Test parameters tuning

06/03/2025 - To do list

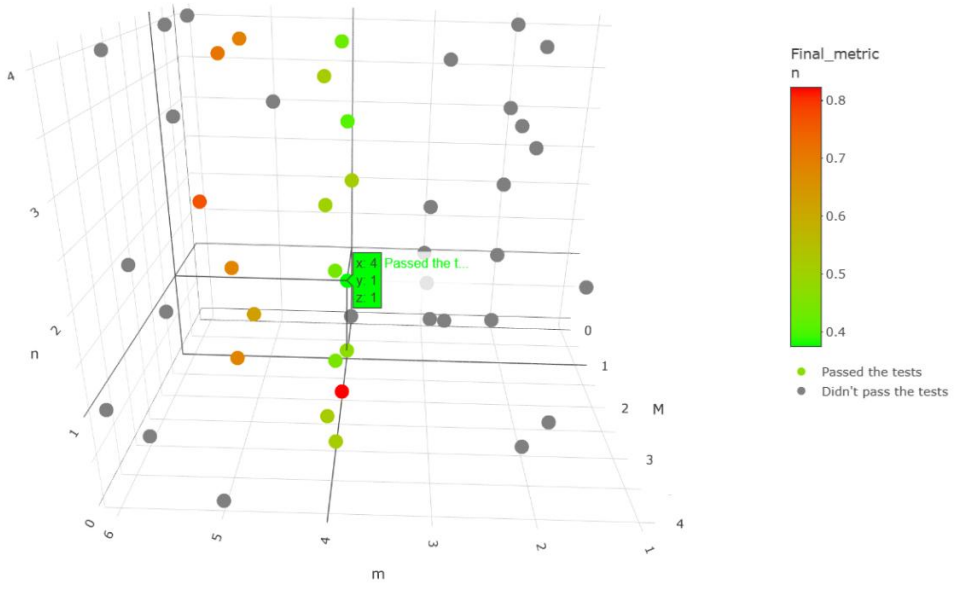
Cose da fare:

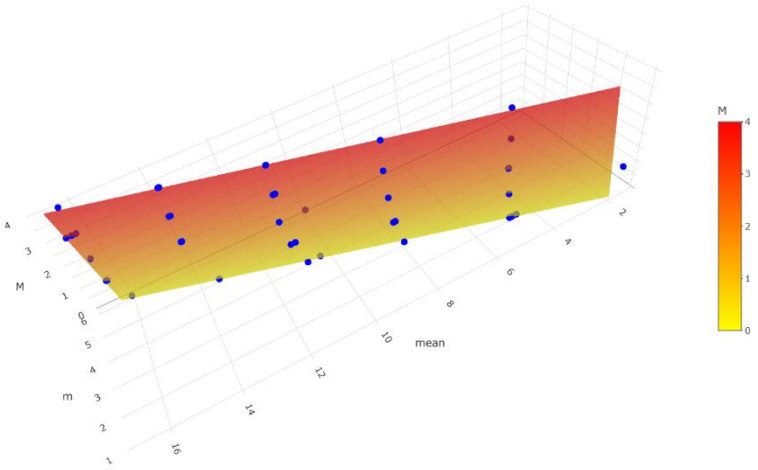
- Testare come generare combinazioni random su bash (FATTO);
- Capire come creare un log delle combinazioni (FATTO);
- Capire come aggiungere le cose nel log (FATTO);
- Testare come controllare se una popolazione è già stata esplorata (FATTO);
- Runnare solo ustacks e capire che output dà e quali file sono utilizzati nei passaggi seguenti (FATTO). Risposta: ustacks runna per ciascuna popolazione e, per ognuna, crea tre file che verranno tutti utilizzati dai seguenti passaggi. Quindi bisognerebbe creare una cartella per ogni combinazione. Viene un casino ma si può fare. Risposta: non ho voglia di farlo;
- Minimizzare la objective function richiede creare dei modelli per predire i valori di mean, sd, SNPs e unique: capire se ci vuole linear Regression o altri tipi di regressione (FATTO).

