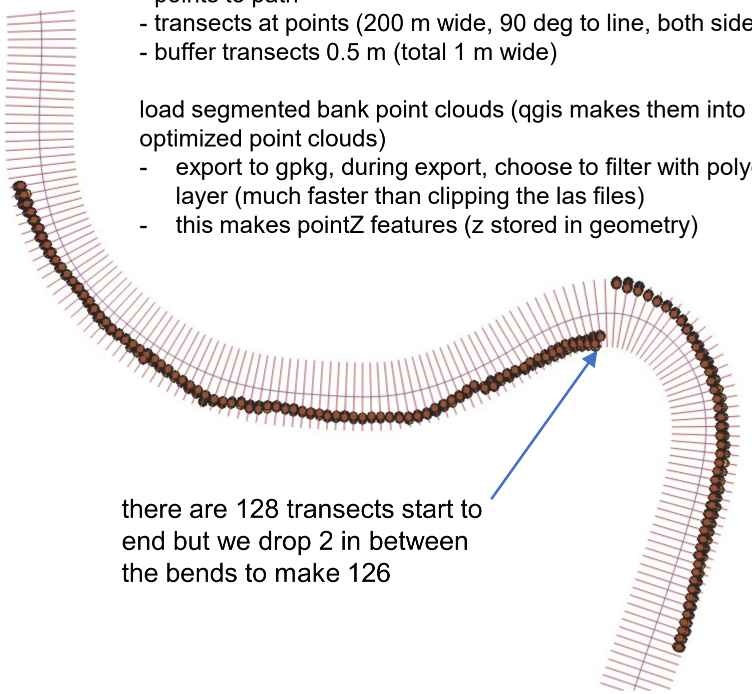
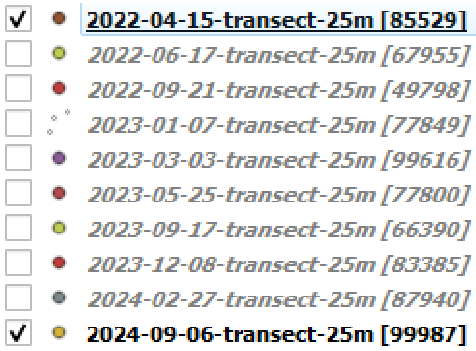
adjusted workflow to clip point clouds to transect polys:
- points along line (25 m spacing)
- points to path
- transects at points (200 m wide, 90 deg to line, both sides)
- buffer transects 0.5 m (total 1 m wide)

load segmented bank point clouds (qgis makes them into laz cloud optimized point clouds)

- export to gpkg, during export, choose to filter with polygon buffer layer (much faster than clipping the las files)
- this makes pointZ features (z stored in geometry)

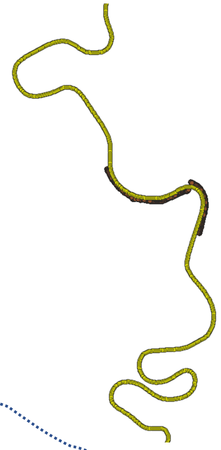


there are 128 transects start to end but we drop 2 in between the bends to make 126



☑ ● 2022-04-15-transect-25m [85529]
☐ ● 2022-06-17-transect-25m [67955]
☐ ● 2022-09-21-transect-25m [49798]
☐ 2023-01-07-transect-25m [77849]
☐ ● 2023-03-03-transect-25m [99616]
☐ ● 2023-05-25-transect-25m [77800]
☐ ● 2023-09-17-transect-25m [66390]
☐ ● 2023-12-08-transect-25m [83385]
☐ ● 2024-02-27-transect-25m [87940]
☑ ● 2024-09-06-transect-25m [99987]

