Yuriko Hashimoto
990853225
Penn State MGIS Capstone
GEOG 596B

Evaluating environmental predictors of breeding waterfowl population distribution and abundance in the Central Interior of British Columbia (2007-2017)

ABSTRACT

Conservation biology has seen an explosion of species distribution models (SDMs) in the recent published literature and growing numbers of governments and organizations responsible for small-scale regional to global conservation activities are actively utilizing them or have embarked on their own predictive studies (Franklin and Miller 2009; Guisan, Wilfried, and Zimmermann 2017). Waterfowl conservation planning in British Columbia (BC) however currently does not have the benefit of breeding waterfowl SDMs. To address this gap I have developed methods to generate SDMs using 11 years of survey data from BC's annual Breeding Waterfowl Survey between 2007 and 2017. Models utilizing a random forest-based approach were used to produce predictive maps for ten species and five species groups within each of the eight ecosections within the central interior representing core provincial breeding waterfowl habitat. The developed methods, techniques and recommendations form a template for future studies in the generation of species distribution maps for guides in conservation planning.

Keywords: waterfowl, random forest, species distribution model, conservation

INTRODUCTION

While the uses and practical applications of species distribution models in conservation decision-making are diverse (Guillera-Arroita et al. 2015; Guisan et al. 2013; Guisan and Thuiller 2005) the overarching aim is to conserve biodiversity and support better land use planning and environmentally sustainable management practices. The breadth of practical applications include predicting climate change adaptation, invasive species management, critical habitat identification, reserve selection, impact assessment and ecological restoration (Elith and Graham 2009; Franklin and Miller 2009).

Recognizing the lack of waterfowl species distribution models (SDMs) to support conservation actions in response to the urgency of conservation challenges in Canada's boreal forests, Barker et al (2014) published the first national-scale waterfowl SDMs in 2014 based on the traditional Waterfowl Breeding Population and Habitat Survey (WBPHS) database. The WBPHS has been described as "arguably the largest and best-designed population survey in the world" (Murray, Anderson, and Steury 2010), continental in scale and running on a continuous annual basis since 1955, the survey captures what is recognized to be "core waterfowl breeding habitat" in North America. Designed primarily to provide annual breeding population estimates and inform hunting regulations in the U.S. and Canada, these data additionally provide a long-term population monitoring dataset that has since informed countless studies on species-habitat relationships in support of waterfowl conservation. While the models performed well over all, the results of extrapolation to out-of-sample areas were variable for a number of reasons as outlined by the authors. British Columbia (BC) is not within the traditional survey and no survey data from BC informed the models which predicted low total densities for the Western Cordillera between the Pacific coast and Rocky Mountains—a result that conflicts with anecdotal expert opinion (A. Breault, personal communication 2018). As such there are currently no adopted waterfowl

breeding population distribution models in use for the province. This study aims to fill the this gap and help support the mandate of the Canadian Wildlife Service (CWS) to conserve biodiversity (Environment and Climate Change Canada 2019; Environment Canada - Biodiversity Convention Office 1995).In the early 2000s after exploratory pilot surveys in BC's central and sub-boreal highlands determined waterfowl population abundances to be significant enough to justify a regional breeding survey program, the British Columbia Breeding Waterfowl Survey began in earnest in 2006. BC's May surveys, run jointly by the CWS and the U.S. Fish and Wildlife Service and conducted in partnership with Ducks Unlimited Canada, inform the annual population status of migratory game birds in the Central Interior Plateau of BC and contributes to adaptive harvest strategies for mallards in the Pacific Flyway (Zimpfer, Breault, and Sanders 2019).

I developed methods to create SDMs for the top ten most abundant species as well as five group classifications based on life history traits and community descriptors (Baldassarre 2014) (Tables 1 and 2). Guisan et al (2013) have observed that despite the oft-cited assumption of the utility of SDMs in conservation decision-making there is little evidence demonstrating their application in real-world conservation management. Following their recommendation that modellers make explicit the objectives of the framework within which models were developed, I provide guidance on how the SDMs may be improved and implemented as a decision-making platform for conservation planning by the Canadian Wildlife Service and partners in conservation.

MATERIALS AND METHODS

Survey Methods

The study area was designed to capture BC's prime waterfowl breeding habitat of the humid, continental plateaus of the central and sub-boreal Interior between the Coast mountains to the west and Rocky Mountains to the east. The region of interlocking highlands and valleys are mainly forested and sparsely populated. The main industries include forestry, cereal crop agriculture within lowland valleys, ranching and mining (Demarchi 2011).

The survey design consists of latitudinal strip transects 400 metres wide, spaced ten miles apart (16.09 km) and delineated by the boundaries of eight ecosections which represent the finest scale, sub-regional stratum within the Ecoregion Classification System of British Columbia—a small-scale ecosystem management framework stratifying nested regions of "similar climate, physiography, oceanography, hydrology, vegetation and wildlife potential" (Demarchi 2011) (Figure 1). The classification levels ordered from high to low include Ecodomains for global mapping applications, ecodivisions for national-scale, ecoprovinces for provincial scale, ecoregions for regional and ecosections for sub-regional planning. While designed for small-scale mapping ecosection boundaries are drawn at large scale (1:20,000).

The southern two-thirds of the study area falls within the Fraser Plateau Ecoregion. The flat and rolling hills of the region contain numerous meandering streams and low-lying depressions that create an abundance of wetland habitat (Demarchi 2011) while the Central Interior Ecoprovince in which it nests supports 65% of all species known to occur and 61% of all bird species known to breed in BC. with the greatest total abundances of waterfowl occurring in the Cariboo Basin ecosection within the Riske Creek area (Demarchi 2011; Savard, Sean Boyd, and John Smith 1994). The remaining northernmost third of the study area stretches into the Sub-boreal Interior Ecoprovince which supports 57% of all bird species known to occur and 45% known to breed in the province (Demarchi 2011). Table 1 provides general descriptions of study area ecosections.

The surveys are conducted by helicopter and surveyed by a consistent survey crew of two to three experienced observers (Zimpfer, Breault, and Sanders 2019). The transects when originally conceived were designed within the NAD 1983 geographic coordinate system and were based on the mapped Ecoregion Classification System version 2.0 (1995). The ecosection boundaries have since been updated with finer scale vegetation zonation data in version 2.1 (2006) however the annual surveys continue to follow the extent of the original design. All analyses were based on the areal overlap between the two versions with the boundaries redefined in alignment with version 2.1.
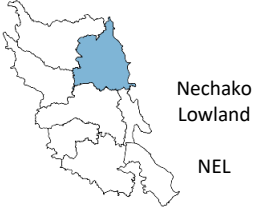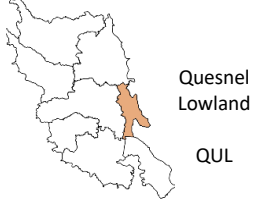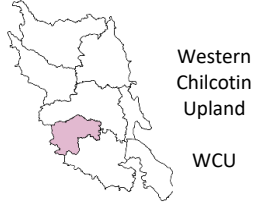
Figure 1. The survey area divided into the ecosections of the Ecoregions of British Columbia Classification system (Demarchi, 2011). Latitudinal strip transects (horizontal lines) are spaced ten miles (~16 km) apart within the Central and Sub-boreal plateaus of BC.

Table 1. General summary descriptions of ecosections within study area as described in Demarchi (2011).

Ecoregion	Ecosection	Topography	Climate	Vegetation	Governance, Protection, Development
Fraser Basin	Babine Upland BAU	Rolling upland with low ridges; many small streams and lakes and several large lakes	Sub-continental--extreme winter cold and snow events, humid and rainy	Sub-boreal spruce dominated forests	Sparsely populated, seasonal lodges; many protected areas
Fraser Plateau	Bulkley Basin BUB	Broad, lowland valleys with many lakes	Rainshadow effect of Coast mountains	Dominated by lodgepole pine; trembling aspen in lower south-facing valleys	Extensive development and agriculture within Yellowhead Hwy corridor and surrounding Francois Lake; protected areas include Francois lake Park
Fraser Plateau	Cariboo Basin CAB	Rolling uplands; many streams, wetlands and lakes	Warmest temperatures within ecoregion; relatively high summer precipitation	Vegetation zonation changes	Small farms, ranches and logging are main industries. Williams Lake is largest city
Fraser Plateau	Chilcotin Plateau CHP	Rolling uplands with many small lakes and wetlands; higher relief in south and northwest abutting Chilcotin Ranges and west Chilcotin; many small streams and rivers	Pronounced rain shadow effect	Douglas fir near rivers and lodgepole pine at higher elevations	Ranching and logging are main industries; Big Creek Park is largest park
Fraser Plateau	Nazko Upland NAU	Rolling uplands with high relief within north-central region; slow moving streams and wetlands	Sub-continental climate--cold winters, warm summers, maximum precipitation in late spring/early summer; subject to cold balasts of Arctic air	Lodgepole pine in south and white spruce/lodgepole pine/subalpine fir in north	Ranching and First Nations reserves but no large communities

Fraser Basin	Nechako Lowland NEL	Flat or gently rolling lowlands	Sub-boreal climate-humid summers and harsh winters	Sub-boreal spruce forests and lodgepole pine	Largest city is Prince George; cereal crop farming in southern lowlying areas and ranching
Fraser Plateau	Quesnel Lowland QUL	Lowland trench bissected by the Fraser River; small lakes and wetlands but no large lakes	Subject to cold Arctic air; increased precipitation as air moves east over Columbia Mountains and in summer due to upland wetlands	Douglas-fir on dry south-facing slopes and trembling aspen, lodgepole pine, white spruce to subalpine fir with increasing elevation	Quesnel is the largest and only city but many communities; farming ranching and forestry (with extensive logging)
Fraser Plateau	Western Chilcotin Upland WCU	Rounded uplanded with a large lowland area containing many lakes	Sub-continental climate subject to blasts of Arctic winter air	Mainly lodgepole pine and white spruce in higher elevations	Anahim Lake is largest community

The survey does not follow the design methodology of the traditional WBPHS, however the survey techniques are consistent with the methods outlined in Smith (1995) and the Standard Operating Procedures (US Fish and Wildlife Service (USFWS) and Canadian Wildlife Service (CWS) 1987). Prior to 2010, the survey technologies employed consisted of paper maps, field notes and GPS units with navigation determined by piloting along bearings to GPS waypoints. In 2010, the mobile GIS software PC-Mapper with Airborne Inspection (version 4.0, Corvallis Microtechnology Inc, 2015) was adopted for both survey navigation and data collection. The software is run on Panasonic Toughbooks (CF-19 and CF-31) with the screen in view of both the pilot and observer. Real-time navigation is guided by the GIS with reference base data containing strip transect boundaries, freshwater polygons and stream segments, ecosection boundaries and fuel waypoints. Digital data collection via georeferenced voice recordings transcribed by the observer post-survey collection have replaced paper analogue methods.

The survey is designed to capture the primary breeding period beginning in early May but is weather and climate dependent and has taken place as early as late April (April 28, 2015) and as late as mid-May (May 13, 2011). Heavy snowpack and/or cold spring temperatures can delay accessibility to open water and wetlands. The study area extent is almost 11 million hectares and takes an average 23 days (~105 flying hours) to survey.

Table 2. Species included in the study and their nesting and feeding guilds based on Baldassarre (2014) and Cornell (2015). Species-specific models for only the top ten most common species were created however less common species were accounted for in the groups outlined in Table 2.

Species Code	Common Name	Scientific Name	Species Group	Foraging Guild	Nesting Guild
AMWI	American Wigeon*	*Mareca americana*	Dabblers	Dabbler - Plants	Ground
BAGO	Barrow's Goldeneye*	*Bucephala islandica*	Divers	Surface - Dive - Insects	Cavity
BUFF	Bufflehead*	*Bucephala albeola*	Divers	Aerial Dive - Insects	Cavity

BWTE	Blue-winged Teal*	*Spatula discors*	Dabblers	Dabbler - Seeds	Ground
CAGO	Canada Goose*	*Branta canadensis*	Geese	Ground Forager - Seeds	Ground
CANV	Canvasback	*Aythya valisineria*	Divers	Surface - Dive - Plants	Floating
CITE	Cinnamon Teal	*Spatula cyanoptera*	Dabblers	Dabbler - Seeds	Ground
COGO	Common Goldeneye	*Bucephala clangula*	Divers	Surface - Dive - Insects	Cavity
COME	Common Merganser	*Mergus merganser*	Mergansers	Surface - Dive - Fish	Cavity
GADW	Gadwall	*Mareca strepera*	Dabblers	Dabbler - Plants	Ground
GWTE	Green-winged Teal*	*Anas crecca*	Dabblers	Dabbler - Seeds	Ground
HADU	Harlequin Duck	*Histrionicus histrionicus*	Divers	Surface - Dive - Insects	Ground
HOME	Hooded Merganser	*Lophodytes cucullatus*	Mergansers	Surface - Dive - Fish	Cavity
LTDU	Long-tailed Duck	*Clangula hyemalis*	Divers	Surface - Dive - Insects	Ground
MALL	Mallard*	*Anas platyrhynchos*	Dabblers	Dabbler - Seeds	Ground
NOPI	Northern Pintail	*Anas acuta*	Dabblers	Dabbler - Seeds	Ground
NSHO	Northern Shoveler*	*Spatula clypeata*	Dabblers	Dabbler - Plants	Ground
RBME	Red-breasted Merganser	*Mergus serrator*	Mergansers	Surface - Dive - Fish	Ground
REDH	Redhead	*Aythya americana*	Divers	Surface - Dive - Plants	Floating
RNDU	Ring-necked Duck*	*Aythya collaris*	Divers	Surface - Dive - Plants	Floating
RUDU	Ruddy Duck	*Oxyura jamaicensis*	Divers	Surface - Dive - Insects	Ground
SCAU**	Lesser Scaup*	*Aythya affinis*	Divers	Surface - Dive - Insects	Ground
SNGO	Snow Goose	*Anser caerulescens*	Geese	Ground Forager - Plants	Ground
TRUS	Trumpeter Swan	*Cygnus buccinator*	Swans	Dabbler - Plants	Ground
WFGO	White-fronted Goose	*Anser albifrons*	Geese	Dabbler - Plants	Ground
WODU	Wood Duck	*Aix sponsa*	Dabblers	Dabbler - Plants	Ground

* Species within the top ten most commonly observed for which a species-specific model was created.

** The waterfowl population survey does not distinguish scaup species however only lesser scaup are observed in the area.

Table 3. Group classifications for which group-specific species distribution models were generated (refer to Table 1 for specific species included within the groups).

Abbreviation	Explanation
sp_div	Index of species richness—number of unique species
sp_tot	Count of total indicated breeding population of all waterfowl
dabblers	Count of total indicated breeding population of dabbling ducks
divers	Count of total indicated breeding population of diving ducks
cavity	Count of total indicated breeding population of cavity nesting species

Population estimates

The surveys are conducted by helicopter along meandering paths at altitudes and speeds lower than fixed-wing aircraft (30-50 m and 40-80 km/h, respectively) therefore no complementary ground surveys are conducted and no visibility correction factor is applied as visibility is assumed to be complete. The total indicated breeding population estimates were derived from raw counts based on species, sex and grouping as outlined in Smith (1995). The temporal subset of the analysis was from 2007 to 2017 inclusive (data from 2006 was excluded due to data inconsistencies and 2018 was excluded due to the absence of interpreted climate data). Survey observation points record the location of the observer within the helicopter and not the bird on the ground. Efforts to correct locations guided by reference data and field observation notes were discontinued from 2012 onward due to resource constraints and inconsistent observational record-keeping. In order to account for this spatial uncertainty, observation location points were collated to the nearest 400m interval along the center line of the strip transect.
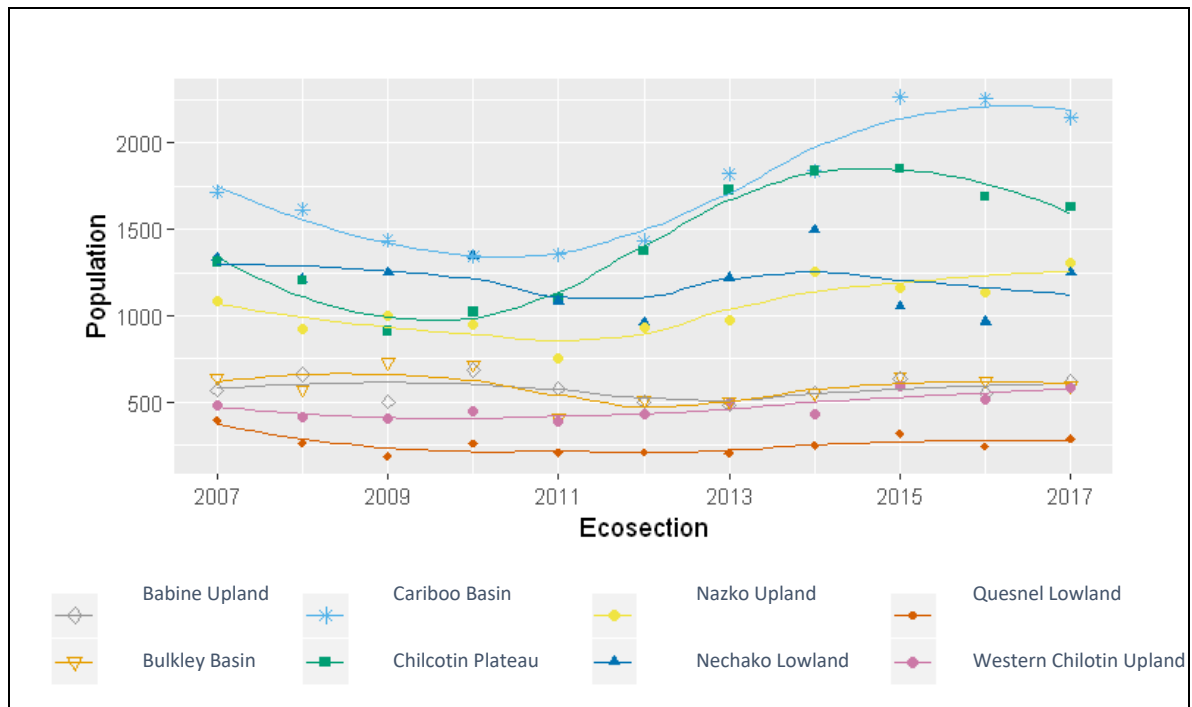
Figure 2. Total indicated breeding population of the top ten most common species observed within the transect (2007-2017).

The Cariboo Basin is by far the most populous ecosection with the greatest total count (Figure 2) and population density (Figure 3) followed by the Chilcotin Plateau, Nechako Lowland and Nazko Upland. The remaining ecosections—Bulkley Basin, Quesnel Lowland, Western Chilcotin Upland and Babine Upland—have similar population densities with the lattermost consistently the least populous. These ecosections share similarly stable year-to-year trends while the densities in the two most populous ecosections, Cariboo Basin and Chilcotin Plateau, display concordant fluctuations and appear to be increasing.
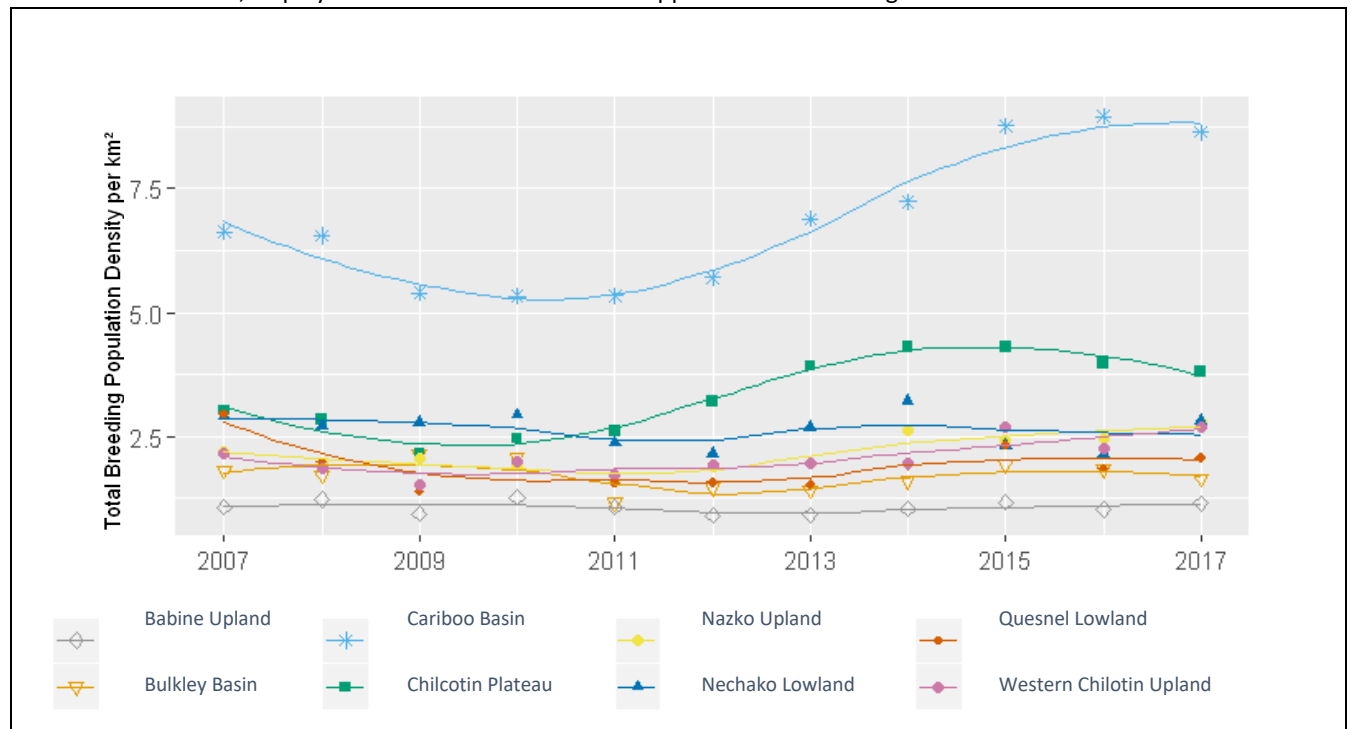
Figure 3. Total annual indicated breeding population density per square kilometre of the top ten most common species by ecosection observed within the transect.

