2005-2006-run

February 23, 2023

1 Imports

[1]:
```
import subprocess
```

[2]:
```
import altair as alt
import numpy as np
import pandas as pd
import statsmodels.formula.api as smf
from tqdm import tqdm
```

[3]:
```
from utils import CHAR_LOOKUP, flatten_columns, get_datadir
```

[4]:
```
# Create new `pandas` methods which use `tqdm` progress
# (can use tqdm_gui, optional kwargs, etc.)
tqdm.pandas()
```

[5]:
```
alt.data_transformers.enable("data_server")
alt.renderers.enable("altair_saver", fmts=["png"], embed_options={"scaleFactor":
  ↪ "4"})
```

[5]:
```
RendererRegistry.enable('altair_saver')
```

[6]:
```
# allow pandas to show more data
pd.set_option("display.max_columns", None)
pd.set_option("display.max_rows", 1000)
```

2 Select year and load file

[7]:
```
year: str = "2005-2006"
```

[8]:
```
datadir = get_datadir(year)
```

[9]:
```
xptfile = datadir / f"paxraw_{CHAR_LOOKUP[year].lower()}.xpt"
zipfilename = f"PAXRAW_{CHAR_LOOKUP[year].upper()}.ZIP"
zipfile = datadir / zipfilename
if not xptfile.exists():
```

```
    print("no extracted xpt file, looking for the zip")
    if not zipfile.exists():
        print("no zip exists, downloading it")
        subprocess.run(
            [
                "wget",
                "-O",
                zipfile,
                f"https://wwwn.cdc.gov/Nchs/Nhanes/{year}/{zipfilename}",
                "--no-use-server-timestamps",
            ]
        )
    print("extracting")
    subprocess.run(["unzip", "-o", zipfile, "-d", datadir])
```

[10]:
```
paxraw = pd.read_sas(xptfile)
```

[11]:
```
paxraw.shape
```

[11]:
```
(74874095, 9)
```

[12]:
```
paxraw.head()
```

[12]:

	SEQN	PAXSTAT	PAXCAL	PAXDAY	PAXN	PAXHOUR	PAXMINUT	\
0	31128.0	1.0	1.0	1.0	1.0	5.397605e-79	5.397605e-79	
1	31128.0	1.0	1.0	1.0	2.0	5.397605e-79	1.000000e+00	
2	31128.0	1.0	1.0	1.0	3.0	5.397605e-79	2.000000e+00	
3	31128.0	1.0	1.0	1.0	4.0	5.397605e-79	3.000000e+00	
4	31128.0	1.0	1.0	1.0	5.0	5.397605e-79	4.000000e+00	

	PAXINTEN	PAXSTEP
0	1.660000e+02	4.000000e+00
1	2.700000e+01	5.397605e-79
2	5.397605e-79	5.397605e-79
3	2.760000e+02	4.000000e+00
4	5.397605e-79	5.397605e-79

2.1 Fix datatypes

[13]:
```
paxraw.dtypes
```

[13]:
```
SEQN        float64
PAXSTAT     float64
PAXCAL      float64
PAXDAY      float64
PAXN        float64
PAXHOUR     float64
```

```
PAXMINUT    float64
PAXINTEN    float64
PAXSTEP     float64
dtype: object
```

[14]:
```python
for col in paxraw.columns:
    print(f"casting {col=} to int")
    try:
        paxraw.loc[:, col] = paxraw.loc[:, col].astype(int)
    except pd.errors.IntCastingNaNError:
        print(f"{col=} has {paxraw.loc[:, col].isna().sum()} NA values, setting␣
 ↪to 0")
        paxraw.loc[:, col] = paxraw.loc[:, col].replace(np.nan, 0).astype(int)
```

```
casting col='SEQN' to int
```

```
/var/folders/92/jlcv07t503q05dghb407p5bd4_6dlc/T/ipykernel_98262/2208879285.py:4
: DeprecationWarning: In a future version, `df.iloc[:, i] = newvals` will
attempt to set the values inplace instead of always setting a new array. To
retain the old behavior, use either `df[df.columns[i]] = newvals` or, if columns
are non-unique, `df.isetitem(i, newvals)`
  paxraw.loc[:, col] = paxraw.loc[:, col].astype(int)
```

```
casting col='PAXSTAT' to int
casting col='PAXCAL' to int
casting col='PAXDAY' to int
casting col='PAXN' to int
casting col='PAXHOUR' to int
casting col='PAXMINUT' to int
casting col='PAXINTEN' to int
casting col='PAXSTEP' to int
col='PAXSTEP' has 62 NA values, setting to 0
```

```
/var/folders/92/jlcv07t503q05dghb407p5bd4_6dlc/T/ipykernel_98262/2208879285.py:7
: DeprecationWarning: In a future version, `df.iloc[:, i] = newvals` will
attempt to set the values inplace instead of always setting a new array. To
retain the old behavior, use either `df[df.columns[i]] = newvals` or, if columns
are non-unique, `df.isetitem(i, newvals)`
  paxraw.loc[:, col] = paxraw.loc[:, col].replace(np.nan, 0).astype(int)
```

2.1.1 Don't add a datetime column, takes too long

```python
paxraw["datetime"] = paxraw.progress_apply(
    lambda x: datetime.datetime(2006, 1, 1) + datetime.timedelta(
        days=int(x.PAXDAY - 1),
        hours=int(x.PAXHOUR),
        minutes=int(x.PAXMINUT)
    ),
    axis=1,
```

```
)
```

[15]:
```
if "PAXSTEP" not in paxraw.columns:
    paxraw["PAXSTEP"] = 0
```

[16]:
```
paxraw.dtypes
```

[16]:
```
SEQN        int64
PAXSTAT     int64
PAXCAL      int64
PAXDAY      int64
PAXN        int64
PAXHOUR     int64
PAXMINUT    int64
PAXINTEN    int64
PAXSTEP     int64
dtype: object
```

[17]:
```
paxraw.head()
```

[17]:

	SEQN	PAXSTAT	PAXCAL	PAXDAY	PAXN	PAXHOUR	PAXMINUT	PAXINTEN	PAXSTEP
0	31128	1	1	1	1	0	0	166	4
1	31128	1	1	1	2	0	1	27	0
2	31128	1	1	1	3	0	2	0	0
3	31128	1	1	1	4	0	3	276	4
4	31128	1	1	1	5	0	4	0	0

2.2 Save parquet

[18]:
```
parquetfile = datadir / f"paxraw_{CHAR_LOOKUP[year].lower()}.parquet"
parquetfile
```

[18]:
```
PosixPath('/Users/mm51929/projects/2022/07-nhanes-
analysis/data/raw/2005-2006/paxraw_d.parquet')
```

[19]:
```
paxraw.to_parquet(parquetfile)
# paxraw = pd.read_parquet(parquetfile)
```

3 Define intensity level cuts and METs

[20]:
```
# cuts defined in literature and in [common software](https://github.com/
 ↪vandomed/nhanesaccel/blob/7ebd7a0cd6e2f169e6f81a66c8c99b1746eacb51/R/
 ↪process_nhanes.R#L267)
int_cuts = [100, 760, 2020, 5999]
```

[21]:
```
# add end ranges for interpolation
int_cuts_endranges = [paxraw.PAXINTEN.min()] + int_cuts + [paxraw.PAXINTEN.
 ↪max() + 1]
int_cuts_endranges
```

[21]:
```
[0, 100, 760, 2020, 5999, 32768]
```

[22]:
```
len(int_cuts_endranges) - 1
```

[22]:
```
5
```

[23]:
```
# MET values corresponding to each cut point
METs = [1, 1, 2, 3.5, 6, 10]
labels = ["Sedentary", "Low", "Light", "Moderate", "Vigorous"]
```

[24]:
```
# linearly interpolate MET values
METs_full = np.interp(
    np.arange(int_cuts_endranges[0], int_cuts_endranges[-1]),␣
 ↪int_cuts_endranges, METs
)
METs_lookup = pd.DataFrame(
    {
        "MET": METs_full,
        "PAXINTEN": np.arange(int_cuts_endranges[0], int_cuts_endranges[-1]),
    }
)
```

3.1 Join METs

[25]:
```
paxraw = paxraw.merge(METs_lookup, how="left", on="PAXINTEN")
```

3.2 Save parquet

[26]:
```
parquetfile = datadir / f"paxraw_{CHAR_LOOKUP[year].lower()}_met.parquet"
parquetfile
```

[26]:
```
PosixPath('/Users/mm51929/projects/2022/07-nhanes-
analysis/data/raw/2005-2006/paxraw_d_met.parquet')
```

[27]:
```
paxraw.to_parquet(parquetfile)
```

4 worn classification

R code here

4.1 Run a sample of data through

[28]:
```
paxraw_sample = paxraw.loc[paxraw.SEQN == paxraw.SEQN.values[0], :].copy()
paxraw_sample.shape
```

[28]:
```
(10080, 10)
```

[29]:
```
min_worn_hours_threshold: int = 10
max_nonzero_count_per_unworn_hour: int = 2
max_of_nonzero_in_unworn_hour: int = 100
MINUTES_PER_HOUR = 60
```

[30]:
```
paxraw_sample.columns
```

[30]:
```
Index(['SEQN', 'PAXSTAT', 'PAXCAL', 'PAXDAY', 'PAXN', 'PAXHOUR', 'PAXMINUT',
       'PAXINTEN', 'PAXSTEP', 'MET'],
      dtype='object')
```

[31]:
```
# set the indicator to True to start
worn = np.ones(paxraw_sample.shape[0])
```

[32]:
```
worn.shape
```

[32]:
```
(10080,)
```

paxraw = paxraw_sample

[33]:
```
paxinten = paxraw_sample.PAXINTEN.values
```

[34]:
```
paxinten.shape[0]
```

[34]:
```
10080
```

4.2 Time a simple algorithm using numpy arrays

[35]:
```
# take the first hour
# assert d.iloc[:MINUTES_PER_HOUR, :].shape[0] == MINUTES_PER_HOUR
if ((paxinten[:MINUTES_PER_HOUR] > 0).sum() <=␣
 ↪max_nonzero_count_per_unworn_hour) and (
    (paxinten[:MINUTES_PER_HOUR] < max_of_nonzero_in_unworn_hour).sum() ==␣
 ↪MINUTES_PER_HOUR
):
    worn[:MINUTES_PER_HOUR] = 0
```

[36]:
```
for i in range(MINUTES_PER_HOUR + 1, worn.shape[0]):
    # assert paxraw_sample.iloc[(i-60):i, :].shape[0] == MINUTES_PER_HOUR
```

```
    if ((paxinten[(i - MINUTES_PER_HOUR) : i] > 0).sum() <=␣
↪max_nonzero_count_per_unworn_hour) and (
        (paxinten[(i - MINUTES_PER_HOUR) : i] < max_of_nonzero_in_unworn_hour).
↪sum()
        == MINUTES_PER_HOUR
    ):
        worn[(i - MINUTES_PER_HOUR) : i] = 0
```

4.2.1 Write that as a function (in `util.py`)

[37]:
```
from utils import worn_indicator, worn_indicator_fast  # noqa: E402
```

[38]:
```
%%timeit
worn_indicator(paxraw_sample.PAXINTEN.values)
```

```
52.2 ms ± 3.35 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

Run it once to compile it

[39]:
```
worn_indicator_fast(paxraw_sample.PAXINTEN.values)
```

[39]:
```
array([1., 1., 1., …, 1., 1., 1.])
```

[40]:
```
%%timeit
worn_indicator_fast(paxraw_sample.PAXINTEN.values)
```

```
1.83 ms ± 96.9 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

4.3 Test an algorithm using pandas

4.3.1 Process out active minutes akin to Fishman (2016)

1. Compute worn/nonworn indicator on each minute, defined as intervals at least 60 minutes of count $= 0$, with up to two count < 100.
2. Sum worn time per day.
3. Discard days with wear time $<$ 10h.
4. Sum up total count per day.
5. Measure average total count per day on valid days, per individual.

[41]:
```
paxraw_sample.head()
```

[41]:

	SEQN	PAXSTAT	PAXCAL	PAXDAY	PAXN	PAXHOUR	PAXMINUT	PAXINTEN	PAXSTEP	\
0	31128	1	1	1	1	0	0	166	4	
1	31128	1	1	1	2	0	1	27	0	
2	31128	1	1	1	3	0	2	0	0	
3	31128	1	1	1	4	0	3	276	4	
4	31128	1	1	1	5	0	4	0	0	

```
        MET
0  1.100000
1  1.000000
2  1.000000
3  1.266667
4  1.000000
```

[42]:
```
paxraw_sample.head()
```

[42]:
```
    SEQN  PAXSTAT  PAXCAL  PAXDAY  PAXN  PAXHOUR  PAXMINUT  PAXINTEN  PAXSTEP  \
0  31128        1       1       1     1        0         0       166        4
1  31128        1       1       1     2        0         1        27        0
2  31128        1       1       1     3        0         2         0        0
3  31128        1       1       1     4        0         3       276        4
4  31128        1       1       1     5        0         4         0        0

        MET
0  1.100000
1  1.000000
2  1.000000
3  1.266667
4  1.000000
```

[43]:
```
# set the indicator to True to start
paxraw_sample.loc[:, "worn"] = True
```

[44]:
```
paxraw_sample.columns
```

[44]:
```
Index(['SEQN', 'PAXSTAT', 'PAXCAL', 'PAXDAY', 'PAXN', 'PAXHOUR', 'PAXMINUT',
       'PAXINTEN', 'PAXSTEP', 'MET', 'worn'],
      dtype='object')
```

[45]:
```
PAXINTEN_col = np.arange(paxraw_sample.shape[1])[paxraw_sample.columns ==␣
↪"PAXINTEN"][0]
```

[46]:
```
worn_col = np.arange(paxraw_sample.shape[1])[paxraw_sample.columns == "worn"][0]
```

[47]:
```
# take the first 60 minutes
assert paxraw_sample.iloc[:60, :].shape[0] == 60
if (paxraw_sample.iloc[:60, PAXINTEN_col] >= 100).sum() <= 2:
    paxraw_sample.iloc[:60, worn_col] = False
```

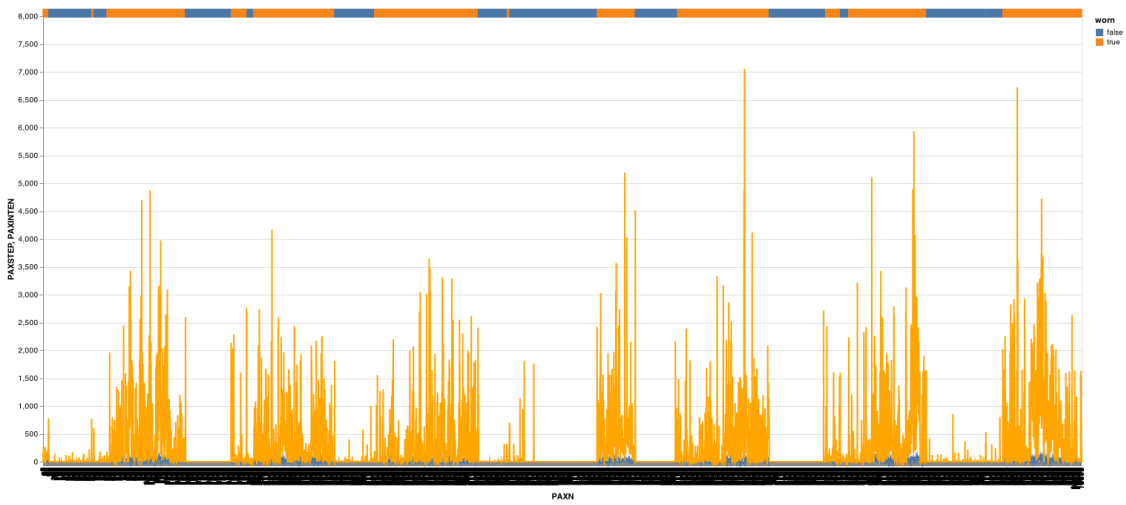
[48]:
```
paxraw_sample.iloc[paxraw_sample.shape[0] - 60 : paxraw_sample.shape[0], :].
↪shape
```

[48]:
```
(60, 11)
```

[49]:
```
for i in range(61, paxraw_sample.shape[0]):
    assert paxraw_sample.iloc[(i - 60) : i, :].shape[0] == 60
    if (paxraw_sample.iloc[(i - 60) : i, PAXINTEN_col] >= 100).sum() <= 2:
        paxraw_sample.iloc[(i - 60) : i, worn_col] = False
```

[50]:
```
(
    alt.Chart(paxraw_sample)
    .mark_bar(width=1)
    .encode(x="PAXN:O", y=alt.value(-10), y2=alt.value(2), color="worn")
    + alt.Chart(paxraw_sample).mark_line().encode(x="PAXN:O", y="PAXSTEP")
    + alt.Chart(paxraw_sample).mark_line(color="orange").encode(x="PAXN:O",␣
 ↪y="PAXINTEN")
).properties(width=1400, height=600)
```

[50]:

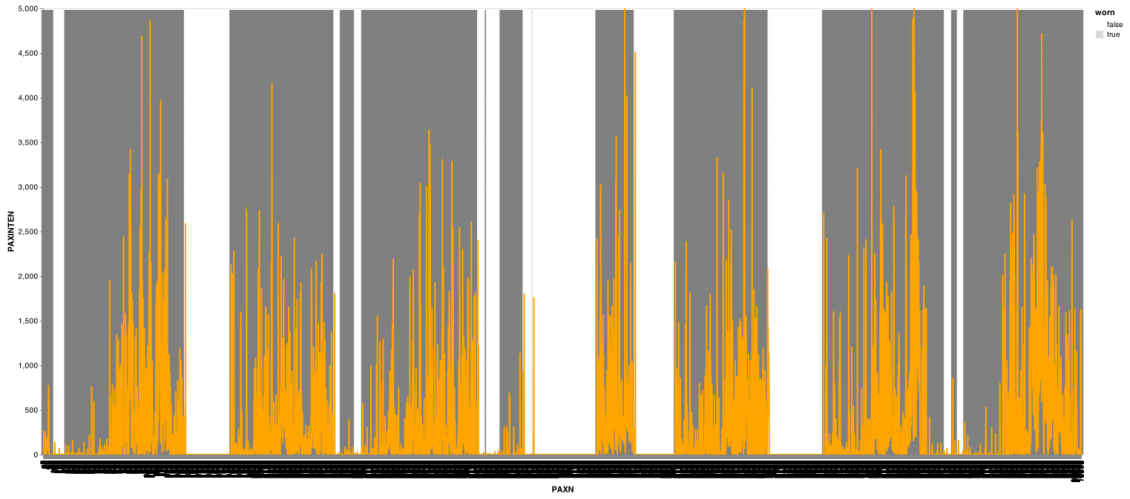


[51]:
```
# set the indicator to True to start
paxraw_sample.loc[:, "worn"] = True
# take the first 60 minutes
assert paxraw_sample.iloc[:60, :].shape[0] == 60
# only 2 allowed > 0, and all 60 are less than 100
if ((paxraw_sample.iloc[:60, PAXINTEN_col] > 0).sum() <= 2) and (
    (paxraw_sample.iloc[:60, PAXINTEN_col] < 100).sum() == 60
):
    paxraw_sample.iloc[:60, worn_col] = False
```

[52]:
```
for i in range(61, paxraw_sample.shape[0]):
    assert paxraw_sample.iloc[(i - 60) : i, :].shape[0] == 60
    if ((paxraw_sample.iloc[(i - 60) : i, PAXINTEN_col] > 0).sum() <= 2) and (
        (paxraw_sample.iloc[(i - 60) : i, PAXINTEN_col] < 100).sum() == 60
    ):
        paxraw_sample.iloc[(i - 60) : i, worn_col] = False
```

[53]:
```
(
    alt.Chart(paxraw_sample)
    .mark_bar(width=5, opacity=0.3)
    .encode(
        x="PAXN:O",
        y=alt.value(600),
        y2=alt.value(2),
        color=alt.Color("worn", scale=alt.Scale(range=["white", "grey"])),
    )
    + alt.Chart(paxraw_sample)
    .mark_line(color="orange", clip=True)
    .encode(x="PAXN:O", y=alt.Y("PAXINTEN", scale=alt.Scale(domain=[0, 5000])))
).properties(width=1400, height=600)
```

[53]:

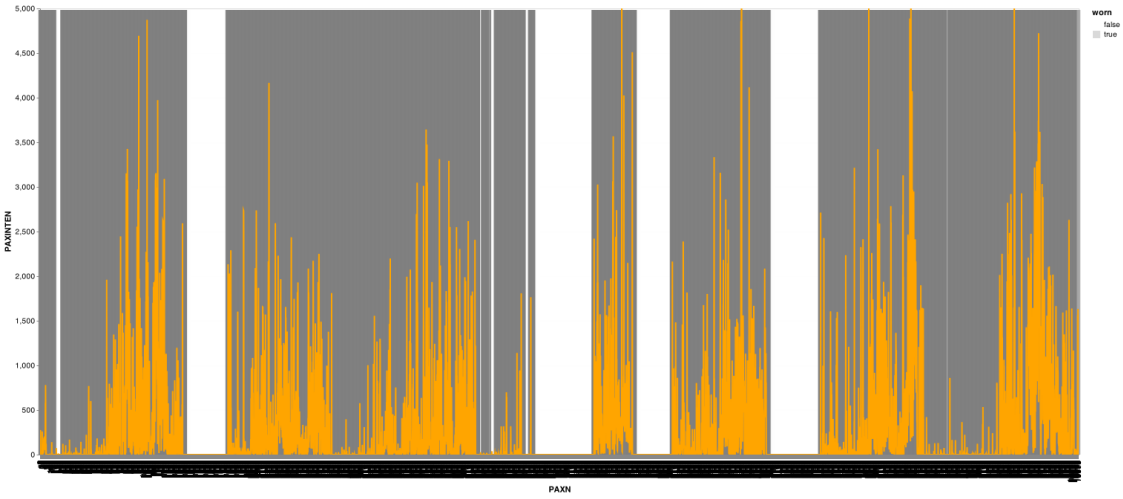


[54]:
```
# set the indicator to False to start
paxraw_sample.loc[:, "worn"] = False
# take the first 60 minutes
assert paxraw_sample.iloc[:60, :].shape[0] == 60
# only 2 allowed > 0, and all 60 are less than 100
if ((paxraw_sample.iloc[:60, PAXINTEN_col] > 0).sum() > 2) or (
    (paxraw_sample.iloc[:60, PAXINTEN_col] >= 100).sum() > 0
):
    paxraw_sample.iloc[:60, worn_col] = True
```

[55]:
```
for i in range(61, paxraw_sample.shape[0]):
    assert paxraw_sample.iloc[(i - 60) : i, :].shape[0] == 60
    if ((paxraw_sample.iloc[(i - 60) : i, PAXINTEN_col] > 0).sum() > 2) or (
        (paxraw_sample.iloc[(i - 60) : i, PAXINTEN_col] >= 100).sum() > 0
    ):
        paxraw_sample.iloc[(i - worn_col) : i, worn_col] = True
```

[56]:
```
(
    alt.Chart(paxraw_sample)
    .mark_bar(width=5, opacity=0.3)
    .encode(
        x="PAXN:O",
        y=alt.value(600),
        y2=alt.value(2),
        color=alt.Color("worn", scale=alt.Scale(range=["white", "grey"])),
    )
    + alt.Chart(paxraw_sample)
    .mark_line(color="orange", clip=True)
    .encode(x="PAXN:O", y=alt.Y("PAXINTEN", scale=alt.Scale(domain=[0, 5000])))
).properties(width=1400, height=600)
```

[56]:



[57]:
```
worn_minutes = paxraw_sample.groupby("PAXDAY").agg({"worn": [sum], "PAXINTEN":␣
↪[sum]})
worn_minutes["valid_day"] = worn_minutes["worn"]["sum"] >␣
↪min_worn_hours_threshold * 60
# filter to valid days
worn_minutes = worn_minutes.loc[worn_minutes.valid_day, :]
np.mean(worn_minutes["PAXINTEN"]["sum"])
```

[57]:
```
345961.4285714286
```

[58]:
```
from utils import get_person_active_count  # noqa: E402
```

[59]:
```
get_person_active_count(paxraw.loc[paxraw.SEQN == paxraw.SEQN.unique()[0], :])
```

[59]:
```
        worn PAXINTEN valid_day
         sum       sum
```

```
PAXDAY
1       1276   377456      True
2       1009   308309      True
3       1273   324734      True
4        599   229846     False
5        911   304957      True
6       1065   388323      True
7       1307   488105      True
```

[60]:
```
%%timeit
get_person_active_count(paxraw.loc[paxraw.SEQN == paxraw.SEQN.unique()[0], :])
```

```
3.5 s ± 75.8 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

Hours:

[61]:
```
3.13 * (paxraw["SEQN"].unique().shape[0]) / 60 / 60
```

[61]:
```
6.481708333333333
```

4.3.2 Test it on a slightly bigger sample

Make sure the groupby object returned makes sense before waiting 8 hours

[62]:
```
person_active_counts = (
    paxraw.loc[paxraw.SEQN.isin(paxraw.SEQN.unique()[:10]), :]
    .groupby("SEQN")
    .progress_apply(get_person_active_count)
)
```

```
100%|
                  | 10/10 [00:37<00:00,  3.73s/it]
```

[63]:
```
person_active_counts
```

[63]:
```
              worn PAXINTEN valid_day
               sum      sum
SEQN  PAXDAY
31128 1       1276   377456      True
      2       1009   308309      True
      3       1273   324734      True
      4        599   229846     False
      5        911   304957      True
      6       1065   388323      True
      7       1307   488105      True
31129 1        329   125225     False
      2        879   260386      True
      3        905   281481      True
```

	4	8	8615	False
	5	982	274492	True
	6	929	271234	True
	7	316	118916	False
31131	1	771	282054	True
	2	885	189285	True
	3	879	202961	True
	4	914	303220	True
	5	1015	343780	True
	6	921	246466	True
	7	583	183323	False
31132	1	907	430315	True
	2	864	409685	True
	3	863	647452	True
	4	343	9231	False
	5	976	717103	True
	6	881	162362	True
	7	574	278420	False
31133	1	2	238	False
	2	766	156406	True
	3	749	237228	True
	4	983	159419	True
	5	994	289835	True
	6	914	217388	True
	7	600	148628	False
31134	1	968	268676	True
	2	137	13585	False
	3	881	454104	True
	4	1039	339230	True
	5	1051	253482	True
	6	600	218650	False
	7	815	369358	True
31137	1	747	285416	True
	2	682	236577	True
	3	531	208456	False
	4	16	3559	False
	5	1022	321241	True
	6	1111	362805	True
	7	1323	526423	True
31139	1	700	96018	True
	2	731	98912	True
	3	760	145537	True
	4	719	99049	True
	5	718	174999	True
	6	578	103628	False
	7	653	121160	True
31140	1	288	257926	False

```
        2        634   285108      True
        3        458   259019     False
        4        691   626162      True
        5        174    50944     False
        6          0       15     False
        7          0        3     False
31141   1        702   130114      True
        2        677   502813      True
        3        868   279641      True
        4        779   234451      True
        5        964   177635      True
        6        754   109035      True
        7        881   142933      True
```

[64]:
```
# check that we can save and load it
# with the heirarchical indexes
parquetfile = datadir / f"paxraw_{CHAR_LOOKUP[year].lower()}_sample_test_write.
 ↪parquet"
person_active_counts.to_parquet(parquetfile)
pd.read_parquet(parquetfile).head()
```

[64]:
```
              worn PAXINTEN valid_day
               sum      sum
SEQN  PAXDAY
31128 1       1276   377456      True
      2       1009   308309      True
      3       1273   324734      True
      4        599   229846     False
      5        911   304957      True
```

4.3.3 Don't apply the numpy-based function to Pandas column, too slow

Because it takes almost 30 minutes.

this would take ~25 minutes

```
paxraw['worn'] = 1

for SEQN in tqdm(pd.unique(paxraw.SEQN.values)):
    paxraw.loc[paxraw.SEQN == SEQN, 'worn'] = worn_indicator(paxraw.loc[paxraw.SEQN == SEQN, '

parquetfile = datadir / f"paxraw_{CHAR_LOOKUP[year].lower()}_met_worn.parquet"
paxraw.to_parquet(parquetfile)
```

4.3.4 Skip a fully pandas-based solution entirely, it's very very slow

FWIW, the `rolling` version should take better advantage of Pandas, but it's still too slow.