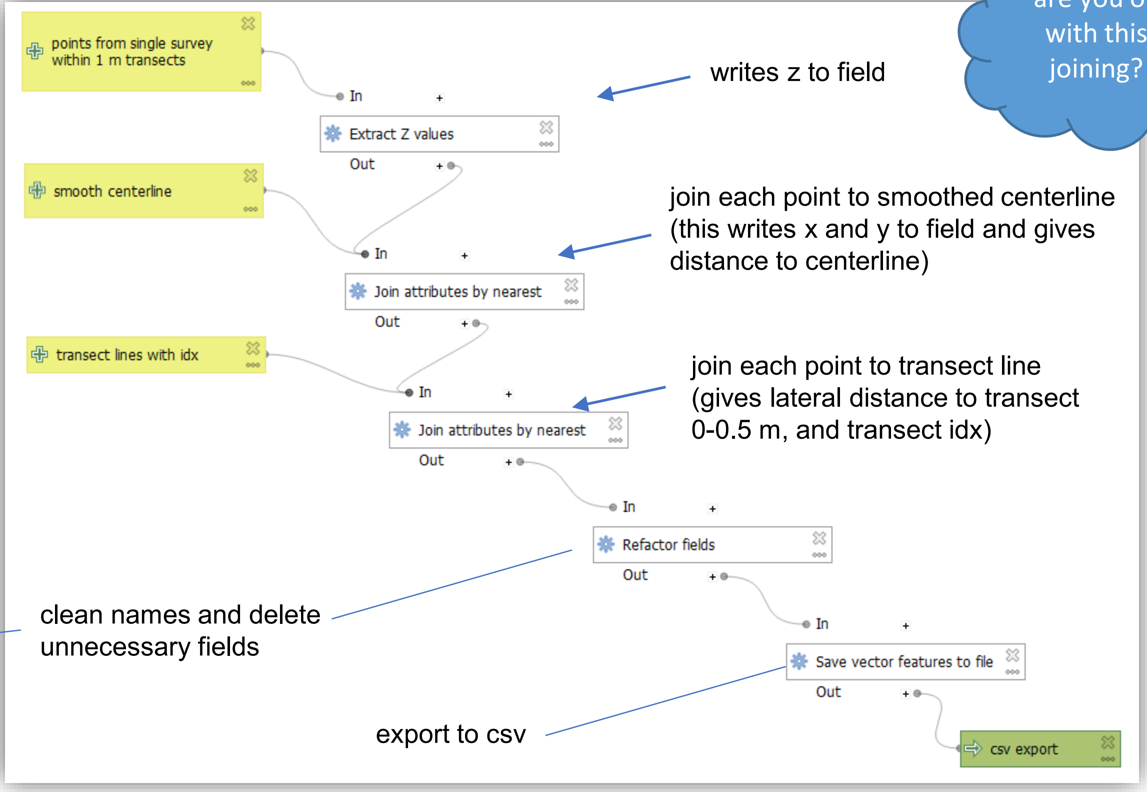
this is number of points



exported the centerline coords at 25 m interval from this larger area, the ones in the bends I wrote a value in the bend # column

and now the fun!

qgis model builder to hack automate the joining process


points from single survey within 1 m transects
In
Extract Z values
Out
smooth centerline
In
Join attributes by nearest
Out
transect lines with idx
In
Join attributes by nearest
Out
In
Refactor fields
Out
In
Save vector features to file
Out
csv export

are you ok with this joining?

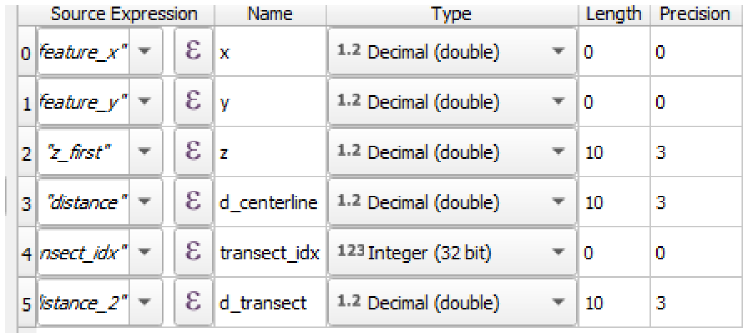
writes z to field

join each point to smoothed centerline (this writes x and y to field and gives distance to centerline)

join each point to transect line (gives lateral distance to transect 0-0.5 m, and transect idx)

