL_EJACUL                 6.3
SCR_LB_SPERMOGR_LBORRES_TOTCONC_SP               182.7
SCR_LB_SPERMOGR_LBORRES_COCENTR_SP                29.0
SCR_LB_SPERMOGR_LBORRES_TOTMOTILIT_SP             56.0
SCR_LB_SPERMOGR_LBORRES_PROGR_MOBIL               14.0
SCR_LB_SPERMOGR_LBORRES_PNONPROGR_MOBIL           35.0
SCR_LB_SPERMOGR_LBORRES_FIXED                     44.0
recalculated_age                              0.457221
age_deviation                                 0.457221
Name: 014-135, dtype: object
```

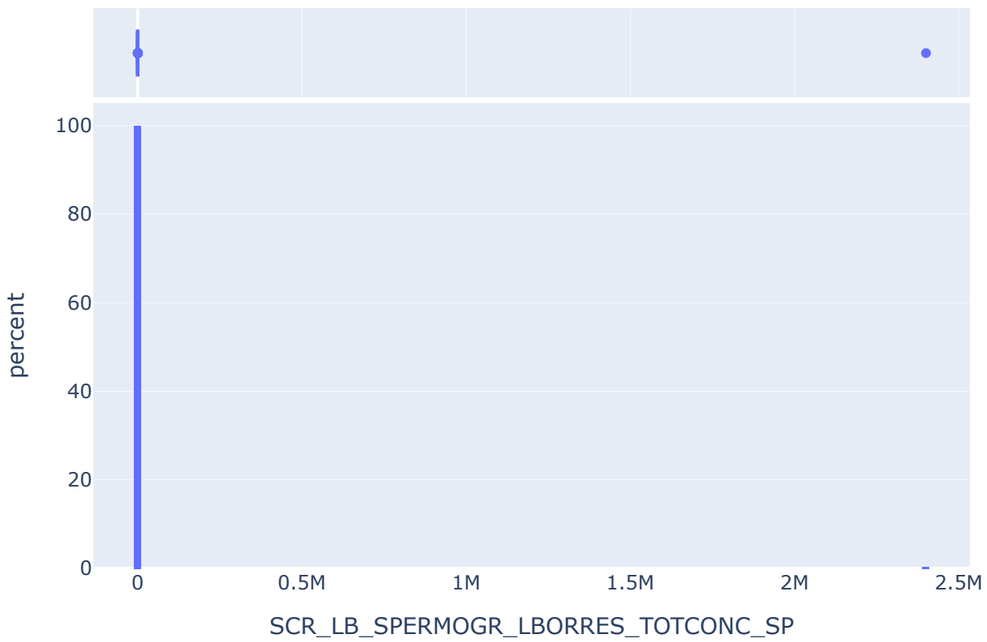
Обзор распределений данных по лабораторным исследованиям

In [ ]:
```
# Распределения по лабораторным исследованиям
for col in spermo_columns:
    fig = px.histogram(
        data_frame=df_cleaned,
        x=col,
        histnorm='percent',
        marginal='box',
        width=700,
        title=f'Распределение признака {col}'
    )
    fig.show()
```

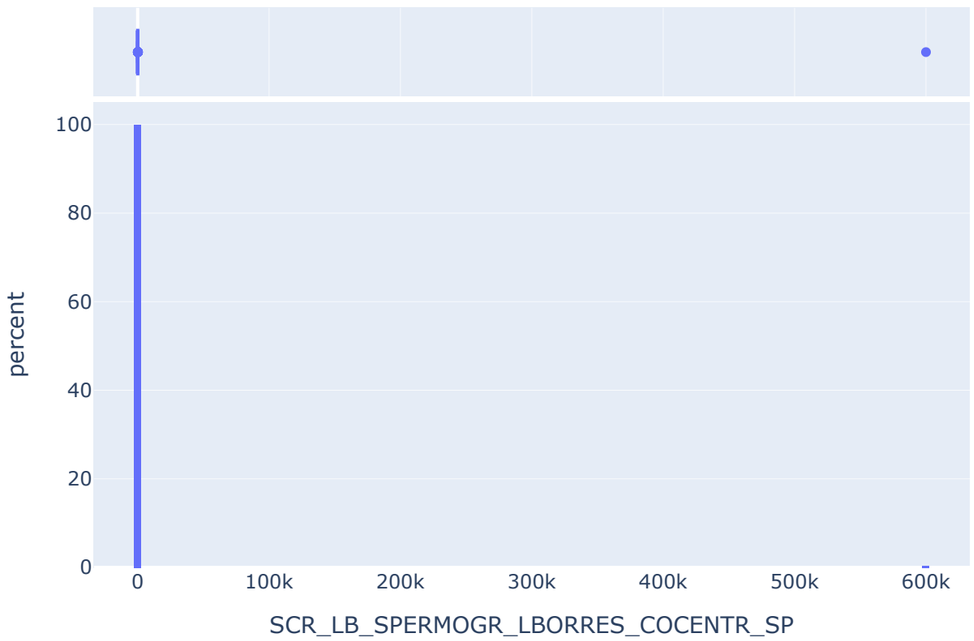
Распределение признака SCR_LB_SPERMOGR_LBORRES_VOL_EJACUL



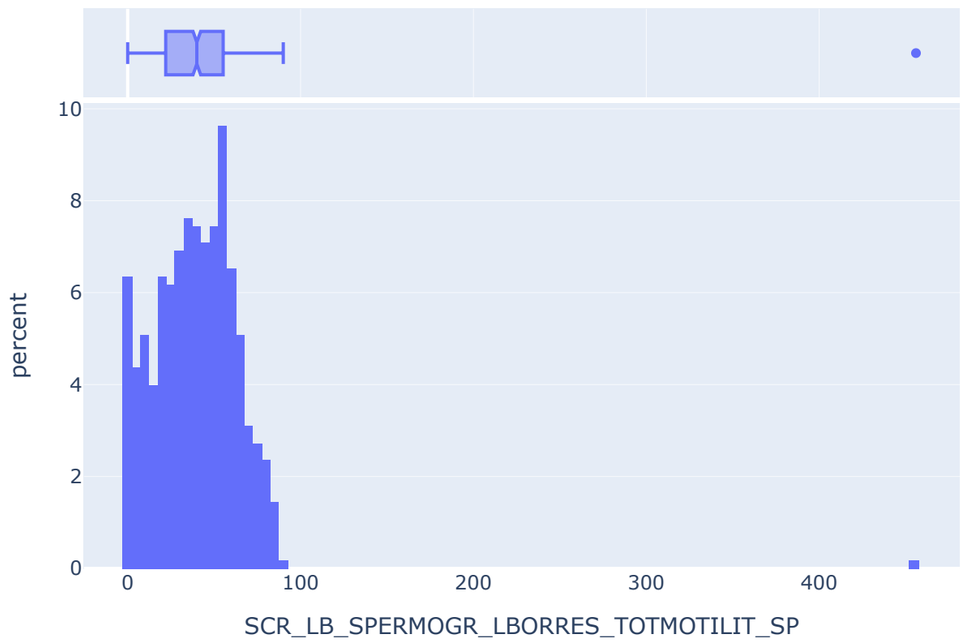
Распределение признака SCR_LB_SPERMOGR_LBORRES_TOTCONC_SP



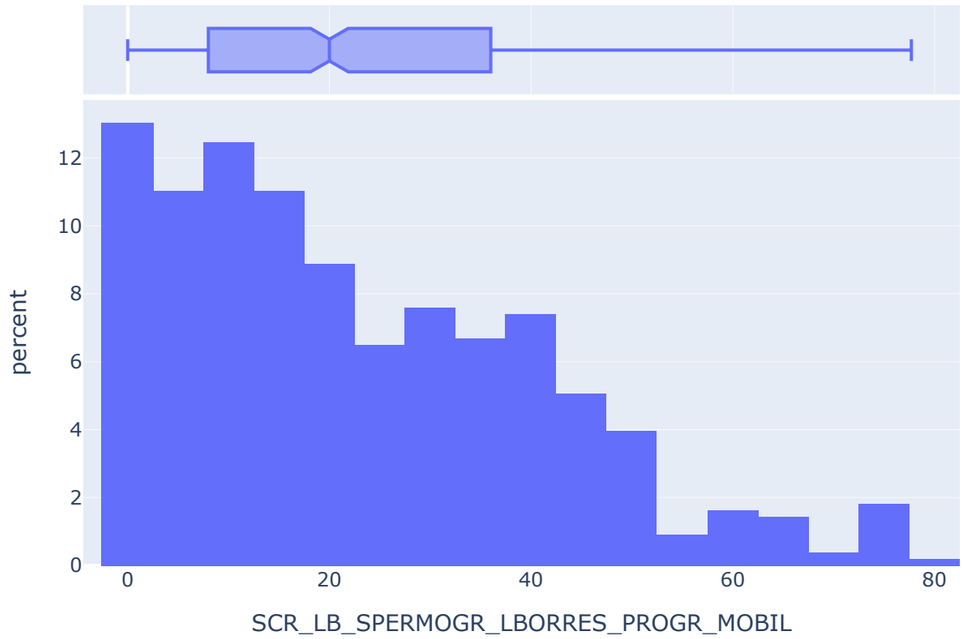
Распределение признака SCR_LB_SPERMOGR_LBORRES_COCENTR_SP



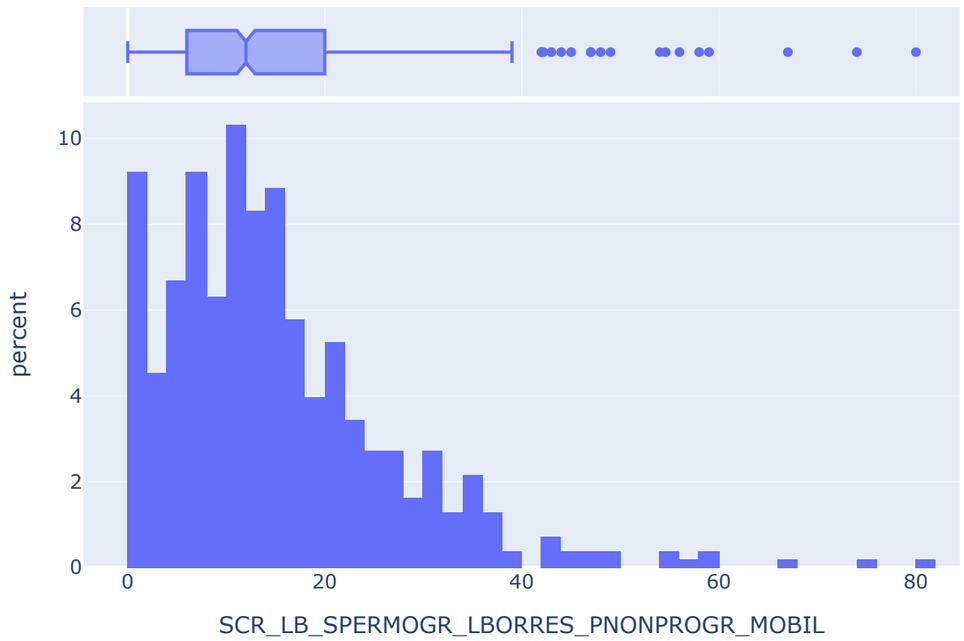
Распределение признака SCR_LB_SPERMOGR_LBORRES_TOTMOTILIT_SP

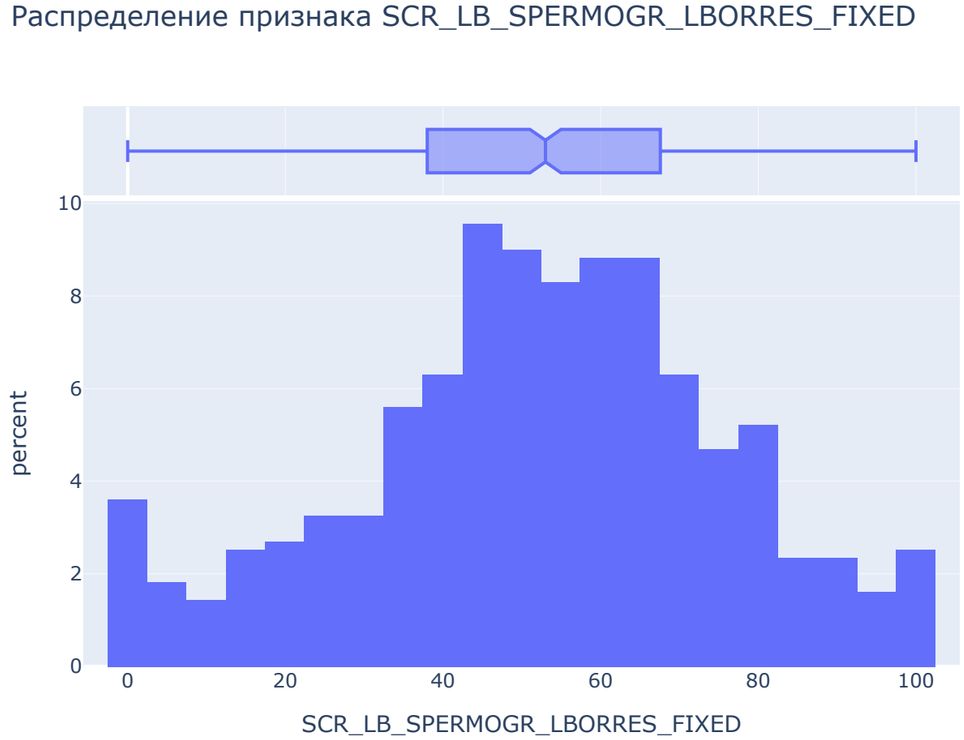


Распределение признака SCR_LB_SPERMOGR_LBORRES_PROGR_MOBIL



Распределение признака SCR_LB_SPERMOGR_LBORRES_PNONPROGR_MOB





In [ ]:
```python
# Find indices of rows where values exceed IQR by 10 times
outlier_indices = set()

