screen time and relation-checkpoint

April 13, 2025

1 Screen time and Parent-child relationship

In todays modern world, screen use is very common in every household. Nowadays, children use screen more as compared to adults. This screen use has positive and negative consequences. Screen use in children help them in learning, play and to become creative. However, the excessive use of screen decrease problem solving ability, sleep disturbace and obesity.

Parent-child interaction is important during early childhood. Children mostly look to their parents for affection, reassurance, guidance and direction. But the use of media affect these interactions, such as parent's give less responses and attention to the children in the presence of screen. If parents have positive attitude towards screen, then children use more screen media and parent's media use is associated with children's media use.

2 Hypothesis

H1: Children's screen use has direct relation with parent's screen use.

H2: Parent screen use is negatively related to relationship quality and child use is negatively related to relationship quality.

2.1 Importing library

[1]:
```
# import library

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats
from scipy.stats import pearsonr
```

2.2 Importing data

[3]:
```
data = pd.read_csv("C:/Users/aiman/OneDrive/Documents/PSY-3035 programming and␣
↪data visualization/exams data/screen time and relation.csv")
```

[4]:
```
data
```

[4]:

	Age of child in months	parent's screen time	child's screen time	screen time together	relationship_conflict	relationship_closeness
0	72	1.50	3.50	6.00	1.89	3.63
1	82	1.00	2.00	1.00	2.11	3.75
2	64	3.00	1.00	1.00	1.44	4.38
3	76	2.00	1.00	5.00	2.33	3.50
4	83	2.50	3.50	5.00	2.33	4.38
5	71	1.50	0.50	0.25	2.78	4.25
6	38	6.00	2.00	3.00	2.33	3.13
7	37	1.00	1.00	1.00	2.22	0.00
8	78	3.00	1.00	1.00	1.67	3.88
9	45	1.00	1.00	1.00	1.78	3.75
10	36	5.00	1.00	1.00	2.11	3.50
11	79	3.00	1.00	1.00	1.78	4.38
12	37	1.00	1.00	1.00	1.89	4.25
13	64	1.50	2.50	1.50	1.67	4.25
14	81	5.00	5.00	0.00	2.22	4.38
15	49	2.50	2.50	1.00	2.78	4.38
16	36	2.50	1.00	3.50	2.11	4.38
17	70	0.29	0.63	1.00	1.67	4.38
18	53	2.00	1.00	2.00	2.33	0.00
19	43	5.00	6.00	5.00	2.22	3.88
20	50	2.00	1.00			
21	50	2.00	0.00			
22	70	1.00	1.00			
23	78	1.00	1.00			

