EOS/GEOG 230

Binary File Types

** download binary_class.ipynb from Week 3 of brightspace for code and data from this class **

Binary

Not Human readable!

- can be opened in notepad++ or some other basic editor, but it presents as a bunch of garbage
  - (although… there *are* binary editors: unleash the *Hexinator*)

- In general, we really hope the fn extension is giving us the correct information
  - Very hard to check
  - But – there are binary editors!

(which reminds me… let's see how to make the fn extension visible in windows)

Binary ***.bin or others**

<u>No particular format</u>

- This is just data written as binary
- Faster than writing in byte format
  - `read(fn,mode='rb') or write(fn,mode='wb')`

- These are base python functions and are not associated with any library.
- Try a simple file of bytes. (Part 1 of class code)
  - Return to the ASCII table (next page)

- Try a group of integers as integers (Part 2)
- and as strings (Part 3)
  - (hexinator on the results)

https://www.tutorialspoint.com/How-to-write-binary-data-to-a-file-using-Python

Dec	Hx	Oct	Char		Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL	(null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH	(start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX	(start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX	(end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT	(end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ	(enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK	(acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL	(bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS	(backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB	(horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF	(NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT	(vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF	(NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR	(carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO	(shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI	(shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE	(data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1	(device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2	(device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3	(device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4	(device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK	(negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN	(synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB	(end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN	(cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM	(end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB	(substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC	(escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS	(file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS	(group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS	(record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US	(unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Binary ***. Lots!**

Back to Pandas library readers: proprietary/commercial formats

- Lots of them:

	binary	MS Excel	read_excel	to_excel
	binary	OpenDocument	read_excel	
"hierarchical data format" – not commercial	~~binary~~	~~HDF5 Format~~	~~read_hdf~~	~~to_hdf~~
"DataFrame binary serializer"	binary	Feather Format	read_feather	to_feather
	binary	Parquet Format	read_parquet	to_parquet
	binary	ORC Format	read_orc	to_orc
	binary	Stata	read_stata	to_stata
	binary	SAS	read_sas	
	binary	SPSS	read_spss	
"DataFrame storage format"	binary	Python Pickle Format	read_pickle	to_pickle

https://pandas.pydata.org/docs/user_guide/io.html

Binary ***.hdf *.hdf4 *.hdf5**

The self-describing gridded binary formats: **Hierarchical Data Format**

- We will ignore commercial formats
  - If you have data in that format, use the appropriate reader.

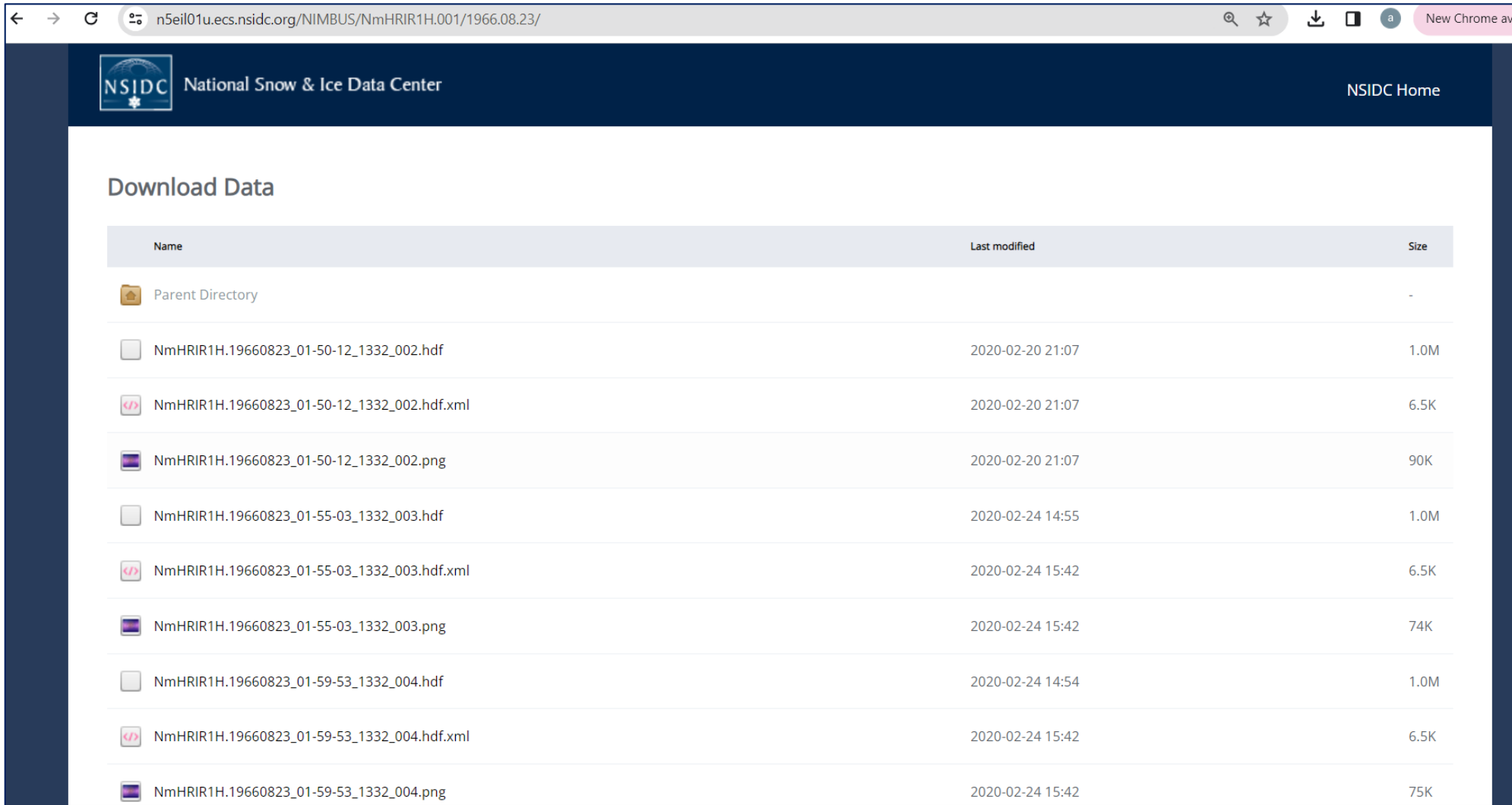
- From the NASA site: "*The Hierarchical Data Format (HDF) is a data model, file format and I/O library designed for storing, exchanging, managing and archiving complex data including scientific, engineering, and remote sensing data. The latest version of HDF, HDF5 allows users to read only the data that they need, not the whole file. Data producers can put images, tables, multidimensional arrays, etc into the same file.*"