Figure 4. Boxplots of total annual indicated breeding population density per square kilometre of the top ten most common species by ecosection observed within the transect.

Figure 5. Total indicated breeding population of the top ten most common species observed within the transect (2007-2017).

Of the top ten most commonly observed species, the species demonstrating the greatest year-to-year variability in abundance include Lesser Sc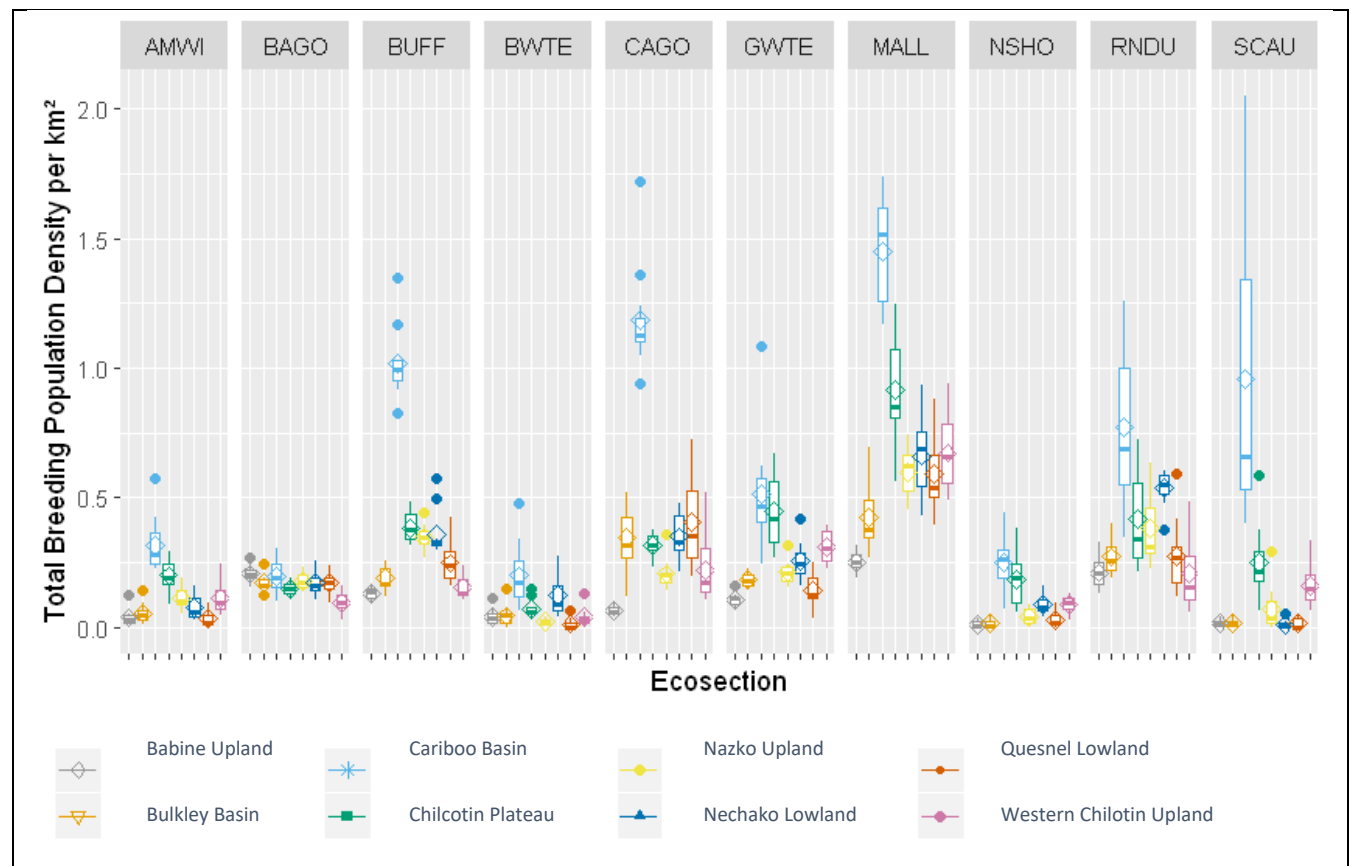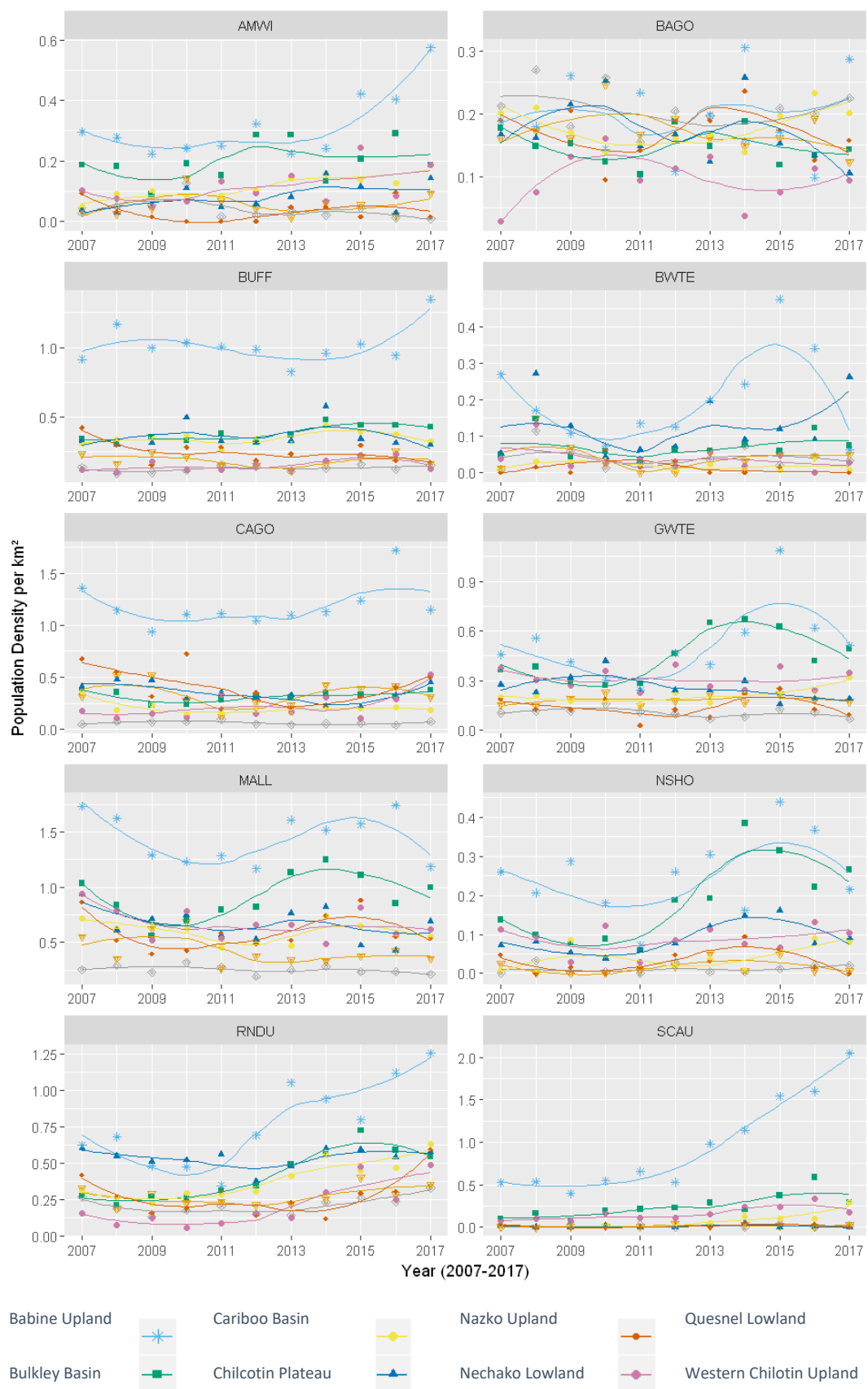aup* (LESC), Ring-necked Duck (RNDU), Canada Goose (CAGO), and Mallard (MALL) (Figure 5). The distributions suggest it is LESC and RNDU species driving the increasing population trends in the Cariboo Basin and the Chilcotin Plateau. Mallard (MALL), Canada Goose (CAGO), Green-Winged Teal (GWTE), Blue-Winged Teal (BWTE), and Northern Shoveler (NOSH) reflect concordant annual fluctuations—decreasing from 2007 to 2010, increasing between 2010 and 2015, and decreasing again to 2017. American Wigeon (AMWI) to a lesser degree follows the main trend but appears to be slightly increasing in the Cariboo Basin. Barrow's Goldeneye (BAGO) distributions remained steady throughout their distributions in all ecosections during the study period.

Environmental data

Environmental predictor variables were pre-selected based on ecological theory of life history traits and physiological processes: hydrological features associated with wetlands, and land use practices and disturbances that can potentially influence waterfowl habitat (Table 2). Where possible direct measures of predictor variables rather than proxy data was selected. For example, measures of temperature and precipitation rather than the indirect measure of elevation were included in the analyses. A number of datasets were collated and evaluated but the final data inputs were constrained by availability, resolution, coverage, and currency, and complicated by interpretation and relevance (Appendix 1). Technical constraints—processing hardware and time—necessitated limiting the number of candidate predictors to improve both interpretation and efficiency however more accurate predictions may be generated with fewer restrictions.

Annual, seasonal, monthly and 30-year normal climatic variables were extracted from ClimateBC software (version 6.10, Wang, Hamann, Spittlehouse, & Carroll, 2016) based on the coordinates and elevation at 400m interval points along the transect center line. Growing degree days and average April temperature were included to capture primary productivity. To represent and characterize hydrological regimes which are driven mainly by snowpack accumation in winter within the study area (Demarchi 2011; Islam et al. 2017; R. Pike et al. 2010) precipitation as snow and the climatic moisture deficit (precipitation minus potential evapotranspiration) were selected. The Biogeoclimatic Zones of British Columbia (BEC) is a standardized provincial dataset that classifies ecosystems within nested classes of regional, site and chronological levels of apex vegetaion (Ministry of Forests, Lands, Natural Resource Operations and Rural Development (FLNRO), 2018) and was included to represent the combined influence of soil chemistry, vegetation, topography and climate. Additionally, the BEC Zone classifications provide an alternative regional classification system to the Ecoregions of BC ecosections for supplemental model development. Moreover, predicted changes to BEC Zone boundaries due to predicted climate change scenarios can be extracted from ClimateBC for future climate impact studies.

Land cover variables to characterize habitat were derived from 2010 Landsat imagery published by the North American Land Change Monitoring Sytem (CCRS/CCMEO/NRCan 2017). Cover classes were aggregated and reclassified to account for imbalanced classes (Appendix #). Topographic data included slope and aspect derived from 1:20,000 DEM (FLNRO, 2014).

Hydrological variables were derived from the provincial reference dataset for standardized hydrological features, and included polygons of lakes, rivers, man-made waterbodies and wetlands as well as stream lines (FLRNO, 2011). Lakes were classified by size class and streams were aggregated by stream order values (Appendix 2). To account for shoreline complexity and avoid correlation a lake perimeter to area index was derived.

The line network of unpaved roads was included to represent anthropogenic disturbance of resource extraction activities (Ministry of Forests, Lands, Natural Resource Operations and Rural Development, 2013).

Land management practices were represented by the Agricultural Land Reserve (Agricultural Land Commission, 2018)—provincial designation designed to identify and conserve agricultural productivity, and by the Protected Areas Database of lands under federal, provincial and municipal protection and land conservancy trusts (CWS, 2018).

Table 4. Environmental predictor variables selected for model inputs.

Predictor - abbreviation	Predictor - description	Resolution	Time Period	Source	Calculation
fwa_1	Freshwater Atlas - Lakes < 1 ha	1:20,000	Static (Variable)	Data BC	Total area within 1.2 km circular radius of centroid point along 400m interval of transect
fwa_2	Freshwater Atlas - Lakes 1-2 ha	1:20,000	Static (variable)	Data BC	Total area within 1.2 km circular radius of centroid point along 400m interval of transect
fwa_3	Freshwater Atlas - Lakes 3-5 ha	1:20,000	Static (variable)	Data BC	Total area within 1.2 km circular radius of centroid point along 400m interval of transect
fwa_4	Freshwater Atlas - Lakes 5-10 ha	1:20,000	Static (variable)	Data BC	Total area within 1.2 km circular radius of centroid point along 400m interval of transect
fwa_5	Freshwater Atlas - Lakes 10-20 ha	1:20,000	Static (variable)	Data BC	Total area within 1.2 km circular radius of centroid point along 400m interval of transect
fwa_6	Freshwater Atlas - Lakes 20-50 ha	1:20,000	Static (variable)	Data BC	Total area within 1.2 km circular radius of centroid point along 400m interval of transect
fwa_7	Freshwater Atlas - Lakes 50-100 ha	1:20,000	Static (variable)	Data BC	Total area within 1.2 km circular radius of centroid point along 400m interval of transect
fwa_8	Freshwater Atlas - Lakes > 100 ha	1:20,000	Static (variable)	Data BC	Total area within 1.2 km circular radius of centroid point along 400m interval of transect
fwa_10ha_plus	Freshwater Atlas - Lakes > 10 ha	1:20,000	Static (variable)	Data BC	Total area within 1.2 km circular radius of centroid point along 400m interval of transect
shorecx_10less	Freshwater Atlas - Shoreline index of lakes < 10 ha	1:20,000	Static (variable)	Data BC	Ratio of total perimeter to area of lakes, multiplied by 100,000
shorecx_10plus	Freshwater Atlas - Shoreline index of lakes > 10 ha	1:20,000	Static (variable)	Data BC	Ratio of total perimeter to area of lakes, multiplied by 100,000
str_s	Freshwater Atlas - Stream order 1-3	1:20,000	Static (variable)	Data BC	Total length of streams of stream orders 1 to 3
str_m	Freshwater Atlas - Stream orders 4-6	1:20,000	Static (variable)	Data BC	Total length of streams of stream orders 4 to 6
fwa_w	Freshwater Atlas - Wetlands	1:20,000	Static (variable)	Data BC	Total area of wetlands within 1.2 km circular radius of centroid point along 400m interval of transect
fwa_r	Freshwater Atlas - River	1:20,000	Static (variable)	Data BC	Total area of rivers within 1.2 km circular radius of centroid point along 400m interval of transect
aspect	Compass direction	1:20,000	Static (2011)	Data BC	Average aspect within 400 x 400 m areal interval of strip transect
slope	Topographic slope	1:20,000	Static (2011)	Data BC	Average slope within 400 x 400 m areal interval of strip transect
alr	Agricultural Land Reserve	1:20,000	Static (2014)	Data BC	Total area within 1.2 km circular radius of centroid point along 400m interval of transect

pa	Protected Areas	1:20,000	Static (2018)	CWS Conservation Areas DB	Total area within 1.2 km circular radius of centroid point along 400m interval of transect
dra_u	Digital Road Atlas, Unpaved Roads	1:20,000	Static (2014)	Data BC	Total density within 1.2 km circular radius of centroid point along 400m interval of transect
cec_urban	Land cover - urban	30 m	Static (2010)	Commission for Environmental Cooperation	Total area within 1.2 km circular radius of centroid point along 400m interval of transect
cec_grassland	Land cover – grassland	30 m	Static (2010)	Commission for Environmental Cooperation	Total area within 1.2 km circular radius of centroid point along 400m interval of transect
cec_shrubland	Land cover - shrubland	30 m	Static (2010)	Commission for Environmental Cooperation	Total area within 1.2 km circular radius of centroid point along 400m interval of transect
cec_mixed_forest	Land cover - mixed forest	30 m	Static (2010)	Commission for Environmental Cooperation	Total area within 1.2 km circular radius of centroid point along 400m interval of transect
cec_broadleaf	Land cover - broadleaf (deciduous)	30 m	Static (2010)	Commission for Environmental Cooperation	Total area within 1.2 km circular radius of centroid point along 400m interval of transect
cec_needleleaf	Land cover - needleleaf (coniferous)	30 m	Static (2010)	Commission for Environmental Cooperation	Total area within 1.2 km circular radius of centroid point along 400m interval of transect
bec_zn	Biogeoclimatic Zone	1:20,000	Static (2018)	Data BC	Majority area within 400 x 400 m areal interval of strip transect
norm_dd5	30 yr normal Growing Degree Days > 5°C	Scale-free 400 m	1980-2010	ClimateBC	Climatic measure at coordinate position and elevation along 400m interval of transect
norm_cmd	30 yr normal Cumulative Moisture Deficit	Scale-free 400 m	1980-2010	ClimateBC	Climatic measure at coordinate position and elevation along 400m interval of transect
pas_wt	Mean Precipitation as Snow in winter	Scale-free 400 m	Annual	ClimateBC	Climatic measure at coordinate position and elevation along 400m interval of transect averaged over survey period
tave04	Average April Temperature	Scale-free 400 m	Annual	ClimateBC	Climatic measure at coordinate position and elevation along 400m interval of transect averaged over survey period

Geoprocessing Methods

Spatial analyses were performed and mapping products were produced in ArcGIS Pro (versions 2.2 to 2.4, ESRI, 2019) and Python (version 3.6.5, Python Software Foundation, 2018). Predictor variables were extracted, projected, rasterized and generalized as required in BC Albers equal area coordinate system within a 1 km buffer of the study area to eliminate edge effects. The location uncertainty of point observations determined the finest scale of the analysis to be a resolution of 400m. Conceptually, survey strip transects were segmented into 400m intervals forming 400m x 400m (16 ha) grid cells—16, 540 in total. Topographic values of slope and aspect were generalized to the mean average within each grid cell. BEC zone classifications were generalized

by majority area within the cell and remained categorical. All remaining predictors were represented by continuous values and generalized to a 1.2 km radius of the interval centroid to capture landscape level effects (Figure 6). Predicted response values were projected to a fishnet grid of 400m cell centroid points with attributed predictor values and rasterized for display. Many of the steps required in the pre-processing of spatial data were automated in a Python Script Toolbox (Appendix 3).
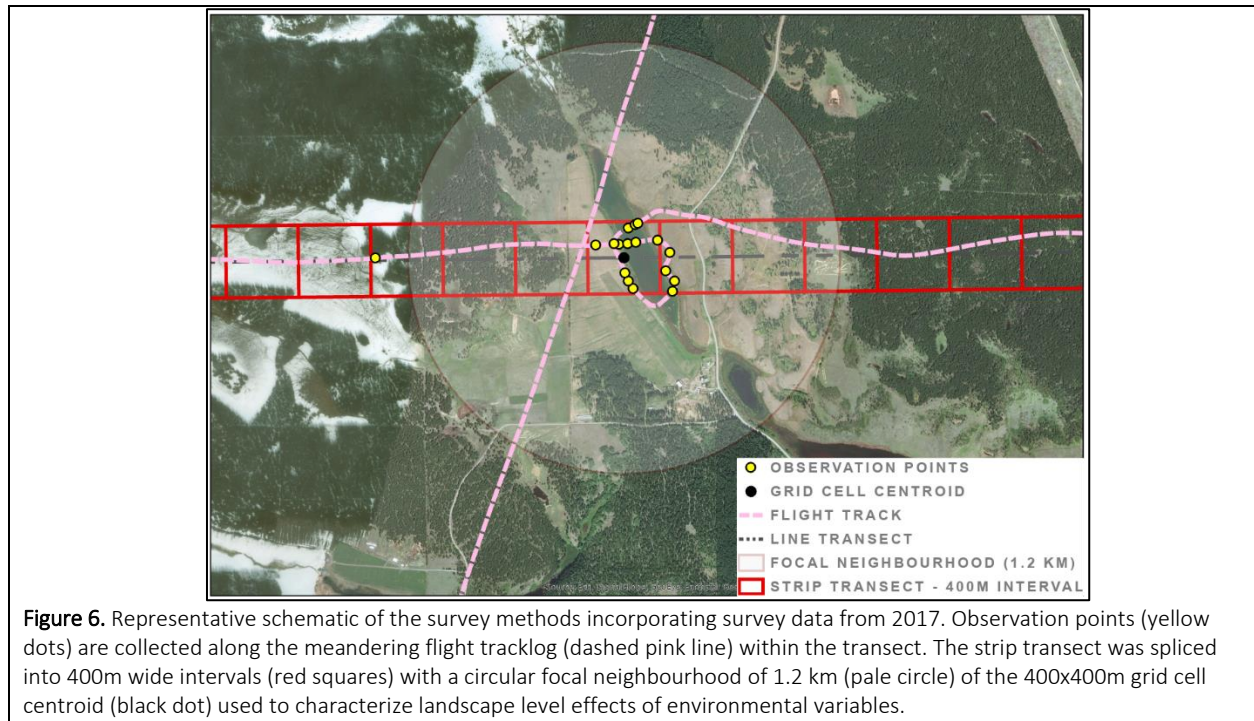
Figure 6. Representative schematic of the survey methods incorporating survey data from 2017. Observation points (yellow dots) are collected along the meandering flight tracklog (dashed pink line) within the transect. The strip transect was spliced into 400m wide intervals (red squares) with a circular focal neighbourhood of 1.2 km (pale circle) of the 400x400m grid cell centroid (black dot) used to characterize landscape level effects of environmental variables.

Statistical Methods

Random forest is an ensemble machine learning algorithm that creates a series of decision trees based on the principles of bagging (or bootstrapped aggregation—the drawing of a large number of data samples by random sampling with replacement) and random permutation. Each individual decision tree is based on a random subset of the data and at each node of the tree the data is partitioned into two branches based on a randomly selected but predetermined number of predictor variables evaluated by the algorithm to produce the best split. As individual trees are highly susceptible to noise in the data and sensitive to local optima, each tree is considered a 'weak learner' (Guisan, Wilfried, and Zimmermann 2017). But by combining and averaging the results of several regression trees, a 'strong learner' is created in the final ensembled prediction.