```
person_active_counts = (
    paxraw.groupby("SEQN").progress_apply(get_person_active_count).reset_index()
```

)

4.4 Apply $_{\text{numpy-based}}$ numba algorithm to full dataset

~Less than 10 min.~

A few seconds.

[65]:
```
from utils import bout_classifier_SEQN_long, worn_indicator_SEQN_long  # noqa: E402
```

[66]:
```
# this should be fast
paxraw["worn"] = worn_indicator_SEQN_long(paxraw.PAXINTEN.values, paxraw.SEQN.values)
```

4.4.1 Compare the full numpy and the numpy function applied to pandas array

It's commented out because we're not running the "numpy function applied to pandas array" version now. They are the same

```
(paxraw['worn'] == worn).sum()
```

```
(paxraw['worn'] != worn).sum()
```

```
(paxraw['worn'] == worn).sum() == worn.shape[0]
```

```
paxraw.loc[(paxraw.worn != worn), :].head()
```

4.5 Save parquet

[67]:
```
parquetfile = datadir / f"paxraw_{CHAR_LOOKUP[year].lower()}_met_worn.parquet"
parquetfile
```

[67]:
```
PosixPath('/Users/mm51929/projects/2022/07-nhanes-analysis/data/raw/2005-2006/paxraw_d_met_worn.parquet')
```

[68]:
```
paxraw.to_parquet(parquetfile)
```

5 Generate indicators for bouts of activity levels

[69]:
```
paxraw["vigorous_bout"] = bout_classifier_SEQN_long(
    paxraw.PAXINTEN.values,
    paxraw.SEQN.values,
    paxraw.worn.values,
    np.zeros(paxraw.PAXINTEN.values.shape[0]),
    upper=int_cuts_endranges[5],
    lower=int_cuts_endranges[4],
    tol_upper_soft=0,
    tol_lower_soft=0,
    m=10,
```

```
    lower_soft=0,  # (int_cuts_endranges[3] + int_cuts_endranges[4])/2,
    check_already_classified=False,
)
```

[70]:
```
paxraw["moderate_bout"] = bout_classifier_SEQN_long(
    paxraw.PAXINTEN.values,
    paxraw.SEQN.values,
    paxraw.worn.values,
    paxraw.vigorous_bout.values,
    upper=int_cuts_endranges[4],
    lower=int_cuts_endranges[3],
    tol_upper_soft=10,
    tol_lower_soft=0,
    m=10,
    lower_soft=(int_cuts_endranges[2] + int_cuts_endranges[3]) / 2,
    upper_soft=int_cuts_endranges[5],
    check_already_classified=True,
)
```

[71]:
```
paxraw["light_bout"] = bout_classifier_SEQN_long(
    paxraw.PAXINTEN.values,
    paxraw.SEQN.values,
    paxraw.worn.values,
    np.maximum(paxraw.moderate_bout.values, paxraw.vigorous_bout.values),
    upper=int_cuts_endranges[3],
    lower=int_cuts_endranges[2],
    tol_upper_soft=10,
    tol_lower_soft=0,
    m=10,
    lower_soft=(int_cuts_endranges[1] + int_cuts_endranges[2]) / 2,
    upper_soft=int_cuts_endranges[5],
    check_already_classified=True,
)
```

[72]:
```
paxraw["low_bout"] = bout_classifier_SEQN_long(
    paxraw.PAXINTEN.values,
    paxraw.SEQN.values,
    paxraw.worn.values,
    np.maximum(paxraw.light_bout.values, paxraw.moderate_bout.values, paxraw.
 ↪vigorous_bout.values),
    upper=int_cuts_endranges[2],
    lower=int_cuts_endranges[1],
    tol_upper_soft=10,
    tol_lower_soft=0,
    m=10,
    lower_soft=(int_cuts_endranges[0] + int_cuts_endranges[1]) / 2,
    upper_soft=int_cuts_endranges[5],
```

```
    check_already_classified=True,
)
```

[73]:
```
paxraw["sed_bout"] = bout_classifier_SEQN_long(
    paxraw.PAXINTEN.values,
    paxraw.SEQN.values,
    paxraw.worn.values,
    paxraw.low_bout.values,
    upper=int_cuts_endranges[1],
    lower=int_cuts_endranges[0],
    tol_upper_soft=0,
    tol_lower_soft=0,
    m=10,
    check_already_classified=False,
)
```

5.0.1 Add them all to the dataframe

[74]:
```
paxraw["no_bout"] = (
    (paxraw["worn"] == 1)
    & (paxraw["sed_bout"] == 0)
    & (paxraw["low_bout"] == 0)
    & (paxraw["light_bout"] == 0)
    & (paxraw["moderate_bout"] == 0)
    & (paxraw["vigorous_bout"] == 0)
) * 1
```

5.1 A single column to label minute-by-minute intensity

[75]:
```
paxraw["intensity"] = pd.cut(
    paxraw["PAXINTEN"].values, int_cuts_endranges, right=False,␣
 ↪labels=range(len(labels))
)
# don't include the labels for size:
# labels=labels
```

[76]:
```
paxraw.head()
```

[76]:

	SEQN	PAXSTAT	PAXCAL	PAXDAY	PAXN	PAXHOUR	PAXMINUT	PAXINTEN	PAXSTEP	\
0	31128	1	1	1	1	0	0	166	4	
1	31128	1	1	1	2	0	1	27	0	
2	31128	1	1	1	3	0	2	0	0	
3	31128	1	1	1	4	0	3	276	4	
4	31128	1	1	1	5	0	4	0	0	

	MET	worn	vigorous_bout	moderate_bout	light_bout	low_bout	\

```
0  1.100000   1.0            0.0            0.0         0.0       0.0
1  1.000000   1.0            0.0            0.0         0.0       0.0
2  1.000000   1.0            0.0            0.0         0.0       0.0
3  1.266667   1.0            0.0            0.0         0.0       0.0
4  1.000000   1.0            0.0            0.0         0.0       0.0

   sed_bout  no_bout intensity
0       0.0        1         1
1       0.0        1         0
2       0.0        1         0
3       0.0        1         1
4       0.0        1         0
```

[77]:
```
paxraw["METh"] = paxraw.MET / 60
paxraw["activeMETh"] = (paxraw.MET - 1) / 60
```

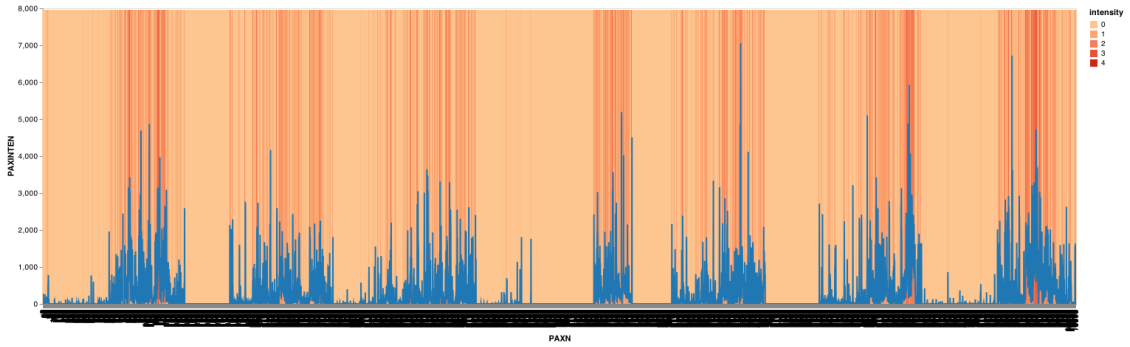
[78]:
```
paxraw_sample = paxraw.loc[paxraw.SEQN == paxraw.SEQN.values[0], :].copy()
paxraw_sample.head()
```

[78]:
```
    SEQN  PAXSTAT  PAXCAL  PAXDAY  PAXN  PAXHOUR  PAXMINUT  PAXINTEN  PAXSTEP  \
0  31128        1       1       1     1        0         0       166        4
1  31128        1       1       1     2        0         1        27        0
2  31128        1       1       1     3        0         2         0        0
3  31128        1       1       1     4        0         3       276        4
4  31128        1       1       1     5        0         4         0        0

        MET  worn  vigorous_bout  moderate_bout  light_bout  low_bout  \
0  1.100000   1.0            0.0            0.0         0.0       0.0
1  1.000000   1.0            0.0            0.0         0.0       0.0
2  1.000000   1.0            0.0            0.0         0.0       0.0
3  1.266667   1.0            0.0            0.0         0.0       0.0
4  1.000000   1.0            0.0            0.0         0.0       0.0

   sed_bout  no_bout intensity      METh  activeMETh
0       0.0        1         1  0.018333    0.001667
1       0.0        1         0  0.016667    0.000000
2       0.0        1         0  0.016667    0.000000
3       0.0        1         1  0.021111    0.004444
4       0.0        1         0  0.016667    0.000000
```

[79]:
```
(
    alt.Chart(paxraw_sample)
    .mark_bar(width=1)
    .encode(
        x="PAXN:O",
        y=alt.value(400),
        y2=alt.value(2),
```

```
        color=alt.Color("intensity", scale=alt.Scale(scheme="orangered"))
        # scale=alt.Scale(range=["white", "grey"])),
    )
    + alt.Chart(paxraw_sample)
    .mark_line(color="#1f77b4", clip=True)
    .encode(x="PAXN:O", y=alt.Y("PAXINTEN", scale=alt.Scale(domain=[0, 8000])))
).properties(width=1400, height=400)
```

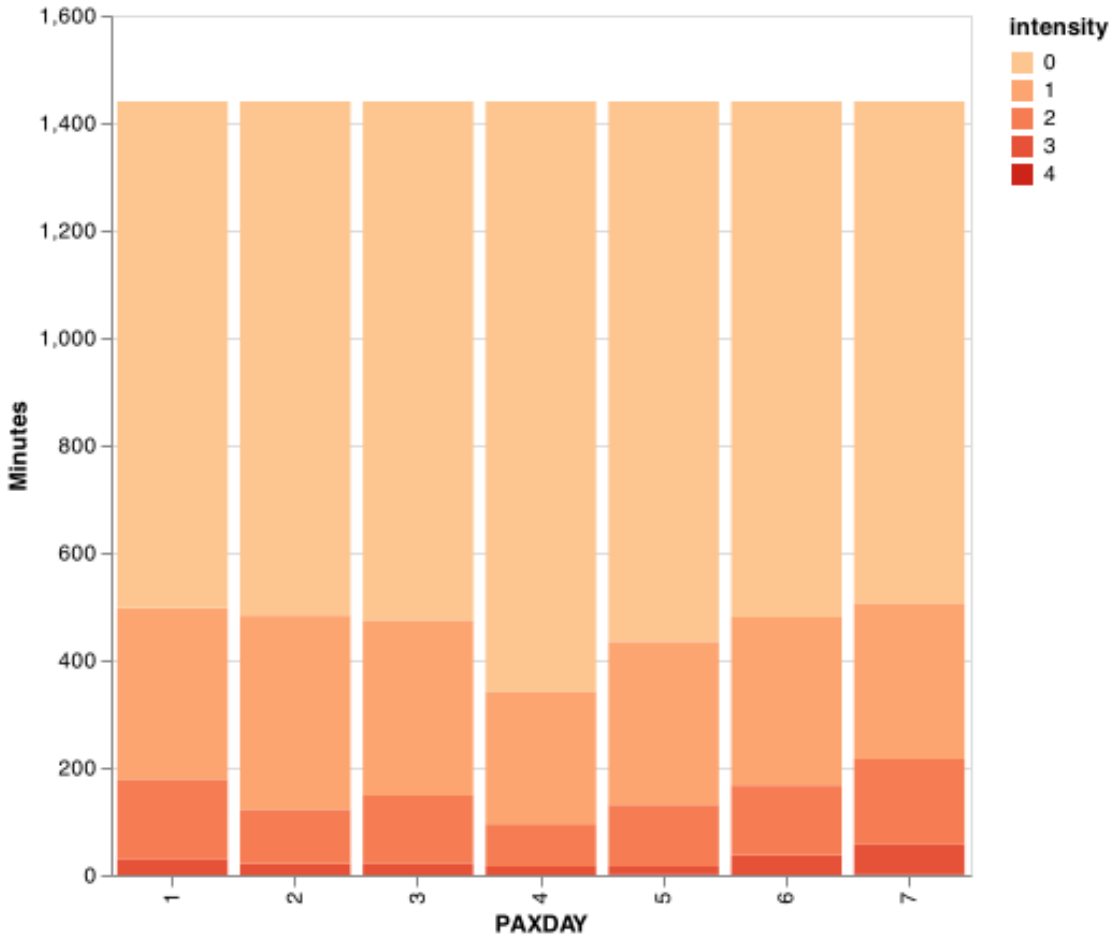
[79]:



[80]:

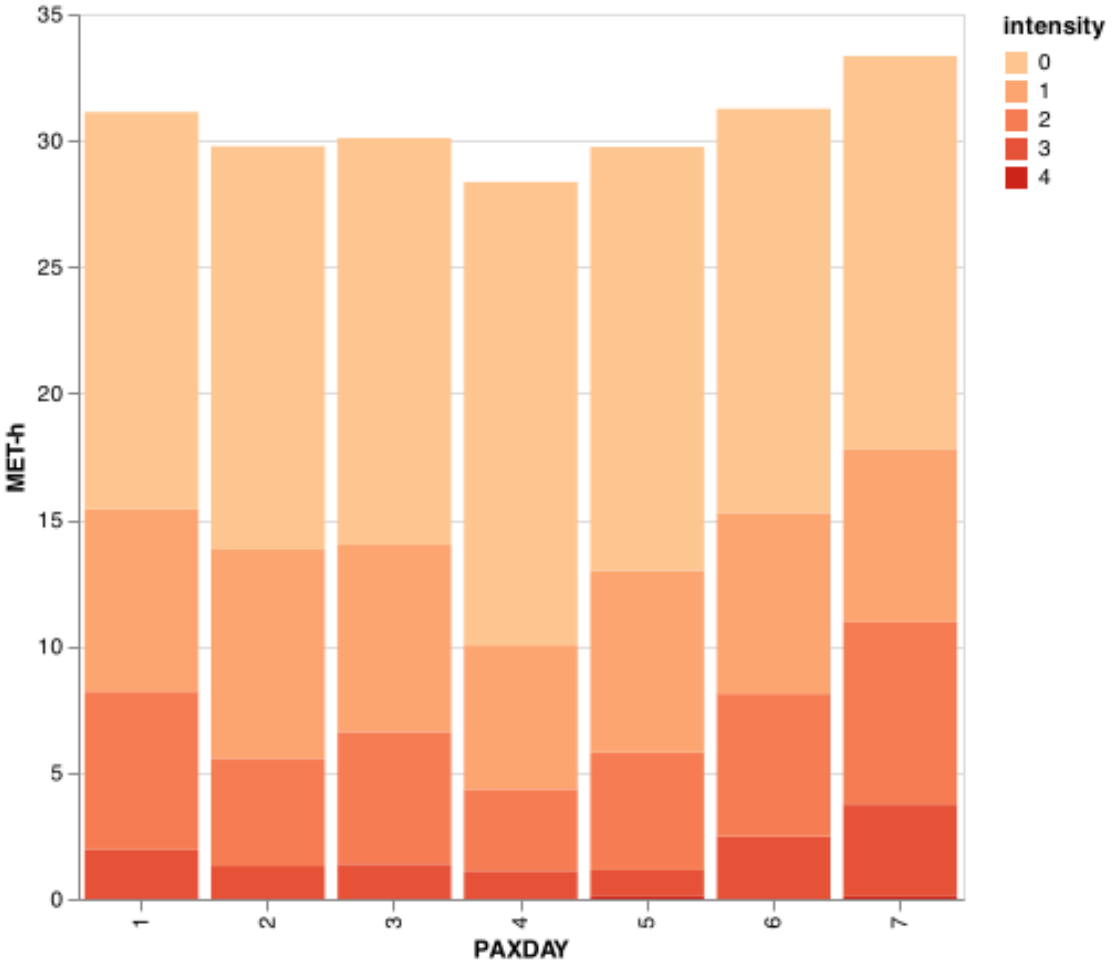
```
(
    alt.Chart(paxraw_sample)
    .mark_bar()
    .encode(
        x="PAXDAY:O",
        y=alt.Y("count()", title="Minutes"),
        color=alt.Color("intensity", scale=alt.Scale(scheme="orangered"))
        # scale=alt.Scale(range=["white", "grey"])),
    )
).properties(width=400, height=400)
```

[80]:



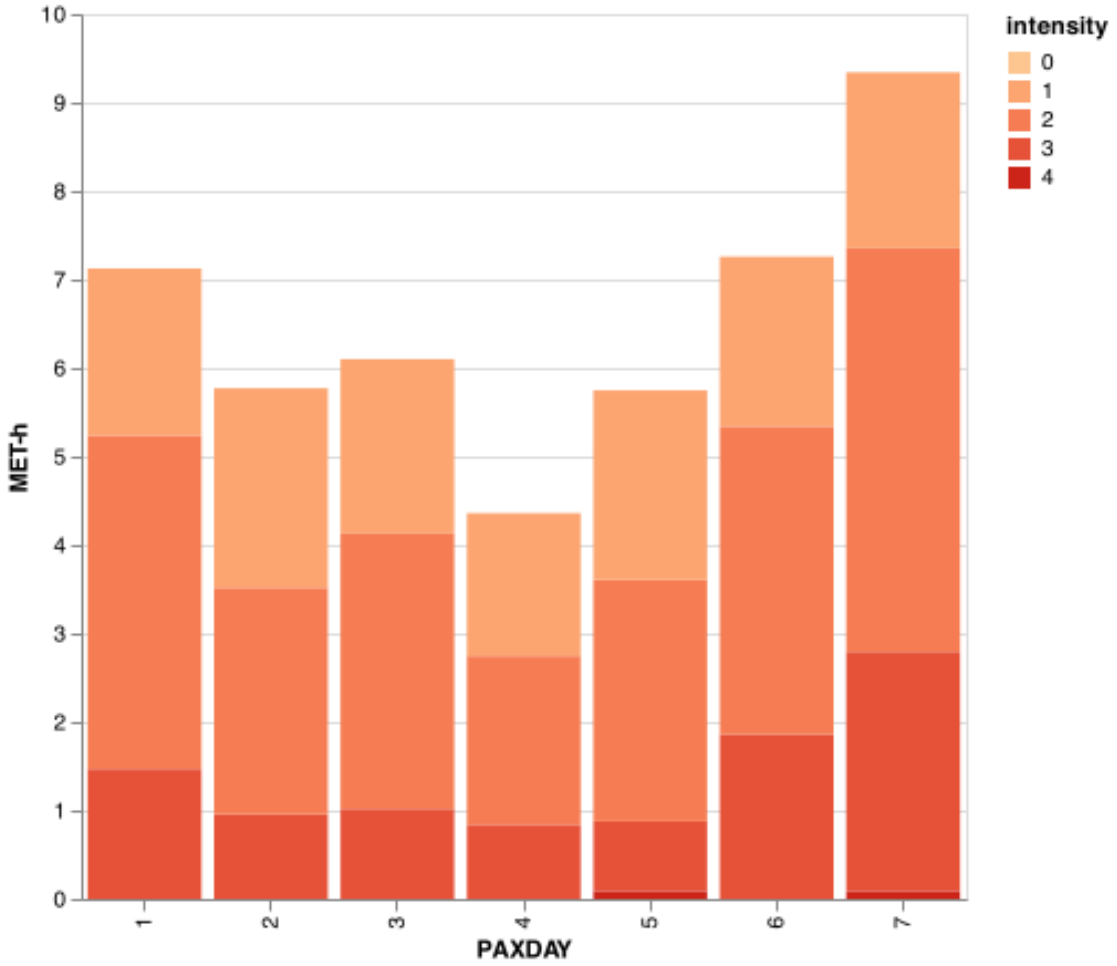
[81]:
```
(
    alt.Chart(paxraw_sample)
    .mark_bar()
    .encode(
        x="PAXDAY:O",
        y=alt.Y("sum(METh)", title="MET-h"),
        color=alt.Color("intensity", scale=alt.Scale(scheme="orangered"))
        # scale=alt.Scale(range=["white", "grey"])),
    )
).properties(width=400, height=400)
```

[81]:



[82]:
```
(
    alt.Chart(paxraw_sample)
    .mark_bar()
    .encode(
        x="PAXDAY:O",
        y=alt.Y("sum(activeMETh)", title="MET-h"),
        color=alt.Color("intensity", scale=alt.Scale(scheme="orangered"))
        # scale=alt.Scale(range=["white", "grey"])),
    )
).properties(width=400, height=400)
```

[82]:



5.2 Save parquet

[83]:
```
parquetfile = datadir / f"paxraw_{CHAR_LOOKUP[year].lower()}_met_worn_bouts.
  ↪parquet"
parquetfile
```

[83]:
```
PosixPath('/Users/mm51929/projects/2022/07-nhanes-
analysis/data/raw/2005-2006/paxraw_d_met_worn_bouts.parquet')
```

[84]:
```
paxraw.to_parquet(parquetfile)
# paxraw = pd.read_parquet(parquetfile)
```

5.3 Check for overlap on intensity bounts

First sum it up

[85]:
```
paxraw.loc[
    :,
    [
        "worn",
        "sed_bout",
        "low_bout",
        "light_bout",
        "moderate_bout",
        "vigorous_bout",
        "no_bout",
    ],
].sum(axis=0)
```

[85]:
```
worn             34326726.0
sed_bout           860091.0
low_bout          5478936.0
light_bout         641593.0
moderate_bout      220815.0
vigorous_bout      862408.0
no_bout          27125291.0
dtype: float64
```

From before the generalized numba function:

```
"worn             34326726.0\n",
"sed_bout         10920764.0\n",
"low_bout           604376.0\n",
"light_bout          62892.0\n",
"moderate_bout       56051.0\n",
"vigorous_bout      395787.0\n",
"no_bout          22352901.0\n",
```

[86]:
```
((paxraw["worn"] == 1) & (paxraw["sed_bout"] == 0)).sum()
```

[86]:
```
33466635
```

[87]:
```
((paxraw["worn"] == 1) & (paxraw["low_bout"] == 1)).sum()
```

[87]:
```
5478936
```

[88]:
```
((paxraw["sed_bout"] == 1) & (paxraw["low_bout"] == 1)).sum()
```

[88]:
```
0
```

[89]:
```
((paxraw["sed_bout"] == 1) & (paxraw["light_bout"] == 1)).sum()
```

[89]:
```
0
```

[90]:
```
((paxraw["sed_bout"] == 1) & (paxraw["moderate_bout"] == 1)).sum()
```

[90]: 0

[91]:
```
((paxraw["sed_bout"] == 1) & (paxraw["vigorous_bout"] == 1)).sum()
```

[91]: 0

6 Get valid days and other filters

6.1 Worn minutes by person-day to compute `valid_day`

[92]:
```
# sum minutes of wear and activity counts per day
worn_minutes = paxraw.groupby(["SEQN", "PAXDAY"]).agg({"worn": [np.sum]})
```

worn_minutes.columns = flatten_columns(worn_minutes.columns.values)

[93]:
```
# compute valid days
worn_minutes["valid_day"] = (
    worn_minutes["worn"]["sum"] > (min_worn_hours_threshold * MINUTES_PER_HOUR)
) * 1
```

[94]:
```
worn_minutes.head(15)
```

[94]:

		worn	valid_day
		sum	
SEQN	PAXDAY		
31128	1	1276.0	1
	2	1009.0	1
	3	1273.0	1
	4	599.0	0
	5	911.0	1
	6	1065.0	1
	7	1307.0	1
31129	1	329.0	0
	2	879.0	1
	3	904.0	1
	4	8.0	0
	5	982.0	1
	6	929.0	1
	7	316.0	0
31131	1	771.0	1

[95]:
```
worn_minutes.columns = flatten_columns(worn_minutes.columns.values)
```

[96]:
```
worn_minutes.head(15)
```

[96]:

SEQN	PAXDAY	worn_sum	valid_day
31128	1	1276.0	1
	2	1009.0	1
	3	1273.0	1
	4	599.0	0
	5	911.0	1
	6	1065.0	1
	7	1307.0	1
31129	1	329.0	0
	2	879.0	1
	3	904.0	1
	4	8.0	0
	5	982.0	1
	6	929.0	1
	7	316.0	0
31131	1	771.0	1

6.2 Other indicators at person-day level that can be used to filter

[97]:
```
agg_columns = ["PAXINTEN", "max_intensity", "out_of_calibration", "unreliable"]
```

[98]:
```
paxraw["max_intensity"] = (paxraw.PAXINTEN == 32767) * 1
paxraw["out_of_calibration"] = (paxraw.PAXCAL == 2) * 1
paxraw["unreliable"] = (paxraw.PAXSTAT == 2) * 1
```

[99]:
```
if "PAXSTEP" not in paxraw.columns:
    paxraw["PAXSTEP"] = 0
    paxraw["zero_steps_with_intensity"] = 0
else:
    paxraw["zero_steps_with_intensity"] = ((paxraw.PAXINTEN > 250) & (paxraw.
 ↪PAXSTEP == 0)) * 1
```

[100]:
```
paxraw["too_many_steps"] = (paxraw.PAXSTEP > 200) * 1
```

[101]:
```
# add a variable for steps_filtered, summing steps only if we have intensity␣
 ↪over 500
paxraw["steps_filtered_500"] = 0
paxraw.loc[paxraw.PAXINTEN >= 500, "steps_filtered_500"] = paxraw.PAXSTEP
paxraw["steps_filtered_300"] = 0
paxraw.loc[paxraw.PAXINTEN >= 300, "steps_filtered_300"] = paxraw.PAXSTEP
```

[102]:
```
agg_columns += [
    "zero_steps_with_intensity",
    "too_many_steps",
    "steps_filtered_500",
```

```
    "steps_filtered_300",
]
```

[103]:
```
tudor2009_filters = (
    paxraw.groupby(["SEQN", "PAXDAY"])
    .agg({col: [np.sum, "last"] for col in agg_columns})
    .reset_index()
)
tudor2009_filters.columns = flatten_columns(tudor2009_filters.columns.values)
```

[104]:
```
tudor2009_filters.head(15)
```

[104]:

	SEQN	PAXDAY	PAXINTEN_sum	PAXINTEN_last	max_intensity_sum	\
0	31128	1	377456	0	0	
1	31128	2	308309	0	0	
2	31128	3	324734	0	0	
3	31128	4	229846	0	0	
4	31128	5	304957	0	0	
5	31128	6	388323	0	0	
6	31128	7	488105	0	0	
7	31129	1	125225	0	0	
8	31129	2	260386	0	0	
9	31129	3	281481	0	0	
10	31129	4	8615	0	0	
11	31129	5	274492	0	0	
12	31129	6	271234	0	0	
13	31129	7	118916	0	0	
14	31131	1	282054	0	0	

	max_intensity_last	out_of_calibration_sum	out_of_calibration_last	\
0	0	0	0	
1	0	0	0	
2	0	0	0	
3	0	0	0	
4	0	0	0	
5	0	0	0	
6	0	0	0	
7	0	0	0	
8	0	0	0	
9	0	0	0	
10	0	0	0	
11	0	0	0	
12	0	0	0	
13	0	0	0	
14	0	0	0	

	unreliable_sum	unreliable_last	zero_steps_with_intensity_sum	\

0	0	0	0
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	0	0	0
11	0	0	0
12	0	0	0
13	0	0	0
14	0	0	0

	zero_steps_with_intensity_last	too_many_steps_sum	too_many_steps_last	\
0	0	0	0	
1	0	0	0	
2	0	0	0	
3	0	0	0	
4	0	0	0	
5	0	0	0	
6	0	0	0	
7	0	0	0	
8	0	0	0	
9	0	0	0	
10	0	0	0	
11	0	0	0	
12	0	0	0	
13	0	0	0	
14	0	0	0	

	steps_filtered_500_sum	steps_filtered_500_last	steps_filtered_300_sum	\
0	9414	0	10819	
1	7643	0	9218	
2	7740	0	9067	
3	4419	0	5600	
4	8001	0	9707	
5	11005	0	12841	
6	13191	0	14263	
7	2720	0	3223	
8	6697	0	7707	
9	7116	0	7968	
10	42	0	42	
11	6758	0	7872	
12	6634	0	7448	

```
13                    1919                          0                    2234
14                    7726                          0                    9250

    steps_filtered_300_last
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
14                        0
```

6.2.1 Join all person-day level indicators

[105]:
```
d_people_days = tudor2009_filters.merge(worn_minutes, how="inner", on=["SEQN",␣
↪"PAXDAY"])
```

[106]:
```
d_people_days.head(15)
```

[106]:
```
      SEQN  PAXDAY  PAXINTEN_sum  PAXINTEN_last  max_intensity_sum  \
0   31128       1        377456              0                  0
1   31128       2        308309              0                  0
2   31128       3        324734              0                  0
3   31128       4        229846              0                  0
4   31128       5        304957              0                  0
5   31128       6        388323              0                  0
6   31128       7        488105              0                  0
7   31129       1        125225              0                  0
8   31129       2        260386              0                  0
9   31129       3        281481              0                  0
10  31129       4          8615              0                  0
11  31129       5        274492              0                  0
12  31129       6        271234              0                  0
13  31129       7        118916              0                  0
14  31131       1        282054              0                  0

    max_intensity_last  out_of_calibration_sum  out_of_calibration_last  \
0                    0                       0                        0
1                    0                       0                        0
```

2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	0	0	0
11	0	0	0
12	0	0	0
13	0	0	0
14	0	0	0

	unreliable_sum	unreliable_last	zero_steps_with_intensity_sum	\
0	0	0	0	
1	0	0	0	
2	0	0	0	
3	0	0	0	
4	0	0	0	
5	0	0	0	
6	0	0	0	
7	0	0	0	
8	0	0	0	
9	0	0	0	
10	0	0	0	
11	0	0	0	
12	0	0	0	
13	0	0	0	
14	0	0	0	

	zero_steps_with_intensity_last	too_many_steps_sum	too_many_steps_last	\
0	0	0	0	
1	0	0	0	
2	0	0	0	
3	0	0	0	
4	0	0	0	
5	0	0	0	
6	0	0	0	
7	0	0	0	
8	0	0	0	
9	0	0	0	
10	0	0	0	
11	0	0	0	
12	0	0	0	
13	0	0	0	
14	0	0	0	

	steps_filtered_500_sum	steps_filtered_500_last	steps_filtered_300_sum	\
0	9414	0	10819	
1	7643	0	9218	
2	7740	0	9067	
3	4419	0	5600	
4	8001	0	9707	
5	11005	0	12841	
6	13191	0	14263	
7	2720	0	3223	
8	6697	0	7707	
9	7116	0	7968	
10	42	0	42	
11	6758	0	7872	
12	6634	0	7448	
13	1919	0	2234	
14	7726	0	9250	

	steps_filtered_300_last	worn_sum	valid_day
0	0	1276.0	1
1	0	1009.0	1
2	0	1273.0	1
3	0	599.0	0
4	0	911.0	1
5	0	1065.0	1
6	0	1307.0	1
7	0	329.0	0
8	0	879.0	1
9	0	904.0	1
10	0	8.0	0
11	0	982.0	1
12	0	929.0	1
13	0	316.0	0
14	0	771.0	1

[107]:
```
parquetfile = datadir / f"paxraw_{CHAR_LOOKUP[year].lower()}_people_days.
↪parquet"
parquetfile
```

[107]:
```
PosixPath('/Users/mm51929/projects/2022/07-nhanes-
analysis/data/raw/2005-2006/paxraw_d_people_days.parquet')
```

[108]:
```
d_people_days.to_parquet(parquetfile)
```

6.3 Sum up to person level

[109]:
```
d_people = d_people_days.groupby("SEQN").agg(
    {
        "zero_steps_with_intensity_sum": np.sum,
        "too_many_steps_sum": np.sum,
        "max_intensity_sum": np.sum,
        "out_of_calibration_sum": np.sum,
        "out_of_calibration_last": "last",
        "unreliable_sum": np.sum,
        "unreliable_last": "last",
        "steps_filtered_500_sum": np.mean,
        "steps_filtered_300_sum": np.mean,
        "valid_day": np.sum,
        "PAXINTEN_sum": np.mean,
    }
)
```

[110]:
```
d_people.head(15)
```

[110]:

SEQN	zero_steps_with_intensity_sum	too_many_steps_sum	max_intensity_sum
31128	0	0	0
31129	0	0	0
31131	0	0	0
31132	0	0	0
31133	0	0	0
31134	0	0	0
31137	0	0	0
31139	0	0	0
31140	0	0	0
31141	2	0	0
31142	0	0	0
31143	0	0	0
31144	0	0	0
31145	0	0	0
31146	0	0	0

SEQN	out_of_calibration_sum	out_of_calibration_last	unreliable_sum
31128	0	0	0
31129	0	0	0
31131	0	0	0
31132	0	0	0
31133	0	0	0
31134	0	0	0
31137	0	0	0

```
31139                     0                     0              0
31140                     0                     0              0
31141                     0                     0              0
31142                     0                     0              0
31143                     0                     0              0
31144                     0                     0              0
31145                     0                     0              0
31146                     0                     0              0

       unreliable_last  steps_filtered_500_sum  steps_filtered_300_sum  \
SEQN
31128                0             8773.285714            10216.428571
31129                0             4555.142857             5213.428571
31131                0             6346.857143             7688.285714
31132                0             9984.857143            10688.571429
31133                0             3301.428571             3956.285714
31134                0             6504.428571             7244.571429
31137                0             7412.428571             9072.142857
31139                0             1543.571429             2102.714286
31140                0             4056.142857             4457.857143
31141                0             4128.857143             5094.714286
31142                0             2581.714286             2850.857143
31143                0             7476.571429             8806.714286
31144                0             2862.000000             3169.857143
31145                0             9016.571429            10095.428571
31146                0             3294.571429             3714.571429

       valid_day   PAXINTEN_sum
SEQN
31128          6  345961.428571
31129          4  191478.428571
31131          6  250155.571429
31132          5  379224.000000
31133          5  172734.571429
31134          5  273869.285714
31137          5  277782.428571
31139          6  119900.428571
31140          2  211311.000000
31141          7  225231.714286
31142          1  150623.142857
31143          7  363330.714286
31144          3  110674.428571
31145          6  454646.857143
31146          0  169655.000000
```

[111]:
```
d_people.shape
```

[111]: (7455, 11)

[112]:
```
parquetfile = datadir / f"paxraw_{CHAR_LOOKUP[year].lower()}_people.parquet"
parquetfile
```

[112]: PosixPath('/Users/mm51929/projects/2022/07-nhanes-analysis/data/raw/2005-2006/paxraw_d_people.parquet')

[113]:
```
d_people.to_parquet(parquetfile)
```

6.4 Use the indicators to filter to people with reliable data

[114]:
```
d_reliable = d_people.loc[
    (d_people.zero_steps_with_intensity_sum <= 10)
    & (d_people.too_many_steps_sum <= 10)
    & (d_people.max_intensity_sum <= 10)
    & (~d_people.out_of_calibration_last)
    & (d_people.unreliable_sum <= 10)
    & (d_people.steps_filtered_500_sum <= 200000),
    :,
]
d_reliable.head(15)
```

[114]:

SEQN	zero_steps_with_intensity_sum	too_many_steps_sum	max_intensity_sum	\
31128	0	0	0	
31129	0	0	0	
31131	0	0	0	
31132	0	0	0	
31133	0	0	0	
31134	0	0	0	
31137	0	0	0	
31139	0	0	0	
31140	0	0	0	
31141	2	0	0	
31142	0	0	0	
31143	0	0	0	
31144	0	0	0	
31145	0	0	0	
31146	0	0	0	

SEQN	out_of_calibration_sum	out_of_calibration_last	unreliable_sum	\
31128	0	0	0	
31129	0	0	0	
31131	0	0	0	
31132	0	0	0	

31133	0	0	0
31134	0	0	0
31137	0	0	0
31139	0	0	0
31140	0	0	0
31141	0	0	0
31142	0	0	0
31143	0	0	0
31144	0	0	0
31145	0	0	0
31146	0	0	0

SEQN	unreliable_last	steps_filtered_500_sum	steps_filtered_300_sum	\
31128	0	8773.285714	10216.428571	
31129	0	4555.142857	5213.428571	
31131	0	6346.857143	7688.285714	
31132	0	9984.857143	10688.571429	
31133	0	3301.428571	3956.285714	
31134	0	6504.428571	7244.571429	
31137	0	7412.428571	9072.142857	
31139	0	1543.571429	2102.714286	
31140	0	4056.142857	4457.857143	
31141	0	4128.857143	5094.714286	
31142	0	2581.714286	2850.857143	
31143	0	7476.571429	8806.714286	
31144	0	2862.000000	3169.857143	
31145	0	9016.571429	10095.428571	
31146	0	3294.571429	3714.571429	

SEQN	valid_day	PAXINTEN_sum
31128	6	345961.428571
31129	4	191478.428571
31131	6	250155.571429
31132	5	379224.000000
31133	5	172734.571429
31134	5	273869.285714
31137	5	277782.428571
31139	6	119900.428571
31140	2	211311.000000
31141	7	225231.714286
31142	1	150623.142857
31143	7	363330.714286
31144	3	110674.428571
31145	6	454646.857143
31146	0	169655.000000

[115]:
```
d_reliable.shape
```

[115]:
```
(6863, 11)
```

[116]:
```
parquetfile = datadir / f"paxraw_{CHAR_LOOKUP[year].lower()}_reliable_people.
↪parquet"
parquetfile
```

[116]:
```
PosixPath('/Users/mm51929/projects/2022/07-nhanes-
analysis/data/raw/2005-2006/paxraw_d_reliable_people.parquet')
```

[117]:
```
d_reliable.to_parquet(parquetfile)
```

6.5 Use filtered people to select rows from full data

[118]:
```
paxraw_reliable = paxraw.merge(
    worn_minutes.loc[worn_minutes.valid_day == 1, :], on=["SEQN", "PAXDAY"]
).merge(d_reliable.loc[:, []], how="inner", on="SEQN")
paxraw_reliable.head(10)
```

[118]:

	SEQN	PAXSTAT	PAXCAL	PAXDAY	PAXN	PAXHOUR	PAXMINUT	PAXINTEN	PAXSTEP	\
0	31128	1	1	1	1	0	0	166	4	
1	31128	1	1	1	2	0	1	27	0	
2	31128	1	1	1	3	0	2	0	0	
3	31128	1	1	1	4	0	3	276	4	
4	31128	1	1	1	5	0	4	0	0	
5	31128	1	1	1	6	0	5	0	0	
6	31128	1	1	1	7	0	6	0	0	
7	31128	1	1	1	8	0	7	0	0	
8	31128	1	1	1	9	0	8	0	0	
9	31128	1	1	1	10	0	9	0	0	

	MET	worn	vigorous_bout	moderate_bout	light_bout	low_bout	\
0	1.100000	1.0	0.0	0.0	0.0	0.0	
1	1.000000	1.0	0.0	0.0	0.0	0.0	
2	1.000000	1.0	0.0	0.0	0.0	0.0	
3	1.266667	1.0	0.0	0.0	0.0	0.0	
4	1.000000	1.0	0.0	0.0	0.0	0.0	
5	1.000000	1.0	0.0	0.0	0.0	0.0	
6	1.000000	1.0	0.0	0.0	0.0	0.0	
7	1.000000	1.0	0.0	0.0	0.0	0.0	
8	1.000000	1.0	0.0	0.0	0.0	0.0	
9	1.000000	1.0	0.0	0.0	0.0	0.0	

	sed_bout	no_bout	intensity	METh	activeMETh	max_intensity	\
0	0.0	1	1	0.018333	0.001667	0	
1	0.0	1	0	0.016667	0.000000	0	

```
2       0.0         1          0  0.016667    0.000000              0
3       0.0         1          1  0.021111    0.004444              0
4       0.0         1          0  0.016667    0.000000              0
5       0.0         1          0  0.016667    0.000000              0
6       0.0         1          0  0.016667    0.000000              0
7       0.0         1          0  0.016667    0.000000              0
8       0.0         1          0  0.016667    0.000000              0
9       0.0         1          0  0.016667    0.000000              0

   out_of_calibration  unreliable  zero_steps_with_intensity  too_many_steps  \
0                   0           0                          0               0
1                   0           0                          0               0
2                   0           0                          0               0
3                   0           0                          0               0
4                   0           0                          0               0
5                   0           0                          0               0
6                   0           0                          0               0
7                   0           0                          0               0
8                   0           0                          0               0
9                   0           0                          0               0

   steps_filtered_500  steps_filtered_300  worn_sum  valid_day
0                   0                   0    1276.0          1
1                   0                   0    1276.0          1
2                   0                   0    1276.0          1
3                   0                   0    1276.0          1
4                   0                   0    1276.0          1
5                   0                   0    1276.0          1
6                   0                   0    1276.0          1
7                   0                   0    1276.0          1
8                   0                   0    1276.0          1
9                   0                   0    1276.0          1
```

[119]:
```
paxraw_reliable.shape
```

[119]: (46186041, 29)

[120]:
```
paxraw_reliable.shape
```

[120]: (46186041, 29)

6.6 Save parquet

[121]:
```
parquetfile = datadir / f"paxraw_{CHAR_LOOKUP[year].
  ↪lower()}_met_worn_bouts_reliable.parquet"
parquetfile
```

[121]:
```
PosixPath('/Users/mm51929/projects/2022/07-nhanes-
analysis/data/raw/2005-2006/paxraw_d_met_worn_bouts_reliable.parquet')
```

[122]:
```
paxraw_reliable.to_parquet(parquetfile)
```

7 Look at intensity distribution and METh thresholds

7.0.1 Group by intensity to sum MET-h levels across days

[123]:
```
groupedMETh = (
    paxraw_sample.groupby(["intensity", "PAXDAY"])
    .agg({"activeMETh": np.sum})
    .groupby(["intensity"])
    .agg({"activeMETh": np.mean})
)
groupedMETh
```

[123]:
```
           activeMETh
intensity
0            0.000000
1            1.969329
2            3.158906
3            1.377102
4            0.024444
```

[124]:
```
groupedMETh.sum()
```

[124]:
```
activeMETh    6.52978
dtype: float64
```

This is the same as just taking the mean of the sum (without grouping by intensity in the middle):

[125]:
```
paxraw_sample.groupby(["PAXDAY"]).agg({"activeMETh": np.sum}).mean()
```

[125]:
```
activeMETh    6.52978
dtype: float64
```

[126]:
```
paxraw_reliable.groupby(["SEQN", "intensity", "PAXDAY"]).agg({"activeMETh": np.
↪sum}).head()
```

[126]:
```
                       activeMETh
SEQN  intensity PAXDAY
31128 0         1             0.0
                2             0.0
                3             0.0
                4             0.0
                5             0.0
```

[127]:
```
paxraw_reliable.groupby(["SEQN", "intensity", "PAXDAY"]).agg({"activeMETh": np.
  ↪sum}).groupby(
    ["SEQN", "intensity"]
).agg({"activeMETh": np.mean}).head()
```

[127]:
```
                  activeMETh
SEQN  intensity
31128 0             0.000000
      1             1.738496
      2             2.886298
      3             1.257598
      4             0.024444
```

[128]:
```
minutes_at_intensity = (
    paxraw_reliable.groupby(["SEQN", "intensity", "PAXDAY"])
    .agg({"activeMETh": "count"})
    .groupby(["SEQN", "intensity"])
    .agg({"activeMETh": np.mean})
    .groupby(["intensity"])
    .agg({"activeMETh": np.mean})
)
minutes_at_intensity
```

[128]:
```
            activeMETh
intensity
0           763.929099
1           182.825856
2            66.517585
3            21.937875
4             1.396948
```

[129]:
```
METh_at_intensity = (
    paxraw_reliable.groupby(["SEQN", "intensity", "PAXDAY"])
    .agg({"activeMETh": np.sum})
    .groupby(["SEQN", "intensity"])
    .agg({"activeMETh": np.mean})
    .groupby(["intensity"])
    .agg({"activeMETh": np.mean})
)
METh_at_intensity
```

[129]:
```
            activeMETh
intensity
0             0.000000
1             1.066732
2             1.711808
3             1.147303
```