Alla fine la selezione Bayesiana sembra andare bene, quindi non farei algoritmi evolutivi che richiedono tanto lavoro. Gli algoritmi evolutivi andrebbero fatti su delle operazioni che richiedono valutazioni più veloci ma spazi di ricerca più grandi e il nostro caso è esattamente il contrario. È stato creato anche un algoritmo su pyhton utilizzando scikit-learn che prova a minimizzare la funzione prevedendo, per le combinazioni non esplorate, il valore di SNPs, unique, mean e sd, usando delle linear regression. Il problema è che le regression (o, almeno, le 4 create) non trovano nessun parametro significativo se non 'm' (in realtà mean utilizza anche M) e quindi suggeriscono M ed n quasi completamente a caso.

07/03/2025 - Risultati

Ho provato 46 combinazioni in tutto: il codice per la valutazione delle metriche si può trovare su Desktop/Tirocinio/06-03/final_metrics_plotted.R. Il miglior risultato rimane sempre m = 4, M = 1 e n = 1, con una final metric (normalizzata per tutte le metriche che hanno passato il test, e solo tra quelle che abbiamo visto) di 0.3745734. Inoltre notiamo come le combinazioni che passano i test hanno sempre m compreso tra 4 e 5, con 4 che ha risultati migliori. m infatti risulta essere l'unico parametro che influenza sd, SNPs e unique, mentre mean è influenzato sia da m che da M. Ho provato a creare dei modelli predittivi su R ed ho ottenuto:

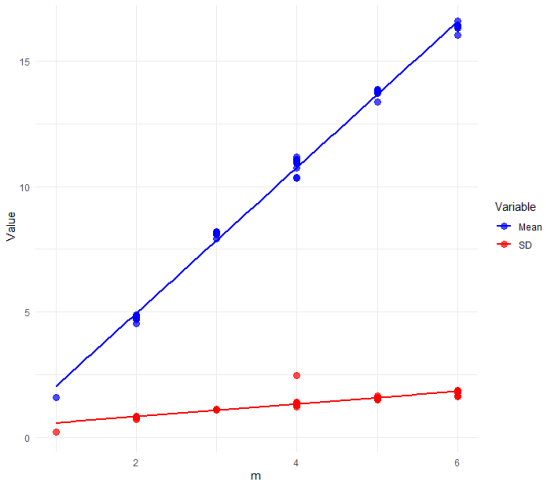




- **Mean** è influenzato sia da m che da M, e assolutamente non da n. Questo ha senso perché i livelli di coverage sono calcolati dopo il primo passaggio, quello con ustacks, che utilizza solo i primi due parametri. Con RStudio ho creato un modello molto robusto (p-value < 2.2e-16) e che spiega il 99.7% della variabilità:

$$mean = -0.94117 + 2.88444m + 0.08466M$$

In cui risulta che m sia MOLTO significativo (p-value < 2e-16) e M abbastanza (p-value = 0.00169). Quindi il valore di m è importantissimo, mentre il valore di M aiuta marginalmente ad aumentare la mean coverage. Non c'è overfitting, in quanto con una 10-fold validation ripetuta 10 volte, ottengo un Rsquared: 0.9980 ± 0.0017.

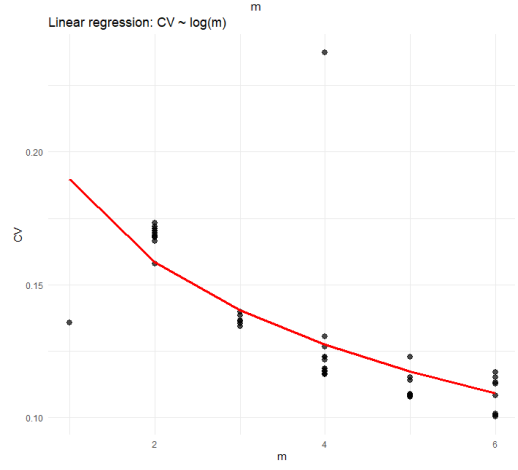


- La **standard deviation** dipende solo da m (p.value < 2.2e-16):

$$sd = 0.31958 + 0.25126m$$

Questo spiega il 78% della variabilità, che è un buon risultato, con 10-fold crossvalidation ripetuto 30 volte ottengo 0.9196 ± 0.1817. Il nostro obiettivo è che la mean sia alta ma sd basso: dato che m aumenta entrambe è necessario avere una giusta via di mezzo, per questo è importante non superare certi valori (infatti il test non è superato per valori di m maggiori di 5). Possiamo provare a plottare sia mean che sd nello stesso grafico.

