9 APPENDICES

APPENDIX I.

Species list for diversity plots sampled at GNP during 1998. Full scientific names, authorities, common names and seasonality are given (Nomenclature: Stubbendieck et al. 1986).

Grasses		
Pascopyrum smithii Rydb.	Western wheatgrass	cool
Pascopyrum dasystachyum (Hook.) Scribn.	Northern wheatgrass	cool
Bouteloua gracilis (H.B.K.) Lag. ex Steud.	Blue grama-grass	warm
Koeleria pyramidata (Lam.) Beauv.	June grass	cool
Poa pratensis L.	Kentucky blue grass	cool
Stipa comata Trin. & Rupr.	Spear grass	cool
Stipa viridula Trin.	Green needle grass	cool
Carex eleocharis †	Low sedge	cool
Forbs ††		
Achillea millefolium L.	Western yarrow	cool
Antennaria rosea Greene	Rosy Everlasting	cool
Chrysopsis villosa (Pursh) Nutt.	Hairy Golden Aster	warm
Gutierrezia sarotha (Pursh.) Britt. & Rusby	Common broomweed	warm
Liatris punctata Hook	Dotted blazingstar	warm
Phlox hoodii Richards	Moss phlox	cool
Selaginella densa Rydb.	Spikemoss	evergreen
Sphaeralcea coccinea (Pursh.) Rydb.	Scarlet mallow	warm
Taraxicum officinale Weber	Common dandelion	cool
Vicea Americana muhl.	American vetch	cool
Shrubs		
Artemisia frigida Willd.	Pasture sage	cool
Artemisia cana Pursh.	Silver sagebrush	cool
Rosa arkansana Porter	Low prairie rose	cool
Lichen		
Xanthoparmelia chlorochroa	Lichen	-

† Grasses included members of the *Cyperaceae*.
†† Forbs are herbs not in the families *Poacea* or *Cyperaceae* (functional definition after Lauenroth 1997).

APPENDIX II. Presence / absence of species found in 0.5m-

SITE	GRASSES						SEDGE	FORBS									SHRUBS			S.DENSA	LICHEN
	Warm	Cool						Warm				Cool									
	Bout. grac.	*Pasc crist.*	*Poa sandb.*	*Stipa com.*	*Pasc. smithii*	*Koel. crist.*	*Carex eleo.*	*Chrys. vill.*	*Liat. punct.*	*Guti. saroth.*	*Sphaer. cocc.*	*Phlox Hood.*	*Ant. parv.*	*Tarax. offic.*	*Vicia Americ.*	*Ach. millef.*	*Art. Frig.*	*Art. Cana*	*Rosa Arkan.*	*Sel. densa*	*Xanth. chlor.*
1	1	0	0	1	1	0	1	0	0	0	0	1	0	0	0	0	1	0	0	1	1
2	1	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1
3	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1
4	1	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	1	1	0	1	1
5	1	0	0	0	1	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1
6	0	0	0	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	1
7	0	0	0	0	1	1	1	0	0	0	1	0	0	0	0	0	1	1	0	0	1
8	0	0	0	0	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	1
9	1	0	0	0	1	0	1	0	0	0	1	0	0	0	0	0	1	1	0	1	1
10	1	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0
11	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1	1
12	1	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1
13	1	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0
14	1	0	0	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	1	1
15	1	0	0	1	1	1	1	0	0	0	1	0	0	0	0	0	1	0	0	1	0
16	1	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	1
17	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1
18	1	0	0	0	1	1	1	0	0	0	1	1	0	0	0	0	1	0	0	1	0
19	1	0	0	1	1	1	1	0	0	0	1	0	0	0	0	0	1	0	0	1	1
20	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	1	0	0	1	1
21	1	0	0	0	1	1	1	0	0	0	1	0	0	0	0	0	1	0	0	1	1
22	1	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	1	0	0	1	1
23	1	0	0	0	1	1	1	0	0	0	1	1	0	0	0	0	1	0	0	1	1
24	1	0	0	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	1	1
25	0	0	0	0	1	0	1	0	0	0	1	0	0	0	0	0	1	0	0	1	1
26	1	0	0	0	1	1	1	0	0	0	1	0	0	0	0	0	1	0	0	0	1
27	1	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	1	0	0	1	1
28	1	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	1
29	1	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	1
30	0	0	0	0	1	0	1	0	0	0	1	1	0	0	0	0	1	0	0	0	1
31	1	0	0	1	1	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0	1
32	0	0	0	0	1	0	1	0	0	0	1	1	0	0	0	0	1	0	0	0	1
33	1	0	0	1	1	0	1	0	0	0	1	1	0	0	0	0	1	1	0	1	1
34	1	0	0	1	1	0	1	0	0	0	0	1	0	0	0	0	0	1	0	1	0
35	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1
36	1	0	0	1	1	1	1	0	0	0	1	1	0	0	0	0	0	0	0	1	1
37	1	0	0	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	1	1
38	1	0	0	1	1	0	0	0	0	0	1	1	0	0	0	0	1	1	0	1	1
39	1	0	0	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	1	1
40	1	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	1	1	0	1	1
41	1	1	0	1	1	0	1	0	0	0	1	0	0	0	0	0	1	0	0	1	1
42	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1	1
43	1	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	1

44	1	0	0	0	1	0	1	0	0	0	1	0	0	0	0	0	1	0	0	1	1
45	1	0	0	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	1
46	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1	1
47	1	0	0	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	1
48	1	0	0	1	1	1	1	0	0	0	0	1	0	0	0	0	0	1	0	1	1
49	1	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1	1
50	1	0	0	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	1	1
51	1	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	1
52	1	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	1
53	1	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1	0
54	1	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	1	1	0	1	0
55	1	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1
56	1	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1
57	1	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1
58	1	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0
59	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1
60	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	1	0	0	1	0
61	1	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1
62	1	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	1	0	0	1	1
63	1	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1	1
64	1	0	0	0	1	1	1	0	0	0	1	0	0	0	0	0	0	1	0	1	1
65	1	0	0	0	1	0	1	0	0	0	1	0	0	0	0	0	1	0	0	1	1
66	1	0	0	0	1	0	1	0	0	0	1	0	0	0	0	0	1	0	0	1	0
67	1	0	0	0	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	1	1
68	0	0	0	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	1	1
69	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1
70	0	0	0	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	1
71	0	0	0	0	1	0	1	0	0	0	1	0	0	0	0	0	1	0	0	1	1
72	0	0	0	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0
73	1	0	0	1	1	0	1	1	0	0	1	0	0	0	1	1	0	0	0	1	0
74	1	0	0	1	1	0	1	0	0	0	1	0	0	1	0	0	0	0	0	1	1
75	1	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0
76	0	0	0	0	1	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0
77	1	0	0	0	1	0	1	0	0	0	1	0	0	1	0	1	0	0	0	0	1
78	1	0	0	1	1	1	1	0	0	0	1	0	0	1	0	0	0	0	0	1	0
79	1	0	0	1	1	1	1	0	0	0	1	0	0	1	0	0	1	0	1	1	1
80	1	0	0	1	1	0	1	0	0	0	1	0	1	1	0	1	0	0	0	1	0
81	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0
82	1	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0
83	0	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	1	0	0	1	0
84	1	0	0	1	1	0	1	0	0	0	1	1	0	0	0	0	0	0	0	1	0
85	1	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0
86	0	0	0	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0
87	1	0	0	1	1	1	1	0	0	0	1	0	0	0	0	0	1	0	0	1	0
88	1	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0
89	1	0	0	1	1	0	1	0	0	0	1	1	0	0	0	0	1	0	0	1	1
90	1	0	0	0	1	0	1	0	0	0	0	0	1	1	0	0	0	0	0	1	0
91	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0
92	1	0	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1	1
93	1	0	0	0	1	0	1	0	0	0	1	0	1	1	1	0	0	0	0	1	0
94	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0