# Find indices of rows where values exceed IQR by 10 times
for col in spermo_columns:
    q1 = df_cleaned[col].quantile(0.25)
    q3 = df_cleaned[col].quantile(0.75)
    iqr = q3 - q1

    # Find outlier indices for the current column
    outliers = df_cleaned[(df_cleaned[col] < q1 - 10 * iqr) | (df_cleaned[col] > q3 + 10 * iqr)].index

    # Update the set of all outlier indices
    outlier_indices.update(outliers)

# Remove rows corresponding to outliers
df_cleaned_filtered = df_cleaned.drop(index=outlier_indices)
print(f'Удалены индексы с аномалиями: {outlier_indices}')
```

```
Удалены индексы с аномалиями: {'024-001', '053-004', '014-127'}
```

In [ ]:
```python
# Calculate IQR for each column in spermo_columns
iqr_values = {}
for col in spermo_columns:
    q1 = df_cleaned[col].quantile(0.25)
    q3 = df_cleaned[col].quantile(0.75)
    iqr = q3 - q1
    iqr_values[col] = iqr

# Find indices of rows where values exceed IQR by 10 times
outlier_indices = {}
```

```python
for col in spermo_columns:
    # Recalculate q1 and q3 for the current column
    q1 = df_cleaned[col].quantile(0.25)
    q3 = df_cleaned[col].quantile(0.75)
    iqr = iqr_values[col]
    outliers = df_cleaned[(df_cleaned[col] < q1 - 10 * iqr) | (df_cleaned[col] > q3 + 10 * iqr)].index
    outlier_indices[col] = outliers.tolist()

# Find indices of rows where values exceed IQR by 10 times
outlier_results = []
for col in spermo_columns:
    # Recalculate q1 and q3 for the current column
    q1 = df_cleaned[col].quantile(0.25)
    q3 = df_cleaned[col].quantile(0.75)
    iqr = iqr_values[col]

    # Find outliers
    outliers = df_cleaned[(df_cleaned[col] < q1 - 10 * iqr) | (df_cleaned[col] > q3 + 10 * iqr)]

    # Add outlier information to the results list
    for index in outliers.index:
        outlier_results.append({'Patient ID': index, 'Feature': col, 'Value': outliers.at[index, col]})