```
4            0.127616
```

[130]:
```
minutes_METh = minutes_at_intensity.rename(columns={"activeMETh": "minutes"}).
 ↪merge(
    METh_at_intensity, how="inner", on="intensity"
)
minutes_METh
```

[130]:

intensity	minutes	activeMETh
0	763.929099	0.000000
1	182.825856	1.066732
2	66.517585	1.711808
3	21.937875	1.147303
4	1.396948	0.127616

[131]:
```
minutes_METh_stack = (
    pd.concat(
        [
            minutes_at_intensity.assign(metric="minutes"),
            METh_at_intensity.assign(metric="MET", activeMETh=lambda d: d.
 ↪activeMETh * 60),
        ]
    )
    .reset_index()
    .merge(
        pd.DataFrame({"label": labels, "intensity": range(5)}),
        how="left",
        on="intensity",
    )
)
minutes_METh_stack
```

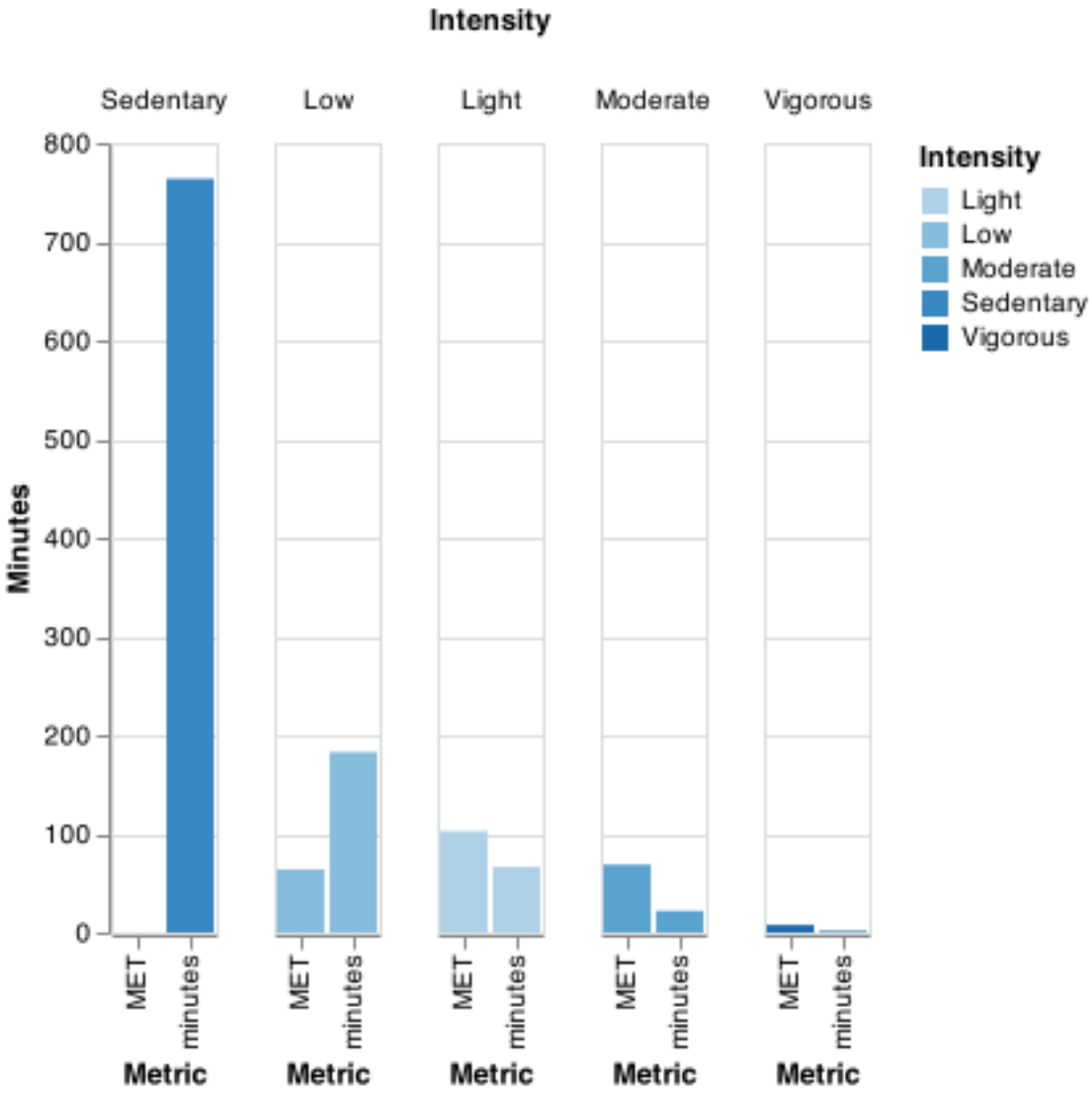
[131]:

	intensity	activeMETh	metric	label
0	0	763.929099	minutes	Sedentary
1	1	182.825856	minutes	Low
2	2	66.517585	minutes	Light
3	3	21.937875	minutes	Moderate
4	4	1.396948	minutes	Vigorous
5	0	0.000000	MET	Sedentary
6	1	64.003916	MET	Low
7	2	102.708509	MET	Light
8	3	68.838166	MET	Moderate
9	4	7.656979	MET	Vigorous

7.0.2 MET Minutes vs Minutes by intensity level

[132]:
```
alt.Chart(minutes_METh_stack).mark_bar().encode(
    x=alt.X("metric:N", title="Metric"),
    y=alt.Y("activeMETh:Q", title="Minutes"),
    color=alt.Color("label:O", title="Intensity"),
    column=alt.Column(
        "label:O", title="Intensity", sort=alt.SortField("intensity",␣
 ↪order="ascending")
    ),
)
```

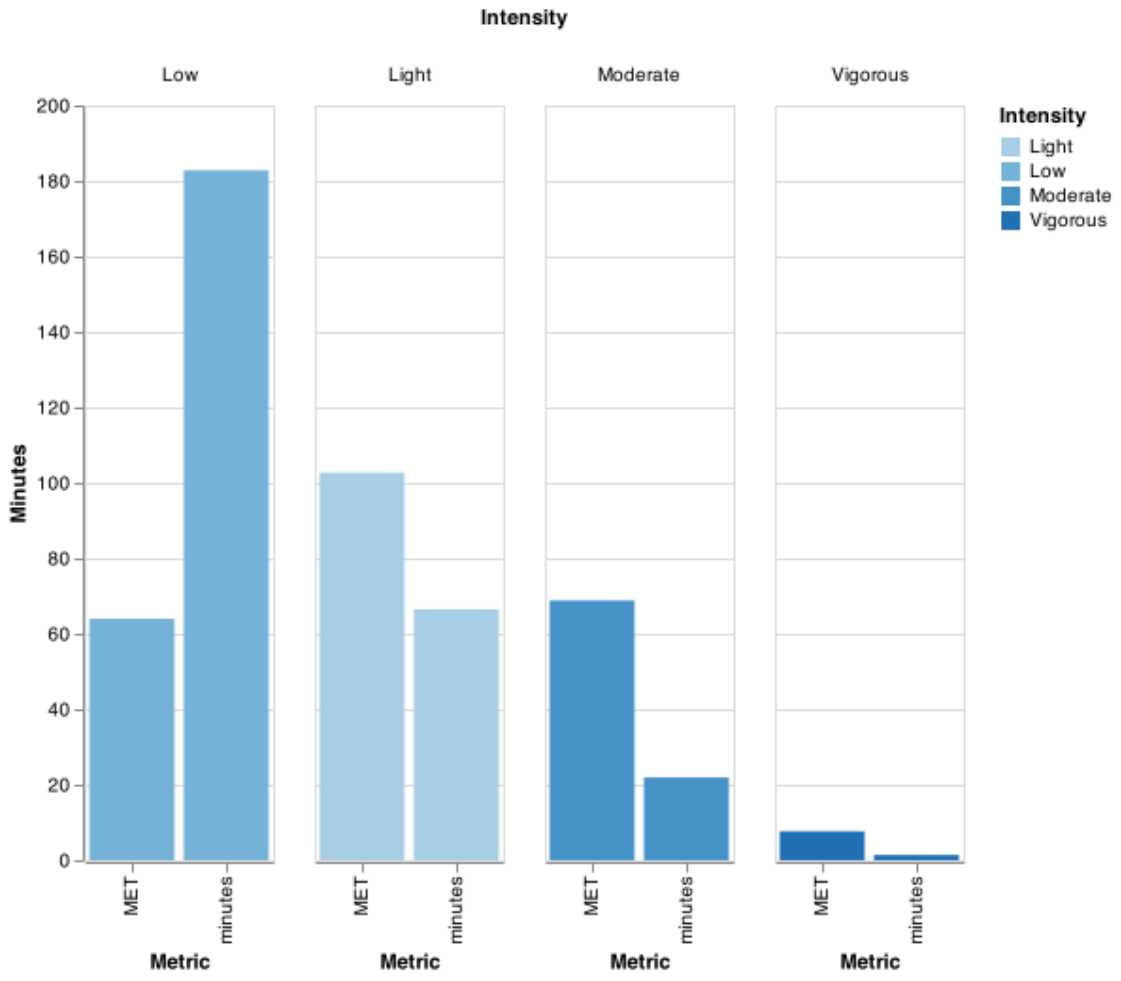
[132]:



Skip sedentary - no METs

[133]:
```
alt.Chart(minutes_METh_stack.loc[minutes_METh_stack.intensity > 0, :]).
 ↪mark_bar().encode(
    x=alt.X("metric:N", title="Metric"),
    y=alt.Y("activeMETh:Q", title="Minutes"),
    color=alt.Color("label:O", title="Intensity"),
    column=alt.Column(
        "label:O", title="Intensity", sort=alt.SortField("intensity",␣
 ↪order="ascending")
    ),
).properties(width=100, height=400)
```

[133]:



7.1 Distribution of weekly METh

[134]:
```
(
    paxraw_reliable.groupby(["SEQN", "intensity", "PAXDAY"])
    .agg({"activeMETh": np.sum})
    .groupby(["SEQN", "intensity"])
    .agg({"activeMETh": np.mean})
    .groupby(["SEQN"])
    .agg({"activeMETh": np.sum})
).mean() * 7
```

[134]:
```
activeMETh    28.374216
dtype: float64
```

[135]:
```
METh = (
    paxraw_reliable.groupby(["SEQN", "intensity", "PAXDAY"])
    .agg({"activeMETh": np.sum})
    .groupby(["SEQN", "intensity"])
    .agg({"activeMETh": np.mean})
    .groupby(["SEQN"])
    .agg({"activeMETh": np.sum})
).assign(weeklyMETh=lambda d: d.activeMETh * 7)
METh
```

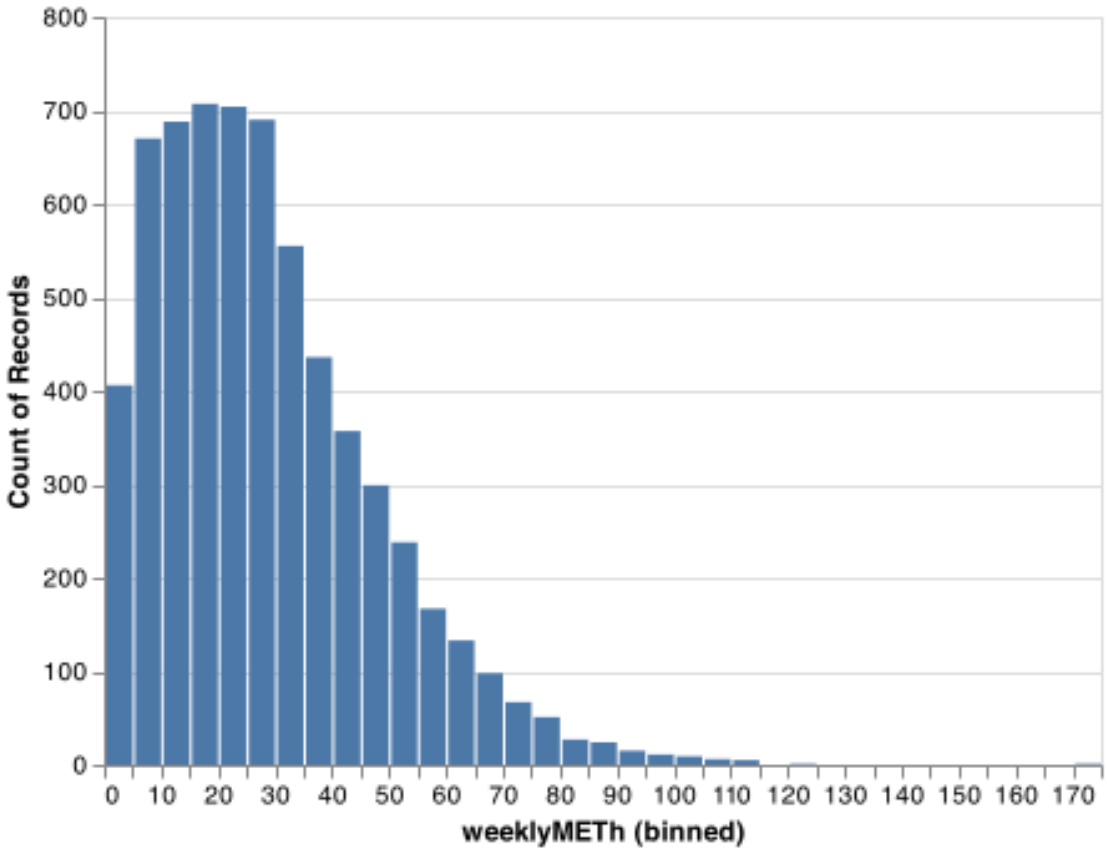
[135]:

SEQN	activeMETh	weeklyMETh
31128	5.906835	41.347847
31129	2.800030	19.600213
31131	4.020895	28.146262
31132	5.833550	40.834849
31133	2.700036	18.900250
…	…	…
41468	2.945727	20.620088
41471	7.180111	50.260778
41472	4.825957	33.781700
41473	2.008668	14.060679
41474	4.896020	34.272141

[6365 rows x 2 columns]

[136]:
```
alt.Chart(METh).mark_bar().encode(
    alt.X("weeklyMETh:Q", bin=alt.BinParams(maxbins=35)),
    y="count()",
)
```

[136]:



7.2 Look at different METh thresholds and % of people getting rewards (and total utilization)

[137]:
```
cut, cuts = pd.qcut(METh.weeklyMETh, 10, retbins=True)
cut
```

[137]:
```
SEQN
31128    (35.438, 43.172]
31129     (16.13, 20.589]
31131    (25.071, 29.654]
31132    (35.438, 43.172]
31133     (16.13, 20.589]
                ...
41468    (20.589, 25.071]
41471    (43.172, 54.424]
41472    (29.654, 35.438]
41473     (11.381, 16.13]
41474    (29.654, 35.438]
Name: weeklyMETh, Length: 6365, dtype: category
Categories (10, interval[float64, right]): [(0.027899999999999998, 6.676] <
```

```
(6.676, 11.381] < (11.381, 16.13] < (16.13, 20.589] … (29.654, 35.438] <
(35.438, 43.172] < (43.172, 54.424] < (54.424, 173.429]]
```

[138]:
```
cuts
```

[138]:
```
array([2.88798701e-02, 6.67554617e+00, 1.13810837e+01, 1.61297168e+01,
       2.05892696e+01, 2.50709474e+01, 2.96544436e+01, 3.54380840e+01,
       4.31720367e+01, 5.44241588e+01, 1.73429322e+02])
```

[139]:
```
cut.cat.categories
```

[139]:
```
IntervalIndex([(0.027899999999999998, 6.676], (6.676, 11.381], (11.381, 16.13],
(16.13, 20.589], (20.589, 25.071], (25.071, 29.654], (29.654, 35.438], (35.438,
43.172], (43.172, 54.424], (54.424, 173.429]], dtype='interval[float64, right]')
```

[140]:
```
utilization = pd.DataFrame(
    {
        "target": np.linspace(cuts[1], cuts[-2], num=50),
        "Total Utilization": [
            np.minimum(METh.weeklyMETh.values / x, 1).mean()
            for x in np.linspace(cuts[1], cuts[-2], num=50)
        ],
        "Max Rewards": [
            (METh.weeklyMETh.values / x >= 1).sum() / METh.shape[0]
            for x in np.linspace(cuts[1], cuts[-2], num=50)
        ],
    }
)
```

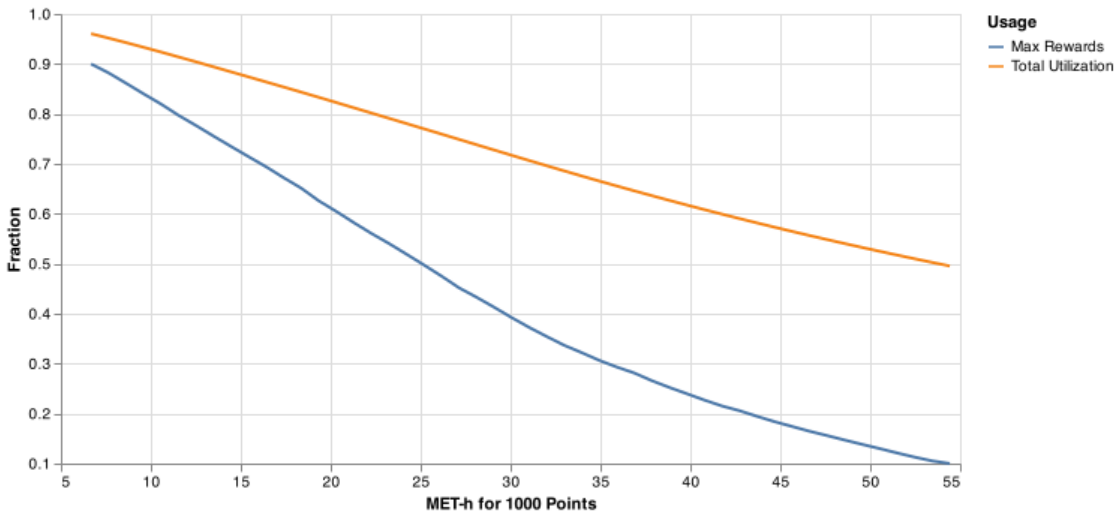
[141]:
```
utilization.head()
```

[141]:

	target	Total Utilization	Max Rewards
0	6.675546	0.960492	0.899921
1	7.650008	0.951684	0.882011
2	8.624469	0.942631	0.861273
3	9.598931	0.933258	0.840063
4	10.573392	0.923710	0.819482

[142]:
```
alt.Chart(utilization).mark_line().transform_fold(
    fold=["Total Utilization", "Max Rewards"], as_=["variable", "value"]
).encode(
    alt.X("target:Q", title="MET-h for 1000 Points"),
    alt.Y("value:Q", title="Fraction", scale=alt.Scale(zero=False)),
    alt.Color("variable:N", title="Usage"),
).properties(
    width=600
)
```

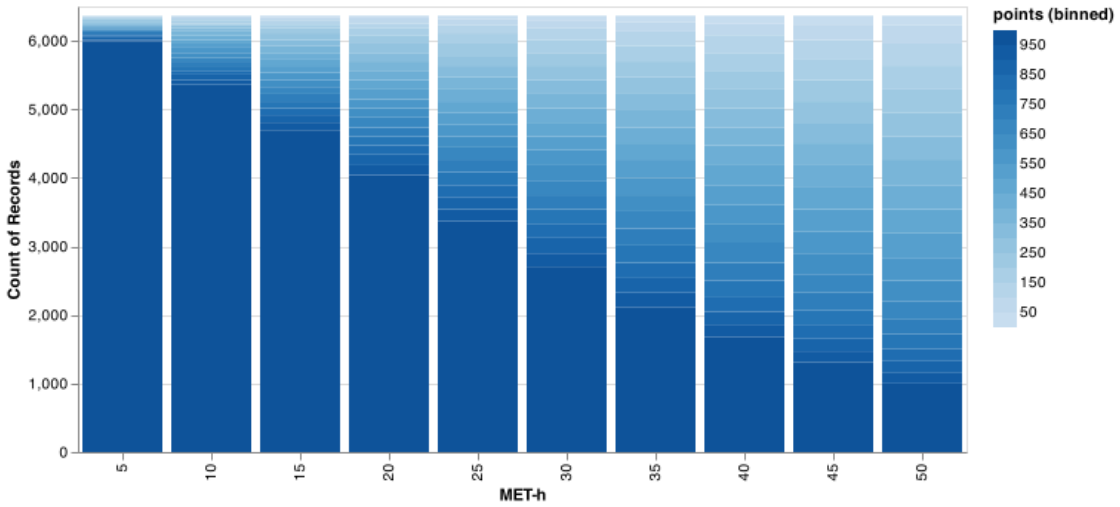
[142]:



[143]:

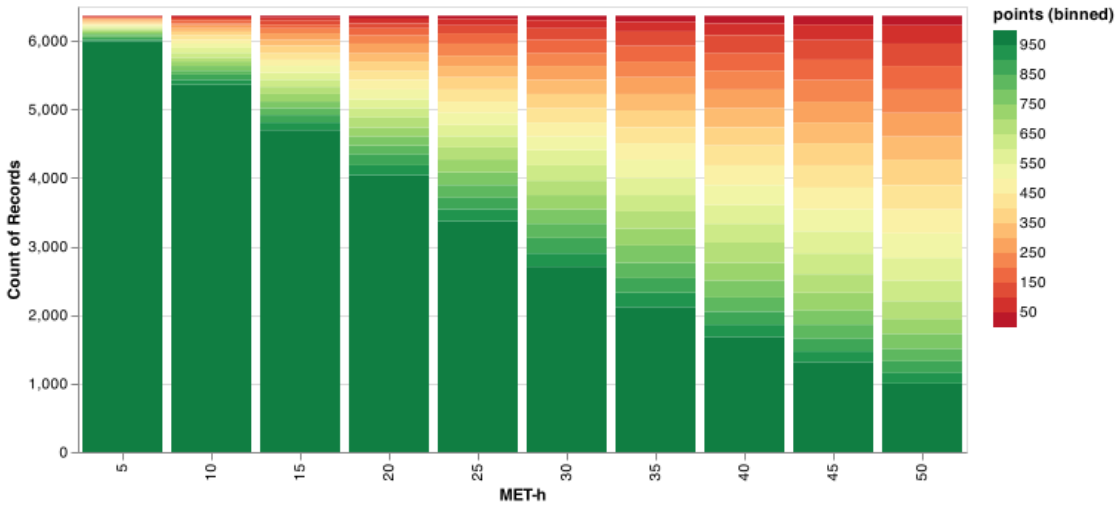
```
alt.Chart(
    pd.concat(
        [
            pd.DataFrame({"points": np.minimum(METh.weeklyMETh.values / x, 1) *␣
↪1000, "MET-h": x})
            for x in np.arange(5, 55, 5)
        ]
    )
).mark_bar().encode(
    alt.Color("points:Q", bin=alt.BinParams(maxbins=20)),
    alt.Y("count()"),
    alt.X("MET-h:O"),
).properties(
    width=600
)
```

[143]:



[144]:
```
alt.Chart(
    pd.concat(
        [
            pd.DataFrame({"points": np.minimum(METh.weeklyMETh.values / x, 1) *␣
↪1000, "MET-h": x})
            for x in np.arange(5, 55, 5)
        ]
    )
).mark_bar().encode(
    alt.Color(
        "points:Q",
        bin=alt.BinParams(maxbins=20),
        scale=alt.Scale(scheme="redyellowgreen"),
    ),
    alt.Y("count()"),
    alt.X("MET-h:O"),
).properties(
    width=600
)
```

[144]:



7.3 Ena's chart of points by intensity for different amounts of points earned

[145]:
```
all_intensities = pd.DataFrame(
    {
        "SEQN": np.repeat(pd.unique(paxraw_reliable.SEQN.values), 5),
        "intensity": np.tile(np.arange(5), pd.unique(paxraw_reliable.SEQN.
 ↪values).shape[0]),
    }
)
all_intensities.head(11)
```

[145]:

	SEQN	intensity
0	31128	0
1	31128	1
2	31128	2
3	31128	3
4	31128	4
5	31129	0
6	31129	1
7	31129	2
8	31129	3
9	31129	4
10	31131	0

[146]:
```
points_by_intensity = (
    paxraw_reliable.groupby(["SEQN", "intensity", "PAXDAY"], dropna=False)
    .agg({"activeMETh": np.sum, "worn": np.sum})
    .groupby(["SEQN", "intensity"], dropna=False)
    .agg({"activeMETh": np.mean, "worn": np.mean})
```

```
    # fill in blank intensities
    .merge(all_intensities, how="outer", on=list(all_intensities.columns))
    .fillna(0)
    # calc points
    .assign(weeklyMETh=lambda d: d.activeMETh * 7, points=lambda d: d.
 ↪weeklyMETh / 25 * 1000)
    .rename(columns={"worn": "dailyMinutes"})
    .reset_index()
)
points_by_intensity.head()
```

[146]:

	index	SEQN	intensity	activeMETh	dailyMinutes	weeklyMETh	points
0	0	31128	0	0.000000	566.714286	0.000000	0.000000
1	1	31128	1	1.738496	273.142857	12.169470	486.778788
2	2	31128	2	2.886298	111.571429	20.204087	808.163492
3	3	31128	3	1.257598	25.571429	8.803185	352.127419
4	4	31128	4	0.024444	0.285714	0.171105	6.844185

[147]:
```
points_by_person = (
    points_by_intensity.groupby("SEQN")
    .agg({"points": np.sum})
    .assign(
        points_capped=lambda d: np.minimum(d.points, 1000),
        point_bin=lambda d: pd.cut(d.points_capped, np.arange(11) * 100,␣
 ↪right=True),
    )
)
points_by_person
```

[147]:

SEQN	points	points_capped	point_bin
31128	1653.913884	1000.000000	(900, 1000]
31129	784.008504	784.008504	(700, 800]
31131	1125.850467	1000.000000	(900, 1000]
31132	1633.393954	1000.000000	(900, 1000]
31133	756.009998	756.009998	(700, 800]
…	…	…	…
41468	824.803500	824.803500	(800, 900]
41471	2010.431113	1000.000000	(900, 1000]
41472	1351.267992	1000.000000	(900, 1000]
41473	562.427144	562.427144	(500, 600]
41474	1370.885635	1000.000000	(900, 1000]

[6365 rows x 3 columns]

[148]:
```
point_thresholds_by_intensity = (
    points_by_intensity.merge(points_by_person, how="left", on="SEQN")
```