95	1	0	0	1	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0
96	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0
97	1	0	0	1	1	0	1	0	0	0	0	0	1	0	1	1	0	0	0	1	0	
98	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	
99	1	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	
100	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	
101	1	0	0	1	1	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	
102	1	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	1	
103	1	0	0	1	1	1	1	0	0	0	1	0	1	0	0	1	1	0	0	1	0	
104	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	
105	1	0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	
106	1	0	0	1	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	1	0	
107	0	0	0	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	
108	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	
109	0	0	0	1	1	0	1	0	0	0	1	1	0	0	1	0	0	0	0	1	0	
110	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	
111	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0	
112	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	
113	1	0	0	1	0	0	1	0	0	0	1	1	0	1	0	0	1	0	0	1	1	
114	1	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	1	0	0	1	1	
115	1	0	0	1	1	1	1	0	0	0	1	0	0	0	0	0	1	0	0	1	1	
116	1	0	0	1	1	1	1	0	0	0	1	0	0	0	0	0	1	0	0	1	1	
117	1	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1	1	
118	1	0	0	1	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0	1	1	
119	1	0	0	1	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	1	1	
120	1	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0	
121	1	0	0	1	1	1	1	0	0	0	1	1	0	0	0	0	1	0	0	1	1	
122	1	0	0	1	1	0	1	0	0	0	0	0	0	0	1	0	1	0	0	1	1	
123	1	0	0	1	1	1	1	0	0	0	0	0	0	0	1	0	1	0	0	1	1	
124	1	0	0	1	1	0	1	0	0	0	1	0	1	0	1	0	1	0	0	1	1	
125	1	0	0	1	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	1	1	
126	1	0	0	1	0	0	1	0	0	0	0	0	0	1	1	0	1	0	0	1	1	
127	1	0	0	1	1	0	1	0	0	0	0	0	0	0	1	0	1	0	0	1	1	
128	1	0	0	1	1	0	1	0	0	0	0	1	0	0	0	0	1	0	0	1	1	
129	1	0	0	1	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	
130	1	0	0	1	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	1	
131	0	0	0	1	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1	0	
132	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	0	
133	1	0	0	1	1	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1	0	
134	1	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	
135	1	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1	0	
136	1	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	
137	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	
138	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
139	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
140	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
141	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
142	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
143	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
144	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	
145	1	0	0	1	1	1	1	0	0	0	0	1	0	0	1	0	0	0	1	1	1	

146	1	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0
147	1	0	0	1	0	0	1	0	1	0	0	1	0	1	0	0	0	0	0	0	1	0
148	1	0	0	0	1	1	1	0	0	0	0	1	0	0	0	0	0	0	1	1	0	
149	1	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	1	1	1	
150	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	
151	1	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	
152	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	
153	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	
154	1	0	0	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	
155	1	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	
156	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	
157	1	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	
158	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
159	1	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	
160	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	
161	1	0	0	1	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	1	0	
162	1	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	1	0	0	1	0	
163	1	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	1	0	0	1	0	
164	1	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	1	0	0	1	0	
165	1	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	1	0	0	1	1	
166	1	0	0	1	1	1	1	0	0	0	1	0	0	0	0	0	1	0	0	1	0	
167	1	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	1	0	0	1	0	
168	1	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	1	0	0	1	1	
169	1	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	1	0	0	1	0	
170	1	0	0	1	1	1	1	0	0	0	1	0	0	0	0	0	1	0	0	1	1	
171	1	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	1	0	0	1	0	
172	1	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0	
173	1	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	1	0	0	1	0	
174	1	0	0	1	1	1	1	0	0	0	1	0	0	0	0	0	1	0	0	1	0	
175	1	0	0	1	1	1	0	0	0	0	1	0	0	0	0	0	1	0	0	1	1	
176	1	0	0	1	1	1	1	0	0	0	1	0	0	0	0	0	1	0	0	1	1	
177	1	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	1	0	0	1	0	
178	1	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	1	0	0	1	1	
179	1	0	0	1	1	0	1	0	0	0	1	0	0	0	1	0	1	0	0	1	1	
180	1	0	0	1	1	1	1	0	0	0	1	0	0	0	0	0	1	0	0	1	0	
181	1	0	0	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0	
182	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	
183	1	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	1	0	0	1	0	
184	1	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	
185	1	0	0	1	1	1	1	0	0	0	1	0	1	0	0	0	1	0	0	1	1	
186	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	1	0	0	1	1	
187	1	0	0	1	0	1	0	0	0	0	1	0	1	0	0	0	1	0	0	1	1	
188	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	
189	1	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	1	0	0	1	1	
190	1	0	0	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	1	1	
191	1	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	1	0	0	1	1	
192	1	0	0	1	1	1	1	0	0	0	0	0	0	0	1	0	1	0	0	1	1	
193	1	0	1	1	0	0	1	0	0	0	1	0	0	0	0	0	1	0	0	1	1	
194	1	0	0	1	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	1	1	
195	1	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0	1	1	
196	1	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	1	0	0	1	1	
197	1	0	0	1	0	1	1	1	0	0	0	1	0	1	0	0	1	0	0	1	1	