A random forest-based approach was selected for its predictive accuracy, resistance to overfitting, ability to account for imbalanced classes, ability to handle both continuous and categorical variables, and its independence from requirements of feature scaling and centering as well as assumptions of normality (Cutler, Cutler, and Stevens 2012; Guisan, Wilfried, and Zimmermann 2017). The observation frequency distribution of bird counts reflected a zero-inflated negative binomial distribution typical of ecological count data (Qian 2010) but not amenable to standard regression-based approaches. Preliminary explorations of zero-inflated generalized mixed models indicated it is a promising approach for future explanatory model building.

All statistical modeling and data manipulation was performed in R (version 3.5.3, R Core Team, 2018). The R 'party' package (version 1.3-3) was selected for its implementation of the random forest and bagging algorithm `cforest_unbiased` function which utilizes conditional inference trees that account for correlation structures between variables in the permutation scheme of variable selection; additionally, the function implements a permutation importance measure, function `varimp`, that is immune to erroneous calculations due to correlated responses and is not biased towards continuous data or categorical variables with many classifications unlike the Gini accuracy of traditional random forest implementations (Strobl et al. 2008; Strobl,

Hothorn, and Zeileis 2009). Slight differences in selected predictor variables between ecosection models were due to zero or near-zero variance, correlated data structures and/or mismatched factor variables in ecosection-based data inputs for model training and prediction.

Preliminary trial results of model inputs of annual survey data indicated low variable importance rankings for the dynamic climate variables. As all other environmental data was static the model input values were based on mean averaged counts and climate measures. Due to high zero-inflation (average of 93.6% frequency of zeroes for the top ten most common species) the data was not subset into training, validation, and test sets to evaluate model performance, instead the internal out-of-bag (OOB) error measures were determined. As each tree is based on a random subset of the training data, a collection of datasets which do not contain a particular record can be ensembled for each record. This collection forms the out-of-bag examples which are used as an unbiased test set to assess prediction accuracy by averaging the error rate. The predetermined number of candidate variables for each node split ('mtry' parameter of `cforest`) was left at the default value of 5 which resulted in better OOB estimates than 8 which was tested using Briemann's rule of thumb to divide the number of candidate variables by three for regression trees (Guisan, Wilfried, and Zimmermann 2017). The stability of 'varimp' ranks helped to inform the number of trees generated for each model which was set to 5,000.

Separate models for each species or species group were developed for each ecosection. Preliminary trials of generalized models on the entire study area indicated significant ecosection differences in predictor variables. Additionally, the computational intensity of the algorithm required for the entire study area was greater than available resources.

The conditional inference trees utilized by `party` package are insensitive to highly correlated data structures but in order to reduce processing time, data preparation included addressing correlation and multicollinearity by step-wise elimination of correlated variables with threshold measures of r = 0.8 followed by elimination of variance inflation factors of 5 and greater corresponding to general thresholds recommended by Guisan et al (2017). The 'varimp' function to derive variable importance can be parameterized to produced unbiased importance measures of highly correlated data however preliminary trial results in Cariboo Basin reflected exaggerated processing times that restricted the application in this study. Neither correlation nor multicollinearity in predictor variables reduces predictive accuracy of random forests however the candidate predictors removed in the preprocessing step for the generation of 'varimp' values were not re-incorporated in the final prediction models. It is recommended these be included in future modeling exercises. The predictive models were re-run setting different seed values in order to confirm the stability of importance measures.

Random forest methods are widely recognized as fast and able to handle large amounts of data and model variables but the unbiased algorithm of the `cforest` implementation is more computationally intensive than standard approaches. For example model training took approximately 10 minutes to process ~3,000 records and 27 variable inputs for mallards in Babine Upland, while model forecasting to the ecosection, an area ~ 40 times greater, took over 18 hours on a 64-bit OS workstation with 48.0 GB RAM and an Intel Xeon(R) CPU 3.60GHz. The R package 'caret' is designed specifically for data preprocessing and model generation and contains a number of machine learning methods including a more traditional random forest implementation and future studies are recommended these be explored.

The R scripts developed for statistical analyses as well as links to a Github repository of latest project updates and an interactive Jupyter Notebook documenting major processing steps are provided in Appendix 4.

RESULTS AND DISCUSSION

Variable Importance

Predictive modelling algorithms like random forest forsake the theoretical hypotheses of explanatory causal models for accurate forecasting (Shmueli 2011). The variable importance measures derived from recursive

partitioning methods are unlike regression coefficients in that they do not provide a linear measure of the relationship between the predictor and response variables, rather the reported measure reflects the drop in prediction accuracy of the model by random permutation of the variable (Guisan, Wilfried, and Zimmermann 2017; Strobl et al. 2008). The most unambiguous application of these values is in variable selection for model building. In summary, variable importance values are not by themselves necessarily interpretive however they may reveal causal mechanisms to inform the development of explanatory models (Shmueli, 2011). Predictor variables were ranked by their variable important measures within each species-ecosection model. The influence of predictors varied between specific ecosections and species but overall the top variable ranks were hydrological—shoreline to lake area index of lakes less than 10 ha in size, area of wetland, shoreline to area index of lakes greater than 10 ha, climatic moisture deficit, and slope which is an indirect measure of potential water pooling followed by small streams, growing degree days and needleleaf land cover (Table 5 and 6, for ecosection and species summary tables, respectively). Lakes smaller than 10 ha do not appear in the overall top five but this is due to their breakdown into four separate smaller classes (fwa_1 through 4). Preliminary trials with a grouped classification of all lakes less than 10 ha indicated a very significant influence that warranted a breakdown into smaller classes.

Predictor variables with the lowest averaged ranking included protected areas, rivers, BEC zone, Agricultural Land Reserve and urban land cover—each of these have low to near-zero variance within the study area extent and are patchily distributed.

Table 5. Sequential average variable importance rank for each ecosection ordered by the overall average rank from high to low. Note that some variables were not input into some ecosecton models.

Predictor	BAU	BUB	CAB	CHP	NAU	NEL	QUL	WCU	Average
shorecx_10less	4	5	3	3	5	6	5	10	5
fwa_w	7	9	7	4	4	5	7	3	6
shorecx_10plus	5	3	6	5	9	5	12	12	7
norm_cmd	NA	7	9	13	6	NA	NA	NA	9
slope	10	6	11	11	7	6	11	11	9
str_s	12	7	3	7	13	13	6	19	10
norm_dd5	9	13	8	14	13	11	5	7	10
needleleaf	13	13	NA	NA	10	7	9	NA	10
fwa_1	8	12	13	10	12	12	9	11	11
fwa_4	10	13	8	10	12	12	11	11	11
fwa_3	11	12	8	5	12	15	16	9	11
fwa_10ha_more	NA	NA	12	14	NA	NA	14	6	11
tave04	NA	NA	NA	NA	12	NA	NA	NA	12
pas_wt	12	12	15	11	17	12	14	13	13
fwa_2	12	23	14	10	9	16	11	18	14
dra_u	16	19	17	17	18	13	10	8	15
broadleaf	13	14	17	18	15	8	18	15	15
str_m	19	10	20	12	12	13	16	21	15
mixed_forest	6	15	20	17	19	16	18	13	15
shrubland	17	11	11	18	20	11	16	20	16
aspect	13	13	20	21	17	11	17	16	16
grassland	16	16	13	13	17	17	18	19	16
urban	17	20	13	23	23	13	12	9	16
alr	23	12	13	10	24	13	15	21	16
bec_zn	14	NA	25	20	13	NA	20	7	16

fwa_r	20	16	24	24	9	21	18	4	17
pa	11	21	15	17	21	21	18	17	18

Table 6. Sequential average variable importance rank for each species ordered by the overall average rank from high to low.

Predictor	AMWI	BAGO	BUFF	BWTE	CAGO	GWTE	MALL	NSHO	RNDU	SCAU	cavity	dabblers	divers	sp_div	sp_tot	Average
shorecx_10less	12	3	1	8	11	3	2	15	2	8	3	3	2	2	2	5
fwa_w	5	12	9	2	5	3	3	5	5	10	14	3	7	3	4	6
shorecx_10plus	10	2	6	12	9	14	8	14	3	5	2	10	3	5	5	7
norm_cmd	6	10	7	9	9	9	11	5	11	7	9	10	9	9	9	9
slope	7	12	14	10	6	5	7	6	11	10	15	6	13	9	8	9
str_s	15	10	8	11	14	6	7	17	8	14	9	7	7	6	8	10
norm_dd5	8	13	10	7	8	11	10	9	11	9	14	10	11	11	10	10
needleleaf	6	12	14	10	9	9	11	6	16	6	9	8	13	12	12	10
fwa_1	13	8	8	9	15	7	10	14	15	13	8	11	10	9	12	11
fwa_4	15	7	5	16	17	15	13	18	5	8	8	14	5	8	8	11
fwa_3	18	7	6	16	18	13	11	16	5	12	7	14	4	8	9	11
fwa_10ha_more	14	5	11	18	13	17	15	17	10	3	4	17	5	12	10	11
tave04	5	22	10	7	16	9	14	13	12	8	12	13	11	15	15	12
pas_wt	11	15	12	10	11	12	14	10	13	13	15	15	15	15	15	13
fwa_2	16	11	8	17	18	14	16	15	13	18	10	16	10	13	15	14
dra_u	14	12	15	15	15	14	14	14	15	15	13	16	17	17	17	15
broadleaf	15	17	18	10	9	12	15	9	18	17	19	13	19	17	16	15
str_m	12	21	20	15	8	15	11	20	18	20	14	13	17	13	14	15
mixed_forest	14	13	17	15	15	15	17	12	16	13	15	16	17	18	17	15
shrubland	12	15	16	12	13	15	16	10	20	16	18	15	19	18	18	16
aspect	16	14	19	15	13	17	19	12	17	12	16	17	17	18	18	16
grassland	14	15	17	12	15	17	20	11	16	15	17	18	19	19	18	16
urban	11	17	19	17	16	18	17	13	19	16	16	15	18	17	16	16
alr	17	21	18	13	10	18	16	14	17	16	22	15	18	18	16	16
bec_zn	18	17	16	19	21	15	15	16	14	15	17	16	16	16	16	16
fwa_r	14	21	18	17	14	18	16	14	20	19	16	15	19	17	17	17
pa	16	16	17	18	19	18	19	17	17	14	17	20	17	19	20	18

Model Performance

Model performance was assessed with the out-of-bag (OOB) estimate. The coefficient of determination (R^2) and mean absolute error (MAE) are reported in Table #. R^2 represents the measure of variance in response values that is explained by the model and corresponds to the correlation between the observed and expected values. The MAE provides a measure of the difference between the observed and expected values and is less sensitive to outliers than the root mean square error (RMSE). In general terms, the lower the MAE and higher the R^2 the better the performance.

Table 7. Model performance of each species-ecosection model estimated by the coefficient of determination (R^2) and mean absolute error (MAE) derived from out-of-bag (OOB) estimates.

Species	Babine Upland R^2	Babine Upland MAE	Bulkley Basin R^2	Bulkley Basin MAE	Cariboo Basin R^2	Cariboo Basin MAE	Chilcotin Plateau R^2	Chilcotin Plateau MAE	Nazko Upland R^2	Nazko Upland MAE	Nechako Lowland R^2	Nechako Lowland MAE	Quesnel Lowland R^2	Quesnel Lowland MAE	Western Chilcotin Lowland R^2	Western Chilcotin Lowland MAE
AMWI	0.10	0.01	0.07	0.02	0.09	0.08	0.18	0.05	0.21	0.03	0.07	0.02	0.02	0.01	0.26	0.02
BAGO	0.26	0.05	0.20	0.04	0.17	0.05	0.11	0.04	0.27	0.04	0.13	0.03	0.09	0.05	0.13	0.03
BUFF	0.18	0.03	0.24	0.04	0.30	0.20	0.23	0.09	0.24	0.08	0.16	0.09	0.10	0.06	0.23	0.04
BWTE	0.04	0.01	0.05	0.01	0.09	0.05	0.14	0.02	0.13	0.01	0.13	0.03	0.02	0.00	0.14	0.01
CAGO	0.08	0.02	0.02	0.00	0.15	0.26	0.18	0.07	0.13	0.05	0.18	0.09	0.05	0.11	0.05	0.06
GWTE	0.11	0.03	0.16	0.05	0.17	0.11	0.19	0.10	0.19	0.05	0.13	0.07	0.04	0.04	0.21	0.07
MALL	0.15	0.06	0.15	0.10	0.24	0.26	0.26	0.17	0.24	0.12	0.20	0.14	0.10	0.14	0.28	0.13
NOSH	0.01	0.00	0.02	0.00	0.11	0.07	0.19	0.05	0.18	0.01	0.10	0.02	0.01	0.01	0.22	0.02
RNDU	0.18	0.05	0.20	0.06	0.20	0.16	0.23	0.09	0.20	0.09	0.21	0.12	0.06	0.07	0.12	0.05
SCAU	0.01	0.00	0.07	0.00	0.20	0.22	0.21	0.06	0.15	0.02	0.05	0.00	0.01	0.01	0.21	0.04
Dabblers	0.16	0.10	0.16	0.16	0.21	0.61	0.29	0.36	0.26	0.21	0.19	0.27	0.08	0.19	0.36	0.25
Divers	0.30	0.16	0.31	0.17	0.29	0.64	0.28	0.25	0.30	0.23	0.25	0.28	0.12	0.20	0.28	0.15
Cavity-Nesters Species	0.26	0.07	0.23	0.06	0.18	0.07	0.11	0.05	0.27	0.06	0.22	0.07	0.09	0.07	0.15	0.03
Diversity	0.29	0.09	0.35	0.12	0.37	0.34	0.37	0.20	0.39	0.14	0.28	0.17	0.18	0.15	0.43	0.11
All Species	0.27	0.24	0.22	0.36	0.27	1.37	0.33	0.63	0.33	0.43	0.24	0.55	0.10	0.43	0.36	0.40
Average	0.16	0.06	0.16	0.08	0.20	0.30	0.22	0.15	0.23	0.10	0.17	0.13	0.07	0.10	0.23	0.09
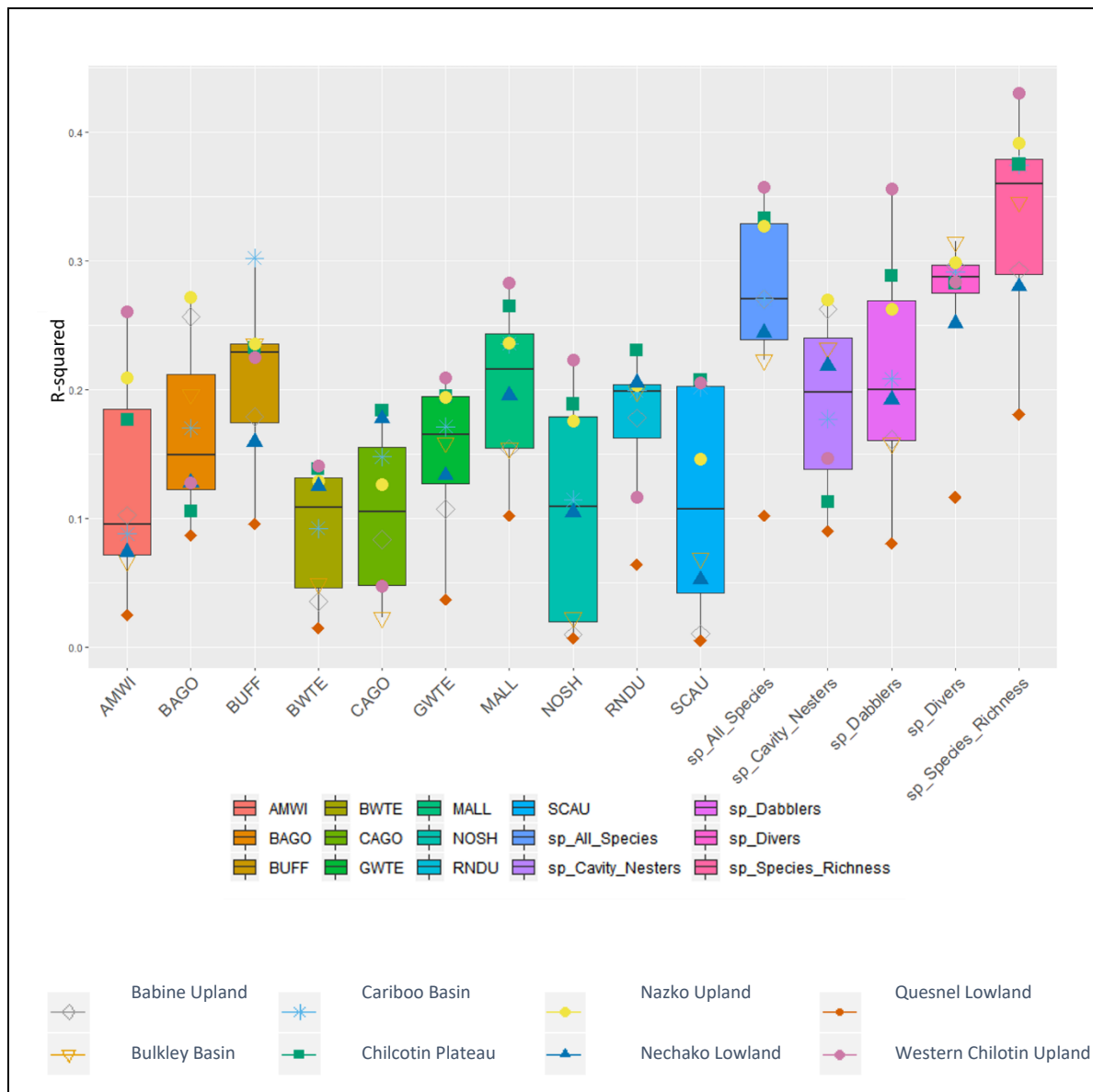
Figure 7. Boxplots of the coefficient of determination (R^2) of each species-ecosection specific model.

Overall, the models on average explained 18% of the variation in species distribution (Figure 7). The generalized groups representing all species("sp_All_species") and species richness ("sp_Species_Richness") performed best followed by divers, dabblers and cavity nesters. The least variability in R^2 values was for divers. Diving ducks tend to prefer larger, deeper lakes that are likely to be better represented in the data than smaller wetlands. Models for Western Chilcotin Upland, Chilcotin Plateau and Nazko Upland performed best while Quesnel Lowland consistently scored poorly. Quesnel Lowland is the smallest ecosection in the study area and the predictors selected do not appear to have sufficient explanatory variability. While the models for Western Chilcotin Upland and Chilcotin Plateau performed best overall their performance for cavity-nesting species, Barrow's goldeneyes and Bufflehead, were relatively poor. Cavity-nesting duck species do not excavate their own nests and rely on abandoned (or natural) cavities within aspen or mixed aspen/coniferous forests (Baldassarre, 2014). Buffleheads as North America's smallest diving duck, prefer entrance holes excavated by northern flickers which are smaller than those made by pileated woodpeckers to which goldeneyes are restricted (Baldassarre, 2014). Model parameters for cavity-nesting species may be improved with species distributions of these cavity excavating land birds.
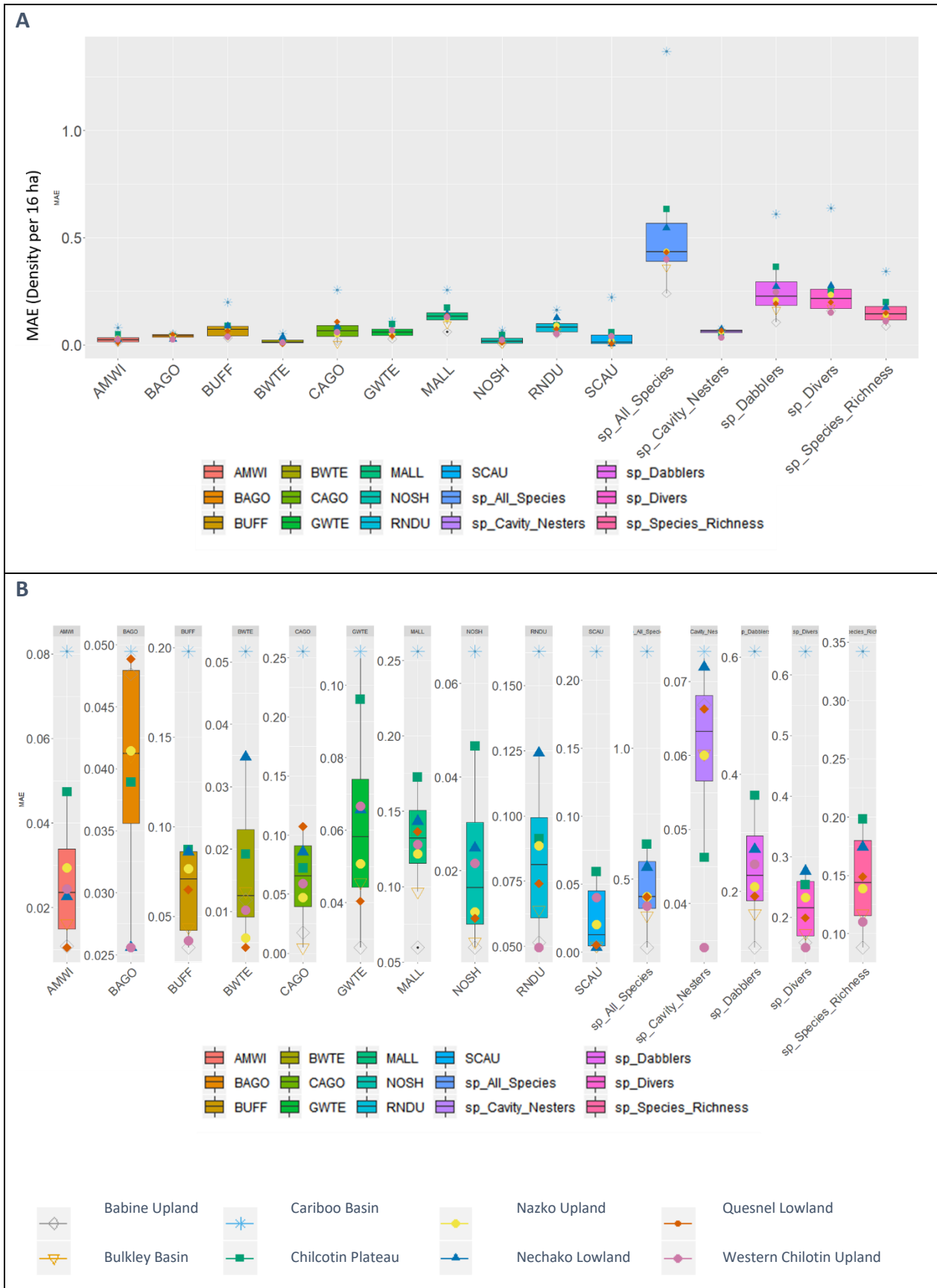
Figure 7. Boxplots of the mean absolute error (MAE) of each species-ecosection specific model (A. Fixed scale, B. dynamic scale range values).

The overall average MAE was 13%, but this varied widely between species and ecosections. The generalized group classifications excluding cavity-nesters had the largest values and greatest range in variability between ecosections. Cariboo Basin ecosection consistently scored the highest error rate. High error rates are likely amplified due to the zero-inflation of observation frequencies—birds are patchily distributed throughput the landscape with zero observations in most cells punctuated by hotspots of high abundance resulting in higher values for the more population dense ecosections and lower values in the least populous.

Model Limitations

We recognize "[d]ucks like water" (Pimm 1994), but these waters have yet to be mapped in detail: Canada, unlike most industrialized nations, does not have a national wetland dataset (Canadian Wetland Inventory — Ducks Unlimited Canada, 2019). The delineation of highly productive wetlands, such as ephemeral wetlands due to increased nutrient mineralization by aerobic microbes (Schlesinger and Bernhardt 2013) and smaller wetlands due to increased light penetration at shallower depths and greater shoreline emergent vegetation, are difficult to capture with freely available, coarser-scale remote sensing imagery (e.g. Landsat imagery at 30m resolution). Techniques for finer-scale wetland feature extraction based on Landsat have been developed for open wetlands but have not yet been fully developed and tested on forested wetlands (Halabisky et al. 2018).

Relative to other provinces, BC has an abundance of fine-scale (1:20,000) base reference spatial data however the creation of much of these data has been driven by the forestry sector which is a significant component of the BC economy. Consequently, much of the mapping of environmental features has been focused on supporting forestry management practices and the delineation of wetlands (areas unsuited to logging) especially at higher elevations is poor (D. Filatow, personal communication April 28, 2018). Moreover, despite mapping standards, BC TRIM data was produced on a mapsheet by mapsheet basis and variability exists between areas. There are ongoing efforts to develop methodologies for wetland mapping coincident with the May surveys utilizing Radarsat-2 technologies in a collaborative effort between the CWS and the Science and Technology branches of Environment and Climate Change Canada which are promising but results are still preliminary and the project is far from complete (K. Moore, personal communication, September 30, 2018). Additionally, the Canadian Wetland Inventory project spearheaded by Ducks Unlimited Canada aims to provide a comprehensive national wetland inventory but BC has yet to be mapped (Canadian Wetland Inventory — Ducks Unlimited Canada, 2019). Global wetland mapping projects include the European Space Agency's Climate Change Initiative which has a number of exciting projects and remote sensing products of varying temporal and spatial resolution and should be monitored for updates (European Space Agency, 2019). Model performance and predictive accuracy can be expected to improve with the incorporation of improved wetland cover products, but proactive management measures should not be delayed while waiting for these data.

Hydrological regimes that create and maintain wetland ecosystems are influenced by a host of factors such as the depth of snowpack, the rate of snowmelt, levels of groundwater, precipitation, glacial retreat, land management practices, and anthropogenic and natural disturbance (Adamus 2014; R. G. Pike et al. 2010). Such complexities are challenging to represent but future studies may be improved by incorporating snow basin, drainage and/or watershed data. Additionally, BC's forested ecosystems have been impacted by the mountain pine beetle outbreak from 1999 to 2015 followed by the extreme fire seasons of 2017 and 2018. The short and longer-term effects of these regional disturbances on forest hydrology are difficult to predict (R. Pike et al. 2010; R. G. Pike et al. 2010) but are worthy of monitoring and future investigation (Bunnell, Fred L, Wells, Ralph 2010; MacKenzie and Moran 2004; Snauffer, Hsieh, and Cannon 2016).

The population estimates reflect sampled observations over a limited period of time and therefore reflect only a snapshot of the species-habitat relationship assumed to be in pseudo-equilibrium (Guisan, Wilfried, and Zimmermann 2017). Moreover, the models did not account for density-dependence, breeding philopatry, site fidelity, territoriality, influence of breeding phenology, species nesting chronologies or lagged response to environmental shifts. A recent study on the breeding phenology of cavity-nesting birds identified observable impacts on nesting activities with critical temperature periods of local temperature as short as 4 days (Drake

and Martin 2018). Daily temperature datasets from Natural Resources Canada at 1 km resolution (McKenney et al. 2011) were collated and extracted but not included in the generation of the models as these data were considered more useful as explanatory variables than predictive determinants. Preliminary trials indicated significant variable importance estimates for averaged weekly temperatures and future efforts are encouraged to explore the incorporation of these and other datasets as identified in Appendix 1.

Species Abundance and Distribution

<insert discussion of predicted distributions and abundance of mapped outputs>

Model Applications

This modeling study had two main, related objectives: first, develop the methods and techniques required to standardize the BC May Survey dataset for analysis and second, create SDMs of relative abundance and distribution of breeding waterfowl to support conservation planning and habitat management. The predicted distribution maps can be used to highlight hotspot areas of high abundance and diversity to evaluate and assess protection or mitigation measures, provide finer-scale population estimates for environmental impact assessments, and help guide adaptive management measures to prepare for climate change. Additonally, the results can inform future studies of ecological relationships, and serve as guidance further model development that addresses the models' limitations.

Changes in BC's hydrological regimes due to increased warming and drying trends are predicted to lead to wetland losses at low to mid-elevations. As higher elevations are less sensitive to temperature changes that can affect snowpack accumulation, BC's higher elevation wetlands may buffer the impacts for waterfowl and other wetland species (Bunnell, Fred L, Wells, Ralph 2010; R. Pike et al. 2010; R. G. Pike et al. 2010). The breeding habitat within BC's Central Interior is by no means as productive as the Prairie Pothole Region (PPR), However the conservation value of our wetlands, especially at elevations greater than 1200 metres (Bunnell, Fred L, Wells, Ralph 2010), may rise with the present and predicted climate-related shifts in wetland productivity of the PPR (N. D. Niemuth, Fleming, and Reynolds 2014; N. Niemuth, Wangler, and Reynolds 2010; Zhao et al. 2016). Future climate impact studies should take into account the wide range in variability of projections between the global models available and employ ensemble methods that encompass this range of variability to account for uncertainty in impact assessments and adaptive management strategies (R. G. Pike et al. 2010; Spittlehouse and Wang 2016).

CONCLUSION

"The single story creates stereotypes, and the problem with stereotypes is not that they are untrue, but that they are incomplete. They make one story become the only story" (Adichie 2009).

With the quote above, the Nigerian author Chimamanda Ngozi Adichie was referring to how a limited perspective can lead to flawed conclusions and racial stereotyping; how any single observation however true is not representative. Models, like stories, must be developed and communicated in meaningful ways. A species distribution model is a single story: a simplified human construct of a complex phenomenon shared to represent and reveal processes and patterns through a single lens. Future studies are encouraged to validate and build upon these results by integrating expert opinion, employing best available methods and data, addressing uncertainty, and continuing to support systematic monitoring and ecological research. These models were created to assist the CWS, governmental and other partners in conservation to assess our efforts, guide action, and inspire us to ask more questions to better our understanding of fundamental biological and ecological theory (Sinclair et al, 2010) to inform planning and decision-making. This study is a single story but it is a story worth telling in support of waterfowl conservation in British Columbia.

ACKNOWLEDGEMENTS

REFERENCES

Adamus, Paul. 2014. "Effects of Forest Roads and Tree Removal In or Near Wetlands of the Pacific Northwest: A Literature Synthesis." (December).

Adichie, Chimamanda Ngozi. 2009. "The Danger of a Single Story." https://www.ted.com/talks/chimamanda_adichie_the_danger_of_a_single_story?language=en.

Baldassarre, Guy. 2014. *Ducks, Geese, and Swans of North America*. Revised an. John Hopkins University Press.

Bunnell, Fred L, Wells, Ralph, Moy Arnold. 2010. "Vulnerability of Wetlands to Climate Change in the Southern Interior Ecoprovince : A Preliminary Assessment 1 Final Report." *Centre for Applied Conservation Research, University of British Columbia* (March).

"Canadian Wetland Inventory — Ducks Unlimited Canada." https://www.ducks.ca/initiatives/canadian-wetland-inventory/ (July 27, 2019).

CCRS/CCMEO/NRCan. 2017. "2010 Land Cover of North America at 30 Meters." http://www.cec.org/tools-and-resources/map-files/land-cover-2010-landsat-30m.

Commission, Agricultural Land. 2018. "Agricultural Land Reserve." https://catalogue.data.gov.bc.ca/dataset/alc-alr-boundary.

Corvallis Microtechnology Inc. 2015. "PC-Mapper with Airborne Inspection."

Cutler, A, DR Cutler, and JR Stevens. 2012. "Random Forests." In *Ensemble Machine Learning*, eds. C Zhang and Y Ma.

Demarchi, Dennis Alvin. 2011. *An Introduction to the Ecoregions of British Columbia*. Victoria. http://www.env.gov.bc.ca/wld/documents/techpub/rn324.pdf.

Drake, Anna, and Kathy Martin. 2018. "Local Temperatures Predict Breeding Phenology but Do Not Result in Breeding Synchrony among a Community of Resident Cavity-Nesting Birds." *Scientific Reports* 8(1): 2756. http://www.nature.com/articles/s41598-018-20977-y (August 1, 2019).

Elith, Jane, and Catherine H. Graham. 2009. "Do They? How Do They? WHY Do They Differ? On Finding Reasons for Differing Performances of Species Distribution Models." *Ecography* 32(1): 66–77. http://doi.wiley.com/10.1111/j.1600-0587.2008.05505.x (July 19, 2019).

Environment and Climate Change Canada. 2019. *Summary of Canada's 6th National Report to the Convention on Biological Diversity*. Gatineau, Quebec.

Environment Canada - Biodiversity Convention Office. 1995. *Canadian Biodiversity Strategy--Canada's Response to the Convention on Biological Diversity*. Hull, Quebec. http://www.biodivcanada.ca/560ED58E-0A7A-43D8-8754-C7DD12761EFA/CBS_e.pdf.

ESRI. 2019. "ArcGIS Pro."

European Space Agency. "Objective | ESA Climate Change Initiative." *Climate Change Initiative*. http://cci.esa.int/objective (August 4, 2019).

Franklin, J, and JA Miller. 2009. *Mapping Species Distributions*. Cambridge: Cambridge University Press.

Guillera-Arroita, Gurutzeta et al. 2015. "Is My Species Distribution Model Fit for Purpose? Matching Data and Models to Applications." *Global Ecology and Biogeography* 24(3): 276–92. http://doi.wiley.com/10.1111/geb.12268 (July 19, 2019).

Guisan, Antoine et al. 2013. "Predicting Species Distributions for Conservation Decisions" ed. Hector Arita. *Ecology Letters* 16(12): 1424–35. http://doi.wiley.com/10.1111/ele.12189 (April 25, 2018).

Guisan, Antoine, and Wilfried Thuiller. 2005. "Predicting Species Distribution: Offering More than Simple Habitat Models." *Ecology Letters* 8(9): 993–1009. http://doi.wiley.com/10.1111/j.1461-0248.2005.00792.x (March 4, 2018).

Guisan, Antoine, Thuiller Wilfried, and Niklaus E. Zimmermann. 2017. *Habitat Suitability and Distribution Models with Applications in R*. Cambridge University Press.

Halabisky, Meghan et al. 2018. "Harnessing the Temporal Dimension to Improve Object-Based Image Analysis Classification of Wetlands." *Remote Sensing* 10(9): 1467. http://www.mdpi.com/2072-4292/10/9/1467 (July 27, 2019).

Islam, Siraj ul et al. 2017. "Future Climate Change Impacts on Snow and Water Resources of the Fraser River Basin, British Columbia." *Journal of Hydrometeorology* 18(2): 473–96. http://journals.ametsoc.org/doi/10.1175/JHM-D-16-0012.1 (April 1, 2018).

MacKenzie, W.H., and J.R. Moran. 2004. *Wetlands of British Columbia: A Guide to Identification. Land Management Handbook 52*. http://www.for.gov.bc.ca/hfd/pubs/Docs/Lmh/Lmh52.htm.

McKenney, Daniel W. et al. 2011. "Customized Spatial Clinate Models for North America." *Bulletin of the American*

Meteorological Society (December): 1611–22.

Ministry of Forests, Lands, Natural Resource Operations, and Rural Development (Organization/Institution). 2011. "Freshwater Atlas." ftp://ftp.geobc.gov.bc.ca/sections/outgoing/bmgs/FWA_Public.

———. 2018. "Biogeoclimatic Zones of British Columbia." https://catalogue.data.gov.bc.ca/dataset/bec-map.

———. 2013. "Digital Road Atlas - Master Partially Attributed Roads." https://catalogue.data.gov.bc.ca/dataset/digital-road-atlas-dra-master-partially-attributed-roads/resource/a06a2e11-a0b1-41d4-b857-cb2770e34fb0.

———. 2014. "TRIM Contours."

Murray, Dennis L., Michael G. Anderson, and Todd D. Steury. 2010. "Temporal Shift in Density Dependence among North American Breeding Duck Populations." *Ecology* 91(2): 571–81. http://doi.wiley.com/10.1890/MS08-1032.1 (March 22, 2018).

Niemuth, ND, B Wangler, and RE Reynolds. 2010. "Spatial and Temporal Variation in Wet Area of Wetlands in the Prairie Pothole Region of North Dakota and South Dakota." *Wetlands* 30.

Niemuth, Neal D., Kathleen K. Fleming, and Ronald E. Reynolds. 2014. "Waterfowl Conservation in the US Prairie Pothole Region: Confronting the Complexities of Climate Change" ed. Ben Bond-Lamberty. *PLoS ONE* 9(6): e100034. http://dx.plos.org/10.1371/journal.pone.0100034 (March 9, 2018).

Pike, R et al. 2010. *Compendium of Forest Hydrology and Geomorphology in British Columbia*. https://www.for.gov.bc.ca/hfd/pubs/docs/lmh/Lmh66.htm (March 11, 2018).

Pike, Robin G et al. 2010. "Climate Change Effects on Watershed Processes in British Columbia." In *Compendium of Forest Hydrology and Geomorphology in British Columbia*,.

Pimm, Stuart L. 1994. "The Importance of Watching Birds from Airplanes." *Trends in Ecology & Evolution* 9(2): 41–43. https://www.sciencedirect.com/science/article/pii/016953479490264X (July 27, 2019).

Python Software Foundation. 2018. "Python Language Reference." https://www.python.org/downloads/.

Qian, Song S. 2010. *Environmental and Ecological Statistics with R*. CRC Press.

R Core Team. 2018. "R: A Language and Environment for Statistical Computing." http://www.r-project.org/.

Savard, Jean-Pierre L., W. Sean Boyd, and G. E. John Smith. 1994. "Waterfowl-Wetland Relationships in the Aspen Parkland of British Columbia: Comparison of Analytical Methods." *Hydrobiologia* 279–280(1): 309–25. http://link.springer.com/10.1007/BF00027864 (March 31, 2018).

Schlesinger, William H, and Emily S Bernhardt. 2013. *Biogeochemistry*. 3rd ed. Elsevier.

Shmueli, Galit. 2011. "To Explain or to Predict?" *Statistical Science* 25(3): 289–310.

Sinclair, SJ, MD White, and GR Newell. 2010. "How Useful Are Species Distribution Models for Managing Biodiversity under Future Climates?" *Ecology and Society* 15(1). http://www.ecologyandsociety.org/vol15/iss1/art8/.

Smith, GW. 1995. 268 Biological Science Report *A Critical Review of the Aerial and Ground Surveys of Breeding Waterfowl in North America*. http://www.ncbi.nlm.nih.gov/pubmed/11438027.

Snauffer, Andrew M., William W. Hsieh, and Alex J. Cannon. 2016. "Comparison of Gridded Snow Water Equivalent Products with in Situ Measurements in British Columbia, Canada." *Journal of Hydrology* 541: 714–26. https://www.sciencedirect.com/science/article/pii/S0022169416304577 (June 30, 2018).

Spittlehouse, Dave, and Tongli Wang. 2016. "Comparison of Climate Change Projections in ClimateBC v5 . 30." : 1–6.

Strobl, Carolin et al. 2008. "Conditional Variable Importance for Random Forests." *BMC Bioinformatics* 9(1): 307. https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-9-307 (May 26, 2019).

Strobl, Carolin, Torsten Hothorn, and Achim Zeileis. 2009. "Party on! A New, Conditional Variable-Importance Measure for Random Forests Available in the Party Package." *R Journal* 1(2): 14–17.

US Fish and Wildlife Service (USFWS) and Canadian Wildlife Service (CWS). 1987. *Standard Operating Procedures for Aerial Waterfowl Breeding Ground Population and Habitat Surveys in North America*.

Wang, Tongli, Andreas Hamann, Dave Spittlehouse, and Carlos Carroll. 2016. "Locally Downscaled and Spatially Customizable Climate Data for Historical and Future Periods for North America." : 1–17.

Zhao, Qing, Emily Silverman, Kathy Fleming, and G. Scott Boomer. 2016. "Forecasting Waterfowl Population Dynamics under Climate Change — Does the Spatial Variation of Density Dependence and Environmental Effects Matter?" *Biological Conservation* 194: 80–88. https://www.sciencedirect.com/science/article/pii/S0006320715301853 (March 22, 2018).

Zimpfer, Nathan L, Andre Breault, and Todd Sanders. 2019. *British Colulmbia Breeding Waterfowl Survey Report, 2019*.

APPENDIX 1 – Dataset Sources

Environmental datasets collated and evaluated for the modeling study. The table describes each dataset and provides a brief description of variables and their assessment.

APPENDIX 2—Reclassification

APPENDIX 3 – Python Script Toolbox
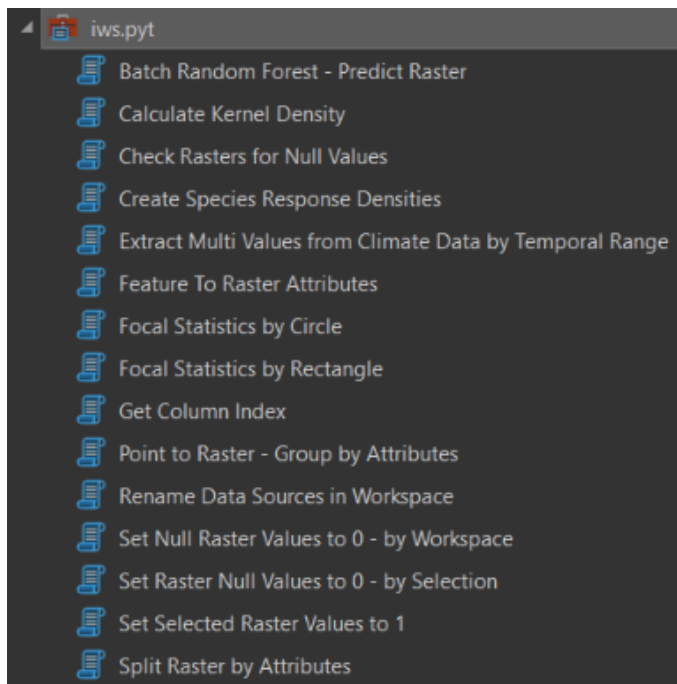
Figure 3. Python Script Toolbox as displayed in ArcGis Pro.