# Create a beautiful table with the results
outlier_table = pd.DataFrame(outlier_results)
print(outlier_table)
```

```
  Patient ID                                Feature      Value
0    053-004     SCR_LB_SPERMOGR_LBORRES_VOL_EJACUL       74.0
1    024-001     SCR_LB_SPERMOGR_LBORRES_TOTCONC_SP  2400000.0
2    024-001     SCR_LB_SPERMOGR_LBORRES_COCENTR_SP   600000.0
3    014-127  SCR_LB_SPERMOGR_LBORRES_TOTMOTILIT_SP      456.0
```

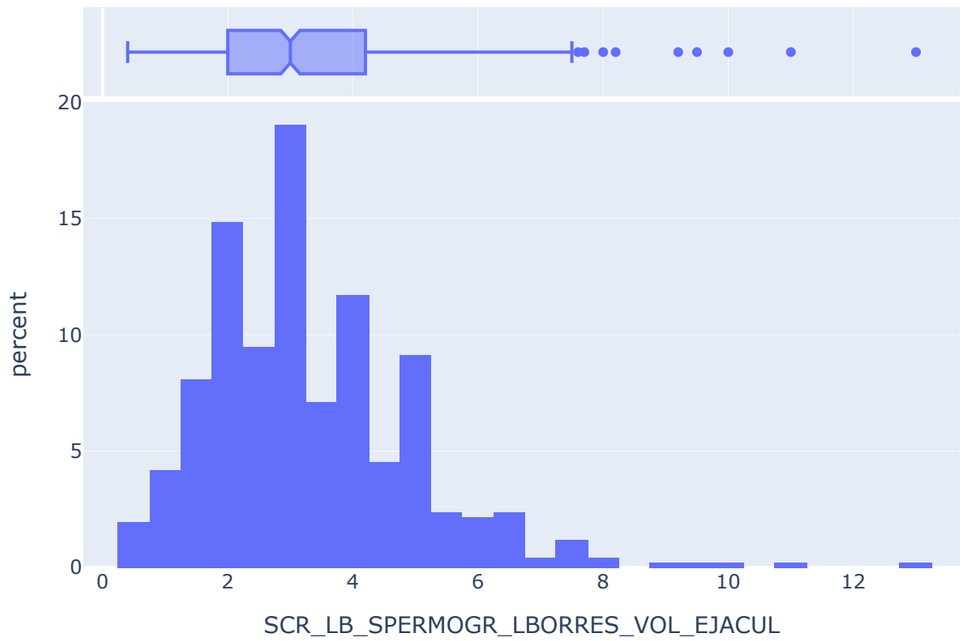
In [ ]:
```python
# Remove rows corresponding to outliers
df_cleaned = df_cleaned.drop(index=(outlier_table['Patient ID'].values))
```

In [ ]:
```python
df_cleaned.shape
```

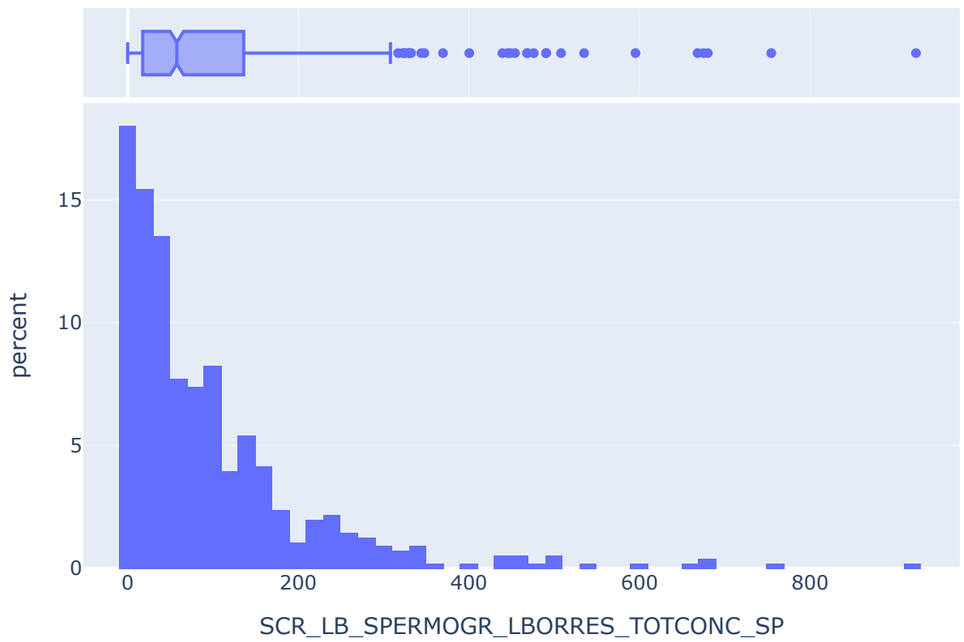
Out[ ]:
```
(557, 16)
```

In [ ]:
```python
# Распределения по лабораторным исследованиям
for col in spermo_columns:
    fig = px.histogram(
        data_frame=df_cleaned,
        x=col,
        histnorm='percent',
        marginal='box',
        width=700,
        title=f'Распределение признака {col}'
    )
    fig.show()
```

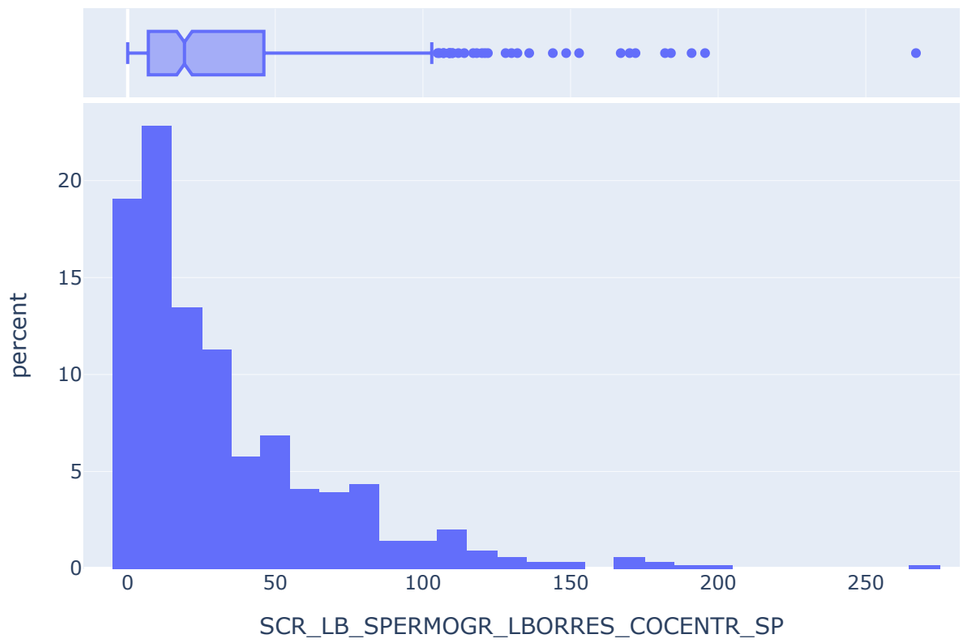
Распределение признака SCR_LB_SPERMOGR_LBORRES_VOL_EJACUL



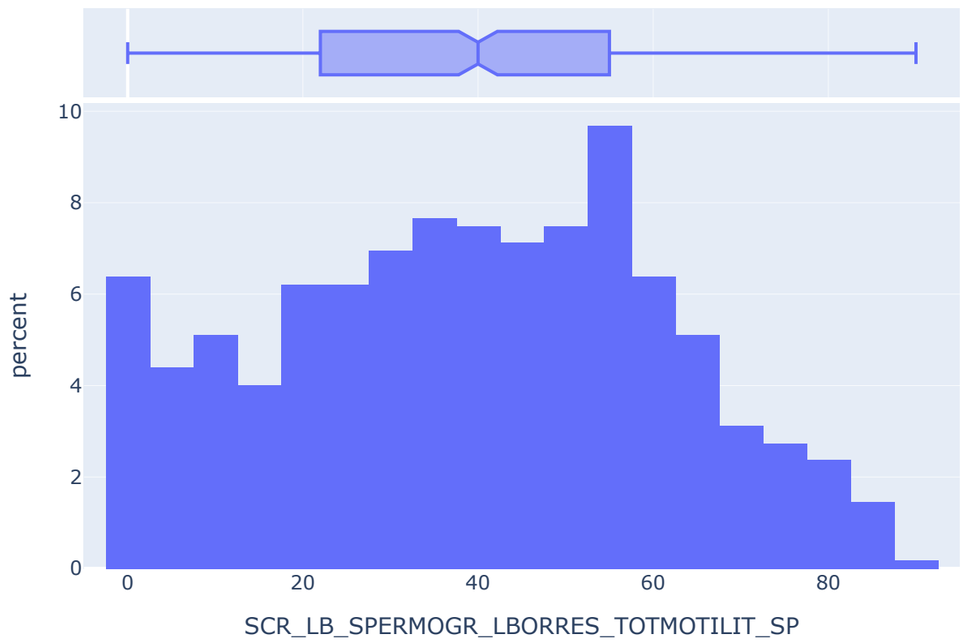
Распределение признака SCR_LB_SPERMOGR_LBORRES_TOTCONC_SP



Распределение признака SCR_LB_SPERMOGR_LBORRES_COCENTR_SP

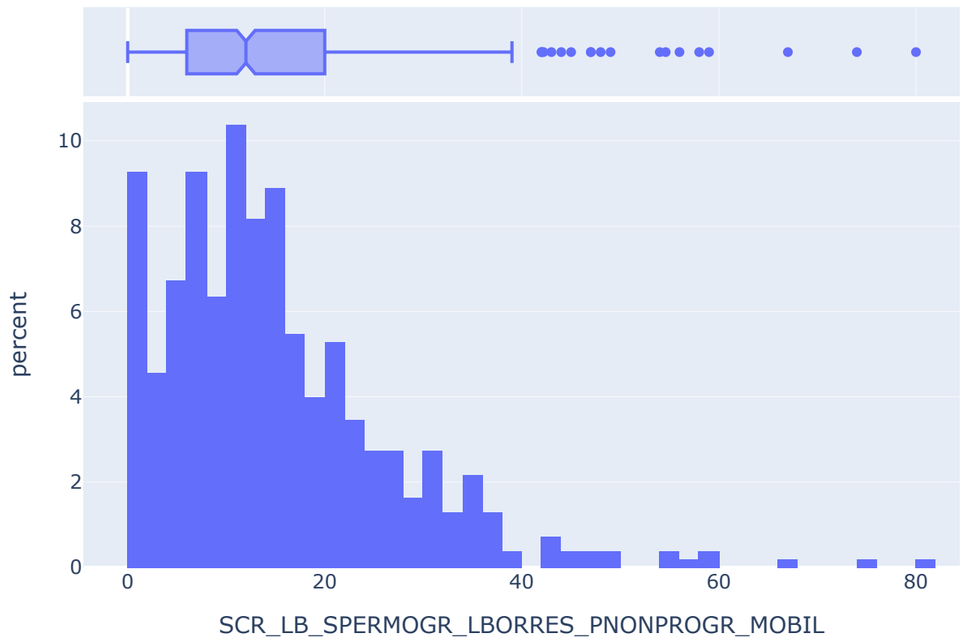


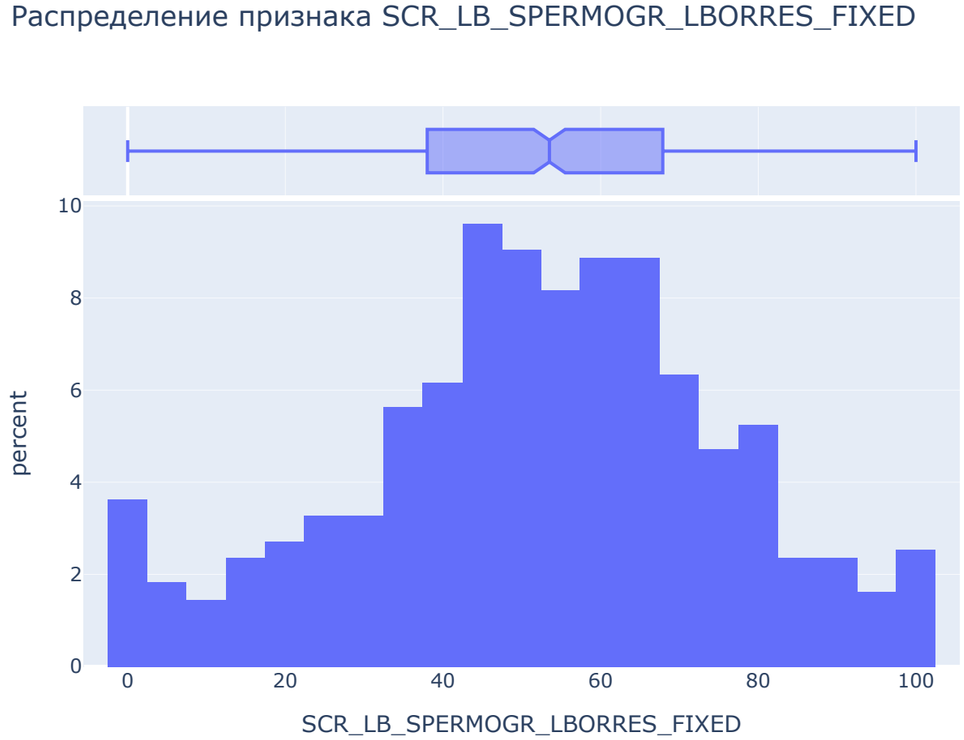
Распределение признака SCR_LB_SPERMOGR_LBORRES_TOTMOTILIT_SP



Распределение признака SCR_LB_SPERMOGR_LBORRES_PROGR_MOBIL

Распределение признака SCR_LB_SPERMOGR_LBORRES_PNONPROGR_MOB



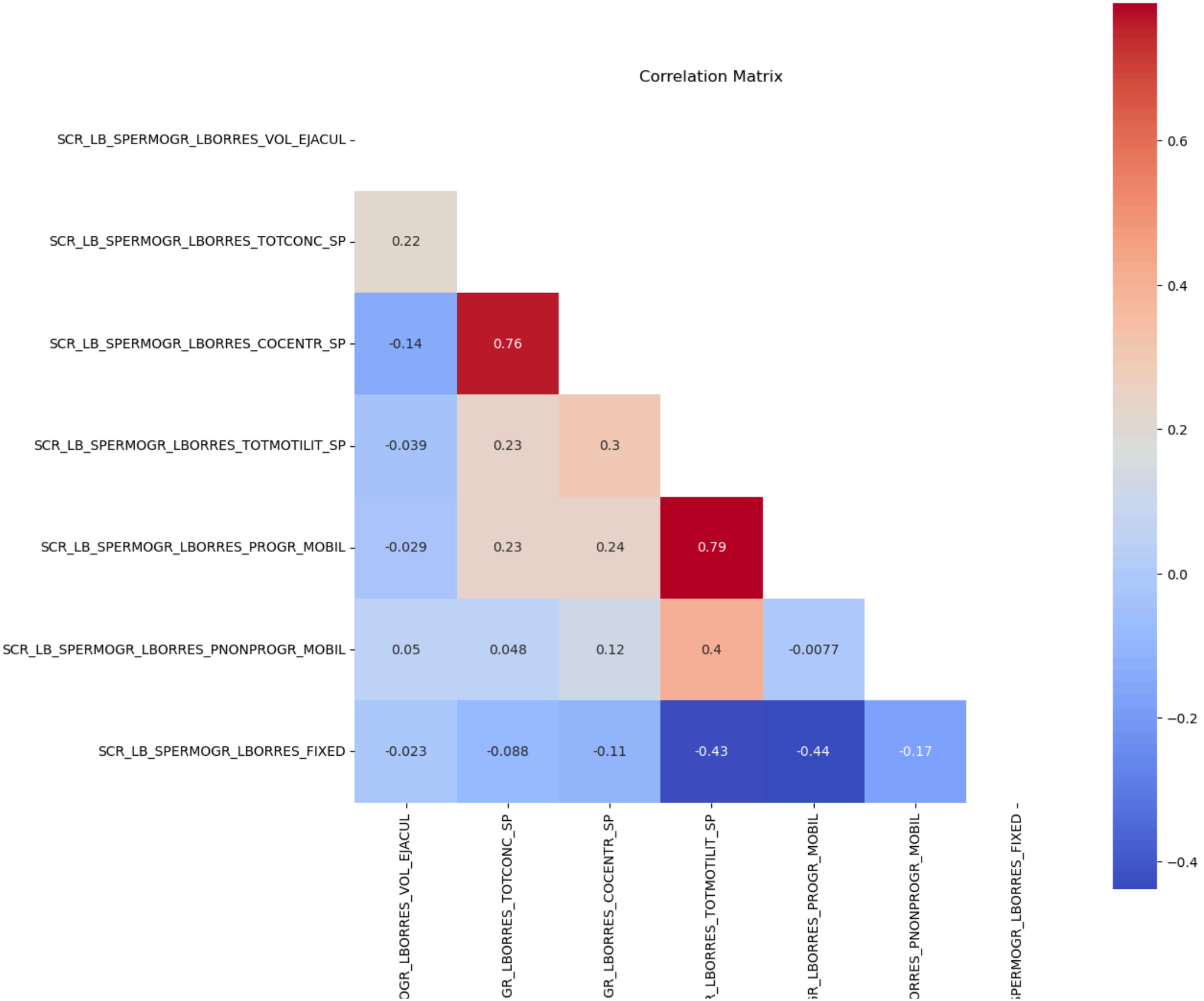


Задача 2: Определение сфальсифицированных признаков

Матрица корреляции признаков

In [ ]:
```
# Select only the columns in spermo_columns
corr_df = df_cleaned[spermo_columns]

# Calculate the correlation matrix
corr_matrix = corr_df.corr()

# Create a mask to remove the upper triangle
mask = np.triu(np.ones_like(corr_matrix, dtype=bool))

# Create a beautiful correlation matrix plot
plt.figure(figsize=(12, 12))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', square=True, mask=mask)
plt.title('Correlation Matrix')
plt.show()
# Get the top correlations
top_corr = corr_matrix.unstack().sort_values(ascending=False).drop_duplicates()

# Print the top correlations
print('Топ корреляции показателей лабораторных результатов')
print(top_corr[top_corr > 0.7])
```

Correlation Matrix




