NCAR | National Center for
UCAR | Atmospheric Research

regClimateChem: An R Package for Data Driven Variable Selection Applied to Atmospheric Carbon Monoxide

William Daniels
Dorit Hammerling
Rebecca Buchholz

NCAR Technical Notes
NCAR/TN-562+STR

National Center for
Atmospheric Research
P. O. Box 3000
Boulder, Colorado
80307-3000
www.ucar.edu

National Science Foundation
NCAR IS SPONSORED BY THE NSF

NCAR TECHNICAL NOTES

http://library.ucar.edu/research/publish-technote

The Technical Notes series provides an outlet for a variety of NCAR Manuscripts that contribute in specialized ways to the body of scientific knowledge but that are not yet at a point of a formal journal, monograph or book publication. Reports in this series are issued by the NCAR scientific divisions, serviced by OpenSky and operated through the NCAR Library. Designation symbols for the series include:

EDD – Engineering, Design, or Development Reports
Equipment descriptions, test results, instrumentation, and operating and maintenance manuals.

IA – Instructional Aids
Instruction manuals, bibliographies, film supplements, and other research or instructional aids.

PPR – Program Progress Reports
Field program reports, interim and working reports, survey reports, and plans for experiments.

PROC – Proceedings
Documentation or symposia, colloquia, conferences, workshops, and lectures. (Distribution maybe limited to attendees).

STR – Scientific and Technical Reports
Data compilations, theoretical and numerical investigations, and experimental results.

The National Center for Atmospheric Research (NCAR) is operated by the nonprofit University Corporation for Atmospheric Research (UCAR) under the sponsorship of the National Science Foundation. Any opinions, findings, conclusions, or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

National Center for Atmospheric Research
P. O. Box 3000
Boulder, Colorado 80307-3000

NCAR/TN-562+STR
NCAR Technical Note

2020-05

regClimateChem: An R Package for Data Driven Variable Selection Applied to Atmospheric Carbon Monoxide

William Daniels
Department of Applied Mathematics and Statistics
Colorado School of Mines, Golden, CO

Dorit Hammerling
Department of Applied Mathematics and Statistics
Colorado School of Mines, Golden, CO

Rebecca Buchholz
Atmospheric Chemistry Observation & Modeling Laboratory
National Center for Atmospheric Research, Boulder, CO

NCAR Laboratory
NCAR Division

NATIONAL CENTER FOR ATMOSPHERIC RESEARCH
P. O. Box 3000
BOULDER, COLORADO 80307-3000
ISSN Print Edition 2153-2397
ISSN Electronic Edition 2153-2400

`regClimateChem`: An `R` Package for Data Driven Variable Selection Applied to Atmospheric Carbon Monoxide

William Daniels*[1], Dorit Hammerling †[1], and Rebecca Buchholz‡[2]

[1]Department of Applied Mathematics and Statistics, Colorado School of Mines, Golden CO
[2]Atmospheric Chemistry Observation & Modeling Laboratory, National Center for Atmospheric Research, Boulder CO

May 15, 2020

Abstract

Carbon monoxide (CO) is a major pollutant, impacting air quality and contributing to the greenhouse effect. Buchholz et al. [1] used a multiple linear regression model to link CO anomalies in the atmosphere to variability in the climate. Measurements of total column CO were retrieved from the Measurements Of Pollution In The Troposphere (MOPITT) instrument onboard the Terra satellite. Anomalies generated by these measurements are used as the response variable in this model. Climate mode indices represent regional variability in the climate, and these indices at various time lags are used as the predictor variables. Buchholz et al. [1] performed this analysis in `MATLAB` using serial algorithms in non-functionalized scripts. Simonson et al. [2] refactored the `MATLAB` codebase, improving both code interpretability and generality and adding parallelization options. In this technical note, we present further updates to the codebase used in the atmospheric CO analysis from Buchholz et al. [1]. These updates stem from four primary objectives. First, transfer the codebase into an `R` package that is available to researchers without a `MATLAB` license. Second, allow for a single climate index to appear in the model multiple times with different lag values. Third, implement an exhaustive variable selection algorithm. Finally, improve upon the non-exhaustive variable selection algorithm implemented in the `MATLAB` code. These updates are discussed, and the accuracy and timing of the new variable selection algorithms are compared.

Keywords: carbon monoxide, climate chemistry, statistical models, multiple linear regression, variable selection, exhaustive search, genetic algorithm, stepwise selection, timing study

*wdaniels@mines.edu
†hammerling@mines.edu
‡buchholz@ucar.edu

Contents

1 Introduction

1.1 Carbon Monoxide Variability

Carbon monoxide (CO) is a major pollutant. In the presence of sunlight, CO is a precursor to tropospheric ozone, a gas with negative health impacts for both humans and vegetation. Furthermore, CO is a primary sink for hydroxyl radicals in the atmosphere that would otherwise react with methane and carbon-containing compounds. Therefore, CO in the atmosphere can indirectly increases the concentration of methane and ozone, two powerful greenhouse gasses [3].

Measurements of total column CO used in Buchholz et al. [1] and this study are retrieved from the Measurements Of Pollution In The Troposphere (MOPITT) instrument onboard the Terra satellite. Data from the MOPITT instrument is publicly available at http://terra.nasa.gov/about/terra-instruments/mopitt. To reduce systematic and random error, we select daytime, land-only retrievals from the thermal infrared product. See Buchholz et al. [1] for details. CO anomalies are created by subtracting a spatial and climatological average of monthly total column CO from 2001 to 2016 from monthly averaged values. Figure 1 shows the measured CO total column and the generated anomaly in Maritime Southeast Asia used in Buchholz et al. [1]. The regions studied in this report will be discussed in detail later in this section.

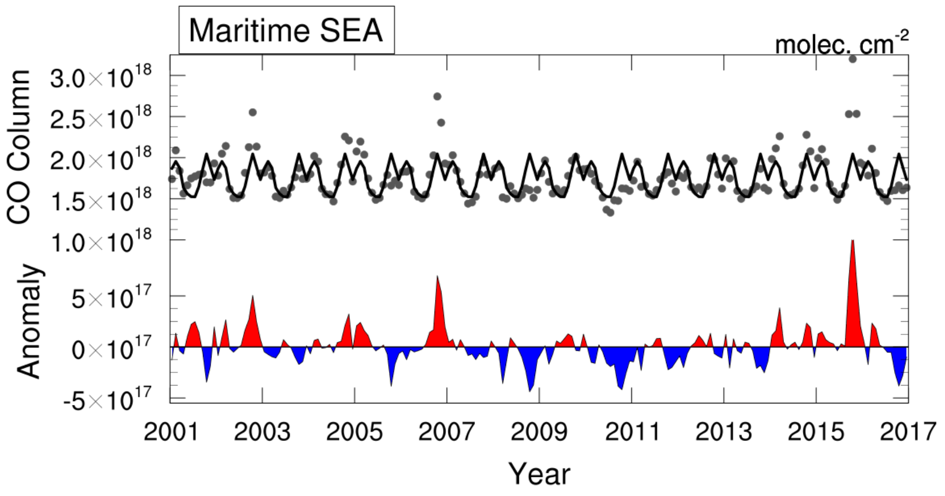


Figure 1: Time series of total column CO and generated anomaly in Southeast Asia. The grey dots are total column CO measurements from MOPITT, and the black line is the seasonal trend in total column CO. CO anomalies are created by subtracting a spatial and climatological average of monthly total column CO from 2001 to 2016 from monthly average values.

Atmospheric CO is directly produced by incomplete combustion and indirectly produced from hydrocarbon oxidation. These reactions are abundant during large burn events, making fires a major source of atmospheric CO [4]. In fact, fires are the primary source of CO variability in the Southern Hemisphere [5]. Therefore, atmospheric CO serves as an easily measurable proxy for fires during fire season.

Table 1: Climate mode indices used in this study with links to their sources.

Climate Mode	Associated Index	Source
ENSO	Niño 3.4	www.cpc.ncep.noaa.gov/data/indices/
IOD	Dipole Mode Index (DMI)	stateoftheocean.osmc.noaa.gov/sur/ind/dmi.php
TSA	Tropical South Atlantic (TSA)	www.cpc.ncep.noaa.gov/data/indices/
AAO	Southern Annular Mode (SAM)	www.cpc.ncep.noaa.gov/products/precip/ CWlink/daily_ao_index/aao/aao.shtml

The intensity and size of fires are closely related to the amount, type, and dryness of available fuel. These factors respond closely to variability in the climate [6]. Climate modes, such as the El Niño-Southern Oscillation, are often used to predict regional climate variability, like rainfall and drought [7, 8]. Climate indices, such as the Nino 3.4 index, are useful metrics for describing the aperiodic variability due to climate modes. Buchholz et al. [1] used indices from four major climate modes to model atmospheric CO anomalies in the Southern Hemisphere. The four indices used in Buchholz et al. and this study are listed in Table 1, as well as the source of each index. Climate mode indices are made publicly available by NOAA. Figure 2 shows the climate indices over the dates of interest in our study.

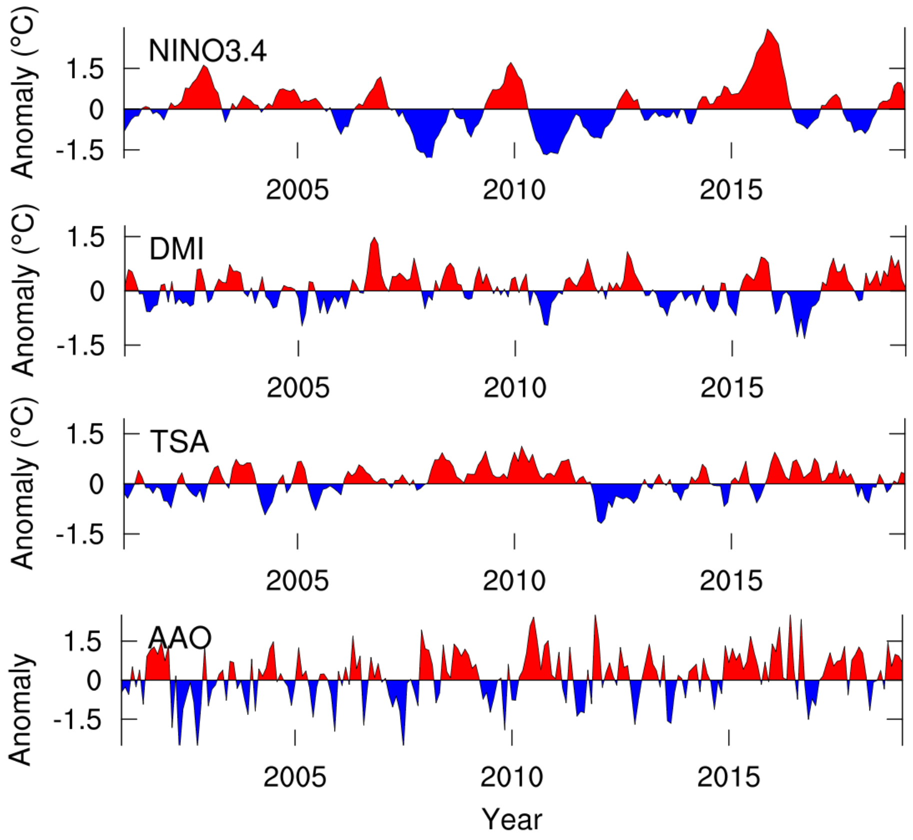


Figure 2: Climate indices over the dates of interest in our study. Positive modes are plotted in red, and negative modes are plotted in blue.

The CO anomalies in Buchholz et al. [1] were aggregated into seven biomass burning regions, and a separate model was created for each region. Figure 3 (a) shows these response regions over the average total column CO from 2001 to 2016. Figure 3 (b) shows the climate indices used in this study over the standard deviation of total column CO from 2001 to 2016 [1].