The code for the Python script toolbox is provided below.

```python
# -*- coding: utf-8 -*-

import arcpy, os
from arcpy import env
arcpy.env.overwriteOutput = True
from arcpy.sa import *
arcpy.CheckOutExtension("Spatial")
arcpy.env.qualifiedFieldNames = False
# scratch_ws = arcpy.CreateScratchName(workspace=arcpy.env.scratchGDB)
arcpy.env.snapRaster = r"C:\Users\hashimotoy\Desktop\ws\base_alb.gdb\dem"
arcpy.env.mask = r"C:\Users\hashimotoy\Desktop\ws\base_alb.gdb\survey_mask"
scratch_ws = r"C:\Users\hashimotoy\Desktop\ws_prep\scratch.gdb"

def checkFieldExists(in_tbl, field_nm, field_type):
    fields = arcpy.ListFields(in_tbl)
    x = False
    for field in fields:
        if field.name == field_nm:
            x = True
    if x == False:
        arcpy.AddField_management(in_tbl, field_nm, field_type)


def getBaseName(in_fc):
    desc = arcpy.Describe(in_fc)
    basename = desc.baseName
    return basename

def unique_values(table, field):
    with arcpy.da.SearchCursor(table, [field]) as cursor:
        return sorted({row[0] for row in cursor})


class Toolbox(object):
    def __init__(self):
        """Define the toolbox (the name of the toolbox is the name of the
        .pyt file)."""
        self.label = "IWS Data Tools"
        self.alias = ""

        # List of tool classes associated with this toolbox
        self.tools = [featureToRasterAttributes,
                      setRasterNullValuesTo0,
                      setRasterNullValuesTo0Workspace,
                      renameDataInWorkspace,
                      kernelDensityCalculations,
                      splitRasterByAttributes,
                      focalStatsByCircle,
                      focalStatsByRectangle,
                      setRasterValuesTo1,
                      pointToRasterGroupByAttributes,
                      getColumnIndex,
                      extractClimateValuesForRange,
                      checkRasterForNullValues,
                      batchRandomForest,
                      createPredictedDensitySurfaces]
```

```python
class Tool(object):
    def __init__(self):
        """Define the tool (tool name is the name of the class)."""
        self.label = "Tool"
        self.description = ""
        self.canRunInBackground = False

    def getParameterInfo(self):
        """Define parameter definitions"""
        # First parameter
        param0 = arcpy.Parameter(
            displayName="Input Features",
            name="in_features",
            datatype="GPFeatureLayer",
            parameterType="Required",
            direction="Input")

        # Second parameter
        param1 = arcpy.Parameter(
            displayName="Sinuosity Field",
            name="sinuosity_field",
            datatype="Field",
            parameterType="Optional",
            direction="Input")

        param1.value = "sinuosity"

        # Third parameter
        param2 = arcpy.Parameter(
            displayName="Output Features",
            name="out_features",
            datatype="GPFeatureLayer",
            parameterType="Derived",
            direction="Output")

        param2.parameterDependencies = [param0.name]
        param2.schema.clone = True

        parameters = [param0, param1, param2]

        return parameters

    def isLicensed(self):
        """Set whether tool is licensed to execute."""
        return True

    def updateParameters(self, parameters):
        """Modify the values and properties of parameters before internal
        validation is performed.  This method is called whenever a parameter
        has been changed."""
        return

    def updateMessages(self, parameters):
        """Modify the messages created by internal validation for each tool
        parameter.  This method is called after internal validation."""
        return

    def execute(self, parameters, messages):
        """The source code of the tool."""
        return

class batchRandomForest(object):
    def __init__(self):
        """Define the tool (tool name is the name of the class)."""
        self.label = "Batch Random Forest - Predict Raster"
        self.description = "Tool uses explanatory rasters to predict to raster. Specify the output workspace" \
                           "for the four resulting outputs: prediction raster, variable importance table, trained " \
                           "features and validation r2. The outputs will be named according to the name of the input feature class" \
                           "and the species selected for prediction."
        self.canRunInBackground = True

    def getParameterInfo(self):
        """Define parameter definitions"""

        param0 = arcpy.Parameter(
            displayName="Input Training Features",
            name="in_fc",
            datatype="DEFeatureClass",
            parameterType="Required",
            direction="Input")

        param1 = arcpy.Parameter(
            displayName="Variables to Predict",
            name="sp_field",
            datatype="Field",
            parameterType="Required",
            direction="Input",
            multiValue=True)

        param1.parameterDependencies = [param0.name]

        param2 = arcpy.Parameter(
            displayName="Output Workspace",
            name="out_ws",
            datatype="DEWorkspace",
            parameterType="Required",
            direction="Input")

        param3 = arcpy.Parameter(
            displayName="Output Files Prefix",
            name="prefix",
            datatype="GPString",
            parameterType="Required",
            direction="Input")

        param4 = arcpy.Parameter(
            displayName="Number of Trees",
```

```python
                name="n_tree",
                datatype="GPLong",
                parameterType="Required",
                direction="Input")

        param5 = arcpy.Parameter(
                displayName="# of Random Variables - mtry",
                name="m_try",
                datatype="GPLong",
                parameterType="Required",
                direction="Input")

        param6 = arcpy.Parameter(
                displayName="Compensate for Sparse Categories",
                name="sparse",
                datatype="GPBoolean",
                parameterType="Required",
                direction="Input")
        param6.value = True

        param7 = arcpy.Parameter(
                displayName="Percent Excluded for Validation",
                name="pct_train",
                datatype="GPLong",
                parameterType="Required",
                direction="Input")
        param7.filter.type = "ValueList"
        param7.filter.list = [10, 20, 30]

        param8 = arcpy.Parameter(
                displayName="Number of Validation Runs",
                name="n_validation",
                datatype="GPLong",
                parameterType="Required",
                direction="Input")

        # Fifth parameter
        param9 = arcpy.Parameter(
                displayName="Select Explanatory Rasters",
                name="in_rasters",
                datatype="DERasterDataset",
                parameterType="Required",
                direction="Input",
                multiValue=True)


        parameters = [param0, param1, param2, param3, param4, param5,
                    param6, param7, param8, param9]

        return parameters

    def isLicensed(self):
        """Set whether tool is licensed to execute."""
        return True

    def updateParameters(self, parameters):
        """Modify the values and properties of parameters before internal
        validation is performed.  This method is called whenever a parameter
        has been changed."""
        return

    def updateMessages(self, parameters):

        return

    def execute(self, parameters, messages):
        ws_fishnet = r"C:\Users\hashimotoy\Desktop\ws_prep\fishnet.gdb"

        in_fc = parameters[0].valueAsText
        lst_sp = parameters[1].valueAsText
        out_ws = parameters[2].valueAsText
        prefix = parameters[3].valueAsText
        n_tree = parameters[4].valueAsText
        m_try = parameters[5].valueAsText
        sparse = parameters[6].valueAsText
        pct_train = parameters[7].valueAsText
        n_validation = parameters[8].valueAsText
        in_rasters = parameters[9].valueAsText

        prediction_type = "PREDICT_RASTER"
        nm = getBaseName(in_fc)
        in_features = in_fc
        arcpy.env.mask = r"C:\Users\hashimotoy\Desktop\ws_prep\raster_attributes.gdb\\" + "eco_" + nm

        lst_rasters = []
        lst_matching = [] # Explanatory raster matching
        for raster in in_rasters.split(';'):
            lst_rasters.append(raster)
            sub_lst = [raster, raster]
            lst_matching.append(sub_lst)


        treat_variable_as_categorical = None
        explanatory_variables = None
        distance_features = None
        explanatory_rasters = lst_rasters
        features_to_predict = None
        explanatory_variable_matching = None
        explanatory_distance_matching = None
        explanatory_rasters_matching = lst_matching
        use_raster_values = True
        number_of_trees = n_tree
        minimum_leaf_size = None
        maximum_level = None
        sample_size = None
        random_sample = m_try
```

```python
            percentage_for_training = pct_train
            output_classification_table = None
            compensate_sparse_categories = True
            number_validation_runs = n_validation

            for sp in lst_sp.split(';'):
                arcpy.AddMessage("Running forest for " + sp)
                variable_predict = sp
                file_prefix = prefix + "_" + nm + "_" + sp + "_"
                output_features = os.path.join(out_ws, file_prefix + "_output_features")
                output_raster = os.path.join(out_ws, file_prefix + "_rtr")
                output_trained_features = os.path.join(out_ws, file_prefix + "_trained")
                output_importance_table = os.path.join(out_ws, file_prefix + "_varimp")
                output_validation_table = os.path.join(out_ws, file_prefix + "_validation")

                arcpy.stats.Forest(prediction_type,
                                   in_features,
                                   variable_predict,
                                   treat_variable_as_categorical,
                                   explanatory_variables,
                                   distance_features,
                                   explanatory_rasters,
                                   features_to_predict,
                                   output_features, output_raster,
                                   explanatory_variable_matching,
                                   explanatory_distance_matching,
                                   explanatory_rasters_matching,
                                   output_trained_features,
                                   output_importance_table,
                                   use_raster_values,
                                   number_of_trees,
                                   minimum_leaf_size,
                                   maximum_level,
                                   sample_size,
                                   random_sample,
                                   percentage_for_training,
                                   output_classification_table,
                                   output_validation_table,
                                   compensate_sparse_categories,
                                   number_validation_runs)

        return


class checkRasterForNullValues(object):
    def __init__(self):
        """Define the tool (tool name is the name of the class)."""
        self.label = "Check Rasters for Null Values"
        self.description = "Interrogates selected rasters for NULL values"
        self.canRunInBackground = True

    def getParameterInfo(self):
        """Define parameter definitions"""
        # First parameter
        param0 = arcpy.Parameter(
            displayName="Select Rasters",
            name="in_rasters",
            datatype="DERasterDataset",
            parameterType="Required",
            direction="Input",
            multiValue=True)

        # Second parameter
        param1 = arcpy.Parameter(
            displayName="Output Workspace",
            name="out_ws",
            datatype="DEWorkspace",
            parameterType="Required",
            direction="Input")

        parameters = [param0, param1]

        return parameters

    def isLicensed(self):
        """Set whether tool is licensed to execute."""
        return True

    def updateParameters(self, parameters):
        """Modify the values and properties of parameters before internal
        validation is performed.  This method is called whenever a parameter
        has been changed."""
        return

    def updateMessages(self, parameters):
        """Modify the messages created by internal validation for each tool
        parameter.  This method is called after internal validation."""
        return

    def execute(self, parameters, messages):
        lst_rasters = parameters[0].valueAsText
        out_ws = parameters[1].valueAsText

        for raster in lst_rasters.split(';'):
            nm = getBaseName(raster)
            arcpy.AddMessage("Reading {0}".format(raster))
            outras = IsNull(raster)
            outras.save("{0}/{1}".format(out_ws, nm))
            arcpy.AddMessage("Saving {0}".format(raster))
        return


class setRasterNullValuesTo0Workspace(object):
    def __init__(self):
        """Define the tool (tool name is the name of the class)."""
        self.label = "Set Null Raster Values to 0 - by Workspace"
```

GEOG 596B Capstone Project

Pennsylvania State University, MGIS Program

```python
        self.description = "Interrogates rasters in a workspace and checks for null values. If present, NA values are set" \
                           "to 0"
        self.canRunInBackground = True

    def getParameterInfo(self):
        """Define parameter definitions"""
        # First parameter
        param0 = arcpy.Parameter(
            displayName="Raster Workspace",
            name="in_rasters",
            datatype="DEWorkspace",
            parameterType="Required",
            direction="Input")

        # Second parameter
        param1 = arcpy.Parameter(
            displayName="Output Workspace",
            name="out_ws",
            datatype="DEWorkspace",
            parameterType="Required",
            direction="Input")

        parameters = [param0, param1]

        return parameters

    def isLicensed(self):
        """Set whether tool is licensed to execute."""
        return True

    def updateParameters(self, parameters):
        """Modify the values and properties of parameters before internal
        validation is performed.  This method is called whenever a parameter
        has been changed."""
        return

    def updateMessages(self, parameters):
        """Modify the messages created by internal validation for each tool
        parameter.  This method is called after internal validation."""
        return

    def execute(self, parameters, messages):
        in_ws = parameters[0].valueAsText
        out_ws = parameters[1].valueAsText
        arcpy.env.workspace = in_ws
        rasters = arcpy.ListRasters()

        for raster in rasters:
            nm = getBaseName(raster)
            arcpy.AddMessage("Reading {0}".format(raster))
            outras = Con(IsNull(raster), 0, raster)
            arcpy.AddMessage("Setting null for {0}".format(raster))
            outras.save("{0}/{1}".format(out_ws, nm))
            arcpy.AddMessage("Saving {0}".format(raster))
        return

class setRasterNullValuesTo0(object):
    def __init__(self):
        """Define the tool (tool name is the name of the class)."""
        self.label = "Set Raster Null Values to 0 - by Selection"
        self.description = "Sets all null raster values to 0 in new raster"
        self.canRunInBackground = True

    def getParameterInfo(self):
        """Define parameter definitions"""
        # First parameter
        param0 = arcpy.Parameter(
            displayName="Input Rasters",
            name="in_rasters",
            datatype="DERasterDataset",
            parameterType="Required",
            direction="Input",
            multiValue=True)

        param1 = arcpy.Parameter(
            displayName="Target workspace",
            name="out_ws",
            datatype="DEWorkspace",
            parameterType="Required",
            direction="Input")

        parameters = [param0, param1]

        return parameters

    def isLicensed(self):
        """Set whether tool is licensed to execute."""
        return True

    def updateParameters(self, parameters):
        """Modify the values and properties of parameters before internal
        validation is performed.  This method is called whenever a parameter
        has been changed."""
        return

    def updateMessages(self, parameters):
        """Modify the messages created by internal validation for each tool
        parameter.  This method is called after internal validation."""
        return

    def execute(self, parameters, messages):
        lst_rasters = parameters[0].valueAsText
        out_ws = parameters[1].valueAsText

        for raster in lst_rasters.split(';'):
            nm = getBaseName(raster)
            arcpy.AddMessage("Reading {0}".format(raster))
```

```python
            outras = Con(IsNull(raster), 0, raster)
            arcpy.AddMessage("Setting null for {0}".format(raster))
            outras.save("{0}/{1}".format(out_ws, nm))
            arcpy.AddMessage("Saving {0}".format(raster))

        return

class featureToRasterAttributes(object):
    def __init__(self):
        """Define the tool (tool name is the name of the class)."""
        self.label = "Feature To Raster Attributes"
        self.description = ""
        self.canRunInBackground = True

    def getParameterInfo(self):
        # Define parameter definitions

        # First parameter
        param0 = arcpy.Parameter(
            displayName="Input Features",
            name="in_features",
            datatype="DEFeatureClass",
            parameterType="Required",
            direction="Input")

        # Second parameter
        param1 = arcpy.Parameter(
            displayName="Attribute Field",
            name="field",
            datatype="Field",
            parameterType="Required",
            direction="Input")

        param1.filter.list = ['TEXT', 'LONG']
        param1.parameterDependencies = [param0.name]

        param2 = arcpy.Parameter(
            displayName="Output workspace",
            name="out_ws",
            datatype="DEWorkspace",
            parameterType="Required",
            direction="Input"
        )
        param2.defaultEnvironmentName = "workspace"

        parameters = [param0, param1, param2]
        return parameters

    def isLicensed(self):
        """Set whether tool is licensed to execute."""
        return True

    def updateParameters(self, parameters):

        return

    def updateMessages(self, parameters):
        """Modify the messages created by internal validation for each tool
        parameter.  This method is called after internal validation."""
        return

    def execute(self, parameters, messages):
        import os

        in_fc = parameters[0].valueAsText
        in_field = parameters[1].valueAsText
        out_ws = r"C:\Users\hashimotoy\Desktop\ws_prep\raster.gdb"
        attributes_ws = r"C:\Users\hashimotoy\Desktop\ws_prep\raster_attributes.gdb"
        out_raster = os.path.join(out_ws, os.path.basename(in_fc))
        # Converting to Raster
        arcpy.AddMessage("Converting to raster: " + out_raster + "...")
        raster = arcpy.FeatureToRaster_conversion(in_fc, in_field, os.path.join(out_ws, out_raster))
        # raster = r"C:\Users\hashimotoy\Desktop\ws_prep\raster.gdb\vri"
        values = unique_values(raster, in_field)
        # Extract By Attributes each unique value
        for value in values:
            if value == "":
                pass
            else:
                print("     Extracting " + value)
                where_clause = in_field + " = '" + value + "'"
                arcpy.AddMessage("     " + where_clause)
                extract = ExtractByAttributes(raster, where_clause)
                out_nm = (in_field + "_" + value).lower()
                extract.save(os.path.join(attributes_ws, out_nm))
        return

class renameDataInWorkspace(object):
    def __init__(self):
        """Define the tool (tool name is the name of the class)."""
        self.label = "Rename Data Sources in Workspace"
        self.description = "The tool will batch rename data within a workspace by one of three options: " \
                            "replace, prefix or add suffix. if "
        self.canRunInBackground = True

    def getParameterInfo(self):
        """Define parameter definitions"""
        # First parameter
        param0 = arcpy.Parameter(
            displayName="Input Workspace",
            name="in_ws",
            datatype="DEWorkspace",
            parameterType="Required",
            direction="Input")
        #param0.filter.type = "Workspace"

        param1 = arcpy.Parameter(
```

```python
                    displayName="Select String Parsing Function",
                    name="parse_function",
                    datatype="GPString",
                    parameterType="Required",
                    direction="Input")

            param1.filter.list = ["REPLACE SUBSTRING", "ADD PREFIX", "ADD SUFFIX"]

            # Second parameter
            param2 = arcpy.Parameter(
                    displayName="Input string",
                    name="in_string",
                    datatype="GPString",
                    parameterType="Required",
                    direction="Input")

            # Third parameter
            param3 = arcpy.Parameter(
                    displayName="Output string",
                    name="out_string",
                    datatype="GPString",
                    parameterType="Optional",
                    direction="Input")

            parameters = [param0, param1, param2, param3]

            return parameters

    def isLicensed(self):
        """Set whether tool is licensed to execute."""
        return True

    def updateParameters(self, parameters):
        parameters[3].enabled = False
        if parameters[1].value:
            if parameters[1].valueAsText == "REPLACE SUBSTRING":
                parameters[3].enabled = True
            else:
                parameters[3].enabled = False
        return

    def updateMessages(self, parameters):
        """Modify the messages created by internal validation for each tool
        parameter.  This method is called after internal validation."""
        return

    def execute(self, parameters, messages):
        in_ws = parameters[0].valueAsText
        parse_function = parameters[1].valueAsText
        in_string = parameters[2].valueAsText
        out_string = parameters[3].valueAsText

        env.workspace = in_ws
        fcs = arcpy.ListFeatureClasses()
        rasters = arcpy.ListRasters()

        lst = fcs + rasters

        for data in lst:
            nm = getBaseName(data)
            if in_string in nm:
                pass
            elif parse_function == "ADD PREFIX":
                out_string = in_string + nm
            elif parse_function == "ADD SUFFIX":
                out_string = nm + out_string
            elif parse_function == "REPLACE SUBSTRING":
                if in_string not in nm:
                    arcpy.AddMessage("\nThe substring " + in_string) + " is not within data source names"
                    pass
                else:
                    arcpy.Rename_management(data, nm.replace(in_string, out_string))
                    arcpy.AddMessage("\nRenamed " + nm + " to " + nm.replace(in_string, out_string))
            arcpy.Rename_management(data, out_string)
            arcpy.AddMessage("\nRenamed " + nm + " to " + out_string)
        return

class kernelDensityCalculations(object):
    def __init__(self):
        """Define the tool (tool name is the name of the class)."""
        self.label = "Calculate Kernel Density"
        self.description = "Tool to generalize vector point and line inputs (use the Focal Statistics tool for raster inputs)"
        self.canRunInBackground = True

    def getParameterInfo(self):
        """Define parameter definitions"""
        """Define parameter definitions"""
        # First parameter
        param0 = arcpy.Parameter(
                displayName="Input Feature Classes",
                name="lst_in_features",
                datatype="DEFeatureClass",
                parameterType="Required",
                direction="Input",
                multiValue=True)

        param0.filter.list = ["Point", "Multipoint", "Polyline"]

        param1 = arcpy.Parameter(
                displayName="Search radius",
                name="search_radius",
                datatype="GPLong",
                parameterType="Required",
                direction="Input")
        param1.value = 564 # pi 'r' squared ~ 1sqkm

        param2 = arcpy.Parameter(
```

```python
                displayName="Cell size",
                name="cell_size",
                datatype="GPLong",
                parameterType="Required",
                direction="Input")

        param2.value = 20 # Default

        param3 = arcpy.Parameter(
                displayName="Density unit value",
                name="area_unit_scale_factor",
                datatype="GPString",
                parameterType="Required",
                direction="Input")
        param3.filter.list = ["SQUARE_MAP_UNITS","SQUARE_KILOMETERS","HECTARES","SQUARE_METERS"]

        param3.value = "SQUARE_METERS"

        param4 = arcpy.Parameter(
                displayName="Output workspace",
                name="out_ws",
                datatype="DEWorkspace",
                parameterType="Required",
                direction="Input")

        parameters = [param0, param1, param2, param3, param4]
        # parameters = [param0, param1, param2, param3]
        return parameters

    def isLicensed(self):
        """Set whether tool is licensed to execute."""
        return True

    def updateParameters(self, parameters):
        """Modify the values and properties of parameters before internal
        validation is performed.  This method is called whenever a parameter
        has been changed."""
        return

    def updateMessages(self, parameters):
        """Modify the messages created by internal validation for each tool
        parameter.  This method is called after internal validation."""
        return

    def execute(self, parameters, messages):

        lst_inputs = parameters[0].valueAsText
        search_radius = parameters[1].valueAsText
        cell_size = parameters[2].valueAsText
        area_unit_scale_factor = parameters[3].valueAsText
        out_ws = parameters[4].valueAsText

        for in_fc in lst_inputs.split(';'):
            nm = getBaseName(in_fc)
            arcpy.AddMessage("\n\nProcessing " + nm)
            kd = KernelDensity(in_features=in_fc,
                                population_field="",
                                cell_size=cell_size,
                                search_radius=search_radius,
                                area_unit_scale_factor=area_unit_scale_factor,
                                out_cell_values="DENSITIES",
                                method="PLANAR")
            kd.save(os.path.join(out_ws, nm))

        return

class splitRasterByAttributes(object):
    def __init__(self):
        """Define the tool (tool name is the name of the class)."""
        self.label = "Split Raster by Attributes"
        self.description = "Output rasters are named according to the attribute field and value"
        self.canRunInBackground = False

    def getParameterInfo(self):
        """Define parameter definitions"""
        # First parameter
        param0 = arcpy.Parameter(
                displayName="Input Raster",
                name="in_raster",
                datatype=["DERasterDataset","GPRasterLayer"],
                parameterType="Required",
                direction="Input")

        # Second parameter
        param1 = arcpy.Parameter(
                displayName="Attribute Field",
                name="field",
                datatype="Field",
                parameterType="Required",
                direction="Input")

        param1.filter.list = ['TEXT', 'LONG']
        param1.parameterDependencies = [param0.name]

        param2 = arcpy.Parameter(
                displayName="Output workspace",
                name="out_ws",
                datatype="DEWorkspace",
                parameterType="Required",
                direction="Input"
                )
        param2.defaultEnvironmentName = "workspace"

        parameters = [param0, param1, param2]
        return parameters

    def isLicensed(self):
```

```python
        """Set whether tool is licensed to execute."""
        return True

    def updateParameters(self, parameters):
        """Modify the values and properties of parameters before internal
        validation is performed.  This method is called whenever a parameter
        has been changed."""
        return

    def updateMessages(self, parameters):
        """Modify the messages created by internal validation for each tool
        parameter.  This method is called after internal validation."""
        return

    def execute(self, parameters, messages):
        in_raster = parameters[0].valueAsText
        in_field = parameters[1].valueAsText
        out_ws = parameters[2].valueAsText

        values = unique_values(in_raster, in_field)
        # Extract By Attributes each unique value
        for value in values:
            if value == "":
                pass
            else:
                print("    Extracting " + value)
                where_clause = in_field + " = '" + value + "'"
                arcpy.AddMessage("    " + where_clause)
                extract = ExtractByAttributes(in_raster, where_clause)
                out_nm = (in_field + "_" + value).lower()
                extract.save(os.path.join(out_ws, out_nm))
        return

class focalStatsByCircle(object):
    def __init__(self):
        self.label = "Focal Statistics by Circle"
        self.description = "Tool to generalize raster inputs (use the Kernel Density tool for vector points and lines)"
        self.canRunInBackground = True

    def getParameterInfo(self):

        param0 = arcpy.Parameter(
            displayName="Input Rasters",
            name="in_rasters",
            datatype="DERasterDataset",
            parameterType="Required",
            direction="Input",
            multiValue=True)


        param1 = arcpy.Parameter(
            displayName="Neighbourhood in map units",
            name="neighbourhood",
            datatype="GPLong",
            parameterType="Required",
            direction="Input")
        param1.value = 1200 # Circular neighbourhood of 1.2km to account for center of 400m cell grid

        param2 = arcpy.Parameter(
            displayName="Statistics type",
            name="stats_type",
            datatype="GPString",
            parameterType="Required",
            direction="Input",
            multiValue=False)
        param2.filter.list = ["MEAN", "MAJORITY", "MAXIMUM", "MEDIAN",
                              "MINIMUM", "MINORITY", "RANGE", "STD",
                              "SUM", "VARIETY"]
        param2.value = "SUM"

        param3 = arcpy.Parameter(
            displayName="Output workspace",
            name="out_ws",
            datatype="DEWorkspace",
            parameterType="Required",
            direction="Input")

        parameters = [param0, param1, param2, param3]
        return parameters

    def isLicensed(self):
        """Set whether tool is licensed to execute."""
        return True

    def updateParameters(self, parameters):
        """Modify the values and properties of parameters before internal
        validation is performed.  This method is called whenever a parameter
        has been changed."""
        return

    def updateMessages(self, parameters):
        """Modify the messages created by internal validation for each tool
        parameter.  This method is called after internal validation."""
        return

    def execute(self, parameters, messages):

        rasters = parameters[0].valueAsText
        neighbourhood = parameters[1].valueAsText
        stats_type = parameters[2].valueAsText
        out_ws = parameters[3].valueAsText

        for in_raster in rasters.split(';'):
            arcpy.AddMessage(in_raster)
            nm = getBaseName(in_raster)
            arcpy.AddMessage("\nProcessing focal statistics for " + nm)
            fs = FocalStatistics(in_raster, NbrCircle(neighbourhood, "MAP"),
```

```python
                                stats_type,"DATA")
                fs.save(os.path.join(out_ws, nm))

        return

class focalStatsByRectangle(object):
    def __init__(self):
        self.label = "Focal Statistics by Rectangle"
        self.description = "Tool to generalize raster inputs (use the Kernel Density tool for vector points and lines)"
        self.canRunInBackground = True

    def getParameterInfo(self):

        param0 = arcpy.Parameter(
            displayName="Input Rasters",
            name="in_rasters",
            datatype="DERasterDataset",
            parameterType="Required",
            direction="Input",
            multiValue=True)

        param1 = arcpy.Parameter(
            displayName="Neighbourhood in map units",
            name="neighbourhood",
            datatype="GPLong",
            parameterType="Required",
            direction="Input")
        param1.value = 1000 # Rectangular neighbourhood 1km x 1km

        param2 = arcpy.Parameter(
            displayName="Statistics type",
            name="stats_type",
            datatype="GPString",
            parameterType="Required",
            direction="Input",
            multiValue=True)
        param2.filter.list = ["MEAN", "MAJORITY", "MAXIMUM", "MEDIAN",
                               "MINIMUM", "MINORITY", "RANGE", "STD",
                               "SUM", "VARIETY"]
        param2.value = "SUM"

        param3 = arcpy.Parameter(
            displayName="Output workspace",
            name="out_ws",
            datatype="DEWorkspace",
            parameterType="Required",
            direction="Input")

        parameters = [param0, param1, param2, param3]
        return parameters

    def isLicensed(self):
        """Set whether tool is licensed to execute."""
        return True

    def updateParameters(self, parameters):
        """Modify the values and properties of parameters before internal
        validation is performed.  This method is called whenever a parameter
        has been changed."""
        return

    def updateMessages(self, parameters):
        """Modify the messages created by internal validation for each tool
        parameter.  This method is called after internal validation."""
        return

    def execute(self, parameters, messages):

        rasters = parameters[0].valueAsText
        neighbourhood = parameters[1].valueAsText
        stats_type = parameters[2].valueAsText
        out_ws = parameters[3].valueAsText

        for in_raster in rasters.split(';'):
            arcpy.AddMessage(in_raster)
            nm = getBaseName(in_raster)
            arcpy.AddMessage("\nProcessing focal statistics for " + nm)
            fs = FocalStatistics(in_raster, NbrRectangle(neighbourhood, neighbourhood, "MAP"),
                                  stats_type,"DATA") #
            fs.save(os.path.join(out_ws, nm))

        return

class setRasterValuesTo1(object):
    def __init__(self):
        """Define the tool (tool name is the name of the class)."""
        self.label = "Set Selected Raster Values to 1"
        self.description = "Sets all raster values to 1 within a specified workspace. For categorical (factor) variables."
        self.canRunInBackground = False

    def getParameterInfo(self):
        """Define parameter definitions"""
        # First parameter
        param0 = arcpy.Parameter(
            displayName="Input Rasters",
            name="in_rasters",
            datatype="DERasterDataset",
            parameterType="Required",
            direction="Input",
            multiValue=False)

        param1 = arcpy.Parameter(
            displayName="Output workspace",
            name="out_ws",
            datatype="DEWorkspace",
            parameterType="Required",
```

```python
                    direction="Input")

            parameters = [param0, param1]
            return parameters

        def isLicensed(self):
            """Set whether tool is licensed to execute."""
            return True

        def updateParameters(self, parameters):
            """Modify the values and properties of parameters before internal
            validation is performed.  This method is called whenever a parameter
            has been changed."""
            return

        def updateMessages(self, parameters):
            """Modify the messages created by internal validation for each tool
            parameter.  This method is called after internal validation."""
            return

        def execute(self, parameters, messages):
            lst_rasters = parameters[0].valueAsText
            out_ws = parameters[1].valueAsText

            for in_raster in lst_rasters.split(';'):
                nm = getBaseName(in_raster)
                arcpy.AddMessage("\n" + nm)
                out_con = Con(~IsNull(in_raster), 1, 0)
                out_con.save(os.path.join(out_ws, nm))

            return


class pointToRasterGroupByAttributes(object):
    def __init__(self):
        """Define the tool (tool name is the name of the class)."""
        self.label = "Point to Raster - Group by Attributes"
        self.description = "Create raster surfaces from one or more attributes in an input point feature class." \
                            " Output rasters will be prefixed by user defined string" \
                            "and appended with the field value. The optional 'group by' parameter will output individual rasters" \
                            "based on attribute."
        self.canRunInBackground = True

    def getParameterInfo(self):
        """Define parameter definitions"""
        # First parameter
        param0 = arcpy.Parameter(
            displayName="Input Point Feature Class",
            name="in_pts",
            datatype="GPFeatureLayer",
            parameterType="Required",
            direction="Input")

        # Second parameter
        param1 = arcpy.Parameter(
            displayName="Attribute fields",
            name="fields",
            datatype="Field",
            parameterType="Required",
            direction="Input",
            multiValue=True)
        param1.parameterDependencies = [param0.name]

        # Third parameter
        param2 = arcpy.Parameter(
            displayName="Output Workspace",
            name="out_ws",
            datatype="DEWorkspace",
            parameterType="Required",
            direction="Input")
        param2.defaultEnvironmentName = "workspace"
        # param2.filter.list = ["Local Database"]

        # Fourth parameter
        param3 = arcpy.Parameter(
            displayName="Prefix for out raster name",
            name="prefix",
            datatype="GPString",
            parameterType="Required",
            direction="Input")

        # Fifth parameter
        param4 = arcpy.Parameter(
            displayName="Cell Assignment",
            name="assignment_type",
            datatype="GPString",
            parameterType="Required",
            direction="Input")
        param4.filter.type = 'ValueList'
        param4.filter.list = ["MEAN", "MAXIMUM", "MOST_FREQUENT",
                            "MINIMUM","RANGE", "STANDARD_DEVIATION",
                            "SUM", "COUNT"]
        param4.value = "MEAN"

        # Sixth parameter
        param5 = arcpy.Parameter(
            displayName="Cell Size",
            name="cell_size",
            datatype="GPLong",
            parameterType="Input",
            direction="Input")
        param5.value = 400

        # Seventh Optional parameter
        param6 = arcpy.Parameter(
            displayName="Optional - Group By attribute",
            name="groupby",
```

```python
                datatype="Field",
                parameterType="Optional",
                direction="Input",
                multiValue = False)

        param6.parameterDependencies = [param0.name]

        parameters = [param0, param1, param2, param3, param4, param5, param6]

        return parameters

    def isLicensed(self):
        """Set whether tool is licensed to execute."""
        return True

    def updateParameters(self, parameters):
        """Modify the values and properties of parameters before internal
        validation is performed.  This method is called whenever a parameter
        has been changed."""
        return

    def updateMessages(self, parameters):
        """Modify the messages created by internal validation for each tool
        parameter.  This method is called after internal validation."""
        return

    def execute(self, parameters, messages):
        # Set environment settings

        in_pts = parameters[0].valueAsText
        fields = parameters[1].valueAsText
        out_ws = parameters[2].valueAsText
        prefix = parameters[3].valueAsText
        assignment_type = parameters[4].valueAsText
        cell_size = parameters[5].valueAsText
        groupby = parameters[6].valueAsText

        env.workspace = out_ws

        arcpy.AddMessage("Running Point to Raster on: " + in_pts)

        for field in fields.split(';'):
            valField = field

            if groupby is None:
                arcpy.AddMessage("\n      Processing " + field)
                inFeatures = in_pts
                outRaster = (prefix + "_" + field).lower()
                arcpy.PointToRaster_conversion(inFeatures, valField, outRaster,
                                               assignment_type, "", cell_size)

            else:
                values = unique_values(in_pts, groupby)
                for value in values:
                    arcpy.AddMessage("\n      Processing " + field + " for " + str(value))
                    nm = getBaseName(in_pts)
                    out_nm = nm + "_" + str(value)
                    where_clause = groupby + " = '" + str(value) + "'"
                    inFeatures = arcpy.Select_analysis(in_pts, out_nm, where_clause)
                    outRaster = (prefix + "_" + field + "_" + str(value)).lower()
                    arcpy.PointToRaster_conversion(inFeatures, valField, outRaster,
                                                   assignment_type, "", cell_size)


        return

class getColumnIndex(object):
    def __init__(self):
        """Define the tool (tool name is the name of the class)."""
        self.label = "Get Column Index"
        self.description = "Using pandas module, read a csv file and get the index of selected columns"
        self.canRunInBackground = True

    def getParameterInfo(self):
        """Define parameter definitions"""
        # First parameter
        param0 = arcpy.Parameter(
            displayName="Input CSV",
            name="csv",
            datatype="DEFile",
            parameterType="Required",
            direction="Input")
        param0.filter.list = ['txt', 'csv']

        # Second parameter
        param1 = arcpy.Parameter(
            displayName="Fields",
            name="fields",
            datatype="Field",
            parameterType="Required",
            direction="Input",
            multiValue=True)
        param1.parameterDependencies = [param0.name]

        parameters = [param0, param1]

        return parameters

    def isLicensed(self):
        """Set whether tool is licensed to execute."""
        return True

    def updateParameters(self, parameters):
        """Modify the values and properties of parameters before internal
        validation is performed.  This method is called whenever a parameter
        has been changed."""
        return
```

```python
    def updateMessages(self, parameters):
        """Modify the messages created by internal validation for each tool
        parameter.  This method is called after internal validation."""
        return

    def execute(self, parameters, messages):
        import pandas as pd

        in_csv = parameters[0].valueAsText
        fields = parameters[1].valueAsText

        arcpy.AddMessage("\n\nGetting indices for selected columns...\n\n")
        csv = pd.read_csv(in_csv)

        for field in fields.split(';'):
            index = csv.columns.get_loc(field)
            arcpy.AddMessage("\n      " + field + " : " + str(index) + "\n")

        return

class extractClimateValuesForRange(object):
    def __init__(self):
        """Define the tool (tool name is the name of the class)."""
        self.label = "Extract Multi Values from Climate Data by Temporal Range"
        self.description = "The tool will extract the values of selected climate variables for" \
                           "the input points for each year in the study range 2007-2017. Basename of filename" \
                           "will be appended with _annual_clim. Note intermediate files will be output to the 'Output" \
                           "workspace'"
        self.canRunInBackground = True

    def getParameterInfo(self):
        """Define parameter definitions"""
        # First parameter
        param0 = arcpy.Parameter(
            displayName="Input Features",
            name="in_features",
            datatype="GPFeatureLayer",
            parameterType="Required",
            direction="Input")

        # Second parameter
        param1 = arcpy.Parameter(
            displayName="Output Workspace",
            name="out_ws",
            datatype="DEWorkspace",
            parameterType="Required",
            direction="Input")

        # Third parameter
        param2 = arcpy.Parameter(
            displayName="Select climate variables",
            name="clim_var",
            datatype="GPString",
            parameterType="Input",
            direction="Input",
            multiValue=True)
        param2.filter.type = 'ValueList'
        param2.filter.list = ["ahm", "bffp","cmd", "dd5", "dd18", "effp", "mat", "map",
                              "ppt", "pas_sp", "pas_wt", "shm", "tave_sp", "tave_wt", "tave04"]

        # Fourth parameter
        param3 = arcpy.Parameter(
            displayName="Workspace containing climate rasters",
            name="clim_ws",
            datatype="DEWorkspace",
            parameterType="Input",
            direction="Input")

        parameters = [param0, param1, param2, param3]

        return parameters

    def isLicensed(self):
        """Set whether tool is licensed to execute."""
        return True

    def updateParameters(self, parameters):
        """Modify the values and properties of parameters before internal
        validation is performed.  This method is called whenever a parameter
        has been changed."""
        return

    def updateMessages(self, parameters):
        """Modify the messages created by internal validation for each tool
        parameter.  This method is called after internal validation."""
        return

    def execute(self, parameters, messages):
        pts = parameters[0].valueAsText
        out_ws = parameters[1].valueAsText
        clim_var = parameters[2].valueAsText
        ws_clim = parameters[3].valueAsText

        base_nm = getBaseName(pts)

        lst_merge = []
        for i in range(2007,2018):
            out_fc = os.path.join(out_ws,base_nm + "clim_" + str(i))
            arcpy.AddMessage("\nRunning value extraction for " + out_fc)
            in_pts = arcpy.CopyFeatures_management(pts, out_fc)
            checkFieldExists(in_pts, "year_txt", "TEXT")
            lst_clim = []
            for var in clim_var.split(';'):
                clim_raster = os.path.join(ws_clim,"clim_" + var + "_" + str(i))
                arcpy.AddMessage("       " + clim_raster)
                lst_clim.append([clim_raster, var])
```

```python
            arcpy.CalculateField_management(out_fc, "year_txt", "'" + str(i) + "'")
            ExtractMultiValuesToPoints(out_fc, lst_clim)
            lst_merge.append(out_fc)
        arcpy.AddMessage("\n\nCompleted extractions by year. \n\nMerging into final dataset...")
        arcpy.Merge_management(lst_merge, os.path.join(out_ws, base_nm + "_annual_clim"))
        return

class createPredictedDensitySurfaces(object):
    def __init__(self):
        """Define the tool (tool name is the name of the class)."""
        self.label = "Create Species Response Densities"
        self.description = "This tool will create density surfaces for predicted SDMs output from 'cforest'"
        self.canRunInBackground = False

    def getParameterInfo(self):
        """Define parameter definitions"""
        # First parameter
        param0 = arcpy.Parameter(
            displayName="Input Directory of CSV response files",
            name="in_csv",
            datatype="DEWorkspace",
            parameterType="Required",
            direction="Input")

        # Second parameter
        param1 = arcpy.Parameter(
            displayName="Geodatabase of fishnets for each ecosection",
            name="ws_fishnet",
            datatype="DEWorkspace",
            parameterType="Required",
            direction="Input")

        # Third parameter
        param2 = arcpy.Parameter(
            displayName="Output Workspace",
            name="ws_out",
            datatype="DEWorkspace",
            parameterType="Required",
            direction="Input")

        # Fourth parameter
        param3 = arcpy.Parameter(
            displayName="Snap Raster",
            name="snap_raster",
            datatype="DERasterDataset",
            parameterType="Required",
            direction="Input")

        parameters = [param0, param1, param2, param3]

        return parameters

    def isLicensed(self):
        """Set whether tool is licensed to execute."""
        return True

    def updateParameters(self, parameters):
        """Modify the values and properties of parameters before internal
        validation is performed.  This method is called whenever a parameter
        has been changed."""
        return

    def updateMessages(self, parameters):
        """Modify the messages created by internal validation for each tool
        parameter.  This method is called after internal validation."""
        return

    def execute(self, parameters, messages):
        ws_response = parameters[0].valueAsText
        ws_fishnet = parameters[1].valueAsText
        ws_output = parameters[2].valueAsText
        snap_raster = parameters[3].valueAsText

        # Loop through the CSV files of predicted densities
        # For each CSV file, get the basename, join it to the appropriate ecosection fishnet
        # Output a raster
        env.workspace = ws_response
        arcpy.env.snapRaster = snap_raster
        files = arcpy.ListFiles("*.csv")
        for f in files:
            f_nm = getBaseName(f)
            arcpy.AddMessage(f_nm)
            eco = f_nm[7:10]
            sp = f_nm[11:len(f_nm)]
            arcpy.AddMessage("Preparing " + sp.upper() + " for " + eco + "...")
            fishnet_lyr = arcpy.MakeFeatureLayer_management(os.path.join(ws_fishnet,eco), "fishnet_lyr")
            joined = arcpy.AddJoin_management(fishnet_lyr, "id_fishnet", f, "id_fishnet")
            arcpy.FeatureToRaster_conversion(joined, sp, os.path.join(ws_output, eco + "_" + sp), cell_size=400)
        arcpy.AddMessage("\n\nCompleted rasterized densities. Check results in " + ws_output)
        return
```

APPENDIX 4 – R Scripts – Project Repository

Latest updates to the project will be uploaded to https://github.com/Yuhash/iws. The repository contains the base survey data inputs as well as an interactive Jupyter notebook running the R-kernel that documents the major script processes in the workflow. For optimal viewing (without interactive code functionality) in Notebook Viewer go to https://nbviewer.jupyter.org/github/Yuhash/iws/blob/master/IWS_Notebook.ipynb.

Workflow Model

```
# For data pre-processing steps please refer to the Github repository and the Jupyter Notebook

# https://journal.r-project.org/archive/2009/RJ-2009-013/RJ-2009-013.pdf
#========================================================================
# cforest model script loops through ecosections and species/species group
# mclapply is a parallelizing function available only on MAC OS. On Windows change 'mclapply' to 'lapply'

useMaxCoreCount <- 2
nTree <- 5000
mTry <- 5
seed <- 42
PREDICT <- TRUE

gitHubRoot <- "/Volumes/Black/Yuri/iws/inputs/by_location/"

outputRoot <- "/Users/hashimotoy/Desktop/iws-results/"

fgcRequire <- function(pkg) {
  if(pkg %in% rownames(installed.packages()) == FALSE) {
    install.packages(pkg)
  }
  #  library(pkg)
}

fgcRequire("ggplot2")
fgcRequire("icesTAF")
fgcRequire("party")
fgcRequire("Hmisc")
fgcRequire("lattice")
fgcRequire("beepr")

#install.packages("parallel") # is base, should not be updated (MacOS / R 3.6.0  / RStudio Version
1.2.1335 )
#install.packages("tools") # is base, should not be updated (MacOS / R 3.6.0  / RStudio Version 1.2.1335
)

library(parallel)
library(lattice)
library(Hmisc)
library(party)
library(tools)
library(icesTAF)
library(beepr)

wd <- gitHubRoot
outwd <- paste0(outputRoot,paste0("results-seed-",seed,"-year-studyarea/"))

set.seed(seed)
mkdir(outwd)
setwd(wd)
getwd()

locList <- c("BAU", "NAU", "NEL", "CHP", "BUB", "CAB", "WCU", "QUL")

spList                                                                                        <-
c("amwi","buff","bago","bwte","cago","gwte","mall","nsho","rndu","scau","dabblers","divers","cavity","
sp_div","sp_tot")

mclapply(spList, function(sp) {
  lapply(locList, function(loc) {
    setwd(wd)
    mydata <- read.csv(paste0(wd,"id1_years_",loc,".csv"))
```

```
    mydata[ 2:ncol(mydata) ] <- lapply(mydata[2:ncol(mydata)], as.numeric)
    #sum(is.na(mydata))

    fishnet <- read.csv(paste0(gitHubRoot,"fishnet_",loc,".csv"))
    fishnet[ 2:ncol(fishnet) ] <- lapply(fishnet[2:ncol(fishnet)], as.numeric)
    setwd(outwd) # DOING OUTPUT NOW

    rdsFilename <- paste0("cf_", sp, "_", loc, ".rds")
    vi_fn <- paste0("VI-",sp,"_",loc,".csv")

    print(paste("START Processing cforest for:", sp))

    cf <- if (
      !file.exists(paste0(outwd, rdsFilename))
    ) {
      #colnames(mydata)[colnames(mydata)==sp] <- "sp" RENAME HACK

      cf <- cforest(
        eval(parse(text = paste0(sp,
                          "~ aspect + slope + alr + pa + cec_urban + cec_shrubland + ",
                          "  cec_mixed_forest + cec_grassland + cec_broadleaf + cec_needleleaf +
",
                          "  fwa_1 + fwa_2 + fwa_3 + fwa_4 + fwa_10ha_more + str_s + str_m + ",
                          "  fwa_r + fwa_w  + dra_u + ",
                          # bec_zn +  # removed becase:  model used for CAB on the eco section
basis
                          #eco + # removed because we have bined by eco
                          "norm_dd5 + norm_cmd + pas_wt + tave04 +",
                          "shorecx_10plus_100000 + shorecx_10less_100000"
        )))
        ,
        data = mydata,
        controls =  cforest_unbiased(ntree = nTree, mtry = mTry)

      )
      print(paste("cforest done:", sp))
      saveRDS(cf, paste0(outwd, rdsFilename))
      cf
    } else {
      readRDS(paste0(outwd, rdsFilename))
    }


    if (!file.exists(paste0(outwd, vi_fn))
    ) {
      vi <- varimp(cf, conditional = FALSE, OOB = TRUE)
      # write.csv(vi, paste(fn, ".csv", sep = ""), row.names = FALSE)
      df_vi <- as.data.frame(vi)
      write.csv(df_vi,vi_fn);
    }

    #### Not incorporated in batch: deriving OOB, Response, Probability
    if (PREDICT) {
      ofResp <- paste0(outwd,sp,"-response_",loc,".csv")
      cf_response <- predict(cf, newdata = fishnet, OOB = TRUE, type = 'response')
      df_result <- data.frame(fishnet$id_fishnet, cf_response)
      write.csv(df_result, ofResp)
    }


    print(paste("COMPLETED Anaylsis of:", sp))
  })
}
, mc.cores = min(useMaxCoreCount,detectCores())
)
#======================================================================================
# Ranking and Plots for varimp
# Written if RDS has to be loaded

setwd(wd)

rdss <- list.files(wd, pattern ="*rds$")
lapply(rdss, function(f){
  cf <- readRDS(f)
```

```r
    fn <- substr(basename(f), 4, nchar(basename(f)) - 4)
    eco <- substr(fn, nchar(fn) - 2, nchar(fn))
    sp <- sub(paste0("_", eco), "", fn)
    vi_fn <- paste0(wd, "vi_", fn, ".csv")
    if (!file.exists(vi_fn)) {
      vi <- varimp(cf, conditional = FALSE, OOB = TRUE)
      df_vi <- as.data.frame(vi)
      v <- as.vector(df_vi$vi)

      vi <- varimp(cf, conditional = FALSE, OOB = TRUE)
      df_vi <- as.data.frame(vi)
      df_vi$X <- names(vi)
      df_vi$eco <- eco
      df_vi$sp <- sp

      # Add metadata fields
      df_vi <- as.data.frame(vi)
      df_vi$X <- names(vi)
      df_vi$eco <- eco
      df_vi$sp <- sp
      write.csv(df_vi, vi_fn, row.names = FALSE)

      # Rank
      df_vi <- df_vi %>%
        mutate(rank = dense_rank(desc(df_vi$vi)))
      colnames(df_vi)[colnames(df_vi) == "rank"] <-paste(eco, sp, sep = "_")
      rank_vi <- subset(df_vi, select = -c(vi, eco, sp))
      write.csv(df_vi, vi_fn, row.names = FALSE)
    }
})
# ========================================================================
## Create plots

library(tools)
setwd(wd)
varimpFiles <- list.files(wd, pattern = "^vi.*csv$")
lapply(varimpFiles, function(f) {
  fn <- substr(basename(f),4,nchar(basename(f))-4)
  print(fn)
  eco <- substr(fn,nchar(fn)-2,nchar(fn))
  sp <- sub(paste0("_",eco),"", fn)

  print(paste("Starting plot", f))
  fn <- file_path_sans_ext(basename(f))
  title <- paste(toupper(sp),eco)
  print(title)
  df_vi <- read.csv(f)
  v <- as.vector(df_vi$vi)
  names(v) <- as.vector(df_vi$X)
  print(names(v))
  dotchart(v[order(v)], cex = 0.9, pch = 19,
          color = "darkblue",
          xlab = "varImp", main = title)
  dev.copy(png, paste(wd, paste("varimp_", title, ".png", sep = ""), sep = "/"))
  dev.off()
  print(paste("plot done:", fn))
})

print("ALL DONE")
```

R Snippets - Data Manipulation

```r
# Create a data frame from a list of csv files converted to a list of data frames merged by common column
values
lst <- list.files(wd, pattern ="^rank.*csv$" )
merged <- do.call("cbind", lapply(ranked, read.csv))

#========================================================================
# Flat correlation matrix with P and r values

library(tidyr)
library(tibble)
library(Hmisc)
```

```r
flat_cor_mat <- function(cor_r, cor_p){
  #This function provides a simple formatting of a correlation matrix
  #into a table with 4 columns containing :
  # Column 1 : row names (variable 1 for the correlation test)
  # Column 2 : column names (variable 2 for the correlation test)
  # Column 3 : the correlation coefficients
  # Column 4 : the p-values of the correlations
  library(tidyr)
  library(tibble)
  cor_r <- rownames_to_column(as.data.frame(cor_r), var = "row")
  cor_r <- gather(cor_r, column, cor, -1)
  cor_p <- rownames_to_column(as.data.frame(cor_p), var = "row")
  cor_p <- gather(cor_p, column, p, -1)
  cor_p_matrix <- left_join(cor_r, cor_p, by = c("row", "column"))
  cor_p_matrix
}

library(corrplot)
cor <- rcorr(as.matrix(mtcars[, 1:7]))

my_cor_matrix <- flat_cor_mat(cor$r, cor$P)
head(my_cor_matrix)


corr <- rcorr(as.matrix(df))
corrplot(corr$r, method='number', number.cex= 12/ncol(df_sub))#To alter font size

# =======================================================================

# Check for NA values

df %>%
  select_if(function(x) any(is.na(x))) %>%
  summarise_each(funs(sum(is.na(.)))) -> extra_NA

# =======================================================================
# Set NA values to 0
df[is.na(df)] <- 0

# =======================================================================
# The function complete.cases() returns a logical vector indicating which cases are complete.
# list rows of data that have missing values

df[!complete.cases(df),]

# =======================================================================
# Check which version of R is running
Sys.getenv("R_ARCH")
# "/i386" or "/x64"

# =======================================================================
# UPDATE PACKAGES
update.packages(checkBuilt=TRUE, ask=FALSE)

# =======================================================================
mypath <- path

multmerge = function(mypath){
  filenames=list.files(path=mypath, full.names=TRUE)
  datalist = lapply(filenames, function(x){read.csv(file=x,header=T)})
  Reduce(function(x,y) {merge(x,y)}, datalist)}

mymergeddata = multmerge(path)

# =======================================================================
# UPDATE R Version

# installing/loading the package:
if(!require(installr)) {
  install.packages("installr"); require(installr)} #load / install+load installr

library(installr)
# using the package:
updateR()
```

```
# ========================================================================
# # Rename a column in R
colnames(data)[colnames(data)=="old_name"] <- "new_name

sp_start <- grep("dabblers", colnames(df))
sp_end <- grep("tot_sp", colnames(df))

print(sp_start)
print(sp_end)

sp <- colnames(df)[sp_start:sp_end]

var_start <- sp_end + 1
var_end <- ncol(df)
print(var_start)
print(var_end)

int_eco <- grep("eco", colnames(df))
int_trans <- grep("trans_id", colnames(df))
int_bec_zn <- grep("bec_zone", colnames(df))
int_bec_sz <- grep("bec_sz", colnames(df))

for (i in 1:length(sp)){
  nm <- sp[i]
  print(nm)
  int = grep(nm, colnames(df))
  data <- df[c(int,int_eco,int_trans,int_bec_zn, int_bec_sz,var_start:var_end)]
  colnames(data)[1]<- "sp"
  write.csv(data, paste(outwd, paste("rf_",scale, "_",nm,".csv", sep = ""), sep = "/"), row.names =
FALSE)
}
```

APPENDIX 5 – Variable Importance Plots

Babine Upland
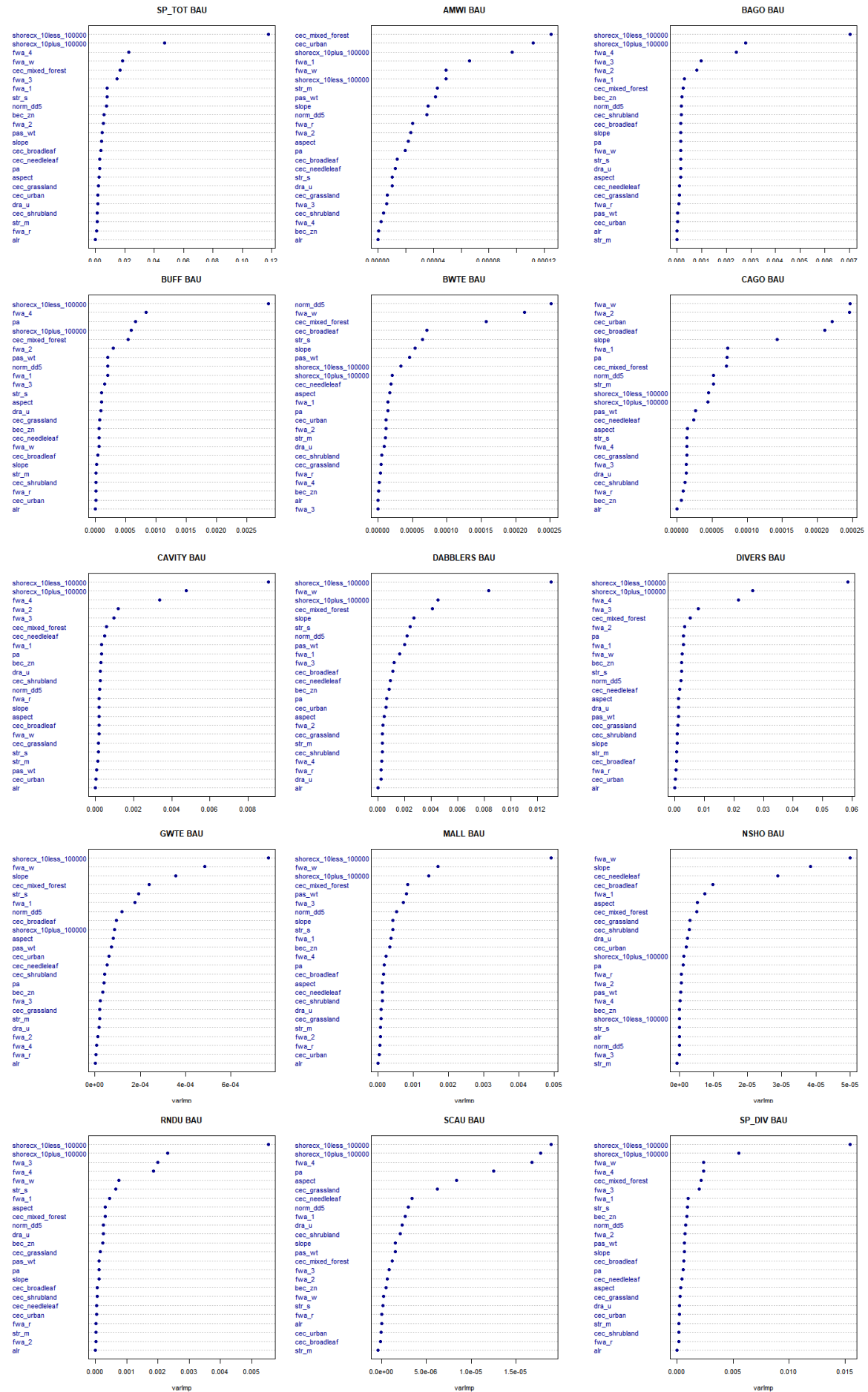
Bulkley Basin
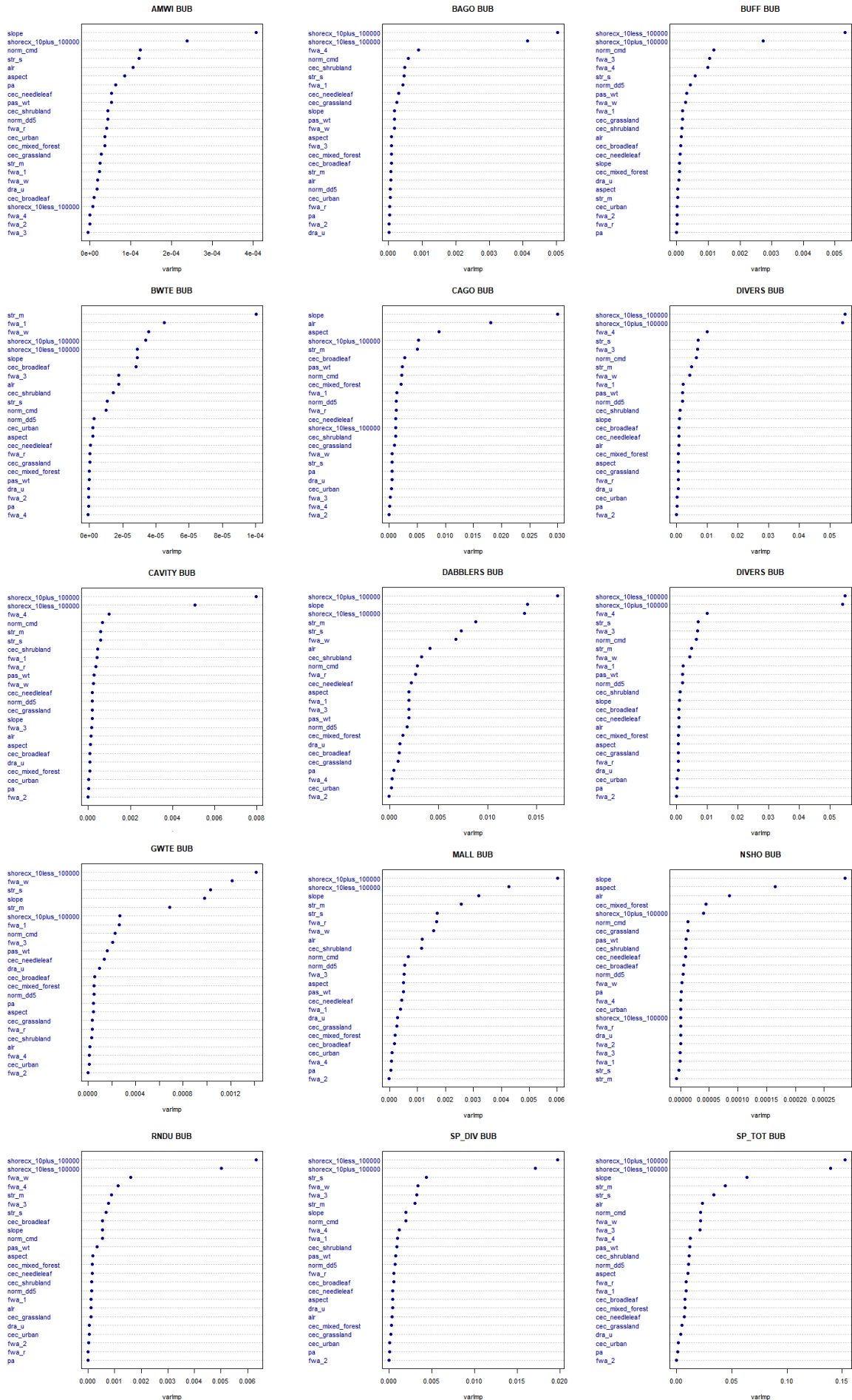
Cariboo Basin

Chilcotin Plateau

Chilcotin Plateau

Nazko Upland
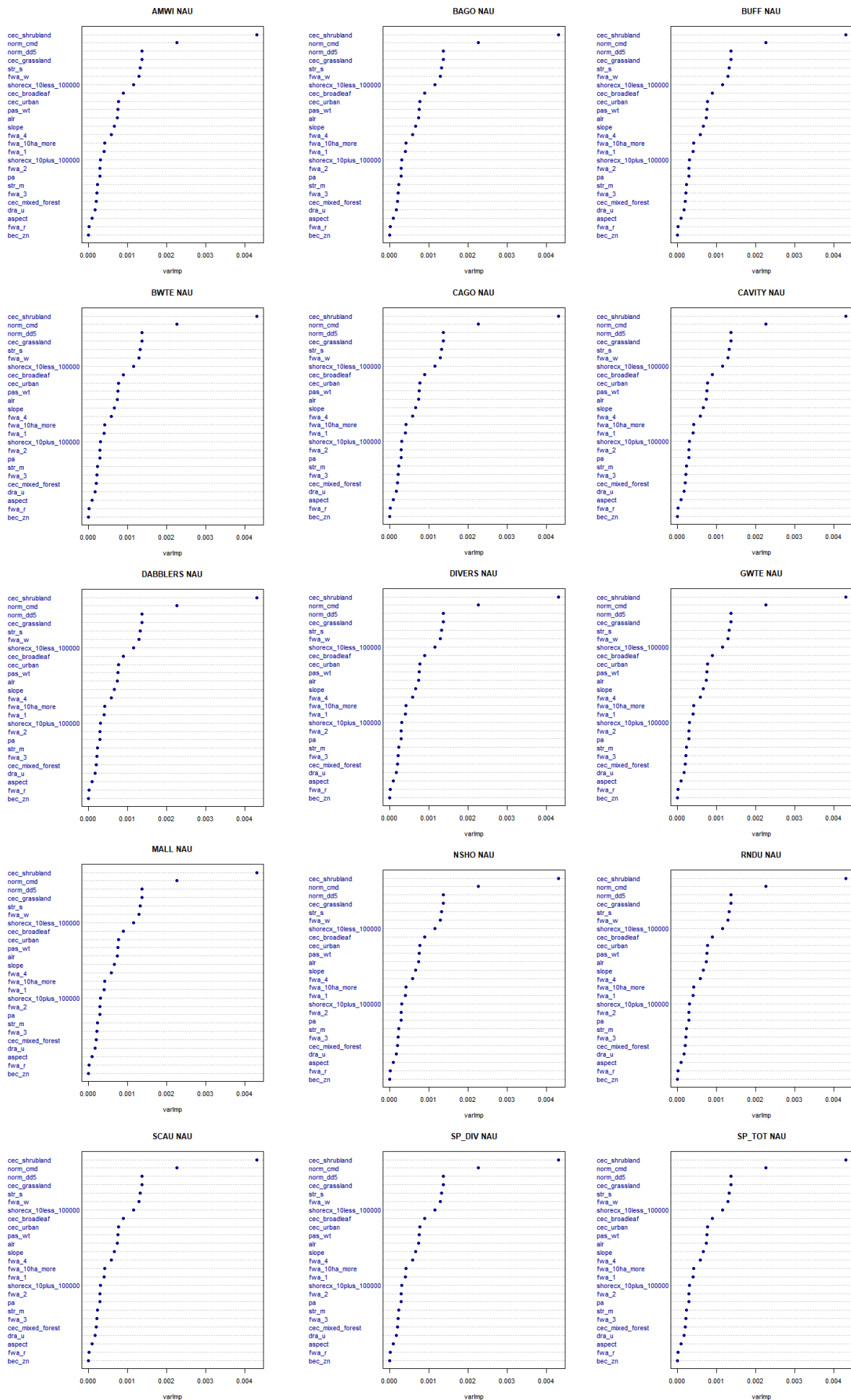
Nechako Lowland

Nechako Lowland

Quesnel Lowland

Western Chilcotin Upland

Variable Importance Summary Table

Table A5.1. Rank order values for each predictor variable by species and ecosection