```
Топ корреляции показателей лабораторных результатов
SCR_LB_SPERMOGR_LBORRES_VOL_EJACUL     SCR_LB_SPERMOGR_LBORRES_VOL_EJACUL     1.000000
SCR_LB_SPERMOGR_LBORRES_TOTMOTILIT_SP  SCR_LB_SPERMOGR_LBORRES_PROGR_MOBIL    0.791283
SCR_LB_SPERMOGR_LBORRES_TOTCONC_SP     SCR_LB_SPERMOGR_LBORRES_COCENTR_SP     0.763382
dtype: float64
```

Вывод: обнаружена пара высокоррелированных признаков

- `SCR_LB_SPERMOGR_LBORRES_TOTMOTILIT_SP`
- `SCR_LB_SPERMOGR_LBORRES_PROGR_MOBIL`

z-score

`Z-score` — это мера, которая описывает положение значения в выборке относительно среднего значения, измеренное в числе стандартных отклонений. Этот метод хорошо работает, если данные имеют нормальное распределение.

Часть наших данных на графиках выше - скошена влево.

Если распределение данных скошено, то z-score может не быть лучшим способом обнаружения выбросов, поскольку оценки среднего и стандартного отклонения могут быть искажены.

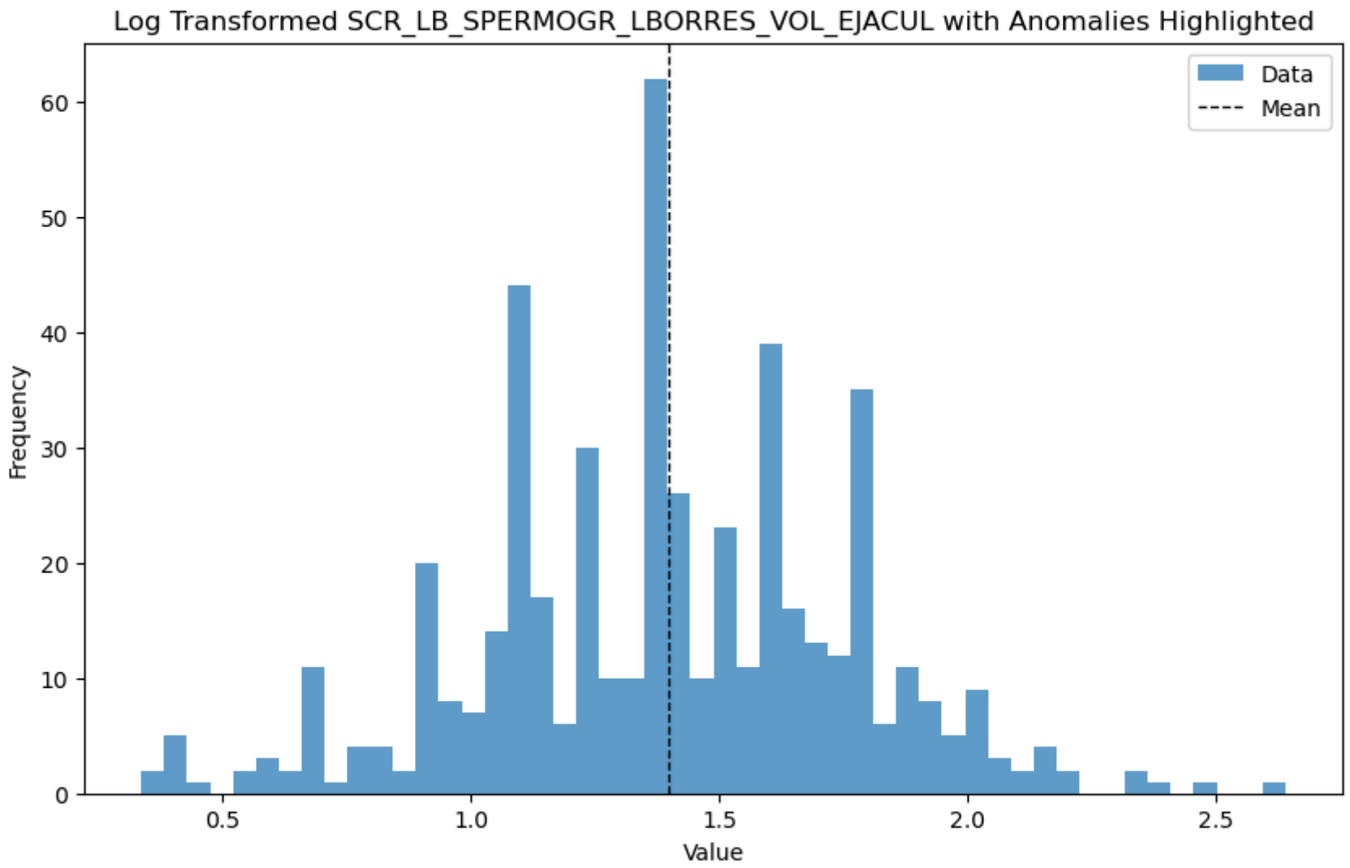
Один из способов обработки скошенных данных — преобразование данных для приближения к нормальному распределению. Примеры таких преобразований включают логарифмическое преобразование, квадратный корень, обратное преобразование и Box-Cox преобразование.

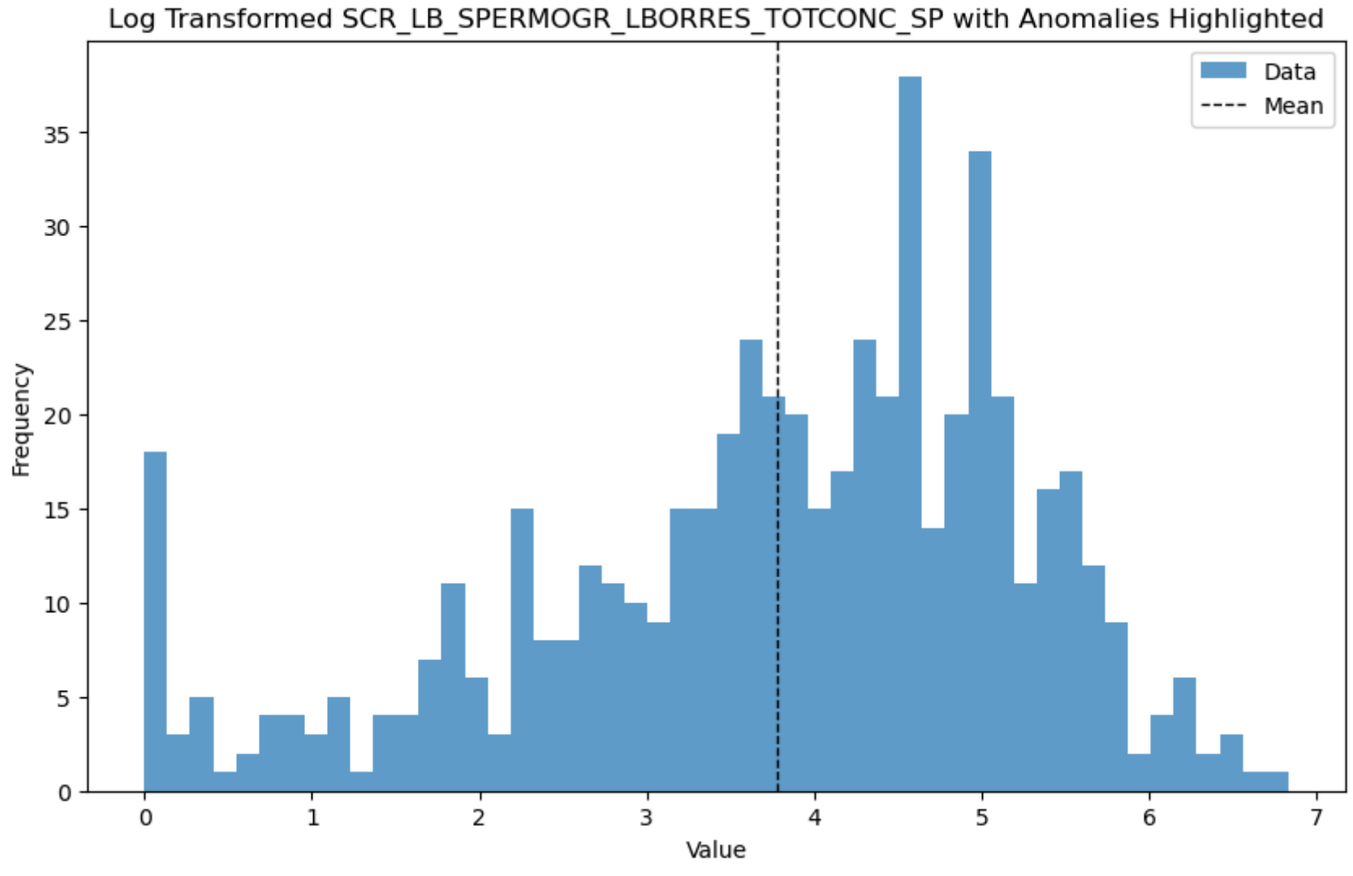
Этот код выведет список аномальных значений и создаст гистограммы логарифмированных признаков с выделенными аномалиями.

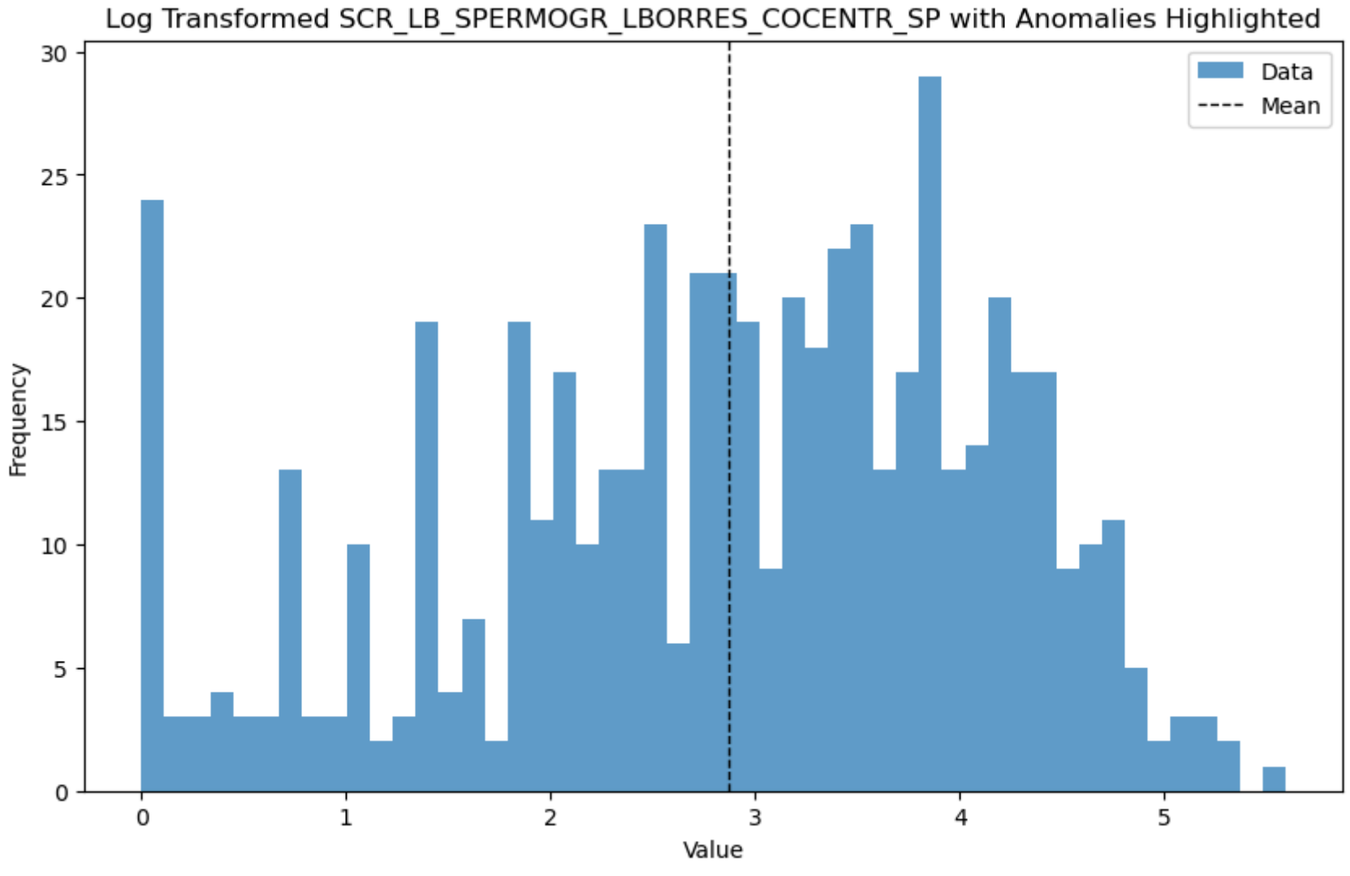
In [ ]:
```python
# Применяем логарифмическое преобразование и вычисляем z-score
for col in spermo_columns:
    df_cleaned[col + '_log'] = np.log1p(df_cleaned[col])
    df_cleaned[col + '_log_zscore'] = zscore(df_cleaned[col + '_log'])