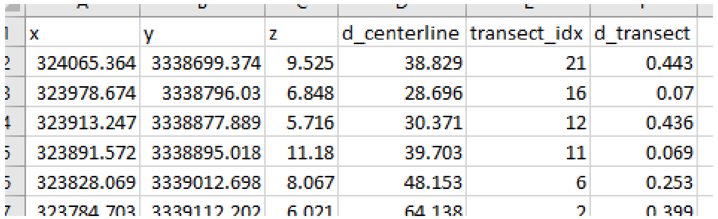
clean names and delete unnecessary fields

export to csv



	Source Expression		Name	Type	Length	Precision
0	feature_x"	ε	x	1.2 Decimal (double)	0	0
1	feature_y"	ε	y	1.2 Decimal (double)	0	0
2	"z_first"	ε	z	1.2 Decimal (double)	10	3
3	"distance"	ε	d_centerline	1.2 Decimal (double)	10	3
4	nsect_idx"	ε	transect_idx	123 Integer (32 bit)	0	0
5	istance_2"	ε	d_transect	1.2 Decimal (double)	10	3

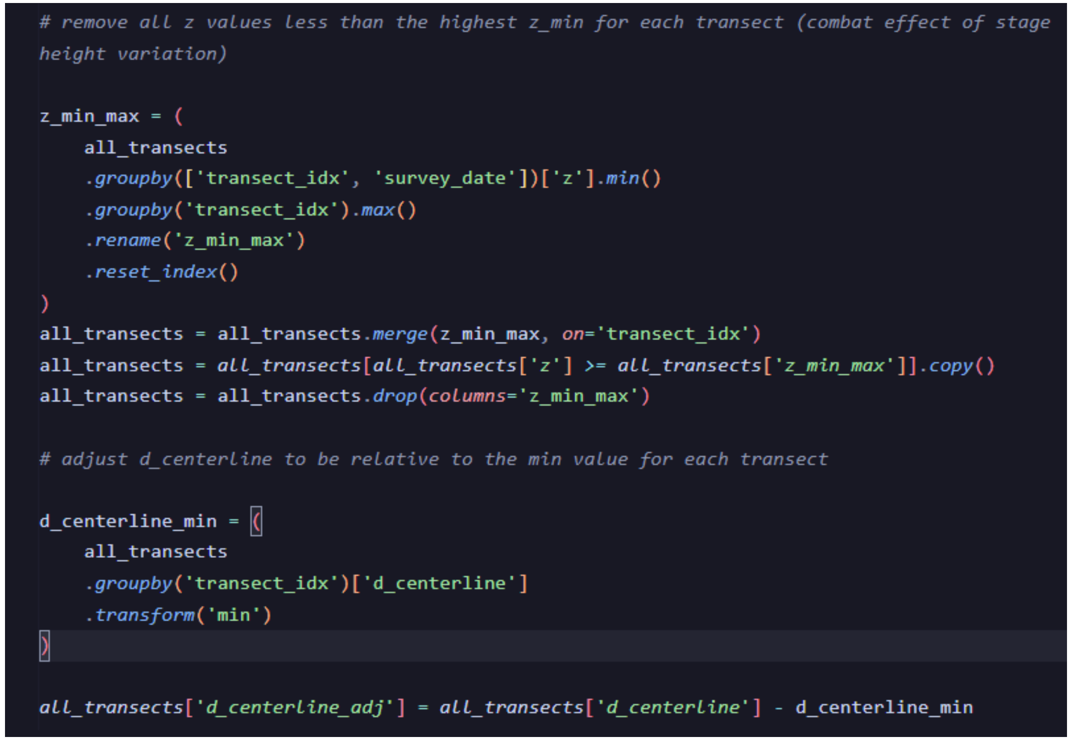
then ran as batch process for each of the above point clouds this is what the outputs look like



	x	y	z	d_centerline	transect_idx	d_transect
1	x	y	z	d_centerline	transect_idx	d_transect
2	324065.364	3338699.374	9.525	38.829	21	0.443
3	323978.674	3338796.03	6.848	28.696	16	0.07
4	323913.247	3338877.889	5.716	30.371	12	0.436
5	323891.572	3338895.018	11.18	39.703	11	0.069
6	323828.069	3339012.698	8.067	48.153	6	0.253
7	323784.703	3339112.202	6.021	64.138	2	0.399

