2003-2004-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]:
```
# Parameters
year = "2003-2004"
```

[9]:
```
datadir = get_datadir(year)
```

[10]:
```
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])
```

[11]:
```
paxraw = pd.read_sas(xptfile)
```

[12]:
```
paxraw.shape
```

[12]:
```
(72250027, 8)
```

[13]:
```
paxraw.head()
```

[13]:

	SEQN	PAXSTAT	PAXCAL	PAXDAY	PAXN	PAXHOUR	PAXMINUT	PAXINTEN
0	21005.0	1.0	1.0	1.0	1.0	5.397605e-79	5.397605e-79	5.397605e-79
1	21005.0	1.0	1.0	1.0	2.0	5.397605e-79	1.000000e+00	5.397605e-79
2	21005.0	1.0	1.0	1.0	3.0	5.397605e-79	2.000000e+00	5.397605e-79
3	21005.0	1.0	1.0	1.0	4.0	5.397605e-79	3.000000e+00	5.397605e-79
4	21005.0	1.0	1.0	1.0	5.0	5.397605e-79	4.000000e+00	5.397605e-79

2.1 Fix datatypes

[14]:
```
paxraw.dtypes
```

[14]:
```
SEQN        float64
PAXSTAT     float64
PAXCAL      float64
PAXDAY      float64
PAXN        float64
PAXHOUR     float64
PAXMINUT    float64
PAXINTEN    float64
dtype: object
```

[15]:
```
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_77784/2208879285.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
```

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

```
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,
)
```

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

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

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

[17]:

	SEQN	PAXSTAT	PAXCAL	PAXDAY	PAXN	PAXHOUR	PAXMINUT	PAXINTEN
0	21005	1	1	1	1	0	0	0
1	21005	1	1	1	2	0	1	0
2	21005	1	1	1	3	0	2	0
3	21005	1	1	1	4	0	3	0
4	21005	1	1	1	5	0	4	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/2003-2004/paxraw_c.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/2003-2004/paxraw_c_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, 9)

[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', '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)
```

```
67.9 ms ± 3.2 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., …, 0., 0., 0.])
```

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

```
2.31 ms ± 46.4 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	MET
0	21005	1	1	1	1	0	0	0	1.0
1	21005	1	1	1	2	0	1	0	1.0
2	21005	1	1	1	3	0	2	0	1.0
3	21005	1	1	1	4	0	3	0	1.0
4	21005	1	1	1	5	0	4	0	1.0

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

[42]:

	SEQN	PAXSTAT	PAXCAL	PAXDAY	PAXN	PAXHOUR	PAXMINUT	PAXINTEN	MET
0	21005	1	1	1	1	0	0	0	1.0
1	21005	1	1	1	2	0	1	0	1.0
2	21005	1	1	1	3	0	2	0	1.0
3	21005	1	1	1	4	0	3	0	1.0
4	21005	1	1	1	5	0	4	0	1.0

[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', '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
```

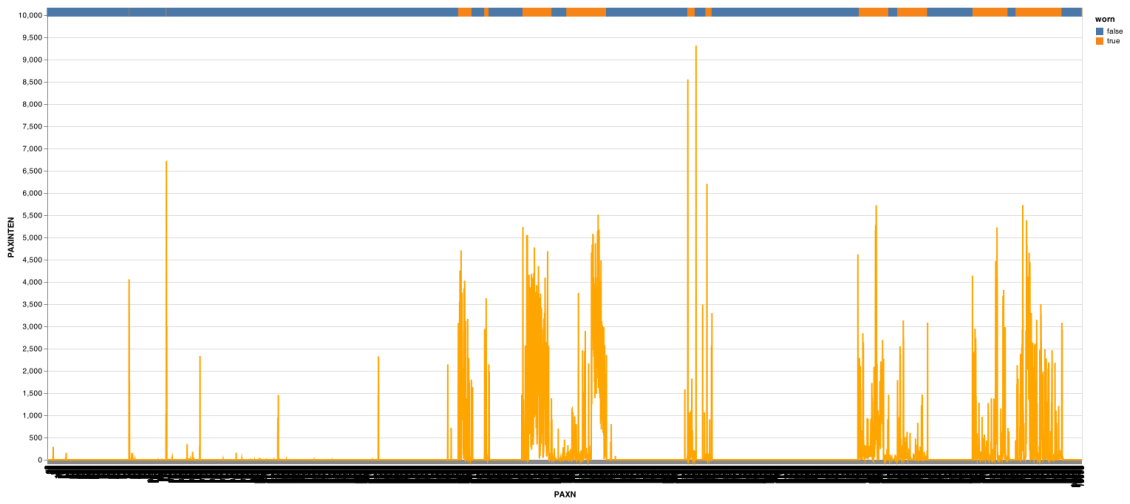
[48]:
```
paxraw_sample.iloc[paxraw_sample.shape[0] - 60 : paxraw_sample.shape[0], :].
 ↪shape
```

[48]:
```
(60, 10)
```

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

[50]:

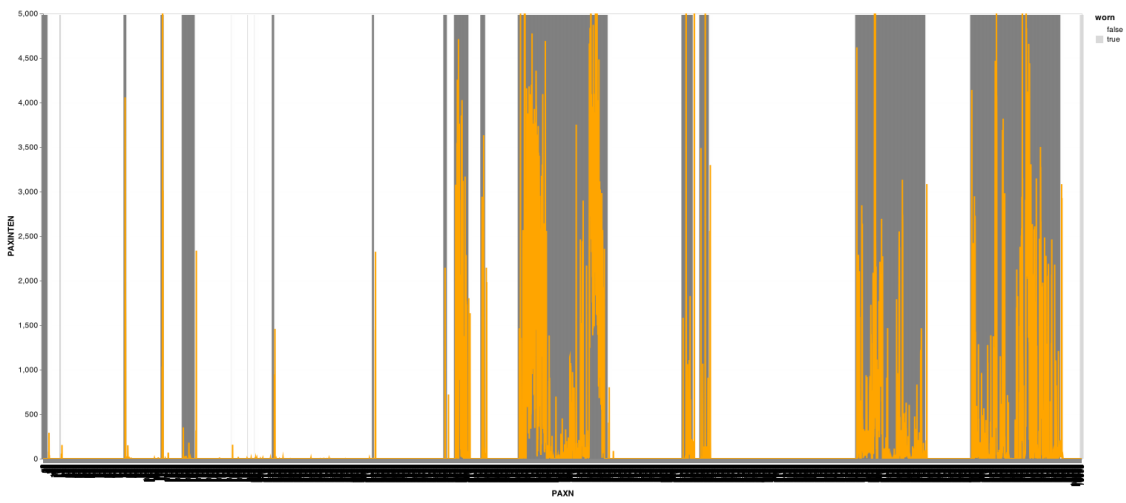


[51]:
```python
# set the indicator to True to start
paxraw_sample.loc[:, "worn"] = True
# take the first 60 minutes
assert paxraw_sample.iloc[:60, :].shape[0] == 60
# only 2 allowed > 0, and all 60 are less than 100
if ((paxraw_sample.iloc[:60, PAXINTEN_col] > 0).sum() <= 2) and (
    (paxraw_sample.iloc[:60,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

[52]:
```python
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]:
```python
(
    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]:

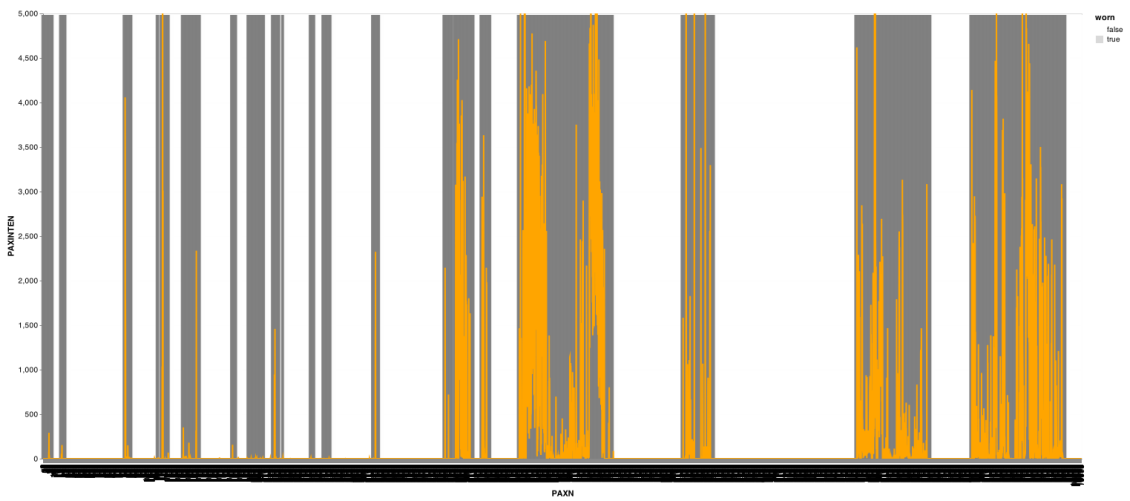


[54]:
```
# set the indicator to False to start
paxraw_sample.loc[:, "worn"] = False
# take the first 60 minutes
assert paxraw_sample.iloc[:60, :].shape[0] == 60
# only 2 allowed > 0, and all 60 are less than 100
if ((paxraw_sample.iloc[:60, PAXINTEN_col] > 0).sum() > 2) or (
    (paxraw_sample.iloc[:60, 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]:
```
393801.25
```

[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	202	32695	False
2	76	5380	False
3	243	143783	False
4	873	851618	True
5	203	74453	False
6	681	249123	True
7	876	469084	True

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

```
4.64 s ± 252 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

Hours:

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

[61]:
```
6.239133333333333
```

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 [00:40<00:00,  4.05s/it]
```

[63]:
```
person_active_counts
```

[63]:

		worn	PAXINTEN	valid_day
		sum	sum	
SEQN	PAXDAY			
21005	1	202	32695	False
	2	76	5380	False
	3	243	143783	False
	4	873	851618	True
	5	203	74453	False
	6	681	249123	True
	7	876	469084	True
21006	1	960	217932	True
	2	712	69560	True
	3	597	206583	False
	4	448	99358	False
	5	581	185269	False
	6	615	150717	True
	7	651	30908	True
21007	1	737	438907	True
	2	910	400517	True
	3	869	316556	True
	4	825	386072	True
	5	1098	293143	True
	6	1284	437497	True
	7	658	255734	True
21008	1	583	605287	False
	2	77	8517	False
	3	420	31406	False
	4	802	7618	True

	5	784	444873	True
	6	536	371741	False
	7	780	192324	True
21009	1	953	366414	True
	2	949	507292	True
	3	981	322404	True
	4	949	578837	True
	5	767	388850	True
	6	645	568553	True
	7	1059	133171	True
21010	1	783	409417	True
	2	824	351251	True
	3	774	224345	True
	4	803	197665	True
	5	617	140445	True
	6	875	283398	True
	7	725	398339	True
21012	1	1367	188752	True
	2	735	77106	True
	3	899	132312	True
	4	1440	118490	True
	5	958	178392	True
	6	781	133291	True
	7	995	87491	True
21013	1	640	133624	True
	2	947	347498	True
	3	2	1801	False
	4	847	146767	True
	5	827	150839	True
	6	969	288338	True
	7	533	144262	False
21015	1	611	84317	True
	2	684	103342	True
	3	632	60211	True
	4	882	84964	True
	5	1356	151834	True
	6	639	71376	True
	7	766	161950	True
21016	1	614	222672	True
	2	907	487651	True
	3	837	280816	True
	4	1026	398366	True
	5	889	222057	True
	6	3	3639	False
	7	510	25167	False

[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			
21005	1	202	32695	False
	2	76	5380	False
	3	243	143783	False
	4	873	851618	True
	5	203	74453	False

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_fast  #␣
 ↪noqa: E402
```

[66]:
```
# this should be fast
paxraw["worn"] = worn_indicator_SEQN_long_fast(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/2003-2004/paxraw_c_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=(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 Save parquet

[75]:
```
parquetfile = datadir / f"paxraw_{CHAR_LOOKUP[year].lower()}_met_worn_bouts.
 ↪parquet"
parquetfile
```

[75]:
```
PosixPath('/Users/mm51929/projects/2022/07-nhanes-
analysis/data/raw/2003-2004/paxraw_c_met_worn_bouts.parquet')
```

[76]:
```
paxraw.to_parquet(parquetfile)
# paxraw = pd.read_parquet(parquetfile)
```

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

[77]:
```
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
```

[78]:
```
paxraw.head()
```

[78]:

	SEQN	PAXSTAT	PAXCAL	PAXDAY	PAXN	PAXHOUR	PAXMINUT	PAXINTEN	MET	\
0	21005	1	1	1	1	0	0	0	1.0	
1	21005	1	1	1	2	0	1	0	1.0	
2	21005	1	1	1	3	0	2	0	1.0	
3	21005	1	1	1	4	0	3	0	1.0	
4	21005	1	1	1	5	0	4	0	1.0	

	worn	vigorous_bout	moderate_bout	light_bout	low_bout	sed_bout	\
0	1.0	0.0	0.0	0.0	0.0	0.0	
1	1.0	0.0	0.0	0.0	0.0	0.0	

```
2   1.0            0.0            0.0         0.0       0.0       0.0
3   1.0            0.0            0.0         0.0       0.0       0.0
4   1.0            0.0            0.0         0.0       0.0       0.0

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

[79]:
```
paxraw["METh"] = paxraw.MET / 60
paxraw["activeMETh"] = (paxraw.MET - 1) / 60
```

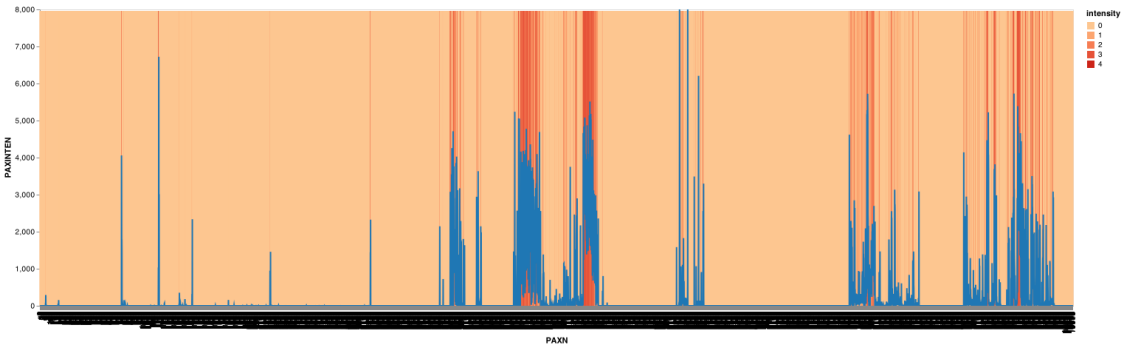
[80]:
```
paxraw_sample = paxraw.loc[paxraw.SEQN == paxraw.SEQN.values[0], :].copy()
paxraw_sample.head()
```

[80]:
```
    SEQN  PAXSTAT  PAXCAL  PAXDAY  PAXN  PAXHOUR  PAXMINUT  PAXINTEN  MET  \
0  21005        1       1       1     1        0         0         0  1.0
1  21005        1       1       1     2        0         1         0  1.0
2  21005        1       1       1     3        0         2         0  1.0
3  21005        1       1       1     4        0         3         0  1.0
4  21005        1       1       1     5        0         4         0  1.0

   worn  vigorous_bout  moderate_bout  light_bout  low_bout  sed_bout  \
0   1.0            0.0            0.0         0.0       0.0       0.0
1   1.0            0.0            0.0         0.0       0.0       0.0
2   1.0            0.0            0.0         0.0       0.0       0.0
3   1.0            0.0            0.0         0.0       0.0       0.0
4   1.0            0.0            0.0         0.0       0.0       0.0

   no_bout intensity      METh  activeMETh
0        1         0  0.016667         0.0
1        1         0  0.016667         0.0
2        1         0  0.016667         0.0
3        1         0  0.016667         0.0
4        1         0  0.016667         0.0
```