# Идентифицируем аномалии
threshold_zscore = 3
anomalies = pd.DataFrame()

for col in spermo_columns:
    z_col = col + '_log_zscore'
    anomalies_col = df_cleaned[(df_cleaned[z_col] > threshold_zscore) | (df_cleaned[z_col] < -threshold_zscore)]
    anomalies_col = anomalies_col[[col, z_col]]
    anomalies_col['Feature'] = col
    anomalies_col.rename(columns={col: 'Value', z_col: 'Z-score'}, inplace=True)
    anomalies = pd.concat([anomalies, anomalies_col[['Feature', 'Value', 'Z-score']]])

# Выводим список аномалий
print(anomalies)

# Визуализация аномалий
for col in spermo_columns:
    plt.figure(figsize=(10, 6))
```

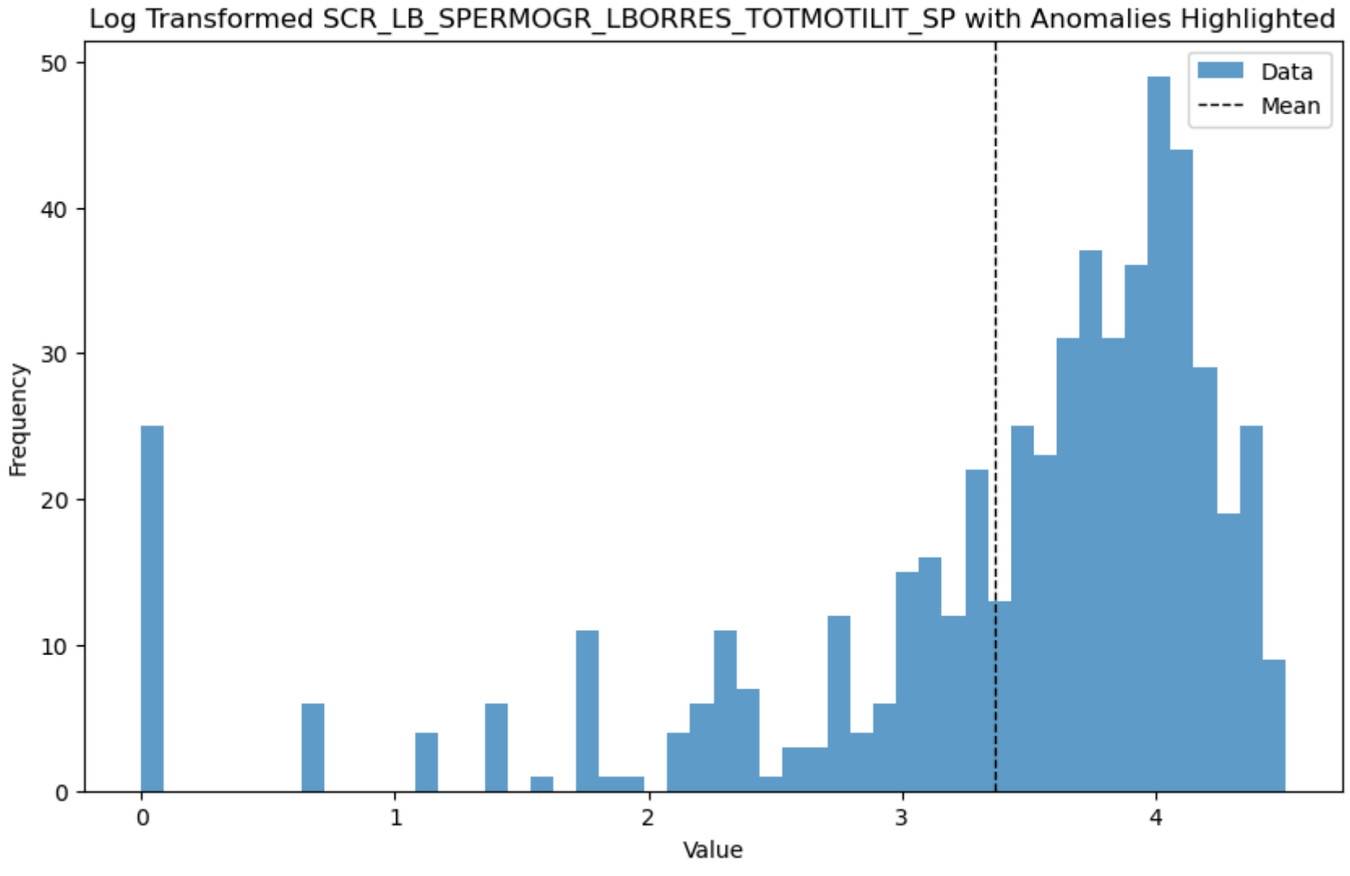
```
    plt.hist(df_cleaned[col + '_log'], bins=50, alpha=0.7, label='Data')
    plt.axvline(df_cleaned[col + '_log'].mean(), color='k', linestyle='dashed', linewidth=1, label='Mean')
    # Подсветка аномалий
    anomalies_col = df_cleaned[(df_cleaned[col + '_log_zscore'] > threshold_zscore) |
                               (df_cleaned[col + '_log_zscore'] < -threshold_zscore)]
    if not anomalies_col.empty:
        plt.scatter(anomalies_col[col + '_log'], [0] * anomalies_col.shape[0], color='r', label='Anomalies')
    plt.title(f'Log Transformed {col} with Anomalies Highlighted')
    plt.xlabel('Value')