198	1	0	0	1	0	1	1	1	0	0	0	0	0	1	0	0	0	0	0	1	1
199	1	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0	1	1
200	1	0	0	1	0	1	1	1	0	0	1	0	0	0	0	0	1	0	0	1	1
201	1	0	0	1	1	0	1	0	0	1	1	1	0	0	0	0	1	0	0	1	1
202	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0
203	1	0	0	1	1	1	1	0	0	0	1	0	0	0	0	0	1	0	0	1	1
204	1	0	0	1	1	0	1	0	0	0	1	1	0	0	0	0	1	0	0	1	1
205	1	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	1	0	0	1	1
206	1	0	0	1	1	0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1
207	1	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	1	0	0	1	1
208	1	0	0	1	1	1	1	0	0	0	1	0	1	0	1	0	1	0	0	1	1
209	1	0	0	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0
210	1	0	0	1	1	1	1	0	0	0	1	1	0	0	0	0	1	0	0	1	0
211	1	0	0	1	1	1	1	0	0	0	1	0	0	0	0	0	1	0	0	1	1
212	1	0	0	1	1	1	1	0	0	0	1	1	0	0	0	0	0	0	0	1	1
213	1	1	0	1	1	1	1	0	0	0	1	1	0	0	0	0	1	0	0	1	1
214	1	1	0	1	1	1	1	0	0	0	1	1	0	0	0	0	1	0	0	1	1
215	1	0	0	1	1	0	1	0	0	0	1	1	0	0	0	0	1	0	0	1	1
216	1	0	0	1	1	1	1	0	0	0	1	1	0	0	0	0	1	0	0	1	1