[81]:
```
(
    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)
```

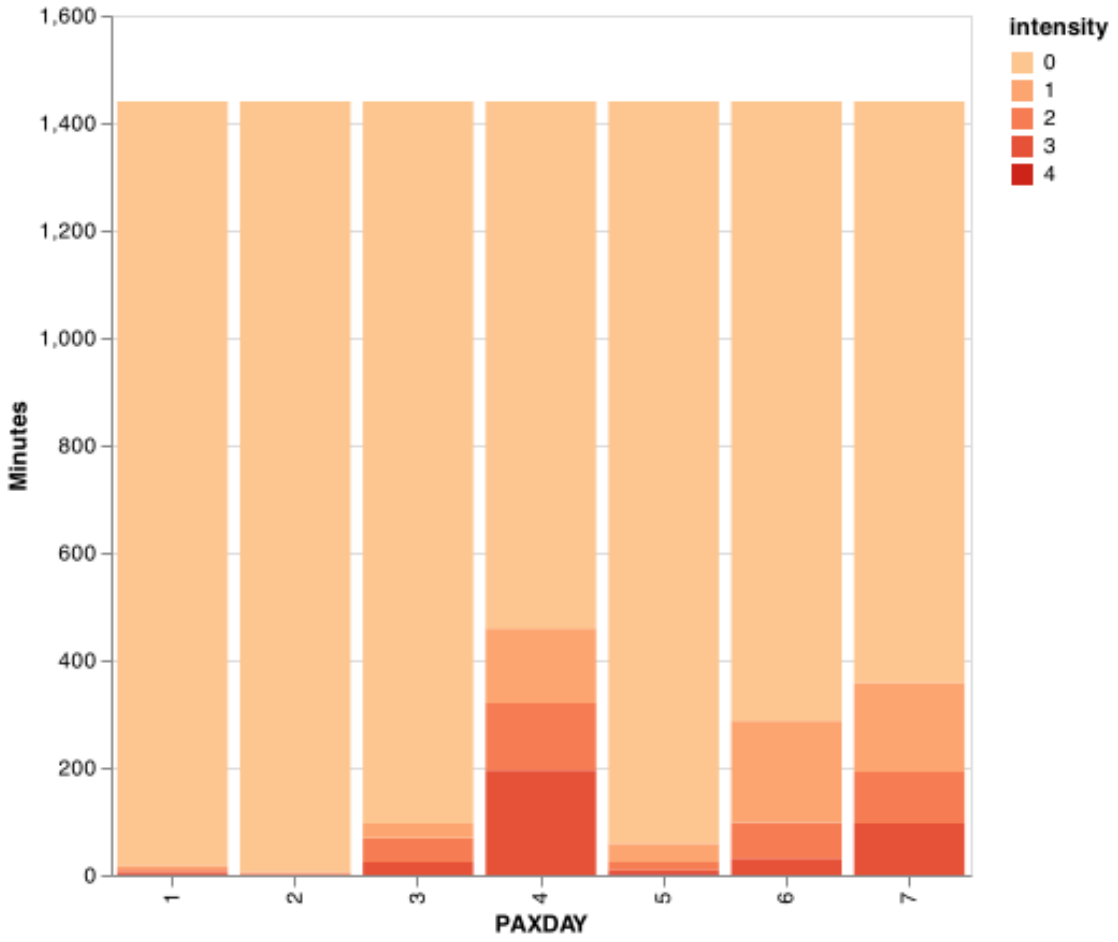
[81]:



[82]:

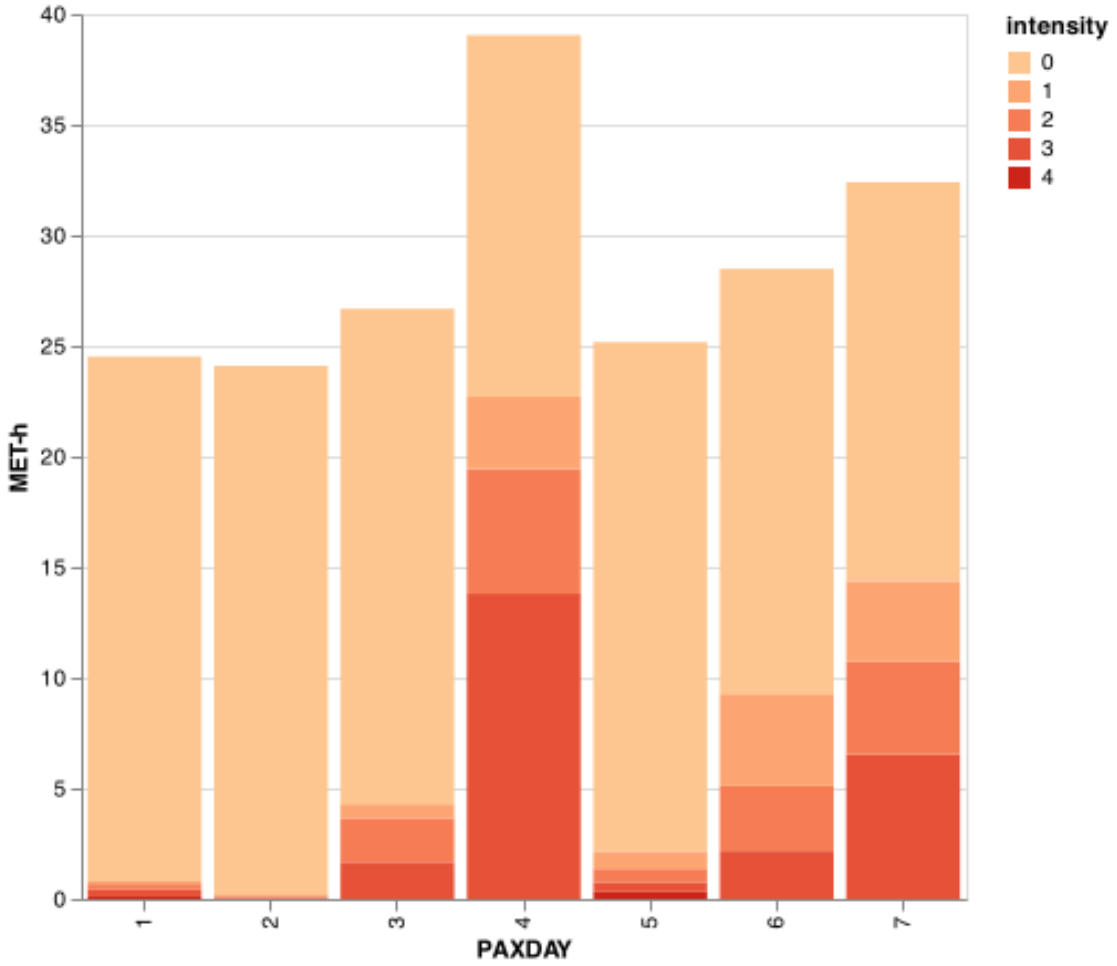
```
(
    alt.Chart(paxraw_sample)
    .mark_bar()
    .encode(
        x="PAXDAY:O",
        y=alt.Y("count()", title="Minutes"),
        color=alt.Color("intensity", scale=alt.Scale(scheme="orangered"))
        # scale=alt.Scale(range=["white", "grey"])),
    )
).properties(width=400, height=400)
```

[82]:



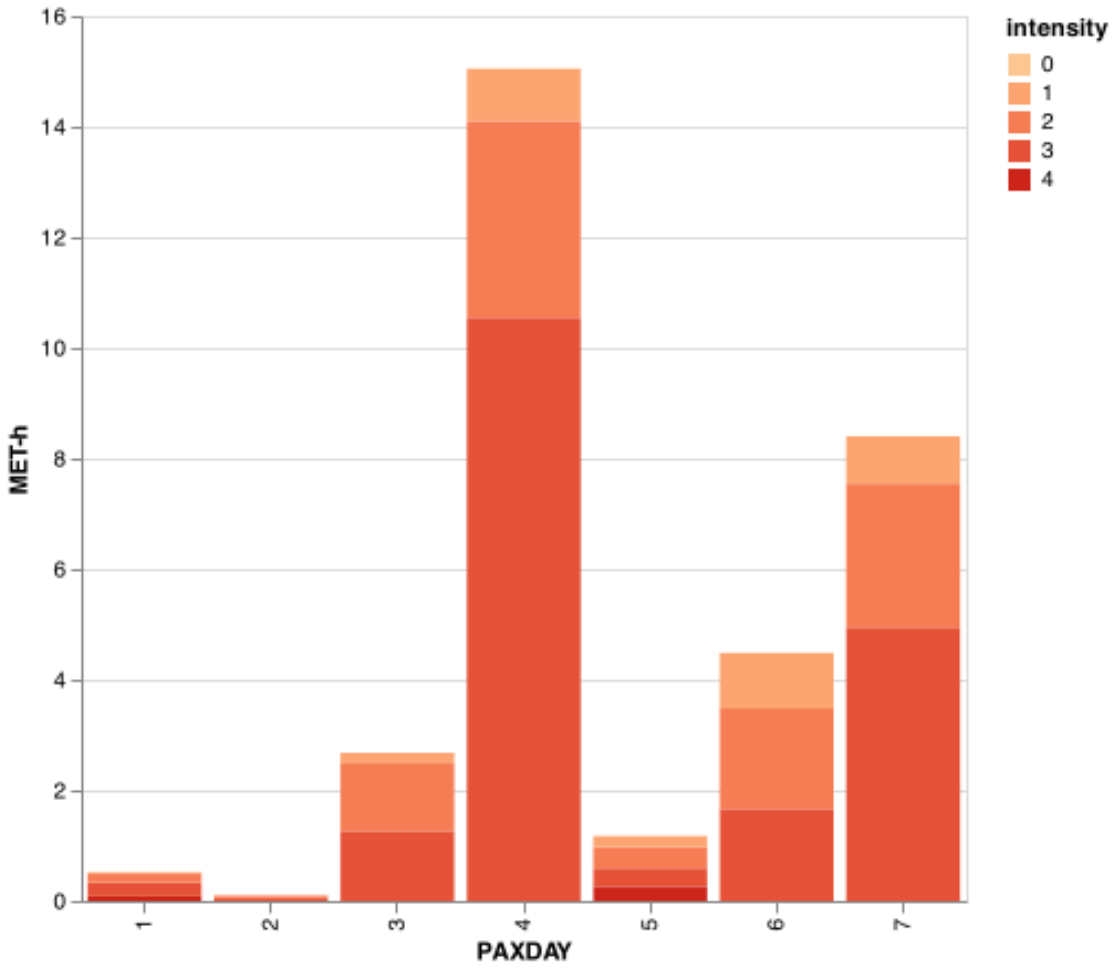
[83]:
```
(
    alt.Chart(paxraw_sample)
    .mark_bar()
    .encode(
        x="PAXDAY:O",
        y=alt.Y("sum(METh)", title="MET-h"),
        color=alt.Color("intensity", scale=alt.Scale(scheme="orangered"))
        # scale=alt.Scale(range=["white", "grey"])),
    )
).properties(width=400, height=400)
```

[83]:



[84]:
```
(
    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)
```

[84]:



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             33134853.0
sed_bout           811951.0
low_bout          5017360.0
light_bout         602056.0
moderate_bout      230393.0
vigorous_bout      832449.0
no_bout          26473093.0
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]:
```
32322902
```

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

[87]:
```
5017360
```

[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		
21005	1	202.0	0
	2	76.0	0
	3	243.0	0
	4	873.0	1
	5	203.0	0
	6	681.0	1
	7	875.0	1
21006	1	960.0	1
	2	712.0	1
	3	597.0	0
	4	448.0	0
	5	581.0	0
	6	615.0	1
	7	650.0	1
21007	1	737.0	1

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

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

[96]:

		worn_sum	valid_day
SEQN	PAXDAY		
21005	1	202.0	0
	2	76.0	0
	3	243.0	0
	4	873.0	1
	5	203.0	0
	6	681.0	1
	7	875.0	1

```
21006 1          960.0          1
      2          712.0          1
      3          597.0          0
      4          448.0          0
      5          581.0          0
      6          615.0          1
      7          650.0          1
21007 1          737.0          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	21005	1	32695	0	0	
1	21005	2	5380	0	0	
2	21005	3	143783	0	0	
3	21005	4	851618	0	0	
4	21005	5	74453	0	0	
5	21005	6	249123	0	0	
6	21005	7	469084	0	0	
7	21006	1	217932	0	0	
8	21006	2	69560	0	0	
9	21006	3	206583	0	0	
10	21006	4	99358	0	0	
11	21006	5	185269	156	0	
12	21006	6	150717	726	0	
13	21006	7	30908	0	0	
14	21007	1	438907	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                        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_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	21005	1	32695	0	0	
1	21005	2	5380	0	0	
2	21005	3	143783	0	0	
3	21005	4	851618	0	0	
4	21005	5	74453	0	0	
5	21005	6	249123	0	0	
6	21005	7	469084	0	0	
7	21006	1	217932	0	0	
8	21006	2	69560	0	0	
9	21006	3	206583	0	0	
10	21006	4	99358	0	0	
11	21006	5	185269	156	0	
12	21006	6	150717	726	0	
13	21006	7	30908	0	0	
14	21007	1	438907	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                        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_300_last  worn_sum  valid_day
0                         0     202.0          0
1                         0      76.0          0
2                         0     243.0          0
3                         0     873.0          1
4                         0     203.0          0
5                         0     681.0          1
6                         0     875.0          1
7                         0     960.0          1
8                         0     712.0          1
9                         0     597.0          0
10                        0     448.0          0
11                        0     581.0          0
12                        0     615.0          1
13                        0     650.0          1
14                        0     737.0          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/2003-2004/paxraw_c_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
21005	0	0	0
21006	0	0	0
21007	0	0	0
21008	0	0	0
21009	0	0	0
21010	0	0	0
21012	0	0	0
21013	0	0	0
21015	0	0	0
21016	0	0	0
21017	0	0	0
21018	0	0	0
21019	0	0	0
21020	0	0	0
21024	0	0	0

SEQN	out_of_calibration_sum	out_of_calibration_last	unreliable_sum
21005	0	0	0
21006	0	0	0
21007	0	0	0
21008	0	0	0
21009	0	0	0
21010	0	0	0
21012	0	0	0
21013	10080	1	0
21015	0	0	0
21016	10080	1	0
21017	0	0	0
21018	0	0	0
21019	0	0	0
21020	0	0	0
21024	10080	1	0

SEQN	unreliable_last	steps_filtered_500_sum	steps_filtered_300_sum
21005	0	0.0	0.0
21006	0	0.0	0.0

```
21007                0                  0.0                  0.0
21008                0                  0.0                  0.0
21009                0                  0.0                  0.0
21010                0                  0.0                  0.0
21012                0                  0.0                  0.0
21013                0                  0.0                  0.0
21015                0                  0.0                  0.0
21016                0                  0.0                  0.0
21017                0                  0.0                  0.0
21018                0                  0.0                  0.0
21019                0                  0.0                  0.0
21020                0                  0.0                  0.0
21024                0                  0.0                  0.0
```

```
       valid_day   PAXINTEN_sum
SEQN
21005          3  260876.571429
21006          4  137189.571429
21007          7  361203.714286
21008          3  237395.142857
21009          7  409360.142857
21010          7  286408.571429
21012          7  130833.428571
21013          5  173304.142857
21015          7  102570.571429
21016          5  234338.285714
21017          6  359232.285714
21018          0   71928.285714
21019          6  438231.285714
21020          6  129705.428571
21024          0  150302.857143
```

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

[111]:
```
(7176, 11)
```

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

[112]:
```
PosixPath('/Users/mm51929/projects/2022/07-nhanes-
analysis/data/raw/2003-2004/paxraw_c_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	out_of_calibration_sum	out_of_calibration_last	unreliable_sum
21005	0	0	0	0	0	0
21006	0	0	0	0	0	0
21007	0	0	0	0	0	0
21008	0	0	0	0	0	0
21009	0	0	0	0	0	0
21010	0	0	0	0	0	0
21012	0	0	0	0	0	0
21015	0	0	0	0	0	0
21017	0	0	0	0	0	0
21018	0	0	0	0	0	0
21019	0	0	0	0	0	0
21020	0	0	0	0	0	0
21025	0	0	0	0	0	0
21026	0	0	0	0	0	0
21027	0	0	0	0	0	0

```
21027                       0                       0               0

       unreliable_last  steps_filtered_500_sum  steps_filtered_300_sum  \
SEQN
21005                0                     0.0                     0.0
21006                0                     0.0                     0.0
21007                0                     0.0                     0.0
21008                0                     0.0                     0.0
21009                0                     0.0                     0.0
21010                0                     0.0                     0.0
21012                0                     0.0                     0.0
21015                0                     0.0                     0.0
21017                0                     0.0                     0.0
21018                0                     0.0                     0.0
21019                0                     0.0                     0.0
21020                0                     0.0                     0.0
21025                0                     0.0                     0.0
21026                0                     0.0                     0.0
21027                0                     0.0                     0.0

       valid_day   PAXINTEN_sum
SEQN
21005          3  260876.571429
21006          4  137189.571429
21007          7  361203.714286
21008          3  237395.142857
21009          7  409360.142857
21010          7  286408.571429
21012          7  130833.428571
21015          7  102570.571429
21017          6  359232.285714
21018          0   71928.285714
21019          6  438231.285714
21020          6  129705.428571
21025          7  299464.428571
21026          5  376165.428571
21027          7  708422.000000
```

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

[115]:
```
(6807, 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/2003-2004/paxraw_c_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	MET	\
0	21005	1	1	4	4321	0	0	0	1.0	
1	21005	1	1	4	4322	0	1	0	1.0	
2	21005	1	1	4	4323	0	2	0	1.0	
3	21005	1	1	4	4324	0	3	0	1.0	
4	21005	1	1	4	4325	0	4	0	1.0	
5	21005	1	1	4	4326	0	5	0	1.0	
6	21005	1	1	4	4327	0	6	0	1.0	
7	21005	1	1	4	4328	0	7	0	1.0	
8	21005	1	1	4	4329	0	8	0	1.0	
9	21005	1	1	4	4330	0	9	0	1.0	

	worn	vigorous_bout	moderate_bout	light_bout	low_bout	sed_bout	\
0	0.0	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	
5	0.0	0.0	0.0	0.0	0.0	0.0	
6	0.0	0.0	0.0	0.0	0.0	0.0	
7	0.0	0.0	0.0	0.0	0.0	0.0	
8	0.0	0.0	0.0	0.0	0.0	0.0	
9	0.0	0.0	0.0	0.0	0.0	0.0	

	no_bout	intensity	METh	activeMETh	max_intensity	out_of_calibration	\
0	0	0	0.016667	0.0	0	0	
1	0	0	0.016667	0.0	0	0	
2	0	0	0.016667	0.0	0	0	
3	0	0	0.016667	0.0	0	0	
4	0	0	0.016667	0.0	0	0	
5	0	0	0.016667	0.0	0	0	
6	0	0	0.016667	0.0	0	0	
7	0	0	0.016667	0.0	0	0	
8	0	0	0.016667	0.0	0	0	
9	0	0	0.016667	0.0	0	0	