    plt.ylabel('Frequency')
    plt.legend()
    plt.show()
```

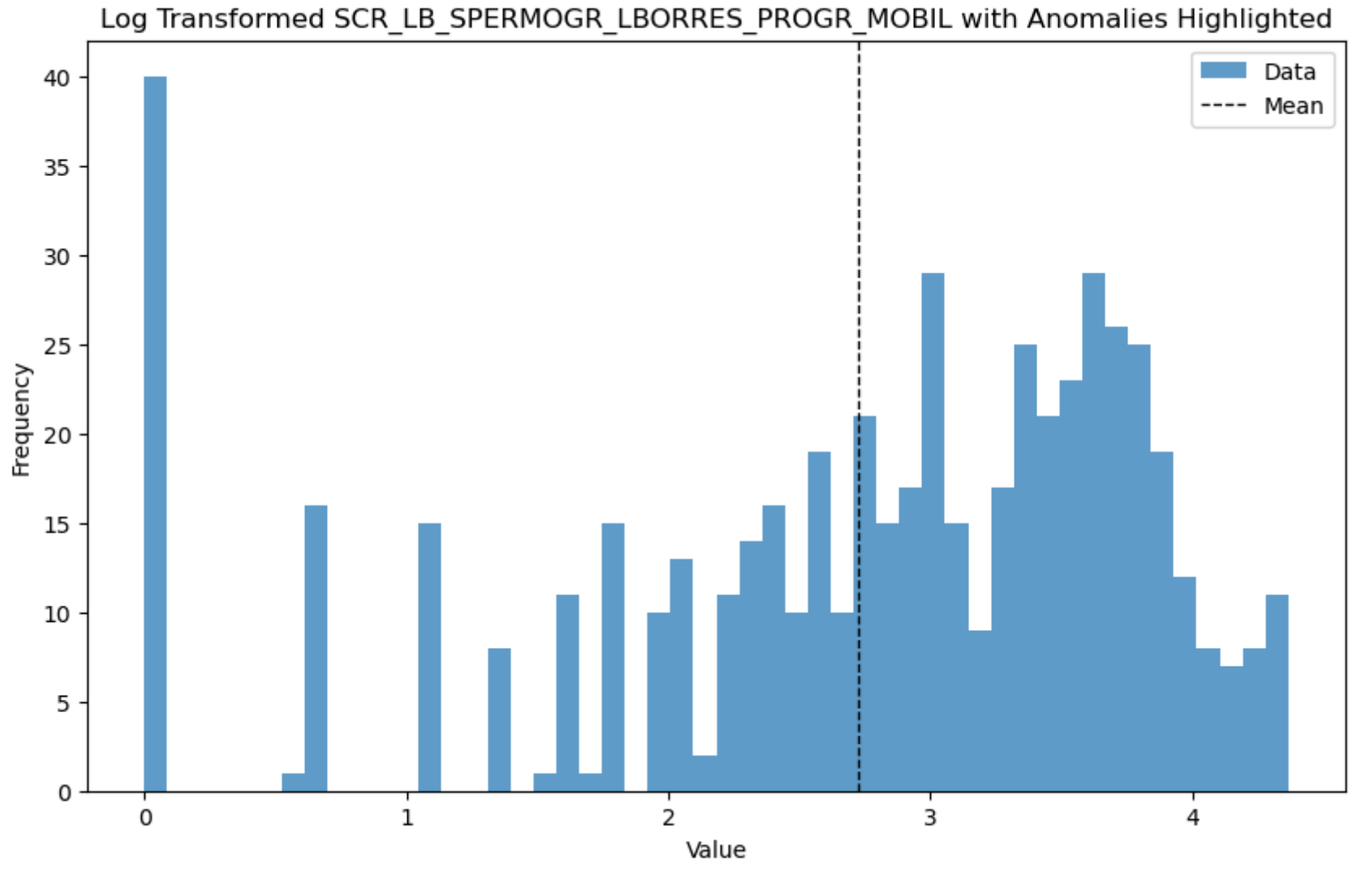
```
Empty DataFrame
Columns: [Feature, Value, Z-score]
Index: []
```

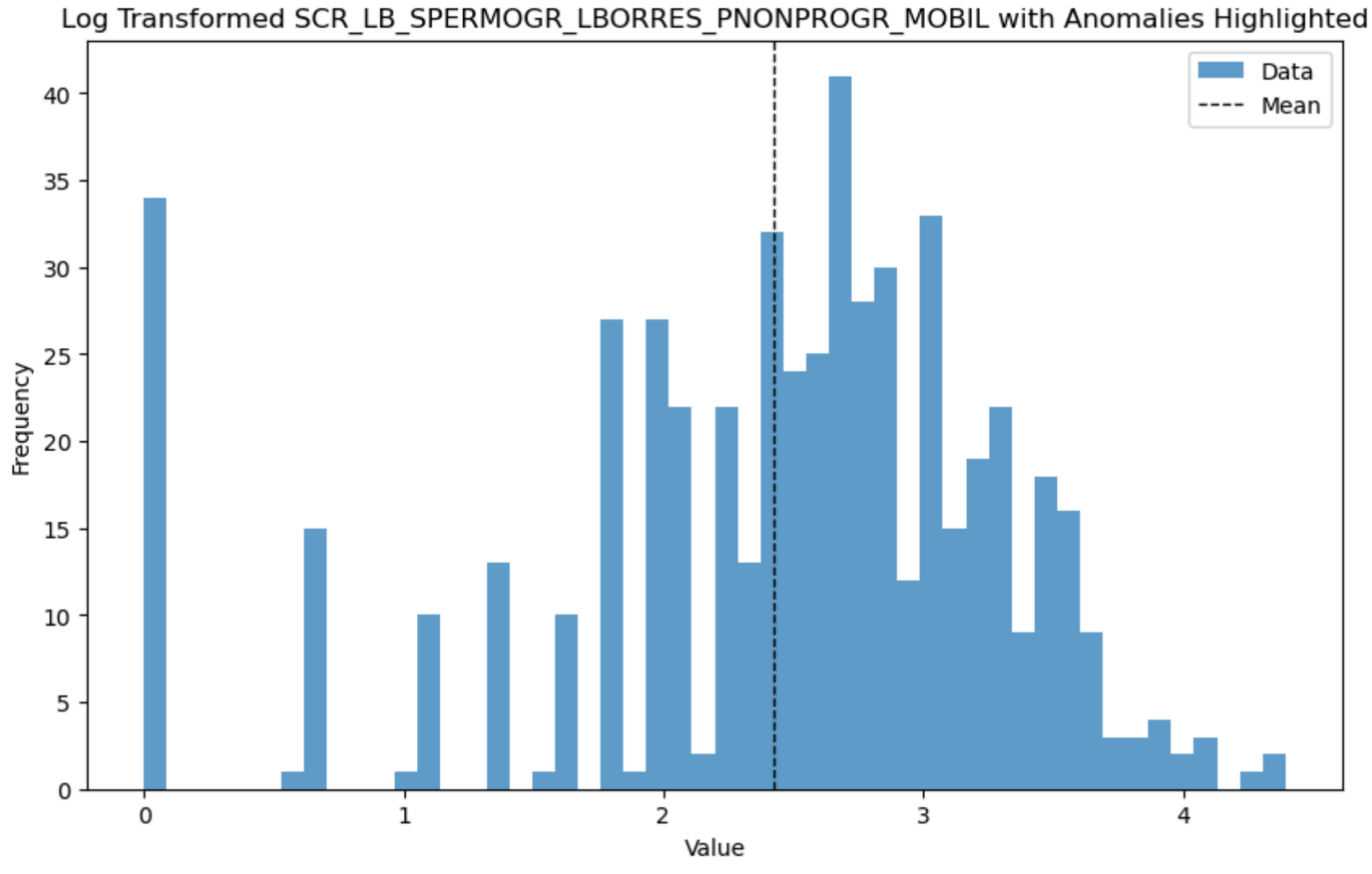


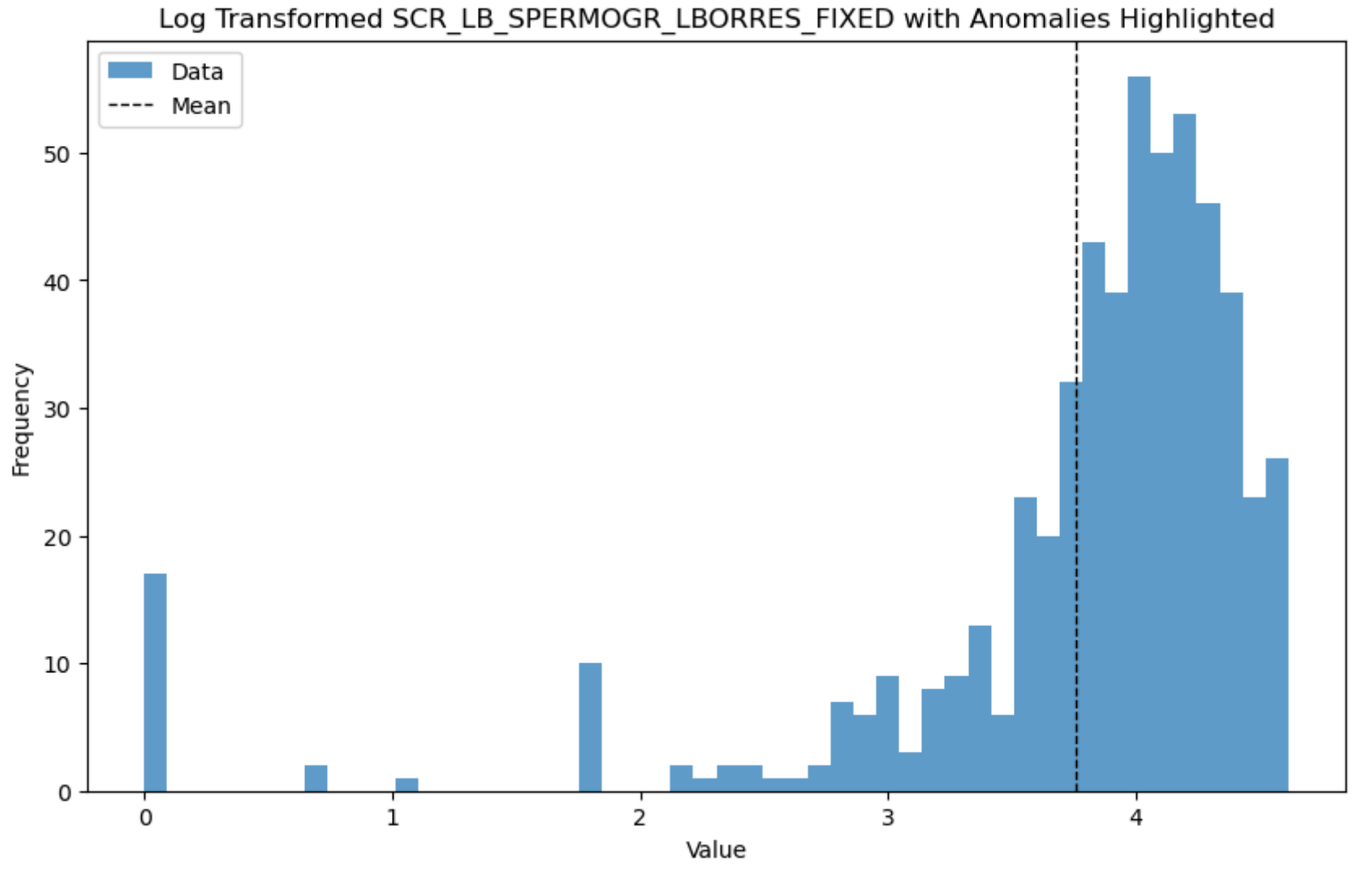

Log Transformed SCR_LB_SPERMOGR_LBORRES_TOTCONC_SP with Anomalies Highlighted
Data
Mean
Frequency
Value
0
5
10
15
20
25
30
35
0
1
2
3
4
5
6
7


Log Transformed SCR_LB_SPERMOGR_LBORRES_COCENTR_SP with Anomalies Highlighted
Data
Mean
Frequency
Value
0
5
10
15
20
25
30
0
1
2
3
4
5


Log Transformed SCR_LB_SPERMOGR_LBORRES_TOTMOTILIT_SP with Anomalies Highlighted
Data
Mean
Frequency
Value
0
10
20
30
40
50
0
1
2
3
4


Log Transformed SCR_LB_SPERMOGR_LBORRES_PROGR_MOBIL with Anomalies Highlighted
Data
Mean
Frequency
Value
0
5
10
15
20
25
30
35
40
0
1
2
3
4


Log Transformed SCR_LB_SPERMOGR_LBORRES_PNONPROGR_MOBIL with Anomalies Highlighted
Data
Mean
Frequency
Value
0
5
10
15
20
25
30
35
40
0
1
2
3
4

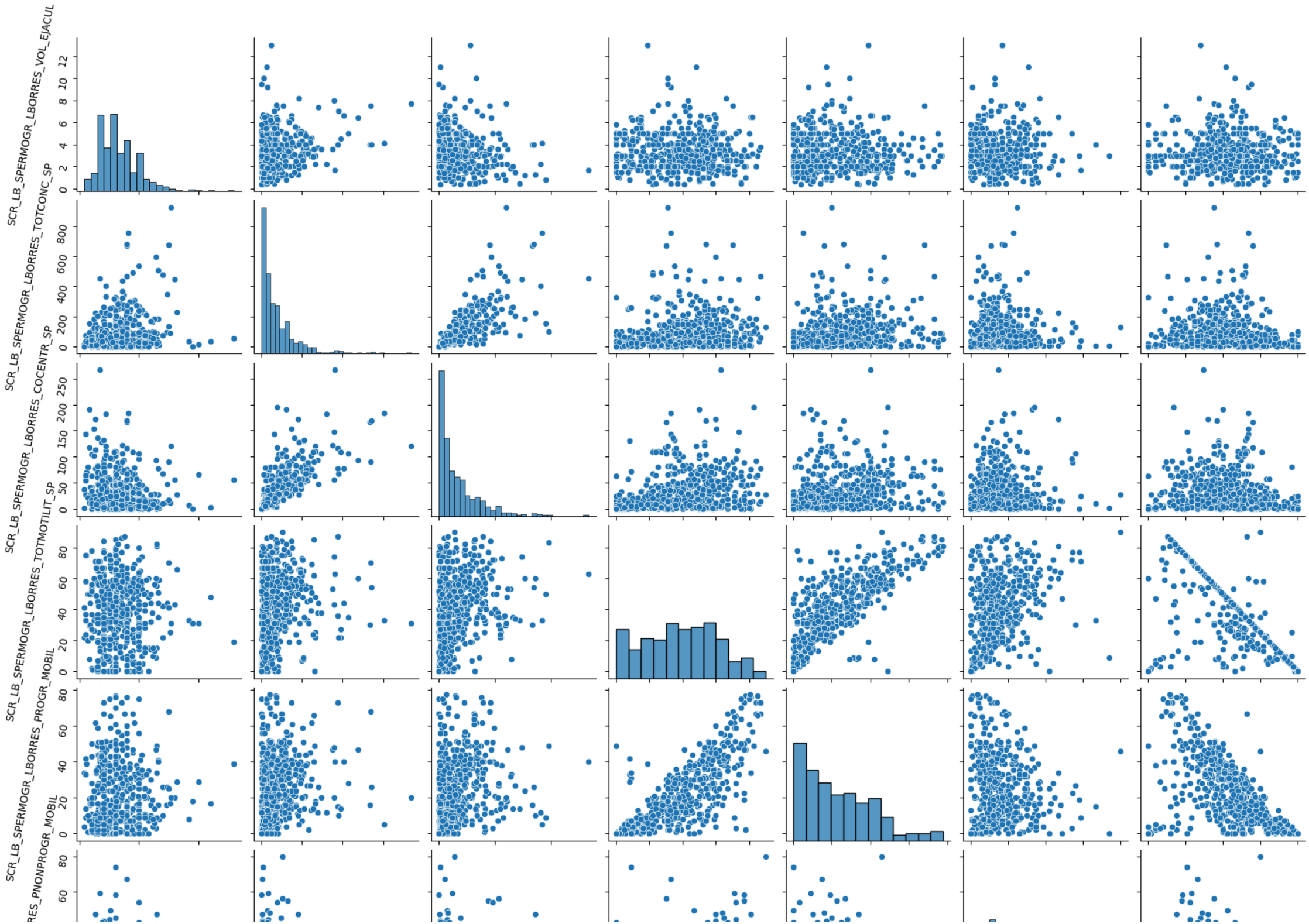


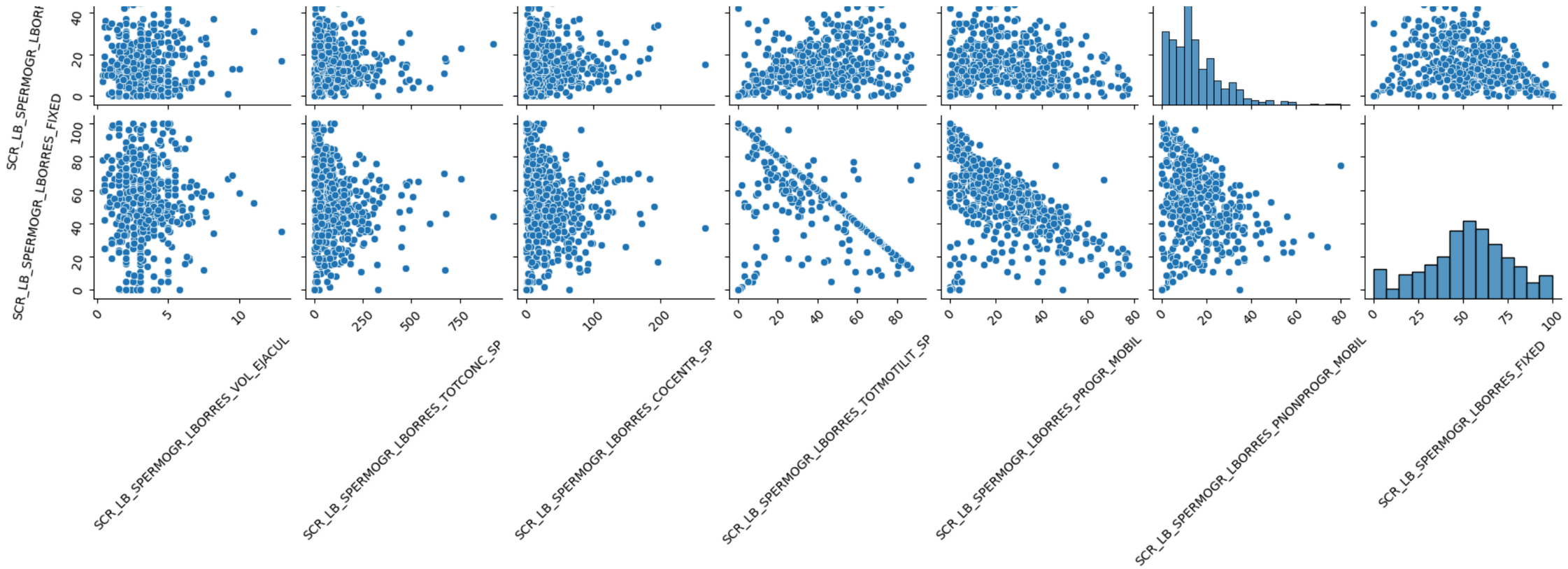
Вывод: методом z-score по логарифмированным данным выбросы и аномалии в сабсете по значениям лабораторных результатов - не обнаружены

Pairplot

In [ ]:
```
# Создаем pairplot для всех признаков в df_cleaned
g = sns.pairplot(df_cleaned[spermo_columns])