APPENDIX III. GRASS shell script code for Divided-differencing interpolation.

```
#! /usr/bin/ksh -f
#
#-----------------------------------------------------------------------------------------------------------------------------------
# This shell script uses the divided difference method to interpolate NDVI values at dates falling between
# bi-weekly NDVI composite values. Interpolations can be used to derive various productivity metrics.
# The desired timestep is specified by the user. The advantage of using divided differencing is that second
# and third order interpolations are based on coefficients derived for a first order (linear) interpolation.
#
# First, second and third order polynomials may be fit through data points using this method. For theory,
# see Atkinson, K (1995). Elementary Numerical Analysis Chapter 5.
#
# This is the fifth version of the script by Andrew Davidson, 2002. Implementation is in GRASS4.3, so
# NDVI data multiplied by a factor of 10 are used. File names must  follow "avhrr.doyxxx.ndvi.x10000"
# convention.
#-----------------------------------------------------------------------------------------------------------------------------------
#-----------------------------------------------------------------------------------------------------------------------------------
# Choose processing approach. Choice of linear, second or third order fits. User prompted for response.
#-----------------------------------------------------------------------------------------------------------------------------------

 echo "Job started on `date`" >> SIM.LOGFILE

 echo "Choose one of the following options:

          1. First Order (Linear) Fit.
          2. Second Order Polynomial.
          3. Third Order Polynomial.
          4. Quit

       Select option:""\n"

 read ANSWER
    if [ $ANSWER = 1 ]; then
     echo "\n"
     echo "You have chosen a First Order (Linear) Fit." | tee -a SIM.LOGFILE
     echo "\n"
   else
    if [ $ANSWER = 2 ]; then
     echo "\n"
     echo "You have chosen a Second Order Polynomial." | tee -a SIM.LOGFILE
     echo "\n"
   else
    if [ $ANSWER = 3 ]; then
     echo "\n"
     echo "You have chosen a Third Order Polynomial." | tee -a SIM.LOGFILE
     echo "\n"
   else
        echo "\n"
        echo "Exiting Program.""\n"
        exit 1
fi
  fi
fi


#-----------------------------------------------------------------------------------------------------------------------------------
# Enter the image dates on which interpolation will be based. Linear interpolation is based on line fitting
# through two points; second order interpolation is based on a polynomial fit through three points; third
# order fit is based on a polynomial fit through four points. Point files are time-series raster coverages.
#-----------------------------------------------------------------------------------------------------------------------------------
```

```
if [ $ANSWER = 1 ]; then
  echo "Choose TWO (2) image dates for use in interpolation.""\n"
  echo "Enter DOY of 1st image:""\n"
  read X0
  echo "\n"
  echo "Enter DOY of 2nd image:""\n"
  read X1
  echo "\n"
     if [ $X1 -lt X0 ]; then
       echo "Second date cannot be earlier than first date""\n"
       echo "Exiting Program.""\n"
       exit 1
     fi
  echo " You have chosen :

           First Date  [X0]:   avhrr.doy$X0.sm.x10000
           Second Date [X1]:   avhrr.doy$X1.sm.x10000" | tee -a SIM.LOGFILE
  echo "\n"
  else
if [ $ANSWER = 2 ]; then
  echo "Choose THREE (3) image dates for use in interpolation.""\n"
  echo "Enter DOY of 1st image:""\n"
  read X0
  echo "\n"
  echo "Enter DOY of 2nd image:""\n"
  read X1
  echo "\n"
  echo "Enter DOY of 3rd image:""\n"
  read X2
  echo "\n"
     if [ $X1 -lt X0 ] || [ $X2 -lt X1 ] ; then
       echo "Later dates cannot be earlier than previous dates""\n"
       echo "Exiting Program.""\n"
       exit 1
     fi
  echo " You have chosen :

           First Date  [X0]:   avhrr.doy$X0.sm.x10000
           Second Date [X1]:   avhrr.doy$X1.sm.x10000
           Third Date  [X2]:   avhrr.doy$X2.sm.x10000" | tee -a SIM.LOGFILE
  echo "\n"
  else
if [ $ANSWER = 3 ]; then
  echo "Choose FOUR (4) image dates for use in interpolation.""\n"
  echo "Enter DOY of 1st image:""\n"
  read X0
  echo "\n"
  echo "Enter DOY of 2nd image:""\n"
  read X1
  echo "\n"
  echo "Enter DOY of 3rd image:""\n"
  read X2
  echo "\n"
  echo "Enter DOY of 4th image:""\n"
  read X3
  echo "\n"
     if [ $X1 -lt X0 ] || [ $X2 -lt X1 ] || [ $X3 -lt X2 ] || [ $X2 -lt X0 ] || [ $X3 -lt X1 ]; then
       echo "Later dates cannot be earlier than previous dates""\n"
       echo "Exiting Program.""\n"
       exit 1
     fi
echo " You have chosen :
```

```
            First Date  [X0]:   avhrr.doy$X0.sm.x10000
            Second Date [X1]:   avhrr.doy$X1.sm.x10000
            Third Date  [X2]:   avhrr.doy$X2.sm.x10000
            Fourth Date [X3]:   avhrr.doy$X3.sm.x10000" | tee -a SIM.LOGFILE
    echo "\n"
    else
      echo "\n"
      echo "Exiting Program.""\n"
      exit 1
    fi
  fi
fi

# Enter the timestep that you want to interpolate in (in units of days).

  echo "Enter desired timestep (in days)""\n"
  read TIMESTEP
  echo "\n"
  echo "You have entered a timestep of $TIMESTEP days" | tee -a SIM.LOGFILE
  echo "\n"

#-----------------------------------------------------------------------------------------------------------------------------------------
# Linear Interpolation :
#
# P1(X) = D0 + (X - X0)D1 (Atkinson, pp106, Eqn 5.36), where:
#
# D0 = f[X0];
# D1 = f[X0,X1] = [f(X1) - f(X0) / (X1 - X0)] (pp101, Eqn 5.21)
#-----------------------------------------------------------------------------------------------------------------------------------------

if [ $ANSWER = 1 ]; then
   # Calculate D0 and D1 using above Eqn 5.21.
   r.mapcalc "D0 = avhrr.doy$X0.sm.x10000"
   r.mapcalc "D1 = ( avhrr.doy$X1.sm.x10000 - avhrr.doy$X0.sm.x10000 ) / ($X1 - $X0)"
   echo "Copying image avhrr.ord1.doy$X0.x10000 from avhrr.doy$X0.sm.x10000" | tee -a SIM.LOGFILE
   r.mapcalc "avhrr.ord1.doy$X0.x10000 = avhrr.doy$X0.sm.x10000"
   i.grey.scale avhrr.ord1.doy$X0.x10000
   let "X = X0 + TIMESTEP"
      while [ $X -lt $X1 ] ; do
        # Calculate P1(X) from above Eqn 5.36. for each timestep.
        echo "Interpolating image avhrr.ord1.doy$X.x10000" | tee -a SIM.LOGFILE
        r.mapcalc "avhrr.ord1.doy$X.x10000 = D0 + (( $X - $X0 )* D1)"
        i.grey.scale avhrr.ord1.doy$X.x10000
        let "X = X + TIMESTEP"
      done
   echo "Copying image avhrr.ord1.doy$X1.x10000 from avhrr.doy$X1.sm.x10000" | tee -a SIM.LOGFILE
   r.mapcalc "avhrr.ord1.doy$X1.x10000 = avhrr.doy$X1.sm.x10000"
   i.grey.scale avhrr.ord1.doy$X1.x10000
   g.remove rast=D0,D1
   else

#-----------------------------------------------------------------------------------------------------------------------------------------
# 2nd Order Interpolation :
#
# P2(X) = D0 + (X - X0)(D1 + (X - X1)D2)) (Atkinson, pp106, Eqn 5.36), where:
#
# D0 = f[X0];
# D1 = f[X0,X1] = [f(X1) - f(X0) / (X1 - X0)]
# D2 = f[X0,X1,X2] = (f[X1,X2] - f[X0,X1]) / (X2 - X0)
# where f[X1,X2] = [f(X2) - f(X1) / (X2 - X1)]; f[X0,X1] = D1; (pp101, Eqn 5.21)
#-----------------------------------------------------------------------------------------------------------------------------------------
```

```
 if [ $ANSWER = 2 ]; then
   # Calculate D0, D1 and D2 using above Eqn 5.21.
   r.mapcalc "D0 = avhrr.doy$X0.sm.x10000"
   r.mapcalc "D1 = ( avhrr.doy$X1.sm.x10000 - avhrr.doy$X0.sm.x10000 ) / ($X1 - $X0)"
   r.mapcalc "fX1X2 = ( avhrr.doy$X2.sm.x10000 - avhrr.doy$X1.sm.x10000 ) / ($X2 - $X1)"
   r.mapcalc "D2 = ( fX1X2 - D1 ) / ($X2 - $X0)"
   echo "Copying image avhrr.ord2.doy$X0.x10000 from avhrr.doy$X0.sm.x10000" | tee -a SIM.LOGFILE
   r.mapcalc "avhrr.ord2.doy$X0.x10000 = avhrr.doy$X0.sm.x10000"
   i.grey.scale avhrr.ord2.doy$X0.x10000
   let "X = X0 + TIMESTEP"
       while [ $X -lt $X1 ] ; do
         # Calculate P2(X) between X0 and X1 from Eqn 5.36 for each timestep.
         echo "Interpolating image avhrr.ord2.doy$X.x10000" | tee -a SIM.LOGFILE
         r.mapcalc "avhrr.ord2.doy$X.x10000 = D0 + ( $X - $X0 ) * (D1 + (( $X - $X1) * D2 ))"
         i.grey.scale avhrr.ord2.doy$X.x10000
         let "X = X + TIMESTEP"
       done
   echo "Copying image avhrr.ord2.doy$X1.x10000 from avhrr.doy$X1.sm.x10000" | tee -a SIM.LOGFILE
   r.mapcalc "avhrr.ord2.doy$X1.x10000 = avhrr.doy$X1.sm.x10000"
   i.grey.scale avhrr.ord2.doy$X1.x10000
   let "X = X1 + TIMESTEP"
       while [ $X -lt $X2 ] ; do
         # Calculate P2(X) between X1 and X2 from Eqn 5.36 for each timestep.
         echo "Interpolating image avhrr.ord2.doy$X.x10000" | tee -a SIM.LOGFILE
         r.mapcalc "avhrr.ord2.doy$X.x10000 = D0 + ( $X - $X0 ) * (D1 + (( $X - $X1) * D2 ))"
         i.grey.scale avhrr.ord2.doy$X.x10000
         let "X = X + TIMESTEP"
       done
   echo "Copying image avhrr.ord2.doy$X2.x10000 from avhrr.doy$X2.sm.x10000" | tee -a SIM.LOGFILE
   r.mapcalc "avhrr.ord2.doy$X2.x10000 = avhrr.doy$X2.sm.x10000"
   i.grey.scale avhrr.ord2.doy$X2.x10000
   g.remove rast=D0,D1,fX1X2,D2
   else

#-------------------------------------------------------------------------------------------------------------------------------------------------
# 3rd Order Interpolation :
#
# P3(X) = D0 + (X - X0)(D1 + (X - X1)(D2 + (X - X2)D3)))(Atkinson, pp106, Eqn 5.46,
# where:
#
# D0 = f[X0];
# D1 = f[X0,X1] = [f(X1) - f(X0) / (X1 - X0)]
# D2 = f[X0,X1,X2] = (f[X1,X2] - f[X0,X1]) / (X2 - X0)
#      where f[X1,X2] = [f(X2) - f(X1) / (X2 - X1)];
#            f[X0,X1] = D1
# D3 = f[X0,X1,X2,X3] = (f[X1,X2,X3] - f[X0,X1,X2]) / (X3 - X0)
#      where f[X1,X2,X3] = (f[X2,X3] - f[X1,X2]) / (X3 - X1)
#                 where f[X2,X3] = f(X3) - f(X2) / (X3 - X2)];
#                           f[X1,X2] = f(X2) - f(X1) / (X2 - X1)], and
#      where f[X0,X1,X2] = D2 ; (pp101, Eqn 5.21)
#-------------------------------------------------------------------------------------------------------------------------------------------------

 if [ $ANSWER = 3 ]; then
   # Calculate D0, D1, D2 and D3 using above Eqn 5.21.
   r.mapcalc "D0 = avhrr.doy$X0.sm.x10000"
   r.mapcalc "D1 = ( avhrr.doy$X1.sm.x10000 - avhrr.doy$X0.sm.x10000 ) / ($X1 - $X0)"
   r.mapcalc "fX1X2 = ( avhrr.doy$X2.sm.x10000 - avhrr.doy$X1.sm.x10000 ) / ($X2 - $X1)"
   r.mapcalc "D2 = ( fX1X2 - D1 ) / ($X2 - $X0)"
   r.mapcalc "fX2X3 = ( avhrr.doy$X3.sm.x10000 - avhrr.doy$X2.sm.x10000 ) / ($X3 - $X2)"
   r.mapcalc "fX1X2X3 = ( fX2X3 - fX1X2 ) / ($X3 -$X1)"
   r.mapcalc "D3 = ( fX1X2X3 - D2 ) / ($X3 -$X0)"
   echo "Copying image avhrr.ord3.doy$X0.x10000 from avhrr.doy$X0.sm.x10000" | tee -a SIM.LOGFILE
```

```
  r.mapcalc "avhrr.ord3.doy$X0.x10000 = avhrr.doy$X0.sm.x10000"
  i.grey.scale avhrr.ord3.doy$X0.x10000
  let "X = X0 + TIMESTEP"
      while [ $X -lt $X1 ] ; do
        # Calculate P3(X) between X0 and X1 from Eqn 5.36 for each timestep.
        echo "Interpolating image avhrr.ord3.doy$X.x10000" | tee -a SIM.LOGFILE
        r.mapcalc "avhrr.ord3.doy$X.x10000 = D0 + ($X - $X0) * (D1 + ($X - $X1) * ( D2 + ($X - $X2)*D3))"
        i.grey.scale avhrr.ord3.doy$X.x10000
        let "X = X + TIMESTEP"
      done
  echo "Copying image avhrr.ord3.doy$X1.x10000 from avhrr.doy$X1.sm.x10000" | tee -a SIM.LOGFILE
  r.mapcalc "avhrr.ord3.doy$X1.x10000 = avhrr.doy$X1.sm.x10000"
  i.grey.scale avhrr.ord3.doy$X1.x10000
  let "X = X1 + TIMESTEP"
      while [ $X -lt $X2 ] ; do
        # Calculate P3(X) between X1 and X2 from Eqn 5.36 for each timestep.
        echo "Interpolating image avhrr.ord3.doy$X.x10000" | tee -a SIM.LOGFILE
        r.mapcalc "avhrr.ord3.doy$X.x10000 = D0 + ($X - $X0) * (D1 + ($X - $X1) * ( D2 + ($X - $X2)*D3))"
        i.grey.scale avhrr.ord3.doy$X.x10000
        let "X = X + TIMESTEP"
      done
  echo "Copying image avhrr.ord3.doy$X2.x10000 from avhrr.doy$X2.sm.x10000" | tee -a SIM.LOGFILE
  r.mapcalc "avhrr.ord3.doy$X2.x10000 = avhrr.doy$X2.sm.x10000"
  i.grey.scale avhrr.ord3.doy$X2.x10000
  let "X = X2 + TIMESTEP"
      while [ $X -lt $X3 ] ; do
        # Calculate P3(X) between X2 and X3 from Eqn 5.36 for each timestep.
        echo "Interpolating image avhrr.ord3.doy$X.x10000" | tee -a SIM.LOGFILE
        r.mapcalc "avhrr.ord3.doy$X.x10000 = D0 + ($X - $X0) * (D1 + ($X - $X1) * ( D2 + ($X - $X2)*D3))"
        i.grey.scale avhrr.ord3.doy$X.x10000
        let "X = X + TIMESTEP"
      done
  echo "Copying image avhrr.ord3.doy$X3.x10000 from avhrr.doy$X3.sm.x10000" | tee -a SIM.LOGFILE
  r.mapcalc "avhrr.ord3.doy$X3.x10000 = avhrr.doy$X3.sm.x10000"
  i.grey.scale avhrr.ord3.doy$X3.x10000
  g.remove rast=D0,D1,D2,D3,fX1X2,fX2X3,fX1X2X3
  else
      echo "\n"
      echo "Exiting Program.""\n"
      exit 1
    fi
  fi
fi
```