```
    .assign(
        points_relative=lambda d: d.points_x / d.points_y,
        dailyMinutesCapped=lambda d: d.dailyMinutes * d.points_capped / d.
 ↪points_y,
    )
    .groupby(["point_bin", "intensity"], dropna=False)
    .agg(
        {
            "points_relative": np.mean,
            "points_capped": np.mean,
            "dailyMinutesCapped": np.mean,
        }
    )
    .assign(
        points=lambda d: d.points_relative * d.points_capped,
    )
    .reset_index()
    .assign(point_bin=lambda d: d.point_bin.astype("str"))
    .merge(pd.DataFrame({"label": labels, "intensity": np.arange(5)}),␣
 ↪how="left", on="intensity")
)
point_thresholds_by_intensity
```

[148]:

	point_bin	intensity	points_relative	points_capped	\
0	(0, 100]	0	0.000000	57.634948	
1	(0, 100]	1	0.526927	57.634948	
2	(0, 100]	2	0.342279	57.634948	
3	(0, 100]	3	0.126602	57.634948	
4	(0, 100]	4	0.004192	57.634948	
5	(100, 200]	0	0.000000	153.820810	
6	(100, 200]	1	0.476642	153.820810	
7	(100, 200]	2	0.369145	153.820810	
8	(100, 200]	3	0.147733	153.820810	
9	(100, 200]	4	0.006481	153.820810	
10	(200, 300]	0	0.000000	248.819090	
11	(200, 300]	1	0.415051	248.819090	
12	(200, 300]	2	0.397417	248.819090	
13	(200, 300]	3	0.175852	248.819090	
14	(200, 300]	4	0.011679	248.819090	
15	(300, 400]	0	0.000000	350.576718	
16	(300, 400]	1	0.434982	350.576718	
17	(300, 400]	2	0.388481	350.576718	
18	(300, 400]	3	0.166080	350.576718	
19	(300, 400]	4	0.010458	350.576718	
20	(400, 500]	0	0.000000	449.786994	
21	(400, 500]	1	0.406409	449.786994	
22	(400, 500]	2	0.408429	449.786994	

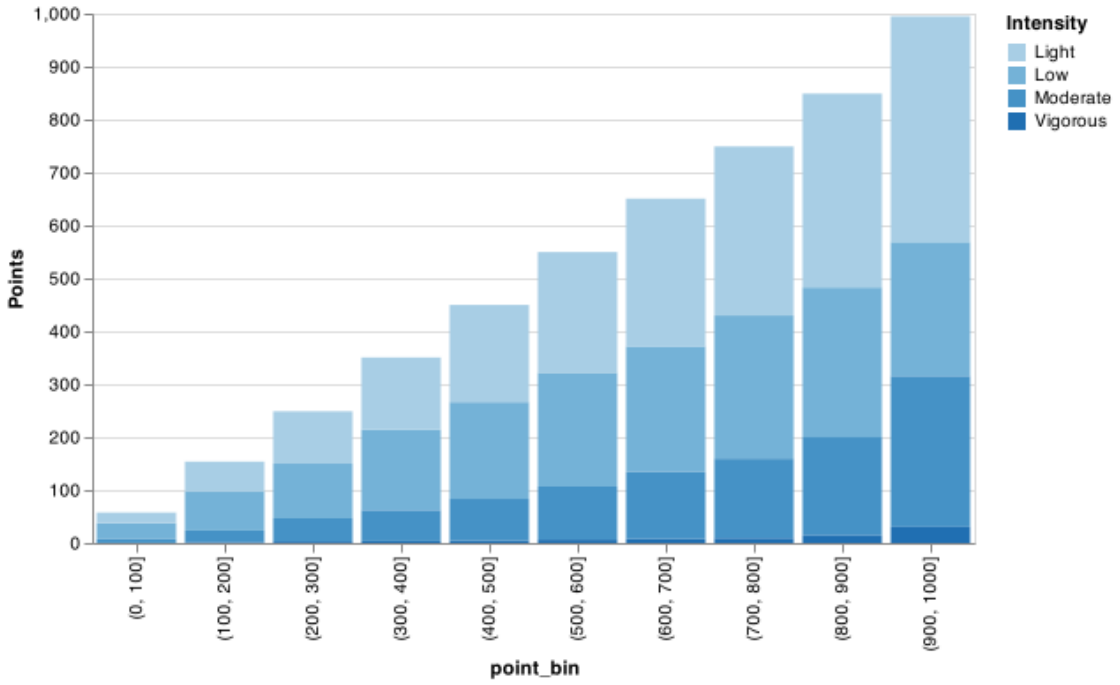
23	(400, 500]	3	0.174974	449.786994
24	(400, 500]	4	0.010188	449.786994
25	(500, 600]	0	0.000000	549.650318
26	(500, 600]	1	0.391661	549.650318
27	(500, 600]	2	0.414225	549.650318
28	(500, 600]	3	0.182648	549.650318
29	(500, 600]	4	0.011466	549.650318
30	(600, 700]	0	0.000000	650.594432
31	(600, 700]	1	0.365191	650.594432
32	(600, 700]	2	0.428574	650.594432
33	(600, 700]	3	0.194465	650.594432
34	(600, 700]	4	0.011770	650.594432
35	(700, 800]	0	0.000000	749.117466
36	(700, 800]	1	0.361057	749.117466
37	(700, 800]	2	0.426217	749.117466
38	(700, 800]	3	0.200695	749.117466
39	(700, 800]	4	0.012031	749.117466
40	(800, 900]	0	0.000000	848.989746
41	(800, 900]	1	0.333225	848.989746
42	(800, 900]	2	0.431347	848.989746
43	(800, 900]	3	0.218446	848.989746
44	(800, 900]	4	0.016982	848.989746
45	(900, 1000]	0	0.000000	995.095917
46	(900, 1000]	1	0.254820	995.095917
47	(900, 1000]	2	0.428406	995.095917
48	(900, 1000]	3	0.286160	995.095917
49	(900, 1000]	4	0.030614	995.095917

	dailyMinutesCapped	points	label
0	169.039514	0.000000	Sedentary
1	27.462006	30.369437	Low
2	2.873354	19.727212	Light
3	0.491388	7.296681	Moderate
4	0.009119	0.241617	Vigorous
5	219.302965	0.000000	Sedentary
6	57.035040	73.317420	Low
7	8.267925	56.782143	Light
8	1.657682	22.724409	Moderate
9	0.041509	0.996838	Vigorous
10	224.849279	0.000000	Sedentary
11	75.412407	103.272682	Low
12	14.190913	98.884989	Light
13	3.079511	43.755441	Moderate
14	0.115334	2.905979	Vigorous
15	296.592253	0.000000	Sedentary
16	106.703457	152.494439	Low
17	19.742191	136.192287	Light

```
18            4.056643   58.223746   Moderate
19            0.146606    3.666247   Vigorous
20          315.754686    0.000000  Sedentary
21          123.906289  182.797604        Low
22           26.389838  183.705913      Light
23            5.533944   78.701168   Moderate
24            0.176177    4.582309   Vigorous
25          326.566460    0.000000  Sedentary
26          143.342857  215.276659        Low
27           32.789234  227.678810      Light
28            6.975983  100.392765   Moderate
29            0.244720    6.302084   Vigorous
30          326.076989    0.000000  Sedentary
31          153.113772  237.591530        Low
32           39.966210  278.827643      Light
33            8.834046  126.517908   Moderate
34            0.299401    7.657351   Vigorous
35          365.257373    0.000000  Sedentary
36          172.365377  270.474387        Low
37           45.692072  319.286757      Light
38           10.534661  150.344050   Moderate
39            0.353121    9.012272   Vigorous
40          383.560471    0.000000  Sedentary
41          178.532873  282.904380        Low
42           51.874594  366.209216      Light
43           12.932630  185.458505   Moderate
44            0.571834   14.417645   Vigorous
45          263.271371    0.000000  Sedentary
46          151.626531  253.570296        Low
47           59.097873  426.304864      Light
48           19.472241  284.756462   Moderate
49            1.198353   30.464296   Vigorous
```

[149]:
```
alt.Chart(
    point_thresholds_by_intensity.loc[point_thresholds_by_intensity.intensity >␣
 ↪0, :]
).mark_bar().encode(
    alt.X("point_bin:O"),
    alt.Y("points:Q", title="Points"),
    alt.Color("label:O", title="Intensity", sort=alt.SortField("label",␣
 ↪order="ascending")),
).properties(
    width=500
)
```

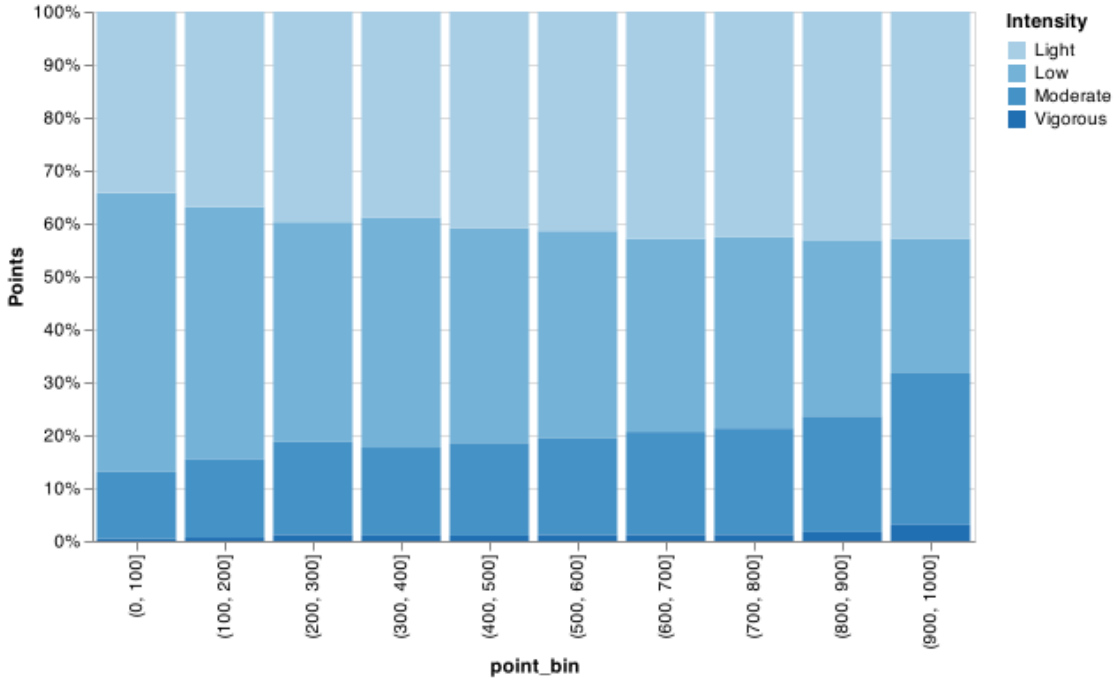
[149]:



7.3.1 Stack the bar chart

[150]:
```
alt.Chart(
    point_thresholds_by_intensity.loc[point_thresholds_by_intensity.intensity >␣
 ↪0, :]
).mark_bar().encode(
    alt.X("point_bin:O"),
    alt.Y("points:Q", title="Points", stack="normalize"),
    alt.Color("label:O", title="Intensity", sort=alt.SortField("label",␣
 ↪order="ascending")),
).properties(
    width=500
)
```

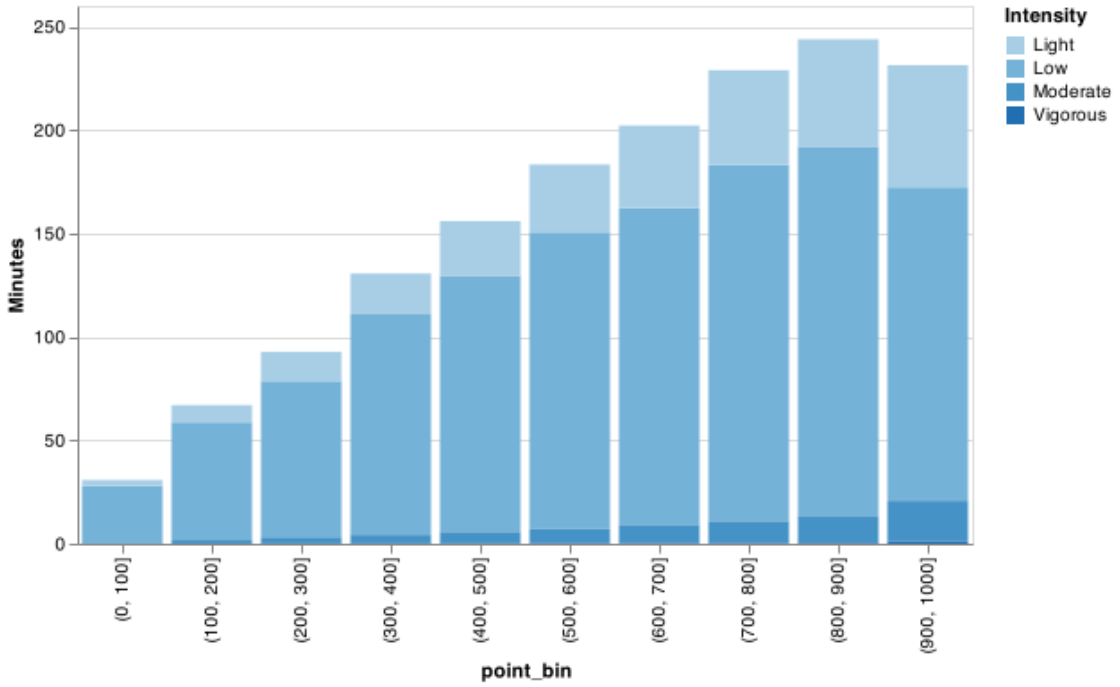
[150]:



7.3.2 Convert this to daily times in zones

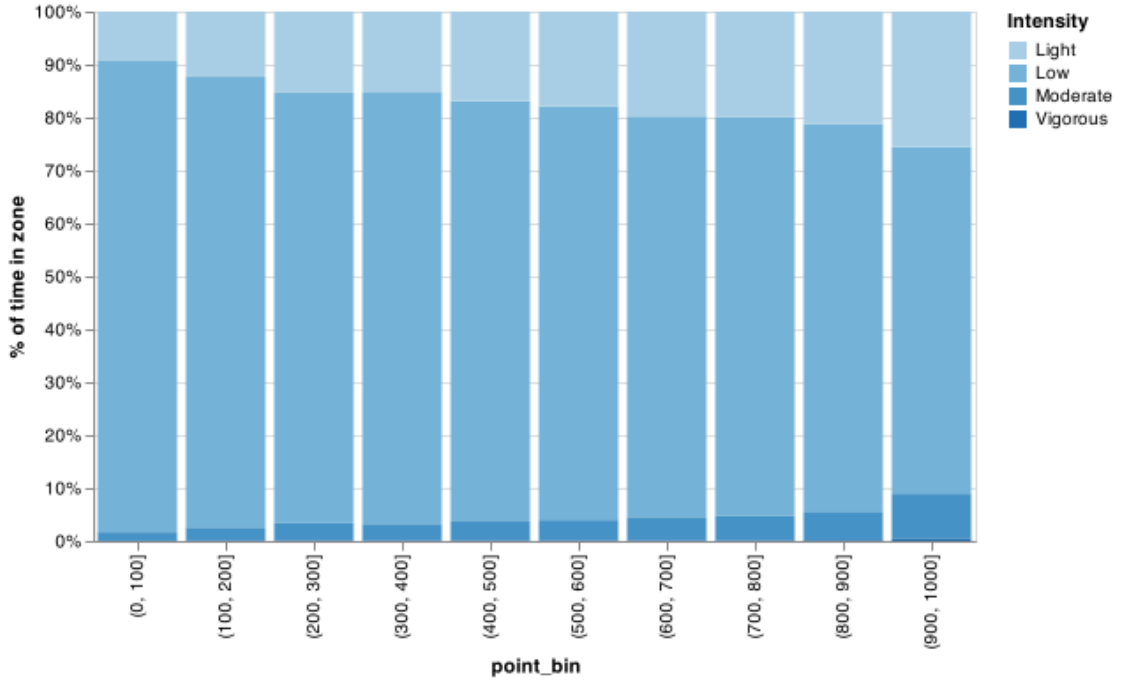
[151]:
```
alt.Chart(
    point_thresholds_by_intensity.loc[point_thresholds_by_intensity.intensity >␣
 ↪0, :]
).mark_bar().encode(
    alt.X("point_bin:O"),
    alt.Y("dailyMinutesCapped:Q", title="Minutes"),
    alt.Detail("label:O", title="Intensity"),
    alt.Color("label:O", title="Intensity", sort=alt.SortField("label",␣
 ↪order="ascending")),
    tooltip=["label", "dailyMinutesCapped", "point_bin", "points"],
).properties(
    width=500
)
```

[151]:



[152]:
```
alt.Chart(
    point_thresholds_by_intensity.loc[point_thresholds_by_intensity.intensity >␣
 ↪0, :]
).mark_bar().encode(
    alt.X("point_bin:O"),
    alt.Y("dailyMinutesCapped:Q", title="% of time in zone", stack="normalize"),
    alt.Detail("label:O", title="Intensity"),
    alt.Color("label:O", title="Intensity", sort=alt.SortField("label",␣
 ↪order="ascending")),
    tooltip=["label", "dailyMinutesCapped", "point_bin", "points"],
).properties(
    width=500
)
```

[152]:



[153]:
```
point_thresholds_by_intensity.loc[point_thresholds_by_intensity.intensity > 0, :
↪].pivot(
    index="point_bin", columns="label", values="dailyMinutesCapped"
)
```

[153]:

label	Light	Low	Moderate	Vigorous
point_bin				
(0, 100]	2.873354	27.462006	0.491388	0.009119
(100, 200]	8.267925	57.035040	1.657682	0.041509
(200, 300]	14.190913	75.412407	3.079511	0.115334
(300, 400]	19.742191	106.703457	4.056643	0.146606
(400, 500]	26.389838	123.906289	5.533944	0.176177
(500, 600]	32.789234	143.342857	6.975983	0.244720
(600, 700]	39.966210	153.113772	8.834046	0.299401
(700, 800]	45.692072	172.365377	10.534661	0.353121
(800, 900]	51.874594	178.532873	12.932630	0.571834
(900, 1000]	59.097873	151.626531	19.472241	1.198353

[154]:
```
point_thresholds_by_intensity.loc[point_thresholds_by_intensity.intensity > 0, :
↪].pivot(
    index="point_bin", columns="label", values="dailyMinutesCapped"
).assign(Moderate_and_Vigorous=lambda d: d.Moderate + d.Vigorous * 2).rename(
    columns={"Moderate_and_Vigorous": "Moderate_and_Vigorous".replace("_", " ")}
).loc[
    :, ["Low", "Light", "Moderate and Vigorous"]
```