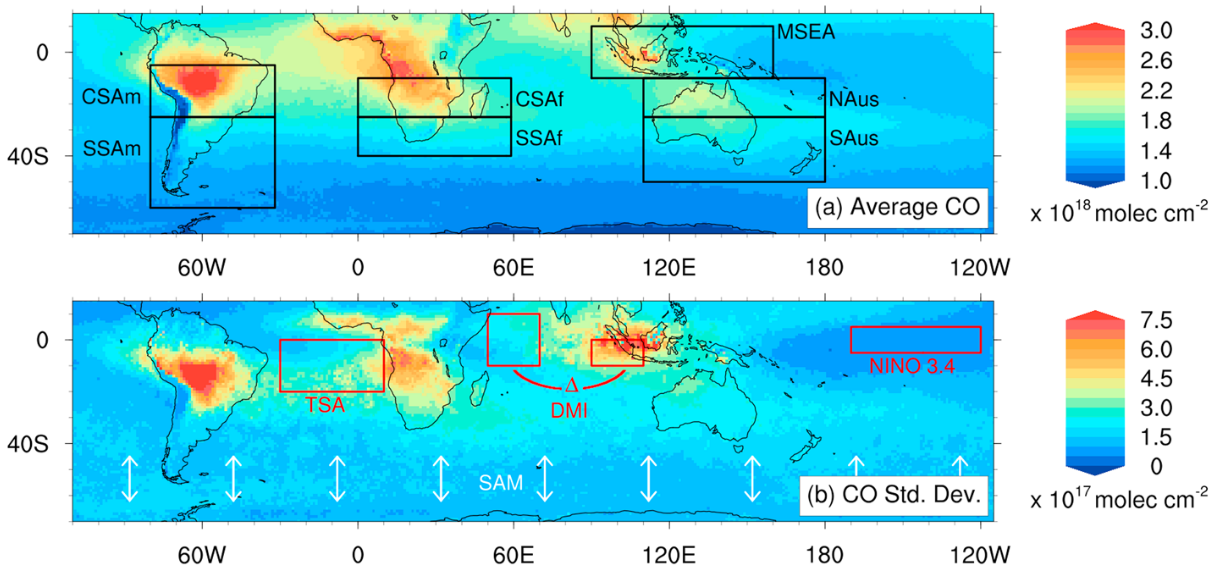


Figure 3: Boxes in (a) define the seven biomass burning regions used in this study. Boxes in (b) define regions used to create the TSA, DMI, and NINO climate mode indices. The white arrows in (b) depict shifting of the westerly winds associated with the SAM climate mode index. The map in (a) depicts average total column CO from September to December between 2001 and 2016. The map in (b) depicts the corresponding standard deviation to the averages in (a). This map was taken from the Buchholz et al. [1] manuscript.

1.2 Statistical Model

A multiple linear regression model with interactions was used in Buchholz et al. [1] to model CO anomalies. The de-seasonalized CO anomaly in each region was used as the response, and the four climate mode indices and their interactions were used as the covariates. Since these models focus on prediction, the climate mode indices were lagged at values ranging from from 1 to 8 months.

For a given region, the form of the regression model with interactions is shown in equation 1.

$$CO(t) = \mu + \sum_{k} a_k \cdot \chi_k(t - \tau_k) + \sum_{i,j} b_{ij} \cdot \chi_i(t - \tau_i) \cdot \chi_j(t - \tau_j) \qquad (1)$$

In equation 1, $CO(t)$ is the CO anomaly at time t, μ is a constant mean displacement, a_k and b_{ij} are coefficients, χ are the climate indices, τ is the lag value for each index in months, and k,i, and j iterate over the number of climate modes used in the analysis. Each climate index can take a different lag value, but once a lag is selected for a main effect, all interactions of that term use the same lag value.

The variable selection and analysis in Buchholz et al. [1] was performed in `MATLAB` using serial algorithms implemented in non-functionalized scripts. A series of nested `for` loops was used to iterate through all possible combinations of lag values. At each combination, stepwise selection was used to find the best model for that set of lag values. After all sets of lag values had been considered, the resulting best models would be directly compared. The best of these models (and the corresponding lag values) would then be selected as the predictive model for that region.

The Bayesian Information Criterion (BIC) was used in Buchholz et al. [1] to perform variable selection, as it selects models with a focus on prediction. The BIC is defined in equation 2.

$$\text{BIC} = \ln(n)k - 2\ln(\hat{L}) \tag{2}$$

In equation 2, n is the number of observations in the dataset used to train the model, k is the number of covariates in the model, and $\hat{L}$ is the maximized value of the likelihood function for the model.

These models are intended for prediction, so it is desirable to have low model complexity, or a small number of covariates. This way the models capture the general trend in the climate mode data without overfitting to the noise. However, we do not want our models to be so simple that they ignore important features in the climate mode data. Therefore, we use the BIC as our variable selection criterion, as it balances both fit and model complexity with an emphasis on minimizing complexity.

A smaller BIC value corresponds to a better model. Therefore, the first term in the BIC equation can be though of as a punishment for model complexity, as it grows with the number of covariates. In fact, the $\ln(n)$ coefficient makes the BIC a particularly harsh criterion in terms of model complexity, as n is often quite large. The second term in the BIC equation can be thought of as a measurement of fit. Comparatively, models with larger maximized likelihood values fit the data better than models with smaller maximized likelihood values. Therefore, a good fit makes the second term in the BIC equation large, reducing the overall BIC value for that model.

Note that the BIC value of a given model is meaningless on its own, as the weight between complexity and fit is arbitrary. The BIC serves only as a comparison between different models.

1.3 Codebase Refactoring

Simonson et al. [2] refactored the initial `MATLAB` codebase, yielding three significant outcomes. First, data structures were implemented to increase code interpretability. Second, the codebase was generalized to work with more flexible periods of interest, more extensive lag space matrices, and any number of climate indices. Third, the codebase was parallelized using the `MATLAB Parallel Computing Toolbox`.

The refactored code in Simonson et al. [2] eliminates the need for nested `for` loops during the variable selection process. Instead, variable selection is broken up into the following steps. First, a range of lag values for each index is specified by the user. Each possible combination of lags is then pre-computed and saved in a grid, which is called the lag space matrix. Each entry of the lag space matrix contains one possible lag value for each climate mode index, which is called a lagset. A single `for` loop can then be used to consider all combinations of lag values by iterating through the lag space matrix, making parallelization possible. Within this loop, variable selection is performed for each lagset using stepwise selection with BIC as the criterion. After considering all lagsets, the best model and corresponding lagset is selected as the predictive model for that region.

This refactoring provided vast improvements in both usability and efficiency of the `MATLAB` codebase. However, there were still some limitations in the codebase left unaddressed:

- `MATLAB` is a fairly expensive software package. Researchers wishing to verify or expand upon the analysis in Buchholz et al. [1] might not have access to `MATLAB` or the `MATLAB Parallel Computing Toolbox`.

- The regression algorithm only searched for models containing one lag per climate mode. In other words, the models could only accommodate one main term per climate mode. One way of exploring more complex relationships between climate and CO would be to include multiple lags of a single mode. This functionality was not built into the Simonson et al. [2] codebase.
- Stepwise selection was the only regression algorithm implemented. This approach works well in many applications. However, stepwise selection is not an exhaustive variable selection algorithm, meaning that it does not explicitly compute a criterion value for every possible model. Therefore, it may not always find the best model. An exhaustive algorithm, on the other hand, will always find the best model. In small problems with few predictor variables, the computational benefit of a non-exhaustive search over an exhaustive search is not significant.
- Stepwise selection is often able to find good models, but there are other non-exhaustive search algorithms that do a better job. For instance, genetic search algorithms are often able to find better models than stepwise selection, but they do not scale as well to larger parameter spaces. The Simonson et al. [2] codebase only contained a stepwise selection algorithm, without providing other non-exhaustive search options.

To address these issues, the codebase has again been refactored and updated. The following items were implemented to address the limitations listed above. These items will be discussed in detail in Section 2 of this technical note.

- The code was rewritten in the open source `R` package `regClimateChem`.
- The lag space matrix was generalized to include multiple lags of a single climate mode.
- An exhaustive search option was added.
- An additional non-exhaustive search option was added. The user can now select between a genetic algorithm and stepwise selection.

The remainder of this report is as follows. Section 2 describes improvements made to the codebase since the Simonson et al. [2] refactoring. Section 3 discusses the default hyperparameter values for each variable selection algorithm implemented in the `regClimateChem` package. Section 4 includes a quick note on the scalability of the variable selection algorithms in `regClimateChem`. Section 5 compares the accuracy of the new variable selection algorithms. In this report, model accuracy refers to each algorithm's ability to find the best possible model for a given lagset. Section 6 compares the run times of the variable selection algorithms. In Section 7, the best model in each response region is reported. We conclude our findings in Section 8. Finally, an updated description of the `regClimateChem` configuration file is found in appendix A, and instructions for downloading the `regClimateChem` `R` package are found in appendix B.

2 Codebase Updates

2.1 `regClimateChem` `R` Package

The `MATLAB` codebase written for Buchholz et al. [1] and refactored in Simonson et al. [2] has been rewritten as the `R` package `regClimateChem`. This package is available on GitHub. See appendix B for instructions on downloading.

The `R` package utilizes several other open source `R` packages:

- *foreach*. Available on CRAN. Used for executing loops in parallel.
- *doParallel*. Available on CRAN. Serves as the parallel backend for the foreach package.
- *gtools*. Available on CRAN. Contains various tools for data manipulation. Required for the combinations function, which is used to create the lag space matrix.
- *glmulti*. Available on CRAN. Package containing exhaustive and genetic regression algorithms. See [9] for details.
- *rJava*. Available on CRAN. Required for *glmulti*. Note that rJava requires Java and the Java Runtime Environment.
- *MASS*. Available on CRAN. Required for stepwise selection.

2.2 Multiple Lags of One Mode

In the `MATLAB` code, the regression algorithm only searched for models containing at most one lag per climate mode. In other words, each climate mode could only appear once as a main term in the generated models. These models are able to explain much of the CO variability, but more complex models (or models with more terms) have potential value. For instance, the ENSO 3.4 index is larger in magnitude and less noisy than the other indices. Models containing two ENSO terms lagged at different values might be better able to predict CO concentrations than models that only contain one ENSO term. This is because ENSO may have different lags that separately impact vegetation growth and fuel dryness. The combination of these two time scales could provide additional predictive power over models that only take into account a single lag.

The new `R` code has been generalized to allow for multiple lagged terms of a single climate mode. This was done via generalizations to the lag space matrix. In the `MATLAB` code, the lag space matrix was generated by simply taking the Cartesian product of the user specified lag limits. The remainder of this section describes the generalized algorithm implemented in `regClimateChem`.

First, the user specifies: 1) the number of lags allowed per climate mode, and 2) the minimum and maximum lag values allowed for each climate mode. These values are set in the configuration file, which is described in detail in Appendix A. At this time, all of the lags of a given climate mode have the same minimum and maximum values. In future package iterations, there could be options for restricting these values for certain lags. However, at this time we want the regression algorithms to search for the best possible model with as few restrictions as possible. With these values defined, the following function call is made for each climate mode:

```
combination.list <- combinations(v = min.lag:max.lag,
                                 n = max.lag - min.lag + 1,
                                 r = num.terms)
```

Here `min.lag` and `max.lag` are the minimum and maximum lag values allowed for each climate mode and `num.terms` is the number of lags allowed per climate mode. These values are used to compute the following quantities. The source vector, `v`, is a vector of lag values starting with the minimum lag value and ending with the maximum lag value. The length of the source vector, `n`, is simply the number of lag values present in the source vector. The length of the target vector, `r`, is the number of lag values present in the target vector. The target vector is the desired output of the `combinations` function. That is, the `combinations` function will create all possible combinations of length `r` using values from the source vector. In this way, the `combinations` function defines all possible lag combinations for a given climate mode.