APPENDIX IV. GRASS shell script code for Calculation of AVHRR-derived Approach I and II TTIs.

CALCULATION OF NDON AND DOYON

```
# !/bin/ksh -f
#-----------------------------------------------------------------------------------------------------------------------------------
# Andrew Davidson, 2002.
# (1) For each grid cell in a series of images, this script calculates the minimum cell
# value. Uses a loop, so user just has to specify start and end dates for sampling.
#
# (2) For each grid cell in image, script also assigns the DOY on which the minimum value
# occurred.
#
# set range to search at beginning of field season.
#-----------------------------------------------------------------------------------------------------------------------------------

 let "STARTDOY = 70"
 let "SECDOY = STARTDOY + 1"
 let "ENDDOY = 130"

 let "STDY = STARTDOY"
 let "TDY = SECDOY"
 let "EDY = ENDDOY"

 r.mapcalc MIN.TMP = 999999

 while [ $STDY -lt $EDY ] ; do
   r.mapcalc "MIN.TMP1 = min(avhrr.ord3.doy$STDY.x10000,avhrr.ord3.doy$TDY.x10000)"
   r.mapcalc "MINBASE = min(MIN.TMP,MIN.TMP1)"
   r.mapcalc "MIN.TMP = MINBASE" | tee -a MIN.RUN.REPORT
   g.remove rast=MIN.TMP1,MINBASE
   let "STDY = STDY + 1"
   let "TDY = TDY + 1"
 done
 r.mapcalc "min.start.ndvi.avhrr.x10000 = MIN.TMP"
 i.grey.scale min.start.ndvi.avhrr.x10000
 g.remove rast=MIN.TMP


# Identifies DOY at which the above minima occur.

 let "STDY = STARTDOY"
 let "EDY = ENDDOY + 1"

 while [ $STDY -lt $EDY ] ; do
   r.mapcalc "DOY.TMP = min.start.ndvi.avhrr.x10000 - avhrr.ord3.doy$STDY.x10000"
   r.mapcalc "DOY.BIN = if(DOY.TMP==0,1,0)"
   r.mapcalc "DOY.$STDY = DOY.BIN * $STDY"
   g.remove rast=DOY.TMP,DOY.BIN
   let "STDY = STDY + 1"
 done

# Adds the above maps together to get a DOY map.

 let "STDY = STARTDOY"
 let "TDY = SECDOY"
 let "EDY = ENDDOY + 1"
 r.mapcalc FINDOY.TMP = 0

 while [ $STDY -lt $EDY ] ; do
   # test to see if cell is already "full".
   r.mapcalc "TESTZERO =  min.start.ndvi.avhrr.x10000 - avhrr.ord3.doy$STDY.x10000"
```

```
  # resets cell to zero if already full
  r.mapcalc "RESETZERO = if(TESTZERO==0,0,1)"
  r.mapcalc "UPDATE = FINDOY.TMP * RESETZERO"
  r.mapcalc "FINDOY.TMP = UPDATE"
  r.mapcalc "FINDOY.TMP1 = FINDOY.TMP + DOY.$STDY"
  r.mapcalc "FINDOY.TMP = FINDOY.TMP1"
  g.remove rast=FINDOY.TMP1,DOY.$STDY,TESTZERO,RESETZERO,UPDATE
  let "STDY = STDY + 1"
 done

  r.mapcalc "min.start.doy.ndvi.avhrr = FINDOY.TMP"
  g.remove rast=FINDOY.TMP
  i.grey.scale min.start.doy.ndvi.avhrr
```

