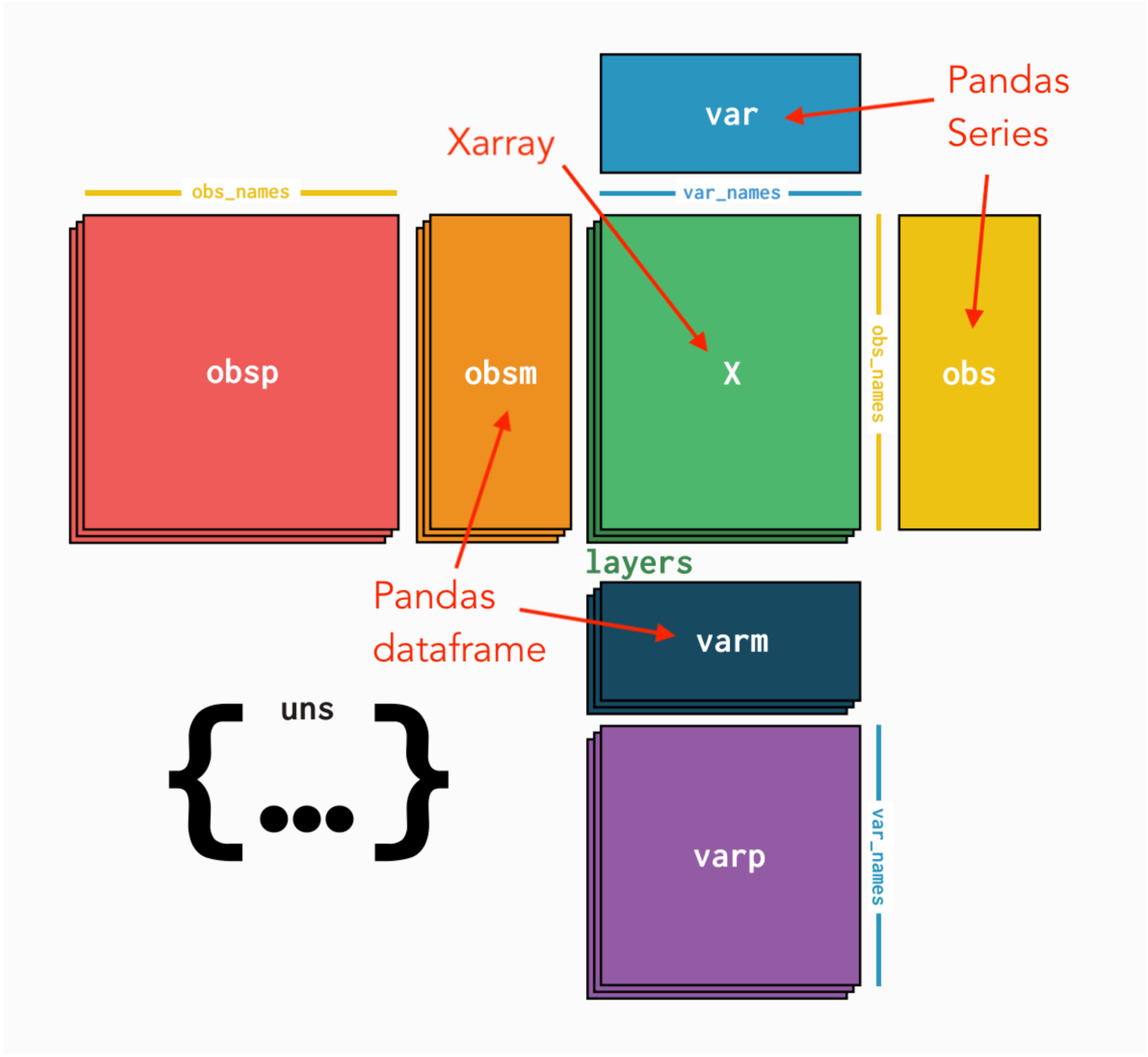
Update on AnnData Extension

Qi An, 2022.12.22



X: data points
Variable: genes, cpgs…
Observations: cells, samples, …

Task:
- Add support for genomic ranges based operations
  - Coordinates: (chromosome, start, end, *args)
  - Slicing with a given coordinate
  - Subset by overlap with a list of coordinates
  - Groupby and aggregation

PyRanges

https://github.com/biocore-ntnu/pyranges

- Nested containment list
  - List of data frames
  - For each chromosome and strand
- Provide internal support for overlapping, intersection, etc.

```
pd.DataFrame(exons)
```

✓ 0.4s

	0	1
0	(chrX, +)	Chromosome Start End ...
1	(chrX, -)	Chromosome Start End ...
2	(chrY, +)	Chromosome Start End ...
3	(chrY, -)	Chromosome Start End ...

```
exons['chrX']['+']
```

✓ 0.3s Python

	Chromosome	Start	End	Name	Score	Strand
0	chrX	135721701	135721963	NR_038462_exon_0_0_chrX_135721702_f	0	+
1	chrX	135574120	135574598	NM_001727_exon_2_0_chrX_135574121_f	0	+
2	chrX	47868945	47869126	NM_205856_exon_4_0_chrX_47868946_f	0	+
3	chrX	77294333	77294480	NM_000052_exon_17_0_chrX_77294334_f	0	+
4	chrX	91090459	91091043	NM_001168360_exon_0_0_chrX_91090460_f	0	+
...	...	...	...	...	...	...
428	chrX	117749562	117749674	NM_144658_exon_29_0_chrX_117749563_f	0	+
429	chrX	129484619	129484705	NM_001282196_exon_5_0_chrX_129484620_f	0	+
430	chrX	70607110	70607311	NR_104391_exon_14_0_chrX_70607111_f	0	+
431	chrX	13587693	13588054	NM_001167890_exon_0_0_chrX_13587694_f	0	+
432	chrX	148059891	148059985	NM_001169123_exon_17_0_chrX_148059892_f	0	+

PyRanges object is not serializable

cannot be saved into `h5ad` data type

- Solution: Convert into data frame and save in `varm` layer
- PyRanges object is re-constructed every time after loading

Immutable indexing

- PyRanges object cannot be edited/operated
- Every operation creates a new PyRanges object and order by chromosome and strand

Solution: Add redundant column matching with var_names

```
pyranges_df
```

✓ 0.2s

	index	Chromosome	Start	End	Name	Score	Strand
0	Gene_0	chrX	51453924	51455226	NR_033773_exon_0_0_chrX_51453925_f	0	+
1	Gene_1	chrX	146363460	146363548	NR_030240_exon_0_0_chrX_146363461_r	0	-
2	Gene_2	chrX	105880989	105881024	NM_001184782_exon_7_0_chrX_105880990_f	0	+
3	Gene_3	chrX	115585489	115585608	NM_007231_exon_9_0_chrX_115585490_f	0	+
4	Gene_4	chrX	30907230	30907511	NM_152787_exon_10_0_chrX_30907231_r	0	-
...	...	...	...	...	...	...	...
995	Gene_995	chrX	137713733	137715147	NM_001139498_exon_0_0_chrX_137713734_r	0	-
996	Gene_996	chrY	1363220	1363354	NM_172246_exon_7_0_chrY_1363221_f	0	+
997	Gene_997	chrX	15657684	15657879	NM_020665_exon_1_0_chrX_15657685_r	0	-
998	Gene_998	chrX	129505537	129505618	NM_001282197_exon_10_0_chrX_129505538_f	0	+
999	Gene_999	chrX	46531985	46532062	NM_001257291_exon_13_0_chrX_46531986_r	0	-

1000 rows × 7 columns

```
pr.PyRanges(pyranges_df)
```

✓ 0.8s

	index	Chromosome	Start	End	Name	Score	Strand
0	Gene_0	chrX	51453924	51455226	NR_033773_exon_0_0_chrX_51453925_f	0	+
1	Gene_2	chrX	105880989	105881024	NM_001184782_exon_7_0_chrX_105880990_f	0	+
2	Gene_3	chrX	115585489	115585608	NM_007231_exon_9_0_chrX_115585490_f	0	+
3	Gene_5	chrX	110463585	110464173	NM_001128173_exon_14_0_chrX_110463586_f	0	+
4	Gene_7	chrX	49315926	49315999	NM_001127345_exon_0_0_chrX_49315927_f	0	+
...	...	...	...	...	...	...	...
995	Gene_973	chrY	15409586	15409728	NR_047626_exon_3_0_chrY_15409587_r	0	-
996	Gene_977	chrY	15522872	15522993	NM_001258270_exon_22_0_chrY_15522873_r	0	-
997	Gene_984	chrY	15526614	15526673	NR_047609_exon_23_0_chrY_15526615_r	0	-
998	Gene_986	chrY	15417278	15417427	NR_047625_exon_6_0_chrY_15417279_r	0	-
999	Gene_989	chrY	15467802	15467898	NM_001258269_exon_14_0_chrY_15467803_r	0	-

1000 rows × 7 columns

Wrapper around AnnData object: RangeAnnData

```python
class RangeAnnData(AnnData):
    def set_coord(self, prange):
        self.varm['coord'] = prange.df.set_index(adata.var_names)

    def subset_by_overlap(self, prange):
        coord = pr.PyRanges(self.varm['coord'].reset_index())
        idx = coord.overlap(prange).index
        return self[:, idx]

    def slice_pyrange(self, chrom, start, end):
        prange = pr.PyRanges(chromosomes=chrom, starts=[start], ends=[end])
        return subset_by_overlap(self, prange)
```

✓ 0.2s

Set Coordinates

Result: object `coord` inside varm layer

```
adata.set_coord(exons)
```
✓ 0.2s

```
adata
```
✓ 0.2s

```
AnnData object with n_obs × n_vars = 100 × 1000
    varm: 'coord'
```

Slicing

Input: RangeAnnData object, chromosome, start, end
Output: Subset view of original object
Shallow copy

```
slice_adata = adata.slice_pyrange('chrX', 1000000, 10000000)
slice_adata.X
```
✓ 0.3s

```
<100x32 sparse matrix of type '<class 'numpy.float32'>'
        with 2020 stored elements in Compressed Sparse Row format>
```

Subset by overlapping with another pyranges object

Input: RangeAnnData object, pyranges object
Output: Subset view of original object

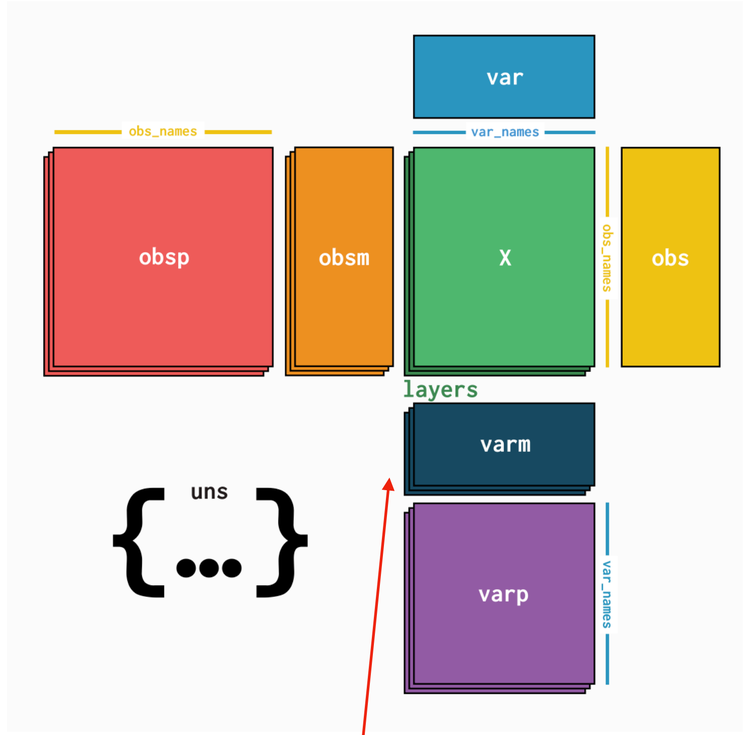
Next step: Add option for partial overlap / containment