# Поворачиваем метки на 45 градусов
for ax in g.axes.flatten():
    # Поворачиваем метки оси X
    if ax is not None:
        ax.set_xlabel(ax.get_xlabel(), rotation=45)
        ax.set_ylabel(ax.get_ylabel(), rotation=80, labelpad=20)
        # Поворачиваем текст меток
        for label in ax.get_xticklabels():
            label.set_rotation(45)
        for label in ax.get_yticklabels():
            label.set_rotation(80)

# Устанавливаем заголовок для всей фигуры, а не для отдельных осей
plt.subplots_adjust(top=0.9)  # Немного поднимаем заголовок, чтобы не перекрывать оси
g.fig.suptitle('Pairplot of all features', fontsize=16)

# Показываем pairplot
plt.show()
```

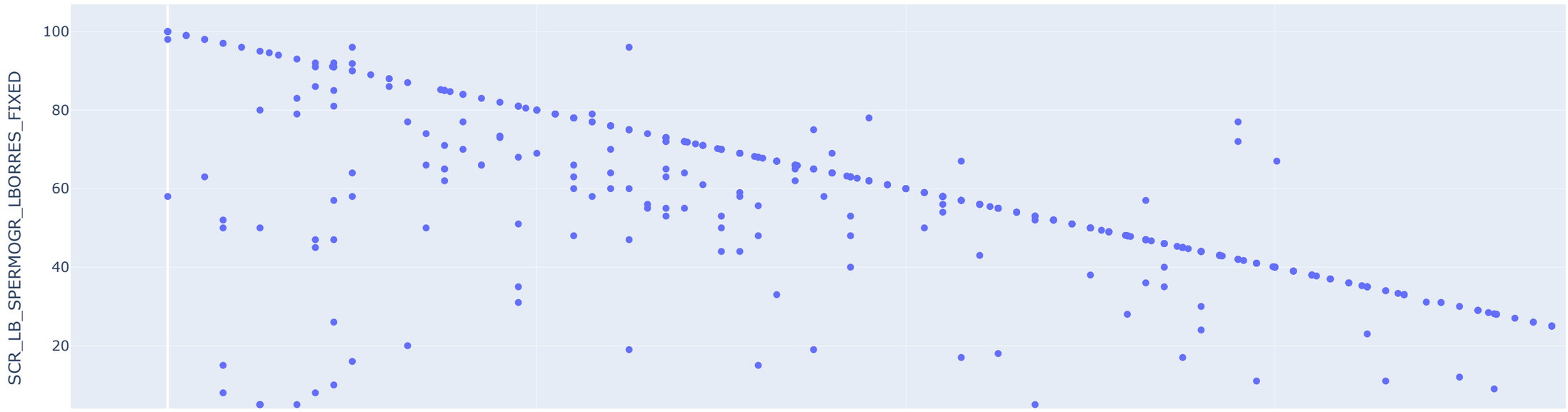
Pairplot of all features





Вывод: построение матрицы парных корреляций признаков указывает на аномальную связь одной пары признаков. Рассмотрим детальнее.

In [ ]:
```python
# Построение scatterplot
fig = px.scatter(df_cleaned, x='SCR_LB_SPERMOGR_LBORRES_TOTMOTILIT_SP', y='SCR_LB_SPERMOGR_LBORRES_FIXED')

# Установка подписей осей и заголовка
fig.update_layout(
    xaxis_title='SCR_LB_SPERMOGR_LBORRES_TOTMOTILIT_SP',
    yaxis_title='SCR_LB_SPERMOGR_LBORRES_FIXED',
    title='Scatterplot of TOTMOTILIT_SP and FIXED'
)

# Показ графика
fig.show()
```



Вывод: на диаграмме разброса четка видна обратная корреляция части данных.

Машинное обучение для выявления аномалий

In [ ]:
```
df_anomaly = df_cleaned[['SCR_LB_SPERMOGR_LBORRES_TOTMOTILIT_SP', 'SCR_LB_SPERMOGR_LBORRES_FIXED']].copy()
df_anomaly.shape
```

Out[ ]:
```
(557, 2)
```

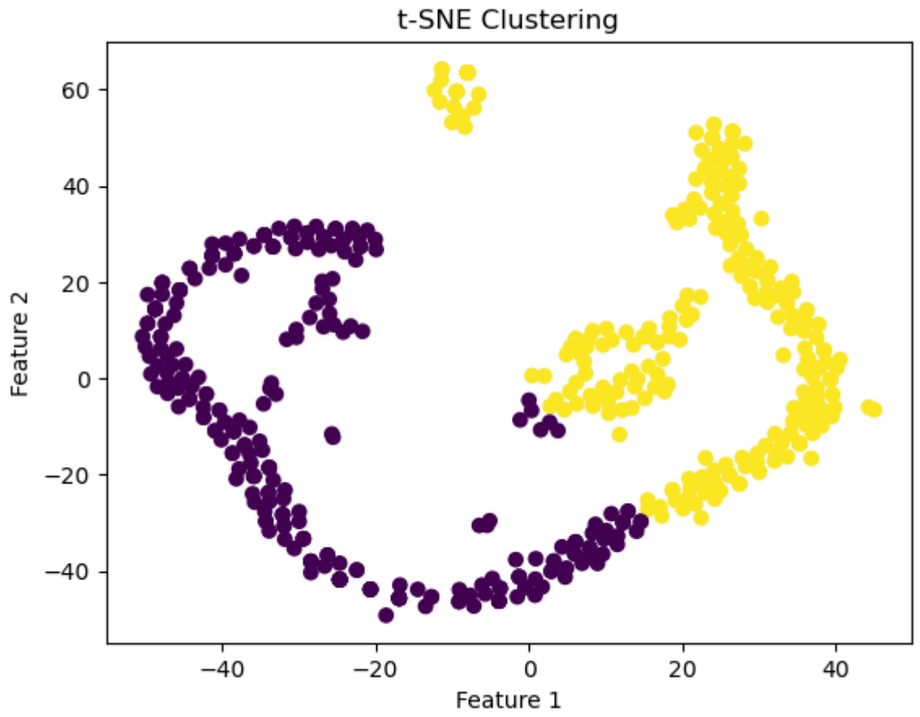
In [ ]:
```
# Удаление строк с NaN значениями для двух столбцов
df_anomaly = df_anomaly.dropna()

# Преобразуем данные в формат, подходящий для sklearn
X = df_anomaly['SCR_LB_SPERMOGR_LBORRES_TOTMOTILIT_SP'].values.reshape(-1, 1)
y = df_anomaly['SCR_LB_SPERMOGR_LBORRES_FIXED'].values

# Создаем и обучаем модель
model = LinearRegression().fit(X, y)

# Вычисляем предсказанные значения
y_pred = model.predict(X)

# Вычисляем ошибки (расстояния от точек до линии регрессии)
errors = np.abs(y - y_pred)

# Определяем порог ошибки для определения "сильной" корреляции
threshold = errors.mean()  # Например, средняя ошибка
```

```
# Получаем индексы пациентов с ошибкой ниже порога
strong_correlation_indices = errors < threshold

# Используем индексы DataFrame для получения id пациентов с сильной корреляцией
patients_with_strong_correlation = df_anomaly.index[strong_correlation_indices]