CALCULATION OF NDMAX AND DOYMAX

```
# !/bin/ksh -f
#-----------------------------------------------------------------------------------------------------------------------------------
# Andrew Davidson, 2002.
# (1) For each grid cell in a series of images, this script calculates the maximum cell NDVI value. Uses a
# loop, so user just has to specify start and end dates for sampling.
#
# (2) For each grid cell in image, script also assigns the DOY on which the maximum value occurred.
#
# Set range to search between peaks of curves at field sites
#-----------------------------------------------------------------------------------------------------------------------------------

 let "STARTDOY = 130"
 let "SECDOY = STARTDOY + 1"
 let "ENDDOY = 270"

 let "STDY = STARTDOY"
 let "TDY = SECDOY"
 let "EDY = ENDDOY"

 r.mapcalc MAX.TMP = 0

 while [ $STDY -lt $EDY ] ; do
  r.mapcalc "MAX.TMP1 = max(avhrr.ord3.doy$STDY.x10000,avhrr.ord3.doy$TDY.x10000)"
  r.mapcalc "MAXBASE = max(MAX.TMP,MAX.TMP1)"
  r.mapcalc "MAX.TMP = MAXBASE" | tee -a MAX.RUN.REPORT
  g.remove rast=MAX.TMP1,MAXBASE
  let "STDY = STDY + 1"
  let "TDY = TDY + 1"
 done
 r.mapcalc "max.ndvi.avhrr.x10000 = MAX.TMP"
 i.grey.scale max.ndvi.avhrr.x10000
 g.remove rast=MAX.TMP

# Identifies DOY at which the above maxima occur.

 let "STDY = STARTDOY"
 let "EDY = ENDDOY + 1"

 while [ $STDY -lt $EDY ] ; do
  r.mapcalc "DOY.TMP = max.ndvi.avhrr.x10000 - avhrr.ord3.doy$STDY.x10000"
  r.mapcalc "DOY.BIN = if(DOY.TMP==0,1,0)"
  r.mapcalc "DOY.$STDY = DOY.BIN * $STDY"
  g.remove rast=DOY.TMP,DOY.BIN
  let "STDY = STDY + 1"
```

```
 done

# Adds the above maps together to get a DOY map.

 let "STDY = STARTDOY"
 let "TDY = SECDOY"
 let "EDY = ENDDOY + 1"
 r.mapcalc FINDOY.TMP = 0

 while [ $STDY -lt $EDY ] ; do
   # test to see if cell is already "full".
   r.mapcalc "TESTZERO = max.ndvi.avhrr.x10000 - avhrr.ord3.doy$STDY.x10000"
   # resets cell to zero if already full; thus taking latermost peak DOY.
   r.mapcalc "RESETZERO = if(TESTZERO==0,0,1)"
   r.mapcalc "UPDATE = FINDOY.TMP * RESETZERO"
   r.mapcalc "FINDOY.TMP = UPDATE"
   r.mapcalc "FINDOY.TMP1 = FINDOY.TMP + DOY.$STDY"
   r.mapcalc "FINDOY.TMP = FINDOY.TMP1"
   g.remove rast=FINDOY.TMP1,DOY.$STDY,TESTZERO,RESETZERO,UPDATE
   let "STDY = STDY + 1"
 done

   r.mapcalc "max.doy.ndvi.avhrr = FINDOY.TMP"
   i.grey.scale max.doy.ndvi.avhrr
   g.remove rast=FINDOY.TMP
```

CALCULATION OF NDEND AND DOYEND

```
# !/bin/ksh -f
#-----------------------------------------------------------------------------------------------------------------------------------------
# Andrew Davidson, 2002.
# (1) For each grid cell in a series of images, this script calculates the minimum cell
# value. Uses a loop, so user just has to specify start and end dates for sampling.
#
# (2) For each grid cell in image, script also assigns the DOY on which the minimum value
# occurred.
#
# set range to search at end of field season.
#-----------------------------------------------------------------------------------------------------------------------------------------

 let "STARTDOY = 260"
 let "SECDOY = STARTDOY + 1"
 let "ENDDOY = 321"


 let "STDY = STARTDOY"
 let "TDY = SECDOY"
 let "EDY = ENDDOY"

 r.mapcalc MIN.TMP = 999999

 while [ $STDY -lt $EDY ] ; do
   r.mapcalc "MIN.TMP1 = min(avhrr.ord3.doy$STDY.x10000,avhrr.ord3.doy$TDY.x10000)"
   r.mapcalc "MINBASE = min(MIN.TMP,MIN.TMP1)"
   r.mapcalc "MIN.TMP = MINBASE" | tee -a MIN.RUN.REPORT
   g.remove rast=MIN.TMP1,MINBASE
   let "STDY = STDY + 1"
   let "TDY = TDY + 1"
 done
 r.mapcalc "min.end.ndvi.avhrr.x10000 = MIN.TMP"
```

```
 i.grey.scale min.end.ndvi.avhrr.x10000
 g.remove rast=MIN.TMP

# Identifies DOY at which the above minima occur.

 let "STDY = STARTDOY"
 let "EDY = ENDDOY + 1"

 while [ $STDY -lt $EDY ] ; do
   r.mapcalc "DOY.TMP = min.end.ndvi.avhrr.x10000 - avhrr.ord3.doy$STDY.x10000"
   r.mapcalc "DOY.BIN = if(DOY.TMP==0,1,0)"
   r.mapcalc "DOY.$STDY = DOY.BIN * $STDY"
   g.remove rast=DOY.TMP,DOY.BIN
   let "STDY = STDY + 1"
 done

# Adds the above maps together to get a DOY map.

 let "STDY = STARTDOY"
 let "TDY = SECDOY"
 let "EDY = ENDDOY + 1"
 r.mapcalc FINDOY.TMP = 0

 while [ $STDY -lt $EDY ] ; do
   # test to see if cell is already "full".
   r.mapcalc "TESTZERO =  min.end.ndvi.avhrr.x10000 - avhrr.ord3.doy$STDY.x10000"
   # resets cell to zero if already full
   r.mapcalc "RESETZERO = if(TESTZERO==0,0,1)"
   r.mapcalc "UPDATE = FINDOY.TMP * RESETZERO"
   r.mapcalc "FINDOY.TMP = UPDATE"
   r.mapcalc "FINDOY.TMP1 = FINDOY.TMP + DOY.$STDY"
   r.mapcalc "FINDOY.TMP = FINDOY.TMP1"
   g.remove rast=FINDOY.TMP1,DOY.$STDY,TESTZERO,RESETZERO,UPDATE
   let "STDY = STDY + 1"
 done

   r.mapcalc "min.end.doy.ndvi.avhrr = FINDOY.TMP"
   g.remove rast=FINDOY.TMP
   i.grey.scale min.end.doy.ndvi.avhrr
```