For instance, if the user specifies that only one lag is allowed per climate mode (ie. the length of the target vector is 1), then by default there are 8 choose 1 possible combinations. In this case, the output of the `combinations` function would be:

$$[1 \quad ; \quad 2 \quad ; \quad \cdots \quad ; \quad 8]$$

If instead the user specifies that two lags are allowed per climate mode (ie. the length of the target vector is 2), then by default there are 8 choose 2 possible combinations. In this case, the output of the `combinations` function would be:

$$[1,2 \quad ; \quad 1,3 \quad ; \quad \cdots \quad ; \quad 2,3 \quad ; \quad 2,4 \quad ; \quad \cdots \quad ; \quad 6,7 \quad ; \quad 6,8 \quad ; \quad 7,8]$$

It is important to emphasize that we do not want models containing multiple terms of a single climate mode with the same lag value. These terms would be identical, resulting in perfectly linearly dependent covariates. Therefore, there are no target vectors containing identical lag values, such as $1,1$ or $2,2$. These vectors would result in a single index being lagged twice at the same value.

After generating a combination list for each climate mode, the lists are spliced together in such a way that all possible combinations of lag values are represented. This is achieved with the following loop structure:

```
for (a in 1:nrow(combination.list[[1]])){
   for (b in 1:nrow(combination.list[[2]])){
     for (c in 1:nrow(combination.list[[3]])){
       for (d in 1:nrow(combination.list[[4]])){

         this.row <- cbind(t(combination.list[[1]][a,]),
                          t(combination.list[[2]][b,]),
                          t(combination.list[[3]][c,]),
                          t(combination.list[[4]][d,]))

         lagset.matrix <- rbind(lagset.matrix, this.row)

       }
     }
   }
 }
```

In this code segment, each element in the `combination.list` data structure is the combination list for a given climate mode. This generalized method pregenerates the lag space matrix, which is more computationally intensive up front but lends itself well to parallelization when doing variable selection. When each lagset is pre-computed, a single loop can be used to iterate through all possible lag value combinations. Within this loop, each iteration is independent of the others, making it suitable for parallelization. This is a desirable feature, as scenarios with many predictor variables or many observations will quickly become infeasible on a single processor.

Note, however, that while pregeneration of the lag space matrix is feasible for this carbon monoxide application, it will become increasingly costly for applications with more predictor variables. For instance, considering four climate mode indices and lag values from one to eight months, the lag space matrix is a 4096×4 matrix taking up 65 KB. However, with five climate mode indices (or two lags of a one mode) and lag values from one to eight months, the lag space matrix becomes 14336×5 matrix taking up 286 KB. So for applications with hundreds of predictor variables, pregeneration of the lag space matrix would require massive amounts of memory. This problem is not addressed in this codebase update, as we do not anticipate that the CO modeling application will require an unreasonable amount of covariates. Even including one or two dozen covariates in the predictive CO models would be possible with this pregeneration method.

Also note that while `regClimateChem` does not impose a limit on the number of lags allowed for a given index, it may not be wise to use more than 2 or 3. This is because climate mode indices are time series data, meaning that they have an auto-correlation structure. Including too many lags of a single index could result in highly correlated covariates, which is undesirable in regression modeling. The optimal number of lags for a single index is an ongoing research topic.

2.3 Exhaustive Search Option

In the `MATLAB` code, stepwise selection was the only variable selection algorithm implemented. This is a non-exhaustive search, meaning that it does not check every possible model. The benefit of non-exhaustive searches (like stepwise selection) is that significant computation time can be saved for large problems (like a large parameter space). The drawback is that for larger problems, non-exhaustive methods are increasingly likely to miss the best possible model. However, for smaller problems (like the CO application with only four covariates), the difference in computation time between an exhaustive and non-exhaustive search is negligible. Therefore, there is no drawback to using an exhaustive search for the CO application with only four covariates.

In `regClimateChem`, an exhaustive search algorithm is implemented via the *glmulti* package. This algorithm considers every possible model for each lagset. It computes the BIC value for each model and selects the model with the best criterion value. Therefore, the exhaustive search always finds the best possible model. To perform an exhaustive search, the user simply specifies this in the configuration script. Section 3 discusses the default parameter values of the exhaustive search. Sections 5 and 6 analyze the accuracy and computation time of the new variable selection algorithms.

2.4 Non-Exhaustive Search Options

With more covariates, however, a non-exhaustive search option is often required. This is because the number of possible models grows exponentially with the number of covariates. As in the `MATLAB` codebase, `regClimateChem` provides a stepwise selection option implemented via the *MASS* package. This algorithm operates under the same general principles as the stepwise algorithm in `MATLAB`. The algorithm begins with a base model. It then begins an iterative process where it checks all models created by either adding or removing a single covariate. If one of these models improves the information criterion, then that model is selected and the algorithm continues. If none of the new models improve the information criterion, then the algorithm stops and the current model is used. Section 3 discusses the default parameter values used for both the stepwise algorithm implemented in `regClimateChem` as well as the Simonson et al. [2] `MATLAB` codebase.

In addition to stepwise selection, `regClimateChem` provides a genetic (or stochastic) variable selection algorithm implemented via the *glmulti* package. Sections 5 and 6 will show that the genetic algorithm is better than the stepwise algorithm at finding the best model in the cases studied here, but is significantly slower. The user can decide which algorithm to use based on the size of their parameter space. The remainder of this subsection describes the genetic algorithm [9].

The genetic algorithm maintains a "population" of models that is smaller than the total number of possible models. Future "generations" of the population are produced by modifying the "parent" models from the current generation to create "children" models in the next generation. There are three different types of modification implemented in the *glmulti* package:

1. *Asexual Reproduction.* The child model is a product of a single parent model. Each possible predictor term has a chance of "mutating." In this CO modeling application, the climate mode indices at a given lag value and their interactions are the predictor terms. If a term mutates, then its role in the child model is the opposite of its role in the parent model. That is, if a mutating term was included in the parent model, then it will not be included in the child, and vise versa. If a term does not mutate, then it has the same role in the child model as in the parent model.

2. *Sexual Reproduction.* Two children models are produced from two parent models. The terms included in the children model are selected randomly from both parents. After the children models are formed, they both undergo asexual reproduction. That is, each term has a chance of mutating after the parent models have been combined.

3. *Immigration.* A child model produced by immigration is completely randomized. Each possible predictor term is either included or not included in the child model with equal probability.

The genetic algorithm ensures that future generations are more likely to contain models with better criterion values. This is achieved by biasing the selection of parent models. Models in the current generation with a better criterion value are more likely to be selected as parent models when creating the next generation. That is, better models are more likely to be used to create the next population of models [9]. This way the population of models converges to one containing the best possible model.

The genetic algorithm stops after a set number of population iterations without significant improvement. Specifically, it examines the best value of the criterion in the population and the average value of the criterion in the population. When both of these values stop improving by more than a set threshold, the algorithm stops.

The degree to which the three modification methods alter the children models are ranked from largest alteration to smallest alteration as follows: immigration, sexual reproduction, asexual reproduction. The immigration method in particular is meant to drastically change the child model. This introduces vastly different models into subsequent populations, which helps to prevent the algorithm from getting stuck at local extrema of the selection criteria.

The rate at which each of these three modification methods occur can be altered. In `regClimateChem`, the rates are set so that the genetic algorithm is more conservative. That is, the immigration rate is higher than the default and the stopping criterion is more strict. This was done to ensure that the genetic algorithm finds the best possible model in most situations, at the expense of rapid convergence. Section 3 discusses the default parameter values of the genetic algorithm in `regClimateChem`.

Sections 5 and 6 provide a detailed comparison of the genetic algorithm implemented in R, the exhaustive search implemented in R, the stepwise search implemented in R, and the stepwise search implemented in the Simonson et al. [2] `MATLAB` codebase. Specifically, Section 5 compares the ability of these algorithms to find the best model, and Section 6 compares the runtime of each algorithm.

3 Default Parameter Settings

3.1 Exhaustive Search

The exhaustive search in `regClimateChem` is implemented using the *glmulti* package. The exhaustive search in the *glmulti* package simply considers all non-redundant model formulas. It does not approximate the exhaustive search with a branch-and-bound algorithm or a simulated-annealing algorithm. Therefore, there are no tuning parameters associated with this search technique.

3.2 Stepwise Selection

Stepwise selection in `regClimateChem` is implemented using the `stepAIC` function from the *MASS* package. The following list defines the `stepAIC` parameters, their default values, and the values used in `regClimateChem`.

- *object.* This is the initial model that stepwise selection will be performed on. In `regClimateChem`, the full model is supplied as the `stepAIC` *object*.
- *scope.* This defines the range of models examined in the stepwise search. The user can provide upper and lower lists, containing terms that will always be included or that will never be included in the model, respectively. If no user input is supplied, then the *object* parameter will be used as the upper limit on the models. In `regClimateChem`, no additional *scope* parameters are provided. Since the full model is supplied as the *object* parameter, this means that no restrictions are placed on the models considered during the stepwise selection.
- *scale.* This is used in the definition of the AIC statistic for selecting the models. The default value of 0 means that the error variance is estimated using maximum likelihood. `regClimateChem` uses the default value.
- *direction.* This determines the mode of stepwise search. It can be "both", "backward", or "forward." The default setting is "both," indicating both forward and backward stepwise selection. `regClimateChem` uses the default value.
- *steps.* This defines the maximum number of steps to be considered. It is typically used to stop the process early. The default is 1000. In most cases, 1000 steps is far more than necessary to converge on a model. Therefore, `regClimateChem` uses the default value.
- *k.* This is the coefficient for the number of degrees of freedom used in the penalty. The default is k = 2, which gives the AIC. `regClimateChem` uses k = log(n), where n is the number of observations. This gives the BIC.

Stepwise selection in the `MATLAB` codebase is implemented using the `stepwiselm` function. The following list defines the `stepwiselm` parameters, their default values, and the values used in the `MATLAB` codebase.

- *modelspec.* This indicates the starting model using in the stepwise selection. In the `MATLAB` codebase, the model with all main terms and no interactions is used as the starting model.
- *criterion.* This specifies the information criterion used in the stepwise selection. By default, `MATLAB` uses the p-value for an F-test of the change in the sum of squared error that results from adding or removing the term. However, the BIC is used in the `MATLAB` codebase, as we are interested in models that focus on prediction.

- *PEnter.* This defines the threshold criterion value for adding or removing a term. The default value for the BIC is 0. In other words, if adding or removing a term decreases the BIC at all, then that model is selected. The `MATLAB` codebase uses the default value.

- *upper.* This defines the largest set of terms to be included in the model. The default setting is to include interaction terms. The `MATLAB` codebase uses the default value.

The only major difference in stepwise algorithms is the starting model. In `regClimateChem`, the full model with interactions is used as the starting model. In the `MATLAB` codebase, the model with main terms and no interactions is used as the starting model. Since main terms are rarely removed in this application, there is little difference between these two starting points. The `regClimateChem` stepwise will simply remove interactions until arriving at the best model, while the `MATLAB` codebase will add interactions until arriving at the best model.

It is important to note, however, that some of the `MATLAB` source code is not publicly visible. As a result, there are potentially other differences beneath the surface of these two algorithms that are not discussed here.

3.3 Genetic Algorithm

The genetic algorithm in `regClimateChem` is implemented using the *glmulti* package. All genetic algorithm parameter values that are changed from their *glmulti* defaults are listed below. If a parameter value is not listed here, then it is set to the default *glmulti* value. See the *glmulti* documentation for a more complete list of parameters [9].

- *marginality.* This dictates whether strong hierarchy is to be enforced. Strong hierarchy says that if a term is included in an interaction, then its main effect must also be included in the model. By default, this parameter is set to FALSE, but in `regClimateChem` it is set to TRUE.

- *popsize.* This defines the number of models maintained in the population. The default value for *popsize* is 100. However, in the CO application, there are only 113 possible models. Therefore, a smaller than default value is used in `regClimateChem`, as this was found to decrease computation time with minimal loss of accuracy. `regClimateChem` uses a value of 40. Users with larger parameter spaces should increase this value.

- *confsetsize.* This defines the number of models reported in the confidence set of models. The default value is 100. However, the CO application is only interested in the best model for each lagset, so only one model is recorded. In other words, `regClimateChem` uses a *confsetsize* value of 1.