- From wiki: "*HDF is self-describing, allowing an application to interpret the structure and contents of a file with no outside information. One HDF file can hold a mix of related objects which can be accessed as a group or as individual objects. Users can create their own grouping structures called "vgroups."* "

Binary

***.hdf *.hdf4 *.hdf5**

I went looking for some data to demonstrate hdf5 and found this at NSIDC. We'll work through reading this in.



Binary *.nc

The self-describing gridded binary formats: **Network Common Data Format**

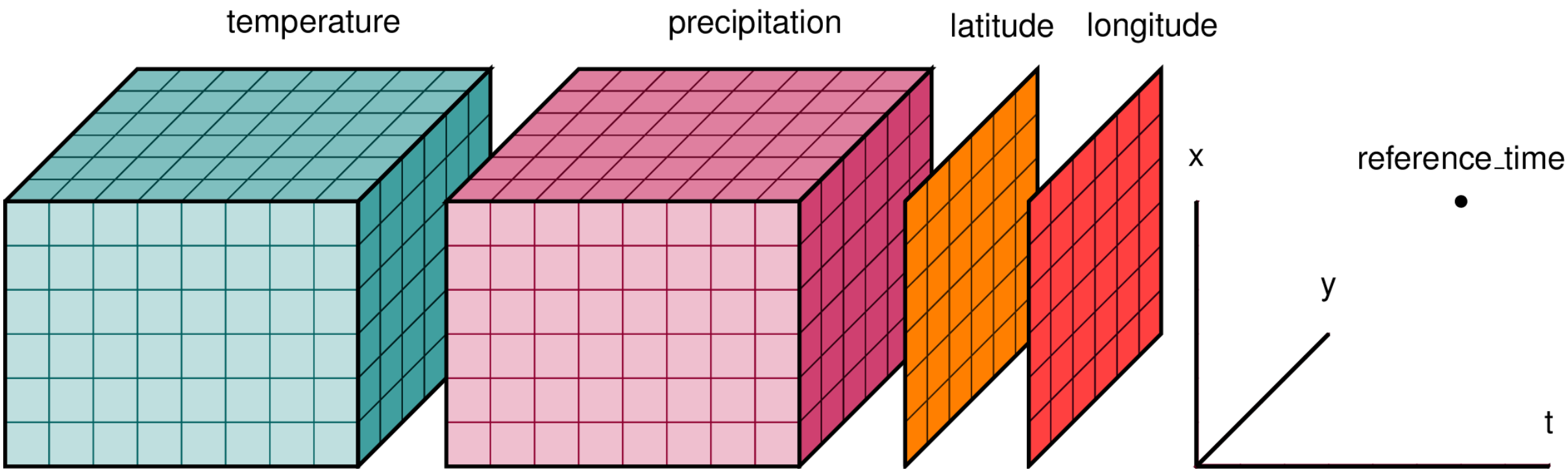
- From wiki: "*NetCDF (Network Common Data Form) is a set of software libraries and self-describing, machine-independent data formats that support the creation, access, and sharing of array-oriented scientific data. The project homepage[2] is hosted by the Unidata program at the University Corporation for Atmospheric Research (UCAR).*"
  - "*It is commonly used in climatology, meteorology and oceanography applications (e.g., weather forecasting, climate change) and GIS applications*".