CALCULATION OF DOYRAN, RAGUP AND RASEN

Specified at command line in GRASS once previous coverages created:

r.mapcalc “DOYRAN = DOYEND – DOYON”

r.mapcalc “RAGUP = ((NDMAX – NDON) / (DOYMAX – DOYON))”

r.mapcalc “RASEN = ((NDEND – NDMAX) / (DOYEND – DOYMAX))”

CALCULATION OF NDTIN

```
#! /usr/bin/ksh -f #
#-----------------------------------------------------------------------------------------------------------------------------------
# Andrew Davidson, 2002.
# This shell calculated TINDVI from the productivity metrics DOYON, DOYEND, NDON, NDEND. These
# coverages must exist before running this script. For details of the metrics used. see Reed at al (1994).
#-----------------------------------------------------------------------------------------------------------------------------------
```

```
# Set ranges for DOYON (from r.stats DOYON) and DOYEND (from r.stats DOYEND).
#
# let "DOYON1 = 70"
# let "DOYON2 = 115"
# let "DOYEND1 = 260"
# let "DOYEND2 = 321"
# let "TIMESTEP = 1"

let "DOYON1 = 70"
let "DOYON2 = 115"
let "DOYEND1 = 260"
let "DOYEND2 = 321"
let "TIMESTEP = 1"

let "ST1 = DOYON1"
let "ST2 = DOYON2 + 1"
let "END1 = DOYEND1"
let "END2 = DOYEND2 + 1"
r.mapcalc "TOTTINDVI.TMP = 0"

# Creates binary images for presence of combinations of start and end dates. Will later mask out values.

while [ $ST1 -lt $ST2 ] ; do
  while [ $END1 -lt $END2 ] ; do
   # Identify cells where start date is $ST1
     r.mapcalc "DOY.$ST1.BIN = if(DOYON==$ST1,1,0)"
   # Identify cells where end date is $END1
     r.mapcalc "DOY.$END1.BIN = if(DOYEND==$END1,1,0)"
   # Identifies cells where start date is $ST1 and end date $END1
     r.mapcalc "DOY.$ST1.$END1.BIN = DOY.$ST1.BIN * DOY.$END1.BIN"
     g.remove rast=DOY.$ST1.BIN,DOY.$END1.BIN
     r.stats DOY.$ST1.$END1.BIN > ZERO.TEST
   # If image is full of 0s, then increment $END1 and do again
     if grep "1" ZERO.TEST; then
        # -- What to do if image contains 1s --
        # Create baseline NDVI over time period from $ST1 to $END1
         echo "Image $ST1.$END1 includes 1s" | tee -a TINDVI.LOGFILE
         rm ZERO.TEST
         r.mapcalc "D0 = avhrr.ord3.doy$ST1.x10000"
         r.mapcalc "D1 = ( avhrr.ord3.doy$END1.x10000 - avhrr.ord3.doy$ST1.x10000 ) / ($END1 -$ST1)"
         let "ST11 = ST1"
         r.mapcalc "TINDVI.TMP = 0"
            while [ $ST11 -lt $END1 ] ; do
              # For each timestep btw $ST1 and $END1 create a linear baseline from $ST1 to $END1
                echo "Creating baseline BASE.$ST1.$END1.$ST11"
                r.mapcalc "BASE.$ST1.$END1.$ST11 = D0 + (( $ST11 - $ST1 ) * D1 )"
              # For each timestep of $ST1 and $END1, create a new productivity estimate (actual -baseline)
                echo "Estimating new productivity ND.$ST1.$END1.$ST11"
          r.mapcalc "ND.$ST1.$END1.$ST11=(avhrr.ord3.doy$ST11.x10000 -BASE.$ST1.$END1.$ST11)"
              # Total NDVI within the range above
                echo "Adding file ND.$ST1.$END1.$ST11 to total"
                r.mapcalc "TOTAL.TMP1 = TINDVI.TMP + ND.$ST1.$END1.$ST11"
                r.mapcalc "TINDVI.TMP = TOTAL.TMP1"
                g.remove rast=TOTAL.TMP1,ND.$ST1.$END1.$ST11,BASE.$ST1.$END1.$ST11
                let "ST11 = ST11 + TIMESTEP"
            done
         # But only create final TINDVI image for cells where this combo occurs (* by binary mask)
           echo "Applying mask DOY.$ST1.$END1.BIN to create TINDVI.$ST1.$END1"
           r.mapcalc "TINDVI.$ST1.$END1 = TINDVI.TMP * DOY.$ST1.$END1.BIN"
           g.remove rast=DOY.$ST1.$END1.BIN,TINDVI.TMP
         # Add this image to the running total (TOTTINDVI.TMP) to create final TINDVI image
```

```
            echo "Adding file TINDVI.$ST1.$END1 to total"
            r.mapcalc "TOTAL.TMP1 = TOTTINDVI.TMP + TINDVI.$ST1.$END1"
            r.mapcalc "TOTTINDVI.TMP = TOTAL.TMP1"
            g.remove rast=TINDVI.$ST1.$END1,TOTAL.TMP1
          # Increment $END1 to do all combos of end for given start date.
            let "END1 = END1 + TIMESTEP"
      else
      # -- What to do if binary image is full of zeros --
      rm ZERO.TEST
      echo "Image $ST1.$END1 all zeros" | tee -a TINDVI.LOGFILE
      g.remove rast=DOY.$ST1.$END1.BIN
      let "END1 = END1 + TIMESTEP"
     fi
   done
   echo "Completed start date $ST1" | tee -a TINDVI.LOGFILE
  # Increment $ST1 to move onto a different start date.
    let "END1 = DOYEND1"
    let "ST1 = ST1 + 1" done
done

 r.mapcalc "NDTIN = TOTTINDVI.TMP"
 i.grey.scale NDTIN
 g.remove rast=TOTTINDVI.TMP,D0,D1
```