- *conseq.* This defines the number of iterations without improvement in the stopping criterion required to stop the algorithm. The default value is 5. In `regClimateChem`, a value of 6 is used, as this helps ensure that a global extrema is found.

- *imm.* This is the rate of immigration in the genetic algorithm. The default value is 0.3, meaning that 3 in 10 models are produced by immigration. In `regClimateChem`, a value of 0.35 is used, as this helps ensure that a global extrema is found.

Note that while some testing was performed to select these parameter values, a more thorough study is ongoing. This ongoing study seeks to optimize all *glmulti* parameters for the atmospheric CO application via an extensive parameter space search.

4 Scalability Considerations

It is important to include a few notes on the scalability of these algorithms before comparing their accuracy and timing in Sections 5 and 6.

- First, this technical note only examines the four and five covariate cases. The results presented for these cases do not necessarily generalize to any number of covariates.
- Stepwise selection is more scalable than heuristic algorithms (like the genetic algorithm). For instance, the number of models considered in both pure forward and backward selection is only $1+p(p+1)/2$, where p is the number of covariates. This is much lower than the 2^p models considered by best subset selection, an exhaustive search [10]. With the genetic algorithm, there is no deterministic formula for the number of models that must be considered. As a result, there is no guarantee on its scalability. Therefore, stepwise selection should likely be used over the genetic algorithm and the exhaustive search for large parameter spaces.
- The stepwise selection algorithm in `MATLAB` is slower than the stepwise selection algorithm in `R` for the four and five covariate case. Again, this may not be the case for larger parameter spaces. Furthermore, this difference could be due to extra features in the `MATLAB` algorithm that are not publicly visibly. For instance, there might be advanced parallelization features built into the `MATLAB` algorithm that make it more appropriate for larger parameter spaces than the algorithm in `R`. Furthermore, the `MATLAB` algorithm might search more of the parameter space than the `R` algorithm. This could potentially explain why the `MATLAB` algorithm takes longer to run than the `R` algorithm, but often finds better models. The timing and accuracy of these two algorithms will be discussed in more detail in Sections 5 and 6.
- The scalability of these algorithms is an ongoing research topic. Future work will study their performance on HPC systems with larger parameter spaces.

5 Accuracy Comparison

In this section, we analyze each variable selection algorithm's ability to find the best model for a given lagset, or the model with the lowest BIC value. The BIC is used as the selection criterion because it is well suited to prediction, as discussed in Section 1.

To evaluate each algorithm's ability to find the best models, we compare them to the exhaustive search. Recall that the exhaustive search will always find the best model, making it a natural benchmark for comparison. The comparison is performed as follows:

1. Start with the first lagset in the lag space matrix. Perform variable selection at these lag values using the exhaustive, genetic, and stepwise algorithms. Each of these three algorithms will come up with their own "best" (lowest BIC) model **for that given lagset**.
2. Compare the BIC of the best model found by the non-exhaustive options (genetic and stepwise) to the exhaustive search. If the BIC values agree, then the non-exhaustive option was able to find the best model for that particular lagset. If the BIC values do not agree, then the non-exhaustive option failed to find the best possible model for that particular lagset.
3. Repeat this process for every lagset in the lag space matrix.

Figure 4 provides a visualization of this comparison using fictitious BIC values. Notice that for some lagsets, both the genetic and stepwise algorithms agree with the exhaustive search (ie. lagset “1,1,1,1”). For other lagsets, however, only one of the non-exhaustive option agrees (ie. lagset “1,1,1,2”). Occasionally, neither non-exhaustive option will agree with the exhaustive search (ie lagset “1,1,1,3”).

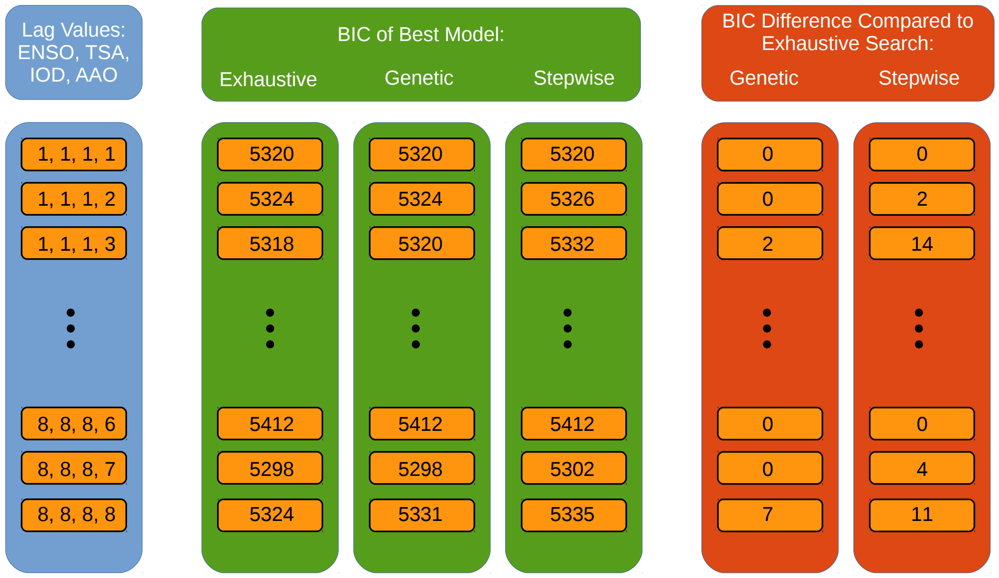


Figure 4: Visualization of model comparison.

The following subsections present plots that summarize the BIC differences for both the four and five covariate cases. Note that two stepwise selection algorithms are considered. The first being the stepwise selection algorithm implemented in the Simonson et al. [2] `MATLAB` codebase. The second being the stepwise selection algorithm implemented in `regClimateChem`.

The plots can be interpreted as follows. The horizontal axis gives the index of each lagset, ranging from 1 to 4096 in the four covariate case and from 1 to 14336 in the five covariate case. These lagsets have been sorted such that the BIC of the best model found by the exhaustive search is increasing as you move from left to right. In other words, the predictive power of the best model is degrading as you move from left to right. The vertical axis gives the BIC value found by each search algorithm for a particular lagset. The red and blue points are the BIC values for the best models found by the exhaustive search and the genetic algorithm, respectively. The orange and green points are the BIC values for the best models found by the stepwise selection algorithm implemented in `R` and the Simonson et al. [2] `MATLAB` codebase, respectively.

Recall that the exhaustive search always finds the best models and that a low BIC value corresponds to a good model. Therefore, the orange, blue, and green points will always be at or above the BIC value of the red points. When one of the non-exhaustive algorithms agrees with the exhaustive search at a particular lagset, the non-exhaustive BIC value is not plotted (for visual clarity). Therefore, the blue, orange, and green points shown in the following plots only correspond to instances in which the genetic or stepwise algorithms failed to find the best model for a particular lagset.

We also include plots showing only the best 15 lagsets. Our conclusions from the plots in this section are discussed in more detail in Section 8.

5.1 Four Covariates

5.1.1 All Lagsets

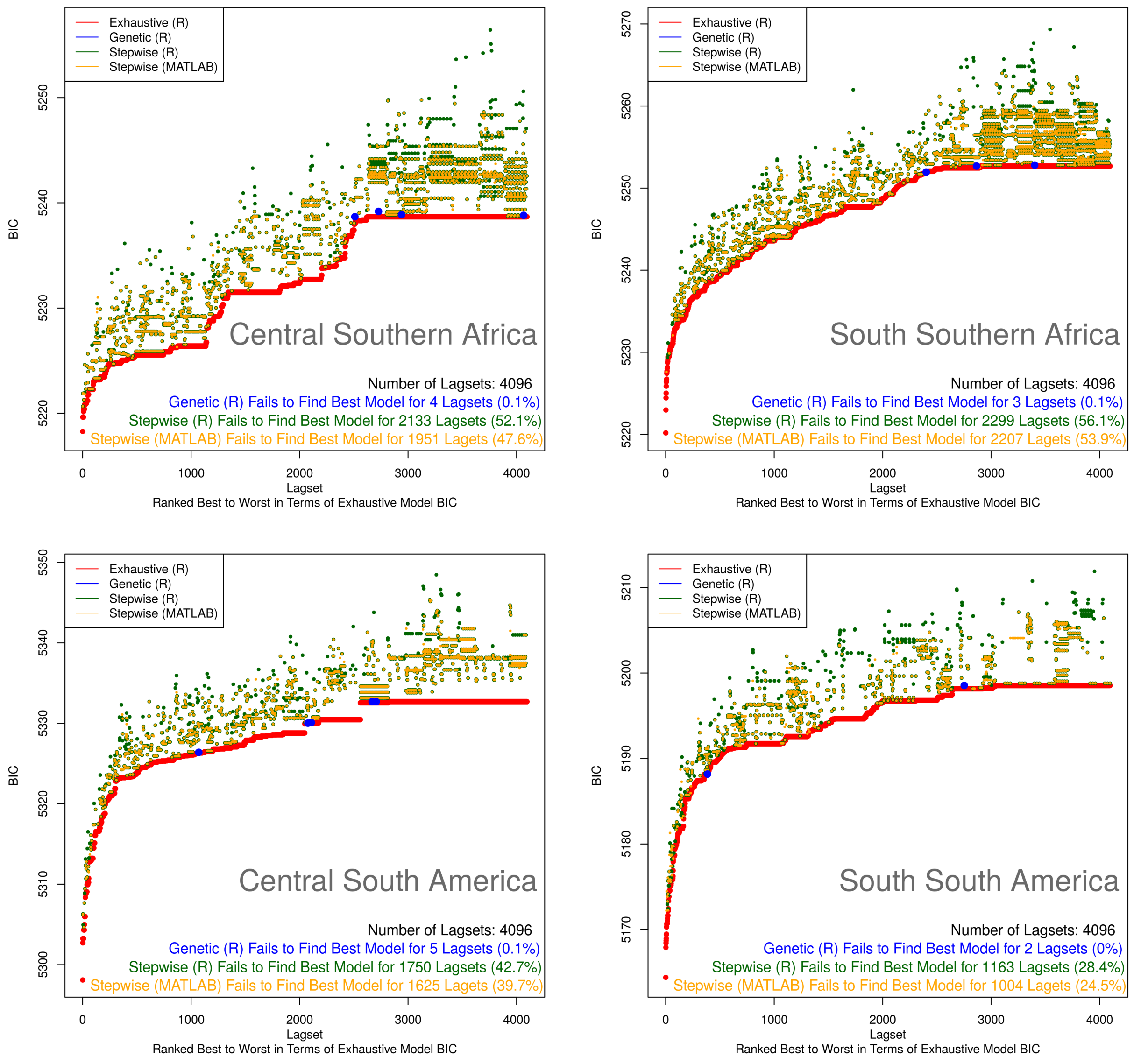


Figure 5: Comparison of variable selection algorithms for the African and South American response regions. Plotted in red are the BIC values of the best model found by the exhaustive search for each lagset. Plotted in blue, orange, and green are lagsets where one of the non-exhaustive algorithms failed to find the best model. Summary statistics are displayed on the plot.