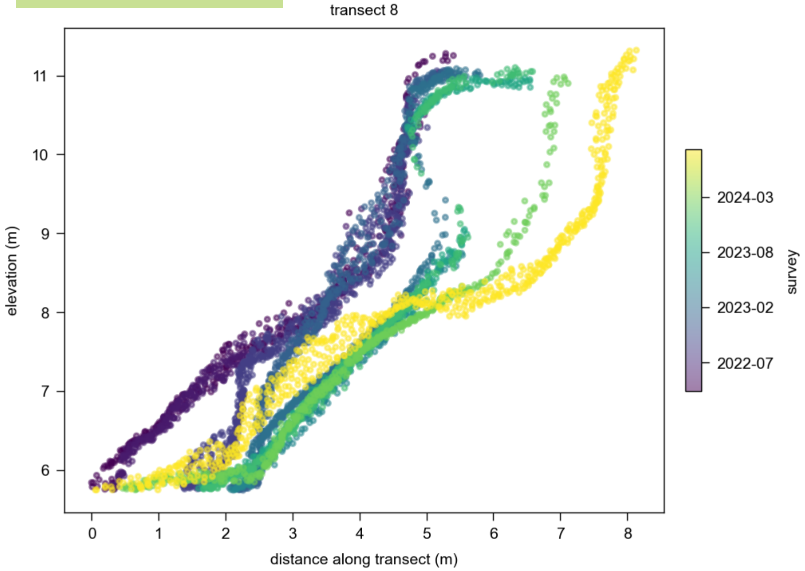
now to python:

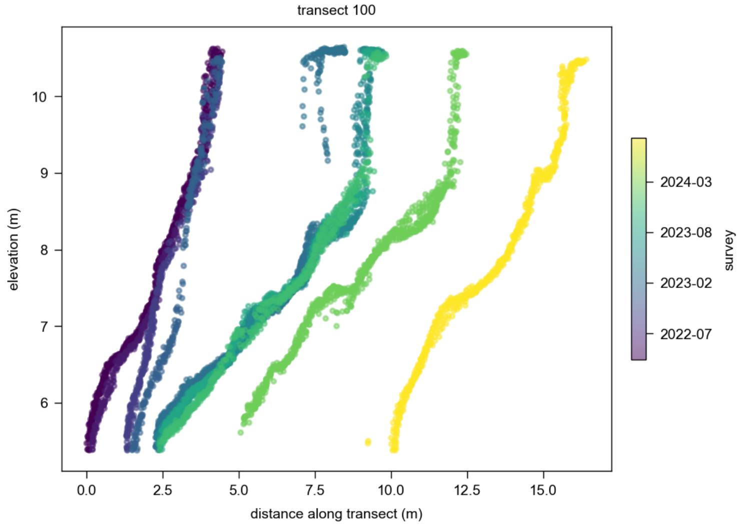
- import csvs and merge to one dataframe (array) with new survey_date column
- import centerline points
- merge df with centerline to write bend values to each point
- compute min z value per transect per survey, get maximum z_min per transect
- get minimum overall d_centerline per transect regardless of survey, subtract from all other d_centerline at that survey