CALCULATION OF GDD TTIS

```
#! /usr/bin/ksh -f #
#------------------------------------------------------------------------------
# Andrew Davidson, 2002.
# This shell calculates the doy at which 10%, 20% … 90% of TINDVI has been accumulated.
# Derived from the productivity metrics DOYON, DOYEND, NDON, NDEND. These coverages must exist # before
running this script. For details of the metrics used. see Reed at al (1994).
#------------------------------------------------------------------------------

g.region raster=vlu_assoc

let "DOYON1 = 70"
let "DOYON2 = 115"
let "DOYEND1 = 260"
let "DOYEND2 = 321"
let "TIMESTEP = 1"
let "PERC = 20"

let "ST1 = DOYON1"
let "ST2 = DOYON2 + 1"
let "END1 = DOYEND1"
let "END2 = DOYEND2 + 1"
r.mapcalc "TOTTINDPER$PERC.TMP = 0"

# Creates binary images for the presence of combinations of start and end dates. Will be
# later to mask out values.
while [ $ST1 -lt $ST2 ] ; do
  while [ $END1 -lt $END2 ] ; do
   # Identify cells where start date is $ST1
     r.mapcalc "DOY.$ST1.BIN = if(DOYON==$ST1,1,0)"
   # Identify cells where end date is $END1
     r.mapcalc "DOY.$END1.BIN = if(DOYEND==$END1,1,0)"
   # Identifies cells where start date is $ST1 and end date $END1
     r.mapcalc "DOY.$ST1.$END1.BIN = DOY.$ST1.BIN * DOY.$END1.BIN"
```

```
    g.remove rast=DOY.$ST1.BIN,DOY.$END1.BIN
    r.stats DOY.$ST1.$END1.BIN > ZERO.TEST
  # If image is full of 0s, then increment $END1 and do again
    if grep "1" ZERO.TEST; then
       # -- What to do if image contains 1s --
       # Create baseline NDVI over time period from $ST1 to $END1
        echo "Image $ST1.$END1 includes 1s" | tee -a TINDVI.LOGFILE
        rm ZERO.TEST
        r.mapcalc "D0 = avhrr.ord3.doy$ST1.x10000"
        r.mapcalc "D1 = ( avhrr.ord3.doy$END1.x10000 - avhrr.ord3.doy$ST1.x10000 ) / ($END1 -$ST1)"
        let "ST11 = ST1"
        r.mapcalc "TINDVI.TMP = 0"
        r.mapcalc "TINDPER$PERC.TMP = 0"
         while [ $ST11 -lt $END1 ] ; do
            # Check to see if the temporary perc file is "filled"
             r.stats TINDPER$PERC.TMP > NEWZERO.TEST
            # If image contains zeros, do following loop, otherwise, increment $ST11 and start again
             if grep "0" NEWZERO.TEST; then
                # -- What to do if image contains 0s --
                 rm NEWZERO.TEST
                # For each timestep btw $ST1 and $END1 create a linear baseline from $ST1 to $END1
                 echo "Creating baseline BASE.$ST1.$END1.$ST11"
                 r.mapcalc "BASE.$ST1.$END1.$ST11 = D0 + (( $END1 - $ST11 ) * D1 )"
               # For each timestep of $ST1 and $END1, create a new productivity estimate (actual - base)
                 echo "Estimating new productivity ND.$ST1.$END1.$ST11"
r.mapcalc "ND.$ST1.$END1.$ST11 = ( avhrr.ord3.doy$ST11.x10000 - BASE.$ST1. $END1.$ST11)/100"
                # Total NDVI within the range above
                 echo "Adding file ND.$ST1.$END1.$ST11 to total"
                 r.mapcalc "TOTAL.TMP1 = TINDVI.TMP + ND.$ST1.$END1.$ST11"
                 r.mapcalc "TINDVI.TMP = TOTAL.TMP1"
                # Test to see if current total > X% of total NDVI; if so assign $ST11 (DOY) value.
r.mapcalc "TINDPER$PERC.TMP1=if(TINDPER$PERC.TMP==0 && ((TINDVI.TMP*100)/NDTIN) > $PERC,$ST11)"
                 r.mapcalc "TINDPER$PERC.TMP2 = TINDPER$PERC.TMP + TINDPER$PERC.TMP1"
                 r.mapcalc "TINDPER$PERC.TMP = TINDPER$PERC.TMP2"
                # Cleanup
                 let "ST11 = ST11 + TIMESTEP"
             else
             # -- What to do if image is full of nonzeros --
             let "ST11 = ST11 + TIMESTEP"
             fi
         done
        # But only create final TINDPER20 image for cells where this combo occurs (* by binary mask)
         echo "Applying mask DOY.$ST1.$END1.BIN to create TINDPER$PERC.$ST1.$END1"
         r.mapcalc "TINDPER$PERC.$ST1.$END1 = TINDPER$PERC.TMP * DOY.$ST1.$END1.BIN"
         g.remove rast=DOY.$ST1.$END1.BIN,TINDPER$PERC.TMP,TINDVI.TMP
        # Add this image to the running total (TOTTINDPER$PERC.TMP) to create final TINDPER image
         echo "Adding file TINDPER$PERC.$ST1.$END1 to total"
         r.mapcalc "TOTAL.TMP1 = TOTTINDPER$PERC.TMP + TINDPER$PERC.$ST1.$END1"
         r.mapcalc "TOTTINDPER$PERC.TMP = TOTAL.TMP1"
         g.remove rast=TINDPER$PERC.$ST1.$END1,TOTAL.TMP1
        # Increment $END1 to do all combos of end for given start date.
         let "END1 = END1 + TIMESTEP"
    else
    # -- What to do if image is full of zeros --
    rm NEWZERO.TEST
    echo "Image $ST1.$END1 all zeros" | tee -a TINDVI.LOGFILE
    g.remove rast=DOY.$ST1.$END1.BIN
    let "END1 = END1 + TIMESTEP"
    fi
  done
  echo "Completed start date $ST1" | tee -a TINDVI.LOGFILE
 # Increment $ST1 to move onto a different start date.
```

```
    let "END1 = DOYEND1"
    let "ST1 = ST1 + 1" done
done

 r.mapcalc "DOYINT$PERC = TOTTINDPER$PERC"
 i.grey.scale DOYINT$PERC
 g.remove rast=TOTTINDPER$PERC.TMP,D0,D1
```