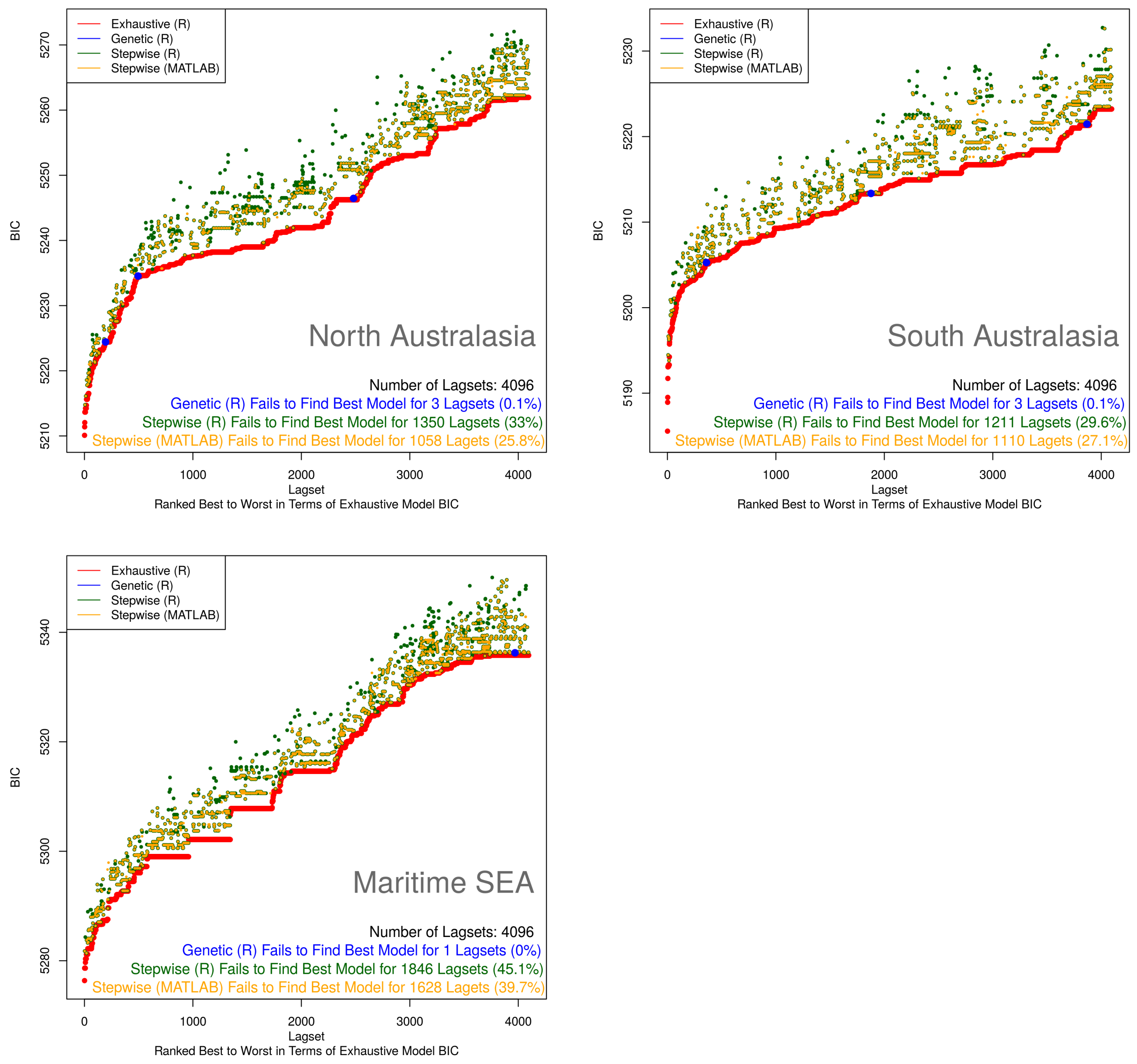


Figure 6: Comparison of variable selection algorithms for the Australasia and Maritime SEA response regions. Plotted in red are the BIC values of the best model found by the exhaustive search for each lagset. Plotted in blue, orange, and green are lagsets where one of the non-exhaustive algorithms failed to find the best model. Summary statistics are displayed on the plot.

Here we see that the genetic algorithm finds the best model for all but a few lagsets, depending on the response region. The stepwise algorithms in `R` and `MATLAB` have similar performance, with the `MATLAB` algorithm finding the best model slightly more often than the `R` version. Both stepwise algorithms, however, fail to find the best model much more often than the genetic algorithm.

5.1.2 Best 15 Lagsets

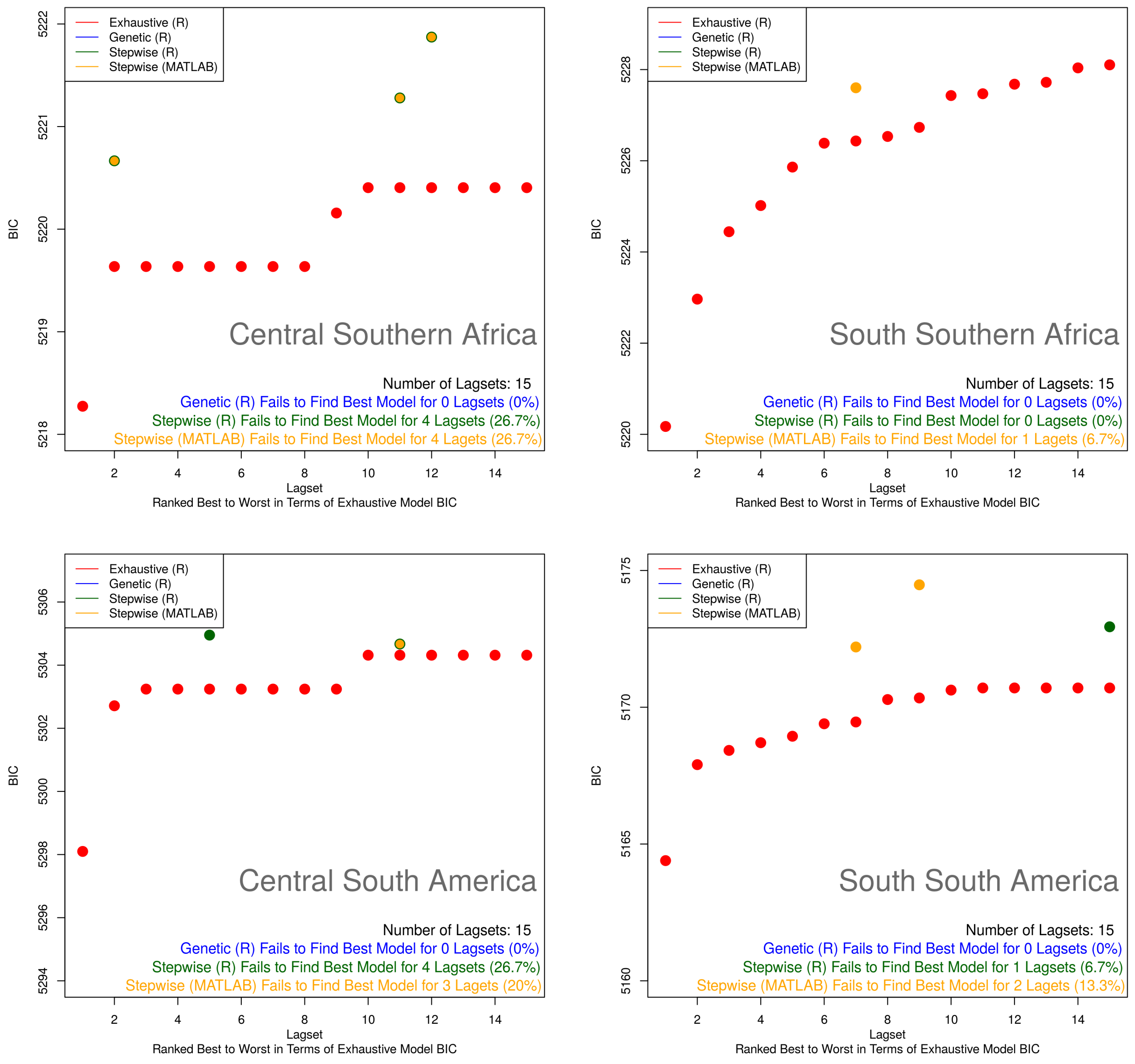


Figure 7: Zoomed in comparison of variable selection algorithms for the African and South American response regions. Only the best 15 lagsets are shown here. Plotted in red are the BIC values of the best model found by the exhaustive search for each lagset. Plotted in blue, orange, and green are lagsets where one of the non-exhaustive algorithms failed to find the best model. Summary statistics only take into account the best 15 lagsets.

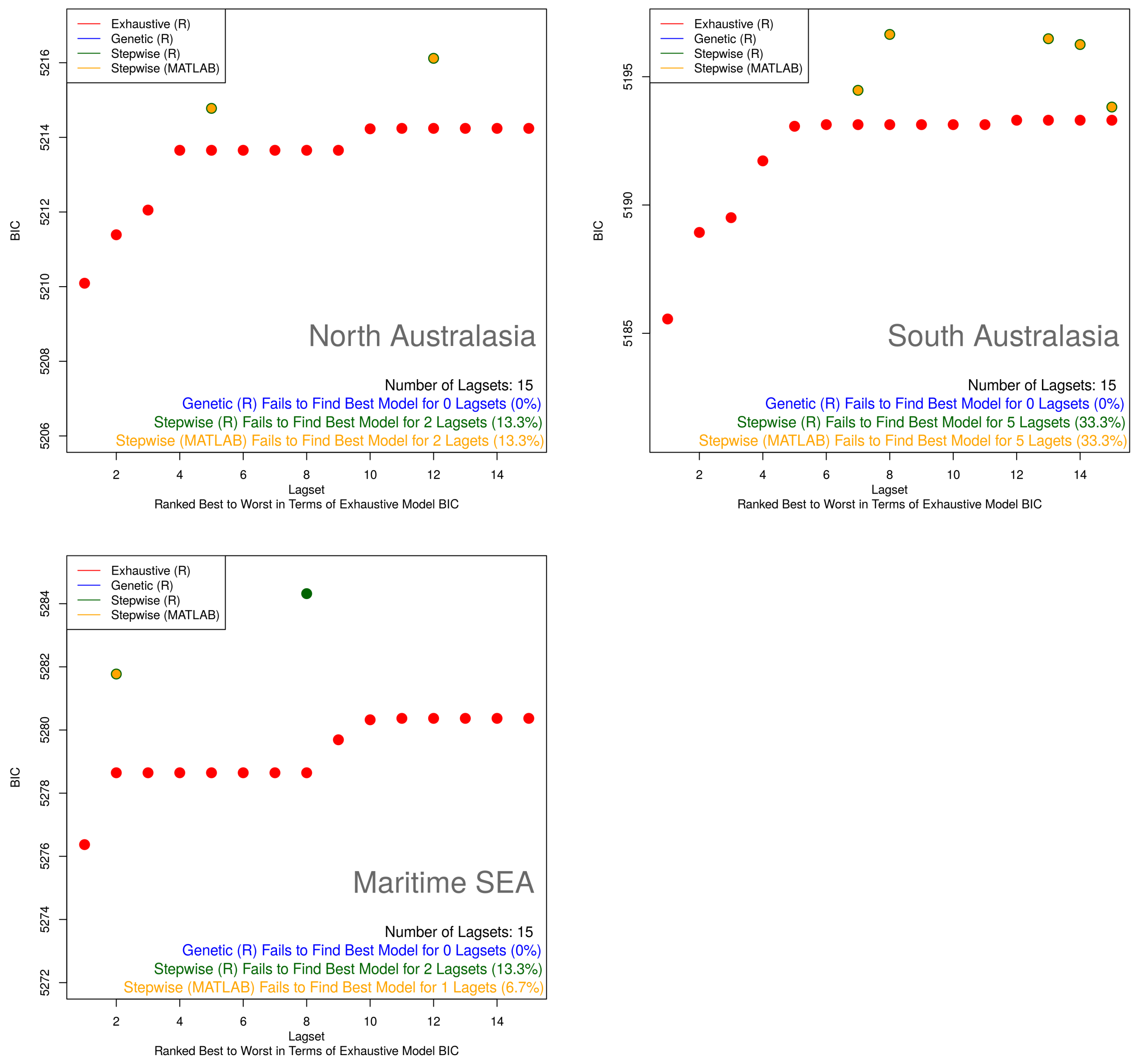


Figure 8: Zoomed in comparison of variable selection algorithms for the Australasia and Maritime SEA response regions. Only the best 15 lagsets are shown here. Plotted in red are the BIC values of the best model found by the exhaustive search for each lagset. Plotted in blue, orange, and green are lagsets where one of the non-exhaustive algorithms failed to find the best model. Summary statistics only take into account the best 15 lagsets.