Dato che mean e sd interagiscono insieme nel valore di CV ho provato a creare un modello per CV che dipende solo da m:



$$CV = 0.189606 - 0.044901 \cdot \log(m)$$

Quindi aumentare in modo spropositato m non aiuta, anche perché questo ha delle conseguenze su SNPs e unique.

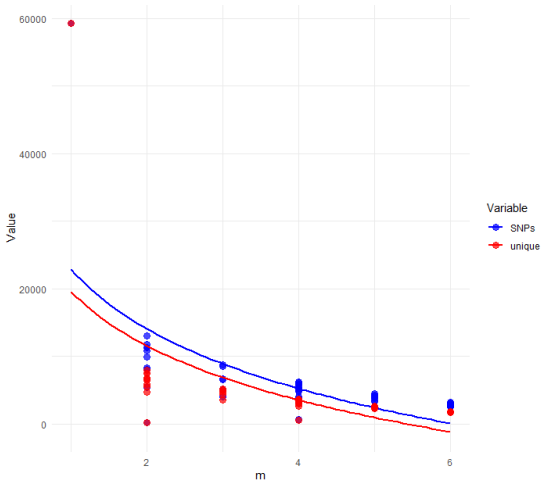
- La relazione tra m e **SNPs e unique** è un po' più complicata. Sembra che diminuisca drasticamente all'inizio e poi si stabilizzi, quindi ho provato una scala logaritmica per m, ma questo spiega solo il 30% della variabilità. Per arrivare al 90% della variabilità, avrei dovuto creare dei modelli polinomiali fino al quarto grado. Rimanendo su log(m), ho in questo modo:

$$SNPs = 22869 - 12708 \cdot \log(m) \qquad unique = -19573 - 11558 \cdot \log(m)$$

I modelli spiegano solo il 30 e 40% della variabilità, però i risultati di un 10-fold cross-validation (fatto su R con la libreria caret) sono migliori. Ho ripetuto la cross validation 30 volte per ciascun modello, ed ho ottenuto in media:

- Rsquared per SNPs: 0.76 ± 0.26
- Rsquared per unique: 0.84 ± 0.27

Che non sono bellissimi, ma se messi a confronto con ciò che ottengo per un modello polinomiale con 4 gradi (che avrebbe spiegato il 90% della variabilità in entrambi i casi):



- Rsquared per SNPs: 0.59 ± 0.33
- Rsquared per unique: 0.52 ± 0.34

è sicuramente un risultato migliore. Questo mi fa pensare che il modello polinomiale che spiega il 90% della variabilità è in realtà un caso di over-fitting.

In generale ecco qui una tabella con tutte le combinazioni provate:

m	M	n	mean	sd	SNPs	unique	FAIL	CV	SNPs_per_locus	CV_norm	SNP_per_locus_norm	unique_norm	Final_metric
3	2	1	8,19	1,118	6546	4788	5	0,1365	1,3672	0,7884	0,7690	0,0808	6,4766
6	3	2	16,60	1,878	2914	1812	5	0,1131	1,6082	0,6534	0,9045	0,0306	6,5273
4	1	0	10,92	1,341	3059	2621	0	0,1228	1,1671	0,5174	0,6644	0,7120	0,4698
4	1	1	**10,92**	1,341	3773	**3087**	0	0,1228	1,2222	0,5174	0,6958	0,8386	**0,3746**
5	2	1	13,84	1,577	3487	2447	0	0,1139	1,4250	0,4801	0,8113	0,6648	0,6266
1	1	1	1,60	0,217	59256	59256	5	0,1356	1,0000	0,7833	0,5625	1,0000	5,3457
3	1	2	8,16	1,129	6746	4938	5	0,1383	1,3661	0,7990	0,7684	0,0833	6,4841
4	4	3	11,01	1,295	6165	3583	0	0,1176	1,7206	0,4956	0,9796	0,9734	0,5018
6	0	4	16,03	1,875	2759	1700	5	0,1169	1,6229	0,6754	0,9128	0,0287	6,5596
3	0	1	7,91	1,107	4067	3530	5	0,1399	1,1521	0,8080	0,6480	0,0596	6,3964
6	4	1	16,30	1,635	3218	1810	5	0,1003	1,7779	0,5792	1,0000	0,0305	6,5487
5	2	4	13,87	1,598	4165	2600	0	0,1152	1,6019	0,4854	0,9120	0,7063	0,6910
2	4	1	4,83	0,812	10851	6741	5	0,1681	1,6097	0,9707	0,9054	0,1138	6,7623
5	4	0	13,70	1,489	3985	2303	5	0,1087	1,7304	0,6276	0,9733	0,0389	6,5620
5	3	2	13,81	1,495	3954	2491	0	0,1082	1,5873	0,4561	0,9037	0,6767	0,6830
4	3	0	11,02	1,279	4824	3136	0	0,1161	1,5383	0,4891	0,8758	0,8519	0,5129
6	1	4	16,40	1,844	2953	1881	5	0,1125	1,5699	0,6497	0,8830	0,0317	6,5010
6	1	3	16,37	1,886	2662	1833	5	0,1152	1,4523	0,6655	0,8168	0,0309	6,4514
4	3	2	11,09	1,290	5484	3489	0	0,1164	1,5718	0,4904	0,8948	0,9478	0,4374
4	0	2	10,75	1,402	4035	2977	0	0,1304	1,3554	0,5496	0,7716	0,8087	0,5125
4	0	0	10,32	1,222	698	621	5	0,1184	1,1240	0,6840	0,6322	0,0105	6,3057
3	1	1	8,09	1,101	5204	4345	5	0,1361	1,1977	0,7862	0,6737	0,0733	6,3866
2	0	4	4,80	0,831	11315	6758	5	0,1731	1,6743	1,0000	0,9417	0,1140	6,8277
3	4	2	8,15	1,094	8516	5094	5	0,1342	1,6718	0,7753	0,9403	0,0860	6,6297
5	0	3	13,38	1,642	3348	2265	5	0,1227	1,4781	0,7088	0,8314	0,0382	6,5020
5	4	3	13,73	1,481	4451	2534	0	0,1078	1,7565	0,4542	1,0000	0,6884	0,7658
2	2	4	4,90	0,829	13104	7910	5	0,1693	1,6566	0,9778	0,9318	0,1335	6,7761
2	0	1	4,70	0,788	5462	4761	5	0,1677	1,1472	0,9686	0,6453	0,0803	6,5335
2	2	3	4,88	0,821	11725	7546	5	0,1685	1,5538	0,9729	0,8740	0,1273	6,7195
6	2	1	16,39	1,774	2584	1779	5	0,1082	1,4525	0,6250	0,8170	0,0300	6,4120
4	3	1	11,05	1,284	5298	3418	0	0,1162	1,5500	0,4896	0,8824	0,9286	0,4435
5	3	1	13,79	1,487	3845	2450	0	0,1079	1,5694	0,4545	0,8935	0,6656	0,6824
4	4	1	11,01	1,291	5809	3449	0	0,1173	1,6843	0,4941	0,9589	0,9370	0,5160
2	0	2	4,75	0,809	8077	5903	5	0,1702	1,3683	0,9830	0,7696	0,0996	6,6530
2	0	0	4,55	0,718	233	219	5	0,1580	1,0639	0,9123	0,5984	0,0037	6,5070
2	1	3	4,84	0,828	11362	7481	5	0,1711	1,5188	0,9880	0,8543	0,1262	6,7160
4	1	3	11,07	1,401	5222	3549	0	0,1266	1,4714	0,5332	0,8377	0,9641	0,4067
2	3	0	4,82	0,802	8262	5566	5	0,1662	1,4844	0,9601	0,8349	0,0939	6,7011
4	2	0	10,35	2,457	5002	3580	0	0,2374	1,3972	1,0000	0,7954	0,9726	0,8229
3	3	4	8,21	1,112	8757	5237	5	0,1354	1,6721	0,7821	0,9405	0,0884	6,6342
6	3	0	16,32	1,653	2669	1680	5	0,1013	1,5887	0,5852	0,8936	0,0284	6,4504
2	0	3	4,78	0,821	9926	6427	5	0,1719	1,5444	0,9926	0,8687	0,1085	6,7528
4	4	4	11,00	1,302	6255	3595	0	0,1184	1,7399	0,4987	0,9906	0,9766	0,5126
6	3	4	16,44	1,656	3153	1858	5	0,1007	1,6970	0,5817	0,9545	0,0314	6,5049
5	3	4	13,83	1,503	4215	2541	0	0,1087	1,6588	0,4578	0,9444	0,6903	0,7119
4	2	4	11,17	1,359	5916	3681	0	0,1216	1,6072	0,5125	0,9150	1,0000	0,4275