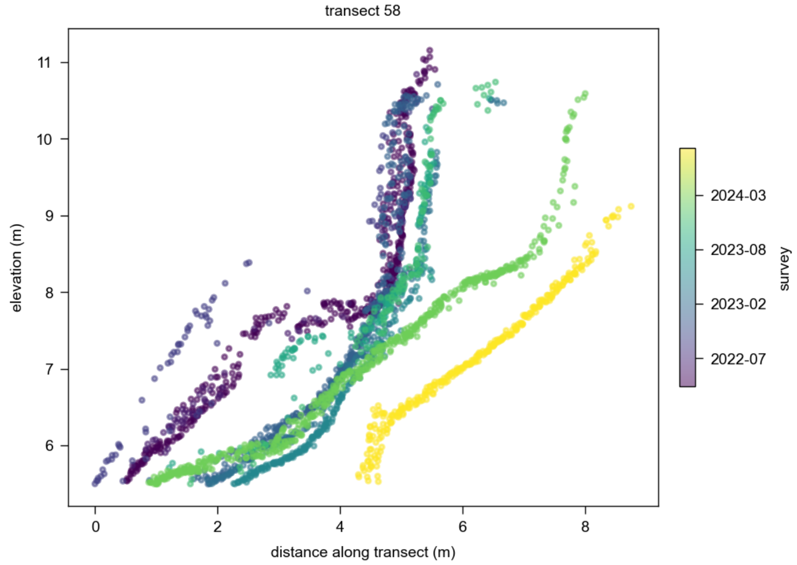


```python
# remove all z values less than the highest z_min for each transect (combat effect of stage
height variation)

z_min_max = (
    all_transects
    .groupby(['transect_idx', 'survey_date'])['z'].min()
    .groupby('transect_idx').max()
    .rename('z_min_max')
    .reset_index()
)
all_transects = all_transects.merge(z_min_max, on='transect_idx')
all_transects = all_transects[all_transects['z'] >= all_transects['z_min_max']].copy()
all_transects = all_transects.drop(columns='z_min_max')

# adjust d_centerline to be relative to the min value for each transect

d_centerline_min = (
    all_transects
    .groupby('transect_idx')['d_centerline']
    .transform('min')
)

all_transects['d_centerline_adj'] = all_transects['d_centerline'] - d_centerline_min
```

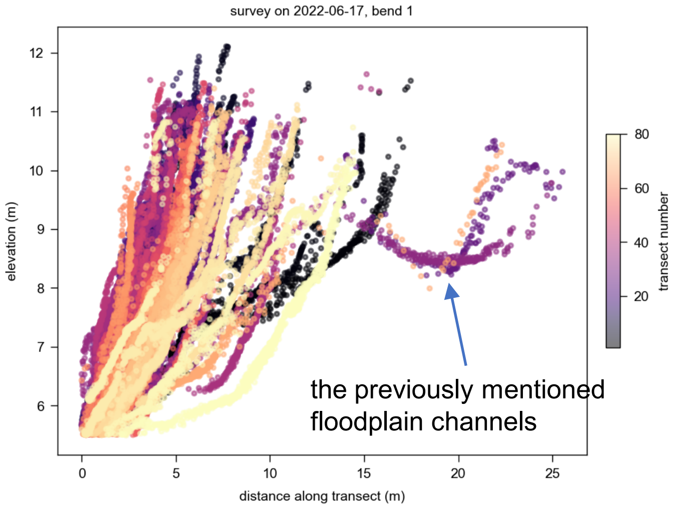
plot by transect!



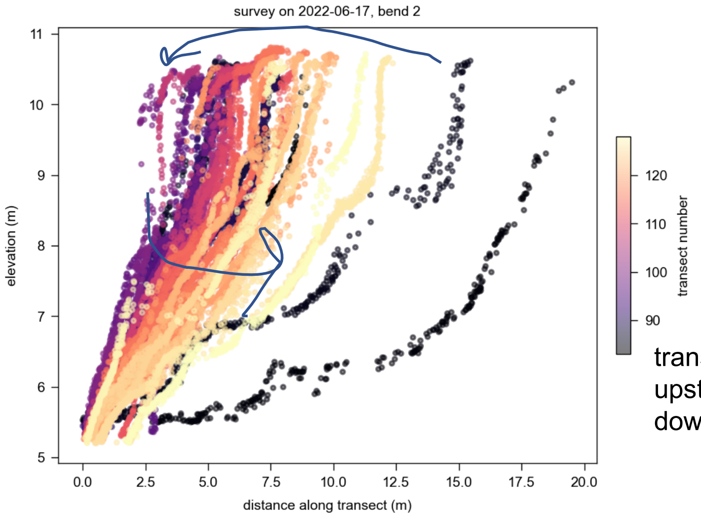




plot by survey and bend!



the previously mentioned floodplain channels



I can convince myself that there is a cool pattern, from bend beginning to end,

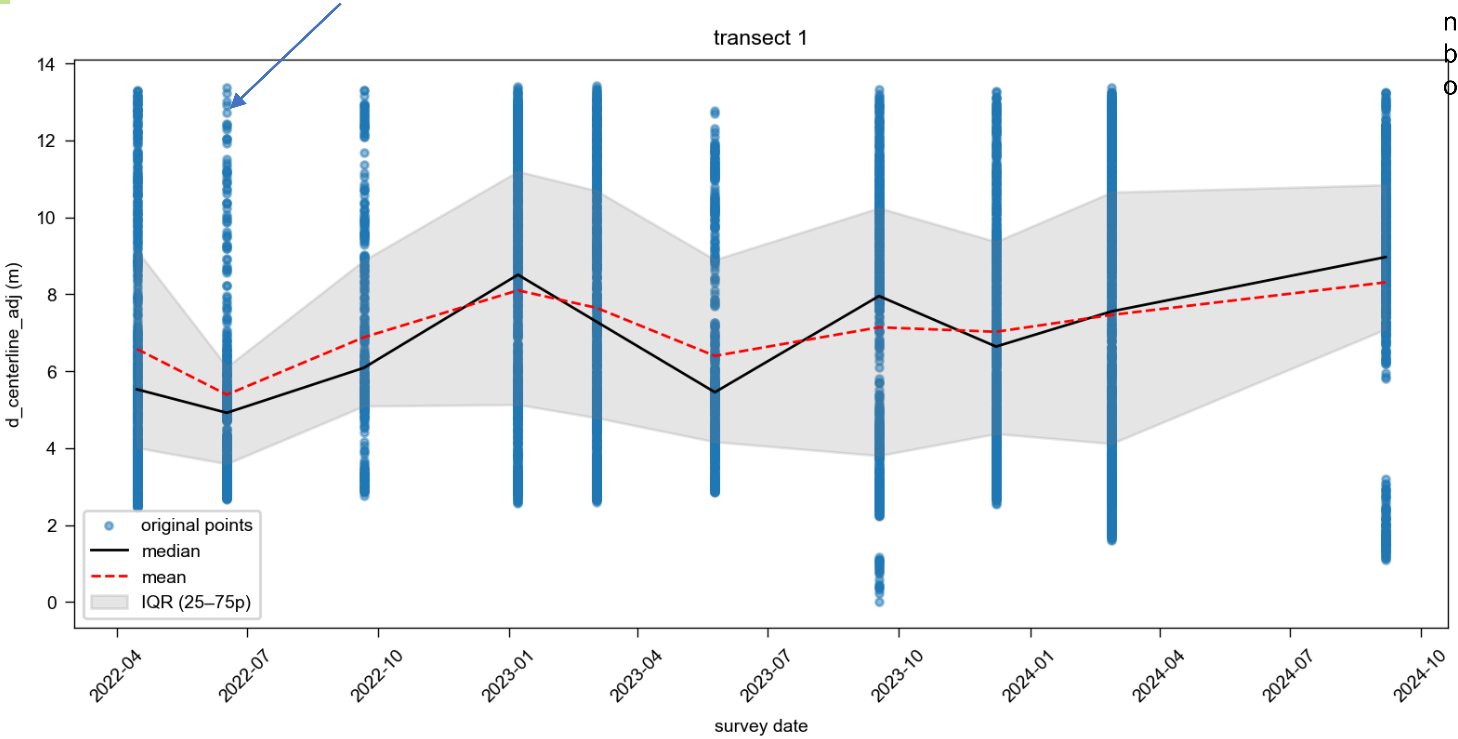
flatter > steeper > flatter

transects numbered upstream to downstream

challenges with erosion calc

this is survey vs aggregated values of distance away from centerline