In the four covariate class, all three non-exhaustive search algorithms happen to agree on the best lagset. This is promising, as the best lagset is the one selected for the predictive model in that response region. However, this will not always occur, as non-exhaustive search options are not guaranteed to find the absolute best model. The user should therefore by wary of using non-exhaustive search algorithms.

5.2 Five Covariates

5.2.1 All Lagsets

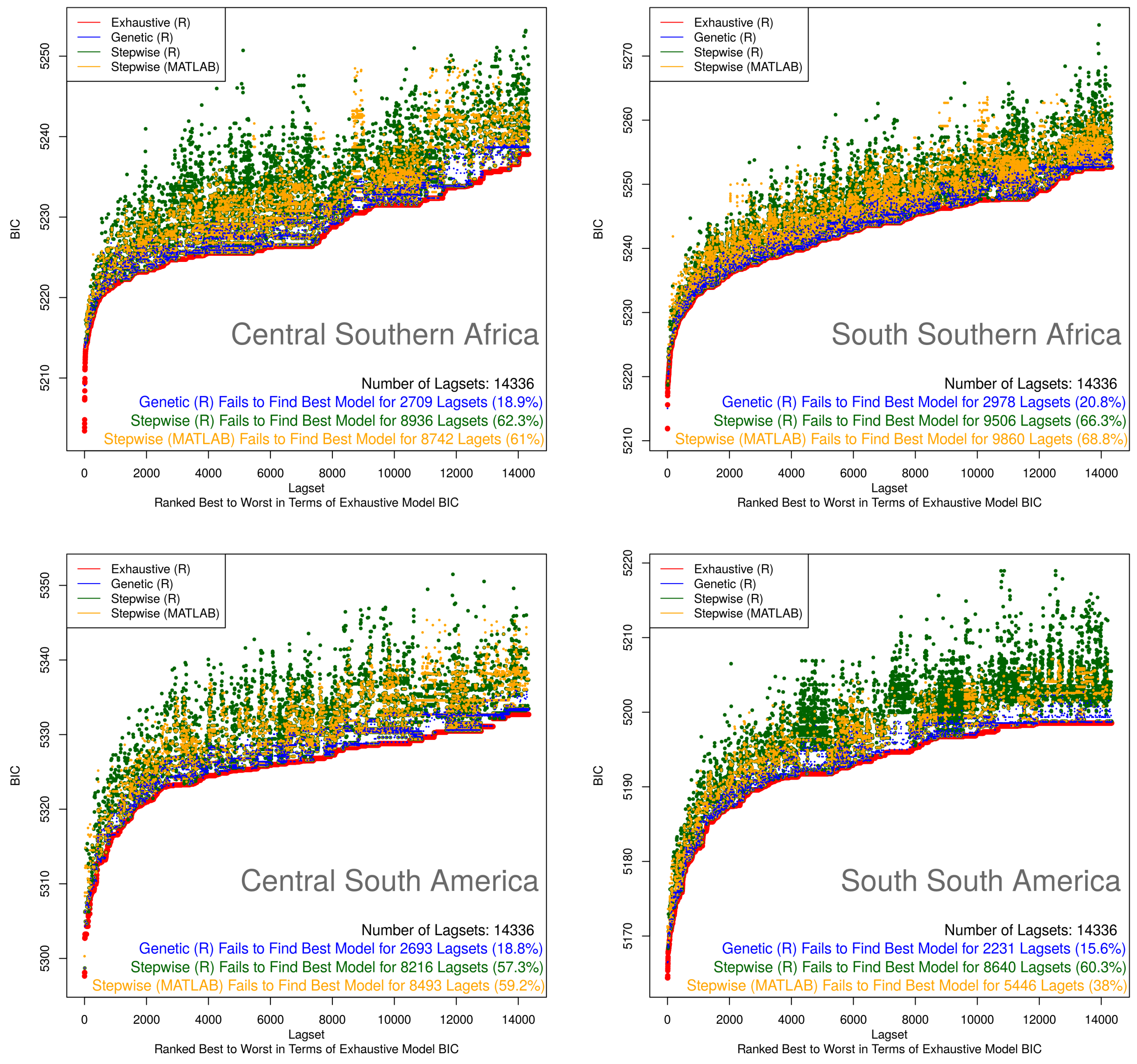


Figure 9: Comparison of variable selection algorithms for the African and South American response regions. Plotted in red are the BIC values of the best model found by the exhaustive search for each lagset. Plotted in blue, orange, and green are lagsets where one of the non-exhaustive algorithms failed to find the best model. Summary statistics are displayed on the plot.

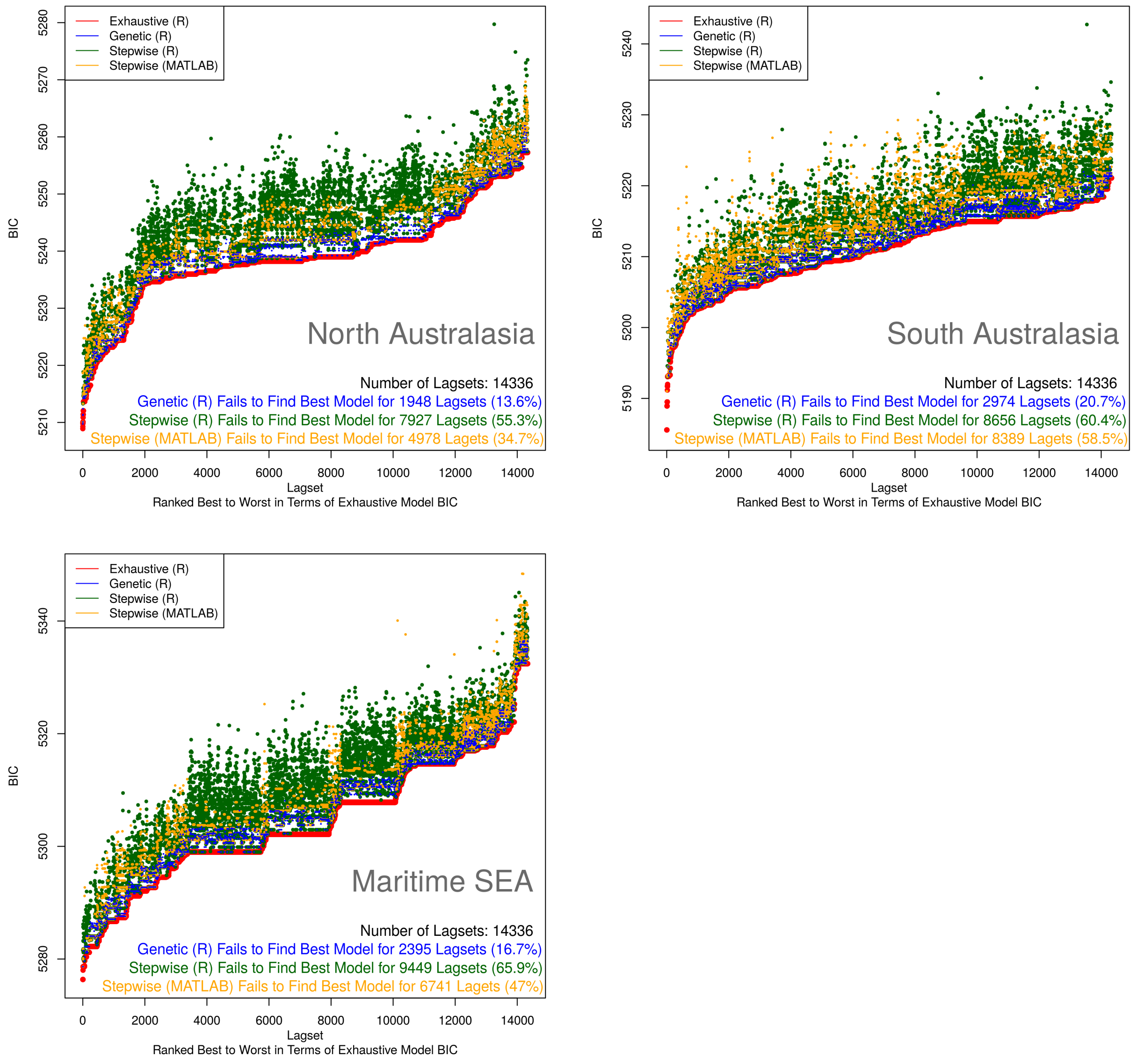


Figure 10: Comparison of variable selection algorithms for the Australasian and Maritime SEA response regions. Plotted in red are the BIC values of the best model found by the exhaustive search for each lagset. Plotted in blue, orange, and green are lagsets where one of the non-exhaustive algorithms failed to find the best model. Summary statistics are displayed on the plot.

In the five covariate case, we see worse performance across all non-exhaustive algorithms. This makes sense, as there are now many more models to consider. The genetic algorithm is again able to find the best model more often than both of the stepwise algorithms. The `MATLAB` stepwise algorithm again performs better than the `R` stepwise algorithm in most response regions.

5.2.2 Best 15 Lagsets

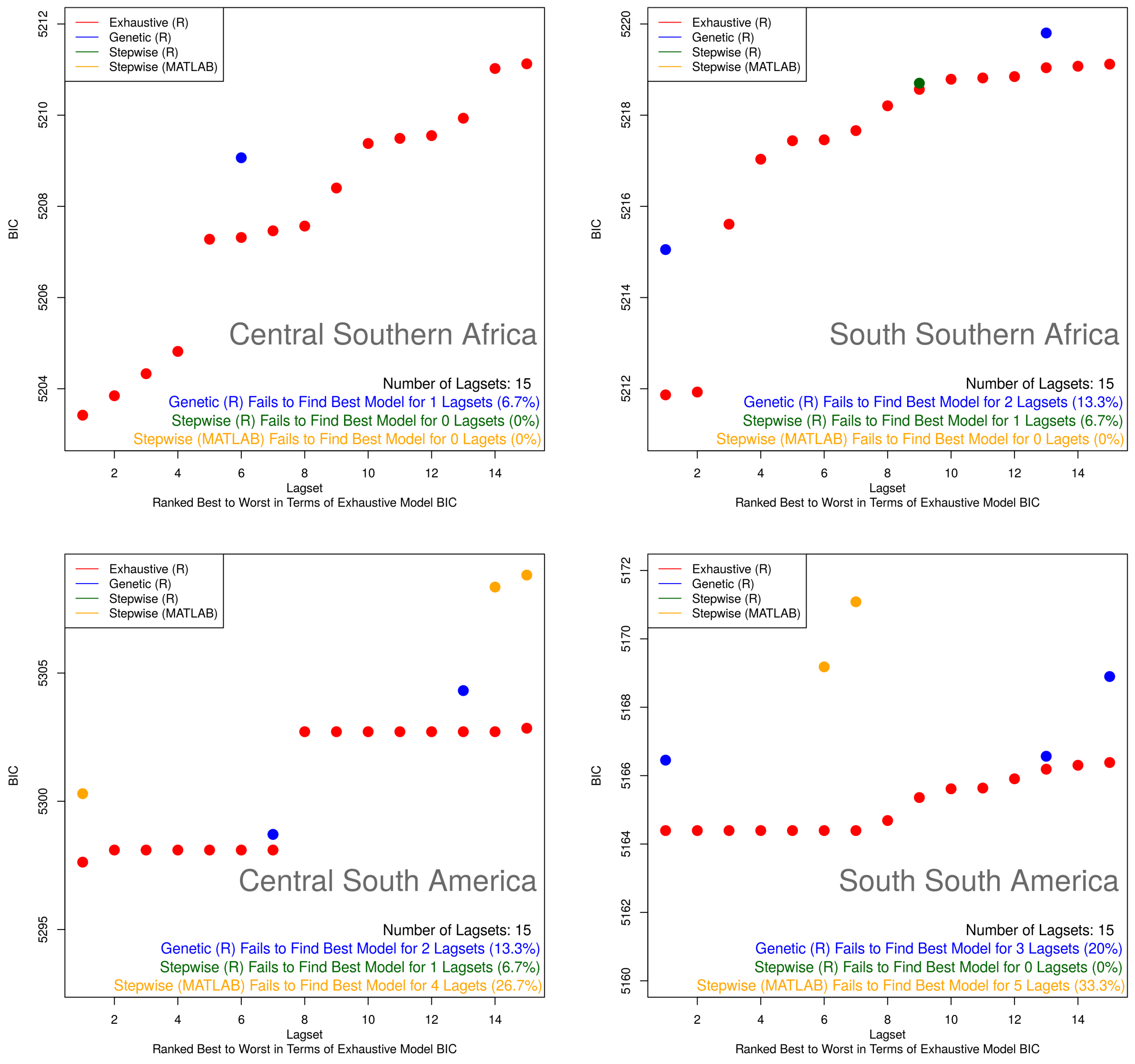
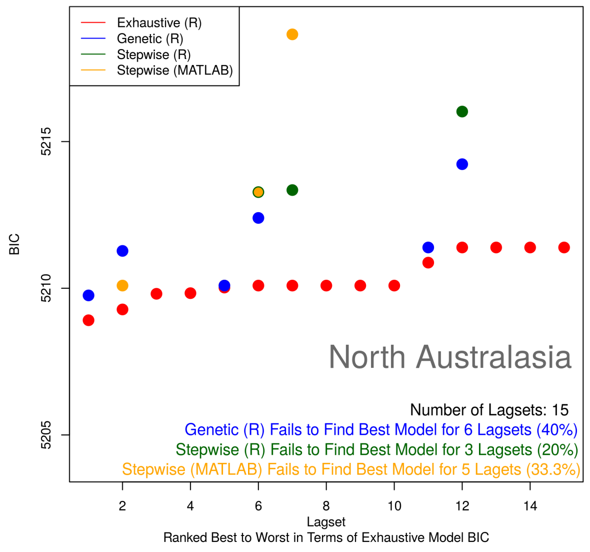
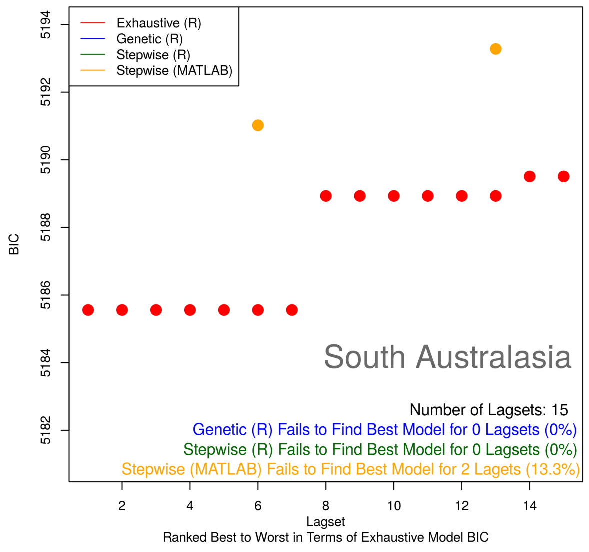