20	1.00	2.33	4.13
21	3.00	2.33	3.63
22	0.00	3.22	4.13
23	1.50	1.22	4.25

[42]:
```
np.loadtxt?
```

```
Signature:
np.loadtxt(
    fname,
    dtype=<class 'float'>,
    comments='#',
    delimiter=None,
    converters=None,
    skiprows=0,
    usecols=None,
    unpack=False,
    ndmin=0,
    encoding='bytes',
    max_rows=None,
    *,
    quotechar=None,
    like=None,
)
Docstring:
Load data from a text file.

Parameters
----------
fname : file, str, pathlib.Path, list of str, generator
    File, filename, list, or generator to read.  If the filename
    extension is ``.gz`` or ``.bz2``, the file is first decompressed. Note
    that generators must return bytes or strings. The strings
    in a list or produced by a generator are treated as lines.
dtype : data-type, optional
    Data-type of the resulting array; default: float.  If this is a
    structured data-type, the resulting array will be 1-dimensional, and
    each row will be interpreted as an element of the array.  In this
    case, the number of columns used must match the number of fields in
    the data-type.
```

```
comments : str or sequence of str or None, optional
    The characters or list of characters used to indicate the start of a
    comment. None implies no comments. For backwards compatibility, byte
    strings will be decoded as 'latin1'. The default is '#'.
delimiter : str, optional
    The character used to separate the values. For backwards compatibility,
    byte strings will be decoded as 'latin1'. The default is whitespace.

    .. versionchanged:: 1.23.0
       Only single character delimiters are supported. Newline characters
       cannot be used as the delimiter.

converters : dict or callable, optional
    Converter functions to customize value parsing. If `converters` is
    callable, the function is applied to all columns, else it must be a
    dict that maps column number to a parser function.
    See examples for further details.
    Default: None.

    .. versionchanged:: 1.23.0
       The ability to pass a single callable to be applied to all columns
       was added.

skiprows : int, optional
    Skip the first `skiprows` lines, including comments; default: 0.
usecols : int or sequence, optional
    Which columns to read, with 0 being the first. For example,
    ``usecols = (1,4,5)`` will extract the 2nd, 5th and 6th columns.
    The default, None, results in all columns being read.

    .. versionchanged:: 1.11.0
        When a single column has to be read it is possible to use
        an integer instead of a tuple. E.g ``usecols = 3`` reads the
        fourth column the same way as ``usecols = (3,)`` would.
unpack : bool, optional
    If True, the returned array is transposed, so that arguments may be
    unpacked using ``x, y, z = loadtxt(…)``.  When used with a
    structured data-type, arrays are returned for each field.
    Default is False.
ndmin : int, optional
    The returned array will have at least `ndmin` dimensions.
    Otherwise mono-dimensional axes will be squeezed.
    Legal values: 0 (default), 1 or 2.

    .. versionadded:: 1.6.0
encoding : str, optional
    Encoding used to decode the inputfile. Does not apply to input streams.
    The special value 'bytes' enables backward compatibility workarounds
```

```
    that ensures you receive byte arrays as results if possible and passes
    'latin1' encoded strings to converters. Override this value to receive
    unicode arrays and pass strings as input to converters.  If set to None
    the system default is used. The default value is 'bytes'.

    .. versionadded:: 1.14.0
max_rows : int, optional
    Read `max_rows` rows of content after `skiprows` lines. The default is
    to read all the rows. Note that empty rows containing no data such as
    empty lines and comment lines are not counted towards `max_rows`,
    while such lines are counted in `skiprows`.

    .. versionadded:: 1.16.0

    .. versionchanged:: 1.23.0
        Lines containing no data, including comment lines (e.g., lines
        starting with '#' or as specified via `comments`) are not counted
        towards `max_rows`.
quotechar : unicode character or None, optional
    The character used to denote the start and end of a quoted item.
    Occurrences of the delimiter or comment characters are ignored within
    a quoted item. The default value is ``quotechar=None``, which means
    quoting support is disabled.

    If two consecutive instances of `quotechar` are found within a quoted
    field, the first is treated as an escape character. See examples.

    .. versionadded:: 1.23.0
like : array_like, optional
    Reference object to allow the creation of arrays which are not
    NumPy arrays. If an array-like passed in as ``like`` supports
    the ``__array_function__`` protocol, the result will be defined
    by it. In this case, it ensures the creation of an array object
    compatible with that passed in via this argument.

    .. versionadded:: 1.20.0

Returns
-------
out : ndarray
    Data read from the text file.

See Also
--------
load, fromstring, fromregex
genfromtxt : Load data with missing values handled as specified.
scipy.io.loadmat : reads MATLAB data files
```

```
Notes
-----
This function aims to be a fast reader for simply formatted files.  The
`genfromtxt` function provides more sophisticated handling of, e.g.,
lines with missing values.

Each row in the input text file must have the same number of values to be
able to read all values. If all rows do not have same number of values, a
subset of up to n columns (where n is the least number of values present
in all rows) can be read by specifying the columns via `usecols`.

.. versionadded:: 1.10.0

The strings produced by the Python float.hex method can be used as
input for floats.

Examples
--------
>>> from io import StringIO   # StringIO behaves like a file object
>>> c = StringIO("0 1\n2 3")
>>> np.loadtxt(c)
array([[0., 1.],
       [2., 3.]])

>>> d = StringIO("M 21 72\nF 35 58")
>>> np.loadtxt(d, dtype={'names': ('gender', 'age', 'weight'),
…                      'formats': ('S1', 'i4', 'f4')})
array([(b'M', 21, 72.), (b'F', 35, 58.)],
      dtype=[('gender', 'S1'), ('age', '<i4'), ('weight', '<f4')])

>>> c = StringIO("1,0,2\n3,0,4")
>>> x, y = np.loadtxt(c, delimiter=',', usecols=(0, 2), unpack=True)
>>> x
array([1., 3.])
>>> y
array([2., 4.])

The `converters` argument is used to specify functions to preprocess the
text prior to parsing. `converters` can be a dictionary that maps
preprocessing functions to each column:

>>> s = StringIO("1.618, 2.296\n3.141, 4.669\n")
>>> conv = {
…     0: lambda x: np.floor(float(x)),  # conversion fn for column 0
…     1: lambda x: np.ceil(float(x)),  # conversion fn for column 1
… }
>>> np.loadtxt(s, delimiter=",", converters=conv)
array([[1., 3.],
```

```
       [3., 5.]])

`converters` can be a callable instead of a dictionary, in which case it
is applied to all columns:

>>> s = StringIO("0xDE 0xAD\n0xC0 0xDE")
>>> import functools
>>> conv = functools.partial(int, base=16)
>>> np.loadtxt(s, converters=conv)
array([[222., 173.],
       [192., 222.]])

This example shows how `converters` can be used to convert a field
with a trailing minus sign into a negative number.

>>> s = StringIO('10.01 31.25-\n19.22 64.31\n17.57- 63.94')
>>> def conv(fld):
…     return -float(fld[:-1]) if fld.endswith(b'-') else float(fld)
…
>>> np.loadtxt(s, converters=conv)
array([[ 10.01, -31.25],
       [ 19.22,  64.31],
       [-17.57,  63.94]])

Using a callable as the converter can be particularly useful for handling
values with different formatting, e.g. floats with underscores:

>>> s = StringIO("1 2.7 100_000")
>>> np.loadtxt(s, converters=float)
array([1.e+00, 2.7e+00, 1.e+05])

This idea can be extended to automatically handle values specified in
many different formats:

>>> def conv(val):
…     try:
…         return float(val)
…     except ValueError:
…         return float.fromhex(val)
>>> s = StringIO("1, 2.5, 3_000, 0b4, 0x1.4000000000000p+2")
>>> np.loadtxt(s, delimiter=",", converters=conv, encoding=None)
array([1.0e+00, 2.5e+00, 3.0e+03, 1.8e+02, 5.0e+00])

Note that with the default ``encoding="bytes"``, the inputs to the
converter function are latin-1 encoded byte strings. To deactivate the
implicit encoding prior to conversion, use ``encoding=None``

>>> s = StringIO('10.01 31.25-\n19.22 64.31\n17.57- 63.94')
```

```
>>> conv = lambda x: -float(x[:-1]) if x.endswith('-') else float(x)
>>> np.loadtxt(s, converters=conv, encoding=None)
array([[ 10.01, -31.25],
       [ 19.22,  64.31],
       [-17.57,  63.94]])

Support for quoted fields is enabled with the `quotechar` parameter.
Comment and delimiter characters are ignored when they appear within a
quoted item delineated by `quotechar`:

>>> s = StringIO('"alpha, #42", 10.0\n"beta, #64", 2.0\n')
>>> dtype = np.dtype([("label", "U12"), ("value", float)])
>>> np.loadtxt(s, dtype=dtype, delimiter=",", quotechar='"')
array([('alpha, #42', 10.), ('beta, #64',  2.)],
      dtype=[('label', '<U12'), ('value', '<f8')])

Quoted fields can be separated by multiple whitespace characters:

>>> s = StringIO('"alpha, #42"       10.0\n"beta, #64" 2.0\n')
>>> dtype = np.dtype([("label", "U12"), ("value", float)])
>>> np.loadtxt(s, dtype=dtype, delimiter=None, quotechar='"')
array([('alpha, #42', 10.), ('beta, #64',  2.)],
      dtype=[('label', '<U12'), ('value', '<f8')])

Two consecutive quote characters within a quoted field are treated as a
single escaped character:

>>> s = StringIO('"Hello, my name is ""Monty""!"')
>>> np.loadtxt(s, dtype="U", delimiter=",", quotechar='"')
array('Hello, my name is "Monty"!', dtype='<U26')

Read subset of columns when all rows do not contain equal number of values:

>>> d = StringIO("1 2\n2 4\n3 9 12\n4 16 20")
>>> np.loadtxt(d, usecols=(0, 1))
array([[ 1.,  2.],
       [ 2.,  4.],
       [ 3.,  9.],
       [ 4., 16.]])
File:      c:\users\aiman\anaconda3\lib\site-packages\numpy\lib\npyio.py
Type:      function
```