low point density at bank edge, low distance to centerline

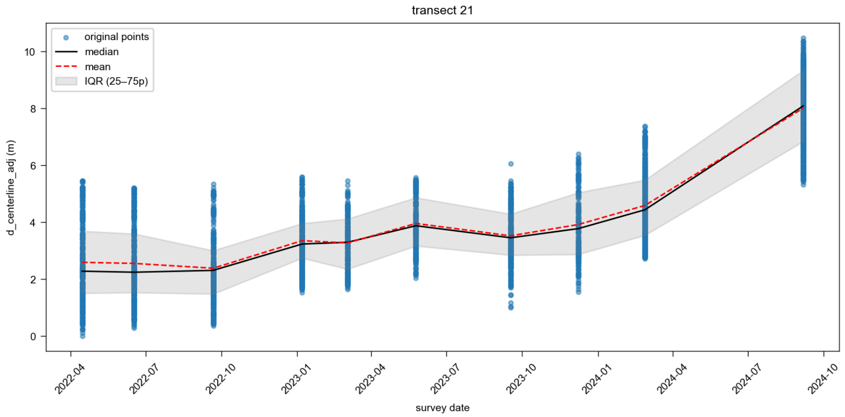


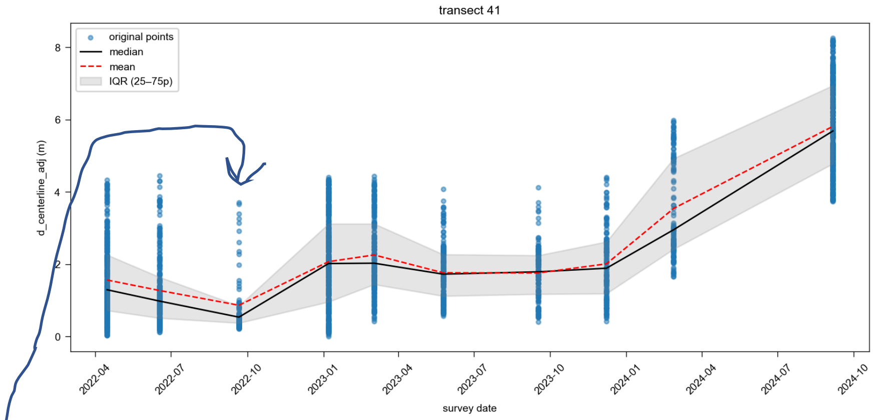
transect 1 basically does not move, but the mean bounces around because of point density obviously signal to noise is higher in ones with more erosion

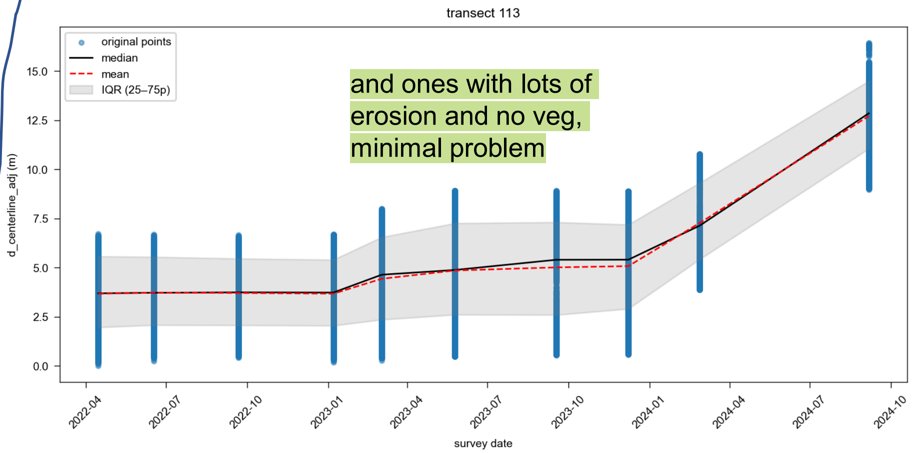
we should probably subsample?