```
   unreliable  PAXSTEP  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     873.0          1
1                   0                   0     873.0          1
2                   0                   0     873.0          1
3                   0                   0     873.0          1
4                   0                   0     873.0          1
5                   0                   0     873.0          1
6                   0                   0     873.0          1
7                   0                   0     873.0          1
8                   0                   0     873.0          1
9                   0                   0     873.0          1
```

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

[119]:
```
(46054080, 29)
```

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

[120]:
```
(46054080, 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/2003-2004/paxraw_c_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            0.456977
2            1.403818
3            2.719815
4            0.050037
```

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

[124]:
```
activeMETh    4.630647
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    4.630647
dtype: float64
```

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

[126]:
```
                        activeMETh
SEQN  intensity PAXDAY
21005 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
21005 0             0.000000
      1             0.399830
      2             1.139354
      3             2.451585
      4             0.000000
```

[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           770.206608
1           183.109975
2            64.498677
3            22.306610
4             1.377937
```

[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             1.066653
2             1.657452
3             1.174534
4             0.124645
```

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

```
)
minutes_METh
```

[130]:
```
              minutes  activeMETh
intensity
0          770.206608    0.000000
1          183.109975    1.066653
2           64.498677    1.657452
3           22.306610    1.174534
4            1.377937    0.124645
```

[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]:

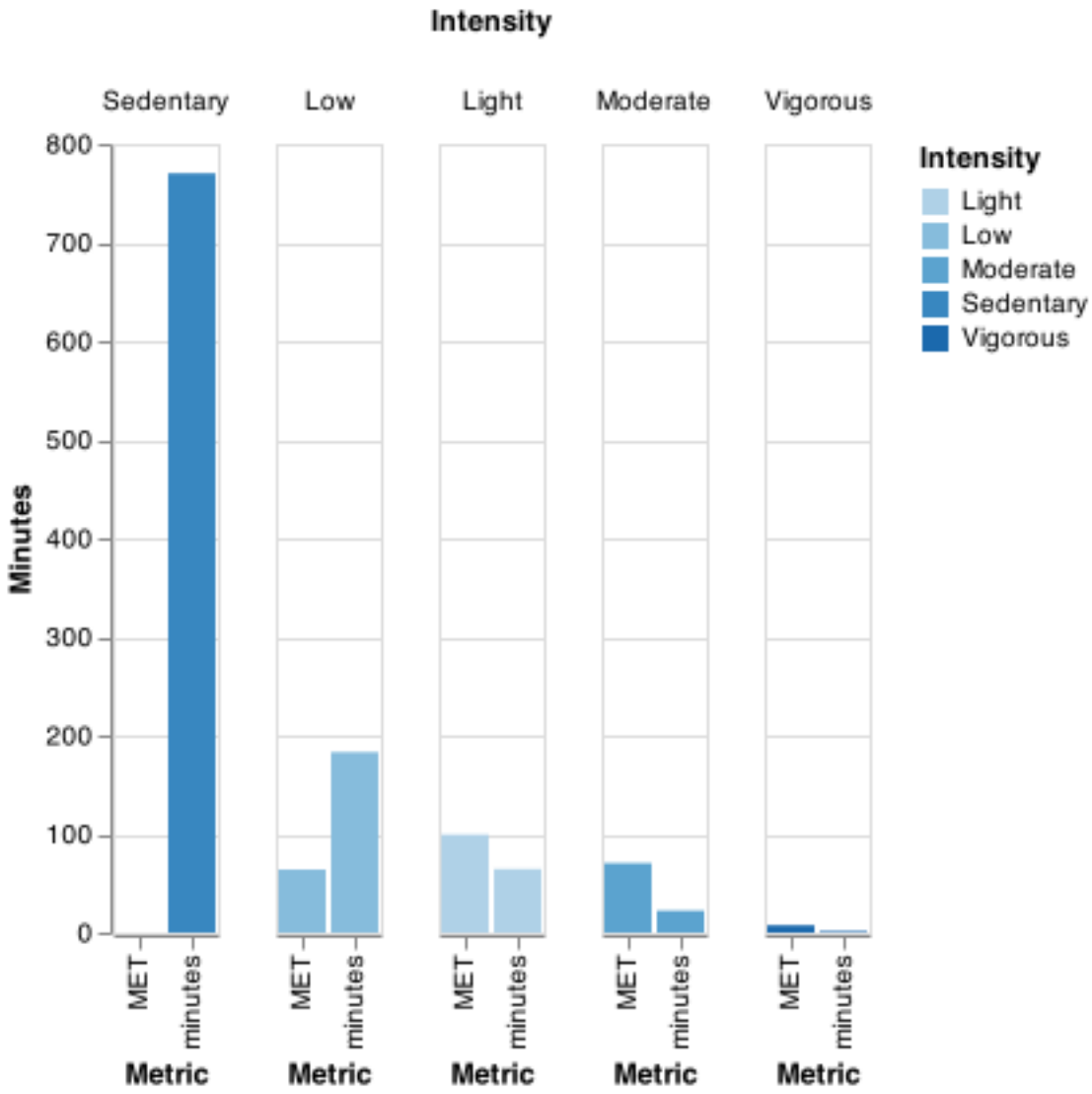
	intensity	activeMETh	metric	label
0	0	770.206608	minutes	Sedentary
1	1	183.109975	minutes	Low
2	2	64.498677	minutes	Light
3	3	22.306610	minutes	Moderate
4	4	1.377937	minutes	Vigorous
5	0	0.000000	MET	Sedentary
6	1	63.999158	MET	Low
7	2	99.447142	MET	Light
8	3	70.472058	MET	Moderate
9	4	7.478697	MET	Vigorous

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]:

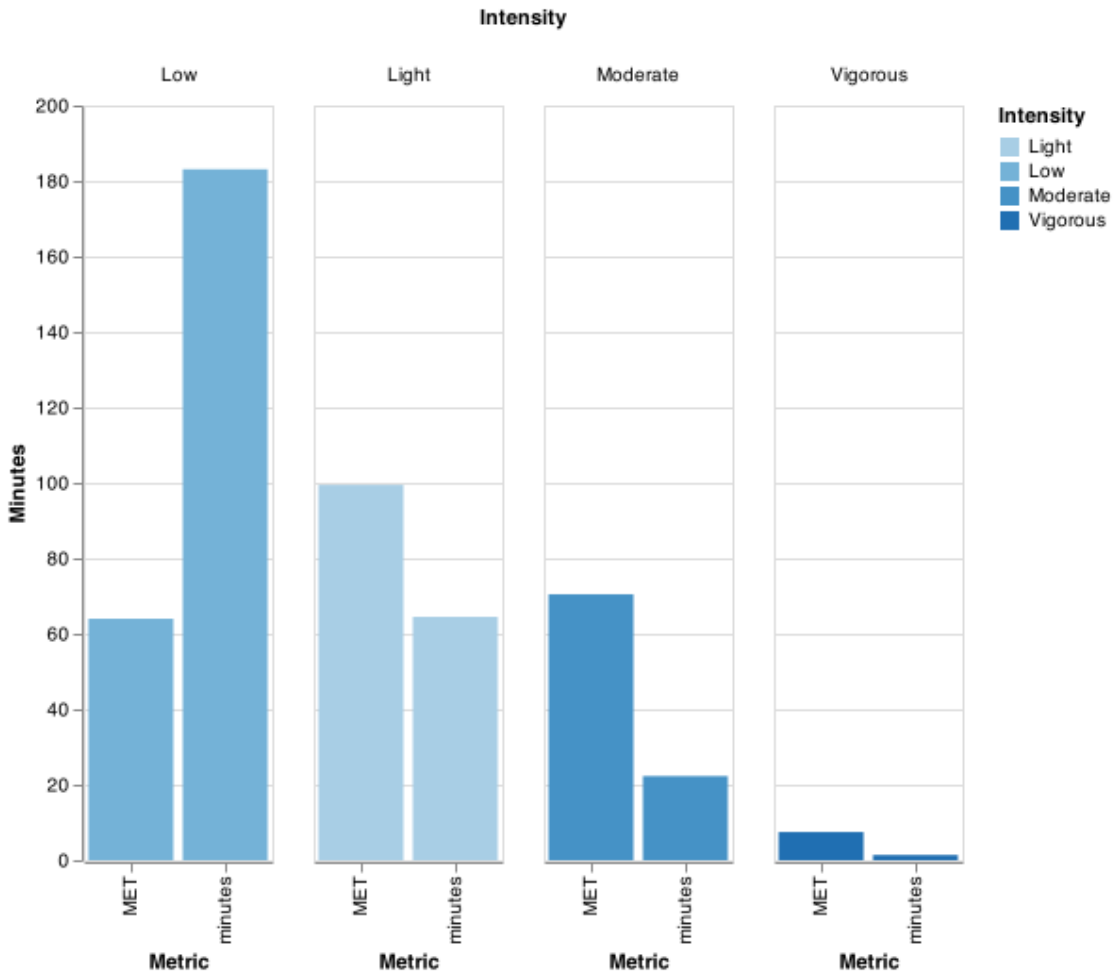


Skip sedentary - no METs

[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    28.16299
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
```

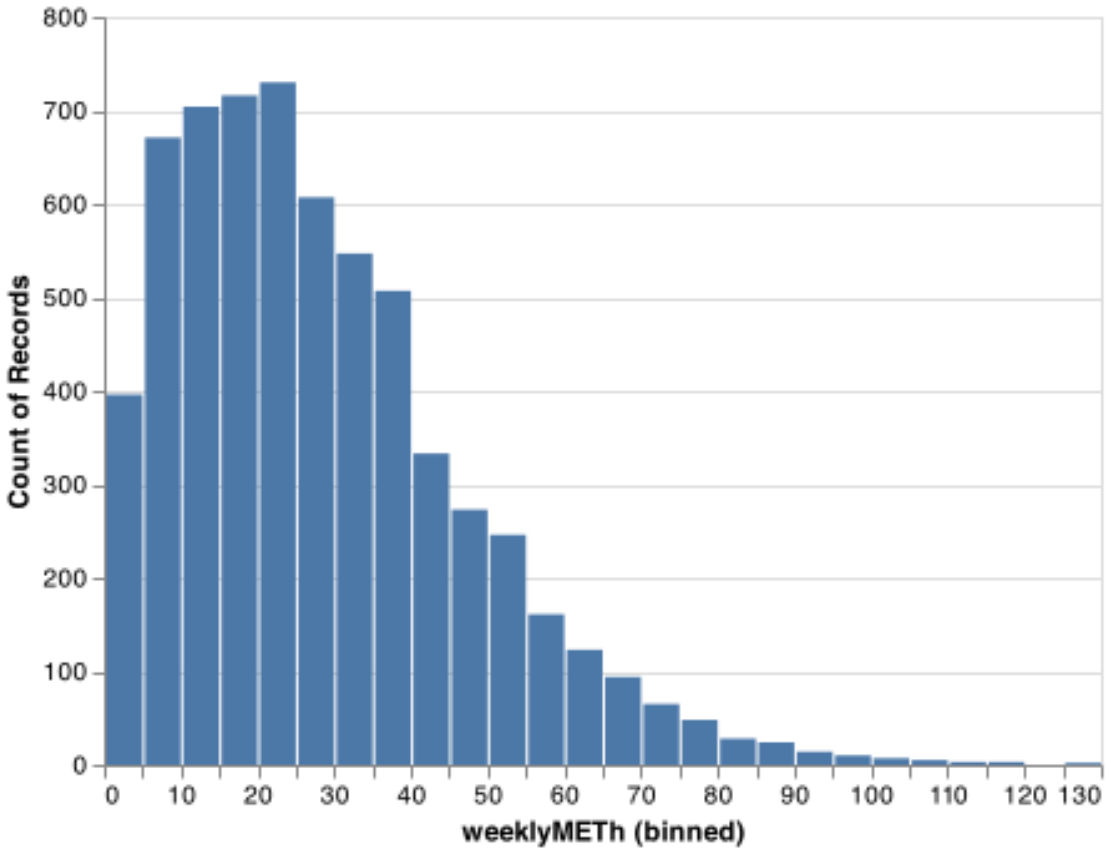
[135]:

SEQN	activeMETh	weeklyMETh
21005	3.990769	27.935385
21006	1.173526	8.214684
21007	6.617710	46.323967
21008	1.590891	11.136237
21009	7.702531	53.917718
…	…	…
31114	3.343096	23.401669
31115	1.903368	13.323576
31119	6.989679	48.927752
31121	3.109121	21.763850
31124	2.853454	19.974178

[6317 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
21005     (24.555, 29.75]
21006     (6.746, 11.633]
21007    (42.428, 54.022]
21008     (6.746, 11.633]
21009    (42.428, 54.022]
                 ...
31114    (20.245, 24.555]
31115    (11.633, 15.772]
31119    (42.428, 54.022]
31121    (20.245, 24.555]
31124    (15.772, 20.245]
Name: weeklyMETh, Length: 6317, dtype: category
Categories (10, interval[float64, right]): [(0.16, 6.746] < (6.746, 11.633] <
```

```
(11.633, 15.772] < (15.772, 20.245] … (29.75, 35.486] < (35.486, 42.428] <
(42.428, 54.022] < (54.022, 129.914]]
```

[138]:
```
cuts
```

[138]:
```
array([  0.16051046,   6.74648629,  11.63271596,  15.77151506,
        20.24492932,  24.55485369,  29.75031654,  35.4864207 ,
        42.42808025,  54.0221748 , 129.91367408])
```

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

[139]:
```
IntervalIndex([(0.16, 6.746], (6.746, 11.633], (11.633, 15.772], (15.772,
20.245], (20.245, 24.555], (24.555, 29.75], (29.75, 35.486], (35.486, 42.428],
(42.428, 54.022], (54.022, 129.914]], 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)
        ],
    }
)
```

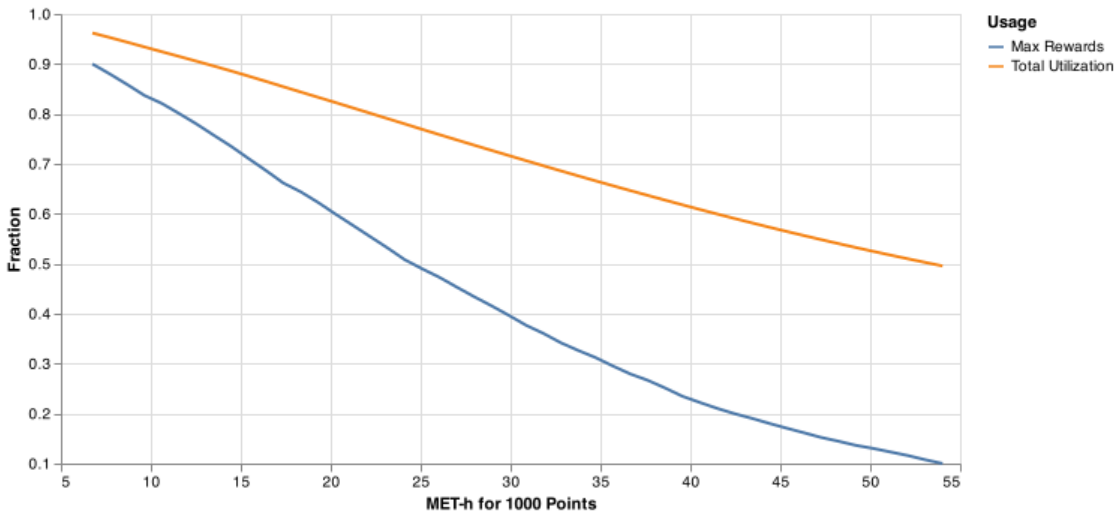
[141]:
```
utilization.head()
```

[141]:

	target	Total Utilization	Max Rewards
0	6.746486	0.961888	0.899953
1	7.711296	0.952972	0.880006
2	8.676106	0.943589	0.858952
3	9.640916	0.934003	0.836948
4	10.605726	0.924469	0.820959

[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
)
```

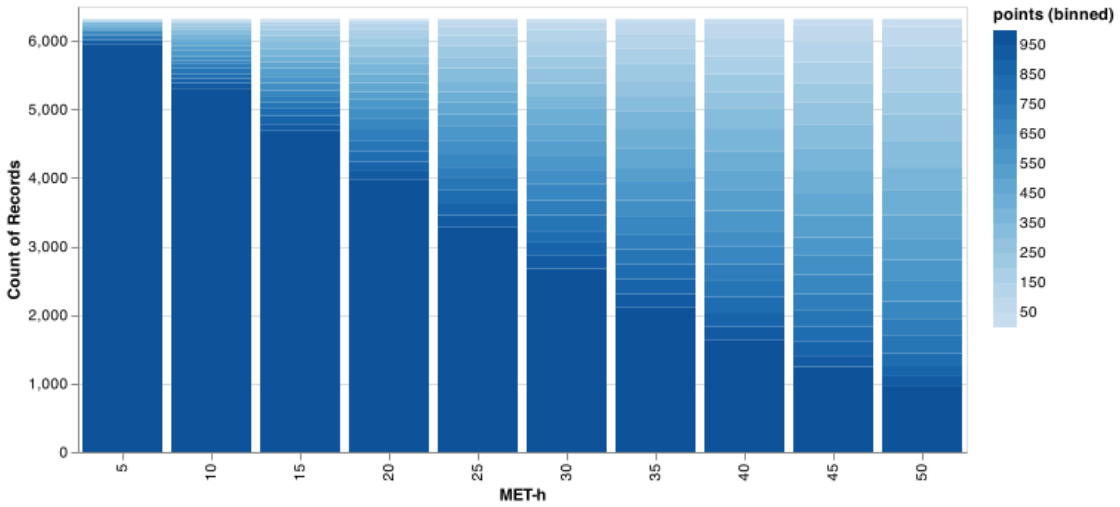
[142]:



[143]:

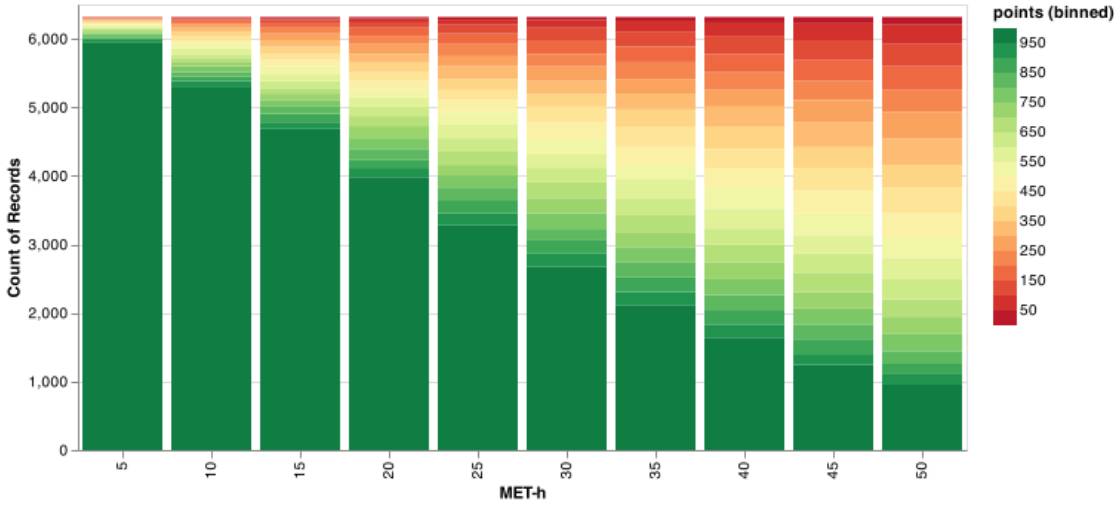
```
alt.Chart(
    pd.concat(
        [
            pd.DataFrame({"points": np.minimum(METh.weeklyMETh.values / x, 1) *␣
↪1000, "MET-h": x})
            for x in np.arange(5, 55, 5)
        ]
    )
).mark_bar().encode(
    alt.Color("points:Q", bin=alt.BinParams(maxbins=20)),
    alt.Y("count()"),
    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	21005	0
1	21005	1
2	21005	2
3	21005	3
4	21005	4
5	21006	0
6	21006	1
7	21006	2
8	21006	3
9	21006	4
10	21007	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	21005	0	0.000000	189.428571	0.000000	0.000000
1	1	21005	1	0.399830	70.428571	2.798813	111.952525
2	2	21005	2	1.139354	41.142857	7.975476	319.019048
3	3	21005	3	2.451585	46.000000	17.161096	686.443830
4	4	21005	4	0.000000	0.000000	0.000000	0.000000

[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
21005	1117.415403	1000.000000	(900, 1000]
21006	328.587368	328.587368	(300, 400]
21007	1852.958677	1000.000000	(900, 1000]
21008	445.449475	445.449475	(400, 500]
21009	2156.708738	1000.000000	(900, 1000]
…	…	…	…
31114	936.066777	936.066777	(900, 1000]
31115	532.943052	532.943052	(500, 600]
31119	1957.110095	1000.000000	(900, 1000]
31121	870.554001	870.554001	(800, 900]
31124	798.967112	798.967112	(700, 800]

[6317 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	64.230375	
1	(0, 100]	1	0.564052	64.230375	
2	(0, 100]	2	0.297794	64.230375	
3	(0, 100]	3	0.125661	64.230375	
4	(0, 100]	4	0.012494	64.230375	
5	(100, 200]	0	0.000000	152.980101	
6	(100, 200]	1	0.485225	152.980101	
7	(100, 200]	2	0.358207	152.980101	
8	(100, 200]	3	0.149147	152.980101	
9	(100, 200]	4	0.007421	152.980101	
10	(200, 300]	0	0.000000	247.575380	
11	(200, 300]	1	0.453118	247.575380	
12	(200, 300]	2	0.367195	247.575380	
13	(200, 300]	3	0.169643	247.575380	
14	(200, 300]	4	0.010044	247.575380	
15	(300, 400]	0	0.000000	346.554535	
16	(300, 400]	1	0.422303	346.554535	
17	(300, 400]	2	0.384059	346.554535	
18	(300, 400]	3	0.181235	346.554535	
19	(300, 400]	4	0.012404	346.554535	
20	(400, 500]	0	0.000000	453.831463	
21	(400, 500]	1	0.419333	453.831463	
22	(400, 500]	2	0.400435	453.831463	

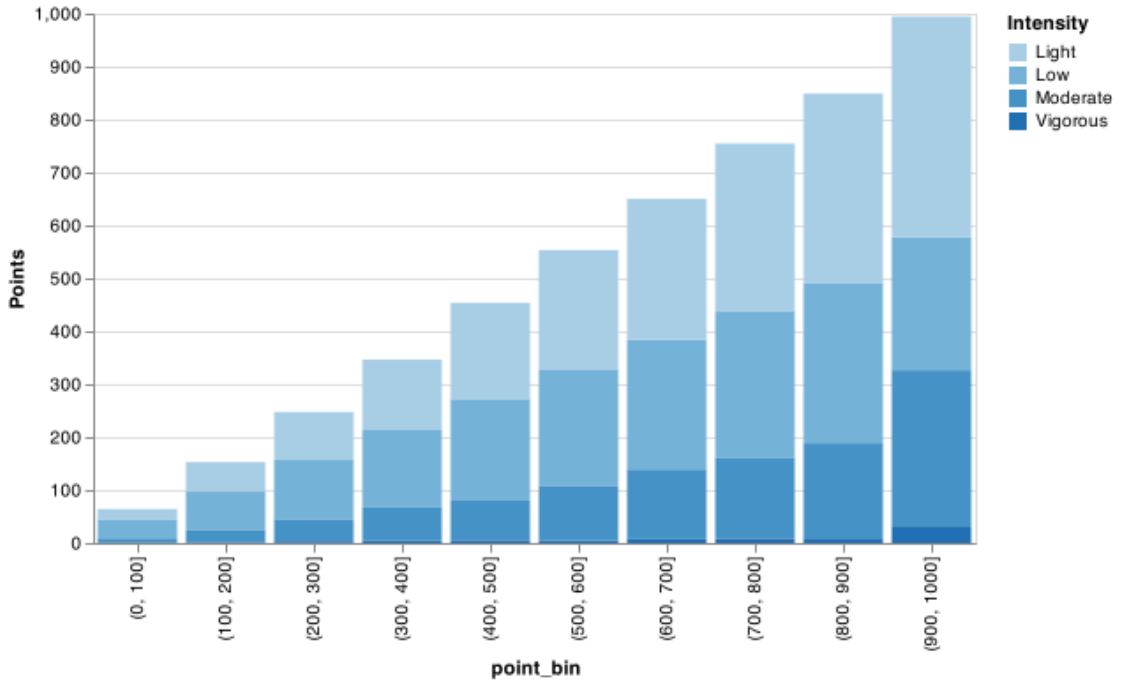
23	(400, 500]	3	0.168370	453.831463
24	(400, 500]	4	0.011862	453.831463
25	(500, 600]	0	0.000000	553.498435
26	(500, 600]	1	0.398930	553.498435
27	(500, 600]	2	0.406199	553.498435
28	(500, 600]	3	0.186593	553.498435
29	(500, 600]	4	0.008278	553.498435
30	(600, 700]	0	0.000000	650.231619
31	(600, 700]	1	0.377926	650.231619
32	(600, 700]	2	0.408501	650.231619
33	(600, 700]	3	0.202391	650.231619
34	(600, 700]	4	0.011181	650.231619
35	(700, 800]	0	0.000000	754.670332
36	(700, 800]	1	0.364894	754.670332
37	(700, 800]	2	0.421201	754.670332
38	(700, 800]	3	0.202787	754.670332
39	(700, 800]	4	0.011119	754.670332
40	(800, 900]	0	0.000000	848.923225
41	(800, 900]	1	0.354546	848.923225
42	(800, 900]	2	0.422568	848.923225
43	(800, 900]	3	0.212162	848.923225
44	(800, 900]	4	0.010723	848.923225
45	(900, 1000]	0	0.000000	994.759198
46	(900, 1000]	1	0.253302	994.759198
47	(900, 1000]	2	0.418776	994.759198
48	(900, 1000]	3	0.297737	994.759198
49	(900, 1000]	4	0.030185	994.759198

	dailyMinutesCapped	points	label
0	214.155280	0.000000	Sedentary
1	32.058385	36.229246	Low
2	2.867081	19.127398	Light
3	0.581366	8.071232	Moderate
4	0.024845	0.802498	Vigorous
5	232.245552	0.000000	Sedentary
6	59.376716	74.229711	Low
7	7.957295	54.798555	Light
8	1.643111	22.816494	Moderate
9	0.045247	1.135342	Vigorous
10	270.923043	0.000000	Sedentary
11	82.987174	112.180915	Low
12	13.190181	90.908522	Light
13	2.911986	41.999400	Moderate
14	0.097302	2.486542	Vigorous
15	284.071018	0.000000	Sedentary
16	103.655172	146.350851	Low
17	19.157225	133.097225	Light

```
18            4.344828   62.807902   Moderate
19            0.173235    4.298558   Vigorous
20          327.832742    0.000000  Sedentary
21          128.510648  190.306357        Low
22           26.314552  181.730143      Light
23            5.330524   76.411481   Moderate
24            0.215173    5.383483   Vigorous
25          333.138743    0.000000  Sedentary
26          143.891174  220.807089        Low
27           32.487659  224.830727      Light
28            7.204188  103.278715   Moderate
29            0.181750    4.581905   Vigorous
30          340.807069    0.000000  Sedentary
31          160.838733  245.739758        Low
32           38.087629  265.620139      Light
33            9.131811  131.601317   Moderate
34            0.285714    7.270404   Vigorous
35          364.809233    0.000000  Sedentary
36          175.909843  275.374414        Low
37           45.518728  317.867573      Light
38           10.632404  153.037295   Moderate
39            0.337544    8.391050   Vigorous
40          388.690222    0.000000  Sedentary
41          189.818564  300.982639        Low
42           51.276165  358.728123      Light
43           12.457219  180.109215   Moderate
44            0.364782    9.103249   Vigorous
45          259.490344    0.000000  Sedentary
46          150.255092  251.974835        Low
47           57.785975  416.581141      Light
48           20.116061  296.176216   Moderate
49            1.190091   30.027006   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
)
```

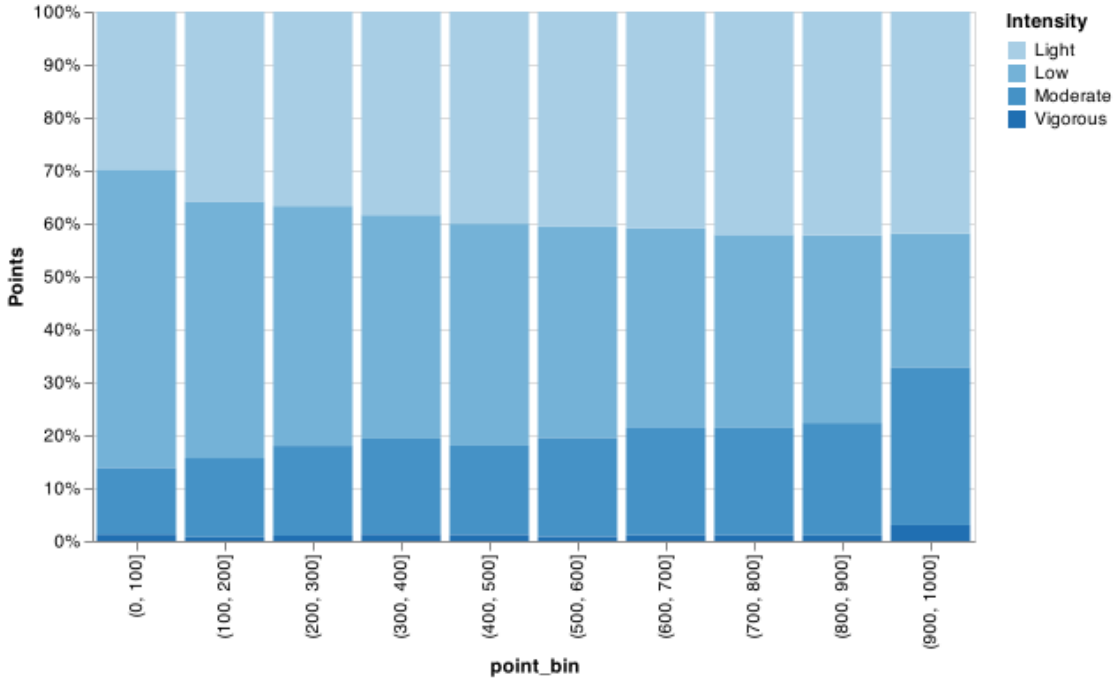
[149]:



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
)
```

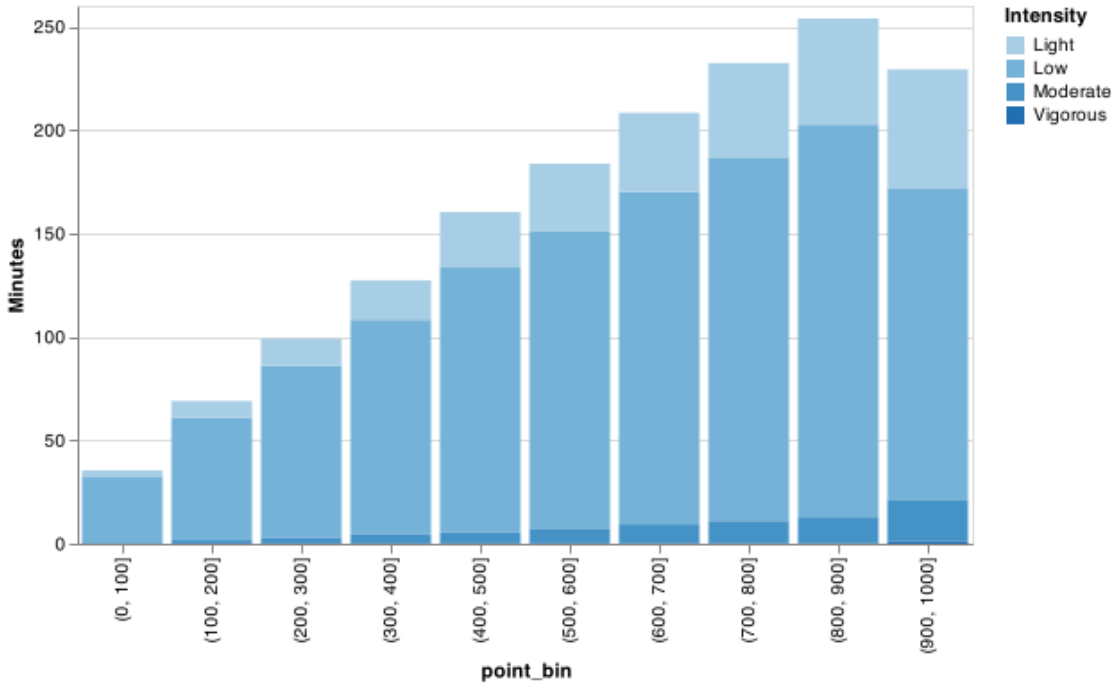
[150]:



7.3.2 Convert this to daily times in zones

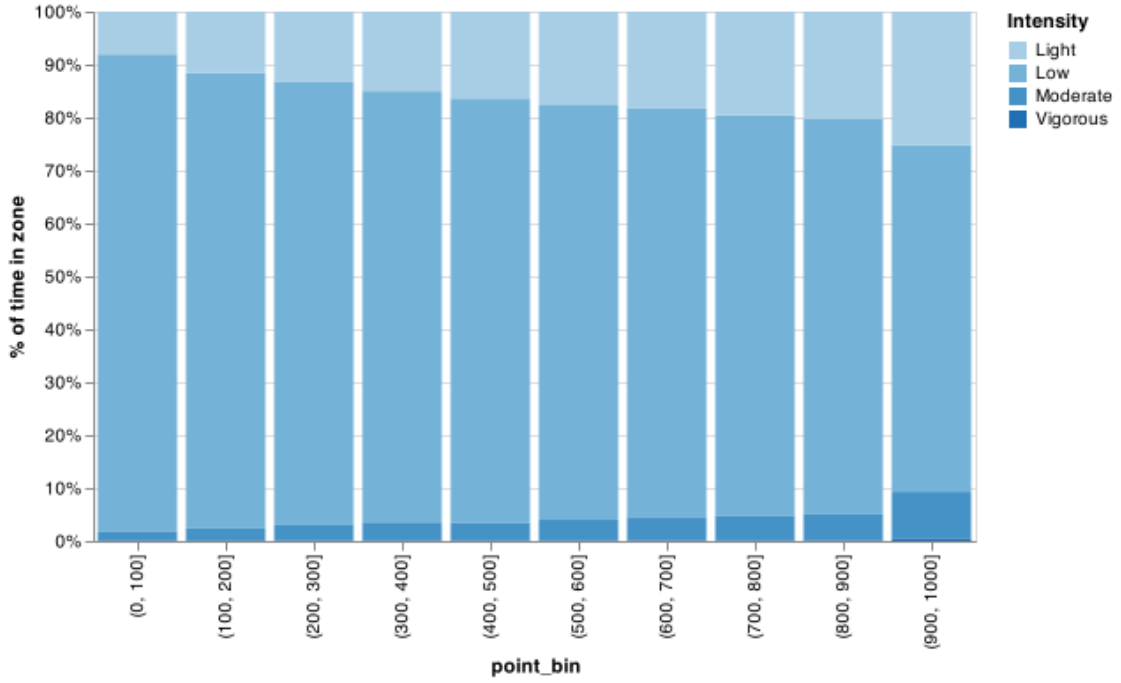
[151]:
```
alt.Chart(
    point_thresholds_by_intensity.loc[point_thresholds_by_intensity.intensity >␣
 ↪0, :]
).mark_bar().encode(
    alt.X("point_bin:O"),
    alt.Y("dailyMinutesCapped:Q", title="Minutes"),
    alt.Detail("label:O", title="Intensity"),
    alt.Color("label:O", title="Intensity", sort=alt.SortField("label",␣
 ↪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]	2.867081	32.058385	0.581366	0.024845
(100, 200]	7.957295	59.376716	1.643111	0.045247
(200, 300]	13.190181	82.987174	2.911986	0.097302
(300, 400]	19.157225	103.655172	4.344828	0.173235
(400, 500]	26.314552	128.510648	5.330524	0.215173
(500, 600]	32.487659	143.891174	7.204188	0.181750
(600, 700]	38.087629	160.838733	9.131811	0.285714
(700, 800]	45.518728	175.909843	10.632404	0.337544
(800, 900]	51.276165	189.818564	12.457219	0.364782
(900, 1000]	57.785975	150.255092	20.116061	1.190091

[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]	32	2	0
(100, 200]	59	7	1
(200, 300]	82	13	3
(300, 400]	103	19	4
(400, 500]	128	26	5
(500, 600]	143	32	7
(600, 700]	160	38	9
(700, 800]	175	45	11
(800, 900]	189	51	13
(900, 1000]	150	57	22

[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]	18	0
(100, 200]	37	1
(200, 300]	54	3
(300, 400]	70	4
(400, 500]	90	5
(500, 600]	104	7
(600, 700]	118	9
(700, 800]	133	11
(800, 900]	146	13
(900, 1000]	132	22

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	21005	1	0	32695	22.704861	6721
1	21005	2	0	5380	3.736111	2336
2	21005	3	0	143783	99.849306	4711
3	21005	4	0	851618	591.401389	5513
4	21005	5	0	74453	51.703472	9313

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

[157]:
```
(50183, 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]:
```
Empty DataFrame
Columns: [SEQN, PAXDAY, PAXSTEP_sum, PAXINTEN_sum, PAXINTEN_mean, PAXINTEN_max]
Index: []
```

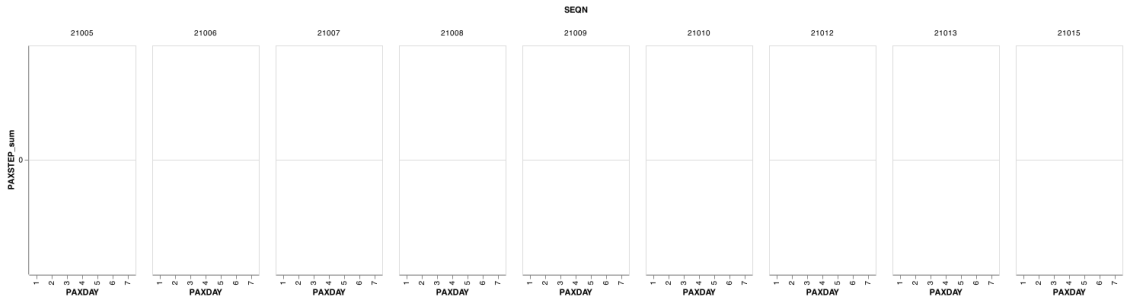
8.2 Daily charts

[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()
```

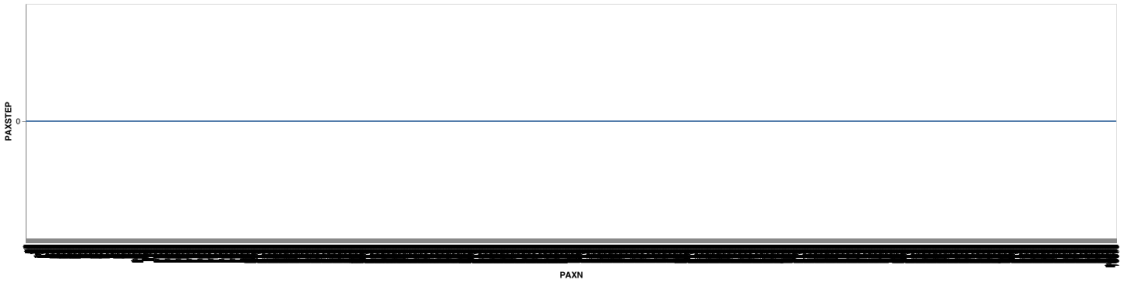
[161]:



8.3 Individual activity/steps

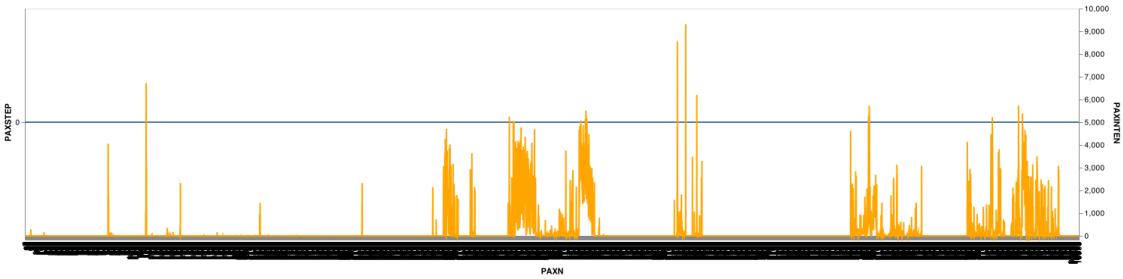
[162]:
```
alt.Chart(paxraw.loc[paxraw.SEQN == by_day.SEQN.unique()[0], :]).mark_line().
↪encode(
    x="PAXN:O", y="PAXSTEP"
).properties(width=1400)
```

[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")
```

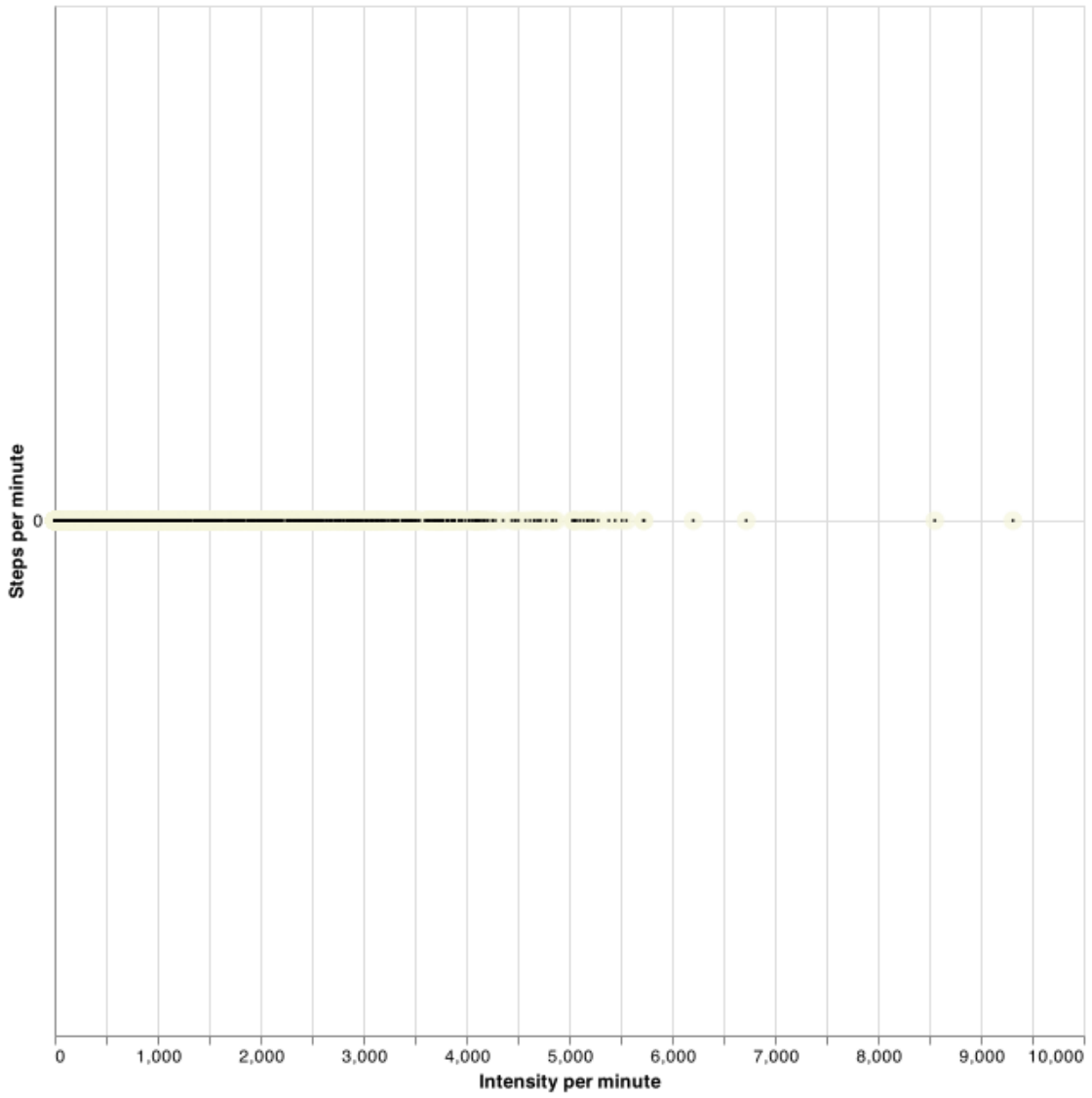
[163]:



[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]:

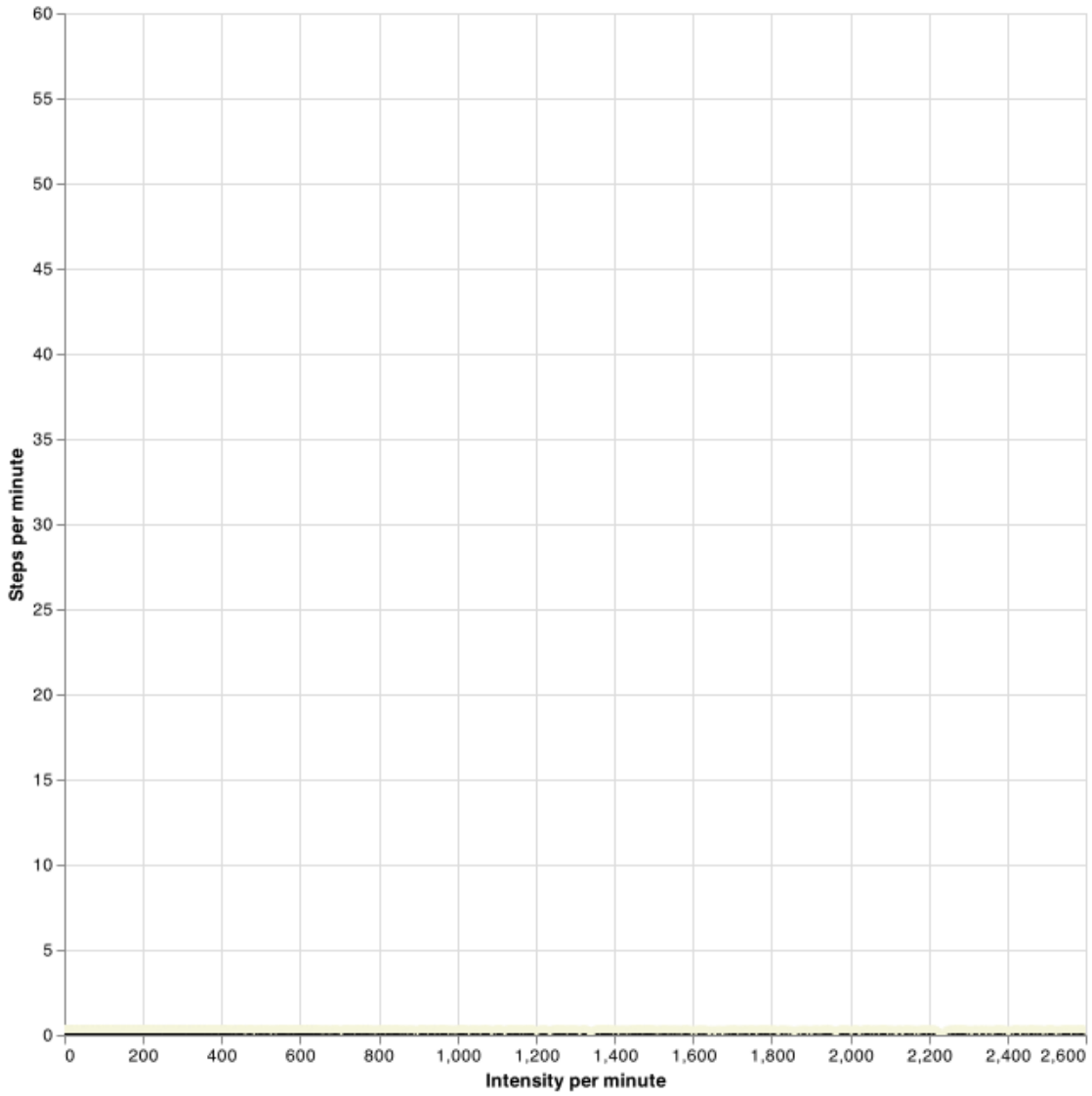


[165]:

```
(
    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]:

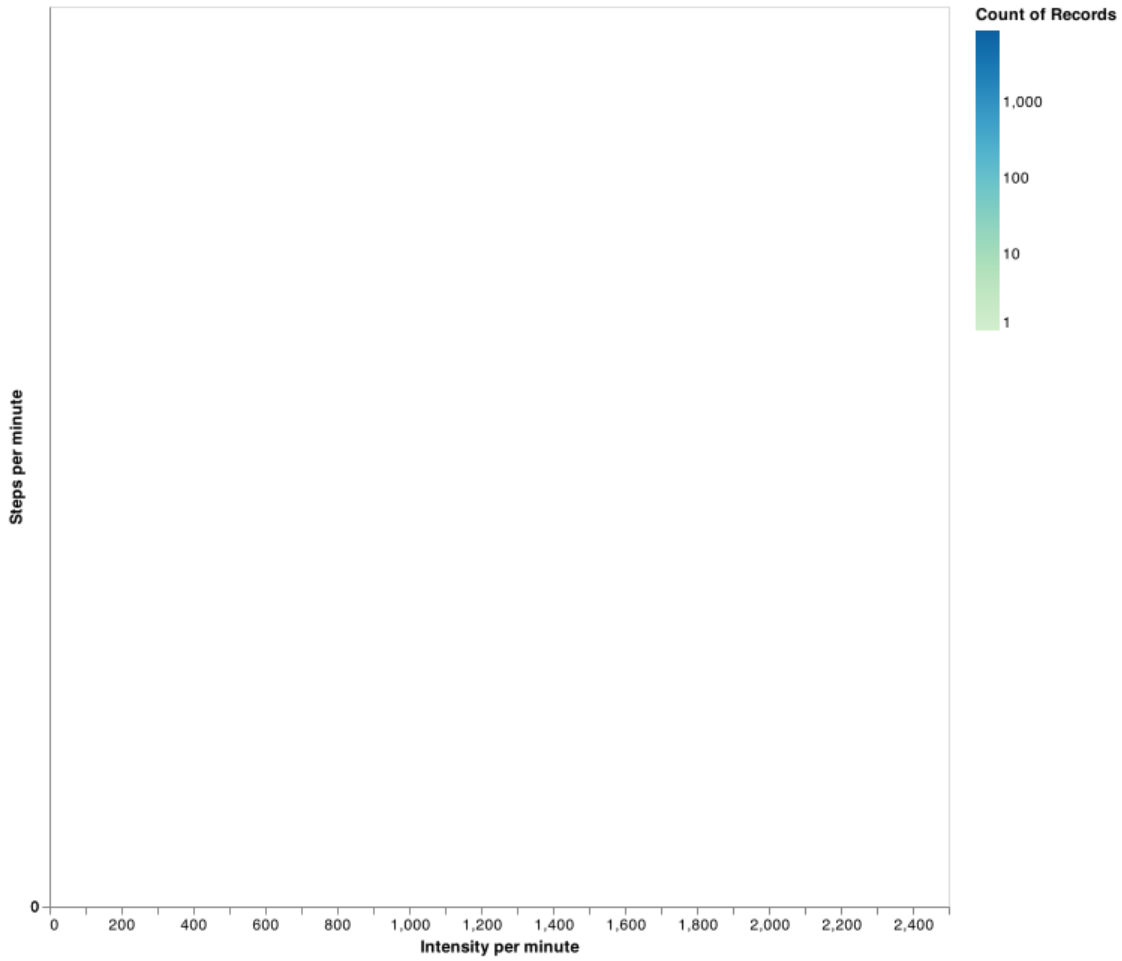


[166]:

```
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]:



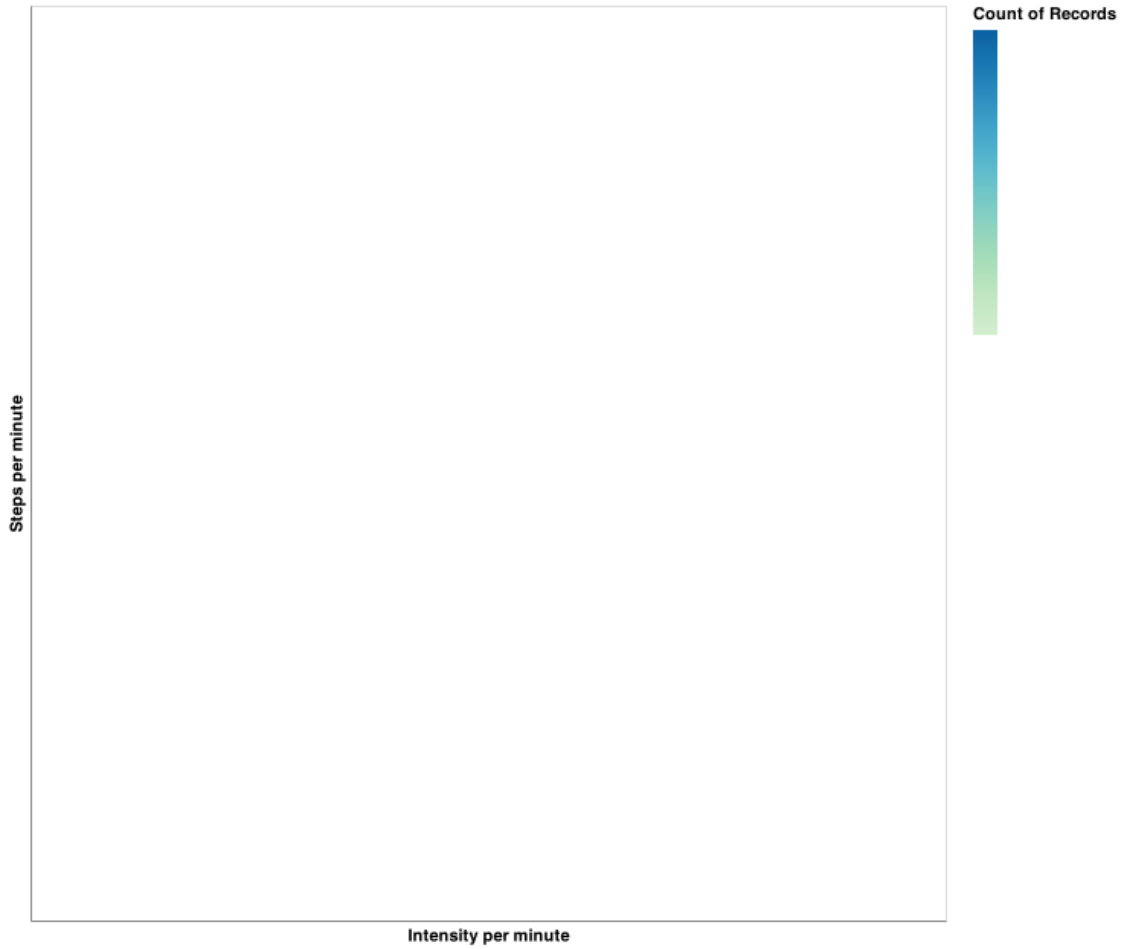
[167]:

```
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
)
```

```
WARN Infinite extent for field "PAXINTEN": [Infinity, -Infinity]
WARN Infinite extent for field "PAXSTEP": [Infinity, -Infinity]
WARN Infinite extent for field "__count": [Infinity, -Infinity]
WARN Log scale domain includes zero: [,]
```

[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	21005	202.0	32695	0
1	21005	76.0	5380	0
2	21005	243.0	143783	0
3	21005	873.0	851618	1
4	21005	203.0	74453	0

[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]:

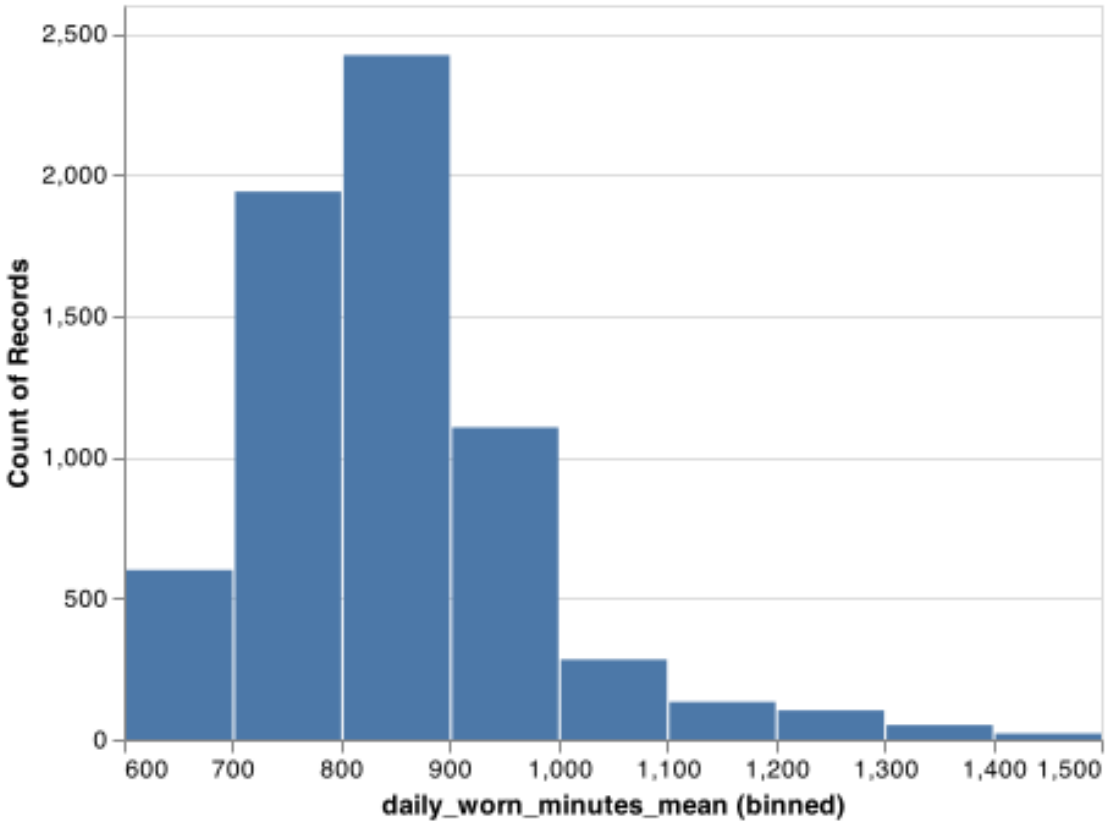
SEQN	daily_worn_minutes_mean	daily_activity_count_sum_mean	n_valid_days
21005	809.666667	523275.000000	3
21006	734.250000	117279.250000	4
21007	911.428571	361203.714286	7
21008	788.333333	214938.333333	3
21009	900.285714	409360.142857	7

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

[172]:
```
(6659, 3)
```

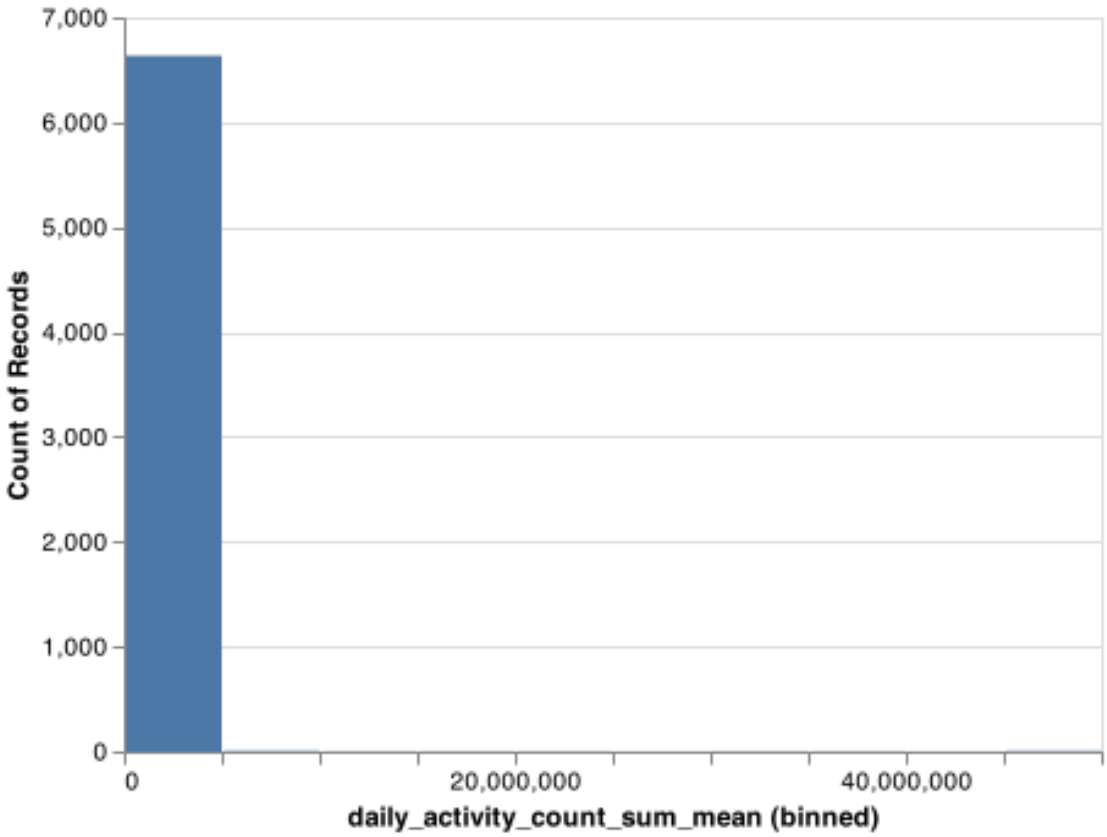
[173]:
```
alt.Chart(person_active_counts_summary).mark_bar().encode(
    alt.X("daily_worn_minutes_mean:Q", bin=True),
    y="count()",
)
```

[173]:



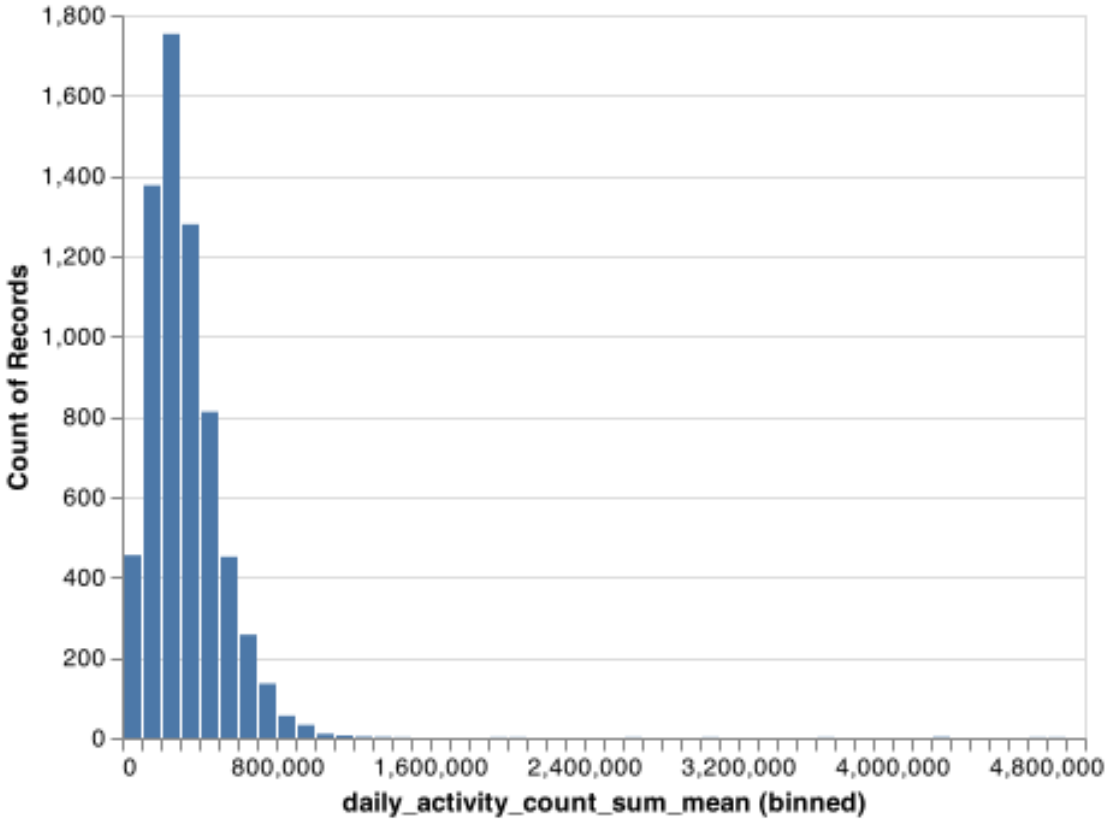
[174]:
```
alt.Chart(person_active_counts_summary).mark_bar().encode(
    alt.X("daily_activity_count_sum_mean:Q", bin=True),
    y="count()",
)
```

[174]:



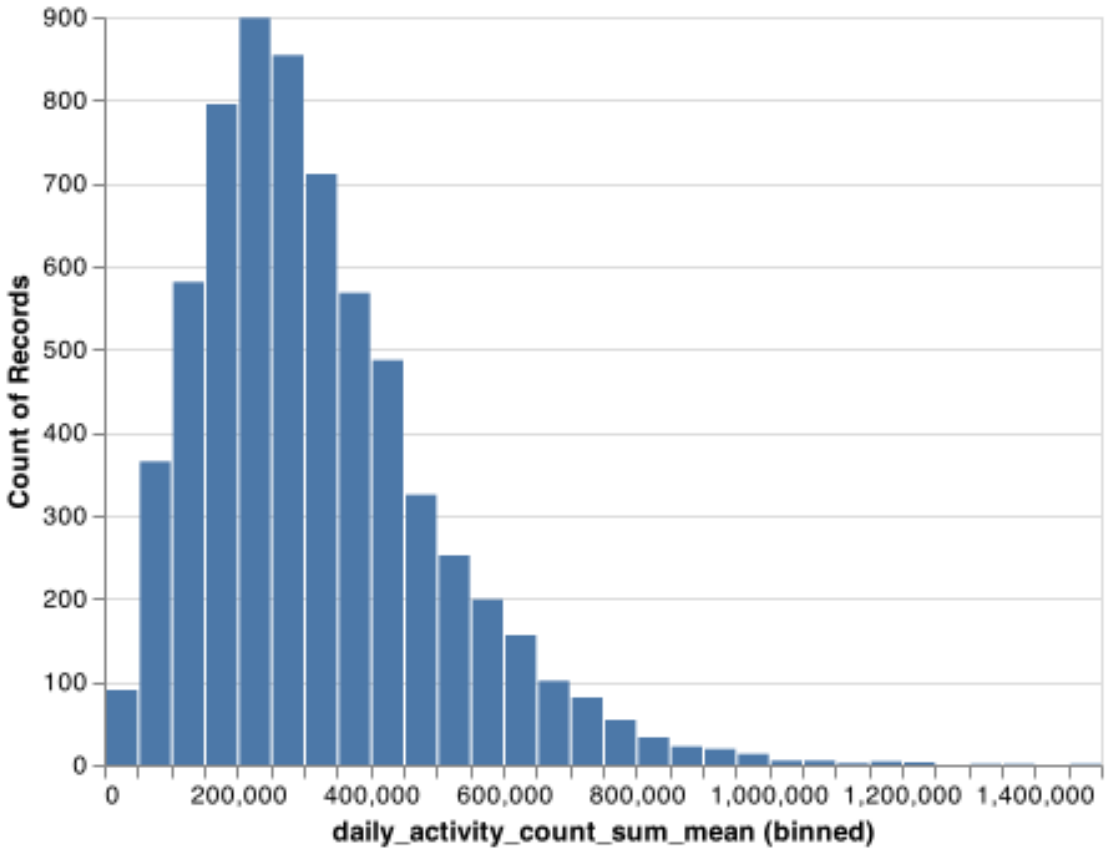
[175]:
```
alt.Chart(person_active_counts_summary).mark_bar(clip=True).encode(
    alt.X(
        "daily_activity_count_sum_mean:Q",
        scale=alt.Scale(domain=[0, 5000000]),
        bin=alt.Bin(maxbins=500),
    ),
    y="count()",
)
```

[175]:



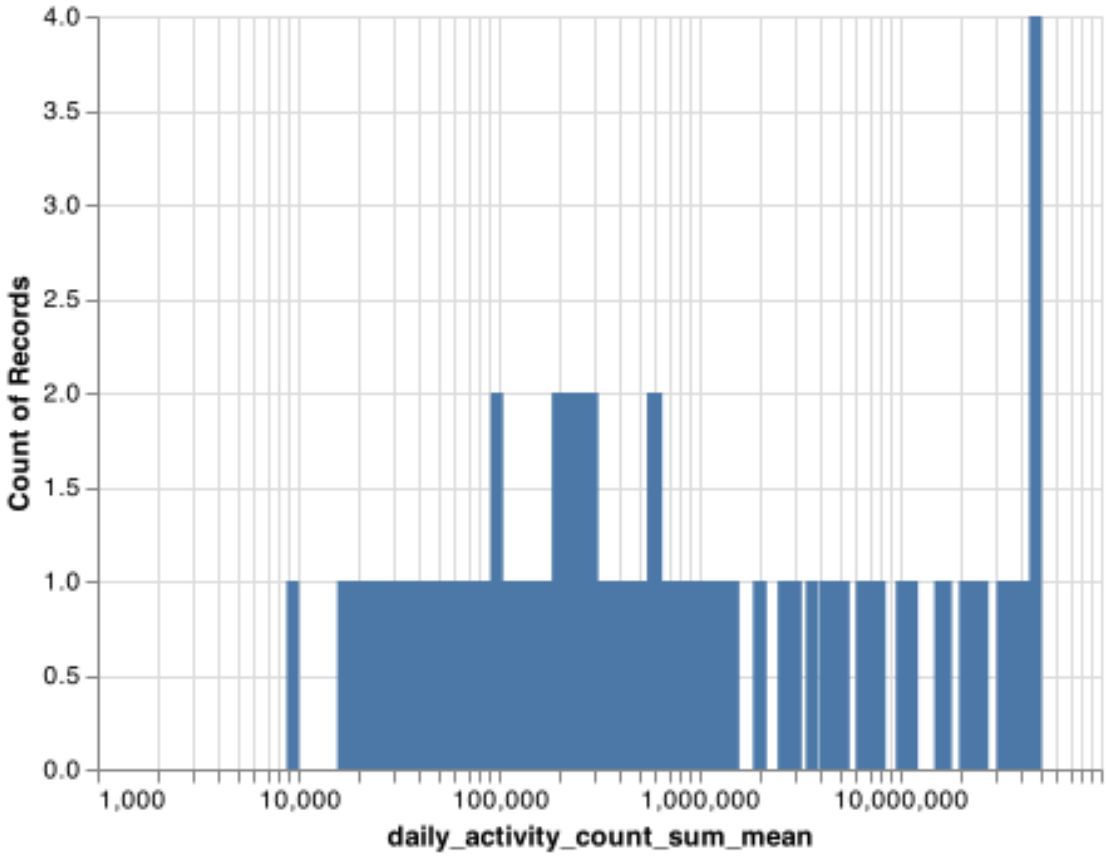
[176]:
```
alt.Chart(person_active_counts_summary).mark_bar(clip=True).encode(
    alt.X(
        "daily_activity_count_sum_mean:Q",
        scale=alt.Scale(domain=[0, 1500000]),
        bin=alt.Bin(maxbins=1000),
    ),
    y="count()",
)
```

[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()",
)
```

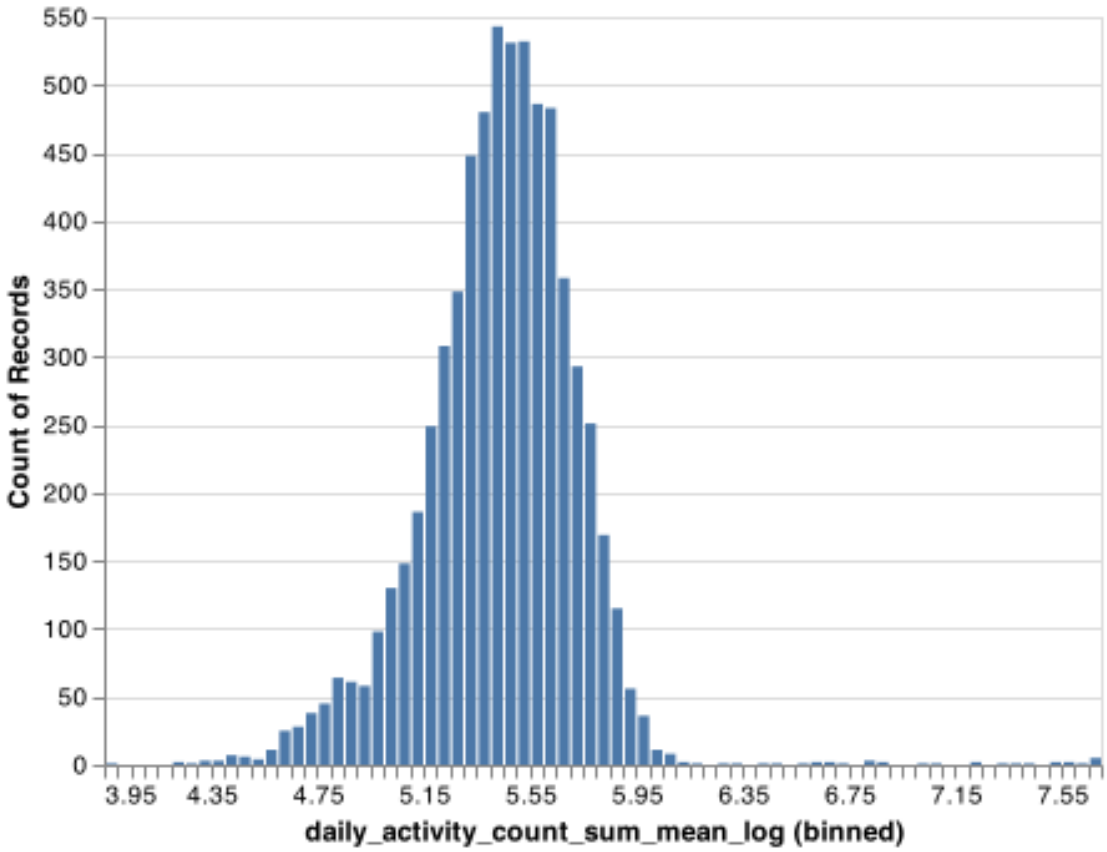
[177]:



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

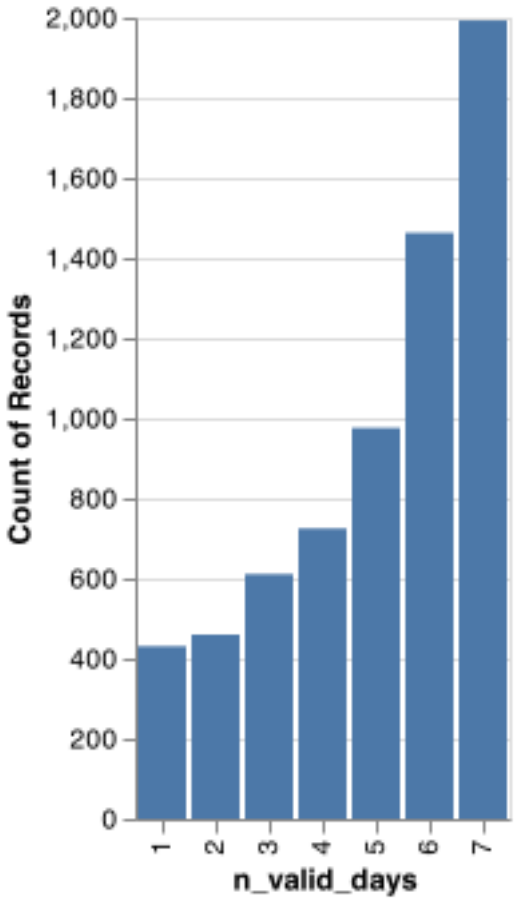
[179]:
```
alt.Chart(person_active_counts_summary).mark_bar(clip=True).encode(
    alt.X(
        "daily_activity_count_sum_mean_log:Q",
        scale=alt.Scale(),
        bin=alt.Bin(maxbins=100),
    ),
    y="count()",
)
```

[179]:



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

[180]:

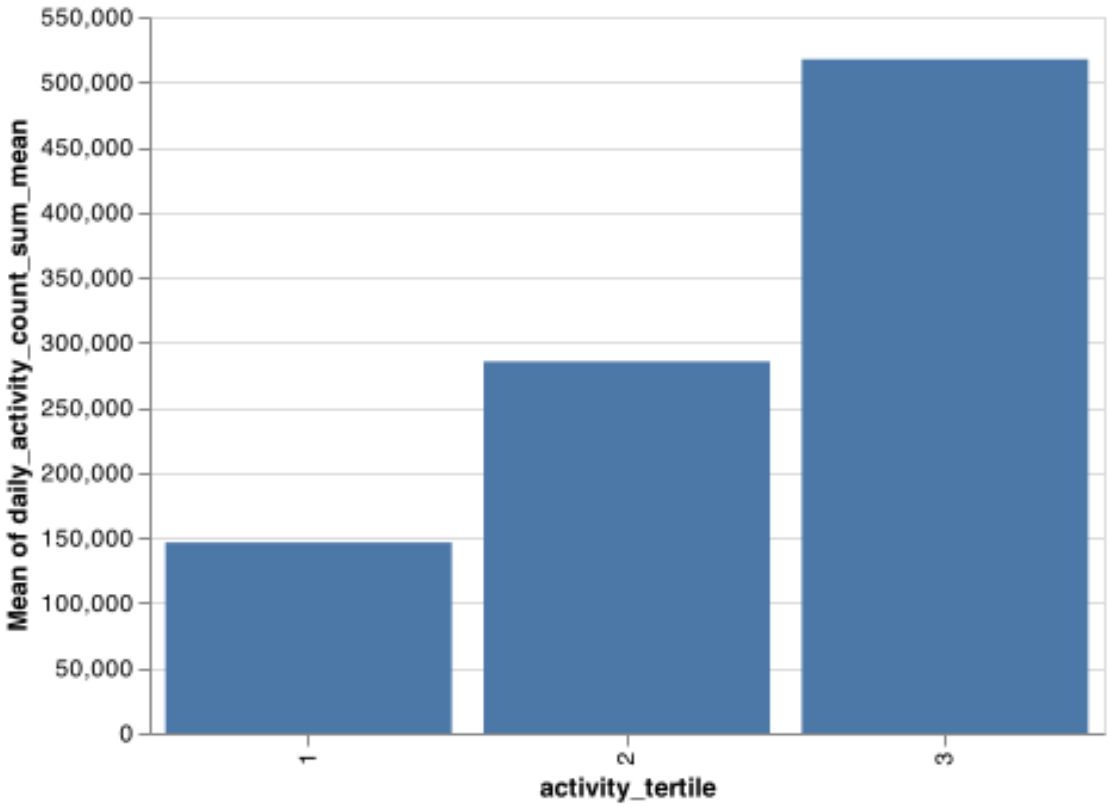


[181]:
```
# clear the crazy data
person_active_counts_summary = person_active_counts_summary.loc[
    person_active_counts_summary.daily_activity_count_sum_mean < 1500000, :
].copy()
```

[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
21316	0	0	858
21901	0	0	14
22055	0	0	10080
22194	0	0	326
22577	0	0	596
22928	0	0	10080
23163	0	0	5338
23772	0	0	10080
23953	0	0	7790
24080	0	0	3287

24719	0	0	1376
26170	0	0	10080
26683	0	0	4936
26976	0	0	1454
26984	0	0	952
27140	0	0	409
27237	0	0	9658
27791	0	0	5333
28310	0	0	41
28744	0	0	2952

SEQN	out_of_calibration_sum	out_of_calibration_last	unreliable_sum	\
21316	10080	1	10080	
21901	0	0	10080	
22055	10080	1	10080	
22194	0	0	10080	
22577	0	0	10080	
22928	10080	1	10080	
23163	0	0	10080	
23772	10080	1	10080	
23953	10080	1	10080	
24080	10080	1	10080	
24719	0	0	10080	
26170	10080	1	10080	
26683	10080	1	10080	
26976	0	0	10080	
26984	0	0	10080	
27140	10080	1	10080	
27237	10080	1	10080	
27791	0	0	10080	
28310	0	0	10080	
28744	10080	1	10080	

SEQN	unreliable_last	steps_filtered_500_sum	steps_filtered_300_sum	\
21316	1	0.0	0.0	
21901	1	0.0	0.0	
22055	1	0.0	0.0	
22194	1	0.0	0.0	
22577	1	0.0	0.0	
22928	1	0.0	0.0	
23163	1	0.0	0.0	
23772	1	0.0	0.0	
23953	1	0.0	0.0	
24080	1	0.0	0.0	
24719	1	0.0	0.0	

```
26170                1                  0.0                  0.0
26683                1                  0.0                  0.0
26976                1                  0.0                  0.0
26984                1                  0.0                  0.0
27140                1                  0.0                  0.0
27237                1                  0.0                  0.0
27791                1                  0.0                  0.0
28310                1                  0.0                  0.0
28744                1                  0.0                  0.0

       valid_day  PAXINTEN_sum
SEQN
21316          7  3.402625e+07
21901          6  6.965657e+05
22055          7  4.718448e+07
22194          0  1.561162e+06
22577          5  3.085106e+06
22928          7  4.718448e+07
23163          3  2.567916e+07
23772          7  4.718448e+07
23953          7  3.705471e+07
24080          7  1.588437e+07
24719          6  6.763425e+06
26170          7  4.718448e+07
26683          5  2.324856e+07
26976          7  7.324363e+06
26984          7  4.832524e+06
27140          5  2.294606e+06
27237          7  4.522121e+07
27791          7  2.563071e+07
28310          2  3.253880e+05
28744          5  1.693966e+07
```

[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
21013                              0                   0                  0
21016                              0                   0                  0
21024                              0                   0                  0
21105                              0                   0                  0
```

21184	0	0	0
21185	0	0	0
21186	0	0	0
21243	0	0	0
21254	0	0	0
21260	0	0	0
21289	0	0	0
21310	0	0	0
21316	0	0	858
21347	0	0	0
21350	0	0	0
21377	0	0	0
21384	0	0	0
21418	0	0	0
21499	0	0	0
21514	0	0	0

SEQN	out_of_calibration_sum	out_of_calibration_last	unreliable_sum	\
21013	10080	1	0	
21016	10080	1	0	
21024	10080	1	0	
21105	10080	1	0	
21184	10080	1	0	
21185	10080	1	0	
21186	10080	1	0	
21243	10080	1	0	
21254	10080	1	0	
21260	10080	1	0	
21289	10080	1	0	
21310	10080	1	10080	
21316	10080	1	10080	
21347	10080	1	10080	
21350	10080	1	0	
21377	10080	1	0	
21384	10080	1	0	
21418	10080	1	0	
21499	10080	1	0	
21514	10080	1	0	

SEQN	unreliable_last	steps_filtered_500_sum	steps_filtered_300_sum	\
21013	0	0.0	0.0	
21016	0	0.0	0.0	
21024	0	0.0	0.0	
21105	0	0.0	0.0	
21184	0	0.0	0.0	

```
21185                0                    0.0                    0.0
21186                0                    0.0                    0.0
21243                0                    0.0                    0.0
21254                0                    0.0                    0.0
21260                0                    0.0                    0.0
21289                0                    0.0                    0.0
21310                1                    0.0                    0.0
21316                1                    0.0                    0.0
21347                1                    0.0                    0.0
21350                0                    0.0                    0.0
21377                0                    0.0                    0.0
21384                0                    0.0                    0.0
21418                0                    0.0                    0.0
21499                0                    0.0                    0.0
21514                0                    0.0                    0.0

       valid_day  PAXINTEN_sum
SEQN
21013          5  1.733041e+05
21016          5  2.343383e+05
21024          0  1.503029e+05
21105          1  4.047857e+04
21184          2  9.412914e+04
21185          7  1.304413e+05
21186          4  2.472670e+05
21243          4  3.139427e+05
21254          0  1.008900e+04
21260          0  9.681071e+04
21289          5  6.196894e+05
21310          7  1.141791e+07
21316          7  3.402625e+07
21347          7  6.753812e+06
21350          2  1.269551e+05
21377          7  3.490004e+05
21384          7  6.331371e+04
21418          6  2.950876e+05
21499          2  2.338179e+05
21514          7  2.610783e+05
```

[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
21013	0	0	0
21016	0	0	0
21024	0	0	0
21105	0	0	0
21184	0	0	0
21185	0	0	0
21186	0	0	0
21243	0	0	0
21254	0	0	0
21260	0	0	0
21289	0	0	0
21310	0	0	0
21316	0	0	858
21347	0	0	0
21350	0	0	0
21377	0	0	0
21384	0	0	0
21418	0	0	0
21499	0	0	0
21514	0	0	0

SEQN	out_of_calibration_sum	out_of_calibration_last	unreliable_sum
21013	10080	1	0
21016	10080	1	0
21024	10080	1	0
21105	10080	1	0
21184	10080	1	0
21185	10080	1	0
21186	10080	1	0
21243	10080	1	0
21254	10080	1	0
21260	10080	1	0
21289	10080	1	0
21310	10080	1	10080
21316	10080	1	10080
21347	10080	1	10080
21350	10080	1	0
21377	10080	1	0
21384	10080	1	0
21418	10080	1	0
21499	10080	1	0
21514	10080	1	0

SEQN	unreliable_last	steps_filtered_500_sum	steps_filtered_300_sum

SEQN			
21013	0	0.0	0.0
21016	0	0.0	0.0
21024	0	0.0	0.0
21105	0	0.0	0.0
21184	0	0.0	0.0
21185	0	0.0	0.0
21186	0	0.0	0.0
21243	0	0.0	0.0
21254	0	0.0	0.0
21260	0	0.0	0.0
21289	0	0.0	0.0
21310	1	0.0	0.0
21316	1	0.0	0.0
21347	1	0.0	0.0
21350	0	0.0	0.0
21377	0	0.0	0.0
21384	0	0.0	0.0
21418	0	0.0	0.0
21499	0	0.0	0.0
21514	0	0.0	0.0