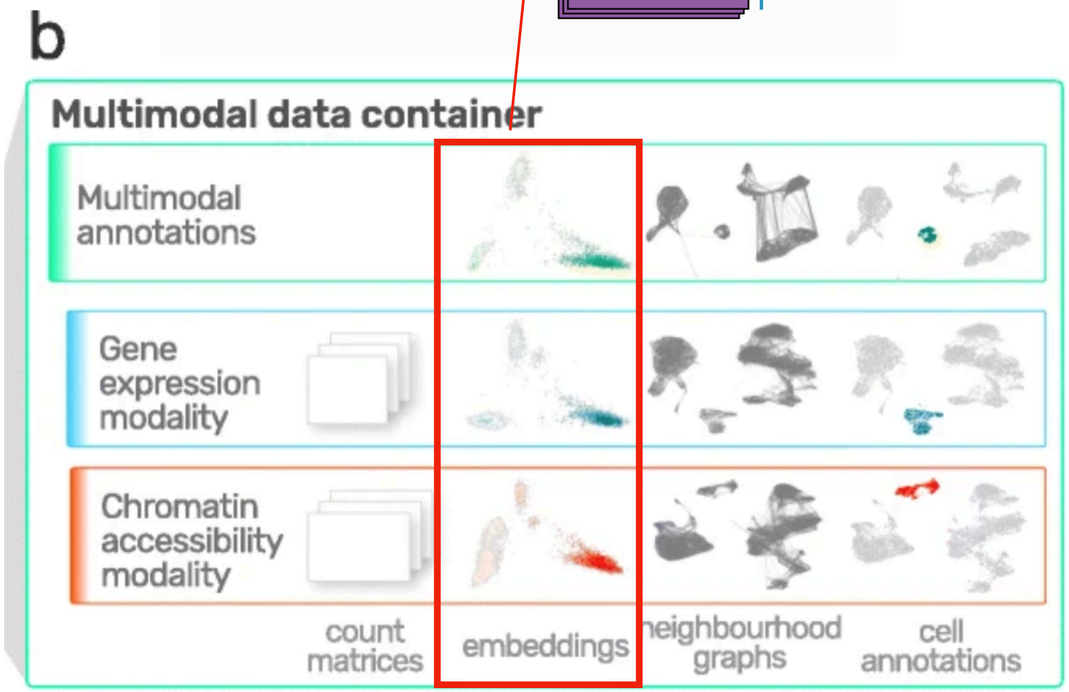
```
subset_adata = adata.subset_by_overlap(gr)
subset_adata.X
```
✓ 0.5s

```
<100x78 sparse matrix of type '<class 'numpy.float32'>'
        with 4899 stored elements in Compressed Sparse Row format>
```

MuData object

- A collection of multiple AnnData object
- Slice each AnnData object and Annotation individually
- Reconstruct MuData





Outlook

- Groupby and aggregation
- Example: List of ranges in TSS -> Average methylation beta value of CpGs inside each TSS region

```
foreach row in TSS PyRanges:
    slice RangedAnnData
    calculate average
```

- More efficient way?

- Memory performance
- PyRanges object is reconstructed every time
- Sort time complexity O(n)

Bioframe

```
import bioframe
bed_column_names = ("chromosome_name", "start_position", "end_position")

query_result = bioframe.select(pbmc.var, "4:0-1000000", cols=bed_column_names)
query_result.head()
```