[44]:
```
type(data)
```

[44]:
```
pandas.core.frame.DataFrame
```

[13]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 24 entries, 0 to 23
Data columns (total 6 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Age of child in months  24 non-null     int64
 1   parent's screen time    24 non-null     float64
 2   child's screen time     24 non-null     float64
 3   screen time together    24 non-null     float64
 4   relationship_conflict   24 non-null     float64
 5   relationship_closeness  24 non-null     float64
dtypes: float64(5), int64(1)
memory usage: 1.3 KB
```

[46]:
```
data.columns
```

[46]:
```
Index(['Age of child in months', 'parent's screen time', 'child's screen time',
       'screen time together', 'relationship_conflict',
       'relationship_closeness'],
      dtype='object')
```

[48]:
```
dir(data)
```

[48]:
```
['T',
 '_AXIS_LEN',
 '_AXIS_ORDERS',
 '_AXIS_TO_AXIS_NUMBER',
 '_HANDLED_TYPES',
 '__abs__',
 '__add__',
 '__and__',
 '__annotations__',
 '__array__',
 '__array_priority__',
 '__array_ufunc__',
 '__arrow_c_stream__',
 '__bool__',
 '__class__',
 '__contains__',
 '__copy__',
 '__dataframe__',
 '__dataframe_consortium_standard__',
 '__deepcopy__',
 '__delattr__',
 '__delitem__',
 '__dict__',
 '__dir__',
 '__divmod__',
```

```
'__doc__',
'__eq__',
'__finalize__',
'__floordiv__',
'__format__',
'__ge__',
'__getattr__',
'__getattribute__',
'__getitem__',
'__getstate__',
'__gt__',
'__hash__',
'__iadd__',
'__iand__',
'__ifloordiv__',
'__imod__',
'__imul__',
'__init__',
'__init_subclass__',
'__invert__',
'__ior__',
'__ipow__',
'__isub__',
'__iter__',
'__itruediv__',
'__ixor__',
'__le__',
'__len__',
'__lt__',
'__matmul__',
'__mod__',
'__module__',
'__mul__',
'__ne__',
'__neg__',
'__new__',
'__nonzero__',
'__or__',
'__pandas_priority__',
'__pos__',
'__pow__',
'__radd__',
'__rand__',
'__rdivmod__',
'__reduce__',
'__reduce_ex__',
'__repr__',
```

```
'__rfloordiv__',
'__rmatmul__',
'__rmod__',
'__rmul__',
'__ror__',
'__round__',
'__rpow__',
'__rsub__',
'__rtruediv__',
'__rxor__',
'__setattr__',
'__setitem__',
'__setstate__',
'__sizeof__',
'__str__',
'__sub__',
'__subclasshook__',
'__truediv__',
'__weakref__',
'__xor__',
'_accessors',
'_accum_func',
'_agg_examples_doc',
'_agg_see_also_doc',
'_align_for_op',
'_align_frame',
'_align_series',
'_append',
'_arith_method',
'_arith_method_with_reindex',
'_as_manager',
'_attrs',
'_box_col_values',
'_can_fast_transpose',
'_check_inplace_and_allows_duplicate_labels',
'_check_is_chained_assignment_possible',
'_check_label_or_level_ambiguity',
'_check_setitem_copy',
'_clear_item_cache',
'_clip_with_one_bound',
'_clip_with_scalar',
'_cmp_method',
'_combine_frame',
'_consolidate',
'_consolidate_inplace',
'_construct_axes_dict',
'_construct_result',
```

```
'_constructor',
'_constructor_from_mgr',
'_constructor_sliced',
'_constructor_sliced_from_mgr',
'_create_data_for_split_and_tight_to_dict',
'_data',
'_deprecate_downcast',
'_dir_additions',
'_dir_deletions',
'_dispatch_frame_op',
'_drop_axis',
'_drop_labels_or_levels',
'_ensure_valid_index',
'_find_valid_index',
'_flags',
'_flex_arith_method',
'_flex_cmp_method',
'_from_arrays',
'_from_mgr',
'_get_agg_axis',
'_get_axis',
'_get_axis_name',
'_get_axis_number',
'_get_axis_resolvers',
'_get_block_manager_axis',
'_get_bool_data',
'_get_cleaned_column_resolvers',
'_get_column_array',
'_get_index_resolvers',
'_get_item_cache',
'_get_label_or_level_values',
'_get_numeric_data',
'_get_value',
'_get_values_for_csv',
'_getitem_bool_array',
'_getitem_multilevel',
'_getitem_nocopy',
'_getitem_slice',
'_gotitem',
'_hidden_attrs',
'_indexed_same',
'_info_axis',
'_info_axis_name',
'_info_axis_number',
'_info_repr',
'_init_mgr',
'_inplace_method',
```

```
'_internal_names',
'_internal_names_set',
'_is_copy',
'_is_homogeneous_type',
'_is_label_or_level_reference',
'_is_label_reference',
'_is_level_reference',
'_is_mixed_type',
'_is_view',
'_is_view_after_cow_rules',
'_iset_item',
'_iset_item_mgr',
'_iset_not_inplace',
'_item_cache',
'_iter_column_arrays',
'_ixs',
'_logical_func',
'_logical_method',
'_maybe_align_series_as_frame',
'_maybe_cache_changed',
'_maybe_update_cacher',
'_metadata',
'_mgr',
'_min_count_stat_function',
'_needs_reindex_multi',
'_pad_or_backfill',
'_protect_consolidate',
'_reduce',
'_reduce_axis1',
'_reindex_axes',
'_reindex_multi',
'_reindex_with_indexers',
'_rename',
'_replace_columnwise',
'_repr_data_resource_',
'_repr_fits_horizontal_',
'_repr_fits_vertical_',
'_repr_html_',
'_repr_latex_',
'_reset_cache',
'_reset_cacher',
'_sanitize_column',
'_series',
'_set_axis',
'_set_axis_name',
'_set_axis_nocheck',
'_set_is_copy',
```

```
'_set_item',
'_set_item_frame_value',
'_set_item_mgr',
'_set_value',
'_setitem_array',
'_setitem_frame',
'_setitem_slice',
'_shift_with_freq',
'_should_reindex_frame_op',
'_slice',
'_stat_function',
'_stat_function_ddof',
'_take_with_is_copy',
'_to_dict_of_blocks',
'_to_latex_via_styler',
'_typ',
'_update_inplace',
'_validate_dtype',
'_values',
'_where',
'abs',
'add',
'add_prefix',
'add_suffix',
'agg',
'aggregate',
'align',
'all',
'any',
'apply',
'applymap',
'asfreq',
'asof',
'assign',
'astype',
'at',
'at_time',
'attrs',
'axes',
'backfill',
'between_time',
'bfill',
'bool',
'boxplot',
'clip',
'columns',
'combine',
```

```
'combine_first',
'compare',
'convert_dtypes',
'copy',
'corr',
'corrwith',
'count',
'cov',
'cummax',
'cummin',
'cumprod',
'cumsum',
'describe',
'diff',
'div',
'divide',
'dot',
'drop',
'drop_duplicates',
'droplevel',
'dropna',
'dtypes',
'duplicated',
'empty',
'eq',
'equals',
'eval',
'ewm',
'expanding',
'explode',
'ffill',
'fillna',
'filter',
'first',
'first_valid_index',
'flags',
'floordiv',
'from_dict',
'from_records',
'ge',
'get',
'groupby',
'gt',
'head',
'hist',
'iat',
'idxmax',
```

```
'idxmin',
'iloc',
'index',
'infer_objects',
'info',
'insert',
'interpolate',
'isetitem',
'isin',
'isna',
'isnull',
'items',
'iterrows',
'itertuples',
'join',
'keys',
'kurt',
'kurtosis',
'last',
'last_valid_index',
'le',
'loc',
'lt',
'map',
'mask',
'max',
'mean',
'median',
'melt',
'memory_usage',
'merge',
'min',
'mod',
'mode',
'mul',
'multiply',
'ndim',
'ne',
'nlargest',
'notna',
'notnull',
'nsmallest',
'nunique',
'pad',
'pct_change',
'pipe',
'pivot',
```

```
'pivot_table',
'plot',
'pop',
'pow',
'prod',
'product',
'quantile',
'query',
'radd',
'rank',
'rdiv',
'reindex',
'reindex_like',
'relationship_closeness',
'relationship_conflict',
'rename',
'rename_axis',
'reorder_levels',
'replace',
'resample',
'reset_index',
'rfloordiv',
'rmod',
'rmul',
'rolling',
'round',
'rpow',
'rsub',
'rtruediv',
'sample',
'select_dtypes',
'sem',
'set_axis',
'set_flags',
'set_index',
'shape',
'shift',
'size',
'skew',
'sort_index',
'sort_values',
'squeeze',
'stack',
'std',
'style',
'sub',
'subtract',
```

```
 'sum',
 'swapaxes',
 'swaplevel',
 'tail',
 'take',
 'to_clipboard',
 'to_csv',
 'to_dict',
 'to_excel',
 'to_feather',
 'to_gbq',
 'to_hdf',
 'to_html',
 'to_json',
 'to_latex',
 'to_markdown',
 'to_numpy',
 'to_orc',
 'to_parquet',
 'to_period',
 'to_pickle',
 'to_records',
 'to_sql',
 'to_stata',
 'to_string',
 'to_timestamp',
 'to_xarray',
 'to_xml',
 'transform',
 'transpose',
 'truediv',
 'truncate',
 'tz_convert',
 'tz_localize',
 'unstack',
 'update',
 'value_counts',
 'values',
 'var',
 'where',
 'xs']
```

[50]:
```
data.describe()
```

[50]:
```
       Age of child in months  parent's screen time  child's screen time  \
count               24.000000             24.000000            24.000000
mean                60.083333              2.345417             1.713750
```

```
std                  17.186741             1.526015             1.461987
min                  36.000000             0.290000             0.000000
25%                  44.500000             1.000000             1.000000
50%                  64.000000             2.000000             1.000000
75%                  76.500000             3.000000             2.125000
max                  83.000000             6.000000             6.000000

       screen time together  relationship_conflict  relationship_closeness
count             24.000000              24.000000               24.000000
mean               1.947917               2.115000                3.690417
std                1.741406               0.446844                1.191982
min                0.000000               1.220000                0.000000
25%                1.000000               1.780000                3.630000
50%                1.000000               2.165000                4.130000
75%                3.000000               2.330000                4.380000
max                6.000000               3.220000                4.380000
```

[19]:
```
data.shape
```

[19]:
```
(24, 6)
```

3 Correlation

Pearson's correlation is calculated below.

[8]:
```
data.corr().round(2)
```

[8]:
```
                        Age of child in months  parent's screen time  \
Age of child in months                    1.00                 -0.22
parent's screen time                     -0.22                  1.00
child's screen time                       0.13                  0.49
screen time together                     -0.02                  0.17
relationship_conflict                    -0.14                  0.09
relationship_closeness                    0.34                  0.07

                        child's screen time  screen time together  \
Age of child in months                 0.13                 -0.02
parent's screen time                   0.49                  0.17
child's screen time                    1.00                  0.40
screen time together                   0.40                  1.00
relationship_conflict                  0.07                 -0.02
relationship_closeness                 0.15                 -0.04

                        relationship_conflict  relationship_closeness
Age of child in months                  -0.14                    0.34
parent's screen time                     0.09                    0.07
child's screen time                      0.07                    0.15
```

```
screen time together                -0.02                 -0.04
relationship_conflict                1.00                 -0.14
relationship_closeness              -0.14                  1.00
```

[10]:
```
corr_matrix = data.corr()
print(corr_matrix)
```

```
                        Age of child in months  parent's screen time  \
Age of child in months                1.000000             -0.220963
parent's screen time                 -0.220963              1.000000
child's screen time                   0.130421              0.486535
screen time together                 -0.019097              0.172066
relationship_conflict                -0.140855              0.093790
relationship_closeness                0.338740              0.065293

                        child's screen time  screen time together  \
Age of child in months             0.130421             -0.019097
parent's screen time               0.486535              0.172066
child's screen time                1.000000              0.399399
screen time together               0.399399              1.000000
relationship_conflict              0.071688             -0.016832
relationship_closeness             0.154126             -0.036540

                        relationship_conflict  relationship_closeness
Age of child in months              -0.140855                0.338740
parent's screen time                 0.093790                0.065293
child's screen time                  0.071688                0.154126
screen time together                -0.016832               -0.036540
relationship_conflict                1.000000               -0.136848
relationship_closeness              -0.136848                1.000000
```

[12]:
```
data.corr()['relationship_conflict'].sort_values(ascending=False)
```

[12]:
```
relationship_conflict     1.000000
parent's screen time      0.093790
child's screen time       0.071688
screen time together     -0.016832
relationship_closeness   -0.136848
Age of child in months   -0.140855
Name: relationship_conflict, dtype: float64
```

[14]:
```
high_corr = corr_matrix.abs() > 0.7
print(high_corr)
```

```
                        Age of child in months  parent's screen time  \
Age of child in months                    True                 False
parent's screen time                     False                  True
child's screen time                      False                 False
```

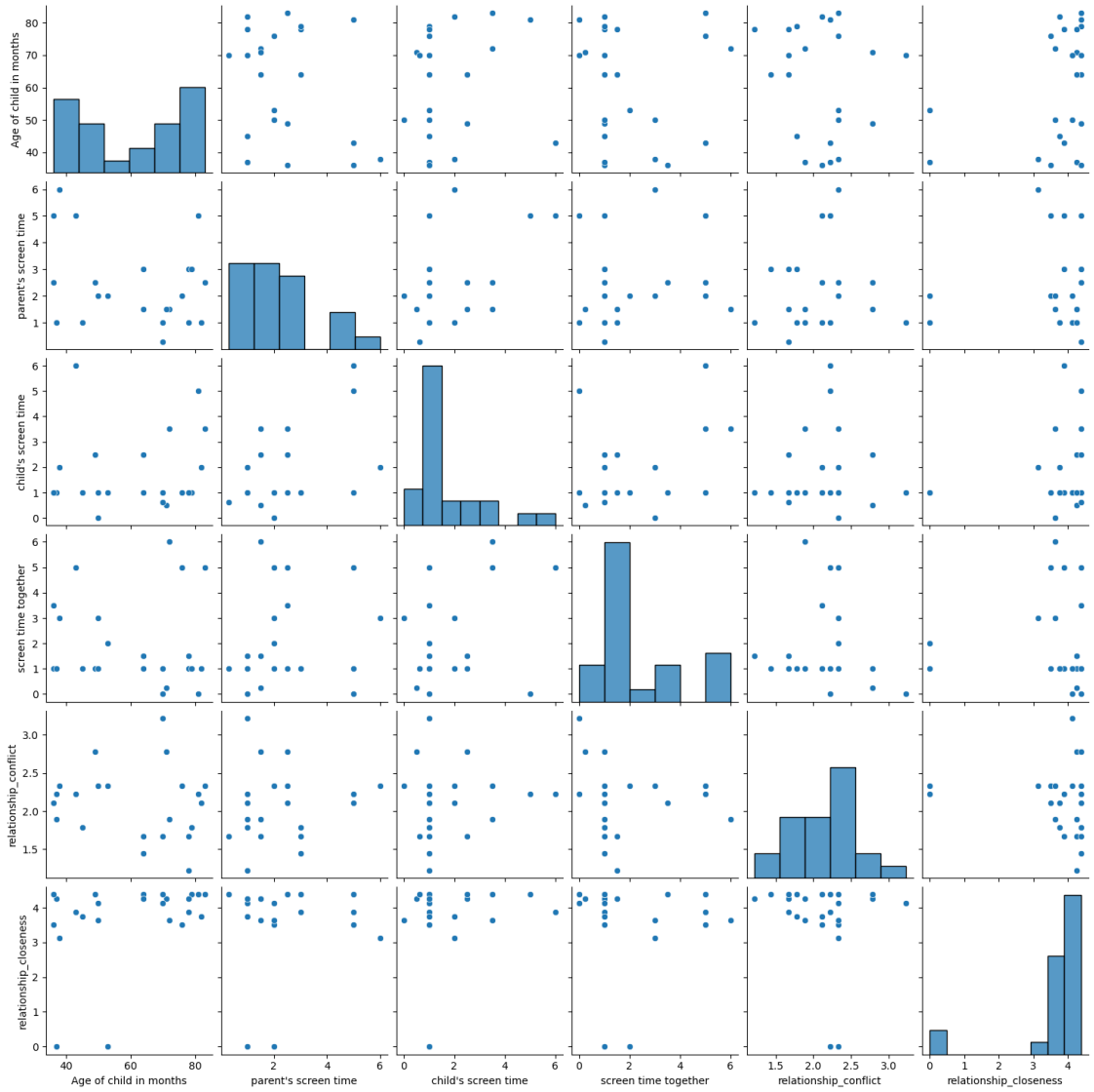
```
screen time together                    False                 False
relationship_conflict                   False                 False
relationship_closeness                  False                 False

                        child's screen time  screen time together  \
Age of child in months                False                 False
parent's screen time                  False                 False
child's screen time                    True                 False
screen time together                  False                  True
relationship_conflict                 False                 False
relationship_closeness                False                 False

                        relationship_conflict  relationship_closeness
Age of child in months                  False                   False
parent's screen time                    False                   False
child's screen time                     False                   False
screen time together                    False                   False
relationship_conflict                    True                   False
relationship_closeness                  False                    True
```

4 Plotting data

[21]:
```
# import seaborn
import seaborn as sns
sns.pairplot(data)
plt.show()
```



5 Heatmap

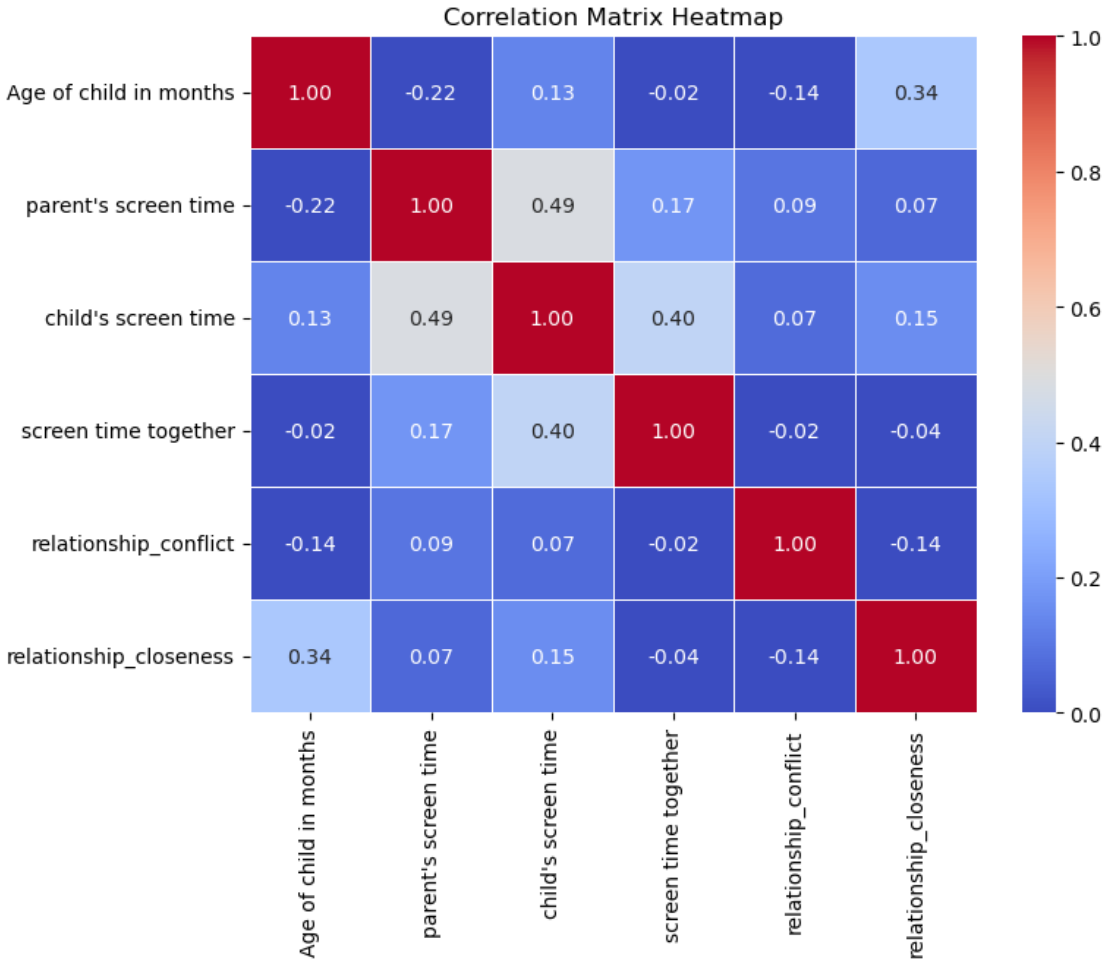
[62]:
```
# Set the figure size
plt.figure(figsize=(8, 6))

# Create correlation matrix in heatmap
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.
 ↪5, vmin=0, vmax=1)



# Display the plot
```

```
plt.title('Correlation Matrix Heatmap')
plt.show()
```

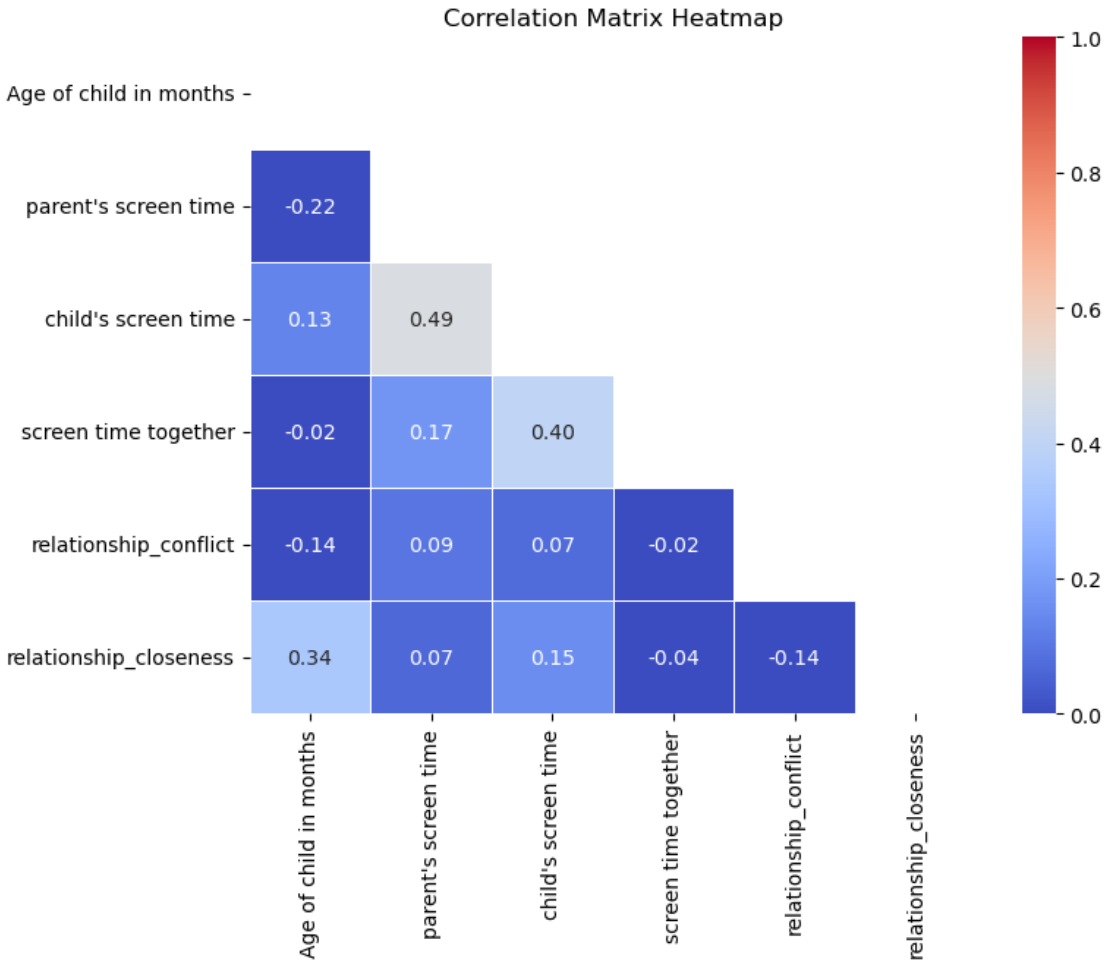


[69]:
```
# Set the figure size
plt.figure(figsize=(8, 6))

# Create correlation matrix in heatmap
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.
 ↪5, vmin=0, vmax=1, mask=my_mask)

my_mask = np.triu(np.ones_like(corr_matrix, dtype=bool))

# Display the plot
plt.title('Correlation Matrix Heatmap')
plt.show()
```



6 Correlation Analysis

H1: Children's screen use has direct relation with parent's screen use.

According to heatmap, children's screen time has a moderate positive correlation (r=0.49) with parent's screen time. This result confirm this hypothesis that as parental screen use increases, children screen use also increases.

H2: Parent screen use is negatively related to relationship quality and child use is negatively related to relationship quality.

The parent's screen time has weak negative correlation with relationship conflict (r=-0.14) and relationship closeness (r=-0.14). The first part of this hypothesis is confirmed that as the parent's screen use increase, the closeness with children become decreases and increase in conflicts occcur.

The children's screen use has a weak but positive correlation with relationship closeness (r=0.07) and relationship conflict (r=0.09). The second part of hypothesis is not confirmed because it doesnot show a negative correlation with relationship quality but shows very weak positive correlation.