SEQN	valid_day	PAXINTEN_sum
21013	5	1.733041e+05
21016	5	2.343383e+05
21024	0	1.503029e+05
21105	1	4.047857e+04
21184	2	9.412914e+04
21185	7	1.304413e+05
21186	4	2.472670e+05
21243	4	3.139427e+05
21254	0	1.008900e+04
21260	0	9.681071e+04
21289	5	6.196894e+05
21310	7	1.141791e+07
21316	7	3.402625e+07
21347	7	6.753812e+06
21350	2	1.269551e+05
21377	7	3.490004e+05
21384	7	6.331371e+04
21418	6	2.950876e+05
21499	2	2.338179e+05
21514	7	2.610783e+05

[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]:

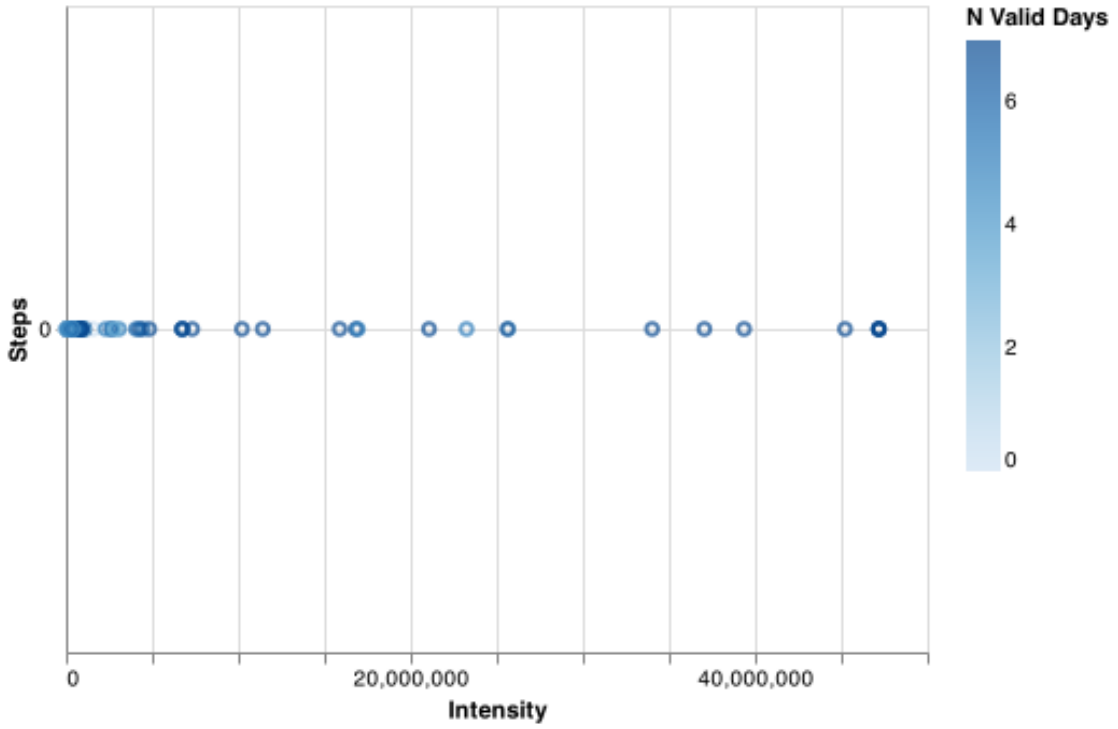
SEQN	zero_steps_with_intensity_sum	too_many_steps_sum	max_intensity_sum	out_of_calibration_sum	out_of_calibration_last	unreliable_sum
21013	0	0	0	10080	1	0
21016	0	0	0	10080	1	0
21024	0	0	0	10080	1	0
21105	0	0	0	10080	1	0
21184	0	0	0	10080	1	0
21185	0	0	0	10080	1	0
21186	0	0	0	10080	1	0
21243	0	0	0	10080	1	0
21254	0	0	0	10080	1	0
21260	0	0	0	10080	1	0
21289	0	0	0	10080	1	0
21310	0	0	0	10080	1	10080
21316	0	0	858	10080	1	10080
21347	0	0	0			
21350	0	0	0			
21377	0	0	0			
21384	0	0	0			
21418	0	0	0			
21499	0	0	0			
21514	0	0	0			

```
21347                  10080                        1           10080
21350                  10080                        1               0
21377                  10080                        1               0
21384                  10080                        1               0
21418                  10080                        1               0
21499                  10080                        1               0
21514                  10080                        1               0

       unreliable_last  steps_filtered_500_sum  steps_filtered_300_sum  \
SEQN
21013                0                     0.0                     0.0
21016                0                     0.0                     0.0
21024                0                     0.0                     0.0
21105                0                     0.0                     0.0
21184                0                     0.0                     0.0
21185                0                     0.0                     0.0
21186                0                     0.0                     0.0
21243                0                     0.0                     0.0
21254                0                     0.0                     0.0
21260                0                     0.0                     0.0
21289                0                     0.0                     0.0
21310                1                     0.0                     0.0
21316                1                     0.0                     0.0
21347                1                     0.0                     0.0
21350                0                     0.0                     0.0
21377                0                     0.0                     0.0
21384                0                     0.0                     0.0
21418                0                     0.0                     0.0
21499                0                     0.0                     0.0
21514                0                     0.0                     0.0

       valid_day  PAXINTEN_sum
SEQN
21013          5  1.733041e+05
21016          5  2.343383e+05
21024          0  1.503029e+05
21105          1  4.047857e+04
21184          2  9.412914e+04
21185          7  1.304413e+05
21186          4  2.472670e+05
21243          4  3.139427e+05
21254          0  1.008900e+04
21260          0  9.681071e+04
21289          5  6.196894e+05
21310          7  1.141791e+07
21316          7  3.402625e+07
21347          7  6.753812e+06
```

```
21350          2  1.269551e+05
21377          7  3.490004e+05
21384          7  6.331371e+04
21418          6  2.950876e+05
21499          2  2.338179e+05
21514          7  2.610783e+05
```

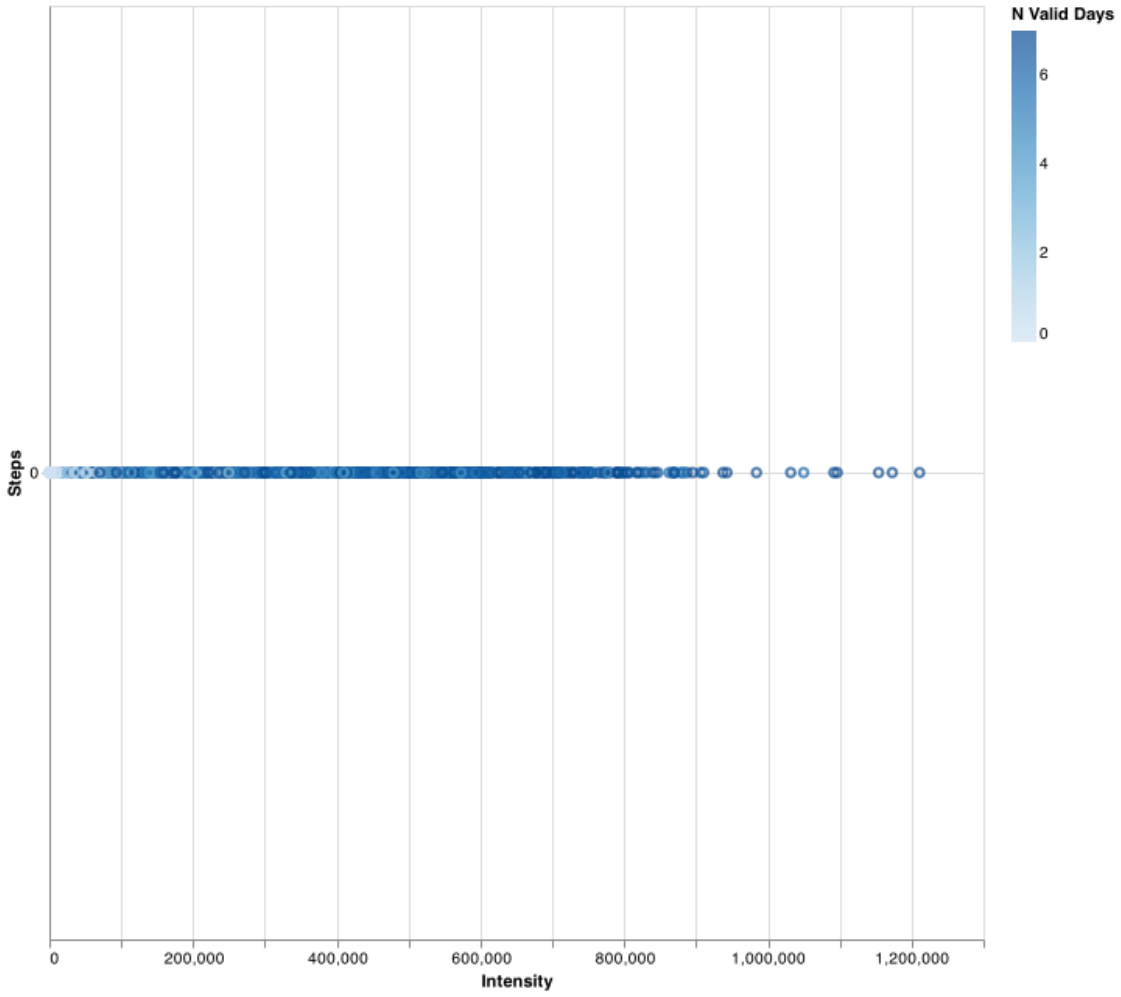
[188]:
```
alt.Chart(d_unreliable).mark_point().encode(
    alt.X("PAXINTEN_sum", title="Intensity"),
    alt.Y("steps_filtered_500_sum", title="Steps"),
    alt.Color("valid_day", title="N Valid Days"),
)
```

[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()
```

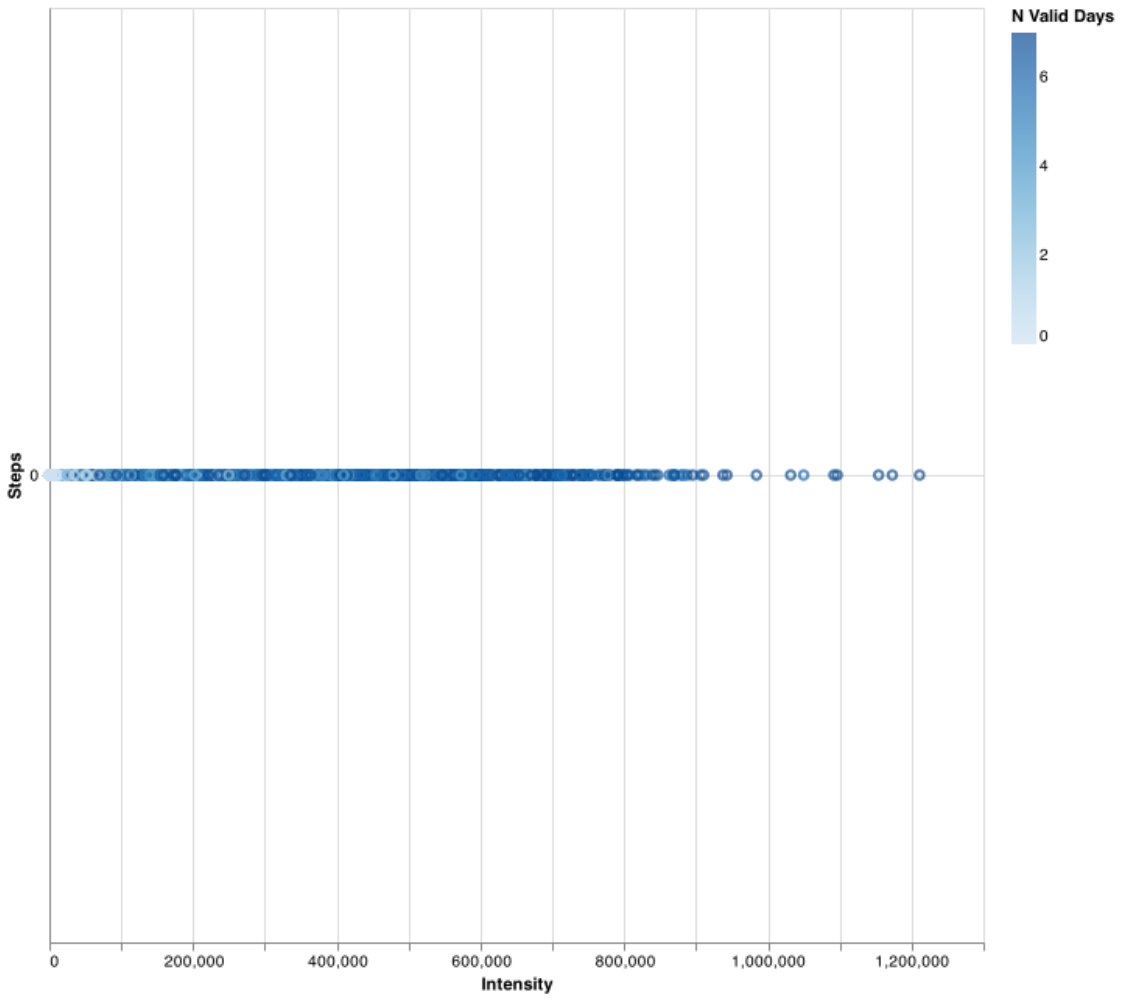
```
/Users/mm51929/.pyenv/versions/3.10.4/lib/python3.10/site-
packages/statsmodels/regression/linear_model.py:1754: RuntimeWarning: invalid
value encountered in double_scalars
  return 1 - self.ssr/self.uncentered_tss
/Users/mm51929/.pyenv/versions/3.10.4/lib/python3.10/site-
packages/statsmodels/regression/linear_model.py:1841: RuntimeWarning: invalid
value encountered in double_scalars
  return self.mse_model/self.mse_resid
/Users/mm51929/.pyenv/versions/3.10.4/lib/python3.10/site-
packages/statsmodels/regression/linear_model.py:940: RuntimeWarning: divide by
```

```
zero encountered in log
  llf = -nobs2*np.log(2*np.pi) - nobs2*np.log(ssr / nobs) - nobs2
/Users/mm51929/.pyenv/versions/3.10.4/lib/python3.10/site-
packages/statsmodels/stats/stattools.py:50: RuntimeWarning: invalid value
encountered in double_scalars
  dw = np.sum(diff_resids**2, axis=axis) / np.sum(resids**2, axis=axis)
```

[190]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                                OLS Regression Results
==============================================================================
===========
Dep. Variable:     steps_filtered_500_sum   R-squared (uncentered):
nan
Model:                                OLS   Adj. R-squared (uncentered):
nan
Method:                     Least Squares   F-statistic:
nan
Date:                    Thu, 23 Feb 2023   Prob (F-statistic):
nan
Time:                            14:14:00   Log-Likelihood:
inf
No. Observations:                   6807   AIC:
-inf
Df Residuals:                       6806   BIC:
-inf
Df Model:                              1
Covariance Type:                nonrobust
==============================================================================
                    coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
PAXINTEN_sum          0          0        nan        nan           0           0
==============================================================================
Omnibus:                          nan   Durbin-Watson:                     nan
Prob(Omnibus):                    nan   Jarque-Bera (JB):                  nan
Skew:                             nan   Prob(JB):                          nan
Kurtosis:                         nan   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()
```

```
/Users/mm51929/.pyenv/versions/3.10.4/lib/python3.10/site-
packages/statsmodels/regression/linear_model.py:1754: RuntimeWarning: invalid
value encountered in double_scalars
  return 1 - self.ssr/self.uncentered_tss
/Users/mm51929/.pyenv/versions/3.10.4/lib/python3.10/site-
packages/statsmodels/regression/linear_model.py:1841: RuntimeWarning: invalid
value encountered in double_scalars
```

```
  return self.mse_model/self.mse_resid
/Users/mm51929/.pyenv/versions/3.10.4/lib/python3.10/site-
packages/statsmodels/regression/linear_model.py:940: RuntimeWarning: divide by
zero encountered in log
  llf = -nobs2*np.log(2*np.pi) - nobs2*np.log(ssr / nobs) - nobs2
/Users/mm51929/.pyenv/versions/3.10.4/lib/python3.10/site-
packages/statsmodels/stats/stattools.py:50: RuntimeWarning: invalid value
encountered in double_scalars
  dw = np.sum(diff_resids**2, axis=axis) / np.sum(resids**2, axis=axis)
```

[192]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                                OLS Regression Results
==============================================================================
===========
Dep. Variable:     steps_filtered_300_sum   R-squared (uncentered):
nan
Model:                                OLS   Adj. R-squared (uncentered):
nan
Method:                     Least Squares   F-statistic:
nan
Date:                    Thu, 23 Feb 2023   Prob (F-statistic):
nan
Time:                            14:14:02   Log-Likelihood:
inf
No. Observations:                   6807   AIC:
-inf
Df Residuals:                       6806   BIC:
-inf
Df Model:                              1
Covariance Type:                nonrobust
==============================================================================
                    coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
PAXINTEN_sum          0          0        nan        nan           0           0
==============================================================================
Omnibus:                          nan   Durbin-Watson:                     nan
Prob(Omnibus):                    nan   Jarque-Bera (JB):                  nan
Skew:                             nan   Prob(JB):                          nan
Kurtosis:                         nan   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]: