Testing out multiple prior inclusions

cool people

Summary of the previous algorithm

Given an $n \times p$ matrix X and an $n \times q$ matrix Z, we hope to find function $\Omega : R^q \to R^{p \times p}$, which models the dependence of variables (represented as columns) in X as a function of extraneous covariates in Z. We do this by performing $n * p$ regressions, one for each observation and variable.

Suggested fix

For each of the n regressions across observations, specify a separate inclusion probability π_i.

Steps of the current algorithm

Below are the steps of the current algorithm that specifically highlight how the algorithm depends on the specification of the prior inclusion probability π:

1. The wrapper `R` function is `covdepGE.R`. It specifies the parameters for the grid search and then calls the workhorse function below for each of the p variables. For our experiment, I think it's okay if we just focus on the brute force `grid_search` for hyper-parameters and ignore other methods.
2. The workhorse `R` function that runs the algorithm is `cavi.R` inside the `R` folder, which returns α, μ, σ^2, the posterior parameters for each of the n spike-and-slab weighted regressions. These parameters have shapes $n \times p$ for each of n regressions and p. The main methods in `cavi.R` for the grid search are `grid_search_c` and `cavi_c`, which are both inside the `covdepGE.cpp` file.
3. The function `grid_search_c` is just a loop that goes through all the hyperparameter specifications.

4. The function `cavi_c` is the function that computes `cavi` updates for each of the n regressions. The specific part that is of interest to us is the update for `alpha`, specified in `alpha_update_c`.
5. Disregarding all the terms that do not depend on **prior inclusion probability** π, the update for α depends on π through $\alpha_1 = \log(\pi/(1-\pi))$, which is currently a `double`.

Proposed changes

Essentially, all the parts of the algorithm above stay the same in terms of the logic / algebra. However, we need to change / specify 2 pieces:

1. The data type of prior inclusion probability π needs to change from `double` to a n-vector. This entails also changing the (algebraic) operations that we do with π, which occur in `cavi_c` and `alpha_update_c` functions.
2. The specification of the grid search for π needs to change. Currently, in `grid_search_c`, candidate π values are specified as `double`. Now, we need to specify candidates for π as the n-vectors. Ideally, the idea is to initialize different n-vectors based on the clustering of the n-observations. This is, however, non-trivial, since we also need to chose clustering algorithm. Two simpler ways to test if this will work is to:
    - Randomly initialize candidate n-vectors, which would correspond to assuming different inclusion probabilities for each of the n regressions;
    - Assume oracle knowledge of the clusters (which we have in simulation settings), and assign different inclusion probabilities $\pi_i = v_{c(i)}$ for $i = 1, \ldots, n$, where $c(i)$ indicates cluster assignment for observation i.

Results

We'll compare results with the baseline under 3 different paradigms:

1) Every row i gets its own π_i
2) The rows are clustered with an oracle mapping $O(i)$ giving the true membership of each row, and each cluster gets its own π
3) The rows are clustered with a mapping learned from the data and each cluster gets its own π

	p	n	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
1	5.00	90.00	0.00	0.00	0.03	0.34	0.47	2.62
2	15.00	90.00	0.00	0.00	0.62	0.82	1.25	4.42
3	25.00	150.00	0.00	0.64	1.08	1.29	1.75	5.17
4	50.00	150.00	0.39	2.68	4.37	4.36	5.80	11.31

Table 1: False positives per sample - Normalized Z, Centered X

The baseline performance (one π value) is listed below. Lower false positives per sample is the goal, as well as keeping the false negatives per sample relatively constant.

% latex table generated in R 4.2.2 by xtable 1.8-4 package % Sat Nov 19 00:00:40 2022

% latex table generated in R 4.2.2 by xtable 1.8-4 package % Sat Nov 19 00:00:40 2022

	p	n	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
1	5.00	90.00	0.00	0.78	1.03	0.99	1.29	2.16
2	15.00	90.00	0.00	0.89	1.32	1.41	1.99	2.98
3	25.00	150.00	0.00	0.77	0.92	0.91	1.10	2.00
4	50.00	150.00	0.08	0.92	1.13	1.19	1.40	3.28

Table 2: False negatives per sample - Normalized Z, Centered X

Oracle mapping

```r
num_workers <- parallel::detectCores() - 1
doParallel::registerDoParallel(cores = num_workers)

oracle_results = simulation_list %>% map(function(setup){
    simulation_func(n_trials, setup, num_workers, normalize = TRUE,
                    pip_assgn = pluck(setup, 1, "interval"))
  })

save(oracle_results, file = "oracle_results.Rda")
```

% latex table generated in R 4.2.2 by xtable 1.8-4 package % Sat Nov 19 00:00:42 2022