Figure 11: Zoomed in comparison of variable selection algorithms for the African and South American response regions. Only the best 15 lagsets are shown here. Plotted in red are the BIC values of the best model found by the exhaustive search for each lagset. Plotted in blue, orange, and green are lagsets where one of the non-exhaustive algorithms failed to find the best model. Summary statistics only take into account the best 15 lagsets.





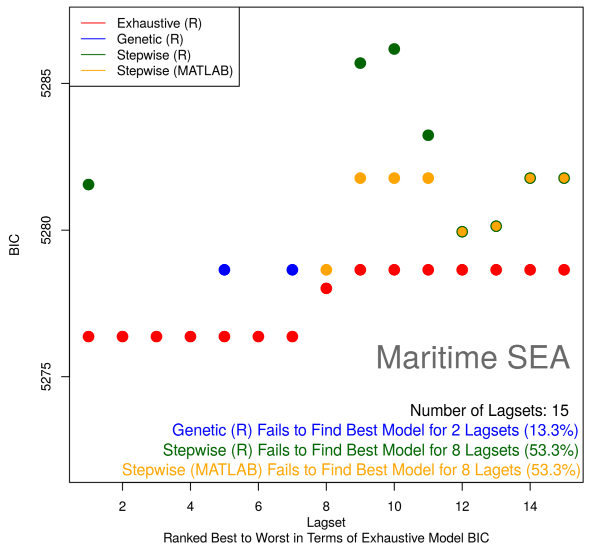


Figure 12: Zoomed in comparison of variable selection algorithms for the Australasia and Maritime SEA response regions. Only the best 15 lagsets are shown here. Plotted in red are the BIC values of the best model found by the exhaustive search for each lagset. Plotted in blue, orange, and green are lagsets where one of the non-exhaustive algorithms failed to find the best model. Summary statistics only take into account the best 15 lagsets.

In the five covariate case, we see that the non-exhaustive search options do not always find the best model for the best lagset. As a result, the user should be wary of using non-exhaustive search methods, especially with larger parameter spaces, as they will not always find the best predictive model for a given region.

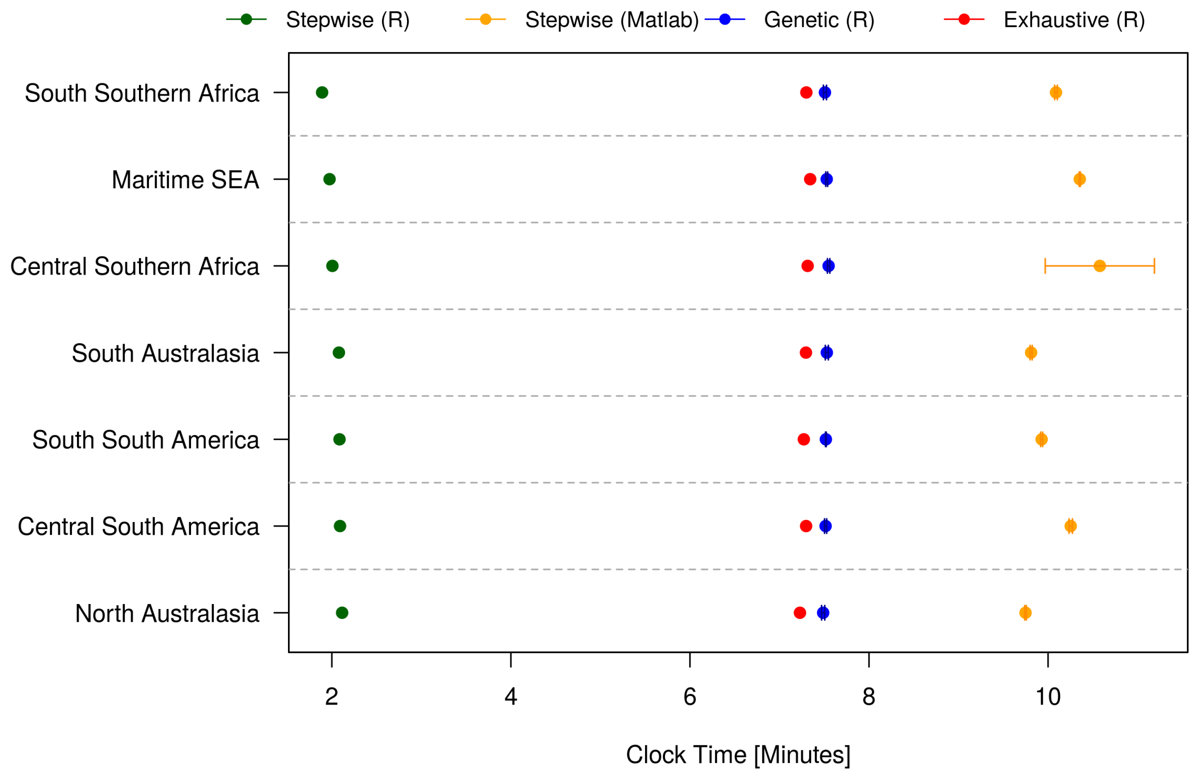
6 Timing Comparison

In this section, we present the run time for the exhaustive, genetic, and stepwise algorithms. To expedite this study, certain algorithms were only run once. These values, therefore, have no error bars associated with them. The algorithms that were run more than once are presented with error bars corresponding to the mean value ± one standard deviation. The algorithms with error bars were run four times. Again, our conclusions are discussed in more detail in Section 8.

Computational trials were performed on a personal computer with an Intel Core i7-4510U CPU @ 2.00GHz, four cores, 8 GB RAM, Ubuntu version 16.04, running 64 bit `R` version 3.6.2 and 64 bit `MATLAB R2019a` version 9.6.0.1174912.

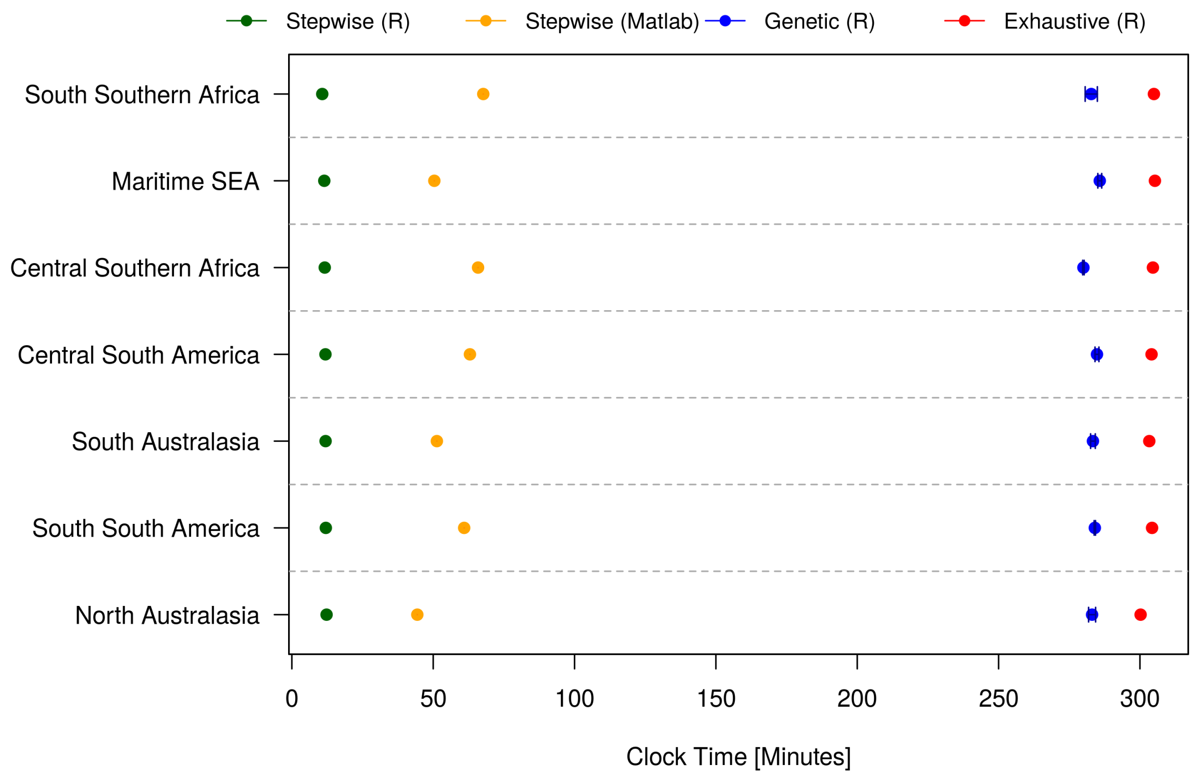
Note that we only consider the four and five covariate cases in this technote. However, the scalability of these algorithms, primarily the genetic and exhaustive search, is an ongoing research topic.

6.1 Four Covariates



Here we see that the stepwise algorithm in `R` runs much faster than all other algorithms. The exhaustive algorithm actually runs slightly faster than the genetic algorithm in the four covariate case. The `MATLAB` stepwise algorithm has the longest runtime of all algorithms considered.

6.2 Five Covariates



Here we see much longer run times, which again makes sense, as there are now many more models to consider. Both stepwise algorithms run much faster than the genetic and exhaustive algorithms, with the `R` version running faster than the `MATLAB` version. In the five covariate case, we begin to see the computational benefit of the genetic algorithm over the exhaustive search. With more predictor variables, this benefit would likely become much more apparent.

7 Best Predictive Models

In Section 5, we considered the best model for each lagset. This provided many different response-covariate relationships that we used to compare the variable selection algorithms. However, in practice we are often only concerned with the best model for the best lagset. As a means of validation, we compared these best predictive models found by the exhaustive search in `regClimateChem` (with four covariates) to the best predictive models reported in Buchholz et al. [1]. The lag values and predictor variables included in these models agree for all response regions.

The best predictive model for each response region is listed in Table 2. Table 2 includes lag values, coefficient estimates, and the adjusted R^2 value for each response region.

Table 2: The best predictive model for each response region found using the exhaustive search in `regClimateChem`.

		Australasia		Southern Africa		South America	
Climate Mode	Maritime SEA	North	South	Central	South	Central	South
				τ			
Nino 3.4	1	3	7	7	8	8	8
DMI	8	1	1	4	4	6	1
TSA	5	7	2	2	2	2	2
SAM	1	1	8	2	2	2	5
				μ			
Constant	-0.29	-0.35	0.05	-0.09	0.04	0.13	-0.06
χ_k				a_k			
Nino 3.4	2.53	1.15	1.45	1.86	1.71	3.14	1.17
DMI	0.67	1.34	0.95	0.79	1.08	3.07	1.19
TSA	0.70	-1.10	-1.86	-1.00	-0.98	-4.84	-0.73
SAM	0.33	-0.19	-0.48	0.40	0.18	0.03	-0.05
χ_i x χ_j				b_{ij}			
Nino 3.4 x DMI	-4.81	1.55	NS	NS	0.73	2.68	NS
Nino 3.4 x TSA	NS	-2.75	-2.73	-2.26	-3.65	-7.60	-1.72
Nino 3.4 x SAM	NS	NS	NS	NS	NS	-1.01	-0.49
DMI x TSA	-5.83	NS	NS	2.88	NS	NS	NS
DMI x SAM	NS	NS	NS	NS	NS	NS	-0.69
TSA x SAM	-2.88	-1.29	NS	NS	2.97	4.45	NS
				Model Fit Statistics			
Adjusted R^2	0.72	0.69	0.56	0.46	0.57	0.60	0.58

8 Conclusions

In this document we have described updates to the data driven variable selection code used to model CO concentrations in Buchholz et al. [1]. These updates include the following main items.

- The code is now written in an open source `R` package.
- It is now possible to create models with multiple lags of a single climate mode.
- An exhaustive search algorithm was added.
- An additional non-exhaustive search option was added. The user can now select between a genetic algorithm and stepwise selection.

In Section 5.1 (the four covariate case), we see much better results from the genetic algorithm than the stepwise algorithms. The stepwise algorithm in `R` fails to find the best model 30-55% of the time, and the stepwise algorithm in `MATLAB` fails to find the best model 25-55% of the time, depending on the response region. The genetic algorithm, on the other hand, fails to find the best model less than 1% of the time in all response regions. Furthermore, in Section 6.1 (the four covariate case), we see that the genetic algorithm runs about 25% faster than the stepwise algorithm in `MATLAB`. The stepwise algorithm in R, however, runs about 80% faster than the stepwise algorithm in `MATLAB`. We also see that the exhaustive search runs faster than the genetic algorithm and the stepwise algorithm in `MATLAB` in the four covariate case. Optimizing the genetic algorithm for the climate chemistry application is an ongoing research topic. Ideally, the genetic runtime will fall between the stepwise search in `R` and the exhaustive search. This way, the genetic algorithm can serve as a slightly more accurate option than stepwise selection, while still providing results faster than the computationally expensive exhaustive search.

In Section 5.2 (the five covariate case), we see that the genetic algorithm fails to find the best model 13-20% of the time. This percentage is much higher than the four covariate case, but understandable since adding a fifth covariate vastly increases the number of possible models. The stepwise search in `R` fails to find the best model 55-65% of the time, and the stepwise search in `MATLAB` fails to find the best model 35-65% of the time. In Section 6.2 (the five covariate case), we see that the genetic algorithm runs about 8% faster than the exhaustive search. With five covariates, the computational benefit of the genetic algorithm begins to appear, as it is slighter faster than the exhaustive method. This benefit would be more apparent with more covariates. The stepwise search in `MATLAB` runs about 80% faster than the exhaustive search, and the stepwise search in `R` runs about 90% faster. In the five covariate case, the scalability of the stepwise search becomes blatantly apparent. Again, the ongoing optimization study into the genetic algorithm seeks to adjust the hyperparameters so that the runtime of the genetic algorithm falls more equally between the stepwise and exhaustive run times.

Examining the best 15 lagsets in both the four and five covariate cases reveals that the non-exhaustive search algorithms do not always find the best model for the best lagset in a given response region. This is seen more clearly in the five covariate case. Therefore, the user should keep in mind that the non-exhaustive search options will not always find the best predictive model for a given response region.

Finally, it is important to emphasize that these results are only for the four and five covariate cases. The performance of the two non-exhaustive options might change for scenarios with many more predictor variables or a different relationship between the response and the predictors.

These results can be summarized as follows:

- Three variable selection algorithms are provided in the `regClimateChem` `R` package: an exhaustive search, a genetic algorithm, and a stepwise search.
- The exhaustive search by definition always finds the best model, regardless of the number of covariates. In the four covariate case, it runs faster than the genetic algorithm and the stepwise search in `MATLAB`. In the five covariate case, it is slower than all of the non-exhaustive search options.
- The genetic search is able to find the best model more often than the two stepwise selection algorithms. In the four covariate case, it is faster than the stepwise selection algorithm in `MATLAB`. In the five covariate case, it is faster than the exhaustive search.
- The stepwise search in `R` is the worst algorithm in terms of finding the best model for each lagset, but it is the most computationally efficient. In both the four and five covariate cases, it runs faster than all other variable selection techniques.
- The user can select between these variable selection algorithms depending on their computational resources and their need for model accuracy.

Based on these results, we present the `regClimateChem` package as an option for `R` users interested in studying the variability in climate chemistry.

Appendix A Description of Updated Configuration File

The configuration file is a convenient way to set the various parameters used in `regClimateChem`. A few changes have been made to the configuration file described in Simonson et al. [2]. The new `R` configuration file contains the following variables that need to be defined by the user (or left at their default values).

- `response.data`
  - `name.of.region`: The name of the response region to analyze. Choose either CentralSAfrica, EastSAmerica, MaritimeSEA, NorthAustralasia, SouthAustralasia, SouthSAfrica, or WestSAmerica.
  - `file.path`: Path to the response region CO concentration .csv file.
  - `file.name`: Name of the response region CO cencentration .csv file.
- `index.data`
  - `name.of.index`: The full name of the climate modes to be included in the analysis. Use spaces and proper formatting.
  - `short.name`: Abbreviation for each climate mode.
  - `file.path`: Path to the climate mode index .csv file.
  - `file.name`: Name of the climate mode index .csv file.
- `model.parameters`
  - `output.period.start`: This is the first month of the time period in which we want to explain CO variability.
  - `output.period.end`: This is the last month of the time period in which we want to explain the CO variability.
  - `regression.selection.criterion`: The criterion that is used to define the "best" model for a given lagset. Choose either "bic" or "aic".
  - `search.algorithm`: The algorithm used to perform variable selection. Use "h" for the exhaustive search, "g" for the genetic search, or "s" for the stepwise selection.
  - `optimization.criterion`: The criterion that is used to select the best lagset and corresponding model. Use either "bic" or "adjr2".
  - `lag.user.limits`: The minimum and maximum lag values that should be considered for each climate mode.
  - `should.search.parallel`: Use "TRUE" to perform the lag space search in parallel. Use "FALSE" to perform the lag space search in serial.
  - `number.of.cores.to.use`: The number of cores to devote to the parallel lag space search. By default, this is set to the number of cores available on the user's machine minus one.
  - `smoothing.parameter`: Window size to use when performing a moving average of the climate mode index data. Note that this number is the number of data points on each side of the current point to be averaged, not the total window size.
  - `max.num.terms`: The maximum number of terms to consider per climate mode. If set to one, then at most one lag value of each climate mode will be included. If set to two or more, then multiple lags of a single climate mode will be considered.

Appendix B Codebase Availability

The `R` package `regClimateChem` can be accessed from GitHub. The repository can be found at:

`https://github.com/wsdaniels/regClimateChem.git`

This repository contains the following:

- *single_run_config_file.R*: This configuration file contains the settings used in the *run_once.R* script.
- *run_once.R*: This script will run the code once, based on the setting in the *single_run_config_file.R*.
- *run_multiple.R*: This script will run the code multiple times, varying certain parameters.
- *data*: This directory contains CO and climate mode index data used in the analysis.
- *R*: This directory contains the R functions used in the analysis.

References

[1] R. R. Buchholz, D. Hammerling, H. M. Worden, M. N. Deeter, L. K. Emmons, D. P. Edwards, and S. A. Monks, “Links between carbon monoxide and climate indices for the southern hemisphere and tropical fire regions.” *Journal of Geophysical Research: Atmospheres*, vol. 123, 2018.

[2] P. Simonson and D. Hammerling, “Refactoring data-driven model selection code for improvements in interpretability, generality, and computational expense,” *NCAR Technical Notes*, 2018.

[3] B. Gaubert, H. M. Worden, A. F. Arellano, L. K. Emmons, S. Tilmes, J. Barré, S. Martinez Alonso, F. Vitt, J. L. Anderson, F. Alkemade, S. Houweling, and D. P. Edwards, “Chemical Feedback From Decreasing Carbon Monoxide Emissions,” *Geophysical Research Letters*, vol. 44, no. 19, pp. 9985–9995, oct 2017.

[4] D. P. Edwards, L. K. Emmons, J. C. Gille, A. Chu, J.-L. Attié, L. Giglio, S. W. Wood, J. Haywood, M. N. Deeter, S. T. Massie, D. C. Ziskin, and J. R. Drummond, “Satellite-observed pollution from Southern Hemisphere biomass burning,” *Journal of Geophysical Research: Atmospheres*, vol. 111, no. D14, 2006. [Online]. Available: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2005JD006655

[5] A. Voulgarakis, M. E. Marlier, G. Faluvegi, D. T. Shindell, K. Tsigaridis, and S. Mangeon, “Interannual variability of tropospheric trace gases and aerosols: The role of biomass burning emissions,” *Journal of Geophysical Research: Atmospheres*, vol. 120, no. 14, pp. 7157–7173, 2015. [Online]. Available: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2014JD022926

[6] G. R. van der Werf, J. T. Randerson, L. Giglio, N. Gobron, and A. J. Dolman, “Climate controls on the variability of fires in the tropics and subtropics,” *Global Biogeochemical Cycles*, vol. 22, no. 3, 2008. [Online]. Available: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2007GB003122

[7] R. V. Andreoli and M. T. Kayano, “Tropical Pacific and South Atlantic effects on rainfall variability over Northeast Brazil,” *International Journal of Climatology*, vol. 26, no. 13, pp. 1895–1912, 2006. [Online]. Available: https://rmets.onlinelibrary.wiley.com/doi/abs/10.1002/joc.1341

[8] H. H. Hendon, D. W. J. Thompson, and M. C. Wheeler, “Australian Rainfall and Surface Temperature Variations Associated with the Southern Hemisphere Annular Mode,” *Journal of Climate*, vol. 20, no. 11, pp. 2452–2467, jun 2007. [Online]. Available: http://journals.ametsoc.org/doi/10.1175/JCLI4134.1

[9] V. Calcagno and C. de Mazancourt, “glmulti: An R package for easy automated model selection with (generalized) linear models,” *Journal of Statistical Software*, vol. 34, no. 12, p. 29, 2010.

[10] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: With Applications in R.* Springer Publishing Company, Incorporated, 2014.