- Multi-dimensional structures supported (5-d):

https://www.earthinversion.com/utilities/reading-NetCDF4-data-in-python/

Binary *.nc

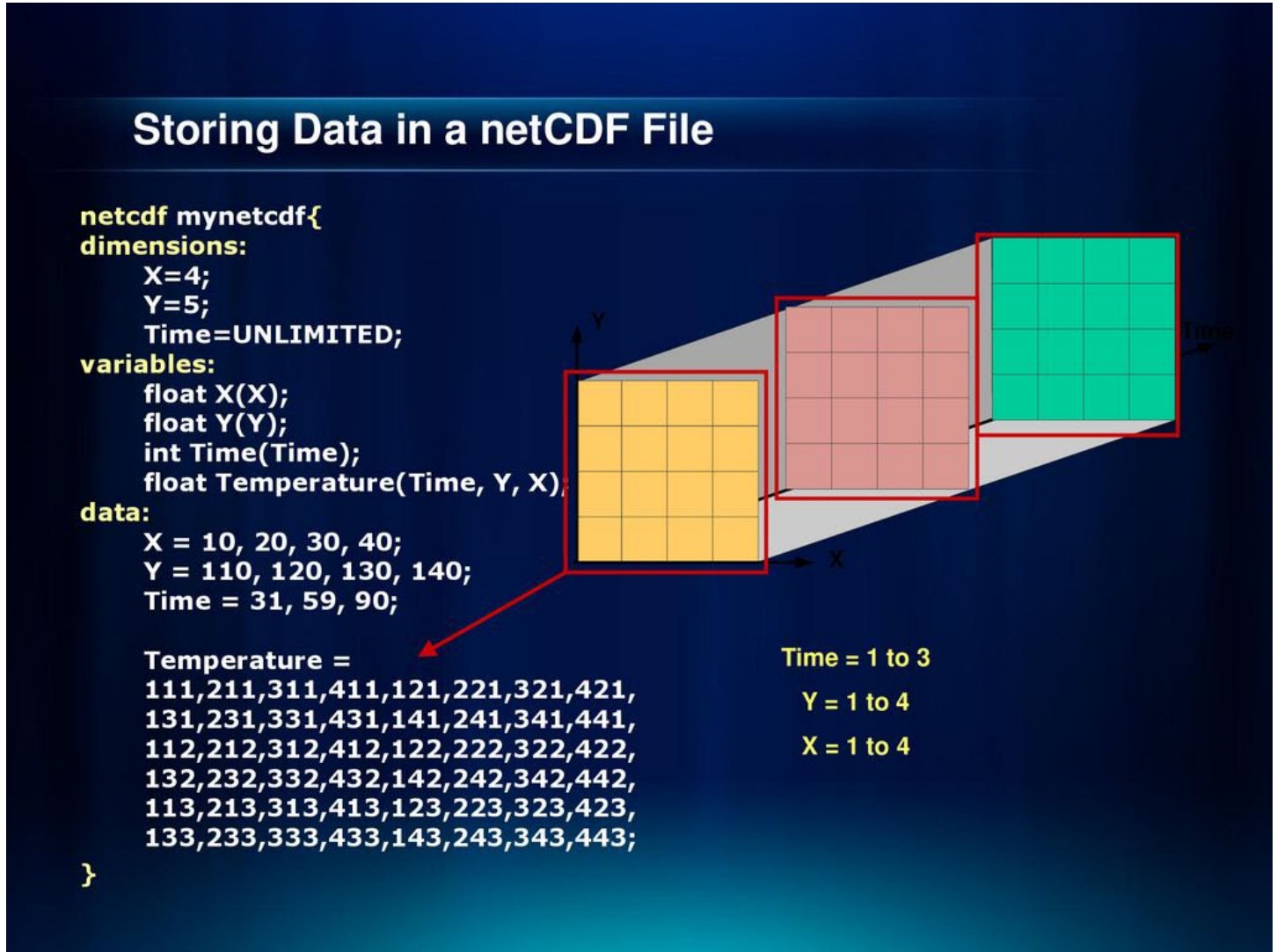
The self-describing gridded binary formats: **Network Common Data Format**



https://www.guillaumedueymes.com/courses/tutorial_python_netcdf/1-netcdf_tutorial/

Binary *.nc

The self-describing gridded binary formats: **Network Common Data Format**



Binary ***.grb *.grib2**

The self-describing gridded binary formats: **GRidded Binary**

- From wiki: "*GRIB (GRIdded Binary or General Regularly-distributed Information in Binary form[1]) is a concise data format commonly used in meteorology to store historical and forecast weather data. It is standardized by the World Meteorological Organization's Commission for Basic Systems…*"

- Similar to netCDF, but fewer dimensions

- No demo because I don't want to install the extra packages ☹