```
].astype(
    "int"
)
```

[154]:
label	Low	Light	Moderate and Vigorous
point_bin			
(0, 100]	27	2	0
(100, 200]	57	8	1
(200, 300]	75	14	3
(300, 400]	106	19	4
(400, 500]	123	26	5
(500, 600]	143	32	7
(600, 700]	153	39	9
(700, 800]	172	45	11
(800, 900]	178	51	14
(900, 1000]	151	59	21

[155]:
```
point_thresholds_by_intensity.loc[point_thresholds_by_intensity.intensity > 0, :
↪].pivot(
    index="point_bin", columns="label", values="dailyMinutesCapped"
).assign(
    Moderate_and_Vigorous=lambda d: d.Moderate + d.Vigorous * 2,
    Light=lambda d: d.Low * 0.5 + d.Light,
).rename(
    columns={"Moderate_and_Vigorous": "Moderate_and_Vigorous".replace("_", " ")}
).loc[
    :, ["Light", "Moderate and Vigorous"]
].astype(
    "int"
)
```

[155]:
label	Light	Moderate and Vigorous
point_bin		
(0, 100]	16	0
(100, 200]	36	1
(200, 300]	51	3
(300, 400]	73	4
(400, 500]	88	5
(500, 600]	104	7
(600, 700]	116	9
(700, 800]	131	11
(800, 900]	141	14
(900, 1000]	134	21

8 Additional plots

8.1 Aggregate to daily

[156]:
```
by_day = (
    paxraw.groupby(["SEQN", "PAXDAY"])
    .agg({"PAXSTEP": [sum], "PAXINTEN": [sum, np.mean, max]})
    .reset_index()
)
by_day.head()
```

[156]:

	SEQN	PAXDAY	PAXSTEP	PAXINTEN		
			sum	sum	mean	max
0	31128	1	12668	377456	262.122222	4873
1	31128	2	11920	308309	214.103472	4166
2	31128	3	11169	324734	225.509722	3644
3	31128	4	6824	229846	159.615278	5190
4	31128	5	11342	304957	211.775694	7058

[157]:
```
by_day.shape
```

[157]:
```
(52056, 6)
```

[158]:
```
# by_day.columns = by_day.columns.get_level_values(0)
by_day.columns = flatten_columns(by_day.columns)
```

[159]:
```
by_day.loc[by_day.SEQN == 31128.0, :]
```

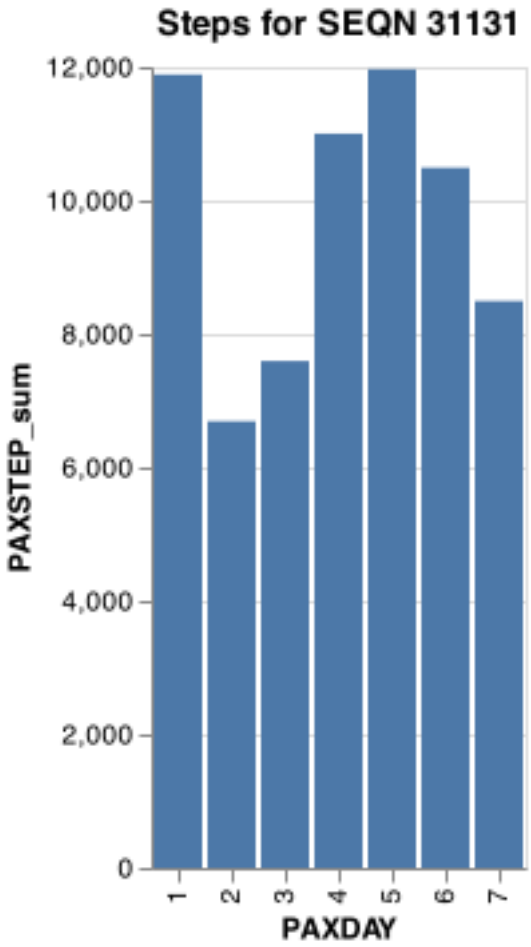
[159]:

	SEQN	PAXDAY	PAXSTEP_sum	PAXINTEN_sum	PAXINTEN_mean	PAXINTEN_max
0	31128	1	12668	377456	262.122222	4873
1	31128	2	11920	308309	214.103472	4166
2	31128	3	11169	324734	225.509722	3644
3	31128	4	6824	229846	159.615278	5190
4	31128	5	11342	304957	211.775694	7058
5	31128	6	15475	388323	269.668750	5933
6	31128	7	15712	488105	338.961806	6722

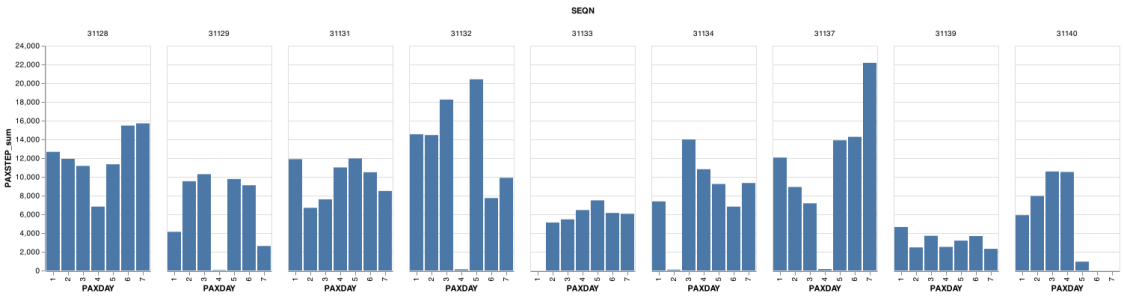
8.2 Daily charts

[160]:
```
id = 2
alt.Chart(by_day.loc[by_day.SEQN == by_day.SEQN.unique()[id], :]).mark_bar().
  ↪encode(
    x="PAXDAY:O", y="PAXSTEP_sum"
).properties(title=f"Steps for SEQN {by_day.SEQN.unique()[id]}")
```

[160]:



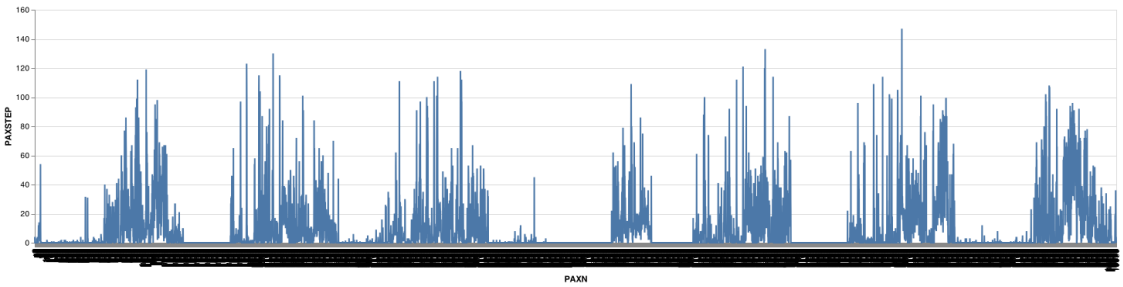
[161]:
```
alt.Chart(by_day.loc[by_day.SEQN.isin(by_day.SEQN.unique()[:9]), :]).mark_bar().
  ↪encode(
    x="PAXDAY:O", y="PAXSTEP_sum", column="SEQN:N"
).properties()
```

[161]:


8.3 Individual activity/steps

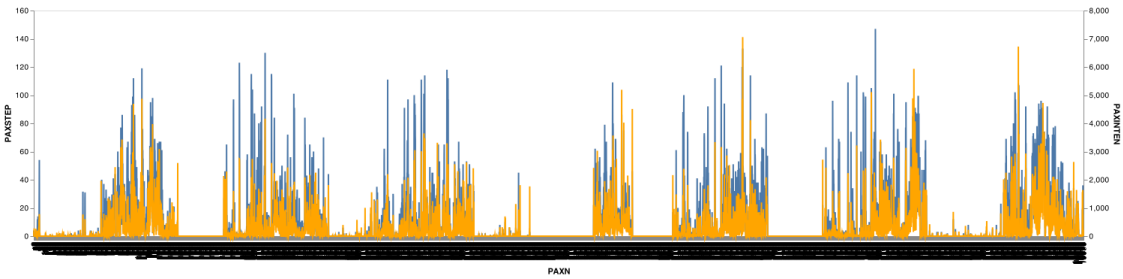
[162]:
```
alt.Chart(paxraw.loc[paxraw.SEQN == by_day.SEQN.unique()[0], :]).mark_line().
↪encode(
    x="PAXN:O", y="PAXSTEP"
).properties(width=1400)
```

[162]:



[163]:
```
(
    alt.Chart(paxraw.loc[paxraw.SEQN == by_day.SEQN.unique()[0], :])
    .mark_line()
    .encode(x="PAXN:O", y="PAXSTEP")
    + alt.Chart(paxraw.loc[paxraw.SEQN == by_day.SEQN.unique()[0], :])
    .mark_line(color="orange")
    .encode(x="PAXN:O", y="PAXINTEN")
).properties(width=1400).resolve_scale(y="independent")
```

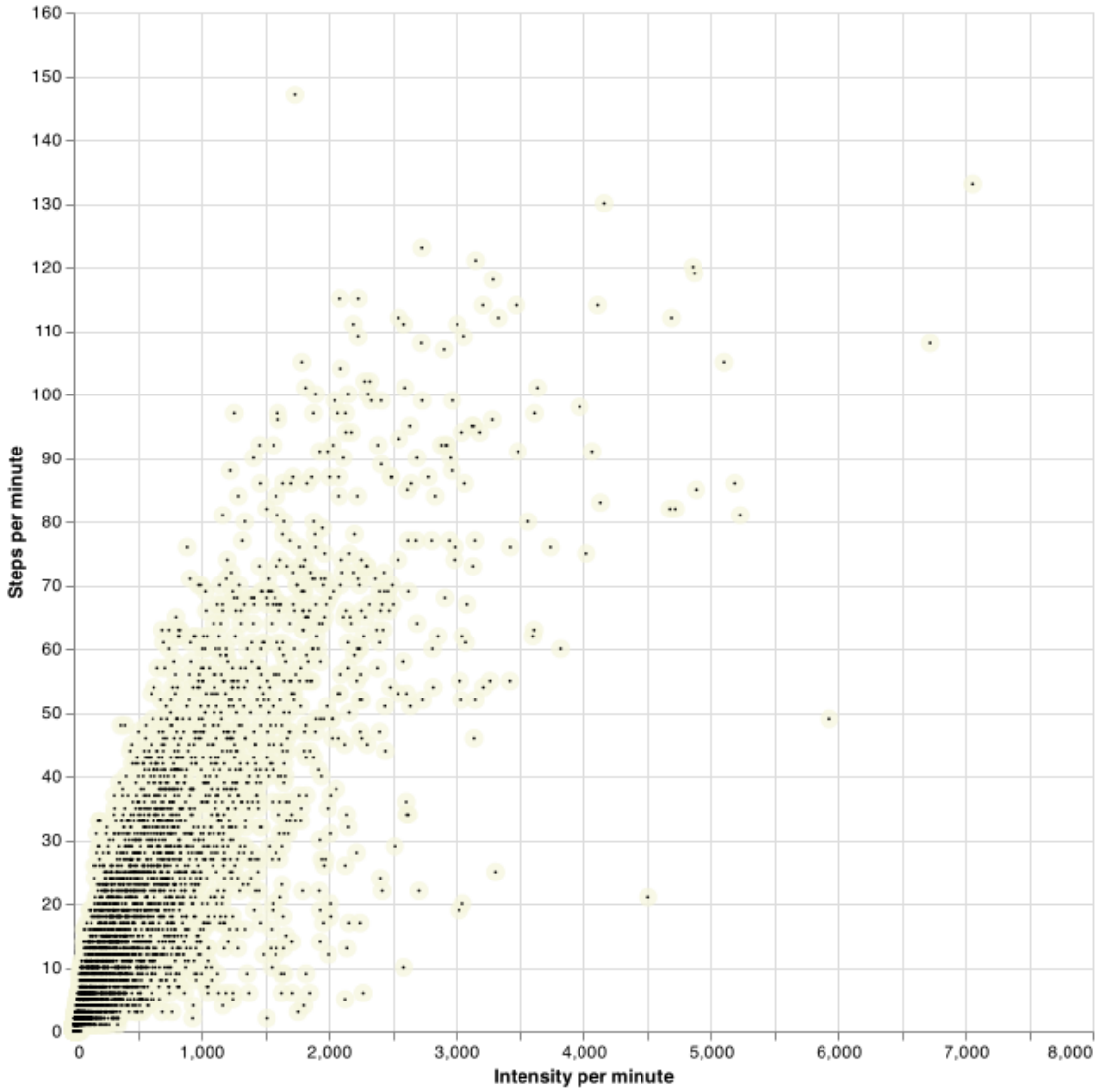
[163]:



[164]:
```
(
    alt.Chart(paxraw.loc[paxraw.SEQN == by_day.SEQN.unique()[0], :])
    .mark_circle(opacity=0.7, color="#F5F5DC", size=120)
    .encode(
        alt.X("PAXINTEN:Q", title="Intensity per minute"),
        alt.Y("PAXSTEP:Q", title="Steps per minute"),
    )
    + alt.Chart(paxraw.loc[paxraw.SEQN == by_day.SEQN.unique()[0], :])
    .mark_circle(opacity=1, color="black", size=3)
```

```
    .encode(x="PAXINTEN", y="PAXSTEP")
).properties(width=600, height=600)
```

[164]:

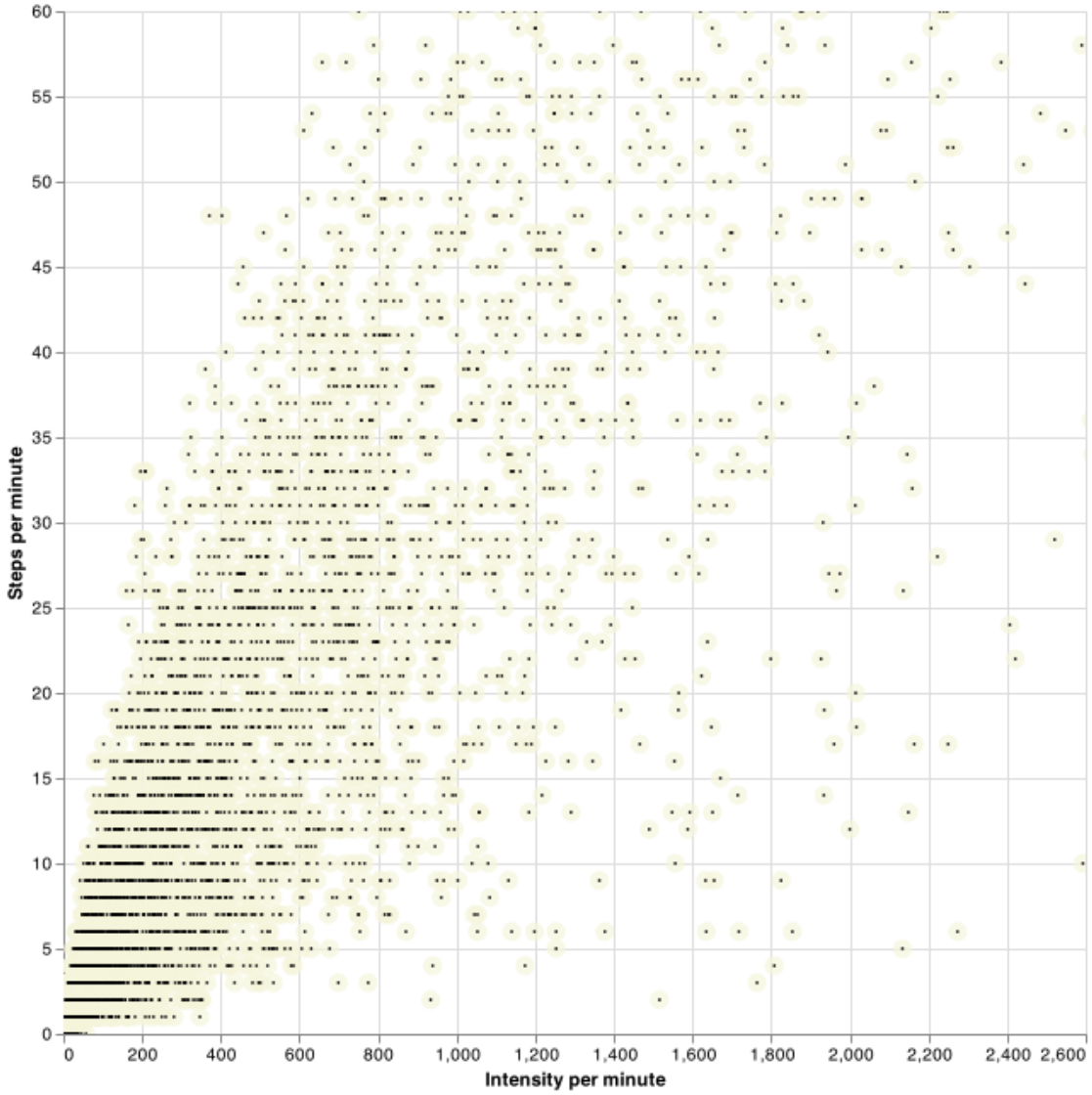


[165]:

```
(
    alt.Chart(paxraw.loc[paxraw.SEQN == by_day.SEQN.unique()[0], :])
    .mark_circle(clip=True, opacity=0.7, color="#F5F5DC", size=120)
    .encode(
        alt.X(
            "PAXINTEN:Q",
            title="Intensity per minute",
            scale=alt.Scale(domain=[0, 2500]),
        ),
```

```
        alt.Y("PAXSTEP:Q", title="Steps per minute", scale=alt.Scale(domain=[0,␣
↪60])),
    )
    + alt.Chart(paxraw.loc[paxraw.SEQN == by_day.SEQN.unique()[0], :])
    .mark_circle(clip=True, opacity=1, color="black", size=3)
    .encode(x="PAXINTEN", y="PAXSTEP")
).properties(width=600, height=600)
```

[165]:

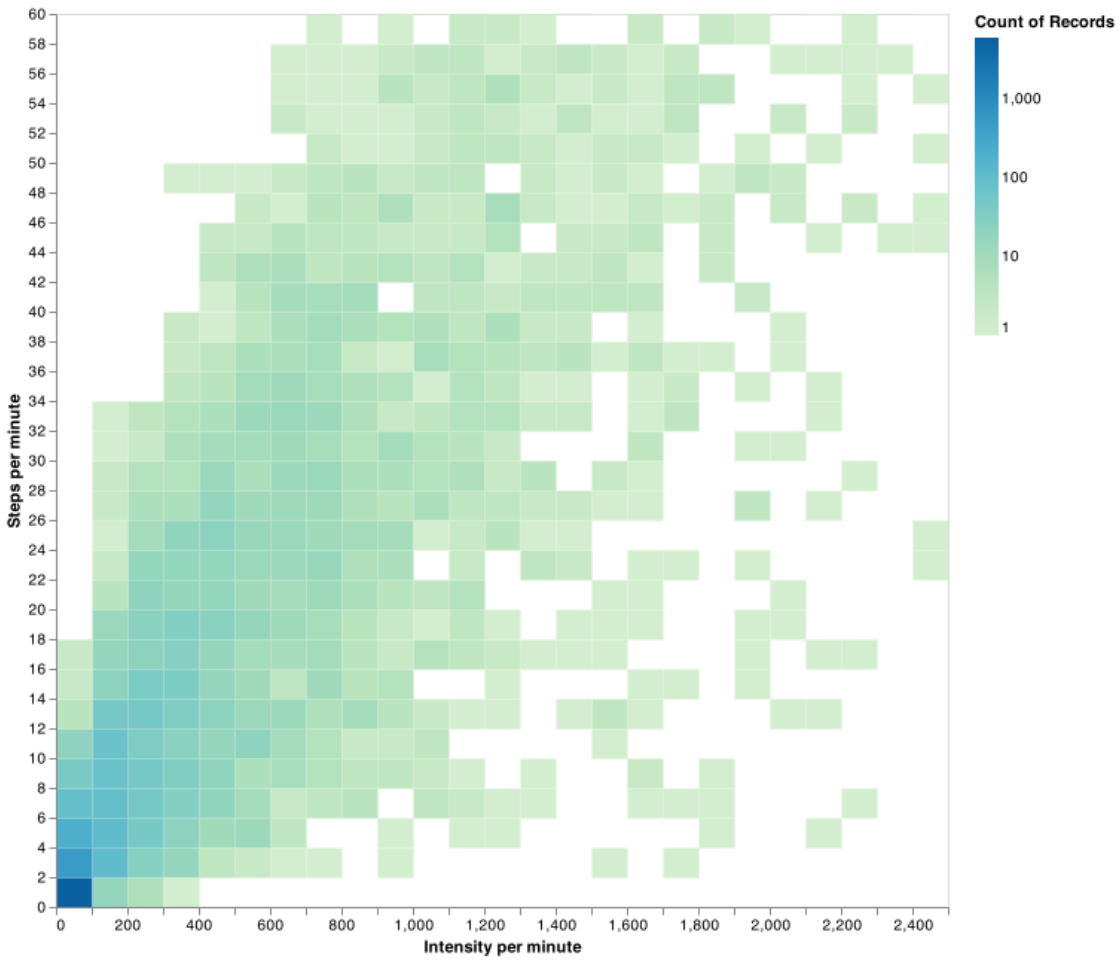


[166]:

```
alt.Chart(paxraw.loc[paxraw.SEQN == by_day.SEQN.unique()[0], :]).
↪mark_rect(clip=True).encode(
    alt.X(
        "PAXINTEN:Q",
```

```
        title="Intensity per minute",
        scale=alt.Scale(domain=[0, 2500]),
        bin=alt.Bin(maxbins=100),
    ),
    alt.Y(
        "PAXSTEP:Q",
        title="Steps per minute",
        scale=alt.Scale(domain=[0, 60]),
        bin=alt.Bin(maxbins=100),
    ),
    alt.Color("count():Q", scale=alt.Scale(scheme="greenblue", type="log")),
).properties(width=600, height=600)
```

[166]:

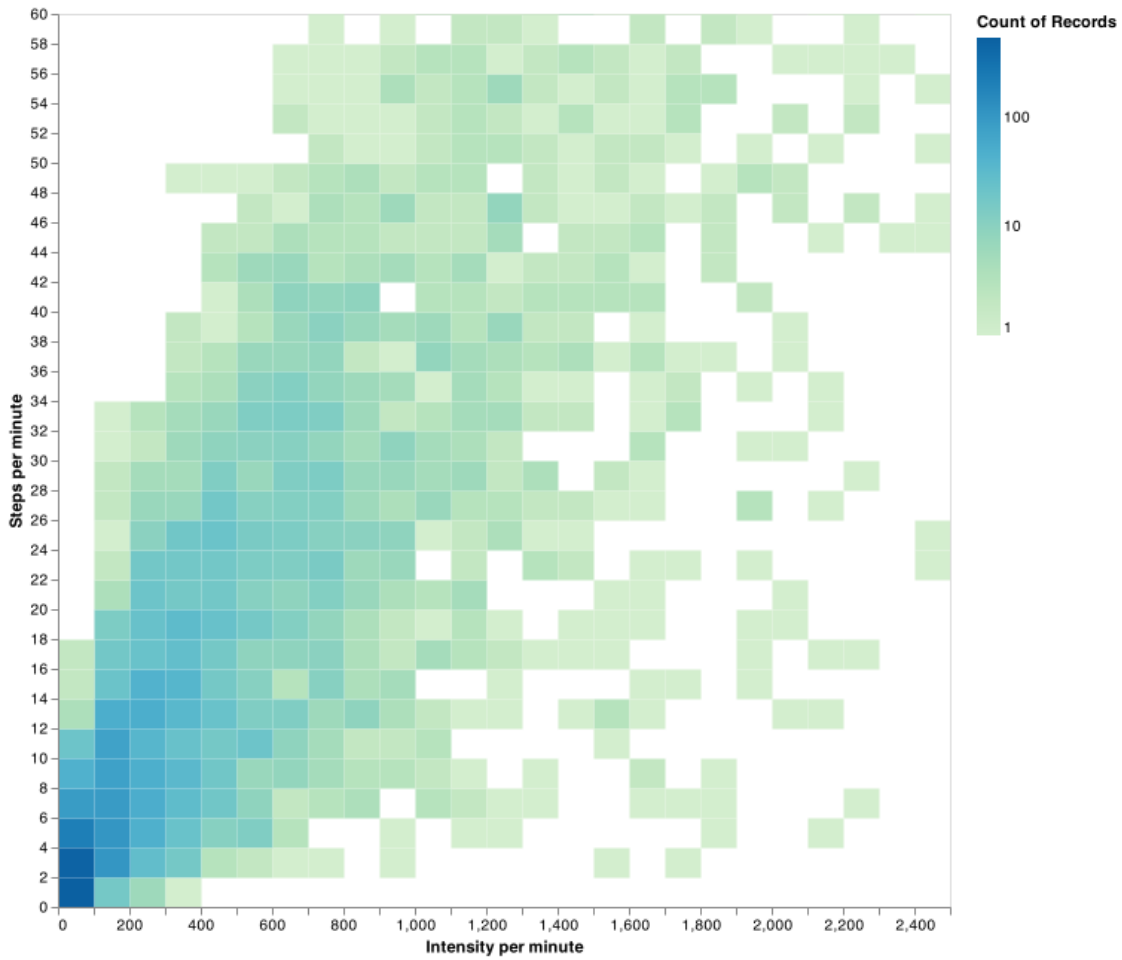


[167]:

```
alt.Chart(
    paxraw.loc[
        (paxraw.SEQN == by_day.SEQN.unique()[0]) & (paxraw.PAXINTEN > 0) &␣
↪(paxraw.PAXSTEP > 0),
```