my thoughts

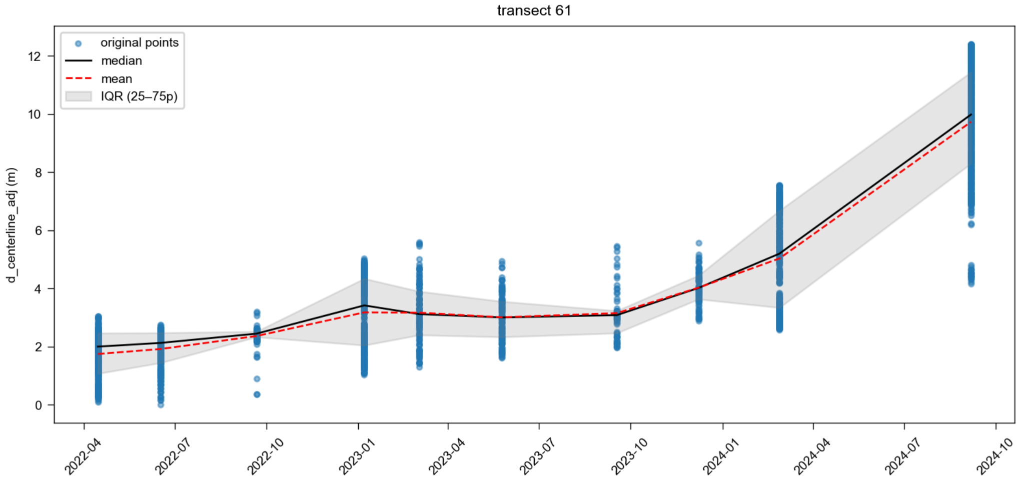
- some type of bin calc to aggregate z along d_centerline_adj
- ideally points are evenly spaced along d_centerline_adj
- then use some 1d interpolation method
- docs.scipy.org/doc/scipy/tutorial/interpolate/1D.html#tutorial-interpolate-1dsection
- then take the mean/median of the interpolated vals
- thought about just taking the midpoint of the range but sometimes there are no points on the bank edge due to veg, so range inconsistent between surveys

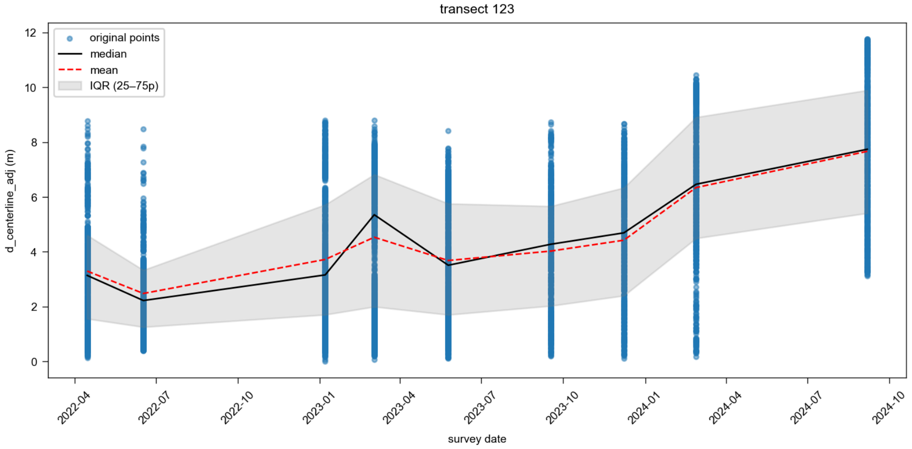


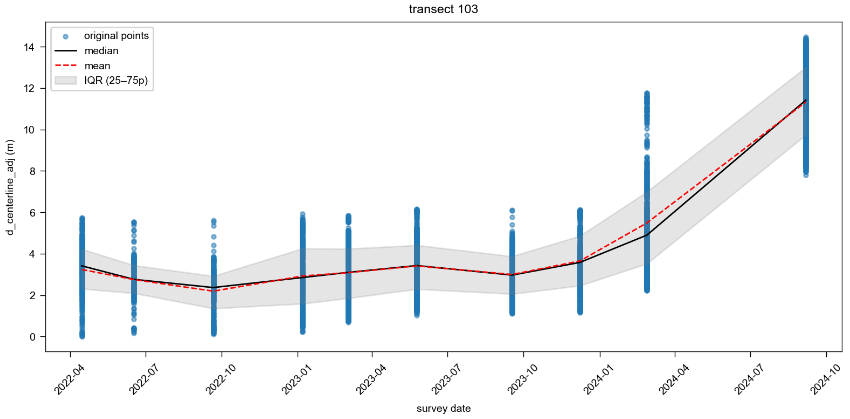




and ones with lots of erosion and no veg, minimal problem


transect 61
original points
median
mean
IQR (25–75p)
d_centerline_adj (m)


transect 123
original points
median
mean
IQR (25–75p)
d_centerline_adj (m)
survey date


transect 103
original points
median
mean
IQR (25–75p)
d_centerline_adj (m)
survey date