Eco	Predictors	AMWI	BAGO	BUFF	BWTE	CAGO	GWTE	MALL	NSHO	RNDU	SCAU	cavity	dabblers	divers	sp_div	sp_tot	Avg
BAU	alr	24	23	24	23	23	24	24	21	24	21	24	24	24	24	24	23
BAU	aspect	13	17	12	11	15	10	15	6	8	5	16	16	14	17	17	13
BAU	bec_zn	23	8	15	22	22	16	11	18	12	17	10	13	10	9	10	14
BAU	broadleaf	15	11	18	4	4	8	14	4	17	23	17	11	21	14	14	13
BAU	grassland	19	19	14	19	17	18	19	8	13	6	19	18	17	18	18	16
BAU	mixed_forest	1	7	5	3	8	4	4	7	9	14	6	4	5	5	5	6
BAU	needleleaf	16	18	16	10	14	13	16	3	19	7	7	12	13	16	15	13
BAU	shrubland	21	10	21	18	20	14	17	9	18	11	12	20	18	22	21	17
BAU	urban	2	22	23	14	3	12	23	11	20	22	23	15	23	20	19	17
BAU	dra_u	18	16	13	17	19	20	18	10	11	10	11	23	15	19	20	16
BAU	fwa_1	4	6	9	12	6	6	10	5	7	9	8	9	8	7	7	8
BAU	fwa_2	12	5	6	15	2	21	21	15	23	16	4	17	6	11	11	12
BAU	fwa_3	20	4	10	24	18	17	6	23	3	15	5	10	4	6	6	11
BAU	fwa_4	22	3	2	21	17	22	12	17	4	3	3	21	3	4	3	10
BAU	fwa_r	11	20	22	20	21	23	22	14	21	20	14	22	22	23	23	20
BAU	fwa_w	5	14	17	2	1	2	2	1	5	18	18	2	9	3	4	7
BAU	norm_dd5	10	9	8	1	9	7	7	22	10	8	13	7	12	10	9	9
BAU	pa	14	13	3	13	7	15	13	13	15	4	9	14	7	15	16	11
BAU	pas_wt	8	21	7	7	13	11	5	16	14	13	22	8	16	12	12	12
BAU	shorecx_10less	6	1	1	8	11	1	1	19	1	1	1	1	1	1	1	4
BAU	shorecx_10plus	3	2	4	9	12	9	3	12	2	2	2	3	2	2	2	5
BAU	slope	9	12	19	6	5	3	8	2	16	12	15	5	19	13	13	10
BAU	str_m	7	24	20	16	10	19	20	24	22	24	21	19	20	21	22	19
BAU	str_s	17	15	11	5	16	5	9	20	6	19	20	6	11	8	8	12
BAU	norm_cmd	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	###
BAU	fwa_10ha_more	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	###
BAU	tave04	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	###
Eco	Predictors	AMWI	BAGO	BUFF	BWTE	CAGO	GWTE	MALL	NSHO	RNDU	SCAU	cavity	dabblers	divers	sp_div	sp_tot	Avg
BUB	alr	5	18	13	9	2	21	8	3	18	14	17	7	16	19	6	12
BUB	aspect	6	13	19	15	3	17	13	2	12	16	18	12	18	17	14	13
BUB	bec_zn	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	###
BUB	broadleaf	20	16	14	7	6	13	20	11	8	15	19	19	14	15	17	14
BUB	grassland	15	9	11	18	16	18	18	7	19	20	14	20	19	21	20	16
BUB	mixed_forest	14	15	17	19	9	14	19	4	13	2	21	17	17	20	18	15
BUB	needleleaf	8	8	15	16	13	11	15	10	14	7	12	11	15	16	19	13
BUB	shrubland	10	5	12	10	15	20	9	9	15	12	7	8	12	11	12	11
BUB	urban	13	20	21	14	21	23	21	16	21	22	22	23	22	22	22	20
BUB	dra_u	19	24	18	21	20	12	17	19	20	17	20	18	21	18	21	19
BUB	fwa_1	17	7	10	2	10	7	16	22	17	13	8	13	9	10	16	12

Eco	Predictors	AMWI	BAGO	BUFF	BWTE	CAGO	GWTE	MALL	NSHO	RNDU	SCAU	cavity	dabblers	divers	sp_div	sp_tot	Avg
BUB	fwa_2	23	23	22	22	24	24	24	20	22	19	24	24	24	24	24	23
BUB	fwa_3	24	14	4	8	22	9	12	21	6	4	16	14	5	5	9	12
BUB	fwa_4	22	3	5	24	23	22	22	15	4	10	3	22	3	9	10	13
BUB	fwa_r	12	21	23	17	12	19	6	18	23	21	9	10	20	14	15	16
BUB	fwa_w	18	12	9	3	17	2	7	13	3	11	11	6	8	4	8	9
BUB	norm_dd5	11	19	7	13	11	15	11	12	16	8	13	16	11	13	13	13
BUB	pa	7	22	24	23	19	16	23	14	24	23	23	21	23	23	23	21
BUB	pas_wt	9	11	8	20	7	10	14	8	11	18	10	15	10	12	11	12
BUB	shorecx_10less	21	2	1	5	14	1	2	17	2	1	2	3	1	1	2	5
BUB	shorecx_10plus	2	1	2	4	4	6	1	5	1	9	1	1	2	2	1	3
BUB	slope	1	10	16	6	1	4	3	1	9	6	15	2	13	7	3	6
BUB	str_m	16	17	20	1	5	5	4	24	5	24	5	4	7	6	4	10
BUB	str_s	4	6	6	11	18	3	5	23	7	3	6	5	4	3	5	7
BUB	norm_cmd	3	4	3	12	8	8	10	6	10	5	4	9	6	8	7	7
BUB	fwa_10ha_more	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	###
BUB	tave04	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	###
Eco	Predictors	AMWI	BAGO	BUFF	BWTE	CAGO	GWTE	MALL	NSHO	RNDU	SCAU	cavity	dabblers	divers	sp_div	sp_tot	Avg
CAB	alr	11	7	17	10	9	21	18	4	13	9	17	14	11	15	13	13
CAB	aspect	23	15	19	21	22	20	22	25	21	13	18	21	16	22	21	20
CAB	bec_zn	25	25	25	25	25	24	25	23	25	24	25	25	25	25	25	25
CAB	broadleaf	8	23	23	17	6	16	15	7	22	23	22	16	23	21	18	17
CAB	grassland	3	18	14	12	11	13	16	10	14	12	20	10	15	14	14	13
CAB	mixed_forest	20	21	20	18	23	15	23	17	20	21	19	22	22	23	23	20
CAB	needleleaf	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	###
CAB	shrubland	1	19	15	8	10	4	6	1	19	16	23	3	20	7	7	11
CAB	urban	10	6	22	15	8	25	12	8	23	7	13	8	17	16	11	13
CAB	dra_u	22	3	18	13	17	19	20	21	17	19	5	23	19	20	22	17
CAB	fwa_1	15	14	6	7	19	5	7	16	15	18	11	13	14	13	16	13
CAB	fwa_2	18	16	7	20	20	18	14	20	9	14	8	18	7	12	15	14
CAB	fwa_3	21	13	3	9	15	3	5	18	4	8	6	9	4	4	5	8
CAB	fwa_4	13	17	5	6	13	6	9	14	6	3	12	5	5	5	4	8
CAB	fwa_r	24	24	24	24	24	23	24	22	24	25	24	24	24	24	24	24
CAB	fwa_w	6	9	10	3	3	8	3	9	7	15	10	4	13	6	6	7
CAB	norm_dd5	4	8	9	4	4	14	10	5	5	11	16	12	8	8	8	8
CAB	pa	17	12	16	16	21	12	19	19	8	10	7	20	9	17	17	15
CAB	pas_wt	9	20	13	14	16	9	17	12	11	20	9	15	18	19	20	15
CAB	shorecx_10less	7	4	1	1	5	1	1	11	1	1	2	1	1	1	1	3
CAB	shorecx_10plus	16	2	4	22	2	10	4	6	3	2	3	6	3	3	3	6
CAB	slope	12	10	8	11	12	7	8	13	10	17	15	11	12	10	12	11
CAB	str_m	19	22	21	23	18	17	13	24	16	22	21	19	21	18	19	20
CAB	str_s	5	5	2	2	1	2	2	3	2	4	4	2	2	2	2	3
CAB	norm_cmd	2	11	12	5	7	11	11	2	12	6	14	7	10	9	9	9
CAB	fwa_10ha_more	14	1	11	19	14	22	21	15	18	5	1	17	6	11	10	12
CAB	tave04	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	###
Eco	Predictors	AMWI	BAGO	BUFF	BWTE	CAGO	GWTE	MALL	NSHO	RNDU	SCAU	cavity	dabblers	divers	sp_div	sp_tot	Avg
CHP	alr	8	25	13	3	3	16	7	9	2	7	24	7	6	7	6	10

CHP	aspect	24	23	22	23	20	22	24	21	23	4	25	23	15	23	23	21
CHP	bec_zn	20	20	16	20	22	18	22	17	15	20	22	20	18	21	22	20
CHP	broadleaf	22	18	18	17	9	21	15	16	19	23	19	18	20	17	18	18
CHP	grassland	15	10	17	5	21	6	18	8	16	11	9	12	19	15	16	13
CHP	mixed_forest	18	6	19	18	19	20	11	15	18	21	7	17	21	20	19	17
CHP	needleleaf	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	###
CHP	shrubland	14	19	20	16	8	17	20	19	20	15	23	19	22	22	21	18
CHP	urban	23	17	24	24	25	24	25	24	24	25	13	24	25	25	25	23
CHP	dra_u	12	22	23	12	15	15	13	11	22	16	21	14	23	19	20	17
CHP	fwa_1	16	8	6	7	23	7	10	14	8	9	12	9	9	8	11	10
CHP	fwa_2	6	4	7	15	17	4	9	13	9	22	4	8	10	9	12	10
CHP	fwa_3	10	2	2	14	11	3	6	3	5	10	1	3	2	3	3	5
CHP	fwa_4	3	7	8	11	6	19	12	23	6	8	5	13	8	10	8	10
CHP	fwa_r	25	24	25	25	18	25	23	25	25	24	17	25	24	24	24	24
CHP	fwa_w	1	16	3	1	1	2	1	1	4	5	20	1	5	2	2	4
CHP	norm_dd5	13	14	15	10	7	14	17	7	12	19	18	15	16	16	15	14
CHP	pa	19	12	14	22	24	13	21	22	21	2	16	22	13	18	17	17
CHP	pas_wt	7	15	9	4	14	12	14	4	10	13	11	11	12	11	13	11
CHP	shorecx_10less	2	1	1	6	5	1	2	2	1	14	2	2	1	1	1	3
CHP	shorecx_10plus	9	3	4	2	2	10	3	10	3	3	3	5	3	4	4	5
CHP	slope	17	9	10	13	12	9	8	12	11	6	15	10	11	12	10	11
CHP	str_m	5	21	21	19	4	8	4	18	17	17	10	6	17	6	7	12
CHP	str_s	4	5	5	8	10	5	5	5	7	18	6	4	7	5	5	7
CHP	norm_cmd	11	13	12	9	16	11	19	6	14	12	14	16	14	14	14	13
CHP	fwa_10ha_more	21	11	11	21	13	23	16	20	13	1	8	21	4	13	9	14
CHP	tave04	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	###

Eco	Predictors	AMWI	BAGO	BUFF	BWTE	CAGO	GWTE	MALL	NSHO	RNDU	SCAU	cavity	dabblers	divers	sp_div	sp_tot	Avg
NAU	alr	26	26	26	25	3	26	26	24	26	26	26	26	26	26	26	24
NAU	aspect	24	17	19	13	9	18	19	17	17	12	20	19	19	17	17	17
NAU	bec_zn	13	18	12	22	11	11	11	16	8	5	17	15	9	14	14	13
NAU	broadleaf	21	11	25	3	12	10	16	2	23	16	21	11	23	20	18	15
NAU	grassland	18	6	24	11	23	21	24	15	14	11	9	22	17	22	20	17
NAU	mixed_forest	15	16	22	16	19	15	20	20	16	19	22	21	24	25	22	19
NAU	needleleaf	2	5	11	9	14	7	13	7	18	4	7	8	13	13	12	10
NAU	shrubland	14	21	23	14	15	22	21	12	24	17	25	20	25	24	24	20
NAU	urban	25	24	18	26	20	24	25	25	25	21	24	25	22	23	25	23
NAU	dra_u	19	10	20	15	22	20	23	19	11	13	14	24	18	19	21	18
NAU	fwa_1	11	14	4	18	10	5	10	11	21	20	11	9	12	8	11	12
NAU	fwa_2	10	13	3	5	24	2	8	3	9	24	10	6	7	6	6	9
NAU	fwa_3	17	4	7	17	25	16	12	10	3	23	4	16	4	11	10	12
NAU	fwa_4	8	3	5	21	26	25	17	21	4	1	3	17	3	12	9	12
NAU	fwa_r	6	25	13	2	7	1	4	1	15	25	13	1	15	7	3	9
NAU	fwa_w	1	9	6	1	2	3	3	4	5	2	19	2	6	2	2	4
NAU	norm_dd5	7	20	14	6	17	13	15	8	13	10	18	12	14	16	16	13
NAU	pa	22	15	21	24	21	23	22	22	22	15	15	23	21	21	23	21
NAU	pas_wt	16	19	16	12	8	19	18	9	19	14	23	18	20	18	19	17

Eco	Predictors	AMWI	BAGO	BUFF	BWTE	CAGO	GWTE	MALL	NSHO	RNDU	SCAU	cavity	dabblers	divers	sp_div	sp_tot	Avg
NAU	shorecx_10less	4	1	1	8	13	8	2	18	1	9	2	3	1	1	1	5
NAU	shorecx_10plus	23	2	9	20	5	17	9	23	2	7	1	14	2	3	4	9
NAU	slope	3	8	15	4	6	4	6	6	6	3	16	4	10	10	8	7
NAU	str_m	12	23	17	19	1	14	1	14	20	22	8	5	16	4	7	12
NAU	str_s	20	7	8	23	18	12	7	26	10	18	6	10	8	9	13	13
NAU	norm_cmd	9	12	2	10	4	6	5	5	7	6	5	7	5	5	5	6
NAU	fwa_10ha_more	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	###
NAU	tave04	5	22	10	7	16	9	14	13	12	8	12	13	11	15	15	12
Eco	Predictors	AMWI	BAGO	BUFF	BWTE	CAGO	GWTE	MALL	NSHO	RNDU	SCAU	cavity	dabblers	divers	sp_div	sp_tot	Avg
NEL	alr	12	21	12	9	14	10	10	6	12	12	22	8	15	17	15	13
NEL	aspect	7	5	19	7	4	17	14	5	19	22	5	11	12	15	7	11
NEL	bec_zn	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	###
NEL	broadleaf	5	14	10	2	8	6	9	2	9	7	17	5	16	8	6	8
NEL	grassland	15	18	17	12	12	15	18	11	21	16	20	17	20	20	20	17
NEL	mixed_forest	17	6	21	16	16	19	20	13	14	5	9	18	17	21	21	16
NEL	needleleaf	2	13	16	4	1	7	4	3	11	8	10	4	10	7	5	7
NEL	shrubland	10	16	11	5	13	8	6	7	20	13	14	6	19	14	10	11
NEL	urban	4	12	18	19	15	12	8	10	18	9	11	10	18	13	14	13
NEL	dra_u	6	9	14	11	7	5	12	8	16	17	15	12	21	19	17	13
NEL	fwa_1	14	8	9	15	17	9	16	9	8	14	8	16	9	9	12	12
NEL	fwa_2	18	20	7	22	18	14	19	16	10	23	18	21	6	11	18	16
NEL	fwa_3	20	3	4	23	23	22	21	21	15	11	3	22	5	16	19	15
NEL	fwa_4	21	4	3	21	21	20	17	22	3	10	4	19	3	6	8	12
NEL	fwa_r	16	22	23	20	19	21	22	18	22	19	21	20	23	22	22	21
NEL	fwa_w	1	15	13	1	3	2	2	1	4	2	19	1	7	3	2	5
NEL	norm_dd5	11	11	5	13	10	16	13	15	5	4	13	14	8	12	11	11
NEL	pa	19	23	22	17	20	23	23	20	17	18	23	23	22	23	23	21
NEL	pas_wt	13	7	6	14	5	11	15	12	7	6	16	15	13	18	16	12
NEL	shorecx_10less	22	1	1	10	11	1	1	19	2	15	1	3	1	1	1	6
NEL	shorecx_10plus	9	2	2	6	6	18	5	14	1	1	2	7	2	2	3	5
NEL	slope	3	10	15	3	2	3	3	4	13	3	12	2	11	5	4	6
NEL	str_m	8	19	20	8	9	13	7	17	23	21	6	9	14	10	13	13
NEL	str_s	23	17	8	18	22	4	11	23	6	20	7	13	4	4	9	13
NEL	norm_cmd	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	###
NEL	fwa_10ha_more	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	###
NEL	tave04	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	###
Eco	Predictors	AMWI	BAGO	BUFF	BWTE	CAGO	GWTE	MALL	NSHO	RNDU	SCAU	cavity	dabblers	divers	sp_div	sp_tot	Avg
QUL	alr	23	21	17	7	6	11	18	23	14	17	20	11	19	12	12	15
QUL	aspect	21	15	23	4	25	15	22	4	23	7	15	15	20	14	25	17
QUL	bec_zn	16	24	25	17	22	18	19	15	20	14	24	19	23	25	23	20
QUL	broadleaf	25	17	20	19	13	10	11	11	24	25	17	12	24	23	19	18
QUL	grassland	9	22	24	8	14	24	24	9	10	19	22	21	25	24	18	18
QUL	mixed_forest	22	10	22	21	17	19	21	10	18	10	18	18	22	17	20	18
QUL	needleleaf	4	18	13	10	5	5	5	6	19	6	11	7	14	9	8	9
QUL	shrubland	5	16	7	9	9	16	25	3	22	24	19	25	15	20	24	16
QUL	urban	8	13	21	20	11	13	7	7	11	12	12	8	12	8	10	12

Eco	Predictors	AMWI	BAGO	BUFF	BWTE	CAGO	GWTE	MALL	NSHO	RNDU	SCAU	cavity	dabblers	divers	sp_div	sp_tot	Avg
QUL	dra_u	3	7	6	18	15	12	4	22	8	20	9	4	10	10	7	10
QUL	fwa_1	7	3	4	5	16	7	6	25	25	8	4	6	6	3	5	9
QUL	fwa_2	20	6	3	14	20	9	13	14	6	11	7	13	5	13	13	11
QUL	fwa_3	17	12	11	22	21	25	14	20	2	18	13	23	7	15	14	16
QUL	fwa_4	19	4	2	23	18	6	9	21	3	16	8	10	4	6	9	11
QUL	fwa_r	14	25	16	25	8	22	20	12	16	13	25	16	18	22	22	18
QUL	fwa_w	2	9	8	6	7	4	8	8	9	22	5	5	8	4	4	7
QUL	norm_dd5	1	11	12	1	1	8	3	1	12	1	10	2	11	5	3	5
QUL	pa	12	20	15	13	24	23	12	13	15	23	21	17	21	21	21	18
QUL	pas_wt	11	19	18	3	10	14	16	5	21	4	16	22	13	16	16	14
QUL	shorecx_10less	13	1	1	11	12	2	1	19	1	15	1	1	1	1	1	5
QUL	shorecx_10plus	10	2	10	16	23	21	23	16	4	2	3	24	3	11	11	12
QUL	slope	6	14	9	24	4	3	10	2	13	21	14	9	17	7	6	11
QUL	str_m	15	23	19	12	3	20	15	18	17	9	23	14	16	18	15	16
QUL	str_s	24	8	5	2	2	1	2	24	5	5	2	3	2	2	2	6
QUL	norm_cmd	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	###
QUL	fwa_10ha_more	18	5	14	15	19	17	17	17	7	3	6	20	9	19	17	14
QUL	tave04	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	###
Eco	Predictors	AMWI	BAGO	BUFF	BWTE	CAGO	GWTE	MALL	NSHO	RNDU	SCAU	cavity	dabblers	divers	sp_div	sp_tot	Avg
WCU	alr	23	23	24	21	20	16	17	21	24	20	24	20	24	22	22	21
WCU	aspect	12	10	20	23	4	18	20	19	16	14	14	19	18	20	17	16
WCU	bec_zn	9	8	5	7	22	2	2	6	6	12	6	3	8	2	4	7
WCU	broadleaf	6	24	13	14	15	15	19	16	21	5	17	15	11	16	15	15
WCU	grassland	15	21	18	11	7	20	22	18	18	24	20	21	22	21	20	19
WCU	mixed_forest	8	20	12	10	11	14	18	10	20	8	15	12	10	15	11	13
WCU	needleleaf	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	###
WCU	shrubland	22	16	19	19	14	21	23	20	19	22	22	22	20	23	21	20
WCU	urban	3	22	7	3	24	7	11	3	8	7	13	4	7	7	5	9
WCU	dra_u	10	5	10	9	2	10	8	4	12	11	5	9	12	9	7	8
WCU	fwa_1	16	3	14	6	21	12	4	7	15	10	4	10	15	12	16	11
WCU	fwa_2	24	4	11	24	19	23	21	22	17	18	7	23	17	19	23	18
WCU	fwa_3	14	7	4	13	8	11	14	14	2	4	8	14	4	6	8	9
WCU	fwa_4	13	18	9	4	10	3	6	12	7	16	23	7	14	10	12	11
WCU	fwa_r	1	6	1	1	1	13	9	2	10	3	3	2	3	3	2	4
WCU	fwa_w	2	14	2	2	6	1	1	1	3	1	11	1	2	1	1	3
WCU	norm_dd5	5	12	8	5	3	4	5	5	11	13	10	5	9	8	6	7
WCU	pa	19	13	21	15	18	22	16	13	14	19	18	18	19	17	18	17
WCU	pas_wt	18	9	15	8	13	8	13	11	9	17	12	13	16	14	14	13
WCU	shorecx_10less	20	11	3	16	16	9	3	17	5	6	9	11	6	5	9	10
WCU	shorecx_10plus	11	1	16	20	17	17	12	23	4	15	1	17	5	13	13	12
WCU	slope	7	19	17	12	9	5	10	8	13	9	21	6	13	11	10	11
WCU	str_m	17	17	23	22	12	24	24	24	22	21	16	24	23	24	24	21
WCU	str_s	21	15	22	17	23	19	15	9	23	23	19	16	21	18	19	19
WCU	norm_cmd	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	###
WCU	fwa_10ha_more	4	2	6	18	5	6	7	15	1	2	2	8	1	4	3	6
WCU	tave04	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	###

APPENDIX 6 – Species Distribution Maps

AMWI - American Wigeon

BAGO - Barrow's Goldeneye

BWTE - Blue-winged Teal

BUFF - Bufflehead

CAGO - Canada Goose

GWTE - Green-winged Teal

MALL - Mallard

NOSH - Northern Shoveler

RNDU – Ring-necked Duck

SCAU – Generic Scaup (Lesser Scaup)

Cavity - Cavity-nesters

Include BAGO, BUFF, COGO, COME, HOME

Dabblers

Divers

sp_div – Species Richness Index

sp_tot – All species