% latex table generated in R 4.2.2 by xtable 1.8-4 package % Sat Nov 19 00:00:42 2022

	p	n	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
1	5.00	90.00	0.00	0.00	0.24	0.45	0.62	3.29
2	15.00	90.00	0.00	0.51	1.09	1.25	1.78	4.33
3	25.00	150.00	0.41	2.23	3.13	3.24	4.13	7.31
4	50.00	150.00	3.64	7.53	9.06	9.16	10.60	14.80

Table 3: False positives per sample - Oracle Clustering

	p	n	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
1	5.00	90.00	0.00	0.80	1.10	1.09	1.42	2.60
2	15.00	90.00	0.00	1.02	1.58	1.53	2.00	3.04
3	25.00	150.00	0.00	0.80	0.98	0.99	1.26	2.21
4	50.00	150.00	0.16	0.80	1.02	1.01	1.24	1.92

Table 4: False negatives per sample - Oracle Clustering

Unique per observation

```
unique_results = simulation_list %>% map(function(setup){
    simulation_func(n_trials, setup, num_workers, normalize = TRUE,
                    pip_assgn = 1:(setup %>% pluck(1, "interval") %>% length()))
  })

save(unique_results, file = "unique_results.Rda")
```

Clustering

We'll use hierarchical clustering, and assume that even if we don't know which observations belong to which cluster we at least know there are either 2, 3, or 6 clusters (in truth there are 3). In a sense, we are testing the model's robustness to misspecification of the number of clusters.

```
k_vec = c(2, 3, 6)

estimate_membership = function(setup, k) {
  setup %>% pluck(1, "Z") %>% dist() %>% hclust() %>% cutree(k = k)
}

cluster_results = simulation_list %>% map(function(setup){
    map(k_vec, function(k) {
```

```
        simulation_func(n_trials, setup, num_workers, normalize = TRUE,
                        pip_assgn = estimate_membership(setup, k))
      })
    })

  save(cluster_results, file = "cluster_results.Rda")
```

% latex table generated in R 4.2.2 by xtable 1.8-4 package % Sat Nov 19 00:00:50 2022

	p	n	clusts	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
1	5.00	90.00	2.00	0.00	0.00	0.22	0.47	0.67	3.18
2	5.00	90.00	3.00	0.00	0.00	0.28	0.47	0.62	3.38
3	5.00	90.00	6.00	0.00	0.00	0.33	0.48	0.69	3.11
4	15.00	90.00	2.00	0.00	0.35	0.82	1.07	1.58	4.69
5	15.00	90.00	3.00	0.00	0.44	1.02	1.21	1.57	5.13
6	15.00	90.00	6.00	0.00	0.68	1.22	1.41	1.82	4.44
7	25.00	150.00	2.00	0.44	1.71	2.52	2.71	3.59	6.24
8	25.00	150.00	3.00	0.93	2.18	2.98	3.20	4.13	6.67
9	25.00	150.00	6.00	1.00	2.62	3.50	3.54	4.46	6.97
10	50.00	150.00	2.00	2.80	6.20	7.68	7.76	9.08	13.40
11	50.00	150.00	3.00	3.25	7.08	8.81	8.78	10.20	14.12
12	50.00	150.00	6.00	4.52	8.37	10.14	10.14	11.60	15.63

Table 5: False positives per sample - Hierarchical Clustering

% latex table generated in R 4.2.2 by xtable 1.8-4 package % Sat Nov 19 00:00:50 2022

	p	n	clusts	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
1	5.00	90.00	2.00	0.00	0.77	1.08	1.08	1.49	2.60
2	5.00	90.00	3.00	0.00	0.78	1.08	1.09	1.47	2.60
3	5.00	90.00	6.00	0.00	0.77	1.12	1.09	1.46	2.60
4	15.00	90.00	2.00	0.00	1.03	1.57	1.53	2.00	3.56
5	15.00	90.00	3.00	0.00	1.02	1.56	1.51	2.00	3.20
6	15.00	90.00	6.00	0.00	1.04	1.57	1.52	2.00	3.02
7	25.00	150.00	2.00	0.00	0.79	0.96	1.00	1.21	2.21
8	25.00	150.00	3.00	0.00	0.80	0.97	0.99	1.25	2.21
9	25.00	150.00	6.00	0.00	0.82	0.99	0.99	1.21	2.21
10	50.00	150.00	2.00	0.28	0.82	1.08	1.06	1.22	2.00
11	50.00	150.00	3.00	0.21	0.80	1.06	1.04	1.24	2.00
12	50.00	150.00	6.00	0.16	0.79	1.02	1.00	1.23	1.88

Table 6: False negatives per sample - Hierarchical Clustering