```
        :,
    ]
).mark_rect(clip=True).encode(
    alt.X(
        "PAXINTEN:Q",
        title="Intensity per minute",
        scale=alt.Scale(domain=[0, 2500]),
        bin=alt.Bin(maxbins=100),
    ),
    alt.Y(
        "PAXSTEP:Q",
        title="Steps per minute",
        scale=alt.Scale(domain=[0, 60]),
        bin=alt.Bin(maxbins=100),
    ),
    alt.Color("count():Q", scale=alt.Scale(scheme="greenblue", type="log")),
).properties(
    width=600, height=600
)
```

[167]:



8.4 Activity plots for a single participant

[168]:
```
person_active_counts = d_people_days.loc[
    :, ["SEQN", "worn_sum", "PAXINTEN_sum", "valid_day"]
].copy()
```

[169]:
```
person_active_counts.columns = ["SEQN", "worn_minutes", "activity_counts",␣
 ↪"valid_day"]
```

[170]:
```
person_active_counts.head()
```

[170]:

	SEQN	worn_minutes	activity_counts	valid_day
0	31128	1276.0	377456	1
1	31128	1009.0	308309	1
2	31128	1273.0	324734	1
3	31128	599.0	229846	0
4	31128	911.0	304957	1

[171]:
```
person_active_counts_summary = (
    person_active_counts.loc[person_active_counts.valid_day == 1, :]
    .groupby(["SEQN"])
    .agg({"worn_minutes": np.mean, "activity_counts": np.mean, "valid_day":␣
 ↪"count"})
)
person_active_counts_summary.columns = [
    "daily_worn_minutes_mean",
    "daily_activity_count_sum_mean",
    "n_valid_days",
]
person_active_counts_summary.head()
```

[171]:

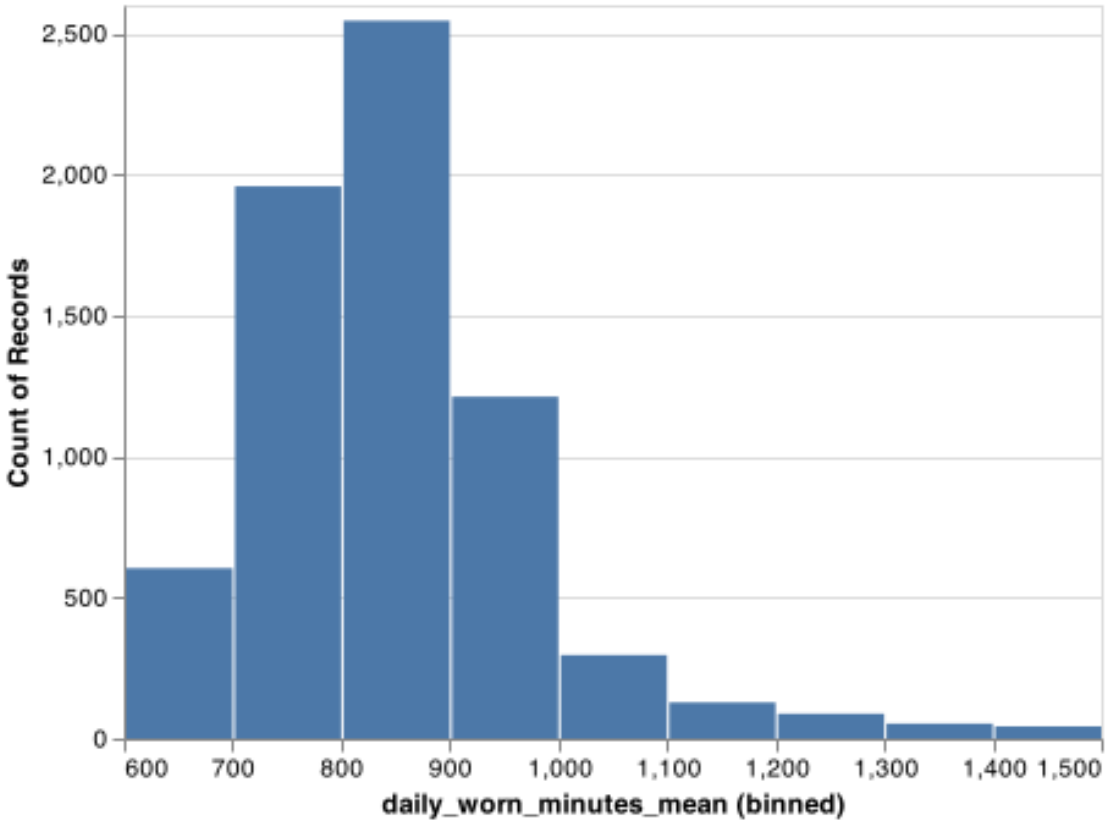
SEQN	daily_worn_minutes_mean	daily_activity_count_sum_mean	n_valid_days
31128	1140.166667	365314.000000	6
31129	923.500000	271898.250000	4
31131	897.500000	261294.333333	6
31132	898.200000	473383.400000	5
31133	881.200000	212055.200000	5

[172]:
```
person_active_counts_summary.shape
```

[172]: (6927, 3)

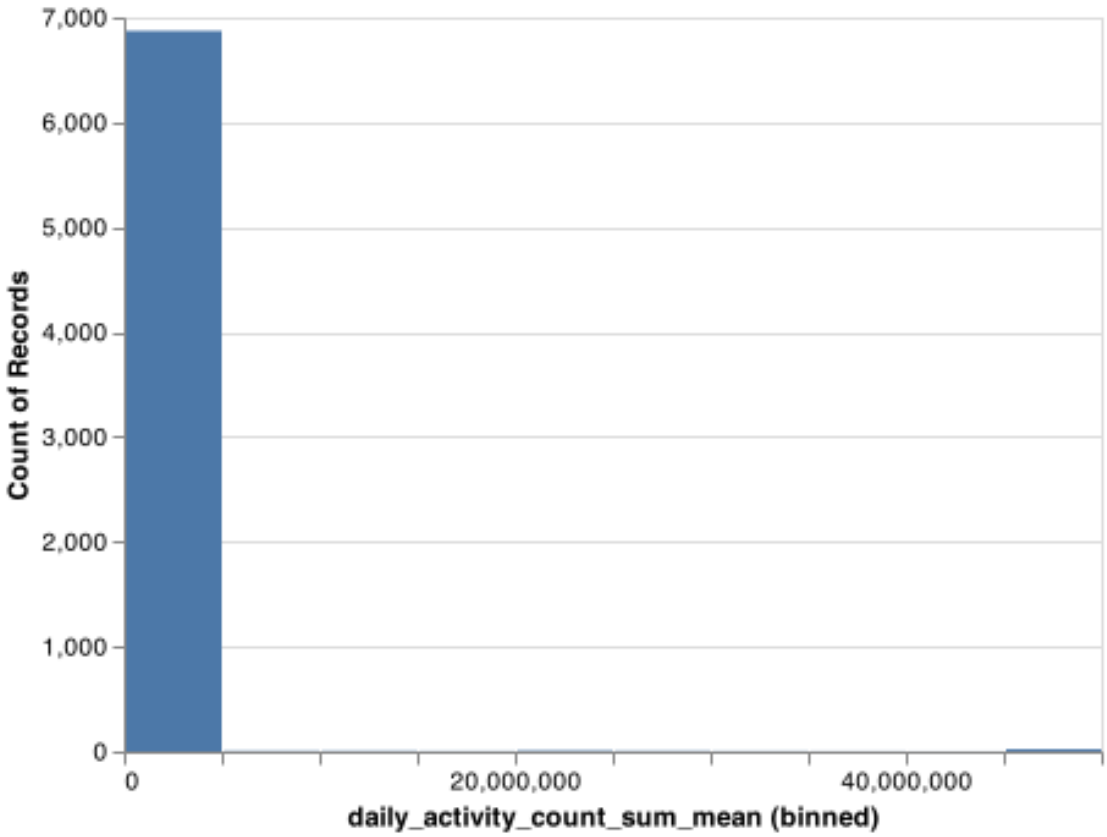
[173]:
```
alt.Chart(person_active_counts_summary).mark_bar().encode(
    alt.X("daily_worn_minutes_mean:Q", bin=True),
    y="count()",
)
```

[173]:



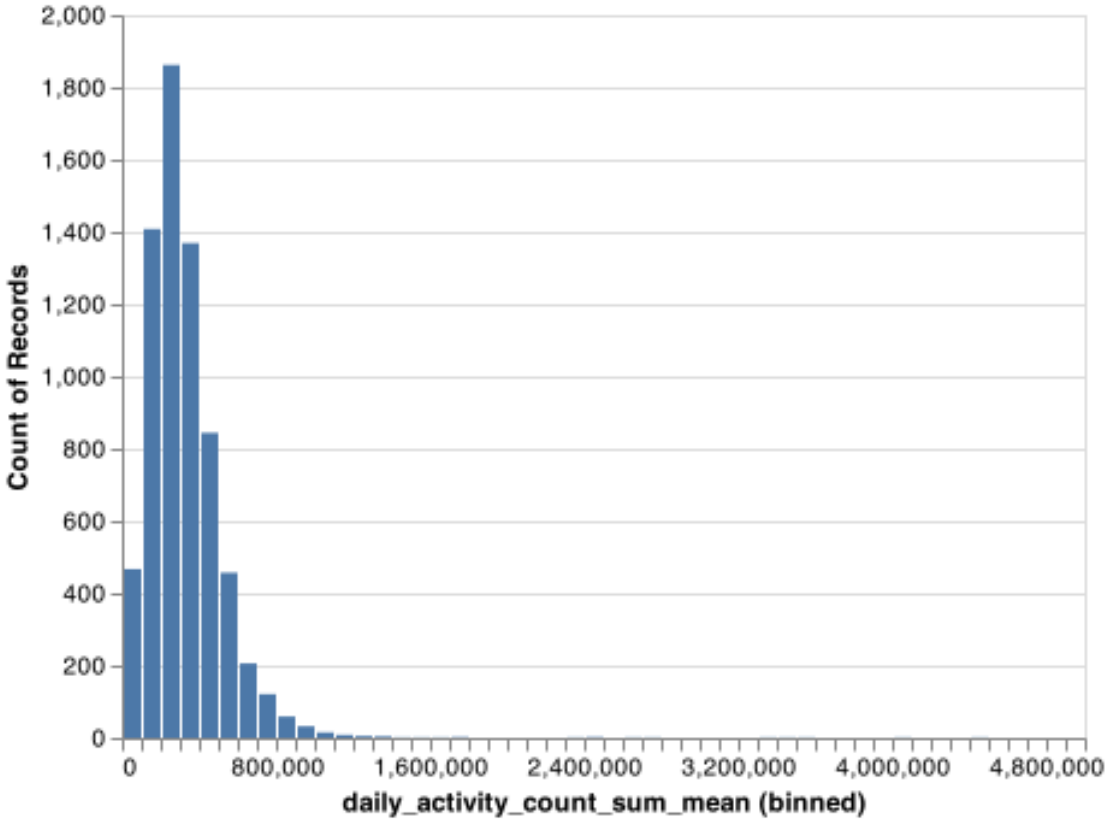
[174]:
```
alt.Chart(person_active_counts_summary).mark_bar().encode(
    alt.X("daily_activity_count_sum_mean:Q", bin=True),
    y="count()",
)
```

[174]:



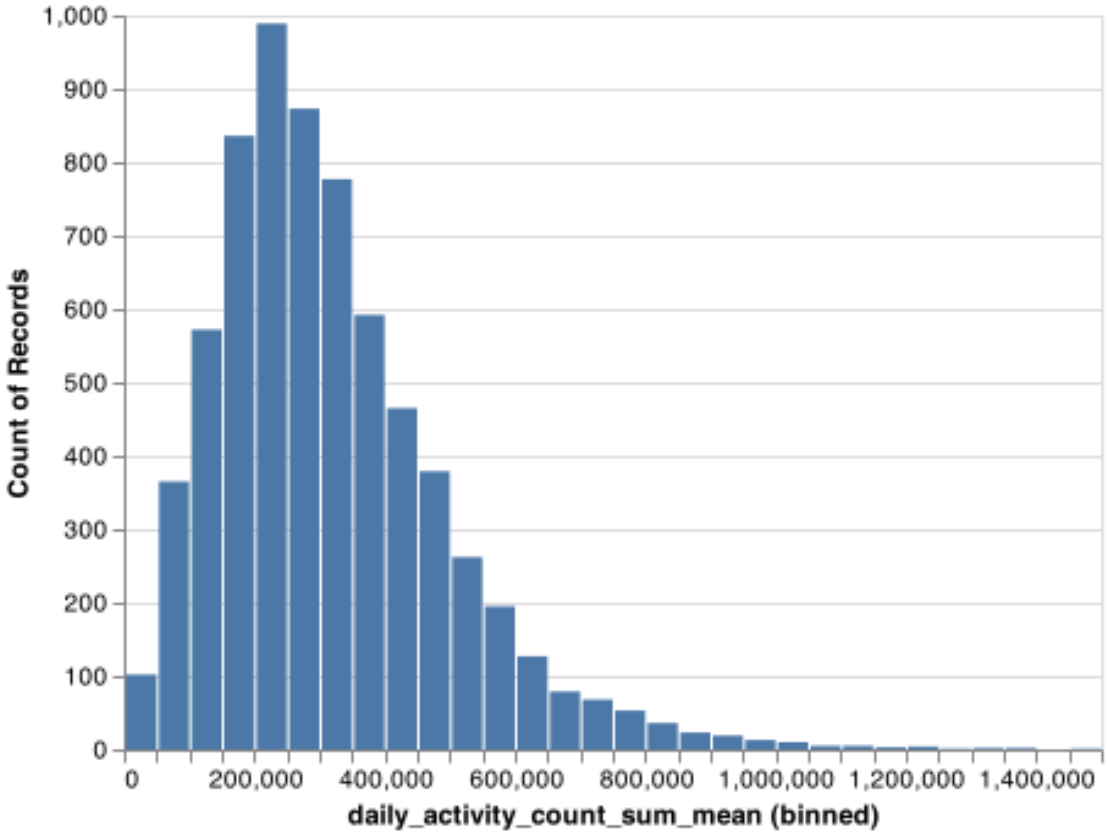
[175]:
```
alt.Chart(person_active_counts_summary).mark_bar(clip=True).encode(
    alt.X(
        "daily_activity_count_sum_mean:Q",
        scale=alt.Scale(domain=[0, 5000000]),
        bin=alt.Bin(maxbins=500),
    ),
    y="count()",
)
```

[175]:



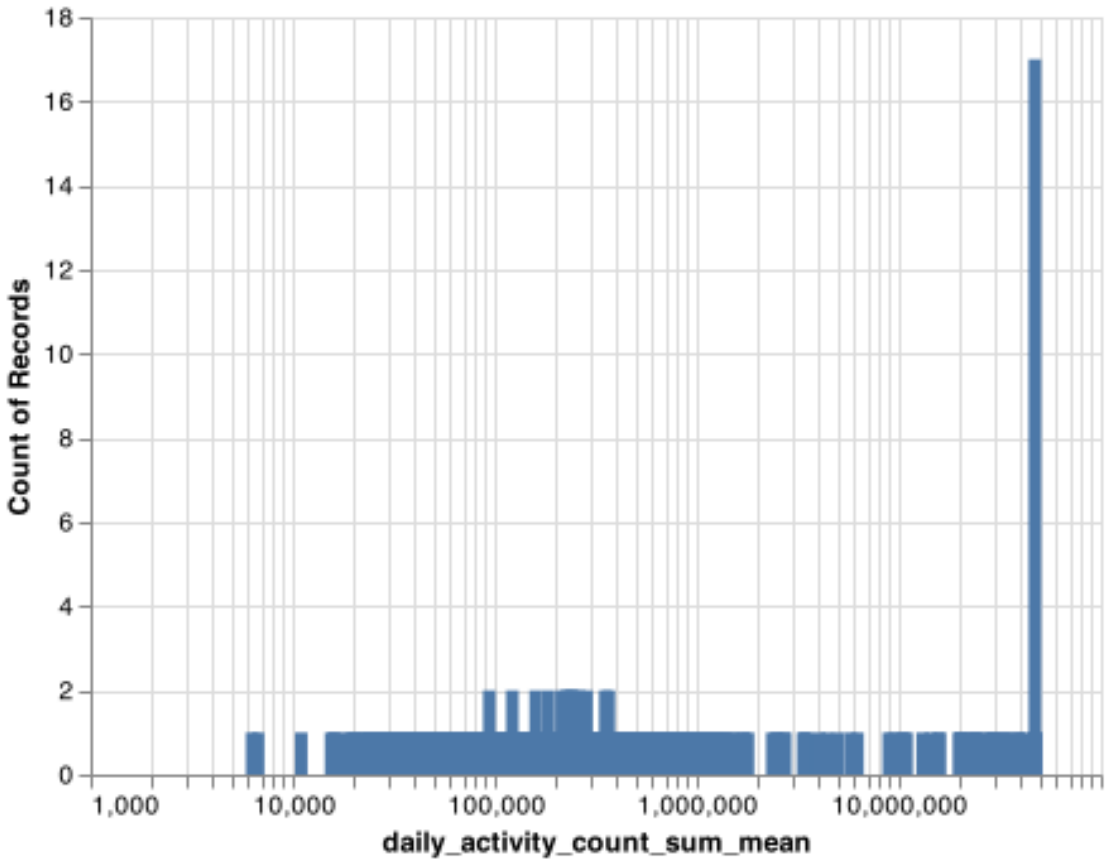
[176]:
```
alt.Chart(person_active_counts_summary).mark_bar(clip=True).encode(
    alt.X(
        "daily_activity_count_sum_mean:Q",
        scale=alt.Scale(domain=[0, 1500000]),
        bin=alt.Bin(maxbins=1000),
    ),
    y="count()",
)
```

[176]:



[177]:
```
alt.Chart(person_active_counts_summary).mark_bar().encode(
    alt.X("daily_activity_count_sum_mean:Q", scale=alt.Scale(type="log")),
    y="count()",
)
```

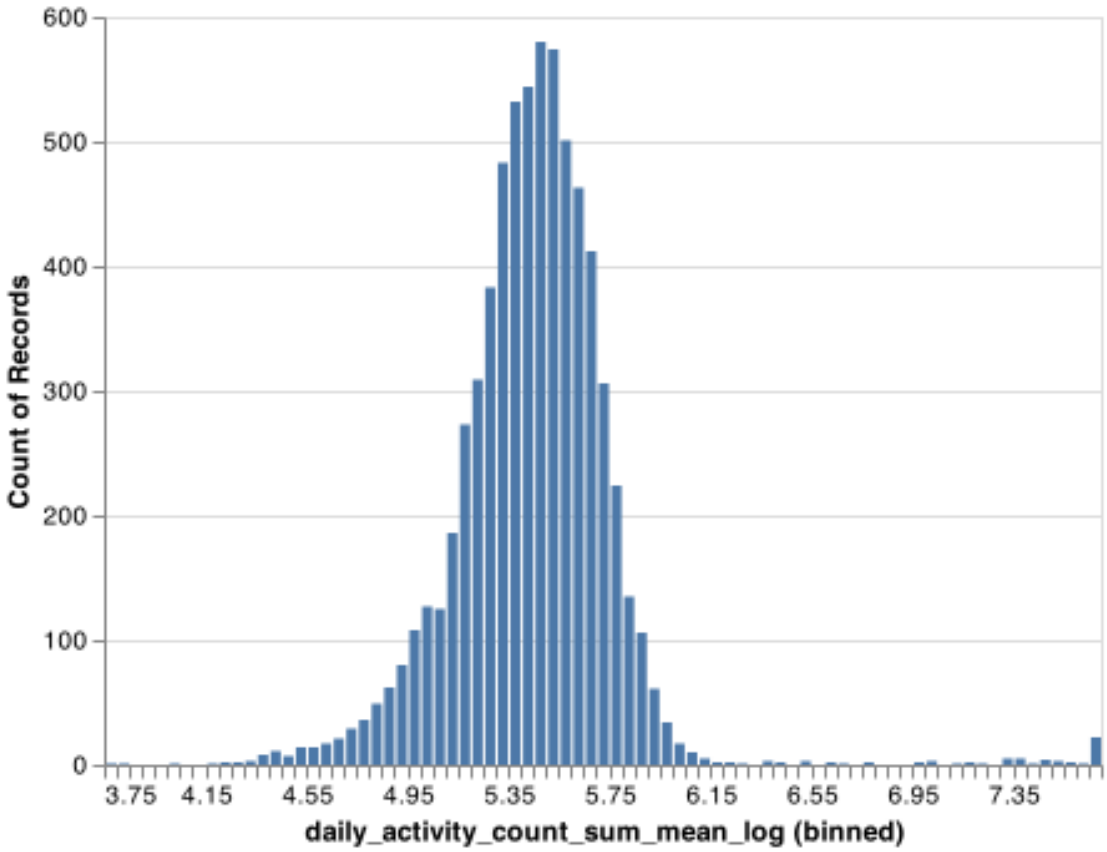
[177]:



[178]:
```
person_active_counts_summary["daily_activity_count_sum_mean_log"] = np.log10(
    person_active_counts_summary["daily_activity_count_sum_mean"]
)
```

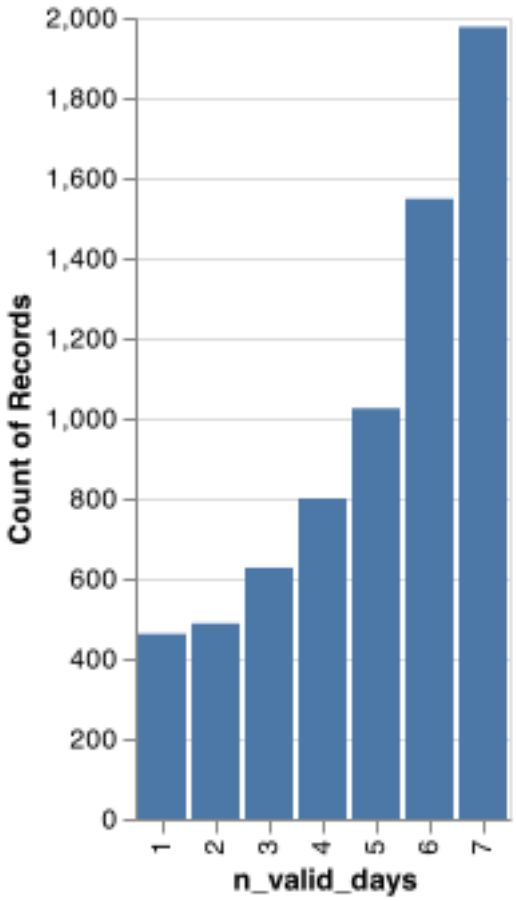
[179]:
```
alt.Chart(person_active_counts_summary).mark_bar(clip=True).encode(
    alt.X(
        "daily_activity_count_sum_mean_log:Q",
        scale=alt.Scale(),
        bin=alt.Bin(maxbins=100),
    ),
    y="count()",
)
```

[179]:



[180]:
```
alt.Chart(person_active_counts_summary).mark_bar().encode(
    alt.X("n_valid_days:O"),
    y="count()",
)
```

[180]:

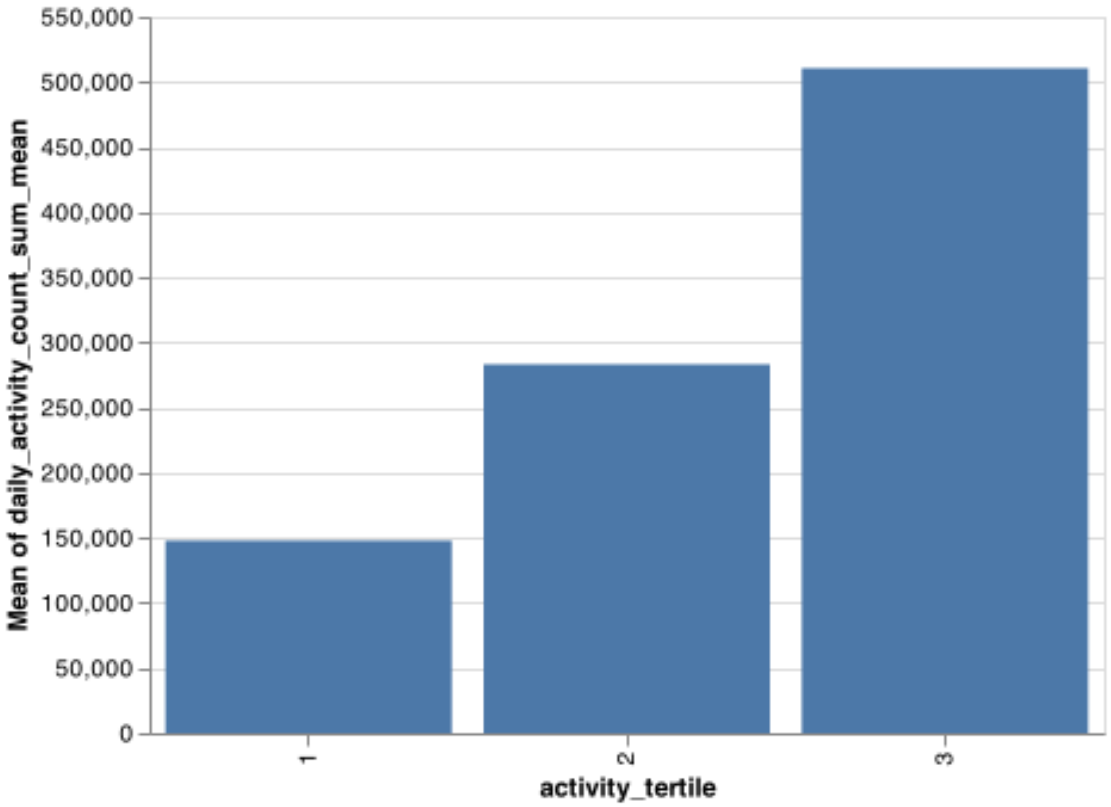


[181]:
```
# clear the crazy data
person_active_counts_summary = person_active_counts_summary.loc[
    person_active_counts_summary.daily_activity_count_sum_mean < 1500000, :
].copy()
```

[182]:
```
cuts = 3
person_active_counts_summary["activity_tertile"] = pd.qcut(
    person_active_counts_summary.daily_activity_count_sum_mean,
    q=cuts,
    labels=range(1, cuts + 1),
)
```

[183]:
```
alt.Chart(person_active_counts_summary).mark_bar(clip=True).encode(
    alt.X("activity_tertile:O", scale=alt.Scale(), bin=False),
    y="mean(daily_activity_count_sum_mean)",
).properties(width=400)
```

[183]:



8.5 Examine the filters for valid people

[184]:
```
d_people.loc[
    (d_people.zero_steps_with_intensity_sum > 10)
    | (d_people.too_many_steps_sum > 10)
    | (d_people.max_intensity_sum > 10),
    :,
].head(20)
```

[184]:

SEQN	zero_steps_with_intensity_sum	too_many_steps_sum	max_intensity_sum
31180	91	709	0
31192	30	345	0
31254	26	1129	0
31257	42	496	0
31263	10080	0	10080
31301	60	498	0
31307	135	0	0
31349	13	0	0
31415	42	279	0
31437	1009	0	0

31502	55	980	0
31550	19	0	0
31637	41	756	0
31662	57	993	0
31674	47	809	0
31698	17	0	0
31700	6797	0	126
31776	2536	0	0
31797	48	901	0
31840	59	521	0

SEQN	out_of_calibration_sum	out_of_calibration_last	unreliable_sum	\
31180	0	0	9092	
31192	0	0	10080	
31254	0	0	10080	
31257	0	0	6904	
31263	10080	1	10080	
31301	0	0	8950	
31307	10080	1	10080	
31349	0	0	10080	
31415	0	0	9114	
31437	0	0	10080	
31502	0	0	7602	
31550	0	0	10080	
31637	0	0	7645	
31662	0	0	8374	
31674	0	0	10080	
31698	0	0	10080	
31700	10080	1	10080	
31776	0	0	10080	
31797	0	0	10080	
31840	0	0	8383	

SEQN	unreliable_last	steps_filtered_500_sum	steps_filtered_300_sum	\
31180	1	37285.571429	51537.714286	
31192	1	13657.000000	18503.714286	
31254	1	72155.000000	91213.428571	
31257	1	14367.800000	23388.600000	
31263	1	0.000000	0.000000	
31301	1	28979.428571	34453.714286	
31307	1	2625.142857	3802.714286	
31349	1	7041.285714	7722.285714	
31415	1	23109.428571	25605.142857	
31437	1	1392.142857	2463.000000	
31502	1	120665.166667	138214.000000	

```
31550                1            1867.857143            2140.571429
31637                1          139785.333333          150074.500000
31662                1           70347.666667           92044.000000
31674                1           33606.285714           44264.857143
31698                1            7950.142857            8558.142857
31700                1            3715.714286            4024.857143
31776                1            9369.571429            9786.428571
31797                1          116551.000000          128244.714286
31840                1           21335.000000           31308.500000

       valid_day  PAXINTEN_sum
SEQN
31180          3  8.668543e+04
31192          1  3.835557e+04
31254          5  1.640576e+05
31257          5  6.801540e+04
31263          7  4.718448e+07
31301          2  6.382729e+04
31307          4  3.338937e+05
31349          6  2.540803e+05
31415          1  4.149614e+04
31437          7  7.873050e+05
31502          2  1.894628e+05
31550          1  7.734857e+04
31637          2  2.039432e+05
31662          4  1.386647e+05
31674          5  9.138557e+04
31698          5  4.072803e+05
31700          7  1.032637e+07
31776          5  1.385192e+06
31797          1  1.621687e+05
31840          3  6.734683e+04
```

[185]:
```
d_people.loc[
    (d_people.zero_steps_with_intensity_sum > 10)
    | (d_people.too_many_steps_sum > 10)
    | (d_people.max_intensity_sum > 10)
    | (d_people.out_of_calibration_last),
    :,
].head(20)
```

[185]:
```
       zero_steps_with_intensity_sum  too_many_steps_sum  max_intensity_sum  \
SEQN
31154                              0                   0                  0
31163                              3                   0                  0
31180                             91                 709                  0
31181                              3                   0                  0
```

31192	30	345	0
31193	0	0	0
31203	0	0	0
31211	0	0	0
31230	0	0	0
31240	0	0	0
31254	26	1129	0
31257	42	496	0
31263	10080	0	10080
31280	0	0	0
31301	60	498	0
31307	135	0	0
31308	0	0	0
31349	13	0	0
31410	0	0	0
31415	42	279	0

SEQN	out_of_calibration_sum	out_of_calibration_last	unreliable_sum	\
31154	10080	1	0	
31163	10080	1	0	
31180	0	0	9092	
31181	10080	1	0	
31192	0	0	10080	
31193	10080	1	0	
31203	10080	1	0	
31211	10080	1	0	
31230	10080	1	0	
31240	10080	1	0	
31254	0	0	10080	
31257	0	0	6904	
31263	10080	1	10080	
31280	10080	1	0	
31301	0	0	8950	
31307	10080	1	10080	
31308	10080	1	0	
31349	0	0	10080	
31410	10080	1	0	
31415	0	0	9114	

SEQN	unreliable_last	steps_filtered_500_sum	steps_filtered_300_sum	\
31154	0	3773.142857	6016.000000	
31163	0	4451.142857	5287.285714	
31180	1	37285.571429	51537.714286	
31181	0	2693.857143	4333.000000	
31192	1	13657.000000	18503.714286	