# Выводим список пациентов
print(patients_with_strong_correlation.tolist())
```

```
['001-001', '001-002', '001-005', '001-010', '001-012', '001-017', '001-018', '001-019', '001-027', '001-034', '001-036', '001-038', '001-039', '001-040', '001-041', '001-042', '001-043', '001-
044', '001-046', '001-047', '001-048', '001-049', '002-001', '004-001', '004-002', '004-003', '004-005', '004-006', '004-007', '006-004', '006-007', '006-009', '006-011', '006-014', '006-016',
'006-017', '006-019', '008-003', '008-004', '008-005', '008-006', '008-007', '008-008', '008-009', '008-013', '008-015', '008-016', '008-017', '008-019', '008-020', '008-021', '008-024', '008-0
26', '008-027', '008-028', '008-029', '008-030', '008-031', '008-032', '008-038', '008-039', '010-004', '012-001', '012-002', '012-003', '012-004', '012-006', '012-007', '012-008', '012-009',
'012-011', '012-012', '012-013', '012-014', '012-015', '013-003', '014-002', '014-004', '014-006', '014-007', '014-008', '014-011', '014-012', '014-014', '014-016', '014-018', '014-019', '014-0
20', '014-021', '014-022', '014-023', '014-024', '014-025', '014-028', '014-029', '014-030', '014-031', '014-032', '014-033', '014-034', '014-035', '014-036', '014-037', '014-038', '014-042',
'014-044', '014-045', '014-047', '014-048', '014-049', '014-050', '014-051', '014-053', '014-054', '014-057', '014-058', '014-059', '014-060', '014-061', '014-062', '014-065', '014-066', '014-0
67', '014-068', '014-071', '014-073', '014-074', '014-075', '014-077', '014-078', '014-079', '014-083', '014-084', '014-085', '014-086', '014-089', '014-090', '014-091', '014-093', '014-094',
'014-096', '014-097', '014-098', '014-100', '014-101', '014-102', '014-103', '014-104', '014-105', '014-106', '014-108', '014-109', '014-110', '014-115', '014-116', '014-118', '014-119', '014-1
20', '014-122', '014-124', '014-125', '014-126', '014-128', '014-130', '014-132', '014-135', '014-136', '014-137', '014-138', '014-139', '014-140', '014-141', '014-142', '014-143', '014-144',
'014-145', '014-147', '014-148', '014-149', '014-150', '014-151', '014-152', '014-153', '014-154', '014-156', '014-157', '014-158', '014-161', '016-005', '017-003', '017-006', '017-007', '017-0
08', '017-010', '017-011', '017-012', '017-013', '017-014', '017-016', '017-017', '017-020', '018-001', '018-003', '018-005', '018-020', '018-021', '018-022', '018-024', '018-025', '018-027',
'018-028', '018-029', '018-030', '018-031', '018-032', '018-034', '018-035', '018-036', '018-037', '018-039', '018-040', '018-044', '018-045', '018-046', '018-049', '018-050', '019-001', '021-0
01', '021-002', '021-003', '029-004', '030-010', '030-013', '030-014', '030-016', '031-008', '031-011', '031-013', '031-014', '031-015', '031-016', '031-017', '031-018', '031-019', '031-020',
'031-021', '031-022', '031-023', '032-001', '032-006', '032-007', '032-008', '032-009', '041-001', '041-003', '041-005', '041-018', '041-019', '041-020', '041-021', '042-001', '042-003', '042-0
04', '042-005', '042-007', '042-008', '042-012', '042-014', '042-016', '042-017', '042-020', '042-021', '042-022', '042-023', '042-024', '042-025', '042-026', '043-001', '043-003', '043-005',
'043-007', '044-002', '044-003', '044-006', '044-007', '044-010', '044-011', '044-012', '044-013', '044-014', '044-015', '044-016', '044-017', '044-018', '044-020', '044-021', '044-023', '046-0
01', '046-002', '046-003', '046-010', '046-011', '046-015', '046-017', '049-003', '049-004', '049-007', '049-010', '049-014', '049-015', '049-016', '049-017', '049-018', '049-019', '050-001',
'051-001', '051-002', '051-003', '051-004', '053-001', '053-002', '053-003', '054-004', '057-002', '057-003', '057-004', '057-005', '057-006', '057-007', '057-008', '057-009', '057-010', '057-0
11', '057-012', '057-013', '057-014', '057-016', '057-021', '059-001', '059-003', '059-005', '059-006', '061-003', '061-004', '061-005', '061-006', '063-001', '063-002', '063-003', '063-004',
'063-005', '063-006', '063-007', '063-008', '063-009', '063-010', '063-011', '063-012', '063-013', '063-014', '063-015', '071-001', '071-002', '071-003']
```

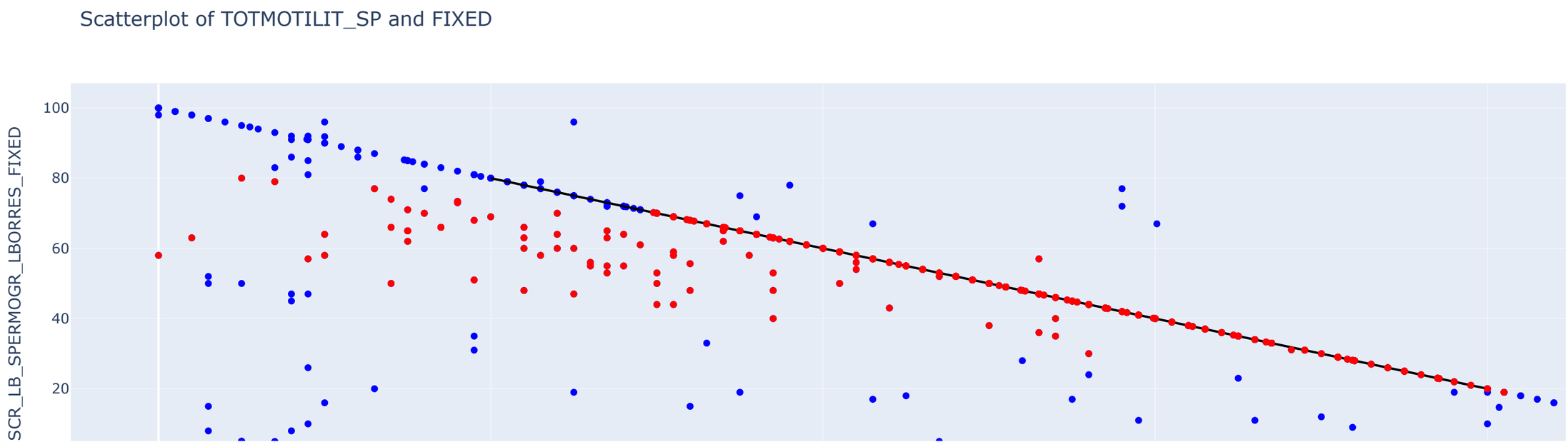
In [ ]:

```
from sklearn.manifold import TSNE
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Perform t-SNE dimensionality reduction
tsne = TSNE(n_components=2, random_state=42)
tsne_df = tsne.fit_transform(df_anomaly)

# Perform K-Means clustering with 2 clusters
kmeans = KMeans(n_clusters=2, random_state=42)
kmeans_labels = kmeans.fit_predict(tsne_df)

# Plot the clusters
plt.scatter(tsne_df[:, 0], tsne_df[:, 1], c=kmeans_labels)
plt.title('t-SNE Clustering')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.show()
```



In [ ]:
```python
# Define the line equation: y = -x + 100
def line_eq(x):
    return -x + 100

# Calculate the y-values for the line
x_line = np.array([20, 80])
y_line = line_eq(x_line)

# Create the main scatterplot
fig = px.scatter(df_cleaned, x='SCR_LB_SPERMOGR_LBORRES_TOTMOTILIT_SP', y='SCR_LB_SPERMOGR_LBORRES_FIXED',
                 color_discrete_sequence=['blue'],  # синий цвет для основного слоя
                 title='Scatterplot of TOTMOTILIT_SP and FIXED')

# Add a line to the plot
fig.add_scatter(x=x_line, y=y_line, mode='lines', line=dict(color='black', width=2), name='Линия тренда')

# Identify points in df_anomaly that lie on the line with a tolerance of +- 1
tolerance = 1
df_anomaly_on_line = df_anomaly[np.abs(df_anomaly['SCR_LB_SPERMOGR_LBORRES_FIXED'] - line_eq(df_anomaly['SCR_LB_SPERMOGR_LBORRES_TOTMOTILIT_SP'])) <= tolerance]

# Save the patients who lie on the line to a list
patients_on_line = df_anomaly_on_line.index.tolist()

# # Add yellow points for the points on the line
# fig.add_scatter(x=df_anomaly_on_line['SCR_LB_SPERMOGR_LBORRES_TOTMOTILIT_SP'],
#                 y=df_anomaly_on_line['SCR_LB_SPERMOGR_LBORRES_FIXED'],
#                 mode='markers', marker=dict(color='yellow'), name='Ручная привязка пациентов к линии тренда')

# Add red points for patients with strong correlation
fig.add_scatter(x=df_cleaned.loc[patients_with_strong_correlation, 'SCR_LB_SPERMOGR_LBORRES_TOTMOTILIT_SP'],
                y=df_cleaned.loc[patients_with_strong_correlation, 'SCR_LB_SPERMOGR_LBORRES_FIXED'],
                mode='markers', marker=dict(color='red'), name='Linear Regression predict')
```

```
# Show the plot
fig.show()
```



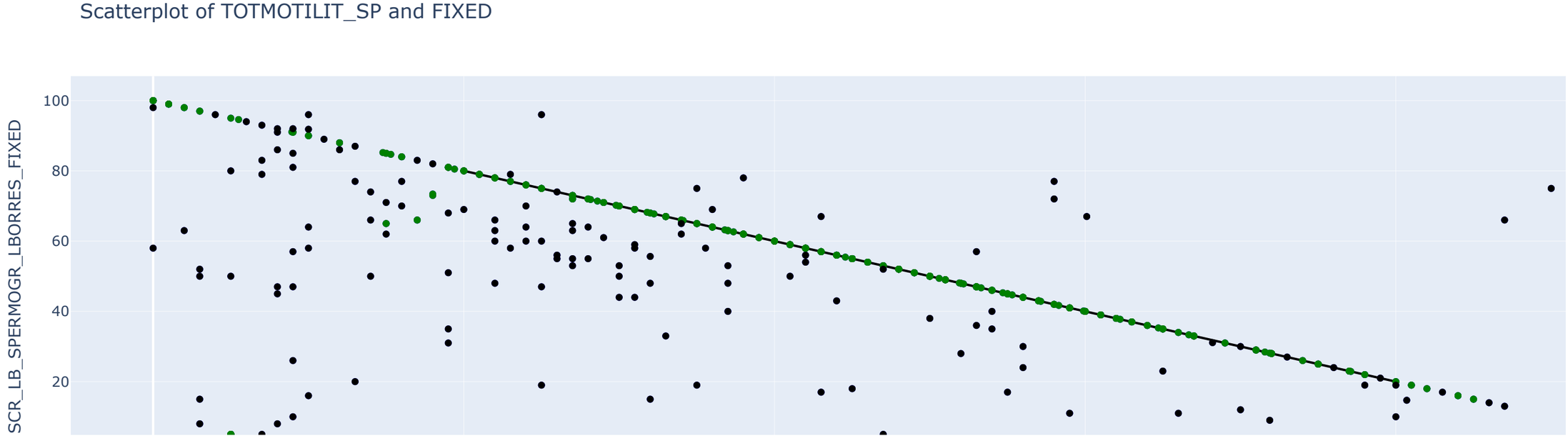
In [ ]:
```
# Perform DBSCAN clustering with 2 clusters
dbscan = DBSCAN(eps=0.7, min_samples=2)
dbscan_labels = dbscan.fit_predict(df_anomaly)

# Replace all cluster labels except -1 with 1
dbscan_labels_binary = np.where(dbscan_labels == -1, -1, 1)

# Create a DataFrame with patient IDs and cluster assignments
cluster_assignments = pd.DataFrame({
    'Patient ID': df_anomaly.index,
    'Cluster': dbscan_labels_binary})
cluster_assignments.set_index('Patient ID', inplace=True)

cluster_assignments = cluster_assignments.assign(
    SCR_LB_SPERMOGR_LBORRES_TOTMOTILIT_SP=df_cleaned['SCR_LB_SPERMOGR_LBORRES_TOTMOTILIT_SP'],
    SCR_LB_SPERMOGR_LBORRES_FIXED=df_cleaned['SCR_LB_SPERMOGR_LBORRES_FIXED']
)
```

In [ ]:
```
cluster_assignments.Cluster.value_counts()
```

Out[ ]:
```
Cluster
 1    411
-1    137
Name: count, dtype: int64
```

In [ ]:
```
# Define the line equation: y = -x + 100
def line_eq(x):
```

```
    return -x + 100

# Calculate the y-values for the line
x_line = np.array([20, 80])
y_line = line_eq(x_line)

# Create the main scatterplot
fig = px.scatter(df_cleaned, x='SCR_LB_SPERMOGR_LBORRES_TOTMOTILIT_SP', y='SCR_LB_SPERMOGR_LBORRES_FIXED',
                 color_discrete_sequence=['blue'],  # синий цвет для основного слоя
                 title='Scatterplot of TOTMOTILIT_SP and FIXED')

# Add a line to the plot
fig.add_scatter(x=x_line, y=y_line, mode='lines', line=dict(color='black', width=2), name='Линия тренда')


# Add green points for patients from BDSCAN clusters == 1
ca_1 = cluster_assignments[cluster_assignments.Cluster == 1]
fig.add_scatter(x=ca_1['SCR_LB_SPERMOGR_LBORRES_TOTMOTILIT_SP'],
                y=ca_1['SCR_LB_SPERMOGR_LBORRES_FIXED'],
                mode='markers', marker=dict(color='green'), name='from DBSCAN clusters == 1 (Аномалии)')

# Add black points for patients from BDSCAN clusters == -1
ca_norm = cluster_assignments[cluster_assignments.Cluster == -1]
fig.add_scatter(x=ca_norm['SCR_LB_SPERMOGR_LBORRES_TOTMOTILIT_SP'],
                y=ca_norm['SCR_LB_SPERMOGR_LBORRES_FIXED'],
                mode='markers', marker=dict(color='black'), name='from DBSCAN clusters == -1 (Норма)')

# Show the plot
fig.show()
```



Шаг 04 Общий вывод

Какие именно данные, возможно, были сфальсифицированы

Список типов исследований с возможной фальсификацией

- 'SCR_LB_SPERMOGR_LBORRES_TOTMOTILIT_SP',
- 'SCR_LB_SPERMOGR_LBORRES_FIXED'

In [ ]:
```
print(f"Количество наблюдений, где анализ спермограммы был сделан до визита: {count_observations_before_visit}")
```

```
Количество наблюдений, где анализ спермограммы был сделан до визита: 101
```

Отметить номера пациентов, для которых данные, возможно, были сфальсифицированы.

In [ ]:
```
len(ca_1), len(df_anomaly_on_line)
```

Out[ ]:
```
(411, 401)
```

In [ ]:
```
ca_1.to_csv('fraud_by_clustering.csv', index=True)
df_anomaly_on_line.to_csv('fraud_by_visual.csv', index=True)
```