```
31193                0            9644.142857           10371.857143
31203                0            6299.428571            7980.142857
31211                0            5260.571429            6351.000000
31230                0            3711.857143            4409.428571
31240                0            4054.285714            4841.857143
31254                1           72155.000000           91213.428571
31257                1           14367.800000           23388.600000
31263                1               0.000000               0.000000
31280                0            8252.428571            9834.285714
31301                1           28979.428571           34453.714286
31307                1            2625.142857            3802.714286
31308                0            4658.428571            6436.285714
31349                1            7041.285714            7722.285714
31410                0            7836.142857            8801.142857
31415                1           23109.428571           25605.142857
```

```
       valid_day  PAXINTEN_sum
SEQN
31154          7  1.001300e+05
31163          6  1.680881e+05
31180          3  8.668543e+04
31181          7  1.222759e+05
31192          1  3.835557e+04
31193          2  3.998269e+05
31203          7  2.749851e+05
31211          6  2.377843e+05
31230          2  1.325093e+05
31240          4  1.724244e+05
31254          5  1.640576e+05
31257          5  6.801540e+04
31263          7  4.718448e+07
31280          7  2.942091e+05
31301          2  6.382729e+04
31307          4  3.338937e+05
31308          7  2.261061e+05
31349          6  2.540803e+05
31410          7  3.116296e+05
31415          1  4.149614e+04
```

[186]:
```
d_people.loc[
    (d_people.zero_steps_with_intensity_sum > 10)
    | (d_people.too_many_steps_sum > 10)
    | (d_people.max_intensity_sum > 10)
    | (d_people.out_of_calibration_last)
    | (d_people.unreliable_last),
    :,
].head(20)
```

[186]:

SEQN	zero_steps_with_intensity_sum	too_many_steps_sum	max_intensity_sum
31154	0	0	0
31163	3	0	0
31180	91	709	0
31181	3	0	0
31192	30	345	0
31193	0	0	0
31203	0	0	0
31211	0	0	0
31230	0	0	0
31240	0	0	0
31254	26	1129	0
31257	42	496	0
31263	10080	0	10080
31280	0	0	0
31301	60	498	0
31307	135	0	0
31308	0	0	0
31349	13	0	0
31410	0	0	0
31415	42	279	0

SEQN	out_of_calibration_sum	out_of_calibration_last	unreliable_sum
31154	10080	1	0
31163	10080	1	0
31180	0	0	9092
31181	10080	1	0
31192	0	0	10080
31193	10080	1	0
31203	10080	1	0
31211	10080	1	0
31230	10080	1	0
31240	10080	1	0
31254	0	0	10080
31257	0	0	6904
31263	10080	1	10080
31280	10080	1	0
31301	0	0	8950
31307	10080	1	10080
31308	10080	1	0
31349	0	0	10080
31410	10080	1	0
31415	0	0	9114

unreliable_last	steps_filtered_500_sum	steps_filtered_300_sum

SEQN			
31154	0	3773.142857	6016.000000
31163	0	4451.142857	5287.285714
31180	1	37285.571429	51537.714286
31181	0	2693.857143	4333.000000
31192	1	13657.000000	18503.714286
31193	0	9644.142857	10371.857143
31203	0	6299.428571	7980.142857
31211	0	5260.571429	6351.000000
31230	0	3711.857143	4409.428571
31240	0	4054.285714	4841.857143
31254	1	72155.000000	91213.428571
31257	1	14367.800000	23388.600000
31263	1	0.000000	0.000000
31280	0	8252.428571	9834.285714
31301	1	28979.428571	34453.714286
31307	1	2625.142857	3802.714286
31308	0	4658.428571	6436.285714
31349	1	7041.285714	7722.285714
31410	0	7836.142857	8801.142857
31415	1	23109.428571	25605.142857

SEQN	valid_day	PAXINTEN_sum
31154	7	1.001300e+05
31163	6	1.680881e+05
31180	3	8.668543e+04
31181	7	1.222759e+05
31192	1	3.835557e+04
31193	2	3.998269e+05
31203	7	2.749851e+05
31211	6	2.377843e+05
31230	2	1.325093e+05
31240	4	1.724244e+05
31254	5	1.640576e+05
31257	5	6.801540e+04
31263	7	4.718448e+07
31280	7	2.942091e+05
31301	2	6.382729e+04
31307	4	3.338937e+05
31308	7	2.261061e+05
31349	6	2.540803e+05
31410	7	3.116296e+05
31415	1	4.149614e+04

[187]:
```
d_unreliable = d_people.loc[
    (d_people.zero_steps_with_intensity_sum > 10)
```

```
    | (d_people.too_many_steps_sum > 10)
    | (d_people.max_intensity_sum > 10)
    | (d_people.out_of_calibration_last)
    | (d_people.unreliable_sum > 10)
    | (d_people.steps_filtered_500_sum > 200000),
    :,
]
d_unreliable.head(20)
```

[187]:

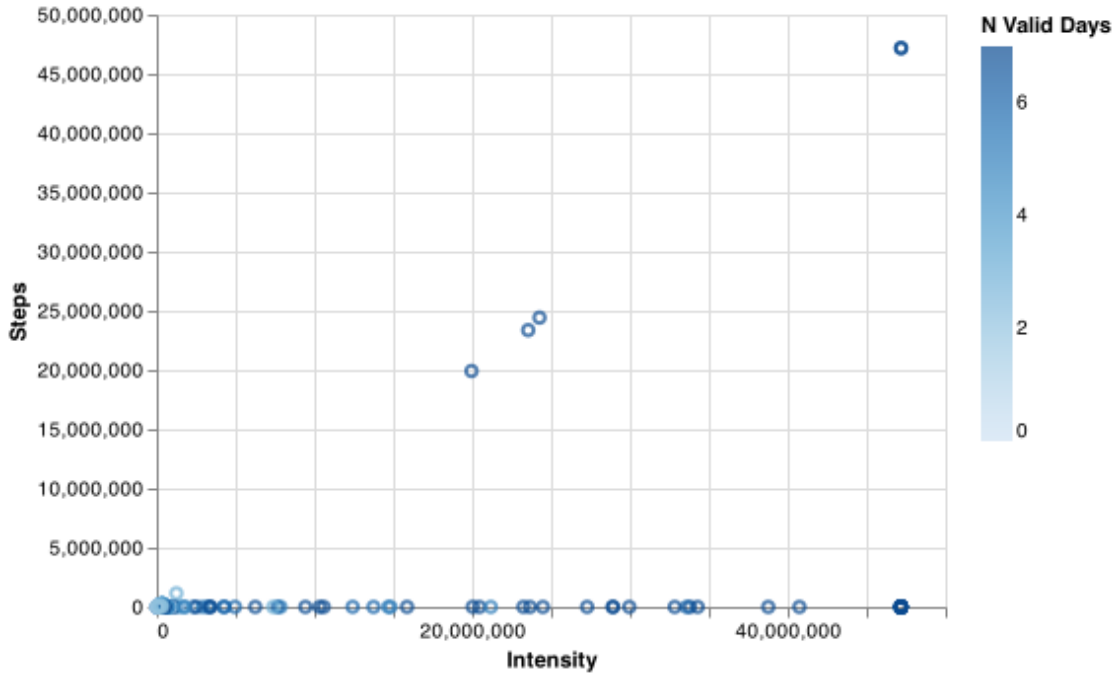
SEQN	zero_steps_with_intensity_sum	too_many_steps_sum	max_intensity_sum	out_of_calibration_sum	out_of_calibration_last	unreliable_sum
31154	0	0	0	10080	1	0
31163	3	0	0	10080	1	0
31180	91	709	0	0	0	9092
31181	3	0	0	10080	1	0
31192	30	345	0	0	0	10080
31193	0	0	0	10080	1	0
31203	0	0	0	10080	1	0
31211	0	0	0	10080	1	0
31230	0	0	0	10080	1	0
31240	0	0	0	10080	1	0
31254	26	1129	0	0	0	10080
31257	42	496	0	0	0	6904
31263	10080	0	10080	10080	1	10080
31280	0	0	0			
31301	60	498	0			
31307	135	0	0			
31308	0	0	0			
31349	13	0	0			
31410	0	0	0			
31415	42	279	0			

```
31280                  10080                        1              0
31301                      0                        0           8950
31307                  10080                        1          10080
31308                  10080                        1              0
31349                      0                        0          10080
31410                  10080                        1              0
31415                      0                        0           9114

       unreliable_last  steps_filtered_500_sum  steps_filtered_300_sum  \
SEQN
31154                0             3773.142857             6016.000000
31163                0             4451.142857             5287.285714
31180                1            37285.571429            51537.714286
31181                0             2693.857143             4333.000000
31192                1            13657.000000            18503.714286
31193                0             9644.142857            10371.857143
31203                0             6299.428571             7980.142857
31211                0             5260.571429             6351.000000
31230                0             3711.857143             4409.428571
31240                0             4054.285714             4841.857143
31254                1            72155.000000            91213.428571
31257                1            14367.800000            23388.600000
31263                1                0.000000                0.000000
31280                0             8252.428571             9834.285714
31301                1            28979.428571            34453.714286
31307                1             2625.142857             3802.714286
31308                0             4658.428571             6436.285714
31349                1             7041.285714             7722.285714
31410                0             7836.142857             8801.142857
31415                1            23109.428571            25605.142857

       valid_day  PAXINTEN_sum
SEQN
31154          7  1.001300e+05
31163          6  1.680881e+05
31180          3  8.668543e+04
31181          7  1.222759e+05
31192          1  3.835557e+04
31193          2  3.998269e+05
31203          7  2.749851e+05
31211          6  2.377843e+05
31230          2  1.325093e+05
31240          4  1.724244e+05
31254          5  1.640576e+05
31257          5  6.801540e+04
31263          7  4.718448e+07
31280          7  2.942091e+05
```

```
31301          2  6.382729e+04
31307          4  3.338937e+05
31308          7  2.261061e+05
31349          6  2.540803e+05
31410          7  3.116296e+05
31415          1  4.149614e+04
```

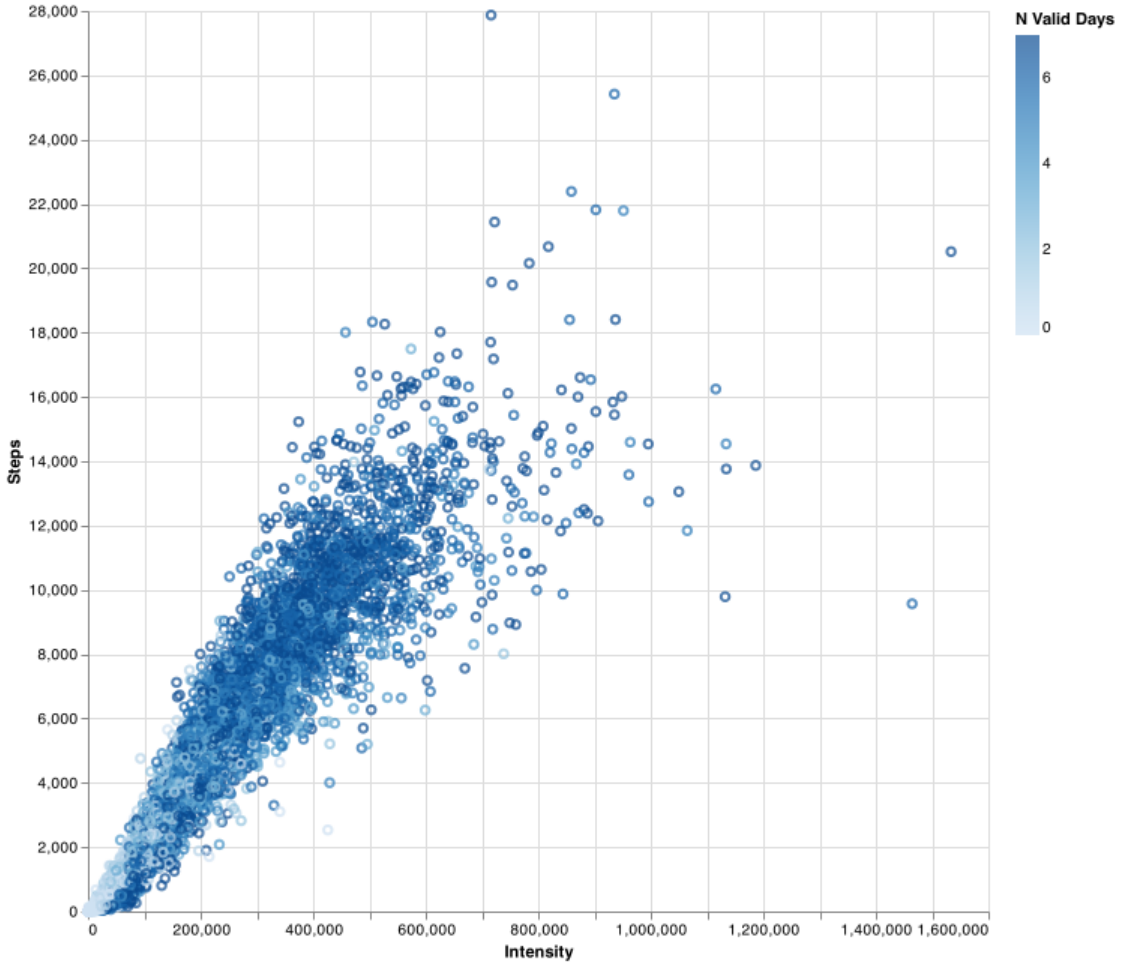
[188]:
```
alt.Chart(d_unreliable).mark_point().encode(
    alt.X("PAXINTEN_sum", title="Intensity"),
    alt.Y("steps_filtered_500_sum", title="Steps"),
    alt.Color("valid_day", title="N Valid Days"),
)
```

[188]:



[189]:
```
alt.Chart(d_reliable).mark_point().encode(
    alt.X("PAXINTEN_sum", title="Intensity"),
    alt.Y("steps_filtered_500_sum", title="Steps"),
    alt.Color("valid_day", title="N Valid Days"),
).properties(width=600, height=600)
```

[189]:



9 Correlation of steps and intensity

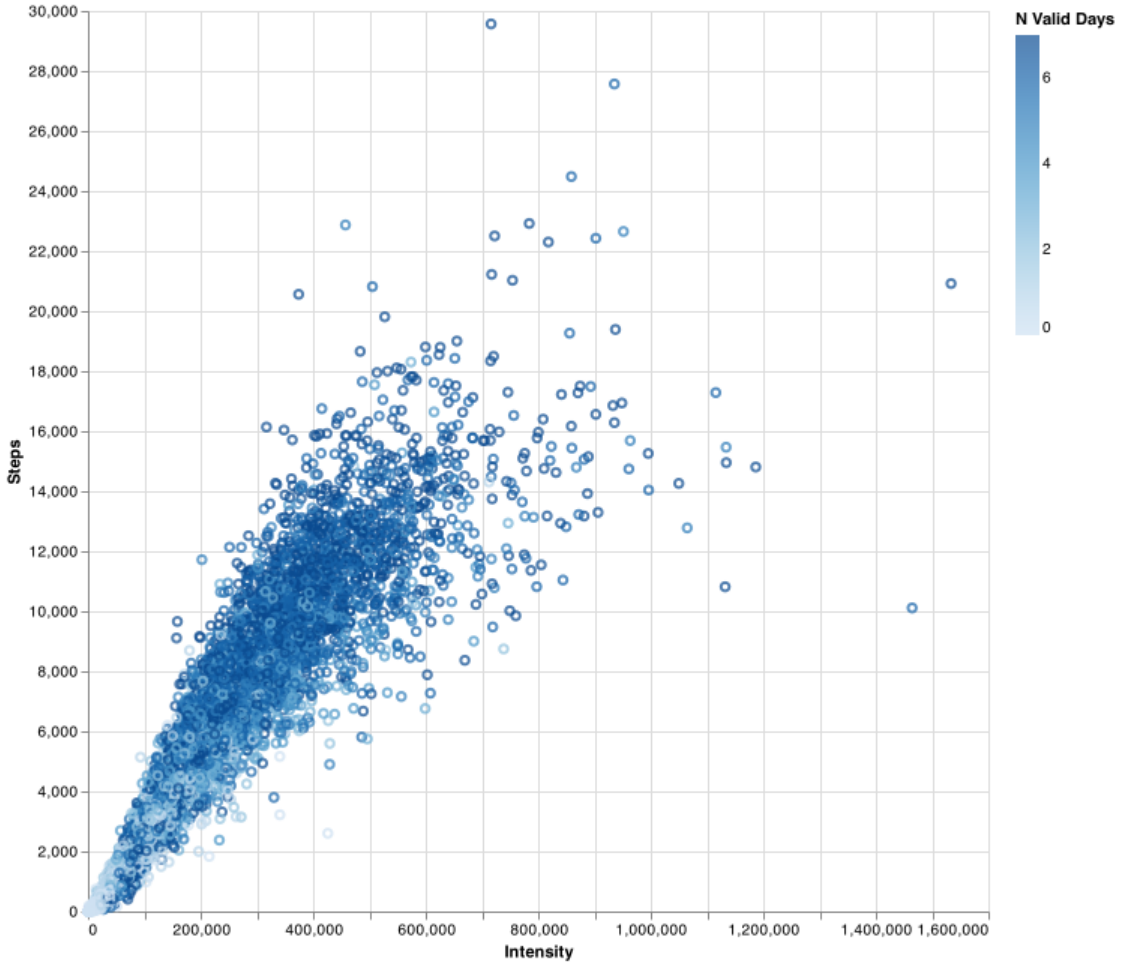
[190]:
```
results = smf.ols("steps_filtered_500_sum ~ PAXINTEN_sum + 0", data=d_reliable).
  ↪fit()
results.summary()
```

[190]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                                 OLS Regression Results
================================================================================
===========
Dep. Variable:     steps_filtered_500_sum   R-squared (uncentered):
0.952
Model:                                OLS   Adj. R-squared (uncentered):
0.952
Method:                     Least Squares   F-statistic:
```

```
1.361e+05
Date:                       Thu, 23 Feb 2023   Prob (F-statistic):
0.00
Time:                              10:31:57   Log-Likelihood:
-59566.
No. Observations:                     6863   AIC:
1.191e+05
Df Residuals:                         6862   BIC:
1.191e+05
Df Model:                                1
Covariance Type:                 nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
PAXINTEN_sum     0.0219   5.95e-05    368.890      0.000       0.022       0.022
==============================================================================
Omnibus:                     3292.380   Durbin-Watson:                   2.004
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           123330.142
Skew:                          -1.631   Prob(JB):                         0.00
Kurtosis:                      23.510   Cond. No.                         1.00
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not
contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

[191]:
```
alt.Chart(d_reliable).mark_point().encode(
    alt.X("PAXINTEN_sum", title="Intensity"),
    alt.Y("steps_filtered_300_sum", title="Steps"),
    alt.Color("valid_day", title="N Valid Days"),
).properties(width=600, height=600)
```

[191]:



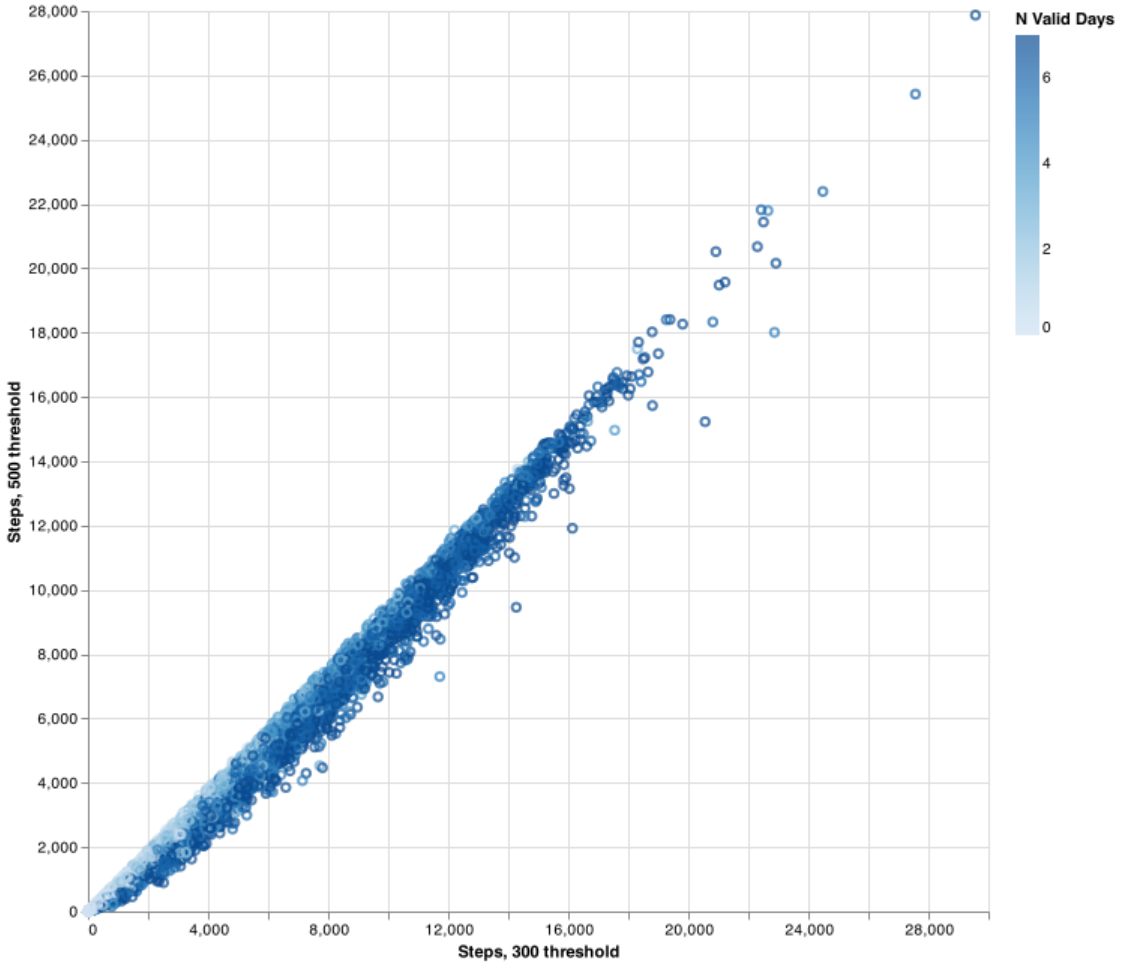
[192]:
```
results = smf.ols("steps_filtered_300_sum ~ PAXINTEN_sum + 0", data=d_reliable).
  ↪fit()
results.summary()
```

[192]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                                 OLS Regression Results
================================================================================
===========
Dep. Variable:     steps_filtered_300_sum   R-squared (uncentered):
0.945
Model:                                OLS   Adj. R-squared (uncentered):
0.945
Method:                     Least Squares   F-statistic:
1.178e+05
Date:                    Thu, 23 Feb 2023   Prob (F-statistic):
0.00
```

```
Time:                            10:31:59   Log-Likelihood:
-60936.
No. Observations:                    6863   AIC:
1.219e+05
Df Residuals:                        6862   BIC:
1.219e+05
Df Model:                               1
Covariance Type:                nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
PAXINTEN_sum     0.0249   7.26e-05    343.254      0.000       0.025       0.025
==============================================================================
Omnibus:                     3685.092   Durbin-Watson:                   1.960
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           120091.436
Skew:                          -1.974   Prob(JB):                         0.00
Kurtosis:                      23.109   Cond. No.                         1.00
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not
contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

[193]:
```
alt.Chart(d_reliable).mark_point().encode(
    alt.X("steps_filtered_300_sum", title="Steps, 300 threshold"),
    alt.Y("steps_filtered_500_sum", title="Steps, 500 threshold"),
    alt.Color("valid_day", title="N Valid Days"),
).properties(width=600, height=600)
```

[193]:


N Valid Days
6
4
2
0
Steps, 500 threshold
28,000
26,000
24,000
22,000
20,000
18,000
16,000
14,000
12,000
10,000
8,000
6,000
4,000
2,000
0
0
4,000
8,000
12,000
16,000
20,000
24,000
28,000
Steps, 300 threshold